

فصل اول

آشنایی با توانایی های

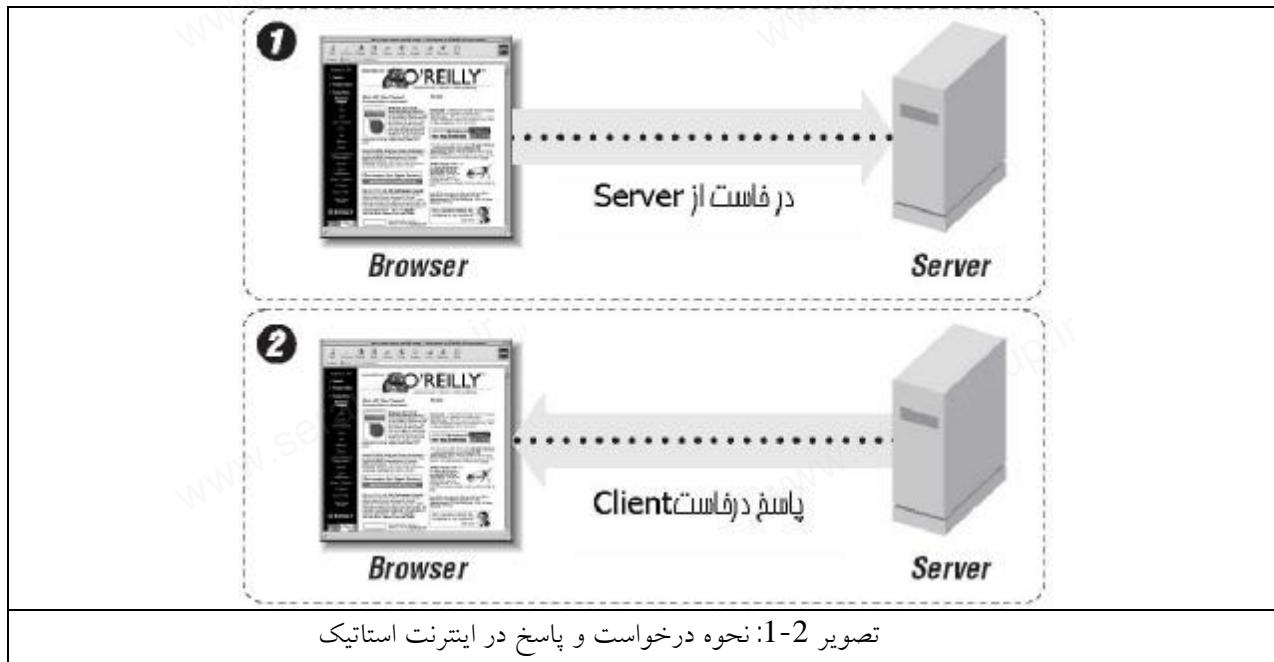
ASP.NET

ASP تکنولوژی است که به شما امکان ساخت برنامه های تحت وب مستقل از مرورگر خاص را با استفاده از اسکریپت طرف سرور می دهد. کد این اسکریپت به هر زبانی می تواند باشد و درون تگ های مخصوصی قرار داده می شود. این اسکریپت چند زبانه درهم که بر روی صفحه قرار دارد هنگام درخواست client برای مضمون به وسیله‌ی Web Server ترجمه می شود.

ایترن特 استاتیک

در اوایل شبکه وب جهانی (world wide web) تمام اطلاعات مورد نیاز مرورگرها به صورت استاتیک بود. به عبارت دیگر محتوای صفحه A برای Client1 همانند محتویات صفحه A برای Client2 می باشد. Web Server به طور دینامیک نمی توانست بخشی از سایت را بسازد و به صورت ساده با فرستادن صفحه HTML استاتیک بار شده بر روی سرور به درخواست‌ها پاسخ می داد. هیچ حالت محاوره‌ای بین استفاده کننده و سرور وجود نداشت. مرورگر درخواست صفحه استاتیک می داد و سرور صفحه را برای او می فرستاد.

اگر چه ایترن特 استاتیک استفاده از گرافیک و صوت را در بر داشت اما هنوز استاتیک بود. در تصویر 1-2 این عملکرد به نمایش در آمده است.



ایترن特 دینامیک:

برنامه‌های کاربردی CGI

یکی از اولین الحاقی‌های ایترن特 استاتیک ساختن رابط بزرگ راه عمومی - (Common Gateway Interface - CGI) این امکان را فراهم کرد که یک مرورگر وب درخواستی را برای اجرا کردن یک برنامه کاربردی وب بر روی Server انتقال دهد، که نتیجه این درخواست فرستادن یک صفحه HTML به مرورگر وب متقاضی می باشد. برنامه‌های

کاربردی CGI انتظارات ما را از وب سایت ها بالا برد و همچنین تحول وب جهانی به اشتراک گذاشتن اطلاعات برای پایگاه ماندگار را در بر داشت.

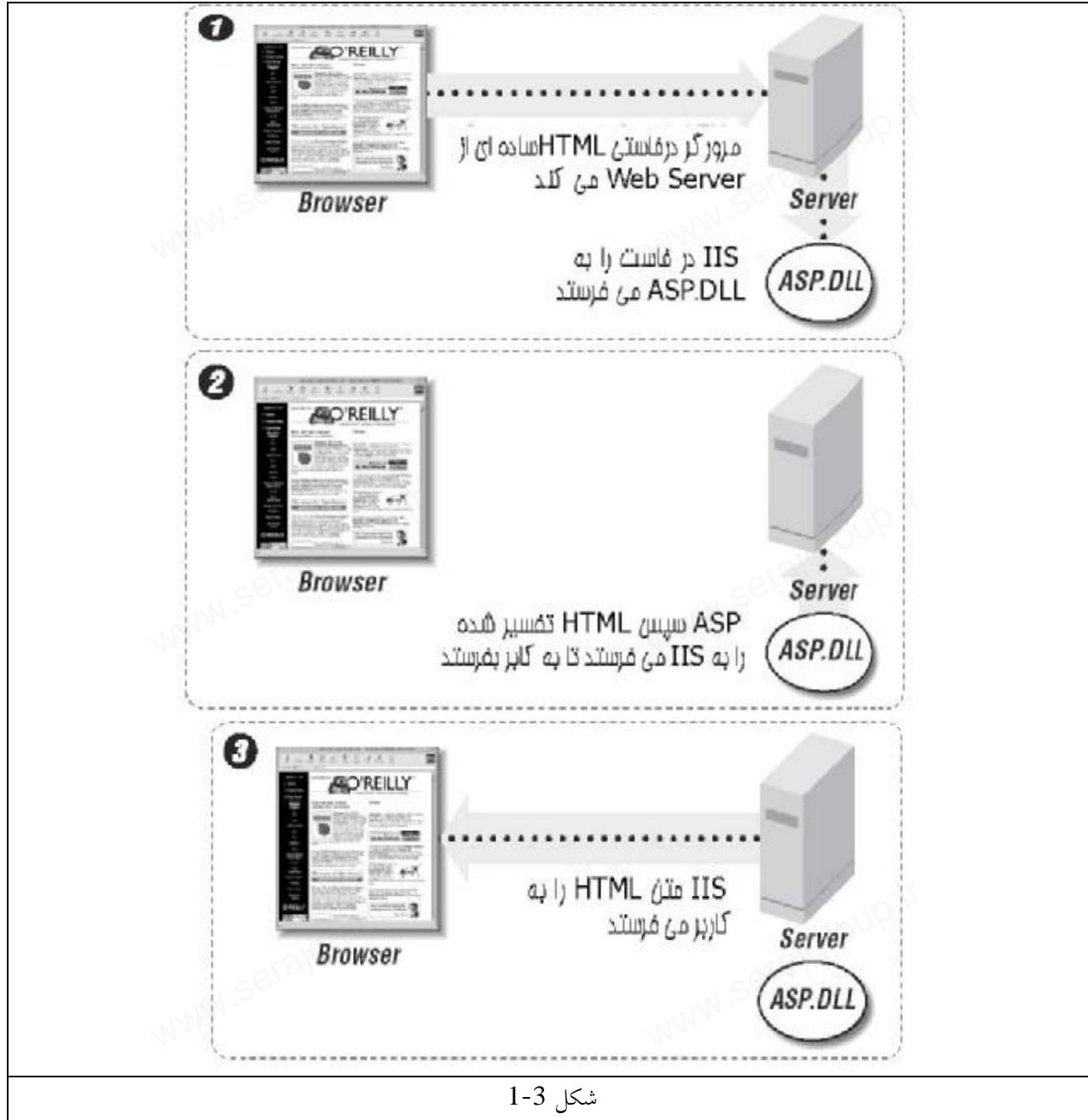
این تکنولوژی باعث ساخته شدن اسکریپت های طرف کاربرشد که این ماشین ها را قادر به انجام بخشی از وظایف می کرد. از مهمترین این ها Netscape's Javascript و Microsoft's VBscript می باشند. در طول رشد این تکنولوژی میکرو سافت (Internet Information Server) خود را منتشر کرد. استفاده آسان، قابلیت اجرا بر روی کامپیوتر های مختلف، ایمن و توسعه پذیر بودن آن باعث محبوب شدنش شد.

ISAPI

در ادامه برای حمایت از ویژگی های CGI میکروسافت Internet Server Application را ساخت. هر زمان که نویی درخواستی برای اجرای یک برنامه کاربردی Web Programming Interface Server کرد نمونه ای جدا از برنامه را اجرا کرده و نتایج اجرای آن را به نوی ارائه می دهد. مشکل این روش بار کردن یک برنامه CGI برای هر درخواست می باشد. اگر درخواست های زیادی وجود داشته باشد منابع سرور جوابگو خواهد بود. این ISAPI بار زیاد را با منتقل کردن آن بر روی Dynamic Link Libraries (DLLs) کاهش داده است. هر برنامه کاربردی ISAPI در قالب ساده ای از DLL که در فضائی مانند فضائی که Web Server برای اولین درخواست برای برنامه کاربردی گماشته است قرار می گیرد. تنها یک بار در حافظه بار شده و DLL در حافظه باقی می ماند و تا زمانی که در حافظه قرار دارد به درخواست ها پاسخ می دهد. ISAPI نسبت به CGI سریعتر عمل می کند زیرا Web Server در هر درخواست برنامه کاربردی جدیدی را معرفی نمی کند. تنها یک بار DLL برنامه کاربردی ISAPI در حافظه بار می شود و Web Server نیازی به بار کردن دوباره آن ندارد. همچنین ISAPI امکان توسعه فیلتر های ISAPI را می دهد. فیلتر ISAPI یک DLL در حافظه Web Server می باشد که توسط Web Server برای پاسخ به هر درخواست HTTP فرخوانی می شود.

Active Server Pages and Active Server Page 2.0

در گذشته بعد از به وجود آمدن IIS 2.0 شرکت میکروسافت تست نسخه بتا تکنولوژی را آغاز کرد که Denali نام داشت. اینک این تکنولوژی به ASP شناخته می شود که یکی از استراتژی های مهم Microsoft's IIS تکنولوژی در بسته ای کوچک (~300K) در قالب DLL به نام ASP.DLL عرضه شده است. این DLL به صورت فیلتر ASP در فضائی از IIS قرار دارد. هر گاه یک کاربر در خاست یک فایل با پسوند ASP را می کند فیلتر ISAPI ASP را اجرا می شود. سپس DLL های مفسر زبان اسکریپتی درخاست شده را در حافظه بار می کند و تمام اسکریپت های طرف سرور صفحه را اجرا می کند و نتیجه HTML آن را به Web Server ارسال می کند، سپس برای مرورگر متقاضی فرستاده می شود. به صورت شکل 1-3.



شکل 1-3

مثال 1-1 یک اسکریپت طرف سرور

```
<%@ LANGUAGE = "VBScript"%>
<HTML>
<HEAD><TITLE>Sample ASP</TITLE></HEAD>
<BODY>
Hello, Welcome to my page.
It is now approximately
<%=time()%>
```

at the server.

```
<% for count=1 to 5 %>
<FONT SIZE=<%=count%>>
Hello size<%=count%>
</FONT>
<%NEXT%>
</BODY></HTML>
```

بعد اجرای این اسکریپت بر روی سرور کد HTML ذیل به کاربر فرستاده می شود.

```
<HTML>
<HEAD><TITLE>Sample ASP</TITLE></HEAD>
<BODY>
Hello, Welcome to my page.
It is now approximately
2:10:20
at the server.
<FONT SIZE=1>Hello size=1</FONT>
<FONT SIZE=2>Hello size=2</FONT>
<FONT SIZE=3>Hello size=3</FONT>
<FONT SIZE=4>Hello size=4</FONT>
<FONT SIZE=5>Hello size=5</FONT>
</BODY>
</HTML>
```

سرور درخواست را پذیرفته و ASP.DLL اسکریپت طرف سرور آن را اجرا و سپس متن HTML حاصل را به کاربر

می فرستد.

مدلهای شی ASP

از شش خصوصیت و متدهای تشکیل شده از قرار ذیل:

- Application
- Object Context
- Request
- Response
- Server
- Session

این شی ها بخشی از ASP.DLL می باشند و همیشه برای برنامه کاربردی ASP شما قابل دسترسی است.

برنامه ASP کاربردی شما را مشخص می کند این شی برای تمامی کاربرانی که وارد برنامه شده اند عمومی است و تنها یک Application Object برای کلیه کاربران وجود دارد. دو رویداد دارد که هنگامی که اولین کاربر درخواست صفحه ای از برنامه شما می کند Application_OnEnd و Application_OnStart که مدیر شبکه برنامه را با استفاده از Microsoft Management Console از روی شبکه بر می دارد. شی Object در واقع بخشی از Microsoft Transaction Server می باشد و تنها درون ASP ربط داده شده. این شی به شما امکان ساخت ASP های با قدرت ثبت تغییرات را می دهد. شی Request روشی را برای عمل کردن با در خاست HTTP کاربر ارائه می دهد. این یکی از مهمترین مدل شی ASP است این شی در میان شی Request که شما در محدوده HTML با

آن کار می کنید و پارامترهایی که با خط آدرس می فرستید. با استفاده از این شی شما از اطلاعات کوکی HTTP و اطلاعات کابر خود می توانید استفاده کنید.

شی Response دستیابی و کنترل شما بر روی HTTP که به عنوان پاسخ به کاربر فرستاده می شود. از طریق این شی شما قادر به فرستادن کوکی ها به کاربر می باشید. شی Server امکان دست یابی به web server را می دهد. درون این شی شما می توانید متغیر timeout برای اسکریپت های خود تعیین کنید این متغیر بیانگر مدت زمانی است که سرور هنگام بروز خطا باید بعد از آن ادامه ندهد. مهمترین متد شی Create Object Server می باشد که امکان ساخت نمونه کامپونت های طرف سرور را می دهد. شی Session دارای اطلاعات منحصر به فرد هر کاربر بر روی سرور می باشد به طوری که هر کابر یک Session مخصوص خود را دارد. شی Web Server برای هر کاربر یک Session در نظر می گیرد.

اسکریپت نویسی

در کل دو نوع اسکریپت داریم این تقسیم بندی بر اساس محل اجرای اسکریپت می باشد یکی بر روی سرور و دیگری بر روی کامپیوتر کاربر. نمونه از اسکریپت طرف client

```
<SCRIPT LANGUAGE = "JavaScript">
<!--
Function AlertJS()
{
    Alert("Hello word.")
}
à
</SCRIPT>
```

```
<SCRIPT LANGUAGE = "VBscript">
<!--
Sub AlertJS()
    MsgBox("Hello word.")
Endsub
à
</SCRIPT>
```

دو علامت (--) و (à) این امکان را می دهد که Browser که تگ <script> را حمایت نمی کند از محتویات درون این دو علامت چشم پوشی کند. کد کامل این اسکریپت به همراه متن HTML برای فرستاده می شود این کد را مرورگر اجرا می کند این کد برای کاربران قابل مشاهده می باشد به همین دلیل کد های زیاد مهمی نیستند و بیشتر برای کنترل روند سایت استفاده می شود مانند چک کردن یک جعبه متن قبل از فرستادن اطلاعات آن به سرور. باید توجه داشت که کد های اصلی برنامه باید طرف سرور باشند زیرا دانستن کد برنامه کمک بزرگی برای هکرهای می باشد تا راحتر سایت را حک کنند. این کد در قالب HTML در هر جایی قابل نوشتن می باشد ولی برای سازمان دهی بهتر به کد آن را در انتهای متن HTML می نویسیم.

```
<HTML>
<BODY>
<%
    Strtime =time()
    Response.write(Strtime)
%>
</BODY>
</HTML>
```

ویژگی خوب اسکریپت طرف سرور این است که کد آن توسط کاربر قابل رویت نیست متن HTML کد بالا که برای مرورگر client فرستاده می شود به صورت زیر است:

```
<HTML>
<BODY>
2:12:50
</BODY>
</HTML>
```

نمونه دیگر

```
<SCRIPT LANGUAGE = "JavaScript" RUNAT="server">
Function AlertJS()
{
    Alert("Hello word.")
}
</SCRIPT>
```

کد های مهم مانند باز و بسته کردن بانک اطلاعاتی ذخیره کردن در بانک در اسکریپت های طرف سرور انجام می پذیرد.

ASP.NET معرفی اولیه

ASP.NET یک نسخه تکمیل شده ASP کلاسیک است. ASP.NET یک محیط کامل جهت گسترش نرم افزارهای تحت وب است. با اینکه ASP.NET از لحاظ گرامر با ASP کلاسیک شباهت هائی را دارد ولی تکنولوژی فوق با ارائه یک مدل جدید برنامه نویسی بهمراه زیر ساخت های لازم، امکان ایجاد نرم افزارهای تحت وب را با امنیت و استحکام بیشتر فراهم می آورد. ASP.NET برخلاف ASP کلاسیک، کامپایل می گردد. در محیط دات نت می توان با استفاده از هر یک از زبانهای برنامه نویسی حمایت شده نظیر: Jscript.Net, C#, VisualBasic.Net اقدام به نوشتن برنامه ها نمود. برنامه های ASP.NET از تمامی توان و پتانسیل های ارائه شده توسط دات نت استفاده می نمایند. در ASP.NET می توان از ویرایشگرهای ویژوال و سایر ابزارهای برنامه نویسی نظیر ویژوال استودیو دات نت استفاده نمود. پیاده کنندگان نرم افزارهای Web تحت وب بکمک ASP.NET می توانند از دو تکنولوژی عمدده فرم های وب (Web Forms) و سرویس های وب (service) برای ایجاد نرم افزار استفاده نمایند.

با استفاده از تکنولوژی فوق می توان صفحات وب متکی بر فرمهای وب قدرتمندی را ایجاد نمود. در زمان ایجاد صفحاتی از این نوع می توان از کنترل های سرویس دهنده ASP.NET برای ایجاد عناصر معمولی در طراحی رابط کاربر (UI) و برنامه نویسی آنها برای انجام عملیات خاص استفاده نمود. استفاده از کنترل های سرویس دهنده باعث سرعت در امر پیاده سازی فرم های وب خواهد داشت.

سرویس های وب XML

این نوع سرویس ها امکان دستیابی به قابلیت ها و پتانسیل های سرویس دهنده را از راه دور فراهم خواهند کرد. با استفاده از سرویس های فوق می توان بخشن منطق و سیاست های راهبردی نرم افزارها و همچنین دستیابی به داده ها را مدیریت نمود. سرویس های وب XML امکان مبادله داده بین سرویس گیرنده و سرویس دهنده و یا بین دو سرویس دهنده را بوجود می آورد. برای تبادل اطلاعات می توان از پروتکل های ارتباطی استانداری نظری http و یا پیامهای XML استفاده نمود. نکته قابل توجه در رابطه با سرویس های فوق توانائی هر برنامه (صرفه از زبان استفاده شده) و تحت هر نوع سیستم عامل برای استفاده از سرویس های فوق است.

دو مدل فوق قادر به استفاده از تمامی مزایای تکنولوژی های ASP.NET خواهند بود. بدیهی است استفاده از پتانسیل های محیط دات نت نیز در این زمینه وجود دارد. در ادامه به برخی از این ویژگی های ASP.NET اشاره می گردد.

شماحت ها و برتریهای ASP Classic با ASP.NET

- اگر دارای تجربی در زمینه پیاده سازی نرم افزار های تحت وب بکمک تکنولوژی ASP باشید، در اولین نگاه به ASP.NET "حتما" متوجه برخی شماحت های موجود خواهید شد. البته مدل اشیاء ASP.NET بصورت کاملاً "آشکار با کلاسیک تفاوت داشته و می توان این ادعا را داشت که ASP.NET بمراتب ساختیافته تر و شی گراء تر شده است. با توجه به مسئله فوق می بایست به این نکته نیز اشاره گردد که ASP کلاسیک بطور کامل سازگار نبوده و تقریباً تمامی صفحات ASP موجود مجبور خواهند بود شاهد برخی تغییرات باشند تا امکان اجرای آنان تحت ASP.NET فراهم گردد. یکی دیگر از تغییرات مهم در این زمینه، وجود VBscript.NET است. در صفحات ASP فعلی از VBscript استفاده شده است که بنوعی این زبان در دات نت مورد توجه قرار نگرفته و VisualBasic.NET جایگزین شده است.

- دستیابی به بانک های اطلاعاتی از طریق برنامه های ASP.NET بعنوان یک نیاز اساسی برای اغلب برنامه های تحت وب مورد توجه خاص قرار گرفته است. در این راستا ASP.NET امکانات بیشتر و بمراتب راحت تر از لحاظ بکارگیری را پیش بینی کرده است و حتی امکان مدیریت بانک اطلاعاتی از طریق کدهای نوشته شده نیز وجود خواهد داشت.

- ASP.NET با ارائه یک مدل ساده به پیاده کنندگان نرم افزارهای تحت وب این امکان را خواهد داد که منطق برنامه های خود را نوشتند و آنها را در سطح Application اجراء نمایند. کدهای فوق را می توان در یک فایل متنی با نام Global.asax و یا در یک کلاس کمپایل شده که بعنوان یک اسمبلی بکار گرفته می شود، استفاده نمود.

- امکانات لازم برای دستیابی به Application State و Session state ASP.NET را ارائه نموده است.

▪ برای پیاده کنندگان حرفه ای تر که قصد استفاده از API را دارند (ISAPI رابط برنامه نویسی است که در نسخه قبلی ASP از آن استفاده می گردد) رابط های جدیدتر و کامل تری را با نام IHttpHandler و IHttpModule را ارائه نموده است.

▪ ASP.NET از امکانات و پتانسیل های موجود در دات نت و CLR بمنظور افزایش کارائی برنامه ها بخوبی استفاده می نماید. تمامی کدهای ASP.NET ترجمه می گردند (تفسیر نمی گردنند) در ASP.NET می توان ماثول هائی را که ارتباطی با برنامه ندارند حذف نمود

(factorable) از سرویس های پیشرفته Caching برای افزایش سرعت و کارائی برنامه ها بخوبی استفاده می نماید. ASP.NET بهمراه یک شمارنده برای سنجش میزان کارائی عملکرد برنامه ها، ارائه شده است. شمارنده فوق این امکان را فراهم می آورد که پیاده کنندگان و مدیران سیستم یک برنامه دات نت، عملکرد شاخص های لازم برای افزایش کارائی برنامه ها را مشاهده، بررسی و در صورت لزوم تجدید نظرهای لازم را اعمال نمایند.

▪ اشکال زدائی برنامه های نوشته ASP.NET بكمک دیباگ برای خواهد گرفت. در این حالت می توان با افزودن چندین خط دیباگ در یک صفحه وب نقطه بروز اشکال را بسرعت و بسادگی تشخیص و درنهایت برطرف نمود. ASP.NET در این راستا کلاس جدیدی با نام TraceContext را ارائه نموده که پیاده کنندگان در زمان نوشتن برنامه، قادر به درج دستورات خاص دیباگ در برنامه برای ردیابی خطاهای احتمالی خواهند بود. دستورات فوق صرف "در زمانی که امکان Tracing فعال شده باشد (برای یک صفحه وب و برای تمام برنامه)، اجرا خواهند شد.

▪ دات نت و ASP.NET دارای امکانات لازم برای Authentication و Authorizing مناسب برای برنامه های تحت وب می باشند. امکانات فوق را می توان بسادگی اضافه و یا با سایر مدل های موجود و مورد نظر جایگزین نمود.

▪ مقادیر مربوط به تنظیمات و پیکربندی برنامه های ASP.NET در فایل های XML ذخیره می گردند با توجه به ماهیت فایل هائی از این نوع خواندن و نوشتن درون آنها بسادگی انجام خواهد یافت. هر برنامه می تواند دارای یک پیکربندی مجزاء بوده که در ادامه حیات برنامه و با توجه به نیازهای مطرح شده اعمال تغییرات بسادگی انجام خواهد گرفت.

▪ برنامه های ASP.NET همانند سایر برنامه های تحت وب از مجموعه ای فایل با نوع های متفاوت و دایرکتوری تشکیل می گردند. این فایل ها می توانند صفحات ASP.NET، کنترل های کاربر (User Controls)، فایل های سرویس های وب و فایل های تنظیمات و پیکربندی و اسembly باشند.

ASP.NET معماری

در این بخش نگاهی سریع به ساختار و معماری بکار گرفته شده در ASP.NET خواهیم داشت. طراحان و ایجاد کنندگان تکنولوژی فوق، نهایت سعی خود را نموده که محصول فوق ماثولار و قابل توسعه باشد. مثلاً "در صورتیکه علاقه ای به داشتن مدیریت Session در صفحات ASP.NET نداشته باشیم، می توان آن را با روتین های مدیریتی خود جایگزین نمود. عملیانی که در ASP کلاسیک امکان تحقق آن وجود نداشت. یکی دیگر از اهداف طراحان تکنولوژی فوق استقلال اجرا و عدم وابستگی به IIS است. بدین منظور آیتمی با نام HTTP زمان اجرا، ایجاد شده است. HTTP زمان اجرا، یک زیرساخت اساسی و بمنظور پردازش سطح پایین HTTP را ایجاد خواهد کرد. امکان فوق جایگزینی مناسب و منطقی برای فیلترهای ISAPI و Aشعبات مربوطه بوده و بگونه ای طراحی شده است که توانایی افزودن، حذف و یا جایگزین نمودن عناصر اساسی ASP.NET

را دارا باشد. زمانیکه درخواستی به بخش **Http** زمان اجرا ارسال می گردد، درخواست فوق از بین تعداد زیادی از ماثول های **Http** عبور داده خواهد شد. ماثول های فوق قبل و بعد از اجرای **Handler** اجرا خواهند شد. این ماثول ها، امکان تفسیر و نهایتاً اجرا را فراهم می نمایند. متدهای خاصی بهمراه ماثول های **Http** توسط رویدادها و فایل های **Global.asax** یکسان سازی خواهند شد. چندین متد ماثولار می توانند به هر یک از رویدادهای در سطح برنامه ها، سینک گردند. مثلاً "ماثول های HTTP handler" با استفاده از رویدادی با نام **OnEnter** در **Windows Authentication** و **Authentication Passport** با استفاده از رویدادی با نام **AuthenticateRequest** سینک خواهند شد. درخواست مورد نظر بین هر ماثول حرکت و درنهایت توسط **URLs** پردازش خواهد شد. هندرلرها، بمنظور پردازش درخواست های منفرد استفاده می گرددند. هندرلرها امکان پردازش **URLs** و یا گروه های از ضمائم **URL** را بهمراه یک برنامه فراهم خواهند کرد. برخلاف ماثول ها، فقط یک هندرلر بمنظور پردازش یک درخواست استفاده می گردد. پس از اینکه هندرلر عملیات مربوط به درخواست را به اتمام رساند، درخواست مسیر خود را بصورت وارونه طی نموده و به ماثول برگردانده تا به حیات آن خاتمه داده شود. در زمان حیات یک درخواست، یک شی با نام **object HTTP Context** مسئولیت کپسوله نمودن تمامی اطلاعات مرتبط با شی را بر عهده خواهد داشت.

نحوه پردازش درخواست های مبتنی بر **ASP.NET**

زمانیکه درخواستی برای یک صفحه **aspx** واصل می گردد، درخواست فوق به **handler** مربوطه داده خواهد شد. در صورتیکه اولین مرتبه ای است که صفحه درخواست می گردد، صفحه مورد نظر ترجمه و با کلاس مربوط به کدهای استفاده شده ترکیب خواهد شد. (**CodeBehind** کلاس چیزی را تولید خواهد نمود که کلاس صفحه نامیده می شود) در حقیقت کلاس ایجاد شده بصورت **Dll** بوده و در یک فهرست موقت ذخیره خواهد گردید. (**Cashed**) در ادامه کلاس فوق، اجرا و تمامی منطق مورد نیاز بمنظور اجرای تگ های مورد نظر **Html** تولید و ماحصل عملیات برای مقاضی ارسال خواهد شد. زمانیکه **Majdada** صفحه فوق درخواست گردد، یک نمونه از کلاس فوق که **Cashe** شده است، ایجاد و مجدداً "تگ های **Html**" تولید و پاسخ مربوطه برای مقاضی ارسال خواهد شد. در این مرحله برخی از عملیات نظیر پارسینگ، ترجمه و ... حذف و قطعاً زمان پاسخ گوی به درخواست مورد نظر کاهش پیدا خواهد کرد.

چرخه حیات یک صفحه **ASP.NET**

یکی از تفاوت های اساسی صفحات **ASP** با **ASP.NET**، روشی است که صفحه پردازش می گردد. در مدل **ASP.NET** پردازش صفحه متنکی بر رویداد است. رویداد **Page_Init** اولین رویدادی است که فعال خواهد شد. روتین پاسخگو در مقابل رویداد فوق، مسئولیت مقداردهی متغیرها و سایر کنترل های استفاده شده در صفحه را بر عهده خواهد داشت. در رویداد فوق تمامی کدهای مربوط به مقدار دهنده مستقر خواهند شد. در ادامه رویداد **Page_Load** فعال خواهد گردید. در این لحظه تمامی کنترل ها و صفحات فعال خواهند گردید. رویداد فوق یکی از پرکاربردترین رویدادهای استفاده شده است. کنترل ها در **ASP.NET** دارای رویدادهای مربوط به خود می باشند. مثلاً "یک کنترل **Text Box**، می تواند دارای رویداد **Change** و یا رویداد **Click** باشد. پس از فعال شدن رویداد **Page_Load** تمامی رویدادهای **Change** مربوط به کنترل ها در ابتدا پردازش و در ادامه رویداد **Click** پردازش خواهد شد. قبل از ارائه نمودن صفحه، رویداد **Page_PreRender** فعال و در ادامه صفحه مورد نظر، پس از فعال شدن رویداد **page_unload** از حافظه خارج خواهد شد. در زمان استفاده از **ASP.NET** بدفعتات از رویدادهای فوق استفاده خواهد شد.

برای برنامه نویسی صفحات ASP.NET، می بایست از یکی از دو مدل تک صفحه ای و یا دو صفحه ای استفاده کرد. در مدل تک نسخه ای صرفاً یک فایل با انشعباب aspx را خواهیم داشت (مشابه ASP کلاسیک) که در آن تگ های Html تگ های مربوط به کنترل ها و خود صفحه قرار خواهند گرفت. (در زمانیکه قصد سوئیچ نمودن از مدل ASP کلاسیک را داشته باشیم مدل فوق بسیار موثر و سریع خواهد بود) در مدل دو صفحه ای که با نام Code-Behind نیز نامیده می شود، از دو صفحه با عملکردهای کاملاً متفاوت استفاده می گردد. در اولین صفحه که با انشعباب aspx خواهد بود تگ های Html و تگ های مربوط به کنترل ها قرار خواهند گرفت. در فایل دوم صرفاً کدهای مربوطه قرار خواهند گرفت. انشعباب فایل فوق با توجه به زبان استفاده شده (VB.NET,C#) بصورت aspx.cs و یا aspx.vb مدل فوق توسط ابزار پیاده سازی ویژوال استودیو مورد استفاده قرار می گیرد. در مدل فوق بصورت واقعی عملیات مربوط به تفکیک کد و محتویات انجام خواهد شد.

معرفی برخی از ویژگی های مهم ASP.NET

در این بخش لازم است که به برخی از ویژگی های اساسی ASP.NET اشاره گردد. دارای امکانات گسترده برای عموم علاقه مندان به برنامه نویسی وب است. اگر شما در صفحه پیاده کنندگان نرم افزار قرار دارید، مشاهده خواهید کرد که ASP.NET "عموماً" با ASP کلاسیک، سازگار است. در این راستا می توان از امکانات وسیع ویژوال استودیو استفاده نمود. با استفاده از ASP.NET می توان مجموعه ای از کنترل های سرویس دهنده را بخدمت گرفت. استفاده از کلاس های پایه کتابخانه ای از دیگر مواردی است که با استفاده از آن می توان تعداد خطوط مورد نظر برنامه نویسی بمنظور انجام یک فعالیت را کاهش داد. برنامه نویسان پس از انتخاب زبان دلخواه قادر به نوشتن کدهای مورد نیاز خواهند بود. در صورتیکه علاقه مند به نوشتن کدهای مورد نظر خود بکمک زبان کوبل نیز باشید، این امر امکان پذیر خواهد بود. تاکنون بیش از بیست زبان برنامه نویسی متفاوت توسط پلات فورم دات نت حمایت شده و تعدادی دیگر در راه می باشد. شما همچنین می توانید ایترفیس API32 ویندوز را مستقیماً و از طریق صفحات aspx فراخوانده و از پتانسیل های آن استفاده نمایید. تمامی زبانهای دات نت نظری VB.NET از یک ساختار ساختیافت به بمنظور برخورد با خطای استفاده می نمایند. اشکال زدائی و ردیابی خطای اشکال موارد قابل توجه و تأمل در ASP.NET است. در این راستا می توان اقدام به اشکال زدائی صفحات ASP نمود (مشابه اشکال زدائی فرمها در ویژوال بیسیک). ASP.NET بمنظور افزایش کارائی (اعتمادپذیری و توسعه با وزن دلخواه) طراحی شده است. در دنیای دات نت هر چیزی ترجمه خواهد شد. کدهای ترجمه شده سرعت را به ارمغان خواهند آورد. بمنظور افزایش کارائی از سیستم Cache API استفاده می گردد. ASP.NET قادر به تشخیص و برخورد مناسب با تمامی خوادثی خواهد بود که در زمان اجرای یک برنامه ممکن است بوجود آید (از بین رفتن پردازه ها، بروز بن بست در سیستم، بروز مشکل در حافظه، نمونه هایی در این راستا می باشد). در چنین مواردی پردازه جدیدی ایجاد و مسئولیت حذف پردازه قبلی با مشکل مواجه شده، به آن سپرده خواهد گردید. تمامی درخواست های معطل مانده، قبل از اینکه به عمر پردازه فوق خاتمه داده شود، توسط پردازه پردازش خواهند شد. در این وضعیت تمامی درخواست های جدید واصل شده، به پردازه جدید داده خواهند شد. نکته جالب در این راستا تنظیم و پیکربندی تمامی پارامترهای ذیربسط، توسط برنامه نویس است.

یکی از مهمترین اهداف دات نت، بکارگیری آسان برنامه ها پس از آماده سازی است. ASP.NET از تکنیک معروف Xcopy (تکثیر فولدر مربوطه بهمراه زیر مجموعه های آن) استفاده می نماید. در زمان استفاده از ASP.NET ضرورتی به استفاده از ریجستری بمنظور تنظیم پارامترهای ذیربط نخواهد بود. در این راستا می توان تمامی تنظیمات دلخواه را در فایل XML ذخیره و بهمراه کدهای نوشته شده بر روی کامپیوتر مورد نظر، تکثیر کرد. با استفاده از امکانات ASP.NET و همراهی Mobile Internet Toolkit ، می توان نرم افزارهای وب خود را بمنظور اجرا بر روی دستگاههای نظری: تلفن های سلولی، PDA و .. آماده کرد. دارای امکانات مناسب برای پیاده سازی سرویس های متکی بر وب، کنترل های بیشتر از بعد مسائل امنیتی و انعطاف پذیری بیشتر در مدیریت Session است.

