



مهندس سعید عاطفی

# برنامه نویسی تحت وب با ASP.NET 4.5



All in one





هیچ کس نباید به دلیل مشکلات مالی، آموزش فود را به تغییر بیندازد.  
ما نمیگذاریم.



کارآموزتر



All in one



ساده تر بیان موزیک

پیش کش به

همه مادران ایران زمین

سعید عاطفی



کارآموزت

کارآموزت

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ



## اين كتاب توسيط آقاي حميد سيفي

به کد ملي: ۱۹۱—۴۱۰— خريداري گردیده است

کد رهگيري: ۱۳۰۰۷۰۶۸۹۶۲



هچ کس نباید به دلیل مشکلات مالی، آموزش خود را به تغییر پیندازد. ما نمی‌گذریم.

اگر روپکرد ما همکاری متقابل آموزشی، ارتقاء سطح فنی و علمی خود و دیگران باشد. چرا محیطی برای این همکاری به وجود نیاوریم و امکان رشد با استفاده از امکانات و خدمات آموزشی، کسب چرخه و زومه کاری از طریق همکاری با افراد متخصص را فراهم نکنیم.

www.KARAMOOZESH.COM  
KARAMOOZESH  
www.KARAMOOZESH.COM

www.KARAMOOZESH.COM  
KARAMOOZESH

# Warring

اخطار

این نسخه از کتاب برای آقای حمید سیفی منتشر و حق استفاده از آن صرفاً برای ایشان محفوظ می‌باشد. هرگونه انتشار و استفاده تجاری از این محصله برای عموم ممنوع بوده و ضمن غیر شرعاً بودن این امر، حق پیگرد قانونی در هر ارجح ذیصلاح نیز برای گروه کارآهوزش محفوظ می‌باشد.



## کتاب سفارشی

یکی از اهداف ما ارائه کتب آموزشی الکترونیکی فارسی با بهترین کیفیت و با دسترسی آسان برای همه افرادی است که خواهان آموزش صحیح و درست در سراسر دنیا هستند، می‌باشند.

با توجه به حرکت در این مسیر، تمایل به استفاده از ایده‌های نو و کارآمد در عرصه ارائه محصولات آموزشی الکترونیکی داریم.

طرح (کتاب سفارشی)، قدمی دیگر برای مطالعه کنندگان عزیز می‌باشد که با استفاده از آن یک کتاب مخصوص شما و به نام شما طراحی می‌شود.

در این طرح شما می‌توانید مشخصات خود را برای ما ارسال کنید تا آن‌ها را در کتاب خودتان قرار دهیم. بنابراین هر زمان و در هر مکان که این کتاب مطالعه شود، مشخص می‌شود که این کتاب برای شما است و شما کم کم دارای یک کتابخانه از کتاب‌های الکترونیک هستید که سفارشی برای خودتان تولید شده است.

همچنین هر کتاب دارای چندین طرح می‌باشد که از نظر گرافیک، رنگ بندی و فرم، متفاوت می‌باشند.

به دلیل آنکه در هنگام مطالعه کتاب اندکی به نکات دیگر نیز توجهی داشته باشد و ذهن خود را از موضوع دور کرده و به خود استراحت دهد علاوه بر تفاوت در موارد بالا، در این سبک از کتاب‌ها طرح‌های کلی درباره تصاویر دیدنی از ایران عزیز، جملات و داستان‌های کوتاه آموزنده و گاهی نیز نقاشی‌های دیدنی از هنرمندان معرف ایران و جهان قرار داده خواهد شد.

امید است با طرح (کتاب سفارشی) بتوانیم رضایت خاطر بیشتر شما را بدست آورده و در مسیر ارائه خدمات آموزشی مناسب به شما عزیزان موفق باشیم.

با ارائه محصولات و خدمات با کیفیت و انجام صحیح امور به یکدیگر احترام خواهیم گذاشت.



## فصل چهارم Error Handling و Logging, Tracing

- پیشگیری از خطاهای رایج
- آشنایی با کنترل خطای Exception Handling
- کلاس Exception و Property های Exception
- زنجیره خطای
  - گرفتن (catch) خطاهای خاص
  - استفاده از Exception Handler های تو در تو Throw
  - استفاده از Page Tracing
  - فعال سازی Tracing - ردیابی اطلاعات ردیابی
- Application State و Session State
- Response Cookies و Request Cookies
- Headers Collection , Form Collection Query String Collection
- ردیابی سطح برنامه

## فصل پنجم State Management

- آشنایی با مشکل State View State
- استفاده از View State محرمانه
- انتقال اطلاعات بین صفحات Cross-Page Posting
- دریافت اطلاعات بیشتر از صفحه مبدا Query String URL Encoding
- استفاده از کوکی ها Cookie
- مدیریت Session State Session Tracking
- پیکربندی Session state Timeout
- Cookieless Mode
- Inproc \*
- Off \*
- StateServer \*
- SQL Server \*
- Custom \*
- Compression \*
- Application State - استفاده از

## بخش دوم: ساخت فرم های وب بهتر

### فصل ششم اعتبارسنجی - Validation

- آشنایی با اعتبارسنجی
- کنترل های اعتبارسنجی
- اعتبارسنجی سمت کلاینت
- ۵ HTML - اعتبارسنجی
- استفاده از کنترل های اعتبارسنجی Regular Expression - اعتبارسنجی با عبارات با قاعده و منظم

### فصل هفتم کنترل های حرفه ای

- کنترل AdRotator
- کنترل Mutiple View
- کار با Wizard
  - Wizard کنترل

### فصل هشتم User Control

- ایجاد User Control
- کار با User Control های مستقل

## بخش اول : توسعه برنامه های ASP.NET

### فصل اول آشنایی با Visual Studio

- ایجاد وب سایت
- ایجاد یک برنامه تحت وب خالی webproject website
- کار با فایل های مخفی Solution Explorer
- استفاده از اضافه نمودن فرم های وب
- بازگردان وب سایت از درون نسخه های قبلی Visual Studio
- طراحی یک فرم وب

- اضافه نمودن کنترل های وب Properties
- استفاده از پنجره Web Form Markup
- - Page Directive
- Doctype
- HTML ملزومات
- \* المان ها
- \* خصیصه ها
- \* فرمت دهنده
- نوشتن کد در Code-Behind
- اضافه کردن Event Handler
- Outlining
- ترسیم IntelliSense
- گرفتن خطای در کد Markup
- Debugging
- عیب یابی و رفع اشکال IIS Express
- وеб سرور
- مرحله ای Debug
- تک BreakPoint
- Variable Watches

### فصل دوم اصول فرم های وب

- آشنایی با ساختا یک برنامه ASP.NET
- انواع فایل های ASP.NET
- پوشش های وب
- معرفی کنترل های سروری HTML
- کنترل های سروری View State
- کلاس های HTML Control
- Event Handling
- Error Handlig
- رویداد های کنترل های HTML
- استفاده از کلاس Page
- فرستادن کاربر به صفحه جدید HTML Encoding
- کار با
- استفاده از رویداد های برنامه Global.asax
- پیکربندی یک برنامه ASP.NET
- کار با فایل Web.Config
- پیکربندی های تودر تو web.config
- قرار دادن تنظیمات سفارشی در
- استفاده از ابزار مدیریت وب سایت

### فصل سوم کنترل های وب - web control

- کلاس های پایه برای وب کنترل
- تگ های وب کنترل
- رویدادهای وب کنترل Auto Postback
- نحوه کار رویدادهای Postback
- چرخه حیات صفحه Page Life Cycle

• پیکربندی ستون ها  
 Visual Studio  
 • ایجاد ستون ها با GridView  
 • فرمت دهنده استایل ها  
 • استفاده از استایل GridView  
 • انتخاب یک ردیف از GridView  
 • اضافه کردن دکمه Select  
 • اشاره کردن یک Data Field به عنوان دکمه انتخاب  
 Master-Detail Selection برای ایجاد صفحات  
 • بکارگیری قابلیت  
 • ویرایش با GridView  
 • Paging Sorting  
 • GridView Template  
 • استفاده از Template  
 • استفاده از چند Template  
 Visual Studio  
 • ویرایش Template ها در Event  
 • هندل کردن یک هندلر  
 • ویرایش رکورد به همراه اعتبارسنجی  
 Command Column  
 • ویرایش رکورد بدون استفاده از Form View و Detail View

## بخش چهارم : امنیت وب سایت

**فصل سیزدهم اصول امنیت**

- آشنایی با نیازمندی های امنیتی
- تست و Deploy کردن تنظیمات امنیتی
- احراز هویت و مجوز - Authentication and Authorization
- Forms Authentication

Web.Config

- \* تنظیمات
- \* قوانین احراز هویت
- \* کنترل دسترسی به پوشش های خاص
- \* کنترل دسترسی به فایل های خاص
- \* کنترل دسترسی برای کاربران خاص
- \* ابزار WAT
- \* صفحه Login
- \* بازیابی هویت کاربر - User Identity
- \* Signing Out
- \* کوکی های مقیم یا ماندگار - Persistent Cookie
- \* Windows Authentication

Web.Config

- \* تنظیمات
- \* یک مثال کاربردی

**فصل چهاردهم Membership**

- ذخیره اطلاعات Membership
- SQL Server Express با Membership
- استفاده از نسخه کامل SQL Server
- پیکربندی Membership Provider
- ایجاد کاربر با WAT
- کلاس های MembershipUser و Membership
- احراز هویت با Membership
- کار با کنترل های امنیتی
- کنترل Login
- کنترل CreateUserWizard
- کنترل PasswordRecovery
- امنیت مبتنی بر Role
- \* ایجاد و اختصاص نقش
- \* محدود کردن دسترسی بر اساس نقش
- کنترل LoginView

**فصل پانزدهم Profile**

- آشنایی با Profile
- کارایی Profile
- نحوه ذخیره سازی داده با Profile

- کار با User Control های یکپارچه  
 User Control  
 - استفاده از رویدادهای ارسال اطلاعات با رویدادها  
 - گرافیکهای پویا  
 Chart Control •

## فصل نهم Master Page و Themes

**Master Page و Themes**

Style -

- \* انواع استایل
- \* Style Builder
- \* ا Rath بری استایل ها
- \* Style Sheet

Themes -

- \* نحوه کار Theme
- \* بکارگیری یک هندلر تداخلی Theme ها
- \* ایجاد چندین Skin برای یک کنترل
- \* Skin های پیشرفته

Master Page

- نحوه ارتباط Content Page و Master Page
- یک Content Page با چند ناحیه
- Relative Path و Master Page
- Layout و Master Page
- تعامل با Master Page از طریق کدنویسی

## بخش سوم : کار با داده

**ADO.NET اصول**

- آشنایی با بانک اطلاعاتی
- پیکربندی بانک اطلاعاتی

SQL Server Express

- \* استفاده از Visual Studio
- \* نمایش و تغییر بانک اطلاعاتی در Connection String
- \* نگهداری رشته اتصال Connection
- ایجاد DataReader
- پرکردن ListBox
- بازیابی رکورد
- ویرایش داده ها
- نمایش مقادیر
- اضافه کردن داده
- حذف رکورد

Disconnected Data Access

**فصل یازدهم Data Binding**

- معرفی Data Binding
- انواع Data Binding

Single-Value Data Binding

Repeated-Value Data Binding

- نحوه کار Data Binding
- Multiple Binding
- Dictionary با Data Binding
- Data Value Field
- استفاده از ADO.NET با Data Binding

**فصل دوازدهم کنترل های داده ای - GridView**

- ایجاد خودکار ستون ها
- تعریف ستون ها

**ASP.NET AJAX**

- آشنایی با AJAX

- ASP.NET در AJAX

Script Manager •

• استفاده از Partial Refresh

• کنترل خطای AJAX

• بروزرسانی مشروط

Trigger \*

• استفاده از Progress Bar

• امکان کنسل کردن رویداد

• بکارگیری refresh زمانبندی شده

• نصب و کار با ASP.NET AJAX Toolkit

**ASP.NET Deploy**

- برنامه های ASP.NET و وب سرور

• نحوه کار وب سرور

Virtual Directory •

• URL های برنامه تحت وب

Web Farm •

IIS •

\* نصب IIS روی نسخه دسکتاپ ویندوز

۲۰۰۸ Windows Server روی IIS

۲۰۱۲ Windows Server روی IIS

\* مديريت وب سايت با IIS Manager

• آشنایی با Application Pool

ASP.NET Account •

• پیکربندی یک وب سايت

• صفحات سفارشی خطا

Windows Authentication •

• محرومگی با SSL و گواهینامه ها

\* ایجاد یک درخواست محرومراه

Deploy - کردن یک سايت

• کامپایل کد

Visual Studio Deploy •

• ایجاد یک Virtual Directory برای یک پروژه جدید

• روش کپی کردن وب سايت

• روش Publish وب سايت

- استفاده از SqlProfileProvider

- بانک اطلاعاتی Profile

- تعریف و استفاده از ویژگی های Profile

- سریالیز کردن Profile

- گروه های Profile

- اندیشه ها و انواع داده ای سفارشی

- نوع سفارشی سریالیز

Anonymous Profile -

**بخش پنجم : ASP.NET پیشرفت****فصل شانزدهم برنامه نویسی مبتنی بر کامپوننت**

- دلیل استفاده از کامپوننت ها

- طراحی سه لایه

- کمپوله سازی

Business Objects •

Data Objects •

- ایجاد یک کامپوننت

- اضافه نمودن reference به کامپوننت

- استفاده از کامپوننت State ها و Property

- کامپوننت Data - Access

- کنترل خطای کامپوننت

**Caching**

- آشنایی با Caching

- چه زمان باید از Caching استفاده کرد

Output Caching •

Query String و Caching •

Fragment Caching •

Cache Profiles •

Data Caching •

- اضافه نمودن آیتم به Cache

- Caching به همراه وایستگی

File Dependencies •

Cache Item Dependencies •

SQL Server Cache Dependencies •

- فعال سازی Service Broker •

**فصل هجدهم Entity Framework و LINQ**

- آشنایی با LINQ

• اصول LINQ

• عبارات LINQ

• Projection

• فیلتر و مرتب سازی

- استفاده از Entity Framework

• ایجاد Entity Data Model

• بروزرسانی Data Model

Entity •

Context •

• کنترل خطای

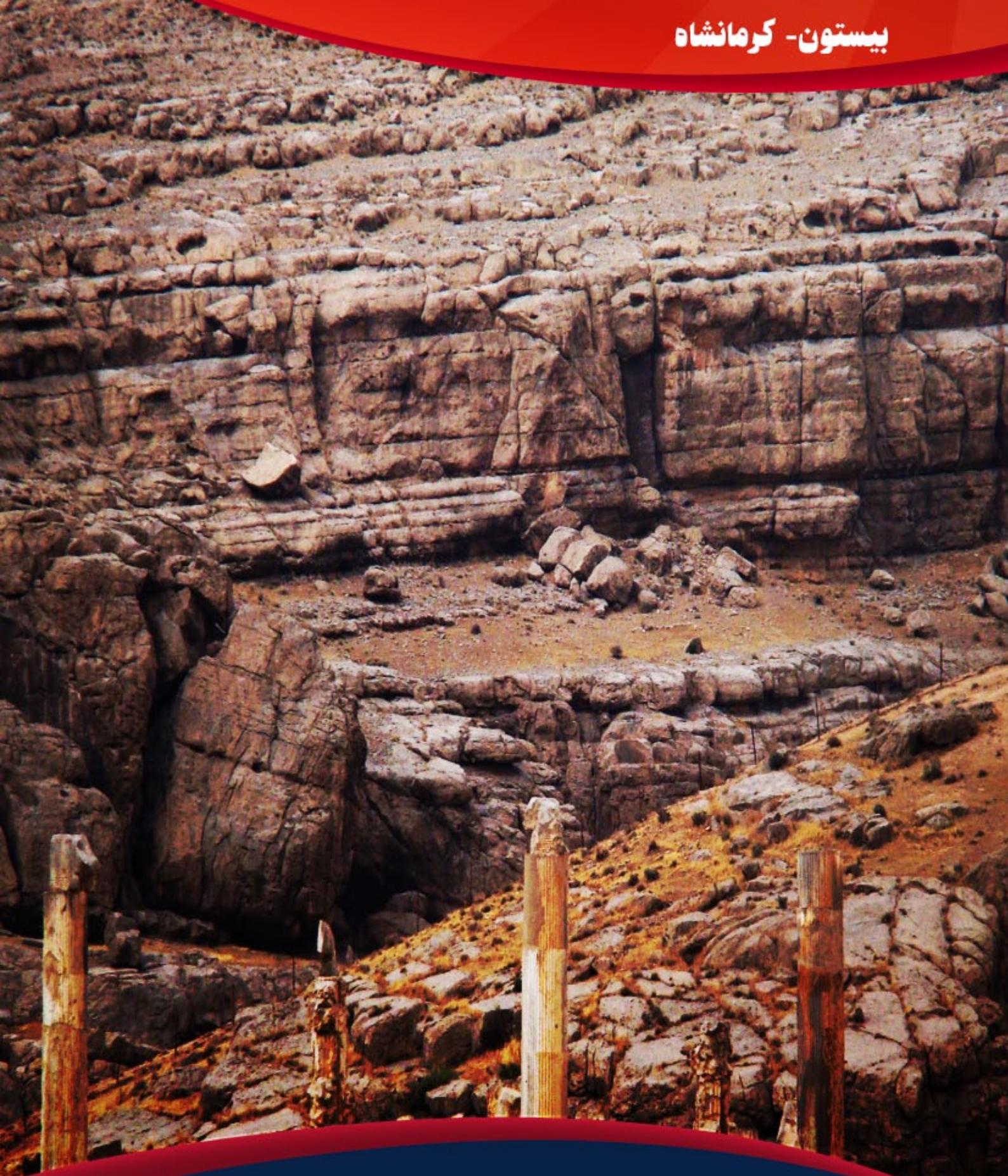
• ارتباطات

LINQ to Entity Query •

• کنترل زمان لود داده

• اجرای حذف ، ایجاد ، ویرایش

• مدیریت همزمانی



طولانی ترین سفرها، حقیقی سفرهای چند هزار کیلومتری  
با یک گام شروع می شوند. «کل تور»

# ASP.NET : توسعه برنامه های بخش اول

فصل اول : آشنایی با **Visual Studio**

~~فصل دوم : اصول فرم های وب~~

~~فصل سوم : کنترل های وب~~

~~فصل چهارم : Error Handling, Logging, Tracing~~

~~فصل پنجم : State Management~~



در روزهای اولیه برنامه نویسی تحت وب، برنامه‌نویسان صفحات وب را با استفاده از ویرایشگر ساده متن مانند Microsoft Notepad ایجاد می‌کردند. اغلب کدهای HTML را با بلوک‌هایی از اسکریپت که در هر بخش از کد که لازم بود اضافه می‌شدند، می‌نوشتند.

Visual Studio، یک جایگزین بسیار بهتر برای NotePad می‌باشد، زیرا به جز نوشتن متن و ذخیره آن دیگر هیچ از ویژگی‌های NotePad در Visual Studio موجود نمی‌باشد.

در این فصل، یاد خواهید گرفت که چگونه با استفاده از Visual Studio، یک برنامه کاربردی تحت وب ایجاد کنید. همچنین خواهید دید که چگونه ویژگی IntelliSense، تعداد خطای شما را کاهش می‌دهد. در انتهای فصل، با کار با Visual Studio و قواعد پایه ای توسعه برنامه‌های تحت وب با استفاده از ASP.NET آشنا شده‌اید.

کلیه برنامه‌های تحت وب، از همان فایل‌های متن قدیمی ساخته شده‌اند. صرف نظر از اینکه این کدها برای برنامه‌های تحت ویندوز یا تحت وب استفاده می‌شوند، کدهای VB درون فایل‌هایی با پسوند .VB و کدهای C# در فایل‌هایی با پسوند .CS، ذخیره می‌شوند. با وجود این موضوع، شما به ندرت برنامه نویسی را می‌بینید که برنامه تحت ویندوزی را بصورت دستی در یک ویرایشگر متن ساده ایجاد کند. این کار نه تنها بسیار خسته کننده می‌باشد، بلکه شما را با خطاهای زیادی نیز روبرو خواهد کرد.

## فواید استفاده از Visual Studio

### کنترل و چک کردن خطای

VS، یک محدوده وسیع از مشکلات مانند خطاهای تبدیل انواع داده‌ای، کلاس‌ها و فضای نام‌های اشتباه، متغیر‌های undefined و ... را تشخیص می‌دهد. همانطور که در حال تایپ کردن هستید، خطاهای شناسایی می‌شوند، زیر آنها خط کشیده می‌شود و به لیست خطاهای اضافه می‌شوند.

برای ایجاد یک صفحه وب در VS، به راحتی می‌توانید کنترل‌های ASP.NET را به مکان مناسبی در صفحه بکشید و Property‌های آن را تنظیم کنید. VS، کدهای HTML مربوط به آن را ایجاد و درون صفحه قرار می‌دهد.

به منظور اجرای برنامه تحت وب ASP.NET، نیاز به یک وب سرور دارید که IIS، روی آن اجرا شده باشد. کار وب سرور دریافت در خواستها و در اختیار قرار دادن صفحه وب مناسب با آن می‌باشد. پیکربندی وب سرور کار سختی نیست ولی چندان هم خوشایند نیست. با تشرک از وب سرور یکپارچه موجود در VS، می‌توانید وب سایتها را مستقیماً از محیط Design اجرا کنید. (البته پس از اتمام تولید برنامه تحت وب، لازم است که آن را روی یک وب سرور حقيقی اجرا کنید)

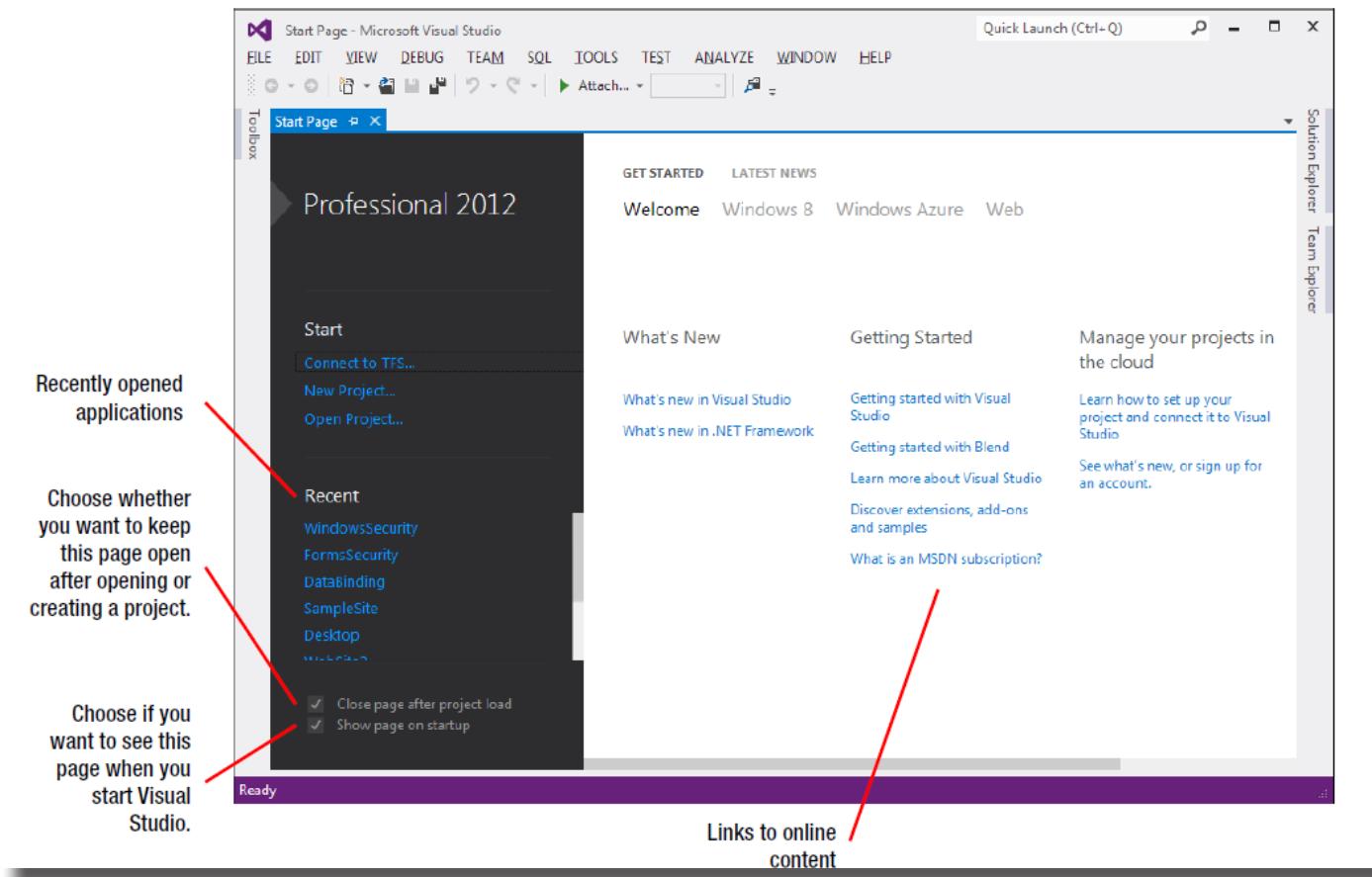
VS، با استفاده از نمایش کدها بصورت کشویی (می‌توان مجموعه‌ای از کدهای درون متدها و ... را بصورت بلوکی از کدها باز و بسته کرد)، تکمیل خودکار دستورات، سینتکس کدهای رنگی، تولید کد را سریعتر و موثر تر کرده است.

VS، به شما اجازه مشاهده کد در زمان اجرا (Watch)، توقف اجرای کد در هر نقطه دلخواه و تعیین مقدار موجود در هر متغیر را می‌دهد. می‌توانید Macro‌های خود را ایجاد کنید، Template پروژه را تغییر دهید، و حتی add-ins ها سفارشی خود را به VS اضافه کنید.

 نکته: تکنیک‌هایی که در این فصل یاد خواهید گرفت با تمامی نسخه‌های VS سازگاری کامل دارد.

## ايجاد WebSite ها

با رفتن به Start>All Programs> Microsoft Visual Studio 2012 می توانيد VS را اجرا کنید.



