به نام خدا

آموزش ASP.NET با زبان ساده

نویسنده:

محمد آقااحمدی

Mahmadi08@gmail.com

پائیز ۱۳۹۰

تقدیم:

این کتاب را به تمامی آن هایی که در جبهه های حق علیه باطل برای دفاع از میهن اسلامی ما و آرمان های انقلاب شجاعانه جنگیدند تقدیم می کنم.

رفتند یاران

چابک سواران...

غلط های املائی و نواقص احتمالی را به بزرگواری خود ببخشید

محمد آقااحمدی

Special Asp.Net Directory

شاخه های مختلفی در یک صفحه ای اس پی وجود دارد که در زیر به اهم آن اشاره می کنیم:

Bin:تمام فایلهای کلاس های جداگانه ی مورد استفاده در صفحات ما که اسمبلی هستند و کامپایل می شوند به صورت dll شده در این شاخه قرار می گیرند.

App_Code:حاوی کلاس های صفحات asp.net می باشد که با پسوند های .vb یا .vc در آن ذخیره می گردد.

App_Data:حاوی پایگاههای داده ی مورد استفاده در سایت مثل SQL و XML .

App_Themes:حاوی فایلهای مربوط به چارچوب کلی تصویری صفحه مثل Skins وThemes .

Global.asax

فایلی که در وب سایت به صورت خودکار یا دستی ایجاد می شود و برای مدیریت شی Application و Session مورد استفاده قرار می گیرد. به این صورت که روی رخداد های مختلف شی Application و Session اجازه ی کد نویسی می دهد.

انواع رخداد ها ی مهم تر موجود در Global.asax:

Application_init :هنگامی که یک شی Application جدید ساخته می شود این رویداد اجرا می شود:

ساخته شدن یک شی Application جدید به نام qqq حاوی مقدار ali:

```
Application.add(“qqq”,”ali”)
```

Application_Disposed:هنگامی که یک شی Application تخریب یا پاک می شود اجرا میگردد.

پاک شدن یک شی Application به نام qqq:

```
Application.Remove("qqq")
Application.Clear()
```

Application_Error:هنگام رویدادن خطا در کارکرد شی Application رخ می دهد.

Application_Start:هنگام آغاز به کار Application پس از ساختن آن اجرا می شود , که پر کاربرد ترین رویداد شی Application میباشد.

Application_End:هنگام پاک شدن آخرین شی Application رخ می دهد.

Application_BeginRequest:هنگام پاسخ به در خواست Application رخ میدهد.

Application_EndRequest:هنگام اتمام پاسخ برنامه به یک در خواست Application رخ می دهد.

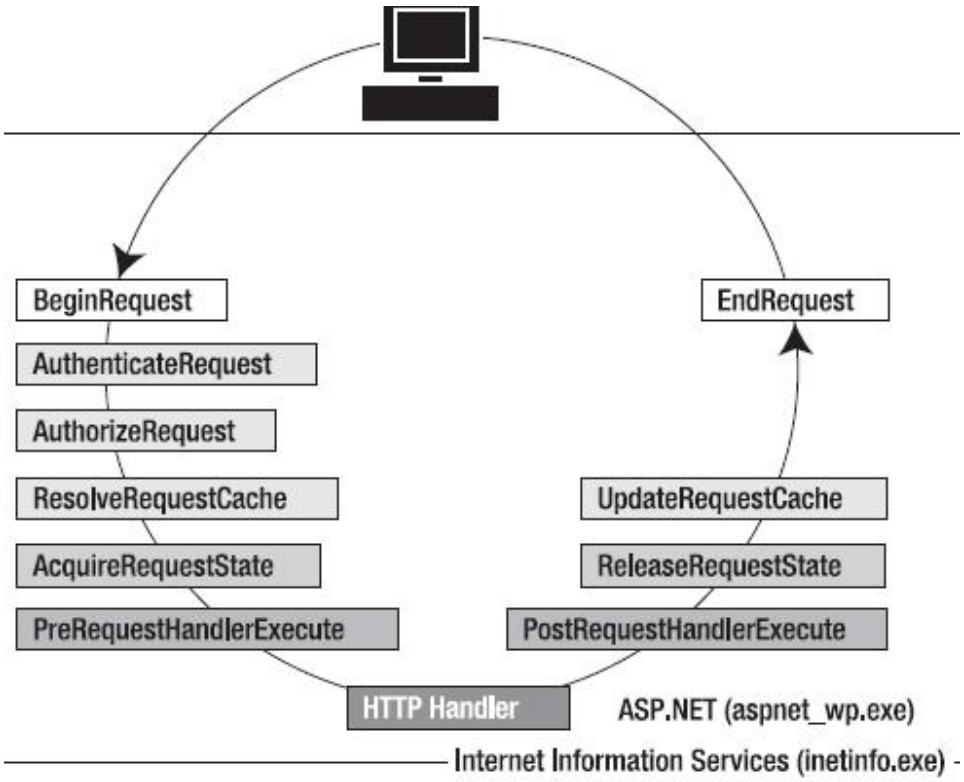
Application_AuthenticationRequest:هنگامی که سیستم از معتبر بودن کاربر مطمین شد رخ می دهد.

Application_AuthorizeRequest:هنگامی رخ می دهد که سیستم مشخص کند کاربر تعیین اعتبار شده , توسط Authentication مجوز دسترسی به منابع را دارد یا خیر.

Session_Start:هنگامی که یک کاربر تعیین اعتبار شد و یک شی Session برایش تعریف شد , این رخداد اجرا می گردد.این عمل زمانی صورت میگیرد که یک رکورد مخصوص کاربر در Session Table ایجاد شد که این عمل با دستور Session Start در Asp.NET صورت می گیرد.

Session_End:هنگامی که Session کاربر تعیین اعتبار شده کارش تمام شود و قصد خروج را دارد , (البته توسط شی Session) رخ می دهد.برای مثال وقتی کوکی مربوط به Session کاربر حذف شد این رویداد فعال می شود.

حالت کلی رخداد ها و نام تمام آنها و ترتیب رخ دادن آنها در شکل زیر آمده است:



برای مثال برای اینکه تعداد بازدید افراد (چه عضو چه غیر عضو) از سایت را داشته باشیم کافی است در رویداد Application_Start یک Application تعریف کنیم و مقدار اولیه آن را ۰ قرار دهیم

```
Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
    Application.Add("ppp", 0)
End Sub
```

سپس با علم به اینکه(شی Application با یک نام خاص مثلا ppp در کل ,یک بار در برنامه ی ما ایجاد شده و تا وقتی که پاک نشده یا برنامه بسته نشده(منظور از بسته شدن برنامه به روز شدن ان و یا سرور است مثلا شما هر بار که برنامه ی خود را در سیستم خود اجرا می کنید و سپس می بندید در اصل برنامه بسته نشده ولی وقتی VisualStudio را می بندیدو سپس آن را باز می کنید برنامه بسته می شود(البته پیش خودتان) ولی وقتی وب سایت خود را Upload میکنید دیگر امکان بسته شدن برنامه بسیار کم می شود مثلا ممکن است سرور به روز شود و یا اینکه شما در کد های سایت خود تغییری ایجاد کنید)و پس از آن برنامه را اجرا می کنید یا به پایان نرسد وجود دارد و مقدار آن نیز حفظ می شود چون اصلا Applicationمخصوص اطلاعاتی که قرار است در

کل برنامه(تمام صفحات) حفظ شوند ساخته شده) باید به دنبال رویدادی بگردیم که با هر بار مراجعه به سایت به مقدارش یکی اضافه شود تا نقش یک شمارنده را دارا باشد .این رویداد میتواند Application_Start باشد ولی چون این رویداد قبلا در ساختن Application مورد استفاده قرار گرفته ازش استفاده نمی کنیم .پس می توان از تنها رویداد ممکن یعنی Application_BeginRequest استفاده می کنیم یعنی هر بار که درخواست شی Application آغاز می شود رخ می دهد :

```
    Sub Application_BeginRequest(ByVal sender As Object, ByVal e As
EventArgs)
        Application("ppp") += 1
    End Sub
```

حتما می پرسید حال چه زمانی اصلا Application آغاز می شود تا آن را بسازیم و چه زمانی Application_BeginRequest می شود که به آن یک واحد اضافه کنیم! کافیست در رویداد page_Load آن را صدا کنیم:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) _Handles Me.Load
        Response.Write(Application("ppp") & "<br>")
    End Sub
```

به این ترتیب با هر بار لود شدن برنامه یک واحد به Application اضافه می شود.البته از Application باید برای اطلاعاتی که قرار است تا پایان برنامه حفظ شود و با بستن آن از بین رود استفاده کنیم نه اطلاعاتی که قرار است پس از پایان برنامه نیز حفظ شود.(منظور از پایان برنامه همانطور که قبلا توضیح دادیم بسته شدن مرورگر نیست بلکه...-در بالا توضیح داده شد-)پس برای نگهداری دایمی اطلاعات باید از پایگاه داده استفاده کنیم.این مثالی که از شمارنده زده شد وقتی صحیح است که برنامه را ه اندازی مجدد نشود زیرا اگر در هر جایی از پیکر بندی برنامه اگرتغییری رخ دهد شی Application از بین می رود در اینجا هم با بستن برنامه(شروع مجدد) دوباره از صفر شروع می شود و در اصل برای شمارش محکم و مطمین باید از پایگاه داده استفاده کرد.

Web.Config

از مهمترین فایل های یک برنامه ی asp.net است که تنظیمات امنیتی مربوط به دادن و گرفتن مجوز از افراد مختلف دادن و گرفتن نقش ها و اطلاعات امنیتی و محلی در آن انجام می گیرد و از جنس xml می باشد(نسبت به حروف کوچک و بزرگ حساس است) و ساختار کلی آن به شکل زیر است:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
<system.web>
<!-- ASP.NET configuration sections go here. -->
</system.web>
</configuration>
```

خط اول که معرف xml بودن صفحه است.اصل کار web.config تگ <configuration> است که بین آن تمام برنامه ها وجود دارد و تگ <system.web> نیز در داخل آن وجود دارد.

معرفی مهمترین تگ های web.config:

<Authentication>

نحوه ی دادن تصدیق را مشخص می کند که وقتی کاربر تقاضای ورود کرد بر چه مبنایی اعتبار وی سنجیده شود که ٤ مقدار را از طریق خصوصیت mode مشخص می کند که عبارتند از Form , Passport ,Windows ,Noneکه اگر مقدار Form را انتخاب کنیم تصدیق از طریق cookie صورت می گیرد که رایج ترین حالت است.و Windows مخصوص استفاده از سرویس امنیتی microsoft است و Passport مخصوص خود IIS است و در نهایت None هم که بدون امنیت است:

```
<authentication mode="Forms">

</authentication>
```

مثلا اگر بخواهیم از دسترسی کاربران غیر مجاز به صفحه ای مثل main.aspx جلو گیری کنیم باید در داخل تگ <Authentication> , از تگ <forms> نیز استفاده کنیم به شکل زیر:

```
<authentication mode="Forms">
    <forms name="reg" loginUrl="login.aspx" protection="All" timeout="20"
path="/" requireSSL="false" slidingExpiration="true"
cookieless="UseDeviceProfile" defaultUrl="main.aspx"></forms>
</authentication>
```

همانطور که میبینید در تگ <form> اولین صفت نام است که اهمیت چندانی ندارد. ولی دومین صفت

LoginUrl است که مشخص می کند در صورت مجوز نداشتن یک کاربر ,اگر وی می خواست به صفحه ی main.aspx برود وی را به صفحه ی Login.aspx بفرستد تا اعتبار لازم را از طریق Login شدن بد ست آورد.صفت بعدی که Protection است مقدار محافظتی را مشخص می کند که سعی کنید همیشه All باشد.صفت بعدی TimeOut است که بر حسب دقیقه مشخص می کند که cookie که کاربر با ورودش ایجاد کرده کی منقضی شود که مقدار پیش فرض آن ۳۰ دقیقه است.Path آدرس محل ذخیره شدن cookie است که یک علامت / کافی است.صفت requireSSL مشخص کننده ی این است که آیا به جای متن عمل رمزگذاری روی cookie انجام شود یا خیر.صفت بعدی slidingExpiration است که سعی کنید همیشه True باشد.این صفت تعیین می کند که اگر کاربروارد سایت شد و cookie آن ساخته شد و ۳۰ دقیقه به طور پیش فرض برای TimeOutآن در نظر گرفته شد , اگر کاربر دوباره به سایت مراجعه کند این ۳۰ دقیقه refresh شود (یعنی مجددا از زمان مراجعه ی دوباره ی وی حساب شود) یا از همان در مراجعه ی اول که True بودن این صفت باعث refresh شدن TimeOut می شود.صفت Cookieless را سعی کنید همیشه با UseDeviceProfile مقدار دهی کنید و در نهایت صفت DefaultUrl که صفحه ی مقصد را مشخص می کند که کاربران دارای مجوز به آن رفته و کاربران بی نام به آدرسی که در LoginUrl است بروند.

<Authorization>

تگ بسیار مهم که مشخص می کند کدام افراد عضو سایت هستند و کدام افراد عضو نیستند .علامت * به معنای کاربران عضو و علامت ؟ به معنای کاربر غیر عضو یا بی نام است. این اعمال از طریق دو تگ مهم <deny> برای ممانعت از ورود و <allow> برای اجازه ی ورود استفاده می شود . کد زیر کاربران بی نام را رد می کند و از ورود آنها جلو گیری می کند:

```
<authorization mode="Forms">
    <deny users="?"/>
</authorization>
```

کد زیر کاربران بی نام را رد می کند و به افرادی که نامشان javadوali است اجازه ی ورود می دهد:

```
<authorization mode="Forms">
    <deny users="?"/>
    <allow users="ali"/>
    <allow users="javad"/>
</authorization>
```

همچنین مدیریت نقش ها نیز توسط تگ های بالا انجام می گیرد با این تفاوت که به جای کلمه ی users کلمه ی roles قرار می گیرد.کد زیر از ورود تمام افراد عضو به جز افرادی که حاوی نقش manager هستند جلوگیری می کند:

```
<authorization mode="Forms">
    <deny  users="*"/>
    <allow roles="manager"/>
</authorization>
```

<Compilation>

تگی است که مربوط به نحوه ی کامپایل برنامه است.که یک خصوصیت مهم آن defaultlanguage است که زبان کامپایل را تعیین می کند.

<CustomError>

تگی است که خطا های رخ داده زمان اجرا (RunTime Errors) را به گونه ای سفارشی می کند.

که دو خصوصیت مهم دارد :

۱-DefaultRedirect:که آدرس صفحه ای است که در صورت بروز خطا به آن فرستاده می شویم البته با تنظیمات صفت دوم تگ <CustomError> به نام Mode.

۲-Mode:که حاوی سه مقدار on, off, Remote Only است که نحوه ی مقابله با خطا را مشخص می کند.که اگر Remote Only باشد در صورت مواجهه با خطا به جای دیدن صفحه ی خطای معمولی asp.net به صفحه ای خاص که آدرس آن در DefaultRedirect است فرستاده می شویم البته تنها کاربران غیر محلی به آن فرستاده می شوند و کاربران محلی مثل خود ما که جلو سرور نشسته ایم به همان صفحه ی asp.net default که به طور دقیق و جزیی خطا را شرح می دهد فرستاده می شویم نمونه ای از این صفحه که ما محلی ها به آن می رویم به صورت زیر است:

و لینک این خطا در آدرس زیر است:

http://localhost:3517/yoursitename/runtimeError.aspx

حال اگر صفت Mode با on تنظیم شود چه محلی ها و چه غیر محلی ها به صفحه ا ی که آدرسش در DefaultRedirect است می روند.و در صورت تنظیم صفت Mode با off همان خطای خودکار asp.net نمایش داده می شود که در شکل بالا میبینید.

حال خود تگ <CustomError> برای اجرا نیاز به یک تگ درونی به نام <error> دارد که می تواند به هر تعدادی باشد که هر کدام نوع خطا ی رخ داده را تشخیص می دهند.این تگ حاوی دو صفت مهم است اولی StatusCode می باشد که کد خطای رخ داده است مثلا همه می دانیم کد ٤٠٤ مخصوص خطای page not found می باشد.دومین صفت هم Redirect است که آدرس صفحه ای است که در صورت رخ دادن خطای ٤٠٤ به آنجا برویم.حال حتما می پرسید صفت Redirect در تگ <error> چیست و صفت DefaultRedirect در تگ <CustomError> چیست!!! باید بگوییم که همانطور که از نامشان بر می آید صفحه ی پیش فرض ما DefaultRedirect می باشد که همه نوع خطای RunTime را در بر می گیرد حال اگر تگ <error> را به داخل <CustomError> اضافه کنیم در صورت رخ دادن خطا ی مشخص شده توسط صفت StatusCode در تگ <error> صفحه به آدرسی که صفت Redirect در تگ <error> مشخص کرده می رود نه DefaultRedirect چون Redirect به DefaultRedirect (در صورت وجود) ارجعیت دارد.

با این مثال مفهوم را بهتر متوجه می شوید:

در صفحه ی web.config تگ های زیر را اضافه کنید:

```
<customErrors defaultRedirect="oo.aspx" mode="On">
    <error statusCode="404" redirect="kkk.aspx" />
</customErrors>
```

سپس دو صفحه ی ساده به نامهای oo.aspx و kkk.aspx طراحی کنید که در هر کدام یک پیغام خطای سفارشی خود شما البته متمایز از هم باشد را دارا باشد.حال یک صفحه ی اصلی ساده(حتی صفحه ی سفید جهت آزمایش) طراحی کنید و آن را اجرا نمایید و سپس آدرس یک صفحه که وجود خارجی ندارد را به آن بدهید ,می بینید که به صفحه ی kkk.aspx فرستاده شده اید.حال اگر لینک بالا ی صفحه را نیز بدهید باز هم به صفحه ی kkk.aspx متصل می شوید.چون اولا mode="On" است و دوما تگ <error> وجود دارد.حال اگر خطای دیگری جز page not found انجام دهید دیگر به kkk.aspx فرستاده نمی شوید چون kkk.aspx فقط برای خطای ٤٠٤ طراحی شده است .دلیل اینکه شما هم که محلی هستید به صفحه ی سفارشی منتقل شدید این است که mode="On" شده ,اگر mode="RemoteOnly " می بود شما به صفحه ی خطای default asp.net منتقل می شدید و غیر محلی ها به صفحه ی سفارشی می رفتند.

<Globalization>

تگی است که ویژگی های فرهنگی و زبانی برنامه ی ما را مشخص می کند مثلا یک صفت آن requestencoding است که شکل رشته های در خواست شده را تعیین می کند مثل Unicode

<sessionState>

تگی که با Session سر و کار دارد که مدت زمان Session و تعیین نحوه ی در خواست را مشخص می کند.همینطور صفت مهم Cookieless در آن تعیین می کند این Session خاص با Cookie کار کند یا با SessionID که مقدار True یعنی با SessionID و مقدار False یعنی Cookie.صفت دیگر این تگ TimeOut است که اعتبارSession را بر حسب دقیقه تعیین می کند.در بخش Session به تفصیل در مورد پیکر بندی این تگ توضیح داده خواهد شد.

<Location>

تگ مهمی که آدرس صفحه ی وبی که برخی تنظیمات باید رویش اعمال شود معلوم می کند.مثلا ما در تگ های امنیتی مثل Authorization دیدیم که چگونه یک سری از افراد اجازه ی ورود داشتند و یک سری اجازه ی ورود نداشتند.حال در کدام صفحه این عمل انجام می گرفت؟کد قبلی به صورت زیر بود که کاربران بی نام را رد میکرد.ولی از چه صفحه ای؟(که صفت Path آن را مشخص می کند)

```
<authorization mode="Forms">
    <deny users="?"/>
</authorization>
```

داریم که مشخص می کند چه صفحه <Location>برای مشخص شدن آن نیاز به تگ ای است که نیاز به ورود کاربران مجاز دارد و از ورود کاربران غیر عضو یا بی نام از آن جلو گیری می شود

.برای این کار تگ <system.web> را روی تگ <Authorization> آورده و پس از آن تگ <Location> را روی تگ <system.web> اضافه می کنیم.به صورت زیر:

```
<location path="main.aspx">
    <system.web>
      <authorization>
          <deny users="?"/>
      </authorization>
    </system.web>
</location>
```

به این ترتیب افراد بی نام یا غیر عضو نمی توانند به صفحه ی main.aspx وارد شوند.

<ConnectionString>

تگی بسیار مهم که خود به خود وقتی یک پایگاه داده را به برنامه ی خود اضافه می کنید ایجاد می شود و به طور کلی به صورت زیر است:

```
<connectionStrings>
  <add            name="ConnectionString"          connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\CDB.mdf;Integrated
Security=True;User Instance=True"
    providerName="System.Data.SqlClient" />
 </connectionStrings>
```

ما به عنوان مثال یک پایگاه داده را به برنامه ی خود اضافه کردیم که این عمل اضافه کردن, با استفاده از تگ <add> در داخل تگ <ConnectionString> صورت گرفت که حاوی ۳ عنصر است.اولی که یک نام معمولی است و کاربردش در استخراج این رشته از جایی دیگر است که با ذکر نام صورت می گیرد.صفت ConnectionString مهمترین

صفت این تگ است که حاوی رشته ی اتصال به پایگاه داده است.درجلوی صفت providerName نیز اگر پایگاه داده ی شما SQL باشد System.Data.SqlClient قرار می گیرد و اگر پایگاه داده ی شما Access باشد System.Data.OleDb قرار می گیرد. میتوانید از تگ <add> چندین بار استفاده کنید .یعنی همزمان به چندین پایگاه داده وصل شوید.

<AppSettings>

این تگ داده های عمومی وب سایت شما را به صورت ایستا در یک شی به نام AppSettings ذخیره می کند.به این صورت می توانید بار ها از آن داده در صفحات سایت خود استفاده کنید.برای تعریف داده ها باید از تگ <add> در داخل <AppSettings> استفاده کنید.میتوانید هر چند بار از تگ <add> در این مکان استفاده کنید .کد زیر دو متغیر به نامهای a وb ایجاد می کند که متغیرa حاوی مقدار ali و متغیر b حاوی مقدار reza است.

```
  <appSettings>
    <add key="a" value="ali" />
    <add key="b" value="reza" />
</appSettings>
```

حال برای بازیابی این دو مقدار کافی است در صفحه ی وب سایت از متد ConfigurationManager استفاده کنید به صورت زیر:

```
Response.Write(ConfigurationManager.AppSettings("a"))
```

که خروجی این کد ali می باشد.

می توان از داخل برنامه نیز تغییراتی را روی web.config انجام داد مثلا به تگ ConnectionString آن دسترسی داشت که به این منظور باید متغیری از نوع ConnectionStringSetting تعریف کرد:

```
Dim connection As ConnectionStringSettings
```

سپس برای بازیابی از تگ ConnectionString در web.config می بایست از یک حلقه ی For Each...next استفاده کرد چون ممکن است شما چندین ConnectionString تعریف کرده باشید.و با استفاده از خصوصیت ConnectionString متد WebConfigurationManager روی حلقه به دنبال ConnectionString موجود در web.config گشت:

```
For Each connection In WebConfigurationManager.ConnectionStrings
   Response.Write("Name: " & connection.Name & "<br />")
```

```
        Response.Write("Connection String: " _
        & connection.ConnectionString & "<br /><br />")
    Next
```

و با connection.Name نام کانکشن و با connection.ConnectionString مقدار آن را بدست آورد.همچنین میتوان با connection.ProviderName نوع پایگاه داده را تشخیص داد.

همچنین برای دسترسی کامل به یک ConnectionString خاص ببه صورت زیر عمل می کنیم.در زیر ConnectionString با نام ali را برای ما به طور کامل استخراج می کند:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
      Response.Write(ConfigurationManager.ConnectionStrings("ali"))
End Sub
```

WebConfigurationManager متد دیگری به نام OpenWebConfiguration دارد که با آن نیز می توان به محتویات web.config دسترسی داشت که این متد مقداری string می پذیرد که همان آدرس محل web.config است که با کد Request.ApplicationPath مشخص می شود.حال می توان متغیری از نوع Configuration تعریف کرد و مقدار اولیه اش را به متد WebConfigurationManager سپرد:

```
Dim con As Configuration =
WebConfigurationManager.OpenWebConfiguration(Request.ApplicationPath)
```

حال متغیر con آماده ی استفاده است.مثلا می توان یک AppSettings جدید ایجاد کرد:

```
con.AppSettings.Settings.Add("aaag", " kharabetam refigh ")
con.Save()
```

این کد باعث اضافه شدن یک AppSettings جدید به web.config می شود.که می توان پس از ایجاد , آن را با کد زیر بازیابی کرد:

```
Response.Write(ConfigurationManager.AppSettings("aaag"))
con.Save()
```

یا می توان مقدار یکAppSettings از پیش تعریف شده را ویرایش کرد:

```
con.AppSettings.Settings("aaag").Value = "I Love Asp.net"
con.Save()
```

در این کد ما مقدار AppSettings , 'aaag' را از kharabetam refigh به I Love Asp.net ویرایش کردیم.

با کد زیر می توان یک AppSettings را حذف نمود:

```
con.AppSettings.Settings.Remove("aaag")
con.Save()
```

یادتان باشد که در هر عملی روی AppSettings باید آن را save کنید.به این صورت که اول نام شی ساخته شده که تمام تغییرات حذف اضافه و ویرایش روی آن اعمال می شود(که در اینجا con می باشد) سپس یک نقطه و بعد از آن انتخاب گزینه ی save :

```
con.Save()
```

موارد و تگ های زیادی هستند که در فایل web.configتعریف می شوند و توسط اشیای مربوطه قابل دسترسی هستند مثل تگ OrderService و...

شاید از خود بپرسید فایل web.config یک فایل بسیار مهم می باشد پس راه جلوگیری از هکر ها و پنهان کردن آن از دیگران چیست؟مسلما دیگران نباید به تگ های این فایل دسترسی پیدا کنند.برای دیدن فایل web.config کافیست در آدرس با صفحه ی سایت web.config را بنویسیم.asp.net راهی مناسب برای پنهان کردن این فایل و هر فایلی را از قبیل pdf,jpg,gif,docو ... را فراهم کرده است و آن استفاده از httpHandlers است که درون تگ <system.web> باید نوشته شود.

پس بهتر است با مفهوم httpHandlers آشنا شویم:

httpHandlers یک Component برای پاسخگویی به در خواست های مختلفی است که در یک برنامه ی Asp.NET وجود دارد.بارز ترین حالت استفاده از httpHandlers استفاده از آن در جهت محافظت از داده هاست.مثلا شاید شما دلتان نخواهد که افرادی که از سایت شما بازدید میکنند به فایل های پسوند doc دسترسی داشته باشند.به این منظر از HttpHandler استفاده می کنیم.این کار با یک کلاس مشتق شده از HttpHandler صورت می گیرد.این کلاس نامش HttpForbidenHandler است.کلا برای استفاده از HttpHandler می بایست از تگ <httpHandlers> در Web.Config استفاده کنیم.در داخل این تگ باید ۳ صفت کلی را با تگ <add> مقداردهی کنیم.

Verb:مشخص کننده ی نحوه ی ارسال درخواست مربوطه است. Get یا Post و یا * که به معنای هر دو است.

Path:آدرس فایلی است که باید درخواست مورد نظر به آن ارسال شود.

Type:نام کلاسی است که در اصل عملیات مربوطه را روی آن آدرسی که در Path دریافت شد انجام می دهد.اگر کلاس حاوی فضای نام هم بود ابتدا فضای نام و سپس نام کلاس آورده می شود:

NameSpace.ClassName

پس ما که در اینجا می خواهیم از یکسری فایل محافظت کنیم اولا قسمت Verb را با * مقدادهی می کنیم.و چون قصد ما حفاظت از تمام فایل ها با پسوند doc. است پس صفت Path را با doc.* مقداردهی می کنیم.* یعنی هر نامی و doc. یعنی با پسوند doc.اگر نام خاصی مد نظرتان بود به جای ستاره آن نام را قرار دهید.حالا می رسیم به عمل.همانطور که گفتم کاری که httpHandler انجام می دهد درون یک کلاس است که نام این کلاس را به همراه فضای نامش باید در صفت Type بیاوریم:

```
  <system.web>
      <httpHandlers>
            <add          verb="*"          path="*.config"          type="
System.Web.HttpForbiddenHandler " />
      </httpHandlers>
</system.web>
```

حالا شاید بخواهید با طرز ایجاد Custom HttpHandlers هم آشنا شوید.برای ایجاد Custom HttpHandlers ابتدا باید کلاسی تعریف کنید و نامش را درون صفت Type بیاورید.این کلاسی که تعریف می کنید حتما باید از IhttpHandler (که HttpHandler از آن مشتق می شود) Implement شود.نکته ی بسیار مهم این است که این کلاس حتما باید حاوی ۱ متد و ۱ خاصیت باشد.

متد ProcessRequest که عملیات اصلی را انجام می دهد.در این متد ما باید به اشیایی نظیر response و Request دسترسی داشته باشیم به این منظور از آرگومان ورودی این متد که از نوع HttpContext است استفاده می کنیم.این شی در حقیقت یک زمینه از اشیایی که ممکن است لازم شود به ما می دهد.این تابع هم باید از Implement IhttpHandler شود.

خاصیت IsReusable که پس از اتمام کار تابع ProcessRequest تعیین می کند آیا این تابع دوباره استفاده شود یا نه. این خاصیت هم باید از Implement IhttpHandler شود.در زیر یک کلاس به نام SimpleHeader با فضای نام HttpExtension ایجاد

کردیم که با استفاده از متد Response شی Context یک نوشته را در خروجی نمایش می دهد.

```
Namespace HttpExtensions
    Public Class SimpleHandler
        Implements IHttpHandler
        Public Sub ProcessRequest(ByVal context As
System.Web.HttpContext)Implements IHttpHandler.ProcessRequest
            context.Response.Write("My custom Http Handler")
        End Sub
        Public ReadOnly Property IsReusable() As Boolean Implements
IHttpHandler.IsReusable
            Get
                Return True
            End Get
        End Property
    End Class
End Namespace
```

حال برای اعمال این HttpHandler کافی است ان را در Web.Config رجیستر کنیم:

```
<httpHandlers>
<add verb="*" path="httphandler.aspx" type="HttpExtensions.SimpleHandler"
/>
</httpHandlers>
```

با این کار اگر صفحه ی httphandler.aspx را اجراکنید نوشته ی My Custom Handler روی آن نمایان می شود.

یک نوع فایل در asp.NET وجود دارد به نام GenericHandler که با آن می توان Custom HttpHandlers ها را به شکل ساده تری ایجاد کرد.برای باز کردن این فایل روی Add New Item رفته و GenericHandler را انتخاب کنبد.آنگاه یک فایل با پسوند ashx ایجاد می شود که کد های زیر به طور پیش فرض در آن قرار دارند:

```
<%@ WebHandler Language="VB" Class="Handler" %>

Imports System
Imports System.Web

Public Class Handler : Implements IHttpHandler

    Public Sub ProcessRequest(ByVal context As HttpContext) Implements
IHttpHandler.ProcessRequest
        context.Response.ContentType = "text/plain"
        context.Response.Write("Hello World")
    End Sub
```

```
    Public ReadOnly Property IsReusable() As Boolean Implements
IHttpHandler.IsReusable
        Get
            Return False
        End Get
    End Property

End Class
```

که می توانید روی آن به دلخواه تغییرات ایجاد کنید و دیگر کلاس HttpHandler را به طور دستی ایجاد نکنید.در بخش های بعدی با نوع پیشرفته ی HttpHandler آشنا می شود.

نحوه ی ایجاد Component:

component ها یک سری کلاس هستند که به وسیله ی آنها می توان با یک بار کد نویسی چندین بار استفاده نمود.ولی نه یک کلاس ساده بلکه یک فایل ClassLibrary و به صورت dll شده.یک کلاس ساده در پوشه ی app_code و در همان برنامه ی وبی که قرار است از آن استفاده کنیم ذخیره می شود و برای اینکه در سایر برنامه ا بتوان از آن استفاده کرد باید آن را Copy Paste کرد.ولی یک کلاس ساده یک بار ایجاد می شود و نوسط Add Referenc به برنامه ی وب اضافه می گردد ولی نه به صورت ک کلاس بلکه یک فایل اسمبلی dll شده.

در اینجا میخواهیم یک کلاس ساده ایجاد کنیم ولی به صورتی آن را کامپایل کنیم و به dll تبدیل کنیم تا خودش به فیل Bin منتقل شود.برای این کار ابتدا از گزینه ی New Project گزینه ی ClassLibrary را انتخاب کنید(این کار را خارج از web Application انجام دهید یعنی در یک پروژه ی جدید). من نام آن را class1می گذارم.تنها چیزی که برای شما در این کلاس نمایش داده می شود کد زیر است:

```
Public Class class1

End Class
```

سپس یک فضای نام برایش ایجاد میکنیم:

```
Namespace calculator

    Public Class class1

    End Class
```

```
End Namespace
```

حال یک کلاس بسیار ساده می زنیم:

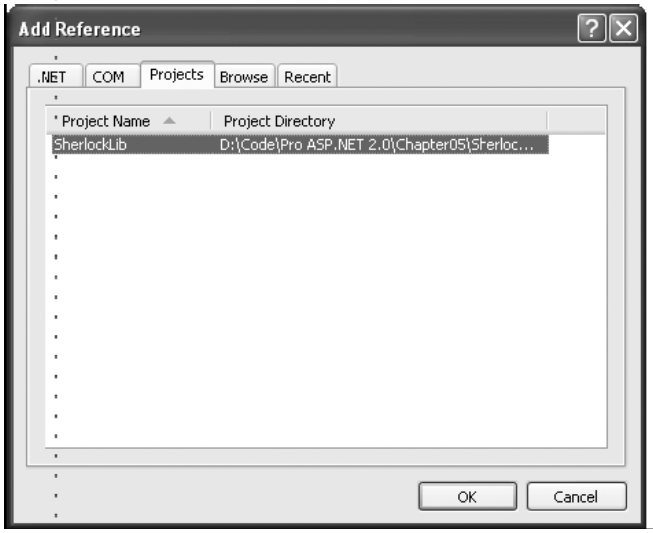
```
  Imports Microsoft.VisualBasic

  Namespace ass

  Namespace calculator
      Public Class Class1
          Private m_f As Integer
          Private m_s As Integer
          Private m_func As String
          Public Property first() As Integer
              Get
                  Return m_f
              End Get
              Set(ByVal value As Integer)
                  m_f = value
              End Set
          End Property
          Public Property second() As Integer
              Get
                  Return m_s
              End Get
              Set(ByVal value As Integer)
                  m_s = value
              End Set
          End Property
          Public Property func() As String
              Get
                  Return m_func
              End Get
              Set(ByVal value As String)
                  m_func = value
              End Set
          End Property
          Public Function calculating() As Integer
              Return calculating(first, second, func)
          End Function
          Public Function calculating(ByVal first1 As Integer, ByVal second1
As Integer, ByVal func1 As String) As Integer
              Select Case func1
                  Case Is = "+"
                      Return first1 + second1
                  Case Is = "-"
                      Return first1 - second1
                  Case Is = "*"
                      Return first1 * second1
```

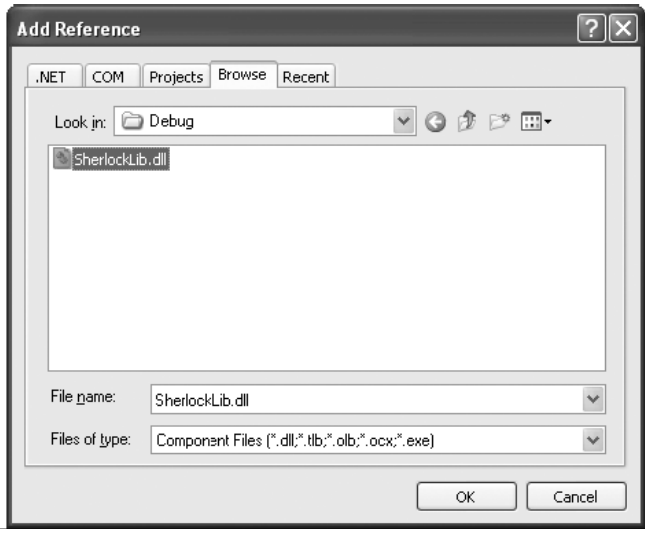
```
                Case Is = "/"
                    Return first1 / second1
            End Select
        End Function
    End Class
End Namespace
```

ما در اینجا یک کلاس ماشین حساب ساده با ٤ عمل اصلی ایجادکردیم.دو مقدار first & Last هم مقایر من هستند که باید عملیاتی که با خاصیت func مشخص می شود روی آن دو اجرا شود.پس از اتمام کار کافی است از منوی Build گزینه ی Bulid Class1 را انتخاب کنیم.در حالت کلی:

Build > Bulid 'Your Class Name'

با این کار عمل کامپایل انجام می شود.حال به وب سایت خود می رویم تا این فایل را به صورت dll و اسمبلی به آن اضافه کنیم.برای این کار در Solution Explorer گزینه ی Add Reference را کلیک کنید سپس در تب Browse نام فایل dll خود را که به صورتClassLibrary آن را Build کرده بودید انتخاب کرده و Ok کنید خود به خود ۳ فایل به پوشه ی bin شما اضافه می شوند:





و حالا کلاس مربوطه در My Web Application قابل دسترسی است:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim k As New ClassLibrary2.calculator.Class1
        k.first = 12
        k.second = 11
        k.func = "+"
        Response.Write(k.calculating())
    End Sub
```

این تکنیک در ایجاد کنترل های سفارشی بسیار کاربرد دارد.
در حالت کلی فایل Web.Config تگ های بسیاری دارد که تعریف هر کدام از آنها در این بخش زیاد جالب نیست و درمورد هر کدام در مباحث مربوطه صحبت خواهد شد.

State Management

مدیریت حالت به مجموعه تکنیک ها یی گفته می شود, که به حفظ حالت برنامه کمک می کنند.مثلا اگر بخواهید برای حفظ حالت یک صفحه ی خاص, اطلاعات کاربر را در اختیار داشته باشید ,مثل نام کاربری, مدیریت حالت می تواند به شما کمک کند.
مدیریت حالت شامل تکنیک های مختلفی می شود که در زیر اهم آنها آمده است:

View State
Query String
Cross-Page Posting
Cookie
Session
Applicatiom
Cache
Profile

View State:

یک شی که مقادیر فرم ها را در خود نگهداری می کند.البته به گونه ی کاملا موقتی!.به این ترتیب که در هر صفحه ای که شما در آن مقادیری را وارد می کنید , View State یک فیلد مخفیHidden Field)) ایجاد می کند و مقادیر داده ای شما را تا وقتی صفحه باز باشد در آن فیلد نگه می دارد که حتی با PostBack شدن صفحه هم از بین نمیرود.ولی بدی View State این است که فقط در Current Page کاربرد دارد ,یعنی تنها در همان صفحه ای که مقادیر را وارد می کنید , نه تمام صفحات.
تعریف آن نیز به صورت زیر است:

```
ViewState.Add("a", "reza")
```

که در این کد ما یک View State به نام a با مقدار reza تعریف کردیم.
میتوان به جای رشته های د ل خواه ,از کنترلهای موجود استفاده کرد.مثلا کد زیر یک View State تعریف می کند با نام یک کنترل TextBox و مقدار Text آن:

```
ViewState.Add(TextBox2.ID, TextBox2.Text)
```

ولی در حالت کلی یادتان باشد که اگر خواهان صفحات سبکتری هستید از View State استفاده نکنید زیرا حجم صفحه را با فیلد مخفی که می سازد زیاد می کند.همچنین از نگهداری اطلاعات سری و مخفی کاربران در View State بپرهیزید .چون هر لحظه امکان از بین رفتن اطلاعات آن می باشد.چون همانطور که گفتیم View State در Current Page کاربرد دارد . می توان به جای آن از کلکسیون های کم حجم تری مثل HashTable , , ArrayList Stack وQueue استفاده کرد.

برای باز یابی View State هم می توان به صورت زیر عمل کرد:

```
label1.Text = ViewState(TextBox1.ID)
```

هنگام بازیابی View State همیشه شرط زیر را بر روی آن قرار دهید تا هر وقت تهی نبود بازیابی شود تا در این صورت با خطا ی NullRefrence مواجه نشوید:

```
  If ViewState(TextBox1.ID) IsNot Nothing Then
      Label1.Text = String.Empty
      Label2.Text += ViewState(TextBox1.ID) & "<br>"
End If
```

برای ذخیره کردن مقادیر مختلفی از نوع string یا integer یا هر چیز دیگر , View State برای شما محدودیتی قایل نیست. به این صورت که View State ابتدا مقادیر ورودی را به stream of bytes تبدیل می کند و سپس این مقادیر stream of bytes را وارد فیلد مخفی می کند. سپس برای بازیابی , از فیلد مخفی آن مقادیر را می خواند و سپس آنها را از byte به string(در اینجا string است و ممکن است هرئچیزی باشد) تبدیل می کند.به این فرایند serialization گفته می شود.مثلا به کلاس بسیار ساده ی زیر اگر دقت کنید . میبینید که نام کلاس customer است و دو متغیر public دارد یکی FirstName و دیگری LastName و در تابع سازنده اش که دو مقدار First و Last را می گیرد و در دو متغیر public خود قرار می دهد.

```
  Public Class Customer
      Public FirstName As String
      Public LastName As String
      Public Sub New(ByVal first As String, ByVal last As String)
          FirstName = first
          LastName = last
      End Sub
End Class
```

اگر متغیری از نوع این کلاس تعریف کنیم به صورت زیر می شود:

```
Dim cust As New Customer("reza", "ahmadi")
```

حال با توجه به این که گفتیم View State هر نوع مقداری را در خود ذخیره می کند ,در اینجا می خواهیم به جای string و integer نوع داده ای را که خودمان ایجاد کردیم قرار دهیم یعنی متغیر cust که از نوع customer است و در کد بالا تعریف کردیم را در View State قرار می دهیم و اسمش را a و مقدارش را cust قرار می دهیم.حال آنکه خود cust حاوی FirstName و LastName است که به ترتیب مقادیر reza و ahmadi را در خود ذخیره کرده است:

```
ViewState.Add("a", cust)
```

به همین سادگی!.ولی اگر این کدها را اجرا کنید یک خطا از شما گرفته می شود و آن هم این است که نوع داده ای شما چون از انواع خود VB.Net نیست پس باید با توجه به آنچه در بالا تر در مورد serialization اشاره شد ,Serializable شود .برای حل این مشکل کافیست به سادگی هر چه تمام تر به کلاس customer عبارت <Serializable()> _ را اضافه کنید:

```
  <Serializable()> _
  Public Class Customer
      Public FirstName As String
      Public LastName As String
      Public Sub New(ByVal first As String, ByVal last As String)
          FirstName = first
          LastName = last
      End Sub
End Class
```

حال View State به درستی مقادیر شما را به توجه به فرایند از پیش تعریف شده ی serialization در خود ذخیره می کند.حال شاید از خود بپرسید نحوه ی بازیابی این View State چگونه است!چگونه می توان مقدار stream of bytes را به customer تبدیل کرد.حتما تا به حال با تابع مهم CType آشنا شده اید.این تابع دو آرگومان ورودی دارد اولی یک متغیر با یک نوع داده ای خاص و دومی یک نوع داده ای دیگر که می بایست اولی به دومی تبدیل شود .مثلا در کد زیر عدد ٤٠٠ که integer است را می خواهیم به string تبدیل کنیم:

```
CType(400, String)
```

در مثال خودمان نیز می خواهیم مقدار نو ع داده ای که در View State ذخیره شده را به customer تبدیل کنیم و آن را در متغیری از نوع customer بریزیم و سپس عمل بازیابی انجام پذیرد:

```
  Dim j As Customer
  j = CType(ViewState("a"), Customer)
  Label1.Text = String.Empty
  Label1.Text += j.FirstName & "<br>"
Label1.Text += j.LastName
```

که خروجی این کد به صورت زیر است:

Reza
Ahmadi

به این اعمالی که انجام شد Storing Objects in View State می گویند یعنی ذخیره کردن اشیا در View State که حتما باید عمل serialization روی آنها انجام پذیرد.همچنین اگر بخواهید کلکسیون ها را نیز در View State ذخیره بکنیدهم باید عمل serialization صورت گیرد ولی چون کلاسش را شما تعریف نکرده اید (و پیش فرض vb.NET است) نیازی به کلمه ی <Serializable()> _ در آن نیست.زیرا این کلمه مخصوص کلاسهایی است که خود شما تعریف کرد ه اید.در مثال زیر کلکسیون HashTable به نام j تعریف شده و مقدار دهی شده و در View State ذخیره شده :

```
  Dim j As New Hashtable
  j.Add("a", 2)
  j.Add("b", 3)
  j.Add("c", 4)
  j.Add("d", 5)
ViewState.Add("ddd", j)
```

و برای بازیابی عمل تبدیل نوع داده ای صورت گرفته است:

```
  Dim ooo As New Hashtable
  ooo = CType(ViewState("ddd"), Hashtable)
  Response.Write(ooo("a").ToString)
  Response.Write(ooo("b").ToString)
  Response.Write(ooo("c").ToString)
Response.Write(ooo("d").ToString)
```

حتما می دانید در هر صفحه ی Asp.Net یک صفت به نام EnableViewState وجود دارد که دو مقدار True,False می گیرد و پیش فرض آن True است.حال اگر این مقدار را False قرار دهید تمام کارهایی که تا اینجا انجام داده ایم بی فایده خواهد بود و بدون هیچ پیغام خطایی نه عمل ذخیره در ViewState انجام می گیرد نه بازیابی.هر

کنترل نیز در Asp.Net این صفت را دارا است که می توان این کار را برای هر کنترل برای جلوگیری از افزایش حجم اطلاعات صفحه در مورد کنترل هایی که نیازی به ذخیره ی اطلاعات آنها نیست انجام داد تا صفحه ی ما بهتر و سریعتر بار گذاری شود.کد زیر خصوصیت EnableViewState برای کل صفحه را False می کند:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="_Default" EnableViewState="false" %>
```

اگر در صفحه ی اجرا شده ی Asp.Net را در مرورگر ببینید که صفات EnableViewState آن با True تنظیم شده باشد ,کلیک راست کرده و ViewSource را بزنید در Source صفحه حتما با خطی مثل خط زیر برخورد می کنید:

```
<input type="hidden" name="__VIEWSTATE" value="dDw3NDg2NTI5MDg7Oz4="/>
```

این خط در اصل همان فیلد مخفی مربوط به ViewState است که اطلاعات در آن قرار می گیرد و نوع آن hidden است و نامش همان ViewState و مقداری عجیب دارد! خیلی ها فکر می کنند این مقدار رمز گذاری شده است در حالی که این طور نیست و اگر شما اطلاعات امنیتی خود را در آن بگذارید هکر ها به سادگی به آنها دسترسی پیدا می کنند.برای بالا بردن امنیت ViewState باید ازصفت EnableViewStateMAC در صفحه استفاده کنید تا عمل حفاظت انجام شود .بهتر است این کار را در خصوصیت تگ <pages> درداخل تگ <system.web> در Web.Config انجام دهید. همچنین برای عمل رمز گزاری روی ViewState از صفت viewStateEncryptionMode استفاده کنید و مقدار آن را مثل فرم زیر Always قرار دهید:

```
  <system.web>
      <pages enableViewStateMac="true" viewStateEncryptionMode="Always"/>
</system.web>
```

در زیرنمونه ای از فیلد مخفی بدون اعمال viewStateEncryptionMode و EnableViewStateMAC آمده است:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwULLTExMjUwMDA5MjBkZNl4Ud3oh12vciFgE+/D6C+46sXS" />
```

و در زیر نونه ای از فیلد مخفی با اعمال viewStateEncryptionMode و EnableViewStateMAC آمده است:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="jV86yUyfgnL9wffdPd7ropliyTYJibG2mCtfIVMHBW4=" />
```

می بینید که Value دردو نمونه ی بالا متفاوت است در حالی که مقدار اصلی ViewState در هر دو یکی است ولی در پایینی این مقدار رمز گذاری شده .به این ترتیب هکر ها نمی توانند به مقادیر ذخیره شده در ViewState های شما به درستی دسترسی پیدا کنند.

می توان این اعمال را در صفحه ی اصلی نیز انجام داد ولی برای کلیت داشتن و پرهیز از کد اضافه و احیانا اشتباه ,بهتر است این دو عمل مهم را در Web.Config انجام دهید.

Query String:

یکی دیگر از تکنیک های مدیریت حالت استفاده از Query String است.که بیشتر برای ارسال داده ها بین صفحات صورت می گیرد.اگر مثلا در google از کلمه ی organic gardening جستجو به عمل بیاورید google به صفحه ی بعدی رفته و لینک زیر را می فرستد:

```
http://www.google.ca/search?q=organic+gardening
```

Query String نیز همین کار را انجام می دهد.

فرمت کلی Query String به صورت زیر است:

```
Response.Redirect("newpage.aspx?var= reza")
```

در این فرمت Response.Redirect نشان دهنده ی ارسال به یک صفحه ی جدید است.newpage.aspx صفحه ای است که مقادیری باید به آن ارسال شود.علامت ؟ نیز بعد از آدرس صفحه ی مبدا ذکر می شود و نشان می دهد که هدف ما ارسال توسط Query String است و بعد از آن نام متغیر ,علامت = و در نهایت مقداری که باید فرستاده شود ذکر شده است.با علامت & می توان چندین مقداررا فرستاد:

```
Response.Redirect("newpage.aspx?var= reza&var1=ali&var2=bahman")
```

مثال زیر یک متغیر تعریف می کند و آن را به صفحه ای خاص می فرستد:

```
  Dim str As String = "."
Response.Redirect("aaa.aspx?ooo=" & str)
```

Response.Redirect همیشه یک مقدار از نوع string به عنوان آدرس صفحه (Url) در یافت می کند و به آن Url می رود.

برای دریافت هم از متد Request.QueryString در صفحه ی مقصد استفاده می کنیم. دقت کنید که حتما باید نام متغیر ارسالی و در یافتی یکی باشد که در اینجا ooo است:

```
Request.QueryString("ooo")
```

اشکال این روش این است که هر چیزی را بفرستید عینا در آدرس بار مرورگر دیده می شود.مثلا در مثال بالا . عینا در آدرس بار مرورگر دیده می شود.که از امنیت کار می کاهد.برای ارسال درست داده بین دو صفحه و جلوگیری از دیده شدن آن در آدرس بار مرورگر باید در طرف ارسال آن را کد گذاری کرده و عمل ارسال را انجام دهیم سپس در صفحه ی دریافت کد آن را واکشی کرده و پیغام را دریافت کنیم.به این منظور از Base64String می توان استفاده کرد برای روشن شدن موضوع با یک مثال توضیح می دهیم:

می خواهیم عبارت .amoo را با عمل کد گزاری به صفحه ی aaa.aspx بفرستیم.ابتدا متغیری تعریف کرده ومقدار .amoo را در آن قرار می دهیم:

```
Dim str As String = ". amoo"
```

سپس برای رمز گزاری متغیری به نام encodetxt از نوع string تعریف می کنیم و عمل تبدیل را با متد Convert و صفت ToBase64String انجام می دهیم:

```
Dim encodetxt As String = Convert.ToBase64String
(System.Text.Encoding.ASCII.GetBytes(str))
```

همانطور که می بینید آرگومان ورودی Convert.ToBase64String داده ای از نوع Byte است پس برای تبدیل متغیر str که از نوع string است و حاوی مقدار .amoo است باید آن را به Byte تبدیل کنیم و سپس به عنوان آرگومان ورودی به Convert.ToBase64String بدهیم .به این منظور از System.Text.Encoding.ASCII.GetBytes استفاده می کنیم که این متد یک مقدار string را به Byte تبدیل می کند.حال کافیست این متغیر رمز گزاری شده را (encodetxt) با Query String بفرستیم:

```
Response.Redirect("aaa.aspx?ooo=" & encodetxt)
```

حال در آدرس بار مرورگر به جای .amoo مقدار رمز گزاری شده به صورت زیر ذکر می شود(زمینه ی آبی):

aaa.aspx?ooo=bWFuaWZlc3QgYW1vbw

حال برای دیکد کردن داده ی رمز شده باید باز از Convert ولی این بار با متد FromBase64String به صورت زیر استفاده کنیم:

```
Dim ttt() As Byte = Convert.FromBase64String(Request.QueryString("ooo"))
```

همانطور که مشخص است چون داده ی ارسالی رمز شده از طریق Base64String بود ابتدا رمز آن با Convert.FromBase64String دیکد شده و به نوع Byte در آمد

که برای استفاده باید در یک متغیر از نوع Byte (در اینجا ttt) قرار گیرد سپس توسط متدی کاملا شبیه متد تبدیل string به Byte این بار از Byte به string تبدیل شود که این متد System.Text.Encoding.ASCII.GetString می باشد که مقدار ورودیش از نوع Byte است و تبدیل شده ی آن را در متغیرstring به نام ID قرار می دهیم :

```
Dim ID As String = System.Text.Encoding.ASCII.GetString(ttt)
```

با این حال سعی کنید از QueryString برای داده های حساس و امنیتی استفاده نکنید.چون باز هم امنیت آن قابل اطمینان نیست از طرفی نمی توان داده های غیر رشته ای را با آن ارسال کرد . مثلا google کلمه ی مورد جستجو را بدون کد کردن می فرستد چون از لحاظ امنیتی موردی ندارد و از نوع رشته است.همینطور شما می توانید اطلاعات یک فرم را با QueryString به صورت زیر بفرستید:

```
Response.Redirect("second.aspx?name=" & myname & "family=" & myfamily & "age=" & myage)
```

امروزه QueryString به مراتب مورد استفاده قرار می گیرد.مثلا شما می توانید یک listBox حاوی نام یک سری مقاله است , ایجاد کنید و به کاربر این امکان را بدهید که با انتخاب هر گزینه از آن لیست به صفحه ی مربوط به آن مقاله ارجاع داده شود و آن را مطالعه کند.به این منظور به جای قرار دادن یک لینک برای تک تک عناصر آن لیست می توانید از QueryString به صورت زیر استفاده کنید:

```
Response.Redirect("aaa.aspx?ooo=" & listbox1.selectedItem.Value)
```

به این ترتیب شما به صفحه ی مربوطه ارسال می شوید بدون اضافه کاری در کد نویسی.البته در صفحه ی مقصد باید به گونه ای بازیابی مقاله ی مورد نظر را بر اساس کلمه ی کلیدی موجود در listBox مشخص کنید به این صورت که پس از ارسال شدن نام مقاله ,یک frame جدید در صفحه ی مقصد (در اینجا aaa.aspx) ایجاد کرده ولینک آن را به صورت زیر مشخص کنید تا مقاله ی بار گزاری شده به درستی بر اساس آیتم انتخاب شده در listBox در آن frame به نمایش در آید:

"Your Web Site Address.../" & ooo & ".aspx"

به این صورت ooo نام مقاله را مشخص می کند .پس صفحه ی مقصد اصلی شما aaa.aspx است که خود حاوی یک صفحه ی دیگر است که با frame ایجاد شده و در آن همان مقاله ای بار گزاری می شود(به وسیله ی لینک Your Web Site Address.../" & ooo & ".aspx") **که شما در listBox صفحه ی مبدا آن را انتخاب کردید.البته می توانید ooo را**

که کلمه ی کلیدی ارسالی از صفحه ی قبل است را عنوان استخراج از پایگاه داده نیز قرار دهید.یعنی مقالات خود را در پایگاه داده ذخیره کنید و سپس آنها را با ooo در صفحه ی aaa.aspx بازیابی کنید تا دیگر به frame و ایجاد لینک نیازی نداشته باشید.

:Cross-Page Posting

یکی دیگر از تکنیک های مدیریت حالت است که تمام خصوصیات یک صفحه را از قبیل عنوان ,نام صفحه ,کنترل ها و... را به صفحه ی دیگر می فرستد.برای بهره گیری از این تکنیک حتما باید از یک صفحه ی خاص به صفحه ای دیگر منتقل شویم تا متد خاص Cross-Page Posting به نام PreviousPage بتواند ویژگی های صفحه ی مبدا را بدست اورد.البته شما برای جلوگیری از هر گونه خطا همیشه یک شرط برای وجود داشتن PreviousPage ایجاد کنید:

```
  If PreviousPage IsNot Nothing Then
      ...
End If
```

این شرط به ما می گوید اگر صفحه ی قبلی وجود داشت وارد if شو.مثلا ساده ترین کدی که می توانید برای Cross-Page Posting بنویسید به صورت زیر است:

```
   If PreviousPage IsNot Nothing Then
      Response.Write("You came from a page titled " & PreviousPage.
Header.Title)
 End If
```

در این کد از متد Header.Title استفاده کردیم تا عنوان صفحه ای را که ازش آمدیم را به ما بدهد.

شما حتی می توانید برای صفحه ای که ازش آمدید هم شرط بگذارید:

```
   If PreviousPage IsNot Nothing Then
      If TypeOf (PreviousPage) Is CrossPage1 Then
       ' (Read some information from the previous page.)
      End If
 End If
```

در این مثال شرط اول که وجود یا عدم وجود PreviousPage را چک می کند.شرط دوم چک می کند اگر نوع صفحه ای که ازش آمدید CrossPage1 باشد وارد ساختار شویم.البته شما در حالت کلی هم می توانید نوع صفحه ی قبلی را با رویداد PreviousPageType در کد صفحه هم مشخص کنید:

```
<%@ PreviousPageType VirtualPath="~/aaa.aspx" %>
```

در این کد نوع صفحه ی قبلی تعیین شده است.این کد جایش در خط دوم کد زیر است.

مثال زیر بهتر مفهوم CrossPage را به شما نشان می دهد.در این مثال ابتدا یک صفحه ی ساده با یک کنترل TextBoxو یک کنترل Button ایجاد کنید و در صفت Button PostBackUrl آدرس صفحه ای مثل q2.aspx را بدهید:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="y.aspx.vb" Inherits="y" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" PostBackUrl="q2.aspx" Text="Button" /></div>
    </form>
</body>
</html>
```

سپس صفحه ی q2.aspx را به برنامه ی خود اضافه کنید و در رویداد Page_Load آن کد زیر را بنویسید:

```
Dim j As TextBox
j = CType(PreviousPage.FindControl("TextBox1"), TextBox)
If Not j Is Nothing Then
  Response.Write(j.Text)
End If
```

خط اول این کد متغیری به نام J از نوع TextBox تعریف می کند.در خط دوم با استفاده از خصوصیت FindControl متد PreviousPage و دادن نامی از نوع String به آن , در حقیقت در صفحه ی قبل که از آن آمدیم به دنبال TextBox1 می گردیم.اگرواقعا وجود داشته باشد آن را با تابع CType به TextBox تبدیل می کنیم و آن رادر متغیر J می ریزیم.سپس چک می کنیم اگر J حاوی

Text باشد آن را در خروجی چاپ کنیم.پس ما توانستیم حتی به Text یک کنترل از یک صفحه ی قبل در صفحه ی دیگر دسترسی داشته باشیم.توجه کنید با نوشتن خط دوم کد بالا دیگر J حاوی تمام صفات TextBox1 از صفحه ی قبل در این صفحه از قبیل ID,ForeColor,BgColor,EnableViewState و ... نیز هست.شما می توانید این کار

را برای کنترل های دیگر نیز انجام دهید. مثلا می توانید این را در نظر بگیرید که مقادیر آن کنترل TextBox۱ که به صفحه ی بعدی ارسال شد اصلا همان username کاربر , جهت Login شدن باشد که اگر Login شد به صفحه ی بعدی بیاید و username آن که همان کنترل TextBox۱ بود کلمه ی کلیدی ما برای استخراج داده های دیگر این کاربر از پایگاه داده شود.به این ترتیب به طریقی امن ,توانستیم username یک کاربر را به یک صفحه ی دیگر ارسال کنیم.

استفاده ی دیگری که می توان از CrossPage کرد Page Validation است.یعنی چک کنیم آیا صفحه ای که ازش آمدیم Validation دارد یا نه.به عنوان مثال یک صفحه به نام crosspage2.aspx ایجاد کنید یک کنترل TextBox یک Button و یک کنترل RequiredFieldValidator کنار TextBox قرار دهید و مقدار ControlToValidate آن را به TextBox مربوط کنید.قبل از ادامه توضیح یک نکته لازم است و آن هم این است که کنترل های تعیین اعتبار حاوی یک صفت به نام EnableClientScript هستند که پیش فرض آن True است.این به این معنی است که در صورت رعایت نکردن قوانین هر کنترل تعیین اعتبار , صفحه بدون PostBackشدن به شما پیغام خطا را می دهد.یعنی خطای شما به Server فرستاده نمی شود و به گونه ی Client-Side از شما خطا گرفته می شود.اگر این صفت را False کنید صفحه PostBack شده و سپس از شما خطا گرفته می شود.در این مثال این صفت را False کنید.صفتPostBackUrl Button را نیز به یک صفحه مثلا y.aspx نسبت دهید.در رویداد Page_load صفحه ی crosspage2.aspx کد زیر را بنویسید:

```
  If Request.QueryString("error") = "true" Then
     Page.Validate()
End If
```

در مورد این کد دوباره توضیح می دهیم.

در رویداد Page_load صفحه ی y.aspx نیز کد زیر را بنویسید:

```
  If PreviousPage IsNot Nothing Then
    If (Not PreviousPage.IsValid) Then
        Response.Redirect(Request.UrlReferrer.AbsolutePath & "?error=true")
    Else
        Response.Write("Thats True ")
     End If
End If
```

در خط دوم این کد شرطی آمده که بیانگر این است که اگر صفحه ای که ازش آمدیم Validate نباشد وارد شرط شو.Validate شدن یک صفحه را با کنترل های تعیین اعتبار می سنجند.با دستور Response.Redirect یک QuerySting به نام error با مقدار True به صفحه ای که ازش آمدیم (Request.UrlReferrer.AbsolutePath) می فرستیم. حال اگر Validate بود یعنی وارد else شدیم , که در آنجا در صفحه ی y.aspx پیغام Thats True را می نویسیم.حال که در این مثال ما از کنترل RequiredFieldValidator استفاده کردیم , کاربر مقداری را در فیلد وارد نکند و روی دگمه کلیک کند یعنی به صفحه ی y.aspx می رود بدون دیدن هیچ واکنشی از سوی RequiredFieldValidator.لازم به ذکر است که اگر صفت EnableClientScript را در کنترل RequiredFieldValidator برابر False تنظیم کنید رویداد PostBackUrl Button به پیغام خطا ی RequiredFieldValidator ارجعیت داده می شود و صفحه منتقل می شود.حال با Request.UrlReferrer.AbsolutePath ما به دلیل Validate نبودن صفحه ای که ازش آمدیم ّبه همانجا برمی گردیم.ولی این بار با یک QuerySting به نام error با مقدار True.این QuerySting به این درد می خورد که ما وقتی به رویداد Page_load صفحه ی crosspage2.aspx رسیدیم این QuerySting در خواست می شود .اگر مقدار error در آن True بود یعنی صفحه Validate نیست و اینجاست که با کد Page.Validate() کاری می کنیم که RequiredFieldValidator فعال شود و پیغام خطای خود را مبنی بر اینکه چرا در TextBox مقداری وارد نکرده اید را ظا هر می کند.پس crosspage در زمینه ی Validate نیز کاربرد دارد.

قبول داریم کمی پیچیده شد.ول یاگر به کلیت نگاه کنید و خودتان آن را اجرا کنید می بینید چقدر ساده است.

:Cookie

مهم تر از ۳ مورد تکنیک مدیریت حالت ذکر شده ,cookie است.Cookie یک فایل کوچک است که هر کاربر با ورودش به سایت ایجاد می کند.البته این ایجاد بدون اطلاع وی در روی هارد درایو او ساخته می شود و نشان دهنده ی آن است که وب سایت ما از این به بعد این کاربر را می شناسد.البته تا زمانی که کوکی منقضی نشود.چون هر کوکی دارای ExpireTime می باشد.

ساخت یک کوکی ۳ مرحله دارد:

۱-ایجاد شی کوکی

```
Dim newcookie As New HttpCookie("mycookie ")
```

۲-دادن مقداربه آن

```
Newcookie.Value= "vb"
```

۳-اضافه کردن ان به صفحه

```
Response.Cookies.Add(newcookie)
```

در مر حله ی اول متغیرnewcookie را new کردیم و یک رشته برای شناختن آن به نام mycookie دادیم.با این کار یک نمونه از کلاس HttpCookie ایجاد کردیم.در مرحله ی دوم مقداری به کوکی دادیم و در مرحله ی آخر با شی Response آن را به وب سایت Add کردیم.توجه شود که اگر چه کوکی یک فایل متنی است که در کامپیوتر Client ذخیره می شود ولی حتما باید دارای Value باشد که Value جزیی از آن فایل متنی به حساب می آید.

می توان زمان انقضا نیز به کوکی داد البته قبل از Add کردن به وب سایت:

```
newcookie.Expires = DateTime.Now.AddMinutes(2)
```

این کد یعنی از الان تا ۲ دقیقه دیگر کوکی منقضی می شود.

طرز استفاده از کوکی نیز با شی Request صورت می گیرد:

```
   If Not Request.Cookies("mycookie") Is Nothing Then
      Response.Write("You Are a My Web Site Member")
   Else
      Response.Write("You Are Not a My Web Site Member")
 End If
```

در این کد درخواست کوکی بر اساس نام آن صورت می گیرد نه متغیری که کوکی درآن ایجاد شده .اگر کوکی وجود داشت یعنی کاربر عضو سایت است و اگر نداشت یعنی کاربر عضوسایت ما نیست.

برای امتحان , کافیست تمام کد های بالا را اسمبل کرده و در رویداد Page_Load یک صفحه قرار دهید:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    If Not Request.Cookies("mycookie") Is Nothing Then
       Response.Write("You Are a My Web Site Member")
    Else
       Response.Write("You Are Not a My Web Site Member")
    End If
    Dim newcookie As New HttpCookie("mycookie")
    newcookie.Value = "vb"
    newcookie.Expires = DateTime.Now.AddMinutes(2)
```

```
    Response.Cookies.Add(newcookie)
End Sub
```

اگر این صفحه ی ساده را اجرا کنید در ابتدا با اجرا شدن شر ط اول ,چون هنوز کوکی ساخته نشده و درخواست انجام می گیرد, پس با پیغام You Are Not a My Web Site Member مواجه خواهید شد, ولی اگر صفحه را تا قبل از ۲ دقیقه(اتمام زمان فعالیت کوکی) بسته و دوباره اجرا کنید چون در دفعه ی اول اجرای صفحه, کوکی ساخته شده بود, پاسخ سرور به وجود کوکی مثبت است و شما با پیغام You Are a My Web Site Member مواجه می شوید.حال اگر پس از گذشت ۲ دقیقه وارد سایت شوید دوباره پیغام You Are Not a My Web Site Memberظاهر می شود چون از تاریخ انقضای کوکی دیرتر وارد سایت شدید.برای پاک کردن کوکی کافیست کد زیر رابنویسید:

```
Request.Cookies.Remove("Mycookie")
```

البته در بعضی منابع ذکر شده که تنها راه از بین رفتن کوکی دادن تاریخ منفی به انقضای آن است:

```
cookie.Expires = DateTime.Now.AddDays(-1)
```

نام کوکی معرف حضور کوکی در وب سایت است.در حقیقت ما به نام کوکی برای در خواست , نیاز داریم مثلا :

Request.Cookies("cookiename")

ولی برای تعریف یا تعیین زمان اعتبار و Add کردن آن به وب سایت , با Value آن کار داریم.الان کوکی که ما با نام mycookie و newcookie Value در مثال بالا ساختیم حاوی داده هایی به صورت زیر است:

```
mycookie
vb
/localhost
١٠٢٤
٩٣٢٩٠٥٤٧٢
٢٩٨٦٩٢١٨
٤٠٣٧٤٠٢٧٦٨
*٢٩٨٦٩٢١٧
```

البته این رشته ها در اصل پشت سر هم و بدون هیچ فاصله ای است.میبینید که نام کوکی اولین کلمه ی آن است و Value آن که vb است پس از آن آمده و بعد نیز آدرس سایت و پورت آن آمده است.

یادتان باشد که در اصل کاربرد کوکی با Login شدن یک کاربر به ب سایت معنا پیدا میکند.به این دلیل که کاربرد کوکی بسیار وسیع است و در Login شدن کاربرا ن کاربرد بیشتری دارد در اینجا تا مباحث مربوط به پایگاه داده و پروفایل و مجوز ها و... گفته نشود مطرح کردن مثا ل کاربردی کوکی امکان پذیر نیست.پس از آموزش ADO.Net و مباحث مهم دیگر به طور عملی با کاربرد کوکی آشنا خواهیم شد.

:Session

شی Session یکی ازپر اقتدار ترین تکنیک های مدیریت حالت است.می توان اطلاعات را در آن ذخیره کرد و در تمام صفحات از آن استفاده کرد .بر خلاف شی ViewState که تنها در صفحه ای خاص بود ویا QueryString,Cross Page که حتما باید از یک صفحه به صفحه ی دیگر ارجاع داده می شد و تنها داده های رشته ای را در خود ذخیره می کردند و حتی Cookie که یک فایل متنی ایجاد می کرد,Session به سادگی ایجاد شده و در تمامی صفحات از آن می توان استفاده کرد البته Cookie می تواند جزیی از Session باشد یا نباشد که این امر بستگی به تنظیم صفات Session در فایل web.config دارد که جلوتر به پیکربندی Session که رسیدیم متوجه خواهید شد .البته این شی حتما جهت امنیت بیشتر, باید پس از خروج کاربر و یا پس از مدتی معین از بین برود. پس می فهمیم این شی هم مفهوم timeOut را داراست و شی پایدار پس از بسته شدن برنامه نیست(نباید هم باشد).این شی در سرور ذخیره می شود.ÅSP.NET هر شی Session را از طریق یک شناسه ی ۱۲۰ بیتی ردیابی می کند.این ۱۲۰ بیت به صورت اختصاصی توسط خود ASP.NET تولید می شود.این مقدار بین کاربران مختلف Unique و کاملا هم تصادفی می باشد و کاربر نمی تواند از طریق مهندسی معکوس به روش تولید آن پی ببرد.این شناسه بخشی از اطلاعات مربوط به شی Session است.وقتی شما در برنامه ی خود از شی session استفاده می کنید باعث تحریک رویداد ()Session_start می شوید و این امر باعث می شود که AsP.Net یک جدول در سرور ایجاد کند به نام Session Table که هر کاربری که شی Session را صدا بزند آنگاه یک سطر جدید از این جدول به آن کاربر تخصیص داده می شود و اگر کاربر از قبل یک سطر در Session داشته باشد دیگر سطر جدیدی برایش ایجاد نخواهد شد.Session Table دارای یک کلید اصلی و Unique است که همان شناسه ی ۱۲۰ بیتی در آن ذخیره می شود.هر بار کاربری از شی Session استفاده کند یک سطر جدید برایش در جدول Session ایجاد می شود و یک

شناسه ی ۱۲۰ بیتی مخصوص وی در آن سطر به عنوان کلید اصلی برایش ذخیره می شود.Session مفهومی مثل Hashtable دارد و می توان هر چند تا داده در آن ذخیره کرد (البته نباید داده ها زیاد باشد چون باعث کم شدن کارایی سرور می شود) و این داده ها هر کدام در سطری که به کاربر تخصیص داده شده بود ذخیره می شود.برای مثال شکل زیر یک نمای فرضی از جدول Session است:

Session ID	Session Data
ryw5r3b4knskos45c00f055	name='ali' lname='jafari' user name='ajafari'
qs35r3b4knske4tgg30f78i	name='asma' lname='mirzaee' color='yellow' username='amirzaeepp' email='asma@uahoo.com'
⋮	⋮

در جدول Session بالای تولید شده در سرور, دو کاربر وجود دارند که هر کدام دارای یک SessionID منحصر به فرد و حتی داده های متفاوتی هستند که این باعث می شو ملا بدانیم در حال حاضر کاربر ali به قسمت رنگ مورد علاقع رجوع نکرده و رنگی را برای خود انتخاب نکرده و یا مثلا ali هنوز آدرس پست الکترونیک خود را جهت ارسال اخبار سایت ثبت نکرده(همه این ها مثال است).پس در اینجا می فهمیم که شی Session می تواند باعث ردیابی کاربر نیز شود.

حال حتما از خود می پرسید که سرور چگونه می تواند این اطلاعات را ردیابی کند.بسیار ساده است.پس از صدا زدن رویداد Session_start توسط کاربر, سرور پس از تولید شناسه ی ۱۲۰ بیتی آن را در قالب یک کوکی در کامپیوتر کاربر ذخیره می کند به نام ASP.NET_sessionid و با استفاده از آن شناسه می تواند به سایر عناصر جدول Session دسترسی پیدا کند.همانطور که در پایگاه داده تا مقدار کلید اصلی را ندانید نمی توانید مقادیر سایر فیلد های سطر خاص را بازیابی کنید در اینجا هم سرور برای ردیابی مثلا نام کاربری شما به سادگی مقدار موجود در کوکی شما را که در کامپیوتر شماست بازیابی می کند و آن را با جدول Session تطبیق می دهد و سطر مورد نظر که اطلاعات شما در آن است را پیدا می کند.حال اگر شما این کوکی را پاک کنید یا به هر نحوی این کوکی از بین برود در حقیقت رابطه ای بین شما وجدول Session نخواهد بود و سرور که

می بینید شناسه ای در کامپیوتر کاربر ذخیره نشده, سطر مربوط به اطلاعات شما را از جدول Session موجود در سرور پاک می کند.حتما تا حالا متوجه شدید که مکانیسم Session بهترین راه برای Login & Logout کردن کاربران است و اینکه بسیاری از سایت ها تعداد کاربران Online را از روی شمارش سطر های جدول Session بدست می آورند.چون Session در قالب کوکی در کامپیوتر شما ذخیره شد , با بستن صفحه ی سایت در مرورگر و یا TimeOut شدن کوکی,Session هم از بین می رود.اگر شما کوکی را در کامپیوتر خود غیر فعال کنید ASP.NET از مکانیسم دیگری در اصطلاح Mung جهت استفاده از Session استفاده می کند .پس شما در این جالت هم اطمینان دارید که اگر کاربر کوکی را در کامپیوتر خود غیر فعال کند هم شی session به درستی کار می کند ولی با سربار بیشتر و پایین آمئ سعت و کارایی سرور.

ولی در هر حالت کوکی سریعتر از Session است زیرا در سرور دیگر جدول Session نمی سازد در حالی که همین جدول Session کلی اطلاعات اضافی ئبه سرور تحمیل می کند و باعث کم شدن کارایی آن می شود پس سعی کنید کمترین مقدار داده ها را در شی Session ذخیره کنید.شی Session در ٤ حالت می تواند از بین برورد:

١-با بستن مرورگر توسط کاربر

٢-اگر کاربر همان صفحه را با مرورگر دیگری باز کند.

٣-وقتی شی TimeOut Session شود.

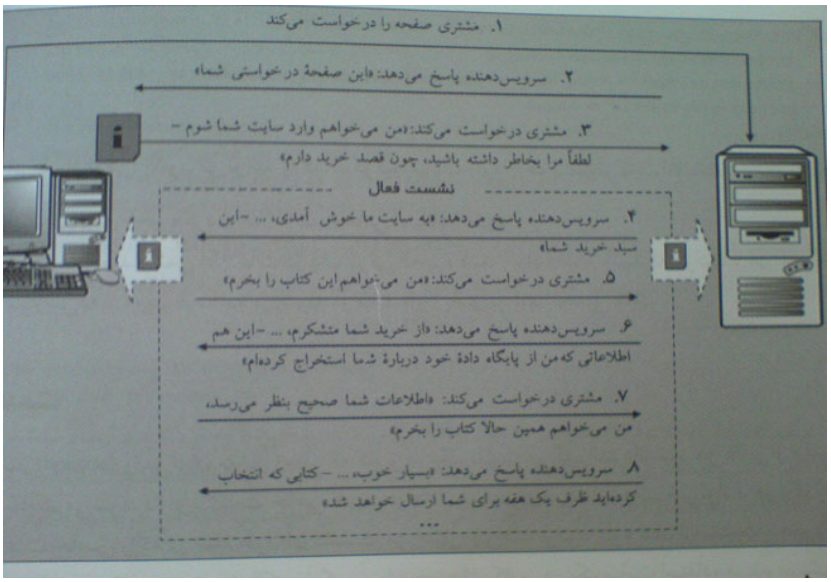
٤-وقتی برنامه نویس از متد هایی جهت از بین بردن شی Session استفاده کند.

٥-وقتی وب سایت به روز می شود یا سرور دوباره راه اندازی می شود.

در دو حالت اول شی Session هنوز در حافظه ی وب سرور وجود دارد ولی دیگر قابل دسترسی نیست و وقتی این اطلاعات از بین می رود که زمان اصلی TimeOut شی Session فرارسد.در مورد مورد پینج هم بگویم که حتما دیدیه اید سایت هایی که در حال بروز رسانی هستند در آن لحظه ی خاص امکان استفاده از آن سایت ها نمی باشد.در چنین حالت هایی باید شی Session را بیرون از این فرایند ها استفاده کنید که در این زمینه هم توضیح خواهیم داد.

کاربرد اصلی Session در تجارت الکترونیک است.وقتی یک مشتری وارد سایتی می شود اطلاعات سبد خرید وی در Session ذخیره می گردد تا هم در تمامی صفحات کاربرد داشته باشد و هم یک ارتباط بین کاربر و سایت ایجاد گردد.ولی کمتر از شی Session در

Login شدن و اطلاعات امنیتی تر مربوط به کاربر استفاده می شود .در آن مباحث , از Cookie استفاده می شود.نحوه ی کار کرد Session در شکل زیر جهت خرید یک کتاب توسط کاربر کاملا واضح است:



به جهت فلش های رسم شده بین سرور و کاربر دقت کنید.

اشیا مختلفی را می توان در شی Session قرار داد.به عنوان مثال در زیر یک شی ا ز کلکسیون HashTable به نام Create را درون یک شی Session به نام cart قرار دادیم:

```
    Dim create As New Hashtable()
    create.Add("a", 1)
    create.Add("b", 2)
Session.Add("cart", create)
```

جهت باز یابی هم باید یادتان باشد که این شی Session حاوی یک کلکسیون HashTable است پس باید آن را به این کلکسیون تبدیل کرد و سپس آن را در متغیری از نوع HashTable قرار داد و سپس عمل بازیابی را انجام داد.چون اینجا هم مثل ViewState برای اشیایی که خودمان ایجاد می کنیم ویا کلکسیونهای VB.NET نیاز به Serializable شدن می باشد. یعنی وقتی ما یک شی مثلا HashTable را در Session قرار می دهیم Session آن شی HashTable را به Bye تبدیل می کند و ذخیره می کند.پس جهت بازیابی نیز باید عکس این عمل را انجام دهیم یعنی Session را به آنچه که به آن تبدیل شده تبدیل کنیم که این اعمال با تابع CType صورت می گیرد:

```
Dim retrieve As New Hashtable()
retrieve = CType(Session("cart"), Hashtable)
```

```
Response.Write(retrieve("a") & "<br>" & retrieve("b"))
```

البته چون ما HashTable را new کردیم نیاز به Serializable شدن داشت اگر آن را new نکنید و از نوع System.Collections.Hashtable تعریف کنید یک راست بدون تبدیل نوع می توانستید Session را در آن قرار دهید:

```
Dim retrieve As System.Collections.Hashtable = Session("cart")
```

همیشه یادتان باشد که اطلاعات زیادی را درSession قرار ندهید چون سرور را تحت فشار قرار می دهد.در تجارت الکترونیک هم از HashTable یا ArrayList برای اطلاعات کاربر در Session استفاده می شود نه متغیر های معمولی!.

تا اینجا با متد Add در Session آشنا شدید که کار اضافه کردن آیتم به Session را بر عهده داشت.حال برای پاک کردن شی Session می توان از صفت Remove استفاده کرد.این صفت یک آیتم از شی Session را پاک می کند مثلا ما در بالا یک شی Session به نام "Cart" ایجاد کردیم و حالا آن را پاک می کنیم:

```
Session.Remove("cart")
```

ولی اگر بخواهید تمام آیتم های Session را پاک کنید بایداز متد Clear استفاده کنید.این متد در عین حال که تمام آیتم ها را از بین می برد ولی شی Session را فعال نگه می دارد و حافظه ای که اشغال کرده(بستگی به صفت Mode دارد که دوباره به آن اشاره می کنیم(مربوط به تگ <sessionState> در web.config می شود) و مکان ذخیره شدن Session را مشخص می کند) سر جایش باقی می ماند:

```
Session.Clear()
```

ولی اگر بخواهید اصلا شی Session را از بین ببرید از متد Abandon استفاده کنید تا Session هم غیر فعال شود و تمام حافظه ی اشغال شده توسط Session از بین برود و اصطلاحا releases all the memory رخ دهد .این متد جهت اتمام عملیات یک کاربر با Session کاربرد دارد:

```
Session.Abandon()
```

صفت Timeout هم مدت زمان بر حسب دقیقه ی انقضای Session می باشد که پس از اتمام آن همان عملیات شرح داده شده درAbandon انجام می گیرد.به طور پیش فرضTimeout ۲۰ دقیقه در نظر گرفته شده است (تنظیم زمان آن در فایل Web.Config است که دو باره به آن اشاره می کنیم که مربوط به تگ <sessionState> در web.config می شود) :

```
Response.Write(Session.Timeout & "<br>")
```

تنها صفتی که می توان مقدار آن را در صفحه ی اصلی نیز تعیین کرد Timeout می باشد:

```
Session.Timeout = ۲۰
```

به عنوان مثال کد زیر را در رویداد Page_Load یک صفحه قرار دهید:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
      If Session("cart") IsNot Nothing Then
           Response.Write(Session("cart").ToString)
      Else
           Response.Write("No session")
           Session.Add("cart", "rrr")
      End If
End Sub
```

در این کد اگر برای اولین بار این صفحه را اجرا کنید می بینید که پیغام No session روی صفحه آمده.حال اگر صفحه را Refresh کنید مقدار آن rrr خواهد بود.چون در دفعه ی اول Session ساخته نشده بود و پاسخ به درخواستآن منفی بود.

دقت کنید چون Session در اینجا یک شی متصل به سرور است اگر کد بالا را برای بار اول اجرا کنید و سپس ببندید و دوباره اجرا کنید باز هم پیغام No session می آید .تنها در صورت Refresh کردن صفحه , rrr در آن نمایش داده می شود.پس دیدیم در کل , شی Session, ٤ متد داشت که میشد از صفحه ی اصلی آن را کنترل کرد(یعنی حذف یا اضافه کرد).

از ویژگی های مهم Session می توان به موارد زیر اشاره کرد که همه در صفحه ی اصلی ReadOnly می باشند(و می توان آنها را در Web.Config تنظیم کرد که دوباره به آن اشاره می کنیم):

:SessionID

شناسه ای ۱۲۰ بیتی و Unique که جهت شناساندن Session به وب سایت استفاده می شود و پیش از این مفصل توضیح داده شد و در این جا می توان آن را بازیابی کرد:

```
Response.Write(Session.SessionID)
```

نمونه ای از خروجی این شناسه ی ۱۲۰ بیتی:

```
22jnjcsblhp5utp55rdks5l55
```

:Count

تعداد آیتم های شی Session می باشد.همان تعداد عناصری که مخصوص یک کاربر خاص ذخیره شده:

```
Response.Write(Session.Count)
```

IsCoookieLess:

مشخص می کند آیا Session با کوکی کار می کند یا خیر.مثلا خروجی کد زیر false است .یعنی Session با کوکی کار می کند:

```
Response.Write(Session.IsCookieless)
```

Mode:

مشخص می کند Session در کجا ذخیره شده .

برای تست درست این ویژگیها کد زیر را در رویداد Page_Load یک صفحه نوشته و آن را ببینید:

```
    Dim create As New Hashtable()
    create.Add("a", 1)
    create.Add("b", 2)
    Session.Add("cart", create)
    Dim retrieve As New Hashtable()
    retrieve = Ctype(Session("cart"), Hashtable)
    Response.Write(retrieve("a") & "<br>" & retrieve("b"))
    Response.Write(Session.SessionID & "<br>")
    Response.Write(Session.Count & "<br>")
    Response.Write(Session.IsCookieless & "<br>")
  Response.Write(Session.Mode & "<br>")
  Response.Write(Session.Timeout & "<br>")
```

نکته ی قابل ذکر این است که این ویژگی ها همه قابل تنظیم هستند.و تنها در صفحه ی اصلی است که ReadOnly هستند تا دور از دسترس کاربر باشند.همه ی این تنظیم ها در تگ <SessionState> در فایل Web.Config قرار دارند.مثلا در آنجا می توان با صفت Mode این را مشخص کرد که Session در کجا ذخیره شود.

تگ <SessionState>: حالت کلی این تگ و آیتم های مهم آن را در زیر مشاهده می فرمایید که خود گویای همه چیز هست. ولی در زیر در مورد تک تک آنها صحبت می کنیم:

```
<sessionState
  mode="[Custom|InProc|Off|StateServer|SQLServer|]"
  stateConnectionString="String"
  stateNetworkTimeout="Integer"
  sqlConnectionString="String"
  sqlCommandTimeout="Integer"
  allowCustomSqlDatabase="[true|false]"
  cookieless="[AutoDetect|UseCookies|UseDeviceProfile|
              UseUri|true|false]"
  cookieName="String"
```

```
    timeout="Integer"
</sessionState>
```

مهمنرین خصوصیت که در خط اول کد بالا ذکر شده mode است.این خصوصیت نحوه ی مدیریت session و اینکه این شی کجا ذخیره شود را مشخص می کند.مهمترین صفت و صفت پیش فرض آن نیز InProc است.

InProc:

این صفت محل ذخیره شدن session را درcurrent application domain تعیین می کند.یعنی تمام اطلاعات session در همان برنامه ی جاری(همان سرور میزبان) ذخیره می شود.که این امر کارایی بالا و دوام پایینی را برای برنامه ی ما به همراه دارد.چون مثلا با Restart شدن سرور اطلاعات session نیز از بین می رود چون بدون شک قسمتی از برنامه ی ما در سرور پردازش می شود(مثلا موارد مربوط به ثبت نام و ورود کاربران که اطلاعاتشان در پایگاه داده ی سرور ذخیره شده و آنجا پردازش می شود و Server-Side است نه Client-Side) و session هم که در برنامه ی ما ست.پس اگر سرور Restart شود یا به هر دلیلی دچار نقص شود session از بین می رود.پس در کل این صفت برای زمانهایی مثل Restart Server یا Update Page و... که امکان Lost شدن session درآنها هست.مناسب نیست چون بالاخره برنامه ی جاری ما می تواند هر لحظه دچار مشکل شود و session کاربر از بین برود و وی از سایت ما ناراضی شود.

State Server:

جهت امنیت بیشتر می توان session را در یک separate Windows service ذخیره کرد.یعنی یک سرور جداگانه .خوب خوبی این کار در این است که دیگراز Lost شدن session نگران نیستیم و session در یک سرور جداگانه ذخیره شه که البته این سرور نقش یک Windows service را بازی می کند که همگام با سرور اصلی سایت اجرا می شود.بدی این کار این است که یک عمل انتقال باید بین دو سرور جهت انتقال session صورت گیرد و از طرف دیگر به اصطلاح Time Delay یا تاخیر زمانی برای انتقال بین دو پردازش افزایش می یابد و ازطرف دیگر هزینه برای ایجاد یک سرور دیگر هم در بر دارد.برای تنظیم سرور جهت State Server به control Panel رفته و Administrative Tools را اجرا کنید.در صفحه ی باز شده Computer Management را اجرا کرده و از لیست سمت چپ آن Service And Application

را Expand کنید.سپس روی گزینه ی Service در آن کلیک کنید و در لیست باز شده سمت چپ به دنبال ASP.NET State Service گشته و برای فعال کردن رویش کلیک راست کرده و سپس start را انتخاب کنید.تا تنظیم صورت گیرد.برای cancel کردن تنظیمات هم رویش کلیک راست کرده و سپس stop را انتخاب کنید.جهت تنظیمات بیشتر رویش کلیک راست کرده و سپس Properties را انتخاب کنید تا وارد پنجره ی جدیدی جهت تنظیمات آن شوید.

Sql Server:

همانطور که از نامش بر می آید جهت ذخیره کردن session در پایگاه داده ی Sql Server است.

Custom:

برای سفارشی کردن session کاربرد دارد و در محدوده ی بحث asp.net نیست.

Off:

برای غیر فعال کردن session است.

در خط دوم کد پیکربندی session, خصوصیت stateConnectionString وجود دارد که مقداری رشته ای را می پذیرد.این ویژگی وقتی فعال است که صفت mode در پیکربندی session با State Server مقداردهی شده باشد و حاوی دو قسمت است یک کد آدرس TCP/IP آن سروری که قرار است session به طور جداگانه در آن ذخیره شود و قسمت دوم شماره ی پورت آن است که شماره ی در گاه میزبان سایت است و امکان همزمان اجرا شدن دو سرور را میدهد و مقدار پیش فرض آن ۱۲۷.۰.۰.۱ می باشد.در زیر نمونه ای از stateConnectionString را می بینید:

```
stateConnectionString="tcpip=127.0.0.1:42424"
```

در خط سوم کد پیکربندی session, خصوصیت stateNetworkTimeout می باشد . این ویژگی وقتی فعال است که صفت mode در پیکربندی session با Sql Server مقداردهی شده باشد و مدت زمانی را نشان می دهد که در هر بار اجرای session در برنامه ی ما , سرور میزبان اطلاعات session , می تواند منتظر درخواست session از سرور اصلی بماند که پس از گذشت آن به در خواست session مورد نظر پاسخ نمی دهد.بر حسب ثانیه است و مقدارپیش فرض آن ۱۰ ثانیه می باشد:

```
stateNetworkTimeout="10"
```

خط بعدی کد پیکربندی session ,خصوصیت sqlConnectionString است که همان رشته ی اتصال به پایگاه داده است و این ویژگی وقتی فعال است که صفت mode در پیکربندی session با Sql Server مقداردهی شده باشد:

```
sqlConnectionString="data source=127.0.0.1;Integrated Security=SSPI"
```

خط بعدی کد پیکربندی session ,خصوصیت sqlCommandTimeout است که وقتی فعال است که صفت mode در پیکربندی session با Sql Server مقداردهی شده باشد. ومدت زمانی را نشان می دهد که در هر بار اجرای session در برنامه ی ما , پایگاه داده ای که حاوی اطلاعات session است, می تواند منتظر درخواست session بماند که پس از گذشت آن به در خواست session مورد نظر پاسخ نمی دهد.بر حسب ثانیه است و مقدارپیش فرض آن ۳۰ ثانیه می باشد:

```
sqlCommandTimeout="30"
```

اگر می خواهید اطلاعات session را در یک پایگاه داده ی دیگر ذخیره نمایید یعنی پایگاه داده ای که خودتان آن را ایجاد کردید ,صفت بعدی یعنی allowCustomSqlDatabase را برابر True قرار دهید.که در این صورت باید به رشته ی اتصال پایگاه داده ی خود در کد پیکربندی session عنصر catalog را بیفزایید و مقدارش را با نام پایگاه داده ی خود تان مقدار دهی کنید ک در اینجا CustDatabase می باشد:

```
  <sessionState allowCustomSqlDatabase="False" sqlConnectionString=
  "data source=127.0.0.1;Integrated Security=SSPI;Initial
Catalog=CustDatabase"
… />
```

صفت بعدی CookieLess است.این صفت استفاده شدن یا نشدن Cookie را در session مشخص می کند که حاوی ٦ گزینه ی زیر است:

:UseCookies

این صفت مشخص می کند که session با کوکی کار کند حتی به این قیمت که مرورگر وب کوکی را پشتیبانی نکند و یا خود کاربر آن را غیر فعال کند که در این صورت مقدار session از بین می رود.این صفت مقدار پیش فرض UseCookies می باشد.

:UseUri

این صفت می گوید از کوکی هرگز استفاده نشود صرف نظر از توانایی های مرور گر یا تنظیمات کاربر.و به جای کوکی SessionID را در URL مرورگر ذخیره می کند.این مورد برای مواقعی است که کوکی کاربر Disable است.

UseDeviceProfile:

هنگامی که Asp.Net با توجه به قابلیت های مرورگر وب تشخیص دهد که این مرورگر کوکی را Support می کند یا نه اگر می کند از کوکی استفاده شود.

Asp.Net:AutoDetect تشخیص می دهد که اگر مرورگر کوکی را Support کند از کوکی استفاده شود و اگر نکند از SessionID در URL مرورگر استفاده شود.که این حالت کامل تر از UseDeviceProfile است.

نمونه ای از لینکی که SessionID در URL مرورگر استفاده شده(پس زمینه ی آبی):

http://localhost/WebApplication/(amfvyc55evojk455cffbq355)/Page1.aspx

دو صفت دیگر به نام True,false وجود دارد که به ترتیب اثر UseUri و UseCookies را دارند.

در خط بعدی پیکر بندی Session نام کوکی(cookieName) مد نظر است.چنانچه کوکی با ویژگی CookieLess فعال باشد آنگاه می توانید نامی به آن بدهید که نام پیش فرض آنASP.NET_SessionId می باشد.

خط بعدی کد پیکربندی session ,خصوصیت TimeOut می باشد که زمان انقضای Session یا دراصل Abandon شدن آن می باشد و بر حسب دقیقه است و پیش فرض آن ۲۰ می باشد.این تنها صفتی است که می توان از داخل صفحه ی اصلی(VB.NET) هم آن را تنظیم کرد:

```
Session.Timeout = ۲۰
```

در زیرنمونه ای از پیکر بندی Session را میبینید:

```
<sessionState
 mode="InProc"
 stateConnectionString="tcpip=127.0.0.1:42424"
 stateNetworkTimeout="10"
 sqlConnectionString="data source=127.0.0.1;Integrated Security=SSPI"
 sqlCommandTimeout="30"
 allowCustomSqlDatabase="False"
 cookieless="UseCookies"
 cookieName="ASP.NET_SessionId"
 timeout="20"
/>
```

البته پیکر بندی Session ویژگی های دیگری نیز دارد که در اینجا به اهم آنها اشاره شد.

اگر چنانچه Session شما با کوکی همراه شد می توان در بازیابی به آن کوکی به صورت زیر امنیت بخشید:

```
Request.Cookies("ASP.NET_SessionId").Secure = True
```

البته راه درست آن این است که هنگام ایجاد SessionID, session آن را رمز گزاری کرد(session ID is encrypted).

Session در فایل Global.asax دو رویداد دارد که پیش تر به آنها اشاره شد و میتوان متناسب با آنها دستوراتی را به ان اضافه کرد.مثلا برای بدست آوردن تعداد کاربران انلاین اگر از مکانسیم Session استفاده کرده باشید می توانید با هر بار تحریک رویداد Session_start یکی به مقدار یک متغیری که در شی Application(جلوتر می گوییم) بود اضافه شود و با هر بار Session_end شدن هم یکی از آن کم شود.

حال برای یک مثال کلی طرز استفاده از Session, یک فروشگاه تجارت الکترونیک ایجاد می کنیم:

البته در این مثال Session بخشی از کار است و ما با بسیاری موارد چه در کنترل های ASP.NET و چه در زبان VB.NET آشنا می شویم.ما می خواخیم صفحه ای ایجاد کنیم که یک سری آیتم برای فروش داشته باشد.هر یک از این آیتم ها حاوی نام , توضیحات , قیمت ,تصویر و دکمه ای جهت خرید می باشند.ما به عنوان کاربر نیز می توانیم هر کدام از آنها را به تعداد دلخواه خریداری کنیم.در نهایت یک لیست کلی از اجناس خریداری شده به همراه تعداد آنها و قیمت تک تک و کل آنها به ما ارایه می شود .ما فعلا تا همینجا پیش می رویم تا وقتی با طریقه ی Login شدن کاربران و همچنین کنترل مهم Wizard آشنا شدید ادامه ی آن را نیز که شامل عضو شدن در سایت و طی مراحل پرداخت پول الکترونیکی است را خواهیم گفت .البته به جز پرداخت پول الکترونیکی!.این را هم بگویم که این اجناس در داخل یک پایگاه داده هستند و ما می بایست آنها را بازیابی کنیم و چون هنوز با ADO.NET آشنا نشد یم از کنترل های آماده ی Visual Studio.NET مثل SqlDataSource استفاده می کنیم.این کنترل ها با اینکه کار ما را خیلی راحت می کنند ولی از دو جهت جالب نیستند.

یک اینکه حجم زیادی را می گیرند و سرعت کار پایین می آید و دوم اینکه مارا برای یک سری عملیات محدود می کنند.نمونه ای از خروجی برنامه را که عمل خرید انجام شده و دکمه ی Check Out که نشانه ی اتمام عملیات خرید و آغاز عضویت و پرداخت پول است زده شده تا صورت حساب کلی به ما عرضه شود را می بینید :

همانطور که می بینید آیتم های مورد فروش در سمت چپ صفحه موجود هستند که با حرکت دادن Scroll Bar می توان تمام آنها را دید.در سمت راست آنها آیتم ها ی انتخاب شده توسط ما و تعداد آنها و قیمت آنها به تفکیک ذکر شده و اگر پس از خرید(زدن دکمه ی Add To Basket) روی دکمه ی Check Out کلیک کنید ضمن تشکر , تعداد آیتم های خریداری شده(به تفکیک نوع آنها یعنی حذف کردن تکراری ها) و همچنین قیمت قابل پرداخت Payable Price را به شما می دهد.

برای شروع بهتر است ابتدا با ظاهر سایت کار کنیم .یعنی طریقه ی قرار گرفتن تصاویر و محتویات هر آیتم و دکمه ی Add To Basket.ما برای این کار از یک کنترل DataList استفاده کردیم.برای شروع شما اوا باید یک جدول مخصوص این کار به پایاه

داده ی خود ایجاد کنید و ستون های آن را عناصر زیر تعریف کنید.ما جدولی به نام Shop به صورت زیر تعریف کردیم:

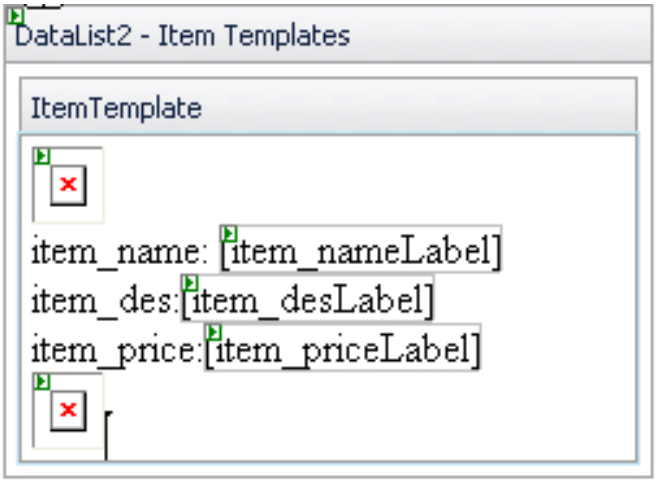
Column Name	Data Type	Allow Nulls
item_no	int	
item_name	nvarchar(25)	
item_des	varchar(MAX)	
item_price	int	
keyword	char(6)	

item_no کلید اصلی و مقداری عددی و شناسه ی هر محصول است که در جدول نباید تکراری باشد.

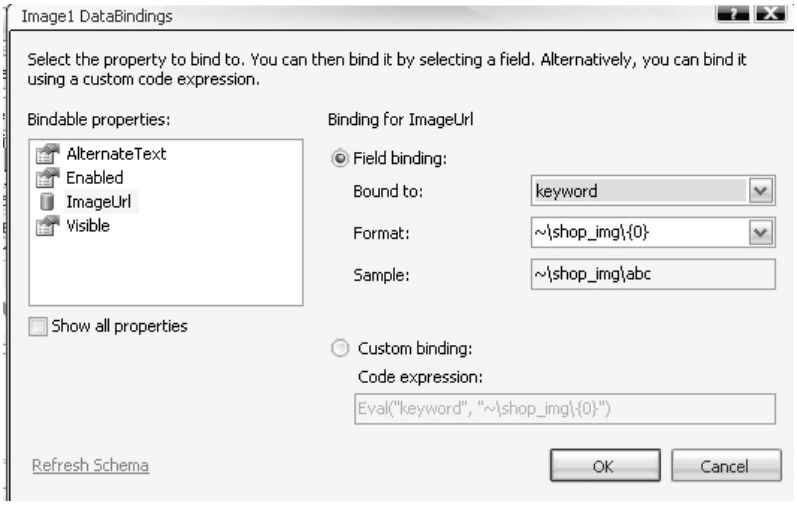
item_name,item_des,item_price به ترتیب قیمت ,توضیح و نام هر کالا است.و در نهایت keyword حاوی نام و پسوند تصاویر است که ٦ کاراکتری است وشما می توانید هر تعدادی بگیرید.مثلا نام یک تصویر ما a3.gif است.

در پر کردن سطرهای جدول مختارید.برای بازیابی یک کنترل SqlDataSource در صفحه قرار دهید و در صفحه ی اول تنظیمات این کنترل نام پایگاه داده و رشته ی اتصال را مشخص کنید و در صفحه ی بعد جدول Shop و آیتم های آن را انتخاب کنید.به این ترتیب ما منبع داده ای داریم.برای نمایش آن از کنترل DataList استفاده می کنیم.و منبع این کنترل را همان کنترل SqlDataSource معرفی کنید.حال کمی با این کنترل DataList کار داریم:

روی تنظیمات این کنترل که به صورت یک پلیکان به سمت راست در گوشه ی بالای سمت راست آن است بروید (Task) و روی EditTemplate کلیک کنید.به این صورت شما دیگر آنچه دوست دارید می توانید درآن قرار دهید و بازیابی کنید. در آن Label های item_no و keyword را حذف کنید و یک کنترل Image روی آن قرار دهید و یک کنترل ImageButton در ته ان که این کنترل ImageButton نقش همان دکمه ی Add To Basket و کنترل Image نقش تصویر جنس مورد نظر می باشد:



لینک آدرس تصویر کنترل ImageButton چون یک تصویر Add To Basket برای تمام آیتم هاست را به طور ثابت بدید.ولی لینک تصویر بالایی چون ممکن است ۱۰۰۰ آیتم هم باشد و پویا است را باید در بخش EditDataBinding کنترل Image تنظیم کنید.کافی است به گوشه ی بالا و سمت راست آن رفته و گزینه یEditDataBinding را انتخاب کرده ودر سمت چپ صفحه در قسمت BindableProperties گزینه ی ImageUrl را انتخاب کرده(سایر صفات این کنترل هم قابل تنظیم هستند) و تنظیمات آن را به صورت زیر در آورید:



که صفت Bound To در آن را با ستون keyword که حاوی نام و پسوند تصاویر است مقدار دهی می کنیم.صفت Format را به آدرس تصویر که برای سایت ما در پوشه ی Shop_img قراردارد به همراه {۰} که نشان دهنده ی همه نوع رشته است مقدار دهی کنید. صفت Sample خودش مقدار دهی می شود.تمام مراحل کار در صفت CustomBinding هم انجام می شود .این صفت مخصوص این است که شاید بخواهید

خودتان مقادیری را به کد تولید شده اضافه کنید. تمام این کار ها را در Source در قسمت مربوط به DataList هم می توانید انجام دهید به صورت زیر:

```
<asp:Image ID="Image1" runat="server" ImageUrl='<%# Eval("keyword", "~\shop_img\{0}") %>' />
```

می بینید همان مقادیری که در CustomBinding بود به صفت ImageUrl کنترل Image هم اضافه شد.پس در اصل وقتی شما ~\shop_img\{0} را می نویسید به منزله ی ~\shop_img\keyword می باشد.حال اگر مثلا نام تصاویر شما در پایگاه داده پسوند نداشت و تنها نام تصویر بود برای دادن پسوند به دنباله ی عبارت {۰} پسوند را اضافه کنید:

~\shop_img\{0}.gif

این اعمال چون سفارشی هستند در صفت CustomBinding انجام می شوند یا در Source صفحه و در داخل کلمه ی کلیدی Eval همانطور که در کد بالا مشخص است.حال می ماند مواردی که در ImageButton قرار دارند.فعلا همین را بدانید که ما می خواهیم کاربر با کلیک کردن روی ImageButton دو مقدار را برای ما بفرستد.یکی نام کالا و دیگری قیمت آن.پس ما باید به دنبال دو خصوصیت Button بگردیم که این دو در آن ذخیره شوند.که به ترتیب CommandArgument و CommandName می باشند.حال شاید از خودتان بپرسید چرا این دو متد؟CommandArgument و CommandName هر دو مخصوص pass کردن یک مقدار اختیاری به رخداد OnCommand در یک button هستند.یعنی به این دو صفت مقداری اختصاص می دهیم سپس آنها را در رویداد OnCommand بازیابی کنیم:

```
<asp:Button id="Button5"
Text="Submit Unknown Command Argument"
CommandName="Submit"
CommandArgument="UnknownArgument"
OnCommand ="CommandBtn_Click"
runat="server"/>
```

در این کد مقدار CommandName , submit است و مقدار UnknownArgument,CommandArgument است و رویداد OnCommand نیز برایش تعریف شده که گفته هر بار که OnCommand شد به رویدادکلیک وصل شو.حال در رویداد کلیک این button, به چه صورت می توان به این دو مقدار دسترسی داشت؟اگر به صورت این رویداد دقت کنید می بینید که یک آرگومان ورودی آن e از نوع

CommandEventArgsاست نه system.EventArgs.این آرگومان خصوصیات مختلف رویداد را در خود حمل می کند حال این رویداد کلیک Button است .پس e حاوی خصوصیات مختلف آن نیز هست.و برای دسترسی به CommandArgument و CommandName کافی است از e استفاده کنیم:

```
  Protected Sub Button2_Click(ByVal sender As Object, ByVal e As
CommandEventArgs) Handles Button2.Command
          Response.Write(e.CommandArgument)
          Response.Write(e.CommandName)
End Sub
```

در این کد دقت کنید در قسمت آرگومان e از نوع CommandEventArgs است نه حالت معمولی آن یعنی system.EventArgs که شما برای بازیابی درست باید یا e را از نوع CommandEventArgs تعریف کنید یا در رویدادی که خود نام آن رویداد حاوی Command باشد به این معنی که ما گفتیم که e خصوصیات مختلف رویداد را در خود حمل می کند که منظور ما این است که این e حتما نباید از نوع CommandEventArgs باشد بلکه کافیست از نوع Command باشد حال از نوع هر کنترل یا رخدادی مثلا از نوع DataListEventArgs باشد.تنها در این صورت ها است که دو پارامتر CommandArgument و CommandName فعال می شوند..در اینجا هم اگر e را در داخل تابع نوشته ونقطه تایپ کنید می بینید که تنها حاوی دو مقدار CommandArgument و CommandName است. خروجی این کد همان مقادیری است که به این دو صفت نسبت دادید.در واقع CommandName دکمه ای را که کاربر کلیک کرده را می شناسد وCommandArgument هم مقدار رشته ای مربوط به آن دکمه را به ما می دهد. در ادامه وقتی به رویداد خاص حاوی Command که رسیدیم دقیقا متوجه شرایط خواهید شد.

کنترل DataList حاوی الگوهای متعددی نیز می باشد که ما از EditItemTemplate آن که اجازه ی ویرایش آیتم ها را می دهد استفاده کردیم.

حال ادامه دهیم.اگر در سمت راست و بالای کنترل ImageButton روی فلش سمت راست کوچک(Task) کلیک کنید یک صفحه شبیه شکل صفحه ی قبل ایجاد می شود(Imagebutton DataBindings).در سمت چپ آن در قسمت BindableProperties ابتدا گزینه ی

CommandArgument را انتخاب کرده و در سمت راست صفحه جلوی صفت Bound To مقدار نام کالا را انتخاب کنید یا در قسمت Custom Binding عبارت زیر را بنویسید:

Eval("item_name")

پس از تنظیم نام کالا به CommandArgument باید قیمت آن را به CommandName نسبت دهیم.برای این کار کافی استCheck Box Show All را Check کنید و در لیست باز شده به دنبال CommandName بگردید و Properties همان مراحلی را که برای CommandArgument طی کردید اینجا هم طی کنید با این تفاوت که به جای نام کالا از پایگاه داده ,قیمت آن را به CommandName نسبت دهید:

Eval("item_price")

خوب حالا کار ما با کنترل DataList تمام شد.می توانید در قسمت Auto format این کنترل را زیبا تر کنید و قالب های آماده ای را برای آن در نظر بگیرید یا به قسمت خواص این کنترل رفته و دستی آن را تنظیم کنید.پس تا اینجا با الگوی مهم کنترل DataList که EditItemTemplate است آشنا شدید و دیدید که می توانید هر گونه که دلتان بخواهد این کنترل را سفارشی کنید و کنترل های مختلفی را به آن اضافه یا کم کنید که ما کنترل ImageButton و Image را به ان اضافه کردیم.حال اگر فرض کنید شما نمی خواستید این همه کار را انجام دهید و به طور ساده تصاویر اجناس و قیمت انها را دستی به سایت اضافه می کردید کارراحت تر بود نه؟بله برای تعدادمحدود کالا!اگر سایت شما تنها ۱۰۰ جنس برای فروش می داشت شما باید ۱۰۰ ImageButton و ۱۰۰ Image و ۱۰۰ بار توضیح ایجاد می کردید و رویداد کلیک هر ImageButton را جداگانه می نوشتید که خود آن علاوه بر کم شدن سرعت و بالا رفتن بی خودی حجم سایت به حدود ۵۰۰ یا ۶۰۰ خط کد نیاز داشت.ولی ما همه ی این کارها را با یک کنترل ImageButton و Image و ... انجام دادیم و وظیفه ی به تعدادکردن آن را به DataList و پایگاه داده ای که به سادگی طراحی کردیم دادیم.کد مربوط به کنترل DataList را در زیر می بینید:

```
  <asp:DataList ID="DataList1" runat="server" DataKeyField="item_no"
DataSourceID="SqlDataSource1" BackColor="#DEBA84" BorderColor="#DEBA84"
BorderStyle="None" BorderWidth="1px" CellPadding="3" CellSpacing="2"
GridLines="Both">
     <ItemTemplate>
         <asp:Image ID="Image1" runat="server" ImageUrl='<%# Eval("keyword",
"~\shop_img\{0}") %>' />
```

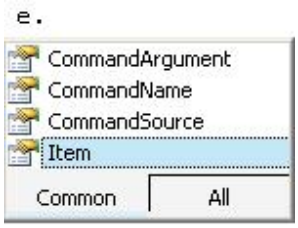
```
        <br />
        <asp:Label    ID="Label1"    runat="server"    Font-Bold="True"    Font-
Names="Verdana" Font-Size="Small" Text="name:"></asp:Label>
        <asp:Label      ID="item_nameLabel"      runat="server"      Text='<%#
Eval("item_name")     %>'     Font-Bold="True"     Font-Names="Verdana"     Font-
Size="Small"></asp:Label>
        <br />
        <asp:Label    ID="Label3"    runat="server"    Font-Bold="True"    Font-
Names="Verdana" Font-Size="Small" Text="Description:"></asp:Label>
        <asp:Label    ID="Label4"    runat="server"    Font-Bold="True"    Font-
Names="Verdana"       Font-Size="Small"       Text='<%#       Eval("item_des")
%>'></asp:Label><br />
        <asp:Label    ID="Label2"    runat="server"    Font-Bold="True"    Font-
Names="Verdana"     Font-Size="Small"     Text="Price:"></asp:Label><asp:Label
ID="item_priceLabel" runat="server" Text='<%# Eval("item_price", "${0}") %>'
Font-Bold="True" Font-Names="Verdana" Font-Size="Small"></asp:Label>
         <br />
         <br />
         <asp:ImageButton           ID="ImageButton1"           runat="server"
CommandArgument='<%# Eval("item_name") %>'
                    CommandName='<%#          Eval("item_price")          %>'
ImageUrl="~/basket.jpg" />
    </ItemTemplate>
        <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
          <SelectedItemStyle      BackColor="#738A9C"      Font-Bold="True"
ForeColor="White" />
            <ItemStyle BackColor="#FFF7E7" ForeColor="#8C4510" />
            <HeaderStyle        BackColor="#A55129"        Font-Bold="True"
ForeColor="White" />
</asp:DataList>
```

می بینید که تمام آیتم های کنترل DataList در تگ <ItemTemplate> قرار دارند.اولی همان کنترل Image است که تصویر کالا را با توجه به لینکی که دارد بر اساس Keyword نمایش می دهد.دومی یک Label ساده با Name Text .سومی یک Label که مقدار نام کالا را از پایگاه داده نمایش می دهد.بعدی هم یک Label ساده با Description Text.بعدی هم توضیحات مربوط به کالا را استخراج کرده ونمایش می دهد. بعدی هم یک Label ساده با Price Text.بدی هم قیمت کالا را به اضافه ی علامت $ نمایش می دهد.و در نهایت یک ImageButton با تنظیمات ذکر شده.در تگ های <FooterStyle><ItemStyle><HeaderStyle> نیز به ترتیب تنظیمات شکل و رنگ عنوان DataList,آیتم های آن و پا ورقی آن مشخص شده است.تنها رویدادی که با کلیک کردن یک کنترل از خانواده ی Button در DataList تحریک می شود

ItemCommand است.یعنی تنها این رویداد است که حس می کند یک دکمه در درون DataList کلیک شده است:

```
  Protected Sub DataList1_ItemCommand(ByVal source As Object, ByVal e As
DataListCommandEventArgs) Handles DataList1.ItemCommand

End Sub
```

از طرفی این رویداد چگونه این تشخیص را می دهد.درست است که ما برای تمامی کالا ها تنها یک دکمه قرار دادیم ولی با تنظیمات مربوط به CommandArgument و DataList, CommandName برای هر کالا دکمه ای جداگانه را حس می کند.و با فرستادن مقدار e از نوع DataList1.ItemCommand می توان تشخیص دادکه کدام ImageButton کلیک شده.فرمان ها ی مربوط به e در شکل زیر مشخص شده:



البته چون e از نوع DataList1.ItemCommand به عنوان آرگومان ورودی تعریف شده شده بود این صفات برایش ظاهر شدند.که اولی و دومی که معرف حضور هستند .پس برای بازیابی کا فی است به صورت زیر عمل کنیم:

```
    itemname = e.CommandArgument
  itemprice = e.CommandName
```

حال به سادگی توانستیم نام و قیمت کالای دکمه ی کلیک شده در DataList را در بین تعداد بسیاری کالا بدست آوریم.البته من خودم از صفت CommandSource استفاده کردم.این صفت همان کنترلی که فرستاده شده با تمام خصوصیاتش را به من می دهد.از آنجایی که کنترل فرستاده شده همان ImageButton است پس کافی است یک متغیر از نوع آن تعریف کنم و مقدارش را با e.CommandSource مقدار دهی کنم:

```
  Dim k As ImageButton
k = e.CommandSource
```

حالا این متغیر K حاوی تمام خصوصیات کنترل ImageButton که کلیک شده از جمله CommandArgument و CommandName می باشد.جلو تر یک تابع به نام AddToBasket تعریف می کنیم که دو آرگومان دارد یکی نام کالا و دیگری قیمت آن (باید به نوع Integer تبدیل شود)که کالای خریداری شده را به سبد خریداضافه می

کند.پس می توان کد کلی رویداد DataList ItemCommand را به صورت زیر تعریف کرد:

```
  Protected Sub DataList1_ItemCommand(ByVal source As Object, ByVal e As
DataListCommandEventArgs) Handles DataList1.ItemCommand
    Dim k As ImageButton
    k = e.CommandSource
    addtobasket(k.CommandArgument, CInt(k.CommandName))
End Sub
```

حال می خواهیم وارد کد نویسی شویم و به Session که اصلا به خاطر آن بود که این مثال را زد یم بپردازیم:

ابتدا تابعی به عنوان initializebasket() تعریف می کنیم.وظیفه ی این تابع ایجاد یک سبد خرید جدید است.ما از کلکسیون HashTable برای هر کالا استفاده می کنیم این تابع را به صورت زیر بنویسید:

```
  Sub initializebasket()
     Dim baskettable As New System.Collections.Hashtable
     Session("basket") = baskettable
     Dim basketprice As New System.Collections.Hashtable
     Session("price") = basketprice
End Sub
```

درخط اول یک HashTable به نام baskettable تعریف کردیم و در خط بعدی مقدارش را (که فغلا خالی است) در یک Session به نام basket قرار دادیم.همین کار را عینا برای قیمت ها هم در دو خط بعدی انجام دادیم.

تا یادمان نرفته یک متغیر عمومی به نام allprice برای اینکه قیمت ایتم های اضافه شده را در آن قرار دهیم تعریف می کنیم و مقدار اولیه ی آن را ۰ قرار می دهیم:

```
Public Shared allprice As Integer = 0
```

رویداد Page_Load هم به صورت زیر بنویسید:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
     If Session("basket") Is Nothing Then
         initializebasket()
     End If
End Sub
```

این رویداد می گوید اگر شی Session به نام basket وجود نداشت(اگر خریدی انجام نگرفته بود) آنگاه یک سبد خرید جدید ایجاد بکن .چون بالاخره این صفحه مربوط به خرید است و باید با بار گذاری صفحه , سبد خریدی برای مشتری ایجاد شود(بدون اینکه متوجه شود) تا وی سر در گم نشود.البته می توان به جای این کار در رویداد

session_start در فایل global.asax عبارات داخای تابع initializebasket را اضافه کنید:

```
  Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
    Dim baskettable As New System.Collections.Hashtable
    Session("basket") = baskettable
    Dim basketprice As New System.Collections.Hashtable
    Session("price") = basketprice
End Sub
```

متد AddToBasket مهم ترین متد ماست که دوآرگومان ورودی دارد یکی نام کالا و دیگری قیمت آن:

```
  Public Sub addtobasket(ByVal itemname As String, ByVal itemprice As
Integer)

End Sub
```

چون نیازی به مقدار برگشتی در این متد نداریم آن را Sub در نظر می گیریم.

سپس هر بار که آیتمی به سبد خرید اضافه شود قیمت آن را به allprice اضافه می کنیم:

```
allprice += itemprice
```

سپس سبد خرید و سبد قیمت که در Session بودند(در تابع مهم initializebasket ایجاد شدند) را در دو HashTable قرار می دهیم چون خودشان هم HashTable بودند(چون ما آنها را در رویداد Session_Start در داخل یک HashTable قرار دادیم):

```
  Dim baskettable As System.Collections.Hashtable = Session("basket")
Dim basketprice As System.Collections.Hashtable = Session("price")
```

حال باید ببینیم که آیا کاربر آیتمی را خریده یا نه؟اگر خریده که هیچ و گرنه باید تعداد آیتم ها را ۰ کنیم تا از این به بعد به آن اضافه شود:

```
  If baskettable(itemname) Is Nothing Then
     baskettable(itemname) = 0
End If
```

یادم رفت بگویم که baskettable HashTable به صورت زیر است:

Baskettable(“itemname”)=number of items

و basketprice HashTable به صورت زیر است:

Basketprice(“itemname”)=items price

پس با این کار اگر کاربر اولین خرید را می خواست انجام دهد تعداد کالاهای وی را ۰ قرار می دهیم تا دوباره یکی یکی به تعداد آن اضافه شود.سپس تعداد آیتم های خریداری شده از هر کالا را در متغیر itemcount ذخیره می کنیم:

```
Dim itemcount As Integer = baskettable(itemname)
```

مثلا اگر کاربر می خواهد کلاه بخرد ابتدا قیمت کلاه به allprice اضافه می شود.سپس سبد های خرید و قیمت اماده می شوند.پس از آن چک می شود که این کلاه اولین کلاه است یا نه اگر اولی بود (که مثلا هست) baskettable(hat) = 0 قرار داده می شود.سپس متغیر itemcount حاوی مقدار ۰ می شود درادامه چون این شخص یک کلاه خریده باید به تعداد کلاه ها یکی اضافه شود:

```
baskettable(itemname) = itemcount + 1
```

پس اگر قرار باشد این شخص یک کلاه دیگر بخرد ابتدا قیمتش به allprice اضافه می شود سپس مجددا سبد ها آماده شده و چون شرط خالی بودن HashTable کلاه ها درست نمی باشد baskettable(hat) به itemcount ریخته می شود و itemcount حالا ۲ می شود.

برای محاسبه ی قیمت هر کالا(به هر تعداد) هم به صورت زیر عمل می کنیم:

```
basketprice(itemname) = itemprice * baskettable(itemname)
```

یعنی قیمت هر کالا که به عنوان آرگومان ورودی تابع در یافت شد ضرب در تعداد هر کالا(مثلا تعداد کلاه) است که اخیرا هم به وسیله ی کد بالا تری(بالا ی بالایی) به روز شد.

حال می ماند نمایش این مواردی که مشخص کردیم.برای نمایش هم دو کنترل DataList به نام های basketlist و gheymat روی صفحه قرار می دهیم و source آنها را به ترتیب دو Session که یکی حاوی HashTable نام و تعداد کالا و دیگری HashTable نام و قیمت کالا به تعداد خریداری شده است قرار می دهیم.البته یادمان باشد نمیتوان Session را به عنوان source کنترل DataList قرار داد ولی HashTable را چرا.این اعمال را در رویداد page_prerender انجام می دهیم چون رویداد مناسبی است و قبل از بارگذاری صفحه انجام می گیرد و با شرطی که در رویداد Page_Load نوشتیم قاطی نمی شود:

```
  Protected Sub Page_PreRender(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.PreRender
    Dim basketTable As System.Collections.Hashtable = Session("Basket")
    basketlist.DataSource = basketTable
    basketlist.DataBind()
    Dim basketprice As System.Collections.Hashtable = Session("price")
```

```
    gheymat.DataSource = basketprice
    gheymat.DataBind()
    If basketTable.Count = 0 Then
    Label5.Text = "No Item(s) Selected!!"
    gheymat.Visible = False
    basketlist.Visible = False
    Else
    gheymat.Visible = True
    basketlist.Visible = True
    Label5.Text = ""
    End If
End Sub
```

بقیه ی موارد بستگی به خودتان دارد مثلا من گفتم اگر basketTable.Count = 0 بود بنویسد آیتمی انتخاب نشده است و در غیر این صورت چیزی ننویسد.حتما با خود می گویید کنترل DataList چگونه موارد مشخص شده را نمایش دهد آخه ما تنها source آن را مشخص کردیم.این کار در source صفحه در قسمت مربوط به کنترل DataList صورت می گیرد:

```
  <asp:DataList ID="basketlist" runat="server">
     <ItemTemplate>
       <asp:Label ID="L10" runat="server" Text='<%# Container.DataItem.Key &
"s: " %>'></asp:Label>
       <asp:Label ID="L20" runat="server" Text='<%# container.Dataitem.Value
%>'></asp:Label>
     </ItemTemplate>
     <HeaderTemplate>
        Item Name/Number
     </HeaderTemplate>
     <HeaderStyle Font-Bold="True" Font-Italic="True" Font-Size="Medium"
ForeColor="Blue" />
</asp:DataList>
```

باز هم در تگ <ItemTemplate> آیتمهای موجود در DataList به نمایش در می آیند.که اولی یک label با مقدار Container.DataItem.Key است که همان کلید HashTable است و بعدی Container.DataItem.value است که همان مقدار HashTable است .به اینجور کد نویسی (درون <%# %>) استفاده از asp.net در html می گویند.در بخش RICH DATA CONTROLS مفصل در مورد طرز استفاده ی درست این گونه دستورا ت در کنترل های مختلف مثل DataList,GridView,DetailsView و ... صحبت خواهیم کرد.

DataList مربوط به قیمتها را نیز به همین صورت تعریف کنید.

حال می ماند دکمه Empty Basket و CheckOut:

CheckOut:

کافی است کد زیر را در رویداد کلیکش بنویسید:

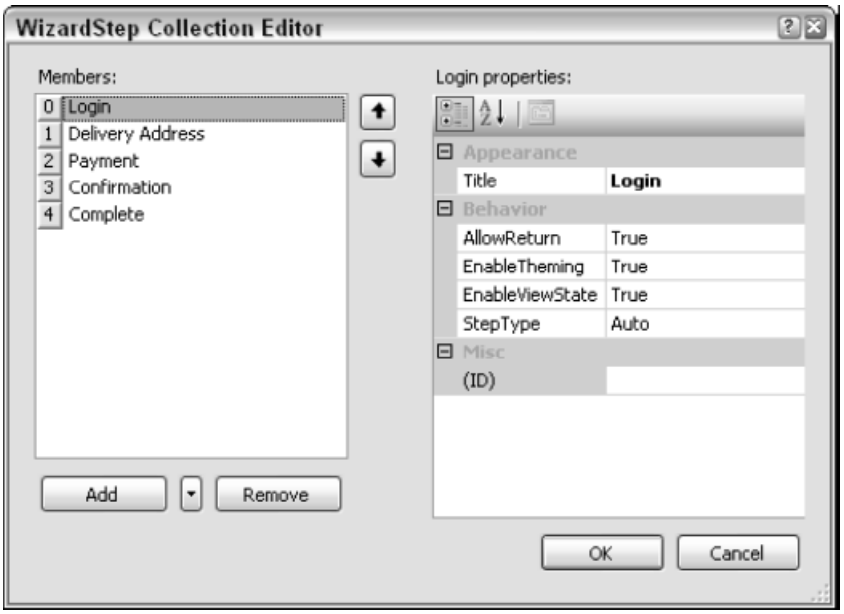
```
  Protected Sub Button2_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button2.Click
    If gheymat.Items.Count = 0 Then
       Label6.Text = "You Not Select Item(s) Yet.Please Select This"
    Else
       Label6.Text = "Thank You For Your Choosing,You Have Selected " &
gheymat.Items.Count & " items From My Online Shop.Your Payable Price is $" &
allprice
    End If
End Sub
```

Empty Basket:

هر بار که تابع initializebasket را صدا می زنیم یک سبد خرید جدید ایجاد می شود پس اگر سبدی هم از قبل وجود داشته باشد با تعریف جدید Session از بین رفت و یک سبد خالی جایش ایجاد میشود.کافی است مقدار allprice را نیز صفر کنیم:

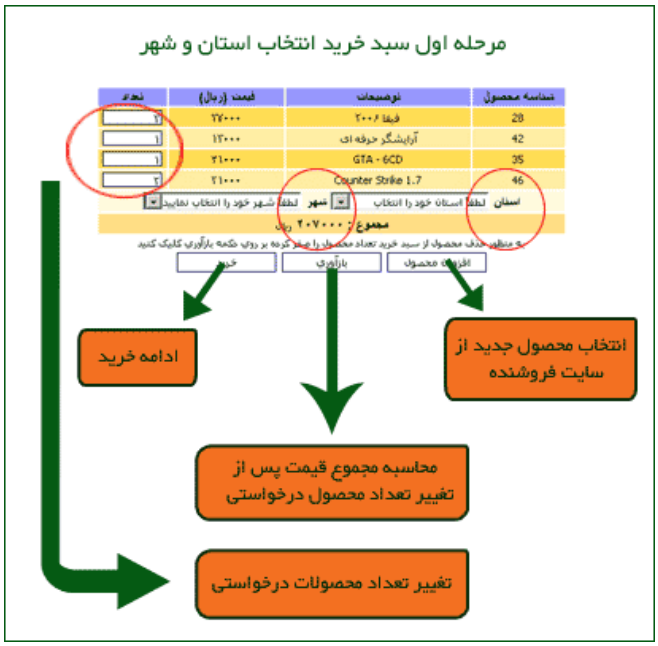
```
  Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
    initializebasket()
    allprice = 0
End Sub
```

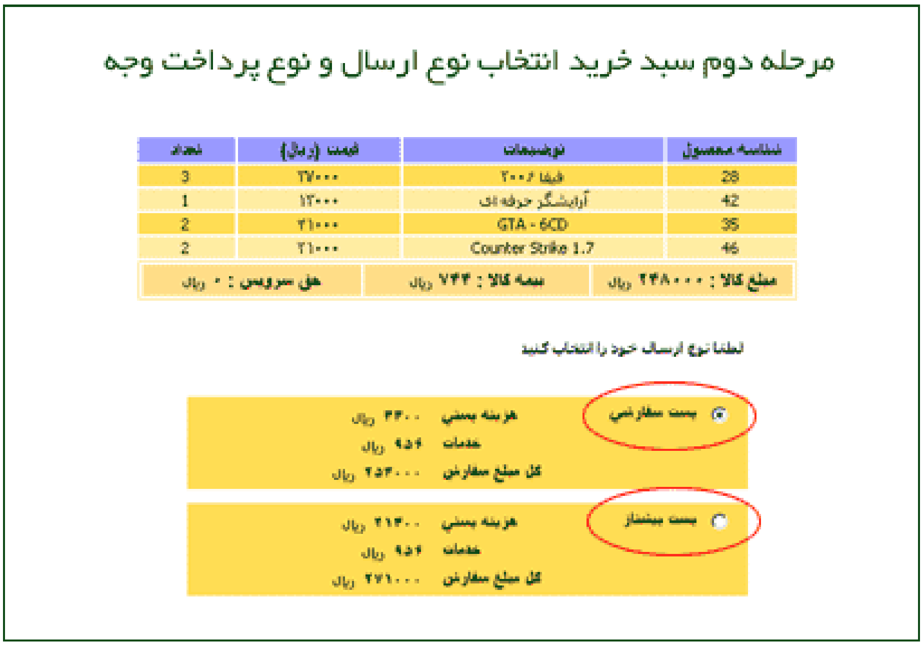
خوب کار ما تمام شد و حالا ما یک فروشگاه الکترونیکی داریم.البته لازم به ذکر است که کلاس معروفی به نام shopping cart وجود دارد که این کار ها را به طریقه ی دیگری با profile انجام می دهد.هر وقت به بحث profile رسیدیم که بحث مهمی است دیگر نیازی به Session نداریم و یک فروشگاه الکترونیکی با profile ایجاد می کنیم.ولی تا همین جا هم غنیمت است.از این به بعد مراحل پرداخت پول الکترونیکی است که در محدوده ی بحث ما نیست. با یک کنترل wizard و قراردادن فرم login در مرحله ی اول ابتدا کاربر وارد سایت می شودو اطلاعات وی هم همراه با وی ارسال می شوند سپس آدرس وی پرسیده می شود پس از پرداخت پول و بحث Cradit Cart آغاز می شود سپس سفارش وی محکم کاری شده و در نهایت پایان کار.پس کنترل wizard شما باید شامل ۵ مرحله باشد:



نحوه ی خرید و پرداخت الکترونیک در ایران در اشکال زیر توضیح داده شده:

مراجعه به یکی از صدها فروشگاه اینترنتی عضو و انتخاب محصول





مرحله دوم سبد خرید انتخاب نوع ارسال و نوع پرداخت وجه

شناسه محصول	توضیحات	قیمت (ریال)	تعداد
28	فیفا ۲۰۰۶	۲۷۰۰۰	3
42	آرایشگر حرفه ای	۱۲۰۰۰	1
35	GTA - 6CD	۲۱۰۰۰	2
46	Counter Strike 1.7	۲۱۰۰۰	2
مبلغ کالا : ۲۴۸۰۰۰ ریال	بیمه کالا : ۷۴۴ ریال	حق سرویس : ۰ ریال	

لطفا نوع ارسال خود را انتخاب کنید

پست سفارشی
هزینه پستی ۴۴۰۰ ریال
خدمات ۹۵۶ ریال
کل مبلغ سفارش ۲۵۴۰۰۰ ریال

پست پیشتاز
هزینه پستی ۲۱۴۰۰ ریال
خدمات ۹۵۶ ریال
کل مبلغ سفارش ۲۷۱۰۰۰ ریال


مرحله سوم سبد خرید ورود اطلاعات ارسال
اطلاعات ارسال
نام
نام خانوادگي
گد ملي
جنسيت
شغل
نام شرکت
پست الکترونيکي
آدرس
کدپستي
تلفن
پيغام شما
ساعت تحويل
اهميتي ندارد

برای آشنایی کامل با پرداخت الکترونیک در ایران و قوانین تجارت الکترونیک در ایران به سایت http://www.pardakht.com رجوع کنید.

Application:

شی که در بحث global.asax به ان اشاره شد.فرق Application با Session این است که Application در سرور به اشتراک گذاشته می شود و اطلاعاتی که ما به آن می دهیم در سرور به صورت اشتراکی ذخیره می شود و مبتنی بر فرد نیست.ما وقتی این سایت تجارت الکترونیک را طراحی کردیم ,هر کاربری در هر جایی از دنیا می توانست خرید کند و برای خود یک Session جدا داشته باشد یعنی هر کاربر یک سبد خرید جدا در شی Session داشت که به بقیه ربطی نداشت.حال Application اصلا اینگونه نیست و یک شی Global است که در تمامی برنامه یکی است در حقیقت یک شی است که مقداری را در آن ذخیره می کنید و می توانید آنرا برای همه ی کاربران بازیابی کنید.و دیگر اینگونه نیست که هر کاربر یک Application جدا داشته باشد که به بقیه ربطی نداشته باشد.ولی از جهت پشتیبانی انواع داده ها و اشیا , مثل Session است.آیتم های موجود

در Application هرگز TimeOut ندارند .اصطلاحا می گویند عمر Application به LifeTime برنامه بستگی دارد.هر گاه برنامه به پایان برسد Application نیز از بین می رود که این امر می تواند به صورت به روز شدن یکی از صفحات یا قسمتی از برنامه ی ما باشد(حتی تغییر یک خط از هر جای کد برنامه که باعث Refresh شدن برنامه می شود) یا Restart شدن سرور.پس تنها در این صورت های ذکر شده است که Application از بین می رود.این به اتمام رسیدن برنامه را اصطلاحا Application Shut Down می گویند.همانطور که دیدید ما یک شمارنده در سایتمان در بحث global.asax داشتیم .حال می خواهیم همان شمارنده را در صفحه ی اصلی خود داشته باشیم:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
      Dim count As Integer = CInt(Application("HitCounterForOrderPage"))
      count += 1
      Application("HitCounterForOrderPage") = count
      lblCounter.Text = count.ToString()
End Sub
```

این کد به درستی کار می کند البته تا وقتی که شی Application از بین نرود(به دلایلی همچون بهروز رسانی صفحه و یا Refresh شدن سرور).

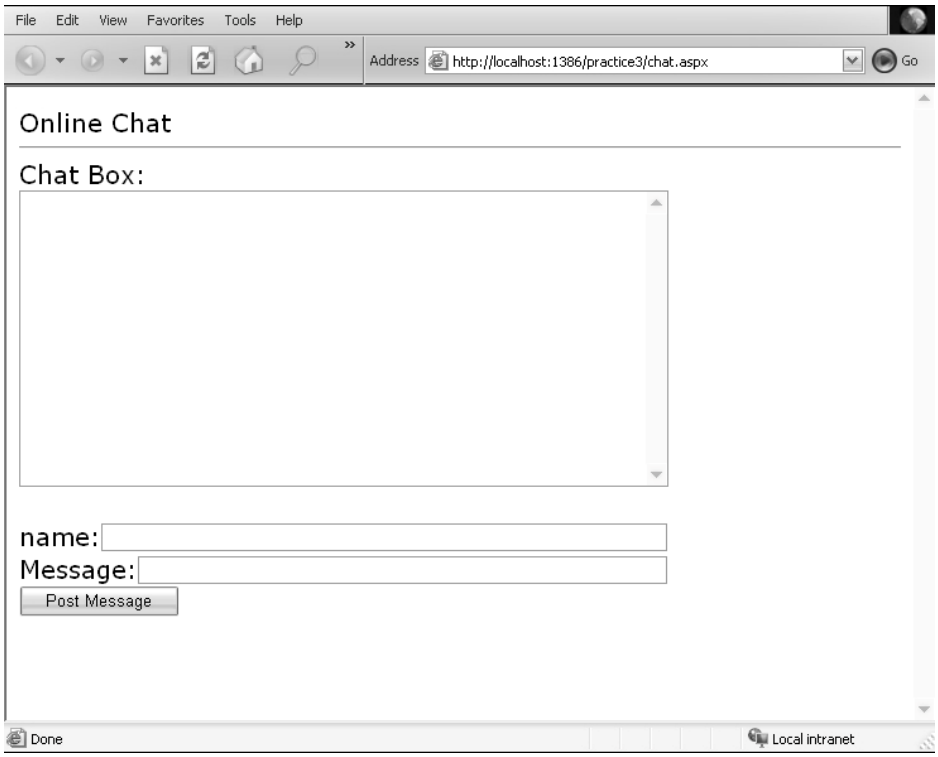
ولی ایراد این کد این است که اگر دو کاربر همزمان وارد سایت ما شوند شمارنده یکی می شمارد.چه بسا به جای دو تا بیش از دو کاربر همزمان وارد سایت شوند و این امر به کلی حساب شمارش را از دست ما خارج می سازد(وقتی که در سایت heavy traffic رخ دهد).زیرا دو کاربر در یک لحظه به یک شی Application واحد رجوع می کنند (بیچاره Application).برای حل این مشکل کافی است در رویداد Page_Load از دو متد lock,unlock استفاده کنیم:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
     Application.Lock()
     Dim count As Integer = CInt(Application("HitCounterForOrderPage"))
     count += 1
     Application("HitCounterForOrderPage") = count
     Application.UnLock()
     lblcounter.Text = count.ToString()
End Sub
```

به این صورت هر کاربری که وارد برنامه شد با وقوع رویداد Page_Load , Application برای سایر کاربران قفل می شود یعنی اجازه ی تغییر ندارد و تنها برای آن کاربر خاص اجازه ی تغییر دارد که به ان یک واحد اضافه شده سپس قفلش باز می شود.حال اگر کاربر جدیدی همزمان با ما وارد شود یکی از ما Application را قفل می کنیم و پس از اضافه کردن یک واحد به آن آن را برای دیگری(دیگران) باز می کنیم.سعی کنید همیشه این کار را انجام دهید.با اینکه متاسفانه باز هم مشکلی پیش می آید و آن این است که کاربران بعدی باید در صف باز شدن قفل Application بمانند.که کارایی برنامه را پایین می آورد ولی خوب بهتر از خروجی اشتباه است.

همانطور که گفتیم Application یک شی است که در سرور ذخیره می شود.و تمام کاربران می توانند آن را بازیابی کنند و یک شی مشترک بین آنان است.پس برای مثالی جامع از طرز کار Application بیایید یک Chat room ساده ایجاد کنیم:

ابتدا یک صفحه مثل صفحه ی زیر ایجاد کنید:



سپس با علم به اینکه نام شما در فیلد name و پیغام شما در فیلد message باید قرار گیرد.رویداد کلیک دکمه را به صورت زیر بنویسید:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
```

```
    Dim newmessage As String = TextBox2.Text + ":" +
Microsoft.VisualBasic.vbTab + TextBox3.Text + Microsoft.VisualBasic.vbCrLf
+ Application("chat")
    If newmessage.Length > 500 Then
      newmessage = newmessage.Substring(0, 499)
    End If
    Application("chat") = newmessage
    TextBox1.Text = Application("chat")
    TextBox3.Text = ""
End Sub
```

ابتدا متغیری به نام newmessage از نوع رشته ای تعریف کردیم و مقدار آن را با نام شخص دو نقطه و فاصله(Microsoft.VisualBasic.vbTab) و سپس پیغام شخص و رفتن به سطر بعد(Microsoft.VisualBasic.vbCrLf) و در نهایت مقادیر حاوی شی Application مشخص کردیم.سپس شرط کردیم اگر پیغام از ۵۰۰ حرف بیشتر شد ۵۰۰ تای اولش را به عنوان پیغام در نظر بگیر.سپس این پیغام را در Application ذخیره کردیم و سپس آن را به صفحه ی مربوط به پیام ها اضافه کردیم تا پیغام ما هم در این صفحه با شد و هم در Application ذخیره گردد.