فصل دوم

آشنایی با IIS و

.NetFramework

در این فصل به صورت بسیار کاربردی با نحوه نصب، تنظیم و راه اندازی IIS را برای ایجاد صفحات پویایی وب توسط تکنولوژی ASP آشنا خواهید شد. همچنین نکاتی نیز در مورد نصب .NET Framework. فراخواهد گرفت. پس از مطالعه این فصل شما می توانید:

- 1 IIS را نصب نمایید.
- 2 NetFramework را نصب نمایید.

3- دایرکتوری Home را در IIS تغییر دهید

4- یک دایرکتوری مجازی به IIS اضافه نمایید.

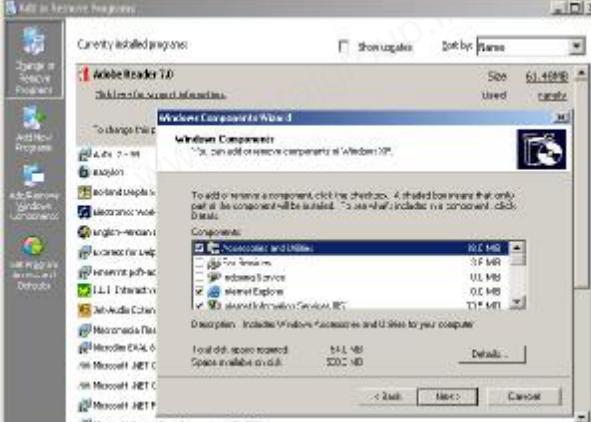
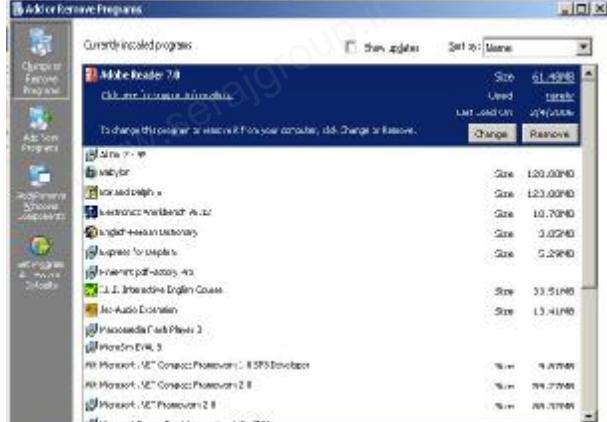
5- صفحه پیش فرض سایت را تعیین نمایید.

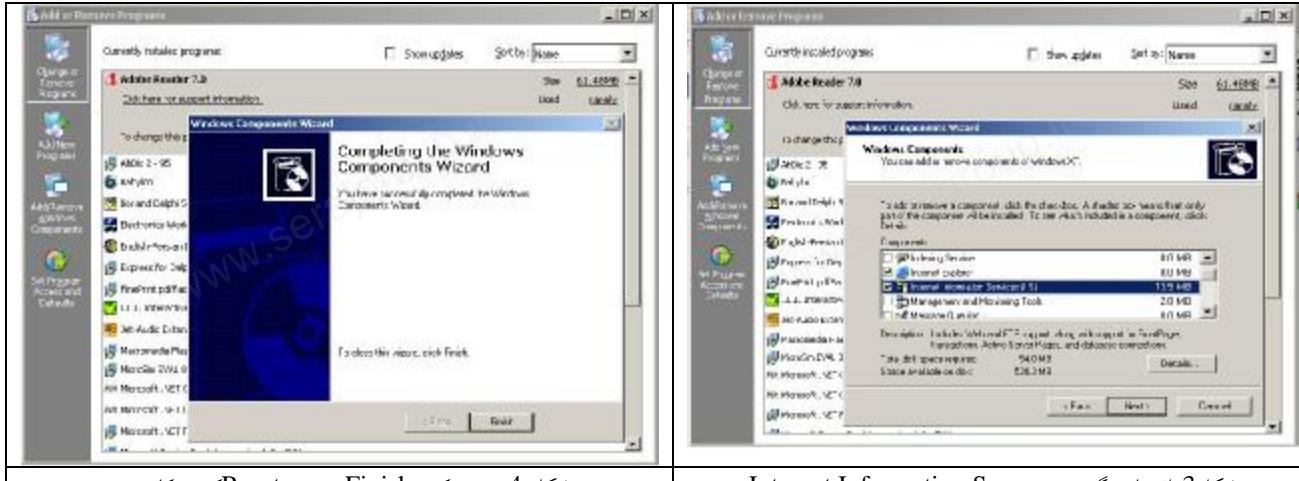
6- خصوصیات فایل Log را برای IIS تغییر دهید

7- یک سایت وب را مکث، متوقف و شروع کنید.

نصب و راه اندازی IIS :

نرم افزار سرور وب مایکروسافت می باشد که برای ایجاد، مدیریت و هاستینگ وب سایت ها مورداستفاده قرار می گیرد. این برنامه بر روی سی دی های ویندوز های 2000 به بالا که بر پایه آن تی هستند موجود می باشد. IIS یک Component ویندوز می باشد که می توان برای نصب به ویزاد مربوط به نصب Component های ویندوز در قسمت Add/Remove programming کنترل پانل مراجعه کرد. در این قسمت می توان واحد مربوط به IIS را انتخاب بعد دکمه NEXT را کلیک کنید تا IIS نصب شود پس از تایید صفحه جاری و فشردن دکمه Next مجموعه IIS نصب می شود (احتمالاً مسیر سی دی ویندوز را هم از شما خواهد پرسید). در این حالت پس از نصب حتماً باید ویندوز را ریست کنید

 <p>شکل 2: انتخاب دکمه Add/Remove Windows Component</p>	 <p>شکل 1: انتخاب Add or Remove Programs</p>
--	--



شکل 4: زدن دکمه Finish و بعد Reset کردن کامپیوتر

شکل 3: انتخاب گزینه Internet Information Service

در گزینه IIS زیر گزینه های زیر موجود می باشد که برای موارد مورد استفاده باید نصب شود:

Documentation: فایل های راهنمایی و مثال های وابسته را نصب می کند.

File Transfer Protocol(FTP) Service: توانایی دانلود و آپلود را به سایت شما اضافه می کند.

FrontPage 2000 server Extention: اگر از ویژوال استودیو یا فرانت پیچ استفاده می کنید بهتر است این گزینه

را انتخاب نمایید.

Internet Service Manager: تکارش تحت وب توانایی های مدیریتی وب سایت.

NNTP Service: اگر به پشتیبانی Network News نیاز دارید آنرا انتخاب نمایید.

SMTP Service: توانایی فرستادن و یا دریافت ایمیل را فراهم می کند.

برای مدیریت IIS می توانید از قسمت Administrative Tools در کنترل پنل Internet service manager را

اجرا کنید



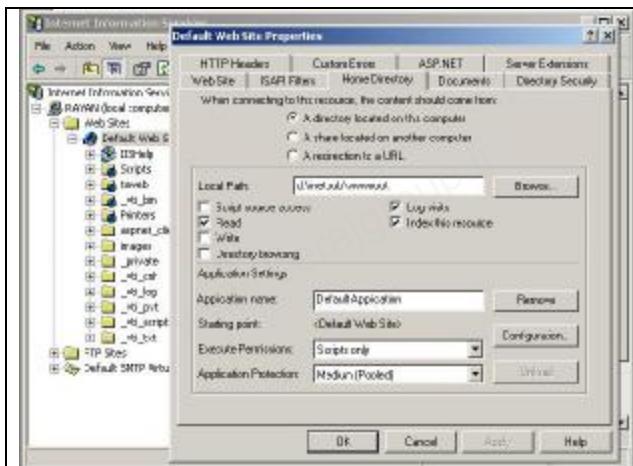
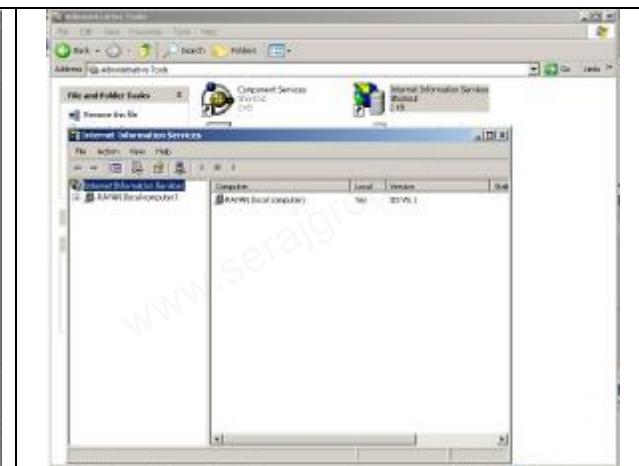
شکل 5: جزئیات قطعه IIS

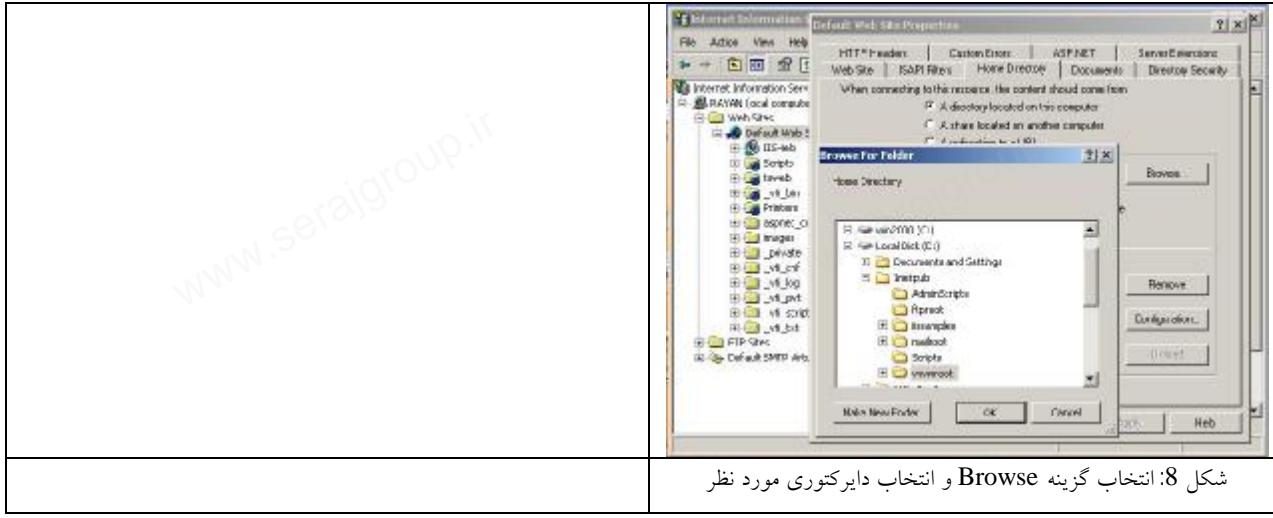
حتماً پس از نصب IIS این کار را انجام دهید. اگر ابتدا آنرا نصب و سپس IIS را نصب نمایید نگارش .NET شما ناقص خواهد شد. این مشکل احتمالاً در نگارش های آتی بر طرف خواهد شد. برای نصب .NET حداقل دوراه وجود دارد راه اول نصب کامل مجموعه ویژوال استودیو است که به همراه آن دات Framework هم نصب خواهد شد. راه دوم استفاده از Setup بیست مگا بایتی .Net Framework است که بر روی سی دی های کامپونت های ویژوال استودیو دات نت، موجود می باشد. نصب آن هیچ نکته خاصی ندارد و فقط بر روی Next کلیک کنید. برای نصب کامل ویژوال استودیو دات نت، چیزی حدود دو گیگابایت را باید کنار بگذارید. بهتر است بر روی کامپیوتر سروری که می خواهید فایل های خودتان را اجرا کنید سی دی کامپونت های دات نت را کامل نصب کنید. حدود 400 مگابایت بیشتر نیست.

تنظیمات IIS

الف) تغییر دایرکتوری Home در IIS

مکانی که فایل های وب سایت شما بر روی آن ذخیره می گردد به نام دایرکتوری Home یا دایرکتوری Root نامیده می شود. شما می توانید مشخص کنید که کدام دایرکتوری روی سرور Web شما دایرکتوری Home باشد. تمام فایل ها و زیر دایرکتوری در داخل این دایرکتوری در سایت شما در دسترس می باشند. به همین خاطر امنیت فایل ها و زیر دایرکتوریها باید بخوبی تعریف شده باشد. مسیر پیش فرض آن <Drive Setup>:\Inetpub\WWWROOT> می باشد و تعویض آن به هر مسیر دیگری توسط IIS امکان پذیر است.

	
<p>شکل 7: انتخاب گزینه Properties کلیک سمت راست روی Default Web Site</p>	<p>شکل 6: انتخاب گزینه Internet Information Service</p>



شکل 8: انتخاب گزینه Browse و انتخاب دایرکتوری مورد نظر

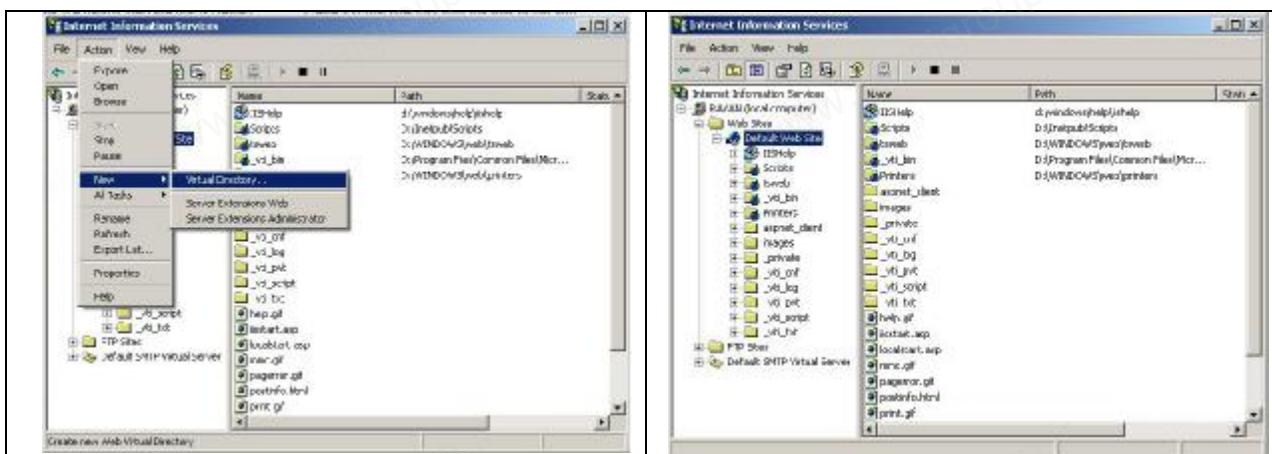
برای این کار Internet Service Manager را از قسمت Administrative tools در کنترل پانل، اجرا کنید. پس از اجرای آن روی Default Web Site کلیک راست کنید و گزینه Properties آن را انتخاب نمایید. صفحه‌ی Home Directory می‌توانید این مسیر پیش فرض را تعویض نمایید. موقعی که کاربران به سایت وب شما وارد شدند، اولین صفحه که می‌بینند صفحه اولیه سایت می‌باشد. این صفحه باید در سایت موجود باشد. نحوه تنظیم کردن آن را در صفحات آتی کتاب بررسی می‌کنیم. در صفحه home Directory گزینه‌های دیگری برای تعیین دایرکتوری Home وجود دارد:

برای مشخص کردن مکانی روی سرور وب به عنوان دایرکتوری Home استفاده می‌شود.	A directory located on computer
در این حالت یک دایرکتوری به اشتراک گذاشته شده روی کامپیوتر دیگر به عنوان دایرکتوری Home در نظر گرفته می‌شود.	A share located on another computer
در این حالت اگر کسی سعی کند به سایت شما دسترسی پیدا کند و به ادرسی دیگر فوروارد خواهد شد.	A redirection to a URL

ب) ایجاد یک دایرکتوری مجازی در IIS

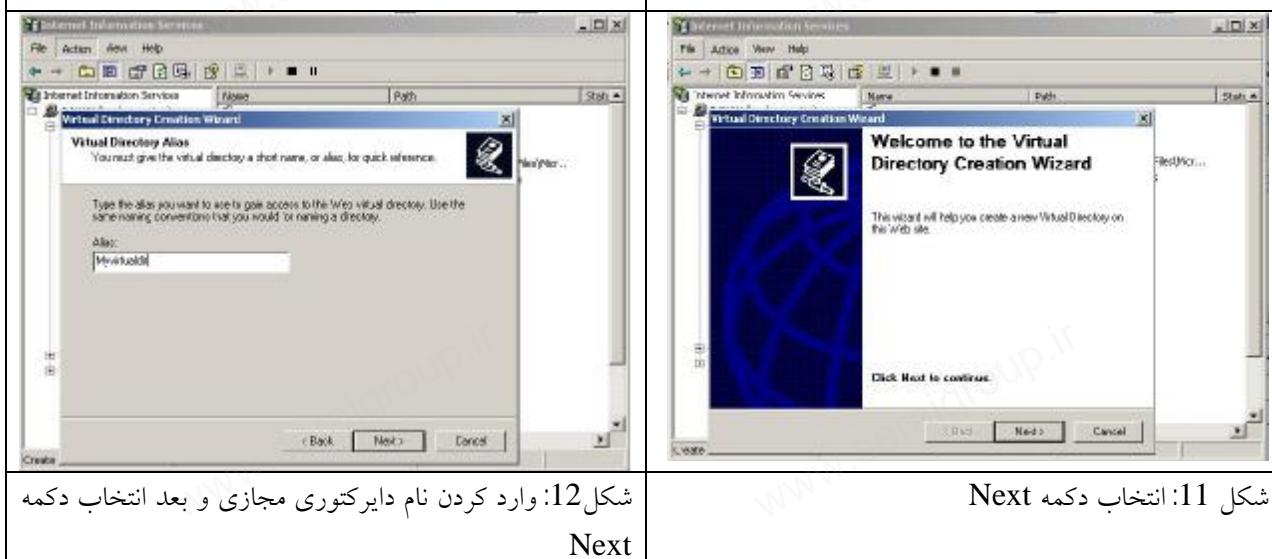
بعد از نصب IIS شما می‌توانید سایت وب خودتان را پیکربندی نمایید. هر سایت وب یک دایرکتوری Home خواهد داشت. برای مثال وب سایت پیش فرض شما بعد از نصب IIS دایرکتوری IIS<install drive>:<inetpub>\wwwroot می‌باشد. تمام زیر دایرکتوری‌ها در این دایرکتوری برای تمام کاربران شما قابل دسترسی هستند. برای مثال فرض کنید که سایت وب شما در www.myweb.com قابل دسترسی باشد. یک زیر دایرکتوری به نام Test در سایت وب پیش فرض اضافه نمایید. بعد از اضافه شدن شما می‌توانید به فایل‌های داخل این دایرکتوری به صورت www.myweb.com/test زیر دسترسی پیدا کنید. با ایجاد دایرکتوری مجازی می‌توان از دایرکتوری‌هایی استفاده کرد که الزاماً زیر دایرکتوری در دایرکتوری وب سایت شما

نیستند. برای مثال از دایرکتوری Home وجود ندارد مانند دایرکتوریهای که در درایور C وجود دارد به سادگی می‌توان با این روش بهره مند شد خصوصاً این روش هنگامیکه شما از چندین سرور استفاده می‌کنید ارزش خودش را نشان می‌دهد.



شکل 10: انتخاب گرینه Action\new\virtual Dirctory

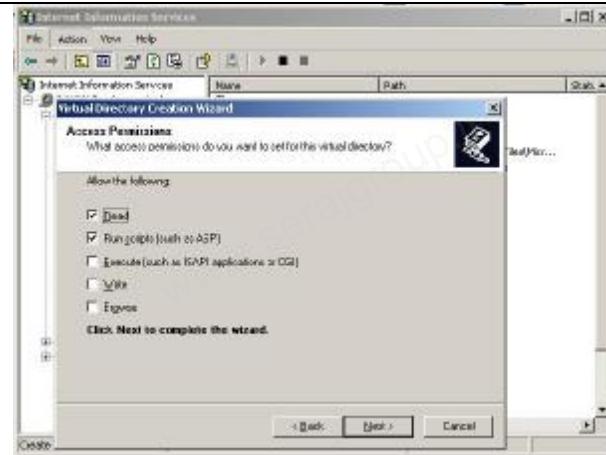
شکل 9: انتخاب گرینه Internet Information Service Administrative tools در



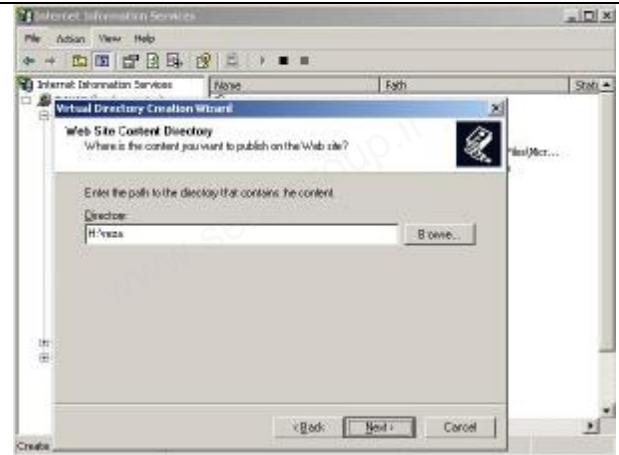
شکل 12: وارد کردن نام دایرکتوری مجازی و بعد انتخاب دکمه Next

شکل 11: انتخاب دکمه Next

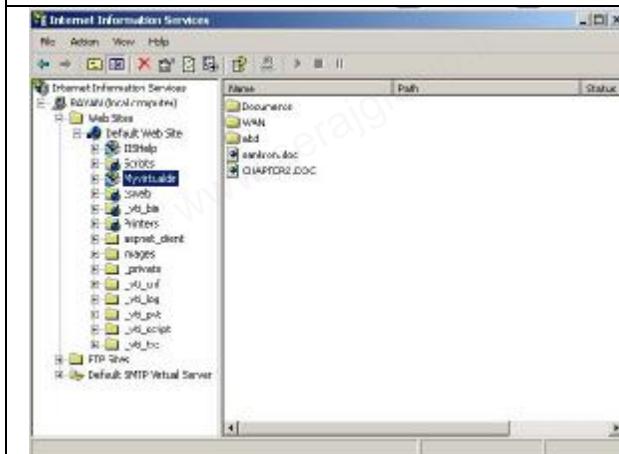
برای ایجاد یک دایرکتوری مجازی Internet Service Manager را اجرا کنید روی دکمه Action از نوار ایزار بالای صفحه آیتم New و سپس Virtual Directory را انتخاب کنید اینکار را با کلیک راست روی آیتم Default Web Site هم می‌توانید انجام دهید سپس در صفحه خوش آمد گویی ظاهر شده روی Next کلیک کنید در صفحه بعد نام دلخواهی را وارد نمایید



شکل 14: مشخص کردن خصوصیات امنیتی و بعد انتخاب Next



شکل 13: مشخص کردن مسیر مورد نظر و بعد انتخاب Next



شکل 16: ایجاد دایرکتوری مجازی



شکل 15: انتخاب دکمه Finsh

بعد از انتخاب نام مناسب مسیر دایرکتوری مورد نظر را مشخص خواهیم کرد، در صفحه بعدی موارد امنیتی مشخص شده اند که پیش فرض انها برای اغلب سایت ها کافی هستند.

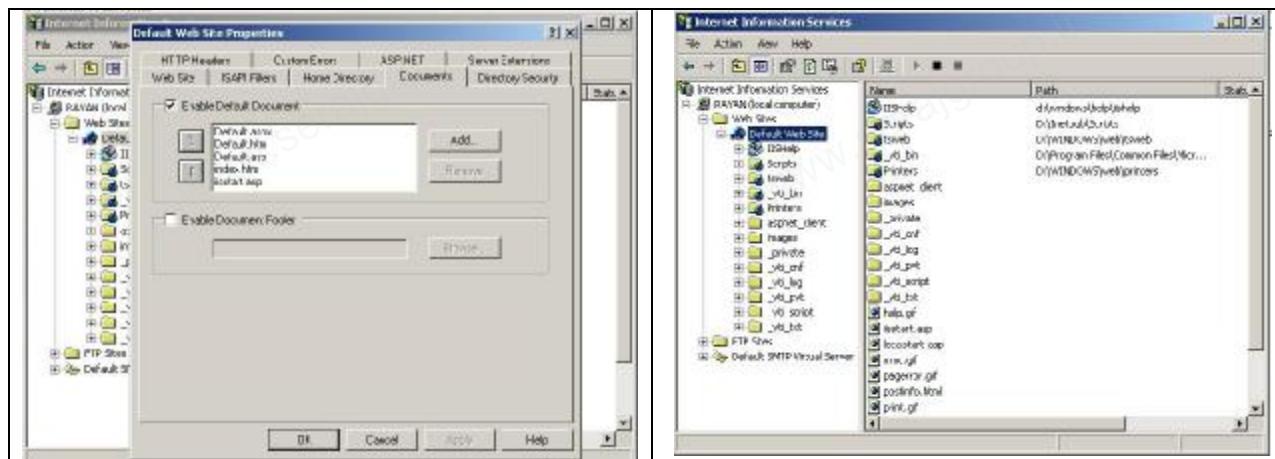
راه دیگری هم برای این کار وجود دارد: با استفاده از Windows Explorer دایرکتوری را که می خواهید به عنوان دایرکتوری مجازی مشخص نمایید، انتخاب کنید. از منوی فایل گزینه Properties را انتخاب نمایید. سپس بر روی Tab مربوط به Web Sharing کلیک کنید و وب سایتی را که می خواهید دایرکتوری مجازی برای آن ایجاد کنید انتخاب نمایید. روی گزینه گزینه Share This Folder کلیک کنید و در صفحه Edit Alias، نام دلخواهی را مشخص نموده و همچنین موارد امنیتی را انتخاب نمایید. سپس روی OK کلیک نمایید. حذف این دایرکتوری مجازی هم در Internet Service Manager امکان پذیر است. فقط کافی است روی آن کلیک راست کرده و Delete را انتخاب کنید. حذف آن خود فایل ها را حذف نمی کند. همانند دایرکتوری home دایرکتوری مجازی هم می تواند یک دایرکتوری را روی سرور مشخص کند یا یک دایرکتوری روی یک کامپیوتر دیگر یا یک URL باشد.

هنگامیکه می خواهید یک دایرکتوری مجازی را ایجاد کنید با 5 گزینه امنیتی بسیار مهم روپرورمی شوید که لازم است مروری بر آنها ارائه شود:

: در این حالت کاربران می توانند به سایت شما دسترسی پیدا کنند و محتویات آنرا مشاهده کنند. (به صورت پیش فرض انتخاب شده است)	Read
توانایی اجرای اسکریپت ها را در دایرکتوری وب ارائه می دهد. در این حالت برای دایرکتوری هایی که صفحات ASP باید در آنها اجرا شوند لازم است. (به صورت پیش فرض انتخاب شده است)	Run Scripts
امکان اجرای برنامه ها را در دایرکتوری مجازی می دهد.	Execute
این مورد برای دایرکتوری مجازی که فایل های ASP موجود در آنها نیاز به ایجاد فایل روی سرور دارند باید فعال شود.	Write
کاربران را قادر می سازد تا تمام سوابق دایرکتوری ها را مشاهده کنند.	Browse

ج) تنظیم صفحه پیش فرض در IIS

اگر صفحه درخواستی مشخص نشود، صفحه که به مرورگر کاربر فرستاده می شود صفحه پیش فرض گفته می شود. در IIS می توان صفر تا تعداد زیادی فایل را برای انجام اینکار مشخص و انتخاب کرد. اگر IIS فایلی را پیدا نکرد یک خطای کاربر نمایش می دهد و اگر امکان Browsing دایرکتوری را شما فعال کرده باشید بجای Error، لیست دایرکتوری ها و فایل ها نمایش داده می شوند. شما می توانید مجموعه مختلفی از صفحات را برای هر دایرکتوری در وب سایت به صورت پیش فرض مشخص کنید. موقع مشخص کردن صفحه پیش فرض همچنین می توانید ترتیب مجموعه صفحات پیش فرض را مشخص کرد. بهتر است از نام های استاندارد زیر برای مشخص کردن این سند پیش فرض استفاده کنید: Index.htm, Default.aspx و Default.htm، مانند اینها.



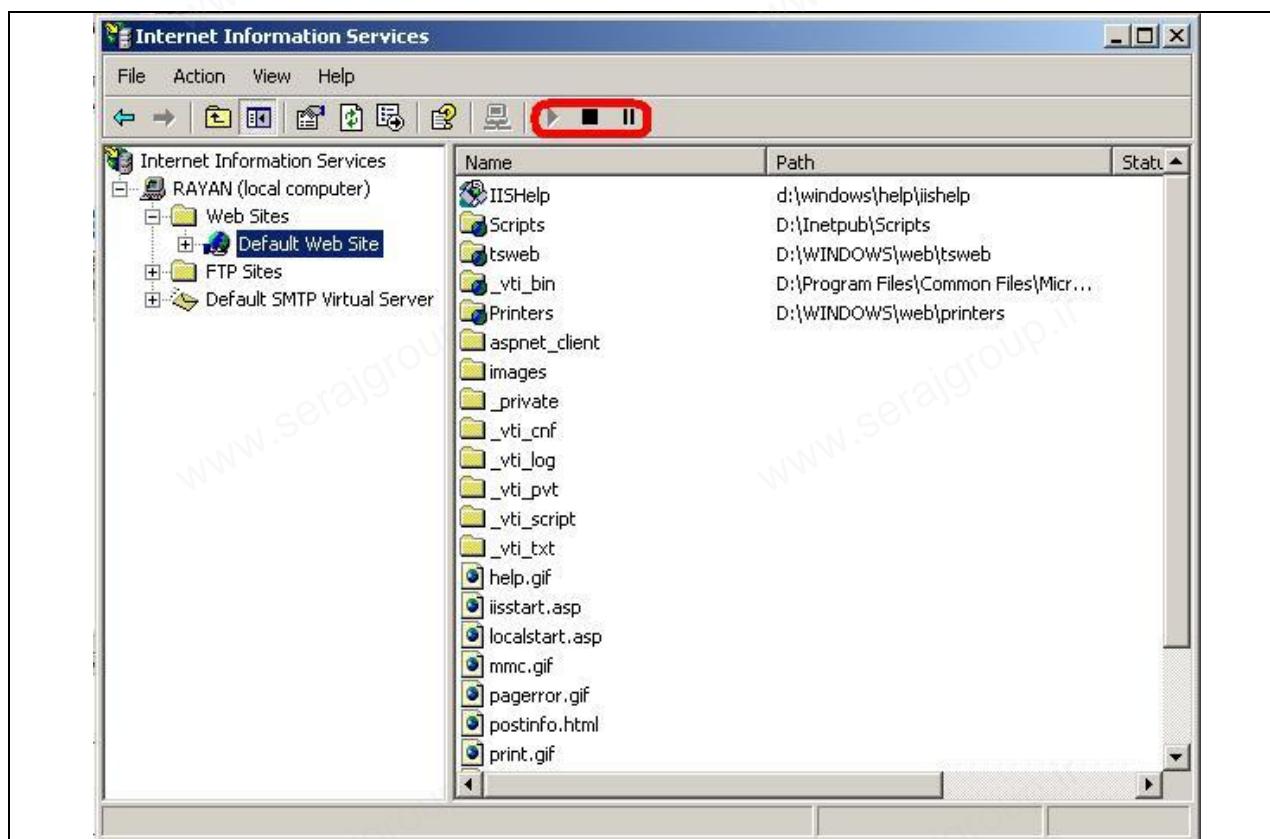
شکل 18: انتخاب زبانه Documents سپس اضافه کردن صفحه پیش فرض

شکل 17: انتخاب گزینه Internet Information Services در Administrative tools Service Properties

برای تنظیم صفحات پیش فرض Internet Service Manager را اجرا کنید. پنجره خواص را Site Selection کنید و Tab به نام Document را انتخاب کنید. Enable Default Document را فعال کرده و نام های پیش فرض را اصلاح کنید. گزینه دیگری که در Tab مربوط به Document در صفحه خواص Default Web Site وجود دارد، Document footer است. بوسیله اینکار می توان به تمام اسناد روی سایت خودتان یک پاورقی اضافه کنید. فرمت آن هم باید مانند یک صفحه HTML باشد بدون داشتن تگ های <Title></Title><Body></Body> و <Bold> باشد مانند <Bold> copyright 2006 </Bold>

د) مکث، متوقف و شروع مجدد یک وب سایت

گاهی از اوقات لازم است برای انجام عملیاتی مانند نگهداری، تهیه پشتیبان و یا ویروس یابی ، سایت را متوقف کرد. متوقف کردن وب سایت در حال دسترسی یک وقفه آنی در سرویس برای همه کاربران ایجاد خواهد کرد. در این حالت سایت را متوقف نمی کند اما از فعالیت های جدید جلوگیری می کند(مکث کردن). برای وب سایت های بسیار پرکار و پر مشغله، مدیر سایت اول این کار را انجام دهد و سپس سایت را متوقف کند. بعضی وب سایتها کار پیکر بندی را موقعی که سایت در حال اجرا است می توانند انجام دهند. ولی موثر نمی باشد تا موقعی که سایت دوباره راه اندازی مجدد نشود. می توان سایت را متوقف سپس آن را اجرا کرد. با استفاده از برنامه های Asp.Net دلایل زیادی برای متوقف کردن یا مکث کردن یک وب سرور وجود ندارد.



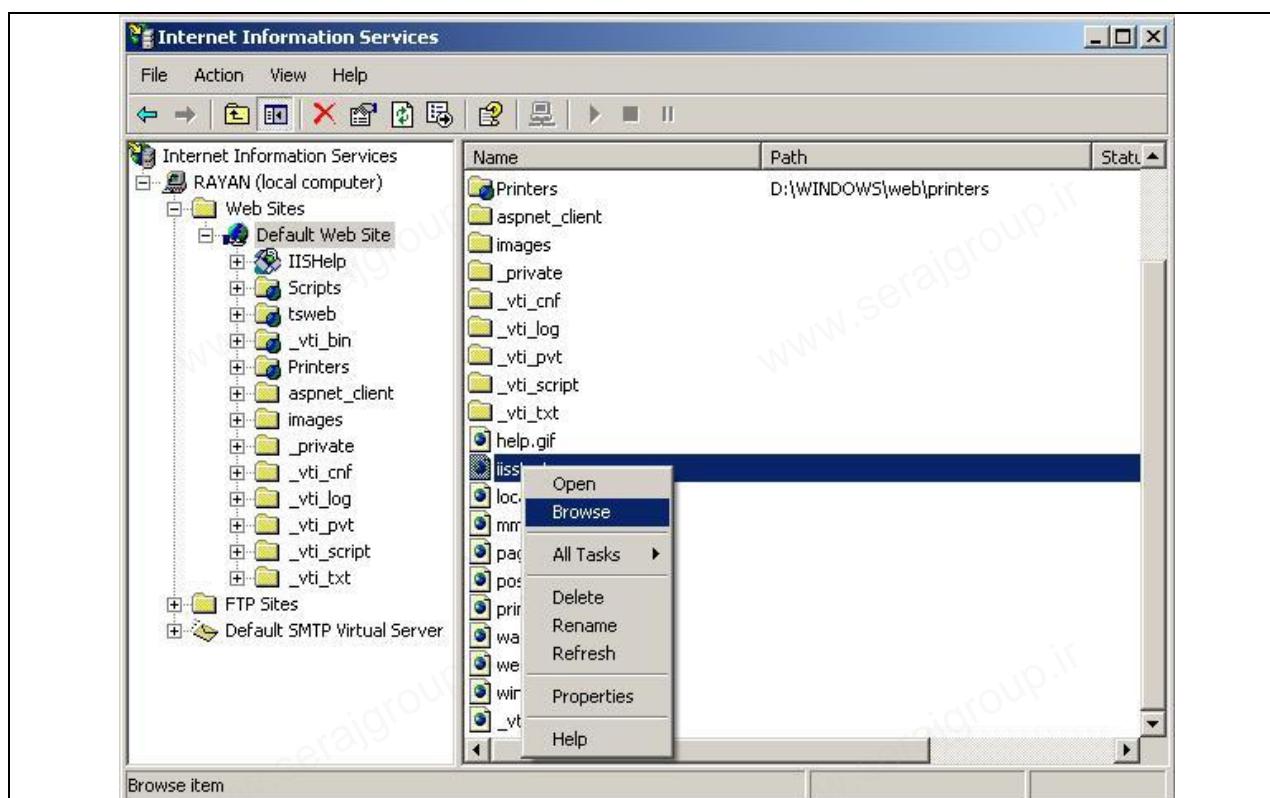
شکل 19 : پنجره Internet Information Service و ستفاده کردن از دکمه های Run، Stop و Pause

از طریق خط فرمان هم می توان کار مکث، توقف و شروع مجدد وب سایت را انجام داد. لیست آن به شرح زیر است:

وب سرور را متوقف و سپس راه اندازی می کند.	iisreset/restart
وب سرور را راه اندازی می کند.	iisreset/start
وب سرور را متوقف می کند.	iisreset/stop
کامپیوتر را ریبوت می کند.	iisreset/reboot
در صورت بروز خطا هر یک از مراحل متوقف سازی یا راه اندازی مجدد وب سرور، کامپیوتر را ریست می کند.	Iisreset/rebootonerror
وب سرور را متوقف و سپس راه اندازی می کند.	Iisreset/status
راهنمای این دستور را نمایش می دهد.	Iisreset/?