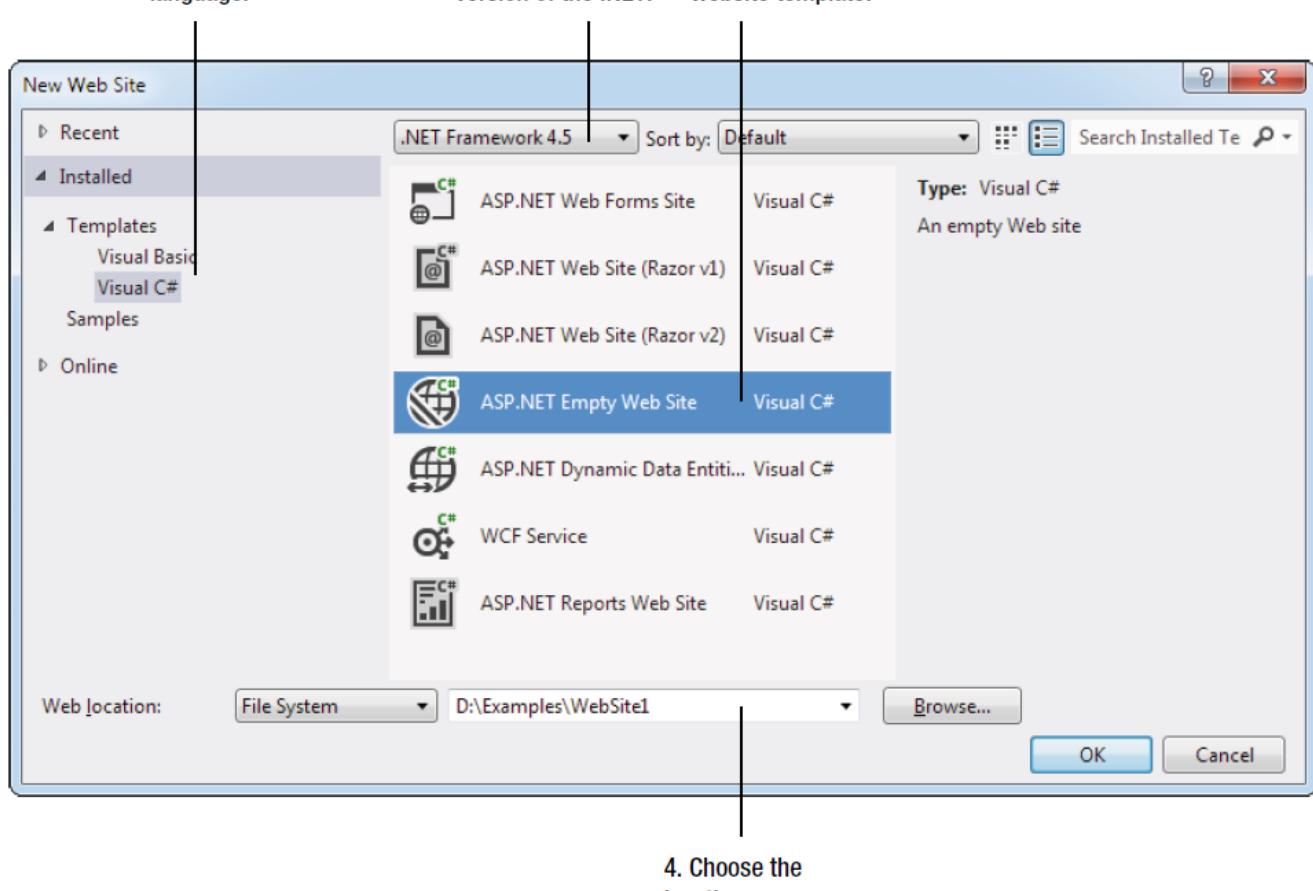
## ایجاد یک Web Application خالی

۱ - مسیر File>New > Web Site را بروید. کادر محاوره ای زیر باز می شود.

1. Choose the language.

2. Choose the version of the .NET.

3. Choose the website template.



4. Choose the location.

۲ - در لیست سمت چپ، زبان برنامه نویسی مورد نیاز برای نوشتن کدها را انتخاب نمایید.

۳ - در dropdown list موجود در بالای پنجره، نسخه دات نتی که می خواهید را انتخاب نمایید. انتخاب پیش فرض .NET FrameWork 4.5 می باشد ولی می توانید نسخه های قدیمی تر را نیز انتخاب نمایید.

۴ - در لیست template های موجود در وسط صفحه، نوع برنامه ای که می خواهید ایجاد کنید را انتخاب نمایید. برای الان، بهترین گزینه ASP.NET Empty Web Site می باشد.

## WEBSITE TEMPLATES

شما می توانید انواع مختلفی از برنامه های ASP.NET را با استفاده از template ها مختلف در VS ایجاد کنید. اینکه از چه template ای استفاده کنید مهم نیست، چون وب سایت با یک روش کامپایل و اجرا می شود. تنها تفاوت در فایل هایی است که بصورت پیش فرض ایجاد می کند. Visual Studio

## ASP.NET Web Forms Site:

این مدل یک وب سایت ASP.NET، با تمامی ویژگی‌ها برای شما ایجاد خواهد کرد. این وب سایت شامل یک master page که طراح کلی را در خود دارد (با header/footer و نوار منو) و دو صفحه با نام default.aspx و about.aspx می‌باشد. همچنین شامل فولدر Accounts با صفحاتی برای ثبت، login و تغییر کلمه عبور کاربر می‌باشد.

## ASP.NET Web Site (Razor):

این مدل یک وب سایت ایجاد می‌کند که از الگوی MVC استفاده خواهد کرد.

برای آموزش و کسب اطلاعات بیشتر درباره MVC، کتاب آموزش کاربردی MVC از مجموعه کتاب‌های all-in-one گروه کارآموزش را مطالعه کنید.

## ASP.NET Empty Web Site:

این مدل شامل یک فایل web.config می‌باشد و نه هیچ چیز دیگر.

## ASP.NET Dynamic Data Entities Web Site:

این مدل شبیه یک وب سایت ASP.NET Dynamic Data می‌باشد که از ویژگی استفاده می‌کند. در حقیقت دو مدل dynamic data وجود دارد که برای اهداف مختلفی در اتصال به بانک اطلاعاتی شما استفاده می‌شود. البته مبحث در این کتاب توضیح داده نخواهد شد.

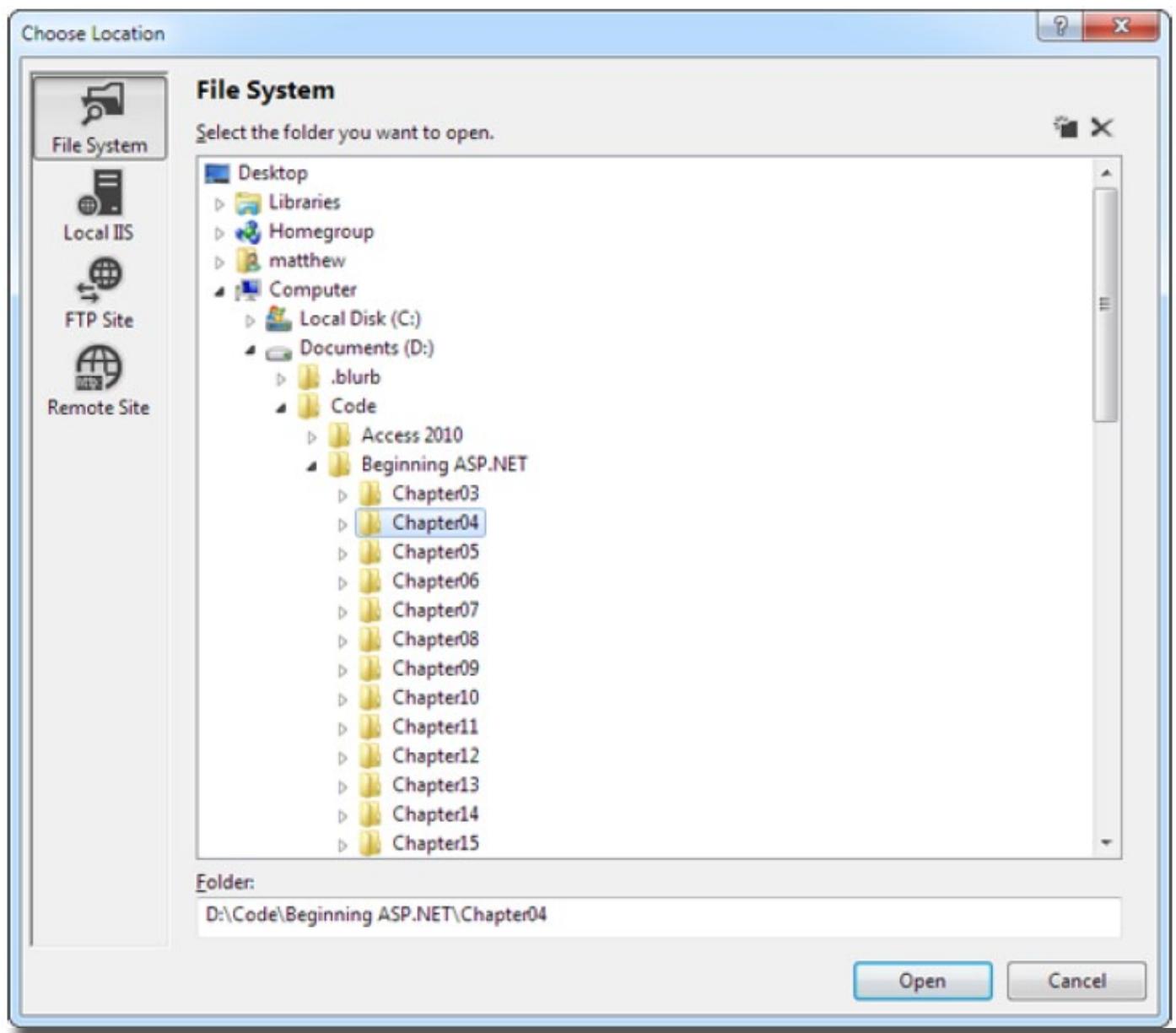
## WCF Service:

یک کتابخانه از متدهای سمت سرور که client های راه دور (مثلاً یک برنامه ویندوزی) می‌توانند از آنها استفاده نمایند.

## ASP.NET Reports Web Site:

این مدل یک وب سایت که از کنترل ReportView و سرویس SQL Server Reporting (ابزاری برای ایجاد گزارش‌های بانک اطلاعاتی که می‌تواند از طریق وب مدیریت و مشاهده شود) استفاده می‌کند، ایجاد خواهد کرد.

۵ - یک محل فیزیکی را برای ذخیره سازی وب سایت خود انتخاب کنید.



۶ - از کادر Choose Location برای جستجوی پوشه مورد نظر خود استفاده نمایید.

۷ - برای ایجاد وب سایت روی OK کلیک کنید. VS، یک وب سایت با یک فایل با نام web.config در آن ایجاد می‌کند. این فایلی است که می‌توانید تنظیمات مختلفی را در آن قرار دهید.

## تفاوت بین Web Project و Website

vS، معمولاً از فایل‌های پروژه برای ذخیره سازی اطلاعات در مورد برنامه ای که شما ایجاد کرده‌اید استفاده می‌کند. برنامه‌های تحت وب اندکی متفاوت می‌باشند، زیرا VS، لزوماً فایل‌های پروژه ای برای آنها ایجاد نمی‌کند. در حقیقت، اگر مراحل ذکر شده در بخش قبلی را انجام داده باشید، هیچ فایل پروژه ای برای شما ایجاد نشده است.

این سیستم که projectless development (توسعه بدون پروژه) نامیده می‌شود، با روشی که VS در مورد سایر انواع برنامه‌ها استفاده می‌کند متفاوت است. این روش طوری طراحی شده است که پوشه‌ی وی سایت شما را تمیز نگاه دارد، بنابراین توسعه و انتشار برنامه تحت وب شما را آسان می‌کند. بنابراین در زمان آپلود وب سایت روی یک وب سرور، می‌توانید کل فolder را بدون نگرانی از فایل‌های خارجی

که به منظور اهداف انتشار و توسعه استفاده می‌شند را، کپی کنید. بدون نیاز به symc کردن فایل‌های پروژه و solution، می‌توانید هر صفحه مشخص کار کنید.

### نوع دیگر، web project یا project base می‌باشد.

با کلیک روی لینک New Project یا کلیک روی File>New>Project >ASP.NET Web Application روی صفحه start می‌توانید یک web project ایجاد کنید.

این مدل از پروژه‌ها، همه ویژگی‌های projectless website ها را دارد می‌باشند، علاوه بر آنکه از فایل‌های اضافی (با پسوند .csproj) نیز استفاده می‌کنند.

فایل‌های web project، فایل‌های پیکربندی، صفحات وب و سایر منابع را نگهداری می‌کنند. فایل‌های پروژه در همان پوشه ای که همه صفحات وب و کد شما وجود دارند ذخیره می‌شود.

### چندین دلیل برای استفاده از web project وجود دارد :

اگر یک web project با VS 2005 ایجاد کرده باشید که بخواهید در VS 2012 آن را باز کنید، این کار بصورت خودکار انجام می‌شود.

اگر بخواهید دو web project را درون یک پوشه website مشابه قرار دهید، این دو پروژه را یک برنامه تحت وب در نظر می‌گیرد. با وجود web project ها، شما این انعطاف پذیری را دارید که در VS روی فایل‌ها بصورت جداگانه کار کنید.

شما از بل دارای وب سایتی هستید که دارای تعداد زیادی از فایل‌های منبع می‌باشد (به عنوان مثال، هزاران عکس). حتی اگر این فایل‌ها بخشی از وب سایت شما باشند، ممکن است نخواهید که آنها را در پنجره Solution explorer در VS مشاهده کنید زیرا محیط برنامه نویسی شما را بسیار کند می‌کنند. اگر از web project ها استفاده کنید، می‌توانید این فایل‌ها را به پروژه اضافه نکنید.

می‌خواهید با استفاده از MSBuild، Visual Studio Web Package، یا Visual Studio Web Package، راه انتشار برنامه تحت وب خود را روی یک سرور خودکار سازید.

### کار با فایل‌های مخفی شده در Solution

همانطور که دیدید، VS به شما امکان ایجاد برنامه های ASP.NET بدون فایل‌های پروژه را می‌دهد. اما کماکان یک نوع از فایل‌های منبع که solution file نامیده می‌شود را ایجاد می‌کند. مفهوم مشابهی با پروژه دارد با این تفاوت که یک solution تنها می‌تواند دارای یک یا چند پروژه باشد. هر زمان که شما با VS کار می‌کنید، دارید با یک solution کار می‌کنید.

VS، فایل‌های solution را در مسیری مانند زیر قرار می‌دهد :

```
c:\Users\[UserName]\Documents\Visual Studio 2012\Projects\[WebsiteFolderName]
```

هر solution، دارای دو solution file با پسوند های .sln و .suo می‌باشد.

به عنوان مثال اگر یک sloution با نام samplesite داشته باشیم دارای دو فایل زیر نیز هستیم :

SampleSite.sln

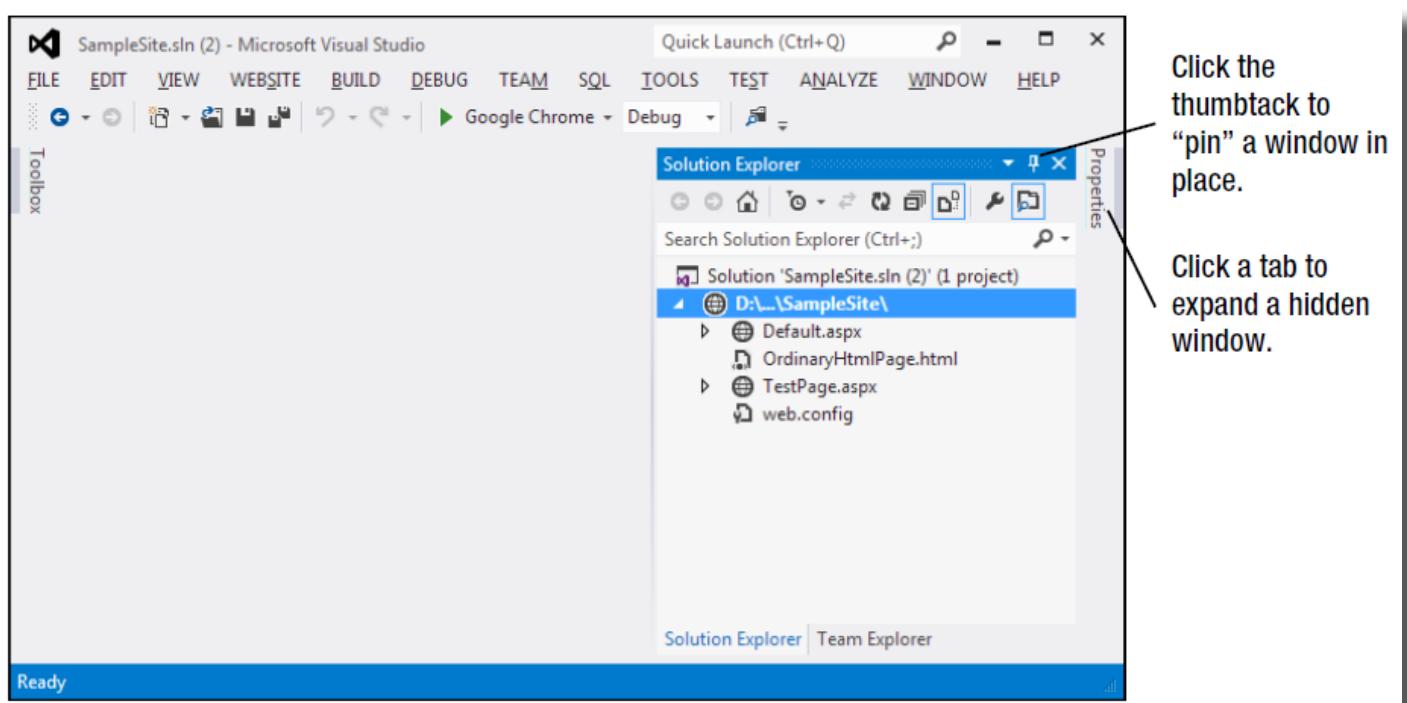
SampleSite.suo

بعضی از جزئیات خاص VS را که مستقیماً به ASP.NET مربوط است، مانند Debugging و تنظیمات نمایش آن، در خود نگهداری می‌کنند. به عنوان مثال، VS فایل‌هایی که در حال حاضر باز هستند را ردیابی می‌کند تا در دفعات بعدی که را باز کردید بتواند همان فایل‌ها را برای شما باز کند و نمایش دهد.

## استفاده از Solution Explorer

این پنجره در سمت راست VS وجود دارد و تمامی فایل‌های موجود در برنامه را برای شما لیست می‌کند. این پنجره را می‌توانید از مسیر زیر نیز مشاهده کنید.

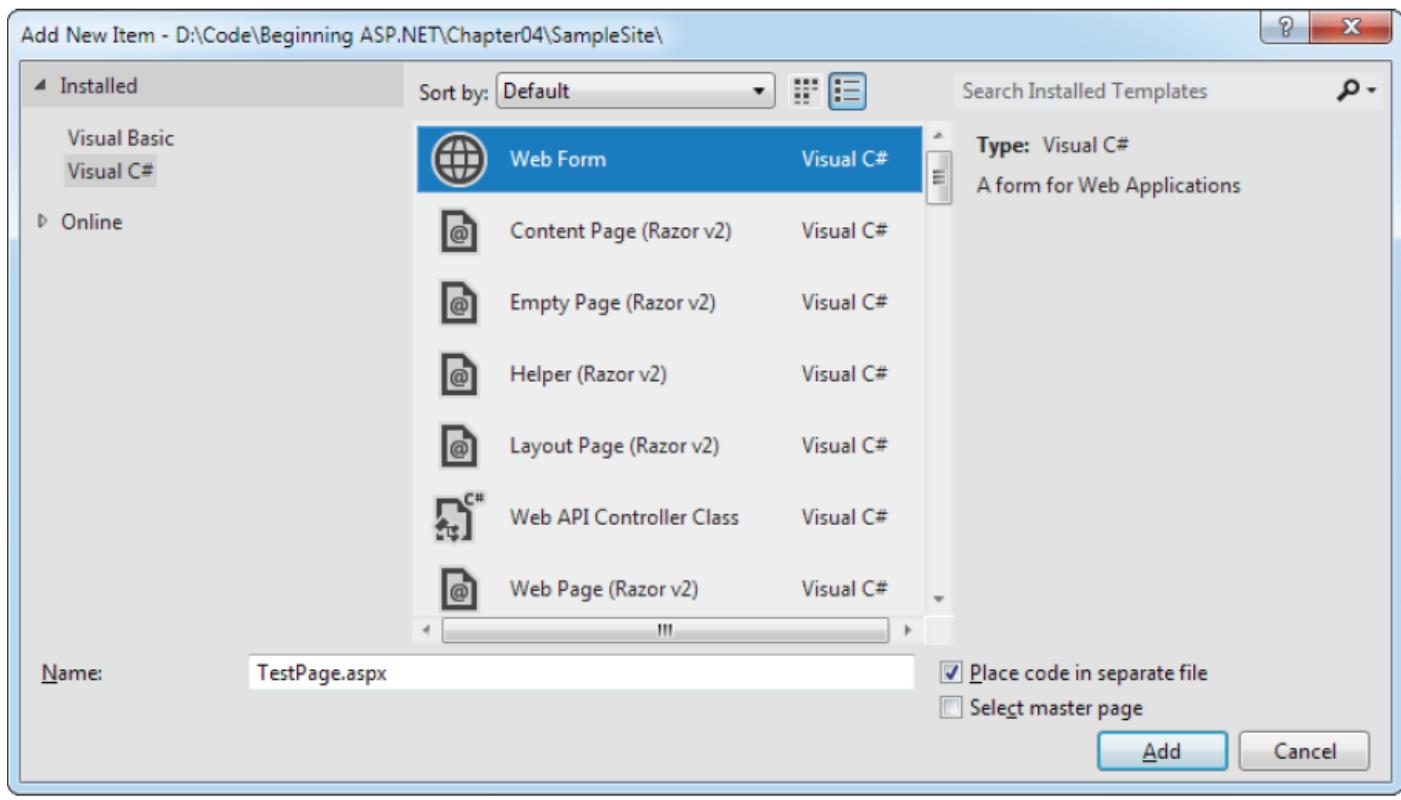
View > Sultion Explorer



این پنجره هر چیزی که در پوشه یک projectless website مشاهده می‌کنید را نمایش می‌دهد. هیچ فایلی مخفی نیست. بنابراین اگر فایلی را درون این پوشه اضافه کنید با کلیک راست روی وب سایت و انتخاب refresh می‌توانید آن فایل را در این پنجره مشاهده کنید.

## اضافه کردن Web Form

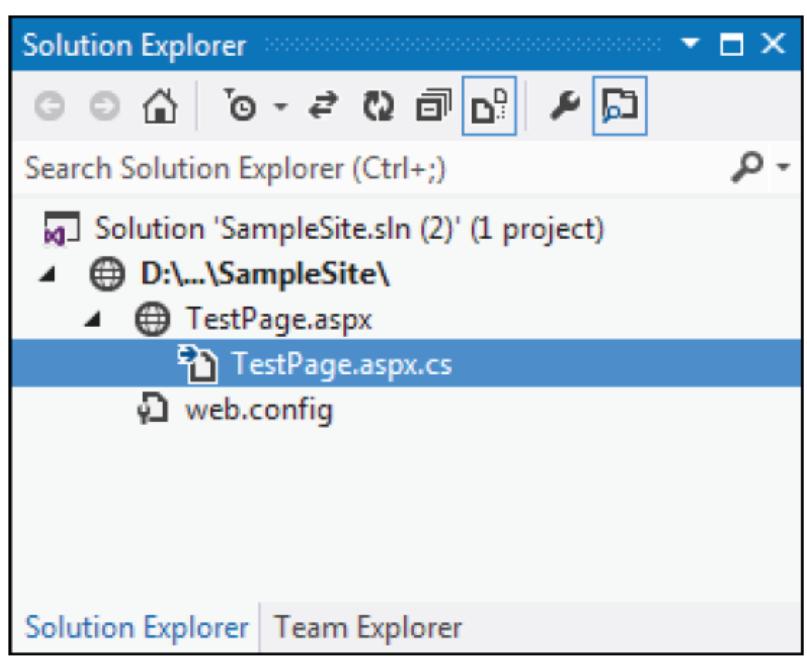
برای اضافه کردن یک صفحه وب جدید یا سایر موارد دیگر، روی Website>Add New Ites کلیک کنید. در کادر بازشده می‌توانید انواع مختلفی از فایل‌ها را اضافه کنید. برای اضافه کردن یک صفحه وب، گزینه Web Form در این کادر انتخاب نمایید.



گزینه Place Code in a Separate File اجازه انتخاب مدل کد نویسی را به شما می‌دهد. اگر این گزینه را انتخاب نکنید، فقط یک فایل برای صفحه وب شما اختصاص می‌دهد. بنابراین باید کدهای C# و HTML را در یک فایل قرار دهید. اگر این گزینه را انتخاب نماید، VS دو فایل مجزا برای صفحه وب شما (یک فایل برای کد markup و یک فایل برای کد C#) ایجاد می‌کند. این کار مدیریت شما در هنگام کار با صفحات پیچیده را راحت‌تر می‌کند.

همچنین می‌توانید گزینه ای دیگری به نام Select Master Page را که به شما اجازه ایجاد یک صفحه که برای Layout استاندارد شما استفاده می‌شود می‌دهد.

روی دکمه Add کلیک کنید.



همچنین اگر فایلی را از قبل درست کرده‌اید، می‌توانید روی website>add existing item کلیک کنید تا فایل شما در وب سایتتان کپی شود.

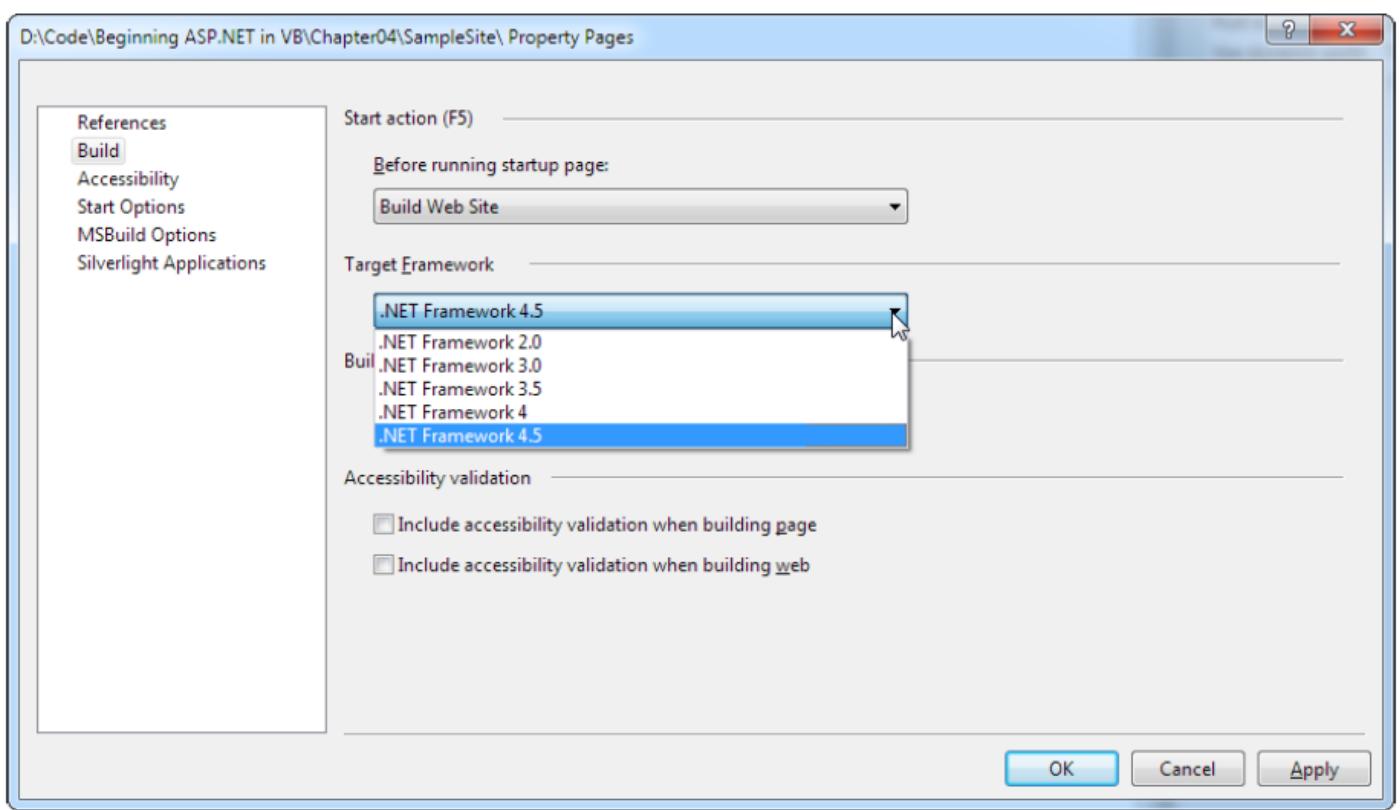
## Multitargeting

Multitargeting، یعنی ایجاد برنامه‌های تحت وب که با .NET 2.0 و .NET 3.0 و .NET 4 و .NET 4.5 کار کنند. می‌توانید حتی بعد از ایجاد وب سایت، نسخه فریمورک دات نت آن را تغییر دهید.