رویداد Page_Load هم باید با نمایش مقادیر شی Application انجام گیرد:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    TextBox1.Text = Application("chat")
End Sub
```

برای زیبایی کار نیز می توانید پیغام زیر را به فایل global.asax اضافه کنید:

```
  Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
    Application("chat") = "Hello To My Chat Page!"
End Sub
```

برای دیدن پیام های سایر کاربرا ن باید یا صفحه Refresh شود یا شما پیغامی بفرستید.می توانید برای جلوگیری از خطا در همزمان سازی صفحه در اینجا هم Application را Lock,Unlock کنید.در کل در .NET از شی Application به ندرت استفاده می شود چون جایگزین های مناسب تر و بهینه تری برایش وجود دارد.مثلا در برنامه ی چت بالا همین طورحجم Application بالا و بالاتر می رود تا آنجا که واقعا برنامه ی ما را کند می کند.البته می توان امکانی برای مدیریت چت گذاشت که پس از هر روز یک بار متد clear شی Application را صدا بزند تا عملیات چت از نو آغاز شود و شی application نفس راحتی بکشد.

بحث StateManagement شامل موارد مهم دیگری همچون Caching و Profileهم هست که به دلیل وسعت مطلب و اهمیت بسیار بالا و اینکه باید پس از پایگاه داده مطرح شود.در جای خود توضیح داده می شود.

در زیر یک مقایسه ی جامع در مورد انواع روشهای مدیریت حالت را می بینید:

ویژگی	View State	Query String	Custom Cookies	Session State	Application State
نوع های داده قابل استفاده	تمامی نوع های داده دات نت با قابلیت سریال شدن	حجم محدودی داده از نوع رشته	داده از نوع رشته	تمامی نوع های داده دات نت	تمامی نوع های داده دات نت
مکان ذخیره سازی	یك فیلد مخفی در صفحه وب جاری	در رشته URL مرورگر	کامپیوتر سرویس گیرنده (در حافظه و یا یك فایل متن کوچك با توجه به تنظیمات انجام شده)	حافظه سرویس دهنده	حافظه سرویس دهنده
طول عمر	نگهداری دائم برای post back به یك صفحه	حذف پس از درج یك URL جدید و یا بستن مرورگر توسط کاربر	وابسته به تنظیمات برنامه نویس (امکان استفاده در چندین صفحه و نگهداری بین چندین ملاقات وجود دارد)	پس از گذشت یك زمان مشخص از بین می روند . (معمولا" ۲۰ دقیقه ولی می توان آن را بطور دستی و یا از طریق کد تغییر داد)	قابل استفاده در مدت زمان حیات برنامه (معمولا" تا زمانی که سرویس دهنده راه اندازی مجدد نگردد)
حوزه دستیابی	محدود به صفحه جاری	محدود به صفحه مقصد	تمامی برنامه ASP.NET	تمامی برنامه ASP.NET	تمامی برنامه ASP.NET (برخلاف سایر روش ها ، داده برنامه برای تمامی کاربران سراسری است)
امنیت	به صورت پیش فرض مقاوم در مقابل تغییرات می باشند ولی امکان خواندن آنها وجود دارد . با استفاده از دایرکتیو صفحه می توان بر رمزنگاری آنها تاکید کرد .	قابل مشاهده بوده و کاربران می توانند به سادگی آنها را تغییر دهند .	غیرایمن بوده و امکان تغییر آنها توسط کاربران وجود دارد .	ایمنی بالائی دارند چون هرگز داده برای سرویس گیرنده ارسال نمی گردد	ایمنی بالائی دارند چون هرگز داده برای سرویس گیرنده ارسال نمی گردد
کارآئی	پائین ، در صورت ذخیره حجم بالائی از اطلاعات ولی بر روی کارآئی سرویس دهنده تاثیر نمی گذارد	تاثیر ندارد ، چراکه حجم داده ناچیز است	تاثیر ندارد ، چراکه حجم داده ناچیز است	پائین ، در صورت ذخیره حجم بالائی از اطلاعات خصوصا" اگر در هر لحظه تعداد زیادی کاربر از برنامه استفاده نمایند. چراکه هر کاربر دارای یك نسخه جداگانه از داده session خواهد بود	پائین ، زمانی که حجم بالائی از اطلاعات ذخیره شده باشد چراکه داده هرگز حذف و یا عمرمفید آن به اتمام نخواهد رسید
متداولترین موارد استفاده	تنظیمات مرتبط با صفحه	ارسال شناسه یك محصول از صفحه نمایش دهنده کلیات به صفحه نمایش دهنده جزئیات	اطلاعات شخصی برای یك وب سایت	ذخیره آیتم هائی در یك سبد خرید	ذخیره هر نوع داده سراسری

با توجه به جدول واضح فوق و اینکه جلوتر با Cache هم آشنا می شویم دیگر باید دانسته باشید در بحث مدیریت حالت از چه چیزی و در کجا استفاده کنیم.

ADO.NET

قبل از آغاز ADO.NET بهتر است یک تعریف تخصصی و کامل را در مورد پایگاه داده ببینیم:

پایگاه داده مجموعه ی داده های ذخیره شده ی پایا(ماندنی-یعنی موقت نیستند) به صورت مجتمع(یعنی کل اطلاعات پایگاه داده مورد استفاده ی یک یا چند کاربر در قالبی مشخص به صورت یکجا و به دور از هرگونه پراکندگی باشد) و مبتنی بر یک ساختار(قانونمند و قاعده مند) و تعریف شده به صورت صوری (یعنی داده ها با یک سری احکام خاص درون پایگاه داده قرار گرفته اند) حداقل افزونگی (تکرار بی رویه ی رکورد ها و داده های جداول آن) مورد استفاده ی یک یا چند کاربر به فرمی همزمان(یعنی همزمان چندین کاربر به داده های درون پایگاه داده دسترسی و حتی آن را کنترل کنند) و غیر همزمان و دارای یک سیستم کنترل متمرکز (سیستمی که بتواند پایگاه داده را مدیریت و کنترل کند) روی آن است.در این تعریف موارد داخل پرانتز توضیحی بر اصطلاحات مهم قرمز رنگ آن می باشد.این تعریف کامل ترین تعریفی است که من به شخصه از پایگاه داده دیده ام.برای آشنایی هر چه بیشتر با اصطلاحات قرمز رنگ بالا و اصلا پایگاه داده باید به مراجع اصول طراحی پایگاه داده مراجعه کنید.در ادامه ما فرض می کنیم شما با پایگاه داده و زبان SQL آشنا هستید.

حال برویم سراغ ADO.NET:

مجموعه اشیا و متد هایی که به وسیله ی آن ها می توان داده های درون DataBase را کنترل کرد را ADO.NET (ActiveX Data Object) می نامند.

یعنی به وسیله ی آن اصلا می توان به پایگاه داده(از قبیل ...,SQL,Oracle,Access) اتصال بر قرار کرد.

ADO.NET Data Providers:

تهیه کننده ی داده کلاسی ویژه برای اتصال به یک DataBase است که به شما اجازه ی بر قراری ارتباط با DataBase خاصی را می دهد تا شما بتوانید دستوراتی را روی DataBase اجرا کرده و کلا آن را کنترل کنید.در حقیقت Data Providers پلی بین برنامه ی شما و محل داده های شما هستند.

انواع Data Providers مهم:

SQL Server provider:شرایط اتصال به پایگاه داده ی SQL Server نسخه ی ۷ به بعد را فراهم می کند.این Provider از پروتکل TDS (Tabular Data Stream) یا داده های جدولی برای اتصال به پایگاه داده استفاده می کند.فضای نام این تهیه کننده System.data.SqlClient است.

OLE DB provider:شرایط اتصال به پایگاه داده ی OLE DB مثل Access و همچنین نسخه های قبل از ۷ SQL Server را فراهم می کند.

Oracle provider: شرایط اتصال به پایگاه داده ی Oracle نسخه ی ۸ به بعد را فراهم می کند.برای دسترسی به آن باید از قسمت Add Reference بین اسملی های .NET. گزینه ی system.Data.OracleClient را به برنامه ی خود Add کنید.

اشیا مهم به کار رفته در ADO.NET در زیرآمده است:

Connection:برای برقراری اتصال به پایگاه داده استفاده می شود.

Command:شی که برای اجرای دستورات SQL به کار می رود.

DataReader:شی که به وسیله ی آن می توان داده های اجرا شده در Command را فقط خواند.

DataSet:مکانی برای ذخیره کردن موقتی اطلاعات استخراج شده از DataBase می باشد.

DataAdapter:شی که می تواند دو عمل مهم انجام دهد:۱-پر کردن DataSet از اطلاعاتی که ازDataBase استخراج شده-۲-ایجاد تغییرات در DataSet و اعمال آنها به DataBase. در حقیقت DataAdapter اتصال به DataBase و استخراج اطلاعات را مدیریت می کند.

زبان مورد استفاده ی ما SQLاست که یک زبان پرس و جوی ساخت یافته است.

ADO.NET کلا دو دسته شی دارد:

۱-Connection Based Object :اشیایی که بیشتر برای اتصال به DataBase و استخراج و مدیریت آن استفاده می شوند نظیر Command-Connection - DataReaderو ...

۲-Content Based Object :اشیایی که برای بسته بندی کردن داده هایی که اشیا Connection Based Object از DataBase استخراج کردند استفاده می شوند مثل DataRow-Data Table-DataSet و ...

مهمترین فضا های نامی که در استفاده از ADO.NET مورد نیاز هستند در زیر آمده اند:

System.Data و System.Data.SqlClient هستند که برای ما جهت اتصال به پایگاه داده ی SQL Serve و مدیریت داده های آن مورد نیاز هستند.البته برای پایگاه داده ی Oracle,Access به ترتیب علاوه بر System.Data, به System.Data.OleDb و System.Data.OracleClient نیاز می باشند که تمام اشیا ذکر شده مربوط به ADO.NET در این فضای نام ها قرار دارند.

The Connection Class:

کلاس مهمی که همانطور که گفته شد جهت اتصال به DataBase مورد استفاده قرار می گیرد و شامل اجزای زیر می شود که به ترتیب تک تک آنها را مورد بحث قرار می دهیم:

Connection String
Testing a Connection
Connection Pooling
Connection Statistics

Connection String:

رشته ی اتصال به DataBase است که اهمیت فوق العاده ای دارد و شامل ۳ بخش عمده است.۱-مکان سروری که DataBase در آن قرار دارد.۲-نام DataBase مورد نظر ۳-نحوه ی مجوز دادن آن DataBase به ورود شما به آن.این سه آیتم در رشه ی اتصال قرار داشته و با ; از هم جدا می شوند.در زیر نمونه ای از این رشته ی اتصال را می بینید:

```
Dim connectionString As String =" Data Source = localhost;Initial
Catalog=Northwind;Integrated Security=SSPI"
```

Data Source همان مکان سرور است که در برنامه ی ما localhost است یعنی ما از سرور محلی و مربوط به IIS یا Visual Studio استفاده می کنیم.Initial Catalog هم نام پایگاه داده را مشخص می کند.Integrated Security هم نحوه ی مجوز دادن را مشخص می کند.

قبلا دو عبارت id وPassword نیز در این رشته به جای Integrated Security به کار برده میشد که به ترتیب یک نام و یک رمز برای ورود مشخص می کرد:

```
Dim connectionString As String ="Data Source=localhost;Initial
Catalog=Northwind; user id=sa;password=opensesame"
```

که در حال حاضر کاربرد خیلی کمی دارد.

اگر از پایگاه داده ی OLE DB استفاده می کنید باید خصوصیت provider را نیز به ان اضافه کنید:

```
Dim connectionString As String ="Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=C:\DataSources\Northwind.mdb"
```

که در این کد ما به پایگاه داده ی از نوع Access متصل شدیم.

کد متداولی که برای اتصال به پایگاه داده ی SQL Server 2005 استفاده می کند به صورت زیر است:

```
Data Source=localhost\SQLEXPRESS; Initial Catalog=Pubs;Integrated Security=SSP
```

البته بهتر است به جای localhost در رشته ی بالا از یک نقطه استفاده کنید:

.\SQLEXPRESS

حال اگر بخواهید به پایگاه داده ی Oracle متصل شوید هم می توانید به صورت زیر عمل کنید:

```
<connectionStrings>
add name="ConnectionString" connectionString="Data Source=mohammad;User >
"ID=mohammad;Password=password;Unicode=True
</ "providerName="System.Data.OracleClient
<connectionStrings/>
```

پایگاه داده ی من در بالا Oracle8i محلی بوده و هم نام کاربری و هم رمز عبور داشته است.

زیاد در بند جزییات Connection String نباشید چون یک راه ذکر شده در بحث web.config برای دسترسی به Connection String موجود در web.config به صورت زیر است:

```
Dim cs As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
```

که این کد را می توان به شرط import کردن Imports System.Web.Configuration به صورت زیر نوشت:

```
Dim cs As String =
WebConfigurationManager.ConnectionStrings("aaa").ConnectionString()
```

یعنی از متد WebConfigurationManager که پیشتر هم توضیح داده شد استفاده کنیم.

که نام این Connection String در web.config , aaa است و ما حالا این رشته را در متغیر cs داریم.بازیابی رشته ی اتصال به روش بالا باعث تسهیل در تغییرا آینده ی صفحات وب ما میشود زیرا به این ترتیب دیگر همه ی صفحات برای اتصال به پایگاه داده از یک منبع رشته را بازیابی می کنند و اگر قرار شد در آینده تغییری در رشته ی اتصال داده شود اگر رشته ی اتصال را در منبع تغییر دهیم همه ی صفحات سایتمان خود را با آن تطبیق می دهند.

موارد بالا را جهت آشنایی با این رشته ی مهم گفتیم.

Testing a Connection:

برای تعریف عنصر Connection به صورت زیر عمل کنید:

```
Dim con As New SqlConnection(con_str)
```

که متغیر con نام Connection ما البته از نوع SQL است که مقدار ورودی این تابع سازنده , Connection String است.

دو متد مهمی که در بحث Connection وجود دارند به ترتیب ()Openو()Close هستند که باز شدن و بسته شدن Connection را بر عهده دارند.عملیات کنترل داده ها از پایگاه داده تنها در صورت باز بودن Connection می تواند صورت گیرد و بسته شدن ان هم به معنای اتمام عملیات است و انجام ان باعث کاهش مشغله ی سرور می شود وحتما یادتان باشد که هر باز شدنی باید بستنی را به همراه داشته باشد در غیر این صورت کارتان اشتباه است.البته در عملکرد سایتمان بستن اتصال خیلی تاثیر ندارد و این به ضرر سرور است زیرا با برقراری اتصال به سرور, سرور یک سری منابع را در

اختیار شما قرار داده حال چه شما از این منابع استفاده کنید و چه استفاده نکنید.حال اگر اتصال را ببندید دیگر آن قسمت از منابع که در اختیار شما بود حالا دیگر در اختیار شما نبوده و این باعث می شود سرور راحت تر به سایر سرویس ها پاسخ دهد و به اصطلاح با فراق بال به سراغ سایر درخواست ها برود.

حال برای تست باز بودن یا بسته بودن Connection مثال زیر را انجام دهید:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim con_str As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(con_str)
        con.Open()
        Response.Write("server version is : " & con.ServerVersion & "<br>")
        Response.Write("connection is : " & con.State.ToString & "<br>")
        Response.Write(con.PacketSize & "<br>")
        Response.Write(con.ConnectionTimeout & "<br>")
        con.Close()
    End Sub
```

خط اول رشته ی اتصال را از web.config استخراج می کند وآن را در متغیری به نام con_str می ریزد.سپس یک sqlConnection به نام con با رشته ی اتصال ورودی تابع سازنده ی con_str تعریف کردیم.سپس جهت استفاده از Connection, آن را باز کرده و سپس از ویژگی های ان استفاده کردیم.مثلا con.ServerVersion نسخه ی سرور را به ما می دهد.con.State.ToString وضعیت Connection را به ما می دهد که باز است یا بسته است.con.packetsize سایز شبکه ی ارتباطی SqlServer را به واحد بایت به ما می دهد و در نهایت con.ConnectionTimeout مدت زمان به ثانیه که Connection باز باشد و پس از آن بسته می شود را می دهد که پیش فرض آن ۱۵ ثانیه است.دقت کنید تمام عملیات ما شاید ۱ ثانیه هم طول نکشد.این کار برای امنیت بیشتر است.

مثال زیر را اگر اجرا کنید به گونه ای دیگر Connection را تست می کنید:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim connectionString As String =
WebConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As SqlConnection = New SqlConnection(connectionString)
        Using con
            con.Open()
            lblInfo.Text = "<b>Server Version:</b> " & con.ServerVersion
```

```
            lblInfo.Text    &=    "<br    /><b>Connection    Is:</b>    "    &
con.State.ToString()
        End Using
        lblInfo.Text &= "<br /><b>Now Connection Is:</b> "
        lblInfo.Text &= con.State.ToString()
End Sub
```

در این مثال دو خط اول که تکراری است ولی ما از بلوک using...end using استفاده کردیم که با end usin شدن شی Con انگار این شی بسته می شود و اگر باز هم con.State را تست کنید می بینید که این با ر close است.خروجی این کد به صورت زیر است:

Server Version: 09.00.1399
Connection Is: Open
Now Connection Is: Closed

Connection Pooling:

می دانیم هر بار Connection به پایگاه داده هزینه هایی از جمله هزینه های زمانی و صرف منابع را در بر دارد زیرا جدا از باز شدن هر Connection باید بسته شود و برای در خواست بعدی باز شود .فرض کنید شما ۱۰۰۰ کاربر دارید که این ها به سایت شما مراجعه می کنند و حضور هر یک منجر به باز و بسته شدن Connection می شود که هزینه های گزافی را در پی دارد.یک راه حل بهینه سازی در این شرایط استفاده از Connection Pooling است.Connection Pooling تعداد دفعات باز و بسته شدن Connection هایی را که نیاز به باز شدن دارند را کاهش می دهد. Connection Pooling به گونه ای Connection ها را مدیریت می کند یعنی Connection های باز را درون خود ذخیره می کند تا در مواقع لازم آنها را به کار گیرد.هر بار که کاربری تقاضای باز شدن Connection را می دهد شی Pooler نگاه می کند اگر آن Connection در Pool قابل دسترسی باشد به جای آنکه Connection را باز کند آن Connection باز شده از قبل که در Pool وجود داشت(توسط کاربری دیگر که زودتر اتصال برقرار کرده) را برای کاربر می فرستد تا صرفه جویی در باز و بسته شدن Connection صورت گیرد .برای بسته شدن نیز به همین صورت است یعنی اگر کاربر تقاضای بسته شدن Connection را بدهد Pool به جای بسته شدن Connection اصلی آن Connection که در خود ذخیره کرده را می بندد.این اعمال به صورت پیش فرض در asp.net وجود دارند و اگر می خواهید به چشم آید در انتهای ConnectionString

صفت pooling را True قرار دهید به این دلیل گفتیم که این اعمال به صورت پیش فرض در asp.net وجود دارند که اگر این کار را هم نکنیدباز هم Pooling=True است.شما همچنین می توانید تعداد Connection های دریافتی در Pool را نیز محدود کنید با دو متد Min Pool Size و Max Pool Size که مقادیر عدد صحیح دارند تنظیم کنید(در قسمت آخر ConnectionString):

```
<connectionStrings>
 <add name="aaa" connectionString="Data Source=.\SQLEXPRESS;Initial
Catalog=sql-test;Integrated Security=True;Min Pool Size=10"
  providerName="System.Data.SqlClient" />
</connectionStrings>
```

دو متد جدید ADO.NET در زمینه ی pooling به ترتیب ClearPool و ClearAllPools است.این دو متد از متد های Connection هستند. که ClearPool می تواند Connection خاص را ازpool پاک کند و ClearAllPools تمام Pool های Provider را پاک می کند.طرز استفاده از ClearAllPools به صورت زیر است که SqlConnection یک کلمه ی کلیدی است:

```
SqlConnection.ClearAllPools()
```

طرز استفاده از ClearPool به صورت زیر است که Connection خاصی (در اینجا Con است) را پاک می کند:

```
SqlConnection.ClearPool(con)
```

Connection Statistics:

از موارد جدید در NET 2. است که به وسیله ی آن می توان آماری از Connection بدست آورد و در اصل یک HashTable از این آمار و ارقام است که برای بدست اوردن آن از متد RetrieveStatistics استفاده می شود.ولی جهت بازیابی چون این متد یک HashTable بر می گرداند پس باید درون یک متغیر HashTable قرار گیرد:

```
Dim connectionString As String =
WebConfigurationManager.ConnectionStrings("aaa").ConnectionString
Dim con As SqlConnection = New SqlConnection(connectionString)
Dim statistics As Hashtable = con.RetrieveStatistics()
```

حتما می پرسید اعضا این HashTable کدامند؟مهم ترین آنها در زیر آمده است:

۱-ServerRoundtrips: تعداد در خواست ها به سرور.

۲-ConnectionTime: تعداد دفعاتی که Connection باز شده است

۳-BytesReceived: مجموع تعداد بایت هایی که از سرور پایگاه داده در یافت شده مانند حجمی از داده هایی که از اجرای دستوات شما بدست امده است.

٤-SumResultSets : تعداد Query هایی که شما اجرا کردید.

و ...

شما می توانید هر بار این امار را در یک جای امن مثل خود پایگاه داده ذخیره کنید و آماری از کارکرد پایگاه داده ی خود بدست آورید:

```
    Dim connectionString As String =
WebConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As SqlConnection = New SqlConnection(connectionString)
    Dim statistics As Hashtable = con.RetrieveStatistics()
  Response.Write("ServerRoundtrips: " &
statistics("ServerRoundtrips").ToString() & "<br>")
    Response.Write("ConnectionTime: " &
statistics("ConnectionTime").ToString() & "<br>")
    Response.Write("Retrieved bytes: " &
statistics("BytesReceived").ToString() & "<br>")
  Response.Write("SumResultSets: " & statistics("SumResultSets").ToString()
& "<br>")
```

:The Command Class

همانطور که از نامش بر می آید کلاسی برای دستورات است.یعنی جزییات کنترل DataBase را مشخص می کند.شامل خواص مهم زیر است:

۱-CommandText: خود دستور است.حال چه SQL و چه StoredProcedure.

۲-CommandType: که نوع دستور را مشخص می کند:

-CommandType.Text که مقدار پیش فرض بوده ودستور را از نوع SQL در نظر می گیرد.

-CommandType.StoredProcedure که نوع دستور را روال ذخیره شده در نظر می گیرد

۳-Connection: مشخص می کند به کدام پایگاه داده وصل است که این کار با دادن یک Connection به این خاصیت میسر است.

۲-Parameters: پارامتر های ورودی هر دستور می باشد که در جای خود در موردش صحبت می کنیم.

کلاس Command شامل ۳ متد مهم برای اجرای دستورات نیز هست که قبل از آشنایی با آنها بهتر است با کلاس DataReader نیز آشنا شوید:

شی DataReader یا داده خوان سریع ترین شی است که می تواند اطلاعات را از پایگاه داده بخواند(البته اطلاعات اجرا شده توسط Command را) .این شی متد های مختلفی دارد که در زیر تعدادی از آنها آمده است:

-Read متدی که نشان می دهد DataReader در حال خواندن داده است.

-GetValue متدی است که مقداری عددی می پذیرد که این مقدار همان اندیس ستون مورد نظر است یعنی کل اطلاعات یک ستون خاص را به ما می دهد:

```
Response.Write(dr.GetValue(0))
```

-GetString متدی که مقداری عددی می پذیرد که این مقدار همان اندیس ستون مورد نظر است یعنی کل اطلاعات از نوع رشته ی یک ستون خاص را به ما می دهد.

- GetInt32 هم مقداری عددی می پذیرد که آن هم ستونی را انتخاب می کند و داده های عددی آن ستون را نمایش می دهد.به همین ترتیب Getfloat,GetDouble,GEtDecimal,GetDate و ...

-NextResult متد مهمی که می توان با استفاده از آن سطر به سطر مقادیر را از جدول مورد نظر خواند که در ادامه مثال مهمی از آن را خواهیم دید.

-Close که باعث بسته شدن شی داده خوان می گردد و فضای اشغال شده ی آن آزاد می شود.

حال به ۳ متد مهم شی Command می پردازیم:

۱-ExecuteNonQuery:حالتی است که تنها دستور را اجرا می کند و هیچ مقدار برگشتی ندارد. مثلا اگر شما بخواهید داده هایی را در پایگاه داده بنویسید خوب نیازی به برگشت مقدار ندارید پس می توانید از این متد استفاده کنید:

```
    Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim cmm As New SqlCommand
    cmm.CommandText = "INSERT INTO customer VALUES('rahmat','manbaab','tehran')"
    cmm.Connection = con
    con.Open()
```

```
    cmm.ExecuteNonQuery()
  con.Close()
```

می بینید که خصوصیت CommandText آن را SQL Query قرار دادیم و Connection آن را نیز همان Connection که بالاتر از آن تعریف کردیم(Con) دادیم. CommandType را نیز مقدار دهی نکردیم چون پیش فرض آن CommandType.Text است که به معنای دستورات SQL است.سپس Connection را باز کردیم (یادتان باشد که همیشه قبل از اجرا(Execute) کردن Connection, Query آن را باز کنید-نه خیلی قبل بلکه بلافاصله قبل از اجرا-) پس از اجرا نیز Connection را بستیم.به این ترتیب یک سطر به سطر های جدول customer اضافه شد.

۲-ExecuteScalar:متدی که ضمن اجرایQuery مقداری از نوع عدد یا رشته یا اصلا بهتر است بگوییم همان نوع داده ای که از پایگاه داده گرفته را بر می گرداند(این مقدار فقط یک مقدار است نه یک جدول و نه یک سطر است و اگر شما درQuery خود جدول و یا سطر را انتخاب کنید تنها اولین مقدار آن را بر می گرداند مثلا اگر من یک جدول را انتخاب کنم و سپس با ExecuteScalar آن را اجرا کنم خروجی من مقدار موجود در سطر اول و ستون اول از آن جدول است :

```
    Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim cmm As New SqlCommand
    cmm.CommandText = "SELECT customer_name FROM customer WHERE customer_street='gohar'"
    cmm.Connection = con
    Dim j As String
    con.Open()
    j = cmm.ExecuteScalar()
    con.Close()
  Response.Write(j)
```

می بینید که در این تابع یک متغیر به نام j از نوع string تعریف کردیم سپس Connection را باز کرده و آن را با ExecuteScalar اجرا کردیم.اگر به خود Query نگاه کنید عبارت Select را می بینید این یعنی مقداری باید بر گشت داده شود و این مقدار از جدول Customer در جایی که Customer_streetآن برابر گوهر است.پس این مقدار در j قرار می گیرد و Connection بسته شده و مقدار j در خروجی چاپ می شود.

۲-ExecuteReader:علاوه بر اجرای Query می تواند یک شی DataReader بر گرداند که برای بازیابی آن می بایست یک شی از نوع DataReader تعریف کنید(dr) و سپس مقدار حاصل از Execute Reader را در آن قرار دهید و در نهایت برای نمایش ,آن متغیر(dr) را به عنوان منبع داده ای یک کنترل مثل GridView در نظر بگیرید :

```
    Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim cm As New SqlCommand("SELECT * FROM customer", con)
    Dim dr As SqlDataReader
    con.Open()
    dr = cm.ExecuteReader()
    GridView1.DataSource = dr
    GridView1.DataBind()
    dr.Close()
 con.Close()
```

همانطور که دیدید در آخر هم ما شی DataReader را بستیم و هم شی Connection را.یک متد در بحث Execute Reader وجود دارد به نام CommandBehavior که با استفاده از آن می توان Connection را بست:

```
 dr = cm.ExecuteReader(CommandBehavior.CloseConnection)
```

به این ترتیب دیگر نیازی به بستن Connection در انتهای کار نیست.

حال این دو چه فرقی با هم دارند؟همیشه هر چه قدر فاصله ی بین اجرای Query و بسته شدن Connection کم تر باشد کارایی برنامه بهتر می شود.در کد اول این فاصله زیاد است به این صورت که بین اجرا و بسته شدن ۳ خط کد است.حتی اگر ۰ خط کد هم بود باز هم خوب نیست چون می توان آن را در همان خط هم بست(منظور از خط را اشتباه نگیرید در vb.net می توان چند دستور را در یک خط نوشت ولی منظور ما ترتیب اجرا و م وازی اجرا شدن Close Connection و اجرای آن است) با CommandBehavior به صورت بالا می توان همزمان هم اجرا کرد و هم connection را بست.

نوعی دیگر از خواندن داده ها بدون استفاده از کنترل هایی مثل Gridviewاست.یعنی بدون هیچ کنترلی:

```
    Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim cm As New SqlCommand("SELECT customer_name,customer_street,customer_city FROM customer", con)
    Dim dr As SqlDataReader
```

```
    Response.Write("connection is : " & con.State.ToString & "<br>")
    con.Open()
    dr = cm.ExecuteReader(CommandBehavior.CloseConnection)
    While dr.Read
      Response.Write(dr("customer_name") & Microsoft.VisualBasic.vbTab)
      Response.Write(dr("customer_street") & Microsoft.VisualBasic.vbTab)
      Response.Write(dr("customer_city") & "<br>")
  End While
```

نکته ی این تمرین در حلقه ی While آن است.در این حلقه ما از متد Read استفاده کردیم یعنی تا وقتی که جا دارد از شی DataReader به نام dr بخواندو سطر به سطرجلو رود و مقادیر ستون های مشخص شده را چاپ کند بدون هیچ کنترلی.یعنی ما آرگومان ورودی شی datareader را نام ستون های مورد نظر جهت نمایش قرار دادیم.البته اشکال این کار بی نظمی در نمایش داده ها است.

بحث مهم دیگر در زمینه ی ExecuteReader بحث Processing Multiple Result Sets است:

که در SQL به MARS یا Multiple Active Result Sets معروف است که زمینه ی اجرای چند Query در یک شی Command را فراهم می آورد.بدون استفاده از چندین Connection و یا چندین شی Command.این عمل می تواند در صورتی که شما مقادیر زیادی از داده ها را بازیابی می کنید موثر باشد.برای این عمل کافی است بین Query ها را با علامت ; پر کنیم.در زیر Command ما حاوی ۳ Query است که با ; از هم جدا شده اند:

```
  Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
  Dim sc As New SqlConnection(cs)
  Dim cmm As New SqlCommand
  cmm.CommandText = "SELECT  * FROM customer;SELECT  * FROM borrower;SELECT * FROM account"
  cmm.CommandType = CommandType.Text
cmm.Connection = sc
  Dim a As SqlDataReader
  sc.Open()
a = cmm.ExecuteReader()
```

به این عمل MARS می گویند.حال کار متد NextResult اینجا مشخص می شود:

```
    Dim i As Integer = 1
     Do
       Do While a.Read
         If i = 1 Then
          GridView1.DataSource = a
```

```
            GridView1.DataBind()
          ElseIf i = 2 Then
            GridView2.DataSource = a
            GridView2.DataBind()
          Else
            GridView3.DataSource = a
            GridView3.DataBind()
          End If
        Loop
        i += 1
      Loop While a.NextResult
    sc.Close()
  a.Close()
```

شاید این کد کمی پیچیده به نظر آید ولی ابتدا بگذارید در مورد خروجی صحبت کنیم.ما ۳ Query داشتیم و می خواهیم خروجی هر کدام را در یک GridView نمایش دهیم.حال هر ۳ تای این Query ها به روش MARS در یک Command اجرا شده اند وهمه در یک شی Reader به نام a قرار گرفته اند.و ما بایداین سه را از هم تفکیک کرده و هر کدام را در یک GridView خاص نمایش دهیم.قبل از توضیح نحوه ی کار بدانید که ما یک حلقه داریم به نام Do...Loop While که شرط این حلقه جلوی Loop While ذکر می شود تا کد های داخل حلقه حداقل یک بار اجرا شوند و حلقه ی دیگری به نام Do While...Loop که شرط آن جلو Do While می آید.ما برای تفکیک خروجی این ۳ Query که ۳ جدول هستند از شگرد برنامه نویسی استفاده کردیم.به این صورت که یک متغیر به نام i تعریف کردیم که اگر ۱ باشد a به GridView1ریخته شود اگر ۲ باشد a به GridView2ریخته شود و اگر ۳ باشد...حال چه موقع یک است؟در اول ورود به حلقه.پس GridView1 پر می شود به این صورت که تا جایی که می خواند آن را به GridView1 بریزد.سپس از حلقه خارج شده و i= ۲ می شود و NextResult به کار افتاده و شی DataReader به سراغ خروجی Query دوم می رود و به همین ترتیب GridView دو و سه هم پر می شوند.در نهایت هم Connection و شی DataReader را می بندیم.

نکته ی مهم اینجا این است که چرا ما با CommandBehavior , Connection را نبستیم تا کارایی سایت بالا رود؟به این دلیل که اگر میبستیم دیگر در خطوط بعدی که برنامه شی DataReader که Connection آن همان Connection اصلی سایت است وجود داشته اجرا نمی شد.پس هر وقت دیدید که پس از ExecuteReader از شی

DataReader یا Connection آن استفاده نشد می توانید از CommandBehavior و متعاقبا خصوصیت CloseConnection آن استفاده کنید.

SQL Injection Attacks

فرض کنید می خواهید یک بازیابی از پایگاه داده ی خود انجام دهید.مثلا افرادی را از جدول Customer که نام خیابانشان آنچه که خود کاربر بگوید باشد(چه بسا یک نفر).خوب شما این کا را با اضافه کردن عبارت WHERE به عبارت SQLمی کنید.ولی چگونه نام خیابانی که مد نظر کاربراست را به پایگاه داده بفرستید؟ خوب یک TextBox به صفحه اضافه کرده و نام خیابان را از TextBox از کاربر می خواهید.خوب چگونه محتویات این TextBox را به دستور SQL اضافه کنیم.برای این کار تمام کار هایی که تا حالا در رویداد Page_Load انجام می دادیم را به صورت یک تابع در می آوریم به نامret که ورودی آن نام خیابان است و مقدار برگشتی آن نام ان Customer :

```
    Public Function ret(ByVal street_name As String) As String
     Dim              cs              As              String              =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
     Dim con As New SqlConnection(cs)
     Dim cmm As New SqlCommand
     cmm.CommandText    =    "SELECT    customer_name    FROM    customer    WHERE
(customer_street='" & street_name & "')"
     cmm.Connection = con
     con.Open()
     Dim a As SqlDataReader
     a = cmm.ExecuteReader()
     Dim j As String = ""
     While a.Read
      j += (a("customer_name").ToString)
     End While
     con.Close()
     Return j
 End Function
```

همه چیز مثل قبل است به جز عبارتSQL .ما گفتیم اسامی مشتریانی را بده که نام خیابانشان برابر متغیر Street_Name (که آگومان ورودی تابع است) باشد.این کار را با آوردن عبارت زیر جلوی WHERE انجام دادیم:

```
  WHERE (customer_street='" & street_name & "')"
```

عبارت بین دو & همان نام خیابان است.حال کافی است در رویدادی مثل Page_Load آن را نمایش دهید:

```
Response.Write(ret(TextBox1.Text))
```

باید توجه کنید که عبارات ثابت رشته ای در SQL باید حتما در داخل '' قرار گیرند.در اینجا هم ما مقدار TextBox1 را بدون این علامت به تابع Ret فرستادیم چون در خود عبارت SQL علامت '' را به دو طرف ان عبارت اضافه کرده بودیم. اگر آنجا این کار را نمی کردید می بایست تابع را به صورت زیر صدا بزنید:

```
Response.Write(ret("'" & TextBox1.Text & "'"))
```

حتما می پرسید بحث SQL Injection Attacks پس چه شد؟خوب این کاری که ما در بالا انجام دادیم به هکر ها اجازه ی نفوذ می دهد.ما نباید برای در یافت مقداری از صفحه ی اصلی آن را به صورتی بین دو & در یافت کنیم.بلکه باید از روش Parametrize استفاده کنیم:

شی SqlParameters برای چنین مواقعی ساخته شده است.این شی ۳ متد اصلی دارد:

۱-Value:که مقدار اصلی Parameter است.

۲-ParameterName: که نام متغیری است که در عبارت Sql آمده است.

۳-DbType:که نوع این پارامتر را مشخص می کند.

برای درک بهتر مثال بالا را از روشParametrize حل می کنیم:

```
   Public Function ret(ByVal street_name As String) As String
    Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim cmm As New SqlCommand
    cmm.CommandText = "SELECT customer_name FROM customer WHERE (customer_street=@strname)"
    cmm.Connection = con
    Dim par As New SqlParameter
    par.Value = street_name
    par.ParameterName = "@strname"
    par.DbType = DbType.String
    cmm.Parameters.Add(par)
    con.Open()
    Dim a As SqlDataReader
    a = cmm.ExecuteReader()
    Dim j As String = ""
    While a.Read
    j += (a("customer_name").ToString)
    End While
```

```
    con.Close()
    Return j
End Function
```

نترسید شاید کمی کد نویسی دارد ولی در عوض امنیت بیشتری در مقابل هکر ها دارد.شی Parameter را با New کردن ساختیم:

```
Dim par As New SqlParameter
```

مقدار آن را(Street_Nsme)همان ورودی تابع دادیم :

```
par.Value = street_name
```

و نام ان را (که حتما چون پارامتر است بایداولش @ باشد) همان نامی دادیم که در آخر دستور SQL مشاهده می کنید (@strname):

```
par.ParameterName = "@strname"
```

و سپس نوع آن را String تعیین کردیم :

```
par.DbType = DbType.String
```

و سپس آن را به شی SqlCommand که در اینجا cmm است Add کردیم:

```
cmm.Parameters.Add(par)
```

دقت کنید علامت @ باید هم در کنار نام پارامتر در عبارت SQL باشد و هم در کنار نام پارامتر در قسمت Par.ParameterName.

اگر این اعمال در کد نویسی باعث زیاد شدن بی رویه ی خطوط می شود(مخصوصا وقتی تعداد Parameter ها بیش از یکی می شود) می توان آن را در یک خط هم نوشت:

```
cmm.Parameters.Add("@strname", SqlDbType.VarChar).Value = street_name
```

که در این کد می بینید که اصلا نیازی به معرفی شی Parameter هم نیست وبسیار ساده تر هم هست ولی قبلی خوانا تر بود.

حتما شاید از خود بپرسید Sql Injection Attack در حالت قبلی چگونه می توانست پیش آید؟ به این کد نگاهی بیندازید با علم اینکه ما یک جدول در پایگاه داده ی خود داریم به نام borrower که شامل دو ستون به نام های customer_name وloan_number است که اولی مقداری رشته ای و دومی مقداری عددی است و می خواهیم با گرفتن شماره ی وام از یک TextBox نام صاحب یا صاحب های آن وام را به خروجی ببریم بدون استفاده از روش پارامتری سازی:

```
   Public Function ret1(ByVal cn As String) As String
    Dim              cs              As              String              =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim cmm As New SqlCommand
```

```
    cmm.CommandText = "SELECT customer_name FROM borrower WHERE loan_number="
& cn
    cmm.Connection = con
    con.Open()
    Dim a As SqlDataReader
    a = cmm.ExecuteReader()
    Dim j As String = ""
    While a.Read
    j += (a("customer_name").ToString & Microsoft.VisualBasic.vbCrLf)
    End While
    con.Close()
    Return j
 End Function
```

کافیست برای اجرا یک TextBox و یک Button در صفحه قرار دهید و در رویداد کلیک دکمه کد زیر را بنویسید:

```
   Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
     Response.Write(ret1(TextBox1.Text))
 End Sub
```

پس مثلا اگر من در این TextBox عدد ٥ را بنویسم صاحبان وام شماره ی ٥ نمایش داده می شوند.حال اگر شما قدری باهوش باشید میبینید که اگر هکر ها به جای ٥ عبارت زیر را در TextBox بنویسند چه پیش می آید:

5;DELETE FROM borrower WHERE loan_number=4

بله یک فاجعه رخ می دهد.این کد اجرا می شود و تمام صاحب وام ها با شماره ی ٥ نمایش داده می شوند ولی تمام صاحب وام ها با شماره ی ٤ از پایگاه داده ی ما پاک می شوند.حال این یک شکل از SQL Attack است این امر اشکال مختلفی دارد.روش ما در بالا زیاد جالب نبود چون برای جلوگیری از کار هکر کافی است ورودی تابع را از String به Integer تبدیل کنیم(از عمد این کار را نکردیم تا طرز کا را متوجه شوید) ولی در مثال زیر کاملا عادی هکر کارش را (بازرنگی) انجام می دهد:

مثلا فرض کنید تابع شما به صورت زیر است که با گرفتن نام خیابان از شما , نام مشتری که در آن خیابان است را بر می گرداند:

```
   Public Function ret3(ByVal cn As String) As String
    Dim cs As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim cmm As New SqlCommand
  cmm.CommandText = "SELECT customer_name FROM customer WHERE
customer_street='" & cn & "'"
```

```
    cmm.Connection = con
    con.Open()
    Dim a As SqlDataReader
     a = cmm.ExecuteReader()
    Dim j As String = ""
    While a.Read
      j += (a("customer_name").ToString & Microsoft.VisualBasic.vbCrLf)
    End While
    con.Close()
    Return j
End Function
```

و مثل قبل یک TextBox و یک Button هم دارید و رویداد کلیک دکمه مثل زیر است:

```
   Protected Sub Button2_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Response.Write(ret3(textbox2.Text))
End Sub
```

حال از کاربر می خواهید نام خیابان را واردکند و وی عبارت زیر را وارد می کند:

azadi';DELETE FROM borrower WHERE loan_number=8;--

چه پیش می آید؟ما در این مثال جهت امنیت بیشتر علامت " را در خود عبارت SQL قرار دادیم تا هکر حتما نامی را که وارد می کند بین این دو علامت قرار گرفته و به عنوان نام خیابان باشد ولی هکر خودش یک ' به نام خیابان اضافه کرد سپس کد sql مد نظرش را نوشت و سپس با علامت - - علامت ' ما را خنثی کرد.زیرا علامت - - نقش comment را در sql داشته و هر چه پس از آن بیاید توضیح تلقی می شود حتی اگر علامت " را در قسمت صدا زدن تابع هم می نوشتید به صورت زیر:

```
Response.Write(ret3("'" & textbox2.Text & "'"))
```

باز هم تاثیری نداشت.پس دیدید که چگونه شما را فریب می دهند.

پس سعی کنید از روش Parametrize برای ارسال مقادیر به پایگاه داده استفاده کنید.چون آن وقت اگر هکر هر رشته ای را در داخل TextBox بنویسد Parameter آن را به عنوان متغیر در نظر گرفته و به هیچ عنوان دستور SQL تلقی نمی شود.البته در برخی موارد برخی قوانین جامعیت موجودیتی در پایگاه داه وجود دارند که مانع پاک شدن همینطوری داده از جداول می شوند.

برای دسترسی به اطلاعات بیشتر و پیشرفته از این مبحث خطرناک و مهم در زمینه ی امنیت داده ها در داده های Sql می توانید به منبع فارسی و ۷۷ صفحه ای SQL

Injection-wolf رجوع کنید که در سایت ParsTech.org به صورت رتیگان یافت می شود رجوع کنید.

StoredProcedures

یکی از کاربرد های جالب Parametrize استفاده از آن در ساختن StoredProcedures است.به عقیده ی خیلی ها اگر یک سری دستور SQL را کنار هم بچینیم حاصل کار StoredProcedures می باشد.قبل از تعریف خودم از StoredProcedures بگذارید فواید آن را بگوییم:

نگهداری آن بسیار ساده بوده و حجم بسیار کمی را اشغال می کند و همچنین شما می توانید به جای آنکه دستورات SQL خود را در صفحه ی ASP.NET بنویسید می توانید آن را درStoredProcedures قرار داده و از آن استفاده کنید و همچنین مثل تعریف تابع,میتوان آن را یک بار نوشت وچندین بار استفاده کرد. و در کل استفاده از آن باعث افزایش کارایی برنامه ی ما می شود.

StoredProcedures مانند یک تابع است.تابعی که یک عمل خاص را انجام می دهد و می تواند آرگومان ورودی داشته باشد و حتی مقداری را Return کند.این اعمال در یک Query Builder صورت می گیرد.برای ساخت یک StoredProcedures کافی است از کلمه ی کلیدی CREATE PROCEDURE استفاده کنیم.این کار را در Visual Studio هم می توان انجام داد (حتما لازم نیست به نرم افزار Microsoft Sql Server رجوع شود) به شرطی که SQL Server روی سیستم شما نصب باشد. البته از شروع ADO.NET می بایست نصب می شد.کافی است در قسمت مربوطه گزینه ی Add New StoredProcedures را کلیک کنید تا صفحه ای بازشود و الی آخر ...

ولی ما به طور دستی این کار را می کنیم:

```
CREATE PROCEDURE dbo.mystoredprocedure

AS

RETURN
```

متغیر های ورودی بین عبارت Create وAs قرار می گیرند و عملی که روال ذخیره شده انجام می دهد بین AS و Return انجام می گیرد.احیانا اگر می خواستید مقدار برگشتی هم داشته باشید آن را جلوی Return قرار دهید.

حال برای مثال یک StoredProcedures به نام man ایجاد می کنیم.

ابتدا آن را ایجاد می کنیم:

```
CREATE PROCEDURE dbo.man(
```

سپس آرگومان های ورودی آن را به همراه نوع داده ای آنها مشخص می کنیم:

```
  @customer_name varChar(30),
  @customer_city varChar(30),
@customer_street varChar(30))
```

سپس عملی که قرار است انجام شود را پس از AS می نویسیم .ما می خواهیم این آرگومان های ورودی را در جدولCustomer ,Insert کند:

```
  AS

  INSERT INTO customer
  (customer_name,customer_city,customer_street)
  VALUES
(@customer_name,@customer_city,@customer_street)
```

مقداری هم نمی خواهیم بر گردانیم:

```
    RETURN
```

دیدید چه قدر ساده بود؟.حال می خواهیم از این StoredProcedures استفاده کنیم.یعنی با استفاده از آن درون جدول Customer را با مقادیری که ما از کاربر از طریق TextBox دریافت می کنیم پر کنیم.به این منظور ابتدا مراحل اولیه ی ADO.NET را انجام می دهیم:

Imports کردن فضاهای نام مورد نیاز:

```
  Imports System.Data.SqlClient
  Imports System.Data
Imports System.Web.Configuration
```

برای این کار تابعی به نام insert ایجاد می کنیم و آرگومان ورودیش را همان مقادیری که می خواهیم درون جدول Customer قرار دهیم در نظر می گیریم:

```
   Public Sub insert(ByVal name As String, ByVal city As String, ByVal street
As String)

 End Sub
```

مشخص کردن رشته ی اتصال و Add کردن آن به شی SqlConnection:

```
  Dim connectionString As String =
WebConfigurationManager.ConnectionStrings("aaa").ConnectionString()

Dim con As New SqlConnection(connectionString)
```

حال به نقطه ی حساس می رسیم و آن این است که شی Command را تعریف کنیم:

```
Dim cmd As New SqlCommand("man", con)
```

همیشه ما در قسمت اول که مربوط به CommandText است دستورSQL را می نوشتیم ولی حالا که از StoredProcedures استفاده می کنیم این قسمت همان نام StoredProcedures می باشد.ولی Command از کجا بداند که این گونه است؟خوب کاری ندارد همان طور که قبلا گفتیم CommandType نوع دستور را (SQL یا StoredProcedures و...) مشخص می کند :

```
cmd.CommandType = CommandType.StoredProcedure
```

حال نوبت مشخص کردن Parameter است.ما در اینجا از ساده ترین روش استفاده کردیم:

```
    cmd.Parameters.Add("@customer_name", SqlDbType.VarChar).Value = name
    cmd.Parameters.Add("@customer_street", SqlDbType.VarChar).Value = street
  cmd.Parameters.Add("@customer_city", SqlDbType.VarChar).Value = city
```

همانطور که می بینید ما در اینجا مقادیر ورودی تابع را یکی یکی به Value پارامتر ها دادیم و متغیر آنها هم آرگومان های ورودی StoredProcedures است که با علامت @ نشان دادیم که رجوع می کنند به StoredProcedures.

در نهایت هم اجرای دستورات بدون هیچ مقدار برگشتی:

```
    con.Open()
    cmd.ExecuteNonQuery()
  con.Close()
```

برای اجرای این تابع هم کافیست ۳ TextBoxو یک Button در صفحه قرار داده و رویداد کلیک Button را به صورت زیر بنویسید:

```
   Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
     insert(TextBox1.Text, TextBox2.Text, TextBox3.Text)
 End Sub
```

حال شما از این StoredProcedures می توانید بار ها استفاده کنید و کارایی برنامه ی خود را افزایش دهید.جلو تر با StoredProcedures های مختلف(شامل عملیات و ورودی های مختلف) نیز آشنا خواهید شد.

Transactions

با توجه به اینکه امروزه (به دلیل حجم گسترده ی اطلاعات)، Database یکی از ارکان اصلی هر نرم افزار است، رفته رفته نیاز به مدیریت اطلاعات درون آن، بیشتر از پیش احساس می شود. یکی از جنبه های این مدیریت، جلوگیری از ورود اطلاعات نامعتبر به Database است، که امروزه حتی در نرم افزار های کاربردی و خانگی مورد نیاز

است. بحثی فوق العاده مهم و کلیدی در SQL که در زمره مباحث پیشرفته در آن محسوب می شود وبسیار پر کاربرد نیز هستTransaction می باشد.یادتون هست در انتهای بحث تجارت الکترونیک وقتی به مرحله ی CheckOut رسیدیم گفتیم که این مرحله نیاز به یک کنترل Wizard دارد تا مرحله به مرحله کار خرید ما را انجام دهد.فرض کنید در هر مرحله یک جدول در Database تکمیل شود و با پایان پذیرفتن آن کار خرید شما تمام می شود.حال اگر یک دفعه وسط طی کردن مراحل کنترل Wizard ,اگر ارتباط با سرور قطع شود(برق قطع شود-اتصال به اینترنت قطع شود –زلزله بیاید و ...) مثلا یک جدول که حاوی مشخصات فردی است پر شود و قسمت های دیگر خالی بماند با رخ دادن قطعی ارتباط با سرور نه شما خرید خود را انجام داده اید و نه سایت مورد نظر به شما جداول خود را پر کرده است و هر دو به شدت دچار مشکل می شوید(مخصوصا شما ,چون ممکن است در قسمت مربوط به شماره و مبلغ Creadit Card , پول از حساب شما کم شود در حالیکه شما روی دکمه ی Finish کلیک نکرده اید ویا اینکه هنوز آدرس پستی خود را وارد نکرده اید و یا نام شما به لیست ارسالی های سایت واردنشده و ده ها احتمال دیگر!.Transaction از وقوع چنین مشکلاتی جلوگیری می کند.Transaction یک واحد برنامه نویسی است که برای مدیریت دسترسی به پایگاه داده ایجاد شده است.ویژگی های مهم Transaction در زیر آمده است:

۱-اتمیک بودن:یعنی یک یا چند تراکنش یا اجرا می شوند یا اجرا نمی شوند برای مثال بالا هم به این صورت است که یا خرید کالا تا آخر انجام می گیرد و تمام جداول پر می شوند و یا اینکه اصلا نه جدولی پر می شود و هیچ عملی انجام نمی گیرد(از کارت شما پولی کسر نمی شود).

۲-سازگاری:یعنی سازگاری داده ها قبل و بعد از تراکنش حفظ می شود در مثال بالا هم یعنی قبل از تراکنش خرید انجام نشده بود و پس از آن خرید انجام شد .در هر دو حالت همه چیز سر جایش است.

۳-جداسازی:یعنی تراکنش ها روی هم اثر مخرب ندارند. در مثال بالا هم یعنی اگر دو نفر قصد Chek Out داشتند این دو تراکنش هیچ ربط و تاثیری به هم ندارند.تراکنش در دو حالت مهم انجام می شود.اولی Commit است که به این معنی است که تراکنش با موفقیت انجام شده و پایگاه داده با آن تغییر می کند.دومی RollBack است که یعنی تراکنش دچار شکست شده و تغییرات بوجود آمده در پایگاه داه باید به حالت اول در آیند.

Transaction با کلمه ی کلیدی Begin Transaction شروع شده و با Commit یا RollBack خاتمه می یابد.یک مثال ساده از تراکنش شما را متوجه کار می سازد.فرض کنید دستور ماInsert کردن دو رکورد به DataBase است و Connction ما هم مثل قبل:

```
   Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
   Dim con As New SqlConnection(cs)
   Dim cmd1 As New SqlCommand("INSERT INTO customer VALUES ('aliakbar','dinpoor','qazvin')", con)
 Dim cmd2 As New SqlCommand("INSERT INTO customer  VALUES ('reza','hadinezhad','tehran')", con)
```

برای تعریف تراکنش ابتدا آن را تعریف کرده سپس مقدار اولیه ی آن را تهی در نظر می گیرید:

```
Dim tran As SqlTransaction = Nothing
```

آنچه که مهم است این است که در ایجاد هر تراکنش بلوک Try...Catch...Finally...End Try نقش کلیدی دارد.این بلوک یک مکان امن برای اجرای هر تراکنش است.کارکرد آن نیز به این صورت است که عملیات اصلی شما بین Try و Catch نوشته می شود.حال در صورت مواجهه با Error در این عملیات,شما وارد قسمت بین Catch و Finally می شوید (به جای ایراد کلی در برنامه) و در نهایت چه با خطا روبرو شوید چه نشوید قسمت بین Finally وEnd Try حتما انجام می گیرد.مثال کلی از این بلوک را می توان تقسیم به ۰ زد:

```
   Dim a, b, c As Integer
   a = 5
   b = 0
   Try
   c = a / b
   Catch
     Response.Write("Oops, you can’t divide by zero.<br>")
   Finally
     Response.Write(c)
End Try
```

در این کد ابتدا عملیات اصلی در قسمت Try انجام می گیرد که تقسیم a به b و ریختن حاصل در c است.سپس اگر هر مشکلی پیش آید به قسمت Catch رفته و پیغام خطا را می بینید.سپس در هر صورت مقدار c به خروجی می رود.اگر تقسیم به ۰ هم باشد مقدار c ۰ خواهد بود(نمیدونم چرا!!).شما می توانید Error واقعی را نیز به خروجی ببرید به

این صورت که جلوی قسمت Catch یک متغیر از نوع Exception تعریف کنید و سپس به شکل زیر آن را به خروجی ببرید:

```
Dim a, b, c As Integer
a = 5
b = 0
 Try
c = a / b
 Catch d As Exception
   Response.Write(d.Message)
 Finally
  Response.Write(c)
 End Try
```

به این ترتیب پیغام خطای اصلی Asp.NET در خروجی ظاهر می شود.شما حتی می توانید خطا ها را سفارشی کنید.به این ترتیب که متغیر جلوی Catch را از نوع خطای دلخواه خود بگیرید.من در زیر گفتم اگر خطای رخ داده تقسیم بر صفر بود پیغام مربوط و مختص به آن را ظاهر کن:

```
Dim a, b, c As Integer
a = 5
b = 0
Try
c = a / b
Catch d As DivideByZeroException
  Response.Write(d.Message)
Finally
  Response.Write(c)
End Try
```

می توانیم در یک بلوک Try...Catch...Finally...End Try چندین Catch قرار دهیم واز رویدادن خطا های مختلف جلوگیری کنیم. خوب بحث در مورد Try...Catch...Finally...End Try کافیه.حال به ادامه ی مثال توجه کنید.بلوک را با کلمه ی Try آغاز کرده و در اولین قدم Connection را Open می کنیم:

```
Try
 con.Open()

Catch ex As Exception

Finally

End Try
```

حال باید متغیری که از نوع Transaction تعریف کرده بودیم را با متد BeginTransaction Connectionی که پیش از این تعریف کرده بودیم مقدار دهی کنیم:

```
tran = con.BeginTransaction()
```

این یعنی آغاز کار تراکنش.سپس این متغیر را به عنوان تراکنش به هر یک از Command ها نسبت می دهیم.این کار را با متد Transaction شی command انجام می دهیم:

```
   cmd1.Transaction = tran
cmd2.Transaction = tran
```

در نهایت ان ها را اجرا می کنیم:

```
   cmd1.ExecuteNonQuery()
cmd2.ExecuteNonQuery()
```

در این مورد بگویم که چون هردو تا Executenonequery ها درون یک تراکنش هستند پس یا هر دو اجرا می شوند و یا هیچ کدام اجرا نمی شود و این طورنیست که یکی اجرا شود و دیگری ایجاد نشود(این نکته بسیار مهم است چون اصلا کار تراکنش همین است).

```
   tran.Commit()
Response.Write("transAction Commited!")
```

حال بلوک Catch هم وقتی رخ می دهد که تراکنش با موفقیت انجام نگیرد اولا تمام تغییرات رخ داده در DataBase را به حالت اول بر می گردانیم:

```
   Catch ex As Exception
 tran.Rollback()
```

سپس پیغامی برای آگاهی بیشتر به خروجی می بریم که این پیغام همان پیغام خطای asp.net است:

```
Response.Write(ex.Message)
```

بلوک Finally هم بهترین جا برای بستن Connection است چون در هر صورت اجرا می شود.دوست دارید طرز کار را ببینید و از آنچه که انجام دادید لذت ببرید؟برای دیدن طرز کار اول بگویم که چه موقع ممکن است در بلوک Error, Tryپیش آید.ببینید من در جدول Customer سه عنوان دارم یکی نام مشتری یکی خیابان وی و آخری شهر وی است.و من نام مشتری را کلید اصلی در نظر گرفتم.یعنی سیستم دو نام یکسان را قبول نمی کند.حال ادامه ی کار.من از عمد دو Command به نام های cmd1 و cmd2 به کار گرفتم که هر دو یک سری داده را در جدول Customer, insert می کند.و از عمد

نام داده ی اول را aliakbar گرفتم که در پایگاه داده وجود ندارد(یعنی مجاز به insert باشد) و دومی را Reza گرفتم که قبلا در پایگاه داده وجود داشت (یعنی مجاز به insert نباشد) تا تراکنش با Error مواجه شود.از آنجایی هم که گفتیم یا هر دو اجرا می شوند یا هیچ کدام اجرا نمی شود با اجرای این کد در اصل Execute اولی باید اجرا شود(یعنی aliakbar وارد جدول شود) سپس به دومی که رسید یعنی Reza چون در پایگاه داده وجود داشت ان را وارد جدول نکند.ولی می بینیم با اجرای تراکنش هیچ کدام وارد جدول نشدند.چون هر دو با هم عضو یک تراکنش بودند.برای درک بیشتر تابع زیر را که همان کار را بدون تراکنش می کند را اجرا کنید .می بینید که aliakbar وارد جدول شده ولی Reza نشده:

```
    Public Sub withouttran()
    Dim              cs              As              String              =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    Dim con As New SqlConnection(cs)
    Dim    cmd1    As    New    SqlCommand("INSERT    INTO    customer       VALUES
('aliakbar','dinpoor','qazvin')", con)
    Dim    cmd2    As    New    SqlCommand("INSERT    INTO    customer       VALUES
('reza','hadinezhad','tehran')", con)
    Try
     con.Open()
     cmd1.ExecuteNonQuery()
     cmd2.ExecuteNonQuery()
     Response.Write("Insert Compeleted!")
     Catch ex As Exception
     Response.Write(ex.Message)
      Finally
     con.Close()
     End Try
 End Sub
```

این مثالی کامل برای درک تراکنش بود.ولی بهتر است بدانید برای همه چیز از تراکنش استفاده نکنید.سعی کنید برای موارد حساس(مثل بحث اول سایت(تجارت الکترونیک)) از تراکنش استفاده کنید.

از دیگر متد های کلاس SqlTransaction متد save است. این متد که Savepoint هم بهش میگویند,را می توانید هر جای دستورات خود (در داخل بلوک Try) قرار دهید .این کار باعث می شود شما هنگام RollBack شدن تراکنش به آن نقطه ای که Save کرده اید برگردید چون ممکن است همه ی دستورا ت نیاز به RollBack شدن نداشته

باشند.این عمل Save با یک نام صورت می گیرد تا هنگام RollBack شدن مقصد خود را بدانیم.این متد می تواند یک جور راه برای بهینه سازی تراکنش محسوب شود:

```
    ' شروع تراکنش
    Dim tran As SqlTransaction = con.BeginTransaction()
    ' دستورات مورد نیاز جهت اجرای تراکنش
    ' CompletedInsertساختن نقطه ی ذخیره و دادن نامی خاص از نوع رشته به آن
    tran.Save("CompletedInsert")
    ' دستورات مورد نیاز جهت اجرای تراکنش
    ' رول بک شدن به نقطه ذخیره شده در صورت نیاز
    tran.Rollback("CompletedInsert")
    ' کردن تراکنش Commit
 tran.Commit()
```

تراکنش را می توان به صورت کد SQL و حتی در StoredProcedure هم ایجاد کرد.برای این کار کمی باید با SQL بازی کنیم.برای این کار ۳ دستور در SQL وجود دارد به نام های ,Commit Tran Begin Tran و RollBack Tran.قبل از مثال در این روش با یک تابع سیستمی در SQL آشنا شوید به نام @@Error.این تابع هیچ مقداری را نمی پذیرد در حقیقت جزو توابع سیستمی بدون پارامتر SQL Server است.این تابع تعداد خطا های رخ داده در آخرین دستور SQLی که اجرا شده را برمی گرداند.نحوه ی استفاده از آن هم به فرم زیر است:

```
select * from customer
select @@error
```

در کد بالا ابتدا تمامی مقادیر جدول customer به خروجی رفت و سپس تعداد خطاهای رخ داده در آن ذکر شد.این را یادتان باشد که در SQL متغیر ها با علامت @variable name مشخص می شوند مثلا متغیری به نام temp در SQl به صورت @temp شناسایی می شود.البته تعریف متغیر ها با کلمه ی کلیدی DECLARE صورت می گیرد به فرم زیر:

```
Declare <variable name>  <variable data type>;
```

مثلا:

```
DECLARE @aaa varchar(10);
```

مقدار دهی آنها نیز می تواند به فرم زیر با عبارت SET باشد:

SET <variable name>=<variable value>

مثلا:

SET @aaa='ali';

می توان یک جا چندین متغیر را معرفی کرد:

DECLARE @aaa varchar(10),@bbb int;

و حتی می توان یک سری جدول موقتی نیز با Declare ایجاد و روی آنها کار هایی انجام داد.به فرم زیر:

```
DECLARE @MyTable Table
)
,OrderID int
(CustomerID char(5
(
INSERT INTO @MyTable
SELECT OrderID, CustomerID
FROM Northwind.dbo.Orders
WHERE OrderID BETWEEN 10240 AND 10250
SELECT
*FROM @MyTable
```

در کد بالا یک جدول موقت به نام @mytable ساختیم و مقادیری از یک جدول دیگر به نام orders را در آن قرار دادیم و سپس کل مقادیر جدول @mytable را از آن بازیابی کردیم. جدول @mytable در حقیقت وجود موقتی دارد و به عنوان یک جدول در پایگاه داده ی ما ذخیره نمی شود.دقیقا مثل تعریف یک متغیر.البته مقادیر موقتی را می توان با عبارات پرکاربردی همچون WITH نیز ایجاد کرد.یعنی می توان مقادیری را محاسبه کرد و حاصل را با استفاده از with ذخیره کرد و سپس در مکان های محتلف از آن استفاده کرد.

توابع سیستمی دیگری هم در Sql وجود دارند نظیر @@ROWCOUNT که تعداد سطر های Affect شده در آخرین دستوری که اجرا شده را برمی گرداند.منظور از تعداد سطر های Affect شده تعداد سطر های تحت تاثیر واقع شده از دستورات Sql مثل

Select و یا insert است.نکته ی دیگر هم در SQL این است که می توانید از شرط IF (condittion)...ELSE نیز استفاده کنید. مثلا در عبارت زیر ابتدا آخرین آمار در تعداد سطر های Affect شده در متغیر rowaffected قرار می گیرد و سپس چک می شود اگر از صفر بزرگترند متغیر result برابر ۰ و در غیر این صورت برابر ۱ باشد:

```
@rowaffected=@@rowcount
if (@rowaffected >0)
@result=0
else
@result=1
```

حال به مثال تراکنش می رسیم.همان عملکردی که در مثال ADO.NET از تراکنش را دیدیم در SQL هم می خواهیم تکرار کنیم.اگر دیده باشد ما برای اجرای تراکنش از یک بلوک Try...Catch استفاده کردیم تا تشخیص دهیم در اجرای Query خطا رخ داده یا نه.اگر رخ داد به عقب برمی گشتیم وگرنه آن را commit می کردیم.حالا هم با تابع @@error متوجه می شویم در هر یک از دستورات خطا رخ داده یا نه اگر رخ داد با دستور RollBAck Tran به عقب برمی گردیم ولی در صورت عدم رخداد خطا آن را Commit نمی کنیم زیرا در آن صورت با وقوع خطا در دستور بعدی به آخرین نقطه ی Commit شده بر می گردیم و در این صورت استفاده یا عدم استفاده از تراکنش توفیقی ندارد.عمل Commit را در انتهای دستور دوم اجرا می کنیم.آغاز تراکنش هم با Begin Tran خواهد بود:

```
begin tran
insert into customer values('aliakbar','dinpoor','qazvin')
if (@@error>0)
rollback tran
insert into teacher values('reza','hadinezhad','tehran')
if (@@error>0)
rollback tran
else
commit tran
```

به این ترتیب یا هر دو دستور اجرا می شوند و یا هیچ کدام اجرا نمی شوند.نکته ی دیگر هم این است که می توان (و چه بهتر است که) تراکنش را در داخل روال های ذخیره شده به کار برد.برای این کار فقط کافی است پس از عبارت AS در روال ذخیره شده

دستورات خود را با Begin Tran آغاز کنیم.مثال زیر عملکرد درستی ندارد ولی نمونه ای از اجرای تراکنش بالا در Stored Procedures است:

```
CREATE PROCEDURE insert_transaction
(
@f_name varchar(20),
@l_name varchar(20),
@city_name varchar(20)
)
AS
begin tran
insert into customer values('aliakbar','dinpoor','qazvin')
if (@@error>0)
rollback tran
insert into teacher values('reza','hadinezhad','tehran')
if (@@error>0)
rollback tran
else
commit tran
```

نکته ی آخر هم این است که برای تعریف SavePoint در SQL از Save Tran به فرم زیر استفاده کنید:

Save Tran ‘save point name’

روشها و کلاسهای متعددی برای استفاده از تراکنش وجود دارند که در اینجا تنها به دو تا از آنها اشاره شد.

Factory DbProvider

تا بحال اگر دقت کرده باشید تمام کدهای ADO.NET ما با Sql بود.یعنی هیچ یک از آنها با پایگاه داده ی Access کار نمیکرد.مثلا Connection ما از نوع SqlConnection بود.و یا DataReader-Command-Parameter و....پس استفاده از آن توابع در یک پایگاه داه به جز Sql غیر ممکن است.از نظر بعضی ها این یک مشکل محسوب می شود بنابراین ASP.NET 2 این مشکل را با ارایه ی Provider-Agnostic Code حل کرد. یک روش در جهت ایجاد Component است تا همه چیز را تحت نظر بگیرد.کلاس مورد استفاده ی ما در این روش کلاس dbproviderFactory است.با این کلاس شما می توانید تابعی برای اتصال و مدیریت پایگاه داده بنویسید که مانند یک Component عمل می کند و در همه نوع

پایگاه داده قابل استفاده است.برای استفاده از این کلاس باید فضای نام زیر را Imports کنید:

```
Imports System.Data.Common
```

برای تعریف متغیر از نوع این کلاس به صورت زیر عمل کنید:

```
Dim pr As DbProviderFactory
```

این کار یک متغیر به نام Pr ازنوع این کلاس را تعریف کرد.برای انجام هرگونه کاری با این متغیر ابتدا می بایست از متد GetFactory آن استفاده کنید که به نظر یک جور New کردن این کلاس است.برای این کار هم از متد dbProviderFactories استفاده می کنیم:

```
Dim pr As DbProviderFactory = DbProviderFactories.GetFactory("provider name
as string")
```

قسمت اول که اعلان متغیر است.در قسمت دوم اگر به آرگومان ورودی متدGetFactory دقت کنید نوشته شده نام تهیه کننده از نوع رشته ای!!!.این یعنی چه؟این همان فضای نام مربوطه به پایگاه داده است.این فضای نام را باید به آن بدهیم.اگر دستی این کار را کنیم به صورت زیر است:

```
DbProviderFactories.GetFactory("system.data.sqlclient")
```

ولی این اصلا درست نیست چون اگر این طور باشد کار ما فقط مربوط به SqlDataBase می شود و این با گفته های پیشین ما در رابطه با Component بودن این روش در تناقض است.خوب برای حل این مشکل می توانیم این رشته را به عنوان آرگومان تابع دریافت کنیم.ولی این باعث پیچیدگی زیاد می شود که مثلا آرگومان ورودی ما System.data.sqlclient باشد.راه حل بهتر استفاده از AppSetting در Web.Config است:

```
  <appSettings>
      <add key="factory" value="System.Data.SqlClient"/>
</appSettings>
```

حال می توانید این فضای نام را بازیابی کنیم و از آن به عنوان ورودی متد GetFactory استفاده کنیم:

```
   Dim factory As String = ConfigurationManager.AppSettings("factory")
 Dim pr As DbProviderFactory = DbProviderFactories.GetFactory(factory)
```

حال اگر AppSetting را به صورت زیر بنویسید می توانید همه نوع پایگاه داده را پشتیبانی کنید:

```
  <appSettings>
      <add key="factory0" value="System.Data.SqlClient"/>
```

```
        <add key="factory1" value="System.Data.OracleClient"/>
        <add key="factory2" value="System.Data.Oledb"/>
        <add key="factory3" value="System.Data.Odbc"/>
</appSettings>
```

برای استفاده از هر نوع پایگاه داده کافی است نام (Key) آن را بازیابی کنید.

حال این متغیر Pr خودش متد های مختلفی دارد.که در جدول زیر با موارد قبلی مقایسه شده است:

نوع شی	کلاس جدید	کلاس مخصوص	متد
Connection	DbConnection	SqlConnection	CreateConnection()
Command	DbCommand	SqlCommand	CreateCommand()
Parameter	DbDataParameter	SqlParameter	CreateParameter()
DataReader	DbDataReader	SqlDataReader	CreateDataReader()
DataAdapter	DbDataAdapter	SqlDataAdapter	CreateDataAdapter()

آنچه از این جدول بر می اید :مثلا اگر نوع شی ما Connection باشد به جای اینکه متغیر ها را از نوع SqlConnection تعریف کنیم آنها را از نوع DbConnection تعریف می کنیم و متد ایجاد آن CreateConnection() است.

مثلا در زیر من یک Connection ایجاد کردم :

```
  Dim con As DbConnection = pr.CreateConnection
con.ConnectionString =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
```

که قبلا این کار را می کردیم:

```
  Dim cs As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
  Dim con As New SqlConnection(cs)
```

ولی قبلا تنها برای Sql کار می کرد ولی حالا برای همه نوع پایگاه داده کار میکند.

من Query خود را نیزبه همراه فضای نام مربوط به Factory در AppSetting نوشتم تا در اینجا از آن استفاده کنم:

```
  <appSettings>
      <add key="factory" value="System.Data.SqlClient"/>
      <add key="customerQuery" value="SELECT * FROM customer"/>
</appSettings>
```

حال برای تعریف Command مثل Connection عمل می کنیم:

```
  Dim cmd As DbCommand = pr.CreateCommand()
cmd.CommandText = ConfigurationManager.AppSettings("customerQuery")
```

که در خط دوم آن را از WebConfig بازیابی کردیم.

پس تابع ما به صورت زیر است:

```
   Public Function agnostic() As DbDataReader
     Dim factory As String = ConfigurationManager.AppSettings("factory")
```

```
        Dim pr As DbProviderFactory = DbProviderFactories.GetFactory(factory)
        Dim con As DbConnection = pr.CreateConnection
        con.ConnectionString                                                    =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim cmd As DbCommand = pr.CreateCommand()
        cmd.CommandText = ConfigurationManager.AppSettings("customerQuery")
        cmd.Connection = con
        con.Open()
        Dim reader As DbDataReader = cmd.ExecuteReader()
        Return reader
        con.Close()
    End Function
```

دقت کنید که مقدار برگشتی ما هم نباید مثل قبل از نوع SqlDataReader باشد و آن را از نوع DbDataReader تعریف می کنیم.

اگر می خواهید یک Component از آن بسازید قبل از هر چیز باید موارد لازم را در AppSetting فایل WebConfig تعریف کنید.مثل نام Factory ها و Query ها وحتی رشته ی اتصال.سپس می تونید کلاس آن را به طور مجزا به صورت زیر بنویسید:

```
Imports Microsoft.VisualBasic
Imports System.Data.Common
Namespace comp
    Public Class agnostic
        Private fact_name As String
        Private fact_query As String
        Private fact_con As String
        Private factory As String
        Private pr As DbProviderFactory
        Private con As DbConnection
        Private cmd As DbCommand
        Private reader As DbDataReader
        Public Property AppSettings_factoryname() As String
            Get
                Return fact_name
            End Get
            Set(ByVal value As String)
                fact_name = value
            End Set
        End Property
        Public Property connectionstring_name() As String
            Get
                Return fact_con
            End Get
            Set(ByVal value As String)
                fact_con = value
            End Set
```

```
            End Property
            Public Property AppSettings_queryname() As String
                Get
                    Return fact_query
                End Get
                Set(ByVal value As String)
                    fact_query = value
                End Set
            End Property
            Public Function agnostic() As DbDataReader
                factory                                                    =
ConfigurationManager.AppSettings(AppSettings_factoryname)
                pr = DbProviderFactories.GetFactory(factory)
                con = pr.CreateConnection
                con.ConnectionString                                       =
ConfigurationManager.ConnectionStrings(connectionstring_name).ConnectionStrin
g
                cmd = pr.CreateCommand()
                cmd.CommandText                                            =
ConfigurationManager.AppSettings(AppSettings_queryname)
                cmd.Connection = con
                con.Open()
                reader = cmd.ExecuteReader()
                Return reader
                con.Close()
            End Function
        End Class
End Namespace
```

و برای استفاده کد زیر را بنویسید:

```
  Imports System.Data.Common
  Imports comp.agnostic
  Partial Class _Default
      Inherits System.Web.UI.Page

      Protected   Sub   Page_Load(ByVal   sender   As   Object,   ByVal   e   As
System.EventArgs) Handles Me.Load
          Dim reader As DbDataReader
          Dim a As New comp.agnostic
          a.AppSettings_factoryname = "factory"
          a.AppSettings_queryname = "customerquery"
          a.connectionstring_name = "aaa"
          reader = a.agnostic()
          While reader.Read
              Response.Write(reader(0) & "||")
          End While
      End Sub
  End Class
```

برای آب و رنگ دادن به این کلاس می توانید آن را به صورت کنترل های سفارشی به فایل Assembly تبدیل کرده و کامپایل کنید و Dll آن را به برنامه اضافه کنید و سپس از آن استفاده کنید.

حالا که شما به واسطه ی کلاس dbproviderFactory با Component و کلاس ها آشنا شدید بهتر است بحث Component در اتصال و مدیریت پایگاه داده را بیشتر پی بگیریم .ادامه ی این موضوع را با عنوان Data Components پی می گیریم.

Data Components

هرچه بیشتر به جای برنامه نویسی در هم و استفاده ی نابجا از توابع فاصله گرفته و به برنامه نویسی شی گرا روی آورید در همه حال به نفعتان است.در ادامه می خواهیم یک کلاس تمام عیار ایجاد کنیم که از نظر ما همه ی کارهای لازم روی یک جدول در پایگاه داده را انجام دهد و آن را تبدیل به یک Component کنیم تا همه جا بتوان از آن استفاده کرد.کلاسی که با یک StoredProcedure راه اندازی می شودو اعمالی همچون حذف-درج-تغییر-بازیابی(تعداد-تکی و...)-و ... را باآن انجام دهیم ولی نه به صورت هردمبیلی بلکه منظم وقاعده مند.البته کار با پایگاه داده چیزی فراتر ازاین ها است ولی آموختن همین ها هم سر جایش هنر است.

قبل از آن فرض کنید که یک Table در پایگاه داده ی خود به نام Employee با ستون های Emp_ID-Emp_firstname-Emp_lastname-Emp_job تعریف کنید و Emp_ID را کلید اصلی در نظر بگیرید .همه ی این صفات خاصه از نوع رشته ای هستند.تعریف این جدول به صورت زیر است:

```
CREATE TABLE employees
(emp_ID NVarChar(70),
emp_firstame VarChar(30),
emp_lastame VarChar(30),
emp_job VarChar(30)
Primary Key (emp_ID)
)
```

قبل از هر چیز برای مرتب بودن هر چه بهتر کار,نیاز به ایجاد یک کلاس به نام EmpDetails داریم که جزییات هر کارمند در آن ثبت شود.پس یک کلاس ایجاد کرده و نامش را EmpDetails می گذاریم:

```
Imports Microsoft.VisualBasic

Public Class EmpDetails
```

```
End Class
```

سپس برای هر یک از اعضای جدول پایگاه داده یک متغیر خصوصی تعریف می کنیم:

```
    Private m_Employee_ID As String
    Private m_FirstName As String
    Private m_LastName As String
  Private m_job As String
```

حال برای تک تک آنها نیاز به تعریف خاصیت داریم تا در بیرون کلاس بتوان از آنها استفاده کرد:

```
    Public Property Emp_ID() As String
        Get
            Return m_Employee_ID
        End Get
        Set(ByVal value As String)
            m_Employee_ID = value
        End Set
    End Property
    Public Property Emp_FirstName() As String
        Get
            Return m_FirstName
        End Get
        Set(ByVal value As String)
            m_FirstName = value
        End Set
    End Property
    Public Property Emp_LastName() As String
        Get
            Return m_LastName
        End Get
        Set(ByVal value As String)
            m_LastName = value
        End Set
    End Property
    Public Property Emp_job() As String
        Get
            Return m_job
        End Get
        Set(ByVal value As String)
            m_job = value
        End Set
  End Property
```

پس از تعریف خواص و مقادیر Get و Set آنها که به تر تیب مقدار برگشتی و دریافتی می باشد , باید تابع سازنده ای برای آن تعریف کنید.این تابع باید به گونه ای باشد که

بتواند مقادیر Id , نام ,نام خانوادگی و شغل کارمند را به ترتیب از ورودی در یافت کند و سپس این مقادیر در یافتی را به متغیر های خصوصی نظیرشان نسبت بدهیم که به این عمل در اصطلاح برنامه نویسی شی گرا گرانبار کردن می گویند:

```
  Public Sub New(ByVal Employee_ID As Integer, ByVal Employee_FirstName As
String, ByVal Employee_LastName As String, ByVal Employee_job As String)
          Me.m_Employee_ID = CStr(Employee_ID)
          Me.m_FirstName = Employee_FirstName
          Me.m_LastName = Employee_LastName
          Me.m_job = Employee_job
End Sub
```

کلمه ی کلیدی Me به معنای Root یا ریشه ی کلاس ماست.دقت کنید ما یک شماره از نوع عددی برای Employee_ID از کاربر می گیریم و سپس آن را به String تبدیل کرده و به نظیرش نسبت می دهیم دلیل این کار را در ادامه توضیح می دهیم.

نمونه ای از تعریف کارمند را در زیر می بینید:

```
 Dim emp1 As New EmpDetails(3, "hasan", "rajabi", "negahban")
```

حال به تعریف کلاس اصلی می پردازیم.نام آن EmpDB است و Import های لازم آن به همراه تعریف کلی آن را در زیر می بینید:

```
  Imports Microsoft.VisualBasic
  Imports EmpDetails
  Imports System.Data
  Imports System.Data.SqlClient
  Imports System.Web.Configuration

  Public Class EmpDB

End Class
```

همانطور که دیدید ما کلاس EmpDetails را نیز Import کردیم.جلوتر دلیل آن را می فهمید.قبل از هر چیز بیایید تابع سازنده را بنویسیم.ولی این تابع چه وظیفه ای خواهد داشت؟بهترین کاری که این تابع می تواند انجام دهد تعریف رشته ی اتصال در آن است.برای این کار ما متغیر خصوصی از نوع String به نام Cs تعریف می کنیم:

```
    Private cs As String
```

سپس در تابع سازنده آن را مقدار دهی می کنیم:

```
      Public Sub New()
          cs                                                    =
WebConfigurationManager.ConnectionStrings("aaa").ConnectionString
    End Sub
```

همانطور که می بینید چون در کار هایی که ما در این Component انجام می دهیم, نظیر Insert,Get,Update,Delete همگی مستلزم تعریف رشته ی اتصال هستند.پس ما وقتی تابع سازنده را صدا می زنیم در حقیقت رشه ی اتصال را تعریف کرده ایم تا یک بار تعریف برای چندین بار استفده باشد.حال از متغیر Cs در هر جا می توان استفاده کرد.شما می توانید چندین تابع سازنده داشته باشید به شرط آنکه آرگومان های ورودی آنها مثل هم نباشد به عنوان مثال در مثال خودمان , اگر شما خواستید یک Connection با یک نام دیگر را انتخاب کنید (یعنی صرفا نامش aaa نباشد) آنگاه شما می توانید این متغیر رشته ای را به عنوان آرگومان ورودی تابع سازنده ی خود داشته باشید:

```
Public Sub New(ByVal ConnectionsStringName As String)
    cs                                                              =
WebConfigurationManager.ConnectionStrings(ConnectionsStringName).ConnectionSt
ring
End Sub
```

پس دیدید که دو تابع سازنده داریم.حال از این بحث خارج می شویم و به بحث Stored Procedure می پردازیم چون تا آن نباشد نمی توان توابع Insert و ... را ایجاد کرد.ابتدا Stored Procedure را برای تابع Insert می نویسیم:

```
CREATE PROCEDURE InsertEmployee
@emp_fname varchar(30),
@emp_lname  varchar(30),
@emp_job     varchar(30),
@emp_ID Nvarchar(70)
AS
INSERT INTO employees
VALUES (@emp_ID, @emp_fname, @emp_lname, @emp_job );
```

که ٤ ورودی داشتیم و عمل Insert نیز کار این Stored Procedure است.

حال تابع Insert را در کلاس EmpDB تشریح می کنیم.تابع را به صورت زیر تعریف می کنیم:

```
Public Function InsertEmployee(ByVal emp As EmpDetails) As String

End Function
```

دقت کنید که ورودی آن به جای شلوغ بازی که ٤ آرگومان باشد یک متغیر از نوع EmpDetails است که شما برای استفاده ابتدا باید یک متغیر از نوع EmpDetails را New کنید و سپس آن را به عنوان ورودی این تابع استفاده کنید.حالا در ابتدا Connection را بر قرار می کنیم:

```
Dim con As New SqlConnection(cs)
```

سپس Command را تعریف می کنیم به این صورت که چون Sql-Command ما از نوع Stored Procedure است, نام Command همان نام Stored Procedure است و نوع آن هم که Stored Procedure:

```
Dim cmd As New SqlCommand("InsertEmployee", con)
cmd.CommandType = CommandType.StoredProcedure
```

حالا نوبت به تعریف پارامتر ها می رسد.ما به جای اضافه کاری در آن ,به ساده ترین شکل آن را نوشتیم:

```
cmd.Parameters.Add("@emp_fname", SqlDbType.VarChar, 30).Value = emp.Emp_FirstName
cmd.Parameters.Add("@emp_lname", SqlDbType.VarChar, 30).Value = emp.Emp_LastName
cmd.Parameters.Add("@emp_job", SqlDbType.VarChar, 30).Value = emp.Emp_job
```

می بینید که مثلا به جای نام از خاصیت Emp_FirstNameمتغیر emp استفاده کردیم.و به همین ترتیب برای نام خانوادگی و شغل.حتما می پرسید پس Emp_Id چه شد؟این صفت خاصه همان کلید اصلی است و باید Unique باشد.و در هیچ Database این گونه نیست که این شماره را از کاربر بپرسند.شما امروزه در اکثر سایت ها وقتی ثبت نام می کنید یک شناسه ی Unique برای شما ثبت می شود که با آن برای مدیر سایت شناخته می شوید.در اینجا هم می خواهیم این شناسه را خودمان به همراه اجزای دیگری که خود کاربر وارد می کند وارد جدول کنیم.یعنی عملInsert به این گونه باشد که کاربر یک شماره –یک نام-یک نام خانوادگی-نام شغل-وارد کند و ما نیز یک مقدار شناسه ی Unique به آنها اضافه کنیم و در نهایت عمل Insert انجام پذیرد.ما Emp_Id را با آن شناسه +عددی که کاربر وارد کرد پر می کنیم.این شناسه با متد Guid(Global Unique Identifier) و ویژگی NewGuid قابل ایجاد است:

```
Guid.NewGuid()
```

پس پارامتر Emp_Id را به صورت زیر می نویسیم:

```
cmd.Parameters.Add("@emp_ID", SqlDbType.NVarChar, 70).Value = (CStr(emp.Emp_ID) & Guid.NewGuid.ToString).ToString
```

چون مقداری که کاربر وارد می کند Integer است, آن را به String تبدیل کردیم و با Guid (تبدیل شده به String) جمع کردیم و در نهایت برای محکم کاری کل آن را نیز ToString کردیم.حالا دیگر کار پارامترها تمام است و نوبت به باز شدن Connection می شود:

```
con.Open()
```

و اجرای آن:

```
cmd.ExecuteNonQuery()
```

سپس برای عملیات دیگر مثل بازیابی و یا Update کردن باید این شناسه به کاربر ارایه شود که این کار با Return کردن مقدار نهایی Emp_Id میسر است:

```
Return CStr(cmd.Parameters("@emp_ID").Value)
```

و در نهایت بستن connection:

```
con.Close()
```

جهت بهبود کار این عملیات را از آنجا که connection باز شد درون بلوک Try...End Try انجام می دهیم:

```
Try
    con.Open()
    cmd.ExecuteNonQuery()
    Return CStr(cmd.Parameters("@emp_ID").Value)
Catch err As SqlException
    Throw New ApplicationException("Data error.")
Finally
    con.Close()
End Try
```

همه چیز مثل قبل است به جز عبارت زیر:

```
Throw New ApplicationException("Data error.")
```

کلمه ی کلیدی Throw برای مدیریت استثنا ها به کار برده می شود.این کلمه باعث می شود وقتی برنامه در بلوک Try به مشکل بر خورد Error گرفته شود و از آن چشم پوشی نکند.این قطعه کد را در همه ی توابع این کلاس د رداخل بلوک Catch به کار می بریم.شاید از خود بپرسید چرا به جای این کار پیغام خطا را با برگشت دادن err.Message به کاربر نمی دهید.دلیلش این است که مقدار برگشتی آن از نوع EmpDetails است نه رشته ای.پس در کل Throw یک مکانیزم برای مدیریت استثنا ها است که در اینجا می تواند به ما کمک کند چون اگراز ذکر آن جلوگیری می کردیم انگاه با پیغام خطایی مبنی بر اینکه Function مقداربرگشتی در بدنه ی اصلی ندارد مواجه می شدیم پس بهتربن کار استفاده از مکانیزم Throw است.حال اگر مشکلی در بدنه ی Try رخ داد از برنامه خارج می شویم و با پیغام خطا روبرو می گردیم در بحث مقابله با خطا ها , راه هایی برای مدیریت این خطا ها وجود دارد که به آن ها اشاره می شود.

حال نوبت تابع دیگری به نام GetEmployee است.این تابع با گرفتن یک Emp_Id از کاربر نام-نام خانوادگی و شغل وی را به او ارایه می دهد.Stored Procedure آن به شکل زیر است:

```
CREATE PROCEDURE GetEmployee
@emp_ID     NVarchar(70)
AS
SELECT emp_ID,emp_firstname,emp_lastname,emp_job FROM employees
WHERE emp_ID = @emp_ID
```

حال به تابع آن بپردازیم.تابع را با یک آرگومان ورودی Id از نوع رشته ای و مقداری برگشتی آن از نوع EmpDetails می باشد:

```
Public Function GetEployee(ByVal EmployeeID As String) As EmpDetails

End Function
```

سپس مانند تابع قبلی که تعریف کردیم Connection و Command را تعریف می کنیم:

```
Dim con As New SqlConnection(cs)
Dim cmd As New SqlCommand("GetEmployee", con)
cmd.CommandType = CommandType.StoredProcedure
```

برای تعریف پارامتر نز مانند قبل عمل می کنیم یعنی مقدار ورودی تابع را به پارامتر نسبت می دهیم:

```
cmd.Parameters.Add("@emp_ID", SqlDbType.NVarChar, 70).Value = EmployeeID
```

حال Connection را باز می کنیم:

```
con.Open()
```

حالا به قسمت اصلی کار می رسیم.در اینجا ما قرار است با گرفتن Id از کاربر, نام و ... را به وی بدهیم ولی تنها مجاز به برگشت دادن یک مقدار هستیم.اگر بخواهیم یک Reader برگشت دهیم کار غلطی است زیرا با برگشت دادن ان نباید Connection را ببندیم تا بتوانیم در صفحه ی اصلی از Reader استفاده کنیم .که این کارهمانطور که گفتیم باعث کاهش سرعت سایت و ارتباط آن با پایگاه داده می شود.به همین دلیل به جای برگرداندن Reader یک متغیر از نوع کلاس EmpDetails بر می گردانیم.برای این کار ابتدا Command مربوطه را ExecuteReader می کنیم :

```
Dim reader As SqlDataReader = cmd.ExecuteReader()
```

سپس آن را می خوانیم:

```
reader.Read()
```

این کار را قبلا به دلیل ازدیاد رکورد ها ی موجود در Reader , با حلقه ی While انجام می دادیم ولی حالا که تنها یک رکورداست تنها یک بار عمل Read را انجام می دهیم.سپس مقادیر آن را در یک متغیر از نوع EmpDetails قرار می دهیم و آن را بر می گردانیم و در نهایتConnection را می بندیم:

```
   Dim        emp        As        New        EmpDetails(CStr(reader("emp_ID")),
CStr(reader("emp_firstname")),                    CStr(reader("emp_lastname")),
CStr(reader("emp_job")))
   Return emp
 con.Close()
```

حالا همه ی این کار ها را از بعد از باز شدن Connection درداخل بلوک Try...End Try قرار می دهیم:

```
            Try
                con.Open()
                Dim reader As SqlDataReader = cmd.ExecuteReader()
                reader.Read()
                Dim     emp     As     New     EmpDetails(CStr(reader("emp_ID")),
CStr(reader("emp_firstname")),                    CStr(reader("emp_lastname")),
CStr(reader("emp_job")))
                Return emp
            Catch err As SqlException
                Throw New ApplicationException(err.Message)
            Finally
                con.Close()
        End Try
```

حال نوبت تابع UpdateEmployee میرسد.Stored Procedure این تابع به صورت زیر است.توجه کنید که الگوریتم آن به این صورت است که Id از کاربر در یافت می شود و به همراه آن نام-نام خانوادگی و شغل وی نیز در یافت می شود اگر چنین Id در جدول باشد آنگاه نام-نام خانوادگی و شغلی که کاربر , جدید وارد کرده است به جای قبلی ها قرار می گیرد:

```
CREATE PROCEDURE UpdateEmployee
@empID nvarchar(70),
@emp_fname varchar(30),
@emp_lname varchar(30),
@emp_jobs varchar(30)
AS
Update employees
Set emp_firstname=@emp_fname,emp_lastname=@emp_lname,emp_job=@emp_jobs
Where emp_ID=@empID
```

یک نکته ی امنیتی این است که اگر طرف هر Id وارد کن دمی تواند یک رکورد را به دلخواه خویش Update کند آن وقت این یک مشکل است.ولی باید گفت این Id یک رشته است مثل زیر:

```
fe6e2bec-c426-4b5d-9446-b0cb84cf4539
```

که تقلب از آن بعید به نظر می رسد ولی شما جهت امنیت بیشتر عبارت جلوی Where را به جای تطبیق با تنها Id ,آن را با عنصر دیگری تطبیق دهید که خودتان آن را به عنوان یک ستون به جدول پایگاه اضافه کردید و قابل تغیر نیست مثل password:

```
Where emp_ID=@emp_ID And emp_password=@pass
```

به این صورت دیگر امکان هک شدن اطلاعات شما بسیار کم می شود.

حال به تابع آن بپردازیم ک بسیار ساده است.ابتدا آن را تعریف می کنیم و Connection وCommand را تعریف می کنیم:

```
    Public Sub UpdateEmployee(ByVal EmployeeId As String, ByVal
EmployeeFirstName As String, ByVal EmployeeLastName As String, ByVal
EmployeeJob As String)
        Dim con As New SqlConnection(cs)
        Dim cmd As New SqlCommand("UpdateEmployee", con)
        cmd.CommandType = CommandType.StoredProcedure

    End Sub
```

جهت امنیت بالا ورودی تابع UpdateEmployee را به جای متغیری از نوع EmpDetails مقادیر را جدا جدا به عنوان آرگومان ورودی تابع دادیم.پارامتر ها نیز به صورت زیر تعریف می شود:

```
        cmd.Parameters.Add("@empID", SqlDbType.NVarChar, 70).Value =
EmployeeId
        cmd.Parameters.Add("@emp_fname", SqlDbType.VarChar, 30).Value =
EmployeeFirstName
        cmd.Parameters.Add("@emp_lname", SqlDbType.VarChar, 30).Value =
EmployeeLastName
      cmd.Parameters.Add("@emp_jobs", SqlDbType.VarChar, 30).Value =
EmployeeJob
```

و بقیه ی کار که در داخل بلوک Try...End Try قرار گرفته است:

```
        Try
            con.Open()
            cmd.ExecuteNonQuery()
        Catch err As SqlException
            Throw New ApplicationException("Data error.")
        Finally
            con.Close()
```

```
        End Try
```

Stored Procedure تابع DeleteEmpoyee نیز به صورت زیر می نویسیم:

```
  CREATE PROCEDURE DeleteEmployee
  @e_ID NVarChar(70)
  AS
DELETE From employees Where emp_ID=@e_ID
```

و تابع ان را به گونه ای بسیارساده به صورت زیر می نویسیم:

```
      Public Sub DeleteEmployee(ByVal EmployeeID As String)
          Dim con As New SqlConnection(cs)
          Dim cmd As New SqlCommand("DeleteEmployee", con)
          cmd.CommandType = CommandType.StoredProcedure

          cmd.Parameters.Add("@e_ID",    SqlDbType.NVarChar,    70).Value    =
EmployeeID
          Try
              con.Open()
              cmd.ExecuteNonQuery()
          Catch err As SqlException
              Throw New ApplicationException("Data error.")
          Finally
              con.Close()
          End Try

    End Sub
```

Stored Procedureتابع CountEmployees که تعداد کارمندان را برمی گردان به صورت زیر می نویسیم:

```
  CREATE PROCEDURE CountEmployees
  AS
Select Count(emp_ID) From employees
```

و تابع ان را به صورت زیر می نویسیم که هیچ آرگومان ورودی ندارد و مقدار برگشتی آن تعدا کارمندان است:

```
      Public Function CountEmployees() As Integer
          Dim con As New SqlConnection(cs)
          Dim cmd As New SqlCommand("CountEmployees", con)
          cmd.CommandType = CommandType.StoredProcedure

    End Function
```

سپس Connection را open می کنیم وآن را اجرا می کنیم ولی از آنجایی که مقدار برگشتی یک عدد(مقدار-Object) است, آن را با ExecuteScalar اجرا می کنیم و حاصل را درون یک متغیر ریخته وان را بر می گردانیم:

```
          Try
```

```
            con.Open()
            Dim a As Integer
            a = cmd.ExecuteScalar()
            Return a
        Catch err As SqlException
            Throw New ApplicationException("Data error.")
        Finally
            con.Close()
    End Try
```

خوب کار ما به پایان رسید.می توان صد ها تابع دیگر با Stored Prcedure های مختلف را به این کلاس اضافه کرد(بسته به نیاز) که ما به همین جا بسنده می کنیم.نمونه ای از Test توابع این کلاس را در زیر می بینید:

```
        Dim f As New EmpDB
        Response.Write(f.CountEmployees())
        Dim u As EmpDetails
        u = f.GetEployee("fe6e2bec-c426-4b5d-9446-b0cb84cf4539")
    Response.Write(u.Emp_FirstName & " " & u.Emp_LastName & " " &
u.Emp_job)
```

DataSet Object

همانطور که قبلا گفته شد شی DataSet شی است که می تواند اطلاعات درون جداول پایگاه اده را درون خود نگه داری کند حتی وقتی که Connection بسته باشد.این ویژگی باعث شده که از آن استقبال زیادی به عمل آید.این شی کاربرد های زیادی نظیر به اشتراک گذاشتن داده ها – پیمایش عقب و جلو بین حجم انبوه داده ها و ... را دارد.پس شما می توانید اطلاعات یک جدول پایگاه داده را استخراج کرده و درون DataSet قرار دهید و Connection را ببندید و سپس روی آن داده ها در DataSet تغییراتی را ایجاد کنید و پس از اتمام تغییرات , آن را به پایگاه داده اعمال کنید.باز هم می گوییم همه ی این کار ها در Connection بسته انجام می شود و همانطور که قبلا گفتیم, هر چه زمان اتصال به پایگاه داده کمتر باشد کارایی برنامه بالاتر می رود.همچنین DataSet می تواند با Xml هم کار کند و در آن بخواند و بنویسد که در بحث مرتبط با Xml آن ها را فرا خواهید گرفت.برای ذخیره ی اطلاعات یک جدول از DataBase از زیر کلاس DataSet به نام DataTable استفاده می شود و هر شی DataTable

حاوی سطر ها و ستون هایی است که برای دسترسی به آنها به ترتیب از اشیا DataColumn و DataRow استفاده می شود.همه ی این کار هایی که تا اینجا گفته شد تا وقتی شی DataAdapter نباشد بی فایده است.همانطور که گفته شد DataAdapter

مدیریت اطلاعات استخراج شده از پایگاه داده را بر عهده دارد که یک وظیفه ی آن پر کردن DataSet اطلاعات استخراجی از پایگاه داده است.این شی در فضای نام System.Data.SqlClient.SqlDataAdapter قرار دارد.شی DataAdapter دارای ۲ متد مهم است:

Fill:وظیفه ی پر کردن DataSet را با اطلاعاتی که DataAdapter استخراج کرده را بر عهده دارد.

Update:وظیفه ی دسترسی و تغییرات داده ها در DataSet و اعمال این تغییرات به DataSource اصلی پایگاه داده را بر عهده دارد.

حال یک مثال ساده در زیر برای پر کردن DataSet را انجام می دهیم.ابتدا رشته ی اتصال و Connection را مشخص می کنیم:

```
   Dim              cs              As              String              =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
 Dim con As New SqlConnection(cs)
```

سپس Query String مر بوطه را می نویسیم:

```
Dim cmd As String = "SELECT * FROM loan"
```

حالا یک متغیر از نوع شی DataSet تعریف و آن را New می کنیم:

```
Dim ds As New DataSet
```

سپس نوبت می رسد به تعریف و New کردن متغیری از نوع SqlDataAdapter.این شی در اینجا دو آرگومان می پذیرد اولی Query String و دومی Connection مربوطه:

```
Dim adapter As New SqlDataAdapter(cmd, con)
```

در نهایت هم عملیات پر کردن شی DataSet را با متد Fill شی SqlDataAdapter:

```
Adapter.Fill(ds, "loan")
```

این کار با پذیرفتن دو آرگومان صورت می گیرد.اولی که نام شی DataSet است.دومی نام جدول است که نامی دلخواه می توانید انتخاب کنید.حتی می توانید عددی را به آن بدهید تا نام جدول اصلا یک عدد باشد.از آنجا که می توان چندین جدول در شی DataSet قرار داد ,نام این جدول ها است که باعث می شود ما آنها را بتوانیم تفکیک کنیم.حالا تمام مقادیر محتوی جدول Loan در داخل شی DataSet قرار دارد.نکتهی قابل توجه در اینجا این است که ما نه Connection را باز کردیم ونه ان را بستیم.دلیل این کار وجود شی DataAdapter است زیرا در آرگومان دوم ورودی این شی, ما نام Connection را مشخص کردیم و این امر باعث شد که خود به خود Connection باز و بسته شود.پس

هر گاه از شی DataAdapter استفاده کردید نیازی به باز و بستن Connection ندارید چون DataAdapter به طور اتوماتیک و به بهترین نحو این کار را انجام می دهد.نکته ی جالب دیگراین است که ما تمام اطلاعات درون جدولی از پایگاه داده را داریم در حالی که Connection بسته است.و نکته ی دیگر این است که متد Fill از شی DataAdapter همان کارِExecute کردن Query ها را نیز انجام می دهد.پس اصلا نیازی به شی Command نیست. نکته ی دیگر در باره ی متد Fill از شی DataAdapter این است که با صدا زدن این متد علاوه بر اجرای Query , Connection هم بسته می شود.

در نهایت هم نمایش داده ها با کنترل GridView.برای این کار شما باید مقداری را برای خاصیت DataSource از کنترل GridView در نظر بگیرید.قبل از آن بیایید با خاصیتی از شی DataSet به نام Table استفاده کنیم.از آنجا که شی DataSet , می تواند حاوی چندین جدول هم باشد برای مشخص کردن یک جدول خاص و نمایش آن بایداز این خصوصیت استفاده کرد.آرگومان ورودی آن هم نام جدول است.حالت کلی آن به صورت زیر است:

DataSetName.Tables(“Table Name”)

پس ما هم برای نمایش آن به صورت زیر عمل می کنیم و حتویات جدول Loanکه در شی DataSet قرار دارد ,درون GridView نمایش داده می شوند:

```
GridView1.DataSource = ds.Tables("loan")
GridView1.DataBind()
```

حال اگر دلتان نخواهد از کنترلی مثل GridView استفاده کنید می توانید از راهی دیگر برای نمایش داده ها استفاده کنید.مهمانطور که گفتیم برای دسترسی به سطر های یک DataSet از شی DataRow می توانید استفاده کنید.حالا می خواهیم در ادامه ی مثال بالا, مقادیر موجود در DataSet را با شی DataRow نمایش دهیم.از آنجا که بیش از یک سطر در جدول پایگاه داده ی ما که در اینجا Loan است موجود است بایداز یک حلقه ی For استفاده کرد.برای راحتی نیز از حلقه ی ForEach استفاده می کنیم.در ابتدا یک شی از نوع DataRow تعریف می کنیم:

```
Dim dr As DataRow
```

سپس حلقه رابه اینگونه می نویسیم که تا آنجا که درسطر های جدول Loanداخل DataSet سطر وجود دارد ادامه بده:

```
For Each dr In ds.Tables("loan").Rows

Next
```

در حلقه ی For Each توجه کنید که متغیر قبل و بعد از کلمه ی کلیدی In باید از جنس هم باشند.در اینجا چون ما Dr را از نوع DataRow تعریف کردیم و گفتیم تا آنجا که درسطر های جدول Loanداخل DataSet سطر وجود دارد این را مشخص کردیم که در ادامه ی ds.Tables("loan") باید خصوصیت Rows را نیز اضافه کنیم.حالا در هر بار گردش دراین حلقه,تمام داده های هر سطر دست ماست (درون متغیر dr)وبرای نمایش آنها کافیست از این متغیر استفاده کنیم و آرگومان ورودیش را نام ستونهایی که می خواهیم نمایش دهیم بگذاریم:

```
For Each dr As DataRow In ds.Tables("loan").Rows
    Response.Write(dr("loan_number") & "<br>")
    Response.Write(dr("branch_name") & "<br>")
    Response.Write(dr("amount") & "<br>")
    Response.Write("-------------" & "<br>")
Next
```

با حلقه ی زیر هم اسامی جداول موجود در Dataset نمایش داه می شوند:

```
For Each dr As DataTable In ds.Tables
    Response.Write(dr.TableName & "<br>")
Next
```

همانطور که می بینید, برای دسترسی به نام جداول متغیری از نوع DataTable تعریف کردیم و دیگر نباید نام جدول را به ds.Tables دهیم چون اصلا ما نام جداول را می خواهیم.وقتی بایدنام بدهیم که با اجزای جدول خاصی کار داشته باشیم.

همچنین می توانید دادهای خاص را از Dataset استخراج کنید.مثلا داده ای که در سطر ۲و ستون ۲ از آن قرار دارد:

```
Response.Write(ds.Tables("loan").Rows(2)(2).ToString)
```

می بینید که خاصیت Row از متد Tables از شی DataSet , می تواند DataSet را یک ارایه ی ۲ بعدی فرض کند وداده ها را از آن بر این اساس استخراج کند.این خاصیت انقدر انعطاف پذیر است که حتی می توانیدبه جای عدد۲ در آن از نام ستون نیز استفاده کنید:

```
Response.Write(ds.Tables("loan").Rows(2)("branch_name").ToString &
"<br>")
```

که در اینجا مقدار سطر دوم از ستون Branch_name نمایش داده می شود.برای قرار دادن یک جدول دیگر در DataSet می توانید از خاصیت SelectCommand شی

DataAdapter استفاده کنید به این صورت که یک Query String جدید به آن نسبت داده و آنرا در DataSet قرار دهید.در زیر ما یک Query String دیگر به شی DataAdapter نسبت داده ایم که تمامی عناصر یکجدول دیگر است:

```
adapter.SelectCommand.CommandText = "SELECT * FROM depositor"
```

حالا برای Add کردن آن به DataSet کافی است مثل قبل عمل کنیم با این تفاوت که نامی دیگر به آن بدهیم:

```
Adapter.Fill(ds, "depositor")
```

حالا شی DataSet حاوی دو جدول است که به تفکیک نام,قابل دسترسی هستند.در زیر تابعی را می بینیدکه دو جدول را در DataSet قرار داده و شی DataSet حاصل را بر می گرداند:

```
Public Function aaa() As DataSet
        Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(cs)
        Dim cmd1 As String = "SELECT * FROM branch"
        Dim cmd2 As String = "SELECT * FROM account"
        Dim ad As New SqlDataAdapter(cmd1, con)
        Dim ds As New DataSet
        Try
            con.Open()
            ad.Fill(ds, 0)
            ad.SelectCommand.CommandText = cmd2
            ad.Fill(ds, 1)
        Catch ex As Exception
            Response.Write(ex.Message)
        Finally
            con.Close()
        End Try
        Return ds
    End Function
```

چون از Try..End Try استفاده کردیم برای کارایی بیشتر با اینکه شی DataAdapter وجود دارد ما Connection را باز و بسته کردیم تا فرمت Try..End Try هم به هم نخورد.

برای استفاده از این تابع به صورت اتوماتیک عمل می کنیم.ابتدا DataSet برگشتی را بازیابی می کنیم:

```
Dim k As New DataSet
k = aaa()
```

سپس از ٤ حلقه ی For استفاده می کنیم بیرونی ترین حلقه برای دسترسی به جدول ها و دومین حلقه برای دسترسی به سطر های آن جدول ها است. حلقه ی داخلی هم برای پیمایش مقادیر جدول ها به صورت یک ارایه ی دو بعدی هستند:

```
Dim i As Integer = 0
While i < k.Tables.Count
    For Each dr As DataRow In k.Tables(i).Rows
        For o As Integer = 0 To dr.Table.Rows.Count - 1
            For t As Integer = 0 To dr.Table.Columns.Count - 1
                Response.Write(dr.Table.Rows(o)(t).ToString & " ")
            Next
            Response.Write("<br>")
        Next
    Next
    i += 1
End While
```

شمارنده ی I برای کنترل حلقه ی While است.از ۰ شروع می شود و تا i < k.Tables.Count ادامه دارد.k.Tables.Count همانطور که از نامش برمی آید تعداد جدول های موجود در DataSet را برمی گرداند.در داخل تابع نیز نام جدول ها را از عمد عدد دادیم که بتوانیم در حلقه از ان ها استفاده کنیم.دو حلقه ی داخلی هم که اولی سطر و دومی ستون را کنترل می کند و اولی تا dr.Table.Rows.Count – 1 و دومی تا dr.Table.Columns.Count – 1 ادامه می یابد.از طرف دیگر چون دو حلقه سط و ستون را کنترل می کنند نیازی به به کار بردن چند Response.write نیست و تنها یکی کافی است چون به تعداد عناصر موجود در جدول چه از نظر سطر و چه از نظر ستون می چرخد.

از رایطه در Sql چه قدر می دانید؟حتما می دانید که اگر Relation نباشد اصلا پایگاه داده بی معنی است.Ado.NET به طور پیش فرض Relation را Support نمی کند.و برای نمایش رابطه در جداول باید دستی کار کنید.دو جدول زیر را در نظر بگیرید:

جدولAccount	جدول Branch

branch_name	branch_city	branch_assets
emam	zanjan	210000000
noor	qazvin	260000000
payam	tehran	300000000
razi	karaj	400000000

account_number	branch_name	balance
1	emam	337080
2	noor	292136
3	payam	280900
4	emam	382024
5	emam	348316
6	razi	213484
7	razi	189210
8	payam	235956
9	noor	325844

این دو جدول به وسیله ی دو صفت خاصه ی Branch_name به هم متصل شده اند.که Branch_name در جدول Branch کلید اصلی و Branch_name در جدول Account کلید خارجی است.به این صورت که نتیجه ی اتصال آن ها برای شعبه ی emam به صورت زیر باید باشد:

account_number	branch_name	branch_assets	balance
1	emam	210000000	337080
4	emam	210000000	382024
5	emam	210000000	348316

این نتیجه ی اتصال دو جدول است. این اتصل به معنای این است که دو جدول به جای مجزا بودن, بر اساس وجوهات مشترکشان نمایش داده شوند.در جداول بالا این دو جدول وجوه مشترکی به نام Branch_name دارند که می تواند باعث اتصال دو جدول به گونه ای کاملا بهینه و نرمال باشد.

.ین دو جدول در اصل یکی بودند ولی به دلیل افزونگی نرمال و تجزیه شدند.Ado.NET این نتیجه را نشان نمی دهد و اگر آن را اجرا کنید دو جدول را به صورت مجزا نمایش می دهد.برای نمایش Relation از شی DataRelation استفاده می شود.اگر بخواهید مغیری از نوع آن ایجاد کرده و new کنید , ۳ آرگومان ورودی خواهد داشت.اولی یک نام ساده به رابطه است.دومی کلید اصلی و سومی کلید خارجی است:

```
Dim rel As New DataRelation("name","parent column primary key","child
column foeign key")
```

می دانیم کلید همان ستون است و اصلی و خارجی بودن کلید نیز دست ماست.پس برای مشخص کردن یک ستون خاص از خاصیت Column شی DataSet استفاده می کنیم:

```
dss.Tables("branch").Columns("branch_name")
```

در این کد dss نام DataSet است.نام جدول هم که Branch است و با خاصیت Column به ستون های آن دسترسی داریم که Branch_name آن ستون مد نظر ماست و کلیداصلی نیز هست.در زیر کد مربوط به کلید خارجی نیز مشخص شده است:

```
dss.Tables("account").Columns("branch_name")
```

حال کافی است متغیر از نوع DataRelation را تعریف کنیم:

```
Dim rel As New DataRelation("MyRelation",
dss.Tables("branch").Columns("branch_name"),
dss.Tables("account").Columns("branch_name"))
```

پس از تعریف آن باید آن را به شی DataSet Add کنیم:

```
dss.Relations.Add(rel)
```

میبینید که برای Add کردن رابطه , از متد Relation شی DataSet استفاده شده و سپس گزینه ی Add در آن انتخاب شده.این امر باعث می شود DataSet این رابطه را درک کند.البته اگر در اصل این رابطه در پایگاه داده وجود نداشته باشد با مشکل بر می خوریم.حال می رسیم به مرحله ی دشوار کار که دسترسی و نمایش رابطه است.برای این کار ابتدا یک حلقه ی ساده و تکراری For .Each می نویسیم برای پیمایش سطر های جدولی خاص.این جدول در اینجا همان جدولی است که حاوی کلید اصلی بوده:

```
For Each row As DataRow In dss.Tables("branch").Rows

Next
```

سپس ستون هایی از آن را که می خواهید چاپ شوند را Response.Write می کنید.این کار را درون حلقه ی For انجام می دهیم:

```
Response.Write(row("branch_name") & "<br>")
```

بعد از آن به این نکته دقت کنید که در اینجا جدولی که حاوی کلید اصلی است Parent و جدولی که حاوی کلید خارجی است Child نامیده می شود.پس برای پیمایش جدول Child باید اول از خاصیت GetChildRows شی DataRow استفاده کنیم و آن را در آرایه ای از نوع DataRow قرار دهیم:

```
Dim cr As DataRow() = row.GetChildRows(rel)
```

متغیر Row هم که در حلقه ی For تعریف شده بود.و ما از خاصیت GetChildRows آن استفاده کرده و آرگومان ورودی آن را که از نوع relation است را نام متغیر relation قرار دادیم(rel) .و آن را درون متغیری از نوع DataRow قرار دادیم البته از نوع ارایه ای چون GetChildRows چند سطر را برمی گرداند نه یک سطر.حالا این چند سطر را با حلقه ی For .Each نمایش می دهیم:

```
      For Each childRow As DataRow In cr
          Response.Write(childRow("account_number") & "<br>")
    Next
```

پس کد کلی به صورت زیر شد:

```
    Dim          rel          As          New          DataRelation("MyRelation",
dss.Tables("branch").Columns("branch_name"),
dss.Tables("account").Columns("branch_name"))
  dss.Relations.Add(rel)
    For Each row As DataRow In dss.Tables("branch").Rows
      Response.Write(row("branch_name") & "<br>")
      Dim cr As DataRow() = row.GetChildRows(rel)
      For Each childRow As DataRow In cr
        Response.Write(childRow("account_number") & "<br>")
      Next
  Next
```

در کل اگر کمی دقت کنید می بینید بسیار ساده است.دو خط اول که تعریف relation و Add کردن آن به DataSet بود.در حلقه ی For اول هم تمام سطر ها پیمایش می شد و در هر پیمایش نام ستون Branch_name آن سطر چاپ می شد.سپس متد GetChildRows که متدی جزو همان سطری است که در حلقه ی بالاییش پیمایش شده ,تمام سطرهایی را که با ان سطر مورد نظر رابطه دارند را به وسیله ی آرگومان ورودیش که همان رابطه بود پیدا می کند.سپس حلقه ی For داخلی هم آن ها را نمایش می دهد.در حقیقت با توجه به جداول بالا ابتدا emam چاپ شده سپس با متد GetChildRows تمام سطر هایی از جدول بعدی که با emam رابطه دارند (که سطر های ۱و٤و٥) هستند انتخاب شده و نمایش داده می شوند.پس به آرایه بودن متغیر cr از نوع DataRow دقت کنید چون یک سطر نیست بلکه چند سطر است.سوالی که پیش می آید این است که کاربرد نام رابطه که در اینجا MyRelation است کجاست؟ما یک نام از نوع String به ان دادیم و کاربردش در خود دستورات Sql است.جلوتر در قسمت DataViewطرز کار و کاربرد ان را می بینید. نتیجه ی یک رابطه ی معمولی را نمی توان به صورت عادی در GridView نمایش داد به همین دلیل از دو حلقه ی For استفاده کردیم که با کمی تمرین و تکرار ملکه ی ذهنتان می شود.

حتما تا اینجا با طرز کار WHERE در Sql آشنا شده اید.شما با متد Select از شی DataTable می توانید یک WHERE به Query خود اضافه کنید . البته نه در QuerySql بلکه در صفحه ی VB.NET .درست است که این متد عضوی از شی

DataTable است .ولی چون وظیفه اش انتخاب تعدادی سطر است مقدار باز گشتی ان از نوع DataRow می باشد.برای مثال فرض کنید که کد ما به صورت زیر باشد:

```
        Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(cs)
        Dim adp As New SqlDataAdapter("Select * From customer", con)
        Dim ds As New DataSet
        adp.Fill(ds, "a")
```

تا اینجا که مشکلی نیست.قبلا برای نمایش به صورت زیر عمل می کردید.

```
        For Each dr As DataRow In ds.Tables("a").Rows
            Response.Write(dr("customer_name") & "<br>")
            Response.Write(dr("customer_street") & "<br>")
            Response.Write(dr("customer_city") & "<br>")
            Response.Write("=----------------------=" & "<br>")
        Next
```

برای ایجاد متد Select کافیست در قسمت بعد از کلمه ی کلیدی In , به جای ds.Tables("a").Rows از ds.Tables("a").Select("customer_city='karaj'") استفاده کنید:

```
        For Each dr As DataRow In ds.Tables("a").Select("customer_city='karaj'")
            Response.Write(dr("customer_name") & "<br>")
            Response.Write(dr("customer_street") & "<br>")
            Response.Write(dr("customer_city") & "<br>")
            Response.Write("=----------------------=" & "<br>")
        Next
```

در حقیقت همان عبارتی که می خواستید در قسمت Where بنویسید را در داخل متد Select نوشتید.که در اینجا Query اصلی ما انتخاب کل جدول بود و با این متد تنها سطر هایی که نام شهرشان کرج بود را انتخاب و سپس نمایش دادیم.

این کار را می توانید به صورت زیر نیز انجام دهید:

```
        Dim mr As DataRow() = ds.Tables("a").Select("customer_city='karaj'")
        For Each dr As DataRow In mr
            Response.Write(dr("customer_name") & "<br>")
            Response.Write(dr("customer_street") & "<br>")
            Response.Write(dr("customer_city") & "<br>")
            Response.Write("=----------------------=" & "<br>")
        Next
```

در اینجا مثل مثال قبلی از یک ارایه از نوع DataRow استفاده کردیم و تمام سطر هایی که نام شهرشان کرج بود را انتخاب کردیم و سپس آن ها را نمایش دادیم.

ولی سعی کنید از متد Select استفاده نکنید.زیرا متدی است که اولا پارامتر ها را پشتیبانی نمی کند و دوما راه را برای حملات تزریق Sql باز می گذارد.

دلیلی ندارد که شما نتوانید در توابع و کلاس ها مقدار برگشتی از نوع DataSet و DataTable نداشته باشید.حتما میدانید برای برگشت داده های انبوه همیشه مقدار برگشتی شما از نوع DataReader بوده است ولی آیا این کار در کلاس ها و توابع هم کاربرد دارد؟اگر داده ی برگشتی ما از نوع DataReader باشد آن وقت تا وقتی DataReader قابل استفاده است که Connection باز باشد.آیا ما می توانیم اینقدر Connection را باز نگه داریم؟بدون شک نه.ما در مثالی که در زمینه ی Component زدیم و یک کلاس کامل ایجاد کردیم جای متدی به نام GetAllEmployee خالی بود.این متد می توانست به صورت زیر ایجاد گردد:

```
Public Function GetAllEmployees() As SqlDataReader
    Dim con As SqlConnection = New SqlConnection(connectionString)
    Dim cmd As SqlCommand = New SqlCommand("GetAllEmployees", con)
    cmd.CommandType = CommandType.StoredProcedure

    Try
        con.Open()
        Dim reader As SqlDataReader = cmd.ExecuteReader()

        Return reader
    Catch err As SqlException
        Throw New ApplicationException("Data error.")
    Finally

    End Try
End Function
```

این متد هیچ ایراد دستوری ندارد.و داده ی برگشتی از نوع SqlDataReader دارد که در بیرون کلاس می توان از آن استفاده کرد.در این کد ذر بلوک Finally جای Con.Close خالی است.زیرا اگر باشد دیگر داده ی SqlDataReader وقتی Connection بسته باشد معنی نخواهد داشت.از طرفی اگر Connection باز هم باشد آنقدرکارایی سیستم پایین می آیدکه حد ندارد.پس اینجاست که DataSet و DataTable به کار ما می آید.در کلاس زیر متد GetAllEmployee داده یبرگشتی از نوع DataSet دارد در حالیکه Connection بسته است(توسط شی DataAdapter بسته شده):

```
Imports Microsoft.VisualBasic
```

```
Imports System.Data
Imports System.Data.SqlClient
Public Class Class1
    Private cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString

    Public Sub New()
        cs = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
    End Sub

    Public Function GetAllEmployees() As DataTable
        Dim con As New SqlConnection(cs)
        Dim sql As String = "Select * from employees"
        Dim da As New SqlDataAdapter(sql, con)
        Dim ds As New DataSet()
        Try
            da.Fill(ds, "Employees")
            Return ds.Tables("Employees")
        Catch
            Throw New ApplicationException("Data error.")
        End Try
    End Function
End Class
```

و نحوه ی استفاده از این کلاس نیز بسیار ساده است:

```
        Dim j As New Class1
        Dim k As New DataTable
        k = j.GetAllEmployees()
        GridView1.DataSource = k
      GridView1.DataBind()
```

ایجاد تغییرات در DataSet:تا حالا هر چه گفتیم در مورد نمایش داده با DataSet بود.در حالی که شما می توانید در DataSet تغییرات ایجاد کنید و حتی این تغییرات رابه پایگاه داده اعمال کنید.حالا کمی در مورد تغییرات روی DataSet کار کنیم مثل Insert-Delete-Update و....

همه ی این تغییرات را در یک مثال انجام می دهیم.ابتدا کد زیر را بنویسید.jj نام جدول ماست که حاوی ۳ ستون است.۱-شناسه(کلید اصلی) ۲-نام ۳-نام خانوادگی:

```
        Dim cs As String
        Dim qs As String
        Dim con As SqlConnection
        Dim ds As DataSet
        Dim adap As SqlDataAdapter
        cs = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        con = New SqlConnection(cs)
```

```
        qs = "Select * From jj"
        adap = New SqlDataAdapter(qs, con)
        ds = New DataSet
        adap.Fill(ds, "jj")
        GridView1.DataSource = ds.Tables("jj")
    GridView1.DataBind()
```

برای اضافه کردن یک سطر جدید به جدول DataSet باید از خاصیت newRow شی DataTable استفاده کرد.برای این کار ابتدا متغیری از نوع DataRow و DataTable تعریف می کنیم:

```
        Dim NewRow As DataRow
    Dim table As DataTable
```

سپس جدول موجود در DataSet را به متغیری که از نوع DataTable تعریف کرده بودیم نسبت می دهیم:

```
    table = ds.Tables("jj")
```

سپس از خاصیت newRow شی DataTable استفاده می کنیم و آن را درون متغیری که از نوع DataRow ایجاد کرده بودیم می ریزیم:

```
NewRow = table.NewRow
```

یعنی در حقیقت با این کد یک سطر جدید وخالی برای جدول خود ایجاد کردیم.در نهایت هم مقادیر آن را تعیین می کنیم:

```
        NewRow.Item("e_id") = 11
        NewRow.Item("e_fname") = "ahmad"
    NewRow.Item("e_lname") = "rezaali"
```

در نهایت آن را به سطر های شی DataTable اضافه می کنیم:

```
    table.Rows.Add(NewRow)
```

این کار را یک سره و بدون متغیراز نوع DataTable هم می توان انجام داد:

```
    NewRow = ds.Tables("jj").NewRow
```

یک سطر دیگر هم اضافه می کنیم:

```
        NewRow1 = table.NewRow
        NewRow1.Item("e_id") = 12
        NewRow1.Item("e_fname") = "shahla"
        NewRow1.Item("e_lname") = "mosadegh"
    table.Rows.Add(NewRow1)
```

پس عملیات Insert به این گونه است که ابتدا Table را مشخص می کنیم سپس از متد newRow آن استفاده کرده و آن را به متغیری از نوع DataRow نسبت می دهیم.سپس آن متغیر را مقدار دهی کرده و به سطر های جدول Add می کنیم.برای نمایش هم مثل قبل به صورت زیر عمل می کنیم:

```
        GridView2.DataSource = ds.Tables("jj")
    GridView2.DataBind()
```

برای عملیات Update هم به ساگی با در دست داشتن اندیس سطر مورد نظر و مقداردهی جدید به سطر ان کار تمام است:

```
        Dim table As DataTable = ds.Tables("jj")
        Dim R As DataRow
        R = table.Rows(2)
        R.Item("e_fname") = "yadollah"
        R.Item("e_lname") = "akbari"
        GridView4.DataSource = ds.Tables("jj")
    GridView4.DataBind()
```

در اینجا سطر دوم جدول را انتخاب کرده و آن را به یک متغیر از نوع DataRow خالی نسبت می دهیم.سپس کافی است این متغیر را از نو مقدار دهی کنیم.این کار را نیز می توان یک سره انجام داد که در کش کمی مشکل تر است:

```
table.Rows(2).Item("e_id") = 3
```

برای Delete هم کافی است از خاصیت Delete متد Tables.Rows استفاده کنیم.در زیر ما سطر چهارم را پاک کردیم:

```
        table.Rows(3).Delete()
        GridView5.DataSource = ds.Tables("jj")
    GridView5.DataBind()
```

نکته ای که تا اینجا باید به خاطر بسپاری این است که اعمال بالا به صورت یک سره هم قابل انجام هستند همانطور که بیان شد و شما هم سعی کنید آن ها را یک سره انجام دهید.مثلا عمل Updateرا به صورت زیر انجام دهید:

```
        table.Rows(2).Item("e_fname") = "yadollah"
        table.Rows(2).Item("e_lname") = "akbari"
        GridView4.DataSource = ds.Tables("jj")
    GridView4.DataBind()
```

قبل از بحث چگونگی Update شدن DataBase بگذارید با شی CommandBuilder آشنا شویم.این شی همانطور که از نامش بر می آید سازنده ی دستور است و چون در بحث Ado.NET مطرح شده سازنده ی دستورات عمومی Sql است. دستوراتی مثل Select-Update-Delete و....در مثال زیر نمایش این دستورات را می بینیم:

ابتدا کد های عمومی کار:

```
        Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(cs)
        Dim qs As String = "SELECT * FROM jj"
```

```
Dim adp As New SqlDataAdapter(qs, con)
```

سپس شی از نوع SqlCommandBuilder تعریف می کنیم همانطور که می بینید new کردن آن مستلزم ورودی از نوع Dataadapter است تا هم به Query String آن دسترسی داشته باشد و هم به Connection آن تا بتواند دستورات را بسازد:

```
Dim cm As New SqlCommandBuilder(adp)
```

به متد های SelectCommand-DeleteCommand-InsertCommand-UpdateCommand شی SqlDataAdapter نیز باید توجه شود.

حالا برای تولید دستورا ت از متد Get شی SqlCommandBuilder استفاده می کنیم:

```
adp.InsertCommand = cm.GetInsertCommand
adp.DeleteCommand = cm.GetDeleteCommand
adp.UpdateCommand = cm.GetUpdateCommand
```

و برای نمایش به صورت زیر عمل می کنیم:

```
Response.Write(adp.InsertCommand.CommandText & "<br>")
Response.Write(adp.DeleteCommand.CommandText & "<br>")
Response.Write(adp.UpdateCommand.CommandText & "<br>")
```

این کد ها با توجه به Query String که مطرح شد دستورات مبنی بر Insert کردن و به روز یا حذف کردن جدول jj را تولید می کند زیرا وقتی شی SqlCommandBuilder تعریف شد و یک شی از نوعDataAdapter آرگومان ورودی تابع سازنده ی آن شد, این شی به جدول مربوطه در پایگاه داه دسترسی دارد(چون شی DataAdapter هنم به رشته ی اتصال و هم به QueryString ما دسترسی دارد) پس می تواند با توجه به ستون های ان و عوامل دیگر کد های مربوزه را تولید کند.

حتما از خود می پرسید چرا اول همه ی دستورات تولید شده را در متد های شی SqlDataAdapter قرار داده ایم وسپس آن ها را به وسیله ی متد های شی SqlDataAdapter چاپ کردیم.در حالیکه میشد یک راست به صورت زیر عمل کنیم:

```
Response.Write(cm.GetInsertCommand.CommandText & "<br>")
Response.Write(cm.GetDeleteCommand.CommandText & "<br>")
Response.Write(cm.GetUpdateCommand.CommandText & "<br>")
```

پاسخ این سوال کاملا واضح است.ما این دستورات را تولید می کنیم تا در شی SqlDataAdapter از آن ها استفاده کنیم مثلا هر وقت برنامه نیاز بهUpdate شدن داشت به متد UpdateCommand از شی SqlDataAdapter رجوع شود.حال وقتی به ان رجوع شود باید دستوری در آن باشد دیگر.پس نیاز به این داریم که ابتدا Command

های تولید شده را درون متد های شی SqlDataAdapter قرار دهیم و سپس آن ها را چاپ کنیم.

حالا می ماند Update کردن DataSet که تغییرات مختلفی را روی آن انجام دادیم.برای این کار از متد Update شی DataAdapter استفاده می کنیم که ۲ آرگومان ورودی دارد یکی نام DataSet (که مثلا تغییراتی مثل اضافه کردن سطر جدید-تغییرات در آن-حذف یک سطر رویش اعمال شده) و دیگری نام جدولی که در پایگاه داده است و این تغییرات باید رویش اعمال شود.دقت کنید این متد بدون وجود دستورات مربوطه این کار ها را نمی کند که ما این دستورات را با CommandBuilder تولید به متد های نظیرش در شی DataAdapter قرار دادیم.اگر بدون انها کد زیر را بنویسید هیچ تغییری در پایگاه داده رخ نمی دهد:

```
adap.Update(ds, "jj")
```

کد کامل تغییرات روی DataSet و اعمال آن به پایگاه داده ی اصلی را می بینید:

```
Imports System.Data
Imports System.Data.sqlclient
Partial Class Default4
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim cs As String
        Dim qs As String
        Dim con As SqlConnection
        Dim ds As DataSet
        Dim adap As SqlDataAdapter
        cs = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        con = New SqlConnection(cs)
        qs = "Select * From jj"
        adap = New SqlDataAdapter(qs, con)
        ds = New DataSet
        adap.Fill(ds, "jj")
        GridView1.DataSource = ds.Tables("jj")
        GridView1.DataBind()
        '//////////////////////
        '//////////////////////
        Dim NewRow As DataRow
        Dim NewRow1 As DataRow
        Dim table As DataTable
        table = ds.Tables("jj")
        NewRow = table.NewRow
        NewRow.Item("e_id") = 11
```

```
            NewRow.Item("e_fname") = "ahmad"
            NewRow.Item("e_lname") = "rezaali"
            NewRow1 = table.NewRow
            NewRow1.Item("e_id") = 12
            NewRow1.Item("e_fname") = "shahla"
            NewRow1.Item("e_lname") = "mosadegh"
            table.Rows.Add(NewRow1)
            table.Rows.Add(NewRow)
            GridView2.DataSource = ds.Tables("jj")
            GridView2.DataBind()
            '//////////////////////
            '//////////////////////
            table.Rows(2).Item("e_fname") = "yadollah"
            table.Rows(2).Item("e_lname") = "akbari"
            GridView4.DataSource = ds.Tables("jj")
            GridView4.DataBind()
            '//////////////////////
            '//////////////////////
            table.Rows(3).Delete()
            GridView5.DataSource = ds.Tables("jj")
            GridView5.DataBind()
            '//////////////////////
            '//////////////////////
            Dim cm As New SqlCommandBuilder(adap)
            adap.InsertCommand = cm.GetInsertCommand
            adap.DeleteCommand = cm.GetDeleteCommand
            adap.UpdateCommand = cm.GetUpdateCommand
            Response.Write(adap.InsertCommand.CommandText & "<br>")
            Response.Write(adap.DeleteCommand.CommandText & "<br>")
            Response.Write(adap.UpdateCommand.CommandText & "<br>")
            adap.Update(ds, "jj")
        End Sub
    End Class
```

همه ی تغییرات مرحله به مرحله هر یک در یک GridView نمایش داده شده اند.نکته هایی که می ماند این است که شی CommandBuilder مورد پسند برنامه نویسان نیست چون ان ها را به راحت طلبی وادار می کند همانطور که دیدید خودش کد تولید می کرد ولی کد های ساده! در کل برای سناریو های پیچیده نمی توان از CommandBuilder استفاده کرد و باید تنبلی را کنار گذاشت و خود دست به کار شد و کد هایSql نوشت.اگر دلتان نخواست که از CommandBuilder استفاده کنید کافی است جلوی خواصی همچون adap.InsertCommand خودتان دستی کد بنویسید:

```
adap.InsertCommand = "INSERT INTO jj VALUES(2,'hamed','haghani')"
```

درک کد های تولید شده توسط CommandBuilder نیز ساده است کافی است در مثال بالا که آن ها را نمایش دادیم, کمی به کد های تولید شده بنگرید.

در کل سعی کنید هنگامی که می خواهید تغییرات انبوهی در پایگاه داده بدهید از شی DataSet استفاده کنید ولی اگر تغییرات اندک هستند از همان روش های قبلی استفاده کنید.چون به هر حال DataSet هم کلاس پیچیده ای است چون مثلا می تواند یک دفعه محتویات ۱۰ جدول مختلف را در خود نگه داری کند.

DataView Class:

شی DataView در Ado.NET همان View در Sql Server است.شی است که می توان به وسیله ی آن یک دید ایجاد کرد.به عبارت دیگر اگر اطلاعات داخل یک DataTable را از یک صافی بگذرانید و یا بر اساس مقادیر مختلف مرتب کنید DataView ایجاد می شود.شی DataView در مبحث DataBindingنیز کاربرد دارد.اولین مبحثی که در مورد DataView پیش می آید مرتب کردن داده ها است.با یک مثال متد Sort از شی DataView را توضیح می دهیم.

در ابتدا یک سری کد عمومی:

```
        Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(cs)
        Dim ds As New DataSet
        Dim qs As String = "SELECT TOP 9 * FROM customer"
        Dim adp As New SqlDataAdapter(qs, con)
        adp.Fill(ds, "customer")
      GridView1.DataSource = ds.Tables("customer")
```

فقط در مورد QueryString یک نکته بگویم که Top 9 به معنی ۹ سطر اول از جدول است.بقیه ی موارد تکراری است.حالا یک متغیر از نوع DataView تعریف می کنیم(new می کنیم) و آرگومان ورودی آن را که از جنس Table است نام جدولی را می دهیم که قرار است ازش یک View ایجاد کنیم:

```
Dim dv1 As New DataView(ds.Tables("customer"))
```

نام متغیر ما dv1 و نام جدول را Customer که در DataSet قرار داشت پر کردیم گذاشتیم.حالا می خواهیماز متد Sort استفاده کنیم.این متد یک مقدار نیاز دارد که همان نام ستونی است که داده ها باید بر اساس آن مرتب شوند.مثلا یادتان هست که درSql مرتب

کردن با کلمه ی کلیدی ’column name‘ ORER BY بود .در اینجا به صورت زیر است:

```
dv1.Sort = "customer_name"
```

یعنی مرتب سازی Dataview بر اساس ستون Customer_name است.برای نمایش هم از GridView استفاده می کنیم:

```
GridView2.DataSource = dv1
```

خوب شاید از خود بپرسید که چرا از کلمه ی کلیدی ’column name‘ ORER BY در Query String استفاده نکردیم؟جواب این است که اگر تعداد این مرتب سازی ها بیش از یکی باشد در روش Query String باید چند Query String نوشت و تک تک را اجرا کرد در حالیکه در اینجا با یک خط کد این کار انجام می شود. مثلا در زیر مرتب سازی بر اساس شهر مشتری است:

```
  Dim dv2 As New DataView(ds.Tables("customer"))
  dv2.Sort = "customer_city"
GridView3.DataSource = dv2
```

در نهایت برای نمایش کار جدیدی می کنیم:

```
Me.DataBind()
```

به جای اینکه ۳ بار برای هر GridView یک بار DataBind را انجام دهیم یک بار این را انجام دادیم.Me همان ریشه ی صفحه است.

حتی عمل مرتب سازی می تواند بر اساس دو یا بیش از دو نام باشد:

```
dv2.Sort = "customer_name,customer_city"
```

دقت کنید که نام ستون ها در اینجا با کاما از هم جدا می شوند.

همچنین می توانیم مرتب سازی را صعودی و نزولی کنیم.اگر خواستیم صعودی باشد که مقدار پیش فرض است کلمه ی کلیدی ASC که مخفف Ascending است را در ادامه ی قسمتی که نام ستون مورد نظر ذکر شده قرار می دهیم و اگر نزولی باشد کلمه ی DESC که مخفف Descending است را به ان اضافه می کنیم:

```
مرتب سازی نزولی        dv1.Sort = "customer_name DESC"
مرتب سازی صعودی        dv1.Sort = "customer_name ASC"
```

حالا که کمی با DataView آشنا شدیم باید به سراغ متد بعدی آن یعنی RowFilter برویم.این متد به ما اجازه ی فیلتر کردن داده ها را می دهد.و نقش WHERE را بازی می کند.

در مثال زیر ابتدا یک سری کد تکراری می بینید:

```
        Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(cs)
        Dim ds As New DataSet
        Dim qs As String = "SELECT * FROM customer"
        Dim adp As New SqlDataAdapter(qs, con)
        adp.Fill(ds, "customer")
      Dim dv1 As New DataView(ds.Tables("customer"))
```

سپس از متد RowFilter آن استفاده کرده و تمام مشتریانی که ساکن کرج هستند را بدست می آوریم.این یک نوع فیلتر کردن داده ها است:

```
        Dim dv1 As New DataView(ds.Tables("customer"))
        dv1.RowFilter = "customer_city='karaj'"
      GridView1.DataSource = dv1
```

کار دیگر استفاده از کلمه ی کلیدی Like است که اسامی را فیلتر می کند:

```
        Dim dv2 As New DataView(ds.Tables("customer"))
        dv2.RowFilter = "customer_city='karaj' AND customer_name LIKE 'r%'"
      GridView2.DataSource = dv2
```

در این کد تمام افرادی که ساکن شهر کرج هستند و اول اسمشان r دارد را به ما می دهد.علامت % یعنی هر حرفی و به هر تعدادی(حتی ۰ تا).

و یا کد زیر که تمام افرادی که ساکن شهر کرج هستند و اول وسط یا آخر اسمشانیک بار از حرف r استفاده شده را می دهد:

```
        Dim dv3 As New DataView(ds.Tables("customer"))
        dv3.RowFilter = "customer_city='tehran' AND customer_name LIKE '%r%'"
      GridView3.DataSource = dv3
```

خوبی این موارد این است که هر گونه فیلتر داده ها را می توانید بدون اتصال و باز و بسته کردن Connection ایجاد کنید.اگر کمی با Sql آشنا باشید می توانید فیلتر های بسیار جالبی روی داده ها اعمال کنید.کلمات کلیدی از قبیل BETWEEN AND OR <-> NOT ISNULL - + / * و ....

شما همچنین می توانید از توابع جمعی نظیر Min Max و... نیز استفاده کنید.در مثال زیر یکی از توابع جمعی را روی دو جدول که با یک Relation به هم متصل هستند اعمال می کنیم.یعنی نتیجه ی حاصل از رابطه ی بین دو جدول را با اعمال تابع جمعی Max فیلتر کردیم.

ابتدا مثل همیشه یک سری کد عمومی و تکراری:

```
        Dim css As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
```

```
        Dim cons As New SqlConnection(css)
        Dim cmd1 As String = "SELECT * FROM branch"
        Dim cmd2 As String = "SELECT * FROM account"
        Dim ad As New SqlDataAdapter(cmd1, cons)
        Dim dss As New DataSet
        cons.Open()
        ad.Fill(dss, "branch")
        ad.SelectCommand.CommandText = cmd2
    ad.Fill(dss, "account")
```

در این کد ما دو Query را اجرا کردیم و به ترتیب جداول حاصل را در DataSet قرار دادیم.حالا که دو جدول داریم رابطه ای را بین ان ها برقرار می کنیم:

```
Dim rel As New DataRelation("MyRelation",
dss.Tables("branch").Columns("branch_name"),
dss.Tables("account").Columns("branch_name"))
```

باز هم ابتدا یک نام, سپس نام ستونی از جدولی که کلید اصلی است و در نهایت نام ستونی از جدولی درگر که کلید خارجی است.البته این رابطه تکراری است چون پیش تر هم ان را دیده بودید.حالا این رابطه را به Add DataSet می کنیم:

```
    dss.Relations.Add(rel)
```

همانطور که قبلا گفتیم جدولی که Primary Key دارد جدول Parent و جدولی که Foreign Key دارد جدول Child است.حال یک شی از نوع DataView تعریف می کنیم :

```
    Dim dv As New DataView(dss.Tables("branch"))
```

قبل از ادامه ی کار می خواهیم ببینیم معنی عبارت زیر چیست:

```
MAX(child(MyRelation).account_number)
```

Max که تابع جمعی است و مقدار بیشینه ی یک ستون را بین سطر های مختلف بدست می آورد.مقدار ورودی آن هم نام آن ستون است.ولی ما به جای نام آن ستون , از عبارت Child(MyRelation) استفاده کردیم.این به این معنی است که تابع Max باید روی جدول حاصل از رابطه ی MyRelation اعمال شود و عبارت account_number به این معنی است کهMax از ستون account_number محاسبه شود.پس فهمیدیم که نامی که برای رابطه در نظر می گرفتیم کجا به درد می خورد.پس تابع Child هم کارش بدست اوردن جدول حاصل از رابطه ای است که نام آن را به عنوان ارگومان ورودی در یافت می کند.پس کاربرد نام رابطه به نوعی دردستورات Sql است چون در اینجا ما می خواهیم آن را در متد RowFilter به کار ببریم که در اصل همان عبارت جلوی WHERE است:

```
Dim dv As New DataView(dss.Tables("branch"))
dv.RowFilter = "MAX(child(MyRelation).account_number) > 7"
```

در اینجا ابتدا متغیری از نوع DataView تعریف کردیم و به صفت RowFilter آن عبارت زیر را نسبت دادیم :

```
MAX(child(MyRelation).account_number) > 7
```

که آن سطر هایی را از جدول حاصل رابطه جدا می کند که شماره حسابشان از عدد ۷ بزرگتر است.برای نمایش هم کافیست از GridView استفاده کنید:

```
GridView1.DataSource = dv
GridView1.DataBind()
```

دقت کنید که در اینجا ما به دلیل استفاده از DataView و خاصیت RowFilter آن و تاستفاده از توابع جمعی توانستیم حاصل را در GridView قرار دهیم درحالیکه نتیجه ی یک رابطه ی معمولی را نمی توان به صورت عادی در GridView نمایش داد مگر اینکه از خلاقیت برنامه نویسی خود استفاده کنید و راه خاصی بیندیشید.

شما همانطور که می توانستید به جداول DataSet سطر اضافه کنید و سپس آن را Update کنید می توانید به ان ستون نیز اضافه کنید.فایده ی این کار در این است که می توانید مقادیر این ستون جدید را به گونه ای بر اساس مقادیر ستون ها ی قبلی آن محاسبه کنید و نمایش دهید.مثلا یک جدول دارید که در هر سطر از آن مقدار گردش مالی هر فرد به ترتیب ماه به ماه ذکرشده باشد.شما می توانید با تابع جمعی AVG و به کاربردن ان در خصوصیت RowFilter از شی DataRow میانگین گردش مالی هر فرد را محاسبه کرده و سپس مقادیر حاصل را در یک ستون جداگانه و متناظر با آن بدست آورید.این کار ها بدون نیاز به اتصال مجدد به پایگاه داده انجام می شود و خوبیش این است که کاملا به روز می باشد و در هر جای صفحه می توانیدآن ها را محاسبه کنید.در مثال زیر ما جدول Branch را در نظر گرفتیم و می خواهیم که میانگین موجودی تمام شعبه ها را به عنوان ستونی جداگانه نمایش دهیم.حالت اولیه ی این جدول به صورت زیر است:

branch_name	branch_city	branch_assets
emam	zanjan	210000000
noor	qazvin	260000000
payam	tehran	300000000
razi	karaj	400000000

و حالت پس از اجرای مثال به صورت زیر می شود:

branch_name	branch_city	branch_assets	Avg_assets
emam	zanjan	210000000	292500000
noor	qazvin	260000000	292500000
payam	tehran	300000000	292500000
razi	karaj	400000000	292500000

می بینید که یک ستون به ستون های آن اضافه شده.البته این مثال بسیار ساده خواهد بود.و به درد مقایسه می خورد.مثلا پس از محاسبه ی میانگین موجودی ها می توان شعبه هایی را که داراییشان از میانگین کمتر است را بازیابی و طی عملیاتی اصلاح کرد.که در اینجا ۲ شعبه ی اول این مشکل را دارند:

branch_name	branch_city	branch_assets	Avg_assets
emam	zanjan	210000000	292500000
noor	qazvin	260000000	292500000

البته Query های پیچیده ی Sql هست که این مقادیر را یک جا محاسبه کنند و به خروجی ببرند مثلا در اینجا اگر بخواهید یک راست با Sql آن ها را بدست آورید به صورت زیر می شود:

```
  with avg_assets(value) as (select avg(branch_assets) from branch)
  select branch_name,branch_assets
  from branch, avg_assets
where branch.branch_assets < avg_assets.value
```

With در Sql یک متغیر است که می توان مقداری را به آن نسبت داد که در اینجا میانگین دارایی همه ی شعبه ها است.سپس با ضرب کارتزین این مقدار در Branch جلوی عبارت From در حقیقت انگار ستون مورد نظر را به آن اضافه کردیم و از این جدول حاصل آنهایی را که داراییشان از میانگین کمتر است را بدست اوریم.

به دلیل واضح شدن مثال ابتدا خود مثال و خروجی ها را گفتیم حالا به سراغ کد می رویم.برای اضافه کردن یک ستون از شی DataColumn استفاده می کنیم که از خانواده ی DataRow و DataTable است .این شی ۳ آرگومان می گیرد.۱-نام ستون ۲- نوع

داده ای مقادیر ستون ۳-مقادیر ستون که می تواند یک Sql Query باشد که از نوع String است و در حقیقت به شما اجازه ی محاسبه ی مقداری را از ستون های پیشین می دهد دکه ما میانگین ستون Branch_assets را در آن قرار دادیم و نامش را Avg_assets و نوع داده ای آن را عددی قرار دادیم:

```
  Dim dc As New DataColumn("Avg_assets", GetType(Integer),
"AVG(branch_assets)")
```

طریقه یAdd کردن این ستون به جدول نیز بسیار ساده است به این صورت که از متد Column شی DataTable استفاده می کنیم وگزینه ی Add را بر می گزینیم و مقدار آرگومان ورودی آن را نام نتغیری که از نوع DataColumn تعریف کرده ایم قرار می دهیم:

```
  dss.Tables("branch").Columns.Add(dc)
```

حالا برای نمایش به صورت زیر عمل کنید:

```
    GridView1.DataSource = dss.Tables("branch")
  GridView1.DataBind()
```

برای بدست آوردن نام و موجودی شعبه هایی که از میانگین کمتر هستند نیز از یک شی DataView استفاده می کنیم و مقدار RowFilter آن را branch_assets < Avg_assets قرار می دهیم که Avg_assets نام ستون جدید است.باز هم دقت کنید این نام در Query های sql به درد می خورد.در حالت کلی نام متغیر برای Add کردن و نام رشته ای برای استفاده در QuerySql است:

```
        Dim dv As New DataView(dss.Tables("branch"))
        dv.RowFilter = "branch_assets < Avg_assets"
        GridView2.DataSource = dv
    GridView2.DataBind()
```

می بینید که چقدر استفاده از اشیا DataView-DataSet-DataRow و... در محیط Connection بسته به کار ما می آید و می تواند کارایی سایت ما را بالا ببرد. به بحث XMLکه رسیدیم کاربرد های DataSet را در XML نیز بیان خواهیم کرد.

Data Binding

پیوند دادن منبع داده ای به کنتر ل ها ی وب و نمایش آن ها را DataBinding می نامند.مثلا اگر از یک منبع داده مقداری را در یک کنترل وب مثل یک TextBox قرار دهید در حقیقت داده ها را Bind کردید.این عمل در ابتدا به این صورت قابل اجرا است که

شما متغیری عمومی یا محافظت شده درCode-Behind تعریف کنید و سپس به آن در داخل صفحه ی Aspx دسترسی داشته باشید.برای مثال یک متغیر به صورت زیر تعریف کنید:

```
Public db As String = "Hello Word!"
```

یک جز از DataBinding دسترسی به اجزای Asp.NET از داخل صفحه است.برای اینکه در بین تگ های صفحه بتوانید این متغیر را صدا زده و از آن استفاده کنید باید از تگ <...#%> استفاده کنید و در داخل آن به متغیری که در Code-Behind تعریف کرده بودید دسترسی داشته باشید.در کد زیر ما به متغیر عمومی که در بالا تعریف کردیم دسترسی داریم:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>---Data Binding Page---</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
          <%#db%>
       </div>
    </form>
</body>
</html>
```

ولی اگر این صفحه را اجرا کنید چیزی در صفحه مشاهده نمی کنید.تا بحال ما برای کنترل های مختلف DataSource های مختلفی ایجاد کردیم .ولی هیچ کدام برای نمایش آنها کافی نبود تا اینکه آن ها را DataBind می کردیم مثل کنترل GridView.در اینجا هم تا حالا ما DataSource را مشخص کردیم و حالا باید صفحه را DataBind کنیم تا مقدار این متغیر که Hellow Word است به خروجی منتقل شود .این کار را در رویداد Page_Load انجام می دهیم:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    Me.DataBind()
End Sub
```

به این عمل که ما انجام دادیم و مقدار یک متغیر رادر صفحه ی Aspx به کار بردیم DataBinding به روش Single-Value می گویند.برای آشنایی هر چه بیشتر با این روش در زیر مثال های دیگری در این زمینه را می بینیم.می خواهیم از مقدار برگشتی یک تابع در یک Label استفاده کنیم.ابتدا تابع را به صورت زیر می نویسیم:

```
    Public Function return_title() As String
        Return Me.Title.ToString
    End Function
```

این تابع عنوان یا Title یک صفحه را از نوع رشته ای بر می گرداند.حال یک کنترل Label به صفحه اضافه کرده و می خواهیم خاصیت Text آن را برابر مقدار برگشتی این تابع قرار دهیم.همانطور که گفتیم برای دسترسی به اجزای اجزای Asp.NET از داخل صفحه , از تگ <...#%> استفاده می کنیم.در اینجا هم جلوی خاصیت Text این Label از این تگ استفاده کرده و در داخلش تابع را صدا می زنیم:

```
<asp:Label ID="label1" runat="server" Text='<%# return_title() %>'>
```

نکته ی قابل توجه این است که این تگ وقتی در جلوی خاصیتی از یک کنترل به کار برده می شود حتما باید داخل "" و یا '' قرار گیرد.

به کد زیر دقت کنید.خروجی آن AliReza است.علامت + دو رشته ی طرفینش را به هم اتصال می دهد مثل کدهایی که در Code-Behind می نوشتید:

```
<%#"ali" + "Reza"%>
```

کد زیر یک خاصیت فقط خواندنی ایجاد کرده و نام یک تصویر را از نوع String برمی گرداند:

```
    Public ReadOnly Property FilePath() As String
        Get
            Return "apress.jpg"
        End Get
    End Property
```

و ما ان را در خاصیت ImageUrl از کنترل image استفاده کردیم تا خروجی از نوع تصویر باشد:

```
<asp:Image ID="image1" runat="server" ImageUrl='<%# FilePath %>' />
```

و یا کد زیر متغیری عددی به نام a تعریف کرده و مقدار اولیه اش را ٤ قرار می دهد:

```
    Public a As Integer = 4
```

و ما از آن به شکل زیر استفاده کردیم:

```
<asp:label ID="label2" runat="server" Text='<%# (a*2)+15 %>'></asp:label>
```

چون در آن از تگ <...#%> استفاده شده مقدار a را از Code-Behind بازیابی کرده و سپس آن رادر ۲ ضرب می کند و حاصل را + ۱۵ می کند و جواب نهایی را در خاصیت Text کنترل label قرار می دهد.همچنین شما می توانید توابعی با آرگومان ورودی را صدا بزنید. در زیر تابعی ساده داریم که یک ورودی عددی دارد که آن را با ٦ جمع کرده و حاصل را برمی گرداند:

```
    Public Function return_sum(ByVal i As Integer) As Integer
```

```
        Return i + 6
    End Function
```

و ما آن تابع را با ورودی متغیر a که بالاتر تعریف کردیم صدا زدیم و خروجیش ۱۰ شد:

```
<%# return_sum(a) %>
```

حتما تا به حال دستور Request.Browser.Browsers را دیده اید.این یک دستور ساده مثل Page.Title است.که ما Page.Title را در یک تابع به کار بردیم و خروجیش را بر گرداندیم.ولی می توان بدون استفاده از تابع نیز این کار را انجام داد یعنی خود دستور را در داخل تگ به کار برید:

```
  <%# Request.Browser.Browser %>
<%#Page.Title%>
```

نکته ی قابل توجه این است که خیلی ها این تگ را با تگ script block اشتباه می گیرند.تگ script block تگی است که در آن می توان کد های VB.Net را در آن و در داخل صفحه ی ASPx نوشت.نمونه ای از آن به صورت زیر است:

```
  <%@ Page Language="vb" %>
  <script runat="server">

      Sub Page_Load()
      time.text=Hour(Now) & ":" & Minute(Now) & ":" & Second(Now)
      End Sub

  </script>
  <html>
  <head>
      <title>The Punctual Web Server</title>
  </head>
  <body>
      In WebServerLand the time is currently: <asp:Label id="time"
runat="server"></asp:Label>
  </body>
</html>
```

خوب این دو تگ که اصلا شبیه هم نیستند؟منظور از اینکه گفتیم خیلی ها این تگ را با تگ script block اشتباه می گیرند از نظر ظاهری نیست بلکه از نظر طرز کار است.چون تگ script block نیز به هر نحوی به اجزای Asp.NET دسترسی دارد.ولی تفاوت این دو تگ در اینجاست که شما در تگ script block می توانید هر چند خط و هر نوع و تعداد دستوری رابنویسید ولی در تگ <%#...%> شما نمی توانید هر گونه کد نویسی کنید.مثلا کد زیر را اگر بنویسید Error های جور واجوری از شما می گیرد :

```
<%#
    If a = 0 Then
       Response.Write("a is a Zero")
    Else
       Response.Write("a is not a Zero")
    End If
%>
```

پس تگ <%#...> تنها برای DataBinding کاربرد دارد و بس.جلوتر با این تگ بسیار کار خواهیم داشت از جمله کاربردش در کنترل Repeater.

تگ دیگری که در زمینه ی DataBinding کاربرد دارد تگ <%$...> است .در زیر مثال هایی از کاربرد این تگ را می بینید:

```
<asp:Literal ID="literal1" runat="server" Text='<%$ AppSettings:aaa
%>'></asp:Literal><br />
```

در اینجا ما به یک appSettingبه نام aaa (که با علامت :از کلمه ی کلیدی appSetting جدا شده)در فایل Web.Config دسترسی داریم.و آن را در یک کنترلLiteral قرار می دهیم.بر خلاف تگ <%#...> از تگ <%$...> نمی توان در هر جا و جای هر خاصیت کنتر ل های وب استفاده کرد.این تگ تنها در کنترل Literal (کنترلی که برای متونی که نمی خواهند تغییر کنند استفاده می شود) که کنترلی مثل Label است کار می کند.شما همچنین می توانید به ConectionString موجود در فایل Web.Config دسترسی داشته باشید:

```
<asp:literal ID="literal2" runat="server" Text='<%$ ConnectionStrings:aaa
%>'></asp:literal>
```

برای دسترسی به ConectionString ها نیز می توان آن ها را در کنترل SqlDataSource بازیابی کرد که در ادامه در مورد آن صحبت می کنیم.

به مواردی که تا به حال به ان اشاره شد data binding expression گفته می شود.یعنی Bind کردن عبارات.شما می توانید Custom Expression بسازید.مثلا در زیر ما از یک Custom Expression که نامش RandomNumber است استفاده می کنیم و پس از نام ان : و سپس دو عدد در یافت می شود که عدد تصادفی بین ان ها تولید شود:

```
<asp:Literal Runat="server" Text="<%$ RandomNumber:1,6 %>" />
```

Custom Expression ها حتما در داخل تگ <%$...> استفاده می شوند و به همین دلیل یا در کنترل Literal قابل دستیابی هستند و یا در صفت های متناظرشان.از طرفی

ساخت آن ها ساده نیست و بسیار دشوار می باشد و چون در ادامه ی کار نیازی به آن پیدا نمی کنیم از بیان ان صرف نظر می کنیم.

تا اینجا مقدماتی از DataBinding بیان شد.از اینجا به بعد وارد اصل کار می شویم.حتما با کنترل های لیستی مثل ListBox-DropDownList-CheckBoxList-RadioButtonList و .. آشنا هستید.این کنترل ها چند خاصیت مشترک در زمینه ی Data Binding دارند.که در زیر به آن ها اشاره شده:

DataSource	یا همان منبع داده ای که منبع داده ای برای کنترل ها است و داده ها از انجا برای کنترل ها تامین می شوند.این منابع داده ای حتما نباید از پایگاه داده تامین شوند بلکه می توانند از یک کلکسیون مثلHashTable هم تامین شوند.
DataTextField	نام ستون منبع داده است.مثلا برای دسترسی به ستونی از پایگاه داده به نام Emp_name باید این مقدار را Emp_name بگذارید.این خاصیت باعث می شود ستونی از منبع داده در کنترل لیستی نمایش داده شود.
DataValueField	همانطور که میدانید کنترل های لیستی تنها یک ستون بیشتر نداشته و هر سطر آن یک Text و یک Value دارد.شما می توانید Value مقادیری را که در کنترل خود قرارمی دهید نیز تعیین کند مثلا در بالا اگر DataTextField برابر Emp_name است شما می توانید DataValueFieldآن را نام خانوادگی بگذارید که در اینجا هم باید نام ستون منبع داده ذکر شود.

با یک مثال عملی درک بیشتری از این قضیه پیدا می کنید.

از یک HashTable به عنوان DataSource استفاده می کنیم.ومقادیری به آن اضافه می کنیم:

```
    Dim j As New Hashtable
    j.Add(0, "ali")
    j.Add(1, "reza")
    j.Add(2, "jafar")
 j.Add(3, "sadegh")
```

سپس همه نوع کنترل های لیستی را وارد صفحه کرده و منبع داده ای آن ها را همین HashTable قرارداده و همه را DataBind می کنیم:

```
    ListBox1.DataSource = j
    DropDownList1.DataSource = j
    RadioButtonList1.DataSource = j
    CheckBoxList1.DataSource = j
 Me.DataBind()
```

با اجرای صفحه می بینید که مقادیربه درستی نمایش داده نشده اند.زیرا جای هر داده مشخص نیست.اینکار با خصوصیت های DataTextField و DataValueField صورت می گیرد.همانطور که گفتیم باید به این دو مقدار به طور دلخواه نام ستون های منبع داه ای را بدهیم.منبع داده ای ما HashTable است که خود یک جدول است و دارای دو

ستون Key وValue می باشد.پس مثلا برای کنترل ListBox به صورت زیر دو خصوصیت DataTextField و DataValueField را بر اساس منبع داده ای که HashTable است مقدار دهی می کنیم تا تمام مقادیر Key که درHashTable هستنند در کنترل ListBox نمایش داده شوند:

```
   ListBox1.DataSource = j
   ListBox1.DataTextField = "key"
ListBox1.DataValueField = "value"
```

و برای سایر کنترل ها:

```
   DropDownList1.DataSource = j
   DropDownList1.DataTextField = "key"
   DropDownList1.DataValueField = "value"

   RadioButtonList1.DataSource = j
   RadioButtonList1.DataTextField = "key"
   RadioButtonList1.DataValueField = "value"

   CheckBoxList1.DataSource = j
   CheckBoxList1.DataTextField = "key"
CheckBoxList1.DataValueField = "value"
```

و در نهایت Bind کردن همه به صورت یکجا:

```
Me.DataBind()
```

دقت کنید که مقادیری که در تنها ستون این کنترل ها نمایش داده می شوند مقادیر ستون Key از HashTable است.برای دسترسی به Valueآنها , همانطور که در خصوصیت DataValueField مقدار دهی شد می توان یک دکمه نیز به صفحه اضافه کرد و مقادیر Select شده از این کنترل ها را برگرداند:

```
      Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
          Label1.Text += "Selected Item in ListBox1 is :" &
ListBox1.SelectedItem.Text & " And Value is: " & ListBox1.SelectedValue &
"<br>"
          Label1.Text += "Selected Item in DropDownList1 is :" &
DropDownList1.SelectedItem.Text & " And Value is: " &
DropDownList1.SelectedValue & "<br>"
          Label1.Text += "Selected Item in RadioButtonList1 is:" &
RadioButtonList1.SelectedItem.Text & " And Value is: " &
RadioButtonList1.SelectedValue & "<br>"
          Label1.Text += "Selceted Items In CheckBoxList1 is: " & "<br>"
          For Each d As ListItem In CheckBoxList1.Items
              If d.Selected Then
                  Label1.Text += d.Text & " " & d.Value & "<br>"
```

```
            End If
        Next
    End Sub
```

به این ترتیب مقادیر ستون Key از HashTable به عنوان مقادیر Text در کنترل ها به نمایش در می آیند و از آنجا که هر کنترل لیستی مثل DropDownList در هر سطرخود یک Text و یک Value دارد و ما Value را با DataValueField مقداردهی کردیم , برای دسترسی به آن یک دکمه ایجاد کرده و در رویداد کلیک آن گفتیم مثلا:

```
Label1.Text += "Selected Item in ListBox1 is :" &
ListBox1.SelectedItem.Text & " And Value is: " & ListBox1.SelectedValue
```

که Key و Value ان سطری را که انتخاب کردیم را به ما می دهد.

این نکته را بگویم که در این مثال ها باید خاصیت Label EnableViewState هایی که نمایش داده ها را بر عهده دارند برابر False کنید. برای مثال اگر روی دکمه در این مثال برای اولین بار کلیک کنید اطلاعات به درستی به شما نمایش داده می شوند ولی اگر برای بار دوم کلیک کنید اطلاعات جدید در ادامه ی اطلاعات قبلی نمایش داده می شوند که باعث سر در گمی است. اگر Label EnableViewState هایی که نمایش داده ها را بر عهده دارند را برابر False قرار دهید دیگر اطلاعات قبلی که این Label ها حمل می کردند پاک شده و اطلاعات جدید به جایشان قرار می گیرند.ولی این کار کافی نیست زیرا False بودن صفت EnableViewStat وقتی کار می کند که صفحه از نو PostBack شده باشد. برای این کار تمام کدها در این بخش در داخل شرط زیر نوشته می شوند:

```
        If (Not Page.IsPostBack) Then

        End If
```

این شرط باعث می شود که برای هر بار PostBack شدن صفحه (در اثر فشار دکمه ها و ...) صفحه به پایگاه داه متصل نشود و اتصال به آن برقرار نشود تا کارایی سایت بالا رود.در کل این شرط وقتی درست است که صفحه ی مربوطه برای اولین بار به نمایش داده می شوند.

مقدار برگشتیPage.IsPostBack وقتی False است که صفحه برای اولین بار بارگذاری شده و هنوز هیچ دکمه و عامل PostBack روی آن تاثیر نگذاشته ولی اگر True باشد یعنی صفحه پس از بار اول و در صورت کلیک یک دکمه و یا عاملی که باعث

POstBack است بار گذاری شده بنابراین از این خاصیت مهم asp.NET می توانید در خیلی جا ها استفاده کنید از جمله در مثالی که ما در حال حلش هستیم.

پس در کل شرط را بگذارید و صفت EnableViewState کنترل Label را False کنید.چون اگر این کار را نکنید دچار مشکل میشوید.

این شرط را نباید در رویداد کلیک دکمه قرار دهید بلکه باید آن را بر سر برنامه ی اصلی به صورت زیر قرار دهید چون درست است که کاربردش به دکمه مربوط می شود ولی در اصل موقع مقدار دهی کنترل ها باید حضور داشته باشد:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If (Not Page.IsPostBack) Then
            Dim j As New Hashtable
            j.Add(0, "ali")
            j.Add(1, "reza")
            j.Add(2, "jafar")
            j.Add(3, "sadegh")

            ListBox1.DataSource = j
            ListBox1.DataTextField = "key"
            ListBox1.DataValueField = "value"

            DropDownList1.DataSource = j
            DropDownList1.DataTextField = "key"
            DropDownList1.DataValueField = "value"

            RadioButtonList1.DataSource = j
            RadioButtonList1.DataTextField = "key"
            RadioButtonList1.DataValueField = "value"

            CheckBoxList1.DataSource = j
            CheckBoxList1.DataTextField = "key"
            CheckBoxList1.DataValueField = "value"

            Me.DataBind()
        End If
    End Sub
```

همانطور که گفتیم علاوه بر کلکسیون هایی مثل HashTable وArrayList پایگاه داده هم می تواند به عنوان منبع داده باشد.در زیر یک DropDownList را با مقدیری که از یک پایگاه داده در یافت کردیم پر می کنیم:

ابتدا شرط مربوطه را قرار داده و داخلش اتصال به پایگاه داده و Query های مربوطه را می نویسیم:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If (Not Page.IsPostBack) Then
            Dim cs As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
            Dim con As New SqlConnection(cs)
            Dim qs As String = "SELECT customer_name,customer_city + ' '
+customer_street  As Address FROM customer"
            Dim cmd As New SqlCommand(qs, con)

        End If
    End Sub
```

در مورد QueryString توضیحی که باید بدهم این است که در آن ما ابتدا ستون customer_name را انتخاب کردیم و سپس دو ستون Customer_city و Customer_street را به هم وصل کرده و با نام Address آن را انتخاب کردیم.

در ادامه در بلوکگ Connection Try...End Try را باز کرده و سپس command را ExecuteReader کرده و حاصل را در یک متغیر از نوع SqldataReader قرار دادیم و در نهایت آن را به عنوان منبع داده ای برای یک DropDownList انتخاب کرده و قبل از DataBind کردن, برای نمایش درست, ستون های متناظر را به خاصیت های DataTextFieldی و DataValueField دادیم.در نهایت آن را DataBind می کنیم و Reader را می بندیم:

```
            Try
                con.Open()
                Dim reader As SqlDataReader
                reader = cmd.ExecuteReader
                DropDownList1.DataSource = reader
                DropDownList1.DataTextField = "customer_name"
                DropDownList1.DataValueField = "Address"
                DropDownList1.DataBind()
                reader.Close()
            Catch ex As Exception
                Response.Write(ex.Message)
            Finally
                con.Close()
        End Try
```

می توان یک دکمه هم آنجا قرار داد و رویداد کلیکش را به صورت زیر نوشت:

```
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Label1.Text += "Address Of " & DropDownList1.SelectedItem.Text & "
is City & Street: " & DropDownList1.SelectedValue
    End Sub
```

ما یک سری کنترل داریم که به آنها RichDataControls می گویند.مانند GridView و... که در فصل بعد در مورد آنها صحبت می کنیم ولی نکته اینجاست که ما از GridView در راستای Data Binding زیاد استفاده کردیم و برای نمایش آن ,از خاصیت Data Binding کنترل GridView استفاده کردیم.همانطور از شی DataView نیز چند بار به عنوان منبع داده ای کنترل GridView

استفاده کردیم و سپس برای نمایش کنترل GridView را Data Bind کردیم.

نکته ای که مجدد متذکر می شوم این است که هرگاه در رویداد Page_Load صفحه اتصال به پایگاه داده وجود داشت ابتدا شرط if Not Page.IsPostBack را چک کنید و اتصال و دیگر موارد پایگاه داده را د داخل این شرط ذکر کنید.

Data Source Control:

از مباحث دیگر در زمینه ی DataBinding مبحث کنترل های منبع داده ای است.شما تا حالا شیوه های مختلف مدیریت و کنترل داده ها را آموختید.در نسخه ی ۲ Asp.Net چند کنترل جدید قرار داده شده که به وسیله ی آنها می توان به منبع داده وصل شد و اصلا آن را مدیریت کرد بدون هیچ گونه کدنویسی که مهم ترین آن ها به شرح زیر می باشد:

SqlDataSource:

به شما امکان کار کردن با هر پایگاه داده مبتنی بر Sql مثل Sql Server و Oracle را می دهد. در ادامه در باره ی این کنترل و طرز استفاده ی آن توضیح داده می شود.

ObjectDataSource:

به ما امکان کار کردن با کلاس های سفارشی را می دهد به این صورت که شما یک کلاس با توابع و اتصالات مختلف به DataBase مختلف ایجاد می کنید وسپس با این کنترل به اجزای آن کلاس دسترسی دارید.در ادامه در باره ی این کنترل و طرز استفاده ی آن توضیح داده می شود.

XmlDataSource:

به شما امکان کار با DataBase های Xml رامی دهد.که در بخش Xml به آن اشاره می کنیم.

:SiteMapDataSource

به شما امکان پیمایش نقشه ی سایت را می دهد.نقشه ی سایت یک فایل با پسوند sitemap. است که در بخش WebSite Navigation به آن اشاره می شود.

این کنترل ها هیچ نقش فیزیکی در صفحه ی شما نخواهند داشت و اگر برنامه ی خود را اجرا کنید انها را نمیبینید.ولی در نمای Design شما آنها را میبینید تا بتوانید به صورت Visual با آن ها کار کنید.ولی اگر بخواهید در نمای Design هم دیده نشوند از منوی View گزینه ی Non Visual Controls را انتخاب کنید.مراحلی که در زیر میبینید Life Cycle یک صفحه ی Asp.Net است:

1. The page object is created (based on the .aspx file).
2. The page life cycle begins, and the Page.Init and Page.Load events fire.
3. All other control events fire.
4. The data source controls perform any updates. If a row is being updated, the Updating and Updated events fire. If a row is being inserted, the Inserting and Inserted events fire. If a row is being deleted, the Deleting and Deleted events fire.
5. The Page.PreRender event fires.
6. The data source controls perform any queries and insert the retrieved data in the linked controls. The Selecting and Selected events fire at this point.
7. The page is rendered and disposed.

در مرحله ی اول صفحه ایجاد می شود.در مرحله ی ۲ شروع چرخه ی حیات صفحه و آماده سازی آن (initial) و رویداد Page.Init می باشد.سپس تمام رخداد های مشخص شده ی مربوط به کنترل ها ایجاد می گردند.سپس در مرحله ی ٤ همه ی کنترل های منبع داده ای به روز رسانی های مربوطه را انجام می دهند.در مرحله ی ٥ پیش رندر صفحه رخ می دهد یعنی این مرحله قبل از آماده سازی نهایی صفحه صورت می گیرد.سپس کنترل های منبع داده ای پرس و جو های مربوطه شامل Insert کردن و ا ارتباط بین جداول را انجام داده و صفحه آماده ی نمایش می شود.

:SqlDataSource

به صورت زیر تعریف می گردد:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
...></asp:SqlDataSource>
```

و فضاهای نام زیر را پوشش می دهد:

- **System.Data.SqlClient**

- System.Data.OracleClient
- System.Data.OleDb
- System.Data.Odbc

این ها همان Provider های پایگاه داده هستند.و به صورت زیر به تعریف SqlDataSource اضافه می شوند در زیر ما فضای نام مربوط به پایگاه داه ی Sql را به آن نسبت دادیم:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ProviderName="System.Data.SqlClient" ></asp:SqlDataSource>
```

مهم تر از هر چیز رشته ی اتصال است.شما اگر ارز قبل تگ ConnectionStrings را در فایل Web.Config پر کرده باشید با استفاده از تگ <%$..> می توانید به آن دسترسی داشته باشید:

```
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ProviderName="System.Data.SqlClient" ConnectionString="<%$
ConnectionStrings:aaa %>" ></asp:SqlDataSource>
```

مرحله ی بعد ایجاد یک Query برای این کنترل است.اگر هدف شما Select کردن تعدادی سطر باشد از گزینه ی SelectCommand استفاده کنید:

```
    <asp:SqlDataSource
     ID="SqlDataSource1"
     runat="server"
     ProviderName="System.Data.SqlClient"
     ConnectionString="<%$ ConnectionStrings:aaa %>"
     SelectCommand="SELECT * FROM depositor">
    </asp:SqlDataSource>
```

این متدها در قسمت Properties از Visual Studio نیز قابل دسترسی هستند.ولی ما به صورت کد ینگ همه ی آنها را بیان می کنیم تا با صفحه ی Aspx نیز آشنا شوید.

سپس می توانید خروجی این کنترل را در یک جا نمایش دهید.جدول ما ۲ ستون دارد.۱-نام ۲- شماره حساب.می خواهیم نام ها را در یک کنترل ListBox قرار داده و سپس کل داده ها را در یک GridView نشان دهیم.

برای نمایش نام ها در کنترل ListBoxدر رویداد Page_Load به صورت زیر عمل می کنیم:

```
        ListBox1.DataSource = SqlDataSource1
        ListBox1.DataTextField = "customer_name"
    Me.DataBind()
```

البته روش های مختلفی برای این کار وجود دارد مثلا در تگ DataSourceID از کنترل ListBox می توانید نام SqlDataSource مربوطه را ذکر کنید:

```
<asp:ListBox ID="ListBox1" runat="server" DataSourceID="SqlDataSource1"
DataTextField="customer_name"></asp:ListBox>
```

اگر این کار را کنید نیاز به Bind کردن صفحه ندارید چون این کار خودکار انجام می شود.این کار را برای کنترل GridView هم می توانید انجام دهید.صفت DataSourceID شناسه ی(ID) کنترل منبع داده ایاست که ما در بالا نام ان را SqlDataSource۱ گذاشتیم.حال نوبت GridView است:

```
    GridView1.DataSource = SqlDataSource1
  Me.DataBind()
```

در کنترل SqlDataSource یک خاصیت وجود دارد به نام DataSource خوب شاید تعجب کنید ولی این خاصیت دو مقدار بیشتر ندارد ۱-DataSet و ۲-DataReader که مقدار پیش فرض هم هست.شما هم با شی DataSet و هم با شی DataReader آشنایید و می دانید که هر دو می توانند جداول پایگاه داده را درخود ذخیره کنند.با آشنایی که از DataSet داریم می دانیم که به مراتب بهتر از DataReader است.زیرا پشتیبانی آن از Sorting-filtering و... باعث این امر است.علاوه بر این ها در حجم های انبوه سطر در یک جدول اگر از DataReader استفاده کنید حجم حافظه ای بسیار زیادی را اشغال می کند.

حال می خواهیم یک مثال با دستورات Sql حاوی پارامتر را با هم ببینیم.در این مثال یک کنترل DropDownList داریم که حاوی نام یک سرس شهر است و می خواهیم با انتخاب هر شهر دران اسمی مشتریانی که در آن هر حساب دارند در یک GridView نمایش داده شود.پس ابتدا کنترل منبع داده ای برای اسامی شهر ها در نظر می گیریم:

```
<asp:SqlDataSource
ID="sqldatasource1"
runat="server"
ProviderName="System.Data.SqlClient"
ConnectionString="<%$ ConnectionStrings:aaa %>"
SelectCommand="SELECT DISTINCT customer_city FROM customer">
</asp:SqlDataSource>
```

دقت کنید چون اسامی شهر ها در پایگاه داده بسیار زیاد و تکراری بودند ما از عبارت DISTINCT جلوی Select استفاده کردیم تا تکراری ها از بین رفته و از هر شهر فقط یک نام به ما بدهد.سپس این منبع داده ای را به کنترل DropDownList نسبت می دهیم و از آنجایی که هیچ دکمه ای در میان نیست یعنی پس از انتخاب آیتم از DropDownList می خواهیم به صفحه ی مربوطه برویم میبایست خاصیت

AutoPostBack آن را true قرار دهیم این کار علاوه بر این باعث بهبود کارایی سایت می شود و با هر بار PostBack شدن صفحه باعث می شود هر بار کنترل DropDownList از پایگاه داده خود را به روز نکند:

```
<asp:DropDownList ID="DropDownList1" runat="server"
DataSourceId="sqldatasource1" DataTextField="customer_city"
AutoPostBack="true"></asp:DropDownList>
```

حال نوبت به منبع داده ی دوم می رسد:

```
      <asp:SqlDataSource          ID="sqldatasource2"          runat="server"
ProviderName="System.Data.SqlClient"                    ConnectionString="<%$
ConnectionStrings:aaa                %>"                SelectCommand="SELECT
customer_name,customer_street FROM customer WHERE customer_city=@city">
    </asp:SqlDataSource>
```

همه چیز مثل قبلی است به غیر از Query که کاملا واضح است.حالا نوبت به ایجاد پارامتر می رسد.ما در اینجا یک پارامتر داریم و آن هم @city است.برای ایجاد پارامتر باید از تگ مخصوصی در آن استفاده کنیم .لیست تگ های مورد نیاز ما ومربوط به پارامتر های SqlDataSource به شرح زیر است:

SelectParameter:که پارامتر های مخصوص Query های Select را در برمیگیرد.

UpdateParameter: که پارامتر های مخصوص Query های Update را در برمیگیرد.

DeleteParameter: که پارامتر های مخصوص Query های Delete را در برمیگیرد.

در اینجا چون Query ما از نوع Select است از تگ SelectParameter استفاده می کنیم:

```
      <asp:SqlDataSource          ID="sqldatasource1"          runat="server"
ProviderName="System.Data.SqlClient"                    ConnectionString="<%$
ConnectionStrings:aaa  %>"  SelectCommand="SELECT  DISTINCT  customer_city  FROM
customer"></asp:SqlDataSource>
      <asp:SqlDataSource          ID="sqldatasource2"          runat="server"
ProviderName="System.Data.SqlClient"                    ConnectionString="<%$
ConnectionStrings:aaa                %>"                SelectCommand="SELECT
customer_name,customer_street FROM customer WHERE customer_city=@city">
      <SelectParameters>

      </SelectParameters>
    </asp:SqlDataSource>
```

حالا درداخل این تگ چه بنویسیم؟.لیست عناصری که میتوان از آن ها به عنوان پارامتر استفاده کرد در زیرآمده است:

ControlParameter:مشخص می کند Value پارامتر ما از یک کنترل وب تامین شود.مثلا یک TextBoxداریم که خاصیت Text آن را به عنوان Value پارامتر خود در نظر می گیریم.

QueryString Value:مشخص می کند که Value پارامتر ما از یک QueryString که از صفحه ی قبل به صفحه ی جاری فرستاده شده تامین شود.

SessionState Value: مشخص می کند که Value پارامتر ما از یک شی Session بدست آید که قبلا

آن را مقدار دهی کرده بودیم.

Parameter:که مشخص کننده ی حضور یک پارمتر در Query ماست.و در کنترل هایی مثل GridView که مثلا قصد ویرایش و یا حذف را دارد می باشد.

Cookie Value: مشخص می کند که Value پارامتر ما مقدار یک Qookie باشد.

Profile Value: مشخص می کند که Value پارامتر ما از یک Profile تامین شود که در فصول مربوط به Security به آن می پردازیم.

در اینجا پارامتر ما از خصوصیت SelectedValue کنترل DropDownList تامین میشود.شاید از خود بپرسید ما که DataValueField را مقداردهی نکردیم و DataTextField را مقداردهی کردیم پس چرا خصوصیت SelectedValue کنترل DropDownList انتخاب می شود؟در پاسخ بگویم اگر شما در هر کنترل لیستی Text را تعیین کنید ولی Value راتعیین نکنید مقدار Text خود به خود به Value هم منتقل می شود. از آنجاییکه DropDownList کنترل است پس باید پارامتر از نوع ControlParameter باشد:

```
<SelectParameters>
<asp:ControlParameter  />
</SelectParameters>
```

این تگ باید حداقل حاوی ۳ مقدار مهم باشد:

۱-ControlID:که Id کنترلی که پارامتر از آن تامین میشود است که در اینجا DropDownList1 است.

۲-Name:که نام پاراکتر مربوطه است که در اینجا city است.

۳-PropertName که نام خاصیتی از کنترلی است که پارامتر از آن تامین می شود.مثلا اگر کنترل ما TextBox باشد این خاصیت مقدار Text را به خود اختصاص می دهد و چون در اینجا کنترل ما DropDownList است این خاصیت مقدار SelectedValue را به خود اختصاص می دهد.

این متدها و خواص در قسمت Properties از Visual Studio نیز قابل دسترسی و تنظیم هستند.

در نهایت هم کافی است نتیجه ی حاصل را در یک کنترل GridView نمایش دهیم:

```
        <asp:GridView              ID="GridView1"              runat="server"
DataSourceID="sqldatasource2">
        </asp:GridView>
```

همچنین شما می توانید برای جلوگیری از شلوغی که Query های Sql ایجاد میکنند و بالا رفتن کارایی سایت از StoredProcedure استفاده کنید.اگر بخواهیم مثال بالا را با StoredProcedure تنجام دهیم باید نامStoredProcedure را(که در اینجا sss است) به خاصیت SelectCommand نسبت دهیم :

```
SelectCommand="sss"
```

و نوع آن را StoredProcedure در نظر بگیریم:

```
SelectCommandType="StoredProcedure"
```

خود StoredProcedure هم به صورت زیر است:

```
CREATE PROCEDURE sss
@city VarChar(20)
AS
SELECT customer_name,customer_street From customer
Where customer_city=@city
```

پس کد کلی آن به صورت زیر می شود:

```
          <asp:SqlDataSource         ID="sqldatasource1"         runat="server"
ConnectionString="<%$ ConnectionStrings:aaa %>"
              ProviderName="System.Data.SqlClient"       SelectCommand="SELECT
DISTINCT customer_city FROM customer">
          </asp:SqlDataSource>
          <asp:SqlDataSource         ID="SqlDataSource2"         runat="server"
ConnectionString="<%$                ConnectionStrings:aaa                %>"
ProviderName="System.Data.SqlClient"                        SelectCommand="sss"
SelectCommandType="StoredProcedure" >
          <SelectParameters>
          <asp:ControlParameter      ControlID="DropDownList1"      Name="city"
PropertyName="SelectedValue" />
          </SelectParameters>
```

```
            </asp:SqlDataSource>
             </div>
            <asp:DropDownList          ID="DropDownList1"          runat="server"
DataSourceID="sqldatasource1"
                DataTextField="customer_city" AutoPostBack="true">
            </asp:DropDownList><br />
            <asp:GridView              ID="GridView1"              runat="server"
DataSourceID="SqlDataSource2">
        </asp:GridView>
```

همه چیز مثل قبل است از جمله نحوه ی تعریف و مقدار دهی پارامتر ها.

نکته ی کلی این است که در اینجا می توانستید مقادیر مختلف پارامتر را به صرت Visual در قسمت Command And Parameter Editor نیز انجام بدهید تا از کد نویسی خلاص شوید.

حالا در زیر می خواهیم یک مثال با هم ببینیم که پارامتر ها در آن از یک QueryString تامین شوند.برای این مثال این طور در نظر می گیریم که اسامی شهرها در یک ListBox به صورت غیر تکراری قرار گرفته باشند و یک دکمه هم در آنجا باشد.وقتی روی دکمه کلیک کنیم ایتم انتخاب شده در ListBox با QueryString به یک صفحه ی دیگری منتقل شود و در آنجااز این QueryString که منتقل شده به عنوان پارامتر نام شهر برای بازیابی مشتریانی که در آن شهر زندگی می کنند استفاده می کنیم. پس ابتدا صفحه ی اول را طراحی کنیم.اول یک کنترل SqlDataSource دران قرارداده و نام شهر ها را به طورغیر تکراری بازیابی می کنیم:

```
    <asp:SqlDataSource ID="sqldatasource1" runat="server"
    ProviderName="System.Data.SqlClient"
    ConnectionString="<%$ ConnectionStrings:aaa %>"
    SelectCommand="SELECT DISTINCT customer_city FROM customer">
</asp:SqlDataSource>
```

سپس یک کنترلListBox در صفحه قرار داده و منبع داده ای آن را کنترل SqlDataSource در نظر گرفته و یادتان باشد خاصیت DataTextField نیز برایش تعریف کنیم:

```
    <asp:ListBox ID="listbox1" runat="server" DataSourceID="sqldatasource1"
DataTextField="customer_city"></asp:ListBox>
```

سپس یک دکمه در صفحه قرار داده:

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

و در رویداد کلیکش QueryString مربوطه را ارسال میکنیم:

```
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
```

```
        Response.Redirect("Default5.aspx?city=" & listbox1.SelectedValue)
    End Sub
```

پس ما به صفحه ی Default5.aspx منتقل شده و متغیری به نام city داریم که مقدارش Value آیتم انتخاب شده در ListBox است.این نکته راهم بگویم که شما می توانستید یک تابع به صورت زیر بنویسید:

```
    Sub btn_click()
        Response.Redirect("Default5.aspx?city=" & listbox1.SelectedValue)
    End Sub
```

وسپس دکمه را به این صورت تعریف کنید:

```
<asp:Button ID="Button1" runat="server" Text="Button" OnClick="btn_click" />
```

همانطور که میبینید در خصوصیت OnClick در آن تابع مربوطه را صدا زدیم.

حالا کار صفحه ی اول تمام است و به سراغ صفحه ی دوم می رویم.و کد تعریف کنترل SqlDataSource را به صورت زیر می نویسیم:

```
   <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ProviderName="System.Data.SqlClient" ConnectionString="<%$
ConnectionStrings:aaa %>" SelectCommand="sss"
SelectCommandType="StoredProcedure">
</asp:SqlDataSource>
```

همانطور که می بینیددر اینجا ما از همان StoredProcedures به نام sss که بالاتر تعریف کرده بودیم استفاده کردیم.حال نوبت به تعریف پارامتر می رسد.چون پارامتر ما برای عملیات select است پس از SelectCommand استفاده می کنیم.سپس از آنجایی که نوع ان از QueryString تامین می شود پس از تگ <asp:QueryStringParameter> استفاده می کنیم که حداقل خصوصیاتی که در آن باید مقداردهی شئد ۲ تا است اولی نام پارامتر در Query مربوطه و دومی(QueryStringField) نام متغیری که Value مربوطه در ان وجود دارد و از صفحه ی قبلی ارسال شده.

```
    <SelectParameters>
      <asp:QueryStringParameter  Name="city" QueryStringField="city" />
    </SelectParameters>
```

برای نمایش هم از یک GridView استفاده کرده و منبع داده ای آن را کنترل SqlDataSource مربوطه قرار می دهیم:

```
   <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1">
  </asp:GridView>
```

بحث بعدی مدیریت استثنا هاست.همانطور که قبلا دیده بودید ما در آنجا از بلوک Tre...EndTry برای مدیریت استثنا ها استفاده می کردیم ولی این امر در کنترل SqlDataSource امکان پذیر نیست.پس اگر به مشکلی برخوردیم ناچار به خروج از برنامه و مشاهده ی Error مروطه هستیم.ولی ASP.NET یک روش جالب برای آن در نظر گرفته و آن هم SqlDataSourceStatusEventArgs است که وضعیت کنترل SqlDataSource را بر می گرداند.ببینید شما رویداد های مختلفی برای SqlDataSource دارید مثلا برای عمل Select شما ۲ رویداد Selecting و Selected دارید.برای عملیات Update هم Updating و Updated و همین طور برای Delete و....در این مثال عمل ما Select است پس رویداد Selecting در حین انجام عملیات select و Selected پس از اجرای عملیات تحریک می شود.اگر می خواهید مدیریت استثنا ها را انجام دهید می بایست همیشه رویدادهایی را که به ed ختم می شوند را برنامه نویسی کنید.مثلا ما اگر در این مثال قصد عملیات Select را داریم برای مدیریت استثنا ها باید رویداد Selected را صدا بزنیم.دلیل این کار این است که اگر این کار را انجام دهید e که به تابع ارسال می شود از نوع qlDataSourceStatusEventArgs است و به شما اجازه ی کار با استثناها را می دهد:

```
    Protected Sub SqlDataSource1_Selected(ByVal sender As Object, ByVal e
As      System.Web.UI.WebControls.SqlDataSourceStatusEventArgs)      Handles
SqlDataSource1.Selected

    End Sub
```

اگر به ارگومان ورودی e دقت کنید می بینید که از نوع SqlDataSourceStatusEventArgs است و این یعنی اجازه ی مدیریت استثناها را دارید.اصلا درستش هم همین است که پس از اتمام (Selected)کار اگر با مشکلی برخوردیم آن را به خروجی ببریم نه اینکه در حین عملیات(selecting). اگر شما Selecting را انتخاب کنید رویداد شما به شکل زیر میشد:

```
    Protected Sub SqlDataSource1_Selecting(ByVal sender As Object, ByVal e
As     System.Web.UI.WebControls.SqlDataSourceSelectingEventArgs)     Handles
SqlDataSource1.Selecting
    End Sub
```

که e در آنجا از یک نوع دیگر است و به شما اجازه ی مدیریت استثناها را نمیدهد.حال به اصل مطلب بپردازیم که مدیریت استثناها ست.در داخل رویداد Selected شرطی می نویسیم.که بیانگر این مطلب باشد که اگر استثنایی وجود داشت وارد بدنه ی if شو:

```
If e.Exception IsNot Nothing Then

End If
```

IsNot Nothing یعنی وجود داشته باشد.پس اگر وارد بدنه شدیم ابتدا یک پیغام و سپس True قرار دادن Handling استثنا:

```
Response.Write("An exception occurred performing the query.")
e.ExceptionHandled = True
```

پس تابع به شکل زیر است:

```
Protected Sub SqlDataSource1_Selected(ByVal sender As Object, ByVal e
As System.Web.UI.WebControls.SqlDataSourceStatusEventArgs) Handles
SqlDataSource1.Selected
    If e.Exception IsNot Nothing Then
        Response.Write("An exception occurred performing the query.")
        e.ExceptionHandled = True
    End If
End Sub
```

پس اگر به هر گونه مشکلی در اجرای Query در کنترل SqlDataSource شدید به جای رفتن به صفحه ی Asp.NET Error پیغام بالا برایتان در خروجی نمایش داده می شود.

مثال بعدی که برایتان در نظر گرفتیم شامل عملیات حذف و ویرایش داده های است که در کنترل SqlDataSourceبا کنترل GridView.ما تا اینجا از خاصیت SelectCommand کنترل SqlDataSource استفاده کردیم.حالا می خواهیم از خصوصیت UpdateCommand و DeleteCommandآن نیز استفاده کنیم.پس کنترل را به صورت زیر تعریف می کنیم:

```
<asp:SqlDataSource ID="sourceEmployees" runat="server"
ProviderName="System.Data.SqlClient" ConnectionString="<%$
ConnectionStrings:aaa %>"
    SelectCommand="SELECT * FROM employees"
    UpdateCommand="Update employees Set
emp_firstname=@emp_fname,emp_lastname=@emp_lname,emp_job=@emp_jobs Where
emp_ID=@ids "
    DeleteCommand="DELETE FROM employees WHERE emp_ID=@id ">
    <UpdateParameters>
        <asp:Parameter Name="ids" Type="String" />
        <asp:Parameter Name="emp_fname" Type="String" />
```

```
            <asp:Parameter Name="emp_lname" Type="String" />
            <asp:Parameter Name="emp_jobs" Type="String" />
        </UpdateParameters>
        <DeleteParameters>
            <asp:Parameter Name="id" Type="string" />
        </DeleteParameters>
</asp:SqlDataSource>
```

همه چیز مثل قبل است با این تفاوت که یک Query برای Update و یک Query برای Delete به آن اضافه شده که با آنها آشنایید.در ضمن برای عملیات Update ٤پارامتر نیاز است برای عملیاتDelete هم تنها پارامتر Id است که مشخص شده.دقت کنید پارامتر id که برای Updating در نظر گرفته شده باید نامش با پارامتر id که برای Deleting در نظر گرفته شده متفاوت باشد در غیراین صورت دچار Error همسانی می شویم که ما یکی را ids و دیگری را id در نظر گرفتیم.حال به سراغ کنترل GridView برویم.این کنترل منبع داده ایش SqlDataSource مربوطه است.و یک خصوصیت دیگر دارد به نام DataKeyNamesکه ستون کلید اصلی است.این امر به خاطر این است که هنگام ویرایش داده ها به کاربر اجازه ی ویرایش ستونی که کلید اصلی است را ندهد.برای نمایش دکمه های Edit وDelete هم نیاز به تگ <Columns> داریم چون به هر حال این تگ یک ستون به GridView اضافه می کند و ما می خواهیم ستونی باشد که درآن یک دکمه به نام Edit و یکی به نام Delete باشد ولی این دکمه ها را ما خودمان تعریف نکنیم بلکه از دکمه ها ی پیش فرض آن استفاده کنیم برای این کار از تگ <asp:CommandField> در داخل تگ <Columns> استفاده کرده و خصوصیت howEditButton وShowDeleteButton را در آن برابر با True قرار می دهیم:

```
  <asp:GridView ID="GridView1"  runat="server" DataSourceID="sourceEmployees"
DataKeyNames="emp_ID" >
        <Columns>
            <asp:CommandField ShowEditButton="True" />
            <asp:CommandField ShowDeleteButton="true" />
        </Columns>
</asp:GridView>
```

ما در بخش مرتبط با GridView به شرح خصوصیات آن می پردازیم.

همه چیز درست است به جز یک کار کوچک و ان هم کمی کد نویسی در رویداد های Updating وDeleting است.شما باید پارامتر هایی را که تعریف کردید مقدار دهی کرده

و پارامتر های قبلی را پاک کنید.پس رویداد Updating کنترل SqlDataSource را به صورت زیر بنویسید:

```
    Protected Sub sourceEmployees_Updating(ByVal sender As Object, ByVal e
As      System.Web.UI.WebControls.SqlDataSourceCommandEventArgs)      Handles
sourceEmployees.Updating
        e.Command.Parameters("@ids").Value                              =
e.Command.Parameters("@emp_ID").Value
        e.Command.Parameters("@emp_fname").Value                        =
e.Command.Parameters("@emp_firstname").Value
        e.Command.Parameters("@emp_lname").Value                        =
e.Command.Parameters("@emp_lastname").Value
        e.Command.Parameters("@emp_jobs").Value                         =
e.Command.Parameters("@emp_job").Value
        e.Command.Parameters.Remove(e.Command.Parameters("@emp_ID"))
        e.Command.Parameters.Remove(e.Command.Parameters("@emp_firstname"))
        e.Command.Parameters.Remove(e.Command.Parameters("@emp_lastname"))
        e.Command.Parameters.Remove(e.Command.Parameters("@emp_job"))
    End Sub
```

همانطور که میبینید یکی یکی پارامتر ها را مقدار دهی کرده و سپس مقادیر قدیمی را پاک می کنیم.برای Deleting هم همین کار را بکنید:

```
    Protected Sub sourceEmployees_Deleting(ByVal sender As Object, ByVal e
As      System.Web.UI.WebControls.SqlDataSourceCommandEventArgs)      Handles
sourceEmployees.Deleting
        e.Command.Parameters("@id").Value                               =
e.Command.Parameters("@emp_ID").Value
        e.Command.Parameters.Remove(e.Command.Parameters("@emp_ID"))
    End Sub
```

آرگومان e حاوی نام پارامتر های اصلی و فرعی می باشد .این کار را با عنصرParameter متد Command انجام می دهد.

پس ما برای هر پارامتر ۲ کار باید انجام دهیم.اولی تعریف آن در تگ Command مربوطه و دومی مقدار دهی و حذف مقدار قبلیش در داخل متدی که عمل خاصی را انجام می دهد و نامش به ing ختم میشود.در مثال زیر مثال بالا را با StoredProcedures انجام می دهیم.

با علم به اینکه StoredProcedures برای Updating به صورت زیر است:

```
CREATE PROCEDURE UpdateEmployee
@empID nvarchar(70),
@emp_fname varchar(30),
@emp_lname varchar(30),
@emp_jobs varchar(30)
AS
```

```
Update employees
Set emp_firstname=@emp_fname,emp_lastname=@emp_lname,emp_job=@emp_jobs
Where emp_ID=@empID
```

و برای Deleting به صورت زیراست:

```
CREATE PROCEDURE DeleteEmployee
@e_ID NVarChar(70)
AS
DELETE From employees Where emp_ID=@e_ID
```

حال ادامه ی کار که به جز عوض شدن SQl Query با StoredProcedure و نام پارامتر ها چیز دیگری نیست.کد کلی آن را در صفحه ی Aspx در زیر می بینید:

```
      <asp:SqlDataSource          ID="sourceEmployees"          runat="server"
ProviderName="System.Data.SqlClient"                    ConnectionString="<%$
ConnectionStrings:aaa %>" SelectCommand="SELECT * FROM employees"
      UpdateCommand="UpdateEmployee"
      UpdateCommandType="StoredProcedure"
      DeleteCommand="DeleteEmployee"
      DeleteCommandType="StoredProcedure"
      >
      <UpdateParameters>
            <asp:Parameter Name="empID" Type="String" />
            <asp:Parameter Name="emp_fname" Type="String" />
            <asp:Parameter Name="emp_lname" Type="String" />
            <asp:Parameter Name="emp_jobs" Type="String" />
      </UpdateParameters>
      <DeleteParameters>
          <asp:Parameter Name="e_ID" Type="string" />
      </DeleteParameters>
      </asp:SqlDataSource><br />
      <asp:GridView               ID="GridView1"               runat="server"
DataSourceID="sourceEmployees"  DataKeyNames="emp_ID">
                  <Columns>
                        <asp:CommandField               ShowEditButton="True"
ShowDeleteButton="true"/>
                  </Columns>
</asp:GridView>
```

و کد پشت صحنه برای رویداد ها:

```
      Protected Sub sourceEmployees_Deleting(ByVal sender As Object, ByVal e
As      System.Web.UI.WebControls.SqlDataSourceCommandEventArgs)      Handles
sourceEmployees.Deleting
          e.Command.Parameters("@e_ID").Value                                =
e.Command.Parameters("@emp_ID").Value
          e.Command.Parameters.Remove(e.Command.Parameters("@emp_ID"))
      End Sub
```

```
        Protected Sub sourceEmployees_Updating(ByVal sender As Object, ByVal e
As       System.Web.UI.WebControls.SqlDataSourceCommandEventArgs)       Handles
sourceEmployees.Updating
            e.Command.Parameters("@empID").Value                              =
e.Command.Parameters("@emp_ID").Value
            e.Command.Parameters("@emp_fname").Value                          =
e.Command.Parameters("@emp_firstname").Value
            e.Command.Parameters("@emp_lname").Value                          =
e.Command.Parameters("@emp_lastname").Value
            e.Command.Parameters("@emp_jobs").Value                           =
e.Command.Parameters("@emp_job").Value
            e.Command.Parameters.Remove(e.Command.Parameters("@emp_ID"))
            e.Command.Parameters.Remove(e.Command.Parameters("@emp_firstname"))
            e.Command.Parameters.Remove(e.Command.Parameters("@emp_lastname"))
            e.Command.Parameters.Remove(e.Command.Parameters("@emp_job"))
    End Sub
```

نکته ی بسیار مهم در اینجا این است که اگرنام پارامتر هایی که در Sql Query تعریف کردیم عینا مثل نام ستون متناظر باشند آنگاه نیازی به تعریف پارامتر در رویداد Updatingو تگ UpdateParameter نداریم.در مثال مربوط به Update کنترل GridView با SqlQuery اگر کمی دقت کنید میبینید نام پارامتر ها با نام فیلد های متناظرشان متفاوت است.مثلا اگر Sql Query به شکل زیر باشد نیازی به تعریف پارامتر حتی در تگ UpdateParameter نیز نداریم چون به طور اتوماتیک پارامتر هایی که در Sql Query تعریف کردید با فیلد های GridView تطبیق داده خواهند شد:

```
Update customer SET customer_name=@customer_name,customer_city=@customer_city Where
customer_Id=@customer_ID
```

اگر شما از StoredProcedure در این راستا استفاده کنید وضع دقیقا مثل اینجاست.آن پارامتر هایی که به عنوان آرگومان ورودیStoredProcedure در نظر گرفتید اگر با پارامتر هایی که در قسمت Query(در داخل بلوک AS) متفاوت باشند باید در رویداد Updating کنترل SqlDataSource آنها را تعریف کنید(مثل مثال بالا) و شما مجبورید کدهایی که در مثال بالا زدیم را در رویدادUpdating کنترل SqlDataSource بنویسید.در مثال بالا پارامتر های تعریف شده در StoredProcedure و آنهایی که در Query آن نوشته شده بودند به صورت زیر بود که پارامتر ها با ستون های متناظرشان متفاوت است:

emp_ID--------empID
emp_firstname-----emp_fname
emp_lastname------emp_lname

emp_job-------emp_jobs

آن پارامتر هایی که نامشان با ستون متناظرشان متفاوت است حتما باید در رویداد Updating کنترل SqlDataSource تعریف شوند ولی آنهایی که برابرند نیازی به تعریف انها در رویداد Updating کنترل SqlDataSource نیست.

حتما نباید تمامی پارامتر ها با تمامی نام ستون ها برابر باشند یا نباشند.یادتان باشد که هر تعداد که برابر بود را کاری نداشته باشید ولی آنهایی که برابر نبود را در رویداد Updating کنترل SqlDataSource تعریف کنید.این نکته برای SqlQuery و Stored Procedure فرقی نمی کند.

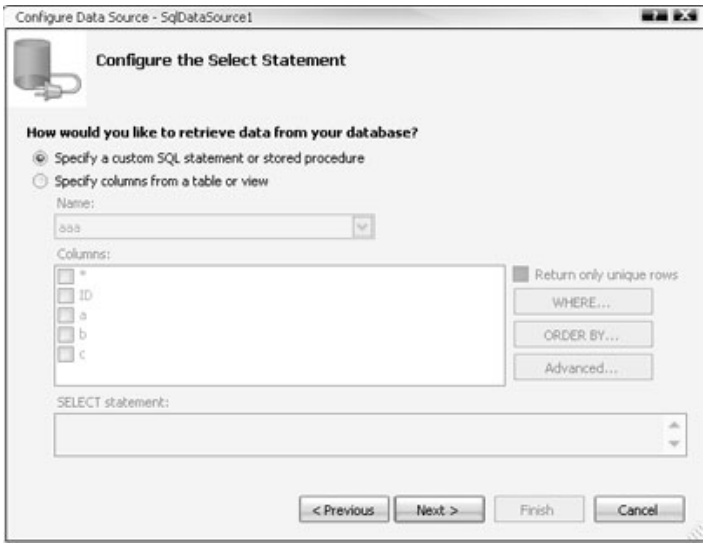
تمام مواردی که گفته شد برای عمل Delete هم کاربرد دارند.

در زیر مثالی را از UpdateوDelete می بینید که بدون نیاز به تعریف پارامتر انجام می گیرد:

```
            <asp:SqlDataSource      ID="SqlDataSource1"      runat="server"
ConnectionString="<%$ ConnectionStrings:aaa %>"
            SelectCommand="SELECT * FROM [employees]"
            UpdateCommand="UPDATE              employees              SET
emp_firstname=@emp_firstname,emp_lastname=@emp_lastname,emp_job=@emp_job
WHERE emp_ID=@emp_ID"
            DeleteCommand="DELETE FROM employees WHERE emp_ID=@emp_ID">
            </asp:SqlDataSource>
        <br />
        <asp:GridView            ID="GridView1"            runat="server"
DataSourceID="SqlDataSource1" DataKeyNames="emp_ID">
            <Columns>
                <asp:CommandField                    ShowEditButton="True"
ShowDeleteButton="true" />
            </Columns>
    </asp:GridView>
```

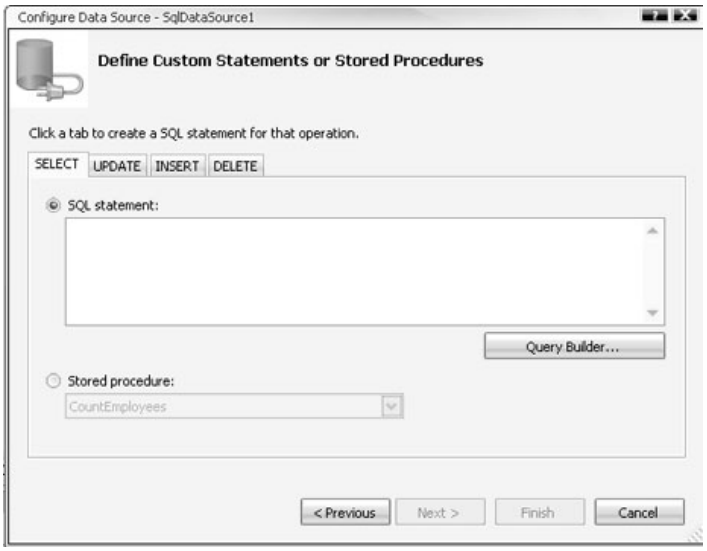
کار ما با این کنترل تمام است.در حالیکه کارهایی که ما انجام دادیم تنها مثال های ساده بود.چه کار های پیچیده ای که نمی توان با SqlDataSource انجام داد.می توان Join ها در جداول و View ها و... را با آن انجام داد که آنها به عهده ی خودتان است.اگر به قسمت Task کنترل SqlDataSource رفته و روی Configure DataSource کلیک کنید پنجره ای باز می شود که از شما نام پایگاه داده را می خواهد و به محض دادن نام ConnectionString لازم را خودش تولید کرده و با زدن کلید Next شما را به صفحه

ای منتقل می کند که لیست تمامی جداول موجود در پایگاه داده وجود دارند.در آنجا شما ۲ انتخاب دارید.صفحه ی مربوطه به صورت زیر است:

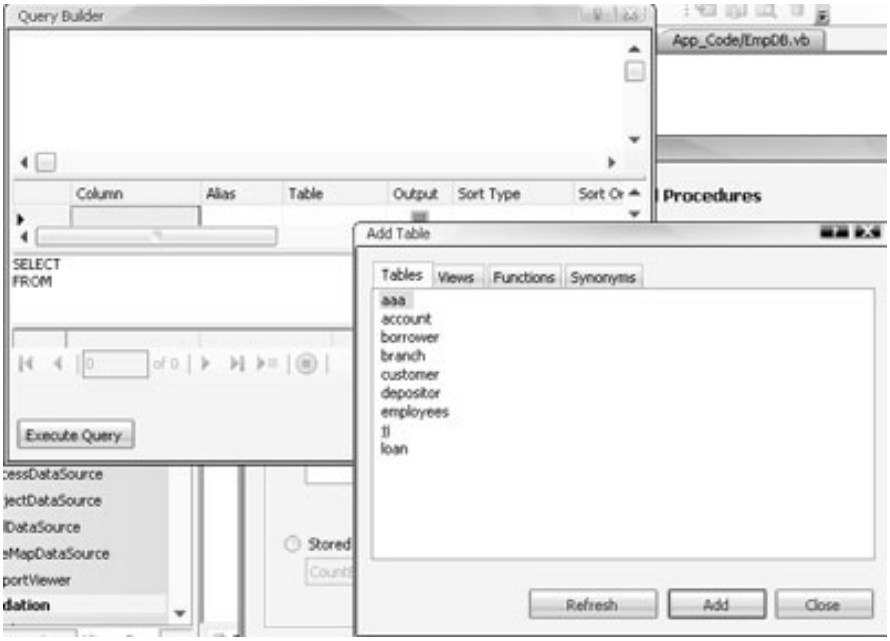


۱-**Specify Columns From a Table View:**که اگر فعال باشد شما می توانید **Query** های ساده و آماده ایجاد کنید.این کتر با ۳ دکمه ایکه در سمت راست صفحه قرار دارند امکان پذیر است.

۲-**Specify a Custom Sql Statement Or Stored Procedures:**که اگر فعال شود و کلید **Next** را بزنید وارد صفحه ی جدیدی به صورت زیر می شوید:



که دراینجا اگر **RadioButton** مربوط به **Sql Statement** را چک زده و روی **QueryBuilder** کلیک کنید وارد صفحه ای به شکل زیر می شوید:



در این صفحه در کنار پنجره ی QueryBuilder که در سمت چپ و بالا ی این شکل مشاهده می کنید یک پنجره به نام AddTable ظاهر می شود که شما می توانید جدول های محتلف را کنار هم قرار دهید و حتی در View Tab نیز شما می توانید View هایی راکه ایجاد کردید به پنجره ی QueryBuilder اضافه کرده و به صورتی کاملا Visual Query های پیچیده بسازید .مثلا با Drag And Drop کردن سطر های هر دو جدول به هم می توانید بدون کد نویسی جدول ها را به هم Join هم کنید و فعالیت ها ی دیگری که بسته به سلیقه ی شما دارد.و حتی آن Query را درهمانجا اجرا هم کنید با زدن دکمه ی Execute Query که در سمت چپ و پایین صفحه ی QueryBuilder وجود دارد.

حال می رسیم به اشکالات کنترل SqlDataSource:

خوب تا حالا طرز استفاده از کنترل SqlDataSource را آموختید و دیدید که این کنترل چه توانایی هایی دارد و شما را از کد نویسی رها می کند ولی یک سری اشکالاتی در این کنترل وجود دارد که در زیر به آنها اشاره می شود:

۱-اگر شما از این کنترل در یک صفحه استفاده کنید تمام Query ها و اهداف شما در همان صفحه ذخیره می شود.یعنی تمام موارد مربوط به پایگاه داده در همان صفحه جاسازی می شوند.در حالیکه شما می توانید یک کلاس جداگانه بنویسید (همانطوری که در بخش Component ها نوشتیم) و در صفحه ی خود از آن استفاده کنید بدون اینکه

بی خودی صفحه را سنگین کرده و شلوغ کنید.به این امر اصلاحا Hard-Coding میگویند.Hard-Coding یعنی کد نویسی مستقیم در صفحه که هنگام تغییرات در صفحه ما را دچار مشکل می کند.تا می توانید از Hard-Coding فاصله بگیرید.

۲-در تعداد صفحه ی زیاد تعداد کنترل SqlDataSource به طور قابل توجهی بالا رفته و علاوه بر حجم زیاد باعث وجود افزونگی می شود.مثلا شما در ۱۰ صفحه از سایت خود نیاز به ۱۰ Query مشابه دارید که اگر بخواهید از کنترل SqlDataSource استفاده کنید باید برای ۱۰ Query مشابه ۱۰ از کنترل SqlDataSource مختلف استفاده کنید.در حالیکه با Component این مشکل حل می شود.

۳-اگر مثلا در یک صفحه ۵ نوع Query جداگانه نیاز به اجرا داشته باشد شما مجبورید ۵ کنترل SqlDataSource در صفحه قرار دهید چون در بحث Query های نسبتا پیچیده هر کنترل SqlDataSource قادر است تنها یک Query اجرا کند.

برای حل چنین مشکلاتی باید به Component ها روی آورد.ولی نه به صورت قبل.Asp.NET 2 یک کنترل دیگر به مجموعه کنترل های منابع داده ای خود اضافه کرده به نام ObjectDataSource که با آن می توان با Component ها ارتباط برقرار کرد.

Object Data Source:

کنترلی که منعطف تر از SqlDataSource است.همانطور که از نامش بر می آید می تواند با اشیایی که شما ساخته اید ارتباط برقرار کند و عملیات مختلفی همچون Updateو.. را انجام دهد.

حالا این شی را از کجا تهیه کنیم؟حتما کلاسی را که در بخش DataComponent تعریف کردیم و در این بخش هم آن را اصلاح کردیم یادتون هست.ما به ان کلاس ۲ تابع کوچک اضافه کردیم به صورتی که کلاس کامل شده رادر زیر می بینید:

```
Imports Microsoft.VisualBasic
Imports EmpDetails
Imports System.Data
Imports System.Data.SqlClient
Imports System.Web.Configuration

Public Class EmpDB
    Private cs As String
```

```
        Public Sub New()
            cs                                                              =
WebConfigurationManager.ConnectionStrings("aaa").ConnectionString
        End Sub

        Public Sub New(ByVal ConnectionsStringName As String)
            cs                                                              =
WebConfigurationManager.ConnectionStrings(ConnectionsStringName).ConnectionSt
ring
        End Sub

        Public Function InsertEmployee(ByVal emp As EmpDetails) As String
            Dim con As New SqlConnection(cs)
            Dim cmd As New SqlCommand("InsertEmployee", con)
            cmd.CommandType = CommandType.StoredProcedure

            cmd.Parameters.Add("@emp_fname",   SqlDbType.VarChar,   30).Value   =
emp.Emp_FirstName
            cmd.Parameters.Add("@emp_lname",   SqlDbType.VarChar,   30).Value   =
emp.Emp_LastName
            cmd.Parameters.Add("@emp_job",    SqlDbType.VarChar,    30).Value    =
emp.Emp_job
            cmd.Parameters.Add("@emp_ID",    SqlDbType.NVarChar,    70).Value    =
(CStr(emp.Emp_ID) & Guid.NewGuid.ToString).ToString
            Try
                con.Open()
                cmd.ExecuteNonQuery()
                Return CStr(cmd.Parameters("@emp_ID").Value)
            Catch err As SqlException
                Throw New ApplicationException("Data error.")
            Finally
                con.Close()
            End Try
        End Function

        Public Sub Insert(ByVal EmployeeId As String, ByVal EmployeeFirstName
As String, ByVal EmployeeLastName As String, ByVal EmployeeJob As String)
            Dim    a    As    New    EmpDetails(EmployeeId,    EmployeeFirstName,
EmployeeLastName, EmployeeJob)
            InsertEmployee(a)
        End Sub

        Public Sub Update(ByVal EmployeeId As String, ByVal EmployeeFirstName
As String, ByVal EmployeeLastName As String, ByVal EmployeeJob As String)
            Dim    a    As    New    EmpDetails(EmployeeId,    EmployeeFirstName,
EmployeeLastName, EmployeeJob)
```

```
            UpdateEmployee(a)
        End Sub

        Public Function GetEmployee(ByVal EmployeeID As String) As EmpDetails
            Dim con As New SqlConnection(cs)
            Dim cmd As New SqlCommand("GetEmployee", con)
            cmd.CommandType = CommandType.StoredProcedure

            cmd.Parameters.Add("@emp_ID",    SqlDbType.NVarChar,    70).Value    =
    EmployeeID
            Try
                con.Open()
                Dim reader As SqlDataReader = cmd.ExecuteReader()
                reader.Read()
                Dim     emp     As     New     EmpDetails(CStr(reader("emp_ID")),
    CStr(reader("emp_firstname")),                  CStr(reader("emp_lastname")),
    CStr(reader("emp_job")))
                Return emp
            Catch err As SqlException
                Throw New ApplicationException(err.Message)
            Finally
                con.Close()
            End Try
        End Function

        Public Function GetAllEmployees() As DataTable
            Dim con As New SqlConnection(cs)
            Dim sql As String = "Select * from employees"
            Dim da As New SqlDataAdapter(sql, con)
            Dim ds As New DataSet()
            Try
                da.Fill(ds, "Employees")
                Return ds.Tables("Employees")
            Catch
                Throw New ApplicationException("Data error.")
            End Try
        End Function

        Public Sub UpdateEmployee(ByVal Emp As EmpDetails)
            Dim con As New SqlConnection(cs)
            Dim cmd As New SqlCommand("UpdateEmployee", con)
            cmd.CommandType = CommandType.StoredProcedure

            cmd.Parameters.Add("@empID",    SqlDbType.NVarChar,    70).Value    =
    Emp.Emp_ID
            cmd.Parameters.Add("@emp_fname",   SqlDbType.VarChar,   30).Value   =
    Emp.Emp_FirstName
```

```
            cmd.Parameters.Add("@emp_lname",   SqlDbType.VarChar,   30).Value   =
Emp.Emp_LastName
            cmd.Parameters.Add("@emp_jobs",   SqlDbType.VarChar,   30).Value   =
Emp.Emp_job

            Try
                con.Open()
                cmd.ExecuteNonQuery()
            Catch err As SqlException
                Throw New ApplicationException("Data error.")
            Finally
                con.Close()
            End Try

        End Sub

        Public Sub DeleteEmployee(ByVal EmployeeID As String)
            Dim con As New SqlConnection(cs)
            Dim cmd As New SqlCommand("DeleteEmployee", con)
            cmd.CommandType = CommandType.StoredProcedure

            cmd.Parameters.Add("@e_ID",    SqlDbType.NVarChar,    70).Value    =
EmployeeID
            Try
                con.Open()
                cmd.ExecuteNonQuery()
            Catch err As SqlException
                Throw New ApplicationException("Data error.")
            Finally
                con.Close()
            End Try

        End Sub

        Public Function CountEmployees() As Integer
            Dim con As New SqlConnection(cs)
            Dim cmd As New SqlCommand("CountEmployees", con)
            cmd.CommandType = CommandType.StoredProcedure
            Try
                con.Open()
                Dim a As Integer
                a = cmd.ExecuteScalar()
                Return a
            Catch err As SqlException
                Throw New ApplicationException("Data error.")
            Finally
                con.Close()
```

```
        End Try
    End Function

End Class
```

ما دو تابع Insert و Update رابه تابع اضافه کردیم و آرگومان ورودی تابع UpdateEmployee را به جای یک خط طولانی از متغیر های ورودی شامل نام و... یک متغیر از نوع EmpDetails گذاشتیم.و وظیفه ی ۲ تابع Insert و Update را تبدیل نوع به EmpDetails و همچنین صدا زدن توابع InsertEmployee و Updateemployee گذاشتیم.با این کار هم می توانید عمل درج و حذف رابا آرگومان ورودی از نوع Empdetails انجام دهید وهم رابا آرگومان های ورودی رشته ای از قبیل نام و...این کار انعطاف بیشتری به کلاس ما می دهد.در ضمن متد GetAllEmployee را نیز با مقدار برگشتی DataTable به کلاس خود اضافه کردیم.حالا می خواهیم با کنترل ObjectDataSource به این کلاس وصل شده و از تمام توابع آن استفاده کنیم.ولی قبل از آن بهتر است با مثال های ساده طرز کار با ObjectDataSource را یاد بگیریم و سپس به سراغ استفاده از این کلاس برویم.در ابتدا نحوه ی معرفی کنترل ObjectDataSource:

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
TypeName="EmpDB"...>
</asp:ObjectDataSource>
```

در این کد صفت TypeName همان شی یا کلاسی است که کنترل ObjectDataSource باید به ان وصل شود که در اینجا نامش EmpDB است.سپس اولین کاری که با این کنترل می توان کرد یک Select بسیار ساده است.برای این کار از صفت SelectMethod استفاده می کنیم و مقدارش را نام متدی که عمل Select را انجام می دهد و در داخل کلاس EmpDB است می گذاریم.که در این کلاس همانطور که می دانید نامش GetAllEmployees است:

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
TypeName="EmpDB" SelectMethod="GetAllEmployees"></asp:ObjectDataSource>
```

برای استفاده و نمایش این کنترل از یک GridView استفاده می کنیم:

```
<asp:GridView ID="GridView1" runat="server"
DataSourceID="ObjectDataSource1"></asp:GridView>
```

و یا اینکه از یک ListBox:

```
<asp:ListBox ID="ListBox1" runat="server" DataSourceID="ObjectDataSource1"
DataTextField="emp_ID"></asp:ListBox>
```

آیا به این فکرکردید که ممکن است از یک ConnectionStringName دیگراستفاده کنید؟ما در کلاس خود ۲ تابع سازنده داریم که یکی به طور پیش فرض نام aaa را به عنوان نام رشته ی اتصال در نظر گرفته ولی دومین تابع سازنده از ما نام رشته ی اتصال را می خواهد.مثلا فرض کنید تگ ConnectionStrings شما در فایل Web.Config به صورت زیر باشد:

```
    <connectionStrings>
      <add   name="aaa"   connectionString="Data   Source=.\SQLEXPRESS;Initial
Catalog=sql-test;Integrated Security=True;Min Pool Size=10"
        providerName="System.Data.SqlClient" />
      <add                  name="bbb"                  connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\ChelseaFC\App_Data\CDB.mdf;Integrated
Security=True;Connect Timeout=30;User Instance=True"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
```

می بینید که دو رشته ی اتصال یکی با نام aaa و دیگری با نام bbb که هر یک به پایگاه داده ی جداگانه متصل هستند در این تگ وجود دارند.اگر در برنامه های معمولی بود به سادگی می توانستید از این تابع سازنده استفاده کنید:

```
Dim k As New EmpDB("bbb")
```

ولی در کنترل ObjectDataSource چگونه می توانید از توابع سازنده استفاده کنید؟برای این کار از رویداد ObjectCreating کنترل ObjectDataSource استفاده می کنید:

```
   Protected Sub ObjectDataSource1_ObjectCreating(ByVal sender As Object,
ByVal e As System.Web.UI.WebControls.ObjectDataSourceEventArgs) Handles
ObjectDataSource1.ObjectCreating
 End Sub
```

سپس در داخل این رویداد از آرگومان e که از نوعObjectDataSourceEventArgs است استفاده کرده و متد InstanceObject ان را انتخاب کرده و برابر مقدار New شده ی تابع سازنده قرار دهید:

```
      Protected Sub ObjectDataSource1_ObjectCreating(ByVal sender As Object,
ByVal e As System.Web.UI.WebControls.ObjectDataSourceEventArgs) Handles
ObjectDataSource1.ObjectCreating
          e.ObjectInstance = New EmpDB("bbb")
    End Sub
```

پس تنها جایی که می توانید از توابع سازنده ی کلاس خود در کنترل ObjectDataSource استفاده و آنها را مقدار دهی کنید اینجاست.

در مثال بعدی می خواهیم Id ها را در یک DropDownList نمایش داده و سپس با انتخاب هر یک از آنها نام-نام خانوادگی-شغل کارمند در GridView نمایش داده شود.پس ابتدا یک کنترل ObjectDataSource را با یک SelectMethod مناسب ایجاد کرده :

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
TypeName="EmpDB" SelectMethod="GetAllEmployees"></asp:ObjectDataSource>
```

سپس تنها فیلد emp_Id آن رادر یک DropDownList نمایش می دهیم و صفت AutoPostBack این کنترل را برابر True قرار می دهیم:

```
<asp:DropDownList ID="DropDownList1" runat="server"
DataSourceID="ObjectDataSource1" DataTextField="emp_ID"
AutoPostBack="true"></asp:DropDownList>
```

سپس یک ObjectDataSource دیگر ایجاد می کنیم و از متد GetEmployee که با گرفتن Id کارند سایر مشخصات وی را می داد به عنوان SelectMethod استفاده می کنیم:

```
      <asp:ObjectDataSource        ID="ObjectDataSource2"        runat="server"
TypeName="EmpDB" SelectMethod="GetEmployee">
    </asp:ObjectDataSource>
```

از آنجایی که این مقدار Id یک مقدار خارجی بوده متوجه می شویم یک پارامتر است و چون در عبارتSelectMethod از آن استفاده می کنیم SelectParameter محسوب می شود وچون از کنترل DropDownList باید تامین شود از نوع ControlParameter:

```
      <asp:ObjectDataSource        ID="ObjectDataSource2"        runat="server"
TypeName="EmpDB" SelectMethod="GetEmployee">
      <SelectParameters>
      <asp:ControlParameter     ControlID="DropDownList1"     Name="EmployeeID"
Type="String" PropertyName="SelectedValue" />
      </SelectParameters>
    </asp:ObjectDataSource>
```

سپس نتایج را برای تنوع به جای GridView در DetailsView نمایش می دهیم:

```
          <asp:DetailsView          ID="DetailsView1"          runat="server"
DataSourceID="ObjectDataSource2">
        </asp:DetailsView>
```

حال نوبت به عملیات Update-Delete-Insert با استفاده از ObjectDataSource است ولی این بار نه با GridView بلکه با TextBox معمولی! یعنی اینکه پارامتر های ما به جای تگ <asp:Parameter> از تگ

<asp:ControlParameter> و از نوع TextBox تعیین می شود.در ضمن البته نمایش داده ها در GridView است. در زیر نمونه ای می بینید:

```
<asp:ControlParameter  ControlID=”TextBox1” Name=”aaa” Type=”String”
ControlProperty=”Text”>
```

همه ی اعمال Delete-Update-Insert-Select با یک کنترل ObjectDataSource انجام می شود.دقت کنید که ما می توانیم این اعمال را همزمان دریک کنترل ObjectDataSource انجام دهیم ولی مثلا ۲ نوع Select و یا دو نوع Update و... را نمی توانیم در یک کنترل ObjectDataSource قرار دهیم پس از هر کدام تنها یکی.حتی اگر نوع Query های ما معروف نباشد یعنی چیزی به جز Delete-Update-Insert-Select باشد برای هر یک از آن نوع Query باید یک کنترل ObjectDataSource مجزا داشته باشیم(البته کلک هایی وجود دارد که بتوان این کارها را انجام داد که در ادامه با یکی از این کلک ها آشنا می شوید)

خوب کار را از تعریف کنترل ObjectDataSource آغاز می کنیم:

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
TypeName="EmpDB"...>
```

در ابتدا صفت های SelectMethod –UpdateMethod-InsertMethod DeleteMethodرا مقداردهی می کنیم:

```
        <asp:ObjectDataSource     ID="ObjectDataSource1"     runat="server"
SelectMethod="GetAllEmployees"
          TypeName="EmpDB" UpdateMethod="Update"  InsertMethod="Insert"
DeleteMethod="DeleteEmployee">
```

در ابتدا به مقدار دهی پارامتر های متد UpdateMethod استفاده می کنیم.دقت کنید که نام پارامتر ها همان آرگومان ورودی توابع آن ها است:

```
            <UpdateParameters>
                <asp:ControlParameter                ControlID="TextBox1"
Name="EmployeeId" PropertyName="Text"
                    Type="String" />
                <asp:ControlParameter                ControlID="TextBox2"
Name="EmployeeFirstName" PropertyName="Text"
                    Type="String" />
                <asp:ControlParameter                ControlID="TextBox3"
Name="EmployeeLastName" PropertyName="Text"
                    Type="String" />
                <asp:ControlParameter                ControlID="TextBox4"
Name="EmployeeJob" PropertyName="Text"
                    Type="String" />
```

```
            </UpdateParameters>
```

طرز کار آن نیز به این ترتیب است که شما هر ٤ فیلد TextBox را پر کرده اگر Id شما در جدول وجود داشته باشد سایر فیلدها مقادیرش پاک شده و مقادیر جدیدی که شما وارد کردید به جای آن ها قرار می گیرد.

سپس پارامتر های InsertMethod را مقدار دهی می کنیم:

```
            <InsertParameters>
                <asp:ControlParameter                    ControlID="TextBox5"
Name="EmployeeId" PropertyName="Text"
                    Type="String" />
                <asp:ControlParameter                    ControlID="TextBox6"
Name="EmployeeFirstName" PropertyName="Text"
                    Type="String" />
                <asp:ControlParameter                    ControlID="TextBox7"
Name="EmployeeLastName" PropertyName="Text"
                    Type="String" />
                <asp:ControlParameter                    ControlID="TextBox8"
Name="EmployeeJob" PropertyName="Text"
                    Type="String" />
            </InsertParameters>
```

این جا هم مواردی که وارد می کنید به جز EmployeeId (که ویرایش شده و سپس وارد می شود) مستقیم وارد جدول پایگاه داده می شود.

سپس پارامتر های DeleteMethod:

```
            <DeleteParameters>
                <asp:ControlParameter                    ControlID="TextBox9"
Name="EmployeeId" PropertyName="Text"
                    Type="String" />
            </DeleteParameters>
```

که با گرفتن شناسه ی واحد , سطر مورد نظر را پاک می کند.پس کنترل ObjectDataSource ما به صورت زیر شد:

```
        <asp:ObjectDataSource     ID="ObjectDataSource1"     runat="server"
SelectMethod="GetAllEmployees"
            TypeName="EmpDB"  UpdateMethod="Update"    InsertMethod="Insert"
DeleteMethod="DeleteEmployee">
            <UpdateParameters>
                <asp:ControlParameter                    ControlID="TextBox1"
Name="EmployeeId" PropertyName="Text"
                    Type="String" />
                <asp:ControlParameter                    ControlID="TextBox2"
Name="EmployeeFirstName" PropertyName="Text"
                    Type="String" />
```

```
                    <asp:ControlParameter                    ControlID="TextBox3"
Name="EmployeeLastName" PropertyName="Text"
                        Type="String" />
                    <asp:ControlParameter                    ControlID="TextBox4"
Name="EmployeeJob" PropertyName="Text"
                        Type="String" />
                </UpdateParameters>
                <InsertParameters>
                    <asp:ControlParameter                    ControlID="TextBox5"
Name="EmployeeId" PropertyName="Text"
                        Type="String" />
                    <asp:ControlParameter                    ControlID="TextBox6"
Name="EmployeeFirstName" PropertyName="Text"
                        Type="String" />
                    <asp:ControlParameter                    ControlID="TextBox7"
Name="EmployeeLastName" PropertyName="Text"
                        Type="String" />
                    <asp:ControlParameter                    ControlID="TextBox8"
Name="EmployeeJob" PropertyName="Text"
                        Type="String" />
                </InsertParameters>
                <DeleteParameters>
                    <asp:ControlParameter                    ControlID="TextBox9"
Name="EmployeeId" PropertyName="Text"
                        Type="String" />
                </DeleteParameters>
            </asp:ObjectDataSource>
```

سپس یک GridView ایجاد کرده و برای نمایش داده ها (GetAllEmployees) منبع داده ی آن را ObjectDataSource قرار می دهیم:

```
            <asp:GridView              ID="GridView1"              runat="server"
DataSourceID="ObjectDataSource1" >
            </asp:GridView>
```

سپس برای مراحل Update وInsert و Delete به ترتیب TextBox هایی که نامشان در صفت ControlID آمده را قرار می دهیم:

```
            ----------------------------------------<br />
            id:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
            fname:<asp:TextBox  ID="TextBox2"  runat="server"></asp:TextBox><br
/>
            lname:<asp:TextBox  ID="TextBox3"  runat="server"></asp:TextBox><br
/>
            job:
            <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox><br />
            <asp:Button ID="Button1" runat="server" Text="Update" /><br />
            ----------------------------------------<br />
```

```
            id:<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox><br />
            fname:<asp:TextBox  ID="TextBox6"  runat="server"></asp:TextBox><br
/>
            lname:<asp:TextBox  ID="TextBox7"  runat="server"></asp:TextBox><br
/>
            job:<asp:TextBox ID="TextBox8" runat="server"></asp:TextBox><br />
            <asp:Button ID="Button2" runat="server" Text="Insert" /><br />
            ----------------------------------------<br />
            id:<asp:TextBox ID="TextBox9" runat="server"></asp:TextBox><br />
            <asp:Button ID="Button3" runat="server" Text="Delete" /><br />
          ----------------------------------------<br />
```

می بینید که ٤ TextBox برای عمل Update و ٤ TextBox برای عمل Insert و ١ TextBox برای عمل Delete در آن قرار دادیم.در بین آنها دکمه هایی را هم میبینید که مخصوص اجرای عملیات است که رویدادهایشان را به صورت زیر می نویسیم:

```
      Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
          ObjectDataSource1.Update()
      End Sub

      Protected Sub Button2_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button2.Click
          ObjectDataSource1.Insert()
      End Sub

      Protected Sub Button3_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button3.Click
          ObjectDataSource1.Delete()
    End Sub
```

کنترل ObjectDataSource دارای متد های Update-Select-Insert-Delete است که هر کدام را سر جایش صدا زدیم.پس اگر فیلد های TextBox مربوط به مثلا Insert را پر کرده و سپس روی دکمه ی Button2 کلیک کنید با اجرای ObjectDataSource1.Insert() عملیات درج انجام می شود به همین ترتیب برای سایر عملیات.

حال می ماند تابع CountEmployees.چون این تابع جزو Select محسوب می شود ما می خواهیم با یک کلک آن را در همان ObjectDataSource قبلی قرار دهیم.برای این کار من یک دکمه به صفحه اضافه می کنم که با زدن آن تعداد کارمندان را به ما بدهد:

```
<asp:Button ID="Button4" runat="server" Text="Take Count Of Records" />
```

سپس در رویداد کلیکش نام selectMethod را عوض می کنیم و آن را CountEmployees می گذاریم:

```
ObjectDataSource1.SelectMethod = "CountEmployees"
```

میبینید که توانستیم به selectMethod در کد پشت صحنه هم دسترسی داشته باشیم.در اصل به تمام ویژگی هایی که در تگ هر کنترل دسترسی داریم در کد پشت صحنه هم دسترسی خواهیم داشت.

سپس متد Select کنترل ObjectDataSource را صدا می زنیم تا selectMethod اجرا شود:

```
ObjectDataSource1.Select()
```

سپس حاصل را که تعداد کارمندان است در یک کنترل DetailsView نمایش می دهیم:

```
DetailsView1.DataSource = ObjectDataSource1
DetailsView1.DataBind()
```

و سریعا SelectMethod را به حالت اولش بر می گردانیم تا کار خراب نشود و همه چیز سر جایش بماند:

```
ObjectDataSource1.SelectMethod = "GetAllEmployees"
```

پس کد کلی به شکل زیر می شود:

```
Protected Sub Button4_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button4.Click
        ObjectDataSource1.SelectMethod = "CountEmployees"
        ObjectDataSource1.Select()
        DetailsView1.DataSource = ObjectDataSource1
        DetailsView1.DataBind()
        ObjectDataSource1.SelectMethod = "GetAllEmployees"
    End Sub
```

نکته ای که باید به آن توجه کنید بحث استثناهاست.شما همانطور که در کنترل SqlDataSource به طور مفصل بحث شد در اینجا هم می توانید دقیقا شبیه همان در رویداد های مختلف Exception Handling کنید.ولی بود و نبودش تاثیری ندارد زیرا ما در سورس کد کلاس خود از بلوک Tre...EndTry استفاده کردیم و همانجا مدیریت استثناها را انجام دادیم و دیگر نیازی به انجام آن در اینجا نیست.

خوب کار ما با کنترل ObjectDataSource تمام است.ولی یک نکته ی مهم در زمینه ی DataSourcecontrols باقی می ماند و آن هم در مثالی است که ما با SqlDataSource زدیم و یک DropDownList درآن به کاربردیم و گفتیم با کلیک بر روی هر آیتم ,آیتم های وابسته ی آن هم نمایش داده می شوند.اگر به آن مثال دقت کنید

می بینید وقتی صفحه بارگذاری می شود DropDownList یک آیتم انتخاب شده از قبل به صورت پیش فرض دارد که همان اولین آیتم موجود است و نسبت به آن موارد در Gridview نمایش داده شده اند.مثلا اگر لیست شهر ها در DropDownList به صورت زیر بود :

Karaj

Tehran

Qazvin

Zanjan

آنگاه برای اولین بار اگر صفحه بارگذاری شود خود به خود اسامی مشتریانی که در شهر کرج حساب دارند نمایش داده می شود واین بسیار غلط است.زیرا در سایت های حال حاضر موجود در وب,اگرDropDownList در آنها به کار رفته باشد هیچ آیتمی از آن به طور Default انتخاب نشده است و یک جمله مثلا PlzChoose و یا در اینجا Choose a City به عنوان اولین آیتم به کار رفته است.اگر آیتمی هم در آن انتخاب شده نباشدآنگاه SqlDataSource از شما ایراد NullReference می گیرد.برای حل این مشکل اگر کد آن مثال یادتان باشد کد صفحه را به صورت زیر بنویسید:

```
<asp:SqlDataSource ID="sqldatasource1" runat="server"
ConnectionString="<%$ ConnectionStrings:aaa %>"
ProviderName="System.Data.SqlClient" SelectCommand="SELECT
DISTINCT customer_city FROM customer">
</asp:SqlDataSource>
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
ConnectionString="<%$ ConnectionStrings:aaa %>"
ProviderName="System.Data.SqlClient" SelectCommand="SELECT
customer_name,customer_street From customer Where customer_city=@city" >
<SelectParameters>
<asp:ControlParameter ControlID="DropDownList1" Name="city"
PropertyName="SelectedValue" />
</SelectParameters>
</asp:SqlDataSource>
<asp:DropDownList ID="DropDownList1" runat="server"
DataTextField="customer_city" AutoPostBack="true">
</asp:DropDownList><br />
<asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource2">
</asp:GridView>
```

که تفاوت آن با قبلی این است که منبع داده ای DropDownList را حذف کردیم.

حالا ادامه ی کار در کد پشت صحنه.به رویداد Page_Load رفته و در آنجا منبع داده ای را برای DropDownList تعیین می کنیم و برای کارایی بیشتر آن را درون شرط PostBack قرارمی دهیم:

```
    If (Not Page.IsPostBack) Then
        DropDownList1.DataSource                                    =
sqldatasource1.Select(DataSourceSelectArguments.Empty)
        DropDownList1.DataBind()

   End If
```

منبع داده ای در اینجا تنها کنترل SqlDataSource نیست بلکه به این دلیل که ما هنگام لود شدن صفحه آیتم انتخاب شده نباید داشته باشیم پس از متد Select آن استفاده کرده و آرگومان ورودیش را DataSourceSelectArgument.Empty دادیم که این باعث می شود آیتم انتخاب شده از قبل درDropDownList وجود نداشته باشد و منبع داده در اولین گام خالی باشد(این به این معنی نیست که منبع داده ای خالی است چون وقتی ما SqlDataSource رابه عنوان منبع انتخاب می کنیم تمام آیتم های آن را انتخاب کردیم ولی متد Select (همان Select که در ObjectDataSource هم بیان شد و باعث اجرای فرمان Select مربوطه می شود)به طور موقتی مواردی را از منبع داده ای فیلتر می کند پس در کل به این معنی است که منبع داده ای ما SqlDataSource1 باشد ولی چون ما می دانیم آیتم اول ما Value ندارد برای اینکه با خطا مواجه نشویم مقدار اولیه اش را فیلتر کردیم(در اینجا با متد Empty آن را خالی کردیم)).سپس در ادامه ۲ عنصر به DropDownList اضافه می کنیم (شما می توانید هر چند تا که دلتان خواست اضافه کند ولی یادتان باشد به ازای هر یک باید مقداردهی لازم را انجام دهید-به ادامه ی مثال توجه کنید متوجه می شوید) و آیتم پیش فرض انتخاب شده را اندیس صفر می گذاریم یعنی هنگام بارگذاری صفحه آن آیتمی که به طور پیش فرض انتخاب شده باشد آیتم با اندیس ۰ است که قبل از تعیینش آن را مشخص کردیم(اندیس ۰ همان Choose a City است):

```
        DropDownList1.Items.Insert(0, "(Choose a City)")
        DropDownList1.Items.Insert(1, "(All Cities)")
    DropDownList1.SelectedIndex = 0
```

اولین عنصر هیچ چیزاست.یعنی یک جمله ی ساده که نه مقدارش و نه Valueش اثری در کار ندارد و اصلا به همین دلیل ما هنگام انتخاب منبع داده آن را به عنوان Value

برای Choose a City در اولین گام تهی در نظر گرفتیم. دومین آیتم هم که All Cities می باشد برای نمایش مشتریان تمامی شهر ها قرار دادیم تا وقتی انتخاب شد اسامی تمام مشتریان نمایش داده شود.حالا باید در پی تحریک یک رویداد باشیم که به آن وسیله, بتوانیم مقداردهی های لازم را برای این دو آیتمی که خودمان اضافه کردیم (برای سایر آیتم ها که شامل نام شهر ها می شود نیازی نیست کاری کنیم چون از قبل با پارامتر قرار دادن انها در کنترل SqlDataSource کار آنها را قبلا انجام داده ایم)انجام دهیم.چون بحث ما Select است پس رویداد Selecting کنترل SqlDataSource را فراخوانی کرده و در آن مشخص می کنیم که اگر Valueآیتم ارسالی (مقداری که به پارامتر City اختصاص داده شده چون مقدار پارامترCity برابر همان SelectedValue کنترل DropDownList است) برابر Choose a City بود کار را Cancel کن و ادامه نده (یعنی نیازی به نمایش داده در GridView نیست)ولی اگر All Cities بود یک Query جدید را اجرا کن و آن هم انتخاب تمام مشتریان است:

```
    Protected Sub SqlDataSource2_Selecting(ByVal sender As Object, ByVal e
As     System.Web.UI.WebControls.SqlDataSourceSelectingEventArgs)     Handles
SqlDataSource2.Selecting
        If  CStr(e.Command.Parameters("@city").Value)  =  "(Choose  a  City)"
Then
            e.Cancel = True
        ElseIf  CStr(e.Command.Parameters("@city").Value)  =  "(All  Cities)"
Then
            e.Command.CommandText = "SELECT * FROM customer"
        End If
    End Sub
```

این تکنیک بسیار مهم است پس هر جا خواستید از DropDownList برای بازیابی پایگاه داده استفاده کنید از این تکنیک استفاده کنید.قبلا اگر می خواستید به جای DropDownList از ListBox استفاده کنید با خطا مواجه می شدید زیرا کنترل ListBox آیتم پیش فرض انتخاب شده ندارد و شما با خطای NullReference مواجه می شدید.ولی در اینجا می توانید از ListBox هم استفاده کنید.

e.Cancel=True هم یعنی صر نظر از ادامه ی کار.

e.CommandText متد Execute ندارد زیرا رویدادی که در آن e.CommandText مقدار دهی شده (Selecting) خودش این Query را اجرا می کند

چون اصلا وظیفه ی آن در این رویداد این است که دستورات مربوط به Selcet Query را اجرا کند.

میبینید که هیچ چیز نداشت.اصل کار قرار دادن DataSourceSelectArgument.Empty به عنوان ورودی متد Selectکنترل SqlDataSource بود.با این نکته ی مهم بحث data Binding هم به انتها می رسد.

Rich Data Controls

تا اینجا زیاد با کنترل GridView کار کرده ایم ولی در این بخش می خواهیم کمی بیشتر روی این کنترل ثروتمد و منعطف کار کنیم.در نسخه ی قبلی Asp.NET کنترلی به نام DataGrid وجود داشت.GridView کامل شده ی این کنترل است که از قابلیت های بسیاری نسبت به آن برخوردار است.

حتما تا اینجا با تگ <Column> در آن آشنا شده اید.این تگ همانطور که از نامش مشخص است به منظور اضافه کردن ستون به کنترل GridView ایجاد شده.هر وقت خواستید ستونی تعریف کنید باید در داخل این تگ این کار را انجام بدهید:

```
<column>
‘Column Defenation
</column>
```

اسامی ستون هایی را که می توانید در این تگ تعریف کنید در زیر آمده است:

<BoundField>:با این تگ می توانید ستونهایی حاوی داده های استخراج شده از پایگاه داده را تعریف کنید.

<ButtonField>:با این تگ می توانید ستونهایی حاوی دکمه ایجاد کنید.

<CheckBoxField>:با این تگ می توانید ستونهایی حاوی CheckBox ایجاد کنید.

<CommandField>:که پیش از این هم استفاده شده بود برای انتخاب و یا ویرایش داده ها از آن استفاده می شود.

<HyperLinkField>:با این تگ می توانید ستونی از ابر لینک ها ایجاد کنید.

<ImageField>:با این تگ می توانید ستونی حاوی تصاویر ایجاد کنید.

<TemplateField>:با این تگ می توانید ستون هایی کاملا سفارشی ایجاد کنید.

در لبتدا می خواهیم کمی با تگ <BoundField> کار کنیم.همانطور که گفتیم با این تگ می توانید ستونهایی حاوی داده های استخراج شده از پایگاه داده را تعریف کنید و آنها را نمایش دهید.قبل از آشنایی با این تگ باید با یک صفت از کنترل GridView آشنا شوید و آن هم صفت AutoGenerateColumns می باشد.این صفت دو مقدار True وFalse می پذیرد که پیش فرض آن هم True است یعنی اگر این صفت را در کنترل GridView به کار نبرید هم مقدارش True خواهد بود.اگرمقدار این صفتTrue باشد به این معنی است که تمامی ستونهایی که در DataSource وجود دارد در GridView هم نمایش داده شوند.ما تا به حال هم همین کار را می کردیم.ولی اگر بخواهید این ستونها را به صورت سفارشی در کنترل GridView ظاهر کنید حتما باید این صفت برابر Falseبوده و سپس از تگ <BoundField> برای تعریف ستونهای سفارشی از DataSource بهره بگیرید.در زیر صفت AutoGenerateColumns برابر False قرار داده شده:

```
  <asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1"
AutoGenerateColumns="false" >
</asp:GridView>
```

همانطور که گفتیم تگ <BoundField> یکی از اجزای تگ <Column> است پس باید درون آن تعریف شودو برای تعریفش باید از ۲ صفت حتما استفاده کنید.اولین صفت DataField است که نام ستون جدول اصلی پایگاه داده است که ما می خواهیم به دلخواه آن را نمایش دهیم و دومی HeaderText که عنوان ستونی که در GridView نمایش داده می شود است.در زیر نمونه ای از تعریف تگ <BoundField> را می بینید:

```
  <asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1"
AutoGenerateColumns="false" >
    <Columns>
       <asp:BoundField DataField="Customer_name" HeaderText="Name" />
       <asp:BoundField DataField="Customer_city" HeaderText="City" />
       <asp:BoundField DataField="Customer_street" HeaderText="Street" />
  </Columns>
</asp:GridView>
```

در این کد ما ۳ ستون را از جدول مربوطه در پایگاه داده را نمایش دادیم.نام ستونهای اصلی در پایگاه داده برای نمایش را درصفت DataField و نامی دلخواه برای هر یک از آنها(نامی به جز نام خود ستون جدول اصلی) را در صفتHeaderText قرار دادیم.توجه کنید حتما صفت AutoGenerateColumns برابر False باشد در غیر این صورت

ستونهایی که ما با تگ <BoundField> تعریف کردیم مازاد به ستونهای اصلی نمایش داده می شوند.

همچنین شما می توانید با برنامه نویسی ستون و یا سطری را در هنگام نمایش به دلخواه حذف کنید.برای این کار کافی است از صفت Visible دو متد مهم کنترل GridView به نام های Columns و Rows استفاده کنید و آن را برابر False قرار دهید..در کد زیر در رویداد Page_Loadمن ستون سوم(عدد ۲ چون از ۰ شمارش می شود ۳ خوانده می شود) و سطر چهارم از جدول حذف می شود:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        GridView1.Columns(2).Visible = False
        GridView1.Rows(3).Visible = False
    End Sub
```

استفاده از <BoundField> و False قرار دادن AutoGenerateColumns باعث تسریع نمایش و سایت و همچنین افزایش سفارشی سازی می شود.

خصوصیات تگ <BoundField>:

این تگ خصوصیات بسیاری علاوه برDataField و HeaderText دارد که در زیربه اهم انها اشاره می کنیم:

DataFormatString:رشته های موجود در فیلدهای GridView رابه اشکال خاص درمی آورد.مثلا می توان باآن جلوی اعداد علامت % و یا $ قرار داد که جلوتر به ان اشاره می کنیم.

ApplyformatInEditMode:باعث میشود شکلی که در قسمت DataFormatString برای یک فیلد تعیین کردیم در هنگام Edit کردن GridView حفظ شود(True) یا نه(False).

HeaderImageUrl:آدرس تصویری را که دوست دارید به جای HeaderText استفاده شود می توان به ان داد.

ReadOnly:اگر True باشد امکان ایجاد تغییر را در ستون مربوطه نمی دهد.

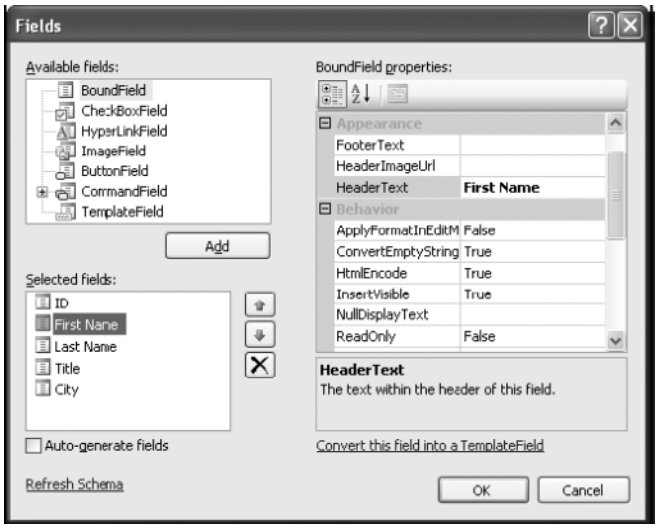
SortExpression:نام فیلد را گرفته و ستون را بر اساس آن مرتب می کند که در قسمت Sorting GridView به آن می پردازیم.

NullDisplayText:مقدار رشته ای می گیرد و آن را در فیلد هایی از جدول که مقدارشان Null است قرار می دهد.

ConvertEmptyStringToNull:اگر True باشد چنانچه در یک سطر از Gridview تغییر ایجاد کردید و در یک فیلد از آن مقداری ننوشتید و خواستید تغییرات را به پایگاه داده اعمال کنید این صفت باعث می شود به جای یک EmptyString مقدار Null به پایگاه داده منتقل شود.

HeaderStyle & ItemStyle:تغییرات در نمای ظاهری و سبک GridView و عناصر داخلی آن را ممکن می سازد.

تمام اعمال بالا در نمای Visual هم میسر است در شکل زیر به صورت Visual می توانید ستون هایی از انواع مختلف حذف و اضافه کنید:



برای دسترسی به این صفحه کافیست از قسمت Task کنترل GridView گزینه ی Edit Columns را انتخاب کنید.

کد زیر یک ستون حاوی دکمه به ستونهای GridViewما اضافه می کند:

```
<asp:ButtonField ButtonType=”Button” Text=”go” HeaderText=”Button
Field” />
```

خاصیت ButtonType نوع دکمه(لینک-تصویر-دکمه) را مشخص می کند. Text نوشته ی روی دکمه را تعیین می کند.

کد زیر به جای HeaderText یک تصویر قرار داده و با صفت ReadOnly اجازه ی تغییر را در آن ستون نمیدهد:

```
<asp:BoundField DataField="Customer_name"  HeaderImageUrl="ttt.jpg"
ReadOnly="True"  />
```

کد زیر یک TemplateField ایجاد می کند.برای ایجاد یک ستون کاملا سفارشی از تگ TemplateField استفاده می شود.در این تگ باید از تگ ItemTemplate استفاده کنید که آیتم های موجود در آن ستون را مشخص می کند:

```
  <asp:templatefield headertext="Address">
    <itemtemplate>
  <asp:label id="address"  runat="server" text= '<%# Eval("customer_city")+ "
" + Eval("customer_street") %>'/>
    </itemtemplate>
</asp:templatefield>
```

همانطور که میبینید ما می توانیم از هر کنترلی در داخل تگ ItemTemplate استفاده کنیم.ما در اینجا یک کنترل Label قرار دادیم و خاصیت Text آن را در داخل تگ <%#...> به تابع Eval سپردیم.

قبل از ادامه این را بگویم که به دو روش می توان در تگ <%#...> به عناصر و اجزای پیگاه داده دسترسی داشت:

۱-استفاده از تابع Eval و ذکر نام ستون به عنوان آرگومان ورودیش:

```
<%# Eval("column name")>
```

این تابع در اصل جزو شی DataBinder است (DataBinder.Eval("..."))که می توان آن را خلاصه به صورت Eval خالی هم در این تگ ذکر کرد.البته جلوتر فرق DataBinder.Eval با Eval خالی را می گوییم.

۲-استفاده از شی Container و صفت DataItem آن:

```
<%# Container.DataItem("column name")>
```

نکته ی مهم در اینجا این است که دو تگ بالا چون مربوط به پایگاه داده هستند حتما باید در داخل DataBound Controls مثلGridView-Repeater و.... استفاده شوند نه هر جایی.

عبارت Container.DataItem یک سطر کامل را بر می گرداند و اگر یک فیلد را به عنوان آرگومان ورودی به آن بدهیم در آن سطر مورد نظر یک ستون خاص را بر می گرداند.

این تابع نام ستونها را به عنوان ارگومان ورودی گرفته و مقادیر آن را در هر سطر نمایش می دهد. در اینجا با علامت + دو تابع Eval را به همراه فاصله به هم متصل کردیم.در اینجا در هر سطر تابع Eval مقدار موجود در ستون Customer_City را با

مقدار موجود در ستون Customer_Street جمع کرده و به عنوان Address آن را در GridVie نمایش می دهد.

اگر خواستید چند ستون سفارشی داشته باشید به همان تعداد از تگ TemplateField استفاده کنید که در هر تگ TemplateField یک تگ ItemTemplate هم قرارمی گیرد.جلوتر وقتی با GridView بیشتر آشنا شده ایم کار های پیشرفته ای با تگ TemplateField انجام می دهیم.

حالا کمی با FormatString کار می کنیم.می دانیم که برای این کار از خاصیت FormatStringاستفاده می کنیم.منبع داده ی ما حاوی یک جدول با چند ستون است که یکی از آنها مقدار موجودی حساب است.می خواهیم از یک حرف $ در کنار هر یک از این مقادیر در GridView استفاده کنیم.به این منظور کافیست عبارت {۰} را بشناسیم.این عبارت که قبلا هم کمی در موردش بحث شده بود یعنی مقدار داده ای که در هر فیلد وجود دارد.حالا اگرمی خواهید به ان علامت $ اضافه کنید با ید آن را به صورت {۰}$ بنویسید:

```
  <asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1"
AutoGenerateColumns="False">
    <Columns>
    <asp:BoundField DataField="account_number" HeaderText="Account Number"
/>
    <asp:BoundField DataField="branch_name" HeaderText="Branch Name" />
    <asp:BoundField DataField="balance" HeaderText="Balance"
DataFormatString="{0}$" />
    </Columns>
</asp:GridView>
```

کنار {۰} می توانید هر گونه رشته ای را بنویسید.تعریف DataFormatString تنها در داخل تگ <BoundField> ممکن است.

اگر دکمه ی Edit را هم در GridView نمایش دهید و از DataFormatStringهم استفاده کرده باشید.اگر روی Edit کلیک کنید مقداری فیلدی که در کنارش از DataFormatString استفاده شده بود حالا بدون DataFormatString ظاهر می شود.برای ظاهرشدن DataFormatString در هنگام Edit کافیست صفت ApplyformatInEditMode را برابر True تنظیم کنید.البته ما این کار را پیشنهاد نمی کنیم:

```
        <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="False">
```

```
            <Columns>
            <asp:BoundField    DataField="account_number"    HeaderText="Account
Number"    />
            <asp:BoundField  DataField="branch_name"  HeaderText="Branch  Name"
/>
            <asp:BoundField       DataField="balance"       HeaderText="Balance"
DataFormatString="{0}$" ApplyFormatInEditMode="true" />
            <asp:CommandField ShowEditButton="true"/>
            </Columns>
        </asp:GridView>
```

حالا می خواهیم از فرمت های آماده ی DataFormatString استفاده کنیم:

حتما می دانید {۰} یعنی چی.Asp.NET کنار این علامت را با یک سری علایم پر کرده و برای هر کدام از آنها هم یک معنی قرار داده.مثلا {۰:C} به معنی قرار گرفتن علامت $ در کنار رشته ی {۰} است.همانطور که گفتیم خصوصیت DataFormatString درون تگ BoundField تعریف می شود.پس اگر در قسمت DataFormatString یک ستون علامت {۰:C} را درج کنیم باید کنار تمام مقادیر آن ستون یک علامت $ یا یورو قرار گیرد ولی لازمه ی این کار False کردن HtmlEncode است چون True بودن آن باعث درست کار نکردن DataFormatString می شود:

```
<asp:BoundField DataField="customer_price" DataFormatString="{0:G}"
HtmlEncode="false" />
```

{۰:C} تنها آیتم نیست بلکه موارد زیرنیز هستند:

این جدول فرمت های عددی را نمایش می دهد(در عبارت {۰:C} به جای c می توانید حروف زیر را قرار دهید):

C	باعث قرار گرفتن علامت هزینه در کنار مقادیر فیلد می شود.
E	باعث تبدیل عدد به مقدار نمایی مثل ۱,۲۳E+۵۰ می شود.
F	باعث ایجاد اعشار در عدد گشته و مقدار پیش فرض آن ۲ رقم اعشار است.اگر خواستید بیشتر یا کمتر کنید جلوی f یک عدد وارد کنید که بیانگر تعداد رقم های اعشار است.{۰:F۳} یعنی عدد با ۳ رقم اعشار نمایش داده شود.اعشار برای اعداد صحیح ۰ می باشد.مثلا اگر این فرمت را روی عدد ۳ اعمال کنید به صورت ۳,۰۰۰ می شود.
P	عدد را در ۱۰۰ ضرب کرده و یک علامت % کنارش قرار می دهد.
G	عدد را به فرمت معمولی نمایش می دهد. مثلا ٦ را به صورت ٦ نمایش می دهد.
N	عدد را به فرمت عددی نمایش می دهد.مثلا ٦ را به صورت ٦,۰۰ نمایش می دهد.

جدول زیر حروف مربوط به زمان و تاریخ را نشان می دهد(به کوچک و بزرگ بودن حروف دقت شود):

d	نمایش دهنده ی نام روزی از ماه است.عددی بین ۱-۳۱.مثلا اگر امروز سوم باشد عدد ۳ را نمایش می دهد.
dd	نمایش دهنده ی نام روزی از ماه است.عددی بین ۱-۳۱.مثلا اگر امروز سوم باشد عدد ۰۳ را نمایش می دهد.و اگر سیزدهم

	باشد ۱۳ را نشان می دهد و فرقش با بالایی در اعداد تک رقمی با صفر و بدون صفر است.
ddd	نشان دهنده ی نام روزی از ماه به طور خلاصه است.مثلا Wed به معنای چهار شنبه و مخفف Wednesday است.
dddd	نشان دهنده ی نام روزی از ماه به طور کامل است.مثلا چهارشنبه به صورت Wednesday خواهد بود.
M	نمایش دهنده ی شماره ی ماه است.بین ۱-۱۲
MM	نمایش دهنده ی شماره ی ماه با پشتیبانی از اعداد تک رقمی.مثلا ماه سوم را به صورت ۰۳ نمایش می دهد.
MMM	نمایش دهنده ی نام ماه به طور خلاصه.
MMMM	نشان دهنده ی نام ماه به صورت کامل.
yy	نشان دهنده ی دو رقم اول سال است.مثلا ما در سال ۰۸ هستیم یعنی ۲۰۰۸.
yyyy	نشان دهنده ی نام کامل سال است.مثلا 2008.
hh یا h	نشان دهنده ی ساعت است.بین ۱-۱۲
HH یا H	نشان دهنده ی ساعت است.بین ۰-۲۳
m	نشان دهنده ی دقیقه است.بین ۰ تا ۵۹
mm	نشان دهنده ی دقیقه است.بین ۰ تا ۵۹ با در نظر گرفتن ۰ کنار تک رقمی ها.
s	نشان دهنده ی ثانیه است.بین ۰ تا ۵۹
ss	نشان دهنده ی ثانیه است بین ۰ تا ۵۹ با در نظر گرفتن ۰ کنار تک رقمی ها.
tt	مشخص کننده ی AM/PM بودن زمان است.
:	جدا کننده ی زمان.
/	جدا کننده ی تاریخ.

حالا که علایم مربوط به زمان وتاریخ را آموختید می رسیم به فرمت هایی که برای DateTime می توان در نظر گرفت(فرمت های زیر را باید جای علامت سوال در عبارت {؟:۰} قرار دهید):

M/d/yyyy	مثل ۱۰/۳۰/۲۰۰۵
dddd, MMMM dd, yyyy	Monday, January 30, 2005
dddd, MMMM dd, yyyy HH:mm tt	Monday, January 30,2005 10:00 AM
dddd, MMMM dd, yyyy HH:mm:ss tt	Monday, January 30, 2005 10:00:23 AM
yyyy-MM-dd HH:mm:ss	۲۰۰۵-01-۳۰ ۱۰:۰۰:۲۳
MMMM dd	January 30
M/d/yyyy HH:mm:ss tt	۱۰/۳۰/۲۰۰۲ ۱۰:۰۰:۲۳AM

البته این فرمتها استاندارد هستند ولی شما به هر گونه ای که دوست دارید می توانید انها را بچینید.فرمت های DateTime تنها روی فیلد هایی از جنس DateTime تاثیر دارند و اگر روی سایر مقادیر آنها را اعمال کنید از شما خطایی گرفته نمی شود ولی نتیجه ی درستی هم ندارد.مثلا اگر روی یک مقدار عددی فرمت M/d/yyyy را اعمال کنیم در خروجی عینا عبارت M/d/yyyy چاپ می شود.در زیر نمونه ای از فرمت تاریخ رامیبینید:

```
<asp:BoundField DataField="pic_DateTime" DataFormatString="{0:M/d/yyyy
HH:mm:ss tt}" HtmlEncode="false" />
```

نکته ی دیگر راجب به فرمتها این است که می توانیدآنها را در DataBinding و تگ <%#...> نیز به کار ببرید.تابع Eval می تواند ۲ آرگومان در یافت کند که اولی مثل همیشه نام ستون و دومی فرمت آن ستون به انتخای شماست.فرمت کلی به شکل زیر است:

<%# Eval(field expression,format expression)>

و یک مثال از آن:

```
<%#Eval("pic_DateTime", "{0:dddd ddd MMMM yy hh}")%>
```

نکته ی دیگر به کار گیری ApplyformatInEditMode در هنگام Edit است که باید مراقب باشید وقتی یک FormatString تعریف می کنیدApplyformatInEditMode را با False(پیش فرض است) تنظیم کنید تا هنگام ویرایش دچار مشکل نشوید.

و در نهایت آخرین نکته در این زمینه این است که FormatString را می توانید با تابع ToString در هر جایی به کاربرید.این تابع میتواند یک آرگومان از نوع رشته ای دریافت کند که فرمت دلخواه شما باشد ولی این بار بدون {} و ۰.مثلا اگر تا الان {۰:C} به کار می بردید اینجا C کفایت می کند چون Tostring تابعی است که از یک متد دیگر که خودش حاوی یک مقدار است:

```
Response.Write(DateTime.Now.ToString("dddd, MMMM dd, yyyy"))
```

خروجی این کد چیزی مثل شکل زیر است. :

Friday, August 24, 2007

.در حالیکه اگر تابع ToString را بدون آرگومان صدا بزنیم خروجی به صورت زیر می شد:

24/08/2007 08:49:47

حالا می خواهیم کمی با Style کار کنیم.

تگ های مختلفی در بحث Style وجود دارد:

۱-Header Style:فرمتی را برای Header مشخص می کند.

۲-Row Style:فرمتی برای سطر ها مشخص می کند.

۳-AlternatingRow Style:علاوه برRow Style تعیین می شود و یکی در میان و به صورت تناوبی فرمتی برای سطر ها مشخص مکی کند.

٤-SelectedRow Style:جلوتر با Selection در GridView آشنا می شویدواین تگ فرمتی را برای سطر انتخاب شده ایجاد می کند.

٥-EditRowSyle:فرمتی برای سطری که می خواهد Edit شود تعیین می کند.

هر یک از تگهای بالا دارای صفت های مشترکی از قبیل صفت های زیر هستند که از نامشان بر می آید وظیفه شان چیست:

Font-Bold----Font-Size---Font-Name---ForeColor---BackColor---Border Style و...

محل تعریف تگهای مربوط به Style در کد زیر مشخص شده است:

```
      <asp:GridView                ID="GridView1"                runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="false">
      Style Tags Here
      <Columns>
      Your Columns Defenation
      </asp:BoundField>
      <asp:BoundField               HeaderText="Customer               Street"
DataField="customer_street" />
      </Columns>
    </asp:GridView>
```

مثالی را دراین زمینه زیر میبینید:

```
      <asp:GridView                ID="GridView1"                runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="false">
      <HeaderStyle BackColor="Blue" ForeColor="White" Font-Bold="true" Font-
Names="Verdana" Font-Size="Medium" BorderStyle="Inset" />
      <RowStyle BackColor="dimgray" ForeColor="White" Font-Bold="true" Font-
Names="Verdana" Font-Size="Small" BorderStyle="Dashed" />
      <AlternatingRowStyle    BackColor="yellow"    ForeColor="black"    Font-
Bold="false" Font-Names="Verdana" Font-Size="Small" BorderStyle="Ridge" />
      <SelectedRowStyle  BackColor="Aqua"  ForeColor="Brown"  Font-Bold="true"
Font-Names="Verdana"          Font-Size="X-Small"          BorderColor="Salmon"
BorderStyle="Dashed" />
      <EditRowStyle  BackColor="CadetBlue"  ForeColor="White"  Font-Bold="true"
Font-Names="Verdana" Font-Size="Small" BorderStyle="Dashed" />
      <Columns>
  ....
      </Columns>
    </asp:GridView>
```

برای مشاهده ی عملکرد تگهای SelectedRow Style و EditRowSyle باید دکمه های Select و Edit را به GridView اضافه کنید:

```
      <Columns>
      <asp:CommandField ShowSelectButton="true"  ShowEditButton="true" />
      ...
    </Columns>
```

این استیل ها که تعریف کردیم استیل های عمومی بودند.یعنی در تمام جدول تاثیر دارند.مثلا اگر ما HeaderStyle تعریف کردیم روی Header تمامی ستون ها تاثیر داشت.ولی شما شاید بخواهید فرمت ستونی خاص را از سایر ستون ها جدا کنید.برای این

کار۲ روش وجود دارد.در اولین روش می توانید در داخل تگ <BoundField> آنها را اضافه کنید:

```
<asp:BoundField HeaderText="Customer Name" DataField="customer_name"
HeaderStyle-BorderColor="ButtonHighlight"
HeaderStyle-Font-Bold="false"
HeaderStyle-Font-Italic="true"
ItemStyle-BackColor="olive"/>
```

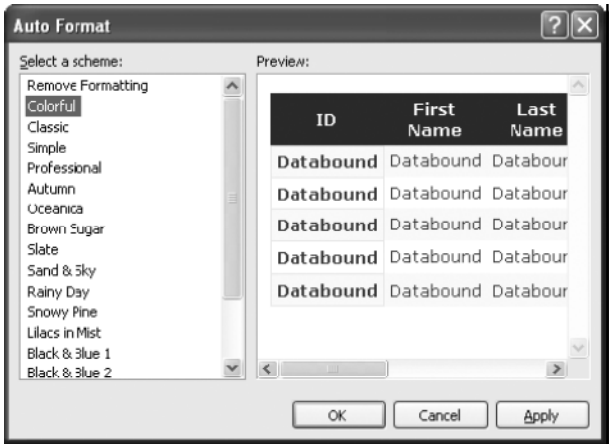
در این کد ما ستونی به نام Customer Name ایجاد کردیم که داده اش را از ستون customer_name جدول پایگاه داده می گیرد که خودش از خواصش استفاده کرده.از جمله ی آن HeadetStyle است که حاوی مواردی چون BorderColor-Font-Bold و ... است می توانید فرمت مربوط به Header این ستون خاص را مشخص کنید.همین طور با صفت ItemStyle می توانید فرمتی برای اجزای داده ای هر ستون تعیین کنید.

دومین روش هم در داخل تگ <BoundField> است ولی به صورت زیر:

```
<asp:BoundField  HeaderText="Customer City" DataField="customer_city">
<ItemStyle  .../>
<HeaderStyle ... />
<ControlStyle ... />
</asp:BoundField>
```

در این کد ما پس از تعریف ستونی با تگ <BoundField> آن را به کل نبستیم .این کار باعث شد تگهایی بین تگ <BoundField> ظاهر شوند از جمله تگ ItemStyle HeaderStyle و... که به هر کدام می توانید خاصیت هایی همچون BackColor-ForeColo-Fontو... را تعیین کرد.

اگر از کدنویسی خوشتان نمی آید می توانید از template/ های آماده ی VisualStudio استفاده کنید.برای این کار به قسمت Task کنترل GridView رفته و گزینه ی Edit Template را برگزینید:



در این صفحه می توانید فرمت دلخواه خود را انتخاب کنید.