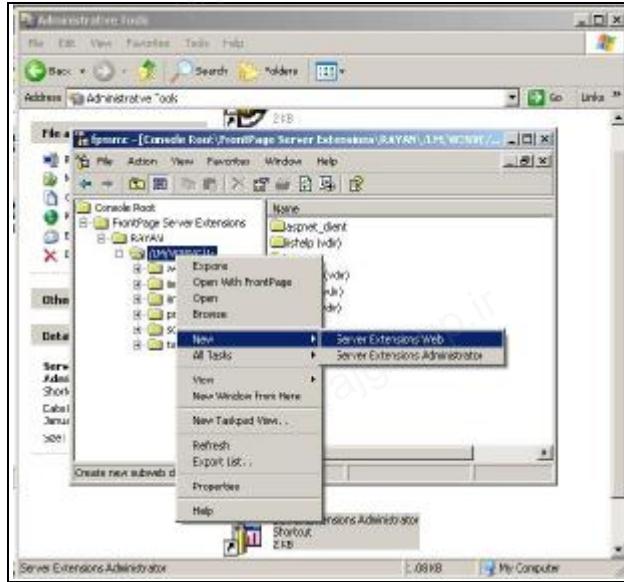
جستجوی وب سایت پیش فرض

شما می توانید از برنامه کاربردی Internet Service Manager(ISM) برای باز کردن صفحات روی وب سایتان در مرورگر استفاده کنید. شما می توانید این وسیله را مستقیماً از منوی Start یا از کنسول کنترل پانل استفاده کرد. این وسیله مدیریتی در پیکربندی کردن سایت شما و همچنین گردش در آن کمک خواهد کرد. همچنین در باز کردن وب سایتان به صورت محلی می توان از آن استفاده کرد. همچنین می توان از ISM برای باز کردن یکی از صفحه های وب سایت بطور جداگانه استفاده کرد.

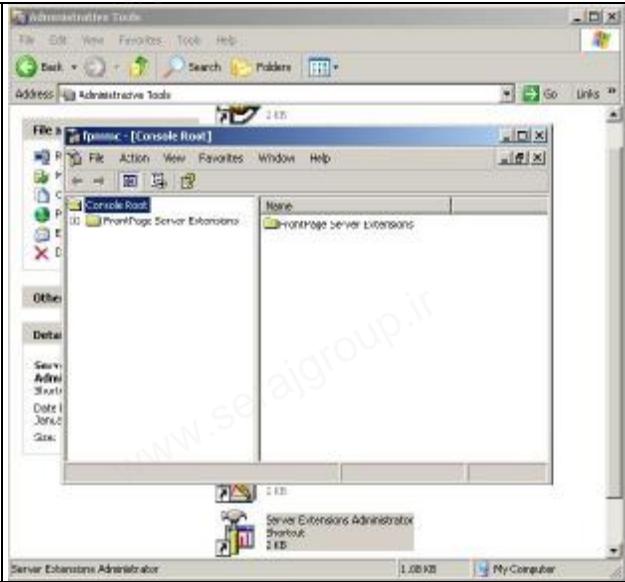


ایجاد Sub Web

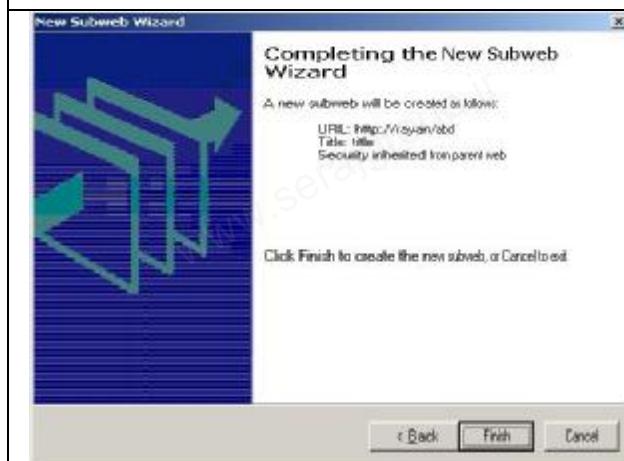
ساب وب یک دایرکتوری مجازی است که حاوی وب سایت شما است. در این حالت با اضافه کردن server Extensions به دایرکتوری مجازی، ویژوال استودیو دات نت را قادر می سازد تا بتواند یک برنامه را در این دایرکتوری ایجاد و نگهداری نماید.



شکل 22: انتخاب گزینه Server Extensions Web



شکل 21: انتخاب گزینه Server Extensions Administrator tools در Administrator tools



شکل 24: زدن دکمه Finish و ایجاد Sub Web



شکل 23: وارد کردن نام دایرکتوری مجازی و توضیحات در Title

برای ایجاد ساب وب، گزینه Server Extensions Administrator Tools را انتخاب

نموده سپس بر روی قسمت درختی کلیک راست نمایید و از منوی ظاهر شده New و سپس Server Extensions Web را انتخاب نمایید. یک صفحه ویزارد باز می شود. نام دایرکتوری را اینجا همان نام دایرکتوری مجازی که در قبل ایجاد کرده اید وارد

نمایید. توضیح مختصری را هم می توانید در قسمت **Title** وارد کنید. روی **Next** کلیک کنید در صفحه بعدی، گزینه پیش فرض را قبول کرده روی **Next** کلیک کنید و تمام! پس از انجام اینکار ، این فولدر در ویژوال استودیو قابل دسترسی می شود.

را تنها می توان روی **Root Web** و یا فولدری داخل آن ایجاد کرد.

فصل سوم

آشنایی با زبان

C#

در این فصل با اصول پایه ای C# و برنامه نویسی آن برای ایجاد صفحات ASP.NET آشنا می شویم اگر با زبان C آشنایی دارید فصل جاری فصلی ساده و بسیار روانی برای شما خواهد بود و در غیر این صورت با کمی پشتکار مشکل حل خواهد شد این مرور بسیار کاربردی و خیلی سریع کد نوشتن را شروع خواهیم کرد بد یهی که فقط برای آشنایی کامل با اساس و شالوده‌ی زبان C# به کتابی کامل نیاز می باشد و نه یک فصل چند صفحه‌ای.

C# ویژگیهای زبان

- 1- زبان C# یک زبان میانی می باشد.
- 2- قابل انعطاف و بسیار قدرتمند می باشد.
- 3- زبان برنامه نویسی سیستم است.
- 4- یک زبان قابل حمل می باشد.
- 5- زبان کوچکی می باشد.
- 6- کاملاً شی گرا می باشد.
- 7- نسبت به حروف حساس می باشد.
- 8- دارای کلاسها و فضاهای نام بسیار زیادی می باشد.

(Namespace) نام فضای

فضایی نام یک مفهومی است که از C++ گرفته شده است و طرح نام گذاری منطقی از گروهی از نوع های به هم مرتبط و همچنین روشی برای مدیریت نام کلاسها و متدها هستند آنها ایجاد شده اند تا تداخلی بین نام های توابع در برنامه شما رخ ندهد این مساله در پروژه های بزرگ خود را نشان می دهد و ممکن است دو آیتم در یک پروژه نام های یکسانی را پیدا کنند بدین وسیله این شанс تصادم و تداخل کاهش پیدا می کند برای ایجاد یک فضای نام به صورت زیر عمل می شود.

namespace mynamespace

{

.....

.....

Class myclass1

{

.....

.....

.....

}

.....

.....

}

فضای نام System قضای نام ریشه برای تمام .NET Framework می باشد. بنابراین شامل تمام کلاس‌های پایه و عمومی برای استفاده شما در ASP.NET می باشد. برای استفاده از آن می توان از کد زیر کمک گرفت:

using System;

فضای نام System شامل نوع داده های ساده مانند int,uint,sbyte,byte,short,ushort,long,ulong,float, double,decimal,string,char,bool که همه اینها از نوع Value type هستند و همچنین شامل نوع داده های می باشد، در C# تمام نمونه های کلاسها از نوع Reference می باشند. که به نوبه خود همه کلاسها از کلاس Object مشتق شده اند.

تمام فضاهای نام به صورت پیش فرض public می باشد و در خارج از کد شما قابل دسترسی هستند روشن استفاده از انها به صورت زیر است:

```
projectname.Namespace.classname.membername;
```

برای مثال اگر کلاس آرایه ای را در دات نت بخواهیم مرتب کنیم حداقل دو راه برای نوشتن وجود دارد:
system.array.sort(strarray);

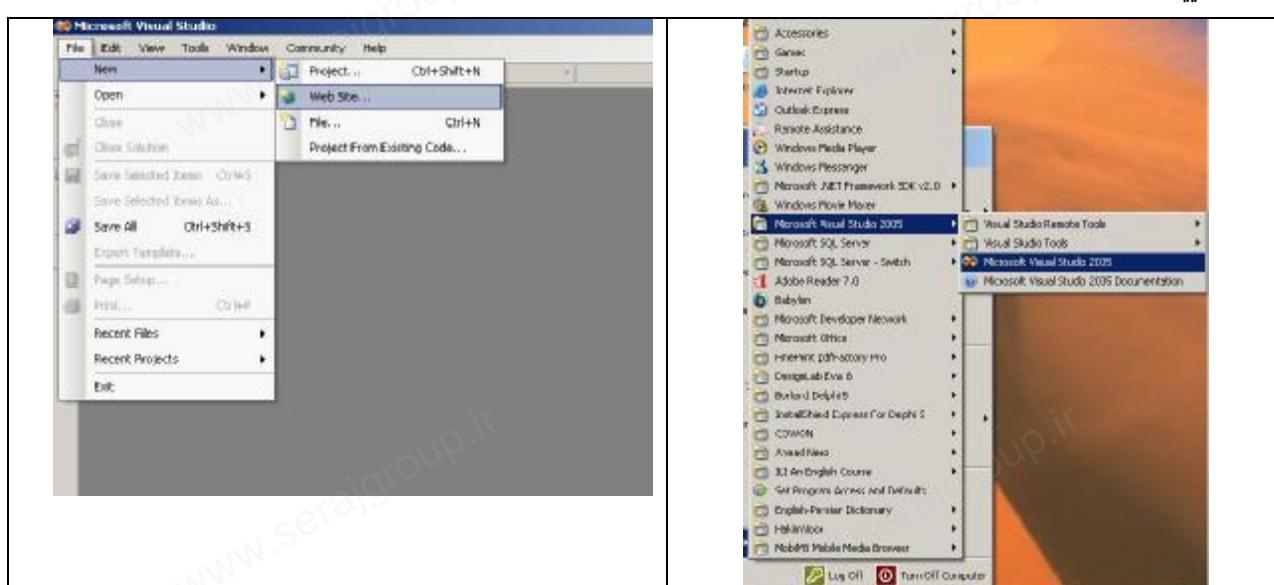
و یا

```
using system;  
array.sort(strarray);
```

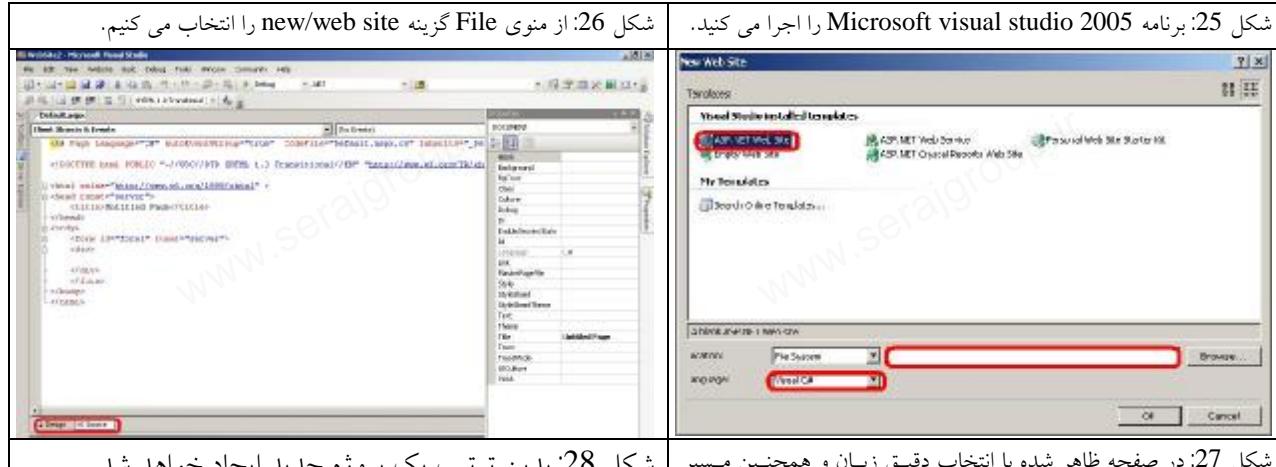
بدین صورت خلاصه نویسی در کد صورت می گیرد.

مثال اول: مروری بر نحوه استفاده از namespace ها، تعریف متغیر و مقدار دهی اولیه به آن، توابع و خواص ها.

قبل از شروع مثال این نکته را باید مذکور شد که تمام دستورات کنترلی در C# تقریباً مشابه دستورات C و C++ می باشد و اگر در دستورات کنترلی تغییرات داده شده باشد در جای خود مذکور خواهیم شد. برای ایجاد یک پروژه جدید به شکل ها نگاه نمایید.

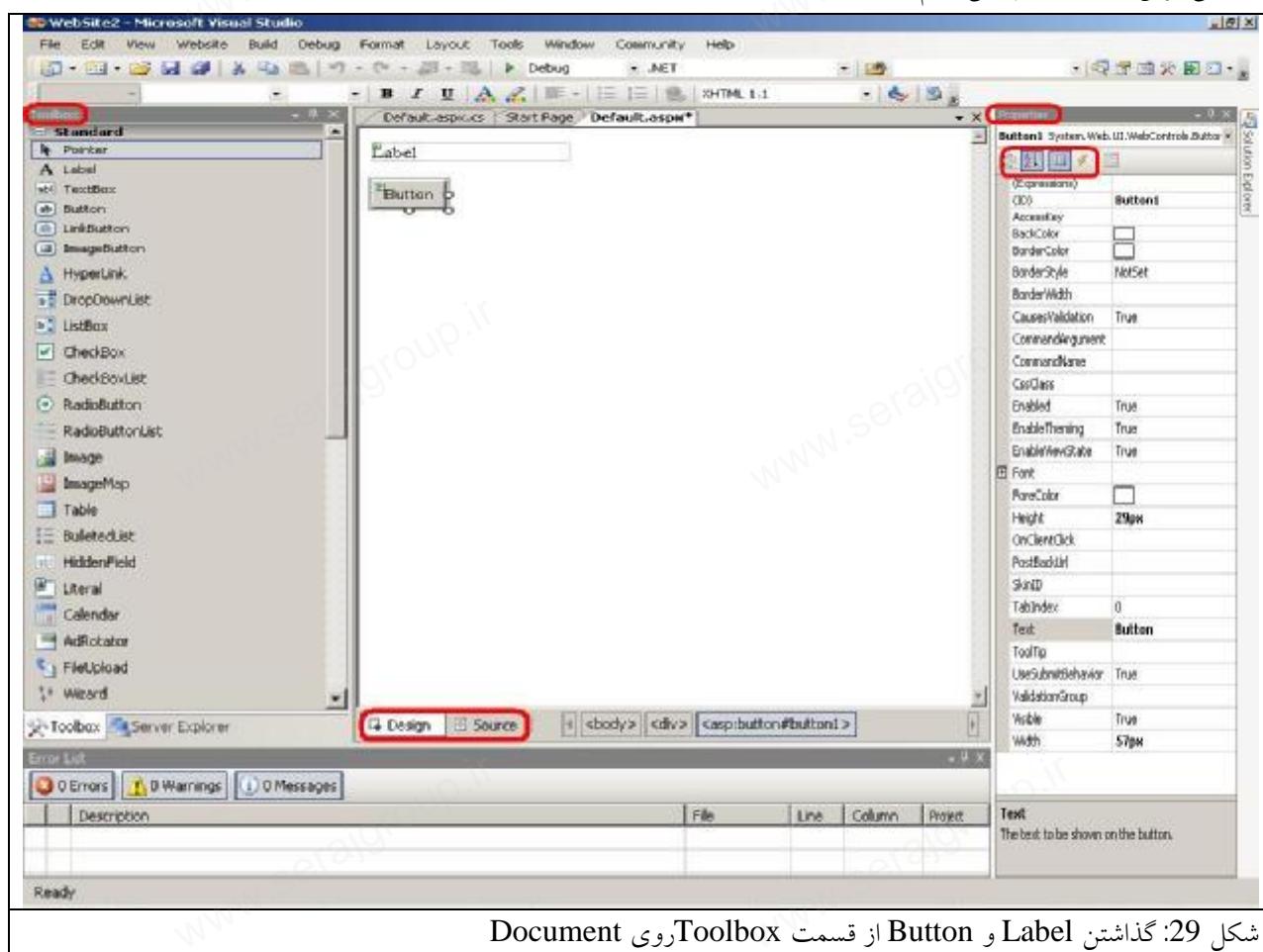


شکل 25: برنامه Microsoft visual studio 2005 را اجرا می کنید.



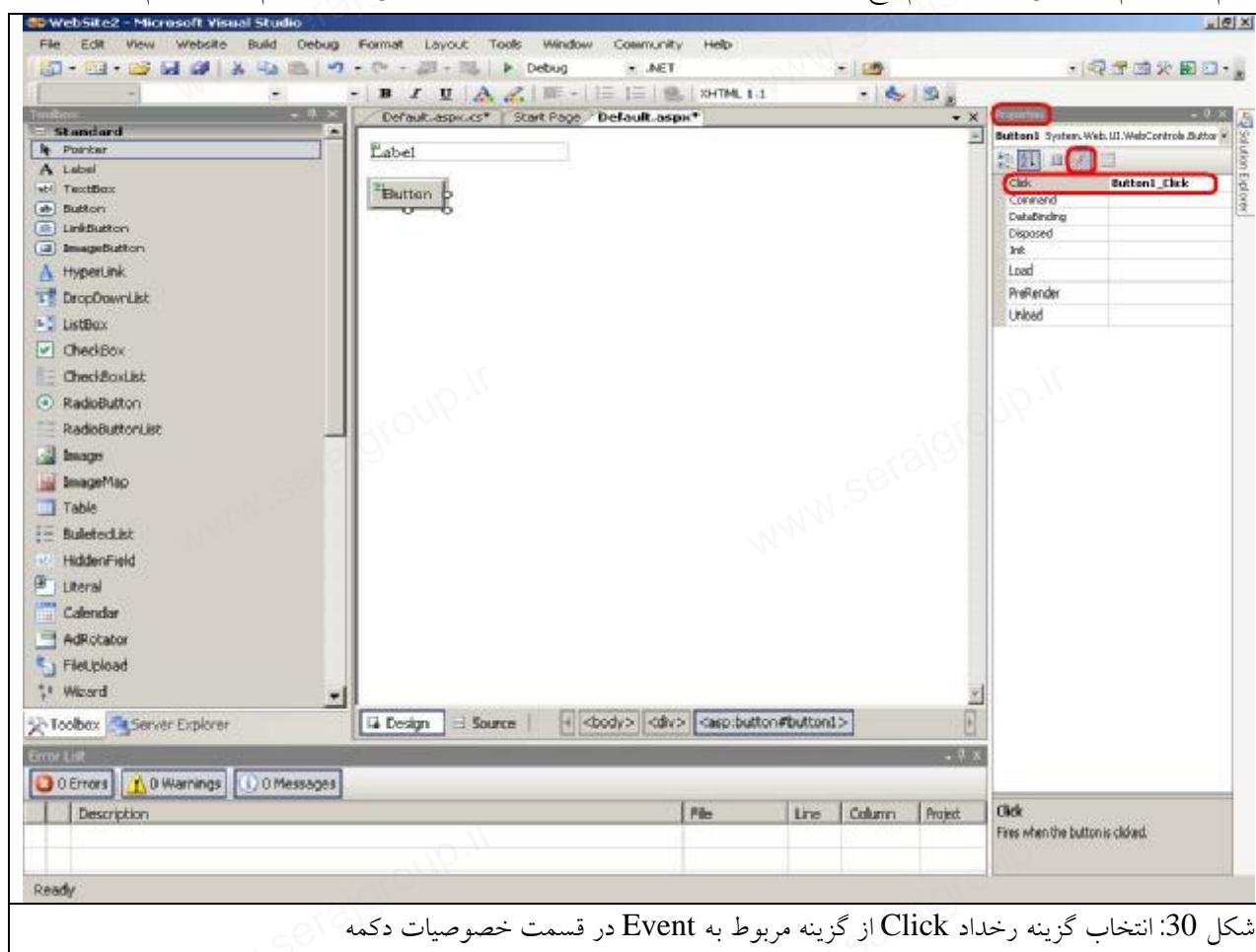
شکل 26: از منوی File گزینه new/web site را انتخاب می کنیم.

حال آماده هستیم که یک پروژه ساده که فقط شامل یک برچسب و یک دکمه باشد، ایجاد کنیم بطوریکه همینکه روی دکمه کلیک کردیم یک پیغام مناسب مبتنی بر اینکه این اولین برنامه می باشد ایجاد کنیم، برای مرور کردن یک سری از اصول ما طولانی ترین راه را انتخاب می کنیم.



شکل 29: گذاشتن Document Label و Button از قسمت Toolbox روی Document

از Toolbox کنار صفحه یک Label و یک دکمه (Button) را روی فرم قرار دهید. حالا روی دکمه دوبار کلیک کنید یا اینکه از قسمت Properties مربوط به دکمه گزینه Events را انتخاب کرده و Event مربوط به کلیک کردن را انتخاب می کنیم تا بتوانیم در تابعی که در هنگام رخداد کلیک شدن بر روی دکمه صدای زده می شود بتوانیم کد بنویسیم.

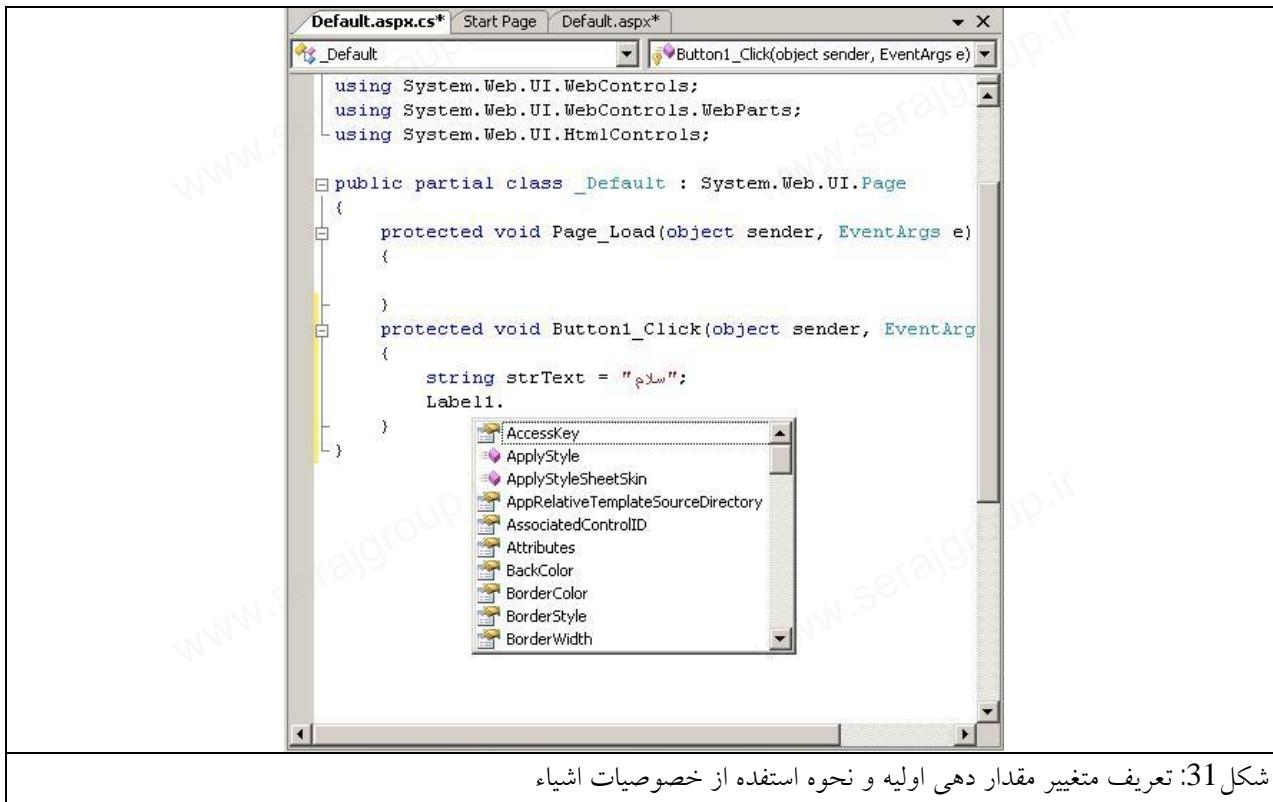


شکل 30: انتخاب گزینه رخداد Click از گزینه مربوط به Event در قسمت خصوصیات دکمه

اگر به صفحه باز شده که به آن Code behind هم می گویند دقت کنید به صورت پیش فرض یک سری از فضاهای نام مفید و لازم در این Source گنجانده شده است. می خواهیم هر بار کاربر روی این دکمه کلیک کرد به او جمله "سلام! این اولین برنامه‌ی من است!" را نشان دهد.

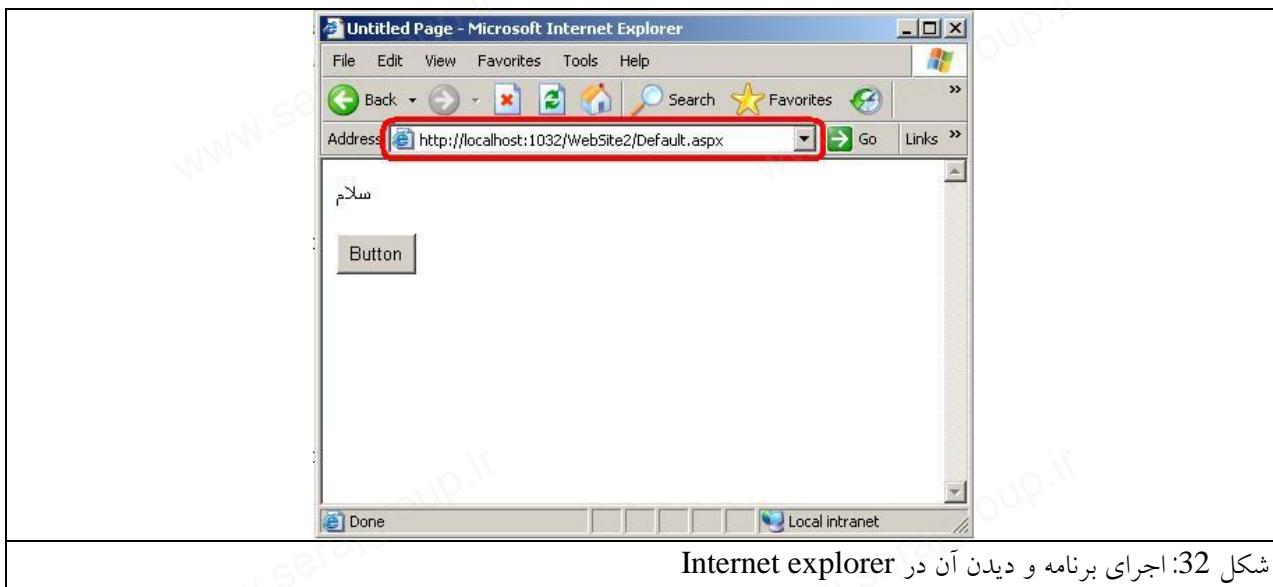
- یک متغیر از نوع String به نام strText تعریف کنید. بهتر است نوع متغیر به صورت خلاصه در ابتدای نام متغیر ذکر شود.
- آن را مقدار دهی اولیه کنید (برای مثال "سلام" و یا جمله بالا).
- به راحتی می توان داخل آن فارسی نوشت. در C# تا متغیر را مقدار دهی اولیه نکنید نمی توان از آن استفاده کرد.
- می خواهیم به خاصیت Text مربوط به Label که روی فرم گذاشته ایم این متغیر را نسبت دهیم. نام Label یعنی Label را بنویسید به همراه یک نقطه در جلوی آن یک منو که تمام توانایی های این کنترل را نمایش می دهد باز

خواهد شد. گزینه **Text** آن را انتخاب کنید و متغیر فوق را به آن نسبت دهید (گر با کامپایلر های ویژوال کار کرده باشید ملاحظه می کنید که همه چیز مانند آنها می باشد).



شکل 31: تعریف متغیر مقدار دهی اولیه و نحوه استفاده از خصوصیات اشیاء

- حالا بر روی دکمه **Run(F5)** کلیک کنید تا برنامه در مرورگر وب اجرا شود . با کلیک کردن بر روی دکمه، سلام ، نمایش داده می شود. به آدرسی که در **Internet explorer** نوشته می شود نیز دقت کنید.



شکل 32: اجرای برنامه و دیدن آن در **Internet explorer**

با توجه به اینکه در این مثال از **Font** فارسی استفاده شده است، باید برای استفاده آن به نکات زیر توجه شود.
از منوی فایل گزینه **Save as** را انتخاب کنید. روی دکمه **Save** یک علامت مثلث قرار دارد. روی آن کلیک نموده تا یک منوی جدید باز شود. حالا روی گزینه **Save With Encoding** کلیک نمایید و صفحه‌ی اخطار باز شده را تأیید کنید و از صفحه‌ی ظاهر شده‌ی بعدی از آیتم **Encoding** گزینه **Unicode(UTF-8 with signature)** را برگزینید.



از این پس با خیال راحت و بدون هیچ نگرانی در مورد به هم ریختن فرمت فارسی برنامه می‌توانید برنامه‌ها را اجرا نمایید.

مروری بر مفاهیم بکار گرفته شده در کد ارائه شده:

فرمت کدن کد:

هر چقدر فرمات نوشتمن کد شما بهتر باشد، خواندن، نگهداری و استفاده مجدد از آن ساده‌تر خواهد بود. دو مورد مهم در برنامه نویسی دندانه دار نویسی و نوشتمن توضیحات و یا کامنت‌ها می‌باشد. نوشتمن توضیحات خصوصاً در برنامه نویسی تیمی بسیار مهم و کار ساز است. در C# از // برای نوشتمن کامنت استفاده می‌شود (مانند +C++) و همانند C هنوز /*....*/ نیز معتبر است.

نکته :

در ویژوال استودیو یک روش نوشتمن کامنت به آن اختلافه شده است که بدین صورت می‌باشد که بهتر است (!) قبل از هر تابع یا خاصیت یا کلاس و ... نوشه شود:

```
///<summary>
///
///
///</summary>
```

خاصیت این نوع نوشتمن کامنت این است که هر موقع روی کلاس یا تابع یا خاصیت مورد استفاده شده که این نوع کامنت برای آن نوشته شده برویم این توضیحات برای آن نمایش داده می‌شود.



تعريف متغير و مقدار دهی به آن :

در هنگام تعريف يك متغير، ناچيه اي از حافظه برای ذخیره سازی داده، اختصاص داده می شود. در #C# بخلاف بعضی از زبان ها که نیازی به تعريف صريح متغير ها ندارند، هم باید نوع متغير را تعريف کنید و هم آنرا مقدار دهی اولیه نمایيد. البته اگر فراموش کردید که متغيری را مقدار دهی اولیه کنید مهم نیست! کامپایلر حتماً آنرا به شما با يك خطأ گوشتزد خواهد کرد!

مقدار دهی اولیه يك متغير از بسياري از خطاهای زمان اجرا مانند جمع زدن دو متغير بدون مقدار جلوگيري خواهد کرد.

استفاده از خواص (property) و اندیکس ساز (Indexer)

دو ويزگي جالب و مهم در #C# خواص و اندیکس ساز می باشد. خواصها در فراخوانی کردن متدهای يك کلاس شما را کمک خواهد کرد و اندیکس سازها در دسترسی به کلکسیون کلاسها که ترکیب Array استفاده شده ما را کمک خواهند کرد. شما به ويزگي های يك شیء با استفاده از خواص آن می توانید دسترسی پیدا کنید. يك Property عضوی است که امكان دسترسی به ويزگي شیء يا کلاس را فراهم می کند. برای مثال طول يك رشته، سایز يك فونت، عنوان يك فرم و نام يك مصرف کننده، خاصیت هستند.

بسیاری از اشیاء ذاتی دات نت فریم ورک، خواص مفید زیادی را به همراه دارند. برای مثال شیء DateTime را نظر بگیرید. با استفاده از خاصیت Today آن می توان تاریخ جاری سیستم را بدست آورد. برای استفاده از يك خاصیت لازم است تا کلاس تعریف کننده شیء در برنامه مهیا باشد. منظور همان استفاده از فضای نام مربوطه می باشد. پس از وارد کردن فضای نام کلاس مورد نظر می توانید از شیء و خواص آن استفاده کنید. همانطور که ذکر شد يا به صورت کامل تمام موارد باید ذکر شوند مانند; System.DateTime.Now و يا وارد کردن فضای نام System كوتاه سازی صورت می گیرد که پیشتر نیز ذکر گردید.

C# : مروری بر آرایه ها و حلقه ها در

در این برنامه می خواهیم آرایه ای از کاراکتر ها را به مقادیر متناظر یونیکد آنها تبدیل و سپس مرتب شده آنها را نمایش دهیم.

هنگامی آرایه ها را ایجاد می شوند که بخواهیم با مجموعه ای از اطلاعات همجنس کار کنیم، برای نمونه در این مثال از آرایه برای ذخیره تعدادی کاراکتر می خواهیم استفاده نماییم. آرایه ها يك نوع متغير هستند پس باید تعريف و مقدار دهی اولیه شوند، نوع و تعداد اعضای آنها باید معین گردد. حد پایین آرایه صفر بوده برای مثال اگر آرایه [chrData][0] ده عضو داشته باشد، اولین عضوان chrData[0] و آخرین عضو آن chrData[9] است. برای تعریف آرایه چندین راه مختلف وجود دارد.

1- تعريف آرایه ای از رشته ها و مقدار دهی اولیه آن.

```
String[] strData = new string[2];
```

2- تعريف و مقدار دهی اولیه

```
string[] strData={“1234”, “abcd”};
```

که آرایه ای از نوع رشته ای به طول 2 عضو با مقدار دهی اولیه ایجاد شده است. در این حالت نیازی به تعیین طول آن نمی باشد.