۱ - گزینه WebSite > Start را انتخاب کنید تا یک کادر برای شما باز شود.

۲- در لیست سمت چپ گروه بندی Build را انتخاب کنید.

۳- در لیست Target Framework، نسخه دات نتی که می‌خواهید را مشخص نمایید.



زمانی که نسخه .NET را انتخاب کردید، تنظیمات موجود در فایل Web.Config را تغییر می‌دهد.

## باز کردن یک وب سایت از درون نسخه قبلی VS

در نسخه‌های قبلی VS، برای باز کردن یک وب سایت قدیمی، نیاز به فرایند مهاجرت از نسخه قدیم به جدید (migration) داشتید. این کار در VS 2012 تغییر کرده، زیرا یک ویژگی به اسم project compatibility به آن اضافه شده است. با تشکر از این ویژگی می‌توانید یک وب سایت ایجاد شده با VS 2010 را به همان راحتی ایجاد یک وب سایت با VS 2012 باز نمایید. دیگر هرگز هیچ اختصار، پیام یا قادر محاوره‌ای مشاهده نخواهد شد و مجبور به گذراندن مراحل upgrade یا ارتقا نیستید. البته توجه داشته باشید که برای پروژه‌هایی که قبل از VS 2010 می‌باشند، مراحل ارتقا کماکان وجود دارد.

ممکن است بخواهید در هنگام تست برنامه به جای استفاده از ASP.NET Developmenet Server از IIS Express استفاده نمایید. برای این کار گزینه WebSite > Use IIS Express را انتخاب (در ادامه توضیح داده خواهد شد) کنید.



## طراحی یک صفحه وب

برای ایجاد یک صفحه وب ساده، روی صفحه ای که می‌خواهید کلیک مضاعف کنید.

VS، به شما سه راه برای مشاهده یک صفحه.aspx. ارائه خواهد داد:

**Design View**: در اینجا یک نمای گرافیکی از آنچه که صفحه شما به نظر خواهد رسید مشاهده خواهد کرد.

**Source View**: در اینجا کدهای HTML با استفاده از markup برای صفحه و تگ‌های کنترل‌های ASP.NET ارائه خواهد شد.

**Split View**: ترکیب دو نمای بالا می‌باشد که به شما اجازه مشاهده هر دو نما را همزمان خواهد داد.

## اضافه کردن کنترل‌های وب

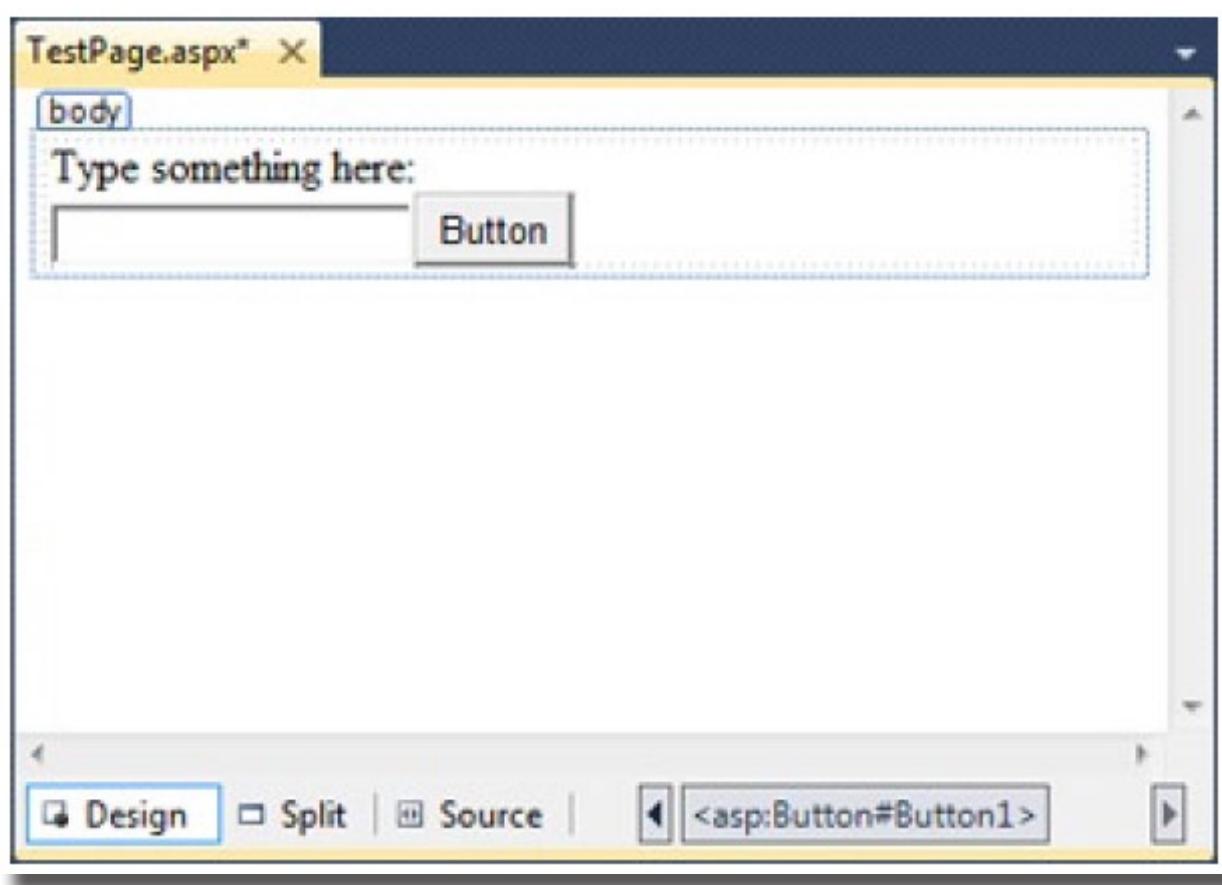
برای اضافه نمودن کنترل‌های وب ASP.NET، آن را از جعبه ابزار موجود در سمت چپ کشیده و به داخل صفحه وب خود بیندازید. اگر از design view استفاده می‌کنید، قرار دادن کنترل در مکان صحیح معمولاً بسیار راحت می‌باشد. اگر کنترل را در source view

بیندازید، ممکن است درون تگ <form> نیفتد که در این صورت به درستی کار نمی‌کند.

کنترل‌های موجود در جعبه ابزار بر اساس عملکردشان به گروه بندی‌های مختلفی تقسیم می‌شوند، اما می‌توانید کنترل‌های پایه مانند lable ,text box ,button و ... را درون زبانه Standard مشاهده کنید.

زمانی که ماوس را از روی جعبه ابزار دور می‌کنید، بصورت اتوماتیک مخفی می‌شود. می‌توانید با کلیک روی آیکن موجود در گوشه راست در بالای جعبه ابزار محل آن را در VS ثابت کنید. اگر آن را ببندید، می‌توانید از View > Toolbox آن را مجدد باز کنید.

کنترل‌های درون یک فرم وب بصورت خط به خط قرار می‌گیرند. برای سازماندهی چند کنترل درون design view، ممکن است نیاز داشته باشد، فضاهایی برای قرار دادن المان‌ها در محلی که می‌خواهید اضافه نمایید.



خواهید دید که بعضی از کنترل‌ها نمی‌توانند تغییر سایز دهند. در عوض، می‌توانید برای تنظیم آنها با محتوای درونشان، آنها را بزرگ و کوچک کنید. برای مثال اندازه یک کنترل label، بسته به آن دارد که چه مقدار متنه درون آن قرار می‌گیرد. در سمت دیگر می‌توانید اندازه یک دکمه یا textbox را با کلیک روی آن و کشیدن آن تغییر دهید.

همانطور که شما وب کنترل‌ها را اضافه می‌کنید، VS، بصورت خودکار، تگ کنترل مرتبط با آن را به فایل aspx. اضافه خواهد کرد.

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TestPage.aspx.cs" %>

<!DOCTYPE html>

<html>
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div style="margin: 3px">
            <asp:Label ID="Label1" runat="server"
                Text="Type something here:" />
            <br />
            <asp:TextBox ID="TextBox1" runat="server" />
            <asp:Button ID="Button1" runat="server" Text="Button" />
        </div>
    </form>
</body>
</html>

```

The screenshot shows the Visual Studio IDE with the Source tab selected for the file TestPage.aspx. The code displays an ASP.NET page structure with a form containing a label, a text box, and a button, all enclosed within a div with a margin of 3 pixels. The code uses the absolute positioning style property to achieve this layout.

اگر از نسخه های قبلی VS استفاده کرده باشید، به یاد خواهید آورد که یک ویژگی با نام grid layout است که در آن وجود داشت که با استفاده از کنترل ها هر جایی که می خواستید، به شما اجازه قرار دادن المان ها با سازگاری absolute، می داد. اگرچه این مدل، مناسب به نظر می رسید، ولی در واقع برای اکثر صفحات وب مناسب نبود چون کنترل ها نمی توانستند هنگامی که محتوای صفحه وب تغییر می کرد (یا متن آن بر اساس خواست کاربر تغییر می کند)، محل قرار گرفتن خود را تنظیم کنند. بنابراین ممکن بود کنترل ها روی همدیگر قرار بگیرند.

```

<asp:Button ID = "cmd" style = "POSITION: absolute; left: 100px; top: 50px;">
    runat = "server" Text = "Floating Button" ... />

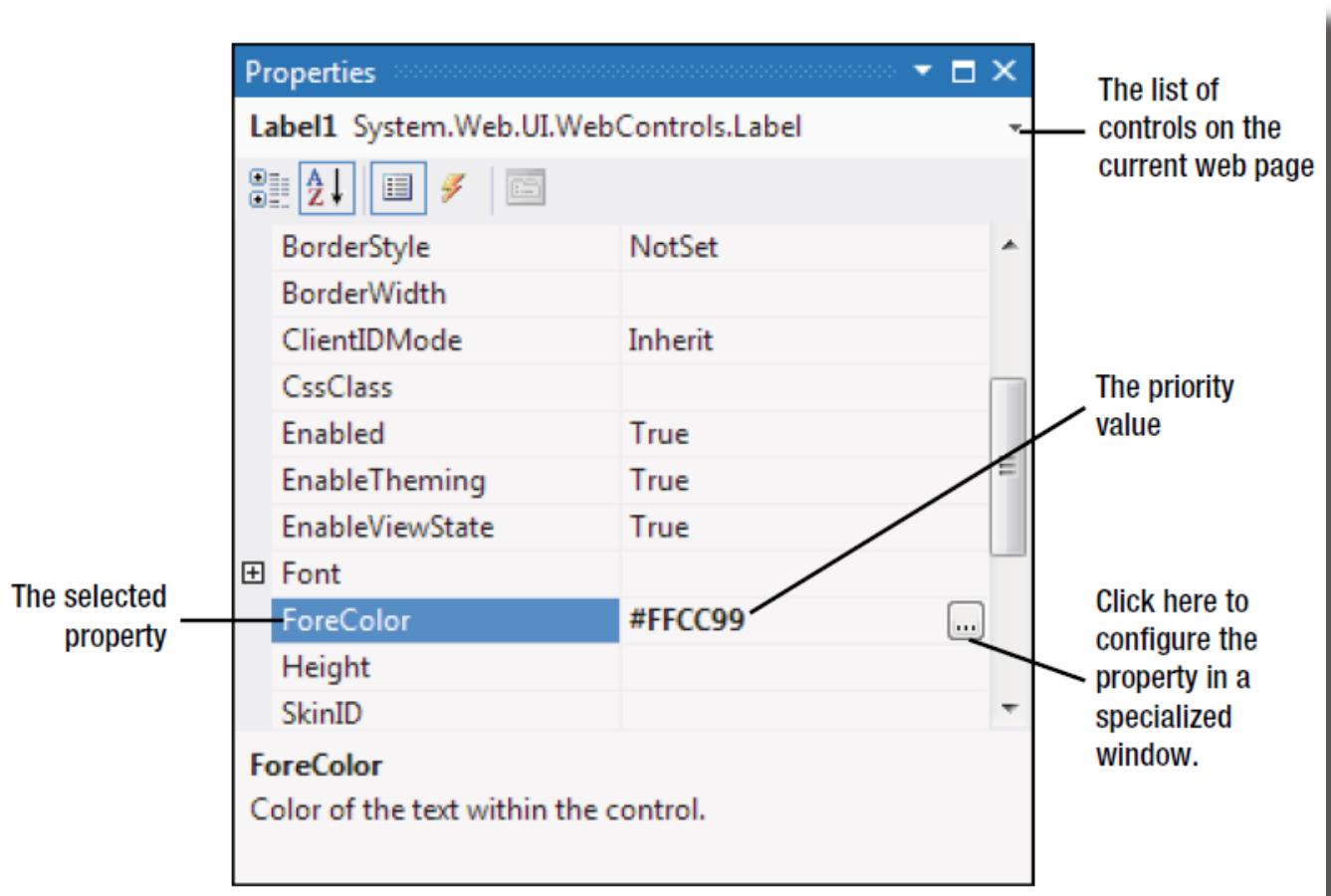
```

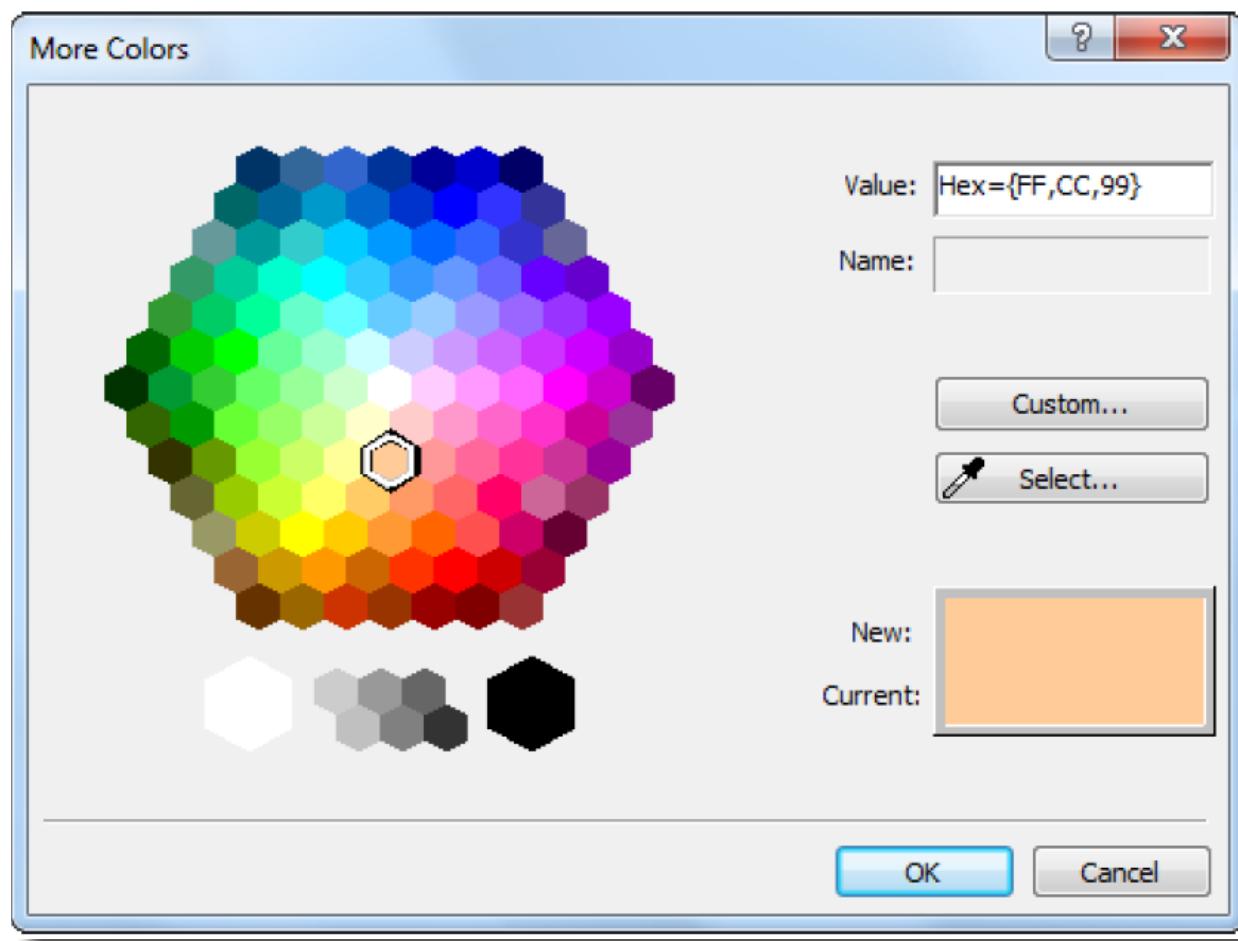
اگر می خواهید از absolute position استفاده نمایید، بهترین ایده بکارگیری یک container (کنترلی که بتوان درون آن از کنترل های دیگر استفاده کرد)، مانند المان <div> می باشد که یک قادر که در حالت عادی غیر قابل دسترس می باشد را در اختیار قرار می دهد و می توانید رنگ پس زمینه، border و سایر فرمتهای آن تنظیم نمایید. با استفاده از absolute position از المان <div> خود را دقیقاً در محل مورد نظر قرار داده اما برای محتوای درون آن از normal flow layout استفاده کنید. این کار به سادگی مقدار استفاده از layout ها برای شما کاهش می دهد. برای نمونه اگر بخواهید یک نوار کناری (sidebar) با لیستی از لینک ها درون آن داشته باشید، بسیار ساده تر است که sidebar را با absolute position قرار دهید و سپس هر لینک را بصورت جداگانه در محل مناسب درون آن قرار دهید.

## Properties از پنجره استفاده

پس از آنکه یک کنترل وب به صفحه خود اضافه کردید، ممکن است بخواهید تغییراتی در آن ایجاد کنید. مثلاً بخواهید متن روی یک دکمه، رنگ label ... را تغییر دهید. همه این کارها را می‌توانید با تغییر در کدهای markup بصورت دستی انجام دهید. VS یک راه ساده‌تر نیز برای شما بوجود آورده است. در زیر پنجره Solution Explorer، در سمت راست – پایین، می‌توانید خصوصیت‌های کنترل انتخاب شده خود را درون پنجره Properties مشاهده کنید.

اگر این پنجره در دسترس قرار نداشت از مسیر زیر آن را فعال کنید. View > Properties Window (یا کلید F4 را فشار دهید).





## آناتومی یک صفحه وب

تا الان، با صفحه وب خود در design view کار کردیم. با توجه به اینکه کار در این مد راحت است ولی انجام بعضی از تغییرات در source view بسیار راحت‌تر می‌باشد.

کد یک صفحه ASP.NET، بصورت کامل و 100 درصد HTML نمی‌باشد. در حقیقت این صفحه یک سند HTML با جزئیات مربوط به کنترل‌های وب ASP.NET می‌باشد.

```
<%@ page language="#c" autoeventwireup="true" codefile="Default.aspx.cs" inherits="_Default" %>

<!DOCTYPE html>

<html>

<head id="Head1" runat="server">

    <title>Untitled Page</title>

</head>

<body>

    <form id="form1" runat="server">

        <div>

            <asp:Label ID="Label1" runat="server" Text="Type something here:" />

        </div>

    </form>

</body>
```

```

<br />

<asp:TextBox ID="TextBox1" runat="server" />

<asp:Button ID="Button1" runat="server" Text="Button" />

</div>

</form>

</body>

</html>

```

بدیهی است جزئیات مربوط به ASP.NET (پرنگ شده‌ها)، برای یک مرورگر وب معنی ندارند زیرا مرورگر وب تنها کدهای HTML می‌شناسد و آنها HTML معتبر نیستند. البته مرورگر وب هرگز این کدها را نخواهد دید زیرا ASP.NET engine، کدهای HTML صفحه وب شما را ایجاد می‌کند. در حقیقت جزئیاتی مانند <asp:Button>، با تگ‌های HTML که ظاهری مشابه دارند جابجا می‌شود و ASP.NET engine، این کد را به مرورگر می‌فرستد.

## Page Directive

صفحه Default.aspx مانند همه فرم‌های وب شامل سه بخش می‌باشد. اولین بخش آن راهنمای صفحه یا page directive می‌باشد.

```

<%@ Page Language = "#c" AutoEventWireup = "true"
CodeFile = "Default.aspx.cs" Inherits = "_Default" %>

```

راهنمای صفحه، اطلاعاتی درباره نحوه کامپایل شدن صفحه در اختیار قرار می‌دهد. این کد، زبانی که می‌خواهید برنامه را با آن بنویسید و روش اتصال به متدها یا event handler ها را مشخص می‌کند. این کد در HTML ای که به مرورگر فرستاده می‌شود وجود نخواهد داشت.

## Doctype

این بخش در زیر راهنمای صفحه قرار می‌گیرد و نوع زبان نشانه گذاری یا همان markup (برای نمونه HTML5 یا XHTML) را مشخص می‌کند. این کد اختیاری می‌باشد ولی VS، آن را بصورت خودکار به صفحه شما اضافه می‌کند. این بخش از این نظر مهم است که تعیین می‌کند صفحه وب شما چگونه در مرورگر وب تفسیر شود. اگر آن را تعیین نکنید، مرورگر از پیش فرض خود برای این کار استفاده می‌کند. برای آنکه صفحه وب شما در همه مرورگرها یکسان به نظر برسد این بخش را در آن قرار دهید.

امروزه اکثر صفحات وب از HTML5 doctype استفاده می‌کنند، مانند زیر:

```
<!DOCTYPE html>
```

در همه مرورگرها کار می‌کند. همچنین می‌توانید از استاندارد قدیمی‌تر HTML4 doctype استفاده کنید.

## مثالی از XHTML 1.0

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

مرورگرها صفحات HTML5 و XHTML را به یک روش پردازش می‌کنند.

## ملزومات HTML

### Elements

المان‌ها، container هایی برای نگهداری محتوای صفحات وب شما می‌باشند. برای نمونه اگر می‌خواهید یک پاراگراف از متن را به یک صفحه اضافه کنید، باید آن را درون المان پاراگراف قرار دهید. ترکیب المان‌های مختلف، ساختار یک صفحه وب را تعريف می‌کند. آنها همچنین نقاطی برای فرمت دادن به صفحات وب می‌باشند. HTML، برای استفاده از این المان‌ها دارای سینتکس خاصی می‌باشد.

تگ آغازی، محتوا، تگ پایانی

```
<p> This is a sentence in a paragraph.</p>
```

یا

```
<h1> A Heading</h1>
```

```
<p> This is a sentence in a paragraph.</p>
```

یا

```
<p> This is a <b> sentence</b> in a paragraph.</p>
```

یا

```
<p> This is line one. <br />
```

This is line two. <br />

This is line three.</p>

در این کتاب فرض بر این است که با المان‌های پایه ای HTML آشنا دارید. در غیر اینصورت حتماً آنها را مرور کنید.

## Attributes

هر سند HTML دارای دو نوع از اطلاعات می‌باشد : محتوای سند و اطلاعاتی در باره نحوه نمایش محتوای سند.

نمایش محتوای سند خود را از سه راه می‌توانید کنترل نماید:

- استفاده از المان‌های مناسب
- تنظیم این المان‌ها برای بدست آوردن ساختار صحیح
- اضافه نمودن خصیصه (attribute) ها به المان‌ها

Attribute ها، بخش‌های مجازی از اطلاعات می‌باشند که شما به یک المان اضافه می‌کنید. یکی از کاربردهای آن این است که می‌توانید یک style را برای تعیین فرمت خاصی برای یک المان به آن اضافه کنید. برای نمونه المان را در نظر بگیرید که برای نمایش تصاویر در صفحات وب از آن استفاده می‌شود. این المان دارای دو بخش اطلاعاتی می‌باشد. آدرس تصویر (image URL) و متن جایگزین (در صورت پیدا نشدن و عدم دسترسی به تصویر) آن.

این دو ویژگی با استفاده از دو attribute زیر با نام‌های src و alt، مشخص می‌شوند :

```
<img src = "happy.gif" alt = "Happy Face" />
```

## Formatting

المان‌های HTML برای تعیین ساختار یک سند بکار می‌روند و نه فرمت آن. اگرچه می‌توانید تعیین رنگ، فونت و بعضی از ویژگی‌های فرمت دهی را با استفاده از المان‌های HTML، انجام دهید ولی بهتر این است که از فرمت دهی با style sheet استفاده کنید. برای نمونه، style sheet می‌تواند به مرورگر بگوید که از فرمت خاصی برای هر المان 

# در صفحه استفاده کند. حتی می‌توانید یک استایل را برای همه صفحات موجود در وب سایت خود استفاده نمایید.

## یک صفحه وب کامل

هر صفحه وب با ساختار زیر شروع می‌شود:

```
<img src = "happy.gif" alt = "Happy Face" />
```

```
<html>
  <head id=""Head1"" runat=""server"">
    <title>Untitled Page</title>
  </head>
  <body>
  </body>
</html>
```

سند HTML، با تگ <html> شروع می‌شود و با تگ </html> پایان می‌گیرد. این المان <html> شامل محتوای کامل یک صفحه

وب می‌باشد.

درون این المان <html> صفحه وب به دو بخش تقسیم می‌شود. بخش اول المان <head> می‌باشد که اطلاعاتی در باره صفحه وب را در خود ذخیره می‌کند. می‌توانید عنوان صفحه خود را که در نوار بالای مرورگر وب نمایش داده خواهد شد را در این بخش قرار دهید. می‌توانید کلمات کلیدی برای جستجو در موتورهای جستجوگر را در اینجا قرار دهید که البته توسط مرورگرهای وب امروزی نادیده گرفته می‌شود. تگ <head> که در VS ساخته می‌شود دارای ویژگی runat="server" می‌باشد که اجازه دستکاری آن را در کد خواهد داد.

بخش دوم المان <body> است که شامل محتوای اصلی صحه می‌باشد که در پنجره مرورگر وب نمایان می‌شود.

در صفحات ASP.NET، حداقل یک المان دیگر به نام <body> نیز وجود دارد که درون تگ <form> قرار می‌گیرد. این بخش می‌تواند اطلاعات را به وب سرور برگرداند. برای نمونه، اطلاعات پر شده در یک textbox باید به مرور فرستاده شود.

```
<html>
  <head id="Head1" runat="server">
    <title>Untitled Page</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
      </div>
    </form>
  </body>
</html>
```

## نوشتن کد

برای آغاز کد نویسی باید به view code-behind page می‌روید. زمانی که به view code می‌روید، کلاس page مربوط به صفحه وب خود را می‌بینید.