حال همه نوع Style- گفته شد به جز Style که به یک سطر خاص اختصاص داده می شود.در صفحه ی AspX ما به ستونها ی GridViewدسترسی داریم ولی به سطرها خیر.برای دادن فرم به یک سطر خاص باید در Code-Behind و با برنامه نویسی اندک این کار را انجام داد.برای این کار یک مثال انجام می دهیم.در این مثال می خواهیم هر سطری که فیلد Customer_name آن برابر Reza است را بهش یک فرمت دلخواه بدهیم.برای این کار ابتدا منبع داده را مشخص می کنیم:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ProviderName="System.Data.SqlClient" ConnectionString="<%$
ConnectionStrings:aaa %>" SelectCommand="SELECT * FROM customer"
></asp:SqlDataSource>
```

سپس GridView را به صورت زیر تعریف می کنیم:

```
      <asp:GridView                ID="GridView1"                runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="false">
      <Columns>
      <asp:BoundField HeaderText="Customer Name" DataField="customer_name" />
      <asp:BoundField   HeaderText="Customer  City"  DataField="customer_city"
/>
      <asp:BoundField              HeaderText="Customer                Street"
DataField="customer_street" />
      </Columns>
    </asp:GridView>
```

سپس باید با یک رویداد به نام RowCreating از کنترل GridView آشنا شوید:

```
      Protected Sub GridView1_RowCreated(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewRowEventArgs) Handles GridView1.RowCreated

    End Sub
```

این رویداد هر بار که سطری در GridView ایجاد می شود تحریک می شود.ما می توانیم به این صورت از این رویداد استفاده کنیم که هر بار که سطری ساخته شد اگر فیلد Customer_name آن سطر برابرReza بود فرمت آن را در یک بدنه ی if تعیین کن.پس در ابتدا باید مشخص کنیم که ما می خواهیم شرط را روی فیلد Customer_name ایجاد کنیم.در کل در هر بار تحریک رویداد RowCreating از کنترل GridView شما می توانید به اطلاعات ستون های آن سطر دسترسی داشته باشید.این هم یادتان باشد که خاصیتDataItem آیتم داده های یک شی است مثلا در هر سطر یک جدول آیتم های متعددی وجود دارند که به مجموعه ی انها DataItem می گوییم و برای استفاده از هر یک(ستون) باید اندیسی به DataItem بدهیم.

از شی DataBinder استفاده می کنیم.این شی جز اشیایی است که باعث دسترسی به عناصر پایگاه (و یا داده های منابع داده های دیگر) داده در کد پشت صحنه و یا تگ <...#%> می شود.یک متد مهم این شی Eval است که با آن در داخل تگ <...#%> آشنا شدید و دیدید که وظیفه ی نمایش داده های پایگاه داده را بر اساس نام ستون دریافتی بر عهده داشت.به همین دلیل هم ما از این متد استفاده کردیم.

حالا هم نیاز به آن داریم ولی نه تنها در تگ <...#%> بلکه در کد پشت صحنه.تعریف آن به صورت زیر است:

```
DataBinder.Eval(container as Object,Expression as String,format as String)
```

همانطور که می بینید این شی 3 آرگومان ورودی دارد.اولی که قبلا هم در متد Eval خالی از آن استفاده کردیم یک شی بود که مرجعی برای نمایش است.ما تا به حال نام ستونی که مقدارش را می خواهیم نمایش دهیم به این مقدار می دادیم ولی حالا می خواهیم به یک دید دیگر به آن نگاه کنیم.سومین آرگومان هم فرمتی برای نمایش داده است و آرگومانی Optional است.ولی آرگومان دوم چیست؟این آرگومان می تواند Property نیز باشد.مثلا اگر شی مرجع من یک SqlConnection باشد (دیگر نام ستون جهت نمایش نباشد) می توانم با مقداردهی این آرگومان با "State" به Value مقدار SqlConnection.State دسترسی داشته باشم.پس این دسترسی به Value یک ویژگی از شی مرجع , توسط متد DataBinder.Eval امکان پذیر است.ولی این امر چگونه میسر است؟برای این کار به مثال زیر دقت کنید.

در این مثال می خواهیم چند تا از ویژگی های یک SqlConnection را در داخل تگ <...#%> نمایش دهم.ولی قبل از آن باید شی SqlConnection را تعریف کنم.من این کار را در رویداد Page_Load می نویسم:

```
Dim k As New
SqlConnection(ConfigurationManager.ConnectionStrings("aaa").ConnectionString)
```

سپس برای دسترسی به ویژگی هایی مثل ServerVesion باید Connection باز باشد:

```
k.Open()
```

من می خواهم از تگ <...#%> در داخل یک FormView استفاده کنم.ولی قبل از تعریف آن باید مشخص کنم آرگومان های ورودی متد DataBinder.Eval چیست.اول باید منبع داده ای مناسبی برای کنترل FormView پیدا کنم.پیش از این شما مثلا می گفتید formview1.datasource=sqldatasource1 و یا یک جدول را به آن می دادید formview1.datasource=dataTable(0): و...پس مرجع شما یک شی کلی مثل یک جدول بود که با متد DataBinder.Eval سطر به سطر آن جدول(بسته به نام ستونی که به آرگومان ورودی متد Eval می دادید), مقادیرش نمایش داده می شد.در اینجا من مرجعی جز یک شی SqlConnection ندارم.و از طرفی در صورتی که formview1.datasource=sqlconnection باشد با خطا مواجه می شوید چون شی sqlconnection یک شی ساده و غیر جدولی است مانند بسیاری از اشیا موجود در .NET. مانند مثلا DirectoryInfo و... و مورد قبول منبع داده ای کنترل های RichData نخواهد بود.پس من باید به دنبال یک مرجع باشم که formview1.datasource آن را بپذیرد و بتوانم از هر یک از سطر های آن مرجع در متد DataBinder.Eval استفاده کنم.یک نوع دیگر داده ای(به جز SqlDataSource) که صفت DataSource کنترل های FormView و... می پذیرد داده های کلکسیونی (Icollection) است. نمونه ای از این داده ها را می توان در HashTable و یا ArrayList نیز یافت.پس من می توانم مثلا بگویم :formview1.datasource=ArrayList. از طرف دیگر چون شی ArrayList هر شی را در خود می تواند ذخیره کند من می توانم شی sqlconnection که تعریف کرده بودم را در آن ذخیره کنم:

```
  Dim arr As New ArrayList
arr.Add(k)
```

پس حالا من یک مرجع کلی دارم(مثل یک جدول) از نوع ArrayList به نام arr که در متد DataBinder.Eval می توانم به ترتیب به هر یک از آیتم هایش (که قبلا سطر های یک جدول بود) دسترسی داشته باشم.پس کافی است منبع داده ای کنترل formview را همان ArrayList در نظر بگیرم:

```
  FormView1.DataSource = arr
FormView1.DataBind()
```

خوب حالا کار ما در پشت صحنه تمام شد و همانطور که می دانید کنترل formview حاوی AutoGenerateColumns نیست و خودمان باید دستی آن را Bind کنیم و این کار را با تگ ItemTemplate در آن انجام می دهیم:

```
  <asp:FormView ID="FormView1" runat="server">
  <ItemTemplate>
    Items To Be Representation...
  </ItemTemplate>
</asp:FormView>
```

حالا می رسیم به استفاده از متد Eval شی DataBinder به گونه ای متفاوت با آنچه تا بحال استفاده کرده بودیم.همانطور که گفتیم این متد ۳ آرگومان دارد.اولی مرجعی برای نمایش است.این مرجع را می توانید با شی Container.DataItem مقدار دهی کنید.همانطور که قبلا هم گفتیم این شی یک سطر از یک جدول پایگاه داده را برمی گرداند و اگر آرگومان ورودی دوم متد Eval شی DataBinder را نام ستونی خاص بدهیم در نهایت مقدار آن ستون از سطر مرجع را در خروجی نمایش می دهد(البته اگر شی Container.DataItem را تنها و بدون متد Eval به کار ببریم می توان به ورودی آن نام ستون یا اندیس آن را داد تا یک راست آن را نمایش دهد مثلا <%# (Container.DataItem(3)%>.حالا دیگر جدولی در کار نیست بلکه یک کلکسیون که در اینجا تنها یک سطر دارد و آن سطر نیز حاوی یک شی SqlConnection است.پس برای دسترسی به این شی از Container.DataItem استفاده کردیم.با این کار من به آن شی دسترسی دارم.آرگومان بعدی هم یک ویژگی از عنصری است که در آرگومان اول مشخص کردیم.و چون آن عنصر شی SqlConnection بود, می توانم نام ویژگی مربوط به آن را در این آرگومان ذکر کنم مثلا:

```
DataBinder.Eval(Container.DataItem, "ServerVersion")
```

حتما یادتان هست که وقتی یک سطر از یک جدول پایگاه داده را به عنوان Container مشخص می کردید به مقادیر تمامی ستون های آن دسترسی داشتید و می

توانستید به دلخواه عمل کنید و بعضی را نمایش دهید و بعضی را نه.در اینجا هم به همین صورت است یعنی وقتی یک شی را به عنوان مرجع نمایش در نظر می گیرید به تمام ویژگی های آن شی نیز دسترسی دارید و به دلخواه می توانید هر کدام را که دوست داشتید نمایش دهید.

پس کد کلی FormView به صورت زیر شد که من تنها ٤ ویژگی آن را نمایش دادم:

```
<asp:FormView ID="FormView1" runat="server">
 <ItemTemplate>
 Server Version :
 <%#DataBinder.Eval(Container.DataItem, "ServerVersion")%>
 <br />
 State :
 <%#DataBinder.Eval(Container.DataItem, "State")%>
 <br />
 Work Station Identifyer:
 <%#DataBinder.Eval(Container.DataItem, "WorkstationId")%>
 <br />
 Packet Size:
 <%#DataBinder.Eval(Container.DataItem, "PacketSize")%>
 </ItemTemplate>
</asp:FormView>
```

خروجی این کد لیست مقادیر وِیژگی های یک SqlConnection است.حال شما با متد Add شی ArrayList می توانید SqlConnectionهای دیگری نیز به آن اضافه کنید زیرا در حقیقت مثال ما مثل یک جدول پایگاه داده بود با تنها یک سطر, متناظر با یک کلکسیون با یک آیتم.شما می توانید هر شی دیگری نیز به جای SqlConnection در ArrayList قرار دهید و سپس از مقدار ویژگی های مخصوص آن استفاده کنید ولی اشیایی که در ArrayList قرار می دهید باید همجنس باشند چون در غیر این صورت ویژگی های آنها برای نمایش مخلوط در نظر گرفته شده و با خطای عدم مالکیت فلان ویژگی برای فلان شی روبرو می شوید.

نکته ی مهم این است که اگر از متد Eval به صورت تنها و بدون مشتق کردن از شی DataBinder استفاده می کردید آنگاه تنها دو آرگومان ورودی داشتید یکی Container as Object و دیگری Format as String و دیگر آرگومانی به نام Expression as String برای نسبت دادن ویژگی نداشتید.البته در متد DataBinder.Eval تنها آرگومان Container as Object و Expression as String اجباری بوده و format as

String آرگومانی Optional خواهد بود.پس فرق Eval خالی و DataBinder.Eval را فهمیدید.

البته راه دیگر نیز برای Bind کردن مقدار Property های یک شی در صفحات aspx نیز وجود دارد و آن هم استفاده از تگ BoundField در GridView است.همانطور که می دانید این تگ قادر است یک ستون مجزا در GridVeiw ایجاد کند.این تگ حاوی یک خاصیت بسیار مهم به عنوان DataField است که با دادن نام ستونی از جدول پایگاه داده, مقادیر آن را در سطری مجزا نمایش دهد.حال فرض منید به جای یک جدول پایگاه داده, یک کلکسیون داریم.و به جای هر سطر از جدول پایگاه داده یک شی SqlConnection.پس می توانیم به جای دادن نام ستونی که می خواهیم نمایشش دهیم به صفت DataFiled , نام ویژگی از شی SqlConnection را که می خواهیم نمایشش دهیم را به DataFiled بدهیم.برای مثال به فرم زیر منبع داده ای کنترل GridView را تعریف کنید(arr همان ArrayListی است که در بالا تعریف کرده بودیم):

```
  GridView1.DataSource = arr
GridView1.DataBind()
```

حال کافی است GridView1 ا به صورت زیر تعریف کنیم:

```
          <asp:GridView              runat="server"              ID="GridView1"
AutoGenerateColumns="false">
          <Columns>
          <asp:BoundField     DataField="ServerVersion"     HeaderText="Server
Version" />
          <asp:BoundField DataField="State" HeaderText="State" />
          <asp:BoundField  DataField="WorkstationId"  HeaderText="Work  Station
Identifyer" />
          <asp:BoundField DataField="PacketSize" HeaderText="Packet Size" />
          </Columns>
        </asp:GridView>
```

می بینید که نام ویژگی ها ی شی SqlConnection را به صفت DataField دادم.البته می توان از آنها در تگی همچون ButtonFiled استفاده کرد ولی تگ ButtonFiled حاوی ویژگی DataField نیست و به جای آن باید از DataTextField استفاده کرد.نکته ی آخر هم در مورد مقدار دهی به صفت DataKeyNames است.هنگامی که منبع ما یک جدول از پایگاه داده بود, این صفت را با نام یک ستون خاص(که اغلب Primary Key) بود مقدار دهی می گردیم و حالا هم می توانیم آن را با یک ویژگی خاص از شی مرجع(که در اینجا SqlConnection) است مقدار دهی کنیم مثلا صفت ConnectionStrings.

اگر در داخل کنترل هایی مثل Repeater بخواهیم داده نمایش دهیم از عبارت زیر استفاده می کنیم:

<%# Container.DataItem("column name")>

اینجا هم شی container وجود دارد.پس شی container شی است که یک مرجع برای داده هاست و به وسیله ی آن منبع داده می تواند داده های خود را دراختیار ما قرار دهد.و اگر مثل کد بالا از آن به تنهایی و بدون متد Eval استفاده کنیم آنگاه می تواند آرگومان ورودی بگیرد و آن هم نام یا اندیس ستونی است که می تواند آن را نمایش دهد ولی اگر مثل کد زیر آن را به عنوان آرگومان ورودی متد Eval در نظر بگیریم آنگاه نمی تواند آرگومانی بپذیرد و تنها یک مرجع برای داده های یک سطر از جدول پایگاه داده است و برای نمایش ستونی خاص از آن باید به رگومان بعدی متد Eval از شی DataBinder استفاده کنیم).پس به این دلیل که container یک مرجع برای شی بوده و در تابع بالا به آن نیاز داریم می توان از آرگومان e که اجزای کنترل را به ما می دهد به عنوان راهی برای رسیدن به container استفاده کنیم.برای این کار از صفتRow و سپس DataItemآن استفاده می کنیم چون تابع Eval برای بازیابی یک ستون خاص از یک سطر ایجاد شده پس در قسمت container باید سطر باشد و برای رسیدن به سطر از e.Row.DataItem استفاده می کنیم:

```
DataBinder.Eval(e.Row.DataItem, "customer_name")
```

پس از این به بعد هر جا می خواستید به عناصر پایگاه داده در کد پشت صحنه دسترسی داشته باشید حتما باید از شی DataBinder و متد Eval آن استفاده کنیم البته درررویداد مناسب ولی اگر در داخل صفحه باشد Eval خالی کفایت می کند (گر چه در آنجا هم همانطور که پیش از این گفته شد می توان از DataBinder و متد Eval آن ااستفاده کرد فقط در صورت لزوم مثل مثال های بالا).اولین آرگومان ورودی آن شی مرجع داده است که در اینجا هر سطر جدولی از پایگاه داده است که به عنوان منبع داده ای Gridview در نظر گرفته شده و ما باید از Row.DataItem آن استفاده کنیم تا به آیتم های سطری که در رویداد RowCreating ساخته می شود دسترسی داشته باشیم و برای استفاده از Row.DataItem یا به عبارت دیگر دسترسی به یک سطر باید مرجعی برایش داشته باشیم که در اینجا آرگومان ارسالی e خواهد بود زیرا اگر به رویداد

RowCreating یک نگاهی بیندازید e در آنجا از جنس GridViewRow خواهد بود پس در حالت کلی مرجع سطر e است که برای دسترسی به خود سطر از Row.DataItem استفاده کردیم. پس یادتان باشد در اینجا آرگومان اول که Container است باید حاوی یک سطر خاص باشد تا با آرگومان بعدی تابع Eval ستون آن هم مشخص شود.حال کافی است نامی که از این متدها استخراج کردیم را در جایی ذخیره کنیم:

```
Dim title As String = DataBinder.Eval(e.Row.DataItem,
"customer_name")
```

سپس آن را چک کنیم که اگر Reza بود آنگاه در فرمت و شکل آن تغییر ایجاد کنیم:

```
If title = "reza" Then
    e.Row.ForeColor = Drawing.Color.DarkBlue
    e.Row.BackColor = Drawing.Color.HotPink
End If
```

در اینجا e همان مرجع سطر ساخته شده است.پس برای تغییر در فرمت سطر,آن رابابد تغییر دهیم.و چون e همان مرجع سطر ساخته شده است پس حاوی تمام متد های مربوط به Style آن هست.یادتان باشد e مرجع سطر ساخته شده است نه خود سطر.برای دسترسی به سطر باید از متد Row آن استفاده کنید همانطور که در بالا از e.Row.DataItem استفاده شد.

ما نمی توانستیم به جای e از نام کنترل GridView استفاده کنیم چون e است که موقعیت سطر ساخته شده را می داند و از نوع GridViewRowEventArgs است.

حالا اگر برنامه را اجرا کنید می بینید که سطری که فیلد custome_name آن Reza باشد با بقیه از نظر رنگ فرق دارد.

این مثالی که برای شما زدیم اهمیت ویژه ای دارد.تمایز یک سطر از سطر های دیگر در یک جدول بسیار بحث پر کاربردی است.

شاید با خود بگویید خوب ما این کار را به صورت دستی نیز می توانیم انجام دهیم.در زیر ForeColorسطر اول را قهوه ای کردیم:

```
GridView1.Rows.Item(0).ForeColor = Drawing.Color.Brown
```

این کار کاملا دستی و استاتیک است.آیا شما همیشه اندیس سطری که نامش رضا است را می دانید؟

GridView Selection:

عمل Select هم علاوه بر Update=Delete و... در GridView وجود دارد.و باعث تحریک رویداد SelectedIndexChanged می شود:

```
    Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles GridView1.SelectedIndexChanged

    End Sub
```

اندیس آیتم انتخاب شده در صفت SelectedIndex کنترل موجود است:

```
    Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles GridView1.SelectedIndexChanged
        Dim i As Integer = GridView1.SelectedIndex

    End Sub
```

برای ایجاد عمل Select باید این دکمه را در GridView نمایش دهید. ما در زیر این کار را کردیم ولی نوع دکمه را Image در نظر گرفتیم:

```
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ProviderName="System.Data.SqlClient" ConnectionString="<%$
ConnectionStrings:aaa %>" SelectCommand="SELECT * FROM customer"
></asp:SqlDataSource>
        <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1" DataKeyNames="customer_name">
        <Columns>
        <asp:CommandField ShowSelectButton="true" ButtonType="Image"
SelectImageUrl="ttt.jpg" />
        </Columns>
      </asp:GridView>
```

با اجرای این کد,دکمه ی Select درکنار تمامی سطر های کنترل GridView نمایش داده می شود و با فشردن هر یک از آنها رویداد SelectedIndexChanged تحریک می شود.اینجاست که اگر تگ SelectedRow Style را تعریف کنید فرمت وشکل سطر انتخاب شده تغییر می کند.حالا می خواهیم پاسخی به رویداد SelectedIndexChanged کنترل GridView بالا که تمامی ستون های جدول Customer را نمایش می دهد بدهیم.همانطور که می دانید جدول Customer بر حسب ستون Customer_name با جدول Depositor رابطه دارد.جدول Depositor هم حاوی نام مشتری و شماره ی حساب وی است.ما می خواهیم اگر کسی در هر سطری از GridView بالا کلیک کند Account_number وی از جدول Depositor برایش نمایش داده شود.برای این کار نیاز به استخراج مقدار ستونی از جدول Customer هستیم که کلید اصلی

است(Customer_name) تا با آن بتوانیم در جدول Depositor به جستجوی شماره ی حسابش باشیم.

قبل از آن بهتر است پاسخ های ساده تری به رویداد SelectedIndexChanged بدهیم:

با کد زیر می توانید به Value صفت DataKeyName کنترل GridView (البته سطری که انتخاب شده)دسترسی داشته باشید.ولی دقت کنید که نام ستونی که اینجا وارد کردید دقیقا همانی باشد که در تگ GridView تعریف کردید:

```
        Dim namea As String =
GridView1.SelectedDataKey.Values("customer_name")
```

اگر می خواهید بدون استفاده از عمل یا دکمه ی select به صفت DataKeyName کنترل GridView دسترسی داشته باشید می توانید از صفت DataKeys کنترل GridView استفاده کنید و اندیس سطر مورد نظر را که می خواهید به DataKeysش دسترسی داشته باشید را به عنوان آرگومان ورودی صفت DataKeys معرفی کنید.یادتان باشد استفاده از این متد تنها در رویداد های مرتبط با GridView امکان پذیر است و وقتی می توانید از آن استفاده کنید که حتما صفت DataKeyName درGridView تعریف شده باشد .مثلا در زیر ما آن را در رویداد On Bound قرار دادیم.این رویداد به هنگام Bind کردن GridView و نمایش داده در آن رخ می دهد :

```
Response.Write(GridView1.DataKeys(1).Value.ToString())
```

با کد های زیر می توانید به داده های سطری که انتخاب شده دسترسی داشته باشید:

```
        Dim k As String = GridView1.SelectedRow.Cells(1).Text
        Dim kk As String = GridView1.SelectedRow.Cells(2).Text
      Dim kkk As String = GridView1.SelectedRow.Cells(3).Text
```

Cell یعنی سلول یا فیلد.که ارگومان ورودیش(اندیس ستون) مشخص می کند کدام ستون از سطری که انتخاب کردیم.

حال به مثال خود بر گردیم.ابتدا query مربوط به پیوند دو جدول Customer و Account را مشخص می کنیم:

```
SELECT depositor.account_number
FROM customer INNER JOIN depositor
ON customer.customer_name = depositor.customer_name
WHERE (customer.customer_name = @cname)"
```

در سطر اول که مقدار برگشتی را مشخص کردیم که همان شماره حساب است.و همانطور که می دانید در Query های sql ابتدا Select سپس Where و در آخرFrom

چک می شود.پس اینگونه می گوییم که از جدول Customer آن نامی را که مقدارش برابر پارامتر cname است در شرط Where قرار بده.سپس در جلوی From با کلمه ی کلیدی INNER JOIN بین دو جدول رابطه بر قرار کردیم که این رابطه بر اساس Customer_name های مشترک بین دو جدول بوده:

Table1 INNER JOIN Table 2 On 'Same Columns'

Query بالا رامی توان به صورت زیر هم نوشت که ابتدا دو جدول را در قسمت From در هم ضرب کارتزین کرده و سپس سطرهای نام برابر را جدا کرده و آنهایی که برابر پارامتر cname بوده را بر می گزینند:

```
SELECT depositor.account_number
FROM customer,depositor
WHERE (customer.customer_name=depositor.customer_name) AND
(customer.customer_name = @cname)
```

پس این query را در یک کنترل SqlDataSource به کار می بریم.البته همراه پارامتر cname:

```
  <asp:SqlDataSource            ID="SqlDataSource2"            runat="server"
ProviderName="System.Data.SqlClient"
  ConnectionString="<%$ ConnectionStrings:aaa %>"
  SelectCommand="SELECT depositor.account_number FROM customer INNER JOIN
depositor ON customer.customer_name = depositor.customer_name WHERE
(customer.customer_name = @cname)">
   <SelectParameters>
     <asp:ControlParameter
  ControlID="GridView1"
  Name="cname"
  PropertyName="SelectedDataKey.Values(customer_name) " />
    </SelectParameters>
</asp:SqlDataSource>
```

نکته ی مورد توجه در اینجا پارامتری است که در اینجا به کاربردیم.این پارامتر از یک GridView تامین می شود.و PropertyName آن SelectedDataKey.Values(customer_name) است.این صفت وقتی قابل دسترسی است که ما مقدار DataKeyName را در تگ GridView مقداردهی کرده باشیم.ما این بازیابی را قبل از این هم در کد پشت صحنه انجام داده بودیم.در غیر این صورت از عبارت زیر به جای آن استفاده می کنیم.

که با توجه به توضیحاتی که دادیم می توان به جای آن از عبارت زیر نیز استفاده کرد:

```
PropertyName="SelectedRow.Cells(1).Text "
```

درکد کنترل SqlDataSource بالا یادتان باشد در قسمت SelectedDataKey.Values(customer_name) نباید customer_name را در داخل "" قرار دهید چون رشته جدا تلقی شده و از شما خطا گرفته می شود.باید آن ر بدون "" به کاربرید(مثل بالا) و یا اینکه به صورت زیر در داخل ":

```
PropertyName="SelectedDataKey.Values('customer_name') "
```

و در نهایت برای نمایش حاصل از GridView استفاده کردیم:

```
<asp:GridView ID="GridView2" runat="server" DataSourceID="SqlDataSource2"
EmptyDataText="No Account"></asp:GridView>
```

در مورد صفت EmptyDataText بگویم که بعضی از افراد در پایگاه داده ی ما فقط وام دارند و نه حساب و برای اینکه فیلذ خالی نشان دادخ نشود از عبارت No Account در آن استفاده کردیم.شاید از خود بپرسید که پس صفت NulldisplayText چه شد؟ما گفتیم اگر از تگ <BoundField> در نمایش ستونها استفاده کنید یک خصوصیت به نام NulldisplayText هم دارد که در فیلدهای خالی آن ستون خاص مقدار رشته ای دلخواه شما را می نویسد:

```
  <Columns>
  <asp:BoundField  DataField="account_number"  HeaderText="ACcount  Number"
NullDisplayText="No Account" />
</Columns>
```

.در حالی که صفت EmptyDataText همین کار را می کند ولی نه برای یک ستون خاص بلکه برای تمامی ستونها .

پس نکته ی مهم در این مثال این بود که ما پارامتر برای یک Query را از یک GridView تامین کردیم.و نکته ی دیگر تنظیمات در GridView بود.ما دو حالت برای انواع تنظیمات فرمت شکل داده و... داشتیم اولی تنظیمات عمومی بود که در کل GridView اعمال می شد ودومی تنظیمات خصوصی که تنها مخصوص ستون و یا سطری خاص بود و می توانستسم ان را علاوه بر تنظیمات عمومی به کار ببریم و همیشه تنظیمات خصوصی به عمومی ارجع هستند.مثلا اگر یک استیل عمومی برای ستونها تعریف کنید و سپس یک استیل خصوصی برای ستونی خاص,در آن ستون به جای استیل عمومی استیل خصوصی اعمال می شود.

Grid View Sorting:

مرتب سازی در یک GridView به روش های مختلفی صورت می گیرد.ساده ترین روش استفاده از صفت SortExpression است.این صفت مقداری رشته ای می گیرد که همان نام فیلدی است که مرتب سازی باید بر اساس ان صورت پذیرد.می دانیم یک جدول دارای ستونهای متعددی است که هر ستون دارای داده هایی می باشد.اینکه می گوییم مرتب سازی بر اساس یک ستون انجام گیرد در مثال زیر واضح است.فرض کنید جدول ما به این صورت است:

First name	Last name	Job
ali	zadloo	karmand
reza	bidari	dabir
bahram	ahmadi	monshi

۱-SortExpression='first name':

First name	Last name	Job
ali	zadloo	karmand
bahram	ahmadi	monshi
reza	bidari	dabir

۲- SortExpression='last name' :

First name	Last name	Job
bahram	ahmadi	monshi
reza	bidari	dabir
ali	zadloo	karmand

۳- SortExpression='job' :

First name	Last name	Job
reza	bidari	dabir
ali	zadloo	karmand
bahram	ahmadi	monshi

اولین نوع مرتب سازی Automatic-Sorting است.این کار با True قرار دادن صفت AllowSorting در تگ Grid View انجام می شود:

```
<asp:GridView ID="gridview1" runat="server" DataSourceId="sqldatasource1"
AllowSorting="true">
```

با انجام این کار HeaderText ها به شکل لینک در آمده و با کلیک بر روی هر یک, جدول بر اساس همان ستونی که بر روی HeaderTextتش کلیک کردیم صعودی مرتب می شود.با کلیک مجدد روی آن به صورت نزولی مرتب می شود.

حال اگر بخواهیم مثلا با کلیک بر روی ستونی خاص مرتب سازی بر اساس ستونی دیگر انجام شود چه کنیم؟این کار نوعی سفارشی سازی است پس باید از SortExpression در تگ BoundField استفاده کنیم.SortExpression در خود تگ GridView قرار ندارد.

برای اینکار من ابتدا ستون های پیش فرضGridView را برداشته و صفت AllowSorting را با True مقداردهی می کنم:

```
   <asp:GridView ID="gridview1" runat="server"
DataSourceId="sqldatasource1" AllowSorting="true"
AutoGenerateColumns="false">
```

سپس ستونهای مورد نظر خود را با تگ Column و سپس BoundField ایجاد می کنم:

```
      <Columns>
      <asp:BoundField DataField="customer_name" HeaderText="Customer Name"
      <asp:BoundField DataField="customer_city" HeaderText="Customer City" />
      <asp:BoundField DataField="customer_street" HeaderText="Customer Street"
/>
   </Columns>
```

سپس از صفت SortExpression به دلخواه استفاده می کنم:

```
      <asp:BoundField  DataField="customer_name"  HeaderText="Customer  Name"
SortExpression="customer_city"/>
      <asp:BoundField  DataField="customer_city"  HeaderText="Customer  City"
SortExpression="customer_name" />
   <asp:BoundField DataField="customer_street" HeaderText="Customer Street"
SortExpression="customer_street" />
```

در اینجا اگر من روی customer_name HeaderText کلیک کنم مرتب سازی بر اساس فیلد Customer_city صورت می گیرد و بر عکس چون خودمان این را خواستیم.

نکته ی قابل توجه این است که مرتب سازی به این روش برای فیلدهای حاوی مقادیر باینری صورت نمی گیرد.

کنترل GridView دارای دو رویداد به نامهای Sorted و Sorting است که می توانید کدهای مورد نظر خود را در آن بنویسید.

:Sorting With ObjectDataSource

مرتب سازی می تواند با ObjectDataSource هم صورت گیرد.اما نه با SortExpression.در این نوع مرتب سازی ما قادریم از کد نویسی در این راستا استفاده کنیم زیرا ObjectDataSource از شی که ما خودمان ساختیم به عنوان منبع داده استفاده می کند. در زیر می خواهیم یک مثال نسبتا جامع در این زمینه بزنیم.

در این مثال یک DropDownList و RadioButtonList قرار دارند که DropDownList حاوی Expression هایی برای مرتب سازی است که عمل مرتب سازی بر اساس آنها انجام می گیرد و RadioButtonList نیز صعودی یا نزولی بودن را

نمایش می دهد.مقادیر در یک کنترل GridView نمایش داده می شوند و منبع داده ی این کنترل هم یک ObjectDataSource است که به کلاس empDB متصل است.

از آنجایی که مرتب سازی بر اساس یک Expression که ما از DropDownList دریافت می کنیم صورت می گیرد پس نیاز به اضافه کردن تابعی است که همان کار GetAllEmployees را انجام دهد ولی با یک آرگومان ورودی به نام Expression.این تابع را همنام با تابع GetAllEmployees و در کنار آن به کلاس اضافه می کنیم(می توان این کار را در هر کلاسی انجام داد به شرط اینکه تعداد یا نوع آرگومان های ورودی برابر نباشد):

```
    Public Function GetAllEmployees(ByVal sortexpression As String) As
DataView

    End Function
```

همانطور که می بینید یک آرگومان ورودی از نوع String به نام Expression داریم که مرتب سازی باید بر اساس آن صورت گیرد.و همینطور مقدار برگشتی به جای DataSet از نوع DataView است.این ما را به یاد شی DataView و متد Sort ان می اندازد.برای نوشتن تابع ابتدا موارد تکراری را می نویسیم:

```
        Dim cs As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(cs)
        Dim sql As String = "Select * from employees"
        Dim da As New SqlDataAdapter(sql, con)
        Dim ds As New DataSet()
        Try
            da.Fill(ds, "Employees")
        Catch
            Throw New ApplicationException("Data error.")
    End Try
```

ما مقدار برگشتی را دیگر در Try قرار ندادیم و در آن تنها DataSet را پر کردیم.در ادامه یک این شیDataSet که حاوی جدول بازیابی شده است را به یک شی DataView می دهیم تا DataView آن را مرتب کند(براساس ورودی تابع) و سپس آن را بر گردانیم:

```
  Dim dv As New DataView(ds.Tables("Employees"))
  dv.Sort = sortexpression
Return dv
```

همانطور که می دانید شی DataView عمل مرتب سازی را بر اساس چند فیلد هم انجام می دهد و اگر در پایان رشته از ASC و DESC هم استفاده کنیم مرتب سازی را به ترتیب صعودی و نزولی می کند.

به خاطر همین ما به دنبال تهیه ی رشته ای هستیم که اولا نام ستونهای مرتب سازی را بدهد و دوما صعودی و نزولی بود را مشخص کند.مثل زیر:

Customer_ID,customer_name DESC

این رشته عمل مرتب سازی را بر اساس دو فیلد Customer_ID,customer_name و به صورت نزولی انجام می دهد.

حال تعریف ObjectDataSource می رویم. ولی قبل از آن DropDownList و RadioButtonList را ایجاد می کنیم:

```
  <asp:RadioButtonList ID="RadioButtonList1" runat="server">
              <asp:ListItem Value="ASC">Ascending</asp:ListItem>
              <asp:ListItem Value="DESC">Descending</asp:ListItem>
</asp:RadioButtonList>
```

می بینید که Value در آیتم های RadioButtonList مشخص کننده ی صعودی و نزولی بودن رشته است.

DropDownList هم به صورت زیر تعریف می کنیم:

```
  Choose a Sort mode: <asp:DropDownList ID="DropDownList1"
runat="server" AutoPostBack="true">
    <asp:ListItem>emp_ID</asp:ListItem>
    <asp:ListItem>emp_firstname</asp:ListItem>
    <asp:ListItem>emp_lastname</asp:ListItem>
    <asp:ListItem>emp_job</asp:ListItem>
    <asp:ListItem>emp_ID,emp_firstname</asp:ListItem>
    <asp:ListItem>emp_firstname,emp_lastname</asp:ListItem>
    <asp:ListItem>emp_firstname,emp_job</asp:ListItem>
    <asp:ListItem>emp_lastname,emp_job</asp:ListItem>
</asp:DropDownList>
```

من این آیتم ها را به صورت دلخواه قرار دادم و صفت AutoPostBack را در آن برابر True گذاشتم.

حال باید رشته ی مرتب سازی را تولید کنیم.این رشته باید به فرمت زیر باشد:

DropDownList1.SelectedValue + " " +
RadioButtonList1.SelectedValue

ما این رشته را در یک Label که به صورت Visible=false است ذخیره کرده:

```
<asp:Label ID="Label2" runat="server" Visible="False"></asp:Label>
```

و آن Label را به عنوان پارامتر به ObjectDataSource معرفی می کنیم:

```
   Protected Sub DropDownList1_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles DropDownList1.SelectedIndexChanged
          Label2.Text    =    DropDownList1.SelectedValue    +    "    "    +
RadioButtonList1.SelectedValue
 End Sub
```

سپس تعریف ObjectDataSource:

```
      <asp:ObjectDataSource        ID="ObjectDataSource1"        runat="server"
TypeName="EmpDB" SelectMethod="GetAllEmployees">
      <SelectParameters>
      <asp:ControlParameter      ControlID="label2"      Name="sortexpression"
PropertyName="text"  />
      </SelectParameters>
    </asp:ObjectDataSource>
```

حال نوبت به تعریف GridView است:

```
   <asp:GridView                  ID="GridView1"                  runat="server"
DataSourceID="ObjectDataSource1" >
 </asp:GridView>
```

مثال ما کامل شد.ولی یک مشکل دارد و آن هم وقتی دکمه ی Select به GridView اضافه می شود مشخص می شود.ما صفت ShowSelectButton را در آن True کرده:

```
    <Columns>
       <asp:CommandField ShowSelectButton="true" />
  </Columns>
```

و یک دکمه و Labelخودمان به صفحه اضافه می کنیم:

```
    <asp:Button ID="Button1" runat="server" Text="Go Selected Value" /><br />
  <asp:Label ID="Label1" runat="server"></asp:Label></div>
```

برای فهمیدن مشکل می خواهیم با کلیک کردن دکمه ID سطری را که انتخاب کردیم در Label نمایش داده شود.برای این کار صفت DatakeyName کنترل GridView را مقدار دهی کرده:

```
DataKeyNames="emp_ID"
```

سپس در رویداد کلیک دکمه عمل بازیابی سطر انتخاب شده را با متد SelectedRow.Cell(i) انجام می دهیم:

```
          If GridView1.SelectedIndex <> -1 Then
              Label1.Text  =  GridView1.SelectedRow.Cells(1).Text  &  "  "  &
GridView1.SelectedRow.Cells(2).Text          &          "          "          &
GridView1.SelectedRow.Cells(3).Text          &          "          "          &
GridView1.SelectedRow.Cells(4).Text
          Else
              Label1.Text = "No Row Selected yet!!!"
```

```
        End If
```

شرط if برای مواقعی است که روی دکمه کلیک کنیم ولی سطری را انتخاب نکرده باشیم.

حال برای فهمیدن اشکال کار روی دکمه ی Select یک سطر کلیک کنید.سپس روی دکمه کلیک کنید . مقادیر سطری را که انتخاب کردید برایتان نمایش داده می شود.سپس عمل مرتب سازی را براساس یکی از آیتم های موجود در DropDownList انجام دهید.پس از آن مجددا روی دکمه ی مربوطه کلیک کنید تا مشخصات سطر انتخاب شده برایتان به نمایش در آید.می بینید با قبلی فرق می کند.برای جلوگیری از این مشکل و حفظ کردن سطر Select شده ی قبلی باید حداقل emp_Idآن را جایی حفظ کنیم و سپس بازیابی کنیم.برای این کار هنگام کلیک روی دکمه ی Select کنترل GridView برای مرتب سازی, emp_ID را در یک ViewState قرار می دهیم:

```
    Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles GridView1.SelectedIndexChanged
        If GridView1.SelectedIndex <> -1 Then
            ViewState("SelectedValue")                                  =
GridView1.SelectedDataKey.Values("emp_ID")
        End If
    End Sub
```

سپس هنگامی که GridView می خواست دوباره خودش را Bind کند از رویداد OnBound (وقتی تحریک می شود که جدول به طور کامل Bind شده و اماده ی نمایش است.در حقیقت قبل از نمایش این رویداد تحریک می شود)آن استفاده کرده و ابتدا ViewState را بازیابی می کنیم:

```
    Protected Sub GridView1_DataBound(ByVal sender As Object, ByVal e As
System.EventArgs) Handles GridView1.DataBound
        Dim selectedValue As String = CType(ViewState("SelectedValue"),
String)

    End Sub
```

سپس چک می کنیم Null نباشد اگر بود بازگشت کن:

```
        If selectedValue Is Nothing Then
            Return
        End If
```

در این قسمت باید آیتم Select شده ی قبلی را با جدیده عوض کنیم.برای این کار روی سطر ها پیمایش با حلقه ی Foe Each انجام می دهیم و چنانچه آیتم کلیدی هر سطری با

مقداری که ما از ViewState بازیابی کردیم برابر بود آن سطر به عنوان آیتم Select شده خواهد بود.

پس ابتدا حلقه ی For...Each را روی سطر های Gridview به صورت زیر می نویسیم:

```
        For Each row As GridViewRow In GridView1.Rows

        Next row
```

سپس در داخل حلقه عنصر DataKey هر سطر را بازیابی می کنیم:

```
Dim keyValue As String = GridView1.DataKeys(row.RowIndex).Value.ToString()
```

RowIndex همان شماره ی سطری است که در حال پیمایش است و جز شی GridViewRow است.

سپس شرط را چک می کنیم:

```
            If keyValue = selectedValue Then
                GridView1.SelectedIndex = row.RowIndex
            End If
```

اگر چنانچه پیدا شد سطر انتخاب شده همان RowIndex خواهد بود که همان سطری است که قبلا انتخاب شده بود ولی با عمل مرتب سازی جایش عوض شد.

نوع دیگر مرتب سازی با sql Query و کلمه ی ORDER By است.

GridView Paging:

صفحه بندی جز عملیات مهمی است که GridView می تواند انجام دهد.به این صورت که به جای نمایش کلی داده ها مخصوصا در حجم بالا آن ها را صفحه بندی می کنیم و سپس در تعدادی صفحه که Navigate بین انها نیز ساده است به کاربر ارایه می دهیم.برای صفحه بندی اتوماتیک کافی است صفت AllowPaging کنترل Gridview را برابر True قرار دهید و با صفت PageSize تعدا سطر هایی که در هر صفحه باید نمایش داده شوند را مشخص کنید:

```
    <asp:GridView ID="GridView1" runat="server"
datasourceId="SqlDataSource1" AllowPaging="true" PageSize=5></asp:GridView>
```

از صفات مهم دیگر در این زمینه می توان به موارد زیر اشاره کرد:

PageIndex:شماره ی صفحه ای را مشخص می کند که با Bind شدن GridVew از آن شروع به نمایش کند.

PagerSetting:نوع و فرمت دکمه یا شماره های قرار گرفته برایNavigate و صفحه بندی را مشخص می کند که در ادامه توضیح داده خواهد شد.

PagerStyle:شکل و قالب اعداد و دکمه های پیمایش صفحه را مشخص می کند.

رویداد مهمی که در این زمینه وجود دارد PagerIndexChanged است که به محض انتقال از یک صفحه به صفحه ای دیگر رخ می دهد.

تگ PagerSetting مقادیر زیر را در صفت Mode خود می پذیرد:

NextPrevious:باعث ایجاد دکمه های Next و Prev در GridView می شود.

NextPreviousFirstLast:علاوه بر لینک های Next و Prev باعث ایجاد دو دکمه می گردد که First به صفحه ی اول و Last به صفحه ی آخر می رود.

Numeric:تنها شماره ی صفحه را نشان می دهد.مثلا اگر یک Gridview ۵ صفحه داشته باشد از ۱ تا ۵ را نمایش می دهد که با زدن هر شماره به صفحه ی متناظر آن می رویم.

NumericFirstLast:علاوه بر شماره , دو لینک First و Last است.

PagerSetting دارای صفت های زیادی پس از Mode است که در زیر به دلیل مشابه بودنشان در زیر تنها به چند تای آنها اشاره می کنیم:

FirstPageText:نوشته ای که روی لینک First می آید.

FirstPageImageUrl:به جای نوشته می توان تصویری جای آن قرار داد.

این دو صفت نه تنها برای first بلکه برای Next-Prevoius-Last نیز وجود دارد.

در مثالهای زیر از GridView می خواهیم از تگ PagerSetting استفاده کنیم.

```
        <asp:GridView            ID="GridView2"            runat="server"
datasourceId="SqlDataSource1" AllowPaging="true" PageSize=5>
        <PagerSettings     Mode="NextPrevious"     PreviousPageText="Back"
NextPageText="Next" />
        </asp:GridView>
        <asp:GridView            ID="GridView3"            runat="server"
datasourceId="SqlDataSource1" AllowPaging="true" PageSize=5>
        <PagerSettings   Mode="NumericFirstLast"   PreviousPageText="Back"
NextPageText="Next" PageButtonCount="3" />
        </asp:GridView>
        <asp:GridView            ID="GridView4"            runat="server"
datasourceId="SqlDataSource1" AllowPaging="true" PageSize=5>
```

```
        <PagerSettings Mode="NextPrevious" NextPageImageUrl="~/next.JPG"
PreviousPageImageUrl="~/prev.JPG" PageButtonCount="3" />
    </asp:GridView>
```

صفت دیگر بین آنها **PageButtonCount** است که تعداد شماره های صفحات است.مثلا اگر به طور **Default** تعداد آنها ۱۰ تاست شما می توانید آن را مثلا ۳ تا کنید که به جای بقیه علامت .. قرار می گیرد و هر گاه روی آخرین شماره بروید(در اینجا ۳) خود به خود شماره ی بعدی جزو ۳ شماره خواهد بود.همه ی این اعمال که روی صفحات انجام دادیم به صورت **Visual** هم در قسمت **Properties** کنترل **GridView** قابل انجام هستند.

می خواهیم کمی بیشتر با **TEmplateField** آشنا شویم.همانطور که گفتیم این تگ باعث ایجاد یک ستون کاملا سفارشی در **GridView** ما می شود.

در مثال زیر ستونهای یک جدول را با هم ترکیب می کنیم و در یک ستون سفارشی نمایش می دهیم.این کار را با تگ < #%> انجام می دهیم:

```
    <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="false"
HorizontalAlign="Left" Font-Names="Verdana">
    <Columns>
    <asp:TemplateField HeaderText="Customer Details" HeaderStyle-Font-
Names="verdana" HeaderStyle-HorizontalAlign="Left" HeaderStyle-Font-
Bold="true">
    <ItemTemplate>
    <%#Eval("emp_firstname") + " " + Container.DataItem("emp_lastname") + "
has a Job name:" + Eval("emp_job")%>
    </ItemTemplate>
    </asp:TemplateField>
    </Columns>
  </asp:GridView>
```

در اینجا هم از **Eval** و هم از **Container.DataItem** استفاده کردیم.ولی به دلایلی **Eval** بهتر است.

در مثال زیر **Id** یک کارمند را از یک **TextBox** می گیریم و با کلیک یک دکمه مشخصات وی را در یک **TemplateField** از **GridView**نمایش می دهیم البته با شکل و فرمی دیگر.

در این راه از **ObjectDataSource** استفاده می کنیم.پس ابتدا **ObjectDataSource** را تعریف می کنیم:

```
  <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
TypeName="EmpDB" SelectMethod="GetEmployee">
```

```
    <SelectParameters>
     <asp:ControlParameter        ControlID="TextBox1"        Name="EmployeeID"
PropertyName="text" />
    </SelectParameters>
</asp:ObjectDataSource>
```

سپس TextBox وButton مربوطه را قرار داده:

```
   <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
 <asp:Button ID="Button1" runat="server" Text="Go" /><
```

و به تعریف GridView می پردازیم:

```
          <asp:GridView              ID="GridView1"              runat="server"
AutoGenerateColumns="false">
          <Columns>
          <asp:TemplateField HeaderText="---|||Employee Details|||---">
          <ItemTemplate>
          <asp:Label  runat="server"  Text='  <%#  Eval("emp_firstname")+"  "  +
Eval("emp_lastname")  %>'
          BackColor="red"    ForeColor="ButtonHighlight"    Font-Names="Tahoma"
Font-Bold="true"></asp:Label>
          </ItemTemplate>
          </asp:TemplateField>
          </Columns>
        </asp:GridView>
```

یک Label در TemplateField قرار دادیم و خاصیت Text آن را ادغام نام ونام خانوادگی مشتری کردیم و در ضمن کمی هم به Label ها شکل و فرم دادیم به این صورت که BackColor و.. برایش تعریف کردیم.برای نمایش خاصیت DataSourceID برای کنترل GridView تعریف نکردیم چون تا مقداری داخل TextBox نوشته نشود و دکمه کلیک نشود اگر این کار را کنیم با خطای NullReference مواجه می شویم.به همین خاطر Bind کردن را در رویداد کلیک دکمه انجام می دهیم:

```
      Protected  Sub  Button1_Click(ByVal  sender  As  Object,  ByVal  e  As
System.EventArgs) Handles Button1.Click
          GridView1.DataSource = ObjectDataSource1
          GridView1.DataBind()
    End Sub
```

این مثال به صورت Visual هم قابل انجام است.یادتون هست در بحث مربوط به تجارت الکترونیم و کنترل DataList در آنجا ما به صورت Visual مشابه این مثال را انجام دادیم.برای ورود به محیط Visual کنترل GridView کافیست در قسمت Task گزینه ی EditTemplate را بزنید تا وارد پنجره ی مربوطه شوید.

تمام کارهایی که تا به حال برای TemplateField انجام داده بودیم برای تگ ItemTemplate بود.تگهای دیگری نیز وجود دارند(GridView Template) مثل :

۱-HeaderTemplate:برای سفارشی کردن هدر.

۲-FooterTemplate:برای سفارشی کردن footer.

۳-AlternatingItem Template:سفارشی کردن یک در میان سطر ها.

٤-EditItemTemplate:هنگام فشردن دکمه ی Edit درGridView(البته اگر ShowEditbutton برابرTrue باشد) باعث سفارشی شدن ستون می شود.

در زیر مثال هایی از GridView Template هایی که در بالا معرفی کردیم با هم می بینیم.

در ابتدا از HeaderTemplateاستفاده می کنیم:

```
  <HeaderTemplate>
    <asp:Label runat="server" Text="My Custom Header" ForeColor="AliceBlue"
BackColor="Black" Font-Names="Verdana"></asp:Label>
</HeaderTemplate>
```

در قسمت Header یک Label قرار دادیم و یک Text یک رنگ پس زمینه و یک رنگ یرایش انتخاب کردیم.

```
      <AlternatingItemTemplate>
      <asp:Label runat="server" Text="OK"></asp:Label>
    </AlternatingItemTemplate>
```

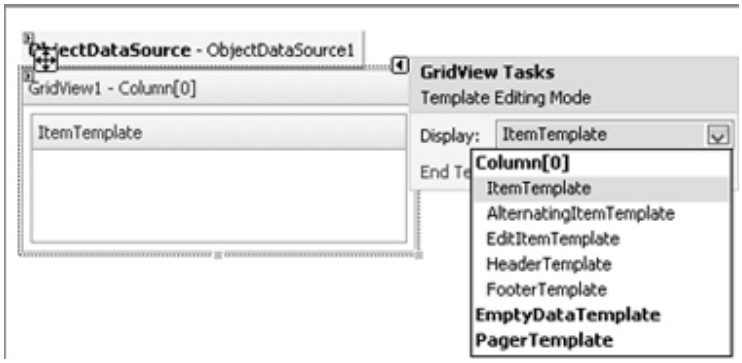
با این کار به صورت یکی در میان نوشته ی Ok در سطر های ستون TemplateField قرار می گیرد.

```
      <EditItemTemplate>
      <asp:Label runat="server" text="Edit View"></asp:Label>
    </EditItemTemplate>
```

نوشته ی Edit View در هنگام فشردن کلید Edit در سطری که این کلید فشرده شد و ستون TemplateField قرار می گیرد.

تگهای دیگری مثل EmptyDataTemplate نیز در GridView Template هستند.هر گاه منبع داده ای شما خالی بود این تگ عمل میکند و می توانید در آن هنگام GridView را سفارشی کنید.

اگر بخواهید به صورت Visual اعمال بالا را انجام دهی باید به قسمت Task کنترل GridView رفته و روی Edit Template کلیک کنید تا پنجره ای مثل پنجره ی زیر نشان داده شود:



در اوایل بحث DataBinding طرز Bind کردن توابع را آموختید.حالا می خواهیم یک مثال جالب را با هم انجام دهیم.در این مثال در یک GridView لیست زیر نمایش داده می شود:

Customer_name—account_number—branch_name---balance

آخرین فیلد مقدار موجودی هر مشتری در بانک است.این مقادیر بین ۱۰۰ تا ۴۰۰ هزار دلار در نوسان هستند.می خواهیم در کنار نام افرادی که در حسابشان کمتر از ۱۹۰هزار دلار است یک تصویر ضربدر قرمز(یعنی بد) و کنار نام افرادی که در حسابشان بیش از ۲۵۰ هزار دلار است یک تیک سبز(یعنی خوب) و در کنار نام افرادی که در حسابشان بین ۲۵۰ و ۱۹۰ هزار دلار است یک خط آبی(بد نیست) نمایش دهیم.برای این کار نیاز به یک ستون سفارشی و مازاد بر ستونهای پیش فرض داریم.که این کار را Templatefield انجام می دهد.تصاویر به شکل زیر هستند:

یک تابع برای کنترل تصاویر بالا ایجاد می کنیم:

```
Public Function GetStatusPicture(ByVal dataitem As Object) As String
End Function
```

این تابع عنصر Balance از GridView را دریافت می کند.برای اطمینان آن ا به int تبدیل کرده و در داخل یک متغیر از نوع Integer می ریزیم.به این دلیل ورودی تابع را Object در نظر گرفتیم که خطایی رخ ندهد و مقدار به سلامت وارد تابع شود سپس آن را به int تبدیل کردیم:

```
Dim price As Integer = CInt(dataitem)
```

سپس روی این مقدار شرط قرار می دهیم و در هر شرط نام تصویر مورد نظر را همراه با پسوندش برمی گردانیم:

```
If price > 250000 Then
    Return ""k.jpg""        ElseIf price < 190000 Then
    Return ""ad.jpg""        Else
```

```
Return ""‘id.jpg""        End If
```

حال به صفحه ی Aspx می رویم:

در ابتدا کنترل منبع داده ای به همراه Sql Query را ایجاد می کنیم.(جدولی که ما در این مثال رویش کار می کنیم از اتصال دو جدول به وجود آمده):

```
<asp:SqlDataSource
ID=""qlDataSource1""runat=""erver""ProviderName=""ystem.Data.SqlClient""Conne
ctionString=""%$ ConnectionStrings:aaa %>""SelectCommand=""ELECT
depositor.customer_name, account.account_number, account.branch_name,
account.balance FROM account INNER JOIN depositor ON account.account_number =
depositor.account_number""</asp:SqlDataSource>
```

سپس کنترل GridView را وارد صفحه می کنیم و منبع داده اش را مشخص می کنیم:

```
      <asp:GridView
ID=""ridView1""runat=""erver""DataSourceID=""qlDataSource1"">

    </asp:GridView>
```

قبل از ادامه این نکته را تکرار می کنیم که می توانیم در تگ <... #%> از توابع با مقدار ورودی و خروجی هم استفاده کنیم.ستونی را با تگ TemplateField درداخل تگ Column اضافه می کنیم:

```
      <Columns>
      <asp:TemplateField>
      <ItemTemplate>

      </ItemTemplate>
      </asp:TemplateField>
    </Columns>
```

چون قصد ما نمایش تصویر است پس از یک کنترل Imageدر داخل تگ ItemTemplate استفاده می کنیم و Url تصویر را مقدار بازگشتی از تابع GetStatusPicture قرار می دهیم:

```
<asp:Image runat=""erver""ImageUrl='’%#
GetStatusPicture(Container.DataItem(""alance"") %>''>
```

می توانستید از یک تگ <img src=”...” نیز به جای کنترل Image استفاده کنید.

دقت کنید که ما تابع GetStatusPicture را با آرگومان ورودی زیر صدا زدیم:

Container.Dataitem(“balance”)

قبلا هم گفتیم که Container.DataItem یک سطر را بر می گرداند و اگر به آن آرگومان ورودی بدهیم(نام ستونی خاص) آن ستون را از آن سطر به ما می دهد.در اینجا هم مقدار موجودی هر سطر را بر می گرداند که ما هم ان را مستقیما به تابع

GetStatusPicture دادیم تا تصویر مربوطه را تولید کند و سپس خود تابع GetStatusPicture را به عنوان ImageURL قرار دادیم.

اگر صفحه بندی و مرتب سازی را True قرار دهیم مشکلی پیش نمی آید:

می توان تابع GetStatusPicture را به فرم زیر نیز صدا زد:

```
"AllowPaging=""rue""AllowSorting=""rue"
GetStatusPicture(Container.DataItem)
```

در اینجا ما یک سطر را به عنوان مرجع به تابع می فرستیم.

آنگاه تابع GetStatusPicture به صورت زیر تغییر می کند:

```
    Public Function GetStatusPicture(ByVal dataitem As Object) As String
        Dim price As Integer = CInt(DataBinder.Eval(dataitem, ""alance""")
        If price > 250000 Then
            Return ""k.jpg""        ElseIf price < 190000 Then
            Return ""ad.jpg""        Else
            Return ""id.jpg""        End If
    End Function
```

با DataBinder.Eval هم که آشنا هستید.آرگومان اول مرجع است که سطر مورد نظر است و آرگومان دوم نام ستون مورد نظر است.

می توان این اشکال را به شکل دکمه در آورد و کاری کرد با کلیک روی هر کدام از آنها نام آیتم کلیک شده را در خروجی نوشت.برای این کار نمی توان از SelectedValue کنترل GridView استفاده کرد چون اصلا ما اینجا دکمه ی Select نداریم پس رویداد SelectedIndexChanged اصلا تحریک نمی شود.پس چاره چیست؟چگونه می توان نام آیتمی را مشخص کرد که در کنارش یک ImageButton کلیک شده که آن هم با TemplateField ایجاد شده.قبلا هم با دو متد CommandArgument و CommandName آشنا شده اید.پس ما از آنها استفاده می کنیم.

```
      <asp:GridView
ID=""ridView1""runat=""erver""DataSourceID=""qlDataSource1""
      <Columns>
      <asp:TemplateField>
      <ItemTemplate>
      <<asp:ImageButton              ID=""mage1""runat=""erver""ImageUrl='%#
GetStatusPicture(Container.DataItem)
%>''CommandName=""lick""CommandArgument='%# Eval(""ustomer_name"" %>''/>
      </ItemTemplate>
      </asp:TemplateField>
      </Columns>
    </asp:GridView>
```

صفت CommandName را با یک مقدار استاتیک رشته ای به نام click مقدار دهی کردیم.

این کار را برای جلوگیری از خطا انجام می دهیم.صفت CommandArgument را هم با مقدار نام مشتری کهImageButton روی سطری که با TemplateField ساخته شده پر می کنیم.

نکته ی مهم این است که ما اینبار(فقط برای تنوع) یک سطر را به عنوان آرگومان ورودی به تابع GetStatusPicture پاس کردیم .پس درخط اول تابع GetStatusPicture مقدار Balance را به صورت زیر بازیابی کنید:

```
Dim price As Integer = CInt(DataBinder.Eval(dataitem, ""alance"")
```

حالا تنها می ماند نمایش آیتم انتخاب شده.این آیتم در صفت CommandArgument imageButton قرار دارد.برای این کار مثل توضیحاتی که در بحث تجارت الکترونیک دادیم باید به دنبال رویدادی باشیم که حاوی Command باشد.خوشبختانه کنترل GridView رویدادی به نام RowCommand دارد.این رویداد وقتی تریگر می شود که دکمه ای در GridView کلیک شود.همانطور که در کنترل DataList دیدید که این رویداد ItemCommandبود.پس از رویداد RowCommand استفاده کرده و ابتدا چک می کنیم هر دکمه ای کلیک نشده باشد(چون ممکن است دکمه های دیگری در GridView باشند) برای این کار میگوییم اگر خاصیت CommandName دکمه ای که کلیک شده click باشد آنگاه داخل شرط برو و صفت CommandArgument آن را در خروجی چاپ کن:

```
      Protected Sub GridView1_RowCommand(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewCommandEventArgs)                  Handles
GridView1.RowCommand
          Dim k As ImageButton = e.CommandSource
          If k.CommandName = ""lick""Then
              Response.Write(""ou  click  The  Row  With  Customer  Name:""&
k.CommandArgument)
          End If
    End Sub
```

البته من از Commandsource استفاده کردم و شما می توانید به صورت زیر که ساده تر هم هست بنویسید:

```
      Protected Sub GridView1_RowCommand(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewCommandEventArgs)                  Handles
GridView1.RowCommand
```

```
        If e.CommandName = ""lick""Then
            Response.Write(""ou  click  The  Row  With  Customer  Name:""&
e.CommandArgument)
        End If
    End Sub
```

به ۳ دلیل عمده عمل Edit با GridView به مشکل بر می خورد:

۱-تنها راه ویرایش عناصر یک جدول, TextBox است.تا حالا حتما با سایت هایی برخورد کردید که هنگام ثبت نام از شما چیزی غیر از TextBox هم در فرمشان بیاید.مثلا خاصیت Gender.این صفت دو مقدار Male & Female دارد.هیچ سایتی این سوال را در داخل TextBox از شما نمی پرسد بلکه در یک DropDownList این کار را می کند.یا مثلا آدرس.آدرس هیچ گاه در یک TExtBox معمولی پرسیده نمی شود بلکه در یک TExtBox با صفت MultiLine پرسیده می شود و موارد دیگر.حال اگر شما این مقادیر را وارد کردید و در یک GridView قصد ویرایششان را داشته باشید همه در قالب TextBox به شما ارایه می شود.و شما می توانید به جای انتخاب هر یک از Male & Female هر رشته ای را در آن قرار دهید و عمل Update به خوبی انجام می شود.یا در قسمت آدرس یک TextBox کوچولو در اختیار شماست که تا کوچکترین کلمه ای در آن می نویسید بقیه ی کلمات از دید شمامحو می شود.

۲-مشکل بعدی نبود هر گونه Validate در TextBox های ویرایش داده در GridView است.شما در فیلد سن می توانید نام خود را وارد کنید.این هم یک مشکل بزرگ است.

۳-فضای اختصاص داده شده به هر TextBox به اندازه ی عرض ستونهای Gridview است.پس طبیعتا فضای کمی برای هر TextBox وجود دارد و ممکن است مقادیر پیش فرض موجود در آن از عرض TextBox تجاوز کند و به طور کامل دیده نشود.

همه ی این ها مشکل امنیتی بزرگی محسوب می شود.چه خوب بود که در هنگام ویرایش هم یک TextBox با MultiLine و یا یک DropDownLis یرای انتخاب Gender وجود داشت. و یا میشد کنترل های Validation را به هر TextBox به طور دلخواه Add کرد و محدودیت های لازم را انجام داد.این اعمال به طور خود به خود در

GridView وجود ندارند.ولی خوشبختانه به صورت دستی می توان آنها را انجام داد.و آن هم تنها با تگ TemplateField و عنصر مهم EditTemplatField قابل انجام است.

در زیر مثالی جامع را با هم حل می کنیم که به وسیله ی آن می توان چنین کارهایی را انجام داد.در ابتدا چون برای ایجاد MultiLine TExtBox نیاز به یک فیلد با خاصیت طولانی تر داریم یک جدول جدید ایجاد می کنیم که دارای چنین فیلدی هم باشد:

```
Create Table NewEmployee(
emp_ID   nvarchar(70)    ,
emp_firstname   nvarchar(30)    ,
emp_lastname   nvarchar(30)    ,
emp_job   nvarchar(30)    ,
emp_comment  nvarchar(500)    )
Primary Key (emp_ID);
```

در این جدول همه چیز مثل جدول Employee است تنها فیلد emp_comment تازه اضافه شده که قرار است در یک MultiLine TExtBox نمایش داده شود.همچنین emp_Job نیز در یک DropDownList به نمایش در خواهد آمد.

برای حل این مثال ابتدا از یک SqlDataSource استفاده می کنیم.همراه با Select وUpdateCommand:

```
  <asp:SqlDataSource
ID=""qlDataSource1""runat=""erver""ProviderName=""ystem.Data.SqlClient""
  ConnectionString=""%$ ConnectionStrings:aaa %>""
SelectCommand=""elect * FROM NewEmployee""UpdateCommand=""PDATE NewEmployee
SET
emp_firstname=@emp_firstname,emp_lastname=@emp_lastname,emp_job=@emp_job,emp_
comment=@emp_comment WHERE emp_ID=@emp_ID"">
</asp:SqlDataSource>
```

از پارامتر استفاده نکردیم (دلیلش را جلو تر متوجه خواهید شد).قبل از هر چیز عملیات Insert در این جدول را انجام خواهیم داد.این کار را به دلیل نمایش استفاده از TExtBox MultiLine و DropDownList انجام می دهیم.پس ابتدا کنترل های لازم را ایجاد می کنیم:

```
id:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
firstname:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br />
lastname:<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox><br />
job:<asp:DropDownList ID="DropDownList1" runat="server">
        <asp:ListItem>negahban</asp:ListItem>
        <asp:ListItem>monshi</asp:ListItem>
        <asp:ListItem>motarjem</asp:ListItem>
        <asp:ListItem>shymist</asp:ListItem>
        <asp:ListItem>computerist</asp:ListItem>
```

```
            <asp:ListItem>modir</asp:ListItem>
            <asp:ListItem>architecture</asp:ListItem>
            <asp:ListItem>barghkar</asp:ListItem>
            <asp:ListItem>hesabdar</asp:ListItem>
        </asp:DropDownList><br />
   comment:<asp:TextBox      ID="TextBox4"      runat="server"      Height="87px"
TextMode="MultiLine"
Width="151px"></asp:TextBox><br />
```

در ۳ خط اول ۳ TextBox قرار دادیم که اولی مخصوص id دومی firstname وسومی Lastnameاست.سپس یک DropDownList ایجاد کردیم و عناوین شغلها را به عنوان ListItem آن قرار دادیم(می شد عناوین شغلها را از پایگاه داه گرفت با دستور Select distinct emp_job from NewEmployee ولی برای این تعداد محدود ترجیح دادم دستی آنها را وارد کنم در ضمن هر چه ارتباظ با پایگاه داده کمتر باشد کارایی سایت بالاتر می رود).و در نهایت یک MultiLine TextBox برای دریافت Comment ایجاد کردیم.

سپس یک دکمه ایجاد می کنیم تا عملیات Add To DataBase را بر عهده گیرد:

```
<asp:Button ID="Button1" runat="server" Text="Add To DataBase" />
```

و از آنجایی که TextBox مربوط به دریافت id سوری بوده و id به رشته ای Unique تبدیل می شود یک Label زیر دکمه قرار می دهیم تا id جدید را به شخصی که در سایت عضو شده ارایه دهد(برای رجیستر شدن های بعدی):

```
<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Names="Verdana"
ForeColor="Green"></asp:Label><br />
```

عملیات Add کردن به پایگاه داده هم توسط تابع زیر که بسیار شبیه تابع InsertEmployee است انجام می دهیم:

```
    Public Function InsertEmployee(ByVal EmployeeId As String, ByVal
EmployeeFirstName As String, ByVal EmployeeLastName As String, ByVal
EmployeeJob As String, ByVal EmployeeComment As String) As String
        Dim cs As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As New SqlConnection(cs)
        Dim qs As String = "INSERT INTO NewEmployee
VALUES(@emp_ID,@emp_fname,@emp_lname,@emp_job,@emp_comment)"
        Dim cmd As New SqlCommand(qs, con)

        cmd.Parameters.Add("@emp_fname", SqlDbType.VarChar, 30).Value =
EmployeeFirstName
        cmd.Parameters.Add("@emp_lname", SqlDbType.VarChar, 30).Value =
EmployeeLastName
```

```
            cmd.Parameters.Add("@emp_job",    SqlDbType.VarChar,    30).Value    =
EmployeeJob
            cmd.Parameters.Add("@emp_comment",  SqlDbType.VarChar,  30).Value  =
EmployeeComment
            cmd.Parameters.Add("@emp_ID",    SqlDbType.NVarChar,    70).Value    =
(CStr(EmployeeId) & Guid.NewGuid.ToString).ToString
            Try
                con.Open()
                cmd.ExecuteNonQuery()
                Return CStr(cmd.Parameters("@emp_ID").Value)
            Catch err As SqlException
                Throw New ApplicationException("Data error.")
            Finally
                con.Close()
            End Try

    End Function
```

سپس برای عملیات Insert باید درون رویداد کلیک دکمه چنین بنویسیم:

```
      Protected  Sub  Button1_Click(ByVal  sender  As  Object,  ByVal  e  As
System.EventArgs) Handles Button1.Click
            Label1.Text    =    InsertEmployee(TextBox1.Text,    TextBox2.Text,
TextBox3.Text, DropDownList1.SelectedValue, TextBox4.Text)
            GridView1.DataSource = SqlDataSource1
            GridView1.DataBind()
    End Sub
```

در اینجا مقدار برگشتی تابع را که Id است در یک Label نمایش دادیم.در ضمن ما GridView را اینجا DataBind کردیم زیرا می خواستیم هر بار که سطری در جدول Insert می شود بلافاصله در GridView هم نمایش داده شود.اگر در همان تگ GridView صفت DataSourceID رامقداردهی می کردیم هر بار که سطری را به پایگاه داده اضافه می کردیم در GridView نمایش داده نمی شد(حتی با Refresh کردن هم نمی شد) و باید صفحه را می بستیم و دوباره اجرا می کردیم.

حالا نوبت به اصل کار می رسد و آن هم GridView است.چون همه کار در تگ TemplateField صورت می گیرد:

```
  <Columns>
<asp:TemplateField HeaderText="Employees">
...
```

پس نحوه ی چیدن عناصر GridView دست خودمان است.من می خواهم عناصر(ستونها) به جای حالت معمولی که در کنار هم هستند این بار روی هم باشند:

```
  <ItemTemplate>
```

```
  <asp:Label          ID="Label5"          runat="server"          Text='<%#
Eval("emp_ID")%>'></asp:Label><br />
  <asp:Label          ID="Label1"          runat="server"          Text='<%#
Eval("emp_firstname")%>'></asp:Label><br />
  <asp:Label          ID="Label2"          runat="server"          Text='<%#
Eval("emp_lastname")%>'></asp:Label><br />
  <asp:Label   ID="Label3"   runat="server"   Text='<%#   Eval("emp_job")%>'
></asp:Label><br />
  <asp:Label          ID="Label4"          runat="server"          Text='<%#
Eval("emp_comment")%>'></asp:Label><br />
</ItemTemplate>
```

در این کد ما در تگ ItemTemplate تنها عناصر را به شیوه ی دلخواه نمایش دادیم.اول id دوم نام سوم نام خانوادگی سپس شغل و در نهایت هم توضیح(Comment).

حالا نوبت به EditTemplate می رسد.ولی قبل از آن ShowEditButton را True می کنیم:

```
<asp:CommandField ShowEditButton="true" />
```

پس تا به حال تمام داده های قبلی در Label نشان داده شده اند و ما می خواهیم هنگام Edit کردن(فشردت دکمه ی edit) فیلد Job به DropDownList و فیلد Comment به MultiLineTextBox تبدیل شود.برای این کار کافی است عناصر دلخواه را به ترتیب در تگ EditTemplate قرار دهیم.

از Id شروع می کنیم.چون این فیلد نباید تغییر کند آن را هنگام تغییر در یک Label نمایش می دهیم:

```
<asp:Label ID="lab" runat="server"
Text='<%#Eval("emp_ID")%>'></asp:Label><br />
```

دو فیلد Firstname و Lastname را در TextBox نمایش می دهیم:

```
  <asp:TextBox          ID="TextBox5"          runat="server"          Text='<%#
Eval("emp_firstname")%>'></asp:TextBox><br />
      <asp:TextBox ID="TextBox6" runat="server" Text='<%#
Eval("emp_lastname")%>'></asp:TextBox><br />
```

فیلد job را در یک DropDownList قرار می دهیم.ولی قبل از آن باید بدانید که این dropDownList علاوه بر اینکه باید حاوی آیتم های مربوط به شغل ها باشد باید هنگامی که DropDownList مورد نظر می خواهد Load شود روی آیتم مشخص شده ی قبلی باشد نه آیتم جدید.مثلا اگر قبلا شخص شغل monshi را انتخاب کرده بود هنگامی که DropDownList مورد نظر Load شد(دکمه ی Edit فشرده شد) هم Monshi را نشان دهد.بدین منظور باید از صفت SelectedIndex استفاده کرده و شماره ی آیتمی را که

شخص قبلا انتخاب کرده بود به آن بدهیم تا وقتی DropDownList بار گذاری شد روی آیتم مورد نظر توقف کرده باشد.حال این شماره را از کجا بیاوریم؟می دانیم عنوان این شغل با Eval("emp_job") قابل بازیابی است.به این منظور ابتدا یک آرایه به نام a() تعریف می کنیم و عناوین شغلها را به همان ترتیبی که در DropDownlist اول(همونی که هنگام Insert تعریف کرده بودیم) در آن درج می کنیم:

```
Public a() As String = {"negahban", "monshi", "motarjem", "shymist",
"computerist", "modir", "architecture", "barghkar", "hesabdar"}
```

از نوع Public تعریف کردیم تا درون تابع زیر قابل شناسایی و استفاده باشد.

سپس تابعی تعریف می کنیم که نام شغل را از صفحه ی aspx توسط Eval("emp_job") به عنوان آرگومان ورودی از نوع Object (برای جلوگیری از خطا) دریافت کند و در یک حلقه روی تک تک عناصر آرایه ی a آن عنصر را مشخص کند:

```
    Protected Function GetSelectedTitle(ByVal title As Object) As Integer
        For i As Integer = 0 To a.Length
            If title = a(i) Then
                Return i
            End If
        Next
    End Function
```

ابتدا نام شفا مورد نظر (title) توسط تابع دریافت شد سپس از ۰ تا آنجایی که آرایه ی a آیتم دارد حلقه می چرخد و با شرط if تک تک عناصر آرایه یa با title مقایسه می شوند و هر کدام که مساوی بود اندیسش Return می شود.

البته این تابع را می توانید اگر با Vb.Net آشنایی کاملتری دارید کمی حرفه ای و به صورت زیر نوشت:

```
    Protected Function GetSelectedTitle(ByVal title As Object) As Integer
        Return Array.IndexOf(a, title.ToString())
    End Function
```

این تابع می گوید در بین اندیس های آرایه یa بگرد دنبال Title و اندیسش را برگردان.

به این ترتیب ما به اندیس شغل انتخاب شده ی قبلی دسترسی داریم و حالا برای استفاده در DropDownlist از صفت SelectedIndex ان استفاده میکنیم:

```
SelectedIndex='<%#GetSelectedTitle(Eval("emp_job"))%>'
```

یادتون باشه تگ <...#%> را در مواردی که با پایگاه داده سر کار دارد درون -" قرار دهید نه "".

نکته ی دیگر این است که آیتم های شغل ها را در این DropDownList هم به همان ترتیب قبلی قرار دهید.حتی یک جابجایی باعث ایجاد مشکل و نتیجه ی غلط می شود:

```
  <asp:DropDownList              ID="DropDownList1"              runat="server"
SelectedIndex='<%#GetSelectedTitle(Eval("emp_job"))%>'>
      <asp:ListItem>negahban</asp:ListItem>
      <asp:ListItem>monshi</asp:ListItem>
      <asp:ListItem>motarjem</asp:ListItem>
      <asp:ListItem>shymist</asp:ListItem>
      <asp:ListItem>computerist</asp:ListItem>
      <asp:ListItem>modir</asp:ListItem>
      <asp:ListItem>architecture</asp:ListItem>
      <asp:ListItem>barghkar</asp:ListItem>
      <asp:ListItem>hesabdar</asp:ListItem>
</asp:DropDownList><br />
```

نکته ی دیگر این است که نام هر دو DropDownListکه در GridView قرار دارند DropDownList1 است و خطایی گرفته نمی شود.دلیلش آن است که این دو در دو تگ جداگانه تعریف شده اند به همین دلیل مشکلی پیش نمی آید.و اصلا صفحه ی Asp کنترلی به این نام را درک نمی کند چون این دو کنترل در داخل GridView قرار دارند.

در نهایت هم Comment را در یک TextBox نشان می دهیم:

```
  <asp:TextBox ID="TextBox7" runat="server" Text='<%#
Eval("emp_comment")%>' TextMode="MultiLine" Height="70"></asp:TextBox><br />
```

پس کد کلی در تگ EditItemTemplate به شکل زیر است:

```
          <EditItemTemplate>
          <asp:Label                  ID="lab"                  runat="server"
Text='<%#Eval("emp_ID")%>'></asp:Label><br />
          <asp:TextBox       ID="TextBox5"       runat="server"       Text='<%#
Eval("emp_firstname")%>'></asp:TextBox><br />
          <asp:TextBox       ID="TextBox6"       runat="server"       Text='<%#
Eval("emp_lastname")%>'></asp:TextBox><br />
          <asp:DropDownList          ID="DropDownList1"          runat="server"
SelectedIndex='<%#GetSelectedTitle(Eval("emp_job"))%>'>
                <asp:ListItem>negahban</asp:ListItem>
                <asp:ListItem>monshi</asp:ListItem>
                <asp:ListItem>motarjem</asp:ListItem>
                <asp:ListItem>shymist</asp:ListItem>
                <asp:ListItem>computerist</asp:ListItem>
                <asp:ListItem>modir</asp:ListItem>
                <asp:ListItem>architecture</asp:ListItem>
```

```
                <asp:ListItem>barghkar</asp:ListItem>
                <asp:ListItem>hesabdar</asp:ListItem>
            </asp:DropDownList><br />
            <asp:TextBox       ID="TextBox7"       runat="server"       Text='<%#
Eval("emp_comment")%>' TextMode="MultiLine" Height="70"></asp:TextBox><br />
        </EditItemTemplate>
```

با این کار تگ Column هم به پایان می رسد.و در ادامه کمی به شکل و فرمت GridView بها دادیم:

```
    </Columns>
        <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
        <RowStyle  BackColor="#FFF7E7"  BorderColor="Red"  BorderStyle="Double"
ForeColor="#8C4510" />
        <SelectedRowStyle          BackColor="#738A9C"          Font-Bold="True"
ForeColor="White" />
        <PagerStyle ForeColor="#8C4510" HorizontalAlign="Center" />
        <HeaderStyle BackColor="#A55129" Font-Bold="True" ForeColor="White" />
  </asp:GridView>
```

حالا تنها یک چیز می ماند.و آن هم پارامتر های Sql است.این پارامتر ها کجا هستند و از کجا تامین می شوند؟.ببینید چون شما از فرمت معمولی GridView برای ویرایش داده ها استفاده نکردید پس نیازی به تعریف تگ <asp:Parameter Name....> ندارید.زیرا پارامتر ها خود به خود از روی Sql Query شناخته می شوند.تنها کاری که باید بکنید مقدار دهی به پارامترهاست.زیرا اگر یادتون باشه وقتی به صورت معمولی با GridView عمل Edit را انجام می دادیم خود کنترل TextBox GridView هایی را که هنگام edit نمایش می داد را به پارامتر های Query شما وصل می کرد پس نیلزی به تامین منبع پارامتر ها نداشتید ولی در اینجا چون خودتان همه چیز را ایجاد کردید GridView به دنبال مقادیری برای پارامتر ها می گردد(پس اینجا هم نیازی به تعریف پارامتر نیست تنها مقدار دهی کافیه) که ما در زیر آن مقادیر را ایجاد کرده و به پارامتر های متناظر نسبت می دهیم.

این کار را در رویداد RowUpdating کنترل GridView انجام می دهیم.این رویداد وقتی که هر سطری می خواهد Update شود تحریک می شود:

```
    Protected Sub GridView1_RowUpdating(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewUpdateEventArgs)                    Handles
GridView1.RowUpdating

    End Sub
```

کاری که در این رویداد باید بکنیم این است که مقادیری را که در کنترل های داخل تگEditItemTemplate مشخص کردیم را به پارامتر متناظرشان نسبت دهیم.از طرفی ما در صفحات Aspx وCode-Behind به این کنترل ها و مقادیرشان دسترسی نداریم.پس باید در کنترل GridView به دنبال این کنترل ها بگردیم.این کار با متد FindControl انجام می شود.این متد در GridView به دنبال کنترل می گردد:

FindControl(“control Id”)

ولی باید از آن درست استفاده کنیم.ما در کل GridView ممکن است ۱۰۰۰ تا DropDownList داشته باشیم(به تعداد رکورد).پس باید در سطری به دنبال آن بگردیم که موقع Update شدن است و DropDownList ان فعال شده.این سطر با آرگومان e(که مرجع ماست) قابل دستیابی است:

e.RowIndex

این کد اندیس سطر مورد را به ما می دهد.پس برای گشتن در این سطر به دنبال یک کنترل از کد زیر استفاده می کنیم:

GridView.Rows(e.RowIndex).FindControl(“controlname”)

قسمت اول کد بالا خود سطر را با استفاده از اندیسی که با e.RowIndex پیدا کرده بودیم و دادن آن به عنوان آرگومان ورودی متد Rows از GridView پیدا می کند.سپس FindControl با گرفتن نام کنترل آن کنترل را پیدا می کند.دقت کنید که چون رویدادی که داریم در آن برنامه نویسی می کنیم حاوی سطر نبود ما مجبوریم از e.RowIndex استفاده کنیم اگر رویداد ما مثلا RowDataBound(وقتی تحریک می شود که هر سطر در GridView تولید و Bind می شود) بود دیگر نیازی به اندیس سطر نداشتیم چون خود رویداد اندیس سطر را در اختیار دارد.و به صورت زیر به دنبال کنترل می گشتیم:

```
e.Row.FindControl(“TextBox5”)
```

برای دسترسی به کنترل از متد CType استفاده می کنیم.در این متد عمل تبدیل به کنترل مورد نظر را انجام می دهیم مثلا در زیر ما به دنبال کنترل با TextBox5 id می گردیم و آن را به TextBox تبدیل می کنیم.این عمل تبدیل جهت محکم کاری و جلوگیری از بروز خطا است:

```
CType(GridView1.Rows(e.RowIndex).FindControl(“TextBox5”), TextBox)
```

حال که عمل تبدیل انجام شد حاصل این کد را درون متغیری متناظر قرار می دهیم:

```
Dim t1 As TextBox =
CType(GridView1.Rows(e.RowIndex).FindControl("TextBox5"), TextBox)
```

حالا می ماند به Add کردن خاصیت Text این کنترل (که ما از این به بعد آن را با نام t1 می شناسیم) Textbox به متغیر متناظرش:

```
e.NewValues.Add("fname", t1.Text)
```

متغیر متناظر آن fname است.عمل Add را به GridView با متد NewValue انجام می دهیم.برای دسترسی به متد NewValue هم از e استفاده می کنیم که با توجه به رویدادی که در آن قرار داریم مرجع نیز هست.

پس کد کلی Add کردن پارامتر ها به صورت زیر است:

```
    Protected Sub GridView1_RowUpdating(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewUpdateEventArgs)                    Handles
GridView1.RowUpdating
        Dim              t0              As              Label              =
CType(GridView1.Rows(e.RowIndex).FindControl("lab"), Label)
        e.NewValues.Add("emp_ID", t0.Text)
        Dim              t1              As              TextBox              =
CType(GridView1.Rows(e.RowIndex).FindControl("TextBox5"), TextBox)
        e.NewValues.Add("emp_firstname", t1.Text)
        Dim              t2              As              TextBox              =
CType(GridView1.Rows(e.RowIndex).FindControl("TextBox6"), TextBox)
        e.NewValues.Add("emp_lastname", t2.Text)
        Dim            t3            As            DropDownList            =
CType(GridView1.Rows(e.RowIndex).FindControl("DropDownList1"), DropDownList)
        e.NewValues.Add("emp_job", t3.Text)
        Dim              t4              As              TextBox              =
CType(GridView1.Rows(e.RowIndex).FindControl("TextBox7"), TextBox)
        e.NewValues.Add("emp_comment", t4.Text)
    End Sub
```

یادتان باشد که متد Add در اینجا دو آرگومان می گیرد یکی نام پارامتری که در Sql Query به کار رفته و دیگری نام و خاصیت کنترلی که Value این پارامتر را تعیین می کند.

این را هم بدانید که متد NewValue جزو متدهای GridView نیست بلکه جزو متد های GridViewUpdateEventArgs است.

پس در اینجا لزوم دادن Id به کنترل هایی که در تگ editItemTemplate ایجاد کرده بودید مشخص شد.

می توانستید دکمه های Edit-Cancel-Update را هم سفارشی کنید.مثلا دکمه ی Edit رابه جای اینکه <asp:CommandField ShowEditButton=True> کار را

بکنید می توانید در انتهای تگ EditItemTemplate این Button را اضافه کنید به هر شکل دلخواه.فقط یادتان باشد در صورتی که این کار را انجام دادید صفت CommandName را به صورت متناظر مقداردهی کنید:

```
<asp:LinkButton runat=”server” Text=”Edit”  CommandName=”Edit” ID=”Linkbutton1” />
```

در این کد CommandName=”Edit” شده.این کار باعث می شود خود به خود رویداد Edit کنترل GridView تحریک شوند و در اینجا وارد تگ EditItemTemplat شویم.پس اگر دکمه ی شما عمل Update را انجام می دهد شما باید CommandName را با Update مقداردهی کنید.

البته من به دلیل پرهیز از حجم بالا, در این مثال از Validation Controls استفاده نکردم.ولی استفاده از انها بسیار ساده است.

این مثال انعطاف GridView را به ما نشان داد.ولی هنوز بحث ما در مورد این کنترل توپ تمام نشده ولی بهتر است قبل از آن کمی به دو کنترل DetailsView و FormView بپردازیم.

در Asp.NEt دو دسته کنترل DataBound قرار دارند.یک دسته از آنها که GridView هم جزوشان است در یک صفحه می توانند چندین رکورد را نمایش دهند و یا آنها را ویرایش کنند.ولی یک دسته ی دیگر هستند که معروفند به Display Single Record Of Data at a Time.یعنی در هر زمان تنها یک رکورد را نمایش می دهند که دو کنترل DetailsView و FormView نیز جزو این دسته هستند.

DetailsView:اگر صفت Allow Paging را True نکنید کلا تنها رکورد اول را نمایش می دهد ولی اگر آن را True کنید در اولین بار رکورد اول را میبینید ولی می توانید بین رکورد ها پیمایش کنید.این کنترل بسیار شبیه GridView نز می باشد(یعنی حالت کلی آیتم ها).این کنترل به طور پیش فرض داده ها را به صورت افقی نمایش می دهد بنابراین هر چه که تا اینجا ستون می نامیدیم از الان باید Row یا Fieldبنامیم.درزیر یک DetailsView را میبینید:

```
      <asp:DetailsView
ID=””etailsView1””runat=””erver””DataSourceID=””qldatasource1””AllowPaging=””
rue””AutoGenerateRows=””alse””AutoGenerateEditButton=””rue””>
      <PagerSettings Mode=NextPrevious />
      <Fields>
      <asp:BoundField DataField=””mp_ID””HeaderText=””D:””/>
      <asp:BoundField DataField=””mp_firstname””HeaderText=””irst Name:””/>
```

```
        <asp:BoundField DataField=""mp_lastname""HeaderText=""ast Name:""/>
        <asp:BoundField DataField=""mp_job""HeaderText=""ob:""/>
        <asp:BoundField DataField=""mp_comment""HeaderText=""omment:""/>
        </Fields>
    </asp:DetailsView>
```

میبینید که بسیار شبیه GridView است.تنها تفاوتش این است که در بخش ستون های سفارشی به جای تگ Column از تگ Field استفاده کردیم.بقیه ی موارد عینا در GridView وجود دارند.با DetailsView می توان کلیه عملیات نظیر Insert-Delete را انجام داد.در مثال زیر این کار انجام شده با علم به این نکته که نام پارامتر های ذکر شده در Sql Query با نام ستون متناظرشان برابر است:

```
  <asp:SqlDataSource ID=""qlDataSource1""runat=""erver""ConnectionString=""%$
ConnectionStrings:aaa %>"" SelectCommand=""ELECT * FROM [employees]""
UpdateCommand=""PDATE employees SET
emp_firstname=@emp_firstname,emp_lastname=@emp_lastname,emp_job=@emp_job
WHERE emp_ID=@emp_ID"" InsertCommand=""NSERT INTO employees
VALUES(@emp_ID,@emp_firstname,@emp_lastname,@emp_job)"" DeleteCommand=""ELETE
FROM employees WHERE emp_ID=@emp_ID""
  </asp:SqlDataSource><br />
  <asp:DetailsView
ID=""etailsView1""runat=""erver""DataSourceID=""qlDataSource1""Height=""0px""
Width=""25px""DataKeyNames=""mp_ID""AllowPaging=""rue""
    <Fields>
  <asp:CommandField
ShowEditButton=""rue""ShowInsertButton=""rue""ShowDeleteButton=""rue""/>
    </Fields>
</asp:DetailsView>
```

ما در این مثال هر ٤ عمل اصلی را قرار داده ایم.صفت AutoGenerateColumns هم برابر مقدار پیش فرض خود یعنی False خواهد بود.تگ های مربوط به style در این کنترل هم مثل GridView موجودند.همینطور تگ های مربوط به Paging.

:FormView

کنترل FormView بسیار شبیه GridView است.ولی یک فرق عمده دارد و ان هم این است که صفتی مثل AutoGenerateColumns را ندارد.این یعنی اینکه وقتی منبع داده ی آن را مثلا SqlDataSurce1 قرار دادید خود به خود چیزی تولید نمی کند و باید ستون ها را دستی برایش تنظیم کنید.این کار با استفاده از تگ های تکراری زیر صورت می گیرد:

ItemTemplate

InsertTemplate

EditTemplate

FooterTemplate

HeaderTemplate

EmptyDataTemplate

که با این تگ ها آشنا هستید.در زیر یک FormView ساده را می بینید که با تگ ItemTemplate ستون هایش مشخص شده.در اینجا نیازی به تگ Column(که در GridView بود) ویا Field(که در DetailsView بود) ندارید:

```
          <asp:FormView
ID=""ormView1""runat=""erver""DataSourceID=""qlDataSource1""
          <ItemTemplate>
          <%#Eval(""mp_Id""%><br />
          <%#Eval(""mp_firstname""%><br />
          <%#Eval(""mp_lastname""%><br />
          <%#Eval(""mp_job""%><br />
          <%#Eval(""mp_comment""%><br />
          </ItemTemplate>
        </asp:FormView>
```

این کنترل هم مثل FormView تمام عملیات اصلی را پشتیبانی میکند.در زیرمثالی کامل از آن را میبینید(SqlDataSource در اینجا همانی است که در بالا بوده):

```
  <asp:FormView
ID=""ormView1""runat=""erver""DataKeyNames=""mp_ID""DataSourceID=""qlDataSour
ce1""AllowPaging=""rue""
              <EditItemTemplate>
                  emp_ID:
                  <asp:Label        ID=""mp_IDLabel1""runat=""erver""Text='''%#
Eval(""mp_ID"" %>''</asp:Label><br />
                  emp_firstname:
                  <asp:TextBox
ID=""mp_firstnameTextBox""runat=""erver""Text='''%# Bind(""mp_firstname"" %>''
                  </asp:TextBox><br />
                  emp_lastname:
                  <asp:TextBox
ID=""mp_lastnameTextBox""runat=""erver""Text='''%# Bind(""mp_lastname"" %>''
                  </asp:TextBox><br />
                  emp_job:
                  <asp:TextBox    ID=""mp_jobTextBox""runat=""erver""Text='''%#
Bind(""mp_job"" %>''
                  </asp:TextBox><br />
```

```
                    <asp:LinkButton            ID=""pdateButton""runat=""erver""
CommandName=""pdate""                    Text=""pdate""
                    </asp:LinkButton>
                    <asp:LinkButton      ID=""pdateCancelButton""runat=""erver""
CommandName=""ancel""                    Text=""ancel""
                    </asp:LinkButton>
                </EditItemTemplate>
                <InsertItemTemplate>
                    emp_ID:
                    <asp:TextBox     ID=""mp_IDTextBox""runat=""erver""Text=''%#
Bind(""mp_ID"" %>''
                    </asp:TextBox><br />
                    emp_firstname:
                    <asp:TextBox
ID=""mp_firstnameTextBox""runat=""erver""Text=''%# Bind(""mp_firstname"" %>''
                    </asp:TextBox><br />
                    emp_lastname:
                    <asp:TextBox
ID=""mp_lastnameTextBox""runat=""erver""Text=''%# Bind(""mp_lastname"" %>''
                    </asp:TextBox><br />
                    emp_job:
                    <asp:TextBox    ID=""mp_jobTextBox""runat=""erver""Text=''%#
Bind(""mp_job"" %>''
                    </asp:TextBox><br />
                    <asp:LinkButton            ID=""nsertButton""runat=""erver""
CommandName=""nsert""                    Text=""nsert""
                    </asp:LinkButton>
                    <asp:LinkButton      ID=""nsertCancelButton""runat=""erver""
CommandName=""ancel""                    Text=""ancel""
                    </asp:LinkButton>
                </InsertItemTemplate>
                <ItemTemplate>
                    emp_ID:
                    <asp:Label         ID=""mp_IDLabel""runat=""erver""Text=''%#
Eval(""mp_ID"" %>''</asp:Label><br />
                    emp_firstname:
                    <asp:Label  ID=""mp_firstnameLabel""runat=""erver""Text=''%#
Bind(""mp_firstname"" %>''
                    </asp:Label><br />
                    emp_lastname:
                    <asp:Label   ID=""mp_lastnameLabel""runat=""erver""Text=''%#
Bind(""mp_lastname"" %>''
                    </asp:Label><br />
                    emp_job:
                    <asp:Label        ID=""mp_jobLabel""runat=""erver""Text=''%#
Bind(""mp_job"" %>''</asp:Label><br />
```

```
                <asp:LinkButton              ID=""ditButton""runat=""erver""
CommandName=""dit""                    Text=""dit""
                </asp:LinkButton>
                <asp:LinkButton            ID=""eleteButton""runat=""erver""
CommandName=""elete""                    Text=""elete""
                </asp:LinkButton>
                <asp:LinkButton              ID=""ewButton""runat=""erver""
CommandName=""ew""                     Text=""ew""
                </asp:LinkButton>
            </ItemTemplate>
        </asp:FormView>
```

میبینید که به ترتیب تگ های InsertTemplate-ItemTemplate-EditTemplate تعریف شده اند.نکته ی دیگر این است که در کنترل FormView دیگر تگ CommandField نداریم که بتوان با آنShowEditButton و یا .. را انجام داد.پس ما به طوردستی با یک LinkButton این کار را در آخر هریک از تگ های InsertTemplate-ItemTemplate-EditTemplate انجام دادیم که نشانه ی آن صفت CommandName در انتهای هر LinkButton است که باعث تحریک رویداد مربوطه توسط FormView می شود.این کار در GridView نیز قابل انجام بود.

در ادامه به ٤ مبحث پیشرفته و مهم در زمینه ی کار با GridView اشاره می کنیم:

١-کار با Footer در GridView و انجام محاسبات در آن.

٢-ایجاد GridView های تو در تو با Sql Query ها ی متصل.

٣-ذخیره ی تصاویربه صورت باینری در پایگاه داده.

٤-بازیابی تصاویر باینری از پایگاه داده با و بدون GridView.

کار با Footer در GridView:

Footer عکس Header است.به معنای پاورقی.ولی در GridViewبیشتر برای انجام محاسبات از آن استفاده می شود.مثلا چه خوب است که در پاورقی یک ستونی از یک جدول که حاوی مقادیر مالی یک شرکت است مقدار میانگین آنها را محاسبه کنیم (به صورت دینامیک).در زیر این مثال را انجام می دهیم.

ابتدا SqlDataSource را ایجاد و Query مربوطه را در آن قرار می دهیم:

```
<asp:SqlDataSource
ID=""qlDataSource1""runat=""erver""ProviderName=""ystem.Data.SqlClient""
```

```
ConnectionString=""%$ ConnectionStrings:aaa %>""
SelectCommand=""ELECT depositor.customer_name, depositor.account_number,
account.branch_name, account.balance FROM depositor INNER JOIN account ON
depositor.account_number = account.account_number""
</asp:SqlDataSource>
```

سپس کنترل GridView را تعریف می کنیم.ولی قبل از آن برای نمایش دادن Footer باید صفت ShowFooter آن را برابر True قرار دهیم.من صفت AllowPaging را هم True قرار دادم و سایز آن را ٥ گرفتم تا مقادیر محاسبه شده صفحه به صفحه باشد.یعنی در هر صفحه مجموع ٥ رگورد موجود را به من بدهد.البته این کار اتوماتیک انجام می شود یعنی صفحه هر چند تا باشد آنها را محاسبه می کند حال چه True AllowPaging باشد چه نباشد تنها به رکورد های یک صفحه دقت می کند چه ١٠٠ تا و چه حتی یکی.درGridViewزیر از Style هم استفاده شده است.در ضمن صفت AutoGenerateColumns را False گذاشتیم و با تگ BoundField خودمان ستونها را تعریف کردیم:

```
          <asp:GridView
ID=""ridView1""runat=""erver""DataSourceID=""qlDataSource1""AllowPaging=""rue
""PageSize=""""ShowFooter=""rue""AutoGenerateColumns=""alse""CellPadding=""""
Font-Names=""erdana""Font-Size=""mall""ForeColor=""333333""GridLines=""one""
          <Columns>
          <asp:BoundField       DataField=""ustomer_name""HeaderText=""ustomer
Name""/>
          <asp:BoundField       DataField=""ccount_number""HeaderText=""ccount
Number""/>
          <asp:BoundField DataField=""ranch_name""HeaderText=""ranch Name""/>
          <asp:BoundField  DataField=""alance""HeaderText=""ustomer  Balance""
DataFormatString=""0:C}""HtmlEncode=""alse"">
          </Columns>
              <FooterStyle                              BackColor=""5D7B9D""Font-
Bold=""rue""ForeColor=""hite""/>
              <RowStyle BackColor=""F7F6F3""ForeColor=""333333""/>
              <EditRowStyle BackColor=""999999""/>
              <SelectedRowStyle                         BackColor=""E2DED6""Font-
Bold=""rue""ForeColor=""333333""/>
              <PagerStyle
BackColor=""284775""ForeColor=""hite""HorizontalAlign=""enter""/>
              <HeaderStyle                              BackColor=""5D7B9D""Font-
Bold=""rue""ForeColor=""hite""/>
              <AlternatingRowStyle BackColor=""hite""ForeColor=""284775""/>
        </asp:GridView>
```

صفت Gridline نوع خطی که در جدول GridView به کار رفته را مشخص می کند.

در اینجا چون می خواهیم عمل محاسبه را در بین سطر های یک GridView انجام دهیم باید قبل از Bind شدن GridView عمل محاسبه را انجام دهیم تا وقتی GridView Bind شد آنوقت Footer هم کهنتیجه ی کار است در زیر آن قرار گیرد.ما باید لیست عناصر تمامی سطر ها را داشته باشیم تا بتوانیم عمل محاسبه را روی آنها انجام دهیم در حالیکه تا عمل Bind انجام نشود ما سطری نخواهیم داشت.ما یک رویداد داشتیم به نام RowDataBound گفتیم این رویداد وقتی تحریک می شود که هر سطر از Gridview تولید می شود.ولی این تولید به نشانه ی نمایش نبود یعنی سطر ها یکی پس از دیگری تولید می شدند و پس از اتمام تولید تمام آنها نمایش داده می شدند.حال یک رویداد مشابه به نام DataBound داریم که همان کاری که برای سطر ها می کرد را این بار برای کل GridView انجام می دهد.یعنی کل جدول Gridview پر شده و اماده ی نمایش است و این همان چیزی است که ما در اینجا نیاز داریم. پس به رویداد DataBound میرویم:

```
Protected Sub GridView1_DataBound(ByVal sender As Object, ByVal e As
System.EventArgs) Handles GridView1.DataBound

End Sub
```

حال باید عمل محاسبه را انجام دهیم.برای دسترسی به تمام عناصر هر صفحه از GridView می توانیم از یک حلقه ی For استفاده کنیم.نکتهی فوق العاده مهم اینجا این است که به چه دلیل عمل محاسبه صفحه به صفحه انجام می شود؟دلیلش استفاده از رویداد DataBound است.این رویداد حتی صفحه ها را هم مشخص کرده و سپس اگر شما در داخل آن با یک حلقه ی For عمل محاسبه را انجام دهید محاسبه صفحه به صفحه صورت می گیرد.

ابتدا یک متغیر تعریف می کنیم تا مقادیر عددی را که از هر سطر بدست می آوریم دران بریزیم:

```
Dim price As Integer = 0
```

سپس حلقه را برای گردش روی سطر های GridView به صورت زیر می نویسیم:

```
For Each row As GridViewRow In GridView1.Rows

Next
```

حالا باید به هر سطری که می رسیم ستون مربوطه را بیازیابی کنیم این کار با عبارت زیر صورت می گیرد:

```
row.Cells(3).Text
```

تا به حال ما از عبارت selectedrow.Cell(i) استفاده کرده بودیم.این عبارت سطر انتخاب شده را برای ما بر می گرداند ولی اینجا ما سطر انخاب شده نداریم.ولی چون با یک حلقه روی تک تک سطر ها حرکت می کنیم پس مرجع لازم را داریم پس به جای selectedrow از row استفاده می کنیم و برای محکم کاری آن را به Int تبدیل کرده و با مقدار قبلی متغیر Price جمع می کنیم:

```
For Each row As GridViewRow In GridView1.Rows
    price += CInt(row.Cells(3).Text)
Next
```

دقت کنید برای دسترسی به عناصر داخل یک GridView بدون هیچ رویدادی و هیچ انتخاب سطری تنها به شرط اینکه Bind GridView شده باشد یعنی عناصرش پر شده باشند به صورت زیر است:

GridViewName.Rows(index Of Row).Cells(index Of cell).Text

مثلا در این مثال به صورت زیر عنصر واقع در سطر چهارم(اگر از ۰ بشمارید می شود سطر سوم) و ستون چهارم((اگر از ۰ بشمارید می شود ستون سوم) را در خروجی چاپ کردیم.این کار را در رویداد Page_Load انجام دادیم که در هنگام این رویدا تمام کنترل ها از جمله GridView مقداردهی های لازم شده اند:

```
Response.Write(GridView1.Rows(3).Cells(3).Text())
```

حال ما مجموع را داریم و کافیست آن را در قسمت Footer کنترلgridView قرار دهیم.

برای این کار از متد FooterRow کنترل GridView استفاده می کنیم:

GridViewName.FooterRow.Cells(index Of Cell).Text=”your String”

میبینید که باید ستون FooterRow هم مشخص شود و ما همان ستونی را انتخاب کردیم که مقادیرش محاسبه شد:

```
GridView1.FooterRow.Cells(3).Text = "Total (on this page) is: " &
CInt(price).ToString("C")
```

در قسمت ToSting با آرگومان ورودی c یک علامت $ به آن اضافه کردیم.

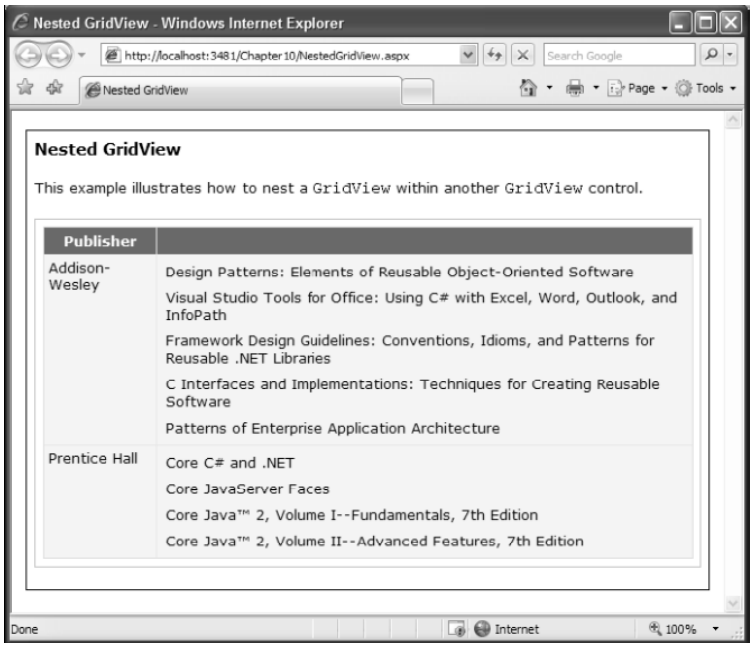
ما تنها مجموع را محاسبه کردیم.شما می توانید عملیات مختلفی انجام دهید.مثلا میانگین را محاسبه کنید یعنی مقدار مجموع را بدست آورده و تقسیم کنید بر GridView.Rows.Count چون تعداد سطر های هر صفحه ممکن است ثابت نباشد(در

آخرین صفحه) و در رویداد DataBound عبارتGridView.Rows.Count تعداد سطر ها در هر صفحه را به شما می دهد:

```
  Dim number_Of_rows As Integer = GridView1.Rows.Count
GridView1.FooterRow.Cells(3).Text = "Average (on this page) is: " &
(CInt(price) / number_Of_rows).ToString("C")
```

ایجاد GridView های تو در تو:

می توان در داخل یک GRidView یک GRidView دیگر تعریف کرد.این کار به وضوح نمایش داده های ما کمک می کند.برای مثال شکل زیر را ببینید:



در این شکل یک GridView(قسمت سمت راست) در داخل یک GridView دیگر قرار گرفته.این کار بسیار مفید است.به GridView اصلی Parent و به ان GridView که در داخل Parent قرار گرفته Child می گویند.ایجاد این نوع ساختار پیچیده نیست.ولی کمی دقت در اتصال جداول پایگاه داده دارد.

در زیر یک مثال عملی در این رابطه با هم انجام می دهیم ولی قبل از آن هدف خود را بیان می کنیم.

دو فیلد زیر نمونه هایی از خروجی برنامه ی ما هستند:

Branch Name	Customer Details

	customer_name	account_number	balance	branch_city
emam	fahimeh	1	337080	zanjan
	fahimeh	5	348316	zanjan
	homa	4	382024	zanjan
	maliheh	1	337080	zanjan
	nader	1	337080	zanjan
	shila	5	348316	zanjan
noor	customer_name	account_number	balance	branch_city
	asghar	2	292136	qazvin
	javad	9	325844	qazvin
	mahrokh	2	292136	qazvin
	rashid	9	325844	qazvin

همانطور که می بینید دو اسمی که در سمت چپ هستند(noorوemam) اسامی شعبه هستند و اسامی سمت راست اسامی مشتریان آن شعبه ها هستند.به نظر ساده می آید ولی این داده ها از ۳ جدول متصل به هم بدست آمده اند. اولین جدول مربوط به افرادی است که حساب دارند(Depositor) که شامل نام و شماره حساب آنهاست.دومین جدول جدول Account است که جدول مربوط به حسابهاست و حاوی ۳ ستون است ۱-شماره حساب ۲-شعبه ی ارایه دهنده ی حساب ۳-موجودی حساب. و درنهایت جدول شعبه ها (Branch) که حاوی نام شعبه-شهر شعبه و دارایی شعبه است.این ۳ جدول به هم وصل شده اند.اولی و دومی توسط Account_Number و دومی و سومی هم با Branch_Name.از طرفی ما با نام مشتری , شعبه ای که در آن حساب باز کرده ,

شماره ی حساب و شهر آن شعبه کار داریم.از طرف دیگر شمابرای ایجاد این چنین GridViewهای تودرتویی همیشه Query پیچیده تر را به Child GridView و Query های ساده تر را به Parent GridView دهید.در اینجا هم ابتدا منبع داده ای برای Parent GridView را ایجاد می کنیم که کارش استخراج تنها نام شعبه است:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ProviderName="System.Data.SqlClient" ConnectionString="<%$
ConnectionStrings:aaa %>" SelectCommand="Select branch_name From
branch"></asp:SqlDataSource>
```

و سپس Query مربوط به ChildGridView را ایجاد می کنیم:

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
ProviderName="System.Data.SqlClient"
ConnectionString="<%$ ConnectionStrings:aaa %>"
SelectCommand="SELECT depositor.customer_name, depositor.account_number,
account.balance, branch.branch_city FROM depositor INNER JOIN account ON
depositor.account_number = account.account_number INNER JOIN branch ON
account.branch_name = branch.branch_name WHERE (branch.branch_name = @aaa)">
```

در این Query ما ۳ جدول را به هم وصل کردیم با الگوریتم زیر:

Table1 INNER JOIN Table2 ON 'same columns' INNER JOIN Table3 ON 'same columns'

همانطور که میبینید این Query نیاز به یک پارامتر دارد که قرار است از کنترل GridView دیگر تامین شود ولی از آنجا که کنترل هایی که در تگ TemplateField قرار می دهیم توسط asp.NET شناخته شده نیستند ما از یک پارامتر معمولی استفاده می کنیم ولی دوباره آن پارامتر را مقدار دهی می کنیم:

```
    <SelectParameters>
    <asp:Parameter Name="aaa" Type="string" />
</SelectParameters>
```

حال به سراغ GridView برویم.

GridView را با false قرار دادن AutoGenerateColumns و همینطور مشخص کردن DatakeyField ایجاد می کنیم و سپس با تگ BoundField نام شعبه را نمایش می دهیم:

```
    <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="false"
DataKeyNames="branch_name">
    <Columns>
<asp:BoundField DataField="branch_name" HeaderText="Branch Name" />
```

سپس با تگ Child GridView TemplateField را در Parent GridView قرار می دهیم.بدون اینکه برایش datasource مشخص کنیم(دلیلش را جلوتر متوجه می شوید):

```
      <asp:TemplateField HeaderText="Customer Details">
      <ItemTemplate>
      <asp:GridView ID="gridView2" runat="server">
      </asp:GridView>
      </ItemTemplate>
    </asp:TemplateField>
```

پس فرمت کلی GridView به شکل زیر شد:

```
      <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="false"
DataKeyNames="branch_name">
      <Columns>
      <asp:BoundField DataField="branch_name" HeaderText="Branch Name" />
      <asp:TemplateField HeaderText="Customer Details">
      <ItemTemplate>
      <asp:GridView ID="gridView2" runat="server">
      </asp:GridView>
      </ItemTemplate>
      </asp:TemplateField>
      </Columns>
    </asp:GridView>
```

حالا نوبت به کمی کد نویسی در Code-Behind است.چون کار ما سطر به سطر پیش می رود یعنی به هر سطری از Parent GridView که رسیدیم باید Child GridView را هم مقداردهی کنیم پس از رویداد RowDataBound استفاده می کنیم.در حقیقت در این رویداد هر سطر از Bind Parent GridView شده و آماده ی نمایش است و اینجاست که باید Child GridView را مقداردهی کنیم چون پارامتری که در Child GridView وجود دارد درParent GridView است و تا آن مقداردهی و Bind نشود نمی توان Child GridView را Bind کرد. سپس چک می کنیم که سطری که می خواهد Bind شود هدر یا فوتر نباشد و DataRow باشد.این کار را با DataControlRowType انجام می دهیم.این شی متد های دیگری مثل header و footer نیز دارد که در کل مشخص کننده ی نوع سطر ارسالی است.این سطر توسط e ارسال می شود:

```
      Protected Sub GridView1_RowDataBound(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewRowEventArgs) Handles
GridView1.RowDataBound
```

```
        If e.Row.RowType = DataControlRowType.DataRow Then

        End If
    End Sub
```

سپس به دنبال کنترل Child GridView می گردیم:

```
Dim gridChild As GridView = CType(e.Row.FindControl("gridview2"), GridView)
```

یک بار دیگر بگویم کنترل هایی که در templateField می دهید توسط asp.net شناخته شده نیستند و می بایست ابتدا آنها را پیدا کرد نامی به آنها داد تا شناخته شوند.

حال نوبت به مقداردهی پارامتری که در ChildGridView sqlDataSource تعریف کردیم می رسد.برای این کار در سطر مورد نظر از ParentGridView به دنبال تنها مقدار داده شده یعنی Branch_name) می گردیم:

```
Dim branch_name As String =
GridView1.DataKeys(e.Row.RowIndex).Value.ToString()
```

این کار را با متد DataKeys کنترل GridView و دادن اندیس مرجع سطری که دارد Bind می شود به عنوان آرگومان ورودیش انجام دادیم.در نهایت با متد selectParameterکنترل sqlDataSource پارامتر تعریف شده را مقداردهی می کنیم:

```
SqlDataSource2.SelectParameters("aaa").DefaultValue = branch_name
```

می بینید که برای این کار نام پارامتری که در sql Query آن تعریف کرده بودیم را اینجا به همراه Value آوردیم.Value آن هم همان مقداریست (Branch_name) که بالاتر آن را بازیابی کردیم.تنها صفتی که می توان Value را به پارامتر داد صفت Default Value است.

در نهایت این کنترل sqlDataSource حاصل را به عنوان منبع داده برای ChildGridView در نظر می گیریم:

```
  gridChild.DataSource = SqlDataSource2
gridChild.DataBind()
```

پس به طور خلاصه ابتدا شرط DataRow بودن را مشخص کردیم سپس ChildGridView را با متد FindControl پیدا کردیم و نامی جدید به آن دادیم سپس در ParentGridView مقدار نام شعبه را بدست آوردیم و آن را به عنوان Value به پارامتر sqlDatasource دادیم و در نهایت sqlDatasource را منبع داده برای ChildGridView انتخاب کردیم و آن را bind کردیم.پس یادتون باشه که ChildGridView را درون صفحه ی aspx برایش منبع داده تعریف نکنید واین کار را

در Code-Behind و پس از مقداردهی به پارامتر SqlDataSource آن را با همین SqlDataSource که Value پارامترش را مشخص کردید DataBind کنید.

ذخیره ی تصاویربه صورت باینری در پایگاه داده:

تصاویر به دو روش در پایگاه داده ذخیره می شوند.روش اول وساده تر این است که لینک تصاویر را در پایگاه داه ذخیره کنیم که اینکار در عین سادگی دردسر هایی دارد و در برخی موارد از نظر امنیتی نیز ایراداتی دارد.ولی روش مطمین تر ذخیره ی خود تصویر در پایگاه داده است.ما کار ذخیره ی تصاویر را به وسیله ی کنترل FileUpload انجام می دهیم یعنی ابتدا تصویر مورد نظر را Browse می کنیم و سپس با یک دکمه به نام Add To DataBase آن تصویر را به پایگاه داده اضافه می کنیم.عمل Add To DataBase هم با Ado.NET انجام می دهیم.قبل از شروع کار, یک جدول جدید با مشخصات زیر ایجاد کنید:

pic_ID id– nvarchar(255) تصویر که با تابع guid تولید می شود.کلید اصلی

pic image----------خود تصویر که از نوع داده ای image است

pic_DateTime datetime-زمان ورود تصویر به پایگاه داده که با تابع DateTime.Now ایجاد می شود

pic_desc nvarchar(300)-- توضیح در مورد تصویر

پس ابتدا کنترل FileUpload را در صفحه قرار می دهیم:

```
<asp:FileUpload ID="FileUpload1" runat="server" />
```

سپس یک TextBox برای دریافت توضیح در مورد تصویر در صفحه قرار می دهیم:

```
<asp:TextBox ID="TextBox1" runat="server" Height="45px"
TextMode="MultiLine" Width="229px"></asp:TextBox>
```

و در نهایت یک دکمه که وظیفه اش اضافه کردن تصویر به پایگاه داده است:

```
<asp:Button ID="Button1" Text="Add To dataBase"  Runat=server />
```

حال به Code-Behind رفته و رویداد کلیک دکمه را باز می کنیم.ما به وسیله ی متد PostedFile کنترل FileUpload می توانیم به مشخصات تصویر ارسال شده دسترسی داشته باشیم.مثلا اگر خود تصویر را بخواهیم از متد InputStream استفاده می کنیم:

```
FileUpload1.PostedFile.InputStream
```

مقدار بر گشتی آن از نوع Stream است پس:

```
Dim Image As Stream = FileUpload1.PostedFile.InputStream
```

و یا اندازه ی تصویر با متد ContentLenght قابل بازیابی است:

```
FileUpload1.PostedFile.ContentLength
```

مقدار برگشتی آن Integer است پس:

```
Dim intImageSize As Integer = FileUpload1.PostedFile.ContentLength
```

ما در اینجا به همین دو خصوصیت یعنی تصویر و اندازه ی آن نیاز داریم.

از آنجایی که نوع تصویری که باید ذخیره شود بایت است و یک بایت(۸ بیت) نیست بلکه چندین بایت است پس آرایه ای از بایت ها را برای آن به اندازه ای که با متد ContentLenght تعیین شده در نظر می گیریم:

```
Dim ImageContent(intImageSize) As Byte
```

حالا باید تصویر را که از نوع Stream است به درون این آرایه ی بایتی انتقال دهیم.برای این کار از متد Read شی Stream استفاده می کنیم.این متد از Stream که متد ازش مشتق شده می خواند ودرون آرایه ی بایتی می نویسد. این متد ۳ ورودی دارد.اولی آرایه ای از بایت است.دومی یک اندیس است که عمل خواندن از آنجا شروع می شود و در نهایت سایز آن آرایه ی بایتی.در این مثال شی Stream که باید مقادیر از آن خوانده شوند همان Image است که توسط متد InputStream خوانده شده بود.و آرایه ی بایتی که باید تصویر دران نوشته شود ImageContent و اندیس شروع ۰ و اندازه ی آن هم ImageSize است:

```
Image.Read(ImageContent, 0, intImageSize)
```

پس مراحل به صورت زیر طی شدند:

```
      دریافت سایز تصویر
  Dim intImageSize As Integer = FileUpload1.PostedFile.ContentLength
      دریافت استریم تصویر
  Dim Image As Stream = FileUpload1.PostedFile.InputStream
       تعریف آرایه ای از بایتها
  Dim ImageContent(intImageSize) As Byte
خواندن مقادیر تصویر از استریم مربوطه و نوشتن آن در آرایه ی بایت ها
Image.Read(ImageContent, 0, intImageSize)
```

حالا آرایه ی ImageContent آماده جهت ارسال به پایگاه داده است.در زیر کد کلی کار را میبینید:

```
      Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
          Dim intImageSize As Integer = FileUpload1.PostedFile.ContentLength
          Dim Image As Stream = FileUpload1.PostedFile.InputStream
          Dim ImageContent(intImageSize) As Byte
          Image.Read(ImageContent, 0, intImageSize)
          Dim con As New
SqlConnection(ConfigurationManager.ConnectionStrings("aaa").ConnectionString)
          Dim qs As String = "INSERT INTO photo Values(@id,@img,@date,@desc)"
          Dim cmd As New SqlCommand(qs, con)
```

```
            cmd.Parameters.Add("@id",     SqlDbType.NVarChar,     255).Value     =
Guid.NewGuid.ToString
            cmd.Parameters.Add("@img", SqlDbType.Image).Value = ImageContent
            cmd.Parameters.Add("@date",         SqlDbType.DateTime).Value         =
DateTime.Now
            cmd.Parameters.Add("@desc",    SqlDbType.NVarChar,    300).Value    =
TextBox1.Text

            Try
                con.Open()
                cmd.ExecuteNonQuery()
                con.Close()
                Response.Write("New Photo successfully added!")
            Catch ex As SqlException
                Response.Write("Insert  Failed.  Error  Details  are:<br/>  "  &
ex.ToString())
            End Try
    End Sub
```

پس حالا به راحتی می توانید تصویر مورد نظر خود را در پایگاه داده قرار دهید.

بازیابی تصاویر باینری از پایگاه داده:

بازیابی تصاویر باینری در حالت معمولی کار بسیار ساده ای است.ولی بدیش این است که در حالت معمولی تنها یک تصویر را می توانید با آن نمایش دهید و آن هم با قسمت Where مشخص می شود که اگر مشخص نکنید همان تصویر موجود در سطر اول نمایش داده می شود.در این روش مقدار باینری تصویر دریافت می شود و درون آرایه ای از بایت ها قرار می گیرد و سپس با متد Response.BinaryWrite در خروجی چاپ می شود این متد دقیقا مثل Response.write است.

عمل اجرا به صورت زیر توسط ExecuteReader انجام می شود:

```
Dim reader As SqlDataReader = cmd.ExecuteReader()
```

سپس عمل خواندن با یک if صورت می گیرد:

```
      If reader.Read() Then

    End If
```

هنگامی که مقدار دریافتی را می خواهیم در یک آرایه از بایت بریزیم عمل تبدیل نوع را انجام می دهیم البته این تبدیل نوع برای محکم کاری است:

```
   Dim bytes As Byte() = CType(reader("pic"), Byte())
```

پس ما با عبارت reader("pic") تصویر را در اختیار داریم و سپس برای محکم کاری آن را تبدیل به آرایه ی بایت می کنیم و در متغیر bytes میریزیم.

حال به سادگی با تابع Response.BinaryWrite به سادگی در خروجی چاپ می کنیم:

```
Response.BinaryWrite(bytes)
```

پس تابع به طور کلی به صورت زیر شد:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim cs As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString()
        Dim con As New SqlConnection(cs)
        Dim qs As String = "SELECT pic FROM photo"
        Dim cmd As New SqlCommand(qs, con)
        Try
            con.Open()
            Dim reader As SqlDataReader = cmd.ExecuteReader()
            If reader.Read() Then
                Dim bytes As Byte() = CType(reader("pic"), Byte())
                Response.BinaryWrite(bytes)
            End If
            reader.Close()
        Catch ex As Exception
            Response.Write(ex.Message)
        Finally
            con.Close()
        End Try
    End Sub
```

دیدید که بازیابی هم ساده است.بحث تبدیل نوع را همیشه در جاهایی که کمترین شکی نسبت به بروز خطا دارید رعایت کنید.

یک نکته در کد بالا اینکه این کد توانایی نمایش بیش از یک تصویر را ندارد که مشکل بزرگی است که جلوتر با GridView آن را حل می کنیم.ولی مشکل دیگر در مورد تصاویر با حجم بالاست.این کد توانایی نمایش حجم مقادیر باینری بالا را ندارد.مثلا یک تصویر با حجم حدود ۲ مگا بایت را ندارد.

ببینید برای تصاویر با حجم بالای باینری شی CommandBehavior حاوی یک متد به نام SequentialAccess است.این متد باعث می شود که reader در خواندن حجم بالای باینری دچار اشکال نشود و تصویر را قطعه به قطعه بخواند و در خروجی بنویسد.

SequentialAccess یک جوری شی reader را در خواندن مقادیر stream توانا

میکند.پس اولین تغییر این است که در داخل ExecuteReader از عبارت زیر استفاده کنیم:

```
CommandBehavior.SequentialAccess
```

ما قبلا آمدیم و یک راست مقدار reader را درون آرایه ی باینری ریختیم ولی حالا با ملاحظه ی سایز این کار را انجام می دهیم با متد GetBytes از شی reader.این متد تعداد بایت هایی که بازیابی شده را برمیگرداند.و ۵ آرگومان ورودی دارد.اولین اندیس اندیسی است که عمل خواندن کلی از آنجا شروع می شود و همیشه باید ۰ باشد.دومی اندیسی است که عمل خواندن جزیی باید از آن شروع شود.سومی آرایه ی اصلی است که مقادیر خوانده شده باید قطعه قطعه در آن ریخته شود.چهارمی هم اندیسی است که عمل نوشتن از آن شروع می شود و باید ۰ باشد.و آخرین آرگومان اندازه ی بافر(محلی آرایه وار برای ذخیره) است که یک تصویر را به آن اندازه قطعه بندی می کنند.

پس من یک متغیر به نام BufferSize تعریف می کنم و اندازه ای که تصاویر بر حسب آن باید قطعه بندی شوند را به آن می دهم من این اندازه را ۱۰۰ در نظر گرفتم (این اندازه دلخواه است ولی سعی کنید نه خیلی زیاد و نه خیلی کم باشد)تا هر تصویری که نوبت خواندن از پایگاه داده و نوشتن در خروجیش شد ۱۰۰ بایت ۱۰۰ بایت خوانده و نوشته شود:

```
Dim bufferSize As Integer = 100
```

سپس یک آرایه از بایت ها را تعریف می کنیم و از آنجا که حتما باید مقدار اولیه داشته باشد به صورت زیر می نویسیم :

```
Dim bytes As Byte() = New Byte(bufferSize - 1) {}
```

این جور تعریف را اعلان با اندازه ی پویا می نامند.

ببینید این آرایه ی بایت هاست که تصویر را در خروجی می نویسید و از آنجا که قرار است تصویر به ۱۰۰ تا ۱۰۰ تا خوانده شود اندازه اش را از ۰ تا ۱۰۰-۱ (99)در نظر می گیریم و چون باید مقدار اولیه داشته باشد به فرم بالا آن را New می کنیم.

حالا ما به یک متغیر کنترل کننده نیاز(جلوتر نحوه ی کارش را می فهمید):

```
Dim bytesRead As Long
```

سپس به متغیری نیاز داریم که مکانی که باید خوانده شود را هر لحظه مشخص کند.مثلا برای بار اول ۰ تا ۹۹ باشد و دفعه ی بعد با مقدار بافر جمع شود و تصویر از بایت صدم شروع به خواندن کند:

```
Dim readFrom As Long = 0
```

از حلقه ی Do...Loop While استفاده می کنیم تا در بار اول حتی اگر هم شرط درست نباشد وارد حلقه شویم.برای درک نحوه ی کار حلقه ی Do...Loop While مثال زیر راببینید:

```
Dim a As Integer = 10
Dim b As Integer = 30
Do
    Response.Write(a)
    a += 2
Loop While a <> b
```

مقدار اولیه ی a=۱۰ از b=30 کمتر است.بار اول بدون توجه به شرط مقدار a در خروجی چاپ شده و به آن یک واحد اضافه می شود.حالا شرط برای ورود مجدد به داخل حلقه چک می شود.اگر a برابر نبود با b آنگاه وارد حلقه شو.پس این حلقه در صورت تداوم درستی شرط ادامه می یابد.پس خروجی این حلقه مقادیر a از ۲ تا ۲۸ دوتا دوتا است.

حلقه ی ما به شکل زیر است:

```
Do
    bytesRead = r.GetBytes(0, readFrom, bytes, 0, bufferSize)
    Response.BinaryWrite(bytes)
    readFrom += bufferSize
Loop While bytesRead = bufferSize
```

متغیر bytesRead تعداد بایتهای خوانده شده در هر مرحله از ۱۰۰ تایی ها را مشخص می کند.همانطور که گفتیم تابع GetBytes تعداد بایت ها ی خوانده شده را بر می گرداند.سپس مقادیر خوانده شده از reader و نوشته شده در bytes توسط متد Response.BinaryWrite در خروجی نوشته می شوند و برای رفتن به مرحله ی بعد هم مقدار readFrom+buffersize می شود.مثلا دفعه ی اول لر بایت ۰ تا ۹۹ خوانده شد.حالا باید از ۱۰۰ شروع شود.دفعه ی اول readfrom ۰ بود پس از بایت ۰ شروع به خواندن کرد به تعداد بافر که ۱۰۰ تا بود حالا نقطه ی شروع خواند باید ۱۰۰ تا جلو رود و از آنجا بخواند و بنویسد و به همین ترتیب.حالا این حلقه یک شرط دارد و آن هم برابر بودن BytesRead و bufferSize است.این یعنی حلقه تا زمانی ادامه دارد که مقدار بایتهای خوانده شده در هر مرحله برابر ۱۰۰(buffereSize) باشد.بیایید Trace انجام دهیم:

فرض کنیم تصویر ۳۴۵ بایت است.(آن را کوچک در نظر گرفتیم تا Trace ساده تر شود):

BufferSize=100

ReadFrom=0

1----------مرحله ی اول

bytesread=100

Response.binaryWrite(“first 100 bytes”)

readfrom=0+100=100

if 100=100 continue do while===True

2---------- مرحله ی دوم

bytesread=100

Response.binaryWrite(“second 100 bytes”)

readfrom=100+100=200

if 100=100 continue do while===True

3---------- مرحله ی سوم

bytesread=100

Response.binaryWrite(“third 100 bytes”)

readfrom=200+100=300

if 100=100 continue do while===True

4---------- مرحله ی چهارم

bytesread=45

Response.binaryWrite(“fourth 100 bytes”)

readfrom=45+300=345

if 45=100 continue do while===False---ExitFrom Doo..loopWhile

پس به این ترتیب از حلقه خارج می شویم.کد کلی را در زیر می بینید:

```
        Dim connectionString As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString()
        Dim con As New SqlConnection(connectionString)
        Dim qs As String = "SELECT pic FROM photo"
        Dim cmd As New SqlCommand(qs, con)
        Try
            con.Open()
            Dim r As SqlDataReader =
cmd.ExecuteReader(CommandBehavior.SequentialAccess)
            If r.Read() Then
                Dim bufferSize As Integer = 100
                Dim bytes As Byte() = New Byte(bufferSize - 1) {}
                Dim bytesRead As Long
                Dim readFrom As Long = 0
                Do
```

```
                    bytesRead = r.GetBytes(0, readFrom, bytes, 0,
bufferSize)
                    Response.BinaryWrite(bytes)
                    readFrom += bufferSize
                Loop While bytesRead = bufferSize
            End If
            r.Close()
        Catch ex As Exception
            Response.Write(ex.Message)
        Finally
            con.Close()
    End Try
```

حتما یادتان باشد که در select Sql Query حتما آن ستونی را انتخاب کنید که در آن تصویر وجود دارد نه ستون دیگر.حال می رسیم به اشکال عمده ای که بین دو نوع کد بالا یکی بود و آن هم Single Result بودن آنها است یعنی تنها یک تصویر به خروجی می دهند.ما می خواهیم علاوه بر خود تصویر Id و Description آن را هم برای تمامی سطر ها نمایش دهیم که در کد های بالا امکان پذیر نبود.برای جلوگیری از این مشکل از یک GridView برای نمایش استفاده می کنیم.یک SqlDataSource به صورت زیر ایجاد می کنیم:

```
<asp:SqlDataSource ID="SqlDataSource1"
ProviderName="System.Data.SqlClient" ConnectionString="<%$
ConnectionStrings:aaa %>"
SelectCommand="SELECT * FROM photo"
runat="server"></asp:SqlDataSource>
```

ببینید اگر یک GridView در صفحه بگذارید و منبع داده ای آن را SqlDataSource بالا قرار دهید هیچ نتیجه ای برای شما در بر ندارد.به همین دلیل و با وجود متد REsponse.BinaryWrite ما نیاز به یک صفحه ی جداگانه برای تولید تصاویر و همچنین تگهای HTML برای نمایش تصاویر داریم.تگ همیشگی تصویر در HTML تگ زیر بود:

```
<img src=.../>
```

ما باید تصویر خود را در جای دیگر تولید کنیم و سپس با استفاده از صفت Source تگ بالا آن تصویر را نمایش دهیم.بهترین راه برای ارتباط هر چه بهتر این صفت و تولید تصویر استفاده از HttpHandler است.ما یک Custom HttpHandler ایجاد می کنیم و تصاویر را در تابع ProcessRequest آن بازیابی می کنیم.برای این کار از HttpHandler آماده استفاده می کنیم.پس از قسمت Add New Items گزینه ی

Generic Handler را انتخاب کنید تا این فایل به وب سایت ما Add شود.و در ابتدا به صورت زیر است:

```
<%@ WebHandler Language="VB" Class="Handler2" %>

Imports System
Imports System.Web

Public Class Handler2 : Implements IHttpHandler

    Public Sub ProcessRequest(ByVal context As HttpContext) Implements
IHttpHandler.ProcessRequest
        context.Response.ContentType = "text/plain"
        context.Response.Write("Hello World")
    End Sub

    Public ReadOnly Property IsReusable() As Boolean Implements
IHttpHandler.IsReusable
        Get
            Return False
        End Get
    End Property

End Class
```

در مورد پیکر بندی این ایل قبلا توضیح داده ایم.در اینجا باید تمام کد های تولید تصویری را که نوشتید درون تابع ProccesRequest قرار دهید.ولی با اندکی تغییر.ببینید قصد ما این است که هر تصویر را در کنار بقیه ی عناصرش مثل Description قرار دهیم.برای این کار هنگام نمایش یک تصویر Id آن را به همراه QueryString به این فایل می فرستیم که نحوه ی ارسال را جلوتر درر بخش GridView میبینید.QueryString هم یک شی است که می توان توسط شی Context که آرگومان ورودی تابع است به آن دسترسی داشت:

Context.Request.QueryString(“query string name”)

ما هنگام نمایش هر تصویر جلوی سایر عناصرش Id آن را توسط QueryString به فایل Handler می فرستیم و سپس توسط آن Id تصویر را بازیابی کرده و به تگ Image وصفت Src می فرستیم.

پس ابتدا Id تصویر را بازیابی می کنیم:

```
Dim id As String = context.Request.QueryString("id")
```

نام QueryStringارسالی id بود.

سپس چک می کنیم اگر id خالی بود ادامه ندهیم:

```
        If id Is Nothing Then
            Throw New ApplicationException("Must specify ID.")
        End If
```

پس دیگر از catch در بلوک Try نیازی نیست.

سپس id بازیابی شده ر به عنوان پارامتر شرط Where در نظر می گیریم:

```
        Dim connectionString As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim con As SqlConnection = New SqlConnection(connectionString)
        Dim qs As String = "SELECT pic FROM photo WHERE pic_id=@ID"
        Dim cmd As SqlCommand = New SqlCommand(qs, con)
        cmd.Parameters.Add("@ID", SqlDbType.NVarChar, 255).Value = id

        Try
            con.Open()
            Dim r As SqlDataReader = cmd.ExecuteReader(CommandBehavior.SequentialAccess)
            If r.Read() Then
                Dim bufferSize As Integer = 100
                Dim bytes As Byte() = New Byte(bufferSize - 1) {}
                Dim bytesRead As Long
                Dim readFrom As Long = 0
                Do
                    bytesRead = r.GetBytes(0, readFrom, bytes, 0, bufferSize)
                    context.Response.BinaryWrite(bytes)
                    readFrom += bufferSize
                Loop While bytesRead = bufferSize
            End If
            r.Close()
        Finally
            con.Close()
        End Try
```

حالا به صفحه ی aspx مربوطه برویم و GridView را با قرار دادن AutoGenerateColumns برابر false و تولید ستونها به صورت سفارشی ادامه می دهیم:

```
    <asp:GridView id="GridView" runat="server" Font-Names="Verdana" Font-Size="X-Small" DataSourceID="SqlDataSource1" AutoGenerateColumns="false">
    <Columns>
    <asp:TemplateField>
        <ItemTemplate>
        <img src='handler.ashx?ID=<%# Eval("pic_ID")%>'/><br />
          <%#Eval("pic_ID")%><br />
```

```
                <%#Eval("pic_DateTime")%><br />
                <%#Eval("pic_Desc")%><br />
            </ItemTemplate>
        </asp:TemplateField>
        </Columns>
</asp:GridVIew>
```

تنها نکته در این کد این است که ما src تصویر را به صورت زیر صدا زدیم:

handler.ashx?id='Value Retrive From pic_ID field'

به عنوان مثال:

handler.ashx?id=dh436hfh45-dh45sh-5ruurrfj

پس ما یک QueryString به نام id با مقداری که از فیلد مربوطه با متد Eval دریافت کردیم را به فایل Handler می فرستیم و ان فایل هم خروجی مربوطه را چاپ می کند.حال می رسیم به توضیحی در مورد ارتباط تگ img و response.BinaryWrite.ببینید فایل Handler ما هیچ گونه مقدار برگشتی ندارد و یک راست تصویر را در خروجی چاپ می کند.ولی اگر فایل را در تگ img استفاده کنیم آنگاه عمل BinaryWrite درون تگ img صورت می گیرد.در حالیکه اگر از کنترل image به جای آن استفاده کنید تصویر به خروجی نمی رود.خروجی کار ما چیزی مثل شکل زیر است:

Caching in ASP.NET 2.0

مبحث caching از جمله مباحث بسیار مهم در زمینه ی کارایی برنامه های وب است.دیگر نمی توان این بحث را جزو مباحث پیشرفته در asp.net مطرح کرد بلکه دیگر با وجود گسترش اطلاعات و کاربرانی که نیاز به دسترسی آن اطلاعات دارند caching یک بحث کاملا ضروری برای هر برنامه ی وب می باشد.کش کردن در asp.net یعنی تهیه ی یک کپی از صفحه یا هر جزیی از صفحه به منظور ارایه به کاربران تا هنگامی که عمل بروز رسانی انجام نشده.

دو نوع cashing در حالت کلی در asp.NET وجود دارند:

OutPut Caching، ساده ترین روش caching است . صفحات aspx . بر اساس فرآیندی در سمت سرویس دهنده ترجمه و پس از اجراء ، ماحصل خروجی آنها به صورت یك صفحه html برای سرویس گیرنده ارسال می گردد . در این روش یك نسخه نهائی از صفحه ترجمه شده html که برای سرویس گیرنده ارسال شده است ، cache می گردد . بدین ترتیب ، زمانی که سرویس گیرنده بعدی درخواست خود برای استفاده از این صفحه را برای سرویس دهنده ارسال می نماید ، در مقابل اجرای صفحه و تولید خروجی لازم (انجام پردازش های سمت سرویس دهنده با توجه به ماهیت و جایگاه عملکردی یك صفحه در یك برنامه وب) ، صفحه ترجمه شده html بطور اتوماتیك برای وی ارسال می گردد . بدین ترتیب ، مدت زمانی که لازم است صفحه به همراه کد درون آن ترجمه و اجراء گردد بطور کامل حذف می گردد .

Data Caching : برای استفاده از این روش می بایست بطور دستی و از طریق کد شرایط آن را فراهم نمود . پیاده کنندگان می توانند بخش های مهمی از اطلاعات را که زمان زیادی صرف ساختن آنها شده است (نظیر یك DataSet بازیابی شده از یك بانك اطلاعاتی) را در cache ذخیره نمایند . در ادامه ، سایر صفحات می توانند وجود این اطلاعات را بررسی و در صورت موجود بودن در cache از آنها استفاده نمایند . بدین ترتیب ، مراحل مورد نیاز برای بازیابی اطلاعات حذف خواهد شد .
روش فوق از لحاظ مفهومی شباهت زیادی به استفاده از شی applicationدارد ولی دارای قابلیت های بیشتری در سمت سرویس دهنده است . به عنوان نمونه زمانی که حجم داده بگونه ای زیاد گردد که کارآئی یك برنامه را تحت تاثیر قرار دهد ، آیتم های موجود در cache بطور اتوماتیك از cache حذف خواهند شد . برای ذخیره آیتم ها در cache می توان یك تاریخ را مشخص تا پس از سپری شدن زمان مورد نظر بطور اتوماتیك از حافظه حذف گردند

علاوه بر این دو روش دو مدل cashing دیگر هم وجود دارند که از ۲ تای بالا مشتق شده اند:

Fragment caching: روش فوق نوع خاصی از output caching است و در مقابل caching کد Html برای تمامی صفحه ، صرفا" بخش هائی خاص از کد cache می گردد. در این روش خروجی کد ترجمه شده Html مربوط به یك كنترل كاربر بر روی یك صفحه ذخیره می گردد . بدین ترتیب در صورت اجراء مجدد صفحه ، کد موجود در صفحه اجراء می گردد و ضرورتی به اجراء کد مرتبط با کنترل کاربر نخواهد بود .

source caching Data : این نوع caching بر اساس کنترل های منبع داده ایجاد و شامل کنترل های منبع داده ObjectDataSource ، SqlDataSource و XmlDataSource می باشد. از لحاظ فنی ، در روش فوق از data caching استفاده می گردد. تنها تفاوت موجود در این رابطه ، عدم نیاز به انجام پردازش های مورد نیاز توسط پیاده کننده است . بدین منظور لازم است که برخی خصلت ها و پیام های کنترل منبع داده که مسئولیت ذخیره و بازیابی caching را برعهده دارند ، توسط پیاده کنندگان پیکربندی گردد.

OutPut Cashing

در مورد OutPut Caching توضیح داده شد و حالا نوبت به یک مثال عملی در زمینه ی آن می رسد.در این مثال تاریخ و ساعت را در یک کنترل Label نگهداری می کنیم و سپس صفحه را برای ۲۰ ثانیه کش می کنیم.برای این کار بالای عنوان صفحه ی aspx از تگ <%@...> استفاده کرده و گزینه ی OutPutCache را انتخاب می کنید:

```
<%@ OutputCache %>
```

این تگ حاوی صفات مختلفی است که مهمترین آن Duration است.این صفت تعیین می کند که صفحه برای چه مدت cache شود.از صفت های مهم دیگر آن می توان به صفت VaryByParam اشاره کرد که وابستگی کش را بر اساس مقادیری که می پذیرد تعیین می کند.فعلا آن را none قرار دهید.در زیر مثالی در این زمینه را می بینید:

```
  <%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="_Default" %>
  <%@ OutputCache Duration="20" VaryByParam="None" %>
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

  <html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Label ID="Label1" runat="server"></asp:Label>
    </div>
    </form>
</body>
</html>
```

این کد کلی صفحه است.که تنها یک OutPutCache و یک Label به آن اضافه شده.در زیر کد پشت صحنه را می بینید که این Label را با تاریخ حال حاضر مقدار دهی کردیم:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Label1.Text = "DateTime Now is:<br/>" & DateTime.Now
End Sub
```

حالا اگر صفحه را برای اولین بار اجرا کنید تاریخ روز را میبینید ولی اگر قبل از ۲۰ ثانیه (انقراض output cache) هر چند بار صفحه را به روز کنید یا اینکه ببندید و سپس باز کنید باز هم همان تاریخی که قبلا نمایش داده بود روی صفحه ظاهر می شود.ولی پس از ۲۰ ثانیه اگر به آن رجوع کنید میبینید که تاریخ مجددا به روز شده.پس ما صفحه را Cache کردیم و تا ۲۰ دقیقه هر کس وارد سایت ما شد با صفحه ای تکراری(نه برای وی) که در cache ذخیره شده روبرو می شود و اصلا کد ها به سرور نمی روند تا کامپایل شوند.البته درست نیست شما تاریخ را کش کنید ما این کار را کردیم تا مفهموم کش را متوجه شوید.

شاید بنظر ۲۰ ثانیه زمان زیادی نباشد ولی برای سایتی که حاوی اطلاعات گسترده ای جهت ارائه به کاربران متعدد است ، این موضوع می تواند کاملا متفاوت باشد. به عنوان نمونه ، فرض کنید می خواهیم لیستی از محصولات قابل عرضه به کاربران را در یك صفحه نمایش دهیم . با caching صفحه به مدت ۲۰ ثانیه ، دستیابی به بانك اطلاعاتی محدود به سه عملیات در یك دقیقه می گردد . بدون caching ، برای هر کاربری که متقاضی مشاهده لیست محصولات است ، می بایست فرآیند ارتباط با بانك اطلاعاتی و نمایش محصولات در یك ساختار نمایشی مناسب (نظیر Gridview) انجام شود . بدیهی است با caching صفحه به مدت ۲۰ ثانیه امکان پاسخگوئی به ده ها درخواست در مدت

زمان فوق و بدون نیاز به دنبال کردن فرآیند ارتباط با بانك اطلاعاتی و نمایش داده انجام می شود.

در زیر صفت های دیگر OutPutCaching را ملاحظه می کنید:

Location:مکانی که کش باید ذخیره شود را مشخص می کند.

VaryByCustom:می توانید عمل کش را سفارشی کنید.

VaryByParameter:می توانید کش را به پارامتر ها وابسته کنید.

VaryByHeader:می توانید کش را به صفحات چند زبانه وابسته کنید.بدین ترتیب که به ان مقدار Accept Language بدهید.

VaryByControl:مربوط به کش کردن userControls می شود.حتما باید کنترل های کاربر را کش کنید چون استفاده ای جز یک نمایش اساتیکی ساده ندارند.این بحث پس از معرفی و کاربرد UserControl مطرح می شود.

صفت Location مکانی که کش باید ذخیره شود را مشخص می کند.در کد زیر کش در Client-Side ذخیره می گردد:

```
<%@ OutputCache Duration="20" VaryByParam="none" Location="Client" %>
```

تنها کاربردی که این صفحه دارد این است که هنگامی شخصی از سایت ما بازدید می کند و درسایت ما پیمایش می کند صفحات در خود کش سمت سرویس گیرنده ذخیره می شود تا مثلا اگر یک دفعه کاربر دکمه back را فشار داد صفحه مجددا بار گذاری نشود و از صفحه ی کش شده استفاده کند و در نتیجه واکنش به زدن دکمه یback بسیار سریع خواهد بود ولی چنانچه وی صفحه را refresh کند صفحه از نو بار گزاری می شود.پس اگر صفت "Location="client باشد تنها در Navigation ها می تواند به کاربر کمک کند و ربطی به Performance سایت ما و ارتباط با سرور ندارد.

صفت Location حاوی مقادیر زیر نیز هست:

Server:یعنی شی کش در سرور ذخیره شود.

none:اصلا OutPutCache را غیر فعال می کند یعنی کش هیچ جا ذخیره نشود.

any:یعنی کش می تواند همه جا ذخیره شود client یا server و ...

ClientAndServer:یعنی کش می تواند در دو سمت سرور یا سرویس گیرنده ذخیره شود.

می خواهیم کمی بیشتر با صفت VaryByParam آشنا شویم:

کاربرد بسیار مهم این صفت در استفاده از queryStringاست.فرض کنید از یک صفحه ی معمولی یک querystring به صفحه ای که می خواهد کش شود ارسال شود.در زیر کد صفحه ی اول را میبینید:

```
<form id="form1" runat="server">
<div>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
    <asp:Button      ID="Button1"      runat="server"      Text="Button"
/> </div>
</form>
```

و کد پشت صحنه ی آن به صورت زیر است که یک queryString به نام k را به صفحه ی querystring2 ارسال می کند.این مقدار از یک textBox گرفته می شود:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim l As String = TextBox1.Text
    Response.Redirect("querystring2.aspx?k=" & l)
End Sub
```

حال به سراغ صفحه ی querystring2 می رویم.درآن صفحه کش را با تگ OutPutcache فعال کنید:

```
<%@ OutputCache Duration="20" VaryByParam="none" %>
```

و مقدار queryString دریافتی را در یک Label بنویسید:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    Label1.Text = Request.QueryString("k").ToString
End Sub
```

با اجرای صفحه ی اول و پر کردن textBox و زدن دکمه میبینید که مقداری که در textBox پر کرده بودید به صفحه ی querystring2 ارسال شده و در Label مربوطه نمایش داده می شود.تا اینجا مشکلی نیست.ولی اگر دوباره قبل از انقضای کش(۲۰ ثانیه) صفحه ی اول را باز کرده و درtextBox چیزی دیگر غیر از آنکه قبلا نوشته بودید وارد کرده و روی دکمه کلیک کنید میبینید که همان مقدار قبلی در صفحه ی querystring2 نمایش داده شده.این یک مشکل بزرگ است که asp.Net با VaryByParam آن را حل کرده.اگر به جای none مقدارش را * بگذارید از این به بعد هر QueryString که دریافت کرد از کش خارج می شود و مقدار جدیدش را نمایش میدهد البته این روش بسیار

نا کار آمد و حتی شاید بعضی اوقات صفحه را از نظر امنیتی دچار مشکل کند.همینطور می توانید نام متغیر QueryString ارسالی را به VaryByParam نسبت دهید تا کش تنها در مورد queryString خاص کنار رفته و صفحه ی جدید بار گذاری شود(این کار بهتر از قبلی(گذاشتن*است):

```
<%@ OutputCache Duration="20" VaryByParam="*" %>
<%@ OutputCache Duration="30" VaryByParam="kk" %>
```

و همچنین می توانید نه تنها یک !متغیر بلکه چندین متغیر QueryString را به VaryByParam نسبت دهید.این کار را با قرار دادن ; بین متغیر ها انجام دهید:

```
<%@ OutputCache Duration="20" VaryByParam="kkk;kkkk" %>
```

جالب اینجاست که اگر در بار دوم مقداری تکراری وارد کنید صفحه ی کش شده به شما نمایش داده می شود (در صورت نگذشتن زمان انقضای کش)ولی اگر چیزی متفاوت با قبل وارد کنید صفحه ی جدید به شما نمایش داده می شود.

VaryByCustom:حتما دیدید که صفحات مختلف ما در مرورگر های مختلف ممکن است به صورت های متفاوتی نمایش داده شود.حال اگر مثلا ما صفحه ی خود را در IE کش کنیم آنگاه ماربری با NetScape تقاضای صفحه ی ما را داشته باشد آنگاه صفحه ای که در Ie نمایش داده شده بود حالا برای مرورگر NetScape فرستاده می شود.حال اگر مواردی باشد که مربوط به اجرای متفاوت صفحه باشد آن صفحه ای که در IE ذخیره شده و حالا در NetScape نمایش داده می شود دچار مشکل خواهد شد پس ما باید نوع مرورگر ها را نیز به کش اضافه کنیم تا دچار این مشکل نشویم.برای حل این مشکل از VaryByCustom استفاده می کنیم و من نام browser را به آن می دهم:

```
<%@ OutputCache Duration="20" VaryByParam="none" VaryByCustom="browser"%>
```

ولی برای درک بیشتر از تابع GetVaryByCustomString استفاده می کنیم.این تابع نوع مرورگر را به کش معرفی می کند تا کش بتواند خود را با مرور گر های مختلف وفق دهد.این تابع را در فایل Global.asax و به صورت زیر بنویسد:

```
    Public Overrides Function GetVaryByCustomString(ByVal context As
HttpContext, ByVal arg As String) As String
        If arg = "browser" Then
            Dim browserName As String
            browserName = context.Request.Browser.Browser
            browserName &= context.Request.Browser.MajorVersion.ToString()
            Return browserName
        Else
```

```
            Return MyBase.GetVaryByCustomString(context, arg)
        End If
    End Function
```

دومین پارامتر این تابع مقداری است که ما به VaryByCustom داده بودیم.ابتدا چک می کنیم که این مقدار همانی است که به VaryByCustom داده بودیم (در اینجا Browser) سپس اگر صحیح بود نام و آخرین ورژن مرور گر را در متغیری رشته ای به نام BrowserName ریخته و سپس بر می گردانیم.

HttpCachePolicy:کلاسی است که به شما اجازه می دهد با کدنویسی عمل کش را انجام دهید.برای این کار از متد response.cache استفاده کنید.مثلا عمل OutPutCache که به عنوان اولین مثال در بحث ما زده شد به صورت زیر است:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Response.Cache.SetCacheability(HttpCacheability.Server)
        Response.Cache.SetExpires(DateTime.Now.AddSeconds(60))
        label1.Text = "The time is now:<br />" & DateTime.Now.ToString()
    End Sub
```

ما در اینجا با متد SetCacheAbility عملا کش را تعریف کردیم و آرگومان ورودی آن نیز Location آن را مشخص می کند که با متد HttpCacheAbility آن را در سرور تنظیم کردیم.همچنین با متد SetExpires مشخص کردیم که تا چه زمانی عمل کش انقضا شود و این زمان را به عنوان آرگومان ورودی به این تابع دادیم.متد DateTime.Now زمان حال حاضر را بیان می کند و متد AddSeconds نیز تعداد ثانیه ی add شده به زمان حال را نشان می دهد.پس ما در بالا گفتیم عمل کش تا ٦٠ ثانیه پس از بارگذاری صفحه(DateTime.Now) به اتمام برسد.البته DateTime.Now حاوی متد های متعددی از جمله AddMinutes می باشد که در آن به جای ثانیه دقیقه اضافه می کند.با انجام مثال زیر انگار مثال اول بخش cache را به روشی دیگر انجام داده اید.کلاس HttpCachePolicy حاوی متد های متعددی در زمینه ی caching است که به دلایل زیر تا همینجا در مورد آن بسنده می کنیم:

١-صفحه ی کد شما دچار آشفتگی می شود.