3- روشي ديگر برای مقدار دهی اولیه

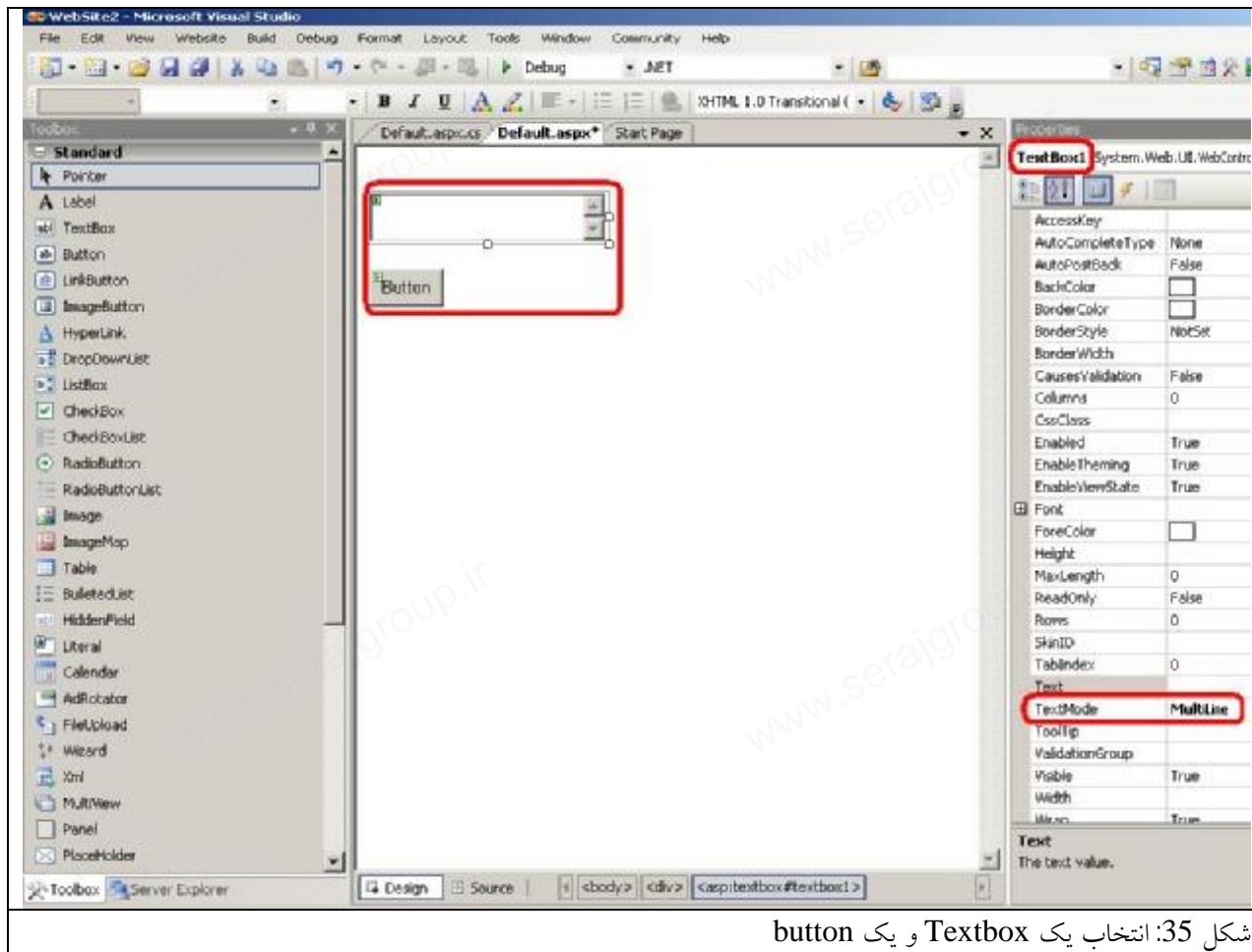
```
strData[0] = "1234";  
strData[1] = "abcd";
```

در دات نت کلاسی به نام **Array** وجود دارد که امکانات جالبی را برای کار با آرایه ها ارائه می دهد. برای مثال تابع **Sort** آن به سادگی یک آرایه را مرتب می کند. برای حرکت بین یک اعضای یک آرایه با تعداد بالا به سادگی می توان از حلقه **for** و یا **foreach** استفاده کرد.

برای مثال

```
for(int i=0; i<strData.Length;i++)  
    Statements;
```

در هنگام کار با آرایه ها حتماً لازم است طول چک شود تا مشکل عدم دسترسی به عضوی که تعریف نشده پیش نماید (عضوی که در کران آرایه قرار ندارد).



شکل 35: انتخاب یک **TextBox** و یک **button**

برای نوشتن برنامه دوم یک آرایه با اعضایی دلخواه به طول 10 تعریف کرده و سپس با استفاده از یک حلقه اعضای آنرا تک تک به مقادیر یونیکد معادل تبدیل نموده و در یک آرایه دیگر ذخیره کرده ایم. برای تبدیل به یونیکد از کد زیر استفاده شده است:

:**(casting)**

```

protected void Button1_Click(object sender, EventArgs e)
{
    // define and initialize the array
    char[] chrArray = {
        'ا', 'ب', 'ج', 'د',
        'پ', 'غ', 'ق', 'ض',
        'ه', 'و', 'ز', 'ش',
        'ک', 'ل', 'ر', 'م'
    };
}

int[] chrTempArray = new int[10]; // define and initialize the array

for (int i = 0; i < chrArray.Length; i++)
    chrTempArray[i] = (int)chrArray[i]; // converting to unicode

Array.Sort(chrTempArray);

foreach (int j in chrTempArray)
    TextBox1.Text += j.ToString() + "\n"; // "\n" means new line
}

```

شکل 36: تعریف دو آرایه، مقداری گذاری اولیه آنها، انتساب و مرتب کردن آنها.

سپس با استفاده از کلاس Array آنرا سورت کرده و سپس خروجی آنرا در یک TextBox نمایش داده شده است. برای اینکه تکست باکس از حالت یک خطی بیرون بیاید و چند خطی شود خاصیت MultiLine TextBox آنرا به تغییر داده ایم.

مروری بر مفاهیم بکار گرفته شده در کد ارائه شده

حلقه ها و پرس در C#

دستورات تکرار و پرس در زبان C# همانند زبان C++ می باشد بدین معنی که C# همانند زبان C++ دارای دستورات for, while, do while, break, continue, return, goto می باشد اما C# داری یک دستور تکرار اضافی بنام foreach می باشد که برای استفاده کردن برای آرایه ها یا در کل برای اعضای شمارشی یک مجموعه بکار بردہ می شود. ساختار آن بدین صورت می باشد.

(یک مجموعه in تعریف یک متغیر از نوع اعضای مجموعه) foreach

بدین معنی می باشد که به برای هر عضو از مجموعه یک بار حلقه تکرار گردد.

آشنایی بیشتر با کلاس ها، متدها

کلاس، مجموعه ای از متغیرها(خصوصیت) و توابعی(متده) است که بر روی این خصوصیات عمل می کنند. نمونه ای از کلاسها را شی گویند. به بیان دیگر متدها یا همان تابع در زبان C، اعضای یک شیء یا کلاس هستند و مجموعه ای از یک سری از کارها را انجام می دهند. با خواص هم که در قسمت های قبل آشنا شدید. بسیاری از کلاسهای دات نت فریم ورک متدها

و یا توابع مفید حاضر و آمده ای را دارند. برای مثال کلاس `DateTime` ، متدهی به نام `ToLongDateString` دارد که تاریخ

را به صورت یک رشته طولانی بر می گرداند. برای تعریف یک کلاس همانطور که گفته شد به صورت زیر عمل می شود:

```
[public | protected | internal | protected internal | private | abstract | sealed ] class className  
{  
}
```

نگاهی گذرا بر کلمات کلیدی این دستور می اندازیم:

کلمه کلیدی	توضیحات
Public	کلاس بصورت عمومی در دسترس قرار دارد.
protected	کلاس فقط توسط کلاسی که شامل این کلاس باشد یا نوعی که از کلاسی که شامل این کلاس است مشتق شده باشد قابل دسترسی می باشد.
internal	کلاس فقط توسط این برنامه قابل دسترسی می باشد.
protected internal	کلاس فقط توسط این برنامه قابل دسترسی است یا نوعی که از کلاسی که شامل این کلاس است مشتق شده باشد.
Private	کلاس فقط در داخل کلاسی که شامل این کلاس می باشد قابل دسترسی می باشد.
abstract	اعضای این نوع کلاس باشد توسط کلاسهای وارث پیاده سازی شود.
Sealed	وراثت از این نوع بعداً امکان پذیر نمی باشد.

برای تعریف یک متدها تابع ابتدا سطح دسترسی به آن وسپس نوع خروجی تابع ذکر می گردد که داخل این پرانتزها می توان ورودی ها تابع یا بقولی آرگومان های ورودی را معرفی کرد. سپس تابع باید با { شروع و با یک } خاتمه یابد، بدین صورت:

```
[ public | protected | internal | protected internal | private | static | virtual | override | abstract |  
extern ]
```

```
[ type | void ] memberName([parameters])
```

```
{
```

```
}
```

هر تابعی می تواند صفر تا تعداد بیشماری آرگومان ورودی و صفر تا تعداد بیشماری خروجی داشته باشد. بوسیله یک تابع می توان پیچیدگی کار را مخفی کرد و صرفاً با صدا زدن نام آن، یک سری از عملیات را انجام داد. گاهی از اوقات لازم می شود دو یا چند تابع با یک نام داشته باشیم بطوریکه پارامترهای ورودی یا مقادیر خروجی و یا نوع آرگومان های ورودی آنها با هم متفاوت باشد به این کار **Overloading** می گویند.

خصوصیت در C# یک متغیر عمومی در کلاس می باشد یا یک دسترسی عمومی می باشد. در مثال زیر دو نوع پیاده

سازی خصوصیت آمده است که روش دوم معمولاً مورد استفاده قرار می گیرد.

پیاده سازی نوع اول:

```
public class calculator
{
    public double Op1;
    public double Op2;
    public double Add()
    {
        return Op1 + Op2;
    }
}
```

پیاده سازی نوع دوم:

```
public class calculator
{
    private double _op1;
    private double _op2;
    public double Operand1
    {
        get
        {
            return _op1;
        }
        set
        {
            _op1 = value;
        }
    }
    public double Operand2
    {
        get
        {
            return _op2;
        }
        set
        {
            _op2 = value;
        }
    }
}
```

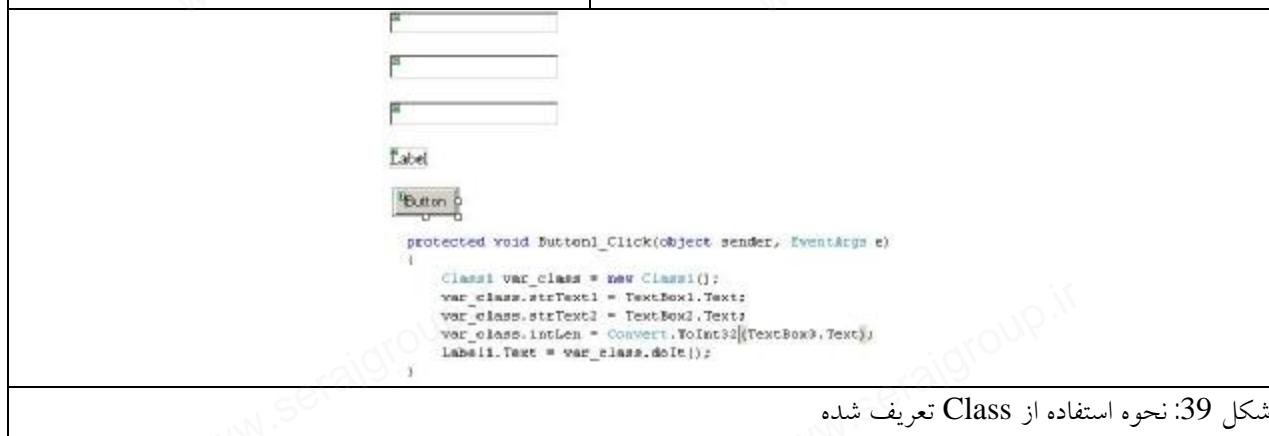
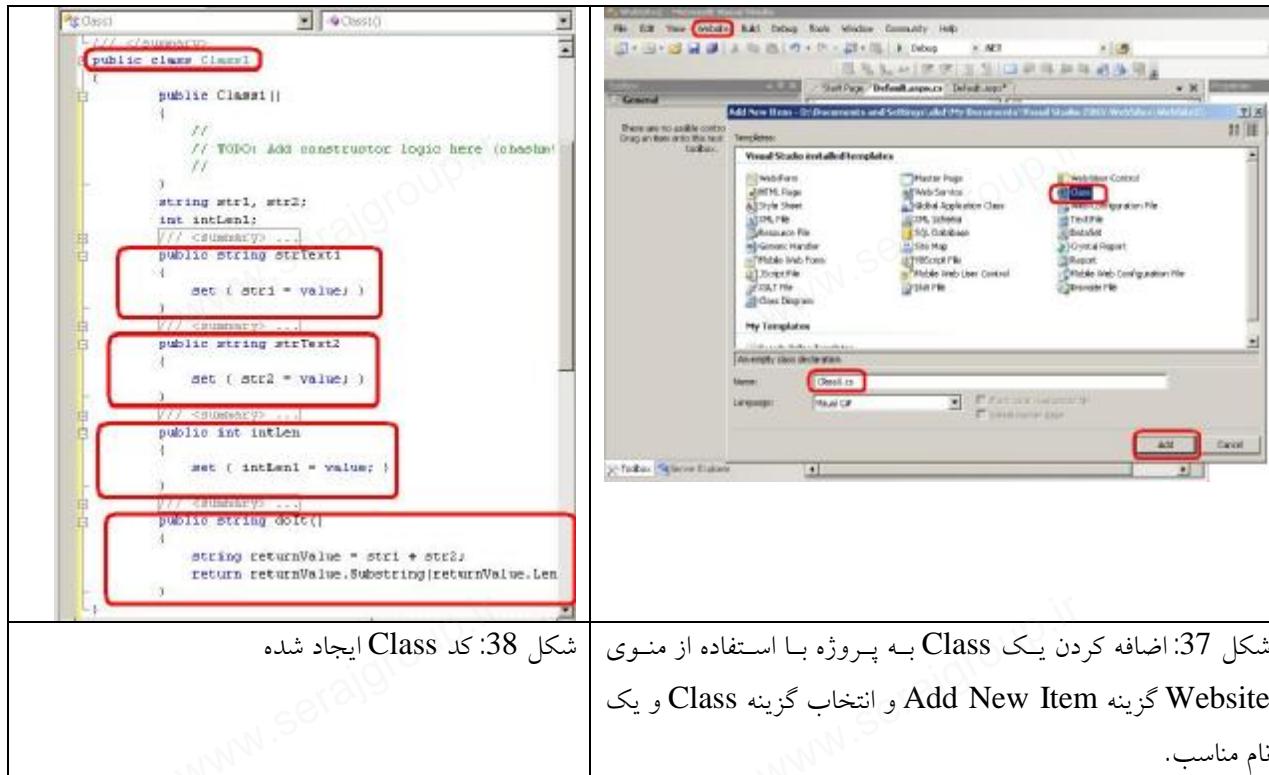
در روش دوم برای خصوصیتی که فقط خواندنی باشد فقط کلمه کلیدی `get` استفاده می شود و برای خصوصیتی که فقط نوشتی باشد فقط از کلمه کلیدی `set` استفاده می شود و وقتی که خواندنی / نوشتی باشد از هر دو کلمه استفاده می شود.

برنامه سوم : تعریف کلاس ، خواص، متدها و مروری بر سطوح دسترسی در کلاس ها

در این برنامه می خواهیم کلاسی را تعریف کنیم که در آن با استفاده از خواص، دو رشته را دریافت و توسط یک متده، این دو رشته به هم متصل گردیده و تعدادی کاراکتر از آن مطابق خاصیتی دیگر که آن طول این رشته جدا شده را از انتهای رشته مشخص می کند، نمایش دهیم.

با توجه به توضیحات ارائه شده در مورد تعریف کلاس ها، توابع و خاصیت، این برنامه ساده می باشد.

فقط برای کار با رشته و پیدا کردن رشته ای از درون رشته ای دیگر، یکی از توابع پر کاربرد `substring` بوده و ساده ترین راه برای تعریف کلاس استفاده از منوی Website قسمت Add New Item و انتخاب Class می باشد که تعاریف اولیه را خود VS.NET انجام می دهد.



لازم به ذکر است که پرکاربردترین سطوح دسترسی به کلاس ها توابع `private` و `public` می باشند.
برای مثال اگر تابعی در کلاس شما یک کار میانی برای ربط دادن دو تابع دیگر را انجام می دهد می توانید آن را `private` تعریف کنید تا هنگام استفاده از کلاس مدیریت کار کردن با تابع گیج کننده نباشد.

بقیه مسائل شی گرای مانند سازنده ها، مخرب ها، نحوه Overload کردن، ارت بری، کپسوله سازی و دیگر مباحث را می توان در کتابهای مرجع یافت. مبحث شیءگرا در `C#` آن قدر مفصل است که می توان یک کتاب 700 صفحه ای راجع به آن نوشت! اگر باور ندارید یک سری به آدرس های زیر بزنید (کتاب `C# Thinking`):

www.thinkigin.net

www.BruceEckel.com

هدف از این فصل مروری سریع بر یک سری از مفاهیم اساسی و پایه ای بودند که در هنگام کار بیشتر با آنها مواجه می شویم. مباحث پیشرفته تر و مفصل تر در این مورد را می توانید در کتابی فوق و یا کتاب های اختصاصی و پایه ای `C#` ملاحظه نمایید.

فصل چهارم

معرفی کنترل

های HTML و نحوه

کاربرد آنها در

صفحات ASP.NET

قبل از معرفی کنترلهای HTML توضیحی درباره صفحات اینترنتی و کد نویسی در ASP.NET خواهیم داشت. یک صفحه اینترنتی به صفحه گفته می شود توسط تگ های HTML در مرورگر اینترنت ایجاد می شود گفته می شود. این تگ ها می توانند استاتیکی یا دینامیکی باشد. استاتیکی بدین معنی که یک فایل HTML توسط کاربر ایجاد شده باشد و توسط مرورگر اینترنت نمایش داده شود. اما دینامیکی بدین معنی که صفحه HTML توسط IIS بر مبنای درخواست کاربر ایجاد و توسط مرورگر اینترنت نمایش داده شود. برای ایجاد HTML استاتیکی معمولاً از تگ ها یا کامپونت های HTML استفاده می شود. اما برای جمع آوری درخواست یا ایجاد صفحات HTML دینامیکی از کنترلهای HTML یا کنترلهای وب استفاده می شود.

کنترل های HTML سرور در حقیقت عناصر استاندارد HTML هستند که در سرور پردازش می شود. تمام کنترل های سرور HTML (که عنوان کنترل های HTML هم شناخته می شوند) دقیقاً معادل یک عنصر HTML تفسیر و اجرا شده و خواص اغلب آنها با عناصر HTML یکسان است.

در طی فصول آتی ما از این کنترل ها به ندرت استفاده خواهیم کرد! کنترل های وب توانای بسیار بیشتری را ارائه می دهند و ادامه‌ی کار تقریباً با آنها تکمیل می گردد. این فصل صرفاً برای یادآوری اسکریپت نویسی کلانتیت ساید با دید VS.NET مفید می باشد و تاثیری آن چنانی بر روی برنامه نویسی سمت سرور ما در فصول آتی نخواهد داشت. این فصل در حقیقت یک نوع DHYML نویسی به سبک VS.NET می باشد.

ASP.NET بر مبنای رخداد عمل می کند بنابراین باید ترتیب رخدادها را بشناسیم. کدها داخل هر رخداد به ترتیب اجرا خواهد شد. ترتیب رخدادها به ترتیب زیر می باشد.

ردیف	رخداد	توضیحات
1	Page_Init	موقع مقدار دهنده اولیه رخ خواهد داد
2	Page_Load	موقعی که صفحه بارگذاری می شود.
3	Control Event	موقعی که یکی از کنترلهای مانند دکمه ها تریگر شود صفحه دوباره بارگذاری خواهد شد.
4	Page_Unload	موقعی که صفحه از حافظه برداشته می شود.

تفاوت بین Page_Load و Page_Init در این است که کنترلها فقط در Page_Load یطور کامل بارگذاری می شود. شما می توانید به کنترلها در هنگام Page_Init دسترسی داشته باشید، اما ViewState بارگذاری نمی شود. بنابراین کنترلها همین مقدار اولیه را خواهند داشت بجای اینکه در هر postback مقدار دهنده مجدد شوند.

پردازش در خواست ها از طرف سرور

یادآور می شود که ما دو نوع اسکریپت نویسی داریم طرف Client و طرف Server. که هر دو نوع اسکریپت نویسی روی کنترلهای HTML انجام می شود. کنترل های HTML در فضای نام SYSTEM.WEB.UI.HTMLCONTROLS تعريف شده اند. شما یک کنترل HTML را در اغلب حالتها با اضافه کردن ویژگی RUNAT="SERVER" به تگ آن، می توانید ایجاد کنید.

با فراموش شدن این مورد نام قابلیت های پردازشی سمت سرور را از دست خواهید داد. بهتر است به هر کنترل یک Id منحصر بفرد اختصاص داده شود تا بتوان به سادگی در برنامه به آن رجوع کرد.

کنترل های سرور HTML از کلاس HTMLInput مشتق شده اند و باید درون کنترل HTMLForm قرار گیرند. با استفاده از کنترل HTMLForm می توان درخواست های رسیده به سرور را پردازش کرد. این کنترل همانند Form معمولی در صفحات HTML است (ازم به ذکر است که در صفحات ASP برای جمع آوری اطلاعات از کاربر مورد توجه قرار می گیرد). بعلاوه اینکه ویژگی RUNAT="server" نیز به آن اضافه می شود. به صورت خودکار وقتی یک پروژه جدید را در ویژوال استودیو باز می کنید اینکار از طرف VS.NET صورت می گیرد و لازم به ذکر است که در هر فایل، شما فقط یک فرم را می توانید تعریف کنید (برخلاف نگارش های قبلی آن).



The screenshot shows the Visual Studio IDE with the Source tab selected. The code editor displays an ASP.NET page (.aspx) with the following structure:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml/DTD/xhtml-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>
            </div>
        </form>
    </body>
</html>
```

A red box highlights the entire `<form>` tag and its content. Another red box highlights the `runat="server"` attribute of the `<form>` tag.

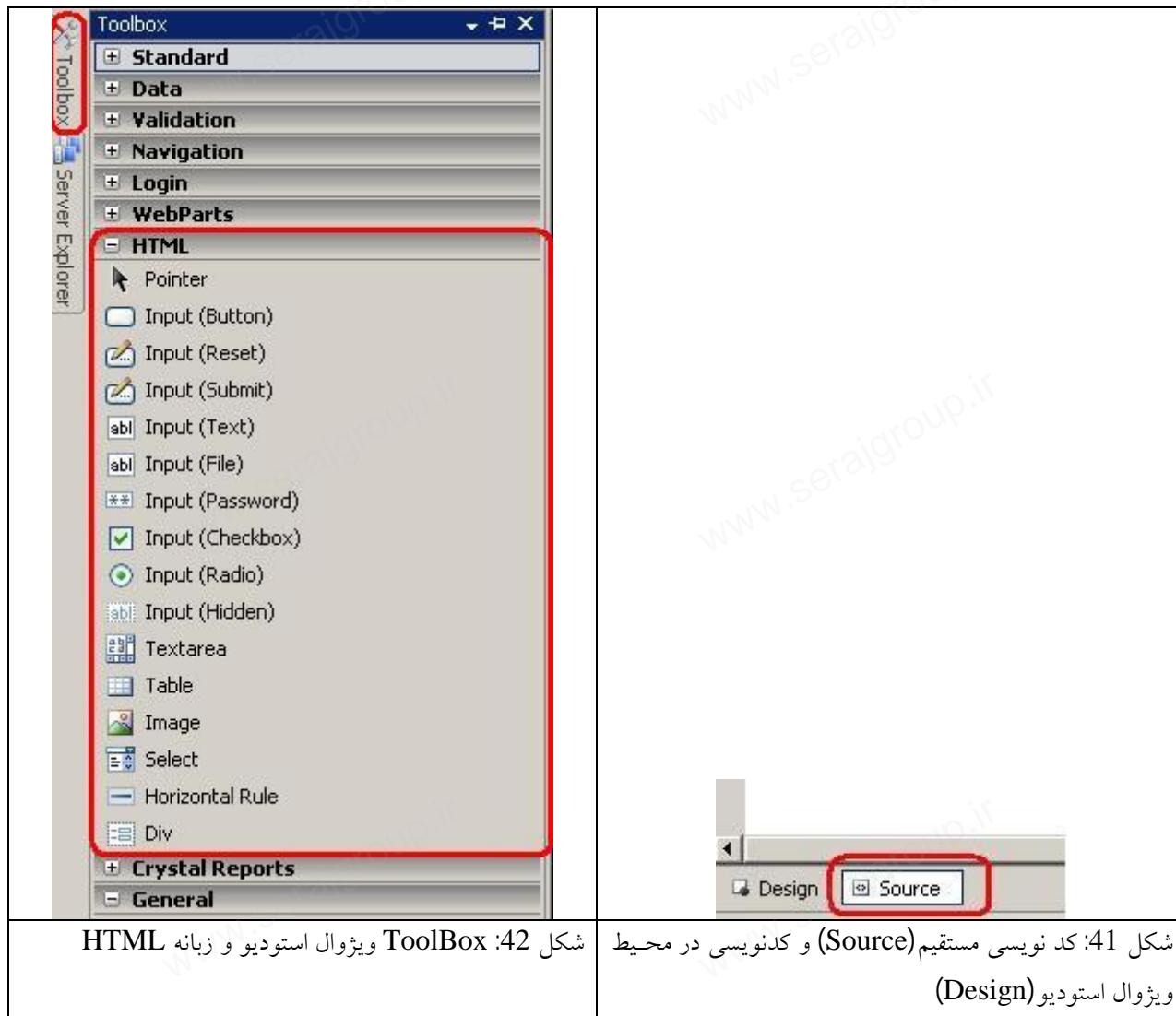
شکل 40: فایل اولیه موقع ایجاد یک صفحه ASP.NET که شامل تعریف زبان و تعریف Header و Body می باشد.

اگر شما به تگ فرم در VS.NET توجه کنید مواردی را ملاحظه می نمایید که مرور شوند. ویژگی Method به مرور می گوید که چگونه اطلاعات را به سرور بفرستد. اگر مساوی GET قرار گیرد، داده ها به صورت یک رشته به URL اضافه شده و فرستاده می شوند و اگر مساوی Post قرار گیرد صورت یک درخواست HTTP فرستاده می شود. چند خاصیت دیگر هم وجود دارند که به صورت پیش فرض در محیط ظاهر نمی شوند.

خصوصیت	توضیحات
Disabled	برای تنظیم وضعیت فرم بکار بردہ می شود. که حالت می تواند فعال یا غیر فعال باشد.
Id	رشته شناسایی فرم را بر می گردد.
name	برای تنظیم و برگرداندن نام فرم بکار بردہ می شود
action	برای مشخص کردن URL که باید داده های فرم را پردازش کند.
length	برای مشخص کردن تعداد عناصر داخل فرم بکار بردہ می شود.

مرواری سریع بر کنترل های

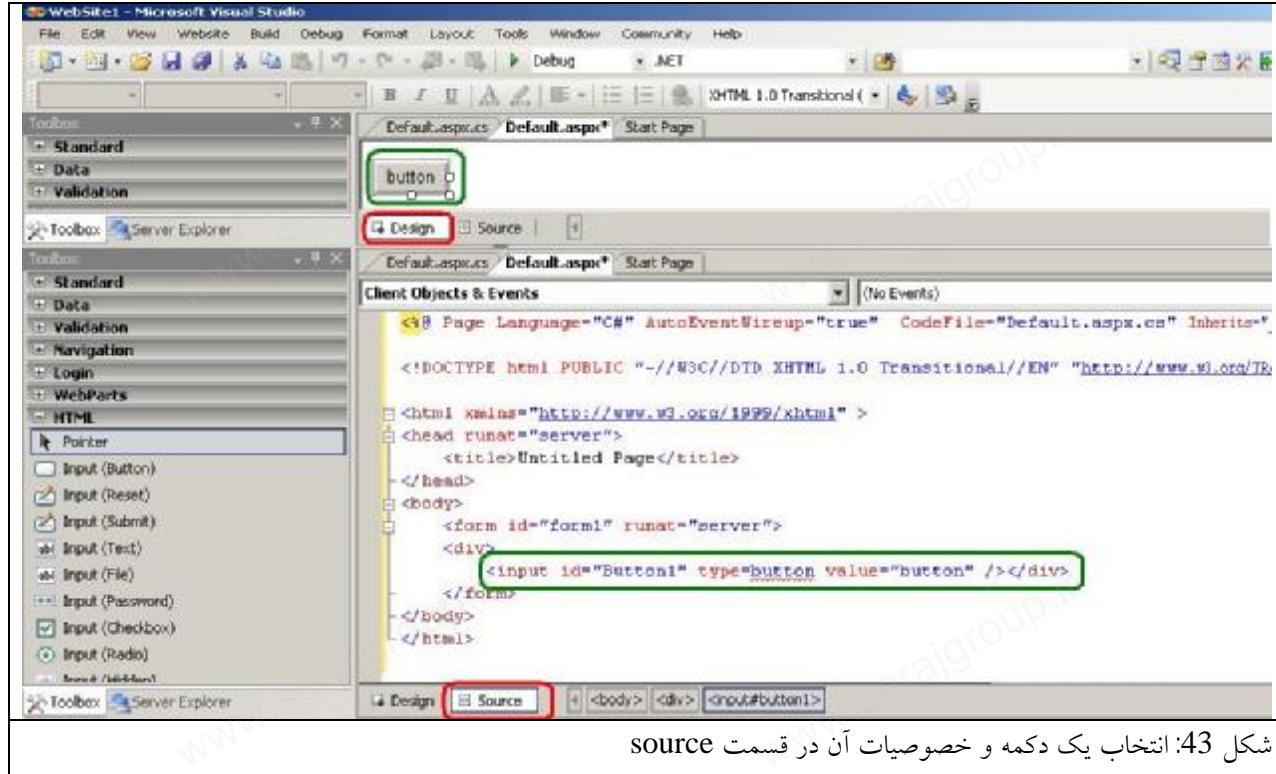
برای اینکار دو راه وجود دارد: 1- کد نویسی مستقیم و 2- استفاده از محیط ویژوال استودیو. بدیهی است که مورد ساده تر بوده و نیازی به محفوظات پیشین ندارد.



شکل 42: کد نویسی مستقیم (Source) و کدنویسی در محیط (Design) ویژوال استودیو

اگر به ToolBox ویژوال استودیو در سمت چپ صفحه توجه کنید چندین Tab مختلف برای طبقه بندی کنترل ها وجود دارد. تمام کنترل های مورد بحث ما در این فصل در Tab HTML قرار گرفته اند. اگر با Visual InterDev کار کرده باشید این Tab برای شما تازگی ندارد!

به راحتی می توان از این Tab یک دکمه (Button) را روی صفحه قرار داد و بقیه کنترل ها هم به همین صورت هستند. اگر به پایین صفحه سمت چپ نگاه کنید (در محیط VS.NET) می توانید بین حالت Design و مشاهده کد (Source) را انتخاب نمایید. با انتخاب کردن حالت Source برای قرار دادن یک دکمه روی صفحه نوشته است را می توان ملاحظه کرد.



شکل 43: انتخاب یک دکمه و خصوصیات آن در قسمت source

نکات مهم مربوط به این کنترل به شرح زیر هستند:

اگر به Source این کنترل دقت کنید و یا به خواص در گوشه‌ی سمت راست صفحه هنگامیکه کنترل Input یا همان دکمه ما انتخاب شده است دقت نمایید، خاصیت Type آنرا می‌توان تنظیم کرد. پیش فرض آن Button است (در این حالت). اگر به Submit تغییر نوع پیدا کنید، می‌توان توسط آن اطلاعات را به سرور فرستاد. و اگر به Reset تغییر یابد می‌توان برای ریست کردن تمام کنترل‌های روی صفحه از آن استفاده کرد. تا هنگامیکه خاصیت "RUNAT="server" به تگ آن اضافه نشود این کنترل کلاینت ساید بوده و فشردن آن باعث ایجاد رخدادی در سمت کاربر می‌شود. اضافه کردن رخداد برای این کنترل و کنترلهای HTML هم دقیقاً مانند اسکریپت نویسی JavaScript می‌باشد.

برای اینکه این کنترل بعنوان یک کنترل سرور بتوانند عمل کند همانطور که ذکر شد یا می‌توان خاصیت RUN AS Server را به صورت دستی وارد کرد و یا روی آن کلیک راست کنید و گزینه‌ی control را انتخاب نمایید.

<pre> using System.Configuration; using System.Web; using System.Web.Security; using System.Web.UI; using System.Web.UI.WebControls; using System.Web.UI.WebControls.WebParts; using System.Web.UI.HtmlControls; public partial class _Default : System.Web.UI.Page { protected void Button1_ServerClick(object sender, EventArgs e) { } } </pre>	
--	--

شکل 46: رخداد ServerClick که با استفاده از دوبار کلیک بر روی دکمه ایجاد می‌شود.

شکل 45: تبدیل کنترل معمول HTML به کنترل سرور

حالا با دوبار کلیک کردن بر روی آن صفحه `Code Behind` باز شده و می توانید در رخداد `Button1_ServerClick` آن کدتان را بنویسید.

اگر کاربر از مرورگر نگارش 4 به بالا استفاده می کند می توان از تگی به نام `<Button>` هم استفاده کرد. اگر نوع `Type` مربوط به تگ `<Image>` را مساوی `Image` قرار دهید می توان از کنترل تصاویر گرافیکی بجای دکمه استفاده کرد و با کمک رخداد `onMouseOver` و `onMouseOut` تصاویر آنها را هنگام نزدیک شدن ماوس تغییر داد. از کنترل `Image` هم می توان برای نمایش دادن تصاویر استفاده کرد و با برنامه نویسی در زمان های لازم تصاویر آنرا عوض کرد. اگر خاصیت `alt` آن مقدار دهی شود، یک `ToolTip` هنگام نگه داشتن ماوس روی آن نمایش داده می شود.

اگر نوع `Type` را مساوی `Text` قرار دهیم یک `TextBox` معمولی حاصل می شود و اگر نوع آن را مساوی `Password` قرار دهیم این `TextBox` هنگام نوشتن حروف کلمه `ی` رمز، ستاره نشان می دهد. برای نشان دادن و دریافت یک `TextArea` باشد در اینجا از کنترل `TextBox` استفاده می شود.

برای دریافت ورودی از نوع `Boolean` از کنترل `CheckBox` استفاده می کنیم و اگر خاصیت `Checked` آن `True` باشد به معنای انتخاب آن از طرف کاربر است.

برای تعریف `RadioButtons` در اینجا که برای انتخاب یک گزینه از بین یک گروه بکار برده می شود می توان از کنترل `RadioButton` استفاده کرد. هر کنترلی در این حالت باید یک `ID` منحصر بفرد داشته باشد اما `Name` آنها باید یکی باشد تا بتوان به راحتی با آنها کار کرد. برای دریافت اطلاعات از آنها برای مثال می توان `Radio[0].Checked` را چک کرد. استفاده از `DropDown List` راه دیگری در مقابل `RadioButton` است هنگامیکه تعداد گزینه های مایلی زیاد باشند. برای `Bind` کردن آن به یک آرایه می توان از خاصیت `DataSource` و `DatBind` آن استفاده کرد. با استفاده از خاصیت `SelectedIndex` می توان گزینه ای را که کاربر مشخص کرده، دریافت کرد. اگر می خواهید کاربر چندین آیتم را با هم در این کنترل انتخاب کند خاصیت `Multiple` آنرا `True` کنید.

برای ایجاد یک لینک از تگ `<a>` استفاده می شود که در خاصیت `HRef` آن آدرس لینک قرار داده می شود. کنترل مهمی که در این مجموعه قرار دارد و در کنترل های وب یافت نمی شود مربوط به `Upload` کردن فایل ها به سرور است که باید راجع به آن قدری بیشتر توضیح داد.

با استفاده از کنترل `HTMLInputFile` کاربران می توانند فایل های خود را به سرور `Upload` کنند. برای اینکار تگ `Form` را باید در سورس `HTML` بهبود بخشد و عبارت `Encrypt="MultiPart/form-data"` را به آن اضافه نمود. این خاصیت سبب می شود تا مرورگر متوجه این مطلب گردد که یکی از کنترل های روی صفحه برای `Upload` کردن فایل به سرور بکار گرفته می شود. برای اینکار از تگ `<Input>` با نوع `Type="File"` یعنی `<File>` استفاده می گردد. بعلاوه گزینه `"RUNAT="server"` هم لازم می باشد. پس از انتخاب فایل از طرف کاربر و فشردن دکمه `Submit` بقیه کار باید به صورت برنامه نویسی انجام شود.