```
using System;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```

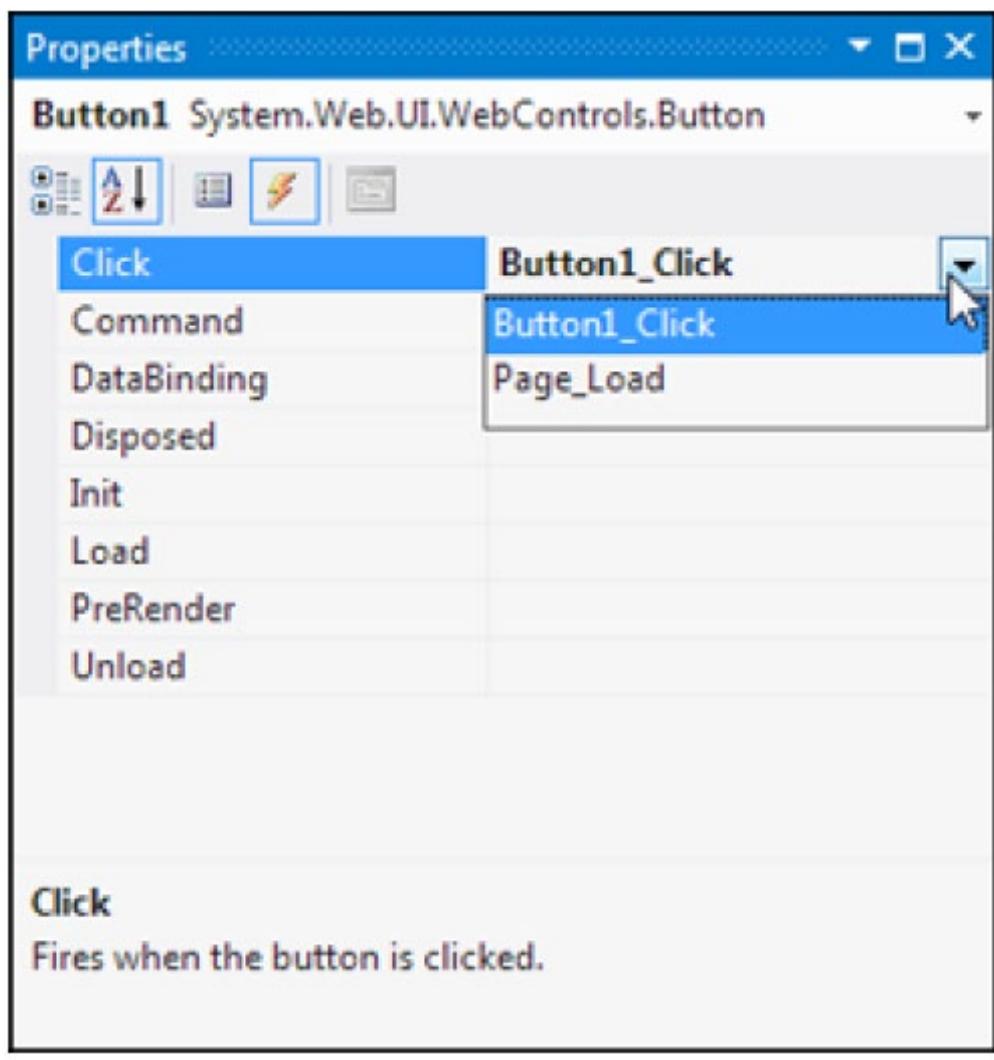
public partial class SimplePage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}

```

## اضافه کردن Event Handler

بیشتر کدهای موجود در صفحه ASP.NET، درون event handler هایی که به رویدادهای کنترل‌های وب عکس‌العمل نشان می‌دهند، قرار گرفته‌اند. با استفاده از VS، سه راه برای اضافه نمودن آنها به کد وجود دارد :

- نوشتن دستی آن‌ها که مستقیماً درون کلاس صفحه در کد فایل C# شما اضافه می‌شود.
- کلیک مضاعف روی یک کنترل در محیط VS. Design برای رویداد پیش فرض کنترل اضافه خواهد نمود. برای نمونه اگر روی یک دکمه دوبار کلیک کنید، VS، یک event handler برای رویداد Click آن ایجاد می‌کند.
- انتخاب رویداد از پنجره Properties : فقط لازم است کنترل مورد نظر را انتخاب کنید و لیست رویدادهای مربوط به این کنترل را در پنجره Properties را مشاهده کنید. روی رویداد مورد نظرتان دوبار کلیک کنید تا VS بصورت خودکار eventhandler آن را ایجاد نماید.



کد موجود در صفحه .aspx

```
<asp:Button ID = "Button1" runat = "server" Text = "Button" OnClick = "Button1_Click" />
```

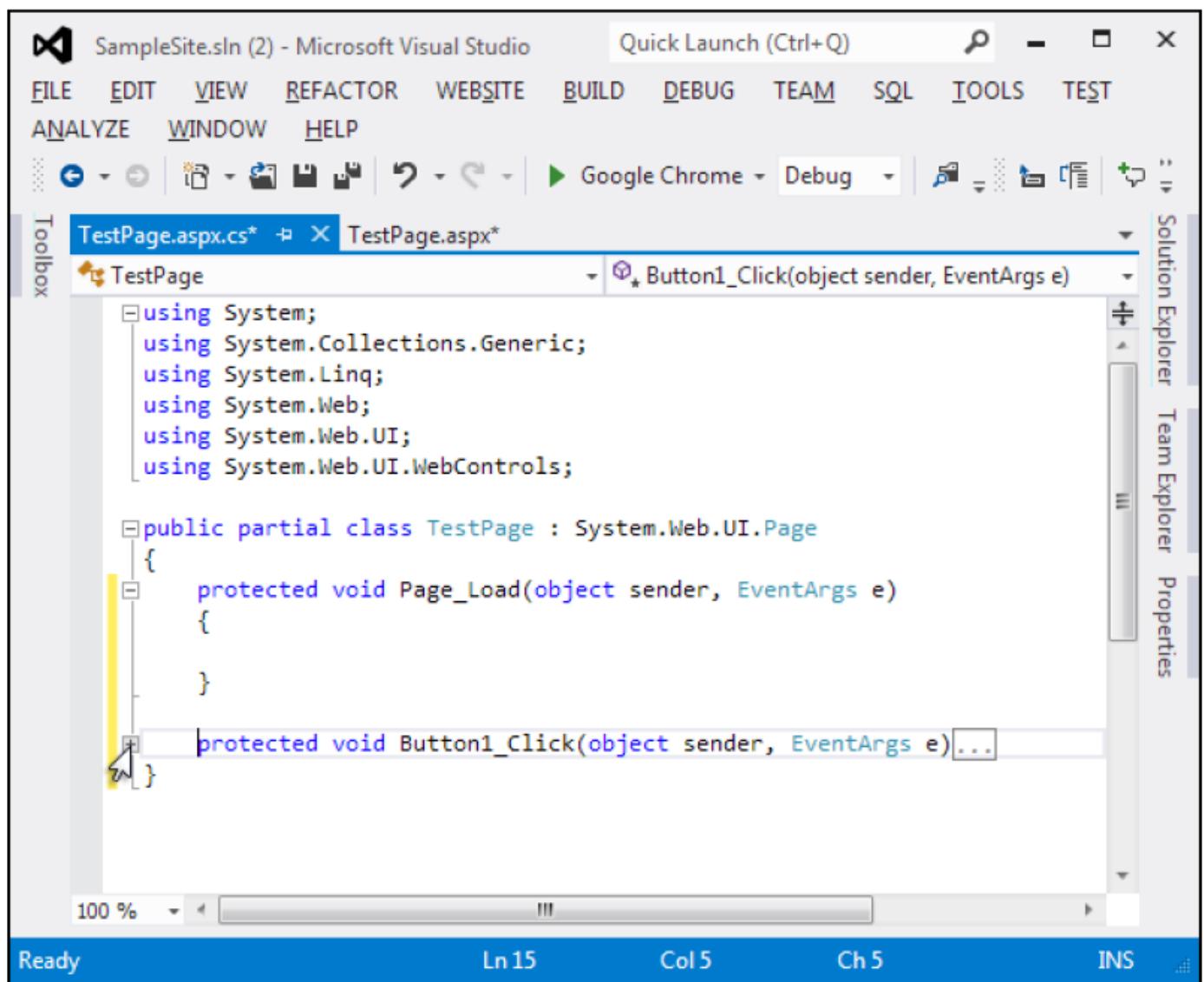
کد موجود در :.aspx

```
protected void Button1_Click(object sender, EventArgs e)
{
    // Your code for reacting to the button click goes here.
}
```

```
protected void Button1_Click(object sender, EventArgs e)
{
    Button1.Text = "Here is some sample text.";
}
```

## Outlining (ترسیم)

Collapse، به اجازه Outlining (جمع کردن متن کد) متده، کلاس و... یا یک ناحیه را به یک خط خواهد داد. برای کردن بخشی از کد، روی علامت (-) در کنار خط اول کلیک کنید. برای expand (باز کردن) کردن آن روی کادر مجدداً کلیک کنید که دارای علامت (+) می‌باشد.



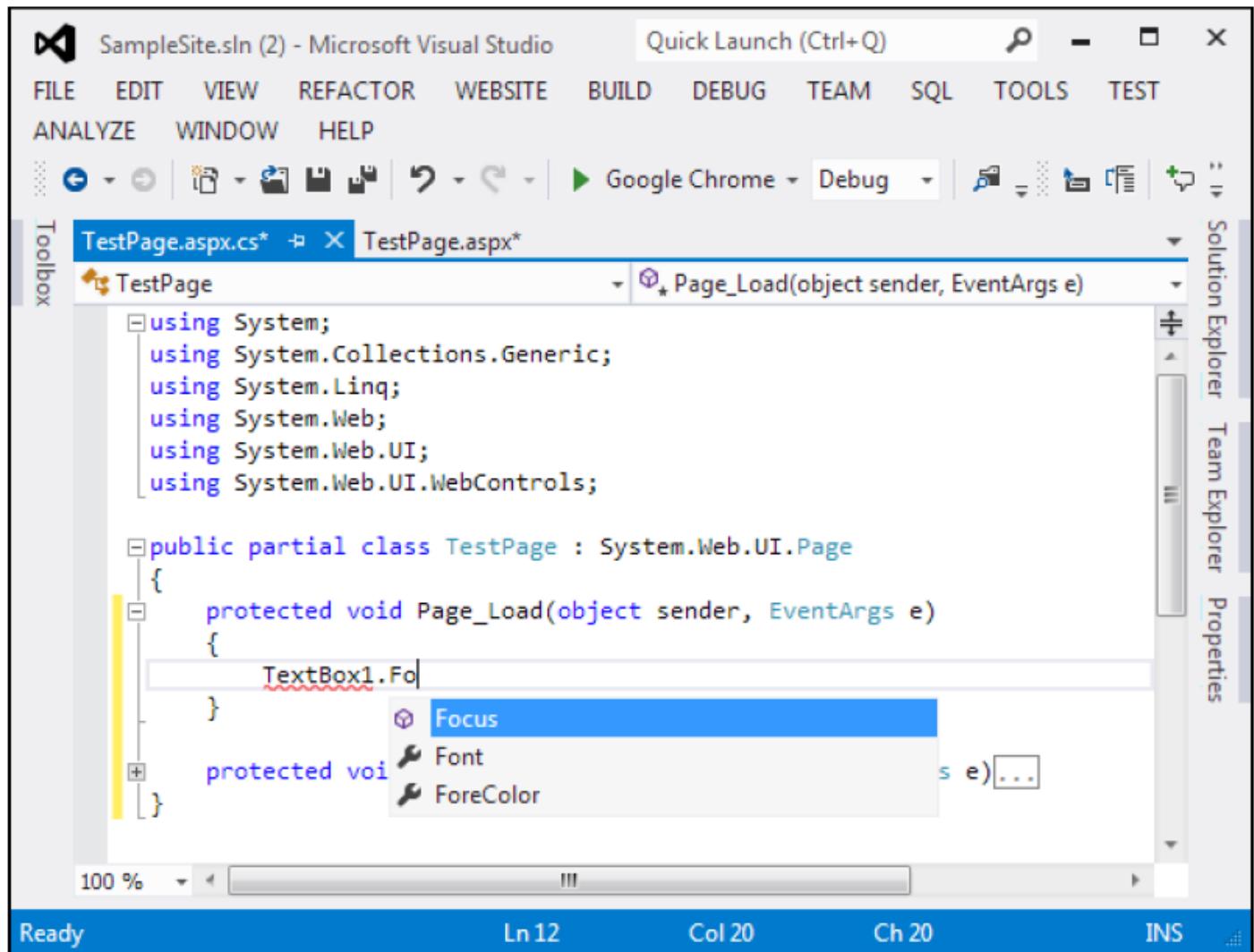
می‌توانید هر متده را با کلیک راست روی هرجایی در پنجره کد و انتخاب Outlining>Collapse to Definition مخفی کنید.

## IntelliSense

VS، برای ساده کردن کار و صرفه جویی در زمان شما تلاش‌های بسیاری کرده است. مهم‌ترین آنها Intellisence است که مجموعه‌ای از پیشنهادهای کد در هنگام تایپ آن می‌باشد.

## استفاده از لیست اعضا

زمانی که نام یک کلاس یا شی را تایپ می‌کنید، لیستی از property‌ها و متدهای قابل دسترس که با متنی که تا آنجا تایپ کرده‌اید و مطابقت خواهد داشت را به شما نشان می‌دهد.



در صورتیکه نام کنترل‌های درون صفحه وب را فراموش کرده باشید، کلمه this را به همراه یک نقطه تایپ کنید. VS، کلیه متدها و property‌های موجود در این کلاس فرم را نمایش می‌دهد.

The screenshot shows the Visual Studio IDE with the code editor open. The file is named 'TestPage.aspx.cs'. The cursor is hovering over the 'Page.Validate()' method call. A tooltip box appears, containing the text: 'Instructs any validation controls included on the page to validate their assigned information.' This illustrates how VS provides detailed documentation through its tooltip system.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Page.Validate();
    }
}

```

## گرفتن خطا در کد

VS، قادر به شناسایی انواع خطاهای مانند متغیرها، undefined property ها و متدهای تبدیلات انواع داده ای نامعتبر و ... می باشد و زیر کد مشکل دار، خط می کشد. می توانید ماوس خود را روی خطابردت تا توضیحات آن را در tooltip ای که نمایش داده می شود مشاهده کنید.

The screenshot shows the Visual Studio IDE with the code editor open. The file is named 'TestPage.aspx.cs'. There is a syntax error in the code where 'Tex' is misspelled as 'Tex'. A tooltip box appears, containing the text: "'System.Web.UI.WebControls.TextBox' does not contain a definition for 'Tex' and no extension method 'Tex' accepting a first argument of type 'System.Web.UI.WebControls.TextBox' could be found (are you missing a using directive or an assembly reference?)'. This is a common way VS highlights errors in the code.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        TextBox1.Tex = "Hello";
    }
}

```

VS، لزوماً خطاه را فوراً نشان گذاری نمی‌کند. اما در هنگامی که برنامه را اجرا (run) یا حتی فقط کامپایل (build) می‌کنید، کل کد را اسکن می‌کند و تمام خطاهایی راه که پیدا کند را نشانه گذاری می‌کند. حتی اگر یک خطای نیز وجود داشته باشد، از شما خواهد پرسید که آیا باید به کار خود ادامه دهد یا نه. اکثراً گزینه cancel را انتخاب خواهد کرد و خطای را رفع می‌کند. ولی در صورتیکه continue را انتخاب کنید، آخرین نسخه کامپایل شده برنامه اجرا می‌شود زیرا VS، نمی‌تواند برنامه‌ای که دارای خطای می‌باشد را Build کند.

Error List				
	2 Errors	0 Warnings	0 Messages	
Description	File	Line	Column	Project
1 'System.Web.UI.WebControls.TextBox' does not contain a definition for 'Tex' and no extension method 'Tex' accepting a first argument of type 'System.Web.UI.WebControls.TextBox' could be found (are you missing a using directive or an assembly reference?)	TestPage.aspx.cs	12	18	D:\Code\SampleSite\
2 The name 'saveFlag' does not exist in the current context	TestPage.aspx.cs	13	15	D:\Code\SampleSite\

ممکن است پس از آنکه که خطاه را برطرف و برنامه را rebuild کردید، با خطاهای بیشتری روبرو شوید. این امر به این دلیل است که VS، همه انواع خطاه را در یک زمان چک نمی‌کند. هنگامی که برنامه را کامپایل می‌کنید، VS، خطاهای پایه ای مانند اسامی کلاس‌های غیر قابل تشخیص و ... را اسکن می‌کنید. اگر این خطاه وجود داشته باشند، سایر انواع خطاه را مورد بررسی قرار نمی‌دهد. به عبارت دیگر، اگر کد شما این سطح از بازبینی را بگذراند، VS، آن را برای خطاهای بیشتری نظیر استفاده از unassigned variable، چک خواهد کرد.

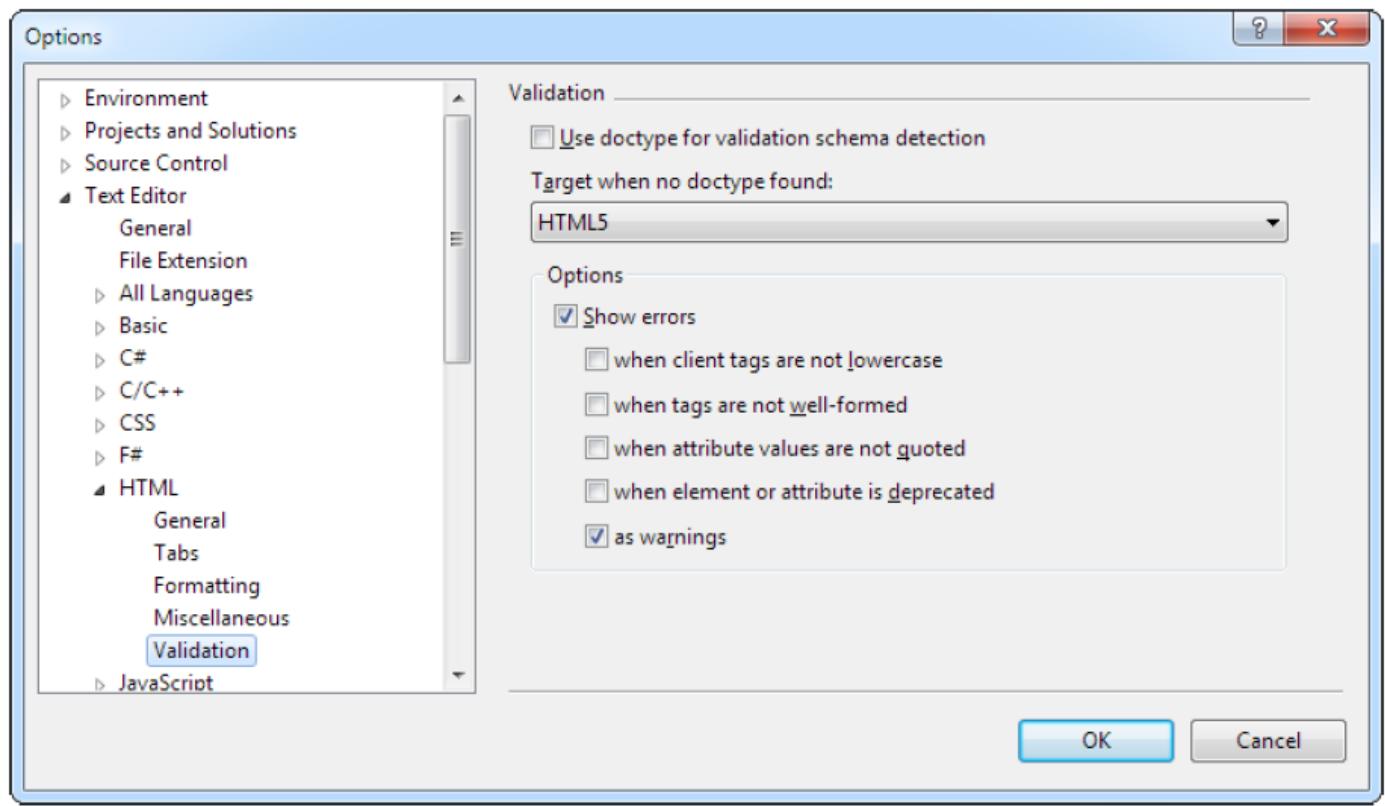
## Markup و کنترل خطای

شما همچنین می‌توانید سطح چک کردن خطاه در VS را برای فایل‌های .aspx، پیکربندی کنید، اما قبل از این کار باید تصمیم بگیرید از چه قانونی می‌خواهید پیروی کنید.

برای نمونه در HTML5 شما می‌توانید عنوانین تگ را با حروف بزرگ داشته باشید. (`<P>` به جای `<p>`). یا تگ‌های خالی بدون (/) در پایان. (`<br>` به جای `<br/>`).

برای تعیین چگونگی اعتبارسنجی کدهای markup صفحه وب خود :

- گزینه Tools> Options را انتخاب کنید تا کادر Options باز شود.
- در درخت سمت چپ، به بخش Text Editor> HTML > Validation بروید.



- گزینه Use Doctype for Validation Schema Detection را انتخاب کنید. این یعنی doctype صفحه وب را مشاهده می‌کند و قوانین مرتبط با آن را در هنگام چک کردن خطا‌های markup بکار می‌گیرد. اگر این گزینه را انتخاب نکنید، قانونی که شما در HTML Source Editing تعیین کرده‌اید (مانند HTML5 و XHTML و ...) را انتخاب می‌کند.
- اطمینان حاصل کنید گزینه Show Errors انتخاب شده است.
- سایر گزینه‌ها را مرور کنید و در صورت نیاز انتخاب نمایید.

## Import کردن خودکار فضای نام

گاهی به دلیل آنکه یک فضای نام را Import نکرده‌اید با خطای روبرو می‌شوید.

```
FileStream fs = new FileStream("newfile.txt", FileMode.Create);
```

این خط یک نمونه از کلاس `FileStream` ایجاد خواهد کرد که درون فضای نام `System.IO` موجود می‌باشد. اگر فضای نام مربوطه را Import نکرده باشید، در هنگام کامپایل با خطای روبرو می‌شوید. VS، یک ابزار مناسب برای کمک به شما ارائه داده است. زمانی که مکان نما را روی نام کلاس ناشناس قرار دهید، آیکن `page` ظاهر خواهد شد. روی این آیکن کلیک کنید تا لیست تصحیح خودکار نمایش داده شود. با استفاده از این گزینه می‌توانید آن خط از کد را به کد با بکارگیری نام کامل کلاس تبدیل کنید یا فضای نام مورد نظر را در بالای کد Import کنید که روش تمیزتری می‌باشد زیرا اگر در محل‌های دیگری از کد نیز از این فضای نام استفاده می‌کنید، نیاز به نوشتن نام کامل کلاس به همراه نام فضای نام نیستید.

```

TestPage.aspx.cs* X
TestPage Page_Load(object sender, EventArgs e)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        FileStream fs = new FileStream("readme.txt", FileMode.Open);
    }
}

```

System.IO.FileStream fs = new FileStream("newfile.txt", FileMode.Create);

لی

```

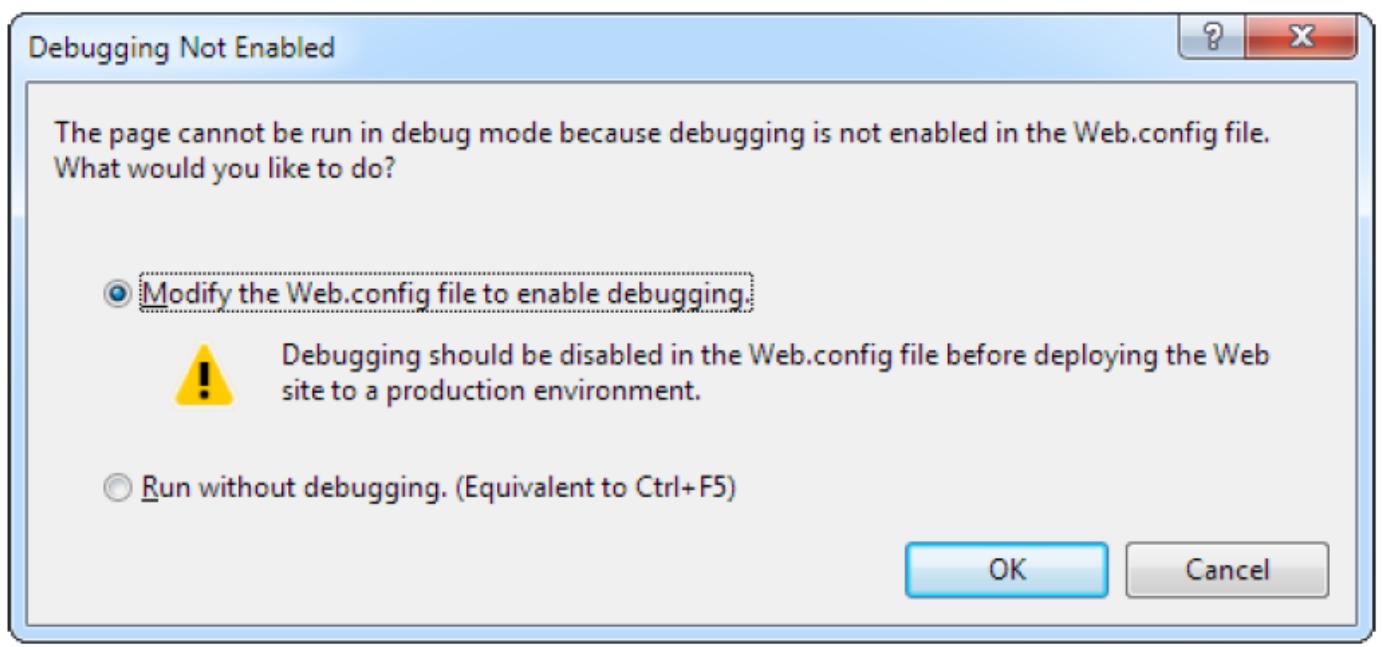
using System.IO;

namespace WebApplication1
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            FileStream fs = new FileStream("newfile.txt", FileMode.Create);
        }
    }
}

```

## Debugging (عیب یابی - اشکال زدایی)

پس از ایجاد یک برنامه، می‌توانید آن را کامپایل و اجرا نمایید. گزینه Debug > Start Debugging را در VS انتخاب نمایید یا روی دکمه Debugging در نوار ابزار کلیک کنید یا دکمه F5 را فشار دهید. VS، مرورگر پیش فرض شما را با درخواست صفحه انتخابی توسط شما باز می‌کند. اولین باری که برنامه را اجرا می‌کنید VS از شما می‌پرسد که آیا می‌خواهید برنامه وب خود را با تنظیم فایل پیکربندی آن اجرا کنید.



## IIS Express Web Server

زمانی که یک برنامه تحت وب را اجرا می‌کنید، یک وب سرور تستی که IIS Express نامیده می‌شود را راه اندازی می‌کند که هاست موقعت سایت شما می‌باشد. این وب سرور طوری طراحی شده است که شبیه وب سرور IIS کامل با اندکی محدودیت کار کند. برای نمونه این وب سرور نمی‌تواند درخواست‌های فرستاده شده از کامپیوترهای دیگر را هندل کند. این وب سرور روی کامپیوتری که کدهای برنامه در آن نوشته می‌شوند به عنوان یک ابزار تست اجرا می‌شود.

پروژه‌های قدیمی مانند پروژه‌های ایجاد شده در ASP.NET Development Server VS 2010 از IIS Express استفاده می‌کنند. IIS Express اندکی به وب سرور واقعی نزدیک راست. برای تغییر آن به IIS Express، گزینه Website > Use IIS Express را انتخاب نمایید.

اگر برنامه را با VS اجرا کنید می‌بینید که آدرس بالای صفحه دارای یک پورت می‌باشد. این پورت برای این است که نشان دهد این برنامه روی کامپیوتر شما اجرا می‌شود و درخواست‌های آن نباید از طریق اینترنت ارسال شوند. همچنین اگر برنامه دیگری روی کامپیوتر شما در حال اجرا باشد و منتظر درخواستی (Listening) باشد، آن درخواست با درخواست برنامه شما تداخل پیدا نکند.

<http://localhost:4235/accounts.aspx>

## عیب یابی تک مرحله ای

محلى از کد را که می خواهید اجرای برنامه در آن نقطه متوقف شود، انتخاب و روی نوار کنار آن کلیک کنید (یا دکمه F9 را فشار دهید) تا یک breakpoint با نقطه قرمز رنگ نمایان شود.

```

TestPage.aspx.cs X
TestPage Button1_Click(object sender, EventArgs e)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        double val;

        // Attempt to convert the text to a number, and only
        // continue if it is a valid number.
        if (Double.TryParse(textBox1.Text, out val))
        {
            val *= 2;
            Label1.Text = "The doubled number is: " + val.ToString();
        }
    }
}

```

اکنون برنامه را اجرا کنید. زمانی که برنامه به خط مورد نظر شما برسد، اجرای آن متوقف می شود و شما به VS منتقل می شوید. خطی که دارای breakpoint می باشد هنوز اجرا نشده است.

شما چندین گزینه پیش رو دارید. می توانید با فشردن F11 خط کنونی را اجرا کنید. خط بعدی با رنگ زرد مشخص می شود که نشان می دهد این خطی است که اجرا خواهد شد.

می توانید با ماوس روی متغیرها رفته تا محتوای کنونی آنها را مشاهده کنید.