٢-درست نیست عمل کش را در رویداد هایی مثل Page_Load به کار ببریم.

البته جلوتر در بحث cacheDependency کمی با HttpCachePolicy کار خواهیم داشت (بسیار جزیی).

کش جزیی در صفحه:گاهی وقت ها دوست ندارید کل صفحه کش شود و تنها قسمت های خاصی را برای کش کردن نیاز دارید.برای این کار از ۲ نوع کش زیر استفاده کنید:

۱-Fraagment caching:که کمی در اوایل فصل در موردش صحبت شد و در بخش UserControl به آن می پردازیم.

۲-Post Cache Substitution:شاید بخواهید کل صفحه ی وب سایت خود را به جز یک قسمت خاص کش کنید.پیش از این ان قسمت هایی که می خواستید به وسیله ی کنترل کاربر طراحی می کردید و سپس کش می کردید که کار را بسیار مشکل می کرد.ولی با Post Cache Substitution این مشکل حل شده است.Post Cache Substitution باعث می شود شما کل صفحه رابه جز قسمتی که در Post Cache Substitution مشخص شده کش کنید. می خواهیم طرز کار Post Cache Substitution را در یک مثال خیلی ساده نمایش دهیم.در ابتدا همان مثالی که در اول بحث کش در مورد تاریخ زده بودیم را اینجا هم ایجاد می کنیم.ابتدا کش را ایجاد کرده:

```
<%@ OutputCache Duration="20" VaryByParam="None" %>
```

سپس تاریخ را درج می کنیم:

```
        Response.Write("This date is cached : ")
        Response.Write(DateTime.Now.ToString() & "<br />")
```

حالا دقت کنید.من برای اینکه از Post Cache Substitution استفاده کنم باید از متد Response.Write Substitution استفاده کنم که یک آرگومان ورودی بسیار مهم به نام HttpResponseSubstitutionCallback دارد که برای مقدار پذیری آن هم باید new شود:

```
Response.Write Substitution(New HttpResponseSubstitutionCallback(AddressOf  “Your shared Method Name”)
```

addressOf باعث دریافت یک متد از نوع shared شده می شود.

متد ما در اینجا GetDate نام دارد پس:

```
        Response.WriteSubstitution(New
HttpResponseSubstitutionCallback(AddressOf GetDate))
```

پس ما نیاز به یک متد از نوع shared داریم که یک ورودی از نوع HttpContext داشته باشد تا با آن بتوانیم متد های متنوعی بنویسیم:

```
    Public Shared Function GetDate(ByVal context As HttpContext) As String
        Return DateTime.Now.ToString()
    End Function
```

نام متد ما در اینجا GetDate است که در اینجا تاریخ حال حاضر را بر می گرداند.شی Context هم باعث می شود در این تابع به عناصری همچون session یا کوکی و ... به وسیله ی request دسترسی داشته باشیم از آنها برای کش نشدن استفاده کنیم.در ضمن رویداد Page_Load را به صورت زیر می نویسیم:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Response.Write("This date is cached : ")
        Response.Write(DateTime.Now.ToString() & "<br />")
        Response.Write("This date is not cache is: ")
        Response.WriteSubstitution(New
HttpResponseSubstitutionCallback(AddressOf GetDate))
    End Sub
```

حال اگر صفحه را اجرا کنید می بینید که در ابتدای صفحه یک تاریخ کش شده وجود دارد و پس از آن یک تاریخ کش نشده.در حقیقت کل صفحه کش شده به جز تاریخ دوم.برای دیدن طرز کار کافیست صفحه را مدام refresh کنید.

در این مثال صفحه برای بار اول توسط کاربر درخواست می گردد سپس بهطور معمول کل صفحه کش می شودولی چون ما پس از نمایش تاریخ از متد Response.Write Substitution استفاده کردیم صفحه توجه خود را معطوف به متدی که برای HttpResponseSubstitutionCallback تعریف کردیم می کند و خروجی آن متد را بلافاصله از کش خارج می کند.

در asp.NET 2.0 یک کنترل وجود دارد به نام Substitution وجود دارد که به وسیله ی آن می توان مثال بالا را ساده تر ایجاد کرد.به این ترتیب که این کنترل مقدار برگشتی متد shared را در خروجی نمایش می دهد.برای استفاده از این کنترل کافیست به صورت زیر عمل کنید:

```
    <asp:Substitution ID="Substitution1" runat="server"
MethodName="GetDate" />
```

صفت MethodName نا م متدی که خروجیش کش نشود را مشخص می کند که ما همان متدی را که قبلا تعریف کرده بودیم را دادیم.برای انجام مثال صفحه را به صورت زیر در آورید:

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Substitution2.aspx.vb" Inherits="Substitution2" %>
```

```
<%@ OutputCache Duration="20" VaryByParam="None" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Label ID="Label1" runat="server"></asp:Label><br />
    <asp:Label ID="label2" runat="server" Text="Date time Is No t cache is
:"></asp:Label><br />
    <asp:Substitution ID="Substitution1" runat="server"
MethodName="GetDate" />
    </div>
    </form>
</body>
</html>
```

و کد پشت صحنه را نیز به صورت زیر:

```
    Public Shared Function GetDate(ByVal context As HttpContext) As String
        Return DateTime.Now.ToString()
    End Function

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Label1.Text = "The DateTime cached Is :" & DateTime.Now
End Sub
```

در رویداد Page_Load ما تاریخ روز را چاپ کردیم و سپس کنترل Substitution نیز همین کار را با فراخوانی متد GetDate انجام می دهد با این تفاوت که مقداری که این کنترل در خروجی چاپ کرده کش نمی شود.

پس در حالت کلی هر گاه از Post Cache Substitution استفاده کردید حتما متدش را به فرم کلی زیر ایجاد کنید:

Public(or Private) Shared "your function name"(context as HttpContext) as "Your Return Value Type"
End Function

دلیل shared بودن تابع هم ایت است که باید در همه جا قابل دسترسی باشد که توابع و متغیر های shared حاوی این خصوصیت هستند.

مشکل Post Cache Substitution روش این است که تنها یک متد را می توانید از کش خارج کنید.ولی کنترل های سفارشی خود به خود از کش خارج می شوند مثلا کنترل AdRotator کنترلی است که خود به خود از کش خارج شده و تصاویر تصادفی به روزی را حتی اگر صفحه OutPutCache هم شده باشد نمایش می دهد.

:CacheProfile

یکی از مسائل در ارتباط با output caching قرار دادن کد درون صفحه است (در بخش aspx markup . و یا در بخش کد کلاس) . با این که استفاده و پیکربندی خصلت های مرتبط با دایرکتیو OutputCache در صفحات وب ساده تر بنظر می آید ولی این روش می تواند مسائل مدیریتی و پشتیبانی مختص به خود را نیز به دنبال داشته باشد (خصوصا" اگر ده ها صفحه cache شده ایجاد شده باشد) . به عنوان نمونه در صورتی که قصد داشته باشیم تغییراتی را در خصوص caching تمامی صفحات فوق انجام دهیم (مثلا" تغییر مدت زمان caching از ۳۰ ثانیه به ۶۰ ثانیه) ، می بایست هر صفحه بطور جداگانه تغییر و در ادامه نیز توسط ASP.NET مجددا" ترجمه گردند .
در ASP.NET 2.0 با معرفی یك راهکار جدید این امکان در اختیار پیاده کنندگان گذاشته شده است تا بتوانند از تنظیمات caching مشابه برای گروهی از صفحات استفاده نمایند . به ویژگی فوق profile cache می گویند و به کمك آن می توان تنظیمات caching را در یك فایل web.config تعریف نمود . بدین ترتیب ، اعمال تغییرات صرفا" از طریق یك نقطه فراهم می گردد .
برای تعریف یك Cache Profile ، از تگ <add> در بخش <outputCacheProfiles> فایل web.config استفاده می گردد . به cache profile ایجاد شده یك نام و مدت زمان مناسب نسبت داده می شود . تعریف باید درون تگ System.Web باشد:

```
<caching>
    <outputCacheSettings>
          <outputCacheProfiles>
                <add name="sharingcache" duration="70"/>
          </outputCacheProfiles>
    </outputCacheSettings>
</caching>
```

حتما باید به آن نام هم بدهید چون برای استفاده در تگ OutPutCache باید از نامش استفاده کنید تا به سایر خصوصیاتش که شما در زیر تعریف خواهید کرد دسترسی ایجاد شود.

برای استفاده از آن نیز از صفت cacheProfile تگ OutPutCache استفاده می کنیم:

```
<%@ OutputCache CacheProfile="sharingcache" VaryByParam="none" %>
```

مقداری را که باید به صفت cacheProfile بدهید همان نامی است که در تگ add به آن دادید.

حتی در web.config می توانید با خصوصیت VarByParam مقادیری را نیز به تگ Add اضافه کنید.مقادیری که در مورد QueryString در مورد آنها بحث شد.در این صورت دیگر نیازی به صفت VarByParam در تگ اصلی OutPutCache نیست.

نوع دیگری از caching به نام Disk caching نیز وجود داشت که برای صفحات و Portion Of Pages با حجم های بالا مناسب بود که کش را در حافظه یا دیسک ذخیره می کرد تا اولا سرور زیاد اذیت نشود و دوما در صورت بروز مشکل در سرور صفحه به درستی کش شود.متاسفانه این ویژگی در ورژن نهایی Asp.NET 2.0 حذف شد.

پیکر بندی کلی فایل web.config برای بحث کش به صورت زیر است:

```
<caching>
<cache
    disableExpiration="[true|false]"
    />
  <outputCache
    enableFragmentCache="[true|false]"
    enableOutputCache="[true|false]"
    sendCacheControlHeader="[true|false]"
    />
  <outputCacheSettings>
    <outputCacheProfiles>
      <clear />
      <remove name="String" />
      <add enabled="[true|false]"
        name="String"
        duration="Integer"
        varyByControl="String"
        varyByHeader="String"
        sqlDependency="String"/>
    </outputCacheProfiles>
  </outputCacheSettings>
</caching>
```

موارد بالا واضح بوده و نیازی به توضیح ندارند.

خوب بحث ما در مورد outPutCaching به پایان رسید و حال می پردازیم به DataCahcing که بحثی فوق العاده مهم و موثر در کارایی وب سایت است.

Data Caching

در اینجا بحث به کلی با OutPut Caching فرق می کند.و ما با یک شی به نام cache سرو کار داریم که باعث کش شدن یک سرس اشیا دیگر می شود.این شی بسیار مثل application است(چون به هر حال هر دو ابزاری برای مدیریت حالت هستند) ولی به دلایلی از جمله وابستگی داده با application متفاوت است که جلوتر با وابستگی کش هم آشنا می شوید.ولی این نوع کش کردن یعنی کش کردن داده ها بسیار مهم و پر کاربرد تر از OutPut Caching است.

برای اضافه کردن آیتمی به شی کش از متد Insert استفاده می کنیم.ولی این متد آرگومان های متفاوتی را در یافت می کند که در زیر مشخص شده است و شما می توانید تنها یکی از آن ها را انتخاب کنید:

```
cache.Insert(Key as string,Value as object)
```

در این حالت Key نامی از نوع رشته و Value هم شی مورد نظر جهت کش است که می تواند آرایه DataSet و یا HashTable و... باشد.

```
Cache.Insert(Key as string,Value as Object,dependency as cachedependency)
```

در این مورد هم عنصر وابستگی هم به آن اضافه شده.این عنصر مشخص می کند که کش به چه فایلی وابسته باشد تا در صورت تغییر آن فایل کش نیزتغییر کند.بحث وابستگی کش را سر جایش مفصل صحبت خواهیم کرد.ولی فعلا می توانید برای استفاده مقدار آن را Nothing بگذارید.

```
Cache.Insert(Key as string,Value as Object,dependency as cachedependency,absoluteExpiration as DateTime,SlidingExpiration as TimeSpan)
```

در اینجا دو مورد به موارد قبلی اضافه شد.اولی یعنی absoluteExpiration تاریخ انقضای آیتم کش را مشخص می کند و از نوع DateTime می باشد.مورد بعدی SildingExpiration است که از نوع TimeSpan بوده و مدت انتظار بین درخواست های متوالی از کش را مشخص می کند که اگر در آن زمان درخواستی صورت نگیرد کش پایان می پذیرد.مثلا اگر آن را برابر ۱۰ دقیقه در نظر بگیرید اگر تا ۰د دقیقه پس از آخرین درخواست از کش درخواست دیگری صورت نگیرد شی کش پاک می شود.این خاصیت می تواند بسیار مفید باشد.مثلا شما کش داده را انجام می دهید و فکر می کنید که

اگر سایت ما شلوغ باشد آنگاه در خواست ها باید زود به زود از کش صورت گیرد پس این مقدار را بسیار کم می دهید ولی اگر سایت خلوت باشد آن را زیادتر می گیرید.

در مورد این دو شی دو متد وجود دارد:

۱-**NoAbsoluteExpirtion**:که به معنای این است که شما زمانی را برای انقضا در نظر نگرفته اید.برای استفاده در مکان آرگومان **absoluteExpiration** عبارت زیر را بنویسید:

cache. NoAbsoluteExpirtion

۲-**NoSlidingExpiration**:یعنی شما زمانی را برای **SildingExpiration** در نظر نگرفته اید. برای استفاده در مکان آرگومان **SildingExpiration** عبارت زیر را بنویسید:

cache. SildingExpiration

یادتان باشد هیچ گاه همزمان نباید(و نمی توانید) دو متد بالا را همزمان به کار ببرید چون به هر حال کش یک وقتی باید پایان بپذیرد.

در صورتی که مطمئن باشیم اطلاعات موجود در یك آیتم **cache** شده در یك بازه زمانی خاص معتبر باقی می ماند (نظیر یك گزارش هواشناسی) ، استفاده از **absolute expiration** توصیه می گردد چون اگر تاریخ انقضا نباشد و شی کش شده(در اینجا گزارش هواشناسی) تغییر کند آنگاه اطلاعات قدیمی (کش شده از قبل)به دست کاربر می رسد نه جدید . در صورتی که داده ذخیره شده در **cache** همواره معتبر باشد (نظیر کاتولوگ یك محصول) ، استفاده از **Sliding expiration** توصیه می گردد .پس همواره بستگی به آیتم کش شده یکی از دو مورد بالا را فعال و دیگری را غیر فعال می کنیم.

به جای **NoSlidingExpiration** می توانید(و بهتر است) از عبارت **TimeSpan.Zero** هم استفاده کنید.

به جای **NoAbsoluteExpirtion** می توانید (و بهتر است) از عبارت **DateTime.MaxValue** استفاده کنید.

```
cache.Insert(Key as string,Value as Object,dependency as cachedependency,absoluteExpiration as DateTime,SlidingExpiration as TimeSpan,Priority as cacheItemPriority,OnRemoveCallBack as cacheItemRemoveCallBack)
```

در مورد دو آیتم جدید اضافه شده هم سر جایش بحث خواهد شد.

برای حذف آیتمی از کش هم به صورت زیر عمل کنید:

cache.Remove(“item name”)

شی cache یک متد دارد به نام Add که دو تفاوت عمده با Insert دارد :

۱-اگر از این متد استفاده کنید باید تمام آرگومان هایش را وارد کنید.آرگومان های Add دقیقا مثل آرگومان های آخرین حالت Insert است که در بالا بحث شد.

۲-متد Add قابلیت OverWrite ندارد.یعنی اگر آیتمی با یک نام خاص در کش وجود داشته باشد با متد Add دیگر نمی توان روی آن نوشت و حتما باید اول آن را پاک کرد و سپس از Add استفاده کرد. ولی Insert قابلیت OverWrite را دارد.

حال که با انواع حالت های Insert در کش آشنا شدید این را بدانید که هر وقت خواستید از کش استفاده کنید باید خالی بودن یا نبودن آن را با شرط زیر چک کنید:

```
If Cache(“your cache name”) IsNot Nothing Then
    Your cache is not empty---call your method
else
    Your cache is empty---Create caceh and then call your method
end if
```

برای پیمایش(چه دسترسی و چه حذف) روی آیتم های شی کش می توان از نوع داده ای DictinaryEntry در یک حلقه ی ForEach استفاده کرد:

```
For Each item As DictionaryEntry In Cache
Cache.Remove(item.Key.ToString())
Next item
```

حال یک مثال ساده از این بحث با هم کار می کنیم:

در رویداد Page_Load عبارت زیر را بویسید:

```
        If Page.IsPostBack Then
            Label1.Text += "The Page Is Posted Back!<br/>"
        Else
            Label1.Text += "The Page Is Created!<br/>"
    End If
```

سپس درادامه رویداد Page_Load برای ایجاد کش همانطور که گفتیم چک می کنیم آیا کش خالی است یا نه:

```
        If Cache("a") Is Nothing Then
        Else
    End If
```

اگر خالی بود یک شی کش ایجاد می کنیم و اگر پر بود شی ایجاد شده از قبل را نمایش می دهیم.شی مورد نظر برای کش در اینجا DateTime.Now است چون با داشتن تاریخ درک کش بسیار راحت می شود:

```
        If Cache("a") Is Nothing Then
            Label1.Text += "Creating Items inCache...<br/>"
```

```
            Dim j As DateTime = DateTime.Now
            Label1.Text += "Storing Items in Cashe...<br/>"
            Cache.Insert("a",   j,   Nothing,   DateTime.Now.AddSeconds(40),
TimeSpan.Zero)
        Else
            Label1.Text += "Retrieving from Cache...<br/>"
            Dim j As DateTime = CDate(Cache("a"))
            Label1.Text += "Time Is :" & j.ToString & "<br/>"
    End If
```

یک سری پیغام هم در بین انجام کار به خروجی می دهیم تا روند کار را مشخص کند.

حال یک مثال پیشرفته تر از قبلی با هم ببینیم.در این مثال من یک سری تاریخ را از یک جدول پایگاه داده استخراج می کنیم و قصد دارم آنها را در یک کنترل تقویم نمایان کنم.و چون این تاریخ ها ثابت هستند و از پایگاه داده دریافت می شوند آنها را کش می کنیم تا هر بار که صفحه بارگذاری شد اتصال با پایگاه داده برقرار نشود.

پس ابتدا باید تابعی به منظور استخراج تاریخ ها از پایگاه داده بنویسیم.برای این کار ابتدا نام تابع را می نویسم و خروجیش را از نوع HashTable قرار می دهیم:

```
  Public Function ret_date() As Hashtable
End Function
```

سپس رشته ی اتصال و موارد دیگر:

```
        Dim          cs          As          String          =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim qs As String = "SELECT date FROM cache"
        Dim con As New SqlConnection(cs)
        Dim cmd As New SqlCommand(qs, con)
    Dim reader As SqlDataReader
```

ما در ادامه می خواهیم اجرای Query مربوطه را بر عهده یExecuteReader گذاشته و سپس نتیجه را در یک sqlDataReader قرار دهیم:

```
            con.Open()
          reader = cmd.ExecuteReader
```

ولی از آنجایی که در مواردی که مقدار برگشتی از نوع SqlDataReader باشد نمی توان آن را بست(در صورتی که تابع شما حاوی مقدار برگشتی است اگر مقدار برگشتی از نوع SqlDataReader باشد نباید آن را ببندیم تا دچار خطا در استفاده از آن مقدار برگشتی نشویم پس با نبستن آن کارایی بسیار پایین می آید)پس ما آن را می خوانیم و در یک HashTable قرار می دهیم:

```
            While reader.Read
                htt(reader("date")) = reader("date")
          End While
```

دقت کنید که ما با این کار تمام مقادیری که در شی reader بود به HashTable انتقال دادیم.و آن ها را به صورت زوج نام و مقدار ذخیره کرده ایم ولی یادتان باشد که نام و مقدار در اینجا یکی هستند.یعنی reader("date") هم نام و هم مقدار است مثلا:

hashtable("4/5/2007 00:00:00")= "4/5/2007 00:00:00"

چون hashtable در قسمت نام می تواند مقداری از نوع DateTime را هم داشته باشد و مثل آرایه نیست که تنها عدد باشد و یا ArrayList که عدد ورشته باشد.

سپس reader را بسته و HashTable را برمی گردانیم.کد کلی این تابع بع همراه بلوک Tre..Catch به صورت زیر می شود:

```
    Public Function ret_date() As Hashtable
        Dim cs As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim qs As String = "SELECT date FROM cache"
        Dim con As New SqlConnection(cs)
        Dim cmd As New SqlCommand(qs, con)
        Dim reader As SqlDataReader
        Dim htt As New Hashtable
        Try
            con.Open()
            reader = cmd.ExecuteReader
            While reader.Read
                htt(reader("date")) = reader("date")
            End While
            reader.Close()
        Catch ex As Exception
            Response.Write(ex.Message)
        Finally
            con.Close()
        End Try
        Return htt
    End Function
```

به این ترتیب ما تاریخ ها را در یک hashtable برگرداندیم. سپس یک تقویم روی صفحه قرار می دهیم:

```
<asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
```

برای مشخص کردن تاریخ ها در تقویم باید به دنبال رویدادی مناسب باشیم که در هنگام رندر شدن تک تک روزها تحریک شود تا ما بتوانیم روزهای مورد نظر را مشخص کنیم.این رویداد DayRender نام دارد.ولی قبل از آن باید HashTable مربوطه را

بازیابی کنیم.این کار را در یک متغیر عمومی انجام می دهیم تا در رویداد DayRender تقویم قابل دسترس باشد:

```
Public ht As Hashtable = ret_date()
```

سپس به رویداد DayRender تقویم رفته و عبارات زیر را می نویسیم:

```
      Protected Sub Calendar1_DayRender(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.DayRenderEventArgs) Handles Calendar1.DayRender
          If ht(e.Day.Date) IsNot Nothing Then
              e.Cell.Style.Add("font-size", "12")
              e.Cell.Style.Add("font-name", "verdana")
              e.Cell.Style.Add("background", "red")
          Else
              e.Cell.Style.Add("font-size", "12")
              e.Cell.Style.Add("font-name", "verdana")
              e.Cell.Style.Add("background", "green")
          End If
    End Sub
```

من چک کردم اگر hashtable برای روز مورد نظر وجود داشته باشد اندازه ی فونت آن را ۱۲ و نامش را verdana و رنگ زمینه ی آن را قرمز کن.همانطور که می دانید مقادیر key در hashtable تاریخ هستند و برای مقایسه ی آن تاریخ با تاریخ روز رندر شده در تقویم باید از e.Day.Date استفاده کنم.e که مرجع است.متد day شماره ی زوز را می دهد ولی چون شماره با مقدار hashtable که از نوع DateTime است نمی خواند پس آن را با متد Date به تاریخ تبدیل کردم.

با متد Cell می توان به هر یک از سلول های تقویم دسترسی داشت و با style هم یه استیل آن ستون.پس با e.Cell.Style.Add می توان استیل تک تک ستون ها را مشخص کرد به شرط اینکه رویداد DayRender باشد چون در این رویداد تک تک روزها که هر یک در یک سلول هستند قابل دسترسیند.

خوب عملیات تا اینجا به درستی انجام می شود.ولی فرض کنید ۲۰۰ نفر وارد سایت شما می شوند.با هر بار ورود هر یک از آنها به سایت شما یک بار تابع ret_date صدا زده می شود و سپس connection باز می شود.از آنجایی هم که کنترل تقویم قابل پیمایش است و در کنارش دکمه ها و عوامل PostBack وجود دارند بدون شک با ۲۰۰ نفر حداقل حدود ٤ یا ٥ برابر یعنی ۱۰۰۰ بار تابع ret_date صدا زده می شود و سپس connection باز می شود.این یک عامل بسیار مهم در افت کارایی سایت شماست.فرض کنید این تاریخ هایی که ما روی تقویم مشخص کده ایم تاریخ های مسابقات

والیبال یک باشگاه باشد.به ندرت تاریخ مسابقات تغییر می کند.این مثل یک لیست محصولات می ماند که ۱۰۰۰ نفر خواهان دیدن آن هستند و این لیست باید از پایگاه داده بازیابی شود. حالا می خواهیم با کش کردن این مشکل را حل کنیم.اول مقداری که به متغیر عمومی از نوع hashtable داده بودیم را حذف می کنیم و می گذاریم به شکل زیر بماند تا با هر بار بارگذاری صفحه تابع ret_date صدا زده نشود و اتصال با پایگاه داه برقرار نشود:

```
Public ht As Hashtable
```

سپس ما hashtable بازیابی شده از تابع ret_date را کش می کنیم چون لیست تاریخ ها در آن قرار دارد.برای این کار به رویدادPreRender (رویدادی که قبیل از آماده سازی نهایی کنترل اتفاق می افتد-در اینجا هم این رویداد قبل از dayRender رخ می دهد) کنترل تقویم رفته وابتدا چک می کنیم اگر کش وجود ندارد آن را ایجاد کن ولی اگر وجود دارد از کشی که قبلا ایجاد شده استفاده کن و دیگر به تابع ret_date وصل نشو:

```
    Protected Sub Calendar1_PreRender(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Calendar1.PreRender
        If Cache("datelist") Is Nothing Then
            ht = ret_date()
            Cache.Insert("datelist",          ht,          Nothing,
DateTime.Now.AddMinutes(2), TimeSpan.Zero)
        Else
            ht = CType(Cache("datelist"), Hashtable)
        End If
    End Sub
```

نام شی کش dateList است.در خط اول عدم وجود کش شرط شده که اگر درست باشد و کش نباشد(صفحه برای اولین بار درخواست شده باشد) آنگاه عمل بازیابی را از تابع ret_date انجام بده و آن را در داخل متغیر عمومی ht از نوع hashtable قرار بده و عمل ایجاد کش را با متدInsert و شیht(همان hashtable بازیابی شده) و زمان انقضای ۲ دقیقه انجام بده.در غیر این صورت یعنی اگر کش از قبل وجود داشته باشد مقدار کش را به ht بده و دیگر به تابع ret_date کاری نداشته باش.

دیگرچیزی تغییر نمی کند.حالا اگر ۲۰۰ نفر وارد سایت ما شوند در هر دو دقیقه تنها یک بار اتصال با پایگاه داده برقرار می شود در حالیکه بدون کش به ۱۰۰۰ بار اتصال نیاز بود.اگر از تاریخ ها مطین تر بودید می توانید تاریخ انقضای کش را بیش از ۲ دقیقه قرار دهید و در عوض SildingExpiration را بیشتر کنید. مثلا:

```
Cache.Insert("datelist", ht, Nothing, DateTime.Now.AddMinutes(10),
TimeSpan.FromMinutes(30))
```

این مثال یک مثال کاربردی از dataCaching همراه با اتصال به پایگاه داده بود.

Cache Priority:عنصری است که وعده اش را قبلا هم داده بودیم.این شی مشخص کننده ی اولویت در آیتم ها ی کش است(در صورتی که بیش از یک شی کش شود).هنگامی که حافظه دارد پر می شود و جایی برای کش کردن نمانده آیتم ها یکی یکی باید حذف شوند.و شما می توانید این حذف را اولویت بندی کنید.مقادیری که Priority دریافت می کند از زودترین حذف تا دیرترین به صورت زیر است:

Low

Below Normal

Normal

Above Normal

High

مقدار پیش فرض این شی Normal است.

مثلا اگر:

```
CacheItemPriority.Low
```

باشد این عنصر زودتر از هم پاک می شود.ولی اگر نخواهید به هیچ وجه عنصری حذف شود گزینه ی NonRemovable را انتخاب کنید.این در صورتی است که ایتم کش شده ی شما برای سایت بسیار حیاتی باشد.

DataSource Caching:

کنترل های ObjectDataSource، SqlDataSource و XmlDataSource بطور ذاتی از امکانات caching حمایت می نمایند . استفاده از caching به همراه کنترل های فوق اکیدا" توصیه می گردد چراکه برخلاف کد سفارشی نوشته شده توسط پیاده کنندگان به منظور دستیابی داده ، کنترل های منبع داده همواره در هر postback یك query را بر روی منبع داده اجراء می نمایند .

کنترل های فوق ، همچنین برای هر کنترل نسبت دهی یك query در سطح منبع داده را اجراء می نمایند . به عنوان نمونه اگر در یك صفحه از سه کنترل نسبت دهی داده در ارتباط با یك منبع داده یکسان استفاده شده باشد ، سه query مجزاء بر روی بانك

اطلاعاتی و قبل از تفسیر و ارسال صفحه برای سرویس گیرنده ، اجراء خواهد شد . بدیهی است حتی با استفاده از امکانات اندك caching به همراه کنترل های منبع داده ، شاهد بهبود چشمگیر کارآئی و کاهش load عملیاتی در سمت سرویس دهنده خواهیم بود.

کنترا های منبع داده از صفت های مشابهی برای عمل caching استفاده می کنند که در زیر اهم آنها آمده:

۱-Enable Caching:دو مقدار true یا false می پذیرد و به ترتیب باعث فعال بودن کش و غیر فعال بودن کش در کنترل منبع داده ی مورد نظر می شود.

۲-CacheDuration:زمان انقضای کش را با توجه به عبارت زیر تعیین می کند.

۳-cacheExpirationPolicy:آیتمی است که دو انتخاب در اختیار شما می گذارد:

sliding:موجب می شود که زمان مشخص شده توسط شما در صفت CacheDuration مربوط به SlidingExpiration شود که همان زمان انقضا در صورت عدم در خواست های متوالی است.

Absolute: موجب می شود که زمان مشخص شده توسط شما در صفت CacheDuration مربوط به AbsoluteExpiration شود.مقدار پیش فرض AbsoluteExpiration می باشد.

٤-CacheKeyDependency & SqlCacheDependency:مربوط به بحث وابستگی داده ها است که سر جایش توضیح داده خواهد شد.

برای مثال یک DropDownList حاوی یک سری شهر ایجاد می کنیم و قصد داریم با انتخاب هر شهر نام مشتریانی که در آن شهر حساب دارند را نمایش دهیم(البته صفتAutoPostBack را برابر True کنید):

```
        <asp:DropDownList        ID="dropDownList1"        runat="server"
AutoPostBack="true">
            <asp:ListItem Text="karaj"></asp:ListItem>
            <asp:ListItem Text="qazvin"></asp:ListItem>
            <asp:ListItem Text="tehran"></asp:ListItem>
            <asp:ListItem Text="zanjan"></asp:ListItem>
        </asp:DropDownList>
```

سپس کنترل sqldatasource را ایجاد می کنیم.با true کردن صفت EnableCacching و همینطور تعیین زمان ۳۰ ثانیه برای انقضای کش:

```
        <asp:SqlDataSource        ID="SqlDataSource1"        runat="server"
EnableCaching="true"  CacheDuration="30"  ProviderName="System.Data.SqlClient"
```

```
ConnectionString="<%$ ConnectionStrings:aaa %>" SelectCommand="Select * From
customer Where customer_city=@city" >
          <SelectParameters>
          <asp:ControlParameter      ControlID="DropDownList1"      Name="city"
PropertyName="selectedValue" />
          </SelectParameters>
        </asp:SqlDataSource>
```

و یک GridView برای نمایش:

```
<asp:GridView ID="gridview1" runat="server" DataKeyNames="customer_name"
DataSourceID="sqldatasource1"></asp:GridView>
```

در مثال فوق ، پس از انتخاب شهر توسط کاربر ، یك query جداگانه اجراء خواهد شد تا لیست کارکنان با توجه به شهر انتخاب شده ، بازیابی و در یك DataSet به میزان ۱۰ دقیقه (۶۰۰ ثانیه) ، cache گردد . در صورت انتخاب یك شهر دیگر توسط کاربر ، پردازش فوق تکرار و مجددا" یك DataSet جدید ایجاد و cache می گردد. در صورت انتخاب یك شهر توسط کاربری که قبلا" توسط کاربران دیگر انتخاب شده است ، DataSet مورد نظر از cache بازیابی خواهد شد (مشروط به عدم اتمام مدت زمان اعتبار حضور آن در cache) .

توجه داشته باشید زمانی که مقدار خصلت DataSourceMode معادل DataSet در نظر گرفته شده باشد (مقدار پیش فرض) ، پتانسیل caching در کنترل منبع داده SqlDataSource به خوبی کار می کند . شی DataReader نمی تواند بطور موثر cache گردد چراکه شی فوق قادر به برقراری یك ارتباط مستقیم و زنده با بانك اطلاعاتی نمی باشد.

یک مشکل در مثال بالا این بود که برای هر پارامتر(در اینجا نام هر شهر)مجبور هستیم یک query جداگانه اجرا کرده و سپس آن query را در کش ذخیره نماییم.فرض کنید به جای ٤ شهر ۱۰۰ شهر داشتیم.پس باید ۱۰۰Query جداگانه اجرا شوند و یکی یکی در کش ذخیره شوند.پس می بینید که کارایی سایت ما باز هم پایین می آید.برای جلوگیری از این مشکل تغییراتی را در کنترل sqlDataSource بالا می دهیم.هدف ما این است که تعداد اجرای query را برای ۱۰۰ شهر از ۱۰۰ به ۱ برسانیم.برای این کار اول Query را به فرم زیر در می آوریم:

Select * From Customer

پس تا اینجا شرط Where را برداشتیم.پس نیازی به تگSelectParameter نیست پس آن را هم پاک کنید.حالا می خواهیم Where را به گونه ای دیگر اعمال کنیم.به این منظوذ به جای تگ SelectParameter از تگ filterParameter استفاده می کنیم.این تگ باعث می شود که جدول کلی انتخاب شده توسط select * From customer را بتوان فیلتر کرد.همانطوری که در شی DataView داشتیم.ولی قبل از آن باید از صفت FilterExpression کنترل SqlDataSource استفاده کنیم.مقداری که این صفن می گیرد همان مقداری است که باید جلوی شرط where به کار ببریم:

filterExpression=”customer_city=’{0}’”

این یعنی ابتدا تمام سطر های جدول انتخاب شده و در dataSet موجود می باشد و سپس آن جدول را فیلتر می کنیم.عبارت {۰} هم می شناسید که بیانگر همه چیز است.یعنی هر رشته ای که در filterParameter ذکر شود.پس ما نام مشتریانی را که شهرشان هرچیز که در filterParameter ذکر شود را بیرون آوردیم.حالا در filterParameter عینا" همان چیزی می آوریم که در مثال قبل درselectParameter آورده بودیم.پس کنترل SqlDataSource ما به شکل زیر در می آید:

```
        <asp:SqlDataSource        ID="SqlDataSource1"        runat="server"
ProviderName="System.Data.SqlClient"                    ConnectionString="<%$
ConnectionStrings:aaa    %>"    SelectCommand="Select    *    From    customer"
FilterExpression="customer_city='{0}'"                   EnableCaching="true"
CacheDuration="30" >
        <FilterParameters>
        <asp:ControlParameter     ControlID="DropDownList1"     Name="city"
PropertyName="selectedValue" />
        </FilterParameters>
      </asp:SqlDataSource>
```

توجه داشته باشید در صورت عدم استفاده از caching ، ضرورتی به فعال کردن فیلترینگ وجود ندارد . چراکه در صورت استفاده از فیلترینگ بدون caching ، در واقع تمامی result set بازیابی خواهد شد تا در ادامه بتوان بخشی از رکوردهای آن را بازیابی کرد . بدین ترتیب ، پس از هر postback و بدون توجیه منطقی تمامی رکوردها (بیش از آن چیزی که مورد نیاز است) ، بازیابی می گردد .

برای کش کردن با کنترل ObjectDataSource باید مثل موردی که در اول بحث

DataCaching ذکر شد عمل کنید یعنی وقتی دارید کلاس مربوطه را ایجاد می کنید خودتان عمل caching را انجام دهید.

این کار را تنها برای توابعی از آن کلاس می توانید ایجاد کنید که مقدار برگشتی آنها از نوع DataSet و یا DataTable باشد و این دو شی را در همان کلاس کش کنید.

Cache Dependencies

مهمترین تفاوت بین شی cache و application وابستگی است. ماهیت منابع داده نظیر یك بانك اطلاعاتی بگونه ای است که به مرور زمان اطلاعات درون آنها تغییر می یابد . در صورتی که در یك برنامه از caching استفاده می گردد ، همواره این احتمال وجود خواهد داشت که اطلاعات موجود در cache متاثر از این تغییرات نباشد و داده بهنگام نشده از cache استخراج و در اختیار کاربران گذاشته شود.
برای کمك در جهت حل این نوع مشکلات ، ASP.NET از caching با وابستگی حمایت می نماید . با استفاده از ویژگی فوق این امکان در اختیار پیاده کنندگان قرار می گیرد تا بتوانند حضور یك آیتم در cache را به منابع دیگری وابسته نمایند . در چنین مواردی ، زمانی که در منبع مورد نظر تغییراتی ایجاد گردید ، آیتم cache شده بطور اتوماتیك از cache خارج می گردد .
ASP.NET از سه نوع وابستگی حمایت می نماید :

وابستگی با سایر آیتم های Cache شده

وابستگی با فایل ها و یا فولدرها

وابستگی با یك query بانك اطلاعاتی

برای ایجاد یک وابستگی از کلاس CacheDependency استفاده می کنیم و آن را New می کنیم.برای new کردن هم باید نام فایلی که کش به ان وابسته می شود را بدهیم:

```
dim dep as new cachedependency("file name")
```

تنها نام فایل در تابع سازنده کفایت نمی کند.به همین خاطر با متد server.MapPath به آدرس آن فایل دسترسی خواهیم داشت.متد server.MapPath به صورت دینامیک آدرس محلی فایلی که نامش را به عنوان آرگومان ورودی دریافت کرده را به ما می

دهد.به این ترتیب که مسیری که پوشه ی وب سایت ما در آن قرار دارد را به نام فایل مورد نظر می چسباند.مثلا اگر به فرم زیر باشد:

server.MapPath(“aaa.txt”)

آنگاه در داخل سرور(که در اینجا پوشه ی وب سایت ماست) با متد mapPath مسیری که پوشه ی وب سایت ما قرار دارد را به مسیری که متد server.MapPath به عنوان ورودی دریافت می کند(حال ممکن است مسیر باشد یا یک نام ساده ی یک فولدر و یا نام یک فایل) می چسباند و حاصل را برمی گرداند خروجی کد بالا از نوع رشته ای و یک چیزی به فرم زیر است:

C:\my practices\practice18\xsd1.xml

البته اگر روی سرور اصلی این کد را بنویسید یک آدرس دیگر به شما داده می شود و آن هم آدرس محلی مربوط به پوشه ی شماست که در داخل سرور قرار دارد ولی چون در اینجا پوشه ی وب سایت شما در درایو C و فولدر my practice قرار داشت آدرس بالا برای شما نمایش داده شد چون در اینجا سرور محلی است در داخلMyComputer به دنبال فایل مربوطه می گردد.حال اگر فایل مربوطه در سرور هم نباشد بازآدرس را می دهد.مثلا فرض کنید فایل ssd در سرور نباشد و من کد زیر را اجرا می کنم:

server.MapPath(“ssd.txt”)

آنگاه باز هم خروجی به شکل زیر خواهد بود:

C:\my practices\practice18\ssd.xml

بنابراین با متد server.MapPath شما دراصل به پوشه ی مورد نظر در سرور دسترسی خواهید داشت چه فایل مورد نظر در پوشه باشد چه نباشد.ولی کلاس ها و متد هایی هستند که از آدرس حاصل از server.MapPath استفاده می کنند که اگر فایل در آن آدرس نباشد آنگاه از شما خطا گرفته می شود برای مثال هم به ادامه ی مثال قبل توجه کنید:

پس اگر من یک فایل ایجاد کنم و آن را در پوشه ی وب سایتم قرار دهم کلاس CacheDependency به صورت زیر می شود:

dim dep as new cachedependency(server.mappath(“file name”))

در این متد اگر فایل مورد نظر وجود نداشته باشد(در پوشه ی سرور) آنگاه از سوی CacheDependency از شما خطا گرفته می شود.

به این ترتیب وابستگی را می توانیم تولید کنیم.حال کافیست آن را به متد Insert شی کش متصل کنیم:

cache.Insert(“cache name”,”item to cache”,dep)

بنابراین دیگر نیازی نیست به کش خود زمان انقضا بدهید.

برای درک وابستگی یک مثال از آن را با هم می بینیم.ولی قبل از آن یادتان باشد از این به بعد در کل مثال ها و دربحث وابستگی کش می بایست رشته ی اتصال خود را چک کنید اگر در آن Pooling تعریف شده بود آن را پاک نمایید.دراین مثال من یک دکمه و یک Labelدر صفحه قرار می دهم.ولی هیچ رویدادی به دکمه نسبت نمی دهم.پس با کلیک دکمه صفحه PostBack می شود.سپس می خواهم وضعیت کش در هر PostBack برای من مشخص شود.کش را به یک فایل txt به نام newdep نسبت می دهم.آیتمی هم که در کش ذخیره می کنم همان تاریخ حال حاضر است.پس یک تابع می نویسم به نام CreateCache که کار ساختن کش و وابستگی را انجام دهد:

```
Public Sub createcache()
    Label1.Text += "Create Dependency Item<br />"
    Dim dependency As New CacheDependency(Server.MapPath("newdep.txt"))
    Dim item As DateTime = DateTime.Now
    Label1.Text += "Adding Item To Cache...<br />"
    Cache.Insert("aaaa", item, dependency)
    Label1.Text += "Item Added to Cache<br />"
End Sub
```

در بین کدها مراحل انجام شده را در یک Label می نویسم تا در خروجی به طور کامل ملاحظه شود.

اگر به هر نحوی در اجرای این مثال با مشکل برخورد کردید یا کش به درستی کار نکرد از متد Add به جای Insert استفاده کنید.از آنجایی که باید تمام آرگومان ها را به آن داد شمایک زمان انقضای نسبتا طولانی مثلا ٦٠٠٠ ثانیه برایش انتخاب کنید .در ضمن قبل از Add کردن باید آیتم قبلی را Remove کنید. تابع بالا به جای متد Insert متد Add را به صورت زیر می توان ایجاد کرد:

```
   Cache.Remove("aaaa")
 Cache.Add("aaaa", item, dependency, DateTime.Now.AddSeconds(6000),
TimeSpan.Zero, CacheItemPriority.Normal, Nothing)
```

من همه ی آرگومان ها را مقدار دهی کردم به جز آیتم آخر که RemovalCallBack نام دارد و جلوتر با آن آشنا می شوید.خوبی متد Add این است که محکم کاری لازم را

انجام می دهد.یعنی آیتم قبلی را به کلی پاک می کند و سپس آیتم جدید را قرار می دهد.و اگر از آن استفاده کنید با مشکل روبرو نخواهید شد.

سپس در ادامه از آنجایی که عمل Test کش با هر PostBack شدن صفحه ایجاد می شود من در رویدادPage_Load شرط مربوط به PostBack شدن را ذکر می کنم:

```
      Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
          If Page.IsPostBack Then

          End If
    End Sub
```

سپس در داخل شرط if که وقتی رخ می دهد که صفحه PostBack شده وجود یا عدم وجود کش را چک می کنم.اگر وجود داشت که محتویات کش را در خروجی چاپ کن و اگر نداشت تاریخ حال حاضر را چاپ کن و سپس کش را با صدا زدن تابع createcache بساز:

```
    If Cache("aaaa") IsNot Nothing Then
        Label1.Text += "retrieving Item From Cache...<br />"
        Label1.Text += "Time Now is" & Cache("aaaa").ToString & "<br/>"
    Else
        Label1.Text += "Cannot retrieving Item From Cache Because cache is
nothing<br />"
        Label1.Text += "Time Now is " & DateTime.Now & "<br/>"
        createcache()
  End If
```

باز هم مراحل انجام شده را در خروجی چاپ می کنیم.

مراحل کار به این صورت است که در ابتدا صفحه بار گذاری می شود و شما تنها یک دکمه روی صفحه می بینید.با اولین بار کلیک روی دکمه پیغام cannot retrieving... برای شما ظاهر می شودچون هنوز کش ساخته نشده.ولی با کلیک بعدی کش ساخته شده و پیغام retrieving Item From... برای شما ظاهر می شود سپس هر چند بار کلیک کنید این پیغام تکرار می شود.حال برای تست وابستگی در عین حال که مرورگر شما باز است کافیست به سراغ فایل newdep.txt رفته و سپس روی آن تغییراتی انجام دهید و آن را ذخیره کنید.حال اگر روی دکمه ی روی صفحه ی مرورگر کلیک کنید این بار با پیغام cannot retrieving... مواجه می شوید.ولی با کلیک بعدی مجددا پیغام retrieving Item From... می آید یعنی ما کش را نیز به روز کردیم.چون با تغییر فایل وابسته کش حذف می شود ولی با حذف کش ما با صدا زدن تابع createcache آن را می سازیم (در

خقیقت یک OverWrite با متد Insert انجام می دهیم)به این صورت که مجددا وابستگی را برای کش تعریف می کنیم(یعنی فایل تغییر داده شده را به عنوان وابستگی کش در نظر می گیریم) و دوباره کش را با همان نام قبلی و مقدار جدید(insert (DateTime.Now می کنیم.دقت کنید تغییر در فایل وابسته منجر به حذف کش می شود نه بروز شدن آن.ما در اینجا دستی خودمان آن را بروز کردیم.

این وابستگی در این مثال تنها به یک فایل بود.شما می توانید وابستگی را به چندین فایل نسبت دهید تا هر یک از فایل ها که تغییر کرد کش هم حذف شود.برای این کار از Aggregate Dependencies استفاده می کنیم.این شی به صورت زیر تعریف می شود:

```
Dim agg As New AggregateDependency()
```

تابع سازنده ی این شی آرگومان ورودی ندارد ولی پس از new کردن می توان از متد add آن استفاده کرد:

```
agg.Add("Array Of CacheDependency")
```

همانطور که میبینید مقداری که متد add در یافت می کند آرایه ای از وابستگی هاست.برای ایجاد این آرایه اول به تعداد دلخواه وابستگی ایجاد می کنیم.سپس آرایه را به صورت زیر تعریف می کنیم:

```
Dim arrdep As CacheDependency=New CacheDependency(){"a","b","c",...}
```

نام های a-b-c همان وابستگی ها هستند.

حال مثال قبل را با چندین وابستگی انجام می دهیم.برای این کار به تابع createcache رفته موارد زیر را در آن می نویسیم.

من ابتدا ۳ وابستگی ایجاد می کنم:

```
  Dim depi As New CacheDependency(Server.MapPath("dependency.txt"))
  Dim depj As New CacheDependency(Server.MapPath("dependency1.txt"))
Dim depk As New CacheDependency(Server.MapPath("dependency2.txt"))
```

سپس آرایه ای از CacheDependency را تشکیل می دهم و اعضایش را وابستگی های بالا تعریف می کنم:

```
Dim dependency As CacheDependency() = New CacheDependency() {depi, depj,
depk}
```

حال AggregateDependency را تعریف می کنم:

```
Dim aggdependency As New AggregateCacheDependency()
```

سپس از متد add استفاده کرده و آرایه ی وابستگی ها را به ان می دهم:

```
aggdependency.Add(dependency)
```

سپس زمان حال حاضر را در یک متغیر ریخته :

```
Dim item As DateTime = DateTime.Now
```

حال در متد insert به جای dependency , aggregateDependency را به کار می بریم:

```
Cache.Insert("bbbb", item, aggdependency)
```

نام کش من در این مثال یا مثال قبلی فرق دارد.یادتان باشد دو کش همنام را هیچ جا به کار نبریم تا دچار مشکل نشویم چون از این حیث کش مثل Application بوده و در سرتاسر برنامه یکی است.پس تابع به صورت کلی به فرم زیر است:

```
      Public Sub createcache()
          Label1.Text += "Create Dependency Item<br />"
          Dim depi As New CacheDependency(Server.MapPath("dependency.txt"))
          Dim depj As New CacheDependency(Server.MapPath("dependency1.txt"))
          Dim depk As New CacheDependency(Server.MapPath("dependency2.txt"))
          Dim dependency As CacheDependency() = New CacheDependency() {depi,
depj, depk}
          Dim aggdependency As New AggregateCacheDependency()
          aggdependency.Add(dependency)
          Dim item As DateTime = DateTime.Now
          Label1.Text += "Adding Item To Cache...<br />"
          Cache.Insert("bbbb", item, aggdependency)
          Label1.Text += "Item Added to Cache<br />"
    End Sub
```

در اینجا هم می توانید به جای متد Insert ازAdd استفاده کنید که به صورت زیر می شود:

```
   Cache.Remove("bbbb")
 Cache.Add("bbbb", item, aggdependency, DateTime.Now.AddSeconds(6000),
TimeSpan.Zero, CacheItemPriority.Normal, Nothing)
```

طرز کار این برنامه مثل مثال قبلی است با این تفاوت که اگر در هر یک از ۳ فایل txt تغییر ایجاد کنید انگاه کش حذف می شود.و ما باز هم یک کاری می کنیم که بروز شود:

```
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
     If Page.IsPostBack Then
        If Cache("bbbb") IsNot Nothing Then
          Label1.Text += "retrieving Item From Cache...<br />"
          Label1.Text += "Time Now is" & Cache("bbbb").ToString & "<br/>"
        Else
          Label1.Text += "Cannot retrieving Item From Cache Because cache is
nothing<br />"
          Label1.Text += "Time Now is " & DateTime.Now & "<br/>"
          createcache()
        End If
```

```
    End If
End Sub
```

حال می خواهیم با RemovedCallBack آشنا شویم.این شی آخرین آرگومان متد Insert پس از Priority است و به وسیله ی آن می توان هنگام رویداد پاک شدن یک آیتم از کش برنامه نویسی کرد.به دو روش می توان ان را ایجاد کرد.

۱-می توان در داخل متد Insert آن را تعریف کرد:

```
Insert(...,New CacheItemRemovedCallBack(AddressOf  "function name")
```

مشخص است که ما یک شی CacheItemRemovedCallBack را New کردیم.این کار را با آرگومان ورودی نام یک تابع و همچنین کلمه ی کلیدی AddressOf(که خودش هنگام باز کردن پرانتز ایجاد می شود و الزامی است).

۲-می توان ان را جدا تعریف کرد و سپس به Insert اضافه کرد:

```
Dim removal As New CacheItemRemovedCallBack(AddressOf "function name")
Insert(...,removal)
```

حال می خواهیم بدانیم این تابع که پس از کلمه ی کلیدی AddressOf می آید چیست.این تابع هنگامی که یک آیتم کش حذف می شود توسط CacheItemRemovedCallBack صدا زده می شود و به ما اجازه ی کد نویسی می دهد و باید به فرم زیر تعریف شود:

```
    Private Sub ItemRemovedCallback(ByVal key As String, ByVal value As
Object, ByVal reason As CacheItemRemovedReason)

    End Sub
```

که نامش اختیاری است ولی آرگومان هایش باید دقیقا مثل بالا باشد.اولین آرگومان همانطور که از نامش بر می آید نام آیتم کش است.دمی هم مقدارش است.یعنی شما با استفاده از آرگومان های Key & Value می توانید به مقادیر کش دسترسی داشته باشید.این آرگومان ها توسط متد CacheItemRemovedCallBack فرستاده می شوند.نوبت می رسد به سومین آیتم.این آیتم نامش reason است به معنای دلیل و آیتمی هوشمند است.این آرگومان دلیل حذف آیتم از کش را مشخص می کند.چون همانطور که گفتیم این تابع وقتی صدا زده می شود چون همانطور که گفتیم این تابع هنگامی صدا زده می شود(توسط CacheItemRemovedCallBack) که آیتمی از کش حذف گردد.مقادیری که reason خواهد داشت در زیر آمده است:

۱-DependencyChanged:دلیل حذف آیتم از کش را تغییر در فایل یا آیتم وابسته می داند البته اگر وجود داشته باشد.

۲-Expired:دلیل حذف آیتم کش را به اتمام رسیدن زمان انقضا می داند.

۳-Removed: دلیل حذف آیتم کش را حذف با برنامه نویس توسط خود کاربر می داند.

٤-UnderUsed: دلیل حذف آیتم کش راحذف به خاطر کمبود حافظه برای ذخیره ی کش می داند.

دراینجا یک مثال با هم می بینیم که در آن دو آیتم کش وجود دارند که می خواهیم با حذف یکی دیگری هم پاک شود.ساب روتین createCahche را به صورت زیر می نویسیم:

```
  Public Sub createcache()
          Dim item1 As String = "cache item A"
          Dim item2 As String = "cache item B"
          Label1.Text += "Adding Item To Cache...<br />"
          Cache.Insert("cccc",  item1,  Nothing,  DateTime.Now.AddSeconds(30),
TimeSpan.Zero,                CacheItemPriority.Default,                New
CacheItemRemovedCallback(AddressOf ItemRemovedCallback))
          Cache.Insert("dddd",  item2,  Nothing,  DateTime.Now.AddSeconds(30),
TimeSpan.Zero,                CacheItemPriority.Default,                New
CacheItemRemovedCallback(AddressOf ItemRemovedCallback))
          Label1.Text += "Item Added to Cache<br />"
 End Sub
```

روال ItemRemovedCallBack را به صورت زیر می نویسیم:

```
      Private Sub ItemRemovedCallback(ByVal key As String, ByVal value As
Object,ByVal reason As CacheItemRemovedReason)
          If key = "cccc" OrElse key = "dddd" Then
              Cache.Remove("cccc")
              Cache.Remove("dddd")
          End If
    End Sub
```

در این روال از reason چیزی به میان نیامده.و چک شده که نام آیتم کش شده ی ارسالی cccc است یاdddd.هر کدام از این دو بودند دیگری هم حذف می شود.برای اینکه نمیدانیم کدامشان حذفشده اند ما هر دو را حذف کردیم(با خاین کار با خطا مواجه نمی شویم).در اینجا می خواهم کمی در مورد OrElse و مشابهینش صحبت کنیم.

OrElse با Or چه فرقی دارد.مثال ی را ببینید:

If a=”ali” Or a=”reza” Then...

در اینجا ابتدا چک می شود که آیا a=”ali” هست یا نه.سپس چک می شود که a=”reza” هست یا نه.

حال به مثال زیر دقت کنید:

If a=”ali” OrElse a=”reza” Then...

در اینجا ابتدا چک می شود که آیا a=”ali” هست یا نه.اگر بود دیگر a=”reza” چک نمی شود.پس تفاوتشان کمی بهینه سازی در OrElse است.

برای And و AndOlso هم ماجرا چنین است:

If a=”ali” And a=”reza” Then...

در اینجا ابتدا چک می شود که آیا a=”ali” هست یا نه.اگر حتی نباشد(یعنی شرط ما نقض شود) باز هم به سراغ a=”reza” می رود ولی اگر به جای and از AndOlso استفاده می کرد با نقض شرط اول اصلا به سراغ شرط دوم نمی رفتیم.

سپس رویداد Page_Load را به صورت زیر می نویسیم:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
      If Page.IsPostBack Then
       If Cache("cccc") IsNot Nothing Then
         Label1.Text += "retrieving Item From Cache indexiA...<br />"
         Label1.Text += "Items In Cache item A:" & Cache("cccc").ToString &
"<br/>"
         Label1.Text += "retrieving Item From Cache indexiB...<br />"
         Label1.Text += "Items In Cache item B:" & Cache("dddd").ToString &
"<br/>"
       Else
         Label1.Text += "Cannot retrieving Item From Cache Because cache is
nothing<br />"
         createcache()
       End If
      End If
  End Sub
```

در اینجا من تنها آیتم cccc را چک کردم.دلیلش این است که در مواقع معمولی(قبل از حذف) هر دو آیتم کش cccc و dddd پر بوده و تست یکی ازآنها مافی است.و وقتی cccc پاک می شود dddd هم با آن پاک می شود و در نتیجه نیازی به چک کردن dddd نیست.به عبارت دیگر یا هر دو هستند و یا هیچکدام نیستند.

حال می ماند عمل حذف.حذف آیتم کش cccc را در رویداد کلیک یک دکمه انجام می دهیم:

```
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Label1.Text += "Removing Item Cache indexA ...<br />"
        Cache.Remove("cccc")
    End Sub
```

در اینجا reason از نوع removed است چون ما دستی آن را حذف کردیم.در این مثال هم باید یک دکمه جهت انجام PostBack بگذارید.با اجرای این برنامه وقتی روی دکمه ی Button1 کلیک کنید آیتم کش cccc پاک می شودو توسط callback با عث حذف dddd خم می شود و پس از آن اگر روی دکمه ی مربوط به PostBack کلیک کنید اثری از هیچ یک نمی بینی و با پیغام ...Cannot Retrieving مواجه خواهید شد.

برای درک reason هم در مثال بالا تغییراتی می دهیم.ابتدا تابع createcache را به صورت زیر بنویسید:

```
    Public Sub createcache()
        Dim item1 As String = "cache item A"
        Dim item2 As String = "cache item B"
        Label1.Text += "Adding Item To Cache...<br />"
        Dim removal As New CacheItemRemovedCallback(AddressOf
ItemRemovedCallback)
        Cache.Insert("ffff", item1, Nothing, DateTime.Now.AddSeconds(30),
TimeSpan.Zero, CacheItemPriority.Default, removal)
        Cache.Insert("eeee", item2, Nothing, DateTime.Now.AddSeconds(30),
TimeSpan.Zero, CacheItemPriority.Default, removal)
        Label1.Text += "Item Added to Cache<br />"
    End Sub
```

رویداد کلیک دکمه را هم به صورت قبل بنویسید با این تفاوت که نام آیتم کش فرق می کند:

```
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Label1.Text += "Removing Item Cache indexA ...<br />"
        Cache.Remove("ffff")
    End Sub
```

در اینجا RemovedCallBack را به روش دوم ایجاد کردیم.هیچ فرقی با بالایی ندارد و می توانید آن را به صورت قبلی هم بنویسید.در این تابع تنها نام آیتم های کش را عوض کردیم.سپس به تابع ItemRemovedBack بروید و یک شرط بر سر شرط قبلی موجود در آن بیاورید به صورت زیر:

```
            If reason = CacheItemRemovedReason.DependencyChanged Then
                If key = "ffff" OrElse key = "eeee" Then
                    Cache.Remove("ffff")
                    Cache.Remove("eeee")
                End If
            End If
```

در این شرط از آرگومان reason ارسالی استفاده کردیم و گفتیم اگر حذف آیتم ارسالی از نوع DependencyChanged باشد آنگاه اقدام به حذف دیگری کن.در حالیکه این طور نیست و حذف از نوع removed بوده است.پس با حذف آیتم ffff از کش دیگر آیتم eeee حذف نمی شود.برای امتحان آن رویداد Page_Load را به صورت زیر بنویسید:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If Page.IsPostBack Then
         If Cache("eeee") IsNot Nothing Then
          If Cache("ffff") IsNot Nothing Then
           Label1.Text += "retrieving Item From Cache indexiA...<br />"
           Label1.Text += "Items In Cache item A:" & Cache("ffff").ToString &
"<br/>"
          End If
           Label1.Text += "retrieving Item From Cache indexiB...<br />"
           Label1.Text += "Items In Cache item B:" & Cache("eeee").ToString &
"<br/>"
         Else
           Label1.Text += "Cannot retrieving Item From Cache Because cache is
nothing<br />"
           createcache()
         End If
        End If
 End Sub
```

در ابتدا که شرط PostBack سرجایش است و با قرار دادن یک دکمه و کلیک آن تحریک می شود.سپس در اینجا هنگام چاپ محتویات کش در خروجی به این دلیل که دیگر مثل مثال قبلی نبوده و اینگونه نیست که یا هر دو یا هیچ یک,در داخل شرط مربوط به چک کردن وجود آیتم کش eeee وجود آیتم کش ffff را نیز خواستار شدیم.چون با حذف ffff دیگر eeee حذف نمی شود و اگر این شرط دوم را نگذاریم با NullReference Error مواجه می شویم.پس به این ترتیب اگر هر دو آیتم وجود داشته باشند هر دو به نمایش در خواهند آمد و با کلیک دکمه ی button1 آیتم ffff از کش خارج شده و دیگر وارد If دوم نخواهیم شد و تنها محتویات آیتم کش eeee در خروجی چاپ می شوند.

پس در این مثال نحوه ی کنترل آیتم های حذف شده را توسط آرگومان مهم reason فهمیدیم.

موارد گفته شده در مورد وابستگی کش تنها مثال هایی ساده جهت آشنایی و درک وابستگی کش بودند.حال می خواهیم به موارد کاربردی در این زمینه بپردازیم که مهمترین آن وابستگی به پایگاه داده ی SQL SERVER است.یعنی محتویات یک جدول پایگاه داده را در کش ذخیره کنیم و به محض به روز شدن آن آیتم از کش خارج شود.به این صورت دیگر نیازی به تعیین زمان انقضا و همینطور ریسک هایی در آن زمینه نداریم.

:Caching With Sql Server

برای کش کردن به گونه ای که به محض تغییر در جدول مربوطه در پایگاه داده می بایست عمل آگاه سازی را انجام دهیم.البته دراینجا هم داریم نوعی وابستگی ایجاد می کنیم. ابتدا پایگاه داده و سپس جدولی را که می خواهیم کش کنیم را از عمل خود آگاه می کنیم.برای این کار قبلا باید یک سری دستورات را در فایل aspnwt_regsql می نوشتید و دردسر های زیادی هم به دنبال داشت.ASP.NET 2.0 روش جدیدی برای آگاه سازی ایجاد کرده و آن هم استفاده از کلاس SqlCacheDependncyAdmin است.با این کلاس که متد های محدودی هم دارد به سادگی می توان عمل آگاه سازی را انجام داد.متد های این کلاس در زیر آمده اند:

۱-EnableNotifications:برای آگاه سازی پایگاه داده برای کش به کار می رود و آرگومان ورودیش ConnectionString مربوطه است.

۲-DisableNotifications:برای غیر فعال کردن آگاه سازی برای عمل کش بوده و آرگومان ورودیش ConnectionString مربوطه است.

۳-EnableTableForNotifications :برای آگاه سازی یک جدول برای عمل کش است.دو آرگومان ورودی دارد اولی ConnectionString مربوطه و دومی نام جدول است.

٤-DisableTableForNotifications : برای غیر فعال کردن آگاه سازی یک جدول برای عمل کش است.دو آرگومان ورودی دارد اولی ConnectionString مربوطه و دومی نام جدول است.

بیایید مثال زیر را انجام دهیم.

یک صفحه به صورت زیر طراحی کنید:

```
      <asp:Button    ID="button1"    runat="server"    Text="enable    database
notification for cache" Width="243px" /><br />
          <asp:Button  ID="Button4"  runat="server"  Text="disable  database
notification for cache"
              Width="242px" /><br />
           Table             Name:<asp:TextBox             ID="TextBox1"
runat="server"></asp:TextBox>
      <asp:Button ID="button2" runat="server" Text="enable table notification
for cache" /><br />
          Table Name:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
          <asp:Button   ID="Button3"   runat="server"   Text="Disable   Table
Notification For Cache"
              Width="292px" /><br />
          ----------------------------------------------------------------
----------------------------<br />
    <asp:Label ID="label1" runat="server"></asp:Label>
```

و کد پشت صحنه را به صورت زیر بنویسید:

```
      Public         Shared         cs         As         String         =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString

      Protected  Sub  button2_Click(ByVal  sender  As  Object,  ByVal  e  As
System.EventArgs) Handles button2.Click
          SqlCacheDependencyAdmin.EnableTableForNotifications(cs,
TextBox1.Text)
      End Sub

      Protected  Sub  button1_Click(ByVal  sender  As  Object,  ByVal  e  As
System.EventArgs) Handles button1.Click
          SqlCacheDependencyAdmin.EnableNotifications(cs)
      End Sub

      Protected  Sub  Button3_Click(ByVal  sender  As  Object,  ByVal  e  As
System.EventArgs) Handles Button3.Click
          SqlCacheDependencyAdmin.DisableTableForNotifications(cs,
TextBox2.Text)
      End Sub

      Protected  Sub  Button4_Click(ByVal  sender  As  Object,  ByVal  e  As
System.EventArgs) Handles Button4.Click
          SqlCacheDependencyAdmin.DisableNotifications(cs)
    End Sub
```

حال به سادگی می توانید عمل آگاه سازی را انجام دهید.با فشار Button1 آگاه سازی پایگاه داده انجام می شود و یک جدول به نام Aspnet_sqlcacheTablesForChangeNotification به لیست جداول پایگاه داده ای که برایش عمل آگاه سازی را انجام دادید اضافه می شود.حال اگر Button2 را فشار دهید یک سطر به جدول Aspnet_sqlcacheTablesForChangeNotification به نام جدولی که آگاه سازی کردید اضافه شده و همچنین یک ستون مجازی هم به جدولی که آگاه سازی کردید اضافه می شود.جدول Aspnet_sqlcacheTablesForChangeNotification حاوی ستون هایی به شرح زیر است:

۱-TableName:نام جدولی که آگاه سازی کردید.

۲-NotificationCreated:زمانی که آگاه سازی برای جدول فوق انجام شد.از نوع DateTime.

۳-ChangeId:تعداد دفعاتی که جدول مورد نظر Update می شود را مشخص می کند.

با فشردن Button4 جدول Aspnet_sqlcacheTablesForChangeNotification حذف می شود و آگاه سازی غیر فعال می شود و با فشردن Button3 هم جدولی که نامش را در TextBox مربوطه وارد کردید از لیست جداول آگاه سازی شده حذف می شود.

حال که عمل آگاه سازی انجام شد بایدآن را یک جوری به asp.NET انتقال دهیم.برای این کار از تگ SqlCacheDependency در فایل web.Config کمک می گیریم:

```
<caching>
    <sqlCacheDependency enabled="true">
    <databases>
          <add name="sql-test" connectionStringName="aaa" pollTime="500"/>
    </databases>
    </sqlCacheDependency>
</caching>
```

همانطور که می بینید در تگ SqlCacheDependency ابتدا صفت enable را true کردیم سپس از تگ DataBase برای اعمال آگاه سازی استفاده کردیم.با تگ add پایگاه داده ی مربوطه را رجیستر کردیم.صفت name نام پایگاه داده ای که آگاه سازی کش را انجام دادی است.صفت ConnectionStringName هم همان نام رشته ی اتصال

است.صفت PollTime هم باز یک تاریخ انقضا برای کش جهت محکم کاری است که حداقل آن ۵۰۰ است و شما اگر از کدنویسی خود اطمینان دارید می توانید آن را بیشتر هم بدهید.

حالا می خواهیم از شی sqlcachedependency استفاده کنیم.در اینجا این شی را به صورت زیر تعریف کنید:

Dim dep As New SqlCacheDependency("Data Base Name","Table Name")

دو آرگومان ورودی پایگاه داده و جدولی هستند که برای کش آگاه سازی شده اند.

حال که عمل رجیستر هم انجام شد بیایید یک مثال عملی انجام دهیم.

در این مثال ابتدا دریک gridView مقادیر موجود در جدول employees را در هر بار تست کردن کش نشان می دهیم و آن را کش می کنیم.سپس با تغییر آن جدول خواهید دید که کش هم از بین می رود.در اینجا کش از بین رفته با هر بار بارگذاری صفحه دوباره ایجاد می شود. عمل کش را به PostBack صفحه نسبت می دهیم.ابتدا تابعی برای بازیابی مقادیر جدول employees ایجاد کنید:

```
    Private dependency As SqlCacheDependency
    Private connectionString As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString

    Private Function GetTable() As DataTable
        Dim con As SqlConnection = New SqlConnection(connectionString)
        Dim sql As String = "SELECT * FROM employees"
        Dim da As SqlDataAdapter = New SqlDataAdapter(sql, con)
        dependency = New SqlCacheDependency("sql-test", "employees")
        Dim ds As DataSet = New DataSet()
        da.Fill(ds, "employees")
        Return ds.Tables(0)
    End Function
```

به این نکته هم دقت کنید که من وابستگی را در تابع GetTable ساختم.

خروجی این تابع از نوع DataTable است.در ضمن رشته ی اتصال و وابستگی را بیرون از تابع و خصوصی تعریف کردیم تا در جاهای دیگر هم بتوان ازآنها استفاده کرد.

به رویداد Page_Load رفته و کد زیر را در آن بنویسید:

```
        If (Not Me.IsPostBack) Then
            Label1.Text &= "Creating dependent item...<br>"
            Cache.Remove("dependencycache")
            Dim dt As DataTable = GetTable()
```

```
            Label1.Text &= "Adding dependent item<br>"
            Cache.Add("dependencycache", dt, dependency,
DateTime.Now.AddSeconds(6000), TimeSpan.Zero, CacheItemPriority.Normal,
Nothing)
        End If
```

برای محکم کاری در اینجاهم از متد Add به جای Insert استفاده کردم با توجه به اینکه remove هم قبل از آن انجام شد.در اینجا ما جهت محکم کاری(اتفاق های پیش بینی نشده) مدت انقضا هم برای کش در نظر گرفتیم (البته طولانی)با توجه به اینکه هنگامی که جدول به روز شد دیگرتوجهی به این زمان نمی شود.یادتان باشد وقتی از متد Add استفاده می کنید حتما دو مقدار Absoluteو Sliding را مقداردهی کنید و آنها را Nothing نگذارید.نکته ی ذکر شده را حتما انجام دهید چون همیشه اتفاقات غیر قابل پیش بینی وجود دارد که احتمال رخ دادنش وجود دارد.

در این مثال هنگامی که صفحه برای بار اول بارگذاری می شود خروجی متد GetTable را بازیابی می کنیم و سپس آن را همراه با وابستگی که تعریف کردیم(در متد GetTable) به شی کش نسبت می دهیم.

سپس دو Button در صفحه قرار دهید .اولی برای تست کش و دومی برای بروز رسانی پایگاه داده باشد.در رویداد کلیک دکمه ی اول وجود یا عدم وجودکش را بررسی می کنیم.اگر وجود نداشت در Label پیامی مبنی بر عدم وجود کش می دهیم ولی اگر وجود داشت آن را نمایش می دهیم به این صورت که داده ی DataTable را با تابع CType از کش خارج کرده و به عنوان منبع داده ای به GridView نسبت می دهیم:

```
        If Cache("teess") Is Nothing Then
            Label1.Text &= "Cache item no longer exits.<br />"
        Else
            Label1.Text &= "Item is still present.<br />"
            GridView1.DataSource = CType(Cache("teess"), DataTable)
            GridView1.DataBind()
        End If
```

سپس در رویداد کلیک دکمه ی دوم هم عمل بروز رسانی را دستی انجام می دهیم:

```
    Protected Sub button2_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim con As SqlConnection = New SqlConnection(connectionString)
        Dim sql As String = "UPDATE employees SET emp_job='monshi' WHERE
emp_firstname='samad'"
        Dim cmd As SqlCommand = New SqlCommand(sql, con)

        Try
```

```
            con.Open()
            cmd.ExecuteNonQuery()
        Finally
            con.Close()
        End Try
        Label1.Text &= "Update completed <br />"
    End Sub
```

حال اگر صفحه را اجرا کنید برای بار اول کش ساخته شده و مقادیر جدول employees به وسیله ی متد GEtTable به کش منتقل می شود.حال اگر هر چند بار روی دکمه ی Button1 کلیک کنید مقادیر از کش در GridView به نمایش در می آیند چون از این به بعد عمل بازیابی از کش صورت می گیرد نه از تابع GetTable.حال Button2 را کلیک کنید (تا عمل به روز رسانی انجام شود) و سپس Button2 را کلیک کنید میبینید که با پیغام no longer exist مواجه می شوید.این یعنی آیتم از کش خارج شده.ولی با بار گذاری مجدد صفحه کش هم دوباره ایجاد می شود و انگار کش به روز شده.

خوب کار ما با کش هم تمام شد.یادتون باشه کش دیگر جزو مسایل پیشرفته نیست.و جزو مسایل حیاتی وب سایت به شمار می رود و حتما باید در وب سایتتان از آن استفاده کنید .حال بسته به سایت شما از انواع کش می توانید استفاده کنید.حال وارد دنیای Xml خواهیم شد.

XML

Xml یا Extensible MarkUp Language یک زبان قاعده مند و با ساختار مبتنی بر تگ است که یک نوع استاندارد جهانی است و از آن می توان در هر گونه برنامه ای استفاده کرد.مثلا اگر شما یک فایل Xml داشته باشد می توان از آن فایل در هر گونه برنامه نظیر asp-aspx-perl-php-cgi و... هم استفاده کرد.این زبان به شدت گسترش پذیر بوده و دنیای وب بیش از هر چیز به سوی آن پیش می رود.XML هیچ کاری انجام نمی دهد.بلکه تنها محیطی قاعده مند برای داده ها ایجاد می کند و برای استفاده ازآن می توان از نرم افزار های متعددی استفاده کرد.XML قابل ایجاد روی یک Text Editor ساده است وچون نیازی به اجرا ندارد همانجا هم می توان آن را ذخیره کرد.XML صرفا شامل تگ هایی است که خودتان آنها را می سازید و قادرید به هر شکل دلخواه آن را

تغییر دهید.نکته ی مهم در مورد این تگها این است که هر آغازی یک پایان خواهد داشت و آن هم به دو صورت زیر است:

<aaa>....</aaa>

<aaa/>

همیشه در کنار پایگاه های داده ی بزرگی مثل Oracle ,Sql و DB2 یک پایگاه داده ی کم حجم از نوع xml هم وجود داشته است که می تواند داده را در خود ذخیره کند.

در زیر یک نمونه سند xml را مشاهده می کنید:

```
<?xml version="1.0" encoding="utf-8" ?>
<productCatalog>
    <catalogName>My Ebooks Catalog</catalogName>
    <expiryDate>2008-01-01</expiryDate>
    <products>
          <product id="1001">
                <productName>Beginning ASP.NET 2.0</productName>
                <Author>Dino Sositto</Author>
                <productPrice>342.10</productPrice>
          </product>
          <product id="1002">
                <productName>ASP.NET 2.0 Steb By Step</productName>
                <Author>Michael Mc Donald</Author>
                <productPrice>982.99</productPrice>
          </product>
          <product id="1003">
                <productName>Pro ASP.NET 2.0</productName>
                <Author>Dave Sussman</Author>
                <productPrice>982.99</productPrice>
          </product>
          <product id="1004">
                <productName>OOP In ASP.NET 2.0 Using C#</productName>
                <Author>Bill Evjen</Author>
                <productPrice>982.99</productPrice>
          </product>
          <product id="1004">
                <productName>OOP In ASP.NET 2.0 Using VB.NET</productName>
                <Author>Brayan Coopper</Author>
                <productPrice>982.99</productPrice>
          </product>
          <product id="1005">
                <productName>XML in ASP.NET 2.0</productName>
                <Author>Randy Nollan</Author>
                <productPrice>982.99</productPrice>
          </product>
```

```
    </products>
</productCatalog>
```

در این سند من یک تگ ریشه یا root به نام ProductCatalog دارم.هر تگ در xml یک گره یا Node است.همیشه هر فایل xml باید یک عنصر ریشه داشته باشد.هر عنصری از جمله عنصر ریشه می تواند دارای فرزند باشد.در اینجا عنصر ریشه ی ProductCatalog دارای ۳ فرزند به صورت زیر است:

```
<ProductCatalog>
    <CatalogName/>
    <ExpiryDate/>
    <Products/>
</Product Catalog>
```

فرزند اول CatalogName.فرزند دوم ExpiryDate و فرزند سوم Products است.

فرزند های اول و دوم خودشان دارای فرزند نیستند ولی حاوی Value هستند.Value فرزند اول My Ebooks Catalog و Value فرزند دوم 1-1-2008است.میبینید که بین نوع داده ی تاریخ و رشته هیچ فرقی نیست و نیاز به تعریف نوع داده ای در اینجا نیست(البته این یک مشکل است و جلوتر اقدام به رفع آن می کنیم).فرزند Products خودش شامل فرزند هایی با نام مشابه ولی مقادیر متفاوت است.نام هر فرزند آن Product است که خود ۳ فزند Author ProductName و ProductPrice است.نکته ی مهم صفات مشخصه هستند.صفات مشخصه عناصری برای توصیف هر چه بیشتر یک گره هستند.در اینجا صفت id را به دلخواه به کاربردم تا مشخص کننده ی یک شناسه برای محصولم باشد.مقادیری که به هر صفت مشخصه نسبت می دهید باید حتما در داخل "" باشد.پس من از سند xml بالا می توانم به عنوان یک پایگاه داده با ٤ رکورد که نام و سایر مشخصات کتاب را مشخص می کند باشم.یک پایگاه داده از کاتالوگ مخصولات که نامش در گره ی CatalogName آمده.Comment در xml به صورت زیر نوشته می شود:

```
<!-- your comment -->
```

مثل:

```
<!--this is my comment-->
```

فاصله ها در xml حفظ می شوند.مثلا اگر یک Value با مقدار Hellow word! در نظر بگیرید فاصله ی بین hellow و word! در خروجی هم حفظ می شود.

یک نکته در مورد صفات مشخصه هم این است که شما در مقدار دهی به آنها می تواند هم از "" و هم از ’’ استفاده کنید ولی در صورت تودرتوبودن در بیرون باید حتما از ’’ استفاده کنید و یا برعکس مثلا:

<aaa name=”ali is a ‘City’ in Tehran”/>

<aaa name=’ali is a “City” in Tehran’/>

xml فواید بسیاری دارد ولی در برنامه های مدرن و امروزی ۳ فایده ی بسیار مهم دارد:

۱-مقبولیت عام:xml در حال حاضر در بسیاری از کمپانی ها مورد استفاده قرار می گیرد و در مبادله و به اشتراک گذاری داده ها بسیار مفید بوده و همین امر باعث شده خیلی ها به سوی xml پیش روند.

۲-قابلیت انعطاف و گسترش:XML هیچ گونه قواعد و یا موارد معنا شناسی را برای استفاده کننده به همراه نداشته و بسیار قابل توسعه است(برخلاف EDI یا مبادله ی الکترونیک داده ها که کمی از پست الکترونیک فراتر می رود ولی یک سری قوانین و محدودیت به همراه دارد) و استفاده از آن بسیار ساده و همچنین کنترل آن نیز ساده است.

۳-استاندارد بودن خود Xml و ابزارهای وابسته اش:هم اکنون xml یک استاندارد جهانی است و کار با آن هم مستلزم نرم افزار خاصی نیست و با اینکه خودش استاندارد هست , یک سری ابزار استاندارد نیز مثل Xml Scema XPath و... را نیز جهت استفاده ایجاد کرده.این استاندارد سازی باعث شده بسیاری از شرکت ها و یا کمپانی های برنامه نویسی می توانند در مورد xml با هم تبادل داده داشته باشند.

مشخصات یک سند Xml خوش ساخت(WellForm):

-هر تگ شروع مستلزم تگ پایانی است.

-عناصر نباید همپوشانی داشته باشند مثلا <aa><bb></bb></aa> درست است ولی <aa><bb></aa></bb> نادرست است.

-یک سند طئم تنها یک ریشه باید داشته باشد.

-تمام صفت ها باید یا در داخل "" و یا ‘’ باشند.
-یک تگ(گره یا عنصر) نباید دارای دو صفت با یک نام باشد.
-دستورات و دستورالعمل های پردازشی نباید در کنار تگ ها بیایند.

NameSpace در Xml:

همانطور که گفتیم بسیاری از شرکت ها می توانند با xml عمل مبادله ی داده ها را انجام دهند ولی به چه صورت؟این عمل با تکیب سند های xml آن ها صورت میگیرد.مثلا ترکیب سند xml یک شرکت تجارب با یک شرکت اقتصادی.دراین گونه موارد منظور از ترکیب یعنی به کار گیری عناصر یک سند در سند دیگر.این امر مشکلاتی در پی دارد مثل همنام بودن برخی عناصر و یا نحوه ی برقراری ارتباط عناصر با یکدیگر.برای مقابله با این مشکل از nameSpace استفاده می شود.مثلا فضای نام سند xmlیک شرکت را در سند دیگری قراردهیم پس به این ترتیب به تمام عناصر آن در دیگری بدون مشکل همنام بودن و نحوه ی ارتباط دسترسی داریم.فضا های نام xml همگی از نوع Url(Universal Identifires (Resource هستند.مثلا http://www.mycompany.com/mystandard می تواند یک فضای نام باشد.برای قرار دادن این فضاهای نام در سند xml از ویژگی xmlns استفاده می کنیم که مخففXml NameSpace است.این ویژگی را در هر تگی به کار ببرید می توانید در نزد فرزندان آن تگ از سند خارجی استفاده کنید.مثلا:

```
<aas xmlns:ord="http://mycompany/OrderML"></aas>
```

در کد بالا من در داخل تک aas یک فضای نام تعریف کردم و چنانچه aas دارای فرزندی باشد تمام آن فرزند ها می توانند از عناصری که از فضای نام مربوطه استخراج می شود استفاده کنند.مثلا اگر سند مربوط به فضای نام مربوطه دارای یک عنصر به نام OrderItem باشد می توان به صورت زیر از آن استفاده کرد:

```
  <aas xmlns:ord="http://mycompany/OrderML">
      <orderItem>...</orderItem>
      <orderItem>...</orderItem>
</aas>
```

شما می توانید فضاهای نام را به صورت بخش های جدا از هم و به چندین تعداد تعریف کنید.برای اینکه بتوانید در یک مکان چندین بار از صفت xmlns استفاده کنید باید برای هر یک یک نام ایجاد کنید:

```
  <order                                    xmlns:ord="http://mycompany/OrderML"
xmlns:cli="http://mycompany/ClientML">

</order>
```

در اینجا من دو فضای نام ایجاد کردم یکی با نام ord و دیگری با نام cli.حال برای استفاده از هر یک باید تگ ها را به صورت پیشوندی بنویسم.یعنی :

<NanemSpace name:your tag neme

مثلا:

```
  <?xml version="1.0" encoding="utf-8" ?>
  <order                                    xmlns:ord="http://mycompany/OrderML"
xmlns:cli="http://mycompany/ClientML">
      <cli:client>
            <cli:firstName>...</cli:firstName>
            <cli:lastName>...</cli:lastName>
      </cli:client>
      <ord:orderItem>...</ord:orderItem>
      <ord:orderItem>...</ord:orderItem>
</order>
```

همانطور که می بینید نام xmlns و نام دلخواه ما با : از هم جدا شده اند.

من از فضای نام دوم که نامش cli بود استفاده کردم و یک گره ی والد به نام client (که می بایست در سند فضای نام اصلی موجود باشد) با دو فرزند به نام های firstnameو lastname ایجاد کردم..و همینطور برای فضای نام اولی.همچنین کد بالا را می توانید به صورت زیر هم بنویسید:

```
  <ord:order                                xmlns:ord="http://mycompany/OrderML"
xmlns:cli="http://mycompany/ClientML">
      <cli:client>
            <cli:firstName>...</cli:firstName>
            <cli:lastName>...</cli:lastName>
      </cli:client>
      <ord:orderItem>...</ord:orderItem>
      <ord:orderItem>...</ord:orderItem>
</ord:order>
```

و یا:

```
  <cli:order                                xmlns:ord="http://mycompany/OrderML"
xmlns:cli="http://mycompany/ClientML">
      <cli:client>
            <cli:firstName>...</cli:firstName>
            <cli:lastName>...</cli:lastName>
      </cli:client>
      <ord:orderItem>...</ord:orderItem>
      <ord:orderItem>...</ord:orderItem>
```

```
</cli:order>
```

واضح است که جهت خوانایی بیشتر عنصر ریشه هم می تواند حاوی نام یکی از فضاهای نام باشد.این روش یعنی به کاربردن عبارات پیش وندی توصیه می گردد چون برای خوانایی بیشتر سند بسیار مهم است و ما از این به بعد همین کار را می کنیم.

حالا نوبت به کار با xml در asp.net است.ولی قبل از آن می خواهیم با یکی از پرکاربرد ترین موارد وابسته به xml آشنا شویم به نام XSD.

XSD:

Xml Schema Definition یک زبان از جنس Xml است که باعث قانونمند شدن xml می شود.مثلا محدود کردن xml به DataType ها و یا محدودیت برای ویژگی ها یا صفت ها و....ما تا به حال گفتیم Xml یک زبان قاعده مند و با ساختار است ولی قانونمند نیست .با این حال با Xsd می توان آن را قانونمند هم ساخت.مثلا فایل Web.Config را در نظر بگیرید.این فایل یک فایل xml جامع و قانونمند است که با زبان Xsd نوشته شده و به شما اجازه ی ویرایش به صورت xml را می دهد.مثلا شما وقتی از تگ ConnectionString استفاده می کنید بین آغاز و پایان این تگ تنها حق استفاده از ۳ تگ فرزند را دارید Add– Remove-Clear و نمی توانید تگ دلخواهی را در آن قرار دهید و یاوقتی که از تگ Add استفاده می کنید یک سری ویژگی هایی مثل Name ConnectionString و ProviderName و...برایتان امکان مقداردهی دارد و بیش از این شما نمی توانید ویژگی جدیدی را تعریف کنید.در حالیکه در فایل های xml شما هر تگی و با هر ویژگی را می توانید تعریف کنید ولی برنامه نویسان Asp.NET با زبان XSd شما را در یک فایل Xmlمحدود کرده اند با یک سری تگ خاص و ویژگی های خاص.پس هر فایل xml که یک Schema رویش قرار گیرد و قانونمند شود می تواند در برنامه های کاربردی مورد استفاده قرار گیرد.حال این قانون ها را با XSD ایجاد می کنیم و به فایل xml معنی می بخشیم تا قابل استفاده باشد.XSD در سال ۲۰۰۱ توسط برنامه نویسان ایجاد شد و مورد تایید کنسرسیوم جهانی وب قرار گرفت به همین دلیل در هر فایل XSD باید فضای نام زیر وجود داشته باشد:

```
http://www.w3.org/2001/xmlSchema
```

جالب اینجاست که XSD خود از جنس xml است ولی با گرامرهایی علاوه بر xml.

شما در XSD می توانید DataType تعریف کنید مثل Integer و یا String.

شما می توانید یک صفت را به عنوان صفت کلید (به عنوان مقداری Unique) تعریف کنید.

می توانید مقدار Null در آن داشته باشید.

در آن می توانید از NameSpace استفاده کنید و با برنامه های دیگر ارتباط برقرار کنید.

برای یک مثال ساده از XSD سند XML ساده ی زیر را در نظر بگیرید:

```
<?xml version="1.0" encoding="utf-8" ?>
<Students>
    <Student>
          <ID>12345</ID>
          <GPA>3.5</GPA>
    </Student>
    <Student>
          <ID>67890</ID>
          <GPA>4.0</GPA>
    </Student>
    <Student>
          <ID>67450</ID>
          <GPA>1.75</GPA>
    </Student>
    <Student>
          <ID>62110</ID>
          <GPA>6.0</GPA>
    </Student>
</Students>
```

یک سند ساده که یک گره ی ریشه به نام Students دارد که ٤ فرزند مشابه به نام Student دارد.هر فرزند نیز حاوی دو فرزند Id و Gpa هستند که به ترتیب مشخص کننده ی شناسه و میانگین امتیاز هر دانشجو است.حال می خواهیم یک XSD برای فایل بالا ایجاد کنیم.به این منظور از Add New Item گزینه ی Xml Schema را انتخا می کنیم تا یک فایل XSD برایم باز شود.سپس فضای نام مربوطه را به همراه یک نام مثل xsd برایش در نظر می گیرم:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

</xsd:schema>
```

همه ی قواعدی که تعریف خواهید کرد باید در داخل تگ xsd:Schema باشد.هر عنصر با تگ element تعریف می شود.مثلا ما در فایل xml یک عنصر داریم به نام

Students که عنصر ریشه نیز هست.برای تعریف آن از تگ به element به صورت زیر استفاده می کنیم:

```
<xsd:element name="students">

</xsd:element>
```

خصلت name نام تگ را مشخص می کند.تگ element حاوی صفات دیگری به جز name نیز هست که در زیر آمده:

۱-Type:نوع داده ای عنصر مربوطه را مشخص می کند(منظورمقداریست که بین دو تگ می آید مثلا <aaa>my name is ali</aaa> که مقدارش my name is ali بود و شما می توانیدDataTypeآن را Date و یا integer و... قرار دهید).

۲-MinOccurs:دو مقدار می گیرد ۰ یا هر عددی بزرگتر ازصفر. ۰ به این معنی است که عنصر مورد نظر به هر تعدادی می تواند استفاده شود.و هر عدد بزگتر از ۰ هم یعنی عنصر مورد نظر تنها به همان تعداد می تواند به کار رود(نه کمتر نه بیشتر).مقدار پیش فرض آن هم ۱ است.مثلا اگر خواستید تگ خاصی مثلا ٤ بار تکرار شود این مقدار را ٤ در نظر بگیرید.در اینجا ما آن را مقداردهی نکردیم چون عنصر ریشه است و هر عنصر ریشه تنها یک بار باید به کار رود.

۳-MaxOccurs:تعداد مجاز عنصر را در سند مشخص می کند.مقدار پیش فرض آن ۱ است.این عنصر سقف تعداد مجاز یک عنصر را مشخص می کند.دو مقدار می گیرد اولی هر عدد بزرگتر از ۰ که به این معنی است که تگ مورد نظر می تواند تا اندازه ی آن عدد تکرار شود.و دومیUnBound ed است که می گوید عنصر مورد نظر به هر تعدادی می تواند تکرار شود.در اینجا ما آن را مقدار دهی نکردیم چون عنصر ریشه است.

تگ بعدی تگ complexType است.این تگ برای عناصری به کار برده می شود که یا فرزند دارند و یا صفت دارند.در این مثال گره ی ریشه خود دارای فرزند های مشابهی به نام student است پس باید در آن از تگ complexType استفاده شود:

```
<xsd:element name="students">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="student" minOccurs="0"
maxOccurs="unbounded" >
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

با تگ squence هم جلوتر آشنا می شویم که حتما باید پس از complexType ذکر شود.

خود student نیز دارای دو فرزند GpaوId است.پس باید حتما در داخل ComlexType تعریف شود.تگ بعدی Squence است.این تگ همیشه پس از ComplexType می آید و نشان دهنده ی توالی در اجزا است.مثلا گره ی Student دو فرزند دارد.اولی Id و دومی GPA.ولی ما مشخص نکردیم آیا ترتیبی برای ایندو وجود دارد یا نه.ولی با Squence این ترتیب را مشخص می کنیم:

```
<xsd:element name="student" minOccurs="0" maxOccurs="unbounded" >
  <xsd:complexType>
    <xsd:sequence>
          <xsd:element     name="id"     type="xsd:string"     minOccurs="1"
maxOccurs="1"/>
          <xsd:element    name="gpa"    type="xsd:integer"    minOccurs="1"
maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

به جایگاه هر تگ توجه کنید.هر تگی که دارای اجزا(فرزند) باشد چیدمان تگها برایش به صورت زیر است:

Element

ComplexType

Squence

child elements....

یعنی اول از همه نام آن عنصر.سپس اگر دارای صفت و یا فرزند باشد از ComplexType در مرحله ی بعد استفاده می کنیم سپس اگر خواهان ترتیب باشیم از sequence استفاده می کنیم و در نهایت با element نام فرزند ها را ذکر می کنیم.حال ممکن است این فرزندان خود نیز دارای فرزند باشند در این صورت آنها هم باید دارای ComplexType و sequence باشند.پس فایل XSd برای سند بالا به صورت زیر شد:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="students">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="student" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
```

```
            <xsd:sequence>
            <xsd:element     name="id"     type="xsd:string"     minOccurs="1"
maxOccurs="1"/>
            <xsd:element     name="gpa"     type="xsd:integer"     minOccurs="1"
maxOccurs="1"/>
              </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

وقتی MinOccurs و MaxOccurs هر دو برابر ۱ باشند یعنی عنصر مربوطه تنها یک بار باید ایجاد شود(این شرط تنها در داخل تگ والدش صحت دارد) نه بیشتر و نه کمتر.

حال می رسیم به نحوه ی تعریف صفت.صفت ها با عنصر Attribute تعریف می شوند و ٤ ویژگی هم دارند:

۱-Name:نام صفت است و الزامی است.

۲-Type:نوع داده ای صفت را مشخص می کند.مثلا خیلی از صفات مقدارشان Boolean است (مثل صفت LockItem در کد زیر) و یا String است مثل ConnectionString درکد زیر:

```
<add name="aaa" connectionString="my connection string"  lockItem="true"
/>
```

۳-Use:نحوه ی استفاده از ویژگی را مشخص می کند و مقادیرش در جدول زیر ذکر شده اند:

مقدار نسبت داده شده	توضیحات
Required	خصلت اجباری است
Default	خصلت دارای مقدار پیش فرض است
Fixed	خصلت دارای مقدار ثابت و غیر قابل تغییر است.
Optional	خصلت اختیاری است

خصلت اجباریست یعنی آن صفت حتما باید در سند xml تعریف و مقدار دهی شود و در غیر این صورت با خطا مواجه می شویم.مثلا به کد زیر درWeb.Config نگاه کنید:

```
  <connectionStrings>
      <add name="aaa" />
</connectionStrings>
```

در تگ add از این کد صفتی به نام ConnectionString جا انداخته شده بنابراین برنامه با خطا مواجه می شود.چون در تعریف این عنصر در قسمت Attribute مقدار Required برایش در نظر گرفته شده بود.

خصلت دارای مقدار پیش فرض است یعنی در صورتی که کاربر آن را مقدار دهی نکند خودمان یک مقدار برایش در نظر می گیریم تا با خطا مواجه نشود.مثل صفت LockItem در کد بالا(در داخل تگ Add) که ما مقداردهی هم نکنیم با خطا مواجه نمی شویم چون در XSD وقسمت Attribute با Default مقداردهی شده.حال این مقدار پیش فرض در Value مشخص می شود.

Fixed یعنی خصلت مقدارش یکی است و شما حق انتخابی ندارید. این مقدار پیش فرض در Value مشخص می شود.

۳-Value:چنانچه مقدار Use یکی از دو نوع Defaultو یا Fixed باشد. Value نشان دهنده ی مقدار پیش فرض است ولی در غیر این صورت نیازی به مقدار دهی Value نیست.

حال که با Attribute آشنا شدید می خواهیم محل استفاده اش را مشخص کنیم. Attribute همیشه در انتهای ComplexType مشخص می شود.زیرا همانطور که قبلا گفته بودیم ComplexType وقتی برای یک عنصر به کار می رود که یا دارای فرزند باشد یا دارای صفت.مثلا فرض کنید فایل xml ما به صورت زیر باشد.یعنی id به جای اینکه یک عنصر باشد یک صفت است:

```
<?xml version="1.0" encoding="utf-8" ?>
<Students>
    <Student Id="12345">
          <GPA>3.5</GPA>
    </Student>
    <Student Id="67890">
          <GPA>4.0</GPA>
    </Student>
    <Student Id="67450">
          <GPA>1.75</GPA>
    </Student>
    <Student Id="62110">
          <GPA>6.0</GPA>
    </Student>
</Students>
```

آنگاه Xsd آن به فرم زیر می شود:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="students">
    <xsd:complexType>
       <xsd:sequence>
         <xsd:element name="student" minOccurs="0" maxOccurs="unbounded" >
          <xsd:complexType>
           <xsd:sequence>
              <xsd:element   name="gpa"   type="xsd:integer"   minOccurs="1"
maxOccurs="1"/>
           </xsd:sequence>
              <xsd:attribute            name="id"            type="xsd:string"
use="required"></xsd:attribute>
         </xsd:complexType>
        </xsd:element>
       </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

ابتدا عنصر ریشه را مشخص کردیم به نام Student و چون دارای فرزند است از complexType در آن استفاده کردیم.سپس عنصر student را مشخص کردیم.و چون این صفت یا فرزند دارد و یا صفت و یا هر دو ملزم به استفاده از ComplexType در زیر آن هستیم. با یک تگ element عنصر GPa را با تعداد تکرار یک بار در داخل تگ Student مشخص می کنیم.در نهایت هم ویژگی id را با تگ Attribute ایجاد می کنیم.نوع داده ای را رشته ای و صفت use را برابر Reqiured در نظر می گیریم .پس نیازی به صفت Value نیست.

دو نوع تشریح در xsd وجود دارد.تشریح نوع اول یا Nested (تو درتو)به همان صورتی بود که در مثال های بالا دیدید.به این صورت که هر عنصری که دارای فرزند باشد پس از تعریف element برای آن می بایست از تگ ComplexType استفاده شود.این کار برای اسناد پیچیده و حجیم(دارای تگ های فرزند تودرتو) ممکن است باعث پیچیدگی زیاد شود.به همین دلیل یک نوع تشریح دیگر وجود دارد به نام Global.در این تشریح پس از تعریف elementی که حاوی فرزند و یا Attribute است مستقیما اقدام به ایجاد تگ ComplexType نمی کنیم و آن را برای پرهیز از شلوغی جدا از همه ی

تگها ولی در داخل تگ Schema تعریف می کنیم.کد زیر به روش nested تشریخ شده و می خواهیم آن را به روش Global تشریح کنیم:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="students">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="student" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
          <xsd:sequence>
          <xsd:element     name="id"     type="xsd:string"     minOccurs="1"
maxOccurs="1"/>
          <xsd:element    name="gpa"    type="xsd:integer"    minOccurs="1"
maxOccurs="1"/>
            </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

برای این کار از خصوصیت name تگ ComplexType استفاده می کنیم:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="students">
          <xsd:complexType>
                <xsd:sequence>
                      <xsd:element       name="student"       type="student"
minOccurs="0" maxOccurs="unbounded"></xsd:element>
                </xsd:sequence>
          </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="student">
          <xsd:sequence>
                <xsd:element   name="ID"   type="xsd:string"   minOccurs="1"
maxOccurs="1"/>
                <xsd:element   name="GPA"   type="xsd:string"   minOccurs="1"
maxOccurs="1"/>
          </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

می بینید که ازتگ ComplexType پس از Student -element استفاده نکردیم و پس از انتهای تگ element عنصر ریشه آن را تعریف کردیم.برای تعریف هم چون به

مشکل برنخوریم و معلوم شود که تگ ComplexType مربوط به کدام element است از صفت name استفاده کردیم و مقدار این صفت را برابر مقداری که برای Typeعنصر مربوطه قرار داده بودیم در نظر می گیریم.پس نکته ی بسیار مهم در اینجا نوع داده ای برای عنصر student است.ما آن را به جای string از نوع خود student در نظر گرفتیم این مقدار همان مقداری است که در صفت name تگ ComplexType که جداگانه ذکرشده می باشد.پس در کل هر عنصری که تگ complextype برایش جدا تعریف می شود باید نوع داده ایش همنام باشد با صفت name تگ complextype.

می توان این دو روش را با یکدیگر ترکیب کرد.یعنی برای برخی از عناصر به روش nested تشریح را انجام دادو برای برخی دیگر به روش global.

برای درک بیشتر فایل xml اول بخش را که مربوط به محصولات تولیدی بود را به نحوی ترکیبی از nested و global به صورت xsd در می آوریم:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="productCatalog">
   <xsd:complexType>
     <xsd:sequence maxOccurs="unbounded">
    <xsd:element name="catalogName" type="xsd:string"></xsd:element>
    <xsd:element name="expiryDate" type="xsd:date"></xsd:element>
    <xsd:element name="products">
     <xsd:complexType>
      <xsd:sequence>
        <xsd:element              name="product"              type="product"
maxOccurs="unbounded"></xsd:element>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
<xsd:complexType name="product">
    <xsd:sequence>
     <xsd:element name="productName" type="xsd:string"></xsd:element>
     <xsd:element name="Author" type="xsd:string"></xsd:element>
     <xsd:element name="productPrice" type="xsd:integer"></xsd:element>
    </xsd:sequence>
    <xsd:attribute              name="id"              type="xsd:integer"
use="required"></xsd:attribute>
  </xsd:complexType>
</xsd:schema>
```

در ابتدا عنصر ProductCatalog را به عنوان یک element ریشه در نظر گرفتیم.سپس چون دارای ۳ فرزند است ابتدا یک تگ complexType و سپس Sequence تعریف کردیم.بعد از آن به ترتیب ۳ فرزند را در تگ element ذکر کردیم و فرزند آخر(Products) چون خودش دارای فرزند است پس برایش complexType و سپس Sequence تعریف کردیم ولی خود فرزند Products که Product است چون حاوی فرزند است می بایست برای آن هم complexType و سپس Sequence تعریف کنیم ولی برای جلوگیری از پیچیدگی زیاد آن را جدا تعریف کردیم.قبل از تعریف جداگانه ی آن صفت type آن را با product مقدار دهی کردم چون هر چیز غیر از آن باعث ایجاد خطا می شود.به این نکته دقت کنید با اینکه عنصر product حاوی Attribute هم هست باز هم آن را می توان جدا تعریف کرد و Attribute مورد نظر را در انتهای آن ذکر کرد.این تلفیقی بود از تشریح به روش nested و global.

خوشبختانه DataType xsd های بسیاری را پشتیبانی می کند (هر آنچه به ذهنتان برسد) ولی خودتان هم می توانید DataType های دلخواه هم تولید کنید.برای این کار از <xsd:simpleType> استفاده می کنیم.نوع های ساده یا همان simpleType بسیار پر کاربرد هستند.خیلی مواقع اینکه یک مقدار صرفا string باشد کافی نیست و می بایست محدودیت های بیشتری برای آن مقدار قایل شد.مثلا حتما بعضی ویژگی ها را در web.config دیدید که تا آنها را انتخاب می کنید یک منو با چند گزینه ظاهر می شود و شما را ملزم می کند تنها یکی از آنها را انتخاب کنید و شما اگر چیزی غیر از آن وارد کنید با خطا مواجه می شوید.برای مثال در کد زیر ما یک نوع داده ای تعریف می کنیم که شک مقدار عددی را بین ۱۰۰۰ و ۳۰۰۰ محدود می کند.

ابتدا از تگ simpleType برای تعریف نوع داده ای استفاده می کنیم و نامی به آن می دهیم:

```
<xsd:simpleType name="simpleType1">

</xsd:simpleType>
```

سپس برای اعمال جزییات از تگ restriction در آن استفاده می کنیم و با ویژگی base نوع داده ای اصلی را تعیین می کنیم که در اینجا من آن را integer در نظر گرفتم:

```
<xsd:simpleType name="simpleType1">
    <xsd:restriction base="xsd:integer">

    </xsd:restriction>
```

```
</xsd:simpleType>
```

سپس برای اعمال محدودیت عددی از یک مقدار مینیمم و ماکسیمم از تگ های minInclusive و maxInclusive استفاده می کنیم:

```
  <xsd:simpleType name="simpleType1">
      <xsd:restriction base="xsd:integer">
            <xsd:minInclusive  value="1000" />
            <xsd:maxInclusive value="3000" />
      </xsd:restriction>
</xsd:simpleType>
```

حالا simpleType1 یک نوع داده ای جدید است که در هر جایی می توانم از آن استفاده کنم مثلا اگر به جای xsd:integer و یا xsd:string از simpleType1 استفاده کنم شما محدود به وارد کردن عدد هستید آن هم عددی بین ۱۰۰۰ تا ۳۰۰۰.مثلا برای مقدار تگ price سند dvd ها آن را اعمال می کنم:

```
<xsd:element name="Price" type="simpleType1" />
```

اگر بخواهید لیستی از انتخابها را ایجاد کنید از تگ <xsd:enumeration> در داخل تگ restriction استفاده می کنید و با استفاده از تگ value مقداری برایش تعیین می کنید مثلا ما می خواهیم همینکه کاربر ویژگی category را نوشت و علامت = را تایپ کرد لیستی برایش باز شود که از آن لیست DVD category مورد نظر را انتخاب کند:

```
  <xsd:simpleType name="simpleType2">
      <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Science Fiction" />
            <xsd:enumeration value="Drama" />
            <xsd:enumeration value="Action" />
            <xsd:enumeration value="Western" />
      </xsd:restriction>
</xsd:simpleType>
```

همچنین می توانید برای یک نوع خاص الگو تعریف کنید این کار را می توان با تگ <xsd:pattern> انجام داد.مثلا \d(3) به معنی ۳ رقم است و یا english|persian به شما می گوید حق دارید یکی از این دو را فقط استفاده کنید.این الگوها را باید در ویژگی Value تگ <xsd:pattern> قرار دهید.

می توانید طول رشته ی وارده را با تگ <xsd:lenght> محدود کنید مثلا:

<xsd:lenght value=”5”/>

طول رشته ی ورودی حتما باید ۵ باشد.

<xsd:minlenght value=”5”/>

طول رشته حداقل ۵ باشد.

<xsd:maxlenght value=”5”/>

طول رشته حد اکثر ۵ باشد.

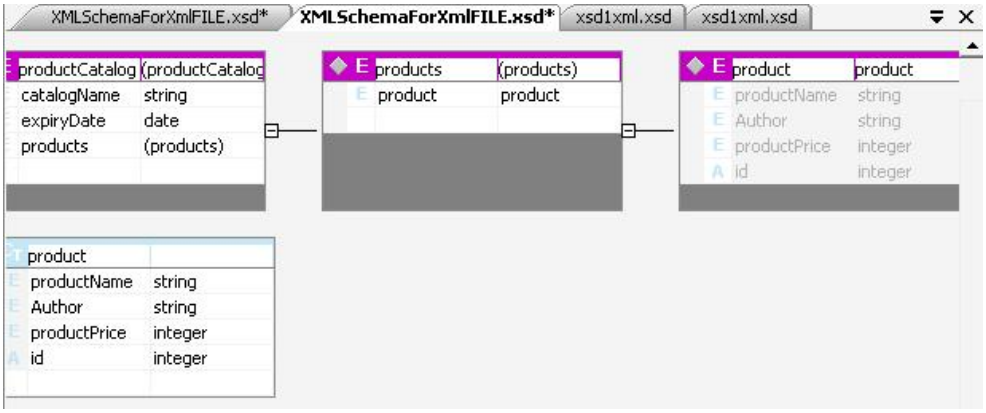
اگر داده ی شما از نوع decimal(اعشاری) باشد می توانید تعداد ارقام صحیح و اعشار آن را محدود کنید.برای مشخص کردن حداکثر تعداد ارقام اعشاری از تگ <xsd:precision> استفاده می کنیم و برای تعیین تعداد ارقام اعشار(سمت راست) از تگ <xsd:scale> استفاده می کنیم:

<xsd:precision value=”5”/>

<xsd:scale value=”2”/>

به این ترتیب تنها اعداد اعشاری که حد اکثر طول 5 دارند و تعداد ارقام اعشارشان حداکثر ۲ تاست مورد قبولند.

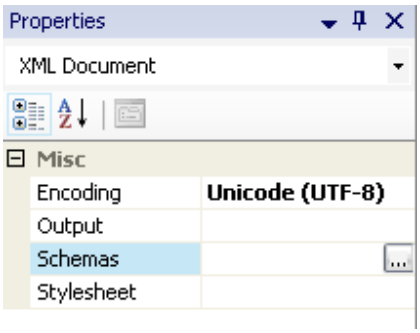
اگر در Visual Studio یک سند xsd ایجاد کنید Visual Studio یک نمای visual هم به شما ارایه می کند تا بتوانید به طور ویژوال هم به طراحی xsd بپردازید.به این منظور هر یک از تگ ها یی که تا اینجا به کار بردیم به شکل کنترل در آمده و می توان با Drag & Drop با آنها کار کرد این کنترل ها در قسمت Tools موجود هستند.نتایج هم به صورت جدول هایی به نمایش در خواهند آمد و هر جدول یک ریشه خواهد بود که فرزندان آن صفات مشخصه هستند.این جداول به رنگ صورتی خواهند بود و اگر تگ complexType را جداگانه تعریف کنید جدولش به رنگ آبی در می آید.نمونه ی خروجی ویژوال مثال اخیر را در زیر مشاهده می کنید:



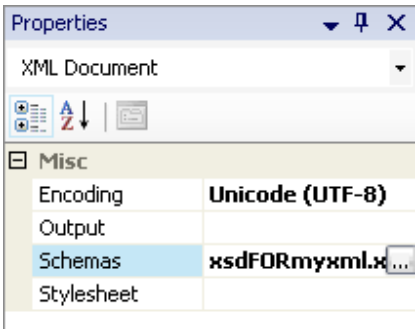
حال که با xsd آشنا شدید می خواهیم آن را روی یک سند xml اعمال کنیم.ابتدا یک فایل xml جدید ایجاد کنید که در صورت ایجاد کد زیر در آن وجود دارد:

```
<?xml version="1.0" encoding="utf-8" ?>
```

حال اگر روی صفحه ی xml باز شده کلیک کنید در سمت راست Visual Studio زیر Solution Explorer یک پنجره ایجاد می شود به نام Properties که مخصوص XML Document است به شکل زیر:



در این پنجره گزینه ای وجود دارد به نام Schema که باید آدرس فایل xsd مربوطه را در آن بنویسید .چون فایل XML و XSD و سایت من همگی در یک فولدر قرار داشتند بنابراین من به ذکر نام فایل بسنده می کنم و تنها نام فایل XSD را وارد می کنم.البته می توانید از دکمه ی Browse که دکمه ای با نوشته ی ... است هم برای یافتن فایل مورد نظرتان اقدام کنید:



بلافاصله پس از انتخاب فایل Schemaی مورد نظر روی صفحه اعمال می شود و شما تنها می توانید بر اساس تعریفی که در فایل xsd مربوطه داشته اید سند xml را بنویسید.

ولی اگر Visual Studio ندارید می توانید کد زیر را به تگ ریشه ی خود اضافه کنید:

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="path Of Your Xsd File"
```

فضای نام اول به تجزیه کننده ی سند xml می گوید این سند قرار است به واسطه ی نامی که برای فضای نام در نظر گرفتیم(در اینجا xsi) Validate شود.ویژگی noNameSpaceSchemaLocation هم محل فیزیکی فایل xsd را مشخص می کند.

xsd بسیار گسترده است و برای آگاهی کامل باید به سایت هایی مثل http://www.w3schools.com/schema رجوع کنید.
کمتر در زمینه ی xsd کتاب مجزا به چاپ رسیده زیرا آنقدر گسترده نیست که به مرحله ی جداگانه ی کتاب برسد ولی در اکثر کتابهای در زمینه ی xml فصل یا فصولی به xsd اختصاص داده شده.

حال می رسیم به کار با xml از طریق Asp.NET 2.0.و در ابتدا می خواهیم یک فایل xml را از داخل asp.net بسازیم.

ایجاد و خواندن فایل XML با برنامه نویسی:
در این قسمت می خواهیم با توابع و متدهایی که کلاس های فضای نام System.XML در اختیار ما قرار داده اند آشنا شده و کار کنیم.در ابتدا می خواهیم با برنامه نویسی یک فایل xml ایجاد کنیم.برای این کار از کلاس XmlTextWriter استفاده می کنیم.این کلاس دارای متد ها و خواص زیادی است.
برای ایجاد یک فایل xml قبل از هر چیز باید مسیری برای آن بیابیم.بدین منظور از Server.MapPath استفاده می کنیم.
تابع سازنده ی کلاس XmlTextWriter دارای ۲ آرگومان ورودی است.اولی آدرس و نام فایلی که باید ساخته شود از نوع String و دومی Encoding فایل مربوطه است.همانطور که می دانید اگر یک فایل xml ایجاد کنید یک صفحه با text زیر برایتان به نمایش در می آید:

```
<?xml version="1.0" encoding="utf-8" ?>
```

صفت encoding نحوه ی کد شدن صفحه را مشخص می کند.پس با دومین آرگومان ورودی تابع سازنده کلاس XmlTextWriter شما در اصل صفت encoding را به صفحه اضافه می کنید.حال می توانید با مقدار Nothing برایش مقداری انتخاب نکنید.

کلاس XmlTextWriter دارای متد هایی ااست که اهم آنها در زیر آمده است:

Formatting:نحوه ی چیدمان تگ ها را مشخص می کند و دو مقدار زیر رامی پذیرد:

۱-Formatting.Indented:ایجاد فاصله(دندانه یا تورفتگی) بین تگ ها در فایل .xml

۲-Formatting.None :نبودن فاصله (دندانه یا تورفتگی) بین تگ ها.

Indention:اگر صفت Formatting با Formatting.Indented مقدار دهی شود آنگاه صفت Indention مقداری عددی به معنای تعداد فضای خالی بین تگ ها را مشخص می کند که پیش فرض آن ۲ است.

WriteStartDocument:شروع نوشتن سند xml را با اعلان <?xml version="1.0" مشخص می کند.یعنی در حقیقت تگ اولیه ی سند xml را که معرف Version زبان xml است را ایجاد می کند.

WriteStartElement:با این متد می توانید تگ ایجاد کنید.آرگومان ورودی این تابع نام تگ مربوطه است.ولی تگی که با این متد ایجاد می کنید فاقد Value است یعنی بین باز و بسته شدن تگ چیزی نوشته نشده(ولی متد هایی هست که بتوان با آن در این گونه تگ ها که فرزند دارند هم چیزی نوشت).به عبارت دیگر این متد برای تعریف و ایجاد تگ هایی است که دارای فرزند هستند.

WriteEndElement:به معنای بستن یک تگ است.ولی آرگومان ورودی ندارد.بنابراین پس از هر WriteStartElement با هرآرگومان ورودی باید یک WriteEndElement ذکر شود.

WriteElementString: با این متد می توانید تگ ایجاد کنید.این متد دو آرگومان ورودی دارد اولی نام تگ و دومی Value(نوشته ای که بین باز و بسته شدن تگ قرار می گیرد) است.این متد برای تعریف و ایجاد تگ هایی است که فاقد فرزند هستند.

WriteAttributeString:برای تعریف خاصیت از آن استفاده می شود.دو آرگومان ورودی دارد اولی نام ویژگی و دومی مقدار آن است.

Close: برای بستن XmlTextWriter به کار می رود که در انتهای کار از آن استفاده می شود.این متد همانقدر ضروری است که برای SqlDataReader ضرورت داشت.

WriteComment:برای نوشتن توضیح به کار می رود .متن توضیح آرگومان ورودی این متد است.در هر جایی که دلتان خواست می توانید از این متد استفاده کنید.

WriteName & WriteValue:از متد هایی که کمتر مورد استفاده قرار می گیرند و برای نوشتن Value برای تگ های فرزند دار و غیر فرزند دار استفاده می شود.

WriteEndDocument:مشخص کننده ی انتهای برنامه نویسی برای سند xml است.مهم نیست این متد را قرار دهید یا نه ولی برای خوش فرم بودن کدهایتان لازم است.به همین ترتیب برای WriteAttributeString هم متد هایی برای نشان دادن انتهایشان وجود دارد مثل WriteEndAttributeو....

برای ایجاد یک سند xml باید به WellForm بودن آن دقت لازم را بکنید.به عبارت دیگر از متد هایی که در بالا معرفی شد درست استفاده کنید.فرمت کلی متد های بالا به شکل زیر است:

```
XmlTextWriter.WriteStartDocument()
 XmlTextWriter.WriteStartElement("...")
   XmlTextWriter.WriteAttributeString("...")
     XmlTextWriterWriteElementString("...","...")
      .
      .
      .
 XmlTextWriter.WriteEndElemetn()
XmlTextWriter.Close()
```

در این ساختار دیدید که ویژگی بعد از معرفی تگ نوشته می شود(برخلاف خواندن از سندxml که جلوتر با آن آنا می شوید و ویژگی قبل از خود تگ باید خوانده شود).پس از آن هم فرزندان تگ مربوطه معرفی می شوند و آنها هم حق دارند برای خود فرزندانی اشته باشند.

حال که با اصول کار با کلاس XmlTextWriter آشنا شدید بهتر است یک مثال عملی را با هم ببینیم:

در زیر یک سند xml می بینید که از این به بعد کار ما با آن خواهد بود.

می خواهیم یک سند xml به فرم زیر ایجاد کنیم که نام dvd های موجود یک شرکت است.این dvd ها هر یک id و دسته بندی خاص خود را دارند.و همچنین هر یک نام فیلم نام کارگردان قیمت و ستارگان هر فیلم را داراست.هر فیلم در تگ <DVD></DVD> قرار دارد.به ویژگی ها و فرزندان هر تگ دقت کنید:

```
<?xml version="1.0"?>
```

```
<DVDList>
   <DVD ID="3221" Category="Science Fiction">
      <Title>The Matrix</Title>
      <Director>Larry Wachowski</Director>
      <Price>18.74</Price>
      <Starring>
         <Star>Keanu Reeves</Star>
         <Star>Laurence Fishburne</Star>
      </Starring>
   </DVD>
   ...
</DVDList>
```

برای ایجاد این چنین فایلی در ابتدا باید آدرس آن را مشخص کنیم:

```
Dim xmlPath As String = Server.MapPath("myxml.xml")
```

سپس از کلاس XmlTextWriter استفاده کرده و آن را new می کنیم:

```
Dim writer As New XmlTextWriter(xmlPath, Nothing)
```

پس از آن فرم سند خود را دندانه دار با حداکثر فضای خالی ٤ ایجاد می کنم:

```
  writer.Formatting = Formatting.Indented
writer.Indentation = 4
```

حال شما آماده ی نوشتن فایل هستید.به این منظور از تگ آغازین و همیشگی هر سند xml (تگ "<?xml version="1.0" ?>)شروع می کنیم که با متد زیر ایجاد می شود:

```
writer.WriteStartDocument()
```

سپس یک Comment هم در سر صفحه ایجاد می کنیم که بیان کننده ی زمان ساخت سند باشد:

```
writer.WriteComment("This File Created At " & DateTime.Now.ToString)
```

پس از آن باید محتوای سند را ایجاد کنیم.همانطور که در سند بالا دیدید تگ ریشه ی ما DvdList است:

```
writer.WriteStartElement("DVDList")
```

سپس نوبت به ایجاد فرزندان این تگ است که نامشان DVD است و چون خود DVD هم فرزند دارد پس آن را نیز با WriteStartElement ایجاد می کنیم:

```
writer.WriteStartElement("DVD")
```

حال نوبت ویژگی های تگ DVD است:

```
  writer.WriteAttributeString("ID", "3221")
writer.WriteAttributeString("Category", "Science Fiction")
```

و سپس فرزندان DVD که خودشان فرزند ندارند را با WriteElementString ایجاد می کنیم:

```
  writer.WriteElementString("Title", "The Matrix")
  writer.WriteElementString("Director", "Larry Wachowski")
```

```
writer.WriteElementString("Price", "18.74")
```

سپس فرزند آخر که خودش فرزند دارد به نام Starring را با WriteStartElement ایجاد می کنیم:

```
writer.WriteStartElement("Starring")
```

و فرزندان تگ Starring که دو تگ با نام مشابه Star است:

```
  writer.WriteElementString("Star", "Keanu Reeves")
writer.WriteElementString("Star", "Laurence Fishburne")
```

و سپس بستن آن تگ هایی که با WriteStartElement ایجاد شد که در این تگ DVD ۲ تا بود:

```
  writer.WriteEndElement()
writer.WriteEndElement()
```

نمونه ی کامل یک تگ DVD را در زیر می بینید:

```
          writer.WriteStartElement("DVD")
          writer.WriteAttributeString("ID", "3224")
          writer.WriteAttributeString("Category", "Drama")
          writer.WriteElementString("Title", "Original Sin")
          writer.WriteElementString("Director", "Henri Wacdfgowi")
          writer.WriteElementString("Price", "23.74")
          writer.WriteStartElement("Starring")
          writer.WriteElementString("Star", "Antonio Banderas")
          writer.WriteElementString("Star", "Anjelina Jolie")
          writer.WriteEndElement()
        writer.WriteEndElement()
```

و بستن تگ ریشه که DvdList نام داشت:

```
writer.WriteEndElement()
```

و در نهایت هم بستن Writer:

```
writer.Close()
```

اعمال بالا را می توانید در رویداد Page_Load بنویسید و یا در یک تابع که مثلا آرگومان ورودی آن تابع نام فایلی باشد که می خواهید بسازید.

نکته ی امنیتی در اینجا نام فایل های ساخته شده است که حتما باید Unique باشد.

حال برویم به سراغ خواندن فایل xml البته با برنامه نویسی.

برای خواندن یک فایل xml با برنامه نویسی ۳ کلاس کلی وجود دارد که به ترتیب هر یک را بررسی می کنیم.

۱-XmlDocument.

۲-XmlPathNavigator.

۳-XmlTextReader.

XmlDocument:

با این کلاس که XmlDOM هم به آن می گویند می توانید سند xml خود را درون شی XmlDocument بارگذاری کنید .این کار با متد ()Load صورت می گیرد.پس از این کار می توانید بازیابی های لازم را از سند مربوطه داشته باشید.حال می خواهیم سند xml که پیش از این ایجاد کرده بودیم را با Xml DOM بخوانیم.همانطور که می دانید ساختار سند xml ساختاری درختی است و تگ ها در آن حکم شاخ و برگ درخت را داشته و به هم متصل و پیوسته هستند.

همانطور که گفتیم در اینجا می خواهیم از کلاس XmlDocument استفاده کنیم.برای این کار ابتدا این کلاس باید New شود و سپس از متد Load آن استفاده شود تا فایل xml را در خود جای دهد.متد Load هم یک آرگومان ورودی دارد و آن هم مسیر فایلی است که قرار است خوانده شود.قبل از هر چیز باید با دو شی مهم آشنا شوید:

XmlNode:نوع داده ای است که می تواند یک تگ را جهت استفاده در خود ذخیره کند.

XmlNodeList: نوع داده ای است که می تواند دسته ای از تگ ها را در خود ذخیره کند.

کلاس XmlDocument حاوی متد های زیادی است که به تدریج با برخی از آنهایی که نیاز داریم آشنا می شویم.یک متد مهم کلاس XmlDocument متد ChildNodes است:

```
Dim node as New XmlDocument()
node.ChildNodes
```

این متد تمام فرزندان گره ی ریشه (یا اولین سطح تگ ها) را مشخص می کند که از نوع XmlNodeList است.مثلا در سند xml مربوط به DvD ها اگر از ChildNodes استفاده کنید تمام فرزندان تگ DvdList در قالب داده ای XmlNodeList را به شما بر می گرداند.

کلاس XmlNode هم حاوی متد هایی است. ما در این مثال به متد NodeType کلاس XmlNode نیازمند هستیم.این متد نوع تگ را مشخص می کند که دارای مقادیر مختلفی است که همگی از کلاس XmlNodeType به ارث برده می شوند نظیر:

XmlDeclaration:تشخیص می دهد گره ی مورد نظر تگ اعلان xml است.

Element:تشخیص می دهد گره ی مورد نظر یک تگ معمولی است.

Text:تشخیص می دهد گره ی مورد نظر حاوی Value(مقداری که بین باز و بسته شدن تگ ذکر می شود) است.

Comment:تشخیص می دهد گره ی مورد نظر یک توضیح است.

Attributeَ:مشخص می کند گره ی مورد نظر ویژگی است.این متد در مثال های ما استفاده ای ندارد دلیلش را نیز در همان مثال ها خواهید دید.

ProcessingInstruction:که مانند <?name value?> هستند.

حال کمی با قواعد فرزند پذیری و فرزند شدنی آشنا می شویم با این توضیح که ساختار درختی سند xml این نیست که هر تگ یک فرزند و یک والدی دارد و شاخ و برگ درخت را تنها تگ ها تشکیل دهند به عبارت دیگر درست نیست که ساختار درختی را با شکل ظاهری سند xml سنجید.این ساختار چیزی متفاوت با آن چیزی است که ما در اسناد xml می بینیم.به این صورت که هر گره از این درخت می تواند (تگ-Value-توضیح و ...) باشد.به چند قاعده ی زیر توجه کنید:

۱-گره ی Attribute نمی تواند فرزند گره ی دیگر باشد.ولی خودش می تواند فرزند داشته باشد.فرزند آن می تواند مقدار آن ویژگی باشد یا به عبارت دیگر Text آن ویژگی .یادتان باشد فضاهای نام مثل xmlns="http... نیز چون مانند ویژگی هستند نمی توانند فرزند گره ای باشند.

۲-Comment نمی تواند فرزندی داشته باشد ولی خود می تواند فرزند گره هایی از نوع تگ باشد.

۳-Element می تواند فرزند گره هایی از نوع Element & Document باشد و می تواند حاوی فرزندانی از نوع Element, Text, Comment باشد.

٤-گره از نوع Text نمی تواند حاوی فرزند باشد ولی می تواند فرزند گره از نوع تگ و یا Attribute باشد.

۵- ProcessingInstruction ها نمی توانند فرزند داشته باشند ولی خود می توانند فرزند گره هایی از نوع تگ باشند.

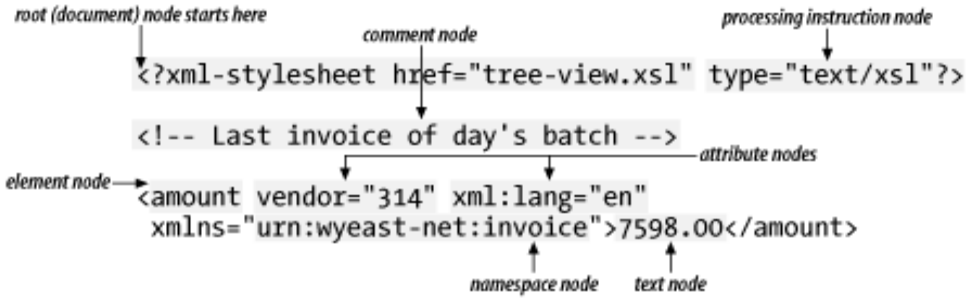
قواعد بالا در درک مثال هایی که در ادامه می زنیم اهمیت بسیاری دارند.

برای درک هرچه بیشتر گره ها سند xml زیر را در نظر بگیرید:

```
<?xml-stylesheet href="tree-view.xsl" type="text/xsl"?>

<!-- Last invoice of day's batch -->

<amount vendor="314" xml:lang="en"
 xmlns="urn:wyeast-net:invoice">7598.00</amount>
```

یک سند ساده ی xml را مشاهده می کنید در خط اول یک ProcessingInstruction تعریف شده (در بحث مربوط به XSLT با ProcessingInstruction آشنا می شوید) که یک فایل Css را به صفحه اعمال می کند.در خط بعدی یک توضیح و در خط بعدی یک تگ به نام amount است که ۲ ویژگی و یک فضای نام و یک مقدار دارد.

شکل زیر گره های مختلف سند بالا را معرفی می کند:.



و عبارت زیر ساختار درختی گره ها را نمایش می دهد:

```
root
 |_ _ _processing instruction target='xml-stylesheet' instruction='href="tree-view.xsl"
type="text/xsl"'
 |_ _ _comment ' Last invoice of day's batch '
 |_ _ _element 'amount' in ns 'urn:wyeast-net:invoice' ('amount')
      | \_ _ _attribute 'vendor' = '314'
      | \_ _ _attribute 'lang' in ns 'http://www.w3.org/XML/1998/namespace' ('xml:lang')
= 'en'
      | \_ _ _namespace 'xml' = 'http://www.w3.org/XML/1998/namespace'
      | \_ _ _namespace 'xmlns' = 'urn:wyeast-net:invoice'
      |_ _ _text '7598.00'
```

طبق این ساختار و قواعدی که ما بیان کردیم گره ی والد تگ root است که در شکل بالا مشخص شده همان <?xml-stylesheet است که تگ آن (نه خود ProcessingInstruction) سه فرزند دارد.اولی از نوع ProcessingInstruction دومی از نوع comment و سومی از نوع element هستند.

ProcessingInstruction و comment که نمی توانند فرزند داشته باشند ولی گره از نوع تگ می تواند فرزند داشته باشد ولی آن ۱ فرزند بیشتر ندارد.همانطور که در شکل مشخص است فرزندان با علامت یک خط صاف به پایین (|) و یک خط صلف به راست (-) مشخص هستند و آنهایی که فرزند نیستند به جای خط راست از خط کج استفاده شده.۲ ویژگی و ۲ فضای نام فرزند نیستند.ولی خودشان می توانند فرزند داشته باشند که تنها فرزندشان مقدارشان است که در شکل مشخص نشده چون اهمیتی ندارد.

رای مثال دیگر سند زیر را در نظر بگیرید:

```
<Servers xmlns="cbtnuggets.com">
    <!--Computers Comment -->
    <Computer ID=1>Test</Computer>
</Servers>
```

در سند بالا ریشه دارای یک فرزند به نام Servers از نوع Element است.خود این Element دارای ۳ فرزند به ترتیب Attribute , Command و Element است و خود Element به نام Computer نیز دارای دو فرزند Attribute و Text می باشد.ولی این حالت ظاهری سند است در حالی که اگر به ۵ قاعده ی بالا توجه کنیم دیگر این گونه نمی شود.کل نکته های بالا(۵ نکته ی مهم) در قالب جدول زیر قابل بیان است:

نوع گره----------Node Type----------	فرزندان----------Children
ریشه ی سند XML	تگ(حداکثر یکی)-پردازشگر دستور-توضیح
تگ(Element)	تگ-پردازشگر دستور-توضیح-متن
ویژگی(Attribute)	متن مربوط به ویژگی
متن(Text)	ندارد
پردازشگر دستور(Processing Instruction)	ندارد
توضیح(Comment)	ندارد

جدول زیر نیز مقادیر برگشتی گره های سند Xml را نشان می دهد:

Node type	nodeName returns	nodeValue returns
Document	#document	null
Element	element name	null

Attr	attribute name	attribute value
ProcessingInstruction	target	content of node
Comment	#comment	comment text
Text	#text	content of node

پس با ٤ عنصر بالا متوجه شدید که هر یک ازآنها یک گره هستند که نیاز به تشخیص دارند.

از دیگر ویژگی های کلاس XmlNode به موارد زیر نیز می توان اشاره کرد:

Attributes:مشخص کننده ی ویژگی های یک گره است.

HasChildNodes:مشخص می کند گره ی مورد نظر حاوی فرزند هست یا نه.

ParentNode:مشخص کننده ی گره ی والد گره ی مورد نظر است.

FirstChild:مشخص کننده ی اولین فرزند گره ی مورد نظر است.

LastChild:مشخص کننده ی آخرین فرزند گره ی مورد نظر است.

Value:مشخص کننده ی Value گره ی مورد نظر است.

ChildNodes:مشخص کننده ی تمام فرزندان گره ی مربوطه(XmlNode) است.

حال که با برخی از متد ها و ویژگی ها آشنا شدید(در ادامه باز هم آشنا می شوید) می خواهیم به سراغ ایجاد یک تابع برویم که به وسیله ی آن از روش Xml DOM برای خواندن از xml استفاده شود.

در ابتدا باید بدانید وقتی ما دسته ای از تگ ها را در یک شی XmlNodeList داشته باشیم رای پیمایش هر فرزند نیاز به یک بار صدا زدن تابعی داریم که آن را به همراه ویژگی ها و فرزندانش برای ما باز یابی کند.در ساختار درختی یک فرمت تودتو وجود دارد که ما نمی دانیم هر فرزند خودش حاوی چند فرزند دیگر است به همین دلیل باید ازیک تابع بازگشتی با یک شرط بازگشت استفاده کنیم.علاوه بر این می خواهیم وقتی سند xml خوانده می شود آن را به فرمت دندانه دار نمایش دهیم نه فرمت معمولی.

قبل از خود تابع در رویداد Page_Load ابتدا مسیر سند مورد نظر را ایجاد کرده:

```
Dim xmlPath As String = Server.MapPath("myxml.xml")
```

سپس یک شی XmlDocument ایجاد کرده :

```
Dim doc As New XmlDocument()
```

سپس از متد Load آن استفاده کرده و آرگومان ورودی آن را هم مسیر مورد نظر در نظر می گیریم:

```
doc.Load(xmlPath)
```

حالا نوبت به صدا زدن تابع است:

```
Label1.Text = xmlreading(doc.ChildNodes, 0)
```

فعلا تا همینجا بدانید که یک تابع به نام xmlreading با دو آرگومان ورودی ایجاد کردیم .اولی لیستی از گره هاست از نوع XmlNodeList.همانطور که می دانید متد XmlDocument.ChildNodes تمام گره های فرزند گره ی ریشه را به ما بر می گرداند.و این متد با متد XmlNode.ChildNodes متفاوت است زیرا XmlNode.ChildNodes تمام فرزندان گره ی مربوطه(به XmlNode) را برمی گرداند.دومین آرگومان هم یک عدد Integer است که کاربردش را در خود تابع خواهید دید.

پس رویداد Page_Load به صورت زیر شد:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim xmlPath As String = Server.MapPath("myxml.xml")
        Dim doc As New XmlDocument()
        doc.Load(xmlPath)
        Label1.Text = xmlreading(doc.ChildNodes, 0)
    End Sub
```

تابع را به صورت زیر تعریف می کنیم:

```
    Private Function xmlreading(ByVal nodeList As XmlNodeList, ByVal level
As Integer) As String

    End Function
```

از آنجایی که می خواهیم سند xml به صورت دندانه دار تو در تو نشان داده شود می بایست فاصله های مور نظر را به تگ ها اعمال کنیم.آرگومان دوم تابع Level هم به همین منظوراستفاده می شود.علامت فاصله در html با   نمایش داه می شود و ما هم از آن به منظور ایجاد فضای خالی یا WhiteSpace استفاده می کنیم.برای ایجاد فضای خالی از یک حلقه ی Do While به صورت زیر استفاده می کنیم:

```
        Dim indent As String = ""
        Dim i As Integer = 0
        Do While i < level
            indent &= "     "
            i += 1
```

```
    Loop
```

طرز کار این حلقه این گونه است که در اولین مرتبه Level=0 به تابع ارسال می شود و چون شرط i<level مورد قبول نیست وارد حلقه نمی شویم و متغیر Indenent بدون فاصله می ماند و ما برای نمایش اولین تگ فاصله ای برایش قایل نمی شویم.ولی در اجرای بازگشتی تابع Level+1 به تابع فرستاده می شود و ما وارد حلقه ی While می شویم و Indent حاوی ۳ فضای خالی می شود.پس هر مرحله که به جلو می رویم ۳ فضای خالی ایجاد می شود و این امر باعث می شود فرمت و شکل دندانه دار xml در خروجی تابع ما هم حفظ شود به این صورت که ما متغیر Indent را پس از چرخش در حلقه ی while به اول نام تگی که می خواهیم نمایش دهیم می چسبانیم.در ادامه نحوه ی نمایش تگ ها و ویژگی ها را می آموزید.