```
fileSuggestionControl.postedFile.SaveAs(...)
```

برای اینکه بیشتر با این کنترلها آشنا شویم به بعضی از خصوصیات این کنترلها همراه با توضیحات در زیر آمده است.

خصوصیات	شرح
Id	رشته را برمی گرداند که مشخص کننده شی می باشد.
Accesskey	برای تنظیم کردن موقعی که شی انتخاب گردید کدام شی فعال شود.
dir	برای تنظیم نحوه خواندن متون روی شی بکار برده می شود، چپ به راست یا راست به چپ.
Disablelead	حالت شی را تنظیم می کند.
long	برای تنظیم زیان شی بکار برده می شود.
name	نام شی را مشخص می کند.
RunAt	طرف جواب دهنده رخدادها را نشان می دهد.
Style	تنظیم یک سبک داخلی برای شی
TabIndex	برای تنظیم ترتیب انتخاب شی موقعی که کلید Tab زده می شود.
Title	برای تنظیم یک اطلاعات کمکی در مورد شی برای کاربر بکار برده می شود.
Type	نوع شی ورودی را مشخص می کند.
Value	مقدار واردہ به شی را مشخص می کند.

برای اینکه اطلاعات ارائه شده در این فصل جتبه‌ی کاربردی پیدا کنند، کنترل‌های یاد شده را به صورت چند برنامه کوچک مورد استفاده قرار خواهیم داد تا نکات مربوط به آنها در عمل تجربه شوند و خصوصاً در هنگام برنامه نویسی اسکریپتی طرف Client در طی فصول آتی آشنایی لازم با پشت صحنه فرم‌های وب حاصل گردد.

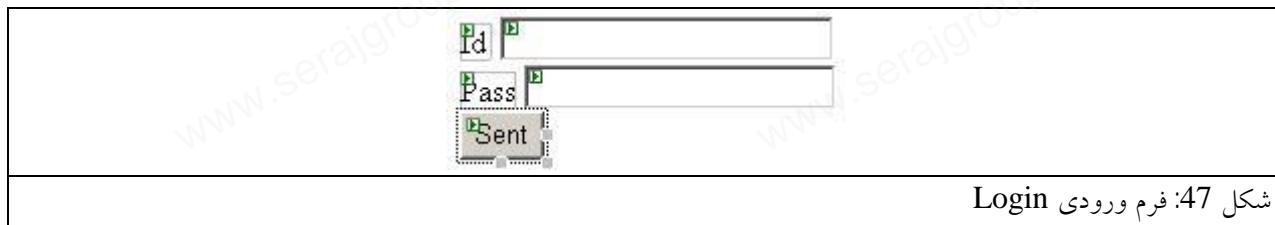
برنامه اول :

می خواهیم یک صفحه‌ی Login بسیار ساده درست کنیم که از کاربر شناسه‌ی کاربری (Id) و رمز عبور (password) را خواسته و پس از کلیک بر روی دکمه Submit در سرور مشخص شود که آیا کاربر و کلمه‌ی رمز او معتبر است یا خیر.

برای نوشتن این برنامه به شی‌ها و تنظیمات داخل جدول زیر نیاز می باشد.

value	(Id)	Type	Tag
-	txtId	text	Input
-	txtPass	password	Input
-	Id	label	Asp tag
-	Pass	label	Asp tag
Sent	btnSubmit	Submit	Input

در پروژه های بزرگ نامگذاری صحیح کنترل ها به شدت کنترل برنامه را ساده می کند. پس هیچگاه از نام های پیش فرض استفاده نکنید. روی تک تک کنترل ها می توان کلیک راست کرد و خاصیت Run As Server Control را انتخاب نمود.



شکل 47: فرم ورودی Login

حالا برروی دکمه که خاصیت اجرا بر روی سرور را پیدا کرده دوبار کلیک نمایید و در صفحه Code Behind در داخل تابعی که رخداد کلیک را نمایش می دهد کد زیر را بنویسید.

```
protected void btnSubmit_ServerClick(object sender, EventArgs e)
{
    if (txtID.Value == "a" && txtPass.Value == "b")
        Response.Write("Ok!");
    else
        Response.Write("Try again!");
}
```

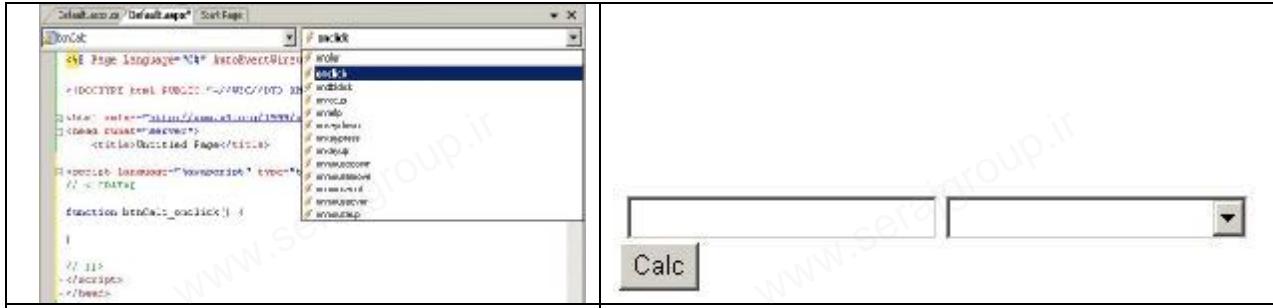
در مورد اشیاء ذاتی Response مانند ASP قبل اشارات شده و در طی فصول آتی بیشتر بحث خواهد شد. بدینهی است که چک کردن پسورد به این صورت نباید در داخل کد قرار گیرد و این مورد صرفاً مثالی از کاربرد کنترل های بودند.

برنامه دوم :

در برنامه دوم می خواهیم نحوه برنامه نویسی در طرف client را یاد بگیریم، در اینجا می خواهیم کاربر با استفاده از یک تکست باکس عددی را وارد نموده و سپس با کلیک کردن بر روی یک دکمه با یک پیغام مناسب حاصلضرب عدد داخل تکست باکس با عدد انتخاب شده در DropDownList نشان داده شود. برای نوشتن این برنامه به شی ها و تنظیمات داخل جدول زیر نیاز می باشد. این اشیاء را روی فرم قرار می دهیم.

value	(Id)	Type	Tag
-	txtNo	text	input
-	selNo	-	select
Calc	btnCalc	text	input

برای دیدن سورس برنامه در پایین صفحه روی زبانه Source کلیک نمایید. حال باید برای رخداد کلیک کردن دکمه برنامه طرف Client بنویسیم. برای این کار در بالای صفحه DropDownList Source دو وجود دارد که یکی برای انتخاب کنترل (btnCalc) را انتخاب می کنیم) و دیگری برای انتخاب رخدادی که پاسخ داده شود (onclick) را انتخاب می کنیم) و بدین صورت به صورت خود کار تابعی که رخداد کلیک را بیان می کند ایجاد می شود و هر دستوراتی که داخل بدنہ ی تابع بنویسید هنگام کلیک شدن بر روی دکمه اجرا می شود. کسانی که قبلاً از محیط های غیر ویژوال برای انجام اینکار استفاده می کردند، می دانند چه نعمت بزرگی به VS.NET اضافه شده است! حالا اگر به تگ مربوط به دکمه هم دقیق نباشد باید صورت خود کار عبارت زیر به آن اضافه شده است. (onclick="return btnCalc_onclick();")



شکل 50: نحوه اضافه کردن رخداد طرف client

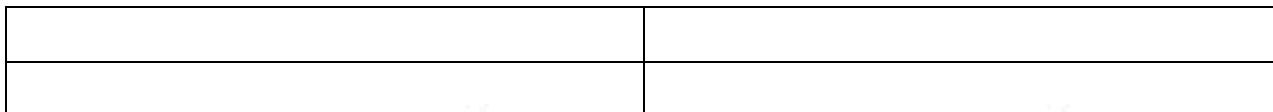
شکل 49: ظاهر فرم برنامه دوم

می خواهیم هنگامی که صفحه می خواهد dropdown list شود به همین دلیل در رخداد onload صفحه، این کار را انجام می دهیم. برای این منظور به همان ترتیبی که برای اضافه کردن تابع مربوط به رخداد کلیک عمل کردند، از منوی پایین افتادنی سمت چپ window را انتخاب می کنیم و از منوی پایین افتادنی سمت راست، گزینه `onload` را انتخاب می کنیم. و بدین صورت بطور خودکار تابع `window_onload` اضافه می شود و اگر به تگ مربوط به body دقت کنید ملاحظه می کنید که عبارت زیر هم به صورت خودکار به آن اضافه شده است.

```
 onload="return window.onload()"
```

اگر در برنامه ای لازم شد یک سری عناصر ثابت را به dropdown list اضافه کنیم روی آن کلیک راست کنید و گزینه خواص را انتخاب کنید. در صفحه‌ی باز شده، می‌توان آیتم‌های جدید را اضافه کرد و وقتی می‌خواهیم آیتمی را با برنامه نویسی به این نوع اضافه کنیم باید با استفاده از شیء سازنده `option` اینکار را انجام دهیم یعنی:

```
newOption=new Option(optionText,optionValue,defaultSelected,selected);
```



در این فصل نحوه‌ی برنامه نویسی کلاینت ساید و یا همان اسکریپت نویسی سنتی را با هم مرور کردیم و دیدیم که در این محیط جدید چقدر این نوع برنامه نویسی ساده‌تر و لذت بخش تر شده است و از اینجا به بعد در طی فصول آتی اگر لازم بود اسکریپت نویسی کلایپت ساید را در موارد معده‌دی به برنامه اضافه کنیم، حداقل روش مکانیزه آنرا به خوبی می‌دانیم. برای مطالعه بیشتر در مورد این نوع برنامه نویسی می‌توان به کتاب (راهنمای آموزش جاوا اسکریپت-ترجمه فرنود عسکری - نشر ادبستان) مراجعه کرد. کتابی مختصر - مفید و کاربردی در طی 220 صفحه می‌باشد.

فصل پنجم

معرفی کنترل های

وب و نحوه استفاده از

آنها در صفحات

ASP.NET

در فصول قبل ما به کنترلهای HTML پرداختیم که مقدمه برای فصول بعدی می باشند از این فصل به بعد برنامه نویسی تحت وب را بطور جدی دنبال خواهیم کرد به همین دلیل در فصل جاری مروری خواهیم داشت بر نحوه استفاده از کنترلهای وب در برنامه ها:

می توان علاوه بر کنترلهای های سرور وب را روی فرم های وب قرار داد. کنترلهای HTML در Client عمل خواهند کرد اما کنترلهای سرور وب در طرف سرور پردازش خواهند شد. کنترلهای سرور وب مزایایی قابل توجهی را نسبت به کنترل های HTML ارائه می دهند که در جدول زیر مرور شده اند:

HTML Controls	Server Controls	ویژگی
تنها می توانند به دخدادهایی در سطح صفحه عکس العمل نشان دهند.	می توانند به رخدادهای مربوط به کنترل پاسخ دهند.	رخدادهای سرور
داده ها نگهداری نمی شوند و باید به صورت دستی و با برنامه نویسی اینکار صورت گیرد.	داده های وارد شده در کنترل بین درخواست ها ثابت باقی میماند.	حفظ حالت
هیچگونه سازگاری اتوماتیکی وجود ندارد و باید با برنامه نویسی اینکار انجام شود.	به صورت خودکار نوع مرورگر را تشخیص می دهد و خود را هماهنگ میکند.	سازگاری
تنها ویژگی های مربوط به HTML در آنها وجود دارد.	از NetFramWork. به ارث رسیده شده است.	خواص

دلایل زیادی برای استفاده کردن از کنترلهای HTML وجود دارد با وجود اینکه کنترلهای سرور وب از نظر کاری خیلی قوی تر است میباشد. اولاً مهاجرت از ASP.NET به سادگی صورت گیرد. زیرا ASP قبلی تنها از عناصر HTML و یا همان کنترل های HTML جدید می توانست استفاده کند. دوماً تمام کنترل های نیازی به رخدادهای سمت سرور و یا حفظ حالت ندارند. کنترل کاملی در مورد شکل نهایی صفحه با کنترل های HTML وجود دارد، زیرا به صورت خودکار نمی توانند نوع مرورگر را حدس بزنند و خود را هماهنگ با آن نمایند (برخلاف کنترل های سرور وب).

در حالت کلی، استفاده از کنترل های سرور وب ساده تر و کارآتر می باشد. در جدول زیر کنترل هایی را که در ویژوال استودیو دات نت می بینند با هم مقایسه شده اند و عملی را که هر کدام انجام می دهند، مرور گردیده است.

HTML Control	Server Control	عملکرد
Text,TextArea,Password	Label, TextBox, Literal	نمایش متن
Tabel	DataGrid, Tabel	نمایش جدول
Select	ListBox, DropDownList, Repeater , DataList	انتخاب از لیست
Button,ResetButoon,SubmitButton	Button, LinkButton, ImageButton	انجام دستورات
Button,ResetButoon,SubmitButton	CheckBox, CheckBoxList, Radio ButtonList, radioButton	تنظیم مقادیر
Image	ImageButton, Image	نمایش تصاویر

<a>	HyperLink	حرکت بین صفحات
-	Placeholder, Panel	کنترل های گروهی
	Calender	کار با تاریخ
	AdRotator	نمایش تبلیغات
Horizontal rule	Literal	نمایش خط افقی
FileField	-	دریافت نام فایل از کلاینت
Inputb Hidden	بوسیله‌ی مدیریت حالت و به صورت خودکار انجام می‌شود.	ذخیره سازی داده‌ها روی صفحه
-		ارزیابی داده‌ها

با این مقایسه دیده می‌شود که دامنه کنترلهای سرور خیلی زیاد و همچنین در زیر خواهیم دید که استفاده کردن از آنها نیز خیلی آسان می‌باشد. اما ما باید توجه کنیم که نوشتن برنامه تحت وب مستلزم اولاً کارایی خوب دوماً دارای سرعت مناسب و سوماً دارای تقسیم بندی مناسب در انجام عملیاتها باشد بطوریکه قسمتهای که نیاز به پردازش سمت سرور نمی‌باشد نباید از کنترلهای سرور استفاده کرد.

کار با متن :

روش‌های زیادی برای نمایش متن روی یک صفحه وجود دارد. برای یک متن فقط خواندنی می‌توان از روش‌های زیر استفاده کرد:

استفاده از دستور (`Response.Write("Some Text")`), استفاده از کنترل `Label` و استفاده از کنترل `TextBox` با خاصیت `True` مساوی `ReadOnly`.

برای نمایش یک متن قابل ویرایش، می‌توان از کنترل سرور `TextBox` استفاده کرد. خواص کلیدی آن در جدول زیر مرور شده‌اند:

خاصیت	نحوه‌ی استفاده
<code>Text</code>	برای دریافت متن از آن و یا نوشتن متن در آن بکار برده می‌شود.
<code>TextMode</code>	حالت یک خطی <code>SingleLine</code> , چند خطی <code>TextArea</code> مانند <code>MultiLine</code> و یا حالت <code>Password</code>
<code>ReadOnly</code>	در صورت <code>True</code> بودن، کابر نمی‌تواند آنرا تغییر دهد.
<code>AutoPostBack</code>	تا زمانی که <code>True</code> نشود نمی‌توان از رخداد <code>TextChanged</code> آن کنترل استفاده کرد و به صورت پیش فرض <code>False</code> است.

نحوه استفاده کردن از این کنترل در قسمتهای قبلی گفته شده و همچنین در مثالهای بعدی نیز دوباره تکرار خواهد شد.

کار با جداول و لیست ها:

برای آراستن متن در ردیف ها و ستون ها باید از یکی از کنترلهای لیست که در جدول بالا نامبرده شدند استفاده شود. از DropDownList و جدول برای جداول و لیست های دینامیک استفاده می گردند. از ListBox و DataGrid و

برای نمایش جدول و لیست های پیچیده مانند آنهایی که حاوی کنترل ها هستند و یا متصل به پایگاه داده اند استفاده می گردد. از

ListBox برای نمایش متنی فقط خواندنی در یک لیست با قابلیت Scroll و از DropDownList برای نمایش متن فقط

خواندنی در یک لیست کرکره ای ساده و از Table برای نمایش متن و یا کنترل ها در ستون ها و ردیف ها و از DataGrid برای نمایش داده ها و کنترلهای پیچیده در جداول استفاده می شود.

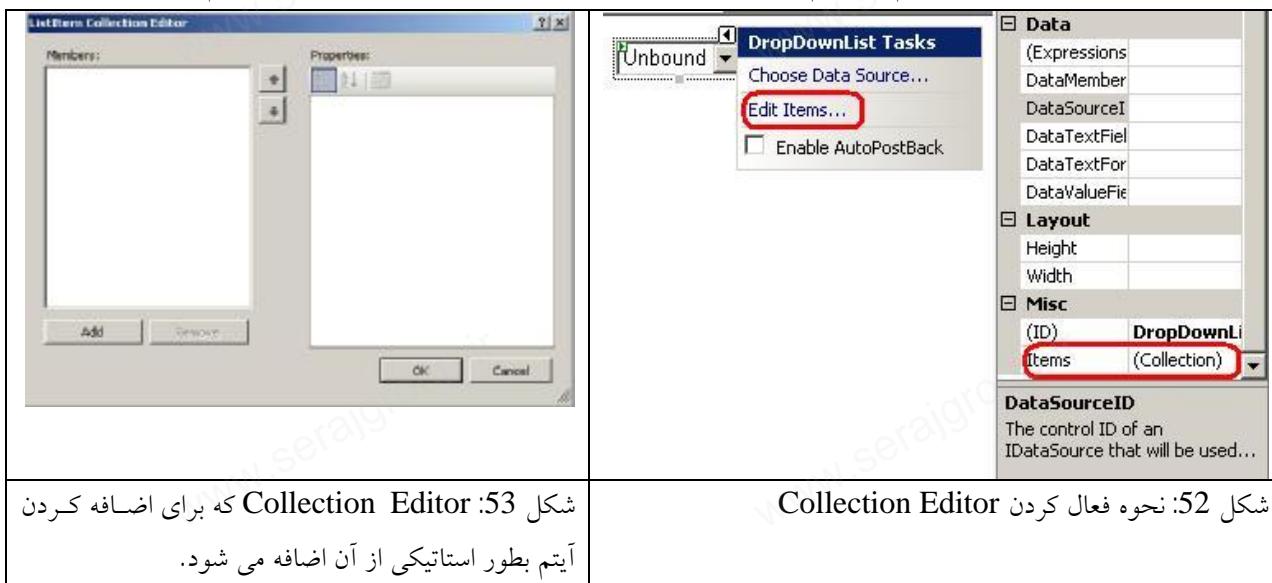
برای اضافه کردن آیتم به Table هم می توان بطور استاتیکی در زمان طراحی و هم

بطور دینامیکی در زمان اجرای برنامه این عمل را انجام داد. برای اضافه کردن آیتم بطور استاتیکی به سه کامپونت بالا می توان از

Collection Editor استفاده کرد. برای فعال کردن آن می توان آیکن مربوطه را انتخاب کرده و از پنجره خصوصیات، برای

کامپونت ها Items DropDownList و برای کامپونت Table خاصیت Rows را انتخاب کنید یا

موقعی که کامپونت مربوطه را روی فرم بگذاریم از منوی ظاهر شده می توان گزینه EditItem را انتخاب کنیم.



برای اضافه کردن دینامیکی آیتم در زمان اجرای برنامه به DropDownList و ListBox می توان با استفاده از متد Add به کلکسیون Items این کامپونت ها استفاده کرد. (...).
ListBox1.Items.Add(...).

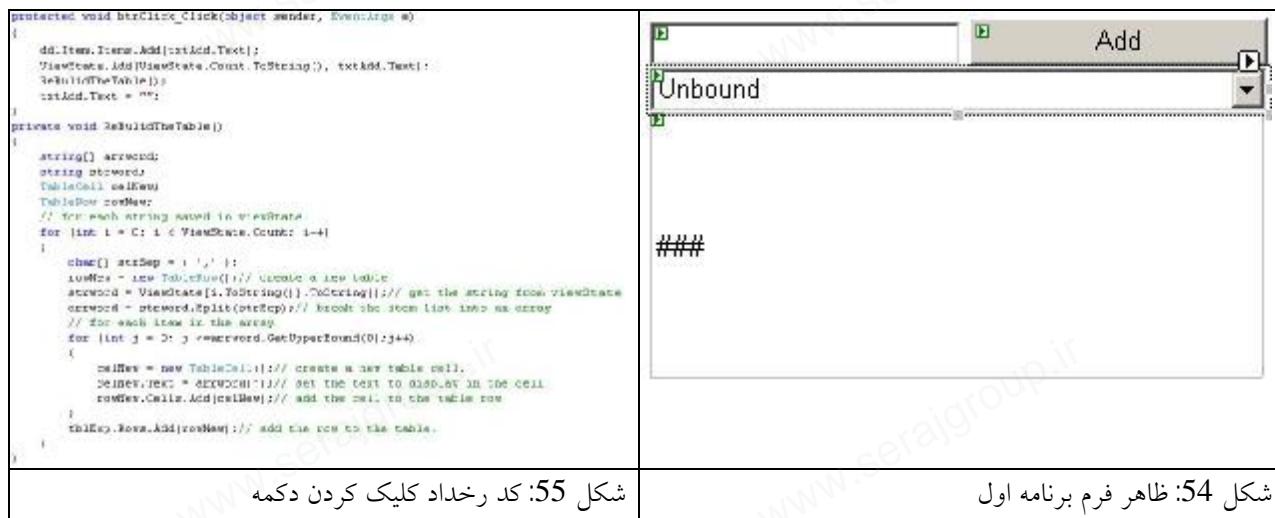
اما برای اضافه کردن عنصر به کامپونت جدول باید طور دیگری عمل کرد. بخارط اینکه این کامپونت تنها اطلاعات جدولی را برای عنصرهای که در زمان طراحی بطور استاتیکی اضافه شده اند نگهداری می کند. اما برای عناصر که در زمان اجرا اضافه می شوند (چه ردیف و چه ستون) باید جدول را از نو ایجاد کرد و عنصرهای قبلی را متناظراً در جای قبلی وارد کرد. در این مورد ما یک مثال در ادامه خواهیم آورد.

مثال اول:

هدف از این مثال یاد گرفتن نحوه اضافه کردن آیتم بطور دینامیکی به کامپونت های DropDownList و Table می باشد. در این مثال می خواهیم که در رخداد کلیک کردن دکمه محتوای یک TextBox را به Table و همچنین به یک DropDownList اضافه کنیم. برای نوشتن این برنامه به شی ها و تنظیمات داخل جدول زیر نیاز می باشد. این اشیاء را روی فرم قرار می دهیم.

Text	(Id)	شی
Add	btnClick	Button
-	txtAdd	TextBox
-	tblExp	Table
-	ddlItem	DropDownList

همانطور که گفته شد در این مثال مفهوم بازسازی کردن جدول در کد پیاده سازی شده است. که در شکل زیر آورده شده.



دریافت آیتم انتخاب شده از یک لیست:

برای دسترسی به آیتم انتخاب شده از خاصیت SelectedItem می توان استفاده کرد. برای نمونه در مثال قبل، می توان از دستور زیر برای بدست آوردن آیتم مورد نظر استفاده کرد.

ddItem.SelectedItem.Text

کردن ساده در کنترل لیست ها: DataBind

کنترلها مقادیرشان را می توانند از هر منبع داده ای در برنامه شما دریافت کنند. برای مثال از یک بانک اطلاعاتی ، آرایه ، خاصیت یک شیء و غیره. در زیر یک نمونه ساده از این مقادیر آورده شده است.

مثال ۲

به مثال قبلی یک آرایه از رشته ها را اضافه کنید با استفاده از دستور `arrData1 = {"a", "b", "c"}` خاصیت `DataSource` کامپونت `DropDownList` را به آرایه نسب بدهید. و در رخداد `Page_Load` دستور `Page.DataBind()` را فراخوانی کنید، در مرحله بعد برنامه را اجرا کنید. می بینید که اعضای آرایه به لیست اضافه شده اند.

نکته:

هنگامیکه از `DataBinding` در کنترل های سرور استفاده می کنید، می توان حفظ مدیریت و حالت را خاموش کنید. این مورد کارایی را افزایش می دهد، زیرا متدهای `Bind()` به صورت اتوماتیک این مدیریت خودکار را جایگزین می کند. برای اینکار، خاصیت `EnableViewState` را `False` کنید.

ترتیب اجرای دستورات:

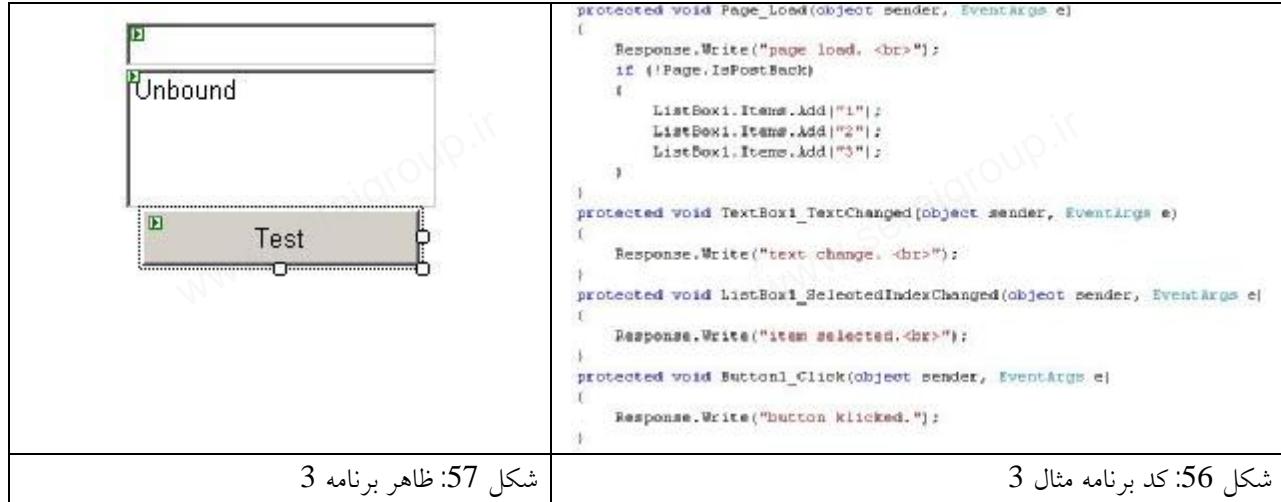
کنترل های سرور `ImageButton`، `LinkButton`، `Button` و `TextBox` برای انجام دستورات بکار برده می شوند. این کنترل ها سبب وقوع رخدادهای به نام `Post-back` می شوند. اینگونه رخدادها از طرف مرورگر درخواست شده و سبب می شوند که سرور به آن پاسخ دهد. برای اینکه ترتیب رخدادهای اتفاق افتاده در یک صفحه را ببینیم به مثال زیر توجه کنید.

مثال ۳

هدف از این تمرین یاد گرفتن کار کردن با `Button`، رخدادهای مربوط به `TextBox` و `ListBox` می باشد به همین دلیل کامپونت های زیر را روی فرم قرار دهید:

Text	(Id)	شی
Test	Button1	Button
-	ListBox1	ListBox
-	TextBox1	TextBox

سپس کدهای زیر را به رخدادهای مختلف صفحه اضافه کنید.



شکل ۵۶: کد برنامه مثال ۳

استفاده از **ImageButton** و **LinkButton** بسیار واضح و سر راست می باشد. کنترل **ImageButton** قابلیت های بیشتری را ارائه می دهد. رخداد کلیک آن حاوی آرگومانهای x و y مکانی هستند که با ماوس روی آن کلیک کرده اید و به آن **ImageMaps** هم می گویند.

در یافت مقادیر کاربر:

با استفاده از کنترل‌های CheckBoxList، RadioButtonList و یا CheckBox، RadioButton می‌توان داده‌های بولی و غیره را از کاربر دریافت کرد. همانند کنترل‌های ListBox و DropDownList می‌توان از ویرایشگر Collection برای اضافه کردن آیتم به CheckBoxList یا RadioButtonList استفاده کرد. برای این کار باید بر روی خاصیت Items آنها در پنجره‌ی خواص کلیک کرد. با استفاده از خواص Checked در آنها می‌توان متوجه این شد که آیا CheckBox یا RadioButton انتخاب شده‌اند یا خیر.

هنگامی که یک **RadioButton** را روی فرم قرار می‌دهید با دیگر **RadioButton**‌ها بر خلاف **RadioButtonList** ها هیچ برهم کنشی ندارد. برای این منظور باید خاصیت **GroupName** آنها را برای هر **RadioButton** مشخص کرد. برای دریافت یا تنظیم مقادیر **RadioButtonList** و یا **CheckBoxList**، از حلقه **foreach** می‌توان استفاده کرد. برای این کترلهای می‌توان خاصیت **Selected** را برای فهمیدن انتخاب شدن یا انتخاب نشدن، می‌توان استفاده کرد.

مثال 4:

می خواهیم یک مثال ساده برای آشنایی با نحوه استفاده از کنترل های `CheckBoxList` و `RadioButtonList` بنویسیم. یک `RadioButtonList` یک دکمه را روی فرم قرار دهید. سپس روی گزینه `Items` کامپونت `RadioButtonList` در صفحه خواص کنترل کلیک کنید. سه گزینه دلخواه به آن اضافه نمایید. و همچنین روی گزینه `Items` کامپونت `CheckBoxList` در صفحه خواص کنترل کلیک کنید. سه گزینه دلخواه نیز به آن اضافه نمایید. در رخداد کلیک کردن دکمه می خواهیم مشخص نماییم که کدام یک از گزینه ها انتخاب و با پیغام مناسب آن را روی فرم گزارش نماییم.

<input checked="" type="checkbox"/> آننم ۱ <input type="checkbox"/> آننم ۲ <input checked="" type="checkbox"/> آننم ۳ <input checked="" type="radio"/> آننم ۴ <input type="radio"/> آننم ۵ <input type="radio"/> آننم ۶ <input type="checkbox"/> Button	<pre> protected void Button1_Click(object sender, EventArgs e) { foreach (ListItem lstItems in RadioButtonList1.Items) { if (lstItems.Selected) Response.Write(lstItems.Text + "RadioButton is selected.
"); } foreach (ListItem lstItems in CheckBoxList1.Items) { if (lstItems.Selected) Response.Write(lstItems.Text + "CheckBoxList is selected.
"); } } </pre>
شکل 59: اجرای برنامه	شکل 58: کد مربوط به رخداد کلیک کردن

نمایش گرافیک و تبلیغات:

روش های مختلفی برای نمایش گرافیک روی فرم وجود دارد

1. بعنوان پس زمینه با استفاده از خاصیت BackGround فرم و ب می توان یک تصویر را روی کل صفحه قرار داد. با استفاده از BackImageURL یک کنترل Panel می توان در قسمتی از صفحه بجای کل صفحه تصویر را نمایش داد.
2. بعنوان پیش نما: با استفاده از کنترل Image در زمان اجرا هم می توان خاصیت ImageURL آنرا تنظیم کرد.

3. بعنوان یک دکمه: با استفاده از کنترل ImageButton

4. بعنوان تبلیغات: با استفاده از کنترل AdRotator برای نمایش تصاویر از لیستی از موارد تبلیغاتی.

موارد بالا بجز مورد چهارمی همه سرراست هستند. بنابراین در مورد چهارم ما بعداً بیشتر صحبت خواهیم کرد و یک مثال در این مورد خواهیم آورد.

کنترل های گروهای:

برای مثال یکی از کاربردهای کنترل های گروهی این است که شما یک صفحه لاگین درست کنید و در یک قسمت صفحه کنترل های مربوط به صورت یک گروه قرار گیرند و پس از لاگین کردن آنها را مخفی کنید و قسمت دیگر صفحه را نمایش دهید.

از کنترل Panel برای اینکار استفاده می شود روی این کنترل نمی توان کنترل ها را ترسیم کرد باید ابتدا کنترل روی فرم قرار گیرد و سپس به روی آن Drag شود.

مثال 5

به عنوان تمرین یک Panel را روی فرم قرار دهید روی TextBox دو برجسب و یک دکمه روی آن فرار می دهیم. در رخداد Click مربوط به دکمه کد زیر را می نویسیم که اگر ورودی TextBox ها درست بود Panel را بردارد و پیغام موفقیت آمیز را بفرستد. اما اگر غلط بود Panel سر جای خود باشد و پیغام نادریت را بفرستد.

	<pre>protected void Button1_Click(object sender, EventArgs e) { if (TextBox1.Text == "naseri" & TextBox2.Text == "password") { form1.Controls.Remove(panel1); Response.Write("نام کاربری و کلمه عبور مطابق درست می باشد"); } else { Response.Write("نام کاربری یا کلمه عبور مطابق درست نمی باشد لطفا دوباره مسأله را بررسی کنید"); } }</pre>
شکل 61: نمایی ظاهری و اجرای مثال 5	شکل 60: کد مربوط به رخداد کلیک دکمه

کار با تاریخ:

با استفاده از کنترل تقویم می توان اطلاعات مربوط به روزها را نمایش داد برای کار با این کنترل از رخدادهای SelectionChanged استفاده می شود و همچنین خواص SelectedDate و SelectionChanged. رخداد SelectionChanged رخدادی post – back می باشد. بنابراین به محض تغییر تاریخ، سرور را مطلع می نماید.

مثال 6

در این مثال می خواهیم که یک کنترل تقویم (Calender) روی فرم قرار دهیم و بمحض اینکه کاربر روز تقویم را عوض کرد روز انتخاب شده در یک برچسب نمایش داده شود به همین منظور یک کنترل تقویم و یک برچسب را روی فرم قرار می دهیم. و در رخداد SelectionChanged کنترل تقویم کد زیر را می نویسیم.