```

TestPage.aspx.cs X
TestPage Button1_Click(object sender, EventArgs e)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        double val;
        // Attempt to convert the text to a number, and only
        // continue if it is a valid number.
        if (Double.TryParse(TextBox1.Text, out val))
        {
            val *= 2;
            Label1.Text = "The doubled number is: " + val.ToString();
        }
    }
}

```

تا هنگامی که در حالت break می‌باشد، می‌توانید با کلیک راست روی صفحه یا استفاده از کلیدهای میانبر از دستورات موجود در جدول زیر استفاده کنید.

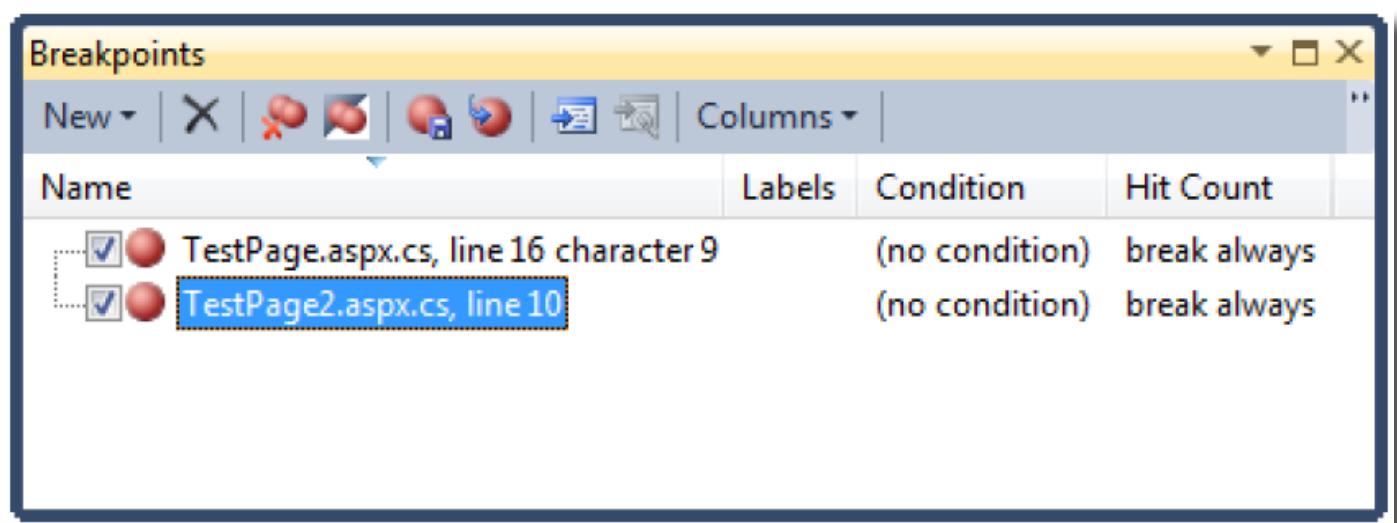
## دستورات قابل دسترس در حالت break

توضیحات	دستور (کلید میانبر)
خط کنونی highlight شده را اجرا و سپس متوقف می‌شود. اگر این خط یک متدا را فراخوانی کرده باشد، اجرای برنامه در اولین خط موجود در داخل متدا متوقف می‌شود (step into)	Step Into (F11)
مانند روش بالا می‌باشد با این تفاوت که با متدا مانند یک خط کد ساده عمل می‌کند و کل متدا را اجرا و به خط بعد می‌رود و در خط بعد از آن متوقف می‌شود.	Step Over (F10)
اجرای همه کدهای موجود در متدا یا رویه کنونی و سپس توقف در خطی بعد از خطی که متدا را فراخوانی کرده است. در حقیقت این کار به شما اجازه می‌دهد به خارج از رویه کنونی منتقل شوید.	Step Out(Shift + F11)
ادامه برنامه تا زمانی که به breakpoint می‌رسد.	Continue(F5)
اجرای همه کدها تا یک خط مشخص ( محلی ) که مکان نما قرار گرفته است	Run to Cursor

<p>به شما اجازه تغییر مسیر اجرای برنامه را در حالت debugging می‌دهد. این دستور برنامه را به خط کنونی که مکان نما در آن قرار دارد هدایت می‌کند. این تکنیک برای از روی حلقه های بزرگ مناسب است.</p>	<p><b>Set Next Statement</b></p>
<p>شما را از خط کد بعدی که VS اجرا می‌کند با خبر خواهد کرد. این دستور در هنگامی که خطی که در آن بودید را فراموش کرده‌اید مناسب می‌باشد.</p>	<p><b>Show Next Statement</b></p>

می‌توانید در هر نقطه‌ای که خواستید، با کلیک روی نقطه مورد نظرتان و کلیک روی دکمه pause در نوار ابزار یا انتخاب برنامه را به حالت Debug > Break All ببرید.

با کلیک روی Breakpoint هایی که در کل برنامه استفاده کرده‌اید را مشاهده کنید.



با دوبار کلیک روی هر آیتم می‌توانید به آن نقطه در کد منتقل شوید.

## های BreakPoint پیشرفته

VS، به شم اجازه می‌دهد تا breakpoint ها سفارشی سازی کنید، بنابراین تنها زمانی فعال می‌شوند که شرط مورد نظر برقرار باشد. برای این کار روی breakpoint کلیک راست نمایید تا گزینه های زیر در منو ظاهر شوند:

- برای مشاهده فایل و خط کد در جایی که این breakpoint گرفته است روی Location کلیک کنید.
- برای تعیین یک عبارت شرطی روی Condition کلیک کنید. هر زمان که آن شرط برقرار باشد، breakpoint در اصطلاح hit می‌شود.
- برای آنکه破點ها پس از اینکه به تعداد دفعات مشخصی hit شد متوقف شود، Hit Count را انتخاب کنید.
- برای آنکه破點 را به thread یا process خاصی محدود کنید روی Filter کلیک کنید.
- برای آنکه در هنگام hit شدن VS کار دیگری مانند اجرای یک ماکرو یا چاپ یک پیام Debug انجام دهد روی When Hit کلیک کنید.

Breakpoint ها بصورت خودکار در فایل‌های solution مربوط به VS ذخیره می‌شوند. البته اگر برنامه را با حالت Release کامپایل کنید، از آنها استفاده‌ای نمی‌شود. این حالت برای کامپایل نهایی برنامه مناسب است زیرا سرعت اجرای برنامه و حجم آن را کم می‌کند.

## Variable Watches

گاهی اوقات ممکن است بخواهید حالت متغیرها را بدون تغییر وضعیت به حالت break مشاهده کنید. در این موارد بهتر است از پنجره‌های Watch و Locals استفاده کنید.



پنجره	توضیحات
Autos	بصورت خودکار متغیرهایی که VS آنها را مهم تشخیص می‌دهد را برای کدهای دستورات کنونی نمایش می‌دهد. برای نمونه، ممکن است شامل متغیرهایی که در خط قبلی استفاده یا تغییر داده شده‌اند، باشد.
Locals	بصورت خودکار تمامی متغیرهایی که در ناحیه متدهای کنونی می‌باشند را نمایش می‌دهد.
Watch	متغیرهایی که شما تعیین می‌کنید را نمایش می‌دهد. برای اضافه کردن یک Watch، هنگامی که در حالت break هستید، روی متغیر موجود در کد خود کلیک راست نمایید و Add Watch را انتخاب کنید. همچنین می‌توانید روی آخرین خط درون این پنجره دوبار کلیک کنید و نام متغیر خود را بنویسید.

Name	Type	Value
this	TestPage {ASP.testpage_aspx}	{ASP.testpage_aspx}
[ASP.testpage_aspx]	ASP.testpage_aspx	{ASP.testpage_aspx}
base	System.Web.UI.Page	{ASP.testpage_aspx}
ApplicationInstance	System.Web.HttpApplication	{System.Web.HttpApplication}
Button1	System.Web.UI.WebControls.Button	{Text = "Button"}
form1	System.Web.UI.HtmlControls.HtmlForm	{System.Web.UI.HtmlControls.HtmlForm}
Head1	System.Web.UI.HtmlControls.HtmlHead	{InnerText = '((System.Web.UI.HtmlControls.HtmlHead)}
Label1	System.Web.UI.WebControls.Label	{Text = "Type something here:"}
Profile	System.Web.Profile.DefaultProfile	{System.Web.Profile.DefaultProfile}
TextBox1	System.Web.UI.WebControls.TextBox	{System.Web.UI.WebControls.TextBox}
base	System.Web.UI.WebControls.TextBox	{System.Web.UI.WebControls.TextBox}
AutoComplete	System.Web.UI.WebControls.AutoComplete	None
AutoPostBack	System.Web.UI.WebControls.AutoPostBack	false
CausesValidation	System.Web.UI.WebControls.CausesValidation	false
Columns	System.Web.UI.WebControls.Columns	0
MaxLength	System.Web.UI.WebControls.MaxLength	0
ReadOnly	System.Web.UI.WebControls.ReadOnly	false
Rows	System.Web.UI.WebControls.Rows	0
Text	System.Web.UI.WebControls.Text	"84"
TextMode	System.Web.UI.WebControls.TextMode	SingleLine
ValidationGroup	System.Web.UI.WebControls.ValidationGroup	""

# ASP.NET : توسعه برنامه های بخش اول

فصل اول - آشنایی با **Visual Studio**

فصل دوم: اصول فرم های وب

فصل سوم: کنترل های وب - **web control**

فصل چهارم: **Error Handling, Logging, Tracing**

فصل پنجم: **State Management**



## اصول و فرم

در این بخش با مواردی آشنا می‌شویم که هر برنامه نویس ASP.NET باید آنها را بداند. فایل و فولدرها، کنترل‌های سروری، ساختار وب فرم‌ها و ... از جمله مواردی می‌باشد که با آنها آشنا می‌شویم.

### آشنایی با آناتومی برنامه ASP.NET

بعضی از وقت‌ها بیان اینکه برنامه تحت وب چه چیزی است مشکل است. برخلاف برنامه دسکتاپ یا smartphone، برنامه‌های تحت وب تقريباً همیشه شامل چند صفحه وب می‌باشند. این یعنی که یک کاربر ASP.NET، از نقاط مختلفی می‌تواند به آن وارد شود و توسط لینک‌هایی از یک صفحه به صفحه دیگری برود.

هر برنامه تحت وب مجموعه‌ای از منابع و تنظیمات پیکربندی را به اشتراک می‌گذارد. صفحات وب سایر برنامه‌های تحت وب این منابع را به اشتراک نمی‌گذارند مگر اینکه روی یک وب سرور باشند.

تعريف استاندارد از برنامه تحت وب ترکیبی از فایل‌ها، صفحات، هندرلرهای ماژول‌ها و کدهای قابل اجرا بی می‌باشد که می‌توانند از درون یک virtual directory درون یک وب سرور فراخوانی شوند. شکل زیر یک وب سرور که چهار برنامه تحت وب مجزا می‌باشد را نشان می‌دهد.

 نکته: یک virtual directory، یک پوشه می‌باشد که در اختیار سایر کامپیوترها روی یک وب سرور قرار می‌گیرد.

### انواع فایل‌های ASP.NET

نام فایل	توضیحات
با پسوند .aspx	صفحات وب ASP.NET می‌باشند که شامل واسطه‌های کاربری و بصورت اختیاری کدهای برنامه می‌باشد.
با پسوند .ascx	این فایل‌ها asp.net user control می‌باشند که شبیه صفحات وب هستند با این تفاوت که کاربر نمی‌تواند مستقیماً به این فایل‌ها دسترسی داشته باشد. در عوض، آنها باید درون صفحات وب قرار بگیرند. user control‌ها به شما اجازه می‌دهند بخش کوچکی از واسط کاربری را ایجاد کنید و آن را در چند صفحه وب متفاوت بکار ببرید (بدون نوشتن مجدد کدها).
Global.asax	این یک فایل عمومی می‌باشد. از این فایل برای تعریف متغیرهای عمومی (که می‌تواند از هر صفحه وب موجود در برنامه فراخوانی شود) و پاسخ به رویدادهای عمومی (مانند اولین باری که برنامه آغاز می‌شود) استفاده می‌شود.
Web.config	این یک فایل پیکربندی برای برنامه شما می‌باشد. این فایل شامل تنظیماتی برای state, security, memory management, management و ... می‌باشد.
با پسوند cs	این فایل code-behind است که شامل کد C# می‌باشد و به شما اجازه جداسازی منطق برنامه از واسط کاربری یک صفحه وب را می‌دهد.

فایل‌های دیگری نظیر HTML, CSS و ... نیز وجود دارند که در ادامه با آنها آشنا خواهیم شد.

## فولدرهای وب ASP.NET

هر برنامه تحت وب از یک محل به نام root folder آغاز می‌شود که درون آن زیر پوشه‌های مختلفی وجود دارد.

نکته: برای ایجاد یک زیر پوشه در VS، روی وب سایت کیک راست کنید و add a new folder را انتخاب کنید.



### پوشه‌های ASP.NET

شامل فایل‌های asp.net که برای تشخیص مرورگرهایی که از برنامه شما استفاده می‌کنند آنها را بکار می‌برد. معمولاً اطلاعات مرورگر در کل وب سرور استاندارد شده است.	App_Browser
شامل فایل‌های که بصورت پویا برای استفاده در برنامه شما کامپایل می‌شوند.	App_Code
منابع عمومی که در دسترس صفحات برنامه شما وجود دارند را ذخیره می‌کند. این پوشه برای localize کردن استفاده می‌شود. زمانی که یک وب سایت با بیش از یک زبان ارائه می‌شود. البته این امر در این کتاب توضیح داده نشده است.	App_GlobalResources
مانند مورد قبلی می‌باشد با این تفاوت که این منابع تنها می‌توانند در دسترس یک صفحه خاص قرار بگیرند.	App_LocalResources
Reference ها به یک وب سرویس را در خود نگهداری می‌کند که می‌توانند از طریق شبکه یا اینترنت فراخوانی شوند.	App_WebResources
داده‌هایی شامل فایل‌های SQL Server Express را در خود نگهداری می‌کند.	App_Data
Theme های استفاده شده برای استاندارد سازی و استفاده مجدد فرمت دهی در صفحات وب در این فولدر ذخیره می‌شوند.	App_Themes
شامل کلیه .NET component (Compiled DLL) هایی که برنامه تحت وب ASP.NET از آنها استفاده می‌کند می‌باشد.	Bin

### معرفی کنترل‌های سروی

کنترل‌های سروری یا سمت سرور به عنوان یک شی ایجاد و پیکربندی می‌شوند. آنها روی سرور وب اجرا و بصورت خودکار HTML خود را تهیه می‌کنند. بنابراین شما خیلی در گیر جزئیات مربوط به کدهای HTML کنترل‌ها نخواهید شد.

ASP.NET، دو مجموعه از کنترل‌های سمت سرور (server-side) را در اختیار قرار می‌دهد:

#### : HTML server control

المان‌هایی مشابه المان‌های HTML استاندارد می‌باشند. زمانی که صفحات قدیمی HTML یا ASP classic را بخواهیم به ASP.NET تبدیل کنیم این کنترل‌ها مفید هستند.

## Web Control

شبیه HTML server control هستند اما اشیای قوی تری با property های مختلفی برای استایل ها و فرمت بندی در اختیار قرار می دهند. همچنین رویدادهای بیشتری نیز ارائه می دهند. این کنترل ها همچنین المان هایی ارائه می دهند که مشابه مستقیمی در HTML برای آنها وجود ندارد. مانند GridView و Calender و ...

## HTML Server Controls

این کنترل ها سه ویژگی کلیدی زیر را ارائه می دهند :

- آنها واسط کاربری خود را ایجاد می کنند :

شما property هایی در کد برای آنها تعیین می کنید و زمانی که صفحه رender شد و به کلاینت فرستاده شد، کدهای HTML آنها بصورت خودکار تولید می شود

- آنها وضعیت یا state خود را حفظ می کنند:

به دلیل آنکه وب stateless می باشد، صفحات قدیمی وب می باشند که اطلاعات در بین درخواست ها می کرددند. این کنترل ها این کارها را بصورت خودکار انجام می دهند. برای نمونه اگر کاربر یک آیتم در listbox را انتخاب کند، آن آیتم در دفعه بعدی که این صفحه به هر دلیل refresh می شود انتخاب شده باقی خواند ماند.

- آنها رویدادهای سمت سرور را fire می کنند:

برای نمونه دکمه ها یک رویداد را در هنگامی که کلیک می شوند fire می کنند. textbox ها زمانی که متن موجود در آنها تغییر کند و ...

## View State

زمانی که برنامه اجرا می کنید اولین چیزی که متوجه آن می شوید، تفاوت اندکی بین HTML موجود در فایل .aspx و HTML ای می باشد که به مرورگر فرستاده شده است. اولین مورد "runat=server" می باشد زیرا چنین attribute یا خصیصه ای برای مرورگر معنی دار نیست و فقط به برای کدهای سمت سرور مناسب است. مورد دوم و مهم تر آنکه، یک فیلد hidden به فرم اضافه شده است.

کد HTML موجود در فایل : .aspx

```
<!DOCTYPE html>
<html>
<head>
<title>Currency Converter</title>
</head>
<body>
```

```

<form id="Form1" runat="server">
  <div>
    Convert: &nbsp;
    <input type="text" id="US" runat="server" />
    &nbsp; U.S. dollars to Euros.
    <br />
    <br />
    <input type="submit" value="OK" id="Convert" runat="server" />
  </div>
</form>
</body>
</html>

```

کد فرستاده شده به مرورگر :

```

<p>
Put content here.

<!DOCTYPE html>
<html>
<head>
  <title>Currency Converter</title>
</head>
<body>
  <form id="form1" method="post" action="CurrencyConverter.aspx">
    <div class="aspNetHidden">
      <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="dDw3NDg2NTI5MDg70z4..." />
    </div>
    <div class="aspNetHidden">
      <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="/WEWAwLr3rrOBgLr797" />
    </div>
    <div>
      Convert: &nbsp;
      <input type="text" id="US" name="US" />
      &nbsp; U.S. dollars to Euros.
    </div>
  </form>
</body>
</html>

```

```

<br />
<br />

<input type=""submit"" value=""OK"" id=""Convert"" name=""Convert"" />
</div>
</form>
</body>
</html>
</p>

```

می‌بینید که ASP.NET در این مثال دو المان `<input hidden>` به فرم اضافه نموده است.

## کلاس‌های HTML Control

تمامی HTML server control ها در فضای نام System.Web.UI.HtmlControls تعریف می‌شوند.

### کلاس‌های کنترل‌های سروری HTML

نام کلاس	المان	توضیحات
HTMLForm	<form>	همه کنترل‌های روی صفحه را پوش می‌دهد. تمامی کنترل‌های ASP.NET باید درون یک کنترل <code>HtmlForm</code> قرار بگیرند بنابراین می‌توانند هنگامی که صفحه <code>submit</code> می‌شود داده خود را به سرور بفرستند. VS المان <code>&lt;form&gt;</code> را به همه صفحات جدید اضافه می‌کند. در هنگام طراحی صفحات باید مطمئن شوید همه کنترل‌ها درون <code>&lt;form&gt;</code> قرار می‌گیرند.
HtmlAnchor	<a>	یک <code>hyperlink</code> که کاربر برای پرس به صفحات دیگر روی آن کلیک می‌کند.
HtmlTable, HtmlTableRow, and HtmlTableCell	<table>, <tr>, <th>, <td>	جدولی که ردیف‌ها و ستون‌هایی از یک متن ثابت را نمایش می‌دهد.
HtmlInputButton, HtmlInputSubmit, and HtmlInputReset	<input type="button">, <input type="submit">, <input type="reset">	یک دکمه که کاربر برای انجام کاری روی آن کلیک می‌کند ( <code>HtmlInputButton</code> ), صفحه را <code>submit</code> می‌کند ( <code>HtmlInputSubmit</code> ), یا کلیه مقادیر موجود در کنترل‌ها را خالی می‌کند ( <code>HtmlInputReset</code> )

<p>دکمه ای که برای انجام کاری کاربر روی آن کلیک می‌کند. این کنترل توسط همه مرورگرها پشتیبانی نمی‌شود بنابراین معمولاً <code>HtmlInputButton</code> بجای آن استفاده می‌شود. تفاوت آن در این است که <code>HtmlButton</code> یک <code>container</code> است که می‌توان چیزهای دیگری مانند متن و تصویر درون آن قرار داد.</p>	<button>	HTMLButton
	<input type="checkbox">	HtmlInputCheckBox
	<input type="radio">	HtmlInputRadioButton
	<input type="text">	HtmlInputText
	<input type="password">	HtmlInputPassword
	<textarea>	HtmlTextArea
<p>مانند تگ <code>&lt;img&gt;</code> می‌باشد اما یک تصویر قابل کلیک می‌باشد تا صفحه <code>submit</code> شود. با استفاده از کدهای سمت سرور می‌توانید تعیین کنید کاربر روی چه ناحیه‌ای از تصویر کلیک کرده است.</p>	<input type="image">	HtmlInputImage
<p>یک <code>button</code> و <code>textbox</code> در کنار هم می‌باشد که برای آپلود فایل روی سرور وب استفاده می‌شود.</p>	<input type="file">	HtmlInputFile
<p>شامل اطلاعاتی می‌باشد که در زمان <code>postback</code> شدن صفحه به سرور فرستاده خواهد شد ولی در مرورگر قابل دسترسی نیستند و نمایش داده نمی‌شوند.</p>	<input type="hidden">	HtmlInputHidden
<p>یک <code>dropdown</code> یا <code>listbox</code> که کاربر می‌تواند یک آیتم از آن را انتخاب کند.</p>	<select>	HtmlSelect
<p>اطلاعات <code>header</code> صفحه را ارائه می‌دهد که در صفحه نمایش داده نمی‌شود. مانند کلمات کلیدی جستجو و عنوان صفحه وب. این‌ها تنها کنترل‌هایی هستند که درون بخش <code>&lt;form&gt;</code> قرار نمی‌گیرند.</p>	<head> <title>	HtmlHead HtmlTitle
<p>این کنترل می‌تواند انواع مختلفی از المان‌های HTML را ارائه دهد. برای نمونه اگر به المان <code>&lt;div&gt;</code> یک خصیصه <code>runat="server"</code> اضافه کنید در سمت کد می‌توانید با آن به عنوان یک المان <code>HtmlGenericControl</code> کار کنید.</p>	دیگر HTML هر المان	HtmlGenericControl

# ویژگی‌های مهم کنترل‌های HTML



ویژگی‌های مهم	کنترل
HRef, Target	HtmlAnchor
Src, Alt, Width, Height	HtmlImage
Checked	HtmlInputCheckBox , HtmlInputRadioButton
Value	HtmlInputText
Value	HtmlTextArea
Src, Alt	HtmlInputImage
Items (collection)	HtmlSelect
InnerText and InnerHtml	HtmlGenericControl

## یک مثال کاربردی

فرض کنید یک فرم داریم که با توجه به عددی که ما وارد می‌کنیم آن را به واحد دیگری تبدیل می‌کند.

```

<html>
  <head>
    <title>Currency Converter</title>
  </head>
  <body>
    <form id="Form1" runat="server">
      <div>
        Convert:
        <input type="text" id="US" runat="server" />
        U.S. dollars to Euros.
        <br />
        <br />
        <input type="submit" value="OK" id="Convert" runat="server" onserverclick="Convert_ServerClick" />
        <br />
        <br />
        <p style="font-weight: bold" id="Result" runat="server">
        </p>
      </div>
    </form>
  </body>
</html>
```

مثال ما اکنون دارای چهار کنترل سروری می‌باشد :

- یک فرم که با شی HtmlForm ارائه می‌شود. این تنها کنترلی می‌باشد که نیازی به دسترسی در code-behind ندارید.
- یک input textbox با نام US ( HtmlInputText)
- یک submit button با نام Convert (HtmlInputButton)
- یک p با نام Result (HtmlGenericControl)

### : code-behind کدهای

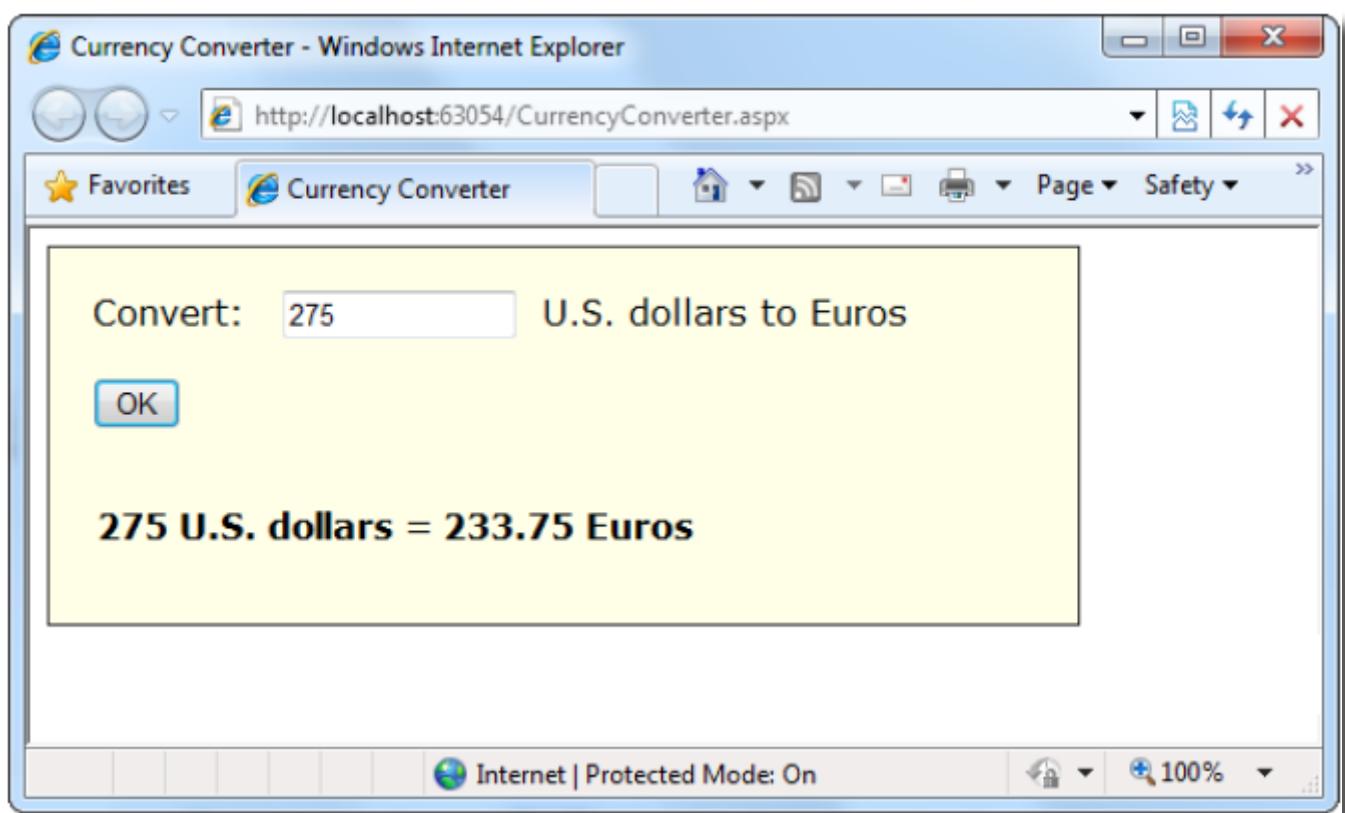
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class CurrencyConverter : System.Web.UI.Page
{
    protected void Convert_ServerClick(object sender, EventArgs e)
    {
        decimal USAmount = Decimal.Parse(US.Value);
        decimal euroAmount = USAmount * 0.85M;
        Result.InnerText = USAmount.ToString() + " U.S. dollars = ";
        Result.InnerText += euroAmount.ToString() + " Euros.";
    }
}
```

 نکته: برخلاف web control ها، نمی‌توانید برای کنترل‌های سروری HTML، با استفاده از پنجره Properties اقدام به ایجاد event handler ها کنید. بجای آن باید متدها را بصورت دستی بنویسید و سپس تگ کنترل را به منظور اتصال به (متدهای مربوطه) تغییر دهید.