ما عمل نمایش را در خارج از تابع انجام می دهیم به این صورت که تمام مقادیر بازیابی شده را در یک متغیر رشته ای ذخیره می کنیم و پس از اتمام کار آن را توسط تابع برمی گردانیم.پس این متغیر را به صورت زیر تعریف می کنیم:

```
Dim str As String = String.Empty
```

از آنجایی که با آرگومان NodeList یک Collection از گره ها در اختیار ماست برای ملاقات هر یک از گره ها باید از یک حلقه ی For Each استفاده کنیم به این صورت که در هر بار چرخش حلقه نوع گره ی انتخاب شده را تشخیص می دهیم و بر اساس این تشخیص آن را در خروجی چاپ می کنیم. چرخش این حلقه توسط XmlNode روی XmlNodeList انجام می شود که XmlNodeList همان NodeList(آرگومان ورودی تابع است).پس ابتدا حلقه را ایجاد می کنیم:

```
    For Each node As XmlNode In nodeList

    Next
```

برای تشخیص گره ها از متد NodeType استفاده می کنیم.زیرا در هر بار چرخش حلقه ما یک گره از نوع XmlNode در اختیار داریم و می توانیم تشخیص دهیم این گره از چه نوعی است(اعلان اولیه-تگ-نوشته ایکه میان باز وبسته شدن تگ ذکر می شود-توضیح و...).این کار را بر عهده ی یک Select Case می گذاریم:

```
Select Case node.NodeType
   Case XmlNodeType.XmlDeclaration
                    str += "XML Declaration: "
                    str += node.Name
                    str += " "
```

```
                    str += node.Value
                    str += "<br/>"

        Case XmlNodeType.Element
                    str += indent
                    str += "Element: "
                    str += node.Name
                    str += "<br/>"

        Case XmlNodeType.Text
                    str += indent
                    str += "Value: "
                    str += node.Value
                    str += "<br/>"

        Case XmlNodeType.Comment
                    str += indent
                    str += "Comment: "
                    str += node.Value
                    str += "<br/>"
End Select
```

برای مثال اگر گره ی مربوطه یک تگ ساده بود ابتدا str+=indent می شود تا با توجه به Level تگ مربوطه فاصله ی درست به آن اختصاص داده شودسپس با Node.Name نام تگ را نمایش می دهیم حال اگر گره حاوی Value هم باشدچون Value هم یک گره محسوب می شود دوباره توسط تابع بازگشتی (در چرخش بعدی حلقه ی For Each) که جلوتر در انتهای حلقه ی for قرار می دهیم وارد Select Case شده و این بار XmlNodeType.Text مورد پذیرش خواهد بود و مقدار تگ با متد Node.Value به رشته ی str اضافه خواهد شد و به همین ترتیب کار ادامه می یابد تا وقتی که حلقه ی For Each به پایان برسد و تمام گره ها بررسی شوند.نکته ی مهم اینجا این است که هر یک از عناصر موجود در سند xml یک گره است که نیاز به تشخیص دارد مثلا Value یک گره است.نکته ی دیگر این است که ما در Select Case, Attribute را تست نکردیم چون همانطورکه می دانید Attribute گره است ولی نمی تواند فرزند گره ای دیگراز جمله تگ باشد.و همینطور ما تابع باز گشتی را با فرزندان یک گره صدا می زنیم(جلوتر نحوه ی صدا زدن تابع بازگشتی را می بینید) .پس نتیجه می گیریم ویژگی ها اصلا جزو گره هایی که ما در داخل تابع به آنها برمی خوریم نخواهد بود و تنها راه ملاقات ویژگی ها این است که یک گره از نوع تگ باشد و ما یک راست به

سراغ ویژگی آن تگ برویم.مثلا فرض کنید در این مثال نوبت به گره ی DVD است.در عبارت Select Case ما DVD را تشخیص می دهیم و نام آن را به str اضافه می کنیم سپس از Select Case خارج می شویم و چک می کنیم گره ی مورد نظر(که با Select Case فهمیدیم تگ dvd است) آیا حاوی ویژگی هست یا نه(که حاوی دو ویژگی Id & Category است).اگر بود با یک حلقه ی For Each پیمایشی روی ویژگی های آن خواهیم داشت .برای چک کردن این که گره ی مورد نظر ویژگی دارد یا نه از متد Attributes شی XmlNode استفاده می کنیم:

```
If Not node.Attributes Is Nothing Then
    For Each attrib As XmlAttribute In node.Attributes
        str += indent
        str += "Attribute: "
        str += attrib.Name
        str += "-Value: "
        str += attrib.Value
        str += "<br/>"
    Next
End If
```

پس با شی XmlAttributeهم آشنا شدید که یک ویژگی را به خود اختصاص می دهد و ما به وسیله ی آن در میان ویژگی های موجود (XmlNode.Attributes) حرکت می کنیم تا تک تک آنها را به همراه Name & Valueشان نمایش دهیم.

یادتان باشد که این کد درون حلقه ی For Each بالا یی قرار می گیرد.

و در نهایت می رسیم به شرط بازگشت که در همه ی توابع بازگشتی باید ذکر شود چون هر تابع بازگشتی بالاخره یک جایی باید به اتمام برسد:

```
If node.HasChildNodes Then
  str += xmlreading(node.ChildNodes, level + 1)
End If
```

در این کد ما تابع را به صورت بازگشتی ولی با آرگومان های متفاوت صدا زدیم.باتوجه به ساختار درختی سند xml هر گره می توان دارای فرزند باشد.اگر این چنین بود وارد شرط if می شویم و تابع ا با فرزندان گره ی مربوطه همچنین یک level بیشتر صدا می زنیم.

و در انتها رشته ی مورد نظر را برگشت می دهیم.این کار را خارج از تمامی حلقه ها اجام می دهیم:

```
Return str
```

فرم کلی تابع به صورت زیر می شود:

```
        Private Function xmlreading(ByVal nodeList As XmlNodeList, ByVal level
    As Integer) As String
            Dim indent As String = ""
            Dim i As Integer = 0
            Do While i < level
                indent &= "     "
                i += 1
            Loop
            Dim str As String = String.Empty
            For Each node As XmlNode In nodeList
                Select Case node.NodeType
                    Case XmlNodeType.XmlDeclaration
                        str += "XML Declaration: "
                        str += node.Name
                        str += " "
                        str += node.Value
                        str += "<br/>"

                    Case XmlNodeType.Element
                        str += indent
                        str += "Element: "
                        str += node.Name
                        str += "<br/>"

                    Case XmlNodeType.Text
                        str += indent
                        str += "Value: "
                        str += node.Value
                        str += "<br/>"

                    Case XmlNodeType.Comment
                        str += indent
                        str += "Comment: "
                        str += node.Value
                        str += "<br/>"
                End Select

                If Not node.Attributes Is Nothing Then
                    For Each attrib As XmlAttribute In node.Attributes
                        str += indent
                        str += "Attribute: "
                        str += attrib.Name
                        str += "-Value: "
                        str += attrib.Value
                        str += "<br/>"
                    Next
```

```
            End If

            If node.HasChildNodes Then
                str += xmlreading(node.ChildNodes, level + 1)
            End If
        Next

        Return str
    End Function
```

می توانید از متد LoadXmlبه جای متد Load استفاده کنید آنوقت دیگر نیازی به نام فایل به عنوان آرگومان ورودی نیست بلکه خود فایل را باید ذکر کنید مثلا:

```
Dim doc As New XmlDocument()
doc.LoadXml("<book ISBN='1-861001-57-5'>" & _
            "<title>Pride And Prejudice</title>" & _
            "<price>19.95</price>" & _
          "</book>")
```

بقیه ی موارد مثل بالا است.

XmlPathNavigator:

روش بعدی استفاده از کلاس XPathNavigator است.قبل از هر چیز باید فضای نام System.Xml.XPath را Imports کنید.کار با این متد بسیار شبیه XmlDocument است و حتی از XmlDocument هم استفاده می کنیم ولی یک تفاوت عمده در آن این است که XPathNavigator در هر بار می تواند تنها حامل یک گره باشد.مثلا اگر بخواهیم تابع بازگشتی را صدا بزنیم باید هر بار یک گره را صدا بزنیم به جای یک دسته از گره ها.همچنین برای پیمایش میان گره ها باید از متد MoveToNext() استفاده کنیم (نه مثلا یک حلقه ی For Each) که به گره ی بعدی اشاره می کند.برای استفاده از XPathNavigator از متد CreateNavigator کلاس XmlDocument استفاده می کنیم(تا یک Navigator در Document ایجاد شود) وبرای استفاده آن را درون متغیری از جنس XPathNavigator قرار می دهیم.پس رویداد Page_Load به صورت زیر می شود:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim xmlPath As String = Server.MapPath("myxml.xml")
        Dim doc As New XmlDocument()
        doc.Load(xmlPath)
        Dim xnav As XPathNavigator = doc.CreateNavigator()
```

```
        Label1.Text = xmlreading(xnav, 0)
    End Sub
```

تابع را نیز به صورت زیر تعریف می کنیم:

```
    Private Function xmlreading(ByVal xnav As XPathNavigator, ByVal level
As Integer) As String

    End Function
```

می بینید که آرگومان ورودی اول در اینجا از جنس XPathNavigator است.

حلقه ی Do While دقیقا مثل روش XmlDocument خواهد بود.

همانطور که گفتیم XPathNavigator در هر بار حاوی یک گره خواهد بود پس SelectCase بدون حلقه ی For Each ایجاد می شود چون یک گره دیگر نیازی به پیمایش ندارد.برای تشخیص نوع گره هم به جای شی XmlNode از متد NodeType شی ارسالی XPathNavigator که در اینجا Xnav است استفاده می کنیم.مقادیر آن هم مثل روش قبل هستند با این تفاوت که از کلاس XPathNodeType به جای XmlNodeType ارث می برد:

```
  Select Case xnav.NodeType
             Case XPathNodeType.Root
                 str += "ROOT"
                 str += "<br/>"
             Case XPathNodeType.Element
                 str += indent
                 str += "Element: "
                 str += xnav.Name
                 str += "<br/>"
             Case XPathNodeType.Text
                 str += indent
                 str += "Value: "
                 str += xnav.Value
                 str += "<br/>"

             Case XPathNodeType.Comment
                 str += indent
                 str += "Comment: "
                 str += xnav.Value
                 str += "<br/>"
End Select
```

کلاس XPathNavigator حاوی متد HasAttribute هم هست که می توان از آن برای تست داشتن یا نداشتن ویژگی استفاده کرد.در این مثال هم مثل مثال قبل و به همان دلیل ذکر شده حتما ویژگی ها را باید بیرون Select Case صدا زد.برای بازیابی ویژگی ها

و نوشتن آنها در متغیر str می بایست از متد MoveToFirstAttribute() شی XPathNavigator استفاده کنیم و هر بار که آن را نمایش دادیم با متد MoveToNextAttribute() به ویژگی بعدی برویم تا....این کار را در یک حلقه ی Do While به صورت زیر انجام می دهیم:

```
If xnav.HasAttributes Then
    xnav.MoveToFirstAttribute()
    Do
        str += indent
        str += " - Attribute: "
        str += xnav.Name
        str += " Value: "
        str += xnav.Value
        str += "<br/>"
    Loop While xnav.MoveToNextAttribute()
    xnav.MoveToParent()
End If
```

همانطور که می بینید در ابتدا چک کردیم اگر گره ی مورد نظر به شرطی که در SelectCase از نوع تگ باشد حاوی ویژگی بود به اولین ویژگی برو و وارد حلقه شو.حلقه را نیز به این صورت نوشتیم تا در اولین بار وارد حلقه شویم چون در صورت داشتن حتی یک ویژگی با شرط HasAttribute قبلا چک شده بود و می توانیم وارد حلقه شویم.متد MoveToNextAttribute هم در هر بار True یا False برمی گرداند پس وقتی آن را در شرط While قرار دادیم هر گاه ویژگی ها تمام شد خودش باعث خروج از حلقه می شود.در انتها هم با متد MoveToParent به گره ی والد برمی گردیم چون یک ویژگی هم یک گره محسوب می شود و می بایست برای ادامه ی کار به خود تگ(گره ای که ازش آمدیم به عنوان آرگومان ورودی تابع ارسال شد) برگردیم تا هنگام صدا زدن تابع بازگشتی با گره ای که به عنوان آرگومان ورودی تابع ارسال شد این بار تابع را با گره ی بعدی آن ارسال کنیم چون متغیر xnav در حال حاضر روی یک ویژگی ایستاده و اگر MoveToParent نباشد ما تابع را با یک ویژگی صدا می زنیم و همانطور که می دانید ChildNodes یک ویژگی مقدار آن است و این کار باعث توقف و بروز خطا در برنامه ی ما می شود.پس این کار باعث می شود در استفاده ی مجدد از تابع بازگشتی دچار مشکل نشویم.به عبارت دیگر برای صدا زدن تابع بازگشتی ابتدا چک می کنیم اگر حاوی فرزند باشد آنگاه به اولین فرزند برو و تابع را به ازای آنها صدا بزن در حالیکه اگر متد MoveToParent به کار نمی رفتXnav حاوی یک ویژگی بود و با متد

MovetofirstChild به آن ویژگی دسترسی داشتیم و با صدا زدن تابع با آن ویژگی حسابی دچار مشکل می شدیم.

نحوه ی صدا زدن تابع بازگشتی و همچنین شرط بازگشت هم به صورت زیر است:

```
If xnav.HasChildren Then
    xnav.MoveToFirstChild()
    Do
        str += xmlreading(xnav, level + 1)
    Loop While xnav.MoveToNext()
    xnav.MoveToParent()
End If
```

در ابتدا چک کردیم اگر گره ی مورد نظر حاوی فرزند باشد وارد if شو.سپس به اولین فرزند گره ی مورد نظر می رویم و با متد MoveToNext فرزند به فرزند تابع را به فرم بازگشتی صدا می زنیم .این متد هم مقدار True و False برمی گرداند و به محض اتمام فرزندان از While خارج می شود و مجددا با متد MoveToParent به والد گره باز می گردیم(این کار در XPathNavigator به صورت دستی انجام می گیرد چون متد MoveToNext وضعیت گره ای که در داخل متغیر xnav قرار دارد را تغییر می دهد و آن را به گره ی فرزند می برد در حالی که برای پیمایش درست گره ها می بایست حتما به والد برگردیم).پس کد کلی تابع به فرم زیر شد:

```
Private Function xmlreading(ByVal xnav As XPathNavigator, ByVal level
As Integer) As String
    Dim str As String = String.Empty
    Dim indent As String = ""
    Dim i As Integer = 0
    Do While i < level
        indent += "     "
        i += 1
    Loop

    Select Case xnav.NodeType
        Case XPathNodeType.Root
            str += "ROOT"
            str += "<br/>"
        Case XPathNodeType.Element
            str += indent
            str += "Element: "
            str += xnav.Name
            str += "<br/>"
        Case XPathNodeType.Text
            str += indent
            str += "Value: "
```

```
                str += xnav.Value
                str += "<br/>"

            Case XPathNodeType.Comment
                str += indent
                str += "Comment: "
                str += xnav.Value
                str += "<br/>"
        End Select

        If xnav.HasAttributes Then
            xnav.MoveToFirstAttribute()
            Do
                str += indent
                str += " - Attribute: "
                str += xnav.Name
                str += " Value: "
                str += xnav.Value
                str += "<br/>"
            Loop While xnav.MoveToNextAttribute()
            xnav.MoveToParent()
        End If

        If xnav.HasChildren Then
            xnav.MoveToFirstChild()
            Do
                str += xmlreading(xnav, level + 1)
            Loop While xnav.MoveToNext()
            xnav.MoveToParent()
        End If

        Return str
    End Function
```

البته انواع گره ی دیگری هم وجود دارند مثل EndElement که مشخص کننده ی </element name> است.

دقت کنید ساختار درختی سند xml که چندبار بهش اشاره کردیم این نیست که هر تگ یک فرزند و یک والدی دارد و درست نیست که ساختار درختی را با شکل ظاهری سند xml سنجید.این ساختار چیزی متفاوت با آن چیزی است که ما در اسناد xml می بینیم طبق مثال ساده ای که از این ساختار به صورت تصویری با هم پیش از این دیدیم.به این صورت که هر گره از این درخت می تواند (تگ-Value-توضیح و ...) باشد و در بالا وقتی می گوییم مثلا MoveToParent ما این Parent را در اصل نمی بینیم و نمی

دانیم که دقیقا چیست(تگ است یا توضیح است و یا شاید ویژگی و یا Value است)ولی برای صحت کار از آن استفاده می کنیم(تشخیص آن به عهده ی Select Case است) و در ساختار درختی اصلی که در XmlDocument ایجاد شده ما بین گره ها حرکت می کنیم.در جایی دیگر گفتیم MoveToNext.این یعنی به گره ی بعدی برو و یا MoveToFirstChild که به اولین فرزند گره برو.در صورتی که عملا درک ساختار این گره ها به میزان آشنایی شما با درخت ها برمی گردد.

در تشریح کلی تابع بالا می توان گفت:

ابتدا یک گره توسط آرگومان تابع در یافت می شود.سپس نوع آن تشخیص داده می شود.اگر هیچ یک از انواع ذکر شده نبود Attribute را چک می کنیم.این کار را با شرط "آیا دارای ویژگی است؟" چک کردیم.اگر ویژگی بود از اولین ویژگی تا آخرین ویژگی را چاپ می کنیم ودر نهایت به والد برمی گردیم یعنی همان گره ای که به عنوان آرگومان ورودی تابع ذکر شده بود زیرا هر ویژگی یک گره است و برای اینکه کار از دست ما در نرود باید به گره ای که ازش آمده ایم (تگی که ویژگی در آن است) برگردیم.پس از آن چک کردیم "آیا فرزندی هنوز باقی مونده؟" اگر مانده بود چون ما نمیدانیم چند تا از اولین فرزند شروع می کنیم تا آخرین فرزند و تابع را به صورت بازگشتی برای تک تک این فرزند ها صدا می زنیم چون هر فرزند هم می تواند حاوی فرزند-توضیح-Value-و... هم باشد.و در انتها به گره ای که ازش آمدیم باز می گردیم و کار تمام می شود.

نکته ی بسیار مهم در اینجا و در ادامه این است که در ساختار درختی سند xml Attribute, به هیچ وجه نمی تواند فرزند هیچ نوع گره ای باشد.ولی خود می تواند فرزند داشته باشد و آن هم Text است.مثلا:

```
<Price Id=”123”></Price>
```

در اینجا گره ی Id فرزند تگ Price نیست ولی خودش فرزند دارد و آن هم 123 است.

پس تفاوت عمده بین دو روش ذکر شده جهت خواندن از فایل xml نحوه ی پیمایش و برخی متد ها یی که اولی داشت و دومی نداشت و یا برعکس بود.مثلا در روش اول ما با یک حلقه پیمایش کردیم ولی در روش دوم با متد های MoveToFirst & MoveToNext.اگر کمی فکر کنید هر دو روش بسیار ساده بود.برای مثال به دو قسمت مشترک هر دو که برای بازیابی Attribute بود توجه کنید:

```
If xnav.HasAttributes Then
   xnav.MoveToFirstAttribute()
```

```
If Not node.Attributes Is Nothing
Then
```

```
   Do
     str += indent
     str += " - Attribute: "
     str += xnav.Name
     str += " Value: "
     str += xnav.Value
     str += "<br/>"
   Loop While
xnav.MoveToNextAttribute()
   xnav.MoveToParent()
End If
```

```
   For Each attrib As XmlAttribute
In node.Attributes
     str += indent
     str += "Attribute: "
     str += attrib.Name
     str += "-Value: "
     str += attrib.Value
     str += "<br/>"
  Next
End If
```

خط اول هر دو:

```
      If xnav.HasAttributes Then
```

```
If Not node.Attributes Is Nothing
Then
```

می بینید که کلاس XPathNavigator دارای متد HasAttribute است ولی XmlNode حاوی متد Attributes بوده و برای استفاده از IsNot Nothing استفاده کردیم.

خط بعدی پیمایش بین ویژگی ها:

```
   xnav.MoveToFirstAttribute()
   Do
   Loop While
xnav.MoveToNextAttribute()
   xnav.MoveToParent()
```

```
For Each attrib As XmlAttribute In
node.Attributes
  Next
```

در کد سمت راست ما یک حلقه برای پیمایش استفاده کردیم چون XmlNode حاوی متدی به نام MoveToFirst یا ... نبود.نکته ی بسیار مهم دیگر این است که در استفاده از XmlDocument کد ها ی بالا همگی در داخل یک حلقه ی For Each قرار داشت زیرا ما با دسته ای از گره ها طرف بودیم در حالیکه در روش XPathNavigator چون با تنها یک گره طرف بودیم خبری از حلقه نبود.ولی در عوض وقتی می خواستیم تابع را به صورت بازگشتی صدا بزنیم کد ها به صورت زیر شد:

```
If xnav.HasChildren Then
  xnav.MoveToFirstChild()
  Do
    str += xmlreading(xnav, level +
1)
  Loop While xnav.MoveToNext()
  xnav.MoveToParent()
End If
```

```
If node.HasChildNodes Then
   str += xmlreading(node.ChildNodes,
level + 1)
End If
```

می بینید که هنگام صدا زدن تابع بازگشتی در سمت راست چون شرط if در داخل حلقه ی For Each قرار دارد نیازی نیست پس از پیمایش هر گره از حلقه استفاده کنیم ولی در کد سمت چپ باید پیمایش صورت می گرفت چون تا قبل از این اصلا حلقه ای به منظور

پیمایش گره ها وجود نداشت .این کار را نیز با متد های MoveToFirstChild و MoveTo Next انجام دادیم.

ببینید در کل هنگامی نیاز به استفاده از MoveToFirstChild و دیگر متد هایی نظیر آن دارید که بحث XPathNavigation مطرح باشد.با اینکه جلوتر با متد Move... های مختلف سروکار خواهیم داشت ولی در آنجا نیاز به این همه دقت نیست.مثلا حتما نباید هنگام بازیابی ویژگی ها ابتدا MoveToFirstAttribute() را صدا بزنیم و در انتها برگردیم به گره ی والد زیرا XPathNavigation پیمایشی بسیار حساس است و از طرف دیگر ما در اینجا کل سند را با تابع باز گشتی صدا زدیم در حالیکه جلوتر ما نیازی به بازیابی کل نداریم و تنها بازیابی های جزیی برایمان مهم است.در کل شما الزامی به استفاده از XPathNavigation ندارید و تنها جهت آشنایی ذکر شد.در جلوتر با روشهایی بسیار ساده تر برای بازیابی داده از xml آشنا می شوید مثل استفاده از کنترل XmlDataSource و یا کنترل قدیمی Xml.

قبل از روش XmlTextReader کمی با جستجو در سند Xml آشنا خواهیم شد:

در اولین روش می خواهیم از متد GetElementByName() کلاس XmlDocument استفاده کنیم و به دنبال محتویات(Value) تگی خاص بگردیم.بنابراین یک تابع می نویسیم تا این کار را انجام دهد و آرگومان ورودی آن نام تگی است که ما به دنبال Value اش هستیم:

```
Private Function searchByTagName(ByVal tagName As String) As String
    Dim xmlPath As String = Server.MapPath("myxml.xml")
    Dim doc As New XmlDocument()
    doc.Load(xmlPath)

End Function
```

برای اینکه به محتویات تگی خاص دسترسی پیدا کنیم می بایست از متد GetElementByName() کلاس XmlDocument استفاده کنیم.این متد به عنوان آرگومان ورودی نام تگ مربوطه را دریافت می کند و مقدار برگشتی آن از نوع XmlNodeList است.یعنی لیست تمام گره ها با نام مربوطه را بر می گرداند:

```
Dim nodelist As XmlNodeList = doc.GetElementsByTagName(tagName)
```

حال ما لیست گره هایی که اولا از نوع تگ هستند (چون متد GetElementByTagName به دنبال تگ می گردد) و دوما هم نامشان آرگومان

ورودی متد GetElementByName است را در اختیار داریم و برای بدست آوردن Value آنها از یک حلقه ی For Each استفاده می کنیم:

```
For Each node As XmlNode In nodelist

Next
```

حتما یادتان هست متد ChildNodes تمام فرزندان یک گره را برمی گرداند.حال اگر به آن یک آرگومان ورودی از نوع Integer بدهیم اندیس فرزندی را که باید به ما بدهد را مشخص می کند در حقیقت:

Return Value Of ChildNodes is XmlNodeList

Return Value Of ChildNodes(i as integer) is XmlNode

در اینجا هم در هر بار چرخش حلقه ی For اگر از ChildNodes(0) استفاده کنیم اولین فرزند تگ مربوطه را به ما می دهد.البته این متد با Value کامل می شود:

```
node.ChildNodes(0).Value
```

پس در حقیقت ChildNodes(0).Value تگ مورد نظر را برمیگرداند.به جای Value می توانید از صفت InnerText هم استفاده کنید.

به عبارت دیگر node که همان تگ Title است.پس Node.ChildNodes فرزندان تگ Title هستند و Node.ChildNodes(0).Value هم Value تگ مورد نظر است.برای مثال اگر آرگومان ورودی این تابع Title باشد خروجی آن عنوان تمام فیلم هاست.یادتان باشد در ساختار درختی سند xml, Attribute به هیچ وجه نمی تواند فرزند هیچ نوع گره ای باشد.به همین دلیل اگر ورودی تابع را نام DVD بدهید به شما خروجی نمی دهد چون متد ChildNodes(0).Value مقدار اولین فرزند تگ DVD را برکی گرداند در حالیکه ویژگی های ID و CategoryName جزو فرزند های DVD حساب نمی شوند و فرزند آن تگ Title است که از نوع Element است و نمی تواند در خروجی چاپ شود.

در ادامه می خواهیم عمل جستجو را کمی پیچیده کنیم.مثلا می خواهیم با دادن عنوان فیلم بازیگران آن فیلم را در خروجی دریافت کنیم.ابتدا تابع مثال قبل را با اندکی تغییر بنویسید:

```
    Private Function searchByTagName(ByVal smn_for_star As String) As
String
        Dim xmlPath As String = Server.MapPath("myxml.xml")
```

```
        Dim doc As New XmlDocument()
        doc.Load(xmlPath)
        Dim nodelist As XmlNodeList = doc.GetElementsByTagName("Title")
        Dim str As String = String.Empty
        For Each node As XmlNode In nodelist
            Dim name As String = node.ChildNodes(0).Value


        Next
        Return str
    End Function
```

در داخل حلقه ی for each به دنبال عنصر مورد نظر خواهیم گشت.در ابتدا چک می کنیم صحت نام فیلم را .به این صورت که Title Value فیلم مورد نظر را با آرگومان ورودی تابع که ما به عنوان نام فیلم مورد نظرمان قرار دادیم مقایسه می کنیم:

```
    If name = smn_for_star Then

  End If
```

ولی دقت کنید ما در حال حاضر در گره ی Title هستیم و به دنبال تگ Satr می گردیم تا اسامی بازیگران را برگردانیم.در حالیکه تگ Satr عضو فرزندان Title نیست.پس یک جوری باید تگی را اختیار کنیم که تگ Satr جزو فرزندانش باشد این کار را با متد ParentNode از شی XmlNode انجام می دهیم:

```
Dim parent As XmlNode = node.ParentNode
```

برای پیدا کردن تگ هایی با نام Star باید از متد GetElementByTagName() استفاده کنیم ولی Parent دارای چنین متدی نیست.برای جلوگیری از این مشکل آن را به تگ تبدیل می کنیم(البته پیش خودمان):

```
Dim a As XmlElement = CType(parent, XmlElement)
```

و سپس به روش عادی محتویات تگ Star را در خروجی چاپ می کنیم:

```
  For Each node1 As XmlNode In nodelist1
                    str += "Found Star : <br/>"
                    str += node1.ChildNodes(0).Value
                    str += "<br/>"
Next
```

پس تابع کلی به شکل زیر می شود:

```
      Private Function searchByTagName(ByVal smn_for_star As String) As
String
        Dim xmlPath As String = Server.MapPath("myxml.xml")
        Dim doc As New XmlDocument()
        doc.Load(xmlPath)
        Dim nodelist As XmlNodeList = doc.GetElementsByTagName("Title")
```

```
            Dim str As String = String.Empty
            For Each node As XmlNode In nodelist
                Dim name As String = node.ChildNodes(0).Value
                If name = smn_for_star Then
                    Dim parent As XmlNode = node.ParentNode
                    Dim a As XmlElement = CType(parent, XmlElement)
                    Dim nodelist1 As XmlNodeList =
a.GetElementsByTagName("Star")
                    For Each node1 As XmlNode In nodelist1
                        str += "Found Star : <br/>"
                        str += node1.ChildNodes(0).Value
                        str += "<br/>"
                    Next
                End If
            Next
            Return str
    End Function
```

حتما می دانید بسیاری از سند های xml حاوی NameSpace هستند.با روش بالا نمی توان تگ هایی را که حاوی فضای نام هستند را در خروجی نمایش داد.برای مثال سند ساده ی زیر را در نظر بگیرید:

```
<?xml version="1.0" encoding="utf-8" ?>
<cli:order  xmlns:cli="http://mycompany/ClientML">
    <cli:client>
          <cli:firstName>ali</cli:firstName>
          <cli:lastName>reza</cli:lastName>
    </cli:client>
</cli:order>
```

ما می خواهیم Value تگ <cli:firstname> را نمایش بدهیم.برای این کار از آرگومان ورودی دوم متد GetElementByTagName استفاده می کنیم و فضای نام مربوطه را به آن میدهیم:

```
GetElementsByTagName("firstName", "http://mycompany/ClientML")
```

پس تابع مخصوص بازیابی آن به شکل زیر می شود:

```
    Private Function searchByTagName() As String
        Dim xmlPath As String = Server.MapPath("namespace.xml")
        Dim doc As New XmlDocument()
        doc.Load(xmlPath)
        Dim nodelist As XmlNodeList = doc.GetElementsByTagName("firstName",
"http://mycompany/ClientML")
        Dim str As String = String.Empty
        For Each node As XmlNode In nodelist
            str += "Found : <br/>"
            Dim name As String = node.ChildNodes(0).Value
            str += name
```

```
            str += "<br/>"
        Next
        Return str
    End Function
```

تا بحال جستجو های ما معیارخاصی نداشت و نمی توان همیشه از موارد بالا استفاده کرد برای رفع این محدودیت ها می توان از XPath استفاده کرد.XPath مثل یک Query است که به سادگی می توان همه نوع جستجو با آن انجام داد .برای این کار می بایست با Syntax XPath آشنا شوید.در زیر مهمترین آنها آمده است:

/

نشان دهنده ی گره ی ریشه است.مثلا:

/DVDList/DVD

تمام تگ هایی با نام DVD که فرزند DVDList باشند را انتخاب می کند.

//

نشان دهنده ی گره ی ریشه است با این تفاوت که همراه فرزندان نوادگان آن را نیز در بر می گیرد.مثلا:

//DVD/Title

تمام گره های Title که جزو نوادگان DVD هستند.

فرق فرزند با نوادگان این است که فرزند یک تگ زیرین است ولی نوادگان شامل هم فرزند و هم فرزندان فرزند یک گره هم می شود.

@

باعث انتخاب ویژگی ها از یک گره(از نوع تگ) می شود.مثلا:

/DVDList/DVD/@Id

ویژگی ID را از تگ DVD که فرزند DVDList است را انتخاب کن.

*

مشخص کننده ی همه است.ولی همه ی فرزندان نه نوادگان. مثلا:

/DVDList/DVD/*

تمام فرزندان DVD را انتخاب کن (DVD هایی که خود فرزند DVDList باشند).

.

مشخص کننده ی گره ی جاری است.

..

مشخص کننده ی والد گره ی جاری است.مثلا اگر گره ی جاری Title باشد .. نشان دهنده ی DVD خواهد بود.

[]

علامت تعیین محدوده ی Select است.مثلا:

```
/DVDList/DVD[Title='Original Sin’]
```

تمام تگ های DVD که تگ Title آنها حاوی Original Sin Value است را به ما می دهد.ویا:

```
/DVDList/DVD[@Id=’234’]
```

تمام تگ های DVD که ویژگی Id در آنها برابر 234 است را به ما می دهد.

Starts-With

با استفاده از آن می توانید تگ هایی را انتخاب کنید که در Value آنها محدودیت قایل شدید.مثلا:

```
/DVDList/DVD[Starts-With(Title,’P’)]
```

تمام تگ های DVD را انتخاب کن که تگ Titleشان حاوی مقادیری باشد که با حرف P شروع شده.

Position

مشخص کننده ی وضعیت تگ انتخاب شده است.مثلا:

```
/DVDList/DVD[position()=2]
```

انتخاب کن فرزند دوم تگ DVDList را با نام DVD.در حقیقت دومین تگ DVD را از بین فرزند هایDVDList انتخاب می کند.در اینجا نیازی به قرار دادن عدد در ” نیست و ’2‘ غلط است.

Count

تعداد یک تگ را بر می گرداند مثلا:

```
Count(DVD)
```

تعداد تگ هایی با نام DVD را بر می گرداند.

Synatx ها در XPath بسیارند و در عین حال ساده.مثلا شما از and یا or هم می توانید استفاده کنید به عنوان مثال:

```
/DVDList/DVD[position()=2 and @Id=’234’]
```

دومین تگ DVD را از بین فرزند هایDVDList انتخاب کن اگر ویژگی id در آن برابر 234 باشد.

برای آشنایی کامل با XPath می توانید به کتب منتشر شده در این زمینه رجوع کنید ولی به نظر من یک مرجع جیبی خیلی بهتر است.البته ما در بحث XSLT باز هم در مورد XPath صحبت خواهیم کرد.

همانطور که می بینید با استفاده از علایم بالا می توانید پرس و جو های پیچیده ای داشته باشید.تا قبل از این یک مثال حل کردیم که نام بازیگران فیلم مشخصی را برای ما برمی گرداند حال آن مثال را با XPath حل می کنیم.برای استفاده از Syntax های XPath باید از متد SelectNodes شی XmlDocument استفاده کنید که ما پیش از این از متد GetElementsByTagName استفاده می کردیم.مقدار برگشتی متد SelectNodes هم از نوع XmlNodeList است.آرگومان ورودی آن هم Synatx XPath خواهد بود:

```
    Private Function searchByTagName() As String
        Dim xmlPath As String = Server.MapPath("myxml.xml")
        Dim doc As New XmlDocument()
        doc.Load(xmlPath)
        Dim nodelist As XmlNodeList =
doc.SelectNodes("/DVDList/DVD[Title='Basic Instinct']/Starring/Star")
        Dim str As String = String.Empty
        For Each node As XmlNode In nodelist
            str += "Found : <br/>"
            Dim name As String = node.ChildNodes(0).Value
            str += name
            str += "<br/>"
        Next
        Return str
    End Function
```

می بینید که تمام این تابع تکراری بود به جز متد SelectNodes.

ما برای بازیابی بازیگران فیلمی خاص (در اینجا با نام Basic Instinct) در ابتدا عبارت زیر را تولید کردیم که تگ DVD مربوط به فیلم Basic Instinct را به ما بر می گرداند:

/DVDList/DVD[Title='Basic Instinct']

حال که تگ DVD مربوطه را داریم باید در آن به دنبال تگ Satr بگردیم که آن هم در داخل Starring است:

/Starring/Star

به همین سادگی می توانید Query های پیچیده تری هم بسازید و حسابی در داخل سند xml به دنبال مقادیر دلخواه خود بگردید.جلوتر وقتی با XSL آشنا شدید به ارزش بسیار بالای XPath پی می برید.

خوب بحث جستجو به پایا ن رسید و حال باید به آخرین روش خواندن سند xml بپردازیم.

:XmlTextReader

با استفاده از این شی هم می توان عمل خواندن را انجام داد.برای استفاده از این شی باید آن را New کرد و به آرگومان ورودیش مسیر فایل xml مورد نظر را داد.برای پیمایش و خواندن گره ها هم از متد Read شی reader در حلقه ی While استفاده می کنیم در حقیقت متد Read در هر بار چرخش حلقه یWhileبه گره ی بعد می رود و شرایط را برای خواندن محیا می کند .در Select Case هم از متد NodeType خود XmlTextReader استفاده می کنیم.خوبی این روش این است که در اینجا از تابع بازگشتی خبری نیست چون خود Do While Reader.Read عمل پیمایش را به درستی انجام می دهد.در حقیقت شی reader در هر لحظه یک گره را بررسی می کند و با متد read به گره ی بعدی می رود ولی در مثال های قبل پیمایش ها دستی بود زیرا متد های مجود برای دو روش XPathNavigation و XmlDocument تنها برای یک فرزند برای هر گره ظرفیت داشتند مثلا متد ChildNodes تنها به فرزند هر گره دسترسی داشت و دیگر به اینکه آیا آن گره و یا فرزندانش هم فرزندی دارند کاری نداشت و ما مجبور بودیم برای هر گره از نوع تگ یک بار تابع را صدا بزنیم ولی در اینجا شی XmlTextReader کل گره ها را یک جا از نوع stream در اختیار دارد و می تواند با یک حلقه و متد ()read روی آنها پیمایش لازم را انجام دهد :

```
Public Sub ReadXML()
        Dim xmlFile As String = Server.MapPath("myxml.xml")
        Dim reader As New XmlTextReader(xmlFile)
        Dim str As String = String.Empty
```

```
        Do While reader.Read()
            Select Case reader.NodeType
                Case XmlNodeType.XmlDeclaration
                    str += "XML Declaration:  "
                    str += reader.Name
                    str += " "
                    str += reader.Value
                    str += "<br/>"

                Case XmlNodeType.Element
                    str += "Element:  "
                    str += reader.Name
                    str += "<br/>"

                Case XmlNodeType.Text
                    str += " - Value:  "
                    str += reader.Value
                    str += "<br/>"

                Case XmlNodeType.Comment
                    str += "Comment:  "
                    str += reader.Value
                    str += "<br/>"
            End Select
            If reader.AttributeCount > 0 Then
                Do While reader.MoveToNextAttribute()
                    str += " - Attribute: "
                    str += reader.Name
                    str += "  Value:  "
                    str += reader.Value
                    str += "<br/>"
                Loop
            End If
        Loop
        reader.Close()
        Label1.Text = str
    End Sub
```

در اینجا هم مانند دو روش قبل Attribute را جدا چک کردیم ولی این بار با متد AttributeCount.چون XmlTextReader متدی مثل HasAttribute ندارد.پس می گوییم اگر تعداد ویژگی هایش بزرگتر از ۰ باشد وارد شرط If شو(این یعنی اگر ویژگی دارد وارد شرط شو).برای پیمایش ویژگی ها هم از متد MoveToNextAttrinute شی XmlTextReader استفاده کردیم.همانطور که قبلا گفته بودیم تنها در استفاده از XPathNavigation ملزم هستید از متد MoveToFirstAttribute استفاده کنید چون

XPathNavigation در هر بار یگ گره را در خود حمل می کند و به این مورد بسیار حساس است ولی XmlTextReader با اینکه دارای متد MoveToFirstAttribute هست ولی نباید از آن استفاده کنید زیرا اصلا نیازی به این کار نیست و خود به خود این کار را انجام می دهد و حتی پس از عمل Move خودش به Parent باز می گردد نیاز نیست ما مثل XPathNavigation دستی آن را به والدش باز گردانیم.در حقیقت طرز کار این حلقه برای تگی مثل DVD که حاوی ۲ ویژگی است به صورت زیر است.

در ابتدا چک می شود گره ی جاری که تگ DVD است اگر دارای ویژگی بود وارد شرط شو.سپس به یک حلقه ی Do While بر می خوریم.در این حلقه ابتدا چک می شود آیا تگ مورد نظر ویژگی بعدی دارد یا نه.در اولین گام MoveToNextAttrinute با توجه به وجود ویژگی Idمقدار True برمی گرداند زیرا در اولین گام NextAttrinute همان اولین ویژگی است.پس وارد حلقه می شویم و نام و مقدار ویژگی Id را به str اضافه می کنیم.در گام بعدی هم ویژگی بعدی چک شده و Category به str اضافه می شود.اگر دقت کنید می بینید که ما درروش قبلی از حلقه ی Do...Loop While استفاده کردیم.در آن حلقه اگر شرط درست هم نبود حداقل یک بار وارد حلقه می شدیم زیرا از متد MoveToFirstAttribute استفاده کرده بودیم(و باید استفاده می کردیم) ولی در اینجا چون از این متد استفاده نکردیم (نباید هم استفاده کنیم)اگر حلقه ی ما به صورت Do...Loop While می بود آنگاه خود تگ DVD یک ویژگی محسوب می شد و به عنوان ویژگی به str اضافه می شد که کار اشتباهی بود.

پس خوبی این روش این بود که از تابع بازگشتی خبری نبود زیرا یک تابع بازگشتی درجه ی رشد برنامه ی ما را بسیار افزایش می دهد و باعث کاهش کارایی آن می شود.هنوز کار ما با XmlTextReader تمام نشده و می خواهیم عمل خواندن را با آن کمی بهینه تر کنیم.برای این کار عمل خواندن را به گونه ای دیگر انجام می دهیم.

متد های زیر از شی XmlTextReader را به خاطر بسپارید:

ReadStartElement:خواندن تگی که حاوی فرزند است.آرگومان ورودیش نام تگ است.

ReadElementString:خواندن Textتگی که حاوی فرزند نیست.آرگومان ورودیش نام تگ است.

Name:نام ویژگی است.

Value:مقدار ویژگی است.

MoveToContent:چک می کند که گره ی مورد نظر حاوی تگ هست یا نه.اگر بود واردش می شود و اگر نبود(یا فضای خالی و یا اعلان xml بود) آن را رد کرده و به محتوی گره ی بعدی می رود.اگر هم نوع گره Attribute باشد این متد به تگی که این ویژگی را داراست بر می گردد.

MoveToAttribute:وارد ویژگی می شود و باعث می شود بتوانیم به آن دسترسی پیدا کنیم.آرگومان ورودیش هم نام ویژگی است.

GetAttribute:با گرفتن نام ویژگی به عنوان آرگومان ورودی مقدار آن را برمی گرداند.

حال که با برخی از متد ها ی شی XmlTextReader آشنا شدید می خواهیم تابعی بنویسیم تا با آن بتوان مقادیر یک سند xml را بخوانیم.در ابتدا تابع را ایجاد می کنیم:

```
Public Sub readXML()

End Sub
```

سپس شی XmlTextReader را New می کنیم:

```
Dim xmlFile As String = Server.MapPath("myxml2.xml")
Dim reader As New XmlTextReader(xmlFile)
```

مانند مثال قبل از یک حلقه ی Do While برای پیمایش گره ها استفاده می کنیم:

```
Do While reader.Read

Loop
```

برای اینکه حلقه ی Do While در هر بار چرخش , وارد فرزندان یک تگ DVD شود و آنقدر این کار را ادامه دهد تا تگ های DVD به پایان برسد لازم است ابتدا مشخص کنیم که حلقه ی فوق تنها باید تگ های فرزند تگ والد DVD را مورد بررسی قرار دهد.پس قبل از نوشتن حلقه ی Do While تگ ریشه را مشخص کنید:

```
reader.ReadStartElement("DVDList")
```

حال که مبدا ما تگ های فرزند DVDList(DVD ها) هستند و حلقه روی تک تک آنها در هر بار چرخش , پیمایش انجام می دهد می توانیم کار را شروع کنیم و پس از مشخص کردن هر یک با متد ,ReadStartElement فرزندان آن را بخوانیم و در رشته ی str بنویسیم:

```
reader.ReadStartElement("DVD")
```

```
    str += "<br/>"
    str += reader.ReadElementString("Title")
    str += "<br/>"
    str += reader.ReadElementString("Director")
    str += "<br/>"
    str += "$" & reader.ReadElementString("Price")
str += "<br/>"
```

در ابتدا با متد ReadStartElement تگ هدف را مشخص کردیم و با متد ReadElementString محتوی Text تگ های فرزند DVD را که مایلید خوانده شود را مشخص کریم.پس این نکته مهم است که بدانید هر بار به تگی که حاوی مقدار نیست و فرزند دارد رسیدیم از ReadStartElement استفاده می کنیم و هر گاه به تگی که حاوی مقدار بود رسیدیم از ReadElementString استفاده می کنیم.مثلا برای خواندن محتویات تگ Star به صورت زیر عمل می کنیم:

```
    reader.ReadStartElement("Starring")
    str += reader.ReadElementString("Star")
    str += "<br/>"
    str += reader.ReadElementString("Star")
    str += "<br/>"
```

به جای اینکه کد های بالا را در داخل حلقه ی Do While بنویسید باید آنها را جهت اطمینان بیشتر در داخل شرط زیر بنویسید:

```
If (reader.Name = "DVD") AndAlso (reader.NodeType = XmlNodeType.Element)
Then
End If
```

این شرط به ما می گوید اگر نام گره ی مورد پیمایش DVD بود و اگر از نوع تگ بود وارد شرط شو.در حقیقت کاربرد این شرط در اسناد xml ناموزون است.مثلا سند خلاصه ی زیر را در نظر بگیرید:

```
<DVDList>
 <DVD
   ...</DVD>
 <DVD
   ...</DVD>
 <DVD
   ...</DVD>
 <MVP>
 ...</MVP>
</DVDList>
```

اگر مثال بالا با سندی مثل سند بالا حل شود و شرط if‌ی که در بالا نوشتیم در خواندن سند ذکر نشود به مشکل برمی خوریم زیرا طبق کد هایی که در بالا نوشتیم که خواندن را بر اساس تگ های DVD و Title و director و Price را انجام می دهد در هر بار چرخش حلقه ی Do While عناصر یک تگ DVD خوانده می شود و در انتها وقتی به تگ MVP رسیدیم باز هم عمل خواندن از تگ های DVD و Title و director و Price انجام می گیرد در حالیکه این تگ ها جزو فرزندان تگ MVP نیستند و در حقیقت اصلا وجود ندارند.پس به مشکل برمی خوریم و آن هم خواندن چیزی است که وجود ندارد.پس باید در هر بار چرخش حلقه ی Do While که عناصر یک تگ فرزند تگ ریشه مورد بررسی و نمایش قرار می گیرد آن فرزند را چک کنیم.اول اینکه نامش DVD باشد(که اگر مثلا MVP بود وارد شرط نشویم و کد هایی که در زیر بیان می شود برایش اجرا نشود) و دوما اینکه گره ی مورد نظر از نوع تگ باشد.

حتما می پرسد چگونه می توان ویژگی ها را بازیابی کرد.نکته ی مهم در اینجا این است که عمل بازیابی ویژگی برای هر تگ, قبل از تگ ReadStartElement صورت می گیرد نه در داخل آن.برای این کار سه روش وجود دارد در روش اول از متد MoveToNextAttribute استفاده می کنیم:

```
  str += "<br/>"
  reader.MoveToNextAttribute()
  str += reader.Name & " : " & reader.Value
  str += "<br/>"
  reader.MoveToNextAttribute()
  str += reader.Name & " : " & reader.Value
  reader.ReadStartElement("DVD")
  str += "<br/>"
  str += reader.ReadElementString("Title")
  str += "<br/>"
  str += reader.ReadElementString("Director")
  str += "<br/>"
  str += "$" & reader.ReadElementString("Price")
str += "<br/>"
```

به این صورت که با متد MoveToNextAttribute در اولین گام وارد اولین ویژگی شده و با متد های name & Value نام و مقدار آن را چاپ کردیم.با به کاربردن دوباره ی متد MoveToNextAttribute وارد ویژگی می شویم.

روش بعدی استفاده از MoveToContent است.روش آن به این صورت است که با متد MoveToContent نوع گره را تشخیص می دهیم.همانطور که گفتیم اگر از نوع ویژگی باشد به تگی که آن ویژگی را داراست برمی گردیم و سپس با متد MoveToAttribute که مخصوص تگی است که ویژگی را داراست به ویژگی مورد نظر دسترسی خواهیم داشت.:

```
  شی خواننده در حال حاضر روی گره ای است که حاوی ویژگی است
  reader.MoveToContent()
  حال شی خواننده روی تگی است که حاوی ویژگی است
  reader.MoveToAttribute("ID")
  حال شی خواننده روی خود ویژگی است و آماده ی خواندن مقدار آن است
  str += reader.Name & " : " & reader.Value
str += "<br/>"
```

اگر هم MoveToContent گره را تگ تشخیص دهد به گره ی بعدی می رود آنقدر می رود تا به ویژگی برسد و سپس به گره ای که آن ویژگی را داراست برسد.بدی این روش این است که باید حتما نام دقیق ویژگی را به عنوان آرگومان ورودی به متد MoveToAttribute بدهیم.در اینجا نیازی نیست پس از خواندن ویژگی به گره ی والد برگردیم (در مثال ما متد ReadStartElement پس از خواندن ویژگی های تگ DVD برای خواندن محتویات تگ Starring به کار برده می شود).

روش سوم برای بازیابی ویژگی استفاده از متد GetAttribute به جای MoveToAttribute است:

```
   reader.MoveToContent()
str += "ID : " & reader.GetAttribute("ID")
```

متد GetAttribute برای بدست آوردن مقدار یک ویژگی دیگر شی reader راوارد خود ویژگی نمی کند (بر خلاف متد MoveToAttribute که شی reader را وارد خود ویژگی می کرد)و سپس عمل خواندن را به reader می سپرد چون پس از به کار بردن متد MoveToAttribute شی reader وارد ویژگی می شود.از طرف دیگر متد GetAttribute چون شی reader را وارد ویژگی نمی کند به نام ویژگی دسترسی ندارد.

در مواردی که تا به حال از XmlTextReader استفاده کردید حتما در انتها باید آن را با متد ()Close ببندید.خوب با ۳ شیوه ی کلی خواندن سند xml و همینطور نحوه ی

جستجو در آن آشنا شدید.حال اشاره ی کوتاهی به Validating Xml برای فایل های xsd خواهیم داشت.

Validating Xml File:

تا اینجا انواع روش برای خواندن و جستجو در اسناد xml را آموختید.ولی تمام آنها تا وقتی سند ما خوش فرم نباشد ارزشی ندارد.همانطور که می دانید با xsd می توانیم یک سند xml را قانونمند کنیم و یک ساختار مشخصی به آن ببخشیم. ما با xsd زیاد کار کردیم و نحوه ی اعمال آن به سندxml را آموختیم و حالا می خواهیم با اشیا و متد های مربوطه در asp.NET نحوه ی Validate یک سند xml را توسطxsd یاد بگیریم.برای این کار با یک سری توابع و متد ها می خواهیم Valid بودن یا نبودن یک سند xml را بر اساس xsd مربوطه تشخیص دهیم.در ابتدا یک xsd برای سند Xml DVDList ایجاد می کنیم و نامش را xsdFormyxml می گذاریم:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="DVDList" >
          <xsd:complexType>
                <xsd:sequence maxOccurs="unbounded">
                      <xsd:element name="DVD" type="DVDType">
                      </xsd:element>
                </xsd:sequence>
          </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="DVDType">
          <xsd:sequence>
                <xsd:element name="Title" type="xsd:string" />
                <xsd:element name="Director" type="xsd:string"  />
                <xsd:element name="Price" type="xsd:decimal"  />
                <xsd:element name="Starring" type="StarringType" />
          </xsd:sequence>
          <xsd:attribute name="ID" type="xsd:integer" />
          <xsd:attribute name="Category" type="xsd:string" />
    </xsd:complexType>

    <xsd:complexType name="StarringType">
          <xsd:sequence maxOccurs="unbounded">
                <xsd:element name="Star" type="xsd:string"/>
          </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

برای تشخیص Valid بودن یا نبودن سند myxml(همان لیست DVD ها) دو سند ایجاد می کنیم یکی Valid و دیگری Invalid تا برنامه تشخیص دهد کدامیک Valid و کدام یک Invalid است.

سند Valid که پیش تر ذکر شده بود (همان لیست DVDها) ولی سندی Invalid ایجاد می کنیم به نام myxml1 که تنها تفاوتی که با myxml دارد این است که ویژگی Id یکی از تگ های DVD در آن به جای نوع داده ای عددی از نوع رشته ای است با علم به خط ۱۸ از سند XSD بالا که نوع داده ای آن را Integer قرار داده بودیم:

```
<?xml version="1.0" encoding="utf-8" ?>
<DVDList>
    <DVD ID="hggt" Category="Science Fiction">
          <Title>The Matrix</Title>
          <Director>Larry Wachowski</Director>
          <Price>18.74</Price>
          <Starring>
                <Star>Keanu Reeves</Star>
                <Star>Laurence Fishburne</Star>
          </Starring>
</DVD>
...
```

پس همین یک خطا محسوب می شود.

برای درک بهتر این مثال صفحه ای به فرم زیر طراحی کنید :

```
    <form id="form1" runat="server">
        <asp:RadioButton          ID="RadioButton1"          runat="server"
GroupName="validation" Text="Use Valid XML" /><br />
        <asp:RadioButton          ID="RadioButton2"          runat="server"
GroupName="validation" Text="Use Invalid XML" /><br />
        <asp:Button     ID="Button1"     runat="server"     Text="Validate"
Width="123px" /><br />
        <asp:Label ID="Label1" runat="server"></asp:Label><br />
    </form>
```

ما دو RadioButton در صفحه قرار دادیم و یک دکمه و یک Label.کاربر باید یکی از دو RadioButton را انتخاب کند و سپس روی دکمه کلیک کند.RadioButton اول که Textش Use Valid XML است قرار است به myxml اشاره کند(که Valid است) و RadioButton. دوم که Textش Use InValid XML است قرار است به myxml۱ اشاره کند(که InValid است).سپس با فشردن دکمه پیغام مربوطه برایش در Label نمایش داده شود به این ترتیب که در صورتی که کاربر RadioButton اول را انتخاب و

سپس روی دکمه کلیک کند پیغام The Xml File Is Valid برایش درLabel ظاهر شود ولی اگر کاربر RadioButton دوم را انتخاب و سپس روی دکمه کلیک کند پیغامی خصوص خطای مربوطه برایش در Label ظاهر شود.به این نکته توجه کنید که وقتی دو RadioButton جداگانه روی صفحه قرار می دهید همزمان می توان هر دو را انتخاب کرد و این یک ایراد است و برای جلوگیری از آن از صفت GroupName استفاده می کنیم.به این صورت که برای هر یک از RadioButton ها گروه مشخص می کنیم اگر هر دو عضو یک گروه باشند در یک لحظه تنها یکی از آنها را می توان انتخاب کرد.

برای عملی کردن توضیحات بالا به رویداد کلیک دکمه بروید:

```
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click

    End Sub
```

در ابتدا متغیری برای تعیین مسیر فایل ایجاد می کنیم:

```
Dim filepath As String = String.Empty
```

سپس با توجه به RadioButton انتخاب شده مسیر فایل را مشخص می کنیم.به این ترتیب که اگر RadioButton اول انتخاب شد مسیر فایل myxml باشد ولی اگر RadioButton دوم انتخاب شد مسیر فایل myxml1 باشد:

```
   If RadioButton1.Checked Then
      filepath = Server.MapPath("myxml.xml")
   ElseIf RadioButton2.Checked Then
      filepath = Server.MapPath("myxml1.xml")
 End If
```

از متغیر filepath سر جایش استفاده خواهد شد.

حال به اصل کار می رسیم.چون با Xml Schema کار می کنیم حتما باید فضای نام System.Xml.Schema علاوه بر System.Xml Imports شود.

حالا به شی نیاز داریم که بتوان فایل xsd مربوطه را در آن قرار داد.شی که می تواند حاوی چنین فایلی باشد شی XmlSchemaSet است که باید new شود و حتما هم () در انتهایش بیاید چون این یک سازنده ی جدید در NET 2.0. است:

```
Dim sch As New XmlSchemaSet()
```

برای دادن فایل xsd مربوطه به شی XmlSchemaSet کافیست از متد Add استفاده کنیم.این متد ۲ آرگومان دارد که اولی فضای نام و دومی مسیر فایل است.ما در اینجا از

فضای نام در فایل xsd استفاده نکردیم(منظور چیزی غیر از فضای نام پیش فرض موجود در اسناد xsd است) پس می توانیم به جایش از nothing استفاده کنیم:

```
sch.Add(Nothing, Server.MapPath("xsdFORmyxml.xsd"))
```

حال که Schema تعریف شد می رسیم به استفاده از آن.برای این کار باید از شی XmlReader استفاده کنیم.این شی با سرعت بالا و بدون توجه به موارد حاشیه ای یک سند xml را می خواند:

```
Dim xr As XmlReader
```

این شی برای خواندن حتما باید با متد Create به فایل مربوطه دسترسی داشته باشد.در زیر مثالی واضح از چگونگی خواندن تگ ها ی سند xml با شی XmlReader را می بینید:

```
        Dim reader As XmlReader = XmlReader.Create(Server.MapPath("myxml.xml"))
        Do While reader.Read
            Response.Write(reader.Name & "<br/>")
        Loop
```

متد Create سازنده های مختلفی دارد. ولی برای اعمال یک Schema به سند xml حتما باید از سازنده ی زیر استفاده کنیم:

```
XmlReader.Create(xml file path as string,xsd setting as xmlreadersettings)
```

آرگومان اول که مسیر سند xml است ولی آرگومان دوم از نوع XmlReaderSettings است که شی حاوی تنظیماتی مربوط به Schema است و در حقیقت پیکربندی شی XmlReader را برعهده دارد.پس قبل از اینکه از شی XmlReader استفاده کنیم باید شی XmlReaderSettings را ایجاد کنیم:

```
Dim readerset As New XmlReaderSettings()
```

از متد ValidationType شی XmlReaderSettings, گزینه ی Validation.Schema را انتخاب کنید زیرا ما می خواهیم عمل ارزشیابی را برای فایل xsd انجام دهیم با علم اینکه استاندارد های قدیمی تری مثل DTD نیز وجود دارند.پس نوع ارزشیابی را با متد ValidationType تعیین می کنیم:

```
readerset.ValidationType = ValidationType.Schema
```

پس از آن باید از متد Schemas برای دادن شی XmlSchemaSet استفاده کنیم.پس ما شی XmlSchemaSet که تعریف کرده بودیم و فایل xsd مربوطه را به آن Add کرده

بودیم را به متد Schemas از شی XmlReaderSettings سپردیم.پس از شی XmlSchemaSet که قبلا تعریف کرده بودیم در اینجااستفاده می کنیم:

```
readerset.Schemas = sch
```

حالا پیکربندی و تنظیمات لازم با شی XmlReaderSettings انجام شد نوبت به تعریف شی XmlReader و استفاده از متد Create آن به صورتی که در بالا اشاره شد می رسد:

```
Dim xr As XmlReader = XmlReader.Create(filepath, readerset)
```

حالا برای شروع عملیات انطباق کافیست حلقه ی زیر را اجرا کنیم:

```
Do While xr.Read
Loop
```

در این حلقه عمل انطباق انجام می شود.حال از یک بلوک Try Catch برای نشان دادن خطا مبنی بر Invalid بودن یک سند استفاده می کنیم و آن را در Label نمایش می دهیم.شکل کلی مثال در رویداد کلیک دکمه در زیر آمده:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim filepath As String = String.Empty
        If RadioButton1.Checked Then
            filepath = Server.MapPath("myxml.xml")
        ElseIf RadioButton2.Checked Then
            filepath = Server.MapPath("myxml1.xml")
        End If
        Dim sch As New XmlSchemaSet()
        sch.Add(Nothing, Server.MapPath("xsdFORmyxml.xsd"))
        Dim readerset As New XmlReaderSettings()
        readerset.ValidationType = ValidationType.Schema
        readerset.Schemas = sch
        Dim xr As XmlReader = XmlReader.Create(filepath, readerset)
        Try
            Do While xr.Read
            Loop
            Label1.Text = "the xml file is Valid"
        Catch ex As Exception
            Label1.Text = ex.Message
        End Try
        xr.Close()
    End Sub
```

پس دیدید که چقدر ساده بود جمع بندی به فرم زیر می شود:

۱-تعریف Schema

۲-تعریف XmlReader و متد Create آن جهت انطباق

۳-تعریف XmlReaderSettings جهت پیکر بندی و تنظیم XmlReader برای انطباق

٤-اضافه کردن Schemaی تعریف شده به XmlReaderSettings

٥-اعمال کلی با یک حلقه ی Do While ساده.

حال که با Validating سند xml بر اساس یک فایل xsd آشنا شدید به سراغ یک بحث فوق العاده مهم به نام انتقال xml به Html به وسیله ی XSLT می رویم.ولی قبل از آن باید بفهمیم XSLT چیست.

XSLT

تا بحال فکر کرده اید اسناد Xml که تا بحال نوشته اید به چه دردی می خورند.شاید بگویید با آن می توان یک پایگاه داده ایجاد کرد.بله ولی تا وقتی پایگاه های داده ی گسترده ای مثل sql و یا DB2 هستند آیا xml مورد استفاده قرار می گیرد؟پس چرا هر چه پیش می رویم استفاده از xml بیشتر می شود؟دلیلش همانی است که گفتیم ولی به شیوه ای دیگر.یعنی از اسناد xml به عنوان پایگاه داده استفاده می شود.پایگاه داده ای که بتوان داده ها را به سرعت و دقت از آن خواند و در ان نوشت.بسیاری از سایت های خبری دنیا خبر های خود را در پایگاه داده ی xml ذخیره می کنند و کاربران برای خواندن آن اخبار عمل بازیابی را از پایگاه داده ی xml انجام می دهدند.خوبی این کار قطع ارتباط با پایگاه های داده ی سنگینی چون sql است.ولی چرا xml؟ اگر نگاهی به بخش اخبار سایت ChelseaFC بیندازید می بینید تمام صفحاتش از نوع html هستند.نه aspx ونه PHP.یعنی صفحاتی هستند ایستا ولی با محتوای پویا!.شاید تعجب کنید ولی اگر درک کاملی از ایستا و پویا داشته باشید می بینید که صفحات ایستا بسیار زودتر از صفحات پویا (مخصوصا آنهایی که با پایگاه داده سروکار دارند) بار گذاری می شوند.راز اینکه صفحه ای که مدام اخبارش به روز می شود و کلی بیننده دارد ولی فرمتش html است تناه در یک چیز است.و آن هم ارتباط بین XML و HTML.یعنی به نوعی مقادیر موجود در XML به HTML تبدیل شده و به کاربران ارایه می شود.راستی چگونه می توان عمل تبدیل را انجام داد.اصلا این یعنی چه کهXML به HTML تبدیل شود؟.پاسخ XSLT است.

XSLT یا Extensible Style Sheet Language Transform زبانی از جنس xml و برای تبدیل فایل xml به Html به کار می رود.این زبان از یک زبان بزرگتر به نام XSL مشتق می شود.همانطور که یک فایل CSS باعث فرمت یک فایل HTML می شود فایل XSL هم باعث فرمت یک سند xml می شود.XSLT مشابه یک زبان برنامه نویسی از جنس xml است که حاوی توابع و متد های زیادی است.XSLT یک زبان مبتنی بر Template است یعنی حتما باید قالب خاصی به آن بدهید تا بتوانید با آن کار کنید این قالب هم از فایل xml بدست می آید تا XSLT بتواند تطبیق داده ها را با آن انجام دهد.

این زبان یک زبان خود بازگشتی است یعنی نیازی نیست برای تمام المان های یک سند پیچیده ی xml برنامه بنویسید زیرا XSLT خودش بین تگ های همنام عمل تکرار را انجام می دهد.عمل تبدیل توسط XSLT به وسیله ی یک ساختار درختی داخلی صورت می گیرد.کاری که ما انجام می دهیم این است که یک سند xml و یک سند XSLT داشته باشیم و با برخی متد های ASP.NET عمل تبدیل به HTML را انجام دهیم.در این راه XPath کمک بسیاری به ما خواهد کرد.Xsd Xslt XPath همه از خانواده های Xml هستند.علاوه بر دستورات ASP.NET برای اجرا و تبدیل توسط XSLT می توان از پردازنده های مخلفی استفاده کرد از جمله XRay 2.0 که یک XML Editor بوده و با آن به سادگی و خارج از محیط سنگین Visual Studio می توان به فراگیری XSLT پرداخت.ما با استفاده از همان Visual Studio این کار را انجام می دهیم.ولی قبل از شروع کار بهتر است نحوه ی تبدیل را با هم ببینیم.

قبل از شروع من مسیر فایل های مورد نیاز را با server.MapPath مشخص می کنم.

در ابتدا فایل xmlی که قصد تبدیلش را دارم:

```
Dim xmlFile As String = Server.MapPath("XMLFile.xml")
```

سپس فایل XSL (فایل های xslt با پسوند XSL ذخیره می شوند) مورد نظر که وظیفه ی اصلی تبدیل بر دوش آن است:

```
Dim xslFile As String = Server.MapPath("XSLTFile3.xsl")
```

در نهایت مسیر فایل HTML را نیز مشخص می کنیم :

```
Dim htmlFile As String = Server.MapPath("result.htm")
```

اگر چه فایل وجود ندارد ولی جلوتر می بینید که خودش با توجه به مسیری که Server.MapPath ساخته است ایجاد می شود.

برای تبدیل xml به HTML از طریق XSLT از شی XslCompiledTransform استفاده می کنیم.برای دسترسی به این شی فضای نام زیر را Import کنید:

```
Imports System.xml.Xsl
```

کافی است یک شی از نوع XslCompiledTransform را New کنیم :

```
Dim transf As New XslCompiledTransform()
```

سپس از متد Load آن استفاده کرده و مسیر فایل XSL مربوطه را به آن بدهید:

```
transf.Load(xslFile)
```

عمل تبدیل یا انتقال به وسیله ی متد Transform شی XslCompiledTransform صورت می گیرد.این متد ۲ آرگومان دارد اولی مسیر فایل xml که می خواهیم تبدیلش کنیم و دومی فایل HTMLی که عمل تبدیل به آن انجام می شود:

```
transf.Transform(xmlFile, htmlFile)
```

پس تابع تبدیل به فرم زیر می شود که برای استفاده کافی است آن را در رویداد Page_Load صدا بزنید:

```
    Public Sub Transform()
        Dim xslFile As String = Server.MapPath("XSLTFile3.xsl")
        Dim xmlFile As String = Server.MapPath("XMLFile.xml")
        Dim htmlFile As String = Server.MapPath("result.htm")
        Dim transf As New XslCompiledTransform()
        transf.Load(xslFile)
        transf.Transform(xmlFile, htmlFile)
    End Sub
```

با این کار سند XMLFile.xml به وسیله ی فایل XSLTFile3.xsl به Html تبدیل شده و نتیجه ی تبدیل در فایل result.htm قرار می گیرد و برای دیدن نتیجه کافی است result.htm را اجرا کنید.پس نتیجه می گیریم متد Transform قادر است فایلی که مسیرش درآرگومان دومش ذکر شده و از قبل وجود نداشت را از نو ایجاد کند.

حال که نحوه ی تبدیل را آموختید می خواهیم به سراغ XSLT برویم.در Visual Studio از Add New Item گزینه ی XSLT File را انتخاب کنید تا سند XSLT باز شود.در صفحه ی باز شده کد های زیر به صورت پیش فرض وجود دارند:

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <html>
    <body>
```

```
        <!--
            This is an XSLT template file. Fill in this area with the
            XSL elements which will transform your XML to XHTML.
        -->
        </body>
        </html>
    </xsl:template>

    </xsl:stylesheet>
```

حال می خواهیم با اجزا و تگ های موجود در این فایل آشنا شویم.

خط اول که مشخص می کند با اینکه این یک فایل XSLT است ولی فرمت کلی XML را داراست.

خط بعدی که تگ Stylesheet است استاندارد XSLT را معرفی می کند و با دادن یک فضای نام و یک نام برای آن به نام xsl مراجعه به تگ های استاندارد فایل XSLT را امکان پذیر می کند.کل سند XSLT باید درون تگ <xsl:stylesheet> و </xsl:stylesheet> ذکر شود.

تگ xsl:template قلب تپنده ی یک فایل XSLT است.همانطور که گفتیم XSLT بر اساس template کار می کند و بدون آن هیچ است.این تگ ویژگی بسیار مهمی دارد و آن هم match است.این ویژگی المان ریشه و یا مبدا را برای کار مشخص می کند.علامت / مبتنی بر XPath بوده و نشان دهنده ی ریشه ی سند xml است.پس هر چیزی که در داخل این تگ template ذکر می شود مربوط به تگ ریشه خواهد بود.ممکن است کار شما با تگی خاص باشدکه در آن صورت به جای / نام آن تگ را به ویژگی match خواهید داد.بعد از خطوطی که توضیح داده شد تگ های html را می بینید که به شما اجازه ی کد نویسی را به زبان html البته در کنارXSLT می دهد.مثلا شاید دوست داشته باشید یک متن ایستای html را در کنار یک متن پویای XSLT قرار دهید که این امر در داخل تگ های Html و body امکان پذیر است.البته اگر دوست نداشتید از html در اینجا استفاده کنید می توانید تگ های آن را پاک کنید و تگ template را به صورت زیر درآورید:

```
<xsl:template match="/" >

</xsl:template>
```

پس تا اینجا مشخص شد برای یادگیری XSLT حتما باید با html آشنایی داشته باشید.

یک تگ بسیار مهم نیز در XSLT وجود دارد به نام تگ outPut که نوع خروجی و پیکربندی آن را مشخص می کند که سر جایش در مورد آن توضیح خواهیم داد.این تگ پس از تگ stylesheet و به فرم زیر تعریف می شود:

```
<xsl:output method="html" indent="yes"/>
```

ویژگی method مشخص می کند کد های تولید شده در چه قالبی به فایل مقصد ارسال شوند.۳ گزینه ی Xml-text-html درآن وجود دارد.اگر آن را برابر html قرار دهید و همچنین تگ های html & body در سند xsl موجود باشند در این صورت یک تگ META که مخصوص فایل های html است به فایل مقصد ما ارسال می شود.ویژگی indent هم حفظ کردن فرمت درختی اسناد مبتنی بر تگ را مشخص می کند بله یا خیر.

کد زیر یک متن ساده به نام hellow word! را در فایل html تولید شده چاپ می کند:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="html" indent="yes"/>
  <xsl:template match="/" >
  hellow word!
  </xsl:template>
</xsl:stylesheet>
```

البته ما هنوز عمل تبدیل را شروع نکرده ایم.

اولین تگی که معرفی می کنم تگ <xsl:apply-template> است.این تگ برای نمایش محتوی سند xml به کار می رود و حتما در داخل تگ template ذکر می شود.اگر این تگ را تنها استفاده کنید به صورت <xsl:apply-template/> مقدار text یا همان Valueفرزندان تگ ریشه را برای شما به نمایش در می آورد.برای مثال فایل xml زیر را با نام msg.xml در نظر بگیرید:

```
  <?xml version="1.0" encoding="utf-8" ?>
  <message>hellow word!</message>
```

فایل xsl را هم به صورت زیر بنویسید:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/" >
      <xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```

اگر با استفاده از کلاس xslCompiledTransform که در موردش بحث شد عمل تبدیل را انجام دهید یک فایل html با کد زیر ایجاد می شود:

```
<?xml version="1.0" encoding="utf-8"?>hellow word!
```

می بینید که اعلان xml در آن دیده می شود.برای حذف این اعلان یا باید فرمت خروجی را html کنید(این کار با ویژگی method از تگ output میسر است) و یا اینکه در همان تگ outPut در صورتی که "method="xml باشد گزینه ی omit-xml-declration را برابر yes قرار دهید:

```
<xsl:output method="xml" indent="yes" omit-xml-declaration="yes"/>
```

اگر چه ما در مثال بالا اصلا از تگ output استفاده نکردیم ولی با این وجود اعلان xml به بالای آن اضافه شد.

تگ <xsl:apply-template/> یک ویژگی به نام select دارد که اگر خواستید مقدار تگی خاص برایتان نمایش داده شود نام آن تگ را به ویژگیselect بدهید.به سند xml زیر توجه کنید:

```
<?xml version="1.0" encoding="utf-8" ?>
<message>
    <a>aaaaa</a>
    <b>bbbbb</b>
    <c>ccccc</c>
</message>
```

سند xsl را به صورت زیر بنویسید:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" indent="yes"/>
    <xsl:template match="message">
          <xsl:apply-templates select="b"/>
    </xsl:template>
</xsl:stylesheet>
```

اگر عمل تبدیل را انجام دهید مقداری که در تگ b بود در خروجی نمایش داده می شود.نکته ی مهم در اینجا این است که برای درست کار کردن تگ apply-templateی که ویژگی select در آن ذکر شده یا باید قسمت match تگ template باید حاوی تگ والد تگی باشد که شما مقدارش را نمایش می دهید و یا اینکه در ویژگی select آن هم نام والد(ها) و هم نام فرزند را ذکر کنید مثلا در اینجا اگر "match="/ می بود برای

نمایش محتوای تگ b مقدار ویژگیSelect باید برابر message/b باشد.در اینجا هم من می خواستم مقدار تگ b را نمایش دهم پس در قسمت match به جای / از والد تگ b یعنی message استفاده کردم.

مثال ساده ی دیگری را با هم می بینیم.سند xml به شکل زیر است:

```
  <?xml version="1.0" encoding="utf-8" ?>
<message>Mohammad Ahmadi</message>
```

و سند xsl را نیز به شکل زیر بنویسید:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="html" indent="yes"/>
  <xsl:template match="/" >
      Name:<xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```

خروجی شما فایل html با سورس زیر می شود:

```
Name:Mohammad Ahmadi
```

پس می بینید که بدون تگ های html هم توانستیم محتوای ایستا(name:) را به یک محتوای پویا (Mohammad Ahmadi) وصل کنیم.

تگ بعدی که معرفی می کنم تگ <xsl:text> است.این تگ تنها یک نوشته روی صفحه قرار می دهد.

فرمتش هم به صورت زیر است:

<xsl:text> Your Text </xsl:text>

فایل xsl بالا را به فرم زیر می نویسیم :

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="html" indent="yes"/>
  <xsl:template match="/" >
      <xsl:text>Name:</xsl:text><xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```

اگر سند msg.xml را به وسیله ی این فایل xsd تبدیل کنید خروجی مثل خروجی قبل می شود:

```
Name:Mohammad Ahmadi
```

اگر دوست داشتید WhiteSpace یا فضای خالی ایجاد کنید می توانید مثلا چند تگ text با فضای خالی بین عبارات خود بیفزایید.البته این فضای خالی تنها در سورس فایل html مقصد دیده می شود و در صورت اجرا فضای خالی دیده نمی شود.

اگر خواستید از علامت های معروف استفاده کنید به جدول زیر دقت کنید:

>gt&	>
<lt&	<
"quot&	"
'apos&	‘
&amp&	&

جایی که این عبارات را باید به کار ببرید مهم نیست فقط باید text باشد .به فایل xsl زیر دقت کنید:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="html" indent="yes"/>
      <xsl:template match="/">
            <xsl:text>your name &amp; family is: </xsl:text><xsl:apply-
templates/>
      </xsl:template>
</xsl:stylesheet>
```

این فایل را با توجه به فایل msg.xml تبدیل می کنیم و سورس html خروجی به شکل زیر می شود:

```
your name &amp; family is: Mohammad Ahmadi
```

و اگر آن را اجرا کنید به صورت زیر می شود:

```
your name & family is: Mohammad Ahmadi
```

اگر خواستید علامت & در سورس فایلhtml هم به جای & ظاهر شود صفت disable-output-escaping تگ xsl:text را برابر yes قرار دهید.به این ترتیب هم در خروجی و هم در سورس فایل html به جای & علامت & قرار می گیرد:

```
<xsl:text disable-output-escaping="no">your name &amp; family is:
</xsl:text><xsl:apply-templates/>
```

نکته های بلا نه تنها برای & بلکه برای سایر مواردی که در جدول مشخص کرده ایم صادق است.

برای اینکه سورس فایل html خروجی به شکل سایر فایل های html درآید باید از تگ های html در سند xsl استفاده کنید.به مثال زیر دقت کنید:

```
  <xsl:stylesheet version="1.0"
```

```
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="html" indent="yes"/>
        <xsl:template match="/">
              <html>
                    <head>My XSLT Convert</head>
                    <body>
                          <p>
                                <xsl:apply-templates/>
                          </p>
                    </body>
              </html>
        </xsl:template>
</xsl:stylesheet>
```

یادتان باشد هر تگی که بدون نام فضای نام در سند xsl ذکر کنید مستقیما به خروجی می رود در اینجا می بینید که تگ های غریبه به رنگ زرشکی و تگ های آشنا به رنگ سبز کم رنگ مشخص شده اند.پس تگ های غریبه ی اینجا خروجی html حضور خواهند داشت و این تگ های آشنا هستند که compile می شوند.من با این مثال در داخل فایل xsl چند تگ htnl به کار بردم از جمله P-head-html که اینها مستقیما به فایل html منتقل می شوند.جالب اینجاست که مثلا نوشته ی بین تگ head ایستا و نوشته ی بین تگ p پویا است.خروجی این فایل در اثر تبدیل سند msg.xml یک فایل html با سورس زیر می شود:

```
  <html>
    <head>
      <META  http-equiv="Content-Type"  content="text/html;  charset=utf-8">My
XSLT Convert</head>
    <body>
      <p>Mohammad Ahmadi</p>
    </body>
</html>
```

اینجاست که ویژگی indent در تگ Output نما پیدا می کند.میبنی نید که تگها به شکل دندانه دار چیده شده اند.علاوه بر آن تگ META هم به آن اضافه شده.

حالا می خواهیم css را از طریق xsl به یک فایل xml نسبت دهیم.

قبل از آن بهتر است با فرمت کلی CSS آشنا شویم.فرمت کلی به صورت زیر است:

H1 { style here}

h1 نشان دنده ی نام تگی است که قرار است style که در داخل {} تعریف می کنیم رویش اعمال شود.مثلا:

h1 { font-family:verdana; font-size:26pt}

عباراتی نظیر font-family و font-size کلماتی کلیدی هستند که بین خودشان و مقدارشان علامت : قرار می گیرد و بین هر جفت نام و مقدار علامت ; قرار می گیرد.برای استفاده نیز به صورت زیر عمل می کنیم:

<h1>hellow word!</h1>

اگر برنامه تان را اجرا کنید نوشته ی hellow word! با استیلی که برای h1 ایجاد کردید نمایش داده می شود یعنی با فونت verdana و اندازه ی ۲۴.

این نمونه ای از css text بود.ولی این به تنهایی کار نمی کند. برای استفاده ی درست از آن از تگ style استفاده می شود که یک ویژگی مهم دارد (type) که نوع استیلی که باید به صفحه اعمال شود را مشخص می کند که معمولا از نوع css است:

<style type=”text/css” Css Syntax></style>

در اینجا عبارات css را که تا بحال آموختید را به جای عبارت Css Syntax قرار دهید مثلا :

<style type=”text/css”

h1 { font-family:verdana; font-size:26pt}></style>

پس از این کار می توانید از تگ h1 جهت اعمال استیل به صفحه اقدام کنید.برای مثال یک فایل xml به فرم زیر ایجاد کنید:

```
<?xml version="1.0" encoding="utf-8" ?>
<doc mystyle="text/css">
    <css>
          h1 {font-family: sans-serif; font-size: 24pt}
          p {font-size: 16pt}
    </css>
    <title>Using Css In xml</title>
    <heading>What Using Css In Xml?</heading>
    <paragraph>
          You can use Css In Xml Document.
          first You need to Learn css syntax then
          you must create xml and xsl file and in end
          convert to html.
    </paragraph>
</doc>
```

این یک فایل ساده با یگ تگ ریشه به نام doc است که یک ویژگی به نام mystyle برایش ایجاد کردیم و مقدارش را text/css قرار دادیم.به عنوان فرزند اول تگی به نام css برایش تعریف کردیم با css syntax .البته در xml تا وقتی xsd نباشد هیچ وحدودیتی در درج نوشته ندارید .در تگ های بعدی نیز text هایی ایجاد کردم.قاعدتا در فایل xml بالا نباید مشکلی داشته باشید.قبل از شرح سند xsl ان , یک بار دیگر این نکته را بگویم که هدف ما اعمال css روی محتوای پویای xml در صفحه ی ایستای html است.ما می خواهیم تگ زیر را در فایل html خروجی تولید کنیم:

```
<"style type="text/css>
h1 {font-family: verdana; font-size: 24pt  }
      p {font-size: 16pt   }
<style/>
```

همانطور که می دانید ما هیچ یک از مقادیر تگ فوق را در html نداریم و قصد داریم آنها را از xml بازیابی کنیم سپس به نحوی درست در xsl قرار دهیم و در نهایت عمل convert را به html انجام دهیم.همینطور قصد داریم title صفحه ی html خود را نیز پویا نمایش دهیم.

حال که از اهداف ما در این مثال آگاه شدید فایل xsl را ایجاد می کنیم.

در ابتدا به دلیل اینکه از ویژگی select تگ apply-template استفاده می کنیم مقدار ویژگی match را برابر نام تگ ریشه قرار میدهیم نه /:

```
<xsl:template match="doc">
```

طبق معمول تگ html , head و body را روی صفحه قرار می دهیم.از آنجایی که می خواهیم title صفحه پویا باشد آن را از تگ title در فایل xml بازیابی می کنیم به این منظور از عبارت زیر استفاده کردیم که value تگ title از فایل xml را بازیابی می کند:

```
<xsl:apply-templates select="title"/>
```

و برای اعمال آن به صفحه ی html آن را در داخل تگ <title></title> قرار می دهیم:

```
<title><xsl:apply-templates select="title"/></title>
```

حال نوبت به اعمال Style است.برای تنوع من نوع style را به عنوان ویژگی تگ doc معرفی کردم.نحوه ی بازیابی یک ویژگی از xml در xsl به صورت {@attribute name} است.پس از این کار ویژگی type در تگ style را برابر مقدار بازیابی شده ی ویژگی mystyle از تگ doc در نظر گرفتم:

```
<style type="{@mystyle}">
```

شاید با خود بگویید به جای این همه دردسر می توانستیم یک راست عبارت زیر را بنویسیم:

```
<style type="text/css">
```

در جئاب باید بگویم اولا این با نمایش محتوای پویا در یک صفحه ی ایستا مغایرت دارد و دوما این مثالی ساده است در موارد پیچیده اگر بخواهید این کار را بکنید دچار سردرگمی میشوید.

در ادامه مقادیر موجود در تگ css در فایل xml را نیز به عنوان Css Syntax بازیابی می کنیم و در ادامه ی تگ style قرار می دهیم:

```
<style type="{@mystyle}">
    <xsl:apply-templates select="css"/>
</style>
```

برای استفاده از استیل هم نوشته ها را درون تگ هایی که به آنها استیل دادیم قرار می دهیم.نوشته ها را نیز به ترتیب از تگ های heading و paragraph می گیریم:

```
<h1><xsl:apply-templates select="heading"/></h1>
<p><xsl:apply-templates select="paragraph"/></p>
```

پس فایل کلی xsl به فرم زیر شد:

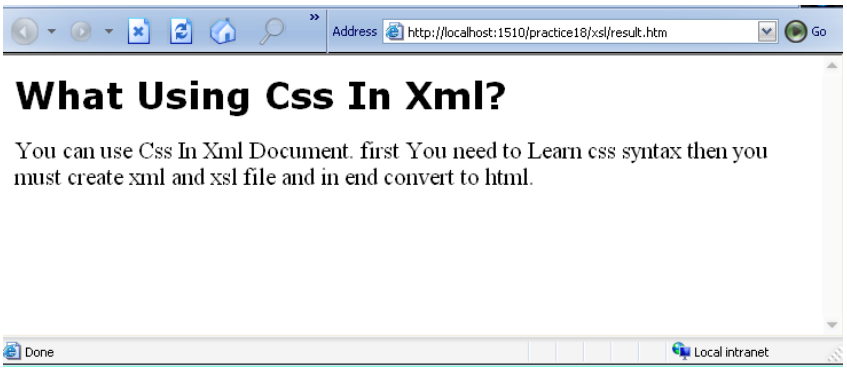
```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" indent="yes"/>
    <xsl:template match="doc">
        <html>
            <head>
                <title><xsl:apply-templates select="title"/></title>
                <style type="{@mystyle}">
                    <xsl:apply-templates select="css"/>
                </style>
            </head>
            <body>
                <h1><xsl:apply-templates select="heading"/></h1>
                <p><xsl:apply-templates select="paragraph"/></p>
            </body>
        </html>
    </xsl:template>
```

```
</xsl:stylesheet>
```

حال اگر عمل تبدیل را انجام دهید سورس فایل html خروجی به شکل زیر خواهد بود:

```
<html>
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Using Css In xml</title>
    <style type="text/css">
          h1 {font-family: verdana; font-size: 24pt}
          p {font-size: 16pt}
    </style>
  </head>
  <body>
    <h1>What Using Css In Xml?</h1>
    <p>
          You can use Css In Xml Document.
          first You need to Learn css syntax then
          you must create xml and xsl file and in end
          convert to html.
    </p>
  </body>
</html>
```

و با اجرای آن خروجی به شکل زیر می شود:



در زیر مرجعی سریع برای عبارات کلیدی Css به همراه یک توضیح ساده می بینید که به سادگی قابل ترجمه است:

background Changes the background color and image.

background-attachment Determines how background images should scroll.background-attachment supports three values:scroll, fixed, and inherit. The default value is scroll, which indicates that images scroll.Specifying fixed means that images are fixed.Specifying inherit indicates that images inherit behavior from their parent.

background-color Defines the background color.

background-image Defines the image to show in the background of the element.

background-position Defines the position of a background image.Defined as two values to specify the horizontal and vertical positioning. These can be top, center, or bottom for vertical alignment; left, center, or right for horizontal alignment; percentages; or fixed values.

background-repeat Indicates whether a background image is repeated and can be repeat, repeat-x, repeat-y, norepeat, or inherit.

border Defines the properties of the border of an element.

border-color Defines the color of a border.

border-collapse Indicates whether borders in tables collapse onto each other. Can be one of collapse, separate, or inherit.

border-spacing Defines the amount of space between borders in a table.
border-style Defines the style of a border, and can be one of none, dotted, dashed, solid, double, groove, ridge, inset, outset, or inherit.
border-top Define the properties of a single border of an element.
border-bottom
border-left
border-right
border-top-color Define the color for an individual border of an element.
border-bottom-color
border-left-color
border-right-color
border-top-style Define the style for an individual border of an element.
border-bottom-style.
border-left-style
border-right-style
border-top-width Define the width for an individual border of an element.
border-bottom-width
border-left-width
border-right-width
border-width Defines the width of the border for an element.
bottom Defines the distance to offset an element from the bottom edge of its parent element.
caption-side Defines where a caption is placed in relation to a table, and can be one of top, bottom, left, right, or inherit.
clear Clears any floating in action, and can be one of none, left, right, both, or inherit.
clip Defines how much of an element is visible and can be a rectangle rect[n,n,n,n], auto, or inherit.
color Defines the foreground color of an element.
content Defines the type of content and how it is to be displayed.
counter-increment Defines the property and amount for auto-incrementing lists.
counter-reset Resets the numerical value of auto-incrementing lists.
cursor Sets the shape of the cursor when over the element. Can be one of auto, crosshair, default, pointer, move, e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help, inherit, or a URL to an image.
direction Indicates the direction for letters, and can be one of ltr (left to right, the default), rtl (right to left), or inherit.
display Determines how to display an element, and can be one of: block, inline, list-item, none, or inherit.
empty-cells Defines how empty table cells are shown, and can be one of hide, show, or inherit.
float Indicates how this element floats within the parent element, and can be one of left, right, none, or inherit.
font Defines the attributes of the font for the element.
font-family Defines the font family for text.
font-size Defines the size of the font for text.
font-size-adjust Defines the aspect value for a font element.
font-stretch Defines how expanded or condensed a font is.
font-style Defines the style of the font, and can be one of italic, normal, oblique, or inherit.
font-variant Defines whether the font is displayed in capital letters, and can be normal, small-caps, or inherit.
font-weight Defines the thickness of the font, and can be absolute weight values (100 to 900 in steps of 100), bold or normal, bolder or lighter
height Defines the height of an element.
left When using absolute positioning, this defines howfar from the left edge of the parent the element is placed.
letter-spacing Defines the amount of space between letters.
line-height Defines the height between lines.
list-style Defines the properties for list elements.
list-style-image Defines the image to be used as the marker for a list element.
list-style-position Defines the positioning of the list relative to the list itself, and can be one of inside, outside, or inherit.
list-style-type Defines the type of marker for the list, and can be one of circle, decimal, disc, square, lower-roman, upper-roman, lower-alpha,
upper-alpha, none, or inherit.
margin Defines the amount of space between the border and the parent element.
margin-top Define the amount of space between the border
margin-bottom and the parent element for an individual side of an element.
margin-left
margin-right
marker-offset Defines the distance between the border of a list marker and the list itself.
marks Defines whether crop marks are shown, and can be one of crop, cross, both, none, or inherit.
max-height Define the maximum height and width of an element.
max-width
min-height Define the minimum height and width of an element.

min-width
orphans Defines the minimum number of lines or paragraph that must be left at the bottom of a page.
outline Defines the properties of a button or form field that has focus.
outline-color Defines the color of a button or form field that has focus.
outline-style Defines the border style of a button or form field that has focus.
outline-width Defines the border width of a button or form field that has focus.
overflow Defines the visibility of content if it doesn't fit within the element, and can be one of auto, hidden, scroll, visible, or inherit.
padding Defines the distance between one or more sides of the content area and its border.
padding-top Define the distance between a side of the content
padding-bottom area and its border.
padding-left
padding-right
page Defines the page type (such as regular or landscape) for printed content.
page-break-after Define how the browser should insert page breaks,
page-break-before and can be one of always, auto, avoid, left, right or inherit.
page-break-inside Defines whether page breaks split an element across pages, and can be auto, avoid, or inherit.
position Defines how an element is positioned relative to the flow of the document, and can be absolute, fixed, relative, static, or inherit.
quotes
right Defines the distance an element should be from its parent's right edge.
size Defines the size of the printing area in a page.
table-layout Defines how the width of table cells is calculated, and can be auto, fixed, or inherit.
text-align Defines the alignment of text, and can be one of center, justify, left, right, or inherit.
text-decoration Defines the decoration on text, and can be one of blink, line-through, none, overline, underline, or inherit.
text-indent Defines the amount of space to indent the first line of a paragraph.
text-shadow Defines the amount of shadow to apply to text.
text-transform Defines the casing to use on text, and can be one of capitalize, lowercase, none, uppercase, or inherit.
top Defines the distance between an element and its parent's top edge.
Unicode-bidi Defines Unicode text as having bidirectional characteristics. vertical-align Defines the vertical alignment of the element.
visibility Defines how, or if, the element is shown, and canbe one of collapse, hidden, visible, or inherit.
white-space Defines how white space should be treated, and can be one of normal, nowrap, pre, or inherit.
widows Defines the minimum number of lines that must be left at the top of a page. width Defines the width of the element.
word-spacing Defines the distance between words.
z-index Defines the depth of an element to allow overlapping elements.

در بخش مربوط به Theme & skin به شرح کامل تری از CSS می پردازیم.

در ادامه شما را با تگ xsl:element معرفی می کنیم.این تگ یک element یا یک تگ را ایجاد می کند.ویژگی name این تگ نام المان خروجی را مشخص می کند.برای مثال می خواهیم فایل xsl زیر را روی msg.xml به html تبدیل کنیم:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" indent="yes"/>
    <xsl:template match="/">
          <xsl:element name="mynewtag"><xsl:apply-templates/></xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

سورس html خروجی به شکل زیر خواهد بود:

```
<mynewtag>hellow word!</mynewtag>
```

می توانید به تگ ایجاد شده ویژگی هم بدهید.این کار را با تگ xsl:attribute در داخل تگ xsl:element انجام می دهیم.نام ویژگی در صفت name تگ xsl:attribute و مقدارش در text بین باز و بسته شدن تگ xsl:attribute مشخص می شود.برای مثال فایل xsl زیر را روی msg.xml به html تبدیل می کنیم:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" indent="yes"/>
    <xsl:template match="/">
        <xsl:element name="mynewtag">
            <xsl:attribute name="ID">3421</xsl:attribute>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

سورس html خروجی به شکل زیر خواهد بود:

```
<mynewtag ID="3421">hellow word!</mynewtag>
```

می توان تگ xsl:attribute را در داخل تگهای معمولی (تگ هایی مثل <html> و <body> که پیش از این با آن ها کارکردیم) نیز قرار داد یعنی ملزم نیستید حتما تگ xsl:attribute را در داخل تگ xsl:element به کار ببیرید.مثلا:

```
<mytag>
    <xsl:attribute name="ID">3511</xsl:attribute>
</mytag>
```

برای ایجاد چندین Attribute و جلوگیری از شلوغی سند xsl و همچنین نظم آن از تگ <xsl:attribute-set> استفاده می کنند.حتما هم باید یک نام برایش تعریف کنید.این کار با ویژگی name انجام می شود.در داخل این تگ می توانید هر چند بار از تگ xsl:attribute استفاده کید(هر چند تا ویژگی ایجاد کنید).این تگ در خارج از تگ xsl:template تعریف می شود.مثل یک تابع است که خارج از برنامه تعریف می شود و هر جا نیاز داشتیم می توانیم از آن استفاده کنیم پس یک حسن دیگر آن یک بار نوشتن و چندین بار استفاده است و برای جلوگیری از بالارفتن بیهوده ی حجم سند xsl بسیار مفید است.در زیر نمونه ای از تگ <xsl:attribute-set> را می بینید:

```
<xsl:attribute-set name="paragraph">
    <xsl:attribute name="priority">medium</xsl:attribute>
    <xsl:attribute name="date">2003-09-23</xsl:attribute>
    <xsl:attribute name="ID">2145</xsl:attribute>
```

```
      <xsl:attribute name="category">drama</xsl:attribute>
</xsl:attribute-set>
```

در اینجا ما یک attribute-set با نام paragraph ایجاد کردیم و در آن ٤ ویژگی تعریف کردیم.برای استفاده از آن ویژگی ها در تگ دلخواه اولا حتما باید تگ را با xsl:element تعریف کنید و سپس از ویژگی use-attribute-sets در تگ xsl:element استفاده کنید و مقدارش را همان نامی بدهید که در صفت name تگ <xsl:attribute-set> مشخص کرده بودید تا تمام ویژگی هایی که در داخل تگ <xsl:attribute-set> معرفی کرده بودید یک جا به تگ مربوطه که با xsl:element تعریف کرده بودید منتقل شود.در زیر نمونه ای از سند xsl با تعریف <xsl:attribute-set> و استفاده از آن را می بینید:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="html" indent="yes"/>
      <xsl:attribute-set name="paragraph">
            <xsl:attribute name="priority">medium</xsl:attribute>
            <xsl:attribute name="date">2003-09-23</xsl:attribute>
            <xsl:attribute name="ID">2145</xsl:attribute>
            <xsl:attribute name="category">drama</xsl:attribute>
      </xsl:attribute-set>
      <xsl:template match="/">
            <xsl:element name="mynewtag" use-attribute-sets="paragraph">
                  <xsl:apply-templates/>
            </xsl:element>
      </xsl:template>
</xsl:stylesheet>
```

سورس خروجی Html آن به صورت زیر می شود:

```
<mynewtag priority="medium" date="2003-09-23" ID="2145"
category="drama">hellow word!</mynewtag>
```

تگ بعدی که معرفی می کنیم تگ <xsl:comment> است.با این تگ می توانید در سورس html خروجی توضیح ایجاد کنید.کافی است تگ <xsl:comment> را ایجاد کنید و عبارتی را که می خواهید به عنوان توضیح بنویسید را بین باز و بسته شدن تگ بنویسید.به مثال زیر که سند msg.xml توجه کنید:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="html" indent="yes"/>
      <xsl:template match="/">
            <xsl:comment><xsl:apply-templates/></xsl:comment>
      </xsl:template>
```

```
</xsl:stylesheet>
```

در اینجا مقدار بازیابی شده از فایل xml را به عنوان توضیح قرار دادیم. سورس خروجی Html آن به صورت زیر می شود:

```
<!--hellow word!-->
```

تگ بعدی که معرفی می کنیم تگ <xsl:processing-instruction> است.کلا تگهایی به فرم زیر را درProcessing Instruction , xml می گویند:

<?name value?>

حال می خواهیم بدانیم تگ <xsl:processing-instruction> چیست.

قبل از توضیح طرز کار این تگ باید به توضیحات زیر توجه شود.

حتما می دانید که فایل هایی با پسوند CSS به طور جداگانه می توانند در سایت ما ذخیره شوند و سپس به صفحات html اعمال شوند.برای ایجاد چنین فایل هایی در Visual Studio کافیست از Add New Item گزینه ی Style Sheet را انتخاب کنید تا صفحه ی مربوطه باز شود.در آن صفحه می توانید کدهای CSS بنویسید.Syntax کدهایی هم که می نویسید مثل Syntax هایی است که پیش تر در موردش به طور مختصر صحبت کردیم.برای اعمال یک CSS خارجی به یک فایل xml می بایست از دو ویژگی بسیار مهم استفاده کنیم اولی Href است که مسیر فایل css است و دیگری type که معمولا مقدارش text/css است.ولی از این صفات کجا باید استفاده کرد؟. از آنها در فایل XSLT در تگ <xsl:processing-instruction> استفاده می کنیم.پس تگ <xsl:processing-instruction> کار اضافه کردن یک فایل css خارجی را در صفحات XSLی را بر عهده دارد که قالب کد خروجیشان xml باشد یعنی <xsl:output method=”xm;” :

```
<xsl:processing-instruction name="myStyleSheet">href="myStyleSheet.css"
type="text/css"</xsl:processing-instruction>
```

همانطور که می بینید من با نام myStyleSheet یک <xsl:processing-instruction> ایجاد کردم و Text زیر را نوشتم:

```
href="myStyleSheet.css" type="text/css"
```

href که مکان فایل css است و type هم که مشخص است.ولی این دو دراینجا ویژگی نیستند و یک نوشته ی ساده هستند.ولی چون این نوشته در داخل تگ <xsl:processing-instruction> تعریف شده هنگام تبدیل عمل اعمال css به صفحه را صورت می دهد.به این صورت که خط زیر را به بالای سند html خروجی زیر اعلان xml قرار می دهد:

<?*name Of processing-instruction element* href="Path Of Css File " type="text/css"?>

پس از ?> مقداری که در صفت name تگ <xsl:processing-instruction> ذکر کرده بودیم به عنوان نام تگ StyleSheet قرار می گیرد و Value که برای تگ <xsl:processing-instruction> مشخص کرده بودیم و حاوی دو ویژگی href & Type بود حالا به عنوان ویژگی به تگ بالا اعمال شده تا همه با هم باعث اعمال CSS به سن خروجی شوند.

این تگ در داخل تگ xsl:template مربوطه ذکر می شود و هنگامی درست کار می کند که ویژگی method تگ outPut برابر xml قرار داده شود.برای مثال یک فایل css به نام myStyleSheet ایجاد کنید و عبارت زیر را در آن بنویسید:

```
paragraph {font-size: 24pt; font-family: tahoma}
```

سپس یک فایل xml به صورت زیر ایجاد کنید:

```
  ?xml version="1.0" encoding="utf-8" ?>
<message>My Name Is Mohammad. I Want To Learning XSLT So Hard</message>
```

و در نهایت فایل XSL را به صورت زیر ایجاد کنید:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="xml" indent="yes"/>

      <xsl:template match="/">
            <xsl:processing-instruction                          name="xml-
stylesheet">href="myStyleSheet.css"          type="text/css"</xsl:processing-
instruction>
                  <xsl:element name="paragraph">
                        <xsl:apply-templates/>
                  </xsl:element>
      </xsl:template>
</xsl:stylesheet>
```

همانطور که می بینید تگ <xsl:processing-instruction> در داخل تگ xsl:template تعریف شد و به خاطر اینکه match="/" بود تمام text های فرزندان ریشه را در برمی گرفت.سپس با ایجاد یک تگی که نامش در داخل فایل Style Sheet به همراه ویژگی های مربوطه تعریف شده (در اینجا دو تگ paragraph و code) و در داخل آن Apply-Template متن را نمایش می دهیم.به این ترتیب هنگام اجرای صفحه ی html , و نمایش محتویات تگ paragraph , توسط برنامه از داخل فایل CSS

مربوطه چک می شود اگر به محتویات تگ Style paragraph اعمال شده بود آن را به خروجی هم اعمال می کند اگر نه آن را به فرمت معمولی نمایش می دهد.

خروجی html به صورت زیر می شود:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="myStyleSheet.css" type="text/css"?>
<paragraph>My Name Is Mohammad. I Want To Learning XSLT So Hard</paragraph>
```

و اگر آن را اجرا کنید نوشته به همان صورتی می شود که در css تعریف کرده بودید.عمل اعمال css به سند با حظور خط زیر صورت گرفت:

```
<?xml-stylesheet href="myStyleSheet.css" type="text/css"?>
```

این خط را تگ <xsl:processing-instruction> در سند ایجاد کرد.مقداری که برای نام تگ <xsl:processing-instruction> در نظر گرفته بودیم در اینجا به نام تگ styleSheet تبدیل شده.

نکته ی مهم در اینجا این است که حتما باید قالب خروجی xml باشد و حتما هم اعلان xml وجود داشته باشد در غیر این صورت تشخیص Styleتوسط مرورگر امکان پذیر نیست.

پس یاد گرفتید که چگونه می توانیم از یک فایل CSS خارجی در صفحه ی XSLT استفاده کنیم به طور خلاصه:

۱-تنظیم Method=”xml” .

۲-ایجاد تگ <xsl:processing-instruction> با نامی دلخواه در داخل تگ xsl:template.

۳-نوشتن متنی حاوی دو ویژگی href و type در بین باز و بسته شدن تگ <xsl:processing-instruction>.

٤-استفاده از تگی که استیلش در فایل CSS مشخص شده و نمایش text بازیابی شده از xml درونش.

برای اعمال Style Sheet به برخی تگ های خاص می توانید طبق مثال زیر عمل کنید.فایل CSS همان فایلی است که در مثال قبل ایجاد کردیم ولی فایل xml را به شکل زیر تعریف کنید:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<message>My Name Is <name>Mohammad</name> . I Want To Learning XSLT So
Hard</message>
```

همانطور که می بینید انگار ما نوشته را قطع کردیم و سپس ادامه دادیم به این عمل Mixed Content می گویند.قصد من این است که نوشته ای که در تگ name ذکر شده از سایر نوشته ها Style متفاوتی داشته باشد.بدین منظور ابتدا فایل xsl را به صورت زیر بنویسید:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="xml" indent="yes" />
      <xsl:template match="/">
            <xsl:processing-instruction                              name="xml-
stylesheet">href="myStyleSheet.css"          type="text/css"</xsl:processing-
instruction>
                  <xsl:element name="paragraph">
                        <xsl:apply-templates/>
                  </xsl:element>
      </xsl:template>

</xsl:stylesheet>
```

تا اینجا همه چیز مثل مثال قبل است و تمام متن (با اینکه در سند xml متن توسط تگ ها جدا و شکسته شده) با استیلی که برای تگ paragraph در فایل mystylesheet.css تعیین کرده بودیم نمایش داده می شود زیرا همانطور که می دانید تگ <xsl:apply-template> وقتی ویژگی select نداشته باشد مقدار Value همه تگ های فرزند ریشه(که در تگ template با متد match مشخص می شود) را نمایش می دهد و کاری ندارد به این که value مال کدام تگ فرزند است.حال اگر در ادامه کد زیر را اعمال کنیم(بیرون از تگ xsl:template و داخل xsl:stylesheet) اتفاق دیگری رخ می دهد:

```
  <xsl:template match="name">
      <xsl:element name="code">
            <xsl:apply-templates/>
      </xsl:element>
</xsl:template>
```

این کد باعث نمایش Value تگ name آن هم در داخل تگ code که در فایل mystylesheet.css به آن Style داده بودیم می شود.به نظر شما خروجیhtml چه می شود؟در این صورت ابتدا متن با فرمت قبلی که در کد بالاتر ایجاد کردیم تولید می شود و سپس متن جدید با فرمت جدید روی آن به نحوی سر جایش میکس می شود زیرا

match="name" است پس در متن موجود به دنبال تگ name می گردد و به محض یافتن آن تگ <code></code> را در خروجی , ما بین محتویات تگ name قرار می دهد.حال هر چند بار که از تگ name استفاده شده باشد این خاصیت رویش اعمال می شود.

سورس html خروجی به صورت زیر می شود:

```
  <?xml version="1.0" encoding="utf-8"?>
  <?xml-stylesheet href="myStyleSheet.css" type="text/css"?>
<paragraph>My Name Is <code>Mohammad</code> . I Want To Learning XSLT So
Hard</paragraph>
```

حال اگر template جدید را رویش اعمال نمی کردید سورس html خروجی به صورت زیر می شد:

```
  <?xml version="1.0" encoding="utf-8"?>
  <?xml-stylesheet href="myStyleSheet.css" type="text/css"?>
<paragraph>My Name Is Mohammad . I Want To Learning XSLT So
Hard</paragraph>
```

و در صفحه ی خروجی چیزی مثل نوشته ی زیر را می بینید:

My Name Is Mohammad . I Want To Learning XSLT So Hard

و یا اگر کلمه ی XSLT در جمله ی بالا را در فایل xml در تگ جدا قرار دهید به صورت زیر:

```
  <?xml version="1.0" encoding="utf-8" ?>
<message>My Name Is <name>Mohammad</name> . I Want To Learning
<learn>XSLT</learn> So Hard</message>
```

و کد زیر را به فایل XSLT اضافه کنید(بیرون از تگ xsl:template و داخل xsl:stylesheet):

```
  <xsl:template match="learn">
      <xsl:element name="code">
            <xsl:apply-templates/>
      </xsl:element>
</xsl:template>
```

سورس html به صورت زیر می شود:

```
  <?xml version="1.0" encoding="utf-8"?>
  <?xml-stylesheet href="myStyleSheet.css" type="text/css"?>
<paragraph>My Name Is <code>Mohammad</code> . I Want To Learning
<code>XSLT</code> So Hard</paragraph>
```

و اگر آن را اجرا کنید به صورت زیر خواهد بود:

My Name Is Mohammad . I Want To Learning XSLT So Hard

پس می بینید که اگر ما از تگ <xsl:processing-instruction> در داخل یک تگ template استفاده کنیم سایر تگ های template دیگر هم به آن دسترسی خواهند داشت و بر اساس مقدار match شان تگ مورد نظر (<code>) را روی عبارات اعمال می کنند.

عمل میکس کاربرد زیادی در طراحی صفحات XSLT دارد و تنها در مورد Style Sheet کاربرد ندارد.

حال تا اینجا که با برخی از تگ ها آشنا شدید می خواهیم یک مثال نسبتا جامع را با هم ببینیم.این مثال روی فایل xml زیر عمل می کند:

```
  <?xml version="1.0" encoding="utf-8" ?>
<message>My Name Is <name>Mohammad</name> . I Want To Learning XSLT So
Hard</message>
```

و فایل Css نیز به فرم زیر است:

```
  heading {display: block; font-size: 16pt; font-family: sans-serif; margin:
8pt 15pt}
  paragraph {display: block; font-size: 12pt; font-family: serif; margin: 5pt
15pt}
code {display: inline; font-size: 21pt; font-family: monospace}
```

حال می رویم به سراغ فایل XSL.

در ابتدا یک Attribute-set تعریف می کنم:

```
  <xsl:attribute-set name="atts">
      <xsl:attribute name="ID">3229</xsl:attribute>
      <xsl:attribute name="notice">true</xsl:attribute>
</xsl:attribute-set>
```

برای اعمال css هم به صورت زیر عمل می کنیم:

```
<xsl:processing-instruction name="xml-styleSheet">href="myStyleSheet.css"
type="text/css"</xsl:processing-instruction>
```

سپس یک تگ به نام doc ایجاد می کنم تا متون خود را در آن بنویسم:

```
  <doc>
</doc>
```

در ابتدا یک توضیح می نویسم:

```
<xsl:comment>This Is a XSLT Summary</xsl:comment>
```

سپس:

```
    <xsl:element name="heading" use-attribute-sets="atts" >
        <xsl:text>You Learn How To Use Different Tag in XSLT</xsl:text>
    </xsl:element>
    <xsl:element name="code" use-attribute-sets="atts">
        <xsl:text>First You Learn About Structure Of XSLT</xsl:text>
</xsl:element>
```

حال کمی کار را دینامیک می کنم و جمله ای را که در سند xml بود به یک جمله ی ایستا می چسبانم:

```
    You Are Learning Xslt So Hard &amp; Your Text is:
<xsl:apply-templates select="message"></xsl:apply-
templates>
```

و حال یک کار پویای دیگر:

```
and Your Name is : <xsl:apply-templates select="message/name"/>
```

و حالا یک تگ template جدا برای تگ name ایجاد می کنم تا هر جا تگ name دید تگ code را رویش اعمال کند:

```
    <xsl:template match="name">
        <xsl:element name="code">
              <xsl:apply-templates/>
        </xsl:element>
</xsl:template>
```

بنابراین کد بالا هر جای متن اگر نوشته ی mohammad وجود داشت آن را در داخل تگ code قرار می دهد تا استیل مربوطه رویش اعمال شود.

کد کلی مثال به شکل زیر خواهد بود:

```
    <?xml version="1.0" encoding="utf-8"?>

    <xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="xml" indent="yes"/>
        <xsl:attribute-set name="atts">
              <xsl:attribute name="ID">3229</xsl:attribute>
              <xsl:attribute name="notice">true</xsl:attribute>
        </xsl:attribute-set>

        <xsl:template match="/">
              <xsl:processing-instruction                                    name="xml-
styleSheet">href="myStyleSheet.css"          type="text/css"</xsl:processing-
instruction>
              <doc>
                    <xsl:comment>This Is a XSLT Summary</xsl:comment>
                    <xsl:element name="heading" use-attribute-sets="atts" >
                          <xsl:text>You  Learn  How  To  Use  Different  Tag  in
XSLT</xsl:text>
```

```
                    </xsl:element>
                    <xsl:element name="code" use-attribute-sets="atts">
                          <xsl:text>First You Learn About Structure Of
XSLT</xsl:text>
                    </xsl:element>
                    You Are Learning Xslt So Hard &amp; Your Text is:
                    <xsl:apply-templates select="message"></xsl:apply-
templates>
                    and Your Name is : <xsl:apply-templates
select="message/name"/>
              </doc>
        </xsl:template>
        <xsl:template match="name">
              <xsl:element name="code">
                    <xsl:apply-templates/>
              </xsl:element>
        </xsl:template>

  </xsl:stylesheet>
```

و سورس html خروجی به شکل زیر می شود:

```
  <?xml version="1.0" encoding="utf-8"?>
  <?xml-styleSheet href="myStyleSheet.css" type="text/css"?>
  <doc>
    <!--This Is a XSLT Summary-->
    <heading ID="3229" notice="true">You Learn How To Use Different Tag in
XSLT</heading>
    <code ID="3229" notice="true">First You Learn About Structure Of
XSLT</code>
                    You Are Learning Xslt So Hard &amp; Your Text is:
                    My Name Is <code>Mohammad</code> . I Want To Learning XSLT
So Hard
and Your Name is : <code>Mohammad</code></doc>
```

بنابراین شمامی توانید با XSLT سایت طراحی کنید.مثلا از تگ های مخصوص table در XSLTاستفاده کنید تا داده هایی که بازیابی می کنید به طور مرتب در صفحه ی خروجی چیده شوند.Style Sheet هم کاربرد بسیاری در صفحات شما دارد به همین خاطر در اینجا رویش بحث زیادی کردیم و دو روش برای اعمال Style Sheet را بیان کردیم که اولی مخصوص قالی داده ی html و دومی مخصوص xml بود.حال در ادامه با سایر تگ ها و قواعد در XSLT آشنا می شویم.

تگ بعدی که معرفی می کنیم تگ <xsl:value-of> است.این تگ هم مانند <xsl:apply-template> عمل می کند و Value یک تگ را برمی گرداند.و طرز کارش

هم مثل <xsl:apply-template> است.ولی تفاوتی که با <xsl:apply-template> دارد در خواندن از توابع است.ما در XSLT و XPath یک سری تابع داریم که با آنها می توانیم اعمال مختلفی را انجام بدهیم.مثلا شما اگر بخواهید مقدار یک Commnet از یک فایل xml را بخوانید می بایست از تابع comment() استفاده کنید ولی کجا؟بله در ویژگی select از تگ <xsl:value-of>.برای مثال فایل xml زیر را در نظر بگیرید:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- hellow word!-->
<message>Hellow.y name is mohammad </message>
```

سند XSL را به صورت زیر بنویسید:

```
<xsl:output method="html" indent="yes"/>
<xsl:template match="/">
    <xsl:value-of select="comment()" />
</xsl:template>
```

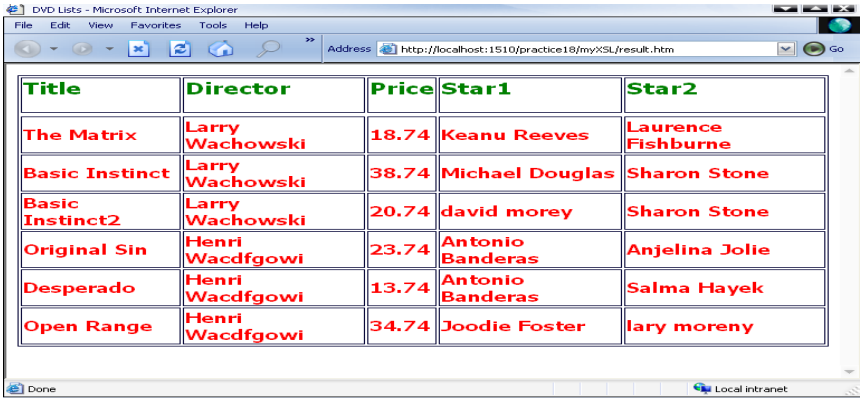
سورس Html خروجی به صورت زیر می شود:

```
hellow word!
```

کم کم با این گونه توابع بیشتر آشنا خواهیم شد.

فرق دیگری که بین <xsl:value-of> و <xsl:apply-template> وجود دارد این است که <xsl:value-of> بدون ویژگی select بی معنی است و چیزی نمایش نمی دهد و اگر <xsl:value-of> را بدون ویژگی select ایجاد کنید از شما خطا گرفته می شود.

حال می خوهیم یک مثال جامع انجام دهیم و آن هم تبدیل سند DVDList به Html به گونه ای که مقادیر آن در سطر ها ستونهای مجزا باشد.می خواهیم صفحه ی خروجی را به شکل زیر در آوریم:



Title	Director	Price	Star1	Star2
The Matrix	Larry Wachowski	18.74	Keanu Reeves	Laurence Fishburne
Basic Instinct	Larry Wachowski	38.74	Michael Douglas	Sharon Stone
Basic Instinct2	Larry Wachowski	20.74	david morey	Sharon Stone
Original Sin	Henri Wacdfgowi	23.74	Antonio Banderas	Anjelina Jolie
Desperado	Henri Wacdfgowi	13.74	Antonio Banderas	Salma Hayek
Open Range	Henri Wacdfgowi	34.74	Joodie Foster	lary moreny

به نظر کار سختی می آید ولی بسیار ساده است.تنها کافی است با مفهوم Table در html کاملا آشنا باشید.در html برای ایجاد یک Table از تگ <table> استفاده می

شود.برای ایجاد سطر از تگ <tr> و برای ایجاد ستون از تگ <td> استفاده می شود.برای مثال کد زیر یک جدول با دو سطر و ۳ ستون ایجاد می کند:

```
<table>
    <tr>
        <td>
        </td>
        <td>
        </td>
        <td>
        </td>
    </tr>
    <tr>
        <td>
        </td>
        <td>
        </td>
        <td>
        </td>
    </tr>
</table>
```

می بینید که دو سطر دارد که در هر سطر ۳ ستون است.

ما می خواهیم عناوین به صورت ایستا باشند و مقادیرشان به صورت پویا.کل کار ما حول کد زیر می چرخد:

```
<table border="1" bordercolor="#000033">
    <tr>
        <td>
            <h1> Title </h1>
        </td>
        <td>
            <h1> Director </h1>
        </td>
        <td>
            <h1> Price </h1>
        </td>
        <td>
            <h1> Star1 </h1>
        </td>
        <td>
            <h1> Star2 </h1>
        </td>
    </tr>
    <tr>
        <xsl:apply-templates/>
    </tr>
```

```
</table>
```

در این کد یک جدول با دو سطر ایجاد کردیم که سطر اول آن حاوی ۵ ستون و سطر دوم آن بدون ستون است.در سطر اول ستون هایی به نامهای Star2 Star1 Price Director Title ایجاد کردیم که نقش عناوین را برعهده دارند.تگ <h1> برای Style است.ولی در سطر دوم از تگ <xsl:apply-template> استفاده کردیم.این تگ کل Value های موجود در سند xml را نمایش می دهد.حال برای ترتیب در نمایش عناصر و جای دادن آنها در جدول از روش میکس کردن استفاده میکنیم.به این صورت که تگ ها را جدا جدا نمایش می دهیم.نحوه ی کار به این صورت است که ما می دانیم که تگ DVDList حاوی چندین تگ DVD است.و ما می خواهیم طبق شکلی که از خروجی دیدید مقادیر هر تگ DVD یک سطر را اشغال کند.پس چون قرار است یک سطر را اشغال کند باید آن را در داخل تگ <tr></tr> قرار دهیم:

```
<xsl:template match="DVD">
    <tr>
        <p>
            <xsl:apply-templates/>
        </p>
    </tr>
</xsl:template>
```

همانطور که می بینید ما یک <xsl:template> جدا تعریف کردیم و نام match آن را هم DVD قرار دادیم.به این ترتیب این تگ در مقادیری که کد بالاتر توسط <xsl:apply-template> ایجاد کرده بود دنبال تگ DVD می گردد و هر تگ DVD که پیدا می کند را در یک سطر قرار می دهد.پس تا اینجا ما به تعداد تگ های DVD +۱ (منظور عنوان است) سطر داریم ولی مقادیر هر سطر آشفته است زیرا در داخل کد بالا نیز از <xsl:apply-template> استفاده شده.با توجه به شکل خروجی نشان داده شده مقادیر موجود در هر سطر در ستون های جداگانه نمایش داده می شود.مثلا Title در یک ستون نشان داده شده.پس ما باید به ازای هر یک از مقادیری که می خواهیم نمایش دهیم یک ستون مجزا تعریف کنیم مثلا برای title به صورت زیر عمل کنیم:

```
<xsl:template match="Title">
    <td>
        <p>
            <xsl:apply-templates/>
        </p>
    </td>
</xsl:template>
```

با توجه به این کد چون می خواستیم مقادیر title را در ستون های مجزا نمایش دهیم مقدارش را در تگ <td></td> قرار دادیم.به همین ترتیب برای سایر موارد نیز عمل می کنیم.کد کلی کار با اعمال برخی موارد نظیر CSS به شکل زیر می شود:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" indent="yes"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>DVD Lists</title>
                <style type="text/css">
                    h1 {font-family: verdana; font-size: 16pt;font-
weight:bold ;color:Green}
                    p {font-family: tahoma;font-size: 14pt;font-
weight:bold;color:Red}
                </style>

            </head>
            <body>
                <table border="1" bordercolor="#000033">
                    <tr>
                        <td>
                            <h1> Title </h1>
                        </td>
                        <td>
                            <h1> Director </h1>
                        </td>
                        <td>
                            <h1> Price </h1>
                        </td>
                        <td>
                            <h1> Star1 </h1>
                        </td>
                        <td>
                            <h1> Star2 </h1>
                        </td>
                    </tr>
                    <tr>
                        <xsl:apply-templates/>
                    </tr>
                </table>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="DVD">
```

```
            <tr>
                  <p>
                        <xsl:apply-templates/>
                  </p>
            </tr>
      </xsl:template>

      <xsl:template match="Title">
            <td>
                  <p>
                  <xsl:apply-templates/>
                  </p>
            </td>
      </xsl:template>

      <xsl:template match="Director">
            <td>
                  <p>
                        <xsl:apply-templates/>
                  </p>
            </td>
      </xsl:template>

      <xsl:template match="Price">
            <td>
                  <p>
                        $ <xsl:apply-templates/>
                  </p>
            </td>
      </xsl:template>

      <xsl:template match="Star">
            <td>
                  <p>
                        <xsl:apply-templates/>
                  </p>
            </td>
      </xsl:template>

</xsl:stylesheet>
```

و سورس Html خروجی به شکل زیر می شود:

```
<html>
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>DVD Lists</title>
    <style type="text/css">
```

```
                                h1 {font-family: verdana; font-size: 16pt;font-
weight:bold ;color:Green}
                                p  {font-family:  tahoma;font-size:  14pt;font-
weight:bold;color:Red}
                            </style>
    </head>
    <body>
      <table border="1" bordercolor="#000033">
        <tr>
          <td>
            <h1> Title </h1>
          </td>
          <td>
            <h1> Director </h1>
          </td>
          <td>
            <h1> Price </h1>
          </td>
          <td>
            <h1> Star1 </h1>
          </td>
          <td>
            <h1> Star2 </h1>
          </td>
        </tr>
        <tr>
      <tr>
            <p>
       <td>
              <p>The Matrix</p>
            </td>
       <td>
              <p>Larry Wachowski</p>
            </td>
       <td>
              <p>$ 18.74</p>
            </td>

             <td>
              <p>Keanu Reeves</p>
            </td>
             <td>
              <p>Laurence Fishburne</p>
            </td>
  ...
  </tr>
    </table>
```

```
    </body>
</html>
```

به دلیل طولانی بودن کد ,ما نام یک فیلم را در آن قرار دادیم.پس به این ترتیب توانستیم محتوای پویا را در یک صفحه ی html قرار دهیم.حال هر چند تا تگ DVD که داشته باشیم نیازی به دستکاری سند XSL نیست و فایل Html خودش آنها را نمایش می دهد.جلوتر که با تغیرات در فایل xml آشنا شدید می بینید که می توانید به سند xml مقادیری را حذف و اضافه کنید و آنها را در فایل Html نمایش دهید.مهمترین کاربرد این مثال در سایت های خبری و درج اخبار است.که به جای پایگاه داده ی SQL از یک فایل xml و تبدیل آن به html برای بهبود بسیار زیاد کارایی سایت استفاده می کنند.یعنی به جای DVD اخبار ردر سند xml قرار می دهند و سپس به شکلی که در بالا گفته شد آن را به صورت HTML نمایش می دهند.

در ادامه با نحوه ی استفاده از XPath در XSLT برای فیلتر کردن مقادیر بازیابی شده آشنا می شویم که مهمترین کاربردش مثل جستجو در سایت های خبری است.خوشبختانه با XPath و نحوه ی کارش در فیلتر کردن اسناد XML با استفاده از اشیا و متد های ASP.NET آشنا شدید و دیدید که چگونه با متد SelectNodes شی XmlDocument می توان داده ای xml را فیلتر کرد و همین کار را برای ما راحت تر می کند.ولی مهمتر از آن کاربرد آن در XSLT است.

در ابتدا برای مثال می خواهیم دنبال بازیگران فیلم Basic Instinct بگردیم و آنها را نمایش دهیم.می دانیم عبارت XPath برای آن به صورت زیر می شود:

```
DVDList/DVD[Title='Basic Instinct']/Starring/Star/
```

محل استفاده از این عبارت به عنوان مقدار برای ویژگی select تگ <xsl:apply-template> است:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="text" indent="yes"/>
      <xsl:template match="/">
            <xsl:apply-templates            select="/DVDList/DVD[Title='Basic
Instinct']/Starring/Star"/>
      </xsl:template>
</xsl:stylesheet>
```

می خواهیم دنبال عناوین فیلم هایی بگردیم که با حرف B شروع می شود:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="text" indent="yes"/>
      <xsl:template match="/">
            <xsl:apply-templates                    select="/DVDList/DVD[starts-
with(Title,'B')]"/>
      </xsl:template>
</xsl:stylesheet>
```

دنبال فیلم هایی می گردیم که در گروه Drama قرار دارند:

```
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="text" indent="yes"/>
      <xsl:template match="/">
            <xsl:apply-templates select="/DVDList/DVD[@Category='Drama']"/>
      </xsl:template>
</xsl:stylesheet>
```

دنبال مقادیر تمام فرزند های DVD می گردیم:

```
  <xsl:template match="/">
      <xsl:apply-templates select="/DVDList/DVD/*"/>
</xsl:template>
```

دنبال عنوان DVD تگ شماره ی ٤ می گردیم:

```
  <xsl:template match="/">
      <xsl:apply-templates select="/DVDList/DVD[4]/Title"/>
</xsl:template>
```

دنبال نام گروه DVD شماره ی چهارم می گردیم:

```
  <xsl:template match="/">
      <xsl:apply-templates select="/DVDList/DVD[4]/@Category"/>
</xsl:template>
```

با عملگر | می توانیم عمل ترکیب بین دو مقدار را انجام دهیم مثلا من عنوان DVD را با نام کارگردان ترکیب می کنم:

```
  <xsl:template match="DVD">
      <xsl:apply-templates select="Title|Director"/>
</xsl:template>
```

می توانیم با مقادیر بازیابی شده محاسبه انجام دهیم.مثلا من قیمت DVD با عنوان Original Sin را با قیمت DVD با عنوان Basic Instinct جمع می کنم:

```
  <xsl:template match="/">
      <xsl:text>Sum Of The Pric Of Original Sin &amp; Basic Instinct is :
</xsl:text>
      <xsl:value-of    select="/DVDList/DVD[Title='Original    Sin']/Price    +
/DVDList/DVD[Title='Basic Instinct']/Price"/>
</xsl:template>
```

یا اینکه قیمت تمامی DVD ها را ٤+ کنم و نمایش دهم(این ٤+ کردن باعث اعمال به سند xml نمی شود و صرفا یک نمایش ساده است):

```
<xsl:template match="DVD">
    <xsl:value-of select="Price + 4"/>
</xsl:template>
```

عملگر های مختلفی در XPath وجود دارند که با آنها می توانید اعمال مختلفی انجام دهید که لیست آنها را در جدول زیر مشاهده می کنید:

عملگر	نوع برگشتی	توضیح
and	Boolean	باعث ایجاد عمل And می شود
or	Boolean	باعث ایجاد عمل Or می شود
=	Boolean	عملگر مساوی
!=	Boolean	عملگر نامساوی
< (<)	Boolean	عملگر کوچکتر
<= (<=)	Boolean	عملگر کوچکتر یا مساوی
> (>)	Boolean	عملگر بزرگتر
>= (>=)	Boolean	عملگر بزرگتر یا مساوی
+	Number	جمع
-	Number	تفریق
*	Number	ضرب
div	Number	تقسیم
mod	Number	باقیمانده

برای مثال لیست DVD هایی را می خواهم که قیمتشان از ۲۵ $ بیشتر باشد:

```
<xsl:template match="/">
    <xsl:apply-templates select="/DVDList/DVD[(Price &gt; 25)]"/>
</xsl:template>
```

در ادامه به بررسی توابع و متد های موجود در XPath و XSLT می پردازیم.

در xml یک ویژگی وجود دارد به نام xml:lang که مخفف language است.که مشخص کننده ی مخفف زبان یک تگ است.مثلا:

<greeting xml:lang=”en”>welcome</greeting>

در کد بالا اگر ویژگی xml:lang را بنویسید و مساوی را کلیک کنید فهرستی از مخفف زبانهای موجود برای شما به نمایش در می آید و شما می توانید از آن فهرست زبانی را انتخاب کنید که ما در اینجا en را انتخاب کردیم.درز بودن این ویژگی دست ماست و این ما هستیم که رعایت گرامر زبان مربوطه را انجام می دهیم.سند xml را به صورت زیر می نویسیم:

```
<?xml version="1.0" encoding="utf-8" ?>
<greet>
    <greeting xml:lang="en">Welcome</greeting>
    <greeting xml:lang="fr">Bienvenue</greeting>
    <greeting xml:lang="es">Bienvenida</greeting>
    <greeting xml:lang="de">Willkommen</greeting>
</greet>
```

قصد ما انتخاب درست کلمه ی خوش آمد گویی در سند XSLT است.به این منظور XPath تابعی به نام lang() در اختیار ما گذاشته تا بتوانیم با توجه به شرایط گزینه ی درست را انتخاب کنیم.شاید کاربردی ترین مورد استفاده از lang و xml:lang در سایت های چند زبانه باشد.برای استفاده کافیست تابع را به فرم زیر بنویسیم:

lang(‘value name’)

مثلا در اینجا:

lang(‘en’)

مقدار برگشتی این تابع True یا false است.در حقیقت ما با استفاده از این تابع می فهمیم آیا زبان تگ مورد نظر انگلیسی هست یا خیر.برای نمایش محتویات تگ در صورت True بودن مقدار برگشتی تابع lang باید از آن در داخل [] استفاده کنیم که ناحیه ی انتخاب است .کد XSL زیر مقدار تگی که به زبان فرانسوی عمل خوش آمد گویی را انجام داده نمایش می دهد:

```
<xsl:template match="greet">
    French : <xsl:apply-templates select="greeting[lang('fr')]"/>
</xsl:template>
```

دقت کافی کنید که match=”Parent Tag” است یعنی ویژگی match با نام تگ والد تگی که می خواهیم نمایشش دهیم مقدار دهی شده.کد بالا را می توانستید به صورت زیر هم بنویسید:

```
<xsl:template match="/">
    French : <xsl:apply-templates select="/greet/greeting[lang('fr')]"/>
```

```
</xsl:template>
```

خروجی کد فوق به صورت زیر خواهد بود:

```
French : Bienvenue
```

پس در حالت کلی هنگام استفاده از تابع lang می بایست آن را در داخل کروشه یا ناحیه ی انتخاب به کار ببرید البته این ناحیه ی انتخاب همیشه باید وابسته به یک تگ باشد همان تگی که می خواهیم مقدارش ر نمایش دهیم:

tag name[lang('value name')]

این تابع از نوع توابع Boolean می باشد و امکان این نیست که آن را تنها صدابزنید مثلا کد زیر اشتباه است:

```
  <xsl:template match="/greet/greeting">
      <xsl:apply-templates select="lang('fr')"/>
</xsl:template>
```

اکثر توابع XPath برای استفاده در XSLT باید در داخل کروشه صدا زده شوند که کم کم با توابع دیگری نیز آشنا می شوید با علم به اینکه تابعی مثل Starts-With که قبلا در موردش توضیح داده ایم هم در داخل کروشه صدا زده شد.

از جمله توابع محاسباتی مهم تابع sum() است که نام یک سری تگ مشابه را به عنوان ورودی دریافت می کند و مقادیر آن تگ ها را با یکدیگر جمع می کند.برای مثال در سند مربوط به DVD ها می خواهیم مقادیر موجود در تگ Price را با هم جمع کنیم.به عبارت دیگر مجموع قیمت DVD ها را می خواهیم:

```
  <xsl:template match="/">
      <xsl:value-of select="sum(DVDList/DVD/Price)"/>
</xsl:template>
```

خروجی کد فوق یک عدد است که آن هم مجموع مقادیر تگ های Price است.

به این نکته دقت کنید که من آرگومان ورودی تابع sum را مسیر دقیق تگ موردنظر یعنی price قرار دادم.

نکته ی مهم دیگر این است که در اینگونه توابع مثل sum که مقدار برگشتیشان یک مقدار عددی است را نمی توان با تگ apply-template نمایش داد و حتما باید از تگ Value-Of استفاده کرد.

از جمله توابع دیگر در XPath تابع round() است.این تابع یک مقدار عددی را به عنوان آرگومان ورودی می پذیرد و آن عدد را رند می کند.مثلا :

round(23.45)=23

round(23.76)=24

در مثال زیر قیمت DVD شماره ی چهارم را رند کرده و نمایش دادیم:

```
  <xsl:template match="/">
      <xsl:value-of select="round(DVDList/DVD[4]/Price)"/>
</xsl:template>
```

قیمت DVD شماره ی چهارم ۲۳,۵۱ بوده که خروجی کد بالا عدد ۲٤ است.

می توان توابع را تودرتو هم صدا کرد(البته اگر در مقدار برگشتی تناقضی با هم نداشته باشند).مثلا ما مقار رند شده ی مجموع قیمت کل DVD ها را به صورت زیر محاسبه می کنیم:

```
  <xsl:template match="/">
      <xsl:value-of select="round(sum(DVDList/DVD/Price))"/>
</xsl:template>
```

از توابع دیگر می توان به تابع concat() اشاره کرد.این تابع هر چند تا آرگومان ورودی می پذیرد و تمام آنها را به هم وصل می کند و رشته ی حاصل از اتصالشان را برمی گرداند.در استفاده از این تابع حتما باید دو عبارت زیر را بلد باشید:

&#۱۰;

باعث رفتن به سطر بعد می شود.

&#۳۲;

باعث ایجاد یک فضای خالی می شود.

البته دو مورد بالا تنها در سورس html خروجی کار می کنند و در هنگام اجرا کار نمی کنند.

برای مثال من مقادیر عنوان فیلم-نام کارگردان –قیمت و بازیگران DVD شماره ی ٤ را به هم وصل کردم:

```
  <xsl:template match="/">
      <xsl:value-of    select="concat('The    Titel    Of    The    Film    is
',/DVDList/DVD[4]/Title,'&#10;','                Director                is
',/DVDList/DVD[4]/Director,'&#32;&#32;','Price      Of      This      DVD      :
',/DVDList/DVD[4]/Price,'&#10;','The    Star    Of    This    DVD    :    ',
'&#10;',/DVDList/DVD[4]/Starring/Star[1],'&#32;','&amp;','&#32;',/DVDList/DVD
[4]/Starring/Star[2])"/>
</xsl:template>
```

سورس html خروجی برای XSL بالا به شکل زیر می شود:

```
  The Titel Of The Film is Original Sin
   Director is Henri Wacdfgowi  Price Of This DVD : 23.51
```

```
  The Star Of This DVD :
Antonio Banderas & Anjelina Jolie
```

و اگر آن را اجرا کنید به صورت زیر خواهد بود:

The Titel Of The Film is Original Sin Director is Henri Wacdfgowi Price Of This DVD : 23.51 The Star Of This DVD : Antonio Banderas & Anjelina Jolie

تابع بعدی که معرفی می کنیم تابع مهم substring() است.همانطورکه از معنی آن نیز پیداست این تابع یک زیر رشته تولید می کند.این تابع می تواند ۲ یا ۳ آرگومان ورودی داشته باشد.اولین آرگومان آن رشته ی مورد نظر است که می خواهیم از آن یک زیر رشته تولید کنیم.دومین آرگومان یک مقدار عددی است.این مقدار بیانگر شماره ی کاراکتر هایی است که از اول رشته کم شود.و همینطور آرگومان ورودی سوم اگر باشد بیانگر این است که از آن تعداد کاراکتر هایی که توسط دومین آرگومان کم شده چقدر به جلو بیاییم.مثلا:

substring('ali reza',2)=li reza

substring('ali reza',2,4)=li r

در مثال اول عدد ۲ باعث شد ما از ابتدای رشته تا سر کاراکتر دوم از رشته بکاهیم.

در مثال دوم هم عدد ۲ باعث شد ما از ابتدای رشته تا سر کاراکتر دوم از رشته بکاهیم تا رشته به صورت li reza در آید حال از این رشته ٤ کاراکتر اول مورد قبول است.

دو تابع دیگر نیز از خانواده ی تابع substring() وجود دارند به نامهای substring-after() و substring-before که کار تابع substring() را به نحو محدودتری انجام می دهند و دو ارگومان ورودی ارند اولی خود رشته و دومی کاراکتری که با آن عمل انطباق انجام می شود.کاربرد این دو تابع در مثال زیر مشخص است:

```
substring-before('1963/02/13', '/')=1963
substring-after('1963/02/13', '/')=02/13
```

در مثال اول عمل انطباق با کاراکتر / از ابتدای رشته انجام شد و به محض برخورد با اولین کاراکتر / هر چه قبل از آن بود به خروجی رفت.

در مثال دوم عمل انطباق با کاراکتر / از ابتدای رشته انجام شد و به محض برخورد با اولین کاراکتر / هر چه بعد از آن بود به خروجی رفت.

تابع بعدی تابع translate() است.این تابع عمل جابجایی کاراکتر را انجام میدهد.این تابع ۳ آرگومان ورودی دارد ولی رشته ی مربوطه دومی کاراکتری که در رشته یمربوطه هست و ما می خواهیم آن را تبدیل کنیم و آرگومان سوم کاراکتری است که عمل تبدیل به

آن صورت می گیرد.مثلا در جمله ی ali reza ramin rahman می خواهم هر چه R داریم را به Z تبدیل کنم:

translate('ali reza ramin rahman','r','z')

کد XSL آن به صورت زیر می شود:

```
  <xsl:output method="html" indent="yes"/>
  <xsl:template match="/">
      <xsl:value-of select="translate('ali reza ramin rahman','r','z')"/>
</xsl:template>
```

و سورس Html خروجی به صورت زیر می شود:

```
ali zeza zamin zahman
```

حال که طرز کار دو تابع substring و translate را فهمیدید می خواهیم یک مثال کاربردی از ترکیب این دو تابع را با هم ببینیم.در این مثال من می خواهیم آدرس داخلی یک کامپیوتر را به آدرس وب تبدیل کنم .به این صورت که آدرس زیر را دریافت می کنم:

```
C:\LearningXSLT\examples\ch05\path.xml
```

و آن را تبدیل به آدرس وب زیر می کنم:

```
http://mysite.com/users/LearningXSLT/examples/ch05/path.xml
```

آنچه که در اینجا واضح است این است که علامت بین پوشه ها در سیستم داخلی یک کامپیوتر به صورت \ است.در حالی که علامت بین صفحات یک سایت / است.پس یکی از کارهای من تبدیل \ به / ذاست که باید با تابع translate انجام شود.همچنین یک مشکل دیگر این است که من در آدرس وب نیازی به C:\ ندارم و باید آن را حذف کنم.این کار را نیز با تابع substring انجام می دهم.پس تنها چیزی که می ماند اضافه کردن آدرس سایت و با احیانا صفحه ای خاص از سایت که با تگ <xsl:text> به سادگی انجام می گیرد.پس ابتدا با تابع substring رشته ی C:\ را از اول رشته ی مربوطه حذف می کنم.قبل از آن , سند xml را به صورت زیر می نویسم:

```
  <?xml version="1.0" encoding="utf-8" ?>
<doc>C:\LearningXSLT\examples\ch05\path.xml</doc>
```

برای حذف از تابع substring به صورت زیر استفاده می کنم:

```
substring(/doc,4)
```

/doc مقدار تگ doc در سند xml را برمی گرداند.عدد ٤ هم یعنی از اول رشته تا سر کاراکتر چهارم را حذف کن.پس تا اینجا من عبارت زیر را تولید کردم:

```
LearningXSLT\examples\ch05\path.xml
```

حالا با یک تابع translate جای تمام \ ها را با / عوض می کنم:

```
translate(substring(/doc,4),'\','/')
```

پس تگ <xsl:value of> به فرم زیر می شود:

```
<xsl:value-of select="translate(substring(/doc,4),'\','/')"/>
```

کافیست text دلخواهی را مبنی برآدرس سایت به تگ بالا بچسبانیم:

```
<xsl:text>http://mysite.com/users/</xsl:text>
```

پس کد کلی XSL به صورت زیر شد:

```
  <xsl:output method="html" indent="yes"/>
  <xsl:template match="/">
    <xsl:text>http://mysite.com/users/</xsl:text><xsl:value-of
select="translate(substring(/doc,4),'\','/')"/>
</xsl:template>
```

خروجی را هم که قبلا دیده بودید.