```
Label1.Text = Calendar1.SelectedDate.ToString();
```

در بخش‌های قبلی ما با بعضی کنترلهای استاندارد آشنا شده ایم کار بقیه کنترلهای بطور مختصر بدین صورت می باشد.

کنترل BulletedList برای ایجاد پاراگرافهای متني که بصورت نکته دار باشد استفاده می شود.

کنترل FileUpload برای فرستادن فایل از طرف Client به طرف سرور.

کنترلهای بخش Data

اضافه کردن آیتم ها به Repeater و DataList و GridView

قبل از اینکه نحوه کار با کامپونت های بالا بیان نماییم، بعضی اصطلاحات می باشد که باید بیان شود:

DataBinding: یعنی اضافه کردن آیتم از منبع داده (Bind).

DataSource: مجموعه ای از المانهای HTML یا کنترلهای سرور می باشد، که برای هر آیتم داده از منبع داده تکرار می شود.

DataRepeater: منبع داده که می توان یک آرایه، یک جدول و یا یک مجموعه می توان باشد.

مثال 7

در این مثال می خواهیم نحوه اضافه کردن ستون های Template را به GridView نشان دهیم و چگونگی Bind کردن کنترل های موجود در آن به یک منبع داده:

-1 - ابتدا یک آرایه از رشته ها را بصورت عمومی در برنامه خود تعریف کنید.

-2 - کنترل GridView را روی فرم قرار دهید.



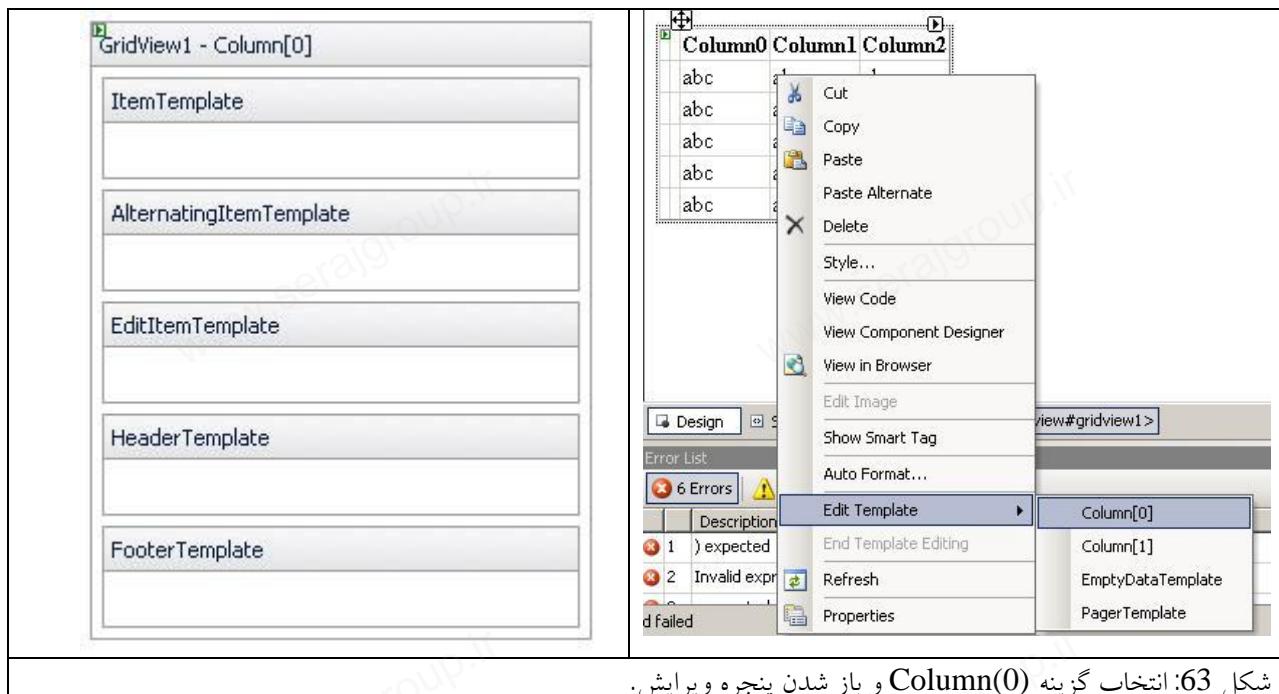
شکل 62: ظاهر خاصیت Columns

-4 در صفحه‌ی که با انتخاب خاصیت Columns ظاهر شده، در قسمت Available fields گزینه Add کلیک نمایید. برای این مثال دو ستون اضافه کنید. سپس دکمه Ok را بزنید.

Ok را انتخاب کرده، سپس بر روی دکمه Add کلیک نمایید. برای این مثال دو ستون اضافه کنید. سپس دکمه Ok را بزنید.

-5 روی GridView کلیک راست کنید و سپس Columns(0) را از منوی کرکره‌ای انتخاب نمایید. ظاهر

کترل به حالت Edit تغییر می‌کند.

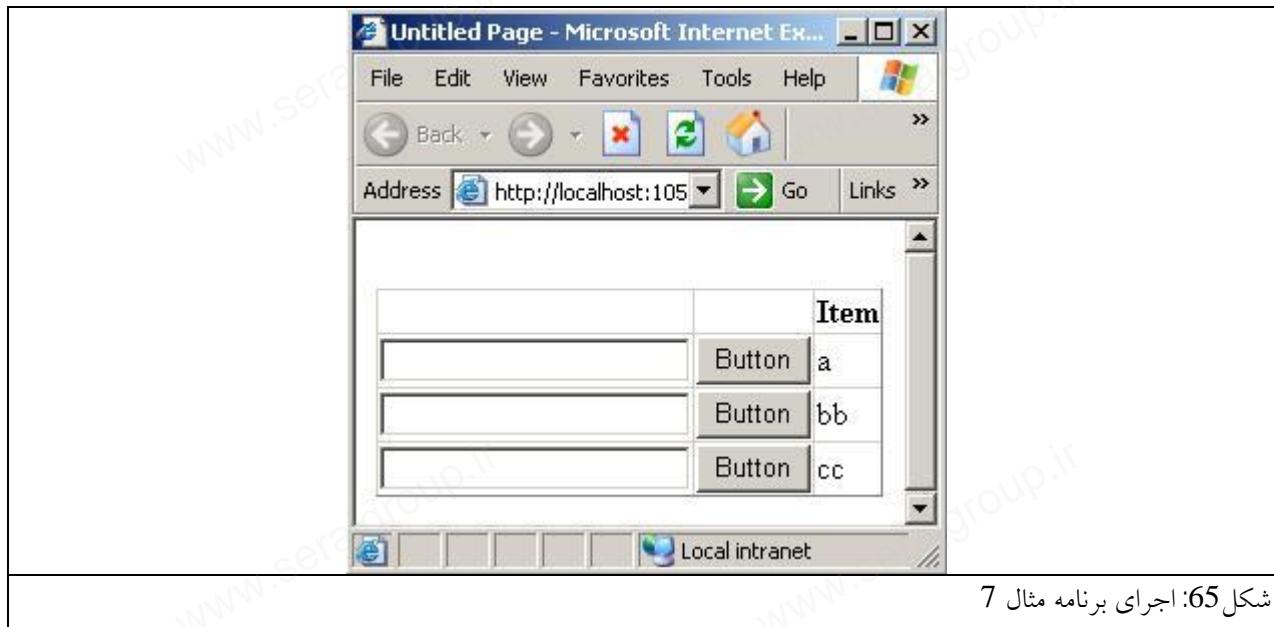


شکل 63: انتخاب گزینه Column(0) و باز شدن پنجره ویرایش.

-6 سایر کترل‌ها را روی فرم وب قرار دهید و سپس به Template مربوط به کترل، Drag کنید تا به

اضافه شوند. برای مثال به این TextBox یک GridView اضافه کنید.

- پس از تنظیمات روی GridView کلیک راست نمایید. و سپس گزینه End Template Editing را انتخاب نمایید. -7
- دومین ستون را ویرایش کنید. برای اینکار بر روی کنترل GridView کلیک راست کنید و سپس Column(1) را از منوی انتخاب نمایید. -8
- مراحل 5 تا 7 را برای این ستون تکرار کنید. برای این مثال، یک دکمه ترسیم کنید و آنرا Drag کنید به GridView در Template Column(1) مریبوط به . -9
- در این مرحله در رخداد مریبوط به Load_Form دستورات زیر که برای Bind کردن یک منبع به GridView می باشد، را وارد کنید و سپس آنرا اجرا نمایید: -10
- ```
GridView1.DataSource = a;
GridView1.DataBind();
```
- یادآوری: a در این مثال نام آرایه عمومی می باشد که بعنوان منبع استفاده شده است.



### ارزیابی داده های ورودی کاربر:

یکی از مهمترین مراحل دریافت داده ها از کاربر این است که اطمینان حاصل کنیم آیا داده های وارد شده از طرف او معتبر هستند یا خیر؟ به این مراحل تعیین اعتبار می گویند. اصول تعیین اعتبار بدین شرح می باشند: آیا کاربر چیزی را وارد کرده است؟ آیا نوع صحیحی از داده را وارد کرده است (برای مثال آدرس ایمیل). آیا داده ورودی در یک بازه خاص قرار دارد؟ و امثال آینها.

ASP.NET یک سری از کنترلهای ارزیابی داده های ورودی را قبل از اینکه داده ها به سرور فرستاده شوند، در سمت کلاینت مهیا کرده است و به این صورت بدون درگیر شده سرور و تحمل بار اضافی به آن صورت می گیرد.

در جدول زیر شش کنترل موجود برای تعیین اعتبار داده های ورودی کاربر، توضیح داده شده اند. این نوع کنترل ها مقدار کنترلی را که در خاصیت ControlToValidate آنها مشخص شده است را بررسی می تمایند.

| کنترل                      | کاربرد                                                                                       |
|----------------------------|----------------------------------------------------------------------------------------------|
| RequiredFieldValidator     | بررسی می کند که آیا کنترل حاوی داده است یا خیر (آیا کاربر چیزی را وارد کرده است?)            |
| CompareValidator           | بررسی می کند که آیا داده وارد شده با داده ای موجود در کنترل دیگر تطابق دارد یا خیر.          |
| RangeValidator             | بررسی می کند که آیا آیتم وارد شده بین دو مقدار تعریف شده قرار دارد یا خیر.                   |
| RegularExpressionValidator | بررسی می کند که آیا داده وارد شده با فرمت مشخص شده مطابقت دارد یا خیر.                       |
| CustomValidator            | اعتبار داده ورودی را توسط اسکریپتی کلاینت سایت یا سمت سرور و یا هر دو انجام می دهد.          |
| ValidationSummary          | تمام موارد بررسی شده را در یک مکان نمایش می دهد یا به صورت کلی فقط یک پیغام را نمایش می دهد. |

برای استفاده از کنترل های تعیین اعتبار باید مراحل زیر طی شود:

- 1 ترسیم و یا قرار دادن یک کنترل اعتبار ورودی روی فرم و تنظیم کردن خاصیت ControlToValidate آن به کنترلی که می خواهد تعیین اعتبار شود. اگر شما از کنترل CompareValidator استفاده می کنید، باید خاصیت ControlToCompare را نیز تنظیم کنید.
- 2 خاصیت ErrorMessage را به پیغامی که می خواهید هنگامیکه داده ای ورودی معتبر نیست نمایش دهد تنظیم کنید.
- 3 خاصیت Text آنرا برای نمایش دادن پیغامی هنگامیکه خطای خطا رخ می دهد، تنظیم کنید. از این مورد برای نمایش دادن توضیحات طولانی تر از خاصیت ErrorMessage استفاده می شود.
- 4 در صورت نیاز یک کنترل ValidationSummary را روی فرم وب برای نمایش تمام پیغام های خطای حاصل از کنترلهای تعیین اعتبار، ترسیم کنید.
- 5 تنها وجود کنترلی که سبب فرستاده شدن یک رخداد Post-Back می شود، سبب انجام بررسی تعیین اعتبار می گردد. پس وجود یک چنین کنترلی (مانند یک دکمه) روی فرم در این حالت ضروری است. برای نمایش خطاهای تعیین اعتبار به صورت یک MessageBox خاصیت کنترل ValidationSummary به نام ShowMessage را کنید.

### ترکیب کنترل های تعیین اعتبار

یک کنترل روی صفحه می تواند از چندین کنترل تعیین اعتبار استفاده کند. برای مثال TextBox ایی که ایمیل کاربر را دریافت می کند می تواند به کنترل RequiredFieldValidator و کنترل RegularExpressionValidator متصل باشد. در مورد کنترل CompareValidator باید به نکات زیر توجه داشت:

اگر کنترل مشخص شده در خاصیت ControlToValidate نتواند به یک نوع داده مناسب تبدیل شود، نتیجه Invalid خواهد بود.

اگر کنترل مشخص شده در خاصیت controlToComare نتواند به یک نوع داده مناسب تبدیل شود، نتیجه Valid خواهد بود. در این حالت باید از یک کنترل دیگر برای تعیین اعتبار بهره جست.

هنگامیکه می خواهید از کنترل CompareValidator استفاده کنیم با تنظیم کردن خاصیت Operator آن می توان نوع مقایسه را انجام داد. برای مثال گاهی از اوقات ورودی یک فیلد باید کمتر یا مساوی فیلد دیگری باشد و امثال اینها. این خاصیت در پنجره خواص کنترل ذکر شده به سادگی قابل تنظیم است.

### مثال 8:

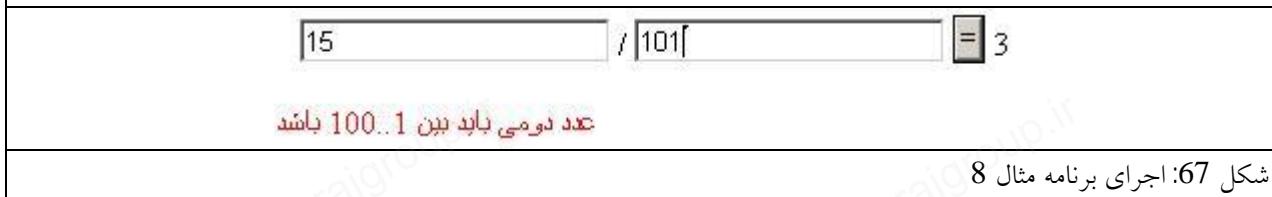
هدف از این مثال استفاده از کنترلهای بالا برای ارزیابی بعضی از کنترلهای می باشد. در این مثال خواهیم دو عدد که در دو TextBox می باشند بر هم تقسیم نماییم بطوریکه هیچکدام از اینها خالی نباشد همچنین عدد دومی بزرگتر از صفر و کوچکتر از 1000 باشد. نهایتاً جواب در یک Label نشان داده شود. به همین منظور دو Label و یک دکمه را روی فرم قرار دهید و برای ارزیابی خالی نبودن دو کنترل TextBox دو ControlToValidator و برای ارزیابی رنج RequiredFieldValidator قرار دهید و برای ارزیابی دویتی بودن دو کنترل TextBox دو RangeValidator کنترل ControlToValidate را روی فرم قرار دهید. خاصیت ControlToValidate دومی یک کنترل RangeValidator را روی فرم قرار دهید. ابتدا خاصیت RequiredFieldValidator TextBox ها را به TextBox دویتی بودن دهید. و در مورد کنترل RangeValidator دکمه کاری ساده برای تقسیم کردن دو عدد دویتی بودن را به TextBox دویتی بودن دهید. پس از این کارها نوبت به تنظیم کردن خاصیت ErrorMessage تک تک کنترل های تعیین اعتبار می باشد. برای هر کدام یک عبارت معنا دار بنویسید. روی دکمه دو بار کلیک کنید و در رخداد Click دکمه کاری ساده برای تقسیم کردن دو عدد دویتی بودن را به integer تغییر دهید.

```
Label2.Text=(Convert.ToInt32(textBox1.Text) /
Convert.ToInt32(textBox2.Text)).ToString();
```

سپس آنرا اجرا نمایند و نتیجه را مشاهده نمایید. اگر در TextBox دومی عددی بزرگتر از یک را وارد نمایید یک پیغام خطای طرف کنترل RangeValidator ظاهر می شود. چرا؟! چون نوع داده ای ورودی را مشخص نکرده ایم! خاصیت Type این کنترل را به integer تغییر دهید.



شکل 66: ظاهر برنامه مثال 8



شکل 67: اجرای برنامه مثال 8

## باطل کردن تعیین اعتبار:

چون تعیین اعتبار قبل از اینکه سرور صفحه ای را پردازش کند اتفاق می افتد، ممکن است کاربر در بین انبوهی از پیغام های خطا گیر بیفتند و نه راه پس داشته باشد و نه پیش! برای باطل کردن این کنترل ها می توان از یک کنترل که خاصیت-Post Nداشته باشد مانند Submit HTML Control استفاده کرد. در این حالت می توان کاربر را با بررسی کردن IsValid شی Page به یک صفحه دیگر هدایت کرد.

### مثال ۹:

هدف از این مثال یاد گرفتن اینکه چگونه از دست کنترلهای ارزیابی فرار کرد و همچنین یاد گرفتن اینکه چگونه می توان یک فرم دیگر را به سایت اضافه کرد و به آن مراجعه کرد. به همین منظور روی فرم یک TextBox، یک RequiredFieldValidator و دو دکمه که یکی سرور ساید و دیگری طرف کلاینت باشد اضافه کنید. خاصیت ControlToValidate کنترل RequiredFieldValidator را به TextBox تنظیم کنید. در این اگر روی دکمه طرف سرور کلیک کنید و TextBox خالی باشد پیغام خطا گرفته می شود ولی اگر روی دکمه طرف کلاینت کلیک شود به صفحه بعدی هدایت می شود. برای برآورد کردن این منظور یک فرم جدید با استفاده از گزینه.. Add New Item.. منوی WebSite پروژه اضافه کنید روی این فرم یک Label اضافه کنید و آن را به Text Hello تغییر دهید. سپس با اضافه کردن HTML ارزیابی کردن آن را غیر فعال کرده سپس در Rخداد onclick="page\_ValidationActive=false;"

کد زیر را وارد می کنید:

```
protected void Page_Load(object sender, EventArgs e)
{
 if (Page.IsPostBack)
 {
 Page.Validate();
 if (!Page.IsValid)
 {
 Response.Redirect("Default2.aspx");
 }
 }
}
```

شکل 68: کد مربوط به Rخداد بار شدن صفحه

این برنامه را می توان ادامه داد....

### تعیین اعتبار سفارشی:

اگر هیچکدام از کنترلهای تعیین اعتبار نیاز شما را برآورده نمی کند می توانید از کنترل CustomValidator استفاده نمایید. اگر لازم است پردازش سمت سرور انجام شود، کد تعیین اعتبار را در Rخداد ServerValidate قرار دهید. برای تعیین اعتیار سمت ClientValidationFunction خاصیت این کنترل را باید تنظیم کرد.

### مثال 10:

هدف از این مثال نحوه کار کردن با کنترل CustomValidator می باشد. برای این منظور در این برنامه می خواهیم برسی کنیم که ورودی یک عدد اول است یا خیر.

روی فرم یک یک دکمه و یک TextBox قرار دهید. خاصیت ControlToValidate مربوط به TextBox را به CustomValidator تنظیم کنید. سپس در رخداد ServerValidate کدهای شکل زیر را وارد کنید در این کد با استفاده از دستور Try-catch استثناهای کنترل شده اند.

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
 try
 {
 // Get value from ControlToValidate (passed as args).
 int iPrime = Int32.Parse(args.Value);
 if (iPrime != 0)
 {
 for (int iCount = 2; iCount <= (iPrime / 2); iCount++)
 {
 // If number is evenly divisible, it's not prime,
 // return False.
 if ((iPrime % iCount) == 0)
 {
 args.IsValid = false;return;
 }
 // Number is prime, return True.
 args.IsValid = true;return;
 }
 }
 else
 {
 args.IsValid = false;return;
 }
 }
 catch (Exception e)
 {
 // If there was an error parsing, return False.
 args.IsValid = false;return;
 }
}
```

شکل 69: کد مربوط به رخداد ServerValidate

## موارد تکمیلی کنترل های وب:

### طريقه‌ی حرکت بین صفحات مختلف در ASP.NET

برای حرکت بین صفحات مختلف، ASP.NET روش ای مختلفی را ارائه داده است که در ادامه بررسی خواهند شد.

| کاربرد                                                                                                                                                        | روش هدایت و حرکت به صفحه ای دیگر |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|
| حرکت به صفحه ای دیگر                                                                                                                                          | کنترل HyperLink                  |
| معادل کلیک بر روی یک کنترل HyperLink می باشد.                                                                                                                 | تابع Response.Redirect(...);     |
| به فرم وب جاری خاتمه بخشیده و اجرای صفحه ای دیگر را آغاز می کند. این روش تنها برای حرکت بین فرم های وب (.aspx). کاربرد دارد.                                  | تابع Server.Transfer(...);       |
| در حالیکه فرم وب جاری در حال نمایش است. اجرای یک فرم وب جدید را آغاز می کند. محتويات هر دو فرم ترکیب خواهند شد. این روش نیز تنها برای فرم های وب کاربرد دارد. | تابع Server.Execute(...);        |

یک صفحه را در یک پنجره جدید مرورگر نمایش می دهد. اگر کاربرد از برنامه هایی مانند pop-up stopper استفاده کند این متدهای کارآیی نخواهد داشت.

## استفاده از HyperLink و Redirection

با تنظیم کردن خاصیت NavigateURL کنترل HyperLink با کلیک کاربر بر روی این کنترل به صفحه ای مشخص شده، هدایت می گردد. این کنترل سبب انجام هیچگونه رخدادی در سمت سرور نمی گردد. در صورت نیاز به پردازش رخداد کلیک می توان از کنترل های LinkButton و یا ImageButton استفاده کرد. در این حالت می توان از تابع Response.Redirect برای هدایت کاربر به صفحه ای دیگر استفاده کرد.

## استفاده از متدهای Transfer

استفاده از این تابع یا متدهای شیوه ای استفاده از Redirect و یا استفاده از HeperLink می باشد با این تفاوت: Transfer می تواند بعضی از اطلاعات مربوط به صفحه ای اصلی را در بین درخواست ها، حفظ و نگهداری کند. تنظیم کردن آرگومان تابع True Transfer به سبب می شود که ViewState و QueryString و پروسیجر رخداد در فرم مقصد نیز مهیا باشند. برای استفاده از این حالت ابتدا باید خاصیت EnableViewStateMac فرم وب را False کنید. به صورت پیش فرض خواهد شد. برای دریافت این اطلاعات در صفحه ای دیگر نیز قابل خواندن ASP.NET اطلاعات ViewState را Hash می کند. با False کردن آن، این اطلاعات در صفحه ای دیگر نیز قابل خواندن سازگاری با ASP قدیمی قرار داده شده است.

توجه: متدهای Transfer و Execute تنها با فرم های وب کار می کنند. هر گونه سعی در استفاده از یک صفحه HTML معمولی با یک خطای زمان اجرا پاسخ داده خواهد شد.

### مثال 4:

هدف از این مثال یادگیری نحوه کار کردن با دستور Transfer می باشد. به این منظور یک پروژه جدید باز کنید. و یک فرم دیگر به آن اضافه کنید. روی فرم اول یک TextBox و یک دکمه قرار دهید. می خواهیم با کلیک کردن روی دکمه روی فرم اولی محتویات فرم اول به فرم دوم منتقل شود. به همین دلیل تنظیمات زیر را انجام دهید:

-1 در رخداد دکمه کد زیر را بنویسید که بتوان به صفحه دوم رفت:

```
.Server.Transfer("default2.aspx", true);
```

-2 در رخداد مربوط به Load\_Page فرم دومی کد زیر را وارد کنید که بتوان اطلاعات رسیده از فرم اول را روی فرم دوم نمایش داد.

```
Label1.Text = "received from webform1:" + Request.Form["TextBox1"].ToString();
```

-3 خاصیت EnableSessionState فرم اول را به False تغییر نماید.

## استفاده از متدهای Execute

با استفاده از متدهای Execute می توان فرم وب دوم را بدون ترک اولین فرم وب، پردازش کرد. این مورد اجازه می دهد نتایج را از یک فرم وب به ناحیه ای در همین صفحه جاری هدایت کنیم. همانند متدهای Transfer، باید صفحه False EnableViewStateMac شود.

هنگامیکه فرم های وب را با استفاده از متده Execute ترکیب می کنید، هر گونه رخداد Post-Back استفاده شده در فرم دوم سبب پاک شدن فرم اول می گردد. برای این منظور استفاده از این روش تنها هنگامی مفید است که فرم دوم حاوی کنترلی نباشد که رخداد Post-Back بی را سبب شود.

# فصل ششم

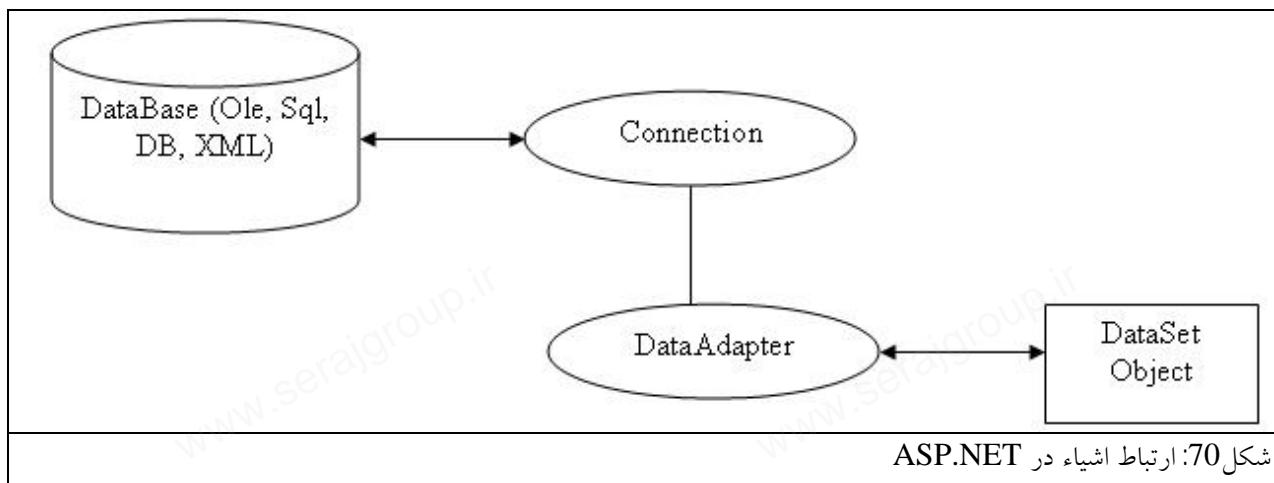
طریقه دستیابی و  
کار با داده ها در  
ASP.NET

VS.NET با استفاده از مجموعه ای از ابزارها و فضاهای نام که به آنها ADO.NET اطلاق می شود به پایگاه داده دسترسی دارد. طریقه دستیابی به داده ها در ADO.NET تقریباً مستقل از منبع داده است و پس از اینکه Connection و ارتباط ایجاد شد جدای از نوع پایگاه، طرز کار و رفتار با آنها بوسیله ی یک سری از اشیاء، خواص و روش ها موجود که برای تمام آنها یکسان است صورت می گیرد. در طی این فصل و فصول آتی نحوه ی اتصال و استفاده از بانکهای اطلاعاتی با استفاده از ابزارها ی جدید فراهم شده را خواهیم آموخت.

### درک پایه ای از ADO.NET

برای دستیابی به داده ها در ADO.NET سه لایه وجود دارد:

- 1- مکان فیزیکی نگهداری داده ها: می توان یک پایگاه داده OLE، پایگاه داده SQL و یا یک فایل XML باشد.
- 2- فراهم کننده ی داده: این مرحله شامل شی Connection و اشیاء Command است که نمایش دهنده سرویس حافظه ای داده ها هستند.
- 3- نمایش دهنده ی درون حافظه ای جداول و ارتباطات بین آنها است. پس از اینکه DataSet ایجاد شد اینکه از کجا آمده است و یا کجا ذخیره گشته است اهمیت ندارد. به این نوع معماری Disconnection هم می گویند زیر DataSet مستقل از ذخیره داده ها است.



دو نوع اتصال داده به پایگاه داده در ADO.NET وجود دارد:

- 1- استفاده از OleDbConnection برای اتصال به پایگاه داده محلی. کانکشن از نوع پایگاه داده Ole از شی OleDataAdapter برای انجام دستورات و بازگشت داده استفاده می شود.
  - 2- استفاده از SqlConnection برای اتصال به یک پایگاه داده بر روی سرور.
- ADO.NET خواص، اشیاء و متدها را به صورت سه فضای نام که در زیر بیان شده است، ارائه می دهد.

| فضای نام    | مواردی را که مهیا می کند                                                           |
|-------------|------------------------------------------------------------------------------------|
| System.Data | کلاس ها، نوع ها و سرویس ها برای ایجاد و دستیابی به DataSets و اشیاء مشتق شده از آن |

|                                                                 |                       |
|-----------------------------------------------------------------|-----------------------|
| نوع ها و کلاس هایی را برای دستیابی به پایگاه داده ای SQL-Server | System.Data.SqlClient |
| نوع ها و کلاس ها برای دستیابی به پایگاه داده OLE                | System.Data.OleDb     |

واضح است که می توان این فضای نام را با استفاده از کلمه **Using** به پروژه اضافه کرد.

برای دستیابی به یک پایگاه داده بوسیله **ADO.NET**, این مراحل را طی کنید:

1- ایجاد یک کانکشن به یک بانک اطلاعاتی با استفاده از شی کانکشن.

شی **Connection**: برای اتصال به پایگاه داده بکار بردہ می شود که قبلًا ذکر گردید که به دو صورت می باشد.

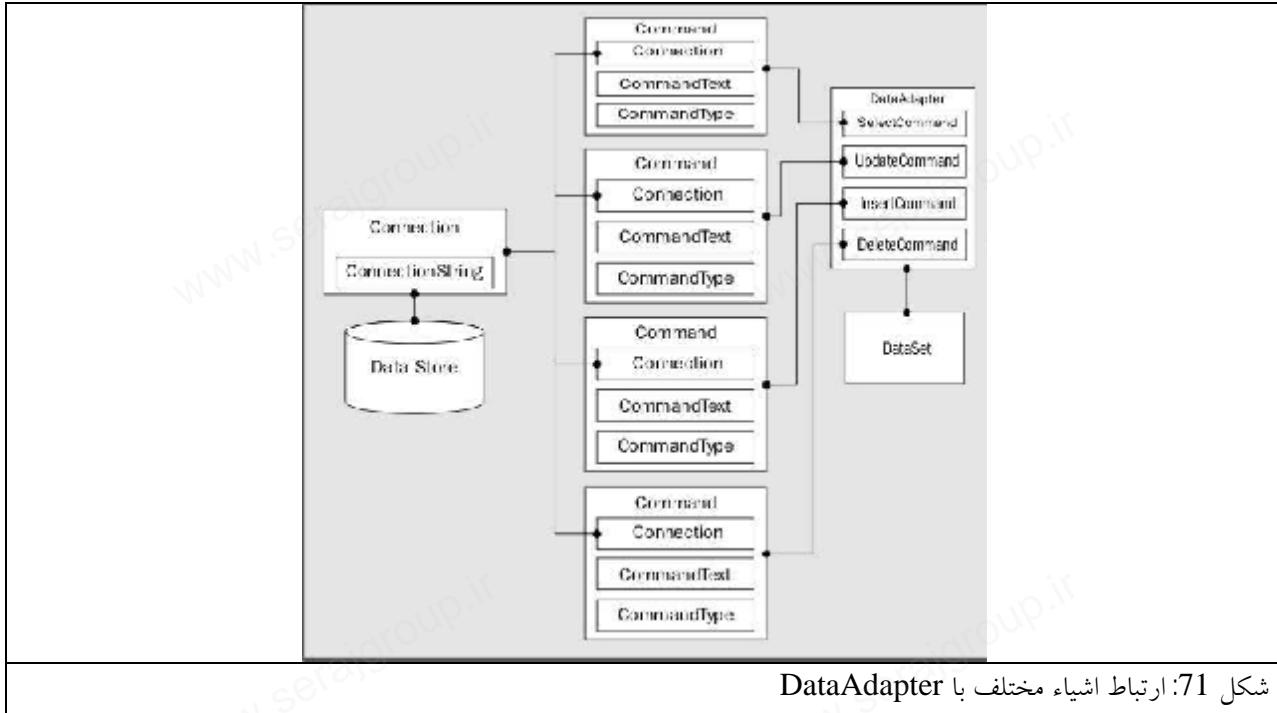
مهمترین متدهای این شی به این صورت می باشد:

| متدها                   | شرح                                                                                                                                     |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>Open</b>             | ارتباط داده شده با خصوصیات تنظیم شده باز می کند. خصوصیات تنظیم شده مانند <b>ConnectionString</b> که ارتباط را برای استفاده مشخص مس کند. |
| <b>Close</b>            | ارتباط را می بندد                                                                                                                       |
| <b>BeginTransaction</b> | شروع به اجرای تراکنش می کند و یک شی <b>Transaction</b> را بر می گرداند که برای قبول یا رد تراکنش می توان استفاده کرد.                   |

2- استفاده از یک دستور برای ایجاد یک **DataSet** با استفاده از شی **.Adapter**.

شی **DataAdapter**: برای متصل کردن یک یا چند شی **DataSet** به شی **Command** بکار بردہ می شود. که به دو

صورت می توان آن را تعریف کرد: **OleDbDataAdapter** و **SqlDataAdapter**. این اشیا چهار خصوصیت تعریف شده برای دستکاری داده ها مورد استفاده قرار می دهند: **SelectCommand**, **InsertCommand**, **UpdateCommand** و **DeleteCommand**



شکل 71: ارتباط اشیاء مختلف با DataAdapter

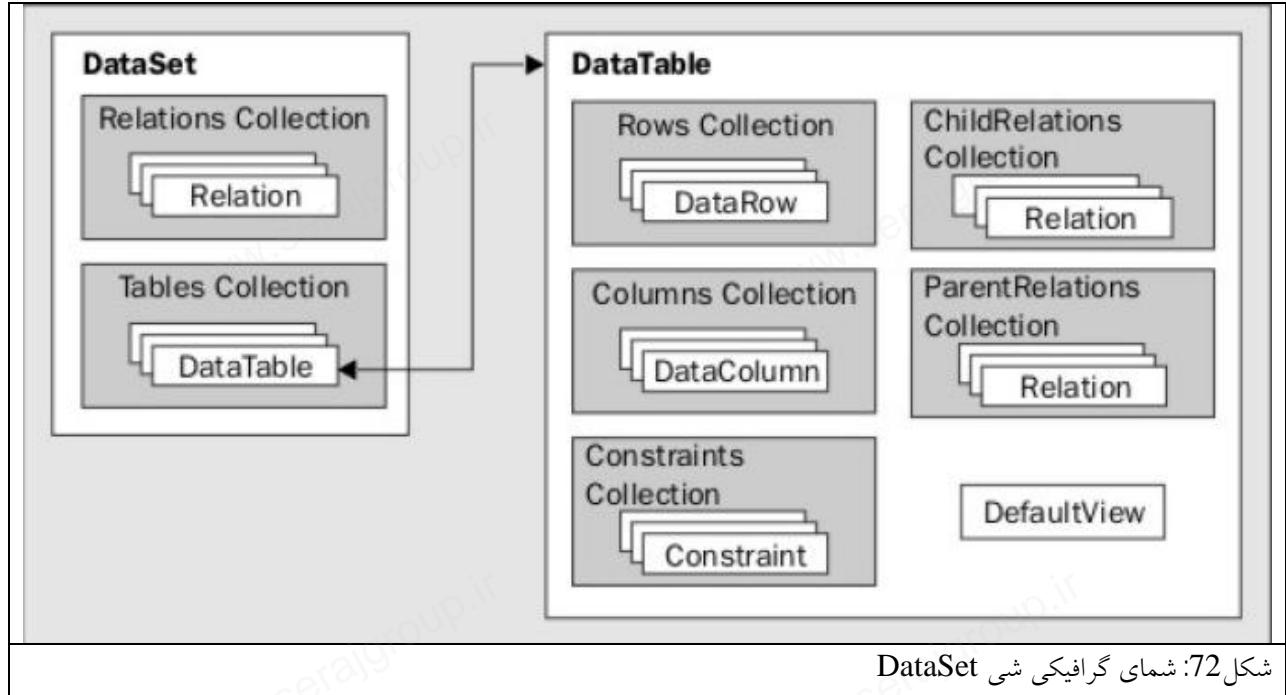
مهمترین متد مورد استفاده در این اشیاء به شرح زیر می باشد:

| شرح                                                                                                                       | متد    |
|---------------------------------------------------------------------------------------------------------------------------|--------|
| اجرای SelectCommand برای پر کردن شی DataSet با داده های که در DataSource مشخص شده است.                                    | Fill   |
| بطور بازگشتنی نوایع DeleteCommand, UpdateCommand, InsertCommand برای هر اضافه شدن، حذف شدن، ویرایش شدن یک ردیف در DataSet | Update |

3- استفاده از از شی DataSet در کد برای نمایش داده ها یا تغییر داده ها در پایگاه داده.

شی DataSet برای فراهم کردن داده های بدون ارتباط با پایگاه داده را استفاده می شود و همچنین کار کردن با این داده ها رابطه ای را فراهم می کند. ما این شی را از داده های ذخیره شده پر می کیم. بدون ارتباط با داده های ذخیره شده با آن کار می کنیم. سپس می توانیم مجدداً با داده های ذخیره شده ارتباط برقرار کنیم و اگر خواستیم داده های تغییر داده شده در DataSet را جایگزین کنیم. هر جدول در DataSet به یک شی DataTable درون TablesCollection نسبت داده می شود.