OnServerClick = "Convert\_ServerClick"

عملگر + = برای اضافه کردن فوری اطلاعات به انتهای label بدون جایگزین کردن متن موجود در آن استفاده می‌شود.



## Event Handling

زمانی که کاربر روی دکمه Convert کلید می‌کند و صفحه به سرور وب ارسال می‌شود، ASP.NET نیاز دارد تا بداند دقیقاً چه کدی را می‌خواهد اجرا کنید. برای ایجاد این ارتباط و اتصال یک رویداد به event handling method، باید یک attribute به تگ کنترل اضافه کنید.

برای نمونه، اگر بخواهید متده کلیک سرور از دکمه Convert را هندل کنید، باید خصیصه OnServerClick را در تگ کنترل با نام متده event handler set نظر خود کنید.

```
<input type = "submit" value = "OK" ID = "Submit1"
OnServerClick = onvert_ServerClick" runat = "server">
```

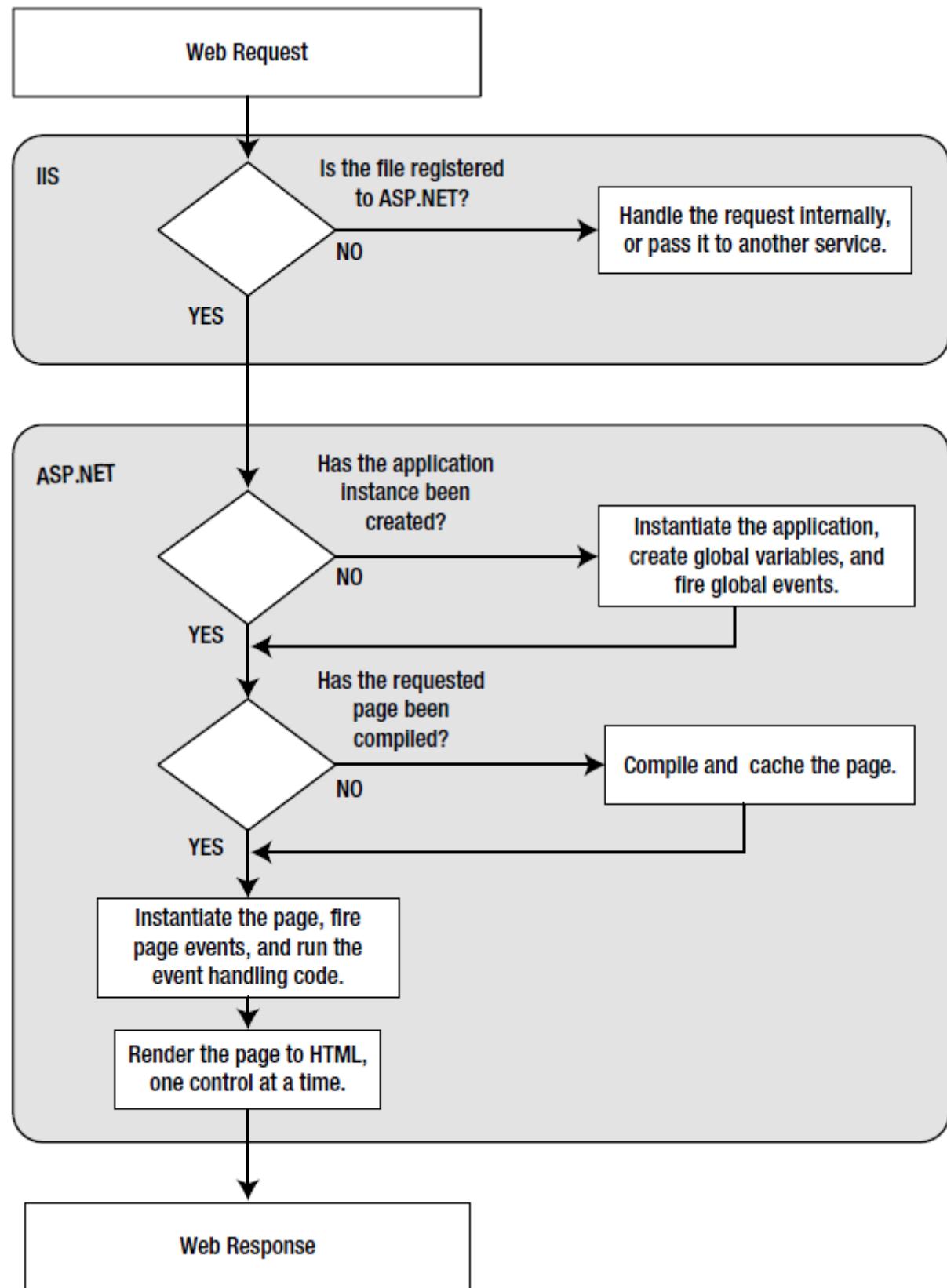
همچنین اجزاه اتصال رویداد به event handler را به روش دیگری به شما می‌دهد. می‌توانید در سمت ASP.NET از دستور زیر استفاده کنید :

```
Convert.ServerClick += this.Convert_ServerClick;
```

## بررسی بیشتر

واقعاً در هنگامی که ASP.NET یک درخواست از صفحه (CurrencyConvert.aspx) دارد، چه اتفاقی می‌افتد؟

- درخواست فرستاده شده از صفحه به وب سرور ارسال می‌شود. اگر یک سایت Live داشته باشد، وب سرور شما قطعاً IIS خواهد بود و اگر برنامه را در VS اجرا کرده باشد، درخواست شما به test server موجود در VS فرستاده می‌شود.
- وب سرور تعیین می‌کند که فایل با پسوند .aspx با ASP.NET ثبت شده است. اگر فایل با پسوند دیگری مانند .html باشد ASP.NET با آن کاری ندارد.
- اگر دفعه اولی است که این صفحه مورد درخواست قرار می‌گیرد، بصورت خودکار یک Application domain ASP.NET ایجاد می‌کند. همچنین همه صفحات وب را برای کارایی بیشتر کامپایل و فایل‌های کامپایل شده را Cache می‌کند. اگر این روال قبل انجام شده باشد، از نسخه کامپایل شده صفحه استفاده می‌کند.
- زمانی که کد صفحه به پایان برسد، ASP.NET از هر کنترل موجود در صفحه خواهد خواست که، خود را درون HTML مربوطه رندر کنند.
- صفحه نهایی به کاربر ارسال شده و برنامه به پایان می‌رسد.



## تکمیل کردن مثال :

می توانیم در مثال بالا، واحدهای بیشتری برای تبدیل اضافه کنیم. بنابراین یک dropdownlist به فرم اضافه می کنیم. در HTML این کنترل با `<select>` نشان داده می شود که برای هر آیتم از المان `<option>` درون آن استفاده می شود. برای کاهش حجم HTML می توان یک dropdownlist خالی در HTML نوشته و آیتمها را در سمت سرور به آن اضافه کرد. توجه کنید که `runat="server"` را برای آن `set` کنید تا از کد بتوان به این کنترل HTML دسترسی داشت.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Currency Converter</title>
</head>
<body>
    <form id="Form1" runat="server">
        <div>
            Convert: &nbsp;
            <input type="text" id="US" runat="server" />
            &nbsp; U.S. dollars to &nbsp;
            <select id="Currency" runat="server" />
            <br />
            <br />
            <input type="submit" value="OK" id="Convert" onclick="Convert_ServerClick"
                   runat="server" />
            <br />
            <br />
            <p style="font-weight: bold" id="Result" runat="server">
            </p>
        </div>
    </form>
</body>
</html>
```

بهترین زمان برای پر کردن لیست هنگام لود شدن صفحه می باشد. برای اینکه هر دفعه این کار تکرار نشود باید تعیین کنیم فقط در بار اولی که صفحه لود می شود این کار انجام شود:

```

protected void Page_Load(Object sender, EventArgs e)
{
    if (this.IsPostBack == false)
    {
        Currency.Items.Add("Euro");
        Currency.Items.Add("Japanese Yen");
        Currency.Items.Add("Canadian Dollar");
    }
}

```

### ذخیره کردن اطلاعات در لیست

هر آیتم دارای یک ویژگی value می‌باشد که می‌توان برای ذخیره یک مقدار برای هر آیتم از آن استفاده کرد.

```

protected void Page_Load(Object sender, EventArgs e)
{
    if (this.IsPostBack == false)
    {
        // The HtmlSelect control accepts text or ListItem objects.

        Currency.Items.Add(new ListItem("Euros", "0.85"));
        Currency.Items.Add(new ListItem("Japanese Yen", "110.33"));
        Currency.Items.Add(new ListItem("Canadian Dollars", "1.2"));
    }
}

```

```

protected void Convert_ServerClick(object sender, EventArgs e)
{
    decimal oldAmount;
    bool success = Decimal.TryParse(US.Value, out oldAmount);

    if (success)
    {
        // Retrieve the selected ListItem object by its index number.

```

```

ListItem item = Currency.Items[Currency.SelectedIndex];

decimal newAmount = oldAmount * Decimal.Parse(item.Value);

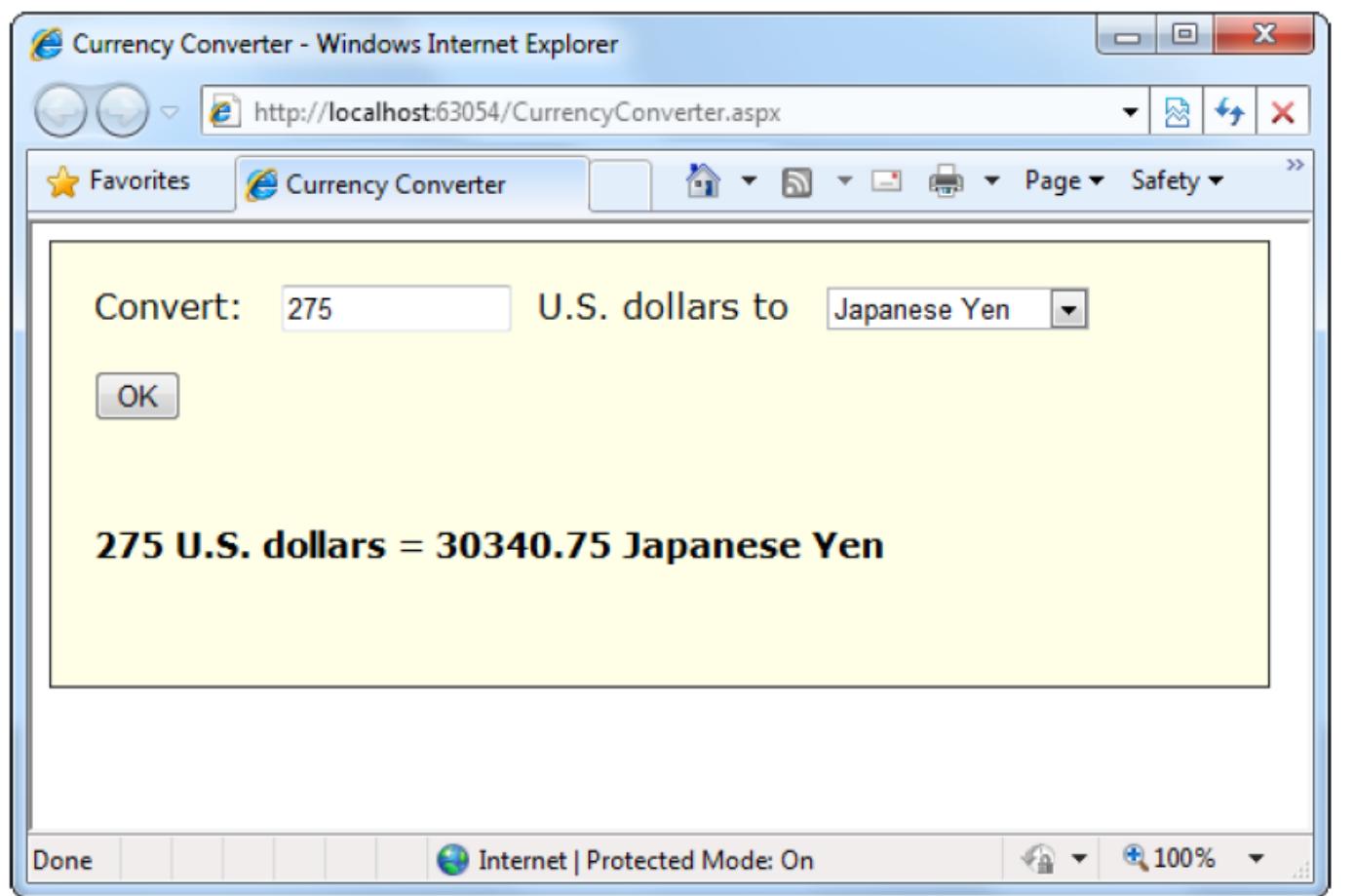
Result.InnerText = oldAmount.ToString() + " U.S. dollars = ";

Result.InnerText += newAmount.ToString() + " " + item.Text;

}

}

```



## اضافه کردن تصاویر لینک دار

فرض کنید می خواهیم یک دکمه برای نمایش گراف تبدیلات به فرم اضافه کنیم. برای این کار باید به برنامه یک دکمه و کنترل image اضافی به فرم اضافه کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>Currency Converter</title>
</head>
<body>
<form id="Form1" runat="server">
<div>
Convert: &nbsp;
<input type="text" id="US" runat="server" />
&nbsp; U.S. dollars to &nbsp;
<select id="Currency" runat="server" />
<br />
<br />
<input type="submit" value="OK" id="Convert" onclick="Convert_ServerClick"
runat="server" />
<input type="submit" value="Show Graph" id="ShowGraph" runat="server" />
<br />
<br />
<img id="Graph" src="" alt="Currency Graph" runat="server" />
<br />
<br />
<p style="font-weight: bold" id="Result" runat="server">
</p>
</div>
</form>
</body>
</html>

protected void Page_Load(Object sender, EventArgs e)
{

```

```

if (this.IsPostBack == false)
{
    Currency.Items.Add(new ListItem("Euros", "0.85"));
    Currency.Items.Add(new ListItem("Japanese Yen", "110.33"));
    Currency.Items.Add(new ListItem("Canadian Dollars", "1.2"));
    Graph.Visible = false;
}
}

```

برای تغییر تصویر با انتخاب هر مورد از لیست کد زیر را می‌نویسیم :

```

protected void ShowGraph_ServerClick(Object sender, EventArgs e)
{
    Graph.Src = "Pic" + Currency.SelectedIndex.ToString() + ".png";
    Graph.Visible = true;
}

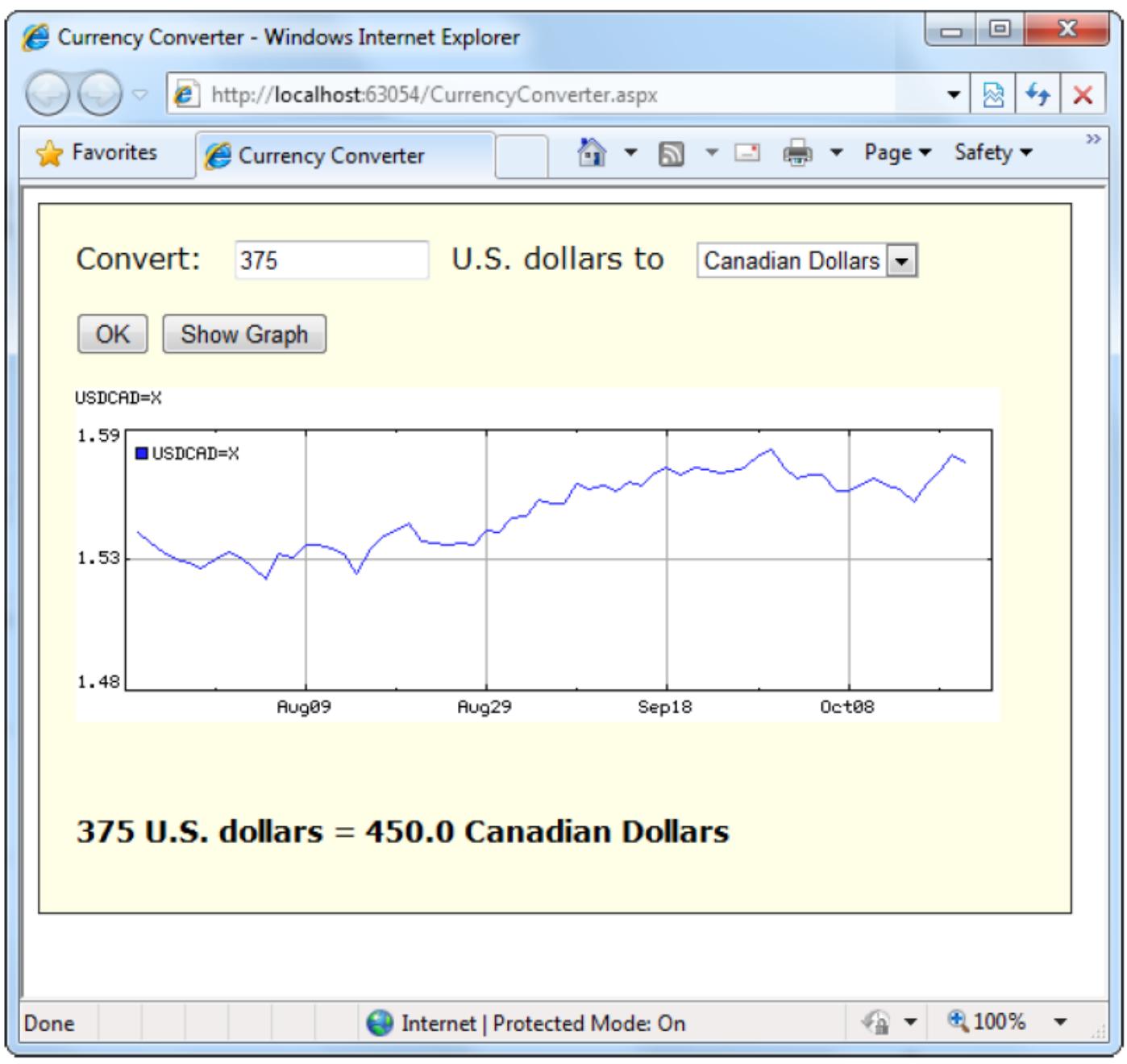
```

برای نمایش تصویر باید رویداد مورد نظر را به متده آن متصل کنیم :

```

<input type = "submit" value = "Show Graph" ID = "Submit1"
OnServerClick = "ShowGraph_ServerClick" runat = "server" />

```



## Setting Styles

علاوه بر ویژگی‌های محدود، هر کنترل HTML امکان دسترسی به خصیصه‌های CSS را از طریق مجموعه‌ای از استایل‌های خود (style collection)، فراهم می‌کند. برای استفاده از آن مجموعه باید نام `css style attribute` و مقداری که می‌خواهید به آن بدهید را مشخص کنید :

```
ControlName.Style["AttributeName"] = "AttributeValue";
```

به عنوان مثال می‌توان تعیین کرد اگر مقدار نا معتبر در فرم وارد شد، رنگ کنترل قرمز شود.

```

protected void Convert_ServerClick(object sender, EventArgs e)
{
    decimal oldAmount;

    bool success = Decimal.TryParse(US.Value, out oldAmount);

    if ((oldAmount <= 0) || (success == false))
    {
        Result.Style["color"] = "Red";
        Result.InnerText = "Specify a positive number";
    }
    else
    {
        Result.Style["color"] = "Black";

        // Retrieve the selected ListItem object by its index number.

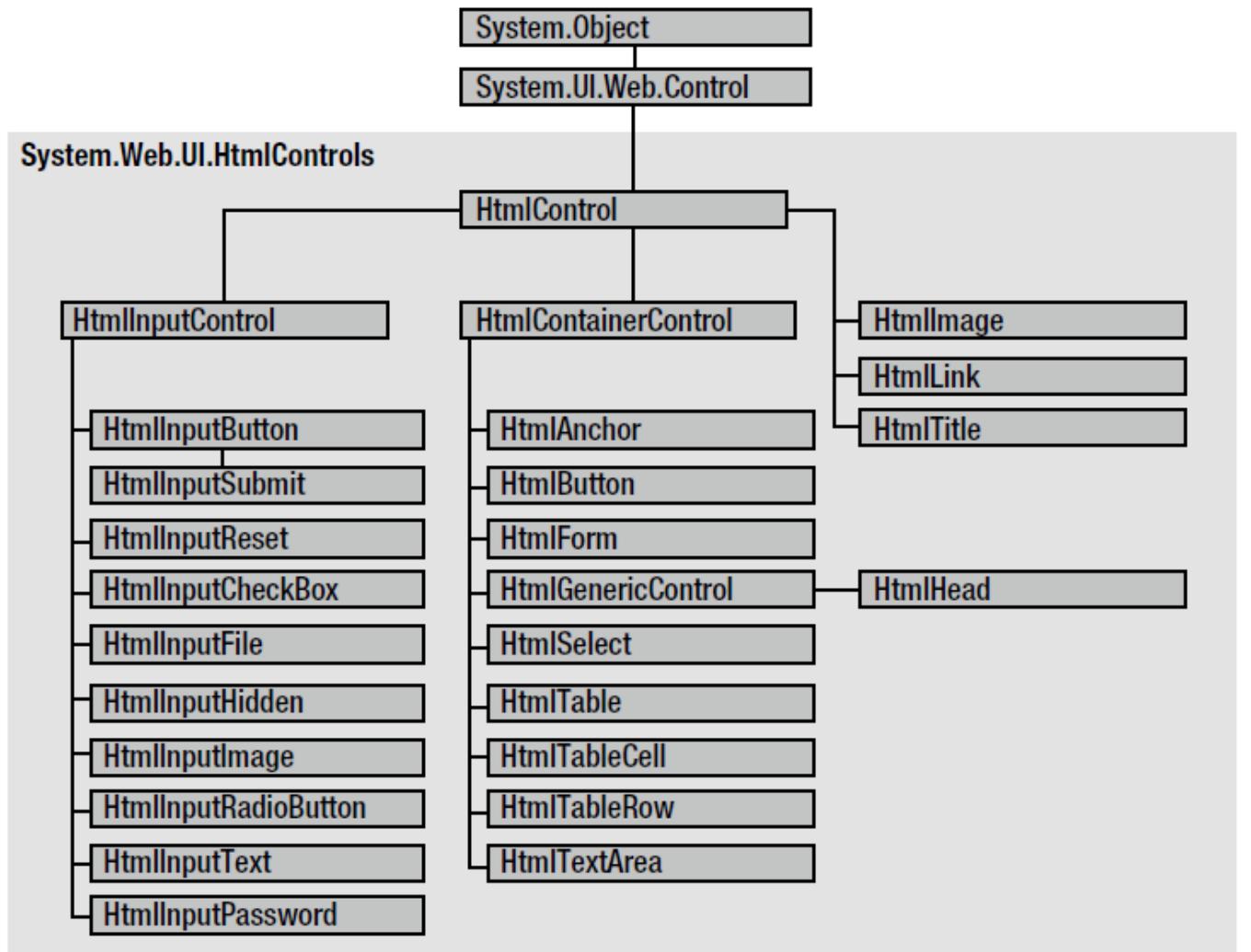
        ListItem item = Currency.Items[Currency.SelectedIndex];

        decimal newAmount = oldAmount * Decimal.Parse(item.Value);

        Result.InnerText = oldAmount.ToString() + " U.S. dollars = ";
        Result.InnerText += newAmount.ToString() + " " + item.Text;
    }
}

```

کلیه کنترل‌های HTML از کلاس HtmlControl ارث بری می‌کنند.



## رویدادهای HTML Control

HTML server control همچنین یکی از دو رویداد ServerClick یا ServerChange را ارائه می‌دهد. رویداد کلیکی می‌باشد که در سمت سرور پردازش می‌شود. این رویداد توسط بیشتر کنترل‌های دکمه‌ای ارائه می‌شود و به کد شما اجازه می‌دهد تا یک عمل فوری انجام دهد. برای نمونه کنترل HtmlAnchor که یک کنترل سروری می‌باشد را در نظر بگیرید. دو راه برای استفاده از آن وجود دارد. روش اول تعیین مقدار URL برای ویژگی Href آن می‌باشد. در اینجا این کنترل شبیه المان <a> در HTML عمل می‌کند. روش دیگر استفاده از رویداد ServerClick می‌باشد.

رویداد ServerChange زمانی رخ می‌دهد که تغییری در یک کنترل متنی یا انتخابی ایجاد شود. این رویداد تا زمانی که صفحه postback نشود رخ نمی‌دهد.

## رویدادهای HTML Control



کنترل‌های فراهم کننده آنها	رویداد
HtmlAnchor, HtmlButton, HtmlInputButton, HtmlInputImage, HtmlInputReset	ServerClick
HtmlInputText, HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputHidden, HtmlSelect, HtmlTextArea	ServerChange

## HtmlContainerControl کلاس

هر کنترلی که نیاز به یک تگ پایانی می‌باشد که آن را می‌بندد مانند `<div></div>`, `<form></form>`, `<a></a>` و ... از کلاس `HtmlContainer` ارث بری می‌کند. بنابراین المان‌هایی مانند `HtmlGenericControl`, `HtmlAnchor`, `HtmlForm` و `stand-alone` از کلاس `HtmlContainerControl` ارث برای می‌کنند ولی المان‌هایی مانند `<input>`, `<img>` و `<img>` از تگ‌هایی تنها یا استفاده می‌کنند.

## ویژگی‌های HtmlContainerControl



توضیحات	Property
محتوای HTML بین تگ باز و بسته هر کنترل را شامل می‌شود. کاراکترهای خاص که از طریق این ویژگی می‌شوند به موجودیت‌های HTML معادل آن تبدیل نخواهد شد. این یعنی می‌توانید از این ویژگی برای بکارگیری فرمت دهی با تگ‌های تودرتو مانند <code>&lt;i&gt;</code> , <code>&lt;b&gt;</code> , <code>&lt;h1&gt;</code> و ... استفاده می‌شود.	InnerHtml
متن بین تگ باز و بسته هر کنترل را شامل می‌شود. کاراکترهای خاص بصورت خودکار به موجودیت‌های HTML تبدیل و به عنوان متن نمایش داده می‌شوند. (برای نمونه کاراکتر " <code>&lt;</code> " به " <code>&amp;lt;</code> "; " <code>&amp;</code> " تبدیل می‌شود و در صفحه وب بصورت " <code>&lt;</code> " نمایش داده می‌شود).	InnerText

## استفاده از کلاس Page

هر صفحه وب یک کلاس سفارشی می‌باشد که از `System.Web.UI.Page` ارث بری کرده است. با ارث برای از این کلاس، کلاس صفحه وب شما دارای تعدادی از `Property`‌ها و متدها می‌شود که می‌توانید در کد از آن استفاده کنید.