البته عبارت تولید شده ی فوق در خروجی یک نوشته ی ساده است و هیچ گونه لینکی ندارد.برای اینکه آن را به HyperLink تبدیل کنیم از تگ مخصوص HyperLink در html استفاده می کنیم یعنی تگ <a> .از این تگ به صورت زیر در html استفاده می شود:

Text

مقداری که برای href در نظر گرفته می شود آدرس صفحه ی وب مقصد است و Value تگ a همان text ی است که اگر رویش کلیک کنیم به آدرس موجود در ویژگی href می رویم.

پس در XSL باید هم تگ a و هم ویژگی href و هم Value برای تگ a تولید می کنیم:

```
  <xsl:template match="/">
      <xsl:element name="a">
            <xsl:attribute name="href">
                  <xsl:text>http://mysite.com/users/</xsl:text>
                  <xsl:value-of
select="translate(substring(/doc,4),'\','/')"/>
            </xsl:attribute>GO
      </xsl:element>
</xsl:template>
```

سورس خروجی کد فوق به صورت زیر خواهد بود:

```
<a
href="http://mysite.com/users/LearningXSLT/examples/ch05/path.xml">GO</a>
```

و اگر آن را اجرا کنید نوشته ی زیر روی صفحه ظاهر می شود که حاوی لینک بالا است:

GO

که اگر رویش کلیک کنیم به آدرس وبی که تولید کرده بودیم خواهیم رفت.

تابع بعدی که معرفی می کنیم تابع generate-id() است.در اسناد معتبر xml هر تگ یک Id دارد که Unique هم هست و به وسیله ی آن از سایر تگ ها قابل شناسایی است.تابع generate-id() می تواند برای تگ های مختلف یک مقدار شناسه ی واحد تولید کند.این تابع نام تگی که قرار است برایش شناسه تولید کند را به عنوان آرگومان ورودی می پذیرد.

در ابتدا یک مثال بسیار ساده با هم می بینیم و سپس یک مثال کمی پیچیده تر مطرح می کنیم.سند xml زیر را در نظر بگیرید:

```
<?xml version="1.0" encoding="utf-8" ?>
<doc>hellow</doc>
```

می خواهم یک خروجی به شکلی ایجاد کنم که برای تگ doc یک ویژگی id تعریف کنم و با استفاده از تابع generate-id() مقداری unique برایش در نظر بگیرم.برای این کار من باید تگ جدیدی به نام doc ایجاد کنم که Value ش را از تگ doc موجود در سند xml دریافت کنم.پس ابتدا تگ template را به صورت زیر می نویسم:

```
<xsl:output method="xml" indent="yes"/>
<xsl:template match="doc">

</xsl:template>
```

سپس در داخلش تگی به نام doc تعریف می کنم با یک ویژگی به نام id که مقدار این ویژگی بر اساس تگ جاری قرار داده شود.به عبارت دیگر می دانیم تگ جاری همان تگی است که نامش در قسمت match آمده.من یک Id بر اساس آن تولید می کنم و به ویژگی id تگ جدیدم نسبت می دهد و همینطور Value تگ جدیدم را هم از تگ جاری می گیرم.با علم اینکه تگ جاری را با . نمایش می دهند:

```
<xsl:element name="doc">
    <xsl:attribute name="id">
          <xsl:value-of select="generate-id(.)"/>
    </xsl:attribute>
    <xsl:value-of select="."/>
</xsl:element>
```

پس کد کلی به فرم زیر شد:

```
<xsl:output method="xml" indent="yes"/>
<xsl:template match="doc">
    <xsl:element name="doc">
          <xsl:attribute name="id">
                <xsl:value-of select="generate-id(.)"/>
```

```
            </xsl:attribute>
            <xsl:value-of select="."/>
      </xsl:element>
</xsl:template>
```

و سورس html خروجی به شکل زیر میشود:

```
  <?xml version="1.0" encoding="utf-8"?>
<doc id="ID0EC">hellow</doc>
```

به مقدار واحدی که به ویژگی id اختصاص داده شده توجه کنید.

حال به مثال پیچیده تر بپردازیم.

برای مثال سند ساده ی xml زیر را در نظر بگیرید:

```
  <?xml version="1.0" encoding="utf-8" ?>
  <doc>
      <a>hellow</a>
      <a>welocome</a>
      <a>thanks</a>
      <a>ok</a>
      <a>good</a>
</doc>
```

من می خواهم برای هر یک از تگ های a یک id واحد تولید کنم.البته این id ها پس از تولید به فایل xml اصلی اعمال نمی شود و تنها جنبه ی نمایش دارند و من می خواهم یک سند xml را در فایل html خروجی نمایش دهم که تمام تگ های a حاوی یک ویگی id با مقداری Unique باشد.برای این کار کد زیر را می نویسم:

```
  <xsl:template match="a">
      <xsl:element name="a">
            <xsl:attribute name="id">
                  <xsl:value-of select="generate-id(.)"/>
            </xsl:attribute>
            <xsl:value-of select="."/>
      </xsl:element>
</xsl:template>
```

من با عبارت match="a" مشخص کردم که با تگ هایی به نام a کار دارم.این به این معنی نیست که من می خواهم به تگ های a شناسه اضافه کنم.بلکه به این معنی است که من می خواهم تگ های جدیدی به نام a تولید کنم که Value آنها را از تگ جاری a در یافت کنم.

سپس یک تگ جدید ایجاد کردم به نام a با یک ویژگی id که مقدار آن را با generate-id(.) تولید کردم.علامت . در داخل generate-id نشان دهنده ی این است که برای تگ جاری می خواهد شناسه تولید کند حال این تگ جاری همان تگی است که در

قسمت match مشخص شده.دقت کنید مقداری که تابع generate-id(.) تولید می کند بر اساس تگ جاری است نه برای تگ جاری.ما مقدار شناسه را با generate-id(.) تولید می کنیم و سپس آن را در ویژگی id تگ جدیدی که ساختیم قرار می دهیم.

مقدارش هم با <xsl:value-of select=”.”> مشخص کردم.این کار باعث می شود مقدار تگ جاری به تگ جدید a انتقال داده شود.

این نکته نیز مهم است که تگ جاری در اینجا a است و تگ جدیدی هم که من ساختم a نام دارد و این دو با هم فرق می کنند.

بالا تر از کد بالا کد زیر را بنویسید:

```
<xsl:template match="/">
    <xsl:element name="doc">
        <xsl:apply-templates select="/doc/a"/>
    </xsl:element>
</xsl:template>
```

این کد مقادیر تمام فرزند های تگ a را نمایش می دهد.

پس کد کلی به شکل زیر شد:

```
<xsl:output method="xml" indent="yes"/>
    <xsl:template match="/">
        <xsl:element name="doc">
            <xsl:apply-templates select="/doc/a"/>
        </xsl:element>
    </xsl:template>
    <xsl:template match="a">
        <xsl:element name="a">
            <xsl:attribute name="id">
                <xsl:value-of select="generate-id(.)"/>
            </xsl:attribute>
            <xsl:value-of select="."/>
        </xsl:element>
</xsl:template>
```

در این مثال هم عمل انطباق یا میکس انجام می شود.به این ترتیب که با تگ template اول, تمام مقادیر تگ های a پشت سر هم نمایش داده می شوند .ولی وقتی تگ template دوم اجرا می شود تگ های تولید شده یکی یکی به جای مقادیر نمایش داده شده قرار می گیرند.برای مثال با تگ template اول, تمام مقادیر تگ های a پشت سر هم نمایش داده می شوند که به صورت زیر می شود:

<doc>hellowwelcomethanksokgood</doc>

که این در نمایش به این صورت است ولی در اصل به شکل زیر است چون هنوز به خروجی نرفته:

```
<doc><a>hellow</a><a>welcome</a><a>thanks<a></a>ok<a>good</a></do
c>
```

سپس با اولین اجرای تگ template دوم یک تگ به صورت زیر ساخته می شود:

```
<a id="ID0EE">hellow</a>
```

سپس عمل میکس انجام می شود و جای تگ <a>hellow با تگ بالا عوض می شود.این عمل به ترتیب برای تمام تگ های موجود انجام می گیرد (تا وقتی که تگی به نام a در عبارت مربوطه وجود داشته باشد).

پس خروجی به شکل زیر می شود:

```
<?xml version="1.0" encoding="utf-8"?>
<doc>
  <a id="ID0EE">hellow</a>
  <a id="ID0EG">welocome</a>
  <a id="ID0EI">thanks</a>
  <a id="ID0EK">ok</a>
  <a id="ID0EM">good</a>
</doc>
```

عمل میکس در XSL کاربرد فراوانی دارد . اگر می خواهید در XSL تبحر پیدا کنید باید آن را به خوبی درک کنید.

تابع بعدی last() است.این تابع Value آخرین تگ مشابه را بر می گرداند مثلا اگر سند xml ما همان سند بالایی باشد کد xsl را به صورت زیر بنویسید:

```
<xsl:template match="doc">
    <xsl:value-of select="a[last()]" />
</xsl:template>
```

عبارت s[last()] عبارتی بسیار ساده است.عبارت داخل کروشه مشخص کننده ی ناحیه ی انتخاب است.حال این ناحیه وابسته به تگی با نام a بوده و مقدارآخرین تگ a را بر می گرداند که اگر اجرایش کنید مقدار good خاهد بود.استفاده های زیادی می توان از تابع last() کرد مثلا خروجی کد زیر مقدار تمام تگ های a به غیر از آخری است:

```
<xsl:template match="doc">
    <xsl:apply-templates select="a[position()!=last()]"/>
</xsl:template>
```

تابع بعدی normalize-space() است.این تابع فضاهای خالی را نرمالیزه می کند.مثلا فرض کنید سند xml به صورت زیر باشد:

```
<?xml version="1.0" encoding="utf-8" ?>
<boulevard>
    <block>      Panorama Street</block>
    <block>Highland    Plaza  </block>
    <block>   Hutchens Avenue</block>
    <block>   Wildwood         Drive</block>
    <block>    Old     Chimney     Road</block>
    <block>  Carrol    Circle</block>
</boulevard>
```

می بینید که Value تگ ها در این سند بسیار آشفته و پر از white-space هستند.اگر ما عمل بازیابی را مثلا از چهارمین تگ block انجام دهیم به صورت زیر:

```
<xsl:template match="/">
    <xsl:value-of select="/boulevard/block[4]" />
</xsl:template>
```

آنگاه خروجی آن به شکل زیر می شود:

```
  Wildwood         Drive
```

می بینید که در خروجی عینا همان چیزی آمده که در سند بود.برای نرمالیزه کردن فضاهای خالی از تابع normalize-space() استفاده می کنیم و آرگومان ورودیش را نام تگی می دهیم که می خواهیم مقدارش نرمالیزه شود و سپس در خروجی چاپ شود:

```
<xsl:template match="/">
    <xsl:value-of select="normalize-space(/boulevard/block[4])" />
</xsl:template>
```

حال خروجی به شکل زیر می شود:

```
Wildwood Drive
```

می بینید که فضاهای خالی اضافه از ابتدا مابین و وسط رشته ی مورد نظر برداشته شد و سپس به خروجی رفت.

تابع مهم بعدی تابع contains() است.این تابع یک تابع boolean است و دو آرگومان ورودی دارد که اولی رشته ی مربوطه و دومی یک(یا چند) کاراکتر است و تشخیص می دهد که آیا آن کاراکتر در آن رشته وجود دارد یا خیر(True Or False).مثلا:

Contains('hellow word','wo')=true

Contains('hellow word','ui')=false

ما کی توانیم از این true & false استفاده کنیم.مثلا بگوییم آن مقادیری را برای من نمایش بده که در آنها این کلمه ی خاص وجود دارد.به سند xml زیر دقت کنید:

```
<?xml version="1.0" encoding="utf-8" ?>
<doc>
    <a>hellow</a>
    <a>hellow word</a>
    <a>thank you</a>
    <a>you are silly</a>
    <a>how are you</a>
</doc>
```

و کد xsl را به صورت زیر بنویسید:

```
<xsl:template match="/">
    <xsl:value-of select="//a[contains(.,'you')]" />
</xsl:template>
```

در این کد بحث ما مقدار تمام تگ هایی با نام a خواهد بود(//a).

در این کد اولین تگ a که مقدارش حاوی رشته ی you باشد بازیابی می شود و آن هم thank you خواهد بود.

اگر بخواهید تمام مقادیری که در آنها از رشته ی you استفاده شده را بازیابی کنید می بایست به جای Value-of از تگ apply-template استفاده کنید(همانطور که پیش تر هم اشاره کرده بودیم یک فرق مهم بین <xsl:value-of> و <xs:apply-template> این است که <xsl:value-of> قادر نیست در مقدار برگشتی برخی توابع بیش از یک مقدار را نمایش دهد ولی <xs:apply-template> قادر است):

```
<xsl:template match="/">
    <xsl:apply-templates select="//a[contains(.,'you')]" />
</xsl:template>
```

خروجی کد فوق به صورت زیر می شود:

```
thank youyou are sillyhow are you
```

ما در رشته ی مورد بحث تابع contains از یک نقطه استفاده کردیم این یعنی تگ جاری.یعنی نوبت به بررسی هر تگی می رسد تنها در همان تگ تابع contains اعمال شود.به جای . نمی توانستیم از a یا هر چیزی که مسیری برای a می بود استفاده می کردیم زیرا آن وقت از پیمایش تمام تگ ها خبری نبود.به عبارت دیگر ما به هر تگی می رسیم باید مقدار برگشتی تابع contains مربوط به همان تگ(تگ جاری) را بررسی کنیم و بگوییم آیا رشته ی you در آن هست یا نه و در غیر این صورت برنامه با یک خطای نحوی روبرو میشد.کاربرد ی که از تابع contain می توان ذکر کرد جستجو بر اساس کلمه ی کلیدی است.

تابع ساده ی بعدی تابع string-lenght() است.این تابع طول یک رشته را محاسبه و برمیگرداند.پس طبیعتا یک آرگومان ورودی دارد و آن هم همان رشته است.مثلا:

string-lenght('ali')=3

تابع بعدی که معرفی می کنیم تابع not() است.این تابع عمل نقیض را انجام می دهد مثلا اگر ورودیش false باشد خروجیش true است.مثلا طبق سند xml مثال مربوط به تابع contains کد xsl را به صورت زیر بنویسید:

```
  <xsl:template match="/">
      <xsl:apply-templates select="//a[not(contains(.,'you'))]" />
</xsl:template>
```

همانطور که می بینید روی تابع contains یک تابع not آمده و همین کار را برعکس می کند و مقادیر تگ هایی به خروجی می روند که حاوی رشته ی you نیستند:

```
hellow hellow word
```

در اینجا به معرفی یک تابع و یک تگ همنام و به هم وابسته می پردازیم.نام تگ <xsl:key> ونام تابع key() است.کاری که این دو با هم انجام می دهند بازیابی داده از سند xml است.به گونه ای که در ابتدا با تگ <xsl:key> یک الگو برای بازیابی مشخص می کنید و سپس با تابع key() عمل بازیابی را انجام می دهید.تگ <xsl:key> یک تگ top-level است یعنی جزو فرزندان تگ <xsl:style-sheet> محسوب می شود و نمی توان از آن در داخل تگ <xsl:template> استفاده کرد.نحوه ی تعریف آن به صورت زیر است:

<xsl:key name=" name" match="pattern name" use="use name"/>

اولین ویژگی name نام دارد که تنها برای شناسایی تگ <xsl:key> برای استفاده در تابع key کاربرد دارد.

ویژگی بعدی match است.این ویژگی یک الگو برای بازیابی مشخص می کند و همان نقشی را در اینجا دارد که match معمولی در تگ <xsl:template> داشت.من خودم کمتر الگو به match موجود در <xsl:template> دادم و بیشتر الگوهای بازیابی را به value-of وapply-template دادم .بهتر است در اینجا هم همین کار را بکنید.مثلا نام گره ای را بدهید که می خواهید از تگ های فرزند آن عمل بازیابی را انجام دهید و یا یک علامت / خالی.درکل اگر بخواهید از الگوهای کامل در match موجود در

<xsl:template>استفاده کنید دستتان برای فیلتر نمایش داده ها در تگ هایی همچون value-of وapply-template بسته خواهد بود.

ویژگی آخر use است.این ویژگی مشخص می کند تابع key() از چه گره ای باید استفاده کند.حال این گره می تواند هم تگ و هم ویژگی باشد.کد زیر بر مبنای سند xml مربوط به DVD هاست.

```
<xsl:key name="myDVD" match="DVD" use="@id"/>
```

در این کد نام تگ <xsl:key> را myDVD قرار دادم.الگویش را هم تگ DVD دادم چون می خواستم عمل بازیابی را از تگ های فرزند آن انجام دهم.

حال که یک الگو ایجاد کردیم و گفتیم حق استفاده از ویژگی id و تگ های فرزند تگ DVD را دارید می رسیم به استفاده از آن که بر عهده ی تابع key است.این تابع عمل بازیابی را انجام می دهد و برای استفاده می بایست در یکی از دو تگ apply-template و یا value-of و... استفاده شودهمانطور که می دانید این تگ ها حتما باید در داخل <xsl:template> باشند.پس در نتیجه ازتابع key() باید در داخل <xsl:template> استفاده شود.این تابع دو آرگومان ورودی دارد.اولی نام تگ <xsl:key> و دومی مقداری است که باید با مقدار ویژگی use همخوانی داشته باشد مثلا اگر ویژگی use معرف یک ویژگی باشد در اینجا آرگومان دوم می توان مقدار آن ویژگی باشد :

```
key('myDVD', '3222')
```

در این کد می بینید که نام key که آرگومان ورودی اول است دقیقا همان نامی است که در ویژگی name تگ <xsl:key> تعریف کرده بودیم.و مقدارش هم 3222 است که می بایست با مقداری که در ویژگی use تعریف شده بود همخوانی داشته باشد.آن مقدار معرف ویژگی id بود و در اینجا هم مقدارش 3222 است پس در نتیجه در تگهایی با نام DVD(که در ویژگی match تگ <xsl:key> مشخص شده بود) به دنبال تگی می گردیم که id=’3222’ باشد.جالب اینجاست که شما محدود به استفاده از مقادیر بازیابی شده توسط تابع key() نیستید و می توانید خودنیز الگوهایی را به آن اضافه کنید:

```
  <xsl:key name="myDVD" match="DVD" use="@id"/>
  <xsl:template match="/">
      <xsl:value-of select="key('myDVD', '3222')/Title"/>
</xsl:template>
```

در این کد من Value تگ title را از تگ DVD که id='3222' دارد بازیابی کردم و خروجی به شکل زیر شد:

```
Basic Instinct
```

پس متوجه می شویم کاربرد عمده ی Key می تواند در جستجو ها باشد.یادتان باشد برای درست کار کردن تابع key() مقدار match تگ <xsl:template> را برابر / قرار دهید.

می توان از چندین تگ <xsl:key> در برنامه ی خود استفاده کرد مثلا:

```
<xsl:key name="myDVD" match="DVD" use="@id"/>
<xsl:key name="DVD" match="DVD" use="Title"/>
<xsl:template match="/">
    <xsl:value-of select="key('myDVD', '3222')/Title"/>
    <xsl:text> ---</xsl:text>
    <xsl:value-of select="key('DVD', 'Basic Instinct')/Starring/Star[1]"/>
    <xsl:text> ---</xsl:text>
    <xsl:value-of select="key('DVD', 'Basic Instinct')/Starring/Star[2]"/>
</xsl:template>
```

در این مثال من نام بازگران DVD با نام Basic Instinct (Title) را نیز بازیابی کردم.تگ دوم Key مقدار ویژگی useش برابر Title است.پس می بینید حتما ملزم نیستید آن را با نام یک ویژگی مقداردهی کنید.پس از آشنایی با موارد دیگر مثل parameter ها باز هم با key کار خواهیم کرد.

تگ بعدی که معرفی می کنیم تگ <xsl:copy-of> است.این تگ بر اساس نام تگی که در ویژگی select خود دریافت می کند یک کپی از تمام انشعاب گره هایی که نامش را با وِیژگی select دریافت کرده ایجاد می کند و به خروجی می برد.منظور از انشعاب , تمام گره های فرزند و نوادگان و حتی ویژگی هاست.

برای مثال ابتدا سند xml زیر را در نظر بگیرید:

```
<?xml version="1.0" encoding="utf-8" ?>
<main>
    <parkway>
        <thoroughfare>Governor Drive</thoroughfare>
        <thoroughfare name="Whitesburg Drive">
            <sidestreet>Bob Wallace Avenue</sidestreet>
            <block>1st Street</block>
            <block>2nd Street</block>
            <block>3rd Street</block>
            <sidestreet>Woodridge Street</sidestreet>
```

```
            </thoroughfare>
            <thoroughfare name="Bankhead">
                  <sidestreet>Tollgate Road</sidestreet>
                  <block>First Street</block>
                  <block>Second Street</block>
                  <block>Third Street</block>
                  <sidestreet>Oak Drive</sidestreet>
            </thoroughfare>
      </parkway>
      <boulevard>
            <block>Panorama Street</block>
            <block>Highland Plaza</block>
            <block>Hutchens Avenue</block>
            <block>Wildwood Drive</block>
            <block>Old Chimney Road</block>
            <block>Carrol Circle</block>
      </boulevard>
</main>
```

در این سند دو نوع راه مشخص شده.یکی بزرگراه و دیگری بولوار. بزرگراه یا parkway می تواند مستقیم باشد و یا حاوی خیابان بزرگ Thoroughfare باشد.هر Thoroughfare یا خیابان بزرگ نیزمی تواند حاوی خیابان کوچک SideStreet و کوچه Block باشد.ولی یک بولوار تنها می تواند حاوی کوچه باشد.هر Thoroughfare اگر حاوی کوچه و یا خیابان بزرگ نباشد نامش به عنوان Value ذکر می شود و اگرحاوی کوچه و یا خیابان بزرگ باشد نامش به عنوان ویژگی name خواهد بود.

به کد زیر نگاه کنید:

```
  <xsl:template match="tag(s) name">
      <xsl:copy-of select="tag(s) anme"/>
</xsl:template>
```

ما دو نوع انتخاب جداگانه داریم.اولی مقداردهی به ویژگی match و دومی مقدار دهی به ویژگی select است.اصل کار مقدار دهی به ویژگی select است.به این صورت که شما نام هر تگی را به آن بدهید عمل کپی ازآن تگ صورت می گیرد و نمایش داده می شود.ولی مقداری که به ویژگی match می دهید ناحیه ای را مشخص می کند که عمل کپی باید به صورت مستثنی صورت گیرد و تنها از Value تگ ها علاوه بر کپی که با ویژگی select مشخص شده بود.

با توجه به سند xml بالا کد xsl را به صورت زیر بنویسید:

```
  <xsl:template match="parkway">
      <xsl:copy-of select="/"/>
</xsl:template>
```

در این کد من ناحیه ای را که باید عمل کپی به درستی از تمام عناصر (نام تگ نام ویژگی و...) گرفته شود را ریشه در نطرگرفتم و ناحیه ای را که عمل کپی باید تنها از Value تگ ها از آن صورت نگیرد را parkway در نظر گرفتم.اگر به کد خروجی نگاه کنید منظور من را در می یابید:

```
<?xml version="1.0" encoding="utf-8"?>
    <main>
    <parkway>
          <thoroughfare>Governor Drive</thoroughfare>
          <thoroughfare name="Whitesburg Drive">
                <sidestreet>Bob Wallace Avenue</sidestreet>
                <block>1st Street</block>
                <block>2nd Street</block>
                <block>3rd Street</block>
                <sidestreet>Woodridge Street</sidestreet>
          </thoroughfare>
          <thoroughfare name="Bankhead">
                <sidestreet>Tollgate Road</sidestreet>
                <block>First Street</block>
                <block>Second Street</block>
                <block>Third Street</block>
                <sidestreet>Oak Drive</sidestreet>
          </thoroughfare>
    </parkway>
    <boulevard>
          <block>Panorama Street</block>
          <block>Highland Plaza</block>
          <block>Hutchens Avenue</block>
          <block>Wildwood Drive</block>
          <block>Old Chimney Road</block>
          <block>Carrol Circle</block>
    </boulevard>
</main>

          Panorama Street
          Highland Plaza
          Hutchens Avenue
          Wildwood Drive
          Old Chimney Road
Carrol Circle
```

می بینید که چون من select="/" در نظر گرفتم کل تگ ریشه به خروجی آمد و از طرف دیگر چون match="parkway" در نظر گرفتم عمل کپی که قرار بود شامل

Value تگ ها باشد را تنها از تگ های باقیمانده یعنی boulevard گرفت و ParkWay از این امر مستثنی شد.حال اگر می خواهید همه ی تگ ها مستثنی شوند کافیست match="/" قرار دهید.در حالت کلی اگر بیش از یک تگ داشتید و برای جلوگیری از شلوغی سند XSL می خواستید text همه ی آنها را پاک کنید می توانید از تگ حذف به صورت زیر استفاده کنید:

<xsl:template match="text()"/>

بله از تابع text() استفاده کردیم این تابع در حقیقت Value یک تگ را به ما می دهد .این کد باعث حذف هرگونه نوشته ی خالی و بدون تگ از خروجی می شود.این نکته را بدانید که در صورت گلچین مقادیر مستثنی نباید از کد بالا استفاده کنید.مثلا ۷ تگ کلی دارید که می خواهید ۳ تای آنها از مستثنی ها باشند و حتی نوشته های خالیشان در خروجی باشد و text بقیه ی تگ ها پاک شود در این صورت نمی توانید از کد بالا استفاده کنید.

بعضی مواقع هم استفاده نکردن از کد بالا برای حذف نوشته های خالی امکان ندارد.در مثال بالا ما میدانستیم که تگ parkway باید مستثنی شود ولی در بعضی اسناد xsl کار آنقدر پیچیده می شود که اصلا متوجه نمی شوید مقادیری که به صورت نوشته ی خالی وجود دارند به درستی مال مدام تگ هستند و یا مقادیر قاطی از تگ های مختلف هستند و...آن موقع است که بهترین راه برای خلاص شدن از شر نوشته ی خالی استفاده از تگ حذف با تابع text() است.

یادتان باشد مقداری که برای ویژگی select در نظر می گیرید بی ربط به مقداری که برای ویژگی match در نظر می گیرید نیست.ربطشان این است که به هیچ وجه نباید گره ای که در match مشخص می کنید نواده ی گره ی ای باشد که در select مشخص می کنید.ولی برعکس این مورد صحیح است.

مثلا خروجی کد زیر بسیار عجیب خواهد بود چون sidestreet جزو نواده ی / است:

```
<xsl:template match="sidestreet">
    <xsl:copy-of select="/"/>
</xsl:template>
```

به سند XSL زیر دقت کنید:

```
<xsl:output method="xml" indent="yes"/>
<xsl:template match="boulevard"/>
<xsl:template match="parkway">
```

```
        <new-parkway>
              <xsl:copy-of select="thoroughfare"/>
        </new-parkway>
</xsl:template>
```

در اینجا به کدی جدید بر می خوریم و آن هم کد زیر است:

```
  <xsl:template match="boulevard"/>
```

این کد به این معنی است که تمام مواردی که مربوط به تگ boulevard می شود را متوقف کن.

برای درک هر چه بیشتر این امر یک میان بر می زنیم.

به سند ساده ی xml زیر دقت کنید:

```
  <?xml version="1.0" encoding="utf-8" ?>
  <docing>
  <doc1>
      <a>hellow</a>
      <a>welocome</a>
      <a>thanks</a>
      <a>ok</a>
      <a>good</a>
  </doc1>
  <doc2>
      <b>salam</b>
      <b>khosh amadid</b>
      <b>mamnoon</b>
      <b>bashe</b>
      <b>khoob</b>
  </doc2>
</docing>
```

و سند XSL را به صورت زیر بنویسید:

```
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="doc2"/>
  <xsl:template match="docking">
      <xsl:apply-templates/>
</xsl:template>
```

در این سند در تگ template دوم هدف چاپ تمامی عناصر و فرزندان و نوادگان تگ docing با استفاده از apply-template است.در حالیکه تگ <xsl:template match=”doc2”/> در حقیقت یک استثنا روی تگ template دوم قایل می شود و محتویاتش را از نمایش محروم می کند.خروجی کد فوق به صورت زیر است:

```
  <?xml version="1.0" encoding="utf-8"?>

  hellow
  welocome
```

```
  thanks
  ok
good
```

حال به مثال اصلی بر می گردیم.در آن مثال هم تگ copy-of یک نسخه از تمام فرزندان و نوادگان و زیر درخت ها ی تگ هایی با نام Thoroughfare می گیرد و سپس استثنا هم چک می شود که parkway است پس آن قسمتی که بدون تگ نمایش داده می شود Value تگ های Boulevard خواهد بود.پس خروجی مثال ما بدون استفاده از تگ حذف (<xsl:template match=" Boulevard"/>) به صورت زیر شد:

```
  <?xml version="1.0" encoding="utf-8"?>
      <new-parkway><thoroughfare>Governor    Drive</thoroughfare><thoroughfare
name="Whitesburg Drive">
                  <sidestreet>Bob Wallace Avenue</sidestreet>
                  <block>1st Street</block>
                  <block>2nd Street</block>
                  <block>3rd Street</block>
                  <sidestreet>Woodridge Street</sidestreet>
            </thoroughfare><thoroughfare name="Bankhead">
                  <sidestreet>Tollgate Road</sidestreet>
                  <block>First Street</block>
                  <block>Second Street</block>
                  <block>Third Street</block>
                  <sidestreet>Oak Drive</sidestreet>
            </thoroughfare></new-parkway>

            Panorama Street
            Highland Plaza
            Hutchens Avenue
            Wildwood Drive
            Old Chimney Road
Carrol Circle
```

حال اگر تگ حذف را قرار دهید دیگر نوشته های بدون تگ ظاهر نمی شوند.

پس اگر از تگ <xsl:template match=" boulevard"/> استفاده می کریم به صورت زیر می شد:

```
  <?xml version="1.0" encoding="utf-8"?>
      <new-parkway><thoroughfare>Governor    Drive</thoroughfare><thoroughfare
name="Whitesburg Drive">
                  <sidestreet>Bob Wallace Avenue</sidestreet>
                  <block>1st Street</block>
                  <block>2nd Street</block>
                  <block>3rd Street</block>
                  <sidestreet>Woodridge Street</sidestreet>
            </thoroughfare><thoroughfare name="Bankhead">
```

```
            <sidestreet>Tollgate Road</sidestreet>
            <block>First Street</block>
            <block>Second Street</block>
            <block>Third Street</block>
            <sidestreet>Oak Drive</sidestreet>
        </thoroughfare></new-parkway>
```

دوباره این نکته را متذکر می شوم که هیچ وقت گره ای که در **match** مشخص می کنید نباید نواده ی گره ای باشد که در **select** مشخص می کنید .البته در این صورت از شما خطایی گرفته نمی شود ولی نتیجه عجیب خواهد بود.

البته من خودم راه ساده تر ی را با **XPath** برای استفاده از **<xsl:copy-of>** به صورت زیر پیشنهاد می کنم:

```
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/">
      <new-parkway>
            <xsl:copy-of select="/main/parkway/thoroughfare"/>
      </new-parkway>
</xsl:template>
```

در اینجا یک راست محتویات تمام تگ هایی با نام **throughfare** کپی شده و به خروجی می روند.

تگ بعدی که معرفی می کنیم تگ **<xsl:copy>** است.این تگ یک کپی از **Value** یک تگ تهیه می کند و با تگ **<xsl:apply-template>** به خروجی می برد.طرز استفاده از این تگ به صورت زیر است:

```
<xsl:copy>
 <xsl:apply-template/>
</xsl:copy>
```

مثلا کد زیر از تمامی **Value** های تگ های موجود در سند مربوط به راهها کپی گرفته و نمایش می دهد:

```
  <xsl:template match="/">
      <xsl:copy>
            <xsl:apply-templates/>
      </xsl:copy>
</xsl:template>
```

خروجی به شکل زیر می شود:

```
  <?xml version="1.0" encoding="utf-8"?>
            Governor Drive
```

```
                Bob Wallace Avenue
                1st Street
                2nd Street
                3rd Street
                Woodridge Street


                Tollgate Road
                First Street
                Second Street
                Third Street
                Oak Drive


            Panorama Street
            Highland Plaza
            Hutchens Avenue
            Wildwood Drive
            Old Chimney Road
            Carrol Circle
```

کپی هایی که توسط تگ <xsl:copy> ایجاد می شوند برای نمایش در خروجی نیازمند این هستند که در داخل یک تگ نمایش داده شوند.این تگ را می توانید در ویژگی match تگ <xsl:template>ی که برسر تگ <xsl:copy> می آورید مشخص کنید.

مثلا اگر مثال بالا را به صورت زیر بنویسید:

```
  <xsl:template match="parkway">
      <xsl:copy>
            <xsl:apply-templates/>
      </xsl:copy>
</xsl:template>
```

که خروجی به صورت زیر می شود:

```
      <parkway>
            Governor Drive

                  Bob Wallace Avenue
                  1st Street
                  2nd Street
                  3rd Street
                  Woodridge Street


                  Tollgate Road
                  First Street
                  Second Street
```

```
                Third Street
                Oak Drive

        </parkway>

            Panorama Street
            Highland Plaza
            Hutchens Avenue
            Wildwood Drive
            Old Chimney Road
Carrol Circle
```

پس با دادن نام یک تگ در ویژگی match در حقیقت ناحیه ای که کپی های گرفته شده باید نمایش داده شوند را مشخص کردیم.حال شما می توانید این ناحیه راکوچکتر کنید.مثلا به جای اینکه match=”parkeway” باشد آن را برابر block قرار می دهیم:

```
<xsl:output method="xml" indent="yes"/>
<xsl:template match="block">
    <xsl:copy>
<name>
          <xsl:apply-templates/>
</name>
    </xsl:copy>
</xsl:template>
```

در کد بالا ناحیه ای که مقادیر کپی شده باید نمایش داده شوند را block دادیم به این ترتیب مقادیر مربوط به block در داخل خود تگ block مشخص شدند.کد خروجی به صورت زیر خواهد شد:

```
            Governor Drive
                Bob Wallace Avenue
                <block><name>1st Street</name></block>
                <block><name>2nd Street</name></block>
                <block><name>3rd Street</name></block>
                Woodridge Street
                Tollgate Road
                <block><name>First Street</name></block>
                <block><name>Second Street</name></block>
                <block><name>Third Street</name></block>
                Oak Drive
            <block><name>Panorama Street</name></block>
            <block><name>Highland Plaza</name></block>
            <block><name>Hutchens Avenue</name></block>
            <block><name>Wildwood Drive</name></block>
            <block><name>Old Chimney Road</name></block>
<block><name>Carrol Circle</name></block>
```

پس می بینید که تگ <xsl:copy> چقدر می تواند کاربردی باشد.نسبت به تگ <xsl:copy-of> تگ <xsl:copy> بسیار پر کاربرد تر است.تا اینجا همیشه از تگ <xsl:apply-template/> در داخل <xsl:copy> استفاده کردیم.این امر باعث نمایش تمام مقادیر کپی گرفته شده در خروجی می شود.برای محدود کردن آن می توانید از تگ حذف(<xsl:template match="tag name to be remove"/> استفاده کنید.

اگر کد ما به صورت زیر بود:

```
<xsl:template match="thoroughfare">
    <xsl:copy>
          <xsl:apply-templates select="sidestreet" />
    </xsl:copy>
</xsl:template>
```

چیز جدید در اینجا این است که تگ apply-template حاوی ویژگی select است.این ویژگی نمایش داده های کپی گرفته شده را محدود می کند.مثلا در بالا اگر ویژگی select نبود خروجی به صورت زیر بود:

```
<?xml version="1.0" encoding="utf-8"?>

          <thoroughfare>Governor Drive</thoroughfare>
          <thoroughfare>
                Bob Wallace Avenue
                1st Street
                2nd Street
                3rd Street
                Woodridge Street
          </thoroughfare>
          <thoroughfare>
                Tollgate Road
                First Street
                Second Street
                Third Street
                Oak Drive
          </thoroughfare>
          Panorama Street
          Highland Plaza
          Hutchens Avenue
          Wildwood Drive
          Old Chimney Road

Carrol Circle
```

که کاملا واضح است.حال ویژگیselect باعث می شود مقادیر نمایش داده شده محدود به نام تگی باشند که در آن مشخص شده.در اینجا نام sidestreet به آن داده شده.پس مقادیری را که قرار است نمایش دهد تنها Value تگ های sidstreet است.که خروجی آن به صورت زیر است:

```
<?xml version="1.0" encoding="utf-8"?>

        <thoroughfare />
        <thoroughfare>Bob Wallace AvenueWoodridge Street</thoroughfare>
        <thoroughfare>Tollgate RoadOak Drive</thoroughfare>
        Panorama Street
        Highland Plaza
        Hutchens Avenue
        Wildwood Drive
        Old Chimney Road

Carrol Circle
```

پس می بینید که می توان مقادیر کپی گرفته شده را محدود کرد.واقعا جالب است که من توانستم مقادیر تگ هایی با نام sidestreet رادر داخل تگ thoroughfare نمایش دهم.

می توان از <xsl:copy-of> و <xsl:copy> با هم استفاده ی بهینه کرد.مثلا از تمام تگ هایی با نام block کپی گرفته و آنها را در داخل تگی boulovared به خروجی ببریم.

برای دسترسی به تمام block های کل سند xml می بایست از یک عبارت XPath استفاده کرد.و آن هم //block است.این عبارت صرف نظر از اینکه والد گره ی block چیست تمام گره هایی با نام block که از تگ ریشه مشتق می شوند را بازیابی میکند.البته از آن باید در جای مناسب استفاده کنیم.ما در مورد تگ <xsl:copy-of> گفتیم که با استفاده از آن می توان از تمام انشعاب هایی از گره ای که به ویژگی select آن داده بودیم کپی تهیه کنیم.پس عبارت را به صورت زیر می نویسیم:

```
<xsl:copy-of select="//block"/>
```

نتیجه ی کد بالا تنها در صورت وجود ویژگی match مشخص می شود که در پایین به آن اشاره می کنیم.

برای اینکه تگ های block به درستی نمایش داده شوند و چیز اضافه ی دیگری نباشد همانطور که گفتیم باید از match="/" استفاده کنیم.ولی در اینجا چون ما می خواهیم تگ

های block را در داخل تگ boulovared قرار دهیم ناچارا باید match=" boulovared" باشد(دلیلش را جلوتر می فهمید).

حال کافیست این کپی که تهیه کردیم را به جای تگ block قرار دهیم.این کار را تگ <xsl:copy> انجام می دهد:

```
  <xsl:template match="boulevard">
      <xsl:copy-of select="//block"/>
</xsl:template>
```

در این کد تمام تگ هایی با نام block به همراه مقدارشان به خروجی می رود ولی اینجا boulovared مستثنی است پس Value تمام تگ هایی که جزو فرزندان و نواده ی تگ ParkWay هستند بدون هیچ تگی به صورت Text نمایش داده می شوند.برای حذف آنها از تگ حذف زیر استفاده می کنیم:

```
<xsl:template match="parkway"/>
```

حال می رسیم به استفاده از تگ <xsl:copy>.این تگ از تمام سند کپی می گیرد ولی صرفا آنهایی را نمایش می دهد که در داخلش مشخص شده.اگر کد را به صورت زیر بنویسیم:

```
  <xsl:template match="boulevard">
      <xsl:copy>
            <xsl:copy-of select="//block" />
      </xsl:copy>
</xsl:template>
```

در اینجا تگ < xsl:copy> از تمام سند کپی می گیرد ولی برای نمایش ناچار است محتویاتی را که تگ <xsl:copy-of> مشخص می کند را نمایش دهد.یادتان هست که قبلا از apply-template استفاده می کردیم.پس از نمایش هم به مقدار ویژگی match نگاه می کند و می بیند که باید مقادیری که می خواهد نمایش دهد در داخل تگ <boulevard> قرار دهد.

خروجی هم به شکل زیر می شود:

```
<?xml version="1.0" encoding="utf-8"?>
    <boulevard>
    <block>1st Street</block>
    <block>2nd Street</block>
    <block>3rd Street</block>
    <block>First Street</block>
    <block>Second Street</block>
    <block>Third Street</block>
    <block>Panorama Street</block>
    <block>Highland Plaza</block>
```

```
        <block>Hutchens Avenue</block>
        <block>Wildwood Drive</block>
        <block>Old Chimney Road</block>
        <block>Carrol Circle</block>
</boulevard>
```

در اینجا چون مقادیر توسط XPath Syntax // تولید شد تگ <boulevard> ناچار است روی تمام تگ ها اعمال شود نه فقط تگ های blockی که در داخل خود <boulevard> بودند زیرا وقتی از // استفاده می کنیم دیگر از والدین تگ block صرف نظر می کنیم پس boulevard نمی داند که کدام تگ block مال خودش است و کدام مال parkway پس ناچار است روی تمام آنها اعمال شود.

در اینجا هم دقت کنید چون از تگ <xsl:copy-of> استفاده شده نباید تگی که در ویژگی match مشخص شده نواده ی تگی باشد که در ویژگی select مشخص شده است.

حال اگر کد xsl ما به صورت زیر باشد:

```
   <xsl:template match="/">
       <xsl:copy-of select="//block"/>
</xsl:template>
```

آنگاه خروجی به شکل زیر می شود:

```
   <?xml version="1.0" encoding="utf-8"?>
   <block>1st Street</block>
   <block>2nd Street</block>
   <block>3rd Street</block>
   <block>First Street</block>
   <block>Second Street</block>
   <block>Third Street</block>
   <block>Panorama Street</block>
   <block>Highland Plaza</block>
   <block>Hutchens Avenue</block>
   <block>Wildwood Drive</block>
   <block>Old Chimney Road</block>
<block>Carrol Circle</block>
```

همانطور که می بینید بی دردسر به تمام تگ هایی با نام block دسترسی داریم ولی نمی توان آنها را به سادگی در اخل تگ <boulevard> قرار داد.

حالا نوبت به تگ <xsl:for-each> است.این تگ دقیقا مثل حلقه های for-each روی یک دشته گره ی خاص پیمایش می کند و ما می توانیم مقادیر دلخواهی از قبیل Value آن تگ ها را از آن بازیابی کنیم.این تگ حاوی یک ویژگی به نام select است که به وسیله

ی آن می توانید محدوده ای که حلقه باید ویش زده شود را مشخص کنید. فرم کلی تگ <xsl:for-each> به صورت زیر است:

```
<xsl:for-each select=”tags Bound”>
your favorite Retrive...
</xsl:for-each>
```

برای مثال در سند مربوط به DVD ها کد xsl زیر را بنویسید:

```
<xsl:template match="/">
    <xsl:for-each select="/DVDList/DVD">
          <xsl:value-of select="name(.)"/>
          <xsl:text>&#10;</xsl:text>
    </xsl:for-each>
</xsl:template>
```

در این کد ساده تنها نحوه ی کار کردن حلقه ی for-each را میبینید.تابع name() تابعی است که نام تگ جاری(.) را بر می گرداند.محدوده ای که حلقه رویش زده شود را /DVDList/DVD مشخص کردیم یعنی حلقه روی تک هایی با نام DVD که فرزند DVDList هستند میزند.سپس به هر یک که رسید نامش را به خروجی می برد.پس خروجی به فرم زیر شد:

```
<?xml version="1.0" encoding="utf-8"?>DVD
DVD
DVD
DVD
DVD
DVD
DVD
```

و اگر کد را به صورت زیر بنویسید:

```
<xsl:template match="/">
    <xsl:for-each select="/DVDList/DVD">
          <xsl:value-of select="Title"/>
          <xsl:text>&#10;</xsl:text>
    </xsl:for-each>
</xsl:template>
```

در این کد من در هر بار گردش در هر تگ Value DVD تگ Title آن را در خروجی نوشتم.

خروجی به شکل زیر می شود:

```
<?xml version="1.0" encoding="utf-8"?>The Matrix
Basic Instinct
Basic Instinct2
```

```
Original Sin
Desperado
Open Range
Last Samurai
```

می توان با حلقه ی for-each عمل ایجاد جداول HTML را انجام داد.ما در مثال زیر این کار را برای لیست DVD ها انجام می دهیم.ابتدا قسمت ایستای کار را انجام می دهیم که همان عنوان ها است:

```
<tr>
 <td>Title</td>
 <td>Category</td>
 <td>Director</td>
 <td>Price</td>
 <td>First Star</td>
 <td>Second Star</td>
</tr>
```

می بینید که یک سطر ایجاد کردم که حاوی ستونهایی با نامای مشخص شده است.

حال حلقه ی for-each را ایجاد می کنم:

```
<xsl:for-each select="/DVDList/DVD">
```

طرز کار به این صورت خواهد بود که در هر بار چرخش حلقه که با یک تگ DVD مواجه می شویم می خواهیم مقادیرآن را در ستون های جداگانه به ترتیبی که در بالا چیده شده به خروجی ببریم.برای این کار کافی است هر عنصر بازیابی شده را درداخل یک ستون(td) قرار دهیم و کل آنها را در یک سطر(tr):

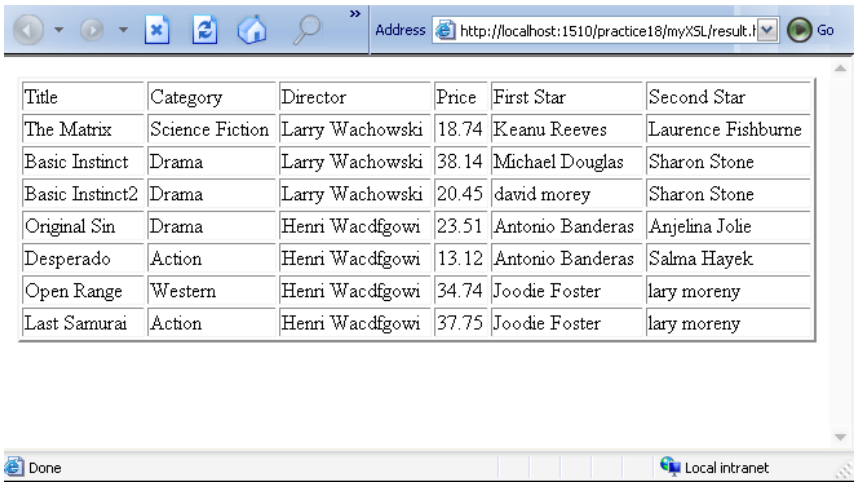
```
<tr>
    <td>
        <xsl:value-of select="Title" />
    </td>
    ...
</tr>
```

پس کد کلی به شکل زیر خواهد بود:

```
    <xsl:template match="/">
    <html>
        <body>
            <table width="100%" border="2">
                <tr>
                    <td>Title</td>
                    <td>Category</td>
                    <td>Director</td>
                    <td>Price</td>
                    <td>First Star</td>
                    <td>Second Star</td>
                </tr>
```

```
                        <xsl:for-each select="/DVDList/DVD">
                              <tr>
                                    <td>
                                          <xsl:value-of select="Title" />
                                    </td>
                                    <td>
                                          <xsl:value-of select="@Category" />
                                    </td>
                                    <td>
                                          <xsl:value-of select="Director" />
                                    </td>
                                    <td>
                                          <xsl:value-of select="Price" />
                                    </td>
                                    <td>
                                          <xsl:value-of
select="Starring/Star[1]" />
                                    </td>
                                    <td>
                                          <xsl:value-of
select="Starring/Star[2]" />
                                    </td>
                              </tr>
                        </xsl:for-each>
                  </table>
            </body>
      </html>
</xsl:template>
```

و شکل جدول خروجی به صورت زیر خواهد بود:



Title	Category	Director	Price	First Star	Second Star
The Matrix	Science Fiction	Larry Wachowski	18.74	Keanu Reeves	Laurence Fishburne
Basic Instinct	Drama	Larry Wachowski	38.14	Michael Douglas	Sharon Stone
Basic Instinct2	Drama	Larry Wachowski	20.45	david morey	Sharon Stone
Original Sin	Drama	Henri Wacdfgowi	23.51	Antonio Banderas	Anjelina Jolie
Desperado	Action	Henri Wacdfgowi	13.12	Antonio Banderas	Salma Hayek
Open Range	Western	Henri Wacdfgowi	34.74	Joodie Foster	lary moreny
Last Samurai	Action	Henri Wacdfgowi	37.75	Joodie Foster	lary moreny

پس با نحوه ی کار با تگ for-each هم آشنا شدید.جلوتر با این حلقه بیشتر کار خواهیم کرد.

حال می خواهیم کار با متغیرها و پارامترها را در XSL یاد بگیریم.

هر زبان برنامه نویسی حاوی متغیر است و XSL هم از این قاعده مستثنی نیست.در این بخش می خواهیم ۲ نوع متغیر را معرفی کنیم.اولین آنها تگ <xsl:variable> است.این تگ حاوی دو ویژگی است اولی name که نام متغیر را مشخص می کند و دومی select که مقدارش را مشخص می کند.فرمت کلی آن به صورت زیر است:

<xsl:variable name=”Var name” Select=”tag name or static name”/>

مقدار یک متغیر را در تگ <xsl:variable> می توان بدون ویژگی select نیز به آن داد:

<xsl:varibale name=”var name”>Value</xsl:variable>

در صورتی که مقدار متغیر شما عددی باشد می بایست آن را تنها درون "" قرار دهید ولی اگر رشته ای باشد حتما باید علاوه بر "" از ‘’ هم استفاده کنید:

```
<xsl:variable name="aaa" select="'ali'"/>
```

از تگ <xsl:variable> هم می توانید درون تگ <xsl:template> و هم بیرون آن استفاده کنید. که اگر آن را در بیرون ایجاد کنید به آن متغیر global می گویند.برای استفاده از یک متغیر تعریف شده از قبل می بایست کنار نامش از علامت $ استفاده کنید.

در زیر نمونه ی ساده ای از استفاده از متغیر را می بینید:

```
  <xsl:template match="/">
      <xsl:variable name="aaa" select="'ali'"/>
      <xsl:value-of select="$aaa"/>
</xsl:template>
```

در این کد در داخل تگ <xsl:template> یک متغیربه نام aaa با مقدار ali ایجاد کردیم و سپس همانجا با تگ Value-of آن را نمایش دادیم.در این کد مقدار متغیر aaa تنها در همان تگ template که توش تعریف شده قابل دسترسی است و بیرون از آن قابل دسترسی نیست.برای قابل دسترس بودن آن می بایست آن را به صورت top-level تعریف می کردیم:

```
  <xsl:variable name="aaa" select="'ali'"/>
  <xsl:template match="/">
      <xsl:value-of select="$aaa"/>
</xsl:template>
```

برای مثال در این زمینه سند xml مربوط به راهها را در نظر بگیرید و کد xsl زیر را بنویسید:

```
    <xsl:variable name="blocks">
        <xsl:copy-of select="//block"/>
    </xsl:variable>
    <xsl:template match="/">
        <xsl:copy-of select="$blocks"/>
</xsl:template>
```

آنچه در ۳ خط اول مشخص است تعریف یک متغیر به نام blocks و مقداردهی به آن است.ولی این مقداردهی دیگر یک رشته ی ساده نیست.در اینجا توسط <xsl:copy-of> یک کپی کلی از تمام تگها با نام block گرفته می شود البته همراه نام تگ و مقدارش.سپس تمام آنها در داخل متغیر blocks ذخیره می شوند.برای نمایش این تگها به همراه مقدارشان می بایست باز از تگ copy-of استفاده کنیم تا تمام تگهای موجود در متغیر blocks را به همراه مقدارشان در خروجی کپی کند.پس خروجی کد فوق به صورت زیر می شود:

```
    <?xml version="1.0" encoding="utf-8"?>
    <block>1st Street</block>
    <block>2nd Street</block>
    <block>3rd Street</block>
    <block>First Street</block>
    <block>Second Street</block>
    <block>Third Street</block>
    <block>Panorama Street</block>
    <block>Highland Plaza</block>
    <block>Hutchens Avenue</block>
    <block>Wildwood Drive</block>
    <block>Old Chimney Road</block>
<block>Carrol Circle</block>
```

اگر برای نمایش محتویات متغیر به جای تگ copy-of از value-of یا apply-template استفاده می کردیم محتویات بدون نام تگ به خروجی می رفتند و از طرف دیگر نمی توانستیم به آنها نام تگ را دستی بدهیم چون در این صورت تگی که ما به آن داده بودیم به صورت کلی روی تمام block ها قرار می گرفت در حالیکه ما برای هر مقدار block یک تگ block نیاز داشتیم.

نوع دوم تعریف متغیر استفاده از تگ <xsl:param> است.این تگ همان کاری را می کند که تگ <xsl:variable> انجام می دهد با همان تعاریف قبلی با این تفاوت که می تواند حاوی مقدار پیش فرض هم باشد یعنی می توانید برایش با ویژگی select مقداردهی لازم را انجام ندهید.می توانید مثال های بالا را بجای varuable با param نیز

انجام دهید و با نتایج مشابه روبرو می شوید.فرق دیگری که بین <xsl:param> و <xsl:variable>وجود دارد این است که مقدار متغیری مه با <xsl:variable> ایجاد می کنید قابل تغییر نیست ولی مقداری را که با <xsl:param> تعریف می کنید قابل تغییر است.این تغییر توسط تگ دیگری به نام <xsl:with-param> صورت می گیرد.این تگ تنها درون تگ apply-template قابل دسترسی است و حاوی دو ویژگی name و select است.name نام متغیریست که با <xsl:param> تعریف شده و select مقدارجدید یا ویرایش شده ی آن متغیر است.برای مثال کد زیر را که مربوط به سند xmlمربوط به راهها است در نظر بگیرید:

```
<xsl:template match="thoroughfare">
    <xsl:apply-templates select="block">
          <xsl:with-param name="precinct" select="5"/>
    </xsl:apply-templates>
</xsl:template>
<xsl:template match="block">
    <xsl:param name="precinct" select="4"/>
    <xsl:value-of select="$precinct"/>
</xsl:template>
```

در این کد تمام مقادیر تگ هایی با نام block برابر ٤ خواهند بود و نمایش داده می شوند ولی عبارت بالایی باعث می شود که مقادیر تگ هایی با نام block که فرزند thoroughfare هستند مقدارشان برابر ٥ شود.

می بینید که کاربرد <xsl:with-param> خیلی گسترده نیست.

پس ما در این بخش با دو نوع تعریف متغیر آشنا شدیم که بسیار شبیه هم بودند.

حال به بحث مرتب سازی می پردازیم.مرتب سازی عملی است که در هر زبانی باید وجود داشته باشد .XSLT نیز با تگ <xsl:sort> این امکان را فراهم کرده.

برای اعمال یک مرتب سازی معمولی بایدعبارت <xsl:sort/> را در جای درست به کار ببرید.

برای مثال در همان سند مربوط به راهها می خواهیم عمل مرتب سازی را روی مقادیر تگ های block به گونه ای انجام دهیم که خروجی به صورت زیر شود:

```
<?xml version="1.0" encoding="utf-8"?>
    <boulevard>
    <block branch="thoroughfare">1st Street</block>
    <block branch="thoroughfare">2nd Street</block>
```

```
        <block branch="thoroughfare">3rd Street</block>
        <block branch="boulevard">Carrol Circle</block>
        <block branch="thoroughfare">First Street</block>
        <block branch="boulevard">Highland Plaza</block>
        <block branch="boulevard">Hutchens Avenue</block>
        <block branch="boulevard">Old Chimney Road</block>
        <block branch="boulevard">Panorama Street</block>
        <block branch="thoroughfare">Second Street</block>
        <block branch="thoroughfare">Third Street</block>
        <block branch="boulevard">Wildwood Drive</block>
</boulevard>
```

ما قبل از مرتب سازی تگ های جدیدی با ویژگی branch ایجاد کردیم که مقدارشان نام بزرگراه یا بولواری است که کوچه یا خیابان مورد نظر از آن انشعاب پیدا کرده.مثلا خیابان 1st Street از thoroughfare انشعاب پیدا کرده.

پس قبل از مرتب سازی بایداین تگ ها را تولید کنیم.برای این کار ابتدا یک template با "match="boulevard ایجاد می کنیم چون قصد داریم تمام مقادیر درآن نمایش داده شود:

```
  <xsl:template match="boulevard">
</xsl:template>
```

سپس در داخل آن برای دسترسی به هر تگ block از //block XPath استفاده می کنیم از طرفی چون یک پیمایش روی تک تک block ها خواهیم داشت و می خواهیم عمل خاصی روی هر تگ block انجام دهیم (به آن یک ویژگی اختصاص دهیم) هیچ چیز بهتر از یک حلقه ی for-each نیست:

```
  <xsl:for-each select="//block">
</xsl:for-each>
```

در بین حلقه ی for-each می توانیم هر کاری روی تگ های block انجام دهیم.چون قصد من اضافه کردن ویژگی است می توانم از تگ <xsl:copy> استفاده کنم.زیرا این تگ در هربار چرخش حلقه تگ یک کپی از Value تگ block می گیرد و به خروجی می برد و ما می توانیم آن مقدار را در تگی محدود کنیم:

```
  <xsl:copy>
      <xsl:attribute name="branch">
            <xsl:value-of select="name(..)" />
      </xsl:attribute>
      <xsl:value-of select="."/>
</xsl:copy>
```

در اینجا تگ کپی باعث می شود محتویات تگ block همراه با نام تگ block نمایش داده شوند چون "match="//block (البته "select="//block است ولی در اینجا نقش match را دارد)است.سپس در این بین یک ویژگی به نام branch به آن اضافه می کند که مقدارش با .. مشخص شده یعنی نام تگ والد(همان بزرگراهی که خیابان مورد نظر از آن انشعاب شده) سپس مقدارش هم با . مشخص شده یعنی مقدار تگ جاری که همان تگ Block است.پس کد کلی xsl به صورت زیر شد:

```
<xsl:template match="text()" />
<xsl:template match="boulevard">
    <xsl:copy>
          <xsl:for-each select="//block">
          <xsl:copy>
                <xsl:attribute name="branch">
                      <xsl:value-of select="name(..)" />
                </xsl:attribute>
                <xsl:value-of select="."/>
          </xsl:copy>
          </xsl:for-each>
    </xsl:copy>
</xsl:template>
```

ما در این کد یک تگ <xsl:copy> دیگر روی حلقه ی for-each قرار دادیم.به این ترتیب از کل مقادیر تولید شده در حلقه ی for-each نیز یک کپی گرفته می شود.این تگ برای این است که بتوانیم مقادیر را درون تگ Boulevard قرار دهیم.

همانطور که می دانید تگ <xsl:copy> بیرونی عمل کپی را از تمامی مقادیر می گیرد(اگر ویژگی select درآن تعریف نشده باشد که در اینجا هم نشده) و سپس حاصل را با مقدار ویژگی match محدود می کند و مقادیر اضافه به صورت نوشته ی خالی باقی می مانند.از آنجایی که "match="boulevard بود یعنی مقادیر تولید شده در تگ boulevard قرار می گیرند و اضافه ها بیرون از آن هستند.برای حذف هر گونه text در صفحه (نوشته ی خالی و بدون تگ) از تابع ()text در تگ حذف استفاده کردیم.

.پس کد خروجی به شکل زیر شد:

```
<?xml version="1.0" encoding="utf-8"?>

    <boulevard>
    <block branch="thoroughfare">1st Street</block>
    <block branch="thoroughfare">2nd Street</block>
    <block branch="thoroughfare">3rd Street</block>
    <block branch="thoroughfare">First Street</block>
```

```
        <block branch="thoroughfare">Second Street</block>
        <block branch="thoroughfare">Third Street</block>
        <block branch="boulevard">Panorama Street</block>
        <block branch="boulevard">Highland Plaza</block>
        <block branch="boulevard">Hutchens Avenue</block>
        <block branch="boulevard">Wildwood Drive</block>
        <block branch="boulevard">Old Chimney Road</block>
        <block branch="boulevard">Carrol Circle</block>
</boulevard>
```

برای اعمال مرتب سازی باید جای درستی برای درج <xsl:sort/> بیابیم.این تگ را باید قبل از هر گونه اقدام برای اضافه کردن ویژگی به تگ های block انجام دهیم.پس بهترین جا دقیقا زیر تگ <xsl:for-each> است تا در هر بار چرخش حلقه ابتدا عمل مرتب سازی انجام گیرد و سپس به تگ ویژگی اضافه شود.البته خود کامپایلر هم در هیچ جا غیر از آن به شما اجازه ی درج <xsl:sort/> را نمیدهد(Vs) بنابراین به سادگی توانستید جای درست <xsl:sort/> را بیابیم.حال اگر به خروجی نگاهی بیندازید می بینید که مرتب سازی بر اساس value تگ های Block و به صورت صعودی انجام شده.به این دلیل مرتب سازی بر اساس value تگ های Block انجام شده که تگ <xsl:sort/> به مقدار match نگاه می کند که در اینجا همان select حلقه ی for-each است (//block) و از آن در می یابد که مرتب سازی بر اساس چه آیتمی صورت گیرد .

با استفاده از ویژگی select تگ <xsl:sort> می توانید عمل مرتب سازی را بر اساس والد گره تغییر دهید برای مثال کد بالا را به صورت زیر بنویسید:

```
  <xsl:template match="text()" />
      <xsl:copy>
            <xsl:for-each select="//block">
                  <xsl:sort select="name(..)"/>
                  <xsl:sort/>
                  <xsl:copy>
                        <xsl:attribute name="branch">
                              <xsl:value-of select="name(..)" />
                        </xsl:attribute>
                        <xsl:value-of select="."/>
                  </xsl:copy>
            </xsl:for-each>
      </xsl:copy>
</xsl:template>
```

ما عبارت "<xsl:sort select="name(..)" /> را بر سر <xsl:sort/> آوردیم.این کار باعث می شود که عمل مرتب سازی به صورت جداگانه توسط مقداری که ویژگی select

تگ sort دارد انجام شود.در اینجا این ویژگی حاوی نام والد گره ی جاری است.گره ی جاری که در حلقه ی for مشخص شده تگ block است پس مرتب سازی بر اساس boulevard و throughfare انجام می شود.به این صورت که عمل مرتب سازی مقادیر تگ های block بر اساس boulevard جدا و بر اساس throughfare هم جدا انجام می شود.خروجی به صورت زیر می شود:

```
<?xml version="1.0" encoding="utf-8"?>

<boulevard>
    <block branch="boulevard">Carrol Circle</block>
    <block branch="boulevard">Highland Plaza</block>
    <block branch="boulevard">Hutchens Avenue</block>
    <block branch="boulevard">Old Chimney Road</block>
    <block branch="boulevard">Panorama Street</block>
    <block branch="boulevard">Wildwood Drive</block>
    <block branch="thoroughfare">1st Street</block>
    <block branch="thoroughfare">2nd Street</block>
    <block branch="thoroughfare">3rd Street</block>
    <block branch="thoroughfare">First Street</block>
    <block branch="thoroughfare">Second Street</block>
    <block branch="thoroughfare">Third Street</block>
</boulevard>
```

همانطور که مشخص است عمل مرتب سازی به طور جداگانه برای تگ های boulevard و throughfare انجام گرفته است.این مدل مرتب سازی را مرتب سازی دسته ای می گویند و بسیار کاربردی است برای درک بیشتر به سند ساده ی xml زیر دقت کنید:

```
<?xml version="1.0" encoding="utf-8"?>
    <freezer>
    <item branch="bakery">bread</item>
    <item branch="bakery">jelly doughnuts</item>
    <item branch="bakery">rolls</item>
    <item branch="freezer">green beans</item>
    <item branch="freezer">ice cream</item>
    <item branch="freezer">peas</item>
    <item branch="freezer">pot pie</item>
    <item branch="produce">apples</item>
    <item branch="produce">bananas</item>
    <item branch="produce">kumquats</item>
</freezer>
```

و کد xsl را به صورت زیر بنویسید:

```
    <xsl:template match="bakery" />
```

```
        <xsl:template match="produce" />
        <xsl:template match="freezer">
              <xsl:copy>
                    <xsl:for-each select="//item">
                          <xsl:sort select="name(..)"/>
                          <xsl:sort/>
                          <xsl:copy>
                                <xsl:attribute name="branch">
                                      <xsl:value-of select="name(..)" />
                                </xsl:attribute>
                                <xsl:value-of select="."/>
                          </xsl:copy>
                    </xsl:for-each>
              </xsl:copy>
</xsl:template>
```

خروجی به شکل زیر می شود:

```
  <?xml version="1.0" encoding="utf-8"?>
      <freezer>
      <item branch="bakery">bread</item>
      <item branch="bakery">jelly doughnuts</item>
      <item branch="bakery">rolls</item>
      <item branch="freezer">green beans</item>
      <item branch="freezer">ice cream</item>
      <item branch="freezer">peas</item>
      <item branch="freezer">pot pie</item>
      <item branch="produce">apples</item>
      <item branch="produce">bananas</item>
      <item branch="produce">kumquats</item>
</freezer>
```

می بینید که مرتب سازی به صورت دسته ای برای تک تک دسته ها جداگانه انجام شده.با ویژگی order تگ <xsl:sort> می توانید صعودی یا نولی بودن مرتب سازی را مشخص کنید مثلا در کد بالا اگر به جای <xsl:sort/> از <xsl:sort order=”descending”/> استفاده کنید مرتب سازی نزولی انجام می شود.ویژگی دیگر تگ sort ویژگی case-order است که ترتیب مرتب سازی بر اساس حروف کوچک و بزرگ را مشخص می کند مثلا از نظر آن Car با car فرق می کند و باید ترتیبی برای آن مشخص شود مثلا اگر برای ویژگی case-order مقدار Upper-first را انتخاب کند ابتدا Car و سپس car انتخاب می شوند.ویژگی دیگر data-type است که نوع داده ای عنصر مورد نظر جهت مرتب سازی را مشخص می کند این ویژگی وقتی کاربرد دارد که عدد ها از نوع رشته باشند یا نباشند.زیرا در بعضی موارد بعضی ها اعداد را رشته در نظر می

گیرند و باید نوع داده ای را درست مقدار دهی کنند ئ در غیر این صورت این باعث مرتب سازی نادرست خواهد بود .

در کد بالا هم به جای دو خط اول می توانستید از تگ حذف و تابع text() استفاده کنید.

با استفاده از سند xml زیر ویژگی های دیگر تگ sort را چک می کنیم:

```
<doc>
    <a id="1">ali</a>
    <a id="3">reza</a>
    <a id="6">asghar</a>
    <a id="9">zahrai</a>
    <a id="2">ahmad</a>
    <a id="7">soosan</a>
    <a id="4">nemat</a>
    <a id="5">jafar</a>
    <a id="8">moahmmad</a>
    <a id="11">omid</a>
    <a id="0">kazem</a>
</doc>
```

اگر کد xsl به صورت زیر باشد

```
<xsl:template match="/">
    <xsl:for-each select="//a">
        <xsl:sort/>
        <xsl:value-of select="."/>
        <xsl:text>&#10;</xsl:text>
    </xsl:for-each>
</xsl:template>
```

به این ترتیب در هر بار چرخش حلقه مرتب سازی انجام می شود و سپس مقدار تگ جاری که مرتب شده در خروجی نمایش داه می شود.خروجی این کد اسامی مرتب شده به صورت صعودی است.

ابتدا کد xsl را به صورت زیر بنویسید:

```
<xsl:template match="/">
    <xsl:for-each select="//a">
        <xsl:sort select="@id"/>
        <xsl:value-of select="."/>
        <xsl:text>&#10;</xsl:text>
    </xsl:for-each>
</xsl:template>
```

طبق کد بالا با ویژگی select حلقه ی for-each مشخص می کنیم که عمل مرتب سازی بر روی چه تگی صورت گیرد که میبینید بر اساس تگ a.ولی با ویژگی select تگ sort توانستم آیتمی که مرتب سازی بر اساس آن انجام گیرد را ویژگی id در نظر

بگیرم.ولی یک مشکل اینجا وجود دارد و آن هم این است که مرتب سازی به درستی انجام نمی گیرد زیرا مقدار ویژگی id در سند xml عددی است ولی اینجا به صورت رشته ای آن را در نظر می گیرد.برای اصلاح آن از ویژگی Data-type استفاده می کنیم و مقدارش را number می دهیم:

```
<xsl:sort select="@id" data-type="number"/>
```

حال به مثالی در زمینه ی key و variable توجه کنید ولی قبل از آن تابع document() را معرفی می کنیم.این تابع باعث می شود ما به سند xml دیگری غیر از آن سندی که عمل تبدیل را با آن انجام میدهیم دسترسی داشته باشیم.این تابع به صورت زیر نوشته می شود:

document('address of xml document')

مقدار برگشتی آن نیز تمام گره ها ی سند مقصد خواهد بود.برای مثال فرض کنید یک سند به نام first.xml به صورت زیر داریم:

```
<?xml version="1.0" encoding="utf-8" ?>
<usstates>
    <western>
          <usstate>Arizona</usstate>
          <usstate>California</usstate>
          <usstate>Idaho</usstate>
          <usstate>Montana</usstate>
          <usstate>Nevada</usstate>
          <usstate>Oregon</usstate>
          <usstate>Washington</usstate>
          <usstate>Utah</usstate>
    </western>
</usstates>
```

این سند بیانگر ایالت های معروف آمریکا است.سند دوم را با نام second.xml به صورت زیر در نظر بگیرید:

```
<?xml version="1.0" encoding="utf-8" ?>
<capitals>
    <capital usstate="Arizona">Phoenix</capital>
    <capital usstate="California">Sacramento</capital>
    <capital usstate="Idaho">Boise</capital>
    <capital usstate="Montana">Helena</capital>
    <capital usstate="Nevada">Carson City</capital>
    <capital usstate="Oregon">Salem</capital>
    <capital usstate="Washington">Olympia</capital>
    <capital usstate="Utah">Salt Lake City</capital>
```

```
</capitals>
```

در اینجا پایتخت هر ایالت مشخص شده.

کد xsl را به صورت زیر بنویسید:

```
  <xsl:template match="/">
      <xsl:apply-templates
select="document('second.xml')/capitals/capital[3]"/>
</xsl:template>
```

در این کد با عبارت document(‘second.xml’) ما به سند second.xml دسترسی داریم و با /capitals/capital[3] به value سومین تگ capital که Boise است دسترسی داریم.یادتان باشد اگراز شی XSLCompiledTransform برای تبدیل استفاده می کنید بایدکد آن را به صورت زیر اصلاح کنید:

```
        Dim xmlFile As String = Server.MapPath("first.xml")
        Dim xslFile As String = Server.MapPath("firsting.xsl")
        Dim htmlFile As String = Server.MapPath("result.htm")
        Dim a As New XsltSettings
        a.EnableDocumentFunction = True
        Dim transform As New XslCompiledTransform
        transform.Load(xslFile, a, New XmlUrlResolver())
      transform.Transform(xmlFile, htmlFile)
```

در این کد با شی xslsettings توانستم EnableDocumentFunction را فعال کنم و برای اضافه کردن آن به شی XSLCompiledTransform کافیست آن شی xslsettings را به عنوان آرگومان دوم به متد loadش بدهیم و با دادن New XmlUrlResolver در حقیقت مشکل وجود تابع document را حل کردیم.

هدف من در این مثال این است که بتوانم دو سند xml بالا را ترکیب کنم به گونه ای که با دادن یک مقدار به یک متغیر, که در اینجا نام ایالت است, پایتخت آن از یک سند دیگر بازیابی شود و در کنار نامش نشان داده شود مثلا اگر من مقدار Arizona را به متغیر بدهم خروجی به شکل زیر شود:

```
Phoenix, Arizona
```

حال بر طبق اسناد xml بالا به کد xsl زیر که ترکیبی از مقادیر دو سند است دقت کنید:

```
    <xsl:key name="Capital" match="capitals/capital" use="@usstate"/>
    <xsl:key name="State" match="usstates/western/usstate" use="."/>
    <xsl:param name="cr">Arizona</xsl:param>
    <xsl:template match="/">
          <xsl:apply-templates select="document('second.xml')/capitals"/>
          <xsl:text>, </xsl:text>
          <xsl:value-of select="key('State', $cr)"/>
    </xsl:template>
```

```
      <xsl:template match="capitals">
            <xsl:value-of select="key('Capital', $cr)"/>
</xsl:template>
```

این کد بسیار ساده است.در این کد از تکنیک میکس استفاده شده.

در دو خط اول دو key با نام های Capital & state تعریف شده و اولی از با مقادیر تگ capital کار دارد و از ویژگی usstate استفاده می کند و دومی با فرزند های تگ usstate استفاده می کند و از تگ جاری استفاده می کند.فعلا هیچ چیز مشخص نیست یعنی این الگوهایی که در ویژگی matchشان مشخص شده فعلا اعتبار ندارد و صرفا الگو هستند.سپس یک متغیر به نام cr با مقدار Arizona تعریف کردیم.پس تا اینجا همه توضیح واضحات بود.

سپس در داخل تگ <xsl:template> و به وسیله ی <xsl:apply-template> سند second.xml را به کار اضافه کردیم.این کار باعث می شود تمام مقادیر تگ های فرزند تگ capitals نمایش داده شوند:

```
<xsl:apply-templates select="document('second.xml')/capitals"/>
```

و با عبارت زیر :

```
<xsl:value-of select="key('State', $cr)"/>
```

نوشته ی Arizona به سایر نوشته ها اضافه می شود.به این صورت که key با نام State طبق تعریف به usstates/western/usstate اشاره می کند و با .(مقدار گره ی جاری) مقدار $cr را که برابر Arizona بود را مقایسه می کند و نام Arizona را به خروجی می برد.همانطور که دیدید در سند second.xml چندین تگ همنام با نامهای usstate وجود داشتند و اگر می خواستیم key دوم را به صورت زیر تعریف کنیم به مشکل چند مقداری برمی خوردیم:

```
<xsl:key name="State" match="usstates/western " use="usstate "/>
```

تا اینجا خروجی به صورت زیر است(تگ صورتی رنگ در خروجی نخواهد بود):

```
  <capitals>
      Phoenix
      Sacramento
      Boise
      Helena
      Carson City
      Salem
      Olympia
      Salt Lake City</capitals>
, Arizona
```

همانطور که میبینید ۸ مقدار اول حاصل عبارت document('second.xml')/capitals هستند پس یک تگ (در بالا به رنگ صورتی مشخص است) باید به صورت بالا بینشان باشد(مخصوص عمل میکس است).این تگ به خروجی نمیرود ولی قبل از رفتن به خروجی این تگ وجود دارد تا شاید عمل میکسی بخواهد انجام شود.پس دقت کنید این تگ صورتی را من خودم جهت فهم بیشتر ایجاد کردم.

یادتان باشد الگوهای مشخص شده در هر دو key در صورت عدم وجود هم باعث ایجاد خطا نمی شود و از طرف دیگر در این کد ازusstates/western/usstate استفاده کردیم که مربوط به سند xml اصلی(اول) می شد و چنانچه از دومی هم استفاده کنید چون با تابع document آن را اضافه کردیم مشکلی پیش نمی آید همانطور که در ادامه کد زیر را نوشتیم:

```
  <xsl:template match="capitals">
            <xsl:value-of select="key('Capital', $cr)"/>
</xsl:template>
```

این کد باعث ایجاد عمل میکس می شود.خروجی این کد به تنهایی به صورت زیر تعیین می شود.

key با نام capital به آدرس capitals/capital اشاره می کند و مقدار $cr را که Arizona است را با ویژگی usstate تگ capital مقایسه می کند و هر کدام که همنام بود برمی گرداند که می دانیم phoenix است.حال به مقدار ویژگی match دقت کنید.این مقدار capitals است.پس در محتویات موجود به دنبال تگی با نام capitals می گردد و به محض یافتن آن مقدار phoenix را به جای مقدار قبلیش جایگزین می کند.این تگ در خروجی بالا به رنگ صورتی نشان داده شده.پس کافیست به جای مقادیر آن مقدار جدید یعنی phoenix قرار گیرد وسپس خروجی به شکل زیر می شود:

```
Phoenix, Arizona
```

در این مثال خروجی اهمیتی ندارد بلکه آنچه که اهمیت دارد ترکیب دو سندxml و استفاده از key و متغیر در کنار هم بود.همینطور باز هم به شکل عمیق تری با عمل میکس آشنا شدید.

در حال حاضر به بحث برنامه نویسی شرطی در XSL می پردازیم.

در ابتدا با تگ <xsl:if> آشنا می شویم.تگ if به معنای اگر است و چک می کند وجود یا عدم وجود یک مقدار را.این کار را با ویژگی test انجام می دهد.در حقیقت مقداری که

ویژگی test می پذیرد عبارت شرطی است.مثلا test="@id='3443'" یعنی ایا مقدار ویژگی id برابر ۳۴۴۳ هست یا نه.و یا شرط زیرکه مثل شرط های یک طرفه ای مثل if a then است که تنها true یا false برایشان مهم است:

```
<xsl:template match="/">
    <xsl:if test="//boulevard">
          <xsl:text>Found a Boulevard!</xsl:text>
    </xsl:if>
</xsl:template>
```

در این کد چک شده آیا //boulevard وجود دارد یا نه.اگر وجود داشت درخروجی نوشته ی Founf a Boulevard را بنویس.

در این مثال که فرم ساده ای دارد تنها نکته اش استفاده از چند if است.این مثال در رابطه با سند xml مربوط به راه هاست:

```
<xsl:template match="text()"/>
<xsl:template match="thoroughfare">
    <xsl:if test="@name">
          <xsl:copy>
            <name>
                <xsl:value-of select="@name"/>
            </name>
            <xsl:if test="@name[contains(., 'Drive')]">
                <drive>
                      <xsl:if test="block">
                            <xsl:copy-of select="block"/>
                      </xsl:if>
                </drive>
            </xsl:if>
          </xsl:copy>
    </xsl:if>
</xsl:template>
```

در این کد مبنای کار روی تگ <xsl:copy> است.عمل کپی ازValue تمام تگهای سند گرفته می شود و سپس به جای مقادیری که از تگ thoroughfare کپی گرفته می شود مقادیری که در داخل تگ <xsl:copy> هستند قرار می گیرد:

```
<?xml version="1.0" encoding="utf-8"?>
          <thoroughfare><name></name></thoroughfare>
          <thoroughfare><name>Whitesburg       Drive</name><drive><block>1st
Street</block><block>2nd                              Street</block><block>3rd
Street</block></drive></thoroughfare>
          <thoroughfare><name>Bankhead</name></thoroughfare>
          Panorama Street
          Highland Plaza
```

```
            Hutchens Avenue
            Wildwood Drive
            Old Chimney Road
Carrol Circle
```

مقادیر تگ های thoroughfare به صورت زیر تولید می شوند.

ابتدا مقدار ویژگی name تگ thoroughfare در داخل تگ <name> قرار می گیرد:

```
<xsl:value-of select="@name"/>
```

البته قبل از آن چک می شود آیا تگ thoroughfare حاوی ویژگی به نام name هست یا نه:

```
  <xsl:if test="@name">
```

سپس چک می شود اگر مقدار ویژگی name حاوی رشته ی Drive هست یا نه.اگر بود تگی به نام Drive برایش ایجاد می شود و در درونش چک می شود آیا تگ thoroughfare حاوی فرزندی به نام block هست یا نه اگر بود از مقدار آن تگ کپی بگیر یعنی مقدارش را درون تگ Drive قرار بده.با تگ حذف نیز نوشته های خالی که در خروجی بالا هم آنها را میبینید حذف می کنیم.

پس خروجی به شکل زیر خواهد بود:

```
  <?xml version="1.0" encoding="utf-8"?>
  <thoroughfare>
    <name>Whitesburg Drive</name>
    <drive>
      <block>1st Street</block>
      <block>2nd Street</block>
      <block>3rd Street</block>
    </drive>
  </thoroughfare>
  <thoroughfare>
    <name>Bankhead</name>
</thoroughfare>
```

همانطور که می بینید در ابتدا به اولین تگ thoroughfare که می رسیم چک می شود آیا حاوی ویژگی هست یا نه چون نیست پس اصلا کاری صورت نمی گیرد.به دومین تگ thoroughfare که می رسیم می بینیم که حاوی ویژگی name هست پس مقدارش را در تگ name می نویسد سپس چک می شود آیا مقدار این ویژگی حاوی رشته ی Drive است یا نه چون هست سپس چک می شود آیا حاوی فرزندی با نام block هست یا نه چون ۳ فرزند با این نام دارد آن ۳ فرزند را در داخل تگ Drive به خروجی میبرد.این

یک مثال نسبتا کامل در زمینه ی استفاده از تگ <xsl:if> بود.می توانستید کد بالا را با حلقه ی for-each روی تک تک تگ های thoroughfare هم بنویسید:

```
<xsl:template match="text()"/>
<xsl:template match="/">
      <xsl:for-each select="//thoroughfare">
      <xsl:if test="@name">
            <xsl:copy>
                  <name>
                        <xsl:value-of select="@name"/>
                  </name>
                  <xsl:if test="@name[contains(., 'Drive')]">
                        <drive>
                              <xsl:if test="block">
                                    <xsl:copy-of select="block"/>
                              </xsl:if>
                        </drive>
                  </xsl:if>
            </xsl:copy>
      </xsl:if>
      </xsl:for-each>
</xsl:template>
```

می بینید که برای دسترسی به تگ هایی با نام thoroughfare از // thoroughfare استفاده کردیم.

حال می رسیم به معرفی ۳ تگ شرطی بعدی که معرفی جداگانه و استفاده ی همزمان دارند:

<xsl:choose> , <xsl:when> , <xsl:otherwise>.

کاربرد این ۳ تگ در کنار هم در عبارت زیر واضح است:

```
<xsl:choose>
<xsl:when test="...">
...
</xsl:when>
<xsl:when test="...">
...
</xsl:when>
<xsl:otherwise>
...
</xsl:otherwise>
```

همانطور که مشخص است با این ۳ تگ می توانید انتخابهای شرطی چند گانه داشته باشید دقیقا مثل ساختار if...else if...else .

در زیر یک مثال روی سند xml مربوط به DVD ها را با هم می بینیم.

در ابتدا تگ Template را به صورت زیر می نویسم.علامت * به معنای این است که تمام فرندان و نوادگان می توانند در این تگ قابل دسترسی باشند و پیمایش شوند.پس نیازی به حلقه ی for-each در اینجا نیست چون خود match این کار را انجام می دهد:

```
<xsl:template match="*"></xsl:template>
```

سپس شرط خود را به این صورت بیان می کنم که به هر تگی که رسیدی اگر نامش Title بود بنویس Found a Title اگر نامش Price بود بنویس Found a Price و اگر نامش Director و Star بود به همین ترتیب.حال در انتها اگر هم تگی وجود داشت که ما مورد بررسی قرارش ندادیم بنویس Found a نام آن تگ.عمل چک کردن تگ ها را نیز تابع name() انجام می دهد:

```
  <xsl:choose>
      <xsl:when test="name() = 'Title'">
            <xsl:text>Found a Title!
              </xsl:text>
      </xsl:when>
      <xsl:when test="name() = 'Director'">
            <xsl:text>Found a Director!
              </xsl:text>
      </xsl:when>
      <xsl:when test="name() = 'Price'">
            <xsl:text>Found a Price!
              </xsl:text>
      </xsl:when>
      <xsl:when test="name() = 'Star'">
            <xsl:text>Found a Star!
              </xsl:text>
      </xsl:when>
      <xsl:otherwise>
            <xsl:text>Found a </xsl:text>
            <xsl:value-of select="name()"/>
            <xsl:text>.
        </xsl:text>
      </xsl:otherwise>
</xsl:choose>
```

حال می خواهیم به محض یافتن مثلا تگ Title مقدار آن را نیز در کنارش قرار دهد.برای این کار کافی است از تگ <xsl:apply-template/> در انتهای کار استفاده کنیم.البته ما می دانیم این تگ تمام مقادیر تگ هایی را که الگویشان در match مشخص شده را به خروجی می بردولی در اینجا ابتدا سرط ها چک می شوند و مقادیری که با تگ

های Text به خروجی فرستادیم مثل Found a Price! دقیقا روی مقادیری قرار خواهند گرفت که <xsl:apply-template/> آنها را به خروجی می برد.به این دلیل <xsl:apply-template/> را در انتها قرار دادم که ترتیب در خروجی رعایت شود یعنی مثلا اول بنویسد Found aTitle! و سپس بنویسد The Matrix.خوشبختانه به دلیل عملکرد درست تگ <xsl:choose> تمام تگ های text سر جای اصلی نوشته ها را چاپ می کنند.نمونه ای از خروجی را در زیر می بینید:

```
Found a DVD.
   Found a Title!
        The Matrix
   Found a Director!
        Larry Wachowski
   Found a Price!
        18.74
   Found a Starring.
         Found a Star!
        Keanu Reeves
         Found a Star!
      Laurence Fishburne
```

یک نکته ی مهم این است که همه جا لازم نیست برای پیمایش روی یک دسته گره حتما از تگ for-each استفاده کنید.به جای آن کافی است مقداردهی درستی به match کنید.مثلا در سند مربوط به DVD ها برای پیمایش هر تگ DVD می توانید از عبارت /DVDList/DVD در match استفاده کنید:

```
<xsl:template match="/DVDList/DVD">
    <xsl:value-of select="Title"/>
    <xsl:text>---</xsl:text>
    <xsl:value-of select="Price"/>
    <xsl:text>---</xsl:text>
    <xsl:value-of select="Director"/>
</xsl:template>
```

پیمایش من در این کد روی تک تک DVD هاست.حال در هر تگ DVD هم عناصری وجود دارد که من با تگ Value-of و دادن نام آن عناصر به ویژگی selectش توانستم مقادیرش را در هر تگ DVD نمایش دهم خروجی کد فوق به صورت زیر است:

```
<?xml version="1.0" encoding="utf-8"?>
    The Matrix---18.74---Larry Wachowski
      Basic Instinct---38.14---Larry Wachowski
    Basic Instinct2---20.45---Larry Wachowski
    Original Sin---23.51---Henri Wacdfgowi
    Desperado---13.12---Henri Wacdfgowi
```

```
      Open Range---34.74---Henri Wacdfgowi
Last Samurai---37.75---Henri Wacdfgowi
```

برای خروجی بالا می توانستیم از حلقه ی for هم استفاده کنیم ولی این کار را بدون آن و باویژگی match هم انجام دادم.حال اگر مثلا match="/" باشد آنگاه دیگر پیمایشی در کار نیست پس باید الگوهای مناسبی بسته با خواسته ی خود به match و یا به ویژگی select تگ های template & value-of بدهید.پس اگر match حاوی نام تگی باشد که به کررات در سند تکرار شده یکی یکی آن ا پیمایش می کند ولی اگر حاوی تنها یک تگ باشد مشخصا پیمایشی در کار نیست.از طرف دیگر با مقداردهی های مختلف به match دست شما در به کار بردن تگهایی نظیر template & value-of بسته خواهد بود پس همه چیز به خودتان بستگی دارد. این نکته در کل مثال های ما در زمینه ی xsl صحت دارد.

تگ بعدی که لازم می دانم آن ا معرفی کنم تگ <xsl:number> است که برای شماره بندی مقادیر بازیابی شده کاربرد دارد.در مثال زیر من هرچه Value تگ block است را بازیابی کردم(به وسیله ی match="block") و برای شماره بندی از تگ <xsl:number> استفاده کردم:

```
  <xsl:template match="//block">
            <xsl:number level="any"/>
            <xsl:text> - </xsl:text>
            <xsl:value-of select="."/>
            <xsl:text>&#10;</xsl:text>
      </xsl:template>
<xsl:template match="text()"/>
```

ویژگی level را any قرار دادم تا شمارش به طور global انجام شود خروجی به صورت زیر می شود:

```
  <?xml version="1.0" encoding="utf-8"?>
  1 - 1st Street
  2 - 2nd Street
  3 - 3rd Street
  4 - First Street
  5 - Second Street
  6 - Third Street
  7 - Panorama Street
  8 - Highland Plaza
  9 - Hutchens Avenue
  10 - Wildwood Drive
  11 - Old Chimney Road
12 - Carrol Circle
```

اگر مقدار level=”single” باشد آنگاه شمارش دسته ای صورت می گیرد به این صورت که هر تعداد تگ block که فرزند والد خاصی باشند جدا شمارش می شوند و خروجی به صورت زیر می شود:

```
  1 - 1st Street
  2 - 2nd Street
  3 - 3rd Street
  1 - First Street
  2 - Second Street
  3 - Third Street
  1 - Panorama Street
  2 - Highland Plaza
  3 - Hutchens Avenue
  4 - Wildwood Drive
  5 - Old Chimney Road
6 - Carrol Circle
```

با ویژگی format تگ <xsl:number> می توانید فرمت شمارنده را مشخص کنید.مقدار پیش فرض آن عددی است ولی مثلا اگر format=”a” بدهید بر اساس حروف الفبا شمارش انجام می گیرد.a-b-c...در صورتی که تعداد بیش از حروف الفبا شد از دو حرف استفاده می کند مثلا بعد از z نوبت aa و سپس ab خواهد بود.اگر مقدار ویژگی format را برابر i	 در نظر می گرفتید شمارش به صورت یونانی صورت می گیرد....i – ii- iii-iv-.

در تگ <xsl:template> یک ویژگی وجود دارد به نام name. این ویژگی یک نام به تگ template مربوطه اختصاص می دهد.حال این نام به چه دردی می خورد؟

یک تگ در XSL وجود دارد به نام <xsl:call-template> که یک template را از جایی دیگر صدا می زند و این صدا زدن بر اساس نامی است که به تگ template اختصاص داده بودیم:

<xsl:call-template name=”name of <xsl:template>”/>

برای مثال سند مربوط به DVD ها را در نظر بگیرید .حالا می خواهیم یک بازیابی ساده از آن توسط xsl انجام دهیم.قبل از آن دو template جداگانه که به ترتیب ویژگی های id و category تگ DVD را بازیابی می کند را ایجاد می کنیم:

```
<xsl:template match="DVD" name="iid">
    <xsl:value-of select="@id"/>
    <xsl:text>&#10;</xsl:text>
</xsl:template>
```

```
  <xsl:template match="DVD" name="cat">
      <xsl:value-of select="@Category"/>
      <xsl:text>&#10;</xsl:text>
</xsl:template>
```

نام template اول iid و نام template دوم cat است.حال کد xsl اصلی را به صورت زیر می نویسم:

```
  <xsl:template match="DVDList/DVD">
      <xsl:value-of select="Title"/>
      <xsl:text>&#10;</xsl:text>
      <xsl:value-of select="Director"/>
      <xsl:text>&#10;</xsl:text>
      <xsl:value-of select="Price"/>
      <xsl:text>&#10;</xsl:text>
      <xsl:text>This DVD Id :</xsl:text>
      <xsl:call-template name="iid"/>
      <xsl:text>This DVD Category :</xsl:text>
      <xsl:call-template name="cat"/>
</xsl:template>
```

در این کد به هر تگ DVD که می رسیم Director Title وPrice ش را به خروجی می بریم و سپس با تگ <xsl:call-template/> یک template با نام iid را صدا می زنم.به این ترتیب به تگ template مربوطه رجوع می شود و id تگ DVD که نوبت بررسیش است بدست آمده و در خروجی چاپ می شود و به همین ترتیب برای <xsl:call-template/> با نام cat.

آخرین بحثی که در زمینه ی xsl مطرح می کنم قالب بندی اعداد با تابع format-number() است.

این تابع دو آرگومان ورودی دارد که اولی عدد مورد نظر و دومی قالبی برای نمایش آن است.علامت # برای عددیست که در صورت صفر بودن نمایش داده نمی شود و علامت ۰ برای عددی است که حتی اگر صفر هم باشد باید نمایش داده شود.علامت . برای جدا کردن قسمت صحیح و اعشاری به کار می رود.می توانید از علامت هایی چون $ و % نیز در قالب خود استفاده کنید:

format-number('3.4135125','#.##')=3.41

format-number('3.41','$#.##')=$3.41

format-number('0.34','%##')=%34

خوب بحث ما در رابطه با XSLT به پایان رسید.در این بحث با خیلی چیزها در XSLT آشنا شدیم و بحث ما فراتر از مقدمات آن بود حال به ادامه ی کار با XML در ASP.NET می پردازیم.

در این بخش می خواهیم با Bind کردن داده های xml در صفحات aspx صحبت کنیم.تا به حال با کد نویسی پشت صحنه خواندن و نوشتن داده های xml را آموختیم و حال می خواهیم از کنترل XmlDataSource در این راستا استفاده کنیم.این کنترل داده ها را نسبت به کنترل های sqlDataSource و ObjectDataSource بسیار سریع تر بازیابی می کند.از طرف دیگر داده های xml به صورت سلسله مراتبی هستند و می توانند تعداد بی شماری level داشته باشند در حالیکه داده های sqldatasource مبتنی بر جدولTabular() هستند.برای استفاده از کنترل XmlDataSource می توان به صورت زیر عمل کرد:

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server" DataFile="myxml.xml"
/>
```

در این کد برای مشخص کردن منبع فایل xml برای کنترل XmlDataSource از صفت DataFile آن استفاده کردم و آدرس و نام سند xml خود را به آن دادم.myxml نام همان سند مربوط به DVD هاست.حالا اگر یک GridView ایجاد کنید و منبع داده اش را XmlDataSource بالا بدهید به صورت زیر:

```
    <asp:GridView                ID="gridview1"                runat="server"
DataSourceID="xmldatasource1" >
</asp:GridView>
```

آنگاه نتیجه ای عجیب به صورت زیر می گیرید:

ID	Category
3221	Science Fiction
3222	Drama
3223	Drama
3224	Drama
3225	Action

3 226	Western

می بینید که تنها ویژگی ها در GridView نمایش داده شده اند.پس نتیجه می گیریم که اگر قرار باشد به همین سادگی عمل کنید نمی توانید مقادیر دلخواه خود را بازیابی کنید.در ضمن اگر تگ های داخلی مثل Starring و یا Titleهم حاوی ویژگی بودند ویژگی آنها در GridView نمایش داده نمی شد و تنها ویژگی های اولین سطح نمایش داده می شوند.پس می بینیم که اصلا به نتیجه ی دلخواه خود نرسیدیم و عملا عمل Bind تنها از گره های سطح اول همچون DVD صورت می گیرد.

برای جلوگیری از مشکل بالا و پیش آمدن مشکلات احتمالی دیگر (که ممکن است در بازیابی از اسناد پیچیده رخ دهد) راههای مختلفی وجود دارد مثل استفاده از XPath در Bind کردن داده ها.

همانطور که از تابع Eval() در Bind کردن مقادیر موجود در جداول sql استفاده می کردید در اینجا هم با متد XPath می توانید این کار را انجام دهید.برای این کار کافی است در تگ <%...#%> از XPath استفاده کنید.آرگومان ورودی که به آن می دهید نیز می تواند نام تگی که می خواهید Valueش را نمایش دهید و یا الگوی مورد نظر برای بازیابی باشد.هنگامی که از آن استفاده می کنید این نکته را به خاطر داشته باشید که مثلا در اینجا می توانید یک راست XPath("Title") را به کار ببرید و نیازی نیست که به جای Title از /DVDList/DVD/Title استفاده کنیم زیرا Xmldatasource تگ top-level را DVD در نظر می گیرد نه تگ DVDList یعنی تشخیص می دهد تگ ریشه DVDList است و تگ پس از آن یعنی DVD را به عنوان مبدا در نظر می گیرد و برای بازیابی مقدار تگ Title هم راست نامش را به کار می بریم و اگر مثلا خواستیم بازیگران را به خروجی ببریم می بایست از Starring/Star استفاده کنیم چون Star فرزند DVD نیست.

در مثال زیر برای Bind کردن تگ ها در GridView من از TemplateField استفاده کردم:

```
    <asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="myxml.xml" />
    <asp:GridView ID="gridview1" runat="server" AutoGenerateColumns="false"
DataSourceID="xmldatasource1" >
     <Columns>
     <asp:TemplateField HeaderText="Title">
```

```
        <ItemTemplate>
        <%# XPath("Title") %>
        </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Director">
        <ItemTemplate>
        <%# XPath("Director") %>
        </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Price">
        <ItemTemplate>
        <%#XPath("Price")%>
        </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Starrs">
        <ItemTemplate>
        <%#XPath("Starring/Star[1]")%><br />
        <%#XPath("Starring/Star[2]")%><br />
        </ItemTemplate>
        </asp:TemplateField>
        </Columns>
</asp:GridView>
```

نحوه ی دیگر Bind کردن مقادیر استفاده از ویژگی XPath خود کنترل XmlDataSource است.به این ترتیب که الگوی مورد نظر خود را به این ویژگی بدهید .البته نکته ی مهم در اینجا این است که در این الگو باید از آدرس درست تگ ها و ویژگی ها استفاده کنید زیرا ویژگی XPath همزمان با DataSource به XmlDataSource داده می شود و الگویXPath وقتی به XmlDataSource می رسد که هنوز شناختی از تگ های سند ندارد بنابراین برای بازیابی تگ Title باید /DVDList/DVD/Title را بنویسید.در ضمن در اینجا برای Bind کردن آن در GridView وقتی از XPath استفاده می کنید باید از تگ جاری (یعنی یک نقطه .)استفاده کنید تا متد XPathی که در تگ <..#%> به کار می رود تنها عمل نمایش تگی را برعهده داشته باشد که با ویژگی XPath کنترل XmlDataSource مشخص کرده بودیم.مثال زیر بیانگر این مطلب است که بازیگران فیلم ها را در GridView نمایش می دهد:

```
      <asp:XmlDataSource    ID="xml1"    runat="server"    DataFile="myxml.xml"
XPath="/DVDList/DVD/Starring/Star"></asp:XmlDataSource>
     <asp:GridView      ID="grid"      Runat="server"      DataSourceID="xml1"
AutoGenerateColumns="false">
     <Columns>
     <asp:TemplateField HeaderText="Star List">
```

```
      <ItemTemplate>
      <%# Xpath(".") %><br/>
      </ItemTemplate>
      </asp:TemplateField>
      </Columns>
   </asp:GridView>
```

از ویژگی Xpath کنترل XmlDataSource می توان استفاده های مختلفی کرد مثلا می توان لیستی از Id ها را در یک DropDownList نگهداری کرد تا با انتخاب هر یک از آنها مشخصات فیلم نمایش داده شوند.نکته اینجاست که مثلا برای بازیابی عنوان DVD باید الگوی XPath ما به صورت زیر باشد:

XPath="/DVDList/DVD[@ID=" & DropDownList1.SelectedValue & "]/Title"

به کار بردن اینگونه الگوها که از مقدار کنترل های دیگر بدست می آید در صفحه ی aspx امکان پذیر نیست بنابراین باید از آنها در کد پشت صحنه استفاده کنیم پس نباید یک راست از ویژگی XPath کنترل XmlDataSource در صفحه ی aspx استفاده کنیم.برای استفاده از Xpath در کد پشت صحنه باید از متد Xpath کنترل XmlDataSource استفاده کنیم.

پس ابتدا یک DropDownList با لیستی از id های موجود در سند xml ایجاد می کنیم:

```
      <asp:DropDownList ID="DropDownList1" runat="server">
      <asp:ListItem Value="3221"></asp:ListItem>
      <asp:ListItem Value="3222"></asp:ListItem>
      <asp:ListItem Value="3223"></asp:ListItem>
      <asp:ListItem Value="3224"></asp:ListItem>
      <asp:ListItem Value="3225"></asp:ListItem>
      <asp:ListItem Value="3226"></asp:ListItem>
   </asp:DropDownList>
```

البته می توان این گونه کار های پیش پا افتاده مثل مقدار دهی یک Drop DownList را با JavaScript انجام داد مثلا در بالا به جای DropDown List ازتگ Html به نام Select استفاده می کردیم.

یک دکمه هم ایجاد می کنیم تا با انتخاب آیتمی از DropDownList و زدن آن دکمه برای مثال عنوان فیلم نمایش داده شود:

```
<asp:Button ID="button1" runat="server"  Text="Go To DVD Title"/>
```

سپس کنترل XmlDatasource را به صورت زیر تعریف می کنیم:

```
   <asp:XmlDataSource ID="xml1" runat="server"
DataFile="myxml.xml"></asp:XmlDataSource>
```

دقت کنید که من صفت XPath را در اینجا مقداردهی نکردم و این کار را در کد پشت صحنه انجام خواهم داد.

سپس از آنجا که از ویژگی XPath کنترل XmlDatasource استفاده می کنیم در GridView از . به عنوان آرگومان ورودی متد XPath استفاده می کنیم تا مقدار تگ جاری را نمایش دهد که آن هم از کنترل XmlDatasource بازیابی می شود:

```
<asp:GridView ID="grid" Runat="server" AutoGenerateColumns="false">
   <Columns>
   <asp:TemplateField HeaderText="DVD">
   <ItemTemplate>
   <%# Xpath(".") %><br />
   </ItemTemplate>
   </asp:TemplateField>
   </Columns>
 </asp:GridView>
```

یادتان باشد که من در اینجا منبع داده را برای GridView مشخص نکردم زیرا می خواهم این کار را در کد پشت صحنه انجام دهم.

در نهایت به رویداد کلیک دکمه رفته و ابتدا الگو را تعیین کنید:

```
xml1.XPath = "/DVDList/DVD[@ID=" & DropDownList1.SelectedValue & "]/Title"
```

نام کنترل XmlDatasource من در اینجا xml1 است و با خصوصیت XPath آن توانستم الگو را تعیین کنم.الگو نیز بسیار مشخص است.سپس باید منبع داده ی کنترل GridView را همان XmlDatasource در نظر بگیریم وعمل Bind کردن را انجام دهیم:

```
        grid.DataSource = xml1
      grid.DataBind()
```

من یک دکمه ی دیگر هم برای بازیابی ستاره های یک DVD ایجاد کردم و رویداد کلیکش را صورت زیر می نویسم:

```
    Protected Sub button3_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles button3.Click
        xml1.XPath = "/DVDList/DVD[@ID=" & DropDownList1.SelectedValue &
"]/Starring/Star"
        grid.DataSource = xml1
        grid.DataBind()
  End Sub
```

پس در این مثال یک بار دیگر آموختیم که در کد پشت صحنه هم می توان به تک تک ویژگی های کنترل ها دسترسی داشت به همان صورتی که در اینجا من عمل اختصاص

منبع داده ای به کنترل GridView و همچنین عمل Bind کردن را در کد پشت صحنه انجام دادم.

پس تا اینجا نکته ای که آموختیم این بود که برای نمایش Text یک تگ در GridView حتما باید از templteField استفاده کنید خودتان برای ایجاد ستون برایش اقدام کنید.

حال می خواهیم طریقه ی ایجاد GridView های تودرتو را برای منبع داده ای XmlDataSource انجام دهیم.چنانچه قبل نحوه ی ایجاد GridView های تودرتو را برای منبع داده ای sqldatasource انجام داده بودیم ولی انجام آن کمی پیچیده بود در حالی که انجام این کار برای منبع داده ای XmlDataSource به نسبت ساده تر است.حتما یادتان هست که در ایجاد GridView های تودرتو را برای منبع داده ای sqldatasource یک GridView والد و یک GridView فرزند داشتیم و با دو sql query آنها را پر می کردیم و sql query منبع داده ای برای GridView فرزند حاوی یک پارامتر بود که آن پارامتر را از مقادیر Bind شده در GridView والد بدست می آوردیم و...

در اینجا کار به نسبت ساده تر است و دیگر نیازی به پارامتر نداریم و حتی نیاز به دو XmlDataSource جداگانه هم نمی باشد.در مثال زیر می خواهیم جزییات هر DVD را به غیر از بازیگرانش نمایش دهیم و بازیگران را در GridView جداگانه نمایش دهیم.پس ابتدا یک XmlDataSource ایجاد می کنیم:

```
<asp:XmlDataSource ID="xml1" runat="server"
DataFile="myxml.xml"></asp:XmlDataSource>
```

سپس جزییات DVD را به غیر از جزییات بازیگرانش را که از تگ Starring/Star بازیابی می شود در یک GrdiView می نویسیم:

```
    <asp:GridView   ID="grid"   runat="server"   AutoGenerateColumns="false"
DataSourceID="xml1">
    <Columns>
    <asp:TemplateField HeaderText="aaa">
    <ItemTemplate>
    Title :<%# XPath("Title") %><br/>
    Director : <%# XPath("Director") %><br/>
    Price :$<%#XPath("Price")%><br/>
    <i>Starring...</i><br />
   </ItemTemplate>
```

در کد بالا منبع داده ای GrdiView را نام XmlDataSourceی که در بالا تعریف کرده بودم دادم سپس یک ستون ایجاد کردم با نام aaa که حاوی ۳ مقدار نام فیلم-کارگردان و قیمت و یک نوشته با فونت <i> ...starring است.در این مرحله من می توانم یک GrdiView دیگر قرار دهم و بازیگران را در آن بازیابی کنم ولی برای تنوع و حجم کمتر از یک repeater استفاده می کنم.همانطور که در کد بالا می بینید من از یک سری text هم استفاده کردم مثلا :Title.پس برای نوشتن text در داخل ItemTemplate مانعی پیش روی شما نیست.

در اینجا نوبت به بیان یک نکته ی مهم می رسد و آن هم این است که متد XPath که در بالا هم از آن استفاده کردیم مقدار برگشتیش از نوع String است.در اینجا من برای DataSource کنترل repeater می خواهم از یک عبارت DataBinding استفاده کنم تا یک دسته از مقادیر گره ها را برای من بازیابی کند.مثلا اگر از XPath استفاده کنیم به صورت زیر می شود:

```
asp:Repeater ID="repeater1" runat="server" DataSource='<%#
XPath("Starring/Star")%>'>
```

در این کد منبع داده ای تنها یک مقدار رشته ای برمی گرداند در حالیکه من نیاز به دسته ای از تگ ها دارم .مثلا در اینجا من می خواهم نام تمام بازیگران در منبع داده ای من باشد تا من با XPath(“.”) یکی یکی آنها را نمایش دهم.بدین منظور یک متد جدید ارایه شد به نام XPathSelect() که آرگومان ورودیش همان الگوی XPath است و یک دسته گره را برای من برمی گرداند پس عبارت DataBinding به صورت زیر می شود:

```
<%# XPathSelect("Starring/Star")%>
```

و کنترل repeater به شکل زیر می شود:

```
    <asp:Repeater     ID="repeater1"     runat="server"     DataSource='<%#
XPathSelect("Starring/Star")%>'>
    <ItemTemplate>
    <h4><%# XPath(".") %></h4><br />
    </ItemTemplate>
  </asp:Repeater>
```

همانطور که می دانید در کنترل repeater دیگر نباید ابتدا تگ Columns و سپس TemplateField و... را تعریف کنیم بلکه کافی است تنها از ItemTemplate استفاده کنیم زیرا repeater خودش ستونی ندارد و تمام ستونهایش را ما خودمان باید ایجاد کنیم.

کنترل repeaterی که در بالا تعریف شد را باید در درون تگ ItemTemplate کنترل GridView و پس از تگ Starring.. قرار دهیم.در ضمن این هم یادتان باشد که متد XPathSelect هم مثل XPath می تواند از سند xmlی که به عنوان منبع داده ی GridView انتخاب شده استفاده کند و تگ DVD آن را به عنوان مبدا در نظر بگیرد.بنابر این اینکه ما در یک کنترل repeater از XPathSelect دلیل این نیست که نتوانیم از سند xml منبع داده ای GridView استفاده کنیم چون به هر حال کنترل repeater در درون GridView تعریف شده.نمونه ای از خروجی را در زیر می بینید:

Title :The Matrix
Director : Larry Wachowski
Price :$18.74
Starring...

Keanu Reeves

Laurence Fishburne

حال می خواهیم کمی در مورد Bind کردن داده های xml سلسله مراتبی در کنترل TreeView صحبت کنیم.می توان داده های xml که به صورت سلسله مراتبی هستند را در کنترل Bind TreeView کرد.

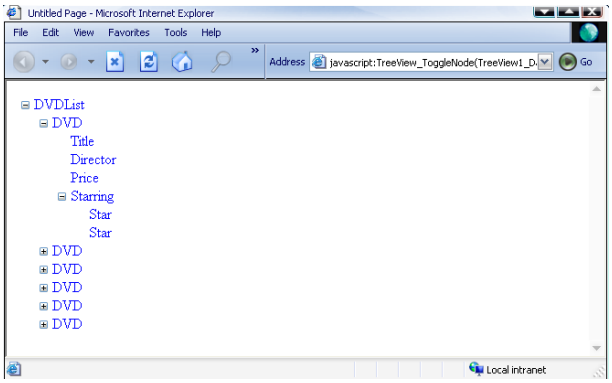
کنترل TreeView از جمله کنترل هایی است که برای پیمایش سایت به کار میروند و شبیه یک درخت پیمایش است و بسیار پرکاربرد هم هست.این کنترل به صورت زیر تعریف می شود:

```
  <asp:TreeView ID="TreeView1" runat="server" DataSourceID="...">
</asp:TreeView>
```

همانطور که می بینید مهم ترین صفت آن صفت DataSource است که منبع داده ای را برای آن مشخص می کند که اغلب از نوع SiteMapDataSource و گاهی هم مثل اینجا از نوع xmldatasource است.حال یک xmldatasource و یک TreeView به صورت زیر تعریف می کنیم:

```
  <asp:TreeView ID="TreeView1" runat="server"
DataSourceID="XmlDataSource1"></asp:TreeView>
    <asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="myxml.xml"></asp:XmlDataSource>
```

همانطور که می بینید کنترل XmlDataSource به سند XML مربوط به DVD ها اشاره دارد و منبع داده ی کنترل TreeView هم همان XmlDataSource در نظر گرفتیم.حال اگر به خروجی نگاه کنید چیزی مثل شکل زیر می بینید:



همانطور که میبینید عمل Bind به خوبی انجام شد ولی به جای Value تگ ها نامشان را نمایش می دهد و این امر نوید کد نویسی دستی را برای Bind کردن xml در TreeView می دهد.پس قبل از هر کاری صفت AutoGenarateDatabindings را برابر با false قرار دهید:

```
<asp:TreeView ID="TreeView1" runat="server" DataSourceID="XmlDataSource1"
AutoGenerateDataBindings="false"></asp:TreeView>
```

این کار دقیقا مثل false کردن AutoGenerateColumns کنترل GridView است و از Bind شدن خودکار سند جلوگیری می شود و به ما اجازه ی Bind کردن به طور دستی را می دهد.برای اعمال Bind کردن به طور دستی می بایست از عنصر DataBindings استفاده کنید که خود آن نیز یک یا چند شی TreeNodeBindings را شامل می شود که هر یک از آنها برای Bind کردن یک تگ مورد استفاده قرار می گیرد:

```
<DataBindings>
<TreeNodeBindings ...>
<TreeNodeBindings ...>
...
</ DataBindings>
```

برای استفاده از TreeNodeBindings باید ابتدا از سطر ریشه شروع به Bind کردن کنید.دو خصوصیت مهم شی TreeNodeBindings به ترتیب DataMember و TextField هستند:

```
<asp:TreeNodeBinding DataMember="..." TextField="..." />
```

DataMember به نام تگ xml اشاره دارد که کنترل TreeView باید به دنبالش بگردد و روی آن تمرکز کند یعنی آماده باشد که عمل بازیابی را از ویژگی ها و یا text آن تگ انجام دهد و صفت TextField نام عنصری است که می خواهید مقدارش را نمایش دهید.مثلا اگر عنصر شما یک ویژگی باشد باید به TextField نام آن ویژگی را بدهید و یا اگر مقدار بین دو تگ باشد (text هر تگ) آنگاه باید مقدار #InnerText را به TextField بدهید.

همانطور که گفتیم کنترل TreeView یک کنترل سلسله مراتبی است.پس وقتی ی خواهیم یک سند xml سلسله مراتبی را در آن نمایش دهیم سند ما باید حتما حاوی گره ی ریشه باشد.پس همیشه اولین شی TreeNodeBindings که در داخل تگ DataBindings تعریف می شود باید به گره ی ریشه اشاره داشته باشد.به عبارت دیگر اگر می خواهید به هر تگی دسترسی داشته باشید باید حتما قبلش والد آن تگ را تعریف کنید.در اینجا هم چون می خواهیم به DVD دسترسی داشته باشیم می بایست والدش را که DVDList است تعریف کنیم.در سند مربوط به DVD ها,گره ی ریشه DVDList بود پس آن را به صورت زیر تعریف می کنیم:

```
<asp:TreeNodeBinding DataMember="DVDList" Text="Root" />
```

دقت کنید که چون تگ ریشه ی ما که نامش DVDList است نه حاوی text است و نه ویژگی , بنابراین جایز نیست از ویژگی TextField کنترل TreeView استفاده کنیم.ویژگی text به ما می گوید می تواند به جای نام اصلی تگ که در اینجا DVDList است یک نام دیگر (که همان مقدار ویژگی text است) تعیین کند.

خوب پس از تعیین گره ی ریشه می توانیم فقط گره ی DVD را تعریف کنیم و به عناصرش دسترسی داشته باشیم چون والدش تعریف شده و اگر بخواهیم مثلا به تگ Title دسترسی داشته باشیم می بایست والدش که dvd است راقبل از آن تعریف کرده باشیم که در اینجا من DVD ها را بر حسب id تفکیک کردم پس ویژگی TextField را برابر id قرار دادم تا ویژگی id تگ DVD نمایش داده شود:

```
<asp:TreeNodeBinding DataMember="DVD" TextField="ID" />
```

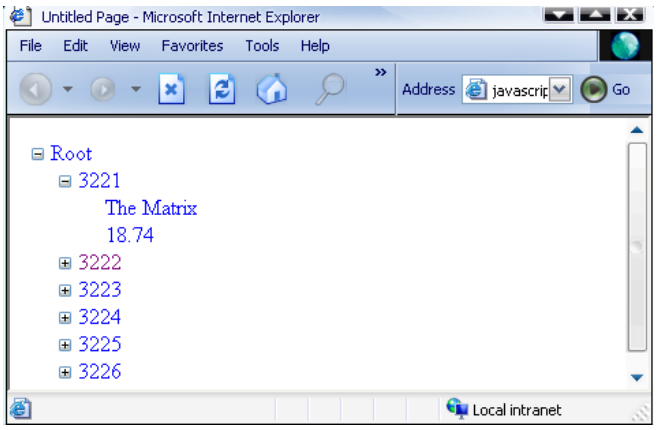
به همین ترتیب عناصر دلخواه دیگری را هم تعریف می کنم:

```
<asp:TreeNodeBinding DataMember="Title" TextField="#InnerText" />
<asp:TreeNodeBinding DataMember="Price" TextField="#InnerText" />
<asp:TreeNodeBinding DataMember="Star" TextField="#InnerText" />
```

پس کد کلی TreeView به فرم زیر شد:

```
      <asp:TreeView                ID="TreeView1"                runat="server"
DataSourceID="XmlDataSource1" AutoGenerateDataBindings="false">
      <DataBindings>
      <asp:TreeNodeBinding DataMember="DVDList" Text="Root" />
      <asp:TreeNodeBinding DataMember="DVD" TextField="ID" />
      <asp:TreeNodeBinding DataMember="Title" TextField="#InnerText" />
      <asp:TreeNodeBinding DataMember="Price" TextField="#InnerText" />
       <asp:TreeNodeBinding DataMember="Star" TextField="#InnerText" />
      </DataBindings>
    </asp:TreeView>
```

و خروجی به فرم زیر می شود:



جلوتر وقتی به بحث WebSite Navigation رسیدیم در مورد کنترل TreeView بیشتر بحث می کنیم.

روش دیگری نیز برای Bind کردن TreeView وجود دارد که انعطاف و راحتی بیشتری دارد و آن هم استفاده از XSLT برای آماده کردن داده ها برای Bind کردن در TreeView است.در این روش شما با XSL سند xml دلخواه خود را از روی سند xml تعریف و ایجاد می کنید و سپس به همان روشی که در بالا به آن اشاره کردیم آن را در Bind TreeView می کنیم.عمل تبدیل XSL به Xml به صورت Built-in در کنترل XmlDataSource وجود دارد پس نیازی به استفاده از کلاس Xslcompiledtransform نیست.نکته ای که راجب به تبدیل xsl وجود دارد این است که اینجا نیاز به تبدیل xml به html نیست چون می خواهیم آن را در aspx و در treeview به کار ببریم باید آن را در کنترل XmlDataSource به عنوان منبع داده ای ذخیره کنیم.برای این کار از ویژگی TransformFile کنترل xmldatasource استفاده می کنیم که باید به آن آدرس فایل xsl را بدهیم تا عمل تبدیل انجام گیرد:

```
<asp:XmlDataSource ID="xml1" runat="server" DataFile="your xml file to
convert" TransformFile="Your xsl File" />
```

با این کار خروجی حاصل از تبدیل شما در داخل کنترل XmlDataSource قرار می گیرد و آماده است برای Bind کردن.

برای مثال من می خواهم یک سند xml مثل زیر از روی سند اصلی بسازم:

```
<Movies>
  <DVDs ID="3221" Title="The Matrix">
    <Stars star1_Name="Keanu Reeves" star2_Name="Laurence Fishburne" />
  </DVDs>
  <DVDs ID="3222" Title="Basic Instinct">
    <Stars star1_Name="Michael Douglas" star2_Name="Sharon Stone" />
  </DVDs>
  ...
</Movies>
```

می بینید که یک تگ ریشه ایجاد کردم به نام Movies که حاوی فرزندان مشابهی به نام DVDs است.سپس هر فرزند دو ویژگی دارد که اولی id فیلم و دومی نام فیلم است و بازیگران هر فیلم هم در تگ فرزند DVDs به نام Stars در دو ویژگی ذخیره شدند.برای تولید سند بالا کد xsl را به صورت زیر می نویسیم.

ابتدا تگ جدیدی به نام Movies ایجاد می کنیم:

```
<xsl:template match="/">
    <xsl:element name="Movies">
        <xsl:apply-templates select="//DVD" />
    </xsl:element>
</xsl:template>
```

همانطور که می بینید در این تگ مقادیر تگهای DVD سابق نمایش داده می شوند.

سپس عمل میکس را برای تگ های DVD که در بالا در داخل تگ جدید Movies لود کردیم انجام می دهیم و می گوییم هر جا تگ DVD دیدی جایش از تگی به نام DVDs استفاده کن و در ضمن دو ویژگی به آن اضافه کن و در انتها هر چه در داخل تگ Starring بود را نیز نمایش بده:

```
<xsl:template match="DVD">
    <xsl:element name="DVDs">
        <xsl:attribute name="ID">
            <xsl:value-of select="@ID"/>
        </xsl:attribute>
        <xsl:attribute name="Title">
            <xsl:value-of select="Title"/>
        </xsl:attribute>
        <xsl:apply-templates select="Starring" />
    </xsl:element>
</xsl:template>
```

در نهایت هم عمل میکس را برای تگ های Starring که در بالا بازیابی شد انجام می دهیم و می گوییم هر جا از خروجی هایی که در بالا تولید شده تگ Starring دیدی جایش را با تگ جدیدی به نام Stars عوض کن و دو ویژگی برای دو بازیگر اصلی به آن اضافه کن:

```
<xsl:template match="Starring">
    <xsl:element name="Stars">
          <xsl:attribute name="star1_Name">
                <xsl:value-of select="Star[1]"/>
          </xsl:attribute>
          <xsl:attribute name="star2_Name">
                <xsl:value-of select="Star[2]"/>
          </xsl:attribute>
    </xsl:element>
</xsl:template>
```

باز هم کاربرد گسترده ی میکس کردن را در مثال بالا دیدید.

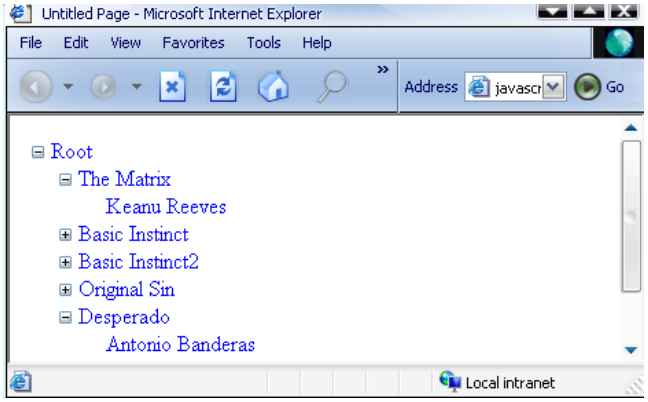
حال این فایل XSL را با نام treeview.xsl ذخیره کنید و به صفت TransformFile کنترل xmldatasource بدهید:

```
<asp:XmlDataSource ID="xml1" runat="server" DataFile="myxml.xml"
TransformFile="treeview.xsl" />
```

حال انگار یک سند xml جدید دارید.پس بر اساس آن عمل بازیابی را انجام بدهید:

```
    <asp:TreeView  ID="tr"  runat="server"  AutoGenerateDataBindings="false"
DataSourceID="xml1">
    <DataBindings>
    <asp:TreeNodeBinding DataMember="Movies" Text="Root" />
    <asp:TreeNodeBinding DataMember="DVDs" TextField="Title" />
    <asp:TreeNodeBinding DataMember="Stars" TextField="star1_Name" />
    </DataBindings>
  </asp:TreeView>
```

خروجی چیزی مثل شکل زیر می شود:



پس می بینید که در صورتی که سند شما بسیار شاخ و برگ داشت و نمی توانستید با روش قبلی Bind در TreeView بازیابی های دلخواه را انجام دهید آنگاه می توانید با XSL یک سند دلخواه از دل سند قبل بیرون بکشید (این کار را با همان XmlDataSource به سادگی آب خوردن تنها با مقداردهی ویژگی TransformFile علاوه بر DataFile انجام می دهید) و سپس آن سند حاصل را در TreeView به همان روش قبل Bind کنید.

این نکته هم یادتان باشد هنگامی که از XmlDataSource استفاده می کنید Cache کردن آن یادتان نرود .چون اصلا استفاده از منبع داده ای Xml به جای Sql باعث افزایش کارایی و سرعت سایت می شود چه برسد به اینکه XmlDataSource ا هم کش کنید.

XML و ADO.NET:

در این قسمت می خواهیم از دو متد Write Xmlو Read Xmlشی DataSet استفاده کنیم.متد WriteXml متدی است که با آن می توان محتویات شی DataSet را به فرم xml ذخیره کرد و متد ReadXml متدی است که با آن می توان محتویات یک فایل xml را خواند و آن را در یک Dataset قرار داد تا بتوان آن را از طریق DataSet نمایش داد.می خواهیم در یک مثال محتویات یک جدول از پایگاه داده را تبدیل به یک سند xml کنیم.این کار از طریق DataSet و به صورت Offline صورت می گیرد.پس ابتدا ۵ رکورد اول جدولی به نام employees را از پایگاه داده بازیابی می کنم و آن را درون DataSet می ریزم و حتی آن را در GridView نمایش می دهم:

```
        Dim connectionString As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim sql As String = "SELECT TOP 5 * FROM employees"
        Dim conn As SqlConnection = New SqlConnection(connectionString)
        Dim da As SqlDataAdapter = New SqlDataAdapter(sql, conn)
        Dim ds As DataSet = New DataSet()

        da.Fill(ds, "employees")
        GridView1.DataSource = ds.Tables("employees")
    GridView1.DataBind()
```

حال باید عمل تبدیل محتویات این dataset را به xml انجام دهم و سپس آن را در یک فایل xml جدید بریزم.پس ابتدا مسیری و نامی برای فایل با متد server.mapPath ایجاد می کنم:

```
Dim xmlFile As String = Server.MapPath("employees.xml")
```

از متد WriteXml شی DataSet برای عمل تبدیل استفاده می کنیم.این متد ۲ آرگومان ورودی دارد که اولی مسیر فایل xmlی است که اولا باید ساخته شود و سپس محتویات تبدیل شده باید در آن ریخته شوند که ما آن را در بالا تعریف کردیم و دومی متدی برای تعیین نوع سند خروجی است.این متد از کلاس XmlWriteMode مشتق شده و حاوی ۳ متد است.اولی WriteSchema است که باعث ایجاد و درج فایل xsd مربوط به جدول مربوطه در پایگاه داده در کنار سند xml تولید شده می شود.دومی IgnoreSchema است که از تولید و به خروجی رفتن فایل xsd مربوط به جدول مربوطه در پایگاه داده جلوگیری می کند و تنها سند xml تولید می شود.من از متد IgnoreSchema استفاده می کنم ولی اگر خواستید فایل xsd مربوط به جدول پایگاه داده را نیز ببینید WriteSchema را انتخاب کنید:

```
ds.WriteXml(xmlFile, XmlWriteMode.IgnoreSchema)
```

در همینجا عمل تبدیل انجام می شود.خروجی (فایل employees.xml) به صورت زیر می شود:

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <employees>
    <emp_ID>315d18d58-cd73-4b4f-94b7-c930067bc4ae</emp_ID>
    <emp_firstname>arezoo</emp_firstname>
    <emp_lastname>nasehi</emp_lastname>
    <emp_job>motarjem</emp_job>
  </employees>
  <employees>
    <emp_ID>3dba9a04e-dcec-441a-86b14fjc3a10eb5</emp_ID>
    <emp_firstname>samad</emp_firstname>
    <emp_lastname>najafi</emp_lastname>
    <emp_job>okgkayr</emp_job>
  </employees>
  <employees>
    <emp_ID>3dba9a04e-dcec-441a-86b1-941ec3a10e09</emp_ID>
    <emp_firstname>mahtab</emp_firstname>
    <emp_lastname>rajabzadeh</emp_lastname>
    <emp_job>monshi</emp_job>
  </employees>
  <employees>
    <emp_ID>3dba9a04e-dcec-441a-86b1-941ec3a10e41</emp_ID>
    <emp_firstname>mahtab</emp_firstname>
    <emp_lastname>keramatipoor</emp_lastname>
    <emp_job>computerist</emp_job>
```

```
    </employees>
    <employees>
      <emp_ID>3dba9a04e-dcec-441a-86b1-941ec3a10eb5</emp_ID>
      <emp_firstname>soozan</emp_firstname>
      <emp_lastname>javadi</emp_lastname>
      <emp_job>monshi</emp_job>
    </employees>
</NewDataSet>
```

می بینید که هر ستون در جدول پایگاه داده تبدیل به یک تگ شده و مقدار آن ستون نیز تبدیل به Text آن تگ شده.هر رکورد نیز در داخل تگ employees قرار دارد که نام جدول مربوطه که در dataset با آن نام ذخیره شده بود است.در ضمن تگ ریشه در اینجا نام پیش فرض DataSet است (NewDataSer) اگر چنانچه خواستید نام تگ ریشه را تغییر دهید قبل از عملیات تبدیل, باید نام شی DataSet را تغییر دهید.این کار با صفت DataSetName شی DataSet میسر است.مثلا:

```
ds.DataSetName = "root"
```

در این صورت نام تگ ریشه root خواهد بود.

برای نمایش محتویات سند xml تولید شده, کافی است از متد ReadXml شی dataSet استفاده کنید و به عنوان آرگومان ورودی مسیر فایل xml مربوطه را به آن بدهید:

```
  Dim dsXml As New DataSet
dsXml.ReadXml(xmlFile)
```

و برای نمایش هم به صورت زیر عمل کنید:

```
  GridView2.DataSource = dsXml
GridView2.DataBind()
```

پس در این مثال با نحوه ی تبدیل جداول پایگاه داده به اسنادxml از طریق DataSet آشنا شدید.