هر شی DataTable شامل یک مجموعه شی DataColumn و مجموعه DataRow می باشد



شکل 72: شمای گرافیکی شی DataSet

#### متدهای اساسی شی DataSet

شی DataSet دارای مجموع ای از متدهای است که برای کار کردن با محتویات جداول یا روابط بین آنها بکار برده می شود. برای مثال، ما می توان یک DataSet را خالی کنیم با داده ها را از دو DataSet با هم ادغام نمایم:

| متدها | شرح متدها                                                 |
|-------|-----------------------------------------------------------|
| Clear | همه داده های داخل DataSet را برمی دارد.                   |
| Merge | محتویات یک DataSet را با یک DataSet دیگر ادغام می نمایید. |

4- اجرای دستورات برای به روز رسانی پایگاه داده از شی Adapter با استفاده از شی DataSet

5- بستن کانکشن- اگر شما به صورت صریح در مرحله 2 با استفاده از متدهای Open آنرا گشوده اید. استفاده از دستورات بدون اجرای دستور Open در ابتدا به صورت ضمنی سبب باز و بسته شدن هر کانکشن با هر درخواست می شود.

در ادامه تمام این مراحل به صورت مفصلی توضیح داده خواهد شد. عموماً در محصولات مایکروسافت دو روش برای کار با پایگاه داده وجود دارد: 1- استفاده از ابزارهای ویژوال در زمان طراحی. 2- کدنویسی مستقیم.

با توجه به اینکه روش اول فقط برای تازه کاران جذاب و مفید می باشد و هیچگونه دید عمیقی را از کار ارائه نداده و با کوچکترین خطایی در برنامه رفع آن واقعاً مشکل می باشد از آن استفاده نخواهد شد. برای مثال در روش اول شما با استفاده از ابزارهای ویژوال می توانید یک ConnectionString را برای اتصال به پایگاه داده ایجاد کنید ولی بدیهی است که با کدنویسی انعطاف پذیری و توانایی بیشتری وجود خواهد داشت و واقعاً کسی مبحث را درست درک کرده که بتواند برای این مجموعه کد نویسد.

## نمايش داده های SQL-Server

کنترل DataGrid برای کار با داده ها بسیار انعطاف پذیر می باشد. آن از ویژگی های پیشرفته ای مانند Paging ویرایش داده ها و مرتب کردن داده ها بهره می برد که مباحث مفصل تر آن و همچنین سایر کنترل های اینگونه در فصلی جداگانه ارائه خواهد گردید. در این فصل برای معرفی مفاهیم کار با داده ها صرفاً از آن برای نمايش داده ها استفاده خواهیم کرد. برای نمايش داده ها حداقل از سه شیء استفاده می شود: SQLConnection، SQLDataAdapter و SQLDataSet. شیء SQLDataSet در فضای نام System.Data.SqlClient در بکار برده می شود. شیء SQLDataAdapter نیز در همین فضایی نام قرار داشته و بیانگر کانکشن و دستوراتی است که روی پایگاه داده اجرا می شوند. شیء SQLDataSet در فضای نام System.Data موجود بوده و SQLDataAdapter را بکار می گیرد تا داده ها را از منبع داده ای SQL-Server بدست آورد.

### مثال 1:

می خواهیم از بانک اطلاعاتی همراه SQL-Server به نام Pubs رکوردهای جدول Titles را بخوانیم و در یک GridView نمايش دهیم.

ابتدا فضاهای نام System.Data.SqlClient و System.Data در یک GridView روی صفحه قرار دهید. سپس بر روی صفحه دوبار کلیک نموده و کد زیر را داخل آن بنویسید.

```
protected void Page_Load(object sender, EventArgs e)
{
 SqlConnection sqlcon = new SqlConnection
 ("Data Source=(local);Initial Catalog=pubs;Integrated Security=True");
 SqlDataAdapter sqldata1 = new SqlDataAdapter("select * from titles",sqlcon);
 DataSet datset1 = new DataSet();
 sqldata1.Fill(datset1);
 GridView1.DataSource = datset1;

 GridView1.DataBind();
}
```

شکل 73: کد رخداد مربوط به Load صفحه

یک بار ایجاد ارتباط با بانک اطلاعاتی باز می کنیم. در ConnectionString آن همانطور که ملاحظه می کنید، نام سرور و نام پایگاه داده را که برنامه می خواهد به آن متصل شود را مشخص می کنیم.

```
SqlConnection sqlcon = new SqlConnection("Data Source=(local);Initial Catalog=pubs;Integrated Security=True");
```

سپس با ایجاد یک SQLDataAdapter و پاس کردن یک عبارت SQL به آن، برای دریافت داده ها اقدام می نماییم و در ادامه یک شیء DataSet ایجاد می کنیم:

```
SqlDataAdapter sqldata1 = new SqlDataAdapter("select * from titles",sqlcon);
DataSet datset1 = new DataSet();
sqldata1.Fill(datset1);
```

SQLDataAdapter را اضافه میکنیم تا DataSet را پیمايش کند. DataSet مربوط به دیتاگرید را به

ایجاد شده تنظیم می کنیم و در نهایت دیتاگرید را به دیتابست متصل می نماییم.

```
GridView1.DataSource = datset1;
```

## اضافه کردن داده ها به بانک اطلاعاتی SQL-Server

کار کردن با بانک های اطلاعاتی بدون آشنایی با دستورات SQL امکان پذیر نمی باشد. اگر با دستور Insert آشنا بی داشته باشید درج کردن داخل یک بانک اطلاعاتی خیلی ساده می شود.

Insert هنگامیکه Command ایجاد شد، با استفاده از شیء Connection می توان دستور SQL را برای اجرای Connection کردن داده ها مورد استفاده قرار داد و نهایتاً Connection باید بسته شود.

### مثال 2 :

در این مثال می خواهیم به جدول titles از پایگاه داده pubs داده اضافه کنیم به همین دلیل روی صفحه چندین TextBox و چندین Label قرار می دهیم. و همچنین روی صفحه یک دکمه می گذاریم و در رخداد Clieck مربوطه کد مربوط به وارد کردن این اطلاعات را وارد می کنیم. این کد شامل ایجاد یک اتصال و ایجاد یک شی Command می باشد و همچنین تنظیم کردن خصوصیات CommandType و Sqlcommand و سپس باز کردن اتصال، اجرای Command بستن اتصال می باشد.

شی Command اجازه اجرا کردن دستورات SQL یا Stored procedure را روی داده ها به ما می دهد. مجموعه ای از ردیف های داده و تعداد آن را ب می گرداند و آن را به شی DataReader یا DataAdapter می دهد. مهمترین متدهای Update Insert ExecuteNonQuery می باشد که برای اجرای دستورات SQL که مقداری را برنمی گرداند مانند Delete بکار برده می شود.

|                                       |                      |
|---------------------------------------|----------------------|
| Title_Id                              | <input type="text"/> |
| Title                                 | <input type="text"/> |
| Type                                  | <input type="text"/> |
| pub_id                                | <input type="text"/> |
| Price                                 | <input type="text"/> |
| Advance                               | <input type="text"/> |
| <input type="button" value="Button"/> |                      |

شكل 74: فرم مثال دوم

```
SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=pubs;Integrated Security=True");
 SqlCommand sqlcom= new SqlCommand();
 string sqlstr="INSERT INTO titles
(title_id,title,type,pub_id,price,advance) VALUES
(" +TextBox1.Text+ "," +TextBox2.Text + "','"+TextBox3.Text + "','" +TextBox4.Text +
"', "+TextBox5.Text + "," +TextBox6.Text+ ")";
 sqlcom.CommandText = CommandType.Text;
```

```
sqlcom.Connection = sqlcon1;
sqlcon1.Open();
sqlcom.ExecuteNonQuery();
sqlcon1.Close();
```

شکل 75: کد مربوط به رخداد کلیک کردن دکمه

## به روز رسانی داده ها و ویرایش آنها

با اجرای دستور SQL Update می توان این کار را انجام داد. نحوه کار هم دقیقاً مشابه عملیات Insert می باشد. به همین دلیل تشریح این کار را با یک مثال دنبال می کنیم.

### مثال 3:

هدف از این مثال آشنایی با دستور SQL Update، ایجاد و استفاده از پروسیجر می باشد به همین دلیل می خواهیم در این مثال هنگامیکه در جدول titles از پایگاه داده هماره pub، فیلد Title\_ID مساوی یک مقدار ورودی باشد مقدار قیمت آن را به 35.00 تغییر دهیم و قبل و بعد از عملیات به روز رسانی این جدول را در یک GridView نمایش دهیم. برای انجام این مثال روی فرم یک TextBox و یک Label و یک Button در قسمت Gridview این دستورات زیر ایجاد می کنیم:

```
public void showdata()
{
 در داخل آن دستورات لازم برای نمایش داده های جدول titles را درون آن می نویسیم به همان صورت که در مثال
 اول این کار را انجام دادیم:
 ابتدا مانند مثالهای قبل فضای نام لازم را به پروژه اضافه می کنیم.
```

```
using System.Data.SqlClient;
 یک Connection را به پایگاه داده Pubs برقار می کنیم:
 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=pubs;Integrated Security=True");
 در ادامه شی SqlDataAdapter را ایجاد و با یک عبارت SQL داده های که می خواهیم نمایش داده شوند را
 انتخاب می کنیم.
```

```
SqlDataAdapter sqldat1 = new SqlDataAdapter("select title,title_id,price from
titles", sqlcon1);
 بعد یک DataTable برای گرفتن داده ها از SqlDataAdapter ایجاد کرده آن را به DataGridView می دهیم:
```

```
DataTable datatabl = new DataTable();
sqldat1.Fill(datatabl);
GridView1.DataSource = datatabl;
GridView1.DataBind();
```

در رخداد LoadForm این تابع را فراخوانی می کنیم. در ادامه برنامه در رخداد کلیک کردن دکمه دستورات مربوط به بروز رسانی را مانند زیر می نویسیم :

| <b>title</b>                                                    | <b>title_id</b> | <b>price</b> |
|-----------------------------------------------------------------|-----------------|--------------|
| The Busy Executive's Database Guide                             | BU1032          | 19.9900      |
| Cooking with Computers: Surreptitious Balance Sheets            | BU1111          | 11.9500      |
| You Can Combat Computer Stress!                                 | BU2075          | 2.9900       |
| Straight Talk About Computers                                   | BU7832          | 19.9900      |
| Silicon Valley Gastronomic Treats                               | MC2222          | 19.9900      |
| The Gourmet Microwave                                           | MC3021          | 35.0000      |
| The Psychology of Computer Cooking                              | MC3026          | 35.0000      |
| But Is It User Friendly?                                        | PC1035          | 22.9500      |
| Secrets of Silicon Valley                                       | PC8888          | 20.0000      |
| Net Etiquette                                                   | PC9999          |              |
| Computer Phobic AND Non-Phobic Individuals: Behavior Variations | PS1372          | 21.5900      |
| Is Anger the Enemy?                                             | PS2091          | 10.9500      |
| Life Without Fear                                               | PS2106          | 7.0000       |
| Prolonged Data Deprivation: Four Case Studies                   | PS3333          | 19.9900      |
| Emotional Security: A New Algorithm                             | PS7777          | 7.9900       |
| Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean | TC3218          | 20.9500      |
| Fifty Years in Buckingham Palace Kitchens                       | TC4203          | 11.9500      |
| Sushi, Anyone?                                                  | TC7777          | 14.9900      |

Input:

شکل 75: اجرای مثال سوم

ابتدا مانند مثال دوم به ترتیب کارهای زیر را انجام می دهیم:

1- ایجاد بک Connection

```
SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=pubs;Integrated Security=True");
```

2- ایجاد بک SqlCommand

```
SqlCommand sqlcom1 = new SqlCommand();
```

3- تنظیم کردن SqlCommand

```
sqlcom1.CommandText = "UPDATE titles SET price=35.00 where title_id = '" + TextBox1.Text + "'";
sqlcom1.CommandType = CommandType.Text;
sqlcom1.Connection = sqlcon1;
```

4- باز کردن Connection

```
sqlcon1.Open();
```

5- اجرای SqlCommand

```
sqlcom1.ExecuteNonQuery();
```

6- بستن Connection

```
sqlcon1.Close();
```

بعد از این عملیات ها رویه showdata() را فراخوانی می کنیم که داده های تغییر داده شده را نشان دهد.

## حذف اطلاعات از جدول

مهمترین و معمولیترین حالت حذف اطلاعات معمولاً مربوط به مدیریت پایگاه داده می شود و بهتر است کاربران دسترسی به یک چنین امکاناتی را نداشته باشند.

روال کار در اینجا نیز همانند اجرای سایر دستورات SQL ذکر شده می باشد و در ابتدا با شی SQLConnection برای ایجاد یک ارتباط به پایگاه داده شروع می کیم. پس از ایجاد ارتباط یک شی SqlCommand ایجاد می شود و یک عبارت SQL که باید روی پایگاه داده اجرا شود. در ادامه، ارتباط باز، دستور اجرا شده و ارتباط بسته می شود.

### :4 مثال

می خواهیم به مثال قبل یک دکمه اضافه کنیم بطوریکه اگر روی دکمه کلیک کردیم داده که آن برابر ورودی باشد. حذف شود. بنابراین بطور خلاصه دستورات زیر را در رخداد کلیک کردن دکمه می نویسیم:

```
SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=pubs;Integrated Security=True");
SqlCommand sqlcom1 = new SqlCommand();
sqlcom1.CommandText = "Delete from titles where title_id=' " + TextBox1.Text +
" '";
sqlcom1.CommandType = CommandType.Text;
sqlcom1.Connection = sqlcon1;
sqlcon1.Open();
sqlcom1.ExecuteNonQuery();
sqlcon1.Close();
showdata();
```

توجه چون جدول titles در پایگاه داده Pubs با دیگر جداول رابطه دارد نمی توان بدین صورت مقادیر داخل آن را حذف کرد به همین دلیل باید این رابطه ها را از آن برداریم. برای برداشتن این رابطه ها باید بدین صورت عمل کرد. در قسمت Design جدول titles وارد قسمت مربوط به ManageRelationships شده در این قسمت تمام رابطه ها با دیگر جداول را قطع می کنیم.

شکل بی روح GridView را می توان بطور دینامیکی تغییر داد. برای انجام این کار روی GridView کلیک راست کنید و از منوی که ظاهر می شود می توان گزینه ... Auto Formating را انتخاب کرد و از شکل های آماده استفاده کرد.

| <b>title</b>                                                    | <b>title_id</b> | <b>price</b> |
|-----------------------------------------------------------------|-----------------|--------------|
| The Busy Executive's Database Guide                             | BU1032          | 19.9900      |
| Cooking with Computers: Surreptitious Balance Sheets            | BU1111          | 11.9500      |
| You Can Combat Computer Stress!                                 | BU2075          | 2.9900       |
| Straight Talk About Computers                                   | BU7832          | 19.9900      |
| Silicon Valley Gastronomic Treats                               | MC2222          | 19.9900      |
| The Gourmet Microwave                                           | MC3021          | 35.0000      |
| The Psychology of Computer Cooking                              | MC3026          | 35.0000      |
| But Is It User Friendly?                                        | PC1035          | 22.9500      |
| Secrets of Silicon Valley                                       | PC8888          | 20.0000      |
| Net Etiquette                                                   | PC9999          |              |
| Computer Phobic AND Non-Phobic Individuals: Behavior Variations | PS1372          | 21.5900      |
| Is Anger the Enemy?                                             | PS2091          | 10.9500      |
| Life Without Fear                                               | PS2106          | 7.0000       |
| Prolonged Data Deprivation: Four Case Studies                   | PS3333          | 19.9900      |
| Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean | TC3218          | 20.9500      |
| Fifty Years in Buckingham Palace Kitchens                       | TC4203          | 11.9500      |
| Sushi, Anyone?                                                  | TC7777          | 14.9900      |

Input:

شکل 76: نمایی از مثال 4

### مرتب کردن داده های یک پایگاه داده

برای مرتب کردن داده بر حسب یکی یا چند تا از فیلدها که کاری اجتناب ناپذیر، موقع سرو کار داشتن با داده ها زیاد می باشد، چندین راه حل وجود دارد:

- 1- استفاده کردن از دستورات SQL
- 2- استفاده کردن از قابلیت های DataSet
- 3- استفاده کردن از قابلیت های GridView

### مثال 5

در این مثال می خواهیم باز هم مثال قبلی را گسترش داده دو دکمه به آن اضافه کرده و در رخداد کلیک هر کدام به یکی از شیوه بالا داده را مرتب کنیم. در رخداد کلیک دکمه اول (SortSQL) کد مربوط به تابع showdata() را می آوریم با این تفاوت که شی SqlDataAdapter بصورت ذیل تغییر می دهیم:

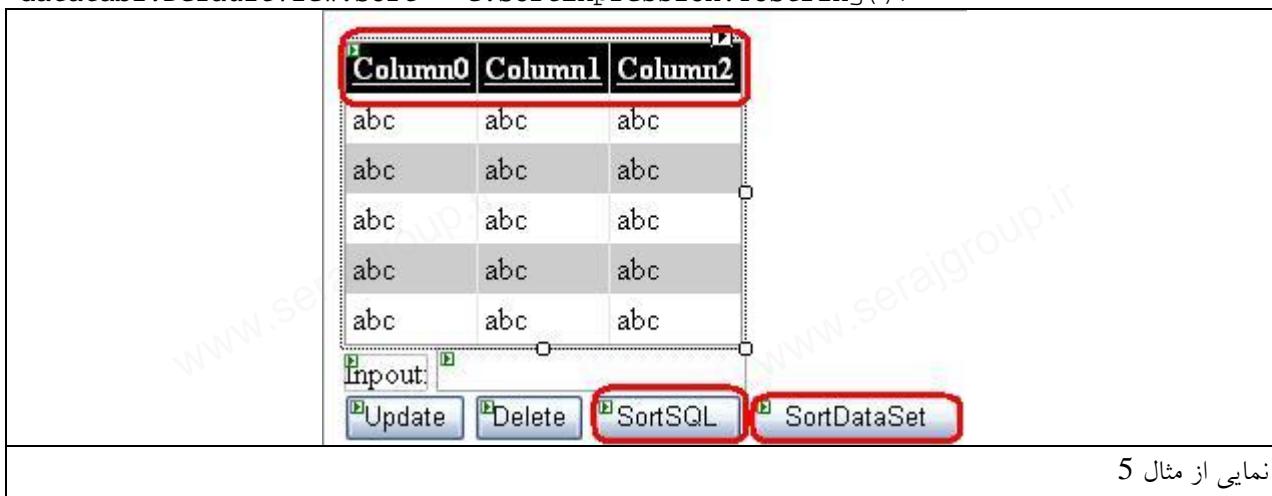
```
SqlDataAdapter sqldat1 = new SqlDataAdapter("select title,title_id,price from titles order by title", sqlcon1);
```

در رخداد کلیک دکمه دوم (SortDataSet) کد مربوط به تابع showdata() را دوباره می آوریم و دستور زیر را بعد از Fill کردن DataSet می آوریم:

```
datatabl.DefaultView.Sort = "title";
```

اما نحوه استفاده کردن از قابلیت سوم بدین صورت می باشد که اگر روی عنوان هر ستون GridView کلیک کردیم بر اساس آن عنوان مرتب شود، به همین دلیل خاصیت AllowSort GridView را True کترول کرده و در رخداد Sorting کترول GridView کد رخداد کلیک کردن دکمه SortDataSet را با تفاوت زیر می آوریم:

```
datatabl.DefaultView.Sort = e.SortExpression.ToString();
```



تا اینجا ما اکثر کارهای لازم برای کار با پایگاه داده ها را بیان نمودیم در ادامه ما موارد تکمیلی از جمله نحوه اجرا کردن رویه های ذخیره شده، انجام عملیات های ریاضی روی ستون ها را برای کار کردن با پایگاه داده بیان خواهیم کرد.

### اجرای رویه های ذخیره شده (Stored Procedure)

استفاده از رویه های ذخیره شده در برنامه ها، برنامه ای سریعتر و امن تر نسبت به حالتی که دستورات SQL به صورت مستقیم روی دیتابیس اجرا می شوند را ایجاد می کند. رویه های ذخیره شده دستورات SQL و پیش کامپایل شده ای بوده و در حافظه سرور پایگاه داده شما Cache خواهد شد. در مثال زیر نحوه استفاده از آن بیان خواهد شد.

#### مثال 6:

هدف از این مثال ایجاد بک پایگاه داده در SQLServer، نحوه ایجاد رابطه، ایجاد یک StoredProcedure و نحوه فراخوانی در یک پروژه ASP.NET می باشد. به همین منظور یک بانک اطلاعاتی جدید به نام MYTestDB ایجاد کنید با سه جدول به صورتی که تداعی جداول یک پایگاه داده انبار ساده باشد. یکی از جداول برای ذخیره مشخصات کالا (Entity)، جدول دوم برای ذخیره کالاهای ورود به انبار (IEntity) و جدول سوم برای ذخیره کالاهای خروج (OEntity) از انبار باشد. می خواهیم از این دو جدول گزارشی تهیه کنیم که به ازای هر تفاضل تعداد ورودی و تعداد خروج هر کالا به صورت یک ستون واحد در یک دیتا گرید نمایش داده شود. بهترین، مطمئن ترین و سریع

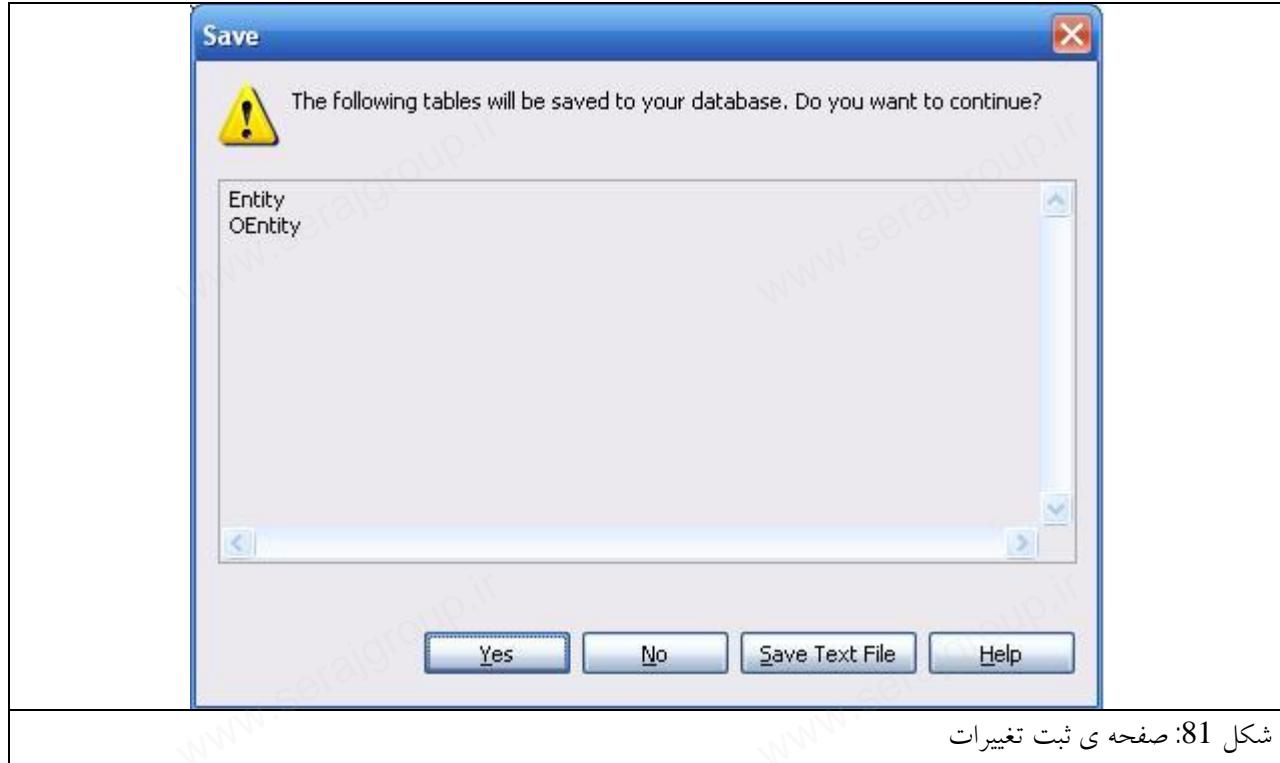
ترین راه برای این نوع مثال ها استفاده از رویه های ذخیره شده می باشد. برای ایجاد این پایگاه داده Enterprise Manager را اجرا کرده در قسمت Database Store یک پایگاه داده یعنام Store ایجاد می کنیم. و در این پایگاه داده سه جدول گفته شده را ایجاد می کنیم بصورت زیر:

شکل 78: مشخصات جداول در حال Design

برای ایجاد رابطه بین این جداول در صفحه Design جداول کلیک راست کرده گزینه Relationship را انتخاب کنید. در صفحه ی ظاهر شده روی New کلیک کنید و تنظیمات صفحه را مانند شکل های داده شده انجام دهید.

|                                                |                                                |
|------------------------------------------------|------------------------------------------------|
| <b>شکل 80:</b> رابطه بین جدول OEntity و Entity | <b>شکل 79:</b> رابطه بین جدول Entity و IEntity |
|------------------------------------------------|------------------------------------------------|

و در آخر پس از بستن این صفحه دیالوگ، صفحه ی نمایش ثبت تغییرات ظاهر خواهد شد. برای ایجاد رویه ذخیره شده، می توان به دو روش این کار را انجام داد: 1- در Query Analyzer -2- در Enterprise Manager



شکل 81: صفحه‌ی ثبت تغییرات

در این مثال ما از روش دوم استفاده کرده و رویه ذخیره شده زیر را ایجاد می‌کنیم که کارش حساب کردن تقاضل ورودی، خروجی هر کالا می‌باشد:

```
create procedure rptDiff
as
select Entity.Eid,(IResult.Isum-OResult.Osum)
from
) select Eid,sum(EI_No) as Isum from IEntity Group by(Eid)) as IResult,
) select Eid,sum(OE_No) as Osum from OEntity Group by(Eid)) as OResult,
Entity
where Entity.Eid=IResult.Eid and Entity.Eid=OResult.Eid
return
```

برای تست کردن آن هم می‌توانید از دستور زیر استفاده کنید:

```
exec rptdiff
```

برای اینکه بتوان با پایگاه داده فوق کار کرد می‌توان یک سری داده را خیلی سریع در محیط وارد نمود.

برای استفاده از این Stored Procedure یک فرم ایجاد کرده روی فرم یک GridView قرار داده و در رخداد بار شدن فرم کد زیر را برای متصل کردن این GridView به Stored Procedure وارد می‌کنیم:

```
protected void Page_Load(object sender, EventArgs e)
{
 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial
```

```

 Catalog=Store;Integrated Security=True");
SqlDataAdapter sqldata1 = new SqlDataAdapter("rptDiff", sqlcon1);
sqldata1.SelectCommand.CommandType = CommandType.StoredProcedure;
DataTable datatabl = new DataTable();
sqldata1.Fill(datatabl);
GridView1.DataSource = datatabl;
GridView1.DataBind();
}

```

## مثال 7:

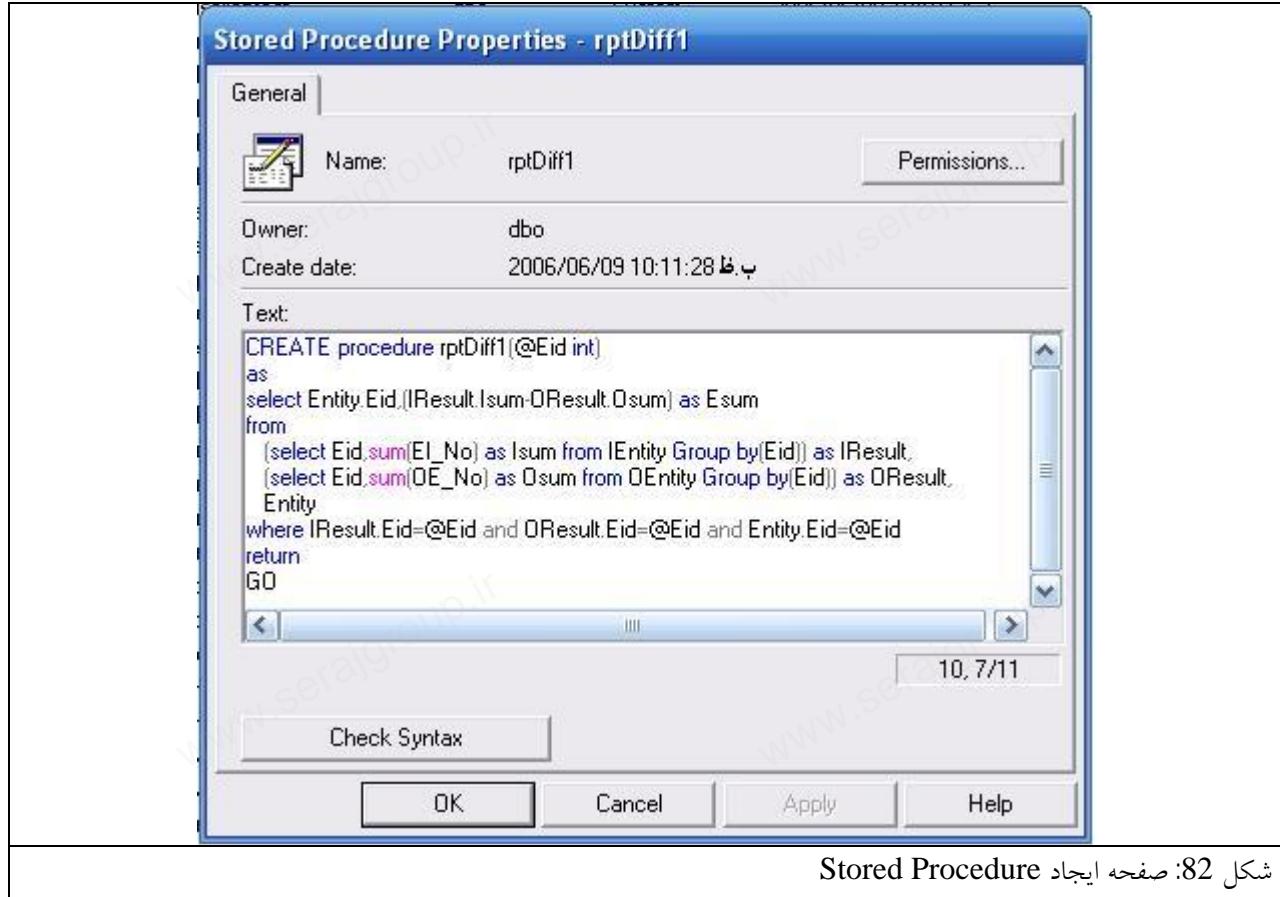
در این مثال می خواهیم نحوه ی پاس کردن متغیرها و ورودی های فرم را به یک رویه ذخیره شده ببریس کنیم. فرض کنیم در پایگاه داده MyTestDB که آنرا در مثال قبل ایجاد کردیم می خواهیم با دادن Eid تفاضل تعداد ورودی و تعداد خروج آنرا بدست آوریم و روی صفحه نمایش دهیم: در ابتدا رویه ذخیره شده زیر را ایجاد نمایید:

```

CREATE procedure rptDiff1(@Eid int)
as
select Entity.Eid,(IResult.Isum-OResult.Osum) as Esum
from
 (select Eid,sum(EI_No) as Isum from IEntity Group by(Eid)) as IResult,
 (select Eid,sum(OE_No) as Osum from OEntity Group by(Eid)) as OResult,
 Entity
where IResult.Eid=@Eid and OResult.Eid=@Eid and Entity.Eid=@Eid
return

```

برای اینکار علاوه بر Query Analyzer می توان از محیط Enterprise Manager نیز استفاده کرد. در لیست درختی مربوط به پایگاه داده MyTestDb روی گزینه **I** Stored Procedure کلیک نمایید تا تمام رویه های ذخیره شده مربوط به بانک اطلاعاتی خودتان را مشاهده نمایید. سپس روی صفحه آن کلیک راست نماییدو گزینه **New Stored Procedure** را انتخاب نمایید. در صفحه **I** باز شده کد بالا را نوشته آن را ذخیره نمایید.



شکل 82: صفحه ایجاد Stored Procedure

برای تست کردن آن هم می توان مثل دفعه قبل در Query analyzer عبارت زیر را وارد کرد:

Exec rptDiff1

برای استفاده کردن از این رویه ذخیره شده روی فرم مثال قبلی یک دکمه و یک TextBox روی فرم قرار داده و در رخداد کلیک دکمه کد زیر وارد کرده بطوریکه اختلاف ورودی و خروجی ورودی TextBox را بدست می آورد:

```
SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=Store;Integrated Security=True");
SqlDataAdapter sqldata1 = new SqlDataAdapter("rptDiff1", sqlcon1);
sqldata1.SelectCommand.CommandType = CommandType.StoredProcedure;
sqldata1.SelectCommand.Parameters.Add("@EId", SqlDbType.Int);
sqldata1.SelectCommand.Parameters["@EId"].Value = TextBox1.Text;
DataTable datatabl = new DataTable();
sqldata1.Fill(datatabl);
GridView1.DataSource = datatabl;
GridView1.DataBind();
```

#### انجام عملیات ریاضی روی فیلدها:

گاهی از اوقات لازم است جمع کل عدهای موجود در یک ستون (فیلد) را محاسبه، یا میانگین ها و امثال اینگونه عملیات را بدست آوریم. یکی از راه حل ها آن بدین صورت است که کل اعداد فیلد را بخوانیم و سپس عملیاتهای ریاضی روی آن انجام دهیم، راه دیگر استفاده از دستورات مخصوص SQL برای اینگونه کارها می باشد. مثال بعدی نحوه انجام اینگونه عملیات را بیان می کند.

هدف از این مثال استفاده از دستورات SQL برای انجام عملیات‌های ریاضی می‌باشد. در این مثال می‌خواهیم بزرگترین عدد موجود در فیلد EI\_No که در مثال اول ایجاد شد را بدست آوریم. برای انجام این کار ما از متدهای ExecuteScalar() از SqlCommand که اولین ستون از اولین سطر مربوط به اجرای Query را برمی‌گرداند استفاده می‌کنیم.

```
protected void Button2_Click(object sender, EventArgs e)
{
 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=Store;Integrated Security=True");
 SqlCommand sqlcom1 = new SqlCommand("select MAX(EI_No) from IEntity",
 sqlcon1);
 sqlcon1.Open();
 int intRes = (int)sqlcom1.ExecuteScalar();
 sqlcon1.Close();
 Response.Write(intRes);
}
```

### استفاده از پایگاه داده های OLEDB

برای استفاده از پایگاه داده های OLEDB مانند اکسیس در ADO.NET باید از فضای نام System.Data.OleDb استفاده کرد. با استفاده از کلاس OleDbCommand می‌توان یک سری از دستورات SQL مانند Select، Insert، Update و Delete و همچنین اجرای روش‌های ذخیره شده را انجام داد.

در این مثال می‌خوایم نحوه کار کردن با بانک‌های اطلاعاتی Access را یاد بگیریم در این مثال قصد داریم اطلاعات را درون بانک اطلاعاتی اکسیس ذخیره کنیم. برای این کار مانند مثال‌های قبلی یک پایگاه داده ایجاد می‌کنیم اما این بار در محیط Access این کار را انجام خواهیم داد.

ابتدا Access XP را باز کنید و سپس از منوی فایل گزینه‌ی New را انتخاب کنید و از پنل سمت راست صفحه روی گزینه‌ی Blank DataBase کلیک نمایید تا یک بانک جدید خالی برای مثال به نام MyTestDb.mdb (مانند پایگاه داده مثال قبل) ایجاد شود. در صفحه‌ی دیالوگ باز شده، روی آیتم Create Table in design View کلیک نمایید تا صفحه‌ی طراحی پایگاه داده که شبیه محیط طراحی پایگاه داده در SQLServer است باز شود. سپس جدول را طراحی کنید. یادتان باشد که فیلد کلید ار یادتان نزود که مشخص کنید که اگر مشخص نکنید Access خود یک فیلد کلید به آن اضافه می‌کند. سپس پنجره را ببندید تا صفحه ذخیره کردن نام جدول ظاهر شود و نام را وارد نمایید.

به پروژه مثال قبلی یک دکمه، دو TextBox و دو برچسب اضافه کرده، در رخداد کلیک کردن این دکمه اضافه کردن اطلاعات به جدول Entity این پایگاه داده را می‌نویسیم و بعد از اضافه کردن محتویات آن را نمایش می‌دهیم. قبل از کد نوشتن در این رخداد فضای نام زیر را وارد می‌کنیم.