توضیحات	Property
این ویژگی <code>Boolean</code> تعیین می‌کند که آیا اولین باری است که صفحه اجرا می‌شود ( <code>false</code> ) یا صفحه در پاسخ به رویداد یک کنترل مجدد <code>submit</code> شده است ( <code>true</code> ). این ویژگی را معمولاً در هندرل <code>Page</code> استفاده می‌کنیم تا مطمئن شویم که مقدار دهی اولیه صفحه وب تنها یک بار انجام شود.	IsPostBack
زمانی که با <code>false</code> شود، ویژگی <code>EnableViewState</code> کنترل‌های موجود در صفحه می‌شود تا مطمئن شود که هیچ کنترلی اطلاعات وضعیت خود را نگهداری نمی‌کند.	EnableViewState

این مجموعه، اطلاعاتی را نگهداری می‌کند که بین تمامی کاربران وب سایت به اشتراک گذاشته می‌شود برای نمونه می‌توانید از Application برای شمارش تعداد دفعاتی که ک صفحه مشاهده شده است استفاده کنید.	Application
این مجموعه، اطلاعاتی را نگهداری می‌کند که برای یک کاربر استفاده می‌شود، بنابراین می‌تواند برای صفحات مختلف استفاده شود. برای نمونه، می‌توانید از این آیتم برای ذخیره سبد خرید کاربر فعلی در یک وب سایت خرید آنلайн استفاده کنید.	Session
این مجموعه به شما اجازه می‌دهد تا اشیایی که ایجاد آنها زمان بر هستند را ذخیره کند، بنابراین آنها می‌توانند در سایر صفحات یا client های دیگر استفاده شوند.	Cache
به شی HttpRequest که اطلاعاتی در مورد درخواست وب فعلی را نگهداری می‌کند اشاره خواهد کرد. می‌توانید از شی HttpRequest برای دریافت اطلاعاتی در مورد مرورگر کاربر استفاده نمایید. همچنین می‌توانید از این شی برای انتقال اطلاعات از یک صفحه به صفحه دیگر با بکارگیری query string استفاده کنید.	Request
به شی HttpResponseMessage اشاره می‌کند، که پاسخ ASP.NET که به کاربر ارسال می‌شود را نگهداری می‌کند. از این شی برای ایجاد cookie نیز استفاده می‌شود. همچنین می‌توانید برای redirect کردن کاربر به یک صفحه وب دیگر از آن استفاده کنید.	Response
به شی HttpServerUtility اشاره می‌کند که به شما اجازه اجرای task های مختلف را می‌دهد. برای encode کردن متن استفاده می‌شود، بنابراین برای قرار دادن آن در URL یا HTML markup صفحه وب شما می‌توان از آن استفاده کرد.	Server
اگر کاربر authenticate شده باشد، این ویژگی با اطلاعات کاربر مقدار دهی می‌شود.	User

## فرستادن کاربر به یک صفحه جدید

در مثال Converter، کلیه چیزها در یک صفحه قرار دارد. در اکثر سایتها، کاربر نیاز دارد تا از یک صفحه به صفحات دیگر برود. راه های مختلفی برای انتقال یک کاربر از یک صفحه به صفحه دیگر وجود دارد. یکی از ساده‌ترین آنها استفاده از المان قدیمی anchor <a>، می‌باشد. در این مثال، کلمه "here" یک لینک به صفحات دیگر می‌باشد.

Click

راه دیگر، فرستادن کاربر به صفحه جدید با استفاده از کد می‌باشد. اگر می‌خواهید از کد خود برای اجرای کار دیگری قبل از فرستادن او به صفحه دیگر انجام دهید این روش مناسب است.

زمانی که از متده استفاده می‌کنید، ASP.NET فوراً پردازش صفحه را متوقف کرده و یک پیام redirect به مرورگر ارسال می‌کند. کدهای بعد از متده Redirect() اجرا نخواهند شد. زمانی که مرورگر پیام redirect را دریافت کرد، یک درخواست برای صفحه جدید به سرور می‌فرستد.

می‌توانید از این متده برای فرستادن کاربر به هر نوع از صفحه استفاده نمایید؛ و حتی او را به سایت دیگری نیز بفرستید.

```
Response.Redirect("http://www.prosetech.com");
```

```
Response.Redirect("newpage.aspx");
```

همچنین گزینه دیگری نیز برای فرستادن کاربر به صفحه دیگر در اختیار شما قرار می‌دهد. می‌توانید از متدهای `HttpServerUtility.Transfer()`

بجای استفاده کنید. شی `Page.Server` از طریق `HttpServerUtility` قابل دسترس می‌باشد.

```
Server.Transfer("newpage.aspx");
```

مزیت استفاده از متدهای `Transfer()` این است که شامل مرورگر نمی‌شود. بجای ارسال یک پیام `redirect` به مرورگر، به سادگی پردازش یک صفحه جدید را شروع می‌کند، در نتیجه آدرس بالای صفحه تغییر نمی‌کند که البته این کمی گیج کننده می‌باشد. این روش در زمان صرفه جویی می‌کند اما محدودیت‌هایی نیز بوجود می‌آورد. نمی‌توانید کاربر را به یک سایت یا صفحه ای به جزی (مانند ASP.NET) HTML ارسال کنید. متدهای `Transfer()` به شما تنها اجازه پرش از یک صفحه به صفحه دیگر در یک وب سایت را می‌دهد. در این روش شما کنترل‌های موجود در یک صفحه را به صفحه دیگر ارسال می‌کنید و در صفحه جدید می‌توان با استفاده از `Request.Form` می‌توان به مقادیر وارد شده در کنترل‌های صفحه قبل دسترسی داشت.

## کار با HTML Encoding

همانطور که از قبل می‌دانید، کarakترهای ویژه، معانی خاصی در HTML دارند. برای نمونه (`<>`) همیشه برای ایجاد تگ بکار می‌رود. اگر بخواهید از این کarakترها در بخشی از صفحه وب استفاده نمایید با مشکل روبرو می‌شوید.

فرض کنید می‌خواهید متن زیر را در صفحه وب نمایش دهید :

`<Enter a word < here`

مشکل اینجاست که مرورگر سعی دارد تا عبارت `<hear>` را به عنوان تگ HTML تفسیر کند.

در عبارت زیر نیز عبارت `<b>` به عنوان یک تگ HTML در نظر گرفته می‌شود و متن بعد از آن بصورت توپر نمایش داده خواهد شد. برای رفع این مشکل باید مقادیر مشکل دار را به معادل HTML آنها تبدیل کنیم. برای نمونه `<lt;` تبدیل می‌شود که مرورگر آن را به عنوان کarakتر `<` نمایش می‌دهد.



کarakتر	عبارت انکد شده
فضای خالی	&ampnbsp
<	&lt;
>	&gt;
&	&amp;
«	&quot;

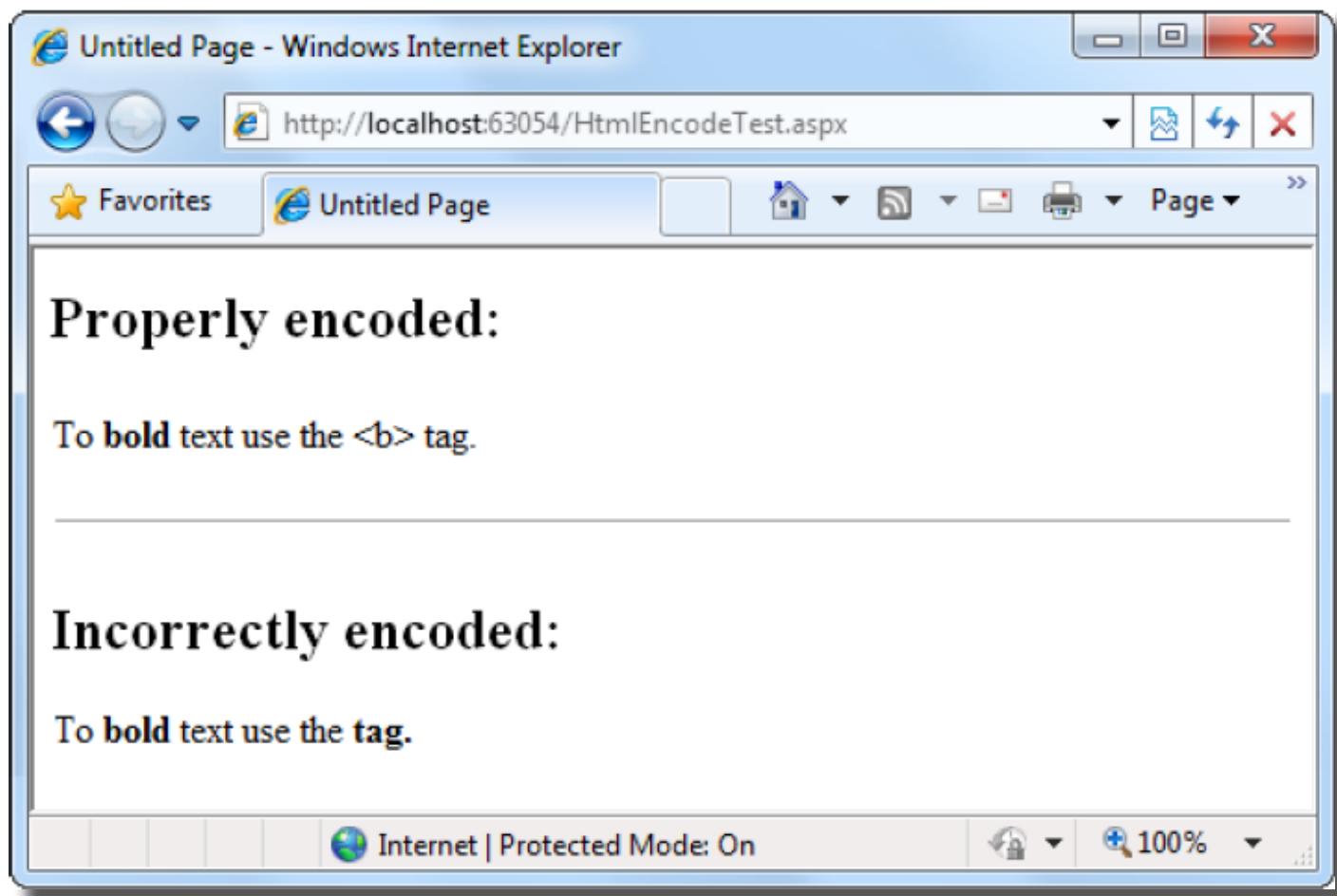
این تبدیلات را می‌توانید هم خودتان انجام دهید و هم از ویژگی `InnerText` یک کنترل HTML server استفاده نمایید. زمانی که محتوای کنترل را با استفاده از `InnerText` برابر قرار می‌دهید، هر کاراکتر اشتباهی بصورت خودکار به معادل HTML خود تبدیل می‌شود. البته اگر متن ترکیبی داشته باشید این روش بکار نمی‌آید. بنابراین می‌توانید از متدهای `HttpServerUtility.HtmlEncode()` برای جایگزینی کاراکترهای خاص استفاده نمایید.

```
// Will output as "Enter a word &lt;here&gt;" in the HTML file, but the
// browser will display it as «Enter a word < here > ».

ctrl.InnerHtml = Server.HtmlEncode("Enter a word < here > ");

ctrl.InnerHtml = "To < b > bold</b > text use the ";

ctrl.InnerHtml += Server.HtmlEncode("< b > ") + " tag.";
```



البته متدهای `HttpServerUtility.HtmlDecode()` نیز برای انجام عکس این عمل نیز وجود دارد.

## بکارگیری رویدادهای Application

در این بخش، خواهید دید که چگونه کنترل‌های ASP.NET، رویدادهایی که شما می‌توانید در کد هندل کنید را fire خواهند کرد. Application event ها به مهمی رویدادهای کنترل‌های سروری نیستند ولی ممکن است برای اجرای کارهای بیشتر از آنها استفاده کنید. برای نمونه با استفاده از رویدادهای برنامه‌ای، می‌توانید کدهای log کردن را بنویسید که هر بار که یک درخواست دریافت می‌شود، اجرا خواهد شد. این رویدادها را نمی‌توانید در code-behind یک صفحه وب هندل کنید بلکه باید در فایل دیگری به نام global.asax این کار را انجام دهید.

### فایل global.asax

این فایل اجازه نوشتن کدی که به رویدادهای عمومی برنامه پاسخ می‌دهد را دارد. این رویدادها در نقاط مختلفی از طول عمر یک برنامه تحت وب اجرا می‌شوند مانند اولین باری که application domain ایجاد می‌شود.

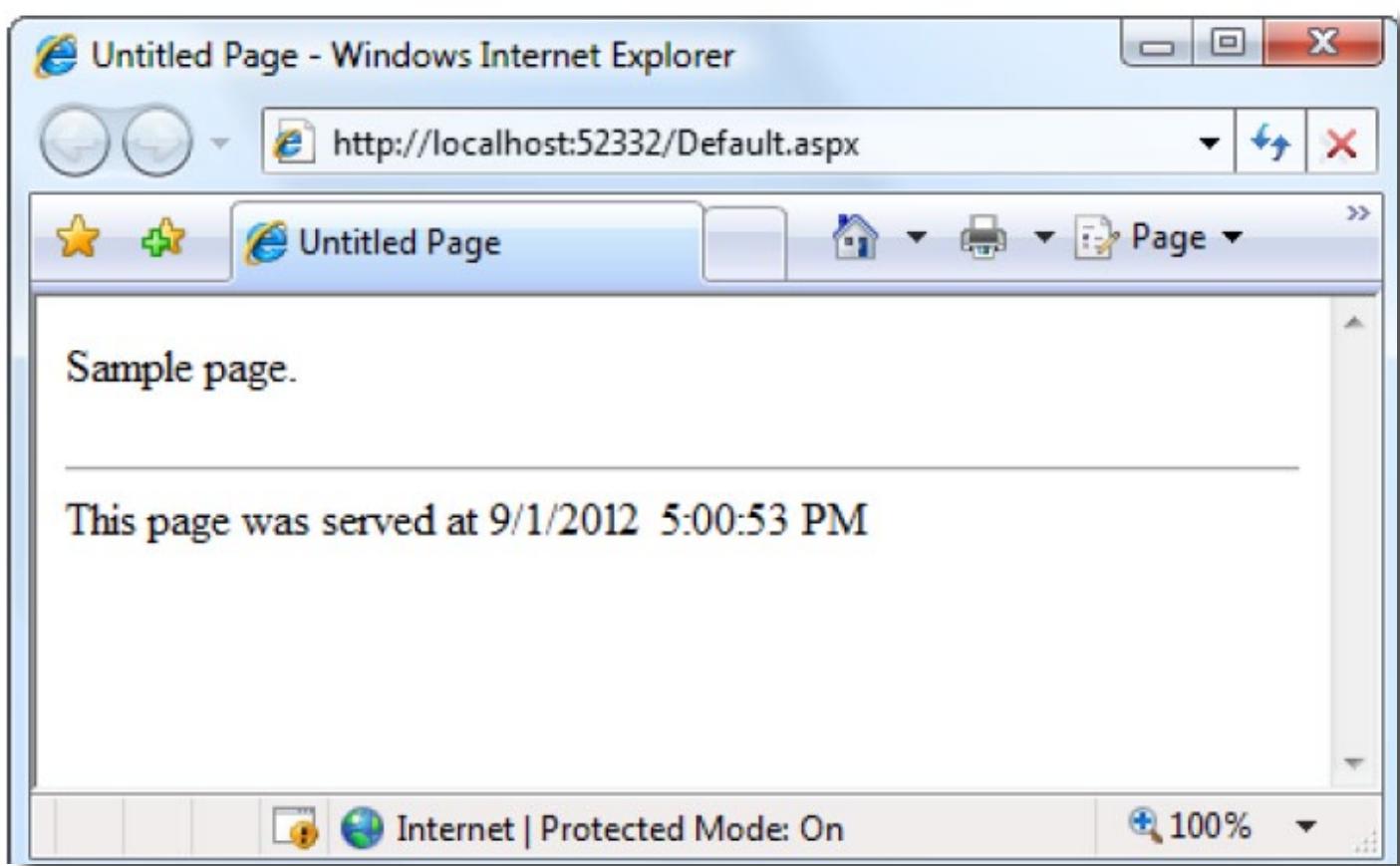
برای اضافه نمودن فایل global.asax به برنامه در VS، از گزینه add new item استفاده نمایید. این فایل شبیه سایر فایل‌های .aspx می‌باشد به جز اینکه هیچ تگ ASP.NET یا HTML ای در آن وجود ندارد.

برای نمونه رویداد زیر از متدهای Response Write() کلاس کاربرد دارد. برای نوشتن یک پاورقی در پایین صفحه با تاریخ و زمان استفاده کرده است.

```
<script language="#c" runat="server">

protected void Application_EndRequest(object sender, EventArgs e)
{
    Response.Write(" < hr />This page was served at " +
    DateTime.Now.ToString());
}

</script>
```



هر برنامه می‌تواند فقط یک فایل global.asax داشته باشد.

## Application Event

متدهندل کردن رویداد	توضیحات
Application_Start()	زمانی که برنامه شروع بکار می‌کند رخ می‌دهد که این اولین باری است که یک درخواست از کاربر دریافت می‌شود. در درخواست‌های بعدی این رویداد رخ نخواهد داد. این رویداد بیشتر برای ایجاد یا کردن بعضی از اطلاعات اولیه که بعداً مورد استفاده قرار می‌گیرد بکار می‌رود.
Application_End()	زمانی رخ می‌دهد که برنامه پایان یابد که معمولاً زمانی که وب سرور reset شود رخ می‌دهد. می‌توانید کدهای cleanup را در اینجا قرار دهید.
Application_BeginRequest()	با هر درخواستی که برنامه دریافت کند، درست قبل از اینکه کدهای صفحه اجرا شوند رخ می‌دهد.
Application_EndRequest()	با هر درخواستی که برنامه دریافت کند، درست بعد از اجرای کدهای صفحه رخ می‌دهد.
Session_Start()	هر زمان که یک درخواست کاربر جدید دریافت شود رخ می‌دهد.
Session_End()	زمانی که زمان session تمام شود (session time out) یا بصورت برنامه نویشی شده پایان یابد این رویداد رخ می‌دهد. تنها وقتی که از روش ذخیره سازی session با نام in-process استفاده کنید این رویداد رخ می‌دهد.
Application_Error()	در پاسخ به خطاهای هندل نشده رخ می‌دهد.

## پیکربندی یک برنامه ASP.NET

### فایل پیکربندی ASP.NET چندین مزیت کلیدی دارد :

- این فایل‌ها هرگز قفل نخواند شد : می‌توانید تنظیمات فایل web.config را در هر زمان انجام دهید، حتی هنگامی که برنامه در حال اجرا می‌باشد. درخواست‌هایی که در طول این تغییر در راه می‌باشند با تنظیمات قبلی پاسخ داده خواهند شد و درخواست‌های جدید با تنظیمات جدید.
- دسترسی آسان : می‌توانید این فایل را در یک کامپیوتر دیگر تغییر و سپس آن را در محل مورد نظر کپی کنید.
- این تنظیمات به راحتی ویرایش و فهمیده می‌شوند: این تنظیمات توسط هر فرد قابل خواندن می‌باشد، و این یعنی برای ویرایش آن نیاز به ابزار خاصی ندارید.

این فایل از یک سری از فرمات‌های XML از پیش تعریف شده استفاده می‌کند که همه آنها در یک المان ریشه به نام `<configuration>` قرار دارند.

```
<?xml version = "1.0" ?>
<configuration>
  <appSettings> . . . </appSettings>
  <connectionStrings> . . . </connectionStrings>
  <system.web> . . . </system.web>
</configuration>
```

این فایل case sensitive می‌باشد بنابراین نم می‌توانید بجای `<appSettings>` از `<AppSettings>` استفاده نمایید. به عنوان یک برنامه نویس شما با سه بخش در این فایل کاردارید. بخش `<appSettings>` به شما اجازه می‌دهد تا اطلاعات بخش‌های گوناگون را در آن قرار دهید.

بخش `<connectionStrings>` به شما اجازه تعریف اطلاعات اتصال به بانک اطلاعاتی را می‌دهد.

بخش `<system.web>` نیز کلیه تنظیمات ASP.NET که شما برای پیکربندی کردن به آنها نیاز دارید را نگهداری می‌کند. در این المان، المان‌های جداگانه‌ای برای هر جنبه از پیکربندی وب سایت موجود می‌باشد. برای نمونه برای تعیین نحوه کار امنیت ASP.NET می‌توانید از المان `<authentication>` و `<authorization>` استفاده کنید.

زمانی که یک وب سایت را ایجاد می‌کنید، دارای فایل web.config نمی‌باشد اما وقتی آن را در VS اجرا می‌کنید و debugging را فعال می‌کنید، VS یک فایل ابتدایی web.config برای آن ایجاد می‌کند.

```

<?xml version = “1.0”?>
<configuration>
  <appSettings>
    <add key = “aspnet:UseTaskFriendlySynchronizationContext” value = “true”/>
    <add key = “ValidationSettings:UnobtrusiveValidationMode” value = “WebForms”/>
  </appSettings>
  <system.web>
    <compilation debug = “true” targetFramework = “4.5”/>
    <httpRuntime requestValidationMode = “4.5” targetFramework = “4.5”
      encoderType = “...”/>
    <httpRuntime requestValidationMode = “4.5” targetFramework = “4.5”
      encoderType = “...”/>
    <pages controlRenderingCompatibilityVersion = “4.5”/>
    <machineKey compatibilityMode = “Framework45”/>
  </system.web>
</configuration>

```

یکی از مهمترین المان ها `<compilation>` می باشد که از دو خصیصه کلیدی استفاده می کند :

**Debug**: این خصیصه به ASP.NET می گوید که آیا می تواند برنامه را با حالت debug کامپایل کند یا نه تا بتوانید از ابزار VS داخل debugging استفاده نمایید.

 نکته: یک از معایب پشتیبانی از dubugging، کاهش سرعت اجرای برنامه می باشد، بنابراین هر زمان که خواستید برنامه را روی یک سرور live اجرا کنید، عبارت `debug=true` را حذف کنید.

**TargetFramework**: این خصیصه به VS خواهد گفت که نسخه .NET ای که می خواهید روی وب سرور استفاده کنید چیست. می توانید با تعیین پوشش بندی های مختلف و قرار دادن یک فایل web.config داخل هر یک تنظیمات خاص برای هر کدام از آنها در نظر بگیرید :

## Machine.config and root web.config files (applies to all web applications on the server)



دو فایل machine.config و web.config در مسیر زیر قرار دارند که هر وب سروری با تعدادی از تنظیمات ابتدایی که در این فایل‌ها وجود دارد اجرا می‌شود :

c:\Windows\Microsoft.NET\Framework\[Version]\Config

به شما اجازه می‌دهد، تا تنظیمات خود را در فایل web.config ذخیره کنید.

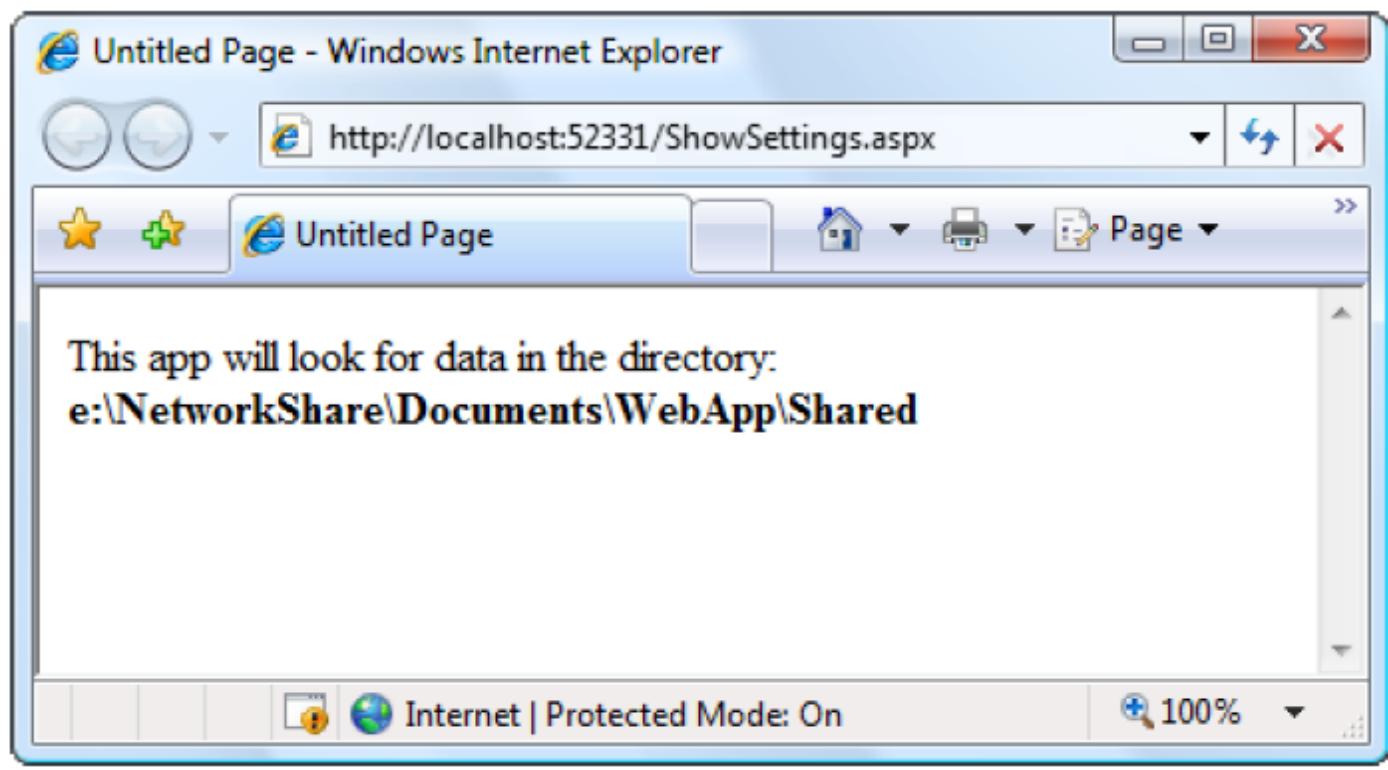
```
<?xml version = "1.0" ?>
<configuration>
  <appSettings>
    <!-- Custom application settings go here. -->
  </appSettings>
  <system.web>
    <!-- ASP.NET Configuration sections go here. -->
  </system.web>
</configuration>
```

برای مرکزیت بخشیدن به تنظیمات مهمی که در صفحات مختلفی بکار می‌رود می‌توانید از متغیرهای بالا استفاده نمایید. برای نمونه برای ایجاد متغیری که بانک اطلاعاتی را در خود نگهداری کند. هر صفحه‌ای که نیاز به این query داشته باشد می‌تواند این