حالا می خواهیم با رابطه ی بین Sql Server و Xml آشنا شویم.شما می توانید با استفاده از Query های معمولی sql , عمل بازیابی از پایگاه داده را انجام دهید و سپس آن را به شکل xml در آورده و از آن عمل بازیابی را انجام دهید.این کار را xQuery یا Execute Xml Query می گویند.این کار چند مرحله دارد اول اینکه باید به انتهای Query که اجرا می کنید عبارت FOR XML را قرار دهید.این عبارت به sql server می گوید داده ها چگونه تنظیم شوند و به فرم Xml در آیند.پس از اضافه کردن عبارت FOR

XML , شما دو انتخاب دارید.اول اینکه صفات خاصه ی جداول پایگاه داده به صورت ویژگی در xml ایجاد شوند و مقدارشان مقدار ویژگی باشد و دوم اینکه صفات خاصه ی جداول پایگاه داده به صورت تگ در آیند و مقدار آنها به صورت text تگ ها باشد.برای عملی کردن مورد اول لازم است عبارت AUTO را به انتهای FOR XML اضافه کنید و برای عملی کردن مورد دوم عبارت AUTO,ELEMENTS را به انتهای FOR XML اضافه کنید.مثلا Query زیر را برای بازیابی نام و نام خانوادگی از جدول employees در نظر بگیرید:

```
SELECT emp_firstname,emp_lastname FROM employees
```

برای عملی کردن مورد اول query بالا به صورت زیر در می آید:

```
SELECT emp_firstname,emp_lastname FROM employees FOR XML AUTO
```

و xmlی که بازیابی از آن انجام می گیرد به صورت زیر می شود:

```
<employees emp_firstname="Nancy" emp_lastname="Davolio"/>
<employees emp_firstname ="Andrew" emp_lastname ="Fuller"/>
<employees emp_firstname ="Janet" emp_lastname ="Leverling"/>
```

و برای عملی کردن مورد دوم query بالا به صورت زیر در می آید:

```
SELECT emp_firstname,emp_lastname FROM employees FOR XML AUTO,ELEMENTS
```

و xmlی که بازیابی از آن انجام می گیرد به صورت زیر می شود:

```
<employees>
< emp_firstname >Nancy</emp_firstname>
< emp_lastname >Davolio</emp_lastname>
<employees/>
<employees>
< emp_firstname >Andrew</emp_firstname>
< emp_lastname >Fuller</emp_lastname>
<employees/>
<employees>
< emp_firstname >Janet</emp_firstname>
< emp_lastname >Leverling</emp_lastname>
employee/>
```

مورد بعدی که باید رعایت کنید این است که در اینجا برای خواندن داده ها از شی XmlReader به جای SqlDataReader استفاده می کنیم و برای اجرای Query هم از ExecuteXmlReader شی SqlCommand به جای ()ExecuteReader استفاده می کنیم.در ادامه مثالی در این زمینه می بینیم که هدفش اجرای query ذکر شده در بالا به روش دوم است.پس ابتدا اشیا مربوطه را ایجاد می کنیم:

```
        Dim connectionString As String = ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim customerQuery As String = "SELECT emp_firstname, emp_lastname FROM employees FOR XML AUTO, ELEMENTS"
        Dim con As New SqlConnection(connectionString)
        Dim com As New SqlCommand(customerQuery, con)
      Dim str As String = String.Empty
```

سپس Connection را باز می کنیم و پس از آن عمل اجرای Query را انجام می دهیم:

```
  con.Open()
Dim reader As XmlReader = com.ExecuteXmlReader()
```

حال که reader حاوی مقادیر سند xml حاصل از اجرای Query است لازم است که بازیابی های لازم از آن انجام شود.برای این کار از یک حلقه ی Do While با شرط reader.read استفاده می کنیم یعنی می گوییم تا وقتی تگ در سند وجود دارد وارد حلقه شو:

```
    Do While reader.Read()
  Loop
```

حال در داخل حلقه چه بنویسیم.با توجه به اینکه متد هایی که شی XmlReader دارد بسیار شبیه متد هایی است که شی XmlTextReader دارد, می توان از متد های مشابه آن استفاده کرد مثلا متد ReadStartElement تگ آغاز را مشخص می کرد و ReadElementString نیز text تگ (های) فرزند تگی را که با ReadStartElement مشخص شده را با گرفتن نام آن تگ به عنوان آرگومان ورودی برمی گرداند.مثلا در سند مربوط به DVD ها, ReadStartElement تگ DVD بود و ReadElementString نیز تگ هایی نظیر Title بودند:

```
  Do While reader.Read()
      reader.ReadStartElement("employees")
      str += reader.ReadElementString("emp_firstname")
      str += " "
      str += reader.ReadElementString("emp_lastname")
      str += "<br/>"
Loop
```

پس کد کلی را به صورت زیر بنویسید.در کد زیر من از یک شرط If جهت محکم کاری و جلوگیری از بروز خطا های احتمالی در ابتدای حلقه ی Do While استفاده کردم و تمام بازیابی ها را در داخل آن شرط انجام دادم که در بحث XmlTextReader راجع به آن شرط مفصل توضیح داده شده بود:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim connectionString As String =
ConfigurationManager.ConnectionStrings("aaa").ConnectionString
        Dim customerQuery As String = "SELECT emp_firstname, emp_lastname
FROM employees FOR XML AUTO, ELEMENTS"
        Dim con As SqlConnection = New SqlConnection(connectionString)
        Dim com As SqlCommand = New SqlCommand(customerQuery, con)
        Dim str As String = String.Empty
        Try
            con.Open()
            Dim reader As XmlReader = com.ExecuteXmlReader()

            Do While reader.Read()
                If (reader.Name = "employees") AndAlso (reader.NodeType =
XmlNodeType.Element) Then
                    reader.ReadStartElement("employees")
                    str += reader.ReadElementString("emp_firstname")
                    str += " "
                    str += reader.ReadElementString("emp_lastname")
                    str += "<br/>"
                End If
            Loop
            reader.Close()
        Finally
            con.Close()
        End Try
        Label1.Text = str.ToString()
    End Sub
```

به عنوان آخرین بحثی که در زمینه ی xml خواهیم داشت می خواهیم با Editable xml DataBinding آشنا شویم.یعنی داده های xml را بتوانیم از داخل صفحات xml حذف-اضافه و ویرایش کنیم.ما این کار را با کنترل XmlDataSource انجام می دهیم ولی با تفاوت زیاد نسبت به کنترل های SqlDataSource & ObjectDataSource.زیرا کنترل XmlDataSource حاوی خصیصه ی قابل ویرایش بودن داده ها نیست.برای فهمیدن این موضوع کافی است داده های کنترل XmlDataSource را در یک GridView نمایش دهید و سپس دکمه ی Edit را در GridView فعال کنید و ویرایش را انجام بدهید و در انتها Update را کلیک کنید آن وقت با پیغام خطایی مبنی بر اینکه XmlDataSource حاوی خصیصه ی قابل ویرایش بودن داده ها نیست روبرو می شوید.

برای ویرایش داده های xml توسط XmlDataSource می توان از متد ()Save آن استفاده کرد.کار این متد جابجا کردن سند xml اصلی(همانی که در ویژگی DataFile کنترل XmlDataSource مشخص شده بود) با سند جاری است.حال این سند جاری کدام است؟برای بدست آوردن سند جاری از متد GetXmlDocument کنترل XmlDataSource استفاده می کنید.این متد یک سند xml با فرمت XmlDocument در اختیار شما قرار می هد و شما می توانید تغییرات دلخواه خود را با متد های مربوطه روی آن انجام دهید و در نهایت متد ()Save کنترل XmlDataSource را صدا بزنید.با صدا زدن این متد, سند xml تغییر یافته (که همان سند جاری محسوب می شود) جایش با سند قبلی عوض می شود.قبل از شروع مثال های عملی لازم است با یک سری متد آشنا شوید.

در ۳ متد مربوط به شی XmlDocument را می بینید:

SelectSingleNode:براساس فرمت XPathی که به عنوان آرگومان ورودی دریافت می کند از سند درون شی XmlDocument یک گره را انتخاب می کند و آن را از جنس XmlNode برمی گرداند.

CreateElement:یک تگ ایجاد می کند که نامش را به عنوان آرگومان ورودی دریافت می کند.مقدار برگشتی آن هم از نوع XmlElement است.

DocumentElement:تگ ریشه ی سند xml را به ما می دهد.یعنی با آن می توانید به تگ ریشه دسترسی داشته باشید.

شی XmlElement نیز حاوی دو متد مهم است :

SetAttribute:پس از ایجاد تگ , با این متد می توانید یک ویژگی به آن اضافه کنید.دو آرگومان ورودی دارد که اولی نام ویژگی و دومی مقدار آن است.

AppendChild: پس از ایجاد تگ, با این متد می توانید برای تگ جاری فرزند انتخاب کنید ولی به شرطی که تگ های فرزند را قبلا مشخص کرده باشید چون آرگومان ورودی این تابع از جنس XmlNode است.یادتان باشد xmlNode یعنی یک گره در سند xml.حال اگراین گره تگ باشد XmlElement نامیده می شود.پس XmlNode حکم یک نوع کلی را برای XmlElement و یا XmlAttribute و... دارد.

در حالت کلی نیز شی XmlNode یک متد به نام RemoveChild دارد.این متد فرزند یک تگ دیگر را پاک می کند .به عبارت دیگر اگر a گره ی والد و b فرزند باشد برای پاک کردن b به صورت زیر عمل می کنیم:

a.RemoveChild(b)

حال که با متد های لازم برای ویرایش سند xml آشنا شدیم به سراغ مثال های عملی در این زمینه می رویم.در ابتدا می خواهیم یک DVD جدید به سند اضافه کنیم که شامل تمام تگ هایی که سایر DVD ها دارند باشد.این کار را با نوشتن یک تابع انجام می دهیم.ولی قبل از آن یک Repeater ایجاد می کنیم و داده های سندمربوط به DVD ها را در آن قرار می دهیم تا در هر لحظه شاهد تغییرات اعمال شده باشیم.پس ابتدا کنترل XmlDataSource را ایجاد می کنیم:

```
<asp:XmlDataSource ID="xml1" runat="server" DataFile="myxml.xml" />
```

سپس کنترل Repeater را به صورت زیر ایجاد می کنیم:

```
      <asp:Repeater ID="Repeater1" runat="server" DataSourceID="Xml1" >
        <ItemTemplate >
            <h1><%#XPath("Title")%> </h1>
            <b>Category:</b><%#XPath("@Category")%>
            <b>Director:</b><%#XPath("Director")%>
            <b>Price:</b><%#XPath("Price")%>
        </ItemTemplate>
  </asp:Repeater>
```

حال به سراغ نوشتن تابعی برای Add کردن یک DVD جدید به سند xml می رویم.این تابع را با نام AddDVD و آرگومان های ورودی به تعداد مقادیری که لازم داریم به صورت زیر می نویسیم:

```
      Public Sub AddDVD(ByVal id As String, ByVal cat As String, ByVal title
As String, ByVal dir As String, ByVal price As String, ByVal star1 As String,
ByVal star2 As String)

    End Sub
```

دقت کنید متغیر های id و Price که به ترتیب از جنس Integer و Decimal هستند را از نوع String دریافت کردیم . هنگام استفاده آنها را با توابع CInt و CDec به مقدار اصلیشان تبدیل می کنیم.

در ابتدا یک شی از نوع XmlDocument تعریف می کنیم :

```
Dim myXml As New XmlDocument
```

سپس با متد getXmlDocument() شی XmlDataSource عمل بازیابی سند xmlی که در خصوصیت DataFile آن تعریف کردیم را انجام می دهیم و چون مقدار برگشتی از جنس XmlDocument است آن را در داخل متغیری که از جنس XmlDocument در بالا تعریف کردیم میریزیم:

```
myXml = xml1.GetXmlDocument()
```

حال می رسیم به پیکربندی تگ DVD که به صورت زیر بود:

```
<DVD Id="..." Category="...">
  <Title>...</Title>
  <Director>...</Director >
  <Price>...</Price >
  <Starring>
    <Star>...</Star>
    <Star>...</Star>
  </Starring>
</DVD>
```

در ابتدا باید تگ ها را ایجاد کنیم.این کار را با متد CreateElement شی XmlDocument انجام می دهیم.در ابتدا یک تگ به نام DVD ایجاد می کنیم و آن را در متغیری از جنس XmlElement با نام dvd ذخیره می کنیم(دلیلش را جلوتر متوجه می شوید):

```
Dim dvd As XmlElement = myXml.CreateElement("DVD")
```

سپس همین کار را برای تگ title انجام می دهیم با این تفاوت که تگ title حاوی text است.برای قراردادن text برای یک تگ از متد InnerText استفاده می کنیم.به این ترتیب که ابتدا تگ را ایجاد کرده و آن را در داخل متغیری از جنس XmlElement قرار می دهیم:

```
Dim tit As XmlElement = myXml.CreateElement("Title")
```

حال که تگ Title را در متغیر tit داریم, می توانیم از متد InnerText شی XmlNode یا همان XmlElement استفاده کنیم و مقدار Titleی را که به عنوان آرگومان ورودی داشتیم را به مقدار آن بدهیم:

```
tit.InnerText = title
```

به همین ترتیب تگ های Director و Price را نیز ایجاد می کنیم:

```
  Dim dirr As XmlElement = myXml.CreateElement("Director")
  dirr.InnerText = dir
  Dim pr As XmlElement = myXml.CreateElement("Price")
pr.InnerText = CDec(price)
```

سپس نوبت به تگ Starring می رسد:

```
Dim Str As XmlElement = myXml.CreateElement("Starring")
```

و دو تگ Star:

```
  Dim str1 As XmlElement = myXml.CreateElement("Star")
  str1.InnerText = star1
  Dim str2 As XmlElement = myXml.CreateElement("Star")
str2.InnerText = star2
```

حال که تعریف تگ ها تمام شد باید دو کار انجام دهیم.اول اینکه رابطه ی تگ ها با خودشان را ایجاد کنیم و دوم اینکه رابطه ی این تگ ها را با سند اصلی ایجاد کنیم.

پس ابتدا دو ویژگی IDو Category را به تگ DVD که با نام dvd تعریف کرده بودیم اضافه می کنیم.این کار را با متد setAttribute انجام می دهیم:

```
  dvd.SetAttribute("ID", CInt(id))
dvd.SetAttribute("Category", cat)
```

سپس فرزندان تگ DVD را با متد AppendChild یکی یک مشخص می کنیم:

```
  'tit is the name Of Title Tag
  dvd.AppendChild(tit)
  dvd.AppendChild(dirr)
  dvd.AppendChild(pr)
dvd.AppendChild(Str)
```

نام هایی که به عنوان آرگومان ورودی به متد های AppendChild داده شده نام متغیر هایی است که تگهایی است که در بالا ایجاد کردیم را در آنها ذخیره کرده بودیم.

سپس دو فرزند تگ Starring (در متغیر str ذخیره شده)را مشخص می کنیم:

```
  Str.AppendChild(str1)
Str.AppendChild(str2)
```

در نهایت باید این تگ DVD جدید را به عنوان فرزند جدید تگ DVDList معرفی کنیم.همانطور که می دانیم تگ DVDList تگ ریشه است و با متد DocumentElement شی XmlDocument می توان به آن دسترسی داشت و هر وقت بخواهیم برایش فزندی ایجاد کنیم کافی است از متد AppendChild برایش استفاده کنیم:

```
myXml.DocumentElement.AppendChild(dvd)
```

در نهایت برای اعمال کارهایی که انجام دادیم به سند xml از متد Save کنترل XmlDataSource استفاده می کنیم تا جای سند قبلی را با سند جاری(که در سند MyXml ایجاد کردیم) عوض کند:

```
xml1.Save()
```

برای نمایش تغییرات داده شده نیز کنترل repeater را یک بار دیگر Bind می کنیم:

```
Repeater1.DataBind()
```

حال کافیست تابع AddDVD را صدا بزنیم.من این کار را به صورت زیر انجام دادم:

```
AddDVD("3229", "Drama", "The Game", "Winski", "40.71",Michael Douglas",
"Larry Belli")
```

حال می خواهیم به سراغ ویرایش داده برویم که بسیار ساده است.من می خواهم قیمت یک DVD خاص را که با id به آن دسترسی خواهم داشت را عوض کنم.پس تابعی به نام editXML می نویسم و آرگومان ورودیش را id و قیمت جدید قرار می دهم تا با id به تگ DVD مربوطه دسترسی داشته باشمودر ضمن هر دو را به صورت String دریافت می کنم و هنگام استفاده تبدیلات لازم را انجام می دهم:

```
    Public Sub editXML(ByVal id As String, ByVal new_price As String)

    End Sub
```

مثل تابع قبل ابتدا بازیابی سند را انجام می دهم:

```
  Dim myXml As New XmlDocument
myXml = CType(xml1.GetXmlDocument(), XmlDocument)
```

چون قصد من ویرایش یک گره ی خاص از نوع تگ است, ابتدا باید آن را پیدا کنم.به همین منظور از XPath استفاده می کنم و ابتدا مسیر آن را مشخص می کنم:

```
Dim path As String = "/DVDList/DVD[@ID=" & CInt(id) & "]/Price"
```

طبق مسیر بالا من می توانم به تگ Price فرزند تگ DVD دسترسی داشته باشم که ویژگی Id در آن برابر مقداری باشد که به عنوان آرگومان ورودی دریافت کردم.

برای استفاده از مسیر بالا و یا به عبارت دیگر برای انتخاب یک گره ی خاص از متد SelectSingleNode شی XmlDocument استفاده می کنم:

```
myXml.SelectSingleNode(path)
```

می توانستیم به جای SelectSingleNode از متد GetElementByTagName هم استفاده کنیم .فرق این دو این است که SelectSingleNode می تواند ویژگی ها را نیز انتخاب کند (یعنی با آن قادرید مقادیر ویژگی ها را نیز تغییر دهید) ولی GetElementByTagName تنها تگ ها را انتخاب می کند.

و برای دسترسی به گره ی مورد نظر آن را در متغیری از نوع XmlNode قرار می دهیم:

```
Dim node As XmlNode = myXml.SelectSingleNode(path)
```

حال ما به گره ی Price که از نوع تگ است دسترسی داریم و کافیست به آن مقدار دهی کنیم.چون از نوع تگ است از متد Innertext استفاده می کنیم:

```
node.InnerText = CDec(new_price)
```

و در نهایت هم عمل Save را انجام می دهیم و سپس Repeater را دوباره Bind می کنم:

```
  xml1.Save()
Repeater1.DataBind()
```

برای استفاده از تابع بالا کافیست آن را به درستی صدا کنیم:

```
editXML("3221", "34.76")
```

کد بالا قیمت DVD با "Id="3221 را به 34.76 تغییر می دهد.

در نهایت با نحوه ی حذف گره ها آشنا می شویم.برای این کار یک تابع به نام RemoveDVD می نویسم که آرگومان ورودیش تنها Id فیلم مورد نظر باشد.یعنی تگ DVD با Id مربوطه را حذف کن:

```
    Public Sub RemoveDVD(ByVal id As String)

    End Sub
```

سپس بازیابی های لازم را انجام می دهم:

```
  Dim myXml As New XmlDocument
myXml = CType(xml1.GetXmlDocument(), XmlDocument)
```

برای حذف تگ DVD باید از تابع RemoveChild استفاده کنیم.همانطور که گفتیم این تابع فرزند یک تگ را حذف می کند. تگ فرزندی که باید حذف شود به عنوان ورودی می گیرد.و گفتیم که اگر a گره ی والد و b فرزند باشد برای پاک کردن b به صورت زیر عمل می کنیم:

a.RemoveChild(b)

در این مثال a=DVDList و b=DVD می باشد.برای دسترسی به a که کافیست از متد DocumentElement استفاده کنیم ولی برای دسترسی به DVD باید مسیرش را مشخص کنیم چون ما می خواهیم یک DVD خاص را پاک کنیم که idیش را به عنوان آرگومان ورودی دریافت کرده ایم.پس یک مسیر برایش مشخص می کنیم:

```
Dim Cpath As String = "/DVDList/DVD[@ID=" & CInt(id) & "]"
```

و سپس آن را انتخاب می کنیم:

```
Dim Cnode As XmlNode = myXml.SelectSingleNode(Cpath)
```

پس از این به بعد Cnode حاوی تگ DVdی است که قرار است پاک شود.پس کافی است آن را به عنوان آرگومان ورودی به متد RemoveChild بدهیم:

```
myXml.DocumentElement.RemoveChild(Cnode)
```

و ...:

```
        xml1.Save()
    Repeater1.DataBind()
```

پس برای حذف یک DVD از سند با id خاص به صورت زیر عمل می کنیم:

```
RemoveDVD("3221")
```

نکته ی مهم این است که برای حذف یک تگ که والدش تگ ریشه نیست باید مسیر والد را جدا مشخص کنیم و والد را با همان متد SelectSingleNode انتخاب کنیم و در نهایت در کد بالا به جای myXml.DocumentElement از تگ انتخاب شده استفاده کنیم.

با این مثال بحث ما در مورد Xml به پایان رسید.

Files and Streams

بحث فایل ها و استریم ها از جمله مباحث بسیار مهمی است که ک برنامه نویس تحت وب باید با آن آشنا باشد زیرا همه چیز پایگاه داده نیست و کار با فایل ها و فولدرهای یک سایت نیز اهمیتی ویژه دارد.مهمترین و پر کاربرد ترین فایل ها و استریم ها در قسمت FileManager (یا Control Panel) وب سایت هایی است که خدمات Hosting ارایه می کنند.حتما می دانید که وب سایتی که شما می سازید در داخل یک فوادر در روی سرور ذخیره می شود و نامش هم همنام وب سایت شماست.سپس در در قسمت FileManager وب سایتی که به شما خدمات ارایه می دهد و در داخل آن فولدر, اجازه ی ایجاد درج – ویرایش و حذف فایلهای گوناگون را دارید.مثلا می توانید صفحات html ایجاد کنید و کد html مربوطه را در آن بنویسید و آنها را ذخیره کنید و سپس اجرا کنید.می توانید فایل های مختف را Upload کنید و حتی می توانید فولدر های جدید بسازید و در کل به آسانی روی فایل ها و فولدر های سایتتان از طریق قسمت FileManager صفحه ی وب ارایه دهنده ی hosting ,پیمایش کنید.پس می بینید که چقدر بحث فایل ها و استریم ها اهمیت دارد.

در اولین گام در این بخش با دو شی Directory و File و متد های آنها آشنا می شویم.

:Directory

شی است که به وسیله ی آن می توان به Directory های مختلف دسترسی داشت و آنها را کنترل کرد.

این شی حاوی متد های مختلفی است که به ترتیب با آنها آشنا می شویم.

CreateDirectory:برای ایجاد یک دایرکتوری یا مسیر استفاده می شود.به عبارت دیگر باعث ایجاد فولدر یا فولدر های تودرتو در یک مسیر خاص می شود.آرگومان وروردی این متد مسیر مورد نظر جهت ایجاد خواهد بود.برای مثال می خواهیم یک فولدر در درایو C ایجاد کنیم.ابتدا مسیر را مشخص می کنم:

```
Dim path As String = "C:\myDir"
```

همانطور که می بینید مسیر ما به صورت C:\myDir مشخص شده.یعنی یک فولدر به نام myDir در درایو C.نکته ی مورد توجه در اینجا این است که مسیر در داخل کامپیوتر محلی با \ و مسیر در وب با / نمایش داده می شود.سپس از متد CreateDirectory استفاده می کنیم و مسیر بالا را به عنوان آرگومان ورودی به آن می دهیم:

```
Directory.CreateDirectory(path)
```

به همین سادگی یک فولدر یا بهتر است بگوییم یک مسیر Directory ایجاد کردیم به این دلیل می گوییم مسیر که می توان چندین فولدر تودرتو هم ایجاد کرد مثلا اگر مسیر من به صورت زیر باشد:

```
Dim path As String = "C:\myDir\aaa\ghg"
```

آنگاه ۳ فولدر تودرتو به نامهای myDir –aaa – ghg ساخته می شود.

نکته ی دیگر این است که اگر مسیر مورد نظر که می خواهیم ایجاد کنیم, قبلا وجود داشته باشد, با خطایی مواجه نمی شویم.

Delete:متدی است که برای حذف یک دایرکتوری مورد استفاده قرار می گیرد.آرگومان ورودی اصلی این متد مسیر مورد نظر جهت حذف است.اگر این کتد را تنها با آرگومان ورودی مسیر صدا بزنیم, تنها برای حذف دایکتوری های خالی کاربرد دارد.برای مثال :

```
Directory.Delete("C:\myDir\aaa\ghg")
```

این عبارت, فولدر ghg را پاک می کند چون خالی است یعنی فایل و فولدری در آن نیست.ولی اگر کد به صورت زیر باشد با خطا مواجه می شویم:

```
Directory.Delete("C:\myDir")
```

زیرا فولدر myDir خالی نیست.

برای حذف یک Directory که خالی نیست لازم است از آرگومان ورودی دوم متد Delete استفاده کنیم.این آرگومان از جنس Boolean بوده و باعث می شود تابع Delete به صورت بازگشتی عمل کند.به عبارت دیگر اگر آن را برابر True قرار دهید, متد Delete به طور بازگشتی از آخرین فرزند دایرکتوری مورد نظر , عمل حذف را شروع می کند تا به خود آن برسد .بنابراین یک دایرکتوری غیر خالی هم به این صورت پاک می شود.برای مثال اگر مسیر کد بالا را به صورت زیر صدا بزنیم, ابتدا ghg- سپسaaa و در نهایت myDir پاک می شود:

```
Directory.Delete("C:\myDir", True)
```

Exist:متدی است که وجود یا عدم وجود یک دایرکتوری را مشخص می کند.مقدار برگشتی آن از جنس boolean بوده و اگر True باشد یعنی دایرکتوری مورد نظر وجود دارد و اگر False باشد یعنی وجود ندارد.آرگومان ورودی این متد نیز مسیر آن دایرکتوری است:

```
Dim path As String = "C:\myDir\aaa\ghg"
  If Directory.Exists(path) Then
      Response.Write("The Directory Exist")
  Else
      Response.Write("The Directory Not Exist")
End If
```

GetCreationTime:زمان ساخت دایرکتوری مورد نظر را به من می دهد.این متد این زمان را از خود فولدر دریافت می کند.این زمان را می توانید در قسمت Properties فولدر مورد نظر هم بیابید:

```
Response.Write(Directory.GetCreationTime(path))
```

GetLastAccessTime:متدی برای بازیابی آخرین دسترسی به دایرکتوری است.ولی این متد به تنهایی کار نمی کند.زیرا جایی را ندارد که از آن بازیابی کند زیرا در قسمت Properties فولدر مورد نظر تنها زمان ساخت فولدر را دارد نه آخرین دسترسی.پس برای استفاده از این متد باید به طور دستی عمل کنید.یعنی با متد متناظرش به نام SetLastAccessTime آن را مقدار دهی کنید و سپس با متد GetLastAccessTime از آن استفاده کنید.مثلا یک تاریخ را با متد SetLastAccessTime به دایرکتوری مورد نظر بدهید و سپس با متد GetLastAccessTime آن را بازیابی کنید:

```
   Directory.SetLastAccessTime(path, New DateTime(2003, 5, 4))
Response.Write(Directory.GetLastAccessTime(path))
```

در این مثال من تاریخ 4-5-2003 را به عنوان آخرین دسترسی به آن دادم و سپس همان را بازیابی کردم.کاربرد این کد هنگامی است که شما می خواهید زمان آخرین دسترسی را جایی ذخیره کنید تا دوباره از آن استفاده کنید.در آن هنگام در رویداد مور نظر خود هر وقت که به مسیر مورد نظر دستررسسی پیدا کردید می توانید آن را ذخیره کنید و هر جایی از آن استفاده کنید.نکته ی مهم این است که وقتی شما این زمان را set می کنید ,این زمان در حقیقت ذخیره می شود و هر جا و هر وقت می توانید آن را بازیابی کنید.

GetLastWriteTime:طرز کارش دقیقا مثل GetLastAccessTime است تنها زمانی کاربرد دارد که در مسیر مورد نظر چیزی نوشته می شود.مثلا یک فایل جدید در مسیر مورد نظر ایجاد می کنید.عمل Set هم با متد setLastWriteTime صورت می گیرد.

GetDirectories:تمام فولدر های موجود در مسیری را که به عنوان آرگومان ورودی میگیرد را به صورت آرایه ای از رشته ها برمی گرداند.بنابراین برای استفاده باید ابتدا آن را درون یک آرایه ی رشته ای قرار دهی و سپس از آن آرایه عمل بازیابی را انجام دهید:

```
Dim path As String = "C:\myDir"
Dim mystr() As String = Directory.GetDirectories(path)
  For Each i As String In mystr
    Response.Write(i)
    Response.Write("<br/>")
Next
```

خروجی کد فوق لیست تمامی دایرکتوری(فولدر) های موجود در درایو C کامپیوتر شماست.

پس ابتدا یک آرایه از رشته ها به نام mystr() تعریف کردم و مقدار برگشتی متد GetDirectories را در آن قرار دادم و سپس با یک حلقه ی for each اسامی فایل ها و فولدر های موجو در آن مسیر را یکی یکی به خروجی بردم.

GetFiles:طرز کارش دقیقا مثل GetDirectories است با این تفاوت که به جای دایرکتوری های موجود در مسیر مورد نظر , فایل های موجود در مسیر مورد نظر را بازیابی می کند:

```
Dim mystr2() As String = Directory.GetFiles("C:\")
For Each i As String In mystr2
    Response.Write(i)
    Response.Write("<br/>")
Next
```

خروجی کد فوق لیست تمامی فایل های موجود در درایو C کامپیوتر شماست.

نکته ی دیگر در زمینه ی دو متد معرفی شده در بالا الگوی بازیابی است.هر دو متد بالا حاوی یک آرگومان ورودی دیگر علاوه بر مسیر هستند که یک رشته به عنوان آرگومان ورودی دریافت می کنند.این رشته الگویی برای جستجوی فایل(دایرکتوری) مورد نظر است.این الگو به صورت زیر است:

filename.fileSuffix

در این الگو علامت * به معنای هر چیز می باشد.مثلا اگر الگوی من در متد GetFiles به صورت *.*a باشد این به این معنا است که تمام فایل هایی که نامشان با حرف a آغاز می شود را به من بده(با هر پسوندی).و یا as*.txt یعنی تمام فایل های متنی که نامشان با as آغاز می شود را به من برگردان و یا در GetDirectories رشته ی *a یعنی تمام دایرکتوری های موجود که نامشان با حرف a آغاز شده را برگردان.کاربرد الگوی جستجو در کد زیر کاملا واضح است:

```
  Dim mystr2() As String = Directory.GetFiles("C:\", "a*.*")
  For Each i As String In mystr2
      Response.Write(i)
      Response.Write("<br/>")
Next
```

غیر از علامت * علامت ؟ نیز وجود دارد که برای تطبیق تک کاراکتر ها به کار می رود.این علامت حداکثر کاراکتر ها را بیان می کند مثلا a??? یعنی دایرکتوری هایی که نامشان با a شروع شده باشد و حداکثر پس از a ۳ حرف دیگر داشته باشد.

GetLogicalDrives:اسامی تمام درایو های موجود در کامپیوتر شما را به صورت آرایه ای از رشته ها برمی گرداند:

```
  Dim mystr3() As String = Directory.GetLogicalDrives
   For Each i As String In mystr3
       Response.Write(i)
       Response.Write("<br/>")
 Next
```

GetCurrentDirectory:این متد مسیری که برنامه ی شما از آن اجرا می شود را بر می گرداند:

```
Response.Write(Directory.GetCurrentDirectory())
```

Move:-عمل انتقال فایل ها و فولدر های یک مسیر(که مسلما قبلا وجود داشته) را به مسیر جدید که قبلا وجود نداشته را انجام می دهد و سپس مسیر اول را پاک می کند.دو آرگومان ورودی دارد.اولی مسیر اصلی است و دومی مسیری است که قبلا وجود نداشته و

داده های مسیر اول باید به داخل آن منتقل شوند.طرز کار این متد دقیقا مثل حرکت یک عنصر از جایی به جای دیگر است.مثلا اگر یک عنصر را از جایی به جای دیگر حرکت دهید دیگر در مکان اولیه قرار ندارد:

```
Directory.Move("C:\myDir", "C:\oooj")
```

در کد بالا تمام فایل ها و فولدر های داخل myDir به داخل oooj می روند و سپس دایرکتوری myDir پاک می شود.متد move شی Directory کاربردش تنها در یک درایو مشترک است و نمی تواند انتقال دایرکتوری ها را بین درایو ها هم انجام دهد.

نکته ی دیگر هم این است که می توان عمل Move شی Directory را یک جور Rename نیز تلقی کرد.

خوب حالا که با متد های مهم شی Directory آشنا شدید به سراغ شی File و متد های آن می رویم.

:File

شی است که قابلیت کنترل و دسترسی به فایل های مورد نظر را به ما می دهد.

این شی حاوی متد های مختلفی است که به ترتیب با آنها آشنا می شویم.

Copy:متدی است که عمل کپی را انجام می دهد.دو آرگومان ورودی دارد که اولی مسیر فایلی است که عمل کپی می خواهد از محتویاتش صورت گیرد.و آرگومان دوم هم مسیر فایلی است که اولا هم جنس فایل اول باشد و دوما قبلا وجود نداشته باشد.مثلا فرض کنید یک فایل txt به نام a در درایو C داریم.و کد زیر را برای آن می نویسیم:

```
  Dim path1 As String = "C:\a.txt"
  Dim path2 As String = "C:\b.txt"
File.Copy(path1, path2)
```

در این کد , ابتدا فایل جدیدی به نام b.txt ساخته می شود و سپس محتوای فایل a.txt در داخل آن کپی می شود.در ضمن پس از عمل کپی , فایل a.txt پاک نمی شود.

اگر خواستید محتویات یک فایل به فایلی که قبلا وجود داشته منتقل شود باید از آرگومان ورودی سوم آن استفاده کنید که OverWrite بودن عمل کپی را True یا False قرار می دهد:

```
        Dim path1 As String = "C:\a.txt"
        Dim path2 As String = "C:\b.txt"
    File.Copy(path1, path2, True)
```

در کد بالا هر دو فایل a.txt & b.txt وجود دارند و هر یک حاوی یک سری نوشته هستند.با اجرا کد بالا هر چیز که در فایل a.txt وجود داشت روی نوشته های موجود در

فایل b.txt OverWrite می شود.در حقیقت عمل کپی صورت می گیرد ولی چون به صورت OverWrite این عمل صورت می گیرد تمام محتوی فایل b.txt پاک می شود و هر چیز که در فایل a.txt وجود داشت در داخل b.txt قرار می گیرد.

Delete:برای حذف کردن یک فایل به کار می رود.مسیر فایل نیز به عنوان آرگومان ورودی به ان داده می شود:

```
File.Delete("C:\a.txt")
```

Exist:وجود یا عدم وجود یک فایل را چک می کند:

```
  If File.Exists("C:\a.txt") Then
     Response.Write("File Exist!")
  Else
     Response.Write("File Not Exist!")
End If
```

GetAttributes & SetAttributes:متدی است که با آن می توان به ویژگی یک فایل دسترسی داشتو یا آن را تنظیم کرد.برای درک منظور از ویژگی یک فایل روی یک فایل راست کلیک کنید و وارد Properties آن شوید و در انتهای آن یک قسمت به عنوان attribute می بینید که حاوی ۳ مقدار Read-Only---Hidden----Archive است که جلوی هر یک CheckBox وجود دارد.متد GetAttributes آن عنصری را که CheckBox جلویش تیک خورده باشد را برمی گراند.مثلا اگر فایلی hidden باشد با متد GetAttributes می فهمیم که آیا مخفی هست یا نه.و همینطور با متد SetAttributes می توان یک فایل را مخفی کرد و یا read-only و ....تنها آرگومان ورودی متد GetAttributes مسیر فایل مورد نظر است:

```
Response.Write(File.GetAttributes("C:\a.txt"))
```

متد SetAttributes دو آرگومان دارد که اولی مسیر فایل مورد نظر و دومی ویژگی است که قرار است به آن بدهیم:

```
File.SetAttributes(path1, FileAttributes.Hidden)
```

همانطور که می بینید با شی FileAttribtes نوع وِیژگی را مشخص کردیم که در اینجا hidden است.

GetCreationTime:زمان ساخت یک فایل را برمی گرداند.

GetLastAccessTime & GetLatsWriteTime:زمان آخرین دسترسی به فایل را مشخص می کند.اگر به قسمت Properties یک فایل بروید می بینید که علاوه بر

Created که زمان ساخت یک فایل را مشخص می کند دو عنر دیگر وجود دارد.اولی Modified که آخرین زمان تغییر در فایل مثلا نوشتن در آن مشخص می شود و دومی Accessed است که آخرین زمان دسترسی (خواندن) فایل در آن ذکر شده.پس در اینجا مثل Directory ها نیازی به استفاده از متد SetLastAccessTime و یا SetLastWriteTime نیست.زیرا زمان های واقعی آخرین تغییر و آخرین دسترسی خود به خود در یک فایل وجود دارد و شما کافی است تنها آن ها را بازیابی کنید:

```
Dim path1 As String = "C:\a.txt"
  Response.Write(File.GetCreationTime(path1))
  Response.Write("<br/>")
  Response.Write(File.GetLastWriteTime(path1))
  Response.Write("<br/>")
Response.Write(File.GetLastAccessTime(path1))
```

Move:عمل انتقال داده های یک فایلی که قبلا وجود داشته را به فایلی که قبلا وجود نداشته را انجام می دهد و فایل اول را پاک می کند.آرگومان ورودی اول مسیر فایل اول (که قبلا وجود داشت) و آرگومان ورودی دوم مسیر فایل دوم است که قبلا وجود نداشته و قرار است ایجاد شود.در ضمن مثل بحث Directory ها پس از عمل انتقال, فایل اول پاک می شود:

```
File.Move("C:\a.txt", "C:\new.txt")
```

در کد بالا ابتدا فایل new.txt ساخته شده و سپس محتویات فایل a.txt به درون ان ریخته شده و در نهایت فایل a.txt پاک می شود.

برخلاف متد Move شی Directory متد Move شیFile قابلیت انتقال بین درایو ها را دارد.

Create:با آن می توان یک فایل با هر پسوند دلخواه ایجاد کرد.در کد زیر من یک فایل با نام a.txt ایجاد کردم:

```
File.Create("C:\a.txt")
```

مقدار برگشتی این متد از جنس FileStream است و این شی به ما اجازه ی نوشتن در فایل ایجاد شده را می دهد.

CreateText: با آن می توان یک فایل با هر پسوند دلخواه ایجاد کرد مثل متد Create , با این تفاوت که مقدار برگشتی آن از جنس StreamWriter است که با آن جلوتر آشنا می شویم و به ما اجازه ی نوشتن در فایل ایجاد شده را می دهد.

OpenRead() & OpenText() & Openwrite: متد OpenText فایل Text که با UTF-8 Unicode نوشته شده را می خواند و توسط یک شی StramReader آن را برمی گرداند. متدOpenRead فایل را برای خواندن باز می کنند و مقدار برگشتی آن از جنس FileStream ولی Read-only است. وOpenwrite نیز برای نوشتن , یک فایل را باز می کند و این کار را با مقدار برگشتی از جنس FileStream انجام می دهد.

ReadAllText:کل مقادیر موجود در یک فایل را می خواند و به صورت یک String ساده بر می گرداند:

```
Response.Write(File.ReadAllText("C:\c.txt"))
```

در ضمن, متد بالا تمام مقادیر فایل c.txt را پشت سر هم بر می گرداند مثلا اگر بین داده ها چندین سطر هم فاصله باشد آن داده ها را کنار هم نمایش می دهد.

ReadAllLines: کل مقادیر موجود در یک فایل را می خواند و خروجیش آرایه ای از رشته ها است.به این صورت که هر خانه ی این آرایه یک خط از فایل مورد نظر است.مثلا فرض کنید فایل a.txt به صورت زیر باشد:

```
ali
reza

javad-              soosan
```

اگر کد ما به صورت زیر باشد:

```
  Dim str() As String = File.ReadAllLines("C:\a.txt")
  For Each s As String In str
     Response.Write(s)
     Response.Write("<br/>")
Next
```

ابتدا تک تک سطر های فایل a.txt در تک تک خانه های آرایه ی str تا آنجا که تعداد سطر ها ادامه پیدا کند ذخیره می شود سپس در هر بار گردش حلقه ی for , یک عنصر از آرایه ی str() نمایش داده می شود و از آنجایی که این عنصر یک سطر مجزا است پس در هر بار چرخش حلقه ی for , یک سطر از فایل مورد نظر به خروجی می رود.

ReadAllBytes:این متد یک ارایه ای از بایت را از یک فایل برمی گرداند و برای فایل های کوچک کاربرد دارد نه فایل های بزرگ.مثلا می توانید یک عکس کوچک را به صورت byte بخوانید و آن را در یک آرایه از بایت ها قرار دهید و سپس از آن استفاده کنید:

```
  Dim j() As Byte = File.ReadAllBytes("C:\leaf.GIF")
  For Each s As Byte In j
```

```
        Response.Write(s)
Next
```

WriteAllText:برای نوشتن در یک فایل کاربرد دارد.دو آرگومان ورودی دارد که ولی مسیر فایلی است که می خواهید چیزی در آن بنویسید و دومی متن مورد نظر است.در این متد رفتن به سطر بعد بی معنی بودهو همه چیز را در یک سطر چاپ می کند:

```
File.WriteAllText("C:\a.txt", "ali reza Javad ")
```

اگر دو بار از این متد استفاده کنید text های اول پاک می شوند و text های جدید به جایش در فایل نوشته می شوند.اگر بخواهید به دنباله ی نوشته های قبلی خود text بیفزایید باید از متد AppendAlltext استفاده کنید که ارگومان هایش مثل WriteAllText است ولی تنها نوشته های جدید را به ادامه ی نوشته های قبلی موجود در فایل اضافه می کند.

WriteAllLines:یک آرایه از رشته ها به عنوان آرگومان وودی دوم دریافت می کند و هر یک از عناصر آن را در یک سطر می نویسد:

```
  Dim str() As String = {"ali", "reza", "javad", "soosan", "hellow word !"}
File.WriteAllLines("C:\a.txt", str)
```

خروجی کد فوق یک فایل a.txt با محتوی زیر است:

```
ali
reza
javad
soosan
! hellow word
```

در متد های بالا اگر فایلی با مسیر داده شده به عنوان آرگومان ورودی وجود نداشت آنگاه آن فایل ایجاد می شود و سپس مقدیر در آن نوشته می شوند ولی اگر فایل هاداز قبل موجود باشند آنگاه نوشن در آنها به صورت OverWrite صورت می گیرد.

به عنوان یک مثال ساده می خواهیم به صورت دینامیک یک فایل txt در درایو C ایجاد کنیم و سپس لیست تمام فایل های درایو C کامپیوتر را در آن قرار دهیم.پس ابتدا تمام فایل ها را بازیابی می کنیم و آنها را در یک آرایه از رشته ها قرار می دهیم:

```
Dim str() As String = Directory.GetFiles("C:\")
```

چون می خواهم از WriteAllLines استفاده کنم پس آرایه ی بالا را به عنوان آرگومان دومش در نظر می گیرم:

```
  Dim newPath As String = "C:\listOfFiles.txt"
File.WriteAllLines(newPath, str)
```

خوب حالا با دو شی FileInfo و DirectoryInfo آشنا می شویم.

این دو متد به ما اجازه می دهند که روی دایرکتوری ها فایل های آنها پیمایش انجام دهیم.از طرف دیگر می توان از این دو به صورت ترکیبی استفاده کرد مثلا می توان به فایلی دسترسی داشت که با DirectoryInfo قابل دسترسی است.پس چون در بعضی از موارد این دو شی ترکیبی عمل می کنند متد های آنها را به ۳ بخش تقسیم می کنیم.بخش اول متد های مشترک-بخش دوم متد های DirectoryInfo و بخش سوم هم متد های FileInfo.نکته ی مهم دیگر این است که این دو شی مخصوص یک مسیر و یا یک فایل خاص هستند و برای استفاده باید حتما متغیری از جنس این دو تعریف کنیم و سپس از آنها استفاده کنیم:

```
  Dim dinfo As New DirectoryInfo("C:\a")
Dim finfo As New FileInfo("C:\a.txt")
```

نکته ی مهم دیگر این است که در دو شی DirectoryInfo & FileInfo متد های Get و Set وجود ندارند.مثلا ما در باره ی fileInfo گفتیم برای دادن یک ویژگی باید از متد SetAttributes و برای بازیابی آن از متد GetAttributes استفاده کنیم.در حالیکه در اینجا یک ویژگی به نام Attribute وجود دارد که عمل read و write یا همان Get و Set را با هم انجام می دهد.

متد های مشترک DirectoryInfo & FileInfo:

مثال هایی که از متد های زیر خواهیم زد بر اساس دو تعریف زیر خواهد بود:

```
  Dim dinfo As New DirectoryInfo("C:\a")
Dim finfo As New FileInfo("C:\a.txt")
```

Attributes:به شما اجازه ی تنظیم و بازیابی ویژگی یک فایل یا یک دایرکتوری را می دهد.برای مثال از بازیابی داریم:

```
Response.Write(dinfo.Attributes)
```

و از تنظیم:

```
dinfo.Attributes = FileAttributes.Archive
```

می بینید که در DirectoryInfo و FileInfo متد های ..GetAtt و ..SetAtt وجود ندارد.

CreationTime:اجازه ی تنظیم و بازیابی زمان ساخت فایل و یا دایکتوری که در سازنده ی DirectoryInfo و FileInfo وجود داشت را می دهد. برای مثال از بازیابی:

```
Response.Write(dinfo.CreationTime)
```

از تنظیم:

```
dinfo.CreationTime = New DateTime(2004, 3, 3)
```

توجه کنید که نیازی به تنظیم در متد CreationTime نیست زیرا هر دو کلاس DirectoryInfo و FileInfo زمان ساخت واقعی را می توانند بازیابی کنند.

LastAccessTime: اجازه ی تنظیم و بازیابی زمان آخرین دسترسی به فایل و یا دایکتوری که در سازنده ی DirectoryInfo و FileInfo وجود داشت را می دهد.برای مثال:

```
  dinfo.LastAccessTime = New DateTime(2005, 4, 26)
Response.Write(dinfo.LastAccessTime)
```

البته یادتان باشد که فایل ها زمان واقعی آخرین دسترسی را در قسمت Properties خود ذخیره می کنند و نیاز به تنظیم ندارند ولی دایرکتوری ها حتما نیاز به تنظیم دارند.

LastWriteTime: اجازه ی تنظیم و بازیابی زمان آخرین تغییرات در فایل و یا دایکتوری که در سازنده ی DirectoryInfo و FileInfo وجود داشت را می دهد.برای مثال:

```
  dinfo.LastWriteTime = New DateTime(2006, 4, 26)
Response.Write(dinfo.LastWriteTime)
```

البته یادتان باشد که فایل ها زمان واقعی آخرین تغییرات را در قسمت Properties خود ذخیره می کنند و نیاز به تنظیم ندارند :

```
Response.Write(finfo.LastWriteTime)
```

ولی دایرکتوری ها حتما نیاز به تنظیم دارند.

Exist:وجود و یا عدم وجود یک فایل یا دایرکتوری را مشخص می کند.برای مثال:

```
  If finfo.Exists Then
      Response.Write("file exist")
  Else
      Response.Write("file Not exist")
End If
```

FullName:نام کامل دایرکتوری یا فایل را به همراه مسیرش برمی گرداند.این نام شامل نام درایو و پسوند نیز می شود:

```
Response.Write(finfo.FullName)
```

خروجی کد بالا C:\a.txt است.

Name:تنها نام و پسوند(اگر فایل باشد) فایل یا دایرکتوری را برمی گرداند:

```
Response.Write(finfo.Name)
```

Extension:پسوند فایل مورد نظر را بر می گرداند:

```
Response.Write(finfo.Extension)
```

Delete:یک فایل و یا یک دایرکتوری خالی را پاک می کند.اگر دایرکتوری خالی نبود آرگومان ورودیش را برابر True قرار دهید:

```
dinfo.Delete(True)
finfo.Delete()
```

Refresh:باعث به روز شدن اطلاعات در بحث همزمان سازی می گردد.زمانی کاربرد دارد که از طریق Windows Explorer تغییری در فایل ایجاد شود که در برنامه ی ما به روز نشده.

Create:یک فایل یا دایرکتوری جدید ایجاد می کند:

```
  Dim j As New FileInfo("C:\ggg.txt")
j.Create()
```

در اینجا یک فایل جدید به نام ggg از نوع متنی در درایو C ایجاد شده.

MoveTo:وظیفه ی انتقال محتوی یک فایل و یا یک دایرتوری را به فایل و یا دایرکتوری جدید بر عهده دارد.فایل و یا دایرکتوری مبدا در تابع سازنده ی مربوطه ایجاد شده اند پس در متد MoveTo کافی است تنها مسیر فایل مقصد را بدهیم.دقت کنید فایل مقصد از قبل وجود نداشته باشد چون متد MoveTo قابلیت OverWrite نداشته و حتما فایل(دایرکتوری) جدید را ایجاد می کند و سپس محتویات فایل(دایرکتوری) قبلی را در آن می ریزد و فایل(دایرکتوری) قدیمی را پاک می کند.در مثال های زیر هم دایرکتوری به نام fff و فایلی به نام ggg.txt از قبل وجود نداشتند:

```
  dinfo.MoveTo("C:\fff")
finfo.MoveTo("C:\ggg.txt")
```

حال می رسیم به متد های مخصوص DirectoryInfo:

Parent & Root:متد Parent فولدر والد دایرکتوری جاری را می دهد:

```
Dim dinfo1 As New DirectoryInfo("C:\a\b")
Response.Write(dinfo1.Parent)
```

خروجی کد فوق a می باشد.root هم نام درایو مورد نظر را برمی گرداند.مثال از root هم به صورت زیر است:

```
Response.Write(dinfo1.Root)
```

خروجی کد فوق هم C:\ خواهد بود.

CreateSubDirectory:یک زیر دایرکتوری ایجاد می کند.اگر مایل باشید این زیر دایرکتوری در داخل دایرکتوری که در تابع سازنده ی DirectoryInfo مشخص شده ایجاد شود کافیست تنها یک نام به عنوان آرگومان ورودی متد CreateSubDirectory بدهید:

```
Dim dinfo As New DirectoryInfo("C:\a")
```

```
dinfo.CreateSubdirectory("rtr")
```

در این کد یک فولدر به نام rtr در داخل C:\a ساخته می شود.و یا در کد زیر ۳ فولدر تودر تو در داخل C:\a ساخته می شود ساخته می شود:

```
dinfo.CreateSubdirectory("y\o\l")
```

مقدار برگشتی متد CreateSubDirectory از جنس DirectoryInfo بوده و پس از ساختن, می توان از آن به عنوان یک DirectoryInfo استفاده کرد:

```
  Dim info As DirectoryInfo = dinfo.CreateSubdirectory("rtr")
info....
```

GetDirectories: دایرکتوری های موجود در داخل دایرکتوری مورد نظر را به عنوان یک آرایه از DirectoryInfo برمی گرداند.در کد زیر اسامی آنها را بازیابی کردیم :

```
  Dim di() As DirectoryInfo = dinfo.GetDirectories()
  For Each d As DirectoryInfo In di
      Response.Write(d.Name)
      Response.Write("<br/>")
Next
```

یک آرگومان وروردی این متد, SearchPattern است که به شما امکان فیلتر کردن دایرکتوری های بازیابی شده را می دهد.در مورد الگوی بازیابی پیش از این توضیح داده شده بود.

GetFiles: فایل های موجود در داخل دایرکتوری مورد نظر را به عنوان یک آرایه از FileInfo برمی گرداند.در کد زیر اسامی آنها را بازیابی کردیم :

```
  Dim di() As FileInfo = dinfo.GetFiles
  For Each d As FileInfo In di
     Response.Write(d.Name)
     Response.Write("<br/>")
Next
```

یک آرگومان وروردی این متد, SearchPattern است که به شما امکان فیلتر کردن دایرکتوری های بازیابی شده را می دهد.در مورد الگوی بازیابی پیش از این توضیح داده شده بود.

حال می رسیم به متد های مخصوص FileInfo:

Directory:نام دایرکتوری که فایل مورد نظر در آن قرار دارد را از جنس DirectoryInfo برمی گرداند.پس می توان از آن نیز استفاده کرد:

```
Response.Write(finfo1.Directory)
```

Lenght:سایز فایل را به واحد Byte(64Bit-integer) به ما می دهد.در کد زیر من حجم فایلی را که درسازنده ی متغیر finfo2 مشخص کرده بودم را بازیابی و سپس به KB تبدیل کردم و به خروجی بردم:

```
Dim finfo2 As New FileInfo("C:\note.GIF")
  Dim a As Integer = finfo2.Length
  Dim b As Decimal = a / 1024
Response.Write(CStr(b) & " KB")
```

CopyTo:عمل کپی کردن مقادیر یک فایل را در یک فایل دیگر انجام می دهد.اگر فایل مقصد وجود نداشته باشد آن را ایجاد می کند و سپس محتویات فایل مبدا را در آن می ریزد ولی اگرفایل مقصد وجود داشته باشد باید OverWrite آن را برابر True کنید تا با خطا مواجه نشوید.

Create:با آن می توان یک فایل با هر پسوند دلخواه ایجاد کرد.در کد زیر فایلی به نام a.txt در درایو C وجود نداشت و من آن را ایجاد کردم:

```
Dim finfo3 As New FileInfo("C:\a.txt")
finfo3.Create()
```

مقدار برگشتی این متد از جنس FileStream است و این شی به ما اجازه ی نوشتن در فایل ایجاد شده را می دهد.

CreateText: با آن می توان یک فایل با هر پسوند دلخواه ایجاد کرد مثل متد Create , با این تفاوت که مقدار برگشتی آن از جنس StreamWriter است که با آن جلوتر آشنا می شویم و به ما اجازه ی نوشتن در فایل ایجاد شده را می دهد.

OpenRead() & OpenText() & Openwrite: متد OpenText فایل Text که با UTF-8 Unicode نوشته شده را می خواند و توسط یک شی StramReader آن را برمی گرداند. متدOpenRead فایل را برای خواندن باز می کنند و مقدار برگشتی آن از جنس FileStream ولی Read-only است. وOpenwrite نیز برای نوشتن , یک فایل را باز می کند و این کار را با مقدار برگشتی از جنس FileStream انجام می دهد.

یک نکته در زمینه ی شی DirectoryInfo باقی می ماند و آن هم این است که این شی مثل FileInfo حاوی متد Lenght نیست که سایز یک فولدر(دایرکتوری) را به ما بدهد.پس برای رسیدن به این مقصد خودمان باید یک تابع برای آن بنویسیم.ابتدا تابه را به صورت زیر می نویسیم که آرگومان ورودی اول شی DirectoryInfo است که همان

فولدری است که می خواهیم سایزش را محاسبه کنیم و ارگومان دوم یک مقدار Boolean است که مشخص می کند آیا فولدر مورد نظر خود حاوی ولدر های دیگری هم هست یا خیر:

```
  Public Function GetDirSize(ByVal dirInfo As DirectoryInfo, ByVal HaveSubDir
As Boolean) As Integer

End Function
```

برای نوشتن این تابع دو نوع سایز متفوت باید حساب شود.اولی سایز تمام فایل های موجود در دایرکتوری است.این کار با بازیابی فایل های یک دایرکتوری با متد GetFiles شی DirectoryInfo و سپس بدست آوردن سایز تک تک این فایل ها با متد lenght امکان پذیر است:

```
        Dim TotalSize As Long = 0
        For Each f As FileInfo In dirInfo.GetFiles
            TotalSize += f.Length
    Next
```

سپس چک می کنیم اگر HaveSubDir برابر True بود یعنی دایرکتوری مورد نظر حاوی زیر شاخه است پس در آن موقع باید برای تک تک SubDirectories آن یک بار تابع را به صورت بازگشتی صدا بزنیم:

```
        If HaveSubDir Then
            For Each dir As DirectoryInfo In dirInfo.GetDirectories
                TotalSize += GetDirSize(dir, True)
            Next
    End If
```

پس از چک کردن شرط بازگشت, تمام فولدر های موجود در فولدری که می خواهیم سایزش را حساب کنیم را با متد GetDirectories به شکل آرایه ای از DirectoryInfo در آوردیم و عینا تابع را مجددا برای هر فولدر صدا زدیم.به این ترتیب اگر ۱۰۰۰ تا فولدر تودر تو هم داشته باشیم سایزشان را می توانیم بدست آوریم.

و در نهایت سایز نهایی را برگردانیم:

```
    Return TotalSize
```

پس تابع کلی به فرم زیر شد:

```
    Public Function GetDirSize(ByVal dirInfo As DirectoryInfo, ByVal
HaveSubDir As Boolean) As Integer
        Dim TotalSize As Long = 0
        For Each f As FileInfo In dirInfo.GetFiles
            TotalSize += f.Length
        Next
        If HaveSubDir Then
```

```
            For Each dir As DirectoryInfo In dirInfo.GetDirectories
                TotalSize += GetDirSize(dir, True)
            Next
        End If
        Return TotalSize
    End Function
```

من از آن به صورت زیر استفاده کردم:

```
Response.Write(GetDirSize(New DirectoryInfo("C:\a"), True))
```

پس می بینید توابعی که تا اینجا معرفی کردیم ساختار پیچیده ای نداشتند.

کلاس جدید و ساده ای که پس از DirectoryInfo & FileInfo معرفی می کنیم کلاس DriveInfo است که خصوصیات درایوی را که به عنوان ارگومان ورودی تابع سازنده به آن می دهیم را بازیابی می کند.متد های آن در زیر آمده:

TotalSize:حجم کلی درایو را بر حسب بایت به ما می دهد.

TotalFreeSpace:حجم کلی خالی درایو را حسب بایت به ما می دهد.

AvilableFreeSpace: حجم کلی خالی قابل استفاده ی درایو را حسب بایت به ما می دهد.حجم قابل استفاده ممکن است گاهی اوقات کمتر از حجم کلی خالی درایو باشد.مثلا من قسمتی از حجم یک درایوم را به Ram اضافه کردم.پس حافظه ی قابل استفاده از تفریق حجم کلی خالی درایو و حجمی که به رم اضافه کردم بدست می آید.

DriveFormat:نام FileSystem را مشخص می کند NTFS یا FAT32.

DriveType:نوع درایو را برمی گرداند مثلا CD-Rom یا RemovableDisc و...

IsReady:مشخص می کند درایو مورد نظر آیا برای خواندن و نوشتن آماده است یا خیر.کاربرد آن در Flash Memory هاست.که گاهی وصل(آماده ی خواندن و نوشتن) و گاهی قطع هستند.

Name:نام درایو را برمی گرداند مثلا C یا E.

```
    Dim j As New DriveInfo("E")
Response.Write(j.TotalFreeSpace)
```

در کد فوق من حجم خالی درایو E را به خروجی بردم(البته بر حسب بایت).

متد FileInfo نمی تواند جزییات فایل های exe و dll را برای ما بازیابی کند. شما می توانید این جزییات را در Version tab از Properties فایل مربوطه ببینید.به همین دلیل

ما نیاز به یک شی داریم که جزییات فایل های exe و dll را برای ما بازیابی کند. این شی FileVersionInfo نام دارد.که در فضای نام system.Diagnostics موجود است.برای استفاده از آن از متد GetVersionInfo استفاده می کنیم و مسیر فایل dll یا exeی مربوطه را به عنوان آرگومان ورودی به آن می دهیم.نحوه ی ایجاد آن به فرم زیر است:

```
Dim path As String = "C:\Program Files\Winamp\winamp.exe"
Dim finfo As FileVersionInfo = FileVersionInfo.GetVersionInfo(path)
```

قبل از استفاده بهتر است با یک سری از ویژگی های شی FileVersionInfo آشنا شویم:

FileVersion:شماره ی نسخه ی فایل.

FileName:نام فایل را از قسمت Description بازیابی می کند.

OriginalFileName:نام اصلی فایل.

FileDescription:نام Internal فایل در صورت وجود.

CompanyName:نام کمپانی تولید کننده ی فایل.

و...

در زیر مثالی را در این زمینه می بینید:

```
Dim path As String = "C:\Program Files\Winamp\winamp.exe"
Dim finfo As FileVersionInfo = FileVersionInfo.GetVersionInfo(path)
Response.Write(finfo.FileVersion)
Response.Write("<br/>")
Response.Write(finfo.FileName)
Response.Write("<br/>")
Response.Write(finfo.ProductName)
Response.Write("<br/>")
Response.Write(finfo.InternalName)
```

حالا می خواهیم با کلاس Path آشنا شویم.

همانطور که می دانید ما تا اینجا با فایل ها و فولدر ها کار کردیم.ولی آیا به این توجه کردید که اجزای برنامه ی ما نیز از یک سری فایل و فولدر تشکیل شده؟مثلا حساس ترین فایل ما که Web.config است نیز یک فایل در فولدر وب سایت ماست.و همچنین سایر صفحات که با پسوند aspx.vb. در دایرکتوری وب سایت ما موجود هستند.آیا ممکن است هکر ها بتوانند به این فایل ها دسترسی داشته باشند؟کلاس path تا حدی جلوی این

مشکلات را میگیرد.برای درک این مشکل بیایید یک مثال با هم ببینیم ولی قبل از آن یک نکته ی مهم را متذکر می شویم.

فرض کنید وب سایت من در آدرس C:\my practices\practice19 قرار دارد.اگر به انتها ی این آدرس عبارت filename/.. را اضافه کنیم اتفاقی که می افتد این است که به مسیر C:\my practices \filename رجوع می شود.یعنی نام فایلی که به انتهای آن افزودیم در فولدر practice19 به آن رجوع می شود.حال اگر در وب سایتم یک فولدر دیگر به نام myfiles وجود داشته باشد برای دسترسی به آن مسیرم به صورت C:\my practices\practice19\myfiles خواهد بود و حالا اگر filename/.. را به آن بیفزاییم جاب است که باز هم به C:\my practices\practice19\filename رجوع می شود.این اتفاقات به این دلیل می افتد که من از /.. استفاده کردم.این عبارت باعث ارتباط به والد دایرکتوری جاری می شود.

حال به مثال خود بپردازیم.یک Button-Label-TextBox روی صفحه قرار دهید ودر رویداد کلیک دکمه چنین بنویسید:

```
Label1.Text = File.ReadAllText("\my practices\practice19\myfiles\" &
TextBox1.Text)
```

می خواهیم محتویات یک فایل را که نامش را شما در یک TextBox می نویسید را در خروجی نمایش دهیم.جالب اینجاست که تمام فایل های برنامه ی من در \my practices\practice19قرار دارد و من برای کمی امنیت بیشتر فایل هایی که شما (به عنوان یک کاربر) می خواهید بازیابی کنید را در یک فولدر در داخل فولدر اصلی سایتم(به نام myfiles) قرار دادم تا در آن فولدر به دنبال فایلی که نامش را شما درTextBox وارد می کنید بگردد.اگر مثال را اجرا کنید می بینید که به راحتی به فایل های فولدرmyfiles دسترسی دارید و می توانید محتویات آنها را ببینید مثلا اگر در TextBox عبارت a.txt را بنویسید محتوای فایل a.txt را ر خروجی می بینید.حال اگر در داخل TextBox عبارت default.aspx.vb/..(که سورس یکی از صفحات سایت شماست.اگر آن را ندارید به جای default می توانید نام فحه ی خود را قرار دهید) را بنویسید چه اتفاقی می افتد؟بله یک فاجعی نظیر فاجعی ای که در بحث Sql Injection بیان شد.سورس یکی از صفحات شما در خروجی به کاربر نشان داده می شود.آیا از این بدتر هم می شود که یک هکر وارد صفحات سایت شما شود مثلا login.aspx و با دیدن نام صفحه, یک عبارت vb. یا cs. به انتهای آن اضافه کند و آن را وارد textbox مربوطه

کند و سورس کد های Server-side شما را ببیند آن هم روی صفحه ای که خودتان ساختید.پس یک مشکل بزرگ نظیر Sql Injection رخ داده ولی خوشبختانه به سادگی قابل حل است.راه حلش هم استفاده از کلاس Path است.این کلاس یک متد به نام GetFileName دارد که این متد نام یک فایل را از دل یک مسیر استخراج می کند مثلا:

Path.GetFileName(“C:\a\b\c\d.txt”)= d.txt

Path.GetFileName(“d.txt”)= d.txt

Path.GetFileName(“..\d.txt”)= d.txt

پس می بینید اگر کاربر از ../ هماستفاده کند شما می توانید به سادگی با متد GetFileName نام اصلی فایل را استخراج کنید.پس ما در مثال بالا استخراج را به صورت زیر انجام می دهیم و نام فایل را در یک متغیر به نام filename ذخیره می کنیم:

```
Dim fileName As String = Path.GetFileName(TextBox1.Text)
```

حال کافی است مسیر اصلی فایل را مشخص کنیم.خوب ما در بالا چنین عمل کردیم:

```
("\my practices\practice19\myfiles\" & TextBox1.Text)
```

در اینجا هم می توان چنین کرد ولی من برای پرهیز از بروز خطا(خطای syntax برای استفاده ی به جا از \ ها) از متد Combine شی Path به صورت زیر استفاده می کنم که دو آرگومان ورودی از نوع String دارد.که هر دو نوعی مسیر به حساب می آیند که متد Combine آنها را به هم می چسباند:

```
Label1.Text = File.ReadAllText(Path.Combine("\my
practices\practice19\myfiles\", fileName))
```

به این ترتیب هکر نمی تواند به فایل های server-side ما دسترسی داشته باشد.

البته مشکل فقط همین یک نوع نیست مثلا شما می دانید که کلاس های شما در یک فولدر به نام App_Code ذخیره می شوند.حال اگر در textBox مثال بالا عبارت ..\App_code\classname.vb نوشته می شد فاجعه ای بدتر از قبلی رخ می داد و کلاس های برنامه ی شما لو می رفت.

یک ذره هم به حاشیه می رویم:

اگر سرو کار شما به Url ها بود برای استفاده ی درست از آنها از شی uri استفاده کنید.این شی می تواند عبارات دلخواه شما را از دل Url بیرون بکشد مثلا اگر Url ما به فرم زیر باشد:

Http://www.mysite.com/page.aspx?cu=hellow

شی uri را به صورت زیر New می کنیم و لینک بالا را به ورودی سازنده ی آن می دهیم:

```
Dim theuri As New Uri("Http://www.mysite.com/page.aspx?cu=hellow")
```

حال برای بازیابی QueryString از متد Query شی uri استفاده می کنیم:

```
Response.Write(theuri.Query)
```

برای بازیابی آدرس صفحه ی محلی(که در اینجا page.aspx است) از متد localPath استفاده می کنیم:

```
Response.Write(theuri.LocalPath)
```

برای بازیابی آدرس سایت اصلی(www.mysite.com)از متد host استفاده می کنیم:

```
Response.Write(theuri.Host)
```

برای بازیابی port نیز از متد port استفاده می کنیم که چون در اینجا http است شماره ی Port ۸۰ خواهد بود:

```
Response.Write(theuri.Port)
```

و متد های دیگری که بسیار ساده هستند.حال از حاشیه خارج می شویم.

شی Path حاوی متد های دیگری نیز هست که در زیر آنها را مختصر معرفی می کنیم:

ChangeExtension:در مسیر فایل مورد نظر پسوند آن را عوض می کند :

```
Response.Write(Path.ChangeExtension("C:\a.txt", ".pdf"))
```

خروجی کد بالا C:\a.pdf خواهد بود.

GetDirectoryName:مسیر دایرکتوری که فایل مورد نظر در آن قرار دارد را می دهد:

```
Response.Write(Path.GetDirectoryName("C:\h\a.txt"))
```

خروجی کد بالا C:\h\ است.

GetFullPath:آدرس کامل مسیر را به من می دهد.مثلا :

```
Response.Write(Path.GetFullPath("a.txt"))
```

خروجی کد زیر علاوه بر نام فایل آدرس محلی دایرکتوری است.مثلا در سیستم من آدرس محلی همان جایی است که برنامه از آن اجرا می شود پس خروجی کد بالا در سیستم من به صورت زیر خواهد بود:

```
C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\a.txt
```

مگر اینکه خودتان از آدرس محلی که وب سایتتان در آنجا ذخیره شده(جایی که وب سایت ذخیره می شود با جایی که از آن اجرا می شود متفاوت است که در اینجا وب سایت من در درایو C قرار دارد در حالیکه لینک محل اجرای آن را در بالا دیدید) آن را مقدار

دهی کنید یعنی یا از آدرس کامل استفاده کنید و یا اینکه اگر یک راست در درایو C قرار دارد تنها یک \ به ابتدایش اضافه کنید:

```
Response.Write(Path.GetFullPath("\a.txt"))
```

خروجی کد بالا C:\a.txt است.

پس در کل اگر یک نام ساده(چه فایل و چه مسیر) بدون \ در ابتدا به ورودی GetFullPath بدهید, مسیر خروجی از همانجاییست که برنامه دارد اجرا می شود.

GetFileName:تنها نام فایل و پسوندش را بر می گرداند.

GetFileNamewithOutExtension:تنها نام فایل را بدون پسوندش بر می گرداند.

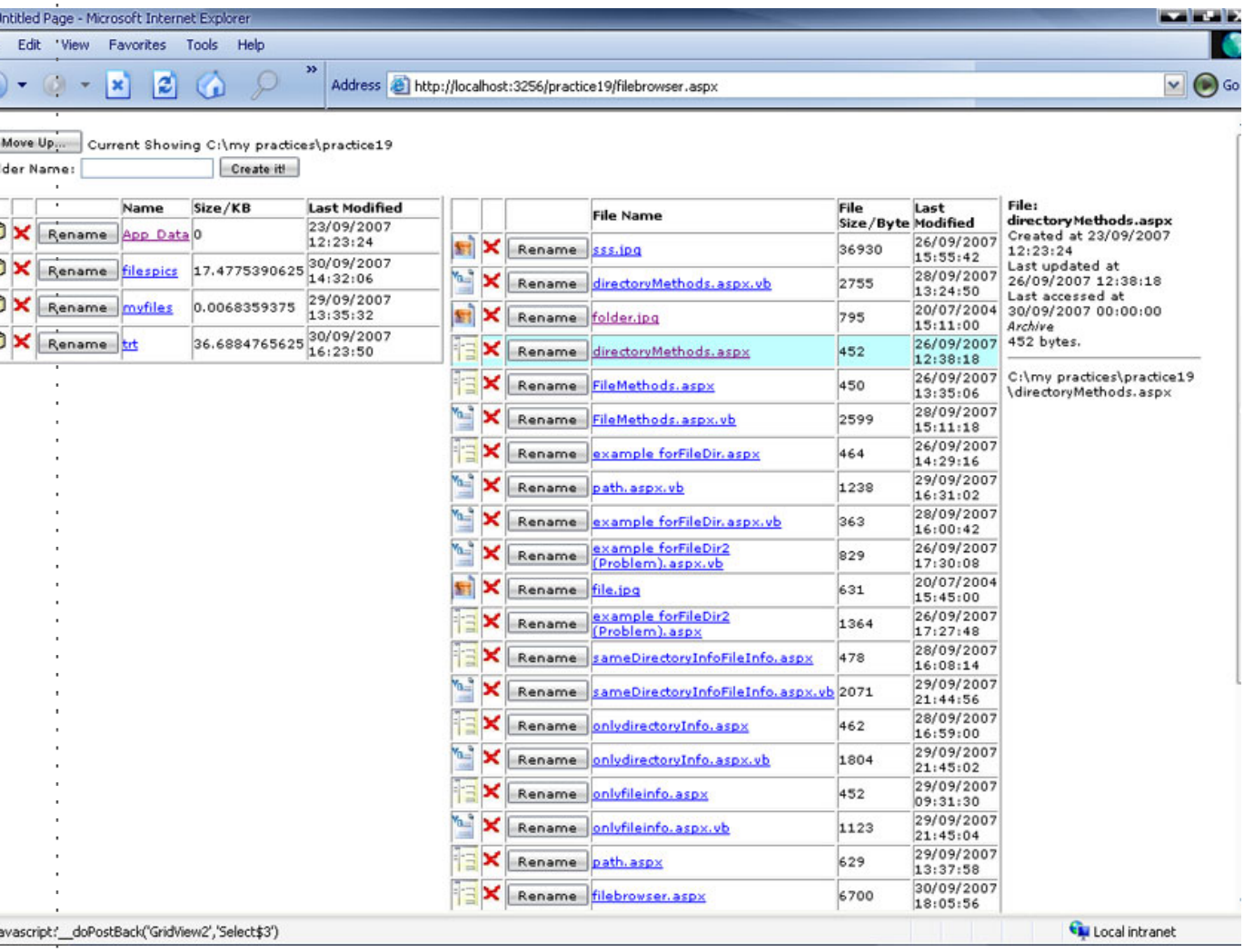
GetPathRoot:نام مسیر ریشه را بر می گرداند که اغلب نام درایو ها هستند.

HasExtension:اگر در آخر مسیر مورد نظر , پسوند وجود داشته باشد True بر می گرداند.

و...

حال می رسیم به اصل کار و آن هم استفاده ی کاربردی از هر آنچه تا اینجا از کار با فایل ها و فولدر ها آموختیم است یعنی طراحی یک File Browser.البته ما هنوز کارمان در بخش Files & Stram تمام نشده.

در این مثال جامع قصد نداریم تمام عملیات را پوشش دهیم ولی در حد معقول یک File Browser معمولی طراحی خواهیم کرد و به گونه ای این کار را انجام می دهیم که شما هم بتوانید آن را تا آنجا که امکان دارد گسترش دهید.قبل از شروع, نمونه ای از خروجی آن را در زیر می بینیم:



همانطور که می بینید برنامه ی ما به دو قسمت کلی فایل ها و دایرکتوری ها تقسیم شده.یک مسیر پیش فرض هم برای آن مشخص شده که به محل فیزیکی جایی که وب سایت ما در آن قرار دارد اشاره می کند به عبارت دیگر مسیر پیش فرض, همان پوشه ی وب سایت اصلی است.یک Gridview در سمت چپ برای نمایش دایرکتوری های داخل مسیر اصلی(که در اینجا پوشه ی وب سایت ماست) و یک Gridview ر سمت راست برای نمایش فایل های داخل مسیر اصلی در نظر گرفته شده.یک FormView هم در سمت راست پست فایل ها قرار دارد که با کلیک روی هر فایل مشخصات جزیی تر آن فایل را به ما نمایش می دهد.در بالای صفحه یک دکمه به نام Move Up قرار دارد که وظیفه اش Back کردن به والد مسیر جاریست در حقیقت کارش مثل دکمه ی Back در پنجره های Windows است.مسیر جاری هم هر لحظه در کنار دکمه ی Move Up ذکر می شود تا مشخص شود کجا هستیم.می توان با کلیک روی هر فولدر در Gridview سمت چپ

محتویات آن را چه فایل باشد و چه فولدر در سمت راست ببینیم.در حقیقت با کلیک روی هر فولدر, محتوای Gridview ها تغییر می کند و محتوای فولدر کلیک شده را نمایش می دهد.در کنار نام هر فایل و فولدری دو دکمه قرار دارد.یکی برای پاک کردن و دیگری برای تغییر نام آن در نظر گرفته شده.تصاویر مربوط به فولدر ها ثابت و تصاویر مربوط به فایل ها بر حسب نوع فایل متفاوت خواهد بود و در کنار نام هر فایل و فولدری سایزش نیز ذکر شده.بخش اعظم این مثال استخراج لیست فایل ها و فولدر ها در هر مسیر و نمایش آنها در GridView است.برای این کار یک تابع به نام ShowDirectoryContent ایجاد می کنیم که ورودیش مسیری باشد که می خواهیم محتویاتش را نمایش دهیم:

```
Private Sub ShowDirectoryContents(ByVal strPath As String)

End Sub
```

ما می خواهیم برای بازیابی فایل ها و دایرکتوری های یک مسیر از شی DirectoryInfo و متد های GetFiles و GetDirectories آن استفاده کنیم.پس در ابتدا درداخل تابع یک شی DirectoryInfo با توجه به مسیر دریافتی توسط فایل ایجاد می کنیم:

```
Dim dir As New DirectoryInfo(strPath)
```

سپس برای دیافت فایل های دایرکتوری مورد نظر از متد GetFiles استفاده می کنیم ولی چون مقدار برگشتی آن آرایه ای از نوع FileInfo است ابتدا آرایه ای از نوع FileInfo تعریف می کنیم و سپس مقادیر بازیابی شده از متد GetFiles را در آن ذخیره می کنیم:

```
Dim files() As FileInfo = dir.GetFiles
```

و به همین ترتیب برای بازیابی دایرکتوری ها نیز عمل می کنیم:

```
Dim dires() As DirectoryInfo = dir.GetDirectories
```

همانطور که می دانید کنترل های Rich Data از جمله GridView قادرند منابع داده ای نوع ICollection را نیز به عنوان منبع داده ای خود قرار دهند پس دو کنترل GridView تعریف می کنیم تا بتوانیم دو آرایه ی بالا را به عنوان منبع داده ای GridView ها قرار دهیم.ولی فعلا کار تابع را ادامه می دهیم:

```
  GridView1.DataSource = dires
GridView2.DataSource = files
```

پس از مشخص کردن منابع داده ای, برای صرفه جویی کل صفحه را Bind می کنیم:

```
Page.DataBind()
```

سپس مسیر جاری را در یک Label قرار می دهیم:

```
Label1.Text = "Current Showing " & strPath
```

از آنجا که قرار است هر گاه روی نام یک فایل در GridView2 کلیک شود جزییات آن در یک FormView نمایش داه شود و دور از حرفه ای هاست که با بارگذاری صفحه, اولین آیتم GridView2 به طور پیش فرض انتخاب شده باشد من عنصر انتخابی آن را با دادن اندیس 1- از بین می برم:

```
GridView2.SelectedIndex = -1
```

به این ترتیب با بارگذاری صفحه جزییات اولین فایل موجود در لیستGridView2 نمایش داده نمی شود تا وقتی خودتان رویش کلیک کنید.

در انتهای کار هم مسیر جاری را در یک ViewState ذخیره می کنیم چون در ادامه به آن نیاز شدید داریم:

```
ViewState.Add("CurrentPath", strPath)
```

پس فرم کلی تابع ShowDirectoryContent به صورت زیر شد:

```
    Private Sub ShowDirectoryContents(ByVal strPath As String)
        Dim dir As New DirectoryInfo(strPath)
        Dim files() As FileInfo = dir.GetFiles
        Dim dires() As DirectoryInfo = dir.GetDirectories
        Label1.Text = "Current Showing " & strPath
        GridView1.DataSource = dires
        GridView2.DataSource = files
        Page.DataBind()
        GridView2.SelectedIndex = -1
        ViewState.Add("CurrentPath", strPath)
    End Sub
```

در قدم بعدی تکلیف دکمه ی MoveUp را که قرار است مسیر را به والد تگ جاری تغییر دهد و محتویات آن را نمایش دهد روشن می کنیم.به همین خاطر هم ابتدا به رویداد کلیک آن می رویم:

```
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click

    End Sub
```

پس نام دکمه ی MoveUp Button1 خواهد بود.همانطور که می دانید اگر به انتهای هر مسیر محلی کامپیوتر, یک ..\ اضافه کنیم مسیر به والد مسیر جاری اشاره می کند.در اینجا هم اگر به انتهای مسیر جاری که در یک ViewState ذخیره کردیم یک ..\ اضافه کنیم می توانیم مسیر را به والدش تغییر دهیم:

```
Dim NewPath As String = ViewState("CurrentPath") & "\.."
```

سپس برای نمایش محتویات مسیر جدید, تابع ShowDirectoryContent را با مسیر جدید صدا می زنیم:

```
ShowDirectoryContents(NewPath)
```

این کد کار می کند ولی شما به عنوان یک حرفه ای نباید به کار کردن بسنده کنید.این کار مشکلاتی را به همراه دارد مثلا می دانید مسیر جدید به چه خواهد بود؟اگر مسیر جاری مثلا C:\my practice\practice19 باشد و به آن رشته ی ..\ را اضافه کنیم حاصل C:\my practice\practice19\.. خواهد بود که به معنی C:\my practice است و اگر ..\..\ اضافه کنیم حاصل C:\my practice\practice19\..\..\ می شود که به معنی C:\ است.و همین نوع مسیرها به عنوان ورودی به تابع ShowDirectoryContent ارسال می شود.و در داخل آن تابع هم همین مسیر در Label نمایش داده می شود.خوب شما می فهمید C:\my practice\practice19\.. یعنی چه ولی آیا همه ی کاربران می دانند C:\my practice\practice19\.. یعنی C:\my practice؟.مشخص است نه پس چه بهتر که به جای مسیر درست ولی نامفهوم C:\my practice\practice19\.. که چه بسا ممکن است ما را دچار مشکل هم بکند از مسیر ساده ی C:\my practice استفاده کنیم.برای این کار نباید یک راست محتوای ViewState را به ..\ پیوند بزنیم.راه درست استفاده از کلاس Path است.در ابتدا با متد Combine کلاس Path عمل پیوند را انجام می دهیم :

```
  Dim spath As String = ViewState("CurrentPath")
spath = Path.Combine(spath, "..")
```

همانطور که می دانید متد combine خودش \ های لازم را بین دو مسیر در حال پیوند اضافه می کند پس ما دیگر نیازی نداریم ..\ به کار ببریم و .. خالی هم کفایت می کند.حالا ما C:\my practice\practice19\.. را در اختیار داریم.برای تبدیل آن به مسیر درست از متد GetFullPath استفاده می کنیم برای مثال:

Path.GetFullPath("C:\a\b\c\..")= C:\a\b

Path.GetFullPath("C:\a\b\c\..\..")= C:\a

پس کافیست آدرس تولید شده را توسط این متد اصلاح کنیم و آن را درون همان متغیر ذخیره کنیم:

```
spath = Path.GetFullPath(spath)
```

حالا ما مسیر درست را داریم و کافیست آن را به تابع مربوطه Pass کنیم:

```
ShowDirectoryContents(spath)
```

پس در کل کد مربوط به رویداد کلیک دکمه ی Move UP با نام Button1 به صورت زیر شد:

```
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim spath As String = ViewState("CurrentPath")
        spath = Path.Combine(spath, "..")
        spath = Path.GetFullPath(spath)
        ShowDirectoryContents(spath)
    End Sub
```

در ادامه چند تابع را با هم می بینیم که کاربردشان را هنگام تعریف GridView ها و FormView تشریح می کنیم:

قبل از دیدن توابع, رویداد Page_Load را به صورت زیر تعریف می کنیم:

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If (Not Page.IsPostBack) Then
            ShowDirectoryContents(Server.MapPath("."))
        End If
    End Sub
```

برای بالا رفتن کارایی سایت, هر بار که صفحه بارگذاری شد من متد ShowDirectoryContents را صدا زدم واین کار را با تگ جاری که وب سایت در آن قرار دارد انجام دادم.

اولین تابع تابعی با نام GetFileVersion است که مشخصات یک فایل که مسیرش را به عنوان ورودی دریافت می کند از نوع رشته ای برمی گرداند:

```
    Protected Function GetVersionInfoString(ByVal path As Object) As String
        Dim info As FileVersionInfo =
FileVersionInfo.GetVersionInfo(CStr(path))
        Return info.FileName & " " & info.FileVersion & "<br>" &
info.ProductName & " " & info.ProductVersion
    End Function
```

همانور که می بینید از متد GetVersionInfo استفاده کردیم و مشخصات یک فایل مثلا FileName و یا FileVersion آن را به صورت یک رشته برگرداندیم.این تابع قرار است مشخصات فایلی را که نامش از لیست فایل ها در GridView2 کلیک شده را در FormView نمایش دهد.

تابع بعدی سایز دایرکتوری ها را برمی گرداند که قبل در مورد آن توضیح داده شد:

```
    Public Function GetDirSize(ByVal dirInfo As DirectoryInfo, ByVal
HaveSubDir As Boolean) As Decimal
        Dim TotalSize As Long = 0
```

```
        For Each f As FileInfo In dirInfo.GetFiles
            TotalSize += f.Length
        Next
        If HaveSubDir Then
            For Each dir As DirectoryInfo In dirInfo.GetDirectories
                TotalSize += GetDirSize(dir, True)
            Next
        End If
        Return TotalSize / 1024
    End Function
```

همانطور که می بینید مقدار برحسب بایت را تقسیم بر ۱۰۲٤ کردم تا مقدار برگشتی بر حسب KB باشد.

این متد قرار است ستون پنجم GridView1 را که سایز هر فولدر است پر کند.

تابع بعدی تابعی است که آدرس تصویری که باید بسته به نوع هر فایل در GridView2 قرار گیرد را برمی گرداند.این متد به جای نام فایل, شی FileInfo را به عنوان ورودی دریافت می کند و سپس با متد Extension پسوند آن را دریافت می کند و سپس با یک Select Case برای پسوند های مختلف آدرس تصویر مرتبط آن را که در پوشه ی filespics قرار دارد برمی گرداند.من تنها برای چند نوع فایل تصویر ایجاد کرم ولی شما سعی کنید بسته به کاربرد سایتتان برای انواع مختلف فایل تصویر ایجاد کنید.مثلا اگر سایت شما host معمولی است کافیست تنها برای Jpg--html و... تصویر ایجاد کنید.برای فایل هایی هم که نمی شناسم(با Case Else مشخص شده) یک تصویر به نام UnKnown ایجاد کردم:

```
    Public Function GetFilePic(ByVal file As FileInfo) As String
        Dim ex As String = file.Extension
        Select Case ex
            Case ".txt"
                Return "filespics\txt.jpg"
            Case ".jpg"
                Return "filespics\jpg.jpg"
            Case ".dll"
                Return "filespics\dll.jpg"
            Case ".gif"
                Return "filespics\gif.jpg"
            Case ".aspx"
                Return "filespics\aspx.jpg"
            Case ".vb"
                Return "filespics\vb.jpg"
            Case ".config"
                Return "filespics\config.jpg"
```

```
            Case Else
                Return "filespics\unknown.jpg"
        End Select
    End Function
```

تابع بعدی به نام CreateDirectory قرار است یک فولدر در مسیر جاری ایجاد کند که نامش را از یک TextBox می گیرد:

```
    Public Sub CreateDirectory(ByVal name As String)

    End Sub
```

ولی ممکن است افراد شیطان به جای یک نام هر چیزی وارد کنند مثلا به جای یک نام معمولی مثلا عبارت a\d\f\g\h را وارد کنند که باعث ایجاد چندین فولدر تودرتو می شود و یا h\..\ که در مسیر والد مسیر جاری فولدری به نام h می سازد پس من با متد GetFileNameWithoutExtension نام دایرکتوری را بازیابی می کنم تا چلوی چنین اعمالی گرفته شود چون همانطور که می دانید:

Path.GetFileNameWithoutExtension("C:\a\d\g")=g

Path.GetFileNameWithoutExtension("C:\a\d\g.txt")=g

Path.GetFileNameWithoutExtension("g")=g

Path.GetFileNameWithoutExtension("g.txt")=g

پس فرقی ندارد فایل باشد یا دایرکتوری.بلکه هر چه باشد یک نام به من برمی گرداند که ٤ مثال بالا را رعایت می کند پس ابتدا نام دایرکتوری را در یک متغیر ذخیره می کنم:

```
Dim newname As String = Path.GetFileNameWithoutExtension(name)
```

سپس برای ایجاد فایل از متد Create شی DirectoryInfo استفاده می کنم ولی چون می خواهم در مسیر جاری فایل را ایجاد کنم مسیر جاری را با نامی که در بالا ایجاد کردم پیوند می دهم:

```
Dim newPath As String = Path.Combine(ViewState("CurrentPath"), newname)
```

حال شی DirectoryInfo را با مسیری که در بالا ایجاد کردم و در متغیر NewPath قرار دادم New می کنم:

```
Dim Dinfo As New DirectoryInfo(newPath)
```

و در نهایت دایرکتوری را می سازم:

```
Dinfo.Create()
```

پس تابع کلی CreateDirectory به صورت زیر شد:

```
  Public Sub CreateDirectory(ByVal name As String)
```

```
    Dim newname As String = Path.GetFileNameWithoutExtension(name)
    Dim newPath As String = Path.Combine(ViewState("CurrentPath"), newname)
    Dim Dinfo As New DirectoryInfo(newPath)
    Dinfo.Create()
End Sub
```

پس با این حساب رویداد کلیک دکمه ای را که با فشردن آن قرار است فولدر مورد نظر ایجاد شود را به صورت زیر می نویسم و تابع بالا را با ورودی TextBox در آن صدا می زنم:

```
    Protected Sub Button2_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button2.Click
        CreateDirectory(TextBox1.Text)
        ShowDirectoryContents(ViewState("CurrentPath"))
    End Sub
```

همانطور که می بینید برای دیدن فولدری که ساخته شده, مجددا متد ShowDirectoryContents را با همان مسیر جاری که قبلا هم در آن بودیم صدا می زنیم.یادتان باشد این کار را اگر نکنیم با کلیک دکمه ی Button2 برای ساخته شدن فولدر با نامی که در TextBox1 مشخص شده, صفحه PostBack می شود و همانطور که در رویداد Page_Load دیدید من برای بالا رفتن کارایی سایت تابع را تنها وقتی صفحه برای اولین بار بارگذاری می شود با تگ جاری صدا زدم نه با PostBack شدن صفحه.پس در اینجا هم چون صفحه مجددا بارگذاری نشده و PostBack می شود برای نمایش فولدر ساخته شده من مجددا متد ShowDirectoryContents را صدا زدم.

یک تابع برای پاک کردن فولدر ایجاد می کنم.البته با فرض اینکه فولدر ها خالی نیستند آرگومان متد Delete را True دادم چون در این صورت اگر فولدر خالی هم باشد باز عمل Delete انجام می گیرد ولی اگر این کار را نمی کردیم تنها فولدر های خالی قابل حذف بودند که این خود یک محدودیت در سایت ما به حساب می آمد:

```
    Public Sub DeleteFolder(ByVal name As String)
        Dim Dinfo As New DirectoryInfo(name)
        Dinfo.Delete(True)
    End Sub
```

یک تابع برای حذف فایل ها می نویسیم:

```
    Public Sub DeleteFile(ByVal name As String)
        Dim finfo As New FileInfo(name)
        finfo.Delete()
    End Sub
```

دو تابع هم برای تغییر نام فولدر ها و فایل ها می نویسیم:

```
        Public Sub RenameFolder(ByVal oldPath As String, ByVal NewPath As
String)
            Directory.Move(oldPath, NewPath)
        End Sub
        Public Sub RenameFile(ByVal oldPath As String, ByVal NewPath As String)
            File.Move(oldPath, NewPath)
    End Sub
```

من در توابع بالا جوانب احتیاط را رعایت نکردم(از کلاس Path استفاده نکردم) زیرا برای تنوع این کار را در رویداد هایی که توابع بالا را صدا می زنند انجام خواهم داد البته لازم به ذکر است وقتی روی یک دکمه کلیک می شود که فایل یا فولدری پاک شود, به نظر نمی رسد کاربر بتواند شیطنتی از خود بروز دهد.

بالاخره نوبت به ایجاد GridView ها می رسد.تعریف اولیه ی GridView1 را در زیر می بینید:

```
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="false"
DataKeyNames="FullName" EmptyDataText="No Directory In This Directory">

    </asp:GridView>
```

اول اینکه صفت AutoGenerateColumns=False است و این یعنی خودمان باید زحمت ایجاد ستون ها را بکشیم.دوما صفت DataKeyNames را برابر FullName قرار دادیم.در این مورد در بحث DataBinding هم صحبت کردیم و گفتیم که اگر منبع داده ای یک Rich Data Control مثل GridView می تواند از جنس ICollection باشد مثل Array.و گفتیم اگر این طور باشد ما برای Bind کردن هر سطر از این GridView به جای سطر های جداول DataBase با یک Object سرو کار داریم که می بایست ویژگی های آن Object را در سطر های GridView نمایش دهیم.یادتان هست که وقتی منبع داده ای ما یک جدول پایگاه داده بود صفت DataKeyNames نام ستون اصلی آن بود.در اینجا هم نام ویژگی اصلی شی است که از آرایه ی مورد نظر بازیابی شده.پس در کل تناظر های زیر را مد نظر داشته باشید:

DataBase Table ~ Array

Each Table Row ~ Each Array Item

Each Table Columns ~ Each Property Of Array Object

با ایجاد تگ Columns ایجاد ستون ها را Start می زنیم:

```
        <Columns>
    ...
```

```
</Columns>
```

اولین ستون یک تصویر خواهد بود که شکل فولدر ها را نمایش دهد:

```
<asp:TemplateField>
<ItemTemplate>
<img src="folder.jpg" />
</ItemTemplate>
</asp:TemplateField>
```

سپس یک دکمه(ImageButton) برای حذف فولدر ها ایجاد می کنیم:

```
<asp:TemplateField>
    <ItemTemplate>
    <asp:ImageButton runat="server" ImageUrl="~/del.JPG"
CommandName="delete" CommandArgument=<%# Container.DataItem %> />
    </ItemTemplate>
</asp:TemplateField>
```

ما در این مثال دو مدل تشخیص دکمه داریم که همانطور که می دانید در یک GridView وقتی یک دکمه فشرده می شود این تشخیص بر عهده ی رویداد RowCommand خواهد بود.اولین مدل مدل عرضی است.یعنی ممکن است چندین دکمه در کنار هم در هر سطر GridView وجود داشته باشند مثلا در اینجا دو دکمه ی Rename و Delete خواهیم داشت.پس برای تشخیص هر یک در رویداد RowCommand برای انجام عملیات مربوطه باید یک فاکتور داشته باشیم.من این فاکتور را بر اساس رشته ای که در صفت CommandName مشخص کردم تعیین کردم.یعنی گفتم اگر روی دکمه ی Delete کلیک شد, مقدار صفت CommandName برابر Delete خواهد بود و متعاقبا اگر روی دکمه ی Rename کلیک شد مقدار صفت CommandName برابر Rename خواهد بود.حال می رسیم به تشخیص طولی .تشخیص طولی یعنی مشخص شود دکمه ی کدام سطر کلیک شده.چون ممکن است صد ها فایل یا فولدر داشته باشیم پس حتما صد ها دکمه هم داریم و وقتی هر یک کلیک شد باید مشخص کنیم مال کدام سطر بوده.برای تشخیص طولی از CommandArgument استفاده می کنیم.ولی مشکل اینجاست که مقادیر در اینجا ثابت نیستند که بگوییم دکمه ی a کلیک شده یا b.بلکه معلوم نیست چند دکمه خواهیم داشت و مشکلات دیگر.در ادامه ی توضیحات بالا درمودر تناظر ICollection با Table به این نکته اشاره می کنیم که همانطور که در بحث DataBinding مطرح شد, متد Container.DataItem برای یک جدول از یک پایگاه داده در هر بار Bind شدن هر سطر از GridView یک سطر را بر می گرداند و اگر آرگومان ورودی اهم از اندیس ستون و یا نام ستونی داشته باشد از آن

سطر مقدار آن ستون را بر می گرداند.در اینجا هم چون داده از نوع ICollection به جای Table است این متد در هر بار Bind شدن هر سطر این GridView یکی از اشیایی که در داخل آرایه قرار دارد را برمی گرداند.حال همانطور که در تابع ShowDirectoryContents مشخص کردیم منبع داده ای این GridView یک آرایه از DirectoryInfo هاست.پس متد Container.DataItem در هر بار Bind شدن هر سطر این GridView یک شی از جنس DirectoryInfo را بر می گرداند.پس اگر ما عبارت Container.DataItem را به عنوان مقدار صفت CommandArgument دکمه ای که کلیک کردیم قرار دهیم Container.DataItem حاوی شیی اخواهد بود که نوبت Bind شدنش است یعنی اینکه نام فولدری که دکمه ی Delete کنارش را کلیک کردیم به رویداد RowCommand ارسال می کند و مشکل تشخیص طولی هم حل می شود.

پس به همین ترتیب دکمه ی Rename را ایجاد می کنیم:

```
  <asp:TemplateField>
      <ItemTemplate>
      <asp:Button       ID="Button1"       runat="server"       Text="Rename"
CommandName="rename" CommandArgument=<%# Container.DataItem %> />
      </ItemTemplate>
</asp:TemplateField>
```

البته این نکته را بگویم که مقداری که Container.DataItem برمی گرداند شی DirectoryInfo است ولی چون این مقدار به صفت CommandArgument داده شده تبدیل به واحد String می شود و سپس ارسال می شود بنابراین هنگام بازیابی ما تنها نام فولدر را در اختیار داریم ولی چون اسامی فولدر ها Unique است به مشکل بر نمی خوریم.

سپس یک ستون برای نمایش نام فولدر ایجاد می کنیم:

```
<asp:ButtonField DataTextField="Name" CommandName="Select"
HeaderText="Name" />
```

باز هم همانطور که در بحث DataBinding اشاره کرده بودیم چون منبع داده ای یک آرایه از یک سری اشیا است پس مقداری که صفت DataTextField می گیرد یک صفت از آن اشیا است و چون در اینجا آن اشیا از جنس DirectoryInfo هستند پس DataTetField=”Name” مقدار DirectoryIno.Name یعنی نام فولدر را نمایش می

دهد. یادتان هست که اگر منبع داده ای یک جدول می بود آنگاه صفت DataTextField برای نمایش باید نام یکی از ستون های آن جدول می بود.

سپس نوبت به نمایش سایز فولدر است.سایز آن را با استفاده از متد GetDirSize که قبلا نوشته بودیم درداخل تگ DataBinding انجام می دهیم:

```
        <asp:TemplateField HeaderText="Size/KB">
        <ItemTemplate>
        <%#GetDirSize(Container.DataItem, True)%>
        </ItemTemplate>
    </asp:TemplateField>
```

به این ترتیب مقدار برگشتی متد GetDirSize که بر حسب کیلوبایت هم هست در اینجا نمایش داده می شود.

در انتها هم یک ستون برای نمایش تاریخ آخرین دسترسی ایجاد می کنیم:

```
<asp:BoundField DataField="LastWriteTime" headertext="Last Modified" />
```

پس در کل GridView1 به صورت زیر تعریف شد:

```
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="false"
DataKeyNames="FullName" CellPadding="0" CellSpacing="1" Font-Names="Verdana"
Font-Size="X-Small" Height="133px" Width="363px" EmptyDataText="No Directory
In This Directory">
        <Columns>
        <asp:TemplateField>
        <ItemTemplate>
        <img src="folder.jpg" />
        </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField>
        <ItemTemplate>
        <asp:ImageButton runat="server" ImageUrl="~/del.JPG"
CommandName="delete" CommandArgument=<%# Container.DataItem %> />
        </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField>
        <ItemTemplate>
        <asp:Button ID="Button1" runat="server" Text="Rename"
CommandName="rename" CommandArgument=<%# Container.DataItem %> Font-
Names="Verdana" Font-Size="X-Small" Height="18px" Width="65px" />
        </ItemTemplate>
        </asp:TemplateField>
        <asp:ButtonField DataTextField="Name" CommandName="Select"
HeaderText="Name" />
        <asp:TemplateField HeaderText="Size/KB">
        <ItemTemplate>
        <%#GetDirSize(Container.DataItem, True)%>
```

```
        </ItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="LastWriteTime" headertext="Last Modified" />
        </Columns>
    </asp:GridView>
```

حال برای تشخیص دکمه ی کلیک شده به رویداد RowCommand از GridView1 می رویمو در ابتدا چک می کنیم دکمه ای که کلیک شده کدام است Rename Or Delete؟.همانطور که گفتیم این تشخیص عرضی بوده و ما آن ا بر عهده ی CommandName گذاشتیم که جزو صفات مقدار ارسالی e و در رویداد RowCommand از نوع CommandEventArg است :

```
    Protected Sub GridView1_RowCommand(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewCommandEventArgs) Handles
GridView1.RowCommand
        If e.CommandName = "delete" Then
            ...
        ElseIf e.CommandName = "rename" Then
            ...
        End If
    End Sub
```

اگر دکمه ی فشرده شده Delete باشد باید تابع Deletefolder را با آرگومان ورودی که مسیر فولدر جهت Delete است صدا بزنم.برای این کار ابتدا نام فولدر را که دکمه ی Delete کنارش در لیست فولدر ها کلیک شده را بازیابی می کنم و جهت اطمینان بیشترو به نوعی احتیاط از کلاس Path برای این کار استفاده می کنم:

```
Dim foldername As String =
Path.GetFileNameWithoutExtension(e.CommandArgument)
```

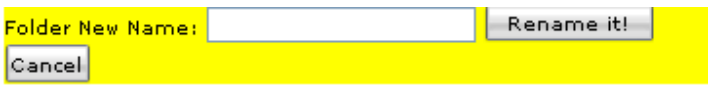
حال که نامش را دارم کافیست مسیرش را با اتصال مسیر جاری به نام فولدر ایجاد کنم:

```
Dim newPath As String = Path.Combine(ViewState("CurrentPath"), foldername)
```

و در نهایت هم تابع Deletefolder را صدا بزنم:

```
DeleteFolder(newPath)
```

حال اگر قصد Rename داشته باشیم باید چه بکنیم.ببینید هر بار که شما روی یک دکمه ی Rename در کنار یک فولدر کلیک می کنید یک Panel به صورت زیر نمایش داده خواهد شد:



که یک Panel با محتوای دو دکمه یک Label و یک TextBox که در ابتدا Panel1.Visible=false بوده و با کلیک کردن دکمه ی Rename ظاهر خواهد شد و تعریفش به صورت زیر خواهد بود:

```
  <asp:Panel  ID="Panel1"  runat="server"  BackColor="Yellow"  Height="22px"
Visible="False" Width="362px">
  <asp:Label  ID="Label3"  runat="server"  Font-Names="Verdana"  Font-Size="X-
Small" Text="Folder New Name:"></asp:Label>
  <asp:TextBox  ID="TextBox2"  runat="server"  Font-Names="Verdana"  Font-
Size="X-Small" Width="132px"></asp:TextBox>
  <asp:Button  ID="Button3"  runat="server"  Text="Rename  it!"  Font-
Names="Verdana" Font-Size="X-Small" Height="19px" Width="87px" />
  <asp:Button ID="Button5" runat="server" Font-Names="verdana" Font-Size="X-
Small" Height="20px" Text="Cancel" Width="44px" />
</asp:Panel>
```

شما باید نام جدید فولدری که دکمه ی Rename کنارش را کلیک کردید در این textBox وارد کنید و جهت تغییر نام هم دکمه ی Rename it! را کلیک کنید و اگر پشیمان شدید هم دکمه ی Cancel را کلیک کنید.پس در رویداد RowCommand اگر روی دکمه ی Rename کلیک کردیم ابتدا باید Panel بیان شده در بالا ظاهر شود:

```
Panel1.Visible = True
```

سپس همه چیز باید به دکمه ی Rename it! یا همان Button3 سپرده شود.ولی در رویداد کلیک این دکمه برای تغییر نام حتما باید نام قبلی را داشته باشیم(به متد Move رجوع شود).پس بهتر است قبل از اینکه همه چیز را به دکمه ی Rename it! بسپاریم نام فولدری که قصد تغییر نامش را داریم نیز به آن ضمیمه کنیم.من این کار را با صفت CommandArgument آن انجام دادم:

```
Button3.CommandArgument = e.CommandArgument
```

e.CommandArgument حاوی نام فولدری است که دکمه ی Rename کنارش در لیست فولدر ها کلیک شده.پس کد کلی رویداد RowCommand به صورت زیر شد:

```
    Protected Sub GridView1_RowCommand(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewCommandEventArgs)                Handles
GridView1.RowCommand
        If e.CommandName = "delete" Then
            Dim          foldername          As          String          =
Path.GetFileNameWithoutExtension(e.CommandArgument)
            Dim newPath As String = Path.Combine(ViewState("CurrentPath"),
foldername)
            DeleteFolder(newPath)
        ElseIf e.CommandName = "rename" Then
```

```
            Panel1.Visible = True
            Button3.CommandArgument = e.CommandArgument
        End If
    End Sub
```

حال نوبت اجرای عمل Rename است.برای این کار به رویداد کلیک Button3 می رویم:

```
    Protected Sub Button3_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button3.Click

    End Sub
```

چون می خواهیم چند بار از مسیر جاری استفاده کنیم ابتدا آن را از ViewState بازیابی کرده و در یک متغیر به نام CurrPath قرار می دهیم:

```
Dim CurrPath As String = ViewState("CurrentPath")
```

سپس نام جدیدی را که کاربر برای فولدر در TexTBox واردکرده را بازیابی می کنم:

```
Dim newName As String = Path.GetFileNameWithoutExtension(TextBox2.Text)
```

حالا مسیر جدید را با پیوند مسیر جاری و نام جدید مشخص می کنم:

```
Dim newPath1 As String = Path.Combine(CurrPath, newName)
```

و پس از آن مسیر قدیم را با بازیابی نامش از صفت CommandArgument دکمه ی Button3 و پیوند با مسیر جاری مشخص می کنم:

```
Dim oldPath1 As String = Path.Combine(CurrPath, Button3.CommandArgument)
```

و تابع RenameFolder را با دو مسیری که بدست آوردم صدا می زنم:

```
RenameFolder(oldPath1, newPath1)
```

برای دیدن تغییر نام تابع ShowDirectoryContens را با همان مسیر جاری صدا می زنم:

```
ShowDirectoryContents(CurrPath)
```

و در نهایت Panelی که حاوی TextBox برای تغییر نام بود را Invisible می کنم:

```
Panel1.Visible = False
```

پس رویداد کلیک Button3 به صورت زیر شد:

```
    Protected Sub Button3_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button3.Click
        Dim CurrPath As String = ViewState("CurrentPath")
        Dim newName As String =
Path.GetFileNameWithoutExtension(TextBox2.Text)
        Dim oldPath1 As String = Path.Combine(CurrPath,
Button3.CommandArgument)
        Dim newPath1 As String = Path.Combine(CurrPath, newName)
        RenameFolder(oldPath1, newPath1)
        ShowDirectoryContents(CurrPath)
```

```
        Panel1.Visible = False
    End Sub
```

یادتان هست که در آن Panel یک دکمه به نام Cancel هم وجود داشت.تنها کاری که این دکمه انجام خواهد داد مخفی کردن Panel است:

```
    Protected Sub Button5_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button5.Click
        Panel1.Visible = False
    End Sub
```

حالا می رویم به سراغ تعریف GridView مخصوص فایل ها یا GridView2.کد کلی آن به صورت زیر است:

```
    <asp:GridView ID="GridView2" runat="server" AutoGenerateColumns="false"
EmptyDataText="No File In This Directory">
    <SelectedRowStyle BackColor="#C0FFFF" />
    <Columns>
      <asp:TemplateField>
      <ItemTemplate>
        <img src='<%# GetFilePic(Container.DataItem) %>' />
      </ItemTemplate>
      </asp:TemplateField>
          <asp:TemplateField>
    <ItemTemplate>
    <asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="~/del.JPG"
CommandName="delete" CommandArgument='<%# container.dataitem %>' />
    </ItemTemplate>
    </asp:TemplateField>
     <asp:TemplateField>
    <ItemTemplate>
    <asp:Button ID="Button7" runat="server" Text="Rename"
CommandName="rename" CommandArgument='<%# container.dataitem %>' Font-
Names="Verdana" Font-Size="X-Small" Height="18px" Width="65px" />
    </ItemTemplate>
    </asp:TemplateField>
      <asp:ButtonField DataTextField="Name" CommandName="Select"
HeaderText="File Name" />
      <asp:BoundField DataField="Length" HeaderText="File Size/Byte" />
      <asp:BoundField DataField="LastWriteTime" HeaderText="Last Modified"
/>
    </Columns>
  </asp:GridView>
```

تگ SelectedRowStyle رنگ پس زمینه ی سطری از GridView که فایلی را که از لیست انتخاب شده از بقیه ی سطر ها متمایز می کند.در ستون اول از تابع GetFilePic استفاده کردیم تا مسیر تصویر مورد نظر را به صفت src تگ img بدهیم.همانطور که قبلا

بار ها ذکر شد در اینجا Container.DataItem خود شی FileInfo که دارد در داخل سطر Bind GridView می شود را بر می گرداند و می دانید که آرگومان ورودی تابع GetFilePic نیز از جنس Fileinfo است و چون ما از Container.DataItem در داخل یک تابع در تگ <%#..> استفاده کردیم بنابراین مثل مقدار صفت CommandArgument نیست که آن را تبدیل به واحد String کند بلکه در اینجا Container.DataItem خود شی FileInfo را بر می گرداند و از مقدار آن به عنوان آرگومان ورودی تابع GetFilePic استفاده کرد.در ستون بعدی هم دکمه ی Delete را قرار دادیم.و پس از آن دکمه ی Rename و سپس ستون هایی برای تشخیص Name و Lenght و LastWriteTime ایجاد کردیم که همانطور که در GridView1 هم بیان ش صفات DataTextFiled و DataFiled هنگامی که منبع داده ای از جنس ICollection است و مرجع ما یک شی باشد مقدار صفت آن شی را نمایش می دهند یعنی مثلا FileInfo.Neme ویا FileInfo.Lenght و...

رویداد RowCommand هم برای GridView2 به صورت زیر است:

```
    Protected Sub GridView2_RowCommand(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewCommandEventArgs)                    Handles
GridView2.RowCommand
        If e.CommandName = "delete" Then
            Dim filename As String = Path.GetFileName(e.CommandArgument)
            Dim newPath As String = Path.Combine(ViewState("CurrentPath"),
filename)
            DeleteFile(newPath)
        ElseIf e.CommandName = "rename" Then
            Panel2.Visible = True
            Button4.CommandArgument = e.CommandArgument
        End If
    End Sub
```

Panel2 هم مثل Panel1 است ولی در بالای GridView2 ظاهر می شود و دکمه ای که عمل تغییر نام را انجام می دهد نیز Button4 نام دارد.رویداد کلیک این دکمه مثل Button3 خواهد بود ولی با یک تفاوت کوچک و آن هم این است که شما نام فایل را می خواهید تغییر دهید.پس کاربر باید تنها یک نام بدون پسوند را در TextBox مربوطه وارد کند و اگر آن را با پسوند وارد کرد ما باید نام را از آن جدا کنیم.چون ممکن است هنگام تغییر نام یک پسوند جدید وارد کند مثلا نام a.txt را به a.pdf تغییر دهد و این یک اشکال در برنامه ی ما محسوب می شود و باید جلوی آن گرفته شود.من این کار را به این

صورت حل کردم که ابتدا با متد GetFileNameWithOutExtension نام اصلی فایلی که کاربر مقدارش را در TextBox وارد کرده بود را بازیابی می کنم.سپس از Button4.CommandArgument استفاده می کنم و در ابتدا پسوند فایل قبلی که جهت تغییر نام روی دکمه ی Rename کنارش کلیک شد را با متد GetExtension بازیابی می کنم و سپس آن را به نام جدید می چسبانم.به این ترتیب فایل جدید دقیقا همان پسوندی را خواهد داشت که فایل قبلی داشته:

```
  Dim CurrPath As String = ViewState("CurrentPath")
  Dim ex As String = Path.GetExtension(Button4.CommandArgument)
  Dim newName As String = Path.GetFileNameWithoutExtension(TextBox3.Text) &
ex
Dim oldPath1 As String = Path.Combine(CurrPath, Button4.CommandArgument)
```

پس کد کلی رویداد کلیک دکمه ی Button4 نیز به صورت زیر شد:

```
    Protected Sub Button4_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button4.Click
        Dim CurrPath As String = ViewState("CurrentPath")
        Dim ex As String = Path.GetExtension(Button4.CommandArgument)
        Dim newName As String =
Path.GetFileNameWithoutExtension(TextBox3.Text) & ex
        Dim oldPath1 As String = Path.Combine(CurrPath,
Button4.CommandArgument)
        Dim newPath1 As String = Path.Combine(CurrPath, newName)
        RenameFile(oldPath1, newPath1)
        ShowDirectoryContents(CurrPath)
        Panel2.Visible = False
    End Sub
```

دکمه ی Cancel نیز به فرم زیر است:

```
    Protected Sub Button6_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button6.Click
        Panel2.Visible = False
    End Sub
```

نکته ای که در مورد Delete هر دو کنترل GridView باقی می ماند این است که وقتی ما سطری را از یک GridView پاک می کنیم به طور خود کار رویداد RowDeleting آن تحریک می شود.از طرف دیگر اگر این عمل حذف توسط دکمه ی Delete پیش فرض خود GridView انجام گیرد(یعنی صفت ShowDeleteButton=True باشد) خود GridView رویداد RowDeleting را Hanlde می کند.ولی وقتی دکمه ی Delete را به طور سفارشی خودمان ایجاد کنیم و باعث حذف یک سطر از GridView شویم دیگر رویداد RowDeleting به طور خود

کار Hanlde نمی شود و شما با خطا مواجه می شوید.برای حل این مشکل کافی است رویداد RowDeleting را برای دو GridView تنها صدا بزنیم.از طرفی ما پس از عمل Delete یک سطر از هر یک از دو GirdView مجددا ShowDirectoryContents را صدا نزدیم تا تغییرات را ببینیم خوب چه بهتر که این کار را در رویداد RowDeleting انجام دهیم(البته این کار را می توانستید در همان رویداد RowCommand هم پس از صدا زدن تابع DeleteFile or DeleteFolder انجام دهید):

```
    Protected Sub GridView2_RowDeleting(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewDeleteEventArgs)                    Handles
GridView2.RowDeleting
        ShowDirectoryContents(ViewState("CurrentPath"))
    End Sub
    Protected Sub GridView1_RowDeleting(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewDeleteEventArgs)                    Handles
GridView1.RowDeleting
        ShowDirectoryContents(ViewState("CurrentPath"))
    End Sub
```

حال می رسیم به اصل کار و آن هم وقتی است که روی یک فولدر در GridView1 کلیک می کنیم تا محتویاتش نمایش داده شود.ما تابع آن را قبلا به نام ShowDirectoryContents نوشتیم.و حالا باید از آن استفاده کنیم.می دانیم که اولا باید روی نام فولدر کلیک کنیم تا محتویاتش نمایش داده شود و دوما همانطور که در تعریف GridView1 دیدید ما یک صفت CommandName به ستونی که در آن نام فولدر را نمایش داده بودیم تعریف کردیم و مقدارش هم Select گذاشتیم.حتما یادتان هست که اگر مقدار CommandName یک ستون در GridView را برابر Select قرار دهید به طور پیش فرض رویداد SelectedIndexChanged کنترل GridView تحریک می شود چون Select یک کلمه ی کلیدی برای CommandName است در حالیکه Rename و Delete کلمه ی کلیدی نبودند.پس حال که این رویداد تحریک می شود کافی است عمل نمایش عناصر داخل دایرکتوری که رویش کلیک کردیم را در رویداد SelectedIndexChanged کنترل GridView1 انجام دهیم:

```
    Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles GridView1.SelectedIndexChanged

    End Sub
```

برای اینکه نام دایرکتوری که رویش کلیک کرده ایم را به مسیر جاری بچسبانیم تا مسیر محتویات دایرکتوری که رویش کلیک کردیم تولید شود, ابتدا باید نام فولدری که رویش کلیک کردیم را بدست آوریم.ولی بهتر است یک کلک بزنیم و به جای نام آن نام کاملش را بدست آوریم که نیازی نباشد آن را به مسیر جاری بچسبانیم.پس حالا موقع استفاده از مقدار صفت DataKeyNames کنترل GridView1 است.حتما می دانید برای بازیابی مقدار صفت DataKeyNames کنترل GridView باید از متد DataKeys این کنترل استفاده کنیم و سپس اندیس سطری که مقدار صفت DataKeyNamesش را می خواهیم به عنوان آرگومان ورودی به آن بدهیم.از طرف دیگر هم چون در رویداد SelectedIndexChanged هستیم پس می توانیم از متد SelectedIndex کنترل GridView برای بدست آوردن اندیس سطری که کلیک شده استفاده کنیم.پس حتما باید حدس زده باشید که کد زیر چه مقداری را بر می گرداند:

```
GridView1.DataKeys(GridView1.SelectedIndex)
```

این کد به صفت FullName شی DirectoryInfoی که رویش کلیک شده اشاره دارد و اگر مقدارFullName را بخواهیم باید آن را به صورت زیر بنویسیم:

```
GridView1.DataKeys(GridView1.SelectedIndex).Value
```

به این صورت به مسیر کامل فولدری که رویش کلیک شده دسترسی داریم و کافی است تابع ShowDirectoryContents را با آن مسیر صدا بزنیم.پس کد کلی رویداد SelectedIndexChanged به صورت زیر خواهد بود:

```
    Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles GridView1.SelectedIndexChanged
        Dim dir As String =
CStr(GridView1.DataKeys(GridView1.SelectedIndex).Value)
        ShowDirectoryContents(dir)
    End Sub
```

در انتها هم نوبت به آن می رسد که جزییات فایلی که در لیست GridView2 رویش کلیک شده را در کنترل FormView نمایش دهیم.در اینجا هم به رویداد SelectedIndexChanged کنترل GridView2 نیاز داریم و در آن در ابتدا نام کامل آن فایل را بازیابی می کنیم:

```
Dim thefile As String =
CStr(GridView2.DataKeys(GridView2.SelectedIndex).Value)
```

سپس آن را تبدیل به یک شی FileInfo می کنیم تا بتوانیم با استفاده از ویژگی های آن به جزییات فایل دسترسی داشته باشیم:

```
Dim finfo As New FileInfo(thefile)
```

خوب ما نمی توانیم یک شی FileInfo را به عنوان منبع داده ای یک Form View قرار دهیم پس ناچاریم از یک کلکسیون برای این کار کمک بگیریم.من شی FileInfo را درون یک ArrayList قرار می دهم و سپس ArrayList را به عنوان منبع داده ای Form View انتخاب می کنم:

```
  Dim arrlist As New ArrayList
  arrlist.Add(finfo)
  FormView1.DataSource = arrlist
FormView1.DataBind()
```

پس کد کلی رویداد SelectedIndexChanged کنترل GridView2 به صورت زیر شد:

```
      Protected Sub GridView2_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles GridView2.SelectedIndexChanged
          Dim thefile As String =
CStr(GridView2.DataKeys(GridView2.SelectedIndex).Value)
          Dim finfo As New FileInfo(thefile)
          Dim arrlist As New ArrayList
          arrlist.Add(finfo)
          FormView1.DataSource = arrlist
          FormView1.DataBind()
    End Sub
```

من کد FormView را به صورت زیر می نویسم:

```
  <asp:FormView ID="FormView1" runat="server" Font-Names="Verdana" Font-
Size="X-Small" Height="151px" Width="155px">
  <ItemTemplate>
  <b>File:
  <%#DataBinder.Eval(Container.DataItem, "Name")%>
  </b>
  <br>
  Created at
  <%#DataBinder.Eval(Container.DataItem, "CreationTime")%>
  <br>
  Last updated at
  <%#DataBinder.Eval(Container.DataItem, "LastWriteTime")%>
  <br>
  Last accessed at
  <%#DataBinder.Eval(Container.DataItem, "LastAccessTime")%>
  <br>
  <i>
  <%#DataBinder.Eval(Container.DataItem, "Attributes")%>
  </i>
  <br>
```

```
    <%#DataBinder.Eval(Container.DataItem, "Length")%>
    bytes.
    <hr>
    <%#GetVersionInfoString(DataBinder.Eval(Container.DataItem, "FullName"))%>
    </ItemTemplate>
</asp:FormView>
```

همانطور که در بحث DataBinding اشاره کرده بودیم اگر از متد DataBinder.Eval به جای Eval خالی استفاده کنید می توانید مرجع را یک شی در نظر بگیرید(در اینجا Container.DataItem است که شی FileInfo است)و سپس با آرگومان بعدی نام ویژگی که می خواهید مقدارش را نمایش دهید ذکر کنید.حتما یادتان هست که اگر منبع داده ی FormView یک جدول بود آنگاه Container.DataItem یک سطر از آن جدول را مشخص می کرد و آرگومان بعدی هم نام ستونی از آن سطر را مشخص می کرد که قرار بود توسط DataBinder.Eval نمایش داده شود ولی در اینجا چون منبع داده ی FormView یک شی ArrayList است بنابراین در هر بار Bind شدن هر سطر FormView(که در اینجا تنها یک سطر خواهد بود زیرا تنها یک شی FileInfo در ArrayList وجود دارد) شی Container.DataItem به هر آیتم ArrayList اشاره می کند که یک شی FileInfo است بنابراین آرگومان بعدی ویژگی از شی FileInfo است که قرار است توسط DataBinder.Eval نمایش داده شود.پس اگر قرار بود مقدار یک ستون خاص از یک سطر یک جدول پایگاه داده را نمایش دهید می توانستید از هر یک از روشهای زیر استفاده کنید:

Eval(“Column Name”)

DataBinder.Eval(Container.DataItem ,“Column Name”)

برای درک بیشتر به بحث DataBinding رجوع کنید.در نهایت هم از متد GetVersionInfoString استفاده کردم تا مشخصات تخصصی تر فایل را اگر Dll Or Exe بود نمایش دهیم.

خوب کار این مثال به پایان رسید.شما می توانید برای نظم بیشتر صفحه ی FileBrowser خود عناصر تعریف شده را در یک Table قرار دهید و همچنین فرمت و قالب فونت ها و رنگشان و... را زیباتر کنید.از طرف دیگر این مثال یک مثال بی نقص

نیست و خطا و اشکال زیاد دارد که در این بحث نمی توان تمام اشکالات آن را بر طرف نمود.ولی برای شروع می توان از بلوک Try-Catch در کد ها استفاده کرد تا اگر استثنایی خ داد بلافاصله با آن مقابله شود و پس از آن از کنترل های تعیین اعتبار استفاده کرد.در ضمن این مثال کاملا قابل گسترش است و شما می توانید عناصر وقابلیت های مختلفی به آن اضافه کنید مثلا قابلیت Caching را به آن بیفزایید و یا پس از این که جلوتر با کنترل FileUpload آشنا شدید می توانید به لیست فایل هایتان فایل هم اضافه کنید و...

حال به ادامه ی بحث این بخش می پردازیم که نوبت StreamReader & StreamWriter است.

شی FileStream قابلیت های زیادی برای نوشتن و خواندن داده ها دارد.وقتی یک شی از آن را New می کنید تابع سازنده ی آن حدود ۱۵ آرگومان ورودی دارد که البته اکثر آنها Optional هستند.اولین و مهمترین آرگومان آن مسیر فایل است.این مسیر فایلی را که قرار است شما آن را کنترل کنید(ایجاد—درج—حذف—تغییر—و...) را مشخص می کند.آرگومان بعدی نوع مدل ایجاد فایل را مشخص می کند.این نوع از Enumeration شی FileMode ایجاد می شود و مهمترین عناصر آن در زیر آمده:

Append:برای اضافه کردن مقادیر به یک فایلی که قبلا وجود داشته کاربرد دارد.

Create:برای ایجاد یک فایل کاربرد دارد ولی اگر فایل قبلا وجود داشته باشد OverWrite صورت می گیرد.

CreateNew: برای ایجاد یک فایل کاربرد دارد ولی اگر فایل قبلا وجود داشته باشد با خطای استثناها مواجه می شویم.

Open:یک فایل که قبلا وجود داشته را باز می کند.

OpenOrCreate: یک فایل که قبلا وجود داشته را باز می کند وی اگر وجود نداشته باشد آن را ایجاد می کند و سپس باز می کند.

آرگومان بعدی شی FileStream مشخص کننده ی عملیاتی است که شما روی فایل انجام می دهید و ازEnumeration شی FileAccess ایجاد می شود و ۳ عنصر دارد:

Read:فایل مورد نظر را برای خواندن کنترل می کند.

Write:فایل مورد نظر را برای نوشتن کنترل می کند.

ReadWrite:فایل مورد نظر را برای هم خواندن و هم نوشتن کنترل می کند.

در زیر نمونه ای از تعریف شی FileStream را می بینید:

```
Dim fs As New FileStream("C:\a.txt", FileMode.Open, FileAccess.Read)
```

نام فایل مورد نظر a.txt در درایو C است و قرار است این فایل برای خواندن باز شود.

حال برای خواندن و نوشتن شی FileStream از دو شی StreamReader و StreamWriter استفاده می کنیم.آرگومان ورودی تابع سازنده ی این دو شی نیز از جنس FileStream است.پس من برای انجام عملیات خواندن از فایل a.txt که شی FileStream آن را در بالا تعریف کردم نیاز به New کردن یک StreamReader دارم.پس آن را New می کنم و آرگومان ورودیش را نیز همان شی FileStream که در بالا تعریف کردم می دهم:

```
Dim sw As New StreamReader(fs)
```

حالا همه چیز آماده ی خواندن محتویات فایل مورد نظر است.شی StreamReader برای عمل خواندن متد های مختلفی دارد.ولی قبل از معرفی آنها با متد peek آشنا می شویم.این متد مشخص می کند آیا هنگام خواندن کاراکتر به کاراکتر و یا دسته ای از کاراکتر ها عنصر بعدی وجود دارد یا خیر.مقدار برگشتی آن از نوع Integer است و تعداد کاراکتر های خوانده نشده را بر می گرداند.از این متد بیشتر به عنوان شرط حلقه ی While استفاده می شود.حال می پردازیم به متد ها ی خواندن شیStreamReader :

Read:عمل خواندن را کاراکتر به کاراکتر انجام می دهد و مقدار برگشتیش کد ASCI کاراکتریست که می خواند.برای مثال اگر حرف اول فایل a.txt حرف A باشد کد زیر مقدار ٦٥ را به خروجی می برد:

```
sw.Read
```

برای خواندن تمامی کاراکتر ها از متد peek در یک حلقه ی While استفاده می کنیم:

```
  While sw.Peek >= 0
     TextBox1.Text += sw.Read
End While
```

این یعنی تا وقتی کاراکتر های باقیمانده, از ۰ بزرگترند , عمل خواندن را ادامه بده و این کد مقادیری که خواندن می شود را به TextBox1 اضافه می کند.برای تبدیل کد ASCI به کاراکتر, از متد Convert.ToChar استفاده می کنیم.به این ترتیب کد زیر محتوای واقعی فایل a.txt را درون TextBox به خروجی می برد:

```
  While sw.Peek >= 0
     TextBox1.Text += Convert.ToChar(sw.Read)
End While
```

همیشه پس از استفاده از StreamReader یا StreamWriter باید آن را ببندید تا پردازش فایل مربوطه برای استفاده ی دیگر پردازنده ها باز شود:

```
sw.Close()
```

پس کد کلی به فرم زیر شد:

```
  Dim fs As New FileStream("C:\a.txt", FileMode.Open, FileAccess.Read)
  Dim sw As New StreamReader(fs)
  While sw.Peek >= 0
     TextBox1.Text += Convert.ToChar(sw.Read)
  End While
sw.Close()
```

ReadLine:عمل خواندن را خط به خط انجام می دهد.مثلا کد زیر تنها خط اول فایل a.txt را در TextBox به خروجی می برد:

```
TextBox1.Text = sw.ReadLine
```

برای تداوم آن باز هم می توان از متد peek در یک حلقه ی While استفاده کرد:

```
  While sw.Peek >= 0
     TextBox1.Text += sw.ReadLine
End While
```

این کد محتویات هر خط از فایل a.txt را به TextBox اضافه می کند ولی رفتن به خط بعد ها را در نظر نمی گیرد در حالیکه متد Read آنها را در نظر می گرفت.اگر هم خودتان یک
 به کد بالا اضافه کنید TextBox آن را به همان صورت
 نمایش می دهد مگر اینکه شما هر خطی که می خوانید را درون یک TextBox جداگانه قرار دهید.

ReadToEnd:کل فایل را به همان صورتی که بود خوانده و به فرم یک String ساده به خروجی می برد:

```
  Dim fs As New FileStream("C:\a.txt", FileMode.Open, FileAccess.Read)
  Dim sw As New StreamReader(fs)
  TextBox1.Text = sw.ReadToEnd
sw.Close()
```

حال می خواهیم کمی با نوشتن در فایل توسط StreamWriter کار کنیم.پس یک FileStream به شکل زیر new می کنیم که یک فایل جدید ایجاد کند(اگر قبلا وجود داشت روی آن بنویسد).نام فایل را نیز از یک TextBox می گیریم:

```
Dim fs As New FileStream("C:\" & TextBox2.Text & ".txt", FileMode.Create,
FileAccess.Write)
```

قبل از نوشتن باید شی StreamWriter را New کنیم:

```
Dim sw As New StreamWriter(fs)
```

برای نوشتن دو متد وجود دارد :

Write:هر چیز به آن بدهید در یک خط می نویسد.

WriteLine:در هر مرتبه داده ای که به آن می دهید را در یک خط می نویسد.

```
  sw.WriteLine(TextBox1.Text)
sw.Close()
```

در این کد من محتوای TextBox1 را خط به خط درون فایل مربوطه می نویسد.

برای اضافه کردن یک متن به متون یک فایل از FileMode.Append استفاده می کنم:

```
  Dim fs As New FileStream("C:\a.txt", FileMode.Append, FileAccess.Write)
  Dim sw As New StreamWriter(fs)
  sw.WriteLine(TextBox1.Text)
sw.Close()
```

در این کد محتویات TextBox1 به ادامه ی محتویات فایل a.txt اضافه می شود.

شی StreamWriter یک آرگومان ورودی دیگر علاوه بر FileStream دارد که نوع Encoding را مشخص می کند که برای استفاده از آن باید از System.Text.Encoding استفاده کنید .مثلا اگر خواستید متنی که در یک فایل می نویسید از نوع UTF8 باشد, باید کد را به صورت زیر بنویسید:

```
  Dim fs As New FileStream("C:\a.txt", FileMode.Create, FileAccess.Write)
  Dim sw As New StreamWriter(fs, System.Text.Encoding.UTF8)
  sw.WriteLine(TextBox1.Text)
sw.Close()
```

می توانستید ابتدا فضای نام System.Text.Encoding را imports کنید و سپس در آرگومان دوم شی StreamWriter از UTF8 تنها(بدون System.Text.Encoding) استفاده کنید.

در زیر مثالی از خواندن با متد FileMode.Open را می بینید:

```
  Dim fs As New FileStream("C:\" & TextBox2.Text & ".txt", FileMode.Open,
FileAccess.Read)
  Dim sr As New StreamReader(fs)
  TextBox1.Text = sr.ReadToEnd
sr.Close()
```

پیش از این هم با متد هایی آشنا شدید نظیر File.CreateText که یک فایل ایجاد می کرد و مقدار برگشتی آن از نوع StreamWriter بود.پس برای نوشتن در آن فایل باید به صورت زیر عمل کنید:

```
  Dim sr As StreamWriter = File.CreateText("C:\sssd.txt")
  sr.Write("Your Text Here...")
sr.Close()
```

و یا مثلا متد OpenText یک فایل را باز می کرد و مقدار برگشتیش از جنس StreamReader بد پس برای خواندن آن نیز باید به صورت زیر عمل کنید:

```
  Dim sr As StreamReader = File.OpenText("C:\a.txt")
  Response.Write(sr.ReadToEnd())
sr.Close()
```

کاربرد مهم نوشتن در فایل برمی گردد به همان بحث وب سایتهایی که خدمات Hosting ارایه می دهند.در File Manager آن وب سایت ها یک قسمت وجود دارد که به شما اجازه ی ساختن فایل های Html(نه Upload کردن) می دهد.وقتی شما خواستید فایل Html بسازید ابتدا نام فایل را از شما می پرسند و سپس یک TextBox به صورت MultiLine ایجاد می شود و از شما می خواهد کد html خود را در آن بنویسید.البته آن وب سایت هیچ مسولیتی مبنی بر صحت و یا عدم صحت کد htmlی که شما می نویسید ندارد.پس از نوشتن کد روی یک دکمه مثلا Create کلیک می کنید و سپس می بینید یک فایل html به لیست فایل هایتان اضافه شده و با اجرای آن, خروجی کد htmlی که نوشتید را می بینید.البته فایل های لازم برای اجرای درست صفحه ی html نظیر تصاویر و یا Flash ها را خودتان باید در سایت Upload کنید و سپس از آدرس درست مسیرشان در صفحه ی html استفاده کنید.جلوتر با نحوه ی Upload درست فایل آشنا می شویم.خوب برای ایجاد چنین سیستمی در وب سایت خود نیاز به دو TextBox و دو دکمه است:

```
  Html File Name:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br
/>
  Html Code:<br />
  <asp:TextBox ID="TextBox1" runat="server" Height="304px"
TextMode="MultiLine" Width="353px"></asp:TextBox><br />
<asp:Button ID="Button1" runat="server" Text="Create" />
```

به رویدا کلیک دکمه می رویم و در ابتدا نام فایل را(از TextBox) با کلاس Path بازیابی می کنیم:

```
Dim filename As String = Path.GetFileNameWithoutExtension(TextBox2.Text)
```

پس حالا ما نام فایل را داریم.مشکلی که ممکن است پیش آید این است که اگر مدل ما FileMode.CreateNew باشد آنگاه به شما اجازه ی ایجاد دو فایل همنام را نمی دهد.پس اگر چنین شود و شما بخواهدی یک فایل ایجاد کنید که قبلا وجود داشته با خطا مواجه می شویم.اگر از FileMode.Create استفاده کنیم OverWrite رخ می دهد و این هم از حرفه ای ها به دور است.اگر Append کنید که اصلا کد html به هم می ریزد

و ....پس برای Handle کردن این استثنا از یک بلوک Try Catch استفاده می کنیم.برای Handle کردن این استثنا که فیلی که دارید می سازید قبلا وجود دارد از IOException استفاده می کنیم.پس قبل از هر چیز دو شی FileStream و StreamWriter را تعریف می کنم ولی New نمی کنم چون به مقدارشان در داخل بلوک Try Catch نیاز دارم تا بتوانم Exception Handling را انجام دهم:

```
  Dim sw As StreamWriter = Nothing
Dim fs As FileStream = Nothing
```

سپس در داخل بلوک Try آنها را New می کنم:

```
  fs = New FileStream("C:\" & filename & ".html", FileMode.CreateNew,
FileAccess.Write)
sw = New StreamWriter(fs, UTF8)
```

فرمت متن html هم Unicode-ft8 در نظر گرفتم.

حال عمل نوشتن را با متد WriteLine انجام می دهم و StreamWriter را می بندم:

```
  sw.WriteLine(TextBox1.Text)
sw.Close()
```

به این ترتیب تک تک خطوطی که کاربر به عنوان کد html وارد می کند با WriteLine به صورت خط به خط به فایل html متقل می شود.

حالا وارد بلوک Catche می شویم و Exception Handling را انجام دهیم:

```
  Catch ex As IOException
    Response.Write("Error Occured," & ex.Message)
End Try
```

پس کد کلی به صورت زیر شد:

```
  Dim filename As String = Path.GetFileNameWithoutExtension(TextBox2.Text)
  Dim sw As StreamWriter = Nothing
  Dim fs As FileStream = Nothing
  Try
     fs = New FileStream("C:\" & filename & ".html", FileMode.CreateNew,
FileAccess.Write)
     sw = New StreamWriter(fs, UTF8)
     sw.WriteLine(TextBox1.Text)
     sw.Close()
  Catch ex As IOException
     Response.Write("Error Occured," & ex.Message)
End Try
```

دقت کنید که من نمی توانستم sw.close را در داخل بلوک Finally قرار دهم چون هنگامی که استثنا رخ می دهد و کد داخل بلوک Try اجرا نمیشود وارد بلوک Finally می

شویم ولی اگر کمی دقت کنید می بینید که sw هنوز مقداردهی نشده (یعنی New StreamWriter(fs,UTF8) اجرا نشده)

بنابراین عمل sw.close برای شی StreamWriter خالی با خطای NullReference مواجه خواهد شد.پس به این دلیل من sw.close را نیز در کنار سایر کد ها دون بلوک Try قرار دادم .البته می توانستید یک شرط if بر سر آن بیاورید تا هنگامی که استثنا رخ داد و sw خالی بود دیگر عمل بسته شدن صورت نگیرد:

```
  Dim filename As String = Path.GetFileNameWithoutExtension(TextBox2.Text)
  Dim sw As StreamWriter = Nothing
  Dim fs As FileStream = Nothing
  Try
  fs = New FileStream("C:\" & filename & ".html", FileMode.CreateNew,
FileAccess.Write)
  sw = New StreamWriter(fs, UTF8)
  sw.WriteLine(TextBox1.Text)
  Catch ex As IOException
      Response.Write("Error Occured," & ex.Message)
  Finally
  If sw IsNot Nothing Then
      sw.Close()
  End If
 End Try
```

در اینجا اگر sw مقداردهی نشود(یعنی New StreamWriter(fs,UTF8) اجرا نشود) خوب sw خالی خواهد بود پس وارد شرط if نمی شویم ولی اگر مقداردهی می شد وارد شرط if می شدیم و sw.close به درستی اجرا می شد.

پس در کل هر جایی خواستید در داخل بلوک Finally از sw.close استفاده کنید شرط if بالا را مبنی بر خالی نبودن آن برسرش قرار دهید.

همه چیز تا اینجا درست است به جز یک قسمت.اگر کد بالا را اجرا کنید با یک خطای امنیت مواجه می شوید و آن هم فرمت داده ای است که درون TextBox نوشته اید.Asp.NET متوجه می شود که متنی که در داخل فایل قرار است نوشته شود یک متن MarkUp بوده و از این کار ممانعت می کند.ولی اگر شما عواقب این کار را بپذیرید مشکلی نخواهید داشت.برای پذیرفتن آن کافی است صفت ValidationRequest را در تگ Page صفحه ی aspx برابر Flase قرار دهید:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="streams.aspx.vb"
Inherits="streams" ValidateRequest="false" %>
```

به این ترتیب کار به درستی انجام می گیرد و شما می توانید این قابلیت را به File Browser خود اضافه کنید.

Security in ASP.NET 2.0

امنیت از جمله مباحث بسیار مهم در نرم افزار ها و مخصوصا نرم افزار های تحت وب است که بدون آن واقعا همه چیز بی معنی است و امروزه کمتر نرم افزاری از امنیت استفاده نمی کند.امنیت امروزه به شبکه های کامپیوتری و فناوری اطلاعات نیز کشیده شده است.طراحی و خلق یک معماری امن مستلزم شناخت کافی شما از محیط اطراف برنامه ی کاربردیتان است.شما نمی توانید یک برنامه ی امن را ایجاد و طراحی کنید تا وقتی ندانید که چه کسانی به برنامه ی شما دسترسی و با آن تعامل دارند و از چه نقطه ای ممکن است به برنامه تان حمله شود.پس مهمترین ملاک در یک طراحی امن درک کامل از محیط اطراف برنامه ی شماست مانند کاربران و داده های ورودی برنامه و نقاط Thread.از این پس کلمه ی کلیدی Threatرا به معنی تهدید به خاطر بسپارید.چرا که امروزه مدل های تهدید در توسعه ی نرم افزار ها اهمیت بسیاری دارند. Threat Modeling یک راه ساخت یافته برای شناخت هر چه بهتر محیط برنامه ی کاربردی و نقاط قوت و ضعف و تهدید ها و تصمیم گیری در مورد تکنیک های کاهش تهدید ها است.برخی از تهدید ها با استفاده از Threat Modeling و تکنیک های آن کاهش پیدا نمی کنند.در اصل Threat Modeling شما را به شناخت و تحلیل محیط پیرامون برنامه تان هدایت می کند و راه هایی را به شما جهت رفع تهدید ها نشان می دهد و صرفا کلید کاهش تهدید نیست.

درست است که راه های درست و تا بحال معرفی شده ی امنیت , برنامه ی شما را به طور کامل امن نمی کند ولی از یک برنامه ی نا امن که بهتر است.لااقل شما اصول آن را رعایت می کنید.در زیر راههایی برای نوشتن کد امن آمده است:

۱- **هیچ وقت به ورودی کاربر اعتماد نکنید:**هیچ وقت در حین نوشتن برنامه با خود فکر نکنید که کاربر چیز درستی به عنوان ورودی به برنامه می دهد و تا می توانید بد بین باشد و Validation های مختلف بر سر راه ورودی کاربر قرار دهید.
۲- **هیچ وقت از اتصال رشته ای در دستورات SQL استفاده نکنید:**به Sql Injection Attack رجوع شود.
۳- **هیچ وقت داده های حساس را در Hidden Fileds ذخیره نکنید:**بهتر است بگویم اصلا از Hidden Fileds استفاده نکنید سنگین تر ید.
٤- **هیچ وقت داده های حساس را در View State ذخیره نکنید.**

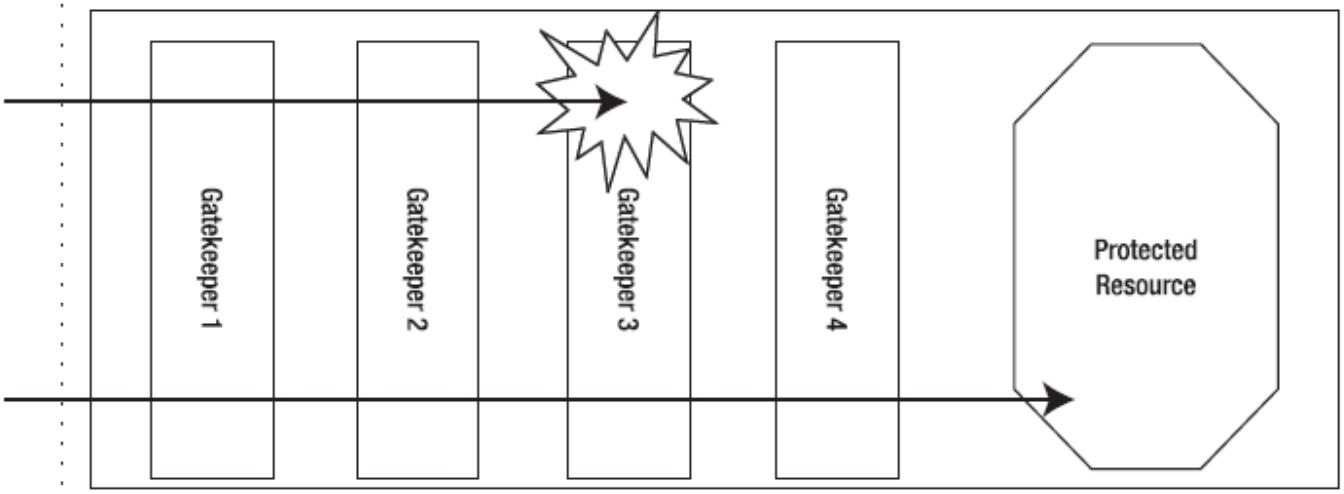
۵- **SSL را در تعیین هویت و اعطا مجوز فعال کنید**:به موقعش در مورد SSL صحبت می کنیم.
۶- **مراقب کوکی ها باشد**:هنگامی که از Forms Authentication استفاده می کنید کوکی ها را دست کم نگیرید.
۷- **از SSL استفاده کنید**:در حالت عمومی برای حفاظت کلی برنامه ی خود از SSL استفاده کنید و فراموش نکنید مسیر ها و فولدر هایی را که حاوی تصاویر و سایر فایل ها است و از طریق برنامه مستقیما از طریق SSL مدیریت نشده.

قبول داریم موارد بالا بسیار کم هستند ولی از هیچ چیز که بهتره.

درک GateKeepers:

یکی از راه های افزایش امنیت برنامه ها است قرار دادن اجزا مختلف بر سر راه اجرای امنیت است.به عبارت دیگر موانع مختلف در بخش حفاظتی بگذاریم تا اگر یکی شکسته شد دیگری نقش امنیت را بر عهده گیرد.GateKeepers یک الگوی مفهومی برای اجرای خط لوله ی زیر ساخت امنیت است.مدلGateKeepers فرض را بر این امر می گذارد که مکانیسم های بسیاری برای حفاظت برنامه ی ما وجود دارند.و هر یک از این مکانیسم ها یک GateKeeper هستند.که بسته به شرایط برنامه هر یک از آنها جهت بالا بردن امنیت به کار می افتند و در صورت شکست هر یک, دیگری جلوی تهدید را می گیرد.

اگر به شکل زیر نگاه کنید مکان GateKeepers و عملکرد خط لوله ی آنها را متوجه می شوید:



در انتهای این خط لوله شما منابع خفاظت شده را می بینید .در فلش پایینی هر یک از GateKeeper ها مجوز ورود Grant Access را دادند و کاربر به سادگی به منابع حفاظت شده رسید ولی در فلش بالایی کاربر پس از گرفتن مجوز از GateKeeper اول و دوم توسط GateKeeper سوم متوقف می شود.ASP.NET حاوی چندین GateKeeper است که با آنها آشنا خواهیم شد ولی قبل از آن سطوح حفاظتی ASP.NET را مرور می کنیم:

۱- **Authentication**:یا تعیین اعتبار که با آن هویت کاربر مشخص می شود.
Authentication به این سوال پاسخ می دهد که چه کسی وارد سایت ما شده و در کل چه کسی با برنامه ی کاربردی ما کار می کند.

۲- **Authorization**:تعیین می کند کاربر تعیین هویت شده به کدام منابع دسترسی دارد و به کدام منابع حق دسترسی ندارد.

۳- **Confidentiality**:یا قابلیت اعتماد به کاربری که تعیین هویت شده و به برخی از منابع دسترسی دارد تضمین می دهد که شخص دیگری به داده های حساس وی دسترسی نخواهد داشت.پس شما باید کانال ارتباطی بین کاربر و سرور را رمز گذاری کنید و از این گذشته نباید حتی خودتان هم به آن داده ها دسترسی داشته باشید(درستش این است ولی خیلی ها سو استفاده ...).

٤- **Integrity**:در نهایت هم باید مراقب باشید داده های انتقالی بین کاربر و یرور توسط کس دیگری تغییر و دخل و تصرف نشود مثلا از امضای دیجیتالی استفاده کنید که به این مورد امانت داری گفته می شود.

البته ASP.NET برای دو مورد اول تدابیر خاصی اندیشیده ولی موارد بعدی را خودتان دستی و با کد نویسی باید فراهم کنید.

Athentication:

فرایندی است که پرده از روی کاربر برمی دارد و هویت وی را مشخص می کند.برای مثال یک جلسه ی کنفرانس را فرض کنید که میز ها ی مختلفی در آن وجود دارد و روی هر میز یک میکروفون وجود دارد.قبل از شروع کنفرانس هر شخص باید یک اعتبارنامه(Credential) داشته باشد و این یک علامت برای شناخت وی است و جلوی

میزش درج شده است و هر کس به نشان شما نگاه بیندازد هویت شما را تشخیص می دهد.مثلا نام و نام خانوادگی.

٤ نوع فرایند برای اجرای Authentication در ASP.NET وجود دارند:

١-Passport Authentication

٢-Forms Authentication

٣-Windows Authentication

٤-Custom Authentication Process

در هر یک از موارد بالا کاربر وقتی به سایت متصل می شود یک اعتبار نامه دریافت می کند .هویت کاربر هم در روشهای بالا ردگیری می شود.به عبارت دیگر در هر یک از روشهای بالا داده ها به نوع متفوتی با دیگری ردگیری می شود.برای مثال سیستم عامل Windows از یک شناسه ی ٩٦بیتی به نام SID یا Security Identify برای تعیین هویت کاربر Log In شده استفاده می کند.در Forms Authentication هویت کاربر در یک کوکی ذخیره می شود.

Authentication بهترین روش برای شخصی سازی وب سایت برای کاربران مختلف است و می توان پیام های مربوط به هر کاربر را روی وب سایت درج کرد که این امر مثلا در سایت Yahoo مشهود است.چنانچه به آن Login کنید می بینید که عبارت Welcome your name در بالای سایت آن درج می شود.ولی تنها Authentication کافی نیست و باید یک Authorization هم داشته باشیم که قبل از توضیح آن با اصطلاح Impersonation یا جعل هویت آشنا می شویم که مربوط به Authentication است.

Impersonation

این قصه ادامه دارد...

یا حق