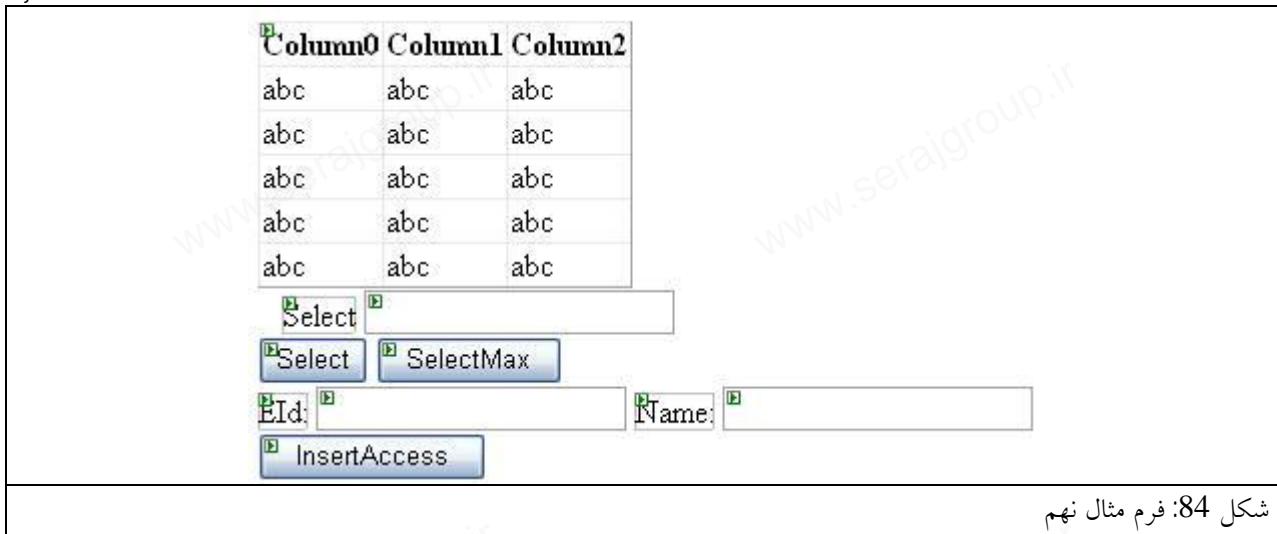
```
using System.Data.OleDb;
dr ضمن ما پایگاه داده که ایجاد می‌کنیم در مسیر پروژه در پوشه App_Data قرار می‌دهیم. در آخر کد زیر را در رخداد دکمه مورد نظر اضافه می‌کنیم:
```

```
protected void Button3_Click(object sender, EventArgs e)
```

```

 string FilePath;
 FilePath = Server.MapPath("App_Data/MyTestDb.mdb");
 OleDbConnection olecon1=new
 OleDbConnection("provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +FilePath);
 string strsqli = "INSERT INTO Entity(EId,name) VALUES(" +TextBox2.Text
 +" , "+TextBox3.Text+")";
 OleDbCommand olecoml=new OleDbCommand(strsqli,olecon1);
 olecon1.Open();
 olecoml.ExecuteNonQuery();
 olecon1.Close();
 OleDbDataAdapter oledatal=new OleDbDataAdapter("select * from
Entity",olecon1);
 DataTable datat1=new DataTable();
 oledatal.Fill(datat1);
 GridView1.DataSource=datat1;
 GridView1.DataBind();
 }
}

```



شکل 84: فرم مثال نهم

در مثال فوق از همان روش قدیمی ASP برای مشخص کردن مسیر فیزیکی فایل mdb اکسس توسط مند Server.MapPath که یک مسیر مجازی از پروژه را می گیرد استفاده کرده ایم. همانطور که ملاحظه می فرمایید همه چیز مانند قبل است فقط بجای OleDb عبارت Sql قرار گرفته است و تمام توضیحات آنها هم تکراری می باشد می خواهیم تک رکوردهای جدول مربوط به بانک اطلاعاتی اکسس را که در طی مثال قبل ایجاد کرده ایم ، در برنامه خوانده و آنرا در یک جدول با استفاده از تگهای ایجاد می کنیم ، نمایش می دهیم . در ادامه می خوایم بک شی به نام OleDbDataReader را معرفی کنیم. این شی مانند یک Recordset فقط خواندنی سیستم قبلی استفاده می شود. در مثال زیر نحوه استفاده از این شی آمده است.

## مثال 10

در این برنامه می خواهیم تمام اطلاعات داخل جدول Entity از پایگاه داده MyTestDb را خوانده و نمایش دهیم اما بدون استفاده از شی GridView. در این مثال بجای استفاده از شی GridView ما تک تک رکوردها را خوانده در یک جدول که بصورت دینامیکی اضافه می کنیم نمایش می دهیم. یک پروژه جدید ایجاد کرده و برای انجام این کار در ابتدا فایل

مریبوط به پایگاه داده را در پوشه App\_Data پروژه جدید کپی می کنیم. بعد روی فرم یک Label اضافه کرده، در رخداد بار شدن فرم کد زیر را وارد می کنیم. قبل از این کار ما فضای نامهای زیر را به برنامه اضافه می کنیم:

```
using System.Data.OleDb;
using System.Text;

protected void Page_Load(object sender, EventArgs e)
{
 string FilePath;
 FilePath = Server.MapPath("App_Data/MyTestDb.mdb");
 OleDbConnection olecon1 = new
OleDbConnection("provider=microsoft.jet.oledb.4.0;data source=" +FilePath);
 OleDbCommand olecom1 = new OleDbCommand("select * from Entity",olecon1);

 olecon1.Open();
 OleDbDataReader oledread1 = olecom1.ExecuteReader();
 //olecon1.Close();

 StringBuilder strbuld1 = new StringBuilder();

 oledread1.Read();
 strbuld1.Append("<Table>");
 do
 {
 strbuld1.Append("<TR><TD>");
 strbuld1.Append(oledread1.GetInt16(0).ToString());
 strbuld1.Append("</TD><TD>");
 strbuld1.Append(oledread1.GetString(1).ToString());
 strbuld1.Append("</TD></TR>");
 } while (oledread1.Read());
 strbuld1.Append("</Table>");
 Label1.Text = strbuld1.ToString();
}
```

در این مثال از شی **StringBuilder** استفاده شده که از شی های مریبوط به فضای نام **System.Text** می باشد که دقیقاً کار شی **String** را انجام می دهد با این تفاوت که برای متصل کردن رشته در این شی بجای عملگر (+) از متده استفاده می شود. در این برنامه از شی **OleDbDataReader** نیز استفاده شده که برای خواندن اطلاعات استفاده می شود. یکی از متدهای این شی **Read()** می باشد که نتیجه آن از نوع **Boolean** می باشد بدین صورت که اگر رکورد بعدی خالی باشد مقدار **False** بر می گردداند، در غیر این صورت مقدار **True** بر می گردداند.

یک روی فرم قرار دهید و سپس فضاهای نام زیر را به برنامه اضافه نمایید :

## کلاس **OleDbDataAdapter**

همانطور که تا به حال ملاحظه کرده اید **Data Adapter** بیانگر دستورات و اتصالاتی است که برای به روز رسانی پایگاه داده بکار گرفته می شود . این کلاس سه خاصیت دستوری دارد که برای به روز رسانی پایگاه داده مورد استفاده قرار می گیرد :

برده می شود.

:Select Command بیانگر یک عبارت SQL است که برای انتخاب رکوردها از بانک اطلاعاتی بکار می رود.

:Delete command بیانگر یک عبارت SQL است که برای حذف رکوردها از پایگاه داده بکار می رود.

## کلاس های **DataTable**, **DataRow**, **DataSet** و **DataTable**

همانطور که قبلا اشاره شد. DataSet کلاسی است عمومی که به وسیله‌ی .NET Framework تهیه شده است.

این کلاس بر روی سمت Client برای ذخیره سازی داده‌ها به روشی که بسیار کاربردی تر و قوی تر نسبت به کاربرد دارد. علاوه بر این داده‌ها در DataSet به فرمت XML موجود بوده و بنابراین برای دستیابی و مدیریت آماده می باشند. فرمت XML آنرا برای کاربرد های وب بسیار مناسب ساخته و دستیابی Cross-Platform را ممکن می سازد. قابلیت ذخیره سازی از چندین جدول و حفظ ارتباطات بین آنها را ممکن می سازد. جداول در اشیا DataTable ذخیره می شوند و DataRelation بیانگر ارتباطات بین جداول است. در شیء های DataColumn و DataRow به ترتیب، ردیف ها و ستون ها در یک جدول ذخیره می شوند.

### مثال 11

هدف از این مثال کار کردن با شیء های DataTable, DataRow و DataColumn می باشد. در این مثال می خواهیم به مثال قبلی یک دکمه اضافه کرده در رخداد کلیک آن کد زیر را وارد کرده که اطلاعات داخل جدول Entity از پایگاه MyTestDb را یک سطر در میان نمایش دهد.

```
protected void Button1_Click(object sender, EventArgs e)
{
 string FilePath;
 FilePath = Server.MapPath("App_Data/MyTestDb.mdb");
 OleDbConnection olecon1 = new
 OleDbConnection("provider=microsoft.jet.oledb.4.0;data source=" + FilePath);
 OleDbCommand olecom1 = new OleDbCommand("select * from Entity", olecon1);

 OleDbDataAdapter oledatal = new OleDbDataAdapter(olecom1);
 DataSet dataset1 = new DataSet();
 oledatal.Fill(dataset1);

 StringBuilder strbuld1 = new StringBuilder();

 foreach (DataTable datatable1 in dataset1.Tables)
 {
 strbuld1.Append("<Table>");
 int i = 0;
 foreach (DataRow datarow1 in datatable1.Rows)
 {
 if (i % 2 == 0)
 {
 strbuld1.Append("<TR>");
 foreach (DataColumn datacolumn1 in datatable1.Columns)
 {
 strbuld1.Append("<TD>");
 strbuld1.Append(datarow1[datacolumn1]);
 }
 }
 else
 {
 strbuld1.Append("<TD>");
```

```

 strbuld1.Append("</TD>");
 }
 strbuld1.Append("</TR>");
 i = i + 1;
}
strbuld1.Append("</Table>");
}

Label1.Text = strbuld1.ToString();
}

```

تا اینجا ما فقط در مورد نحوه ارتباط با پایگاه داده های مختلف صحبت کردیم در ادامه نحوه نمایش اطلاعات را بررسی خواهیم کرد. در ASP.NET کنترل های متنوعی با خاصیت اتصال به پایگاه داده ارائه شده اند که دارای توانایها و انعطاف پذیری خاص خود می باشند که ما به بررسی این موارد می پردازیم:

### نمایش داده های تکرار شونده:

می توان از کنترل Repeater برای نمایش داده ها به صورت لیست های ویژه و سفارشی استفاده کرد. برای فرمت کردن آنها می توان از Templates استفاده نمود که در زیر در مورد آن توضیح داده شده است. برای تغییر دادن و یا بالا بردن قوت ظاهر کنترلها از آن استفاده می شود. انواع مختلفی از Template ها وجود دارد:

: برای فرمت خروجی در هر ردیف کاربرد دارد. ItemTemplate

: برای فرمت خروجی ردیف ها به صورت یک در میان. AlternatingItemTemplate

: برای فرمت خروجی قبل از داده ها. HeaderTempalte

: برای فرمت خروجی بعد از داده ها. FooterTemplate

: برای فرمت کردن بین ردیف های داده. مانند گذاشتن یک خط افقی بین ردیف داده ها. SeperaterTemplate

### مثال 12

هدف از این مثال کار کردن با کنترل Repeater می باشد. در این مثال می خواهیم اطلاعات ستون fname جدول employee از پایگاه داده Pubs را توسط Repeater نمایش بدهیم. برای انجام این کار یک Repeater و یک دکمه به فرم مثال قبل اضافه کرده و عملیات های زیر را انجام خواهیم داد. ابتدا در حالت Source موارد زیر را روی Repeater انجام می دهیم.

در کنترل Repeater یک ItemTemplate، AlternatingItemTemplate، Header Tempalte و FooterTemplate اضافه می کنیم:

```

<asp:Repeater ID="Repeater1" runat="server">
 <HeaderTemplate>
 <table border="1">
 <tr>
 <td>Title Table</td>
 </tr>

```

```

</HeaderTemplate>
<AlternatingItemTemplate>
 <tr>
 <td style="background-color:Gray">

 <%#DataBinder.Eval(Container.DataItem, "fname") %>

 </td>
 </tr>
</AlternatingItemTemplate>
<ItemTemplate>
 <tr>
 <td style="background-color:Navy">

 <%#DataBinder.Eval(Container.DataItem, "fname") %>

 </td>
 </tr>
</ItemTemplate>
<FooterTemplate>
 </table>
</FooterTemplate>
</asp:Repeater>

```

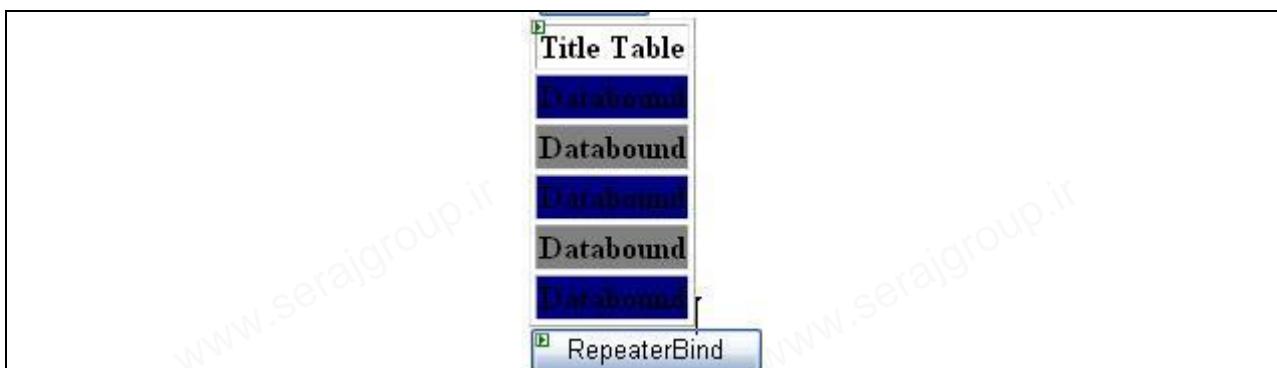
بعد از این به حالت Design برگشته کد زیر را در رخداد کلیک دکمه وارد می کنیم:

```

protected void Button2_Click(object sender, EventArgs e)
{
 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=Pubs;Integrated Security=True");
 SqlDataAdapter sqldata1 = new SqlDataAdapter("Select fname from employee",sqlcon1);
 DataSet dataset1 = new DataSet();
 sqldata1.Fill(dataset1, "employee");
 Repeater1.DataSource = dataset1.Tables["employee"].DefaultView;
 Repeater1.DataMember = "fname";
 Repeater1.DataBind();
}

```

مراحل اجرای کد بدین صورت است که موقعی که دکمه زده می شود، ارتباط با پایگاه برقرار شده، Query مربوط به Repeater از داده پر شده، بعد داده به DataSet داده می شود. بعد این داده توسط دستورات داخل Repeater اجرا شده، HTML نمایش داده می شود. باید به رابطه بین پارامترهای دستورات داخل کد HTML و پارامترهای داخل دستورات داخل رخداد کلیک کردن توجه شود.



## :DataList کنترل

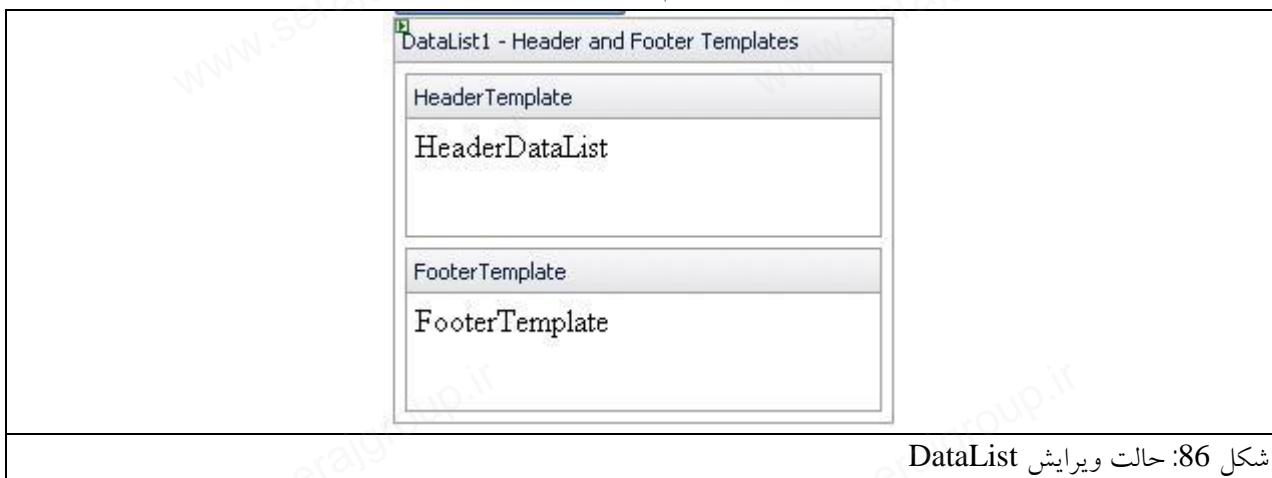
کنترل **DataList** برای نمایش داده ها در فرمت که شما با استفاده **Template** ها و **Style** تعریف کرده اید بکار برده می شود. کنترل **DataList** برای داده ها در هر ساختار تکراری مانند جدول مفید می باشد. کنترل **DataList** می تواند سطر ها را در طرح بندي مختلف مرتب کردن در سطرهای و ستون های نمایش داد.

بسیاری از ویژگی های **DataList** با کنترل **Repeater** یکسان است بعلاوه یک سری گزینه های فرمت کردن **Template** های اضافی و تعیین جهت نمایش داده ها. برای اتصال پایگاه داده به آن مانند **Repeater** از خاصیت **Template** **DataSource** و سپس فراخونی متدهای **DataBind** استفاده می شود. کنترل **DataList** مانند کنترل **Repeater** های زیر را پشتیبانی می کند:

**ItemTemplate**, **AlternatingTemplate**, **HeaderTemplate**, **FooterTemplate**, **SeparatorTemplate**  
علاوه از **SelectedItemTemplate** برای هنگامیکه کاربر آیتمی را انتخاب و **EditItemTemplate** هنگامیکه کاربر آیتمی را ویرایش می کند، بکار می روند.

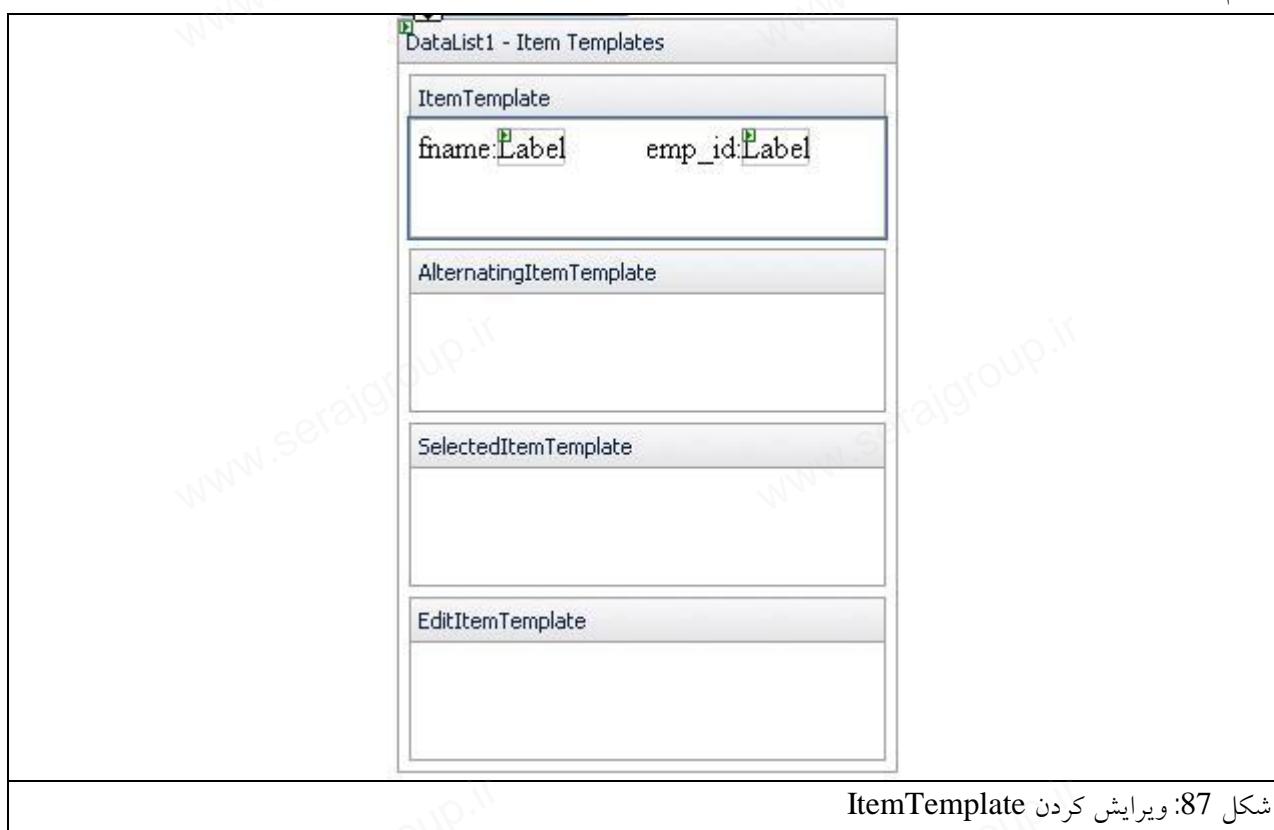
### مثال 13

هدف از این مثال کار کردن با **DataList** می باشد. برای این کار یک پروژه جدید ایجاد کرده روی فرم یک دکمه و یک **Template** قرار دهید برای ویرایش کردن **DataList** ها روی آن کلیک راست نمایید. از منوی ظاهر شده گزینه **Header and footer templates** را انتخاب نمایید. در این حالت **Template** های کنترل قابل ویرایش کردن می شود و می توان متنی یا کنترل دلخواهی مانند دکمه یا **Text** را به آن اضافه کرد. در قسمت سفید رنگی که زیر **Header** و یا **Footer** قرار دارد می توان این عملیات را انجام داد.

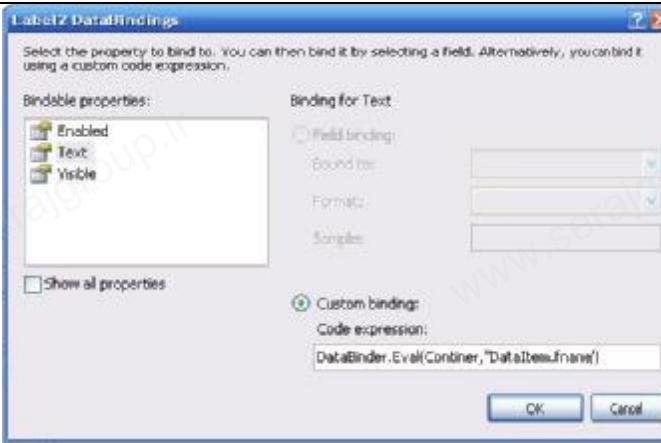


شکل 86: حالت ویرایش **DataList**

برای خاتمه دادن به ویرایش Header and footer دوباره روی کترل کلیک راست نموده گزینه **End** را انتخاب نمایید. حال می خواهیم ItemTemplates را ویرایش کنیم بدین منظور مانند قبل روی کترول کلیک راست کرده این بار از قسمت **Edit Template** گزینه **Item Templates** انتخاب می کنیم. برای این اینکه بتوانیم اطلاعات جدول employee از پایگاه داده pubs را نشان بدهیم دو **Label** را روی قسمت **Item template** قرار می دهیم.



برای بایند کردن داده ها به این **Label** ها باید روی این **Label** ها کلیک راست کرده، در منوی ظاهر شده گزینه **Edit Bindable properties** را انتخاب نمود و سپس در صفحه **Properties** **Text** خاصیت **DataBind** را انتخاب نموده و سپس از عبارت **DataBinder.Eval** مانند پنل سمت راست قسمت **Custom binding expression** را انتخاب نمود و سپس از **Repeater** بهره جست.



شکل 88: مشخص کردن Label ها

برای Label که جهت نمایش دادن emp\_id ها قرار داده ایم عبارت زیر را بنویسید:  
DataBinder.Eval(Container, "DataItem.emp\_id")

و برای لیبل مربوط به عبارت fname زیر را بنویسید:  
DataBinder.Eval(Container, "DataItem.fname")

اکنون با استفاده از کلیک راست روی صفحه می توان به ویرایش کردن آن خاتمه بخشد. برای نمایش اطلاعات جدول

موقع کلیک کردن دکمه کد زیر را در رخداد دکمه می نویسیم:

```
employee
SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=Pubs;Integrated Security=True");
SqlDataAdapter sqldata1 = new SqlDataAdapter("Select emp_id,fname from employee", sqlcon1);
DataSet dataset1 = new DataSet();
sqldata1.Fill(dataset1, "employee");
```

```
DataList1.DataSource = dataset1.Tables["employee"].DefaultView;
DataList1.DataBind();
```

می توانید کد HTML تولید شده را با کد Repeater مقایسه کنید!!

## dropDownList کنترل

برای نمایش یک ستون از یک جدول پایگاه داده با اعمال فیلترهای مختلف می توان استفاده کرد.

### مثال 14

در این مثال می خواهیم نحوه کار کردن با کنترل DropDownList را با نمایش ستون fname از جدول employee از پایگاه داده Pubs در کنترل DropDownList و پس از انتخاب هر آیتم توسط کاربر نام آن در یک نمایش داده شود را بیان نماییم:

به پروژه ای قبلی یک DropDownList و یک Label اضافه کنید. در رخداد Load کردن کد زیر را برای Bind داده های ستون fname از جدول employee وارد کنید. چون تمام دستورات تکراری می باشند ما در مورد آنها زیاد توضیح نخواهیم داد:

```
if (!Page.IsPostBack)
{
```

```

 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial
Catalog=pubs;Integrated Security=True");
 SqlDataAdapter sqldata1 = new SqlDataAdapter("Select emp_id, fname from
employee", sqlcon1);
 DataSet dataset1 = new DataSet();
 sqldata1.Fill(dataset1, "employee");

 DropDownList1.DataTextField = "fname";
 DropDownList1.DataSource = dataset1.Tables["employee"].DefaultView;
 DropDownList1.DataBind();
}

```

همانطور که ملاحظه می فرمایید خاصیت **DropDownList1** این کنترل برای مشخص کردن فیلدی که قرار است نمایش داده شود بکار بردہ می شود.

برای اینکه با انتخاب یک آیتم توسط کاربر در **Label** آن را نمایش دهید. باید خاصیت **AutoPostBack** کنترل **DropDownList** را **True** کنیم. سپس در برگه **SelectedIndexChanged** از دوبار کلیک نمایید تا روال رخداد آن ایجاد شود. سپس به سادگی از کد زیر برای این قسمت استفاده کرد.

```

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
 Label4.Text = DropDownList1.SelectedItem.Text;
}

```

## کنترل :GridView

قبلما کنترل **GridView** را معرفی کردیم در ادامه ای فصل به توانایی های این کنترل که بیش از همه در صفحات **ASP.NET** کاربرد دارد. با ذکر مثالهایی کاربردی، می پردازیم:

### مثال 15

در این مثال می خواهیم یکی از قابلیت های مهم **GridView** که همانا صفحه بنده می باشد را معرفی نماییم. اگر به کنترل های که تابحال که اینکار اصلاً ظاهر خوشایندی ندارد. برای رفع این مشکل قابلیت ذکر شده به دیتاگرید اضافه گردیده است که از آن در طی یک مثال استفاده خواهیم کرد:

پروژه ای جدید را آغاز کرده و یک **GridView** روی صفحه قرار دهید. در صفحه خصوصیات این کنترل خصوصیت **AllowPaging** را به **TRUE** تنظیم نمایید تا اجازه صفحه بنده داده شود. برای تعیین تعداد رکوردها در هر صفحه می توان از خصوصیت **PageSize** که بطور پیش فرض 10 می باشد. استفاده کرد و همچنین برای تعیین کنترلهای صفحات می توان از گزینه **Mode** از زیر خصوصیات **PagerSetting** استفاده کرد. روی صفحه دوبار کلیک نموده و کد زیر را در رخداد بارشدن آن وارد کنید تا بتوان جدول **employee** را به **GridView** متصل کرد:

```

 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial
Catalog=pubs;Integrated Security=True");
 SqlDataAdapter sqldata1 = new SqlDataAdapter("Select emp_id, fname from
employee", sqlcon1);
 DataSet dataset1 = new DataSet();
 sqldata1.Fill(dataset1, "employee");

```

```
GridView1.DataSource = dataset1.Tables["employee"].DefaultView;
```

```
GridView1.DataBind();
```

اگر برنامه را در این حالت اجرا نماید فقط صفحه‌ی اول نمایش داده می‌شوند و بقیه صفحات کار نمی‌کنند. برای اینکه صفحات دیگر را هم بکار بیاندازیم باید برای آنها کد نوشت:

برای فعال کردن صفحات دیگر، خصوصیت **PageIndex** را تغییر می‌دهیم و دوباره اطلاعات را به این کنترل **Bind** خواهیم کرد. برای انجام این کار باید این کارها را در **رخداد PageIndexChanging** نوشت:

```
protected void GridView1_PageIndexChanging(object sender,
 GridViewPageEventArgs e)
{
 GridView1.PageIndex = e.NewPageIndex;
 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=Pubs;Integrated Security=True");
 SqlDataAdapter sqldata1 = new SqlDataAdapter("Select emp_id, fname from employee", sqlcon1);
 DataSet dataset1 = new DataSet();
 sqldata1.Fill(dataset1, "employee");
 GridView1.DataSource = dataset1.Tables["employee"].DefaultView;
 GridView1.DataBind();
}
```

یکی دیگر از قابلیت‌های **GridView** اضافه کردن لینک‌های **Edit** و **Update** به ازای هر ردیف می‌باشد. که با کلیک کردن بر روی **Edit** تمام سلوکهای یک ردیف به شکل **TextBox** قابل ویرایش درآمده و پس از اتمام ویرایش با کلیک بر روی **Update** و البته کد نویسی برای آن، داده‌ها را در دیتابیس ذخیره خواهند شد. حال با یک مثال این قابلیت را تشریح می‌کنیم.

## :16 مثال

پروژه‌ای جدید را ایجاد کرده و سپس یک **GridView** روی فرم قرار دهید. برای اینکه لینک‌های **Edit** به ازای هر ردیف داشته باشیم خاصیت **AutoGenerateEditButton** را **True** می‌کنیم. در این مثال از پایگاه داده **Northwind** که همراه **SQL-Server** ارائه می‌شود به صورت زیر برای **Bind** کردن اطلاعات استفاده می‌کنیم:

```
public void bidtogridview()
{
 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=Northwind;Integrated Security=True");
 SqlDataAdapter sqldata1 = new SqlDataAdapter("select EmployeeID, LastName, FirstName from employees", sqlcon1);
 DataSet dataset1 = new DataSet();
 sqldata1.Fill(dataset1, "sal");
 GridView1.DataSource = dataset1.Tables["sal"].DefaultView;
 string[] strkey = { "EmployeeID" };
 GridView1.DataKeyNames = strkey;
 GridView1.DataBind();
}
protected void Page_Load(object sender, EventArgs e)
{
 if (!Page.IsPostBack)
 bidtogridview();
}
```

برای اینکه وقتی کاربر روی لینک Edit کلیک می کند سلوکها به حالت ویرایش درآیند، باید در رخداد RowEditing ردیفی که باید به حالت ویرایش درآید مشخص شود:

```
protected void GridView1_RowEditing(object sender, GridViewEventArgs e)
{
 GridView1.EditIndex = e.NewEditIndex;
 bidtogramview();
}
```

حال اگر برنامه را اجرا نماییم و روی گزینه Edit هر ردیف کلیک نماییم آن ردیف یه حالت ویرایش در می آید و بجای گزینه Edit گزینه های Cancel و Update جایگزین می شود، حال اگر روی هر کدام از این گزینه کلیک نماییم با خطای مواجهه می شویم. برای اینکه با این گزینه خطای مواجهه نشویم باید برای هر کدام از این لینک ها برنامه نویسی کنیم، به صورت زیر در رخداد مربوط به RowCancelingEdit کد زیر را وارد می کنیم تا موقعی که روی Cancel کلیک کردیم به حالت اول برگردیم:

```
protected void GridView1_RowCancelingEdit(object sender,
GridViewCancelEventArgs e)
{
 GridView1.EditIndex = -1;
 bidtogramview();
}
```

قسمت اساسی این مثال از این لحظه آغاز می شود! از کد زیر برای هنگامی استفاده می کنیم که کاربر روی لینک Update کلیک نموده است و باید داده های تغییر یافته در پایگاه داده ذخیره شوند. برای اینکار باید کد زیر را به روال رخداد RowUpdating اضافه نمود:

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
 string employeeid = GridView1.Rows[e.RowIndex].Cells[1].Text;
 string Firstname =
((TextBox)GridView1.Rows[e.RowIndex].Cells[3].Controls[0]).Text;
 string LastName =
((TextBox)GridView1.Rows[e.RowIndex].Cells[2].Controls[0]).Text;
 string ReportsTo =
((TextBox)GridView1.Rows[e.RowIndex].Cells[4].Controls[0]).Text;
 SqlConnection sqlcon1 = new SqlConnection("Data Source=(local);Initial Catalog=Northwind;Integrated Security=True");
 string strcmd = "update employees set Firstname=' " + Firstname +
" ',ReportsTo=' " + ReportsTo + ",LastName=' " + LastName + " ' where
employeeid=' " + employeeid + " '";
 SqlCommand sqlcoml = new SqlCommand(strcmd, sqlcon1);
 sqlcon1.Open();
 sqlcoml.ExecuteNonQuery();
 sqlcon1.Close();
 GridView1.EditIndex = -1;
 bidtogramview();
}
```

در گزارشات عموماً اضافه کردن نتیجه عددی به انتهای گزارش نیاز می باشد. می خواهیم در مثال زیر نحوه انجام این

کار را با امکانات GridView توضیح بدھیم:

یکی دیگر از امکانات Footer داشتن GridView می باشد که می تواند برای جمع بندی کردن گزارشات بکار برد  
شود. برای استفاده کردن از این خاصیت کارهای زیر را روی پروژه‌ی قبلی انجام می دهیم:

در این مثال می خواهیم جمع تمام مقادیر فیلد ReportsTo از پایگاه Northwind را به انتهای GridView اضافه کنیم. برای انجام این کار اولاً در تابع bidtogridview() دستور زیر را برای اضافه کردن Footer اضافه می کنیم:

```
GridView1.ShowFooter = true;
```

در مرحله بعدی مدیریت رخداد RowDataBound می باشد. چون هنگامیکه هر سطر GridView اضافه می شود رخداد RowDataBound آن به ازای هر سطر، فراخوانی می شود. از این مورد برای خواندن اطلاعات و جمع آن با ردیف های قبلی به سادگی می توان استفاده کرد که نحوه‌ی انجام آن در زیرآورده شده است:

ابتدا یک متغیر عمومی برای نگهداری جمع فیلد ReportsToint اضافه می کنیم:

```
private int ReportsToint = new int();
```

سپس در روای رخداد RowDataBound کار نهایی انجام خواهد شد:

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
 if (e.Row.RowType == DataControlRowType.DataRow)
 {
 try
 {
 ReportsToint += int.Parse(e.Row.Cells[4].Text);
 }
 catch
 {
 }
 }
 else if (e.Row.RowType == DataControlRowType.Footer)
 {
 e.Row.Cells[0].Text = "Total:";
 e.Row.Cells[4].Text = ReportsToint.ToString();
 }
}
```