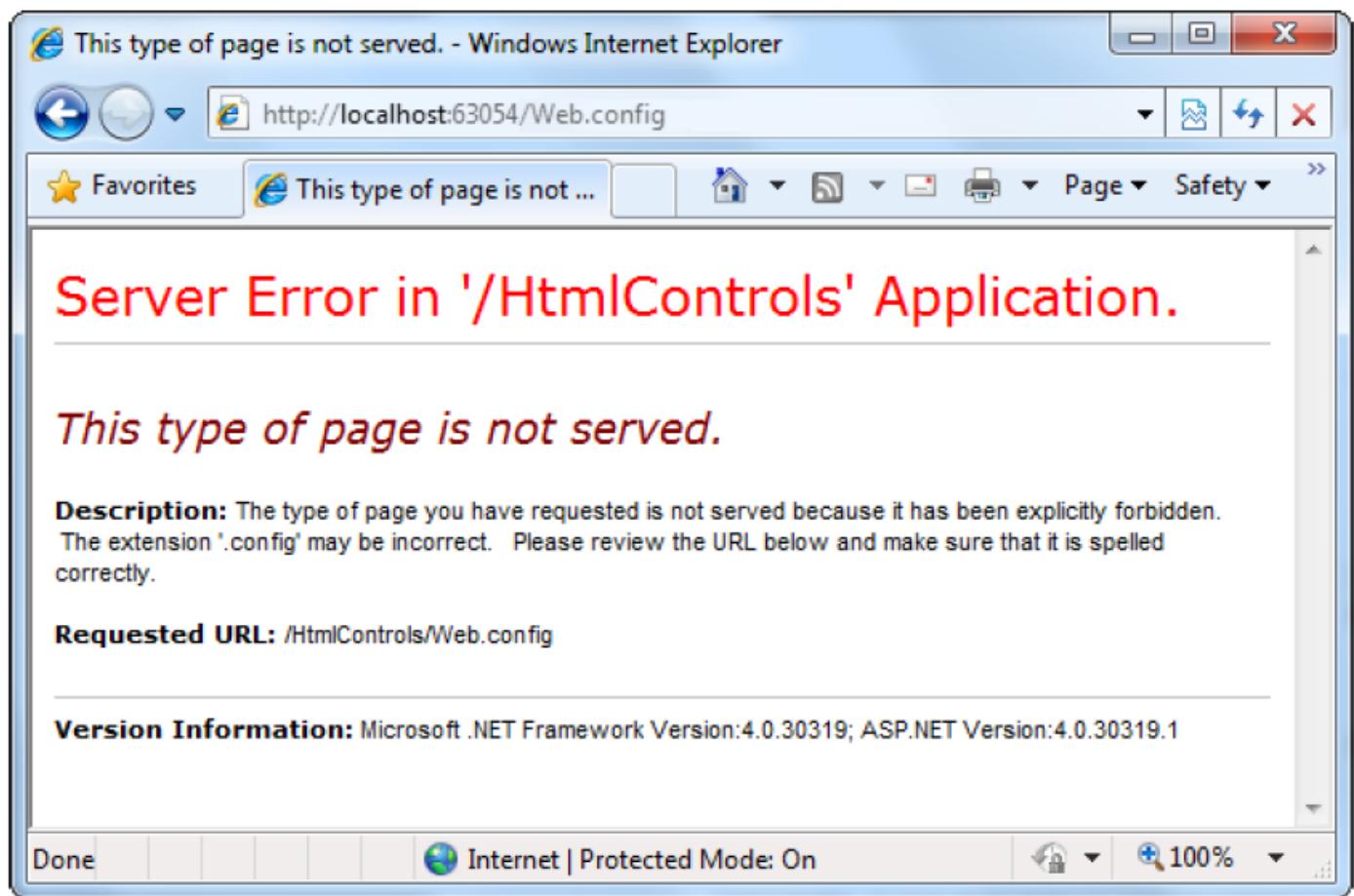
متغیر را بازیابی کند.

```
<appSettings>
  <add key = "DataFilePath"
    value = "e:\NetworkShare\Documents\WebApp\Shared" />
</appSettings>

using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.Configuration;
public partial class ShowSettings : System.Web.UI.Page
{
  protected void Page_Load()
  {
    Results.InnerHtml = "This app will look for data in the directory:<br />";
    Results.InnerHtml += "<b>" + WebConfigurationManager.AppSettings["DataFilePath"] + "</b>";
  }
}
```

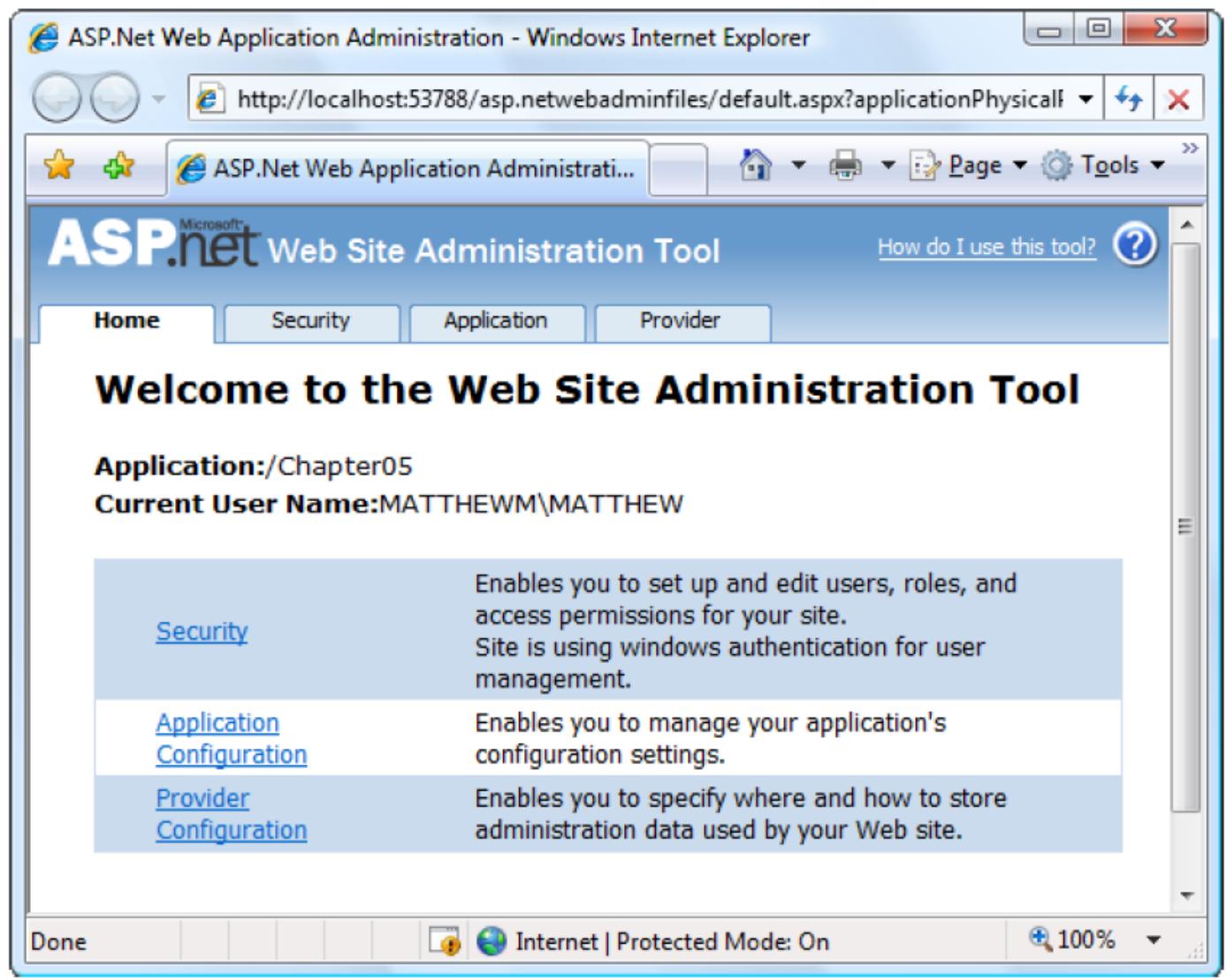


طوری پیکربندی شده است که جلوی هر درخواستی برای فایل‌های config. را بگیرد.



## بکارگیری Website Administration Tool (ابزار مدیریت وب سایت)

ویرایش فایل web.config بصورت دستی بسیار خسته کننده می‌باشد. برای کمک به شما ASP.NET یک ابزار گرافیکی پیکربندی با نام (Website Administration Tool)WAT طراحی کرده که به شما اجازه تغییر در این فایل را از طریق واسطه کاربری می‌دهد. برای راه اندازی WAT، روی آیکن VS کلیک کنید. یک مرورگر وب ظاهر می‌شود.



آنی تنظیمات را به فایل web.config اضافه می‌کند.

 نکته: WAT تنها در زمانی که در حال تولید یک برنامه تحت وب هستید کار می‌کند. به عبارت دیگر، نمی‌توانید از طریق یک وب سرور live به آن دسترسی داشته باشید. برای confige مجدد یک برنامه که در قبل نوشته و deploy شده است، می‌توانید از محیط گرافیکی IIS Manager استفاده نمایید که بعضی از ویژگی‌های مشابه با WAT را در اختیار قرار می‌دهد.

ASP.Net Web Application Administration - Windows Internet Explorer

://localhost:53788/asp.netwebadminfiles/appConfig/ManageAppSettings.aspx

ASP.NET Web Application Administration Tool

How do I use this tool? ?

Home Security Application Provider

Use this page to edit, override, or delete application settings that you do not want to hard-code into your pages. Settings can be local to your application or can be inherited from a default site-wide or computer-wide configuration. If a setting is inherited, you can override it to specify a new value for your application.

Source	Name	Value		
Local	DataFilePath	e:\NetworkShare\Documents\WebApp\Shared	<a href="#">Edit</a>	<a href="#">Delete</a>
Existing application settings: 1				
<a href="#">Create new application setting</a>				

Back

Done

Internet | Protected Mode: On

100%

# ASP.NET : توسعه برنامه های بخش اول

فصل اول - آشنایی با ~~Visual Studio~~

فصل دوم - اصول فرم های وب

فصل سوم - کنترل های وب - web control

فصل چهارم - Error Handling , Logging , Tracing

فصل پنجم - State Management



## دلایل زیادی برای استفاده از وب کنترل‌ها وجود دارد:

- ایجاد یک واسط کاربری قوی

یک کنترل وب به عنوان یک شی برنامه نویسی شده است ولی به یک المان تنها در HTML صفحه مربوط نمی‌شود. برای نمونه ممکن است از یک GridView یا Calender استفاده نمایید که چندین المان HTML در صفحه را ایجاد می‌کنند. زمانی که از برنامه‌های ASP.NET استفاده می‌کنید، نیازی به دانستن جزئیات HTML ندارید. این کنترل‌ها تگ‌های HTML مورد نیاز شما را ایجاد می‌کنند.

- یک مدل شی ثابت بوجود می‌آورند

برای نمونه در HTML، یک textbox توسط یک المان

```
<"input type="password"> <"input type="text">, <textarea>
```

نمایش داده می‌شود. در وب کنترل‌ها این سه المان به عنوان یک کنترل TextBox شناخته می‌شوند. بر اساس ویژگی‌هایی که set می‌کنید، ممکن است المان‌های HTML که توسط ASP.NET تولید می‌شوند، متفاوت باشند.

- آنها خروجی را بصورت خودکار مناسب می‌کنند

آنها می‌توانند نوع مرورگر را تشخیص داده و بصورت خودکار کدهای HTML را تعیین کنند و ویژگی پشتیبانی از javascript را برای آن بنویسند. نیازی نیست شما چیزی در مورد کلاینت بدانید زیرا ASP.NET بصورت خودکار آنها را هندل می‌کند. این ویژگی adaptive rendering نام دارد.

- آنها ویژگی‌های سطح بالایی فراهم می‌کنند

این وب کنترل‌ها به شما اجازه دسترسی به رویدادها، ویژگی‌ها و متدهای اضافی که مستقیماً ربطی به کنترل‌های HTML ندارند را می‌دهند.

## کلاس‌های پایه ای وب کنترل

کلاس کنترل	المان HTML
Label	<span>
Button	<input type="submit"> یا <input type="button">
TextBox	<input type="text">, <input type="password">, or <textarea>
CheckBox	<input type="checkbox">
RadioButton	<input type="radio">
Hyperlink	<a>
LinkButton	<a> with a contained <img> tag
ImageButton	<input type="image">
Image	<img>

<select size="X"> where X is the number of rows that are visible at once	ListBox
<select>	DropDownList
<table> با چند تگ <input type="checkbox"> یک لیست یا	CheckBoxList
<table> با چند تگ <input type="radio"> یک لیست یا	RadioButtonList
<ol> یا یک لیست نامرتب <ul> یک لیست مرتب (شماره گذاری شده)	BulletedList
<div>	Panel
<table>, <tr>, and <td> or <th>	Table, TableRow, and TableCell

## تگ‌های وب کنترل

تگ‌های ASP.NET دارای یک فرمت مناسب هستند. آنها همیشه با پیشوند `asp:` شروع می‌شوند که در ادامه نام کلاس می‌آید. اگر تگ جداگانه‌ای برای بسته شدن تگ مورد نظر ندارند باید با `/>` پایان یابند. خصیصه `runat="server"` در همه آنها موجود است.

```
<asp:TextBox ID="txt" runat="server" />
```

زمانی که یک کلاینت صفحه `.aspx` که کد بالا در آن موجود می‌باشد را درخواست می‌کند، HTML زیر تولید می‌شود :

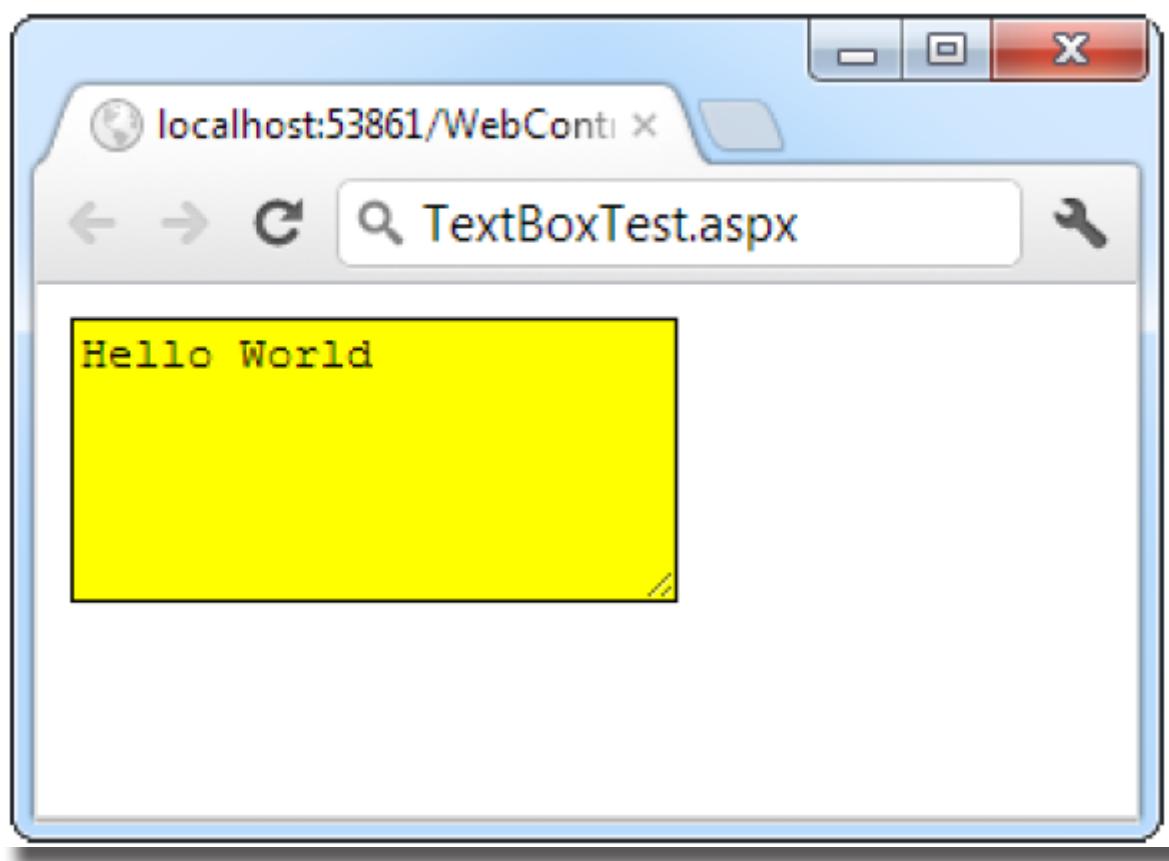
```
<input type="text" ID="txt" name="txt" />
```

کد زیر یک TextBox سفارشی سازی شده است :

```
<asp:TextBox ID="txt" BackColor="Yellow" Text="Hello World"
    ReadOnly="True" TextMode="MultiLine" Rows="5" runat="server" />
```

کد HTML :

```
<textarea name="txt" rows="5" cols="20" readonly="readonly" ID="txt"
    style="background-color:Yellow;">Hello World</textarea>
```



## کلاس‌های وب کنترل

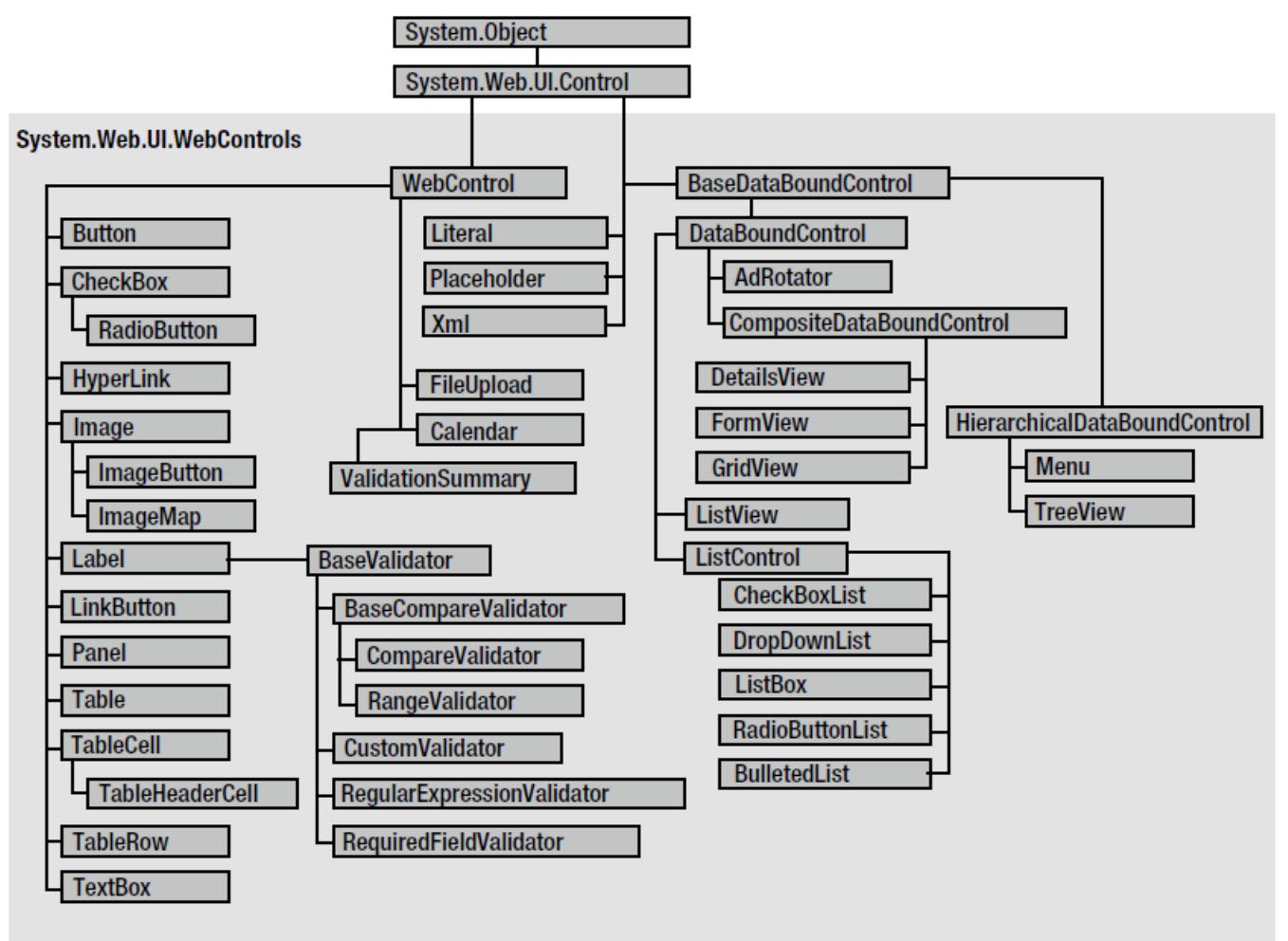
کنترل‌های وب در فضای نام System.Web.UI.WebControls تعریف می‌شوند.

اکثر وеб کنترل‌ها از کلاس پایه webcontrol ارث بری می‌کنند. این کلاس کاربری خاصی برای کارهایی نظیر binding و بعضی از ویژگی‌ها که در بیشتر وеб کنترل‌ها استفاده می‌شوند را دارا می‌باشد.



توضیحات	ویژگی
کلید میانبر تک حرفی را مشخص می‌کند. برای نمونه اگر این ویژگی را برای یک کنترل با Y مقداردهی کنید، ترکیب Alt + Y بصورت خودکار فوکوس را به این وeb کنترل می‌برد.	AccessKey
عرض نوار مرزی کنترل را مشخص می‌کند.	BackColor, Border-, ForeColor Color BorderWidth
مجموعه‌ای از همه کنترل‌هایی که درون کنترل مورد نظر وجود دارد را ارائه می‌دهد. هر شی از نوع System.Web.Control می‌باشد بنابراین باید مرجع را به کنترل مورد نظر خود cast کنید.	BorderStyle Controls
فعال و غیر فعال کردن کنترل	Enabled
فعال و غیر فعال کردن مدیریت خودکار state هر کنترل.	EnableViewState
	Font
	Height and

نامی که شما برای کار با کنترل در کد از آن استفاده می‌کنید.	ID
ارجاعی به صفحه وبی که این صفحه در آن وجود دارد.	Page
ارجاعی به کنترلی که شامل این کنترل می‌باشد. اگر کنترل مستقیماً درون صفحه باشد، این ویژگی ارجاعی به شی صفحه بر می‌گردد.	Parent
ترتیب Tab کنترل‌ها	TabIndex
نمایش یک پیام متنی در زمانی که ماوس را روی کنترل می‌برید.	ToolTip
زمانی که با false مقداردهی می‌شود، کنترل مورد نظر مخفی شده ولی کد HTML ای برای آن رender نمی‌شود.	Visible



## Units

همه کنترل هایی که از واحدهای اندازه گیری مانند `BorderWidth` و ... استفاده می کنند باید از ساختار `Unit` استفاده بکنند که ترکیبی از یک مقدار عددی با نوعی از اندازه گیری (`pixels`, `percentage` و ...) می باشد. زمانی که می خواهید این ویژگی ها را در یک تگ کنترل `set` کنید، برای تعیین نوع `unit` باید مطمئن شوید `px` (برای پیکسل) یا `%` (برای درصد) در انتهای عدد موجود باشد.

```
<asp:Panel Height="300px" Width="50%" ID="pnl" runat="server" />
```

یا

```
// Convert the number 300 to a Unit object
// representing pixels, and assign it.

pnl.Height = Unit.Pixel(300);
```

یا

```
// Create a Unit object.

Unit myUnit = new Unit(300, UnitType.Pixel);

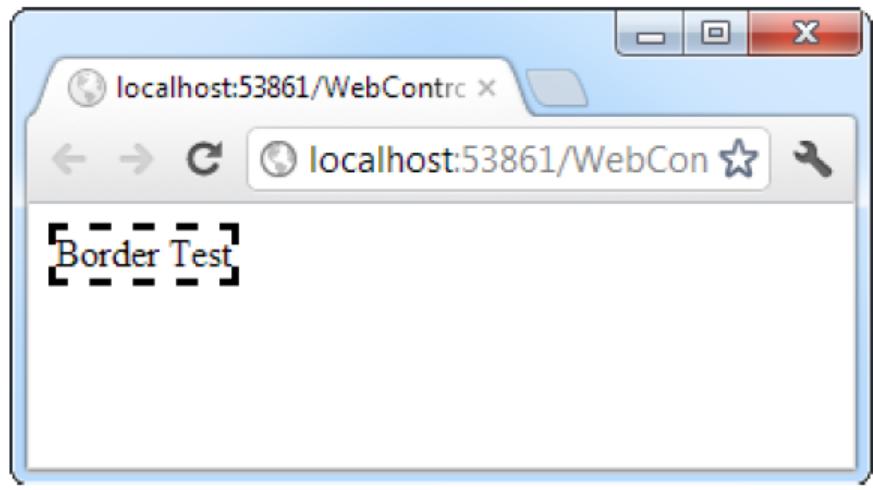
// Assign the Unit object to several controls or properties.

pnl.Height = myUnit;
pnl.Width = myUnit;
```

## Enumerations

برای گروه بندی مقادیر ثابت (constant) در کتابخانه کلاس دات نت، از انواع شمارشی (Enumeration) استفاده می شود. برای نمونه در هنگام تعیین مقداری برای ویژگی `BorderStyle` (نوع کادر)، می توانید یکی از چندین مقدار از پیش تعریف شده را از نوع شمارشی `BorderStyle` انتخاب نمایید.

```
ctrl.BorderStyle = BorderStyle.Dashed;
```



توضیحات	ویژگی
نام فونت را تعیین می‌کند	Name
آرایه ای از string که نام فونت را ذخیره می‌کند. مرورگر اولین فونتی که منطبق با فونتی است که روی دستگاه کاربر نصب می‌باشد را انتخاب می‌کند	Names
اندازه فونت می‌باشد که بصورت absolute یا relative ارائه می‌شود.	Size
ویژگی های Boolean ای می‌باشند که استایل می‌باشند.	Bold, Italic, Strikeout, Underline, and Overline

```

ctrl.Font.Name = "Verdana";
trl.Font.Bold = true;
// Specifies a relative size.
trl.Font.Size = FontUnit.Small;
// Specifies an absolute size of 14 pixels.
trl.Font.Size = FontUnit.Point(14);

```

### : absolute

```

<asp:TextBox Font-Name="Tahoma" Font-Size="40" Text="Size Test" ID="txt"
runat="server" />

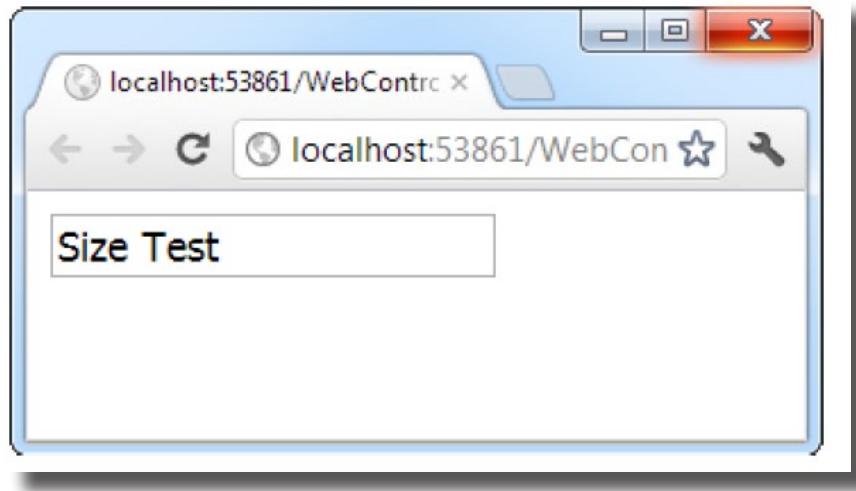
```

### : relative

```

<asp:TextBox Font-Name="Tahoma" Font-Size="Large" Text="Size Test"
ID="TextBox1" runat="server" />

```



اختصاص چندین فونت :

```
<asp:TextBox Font-Names="Verdana, Tahoma, Arial"
Text="Size Test" ID="TextBox2" runat="server" />
```

## Embedded Fonts

اخیراً مرورگرها از ویژگی با نام web font یا embede font پشتیبانی می‌کنند. وقتی کاربری صفحه شما را می‌بیند، مرورگر بصورت خودکار، فونت مربوطه را دانلود و بصورت موقت برای نمایش صفحه از آن استفاده می‌کند. این ویژگی به شما اجازه استفاده از فونت‌های مختلف را بدون نگرانی در مورد وجود این فونت‌ها در دستگاه کاربر می‌دهد.

این ویژگی از نظر تکنیکی مربوط به CSS 3 می‌باشد. در ASP.NET راه مشخصی برای انجام این کار نیست.

## Default Button

ASP.NET به شما اجازه طراحی یک دکمه به عنوان دکمه پیش فرض را می‌دهد که در هنگامی که کاربر دکمه Enter فشار داد، این دکمه کلیک شود. برای این کار باید ویژگی `HtmlForm.DefaultButton` را با شناسه کنترل مورد نظرتان مقدار دهی کنید.

```
<form id="Form2" DefaultButton="cmdSubmit" runat="server">
```

اگر در یک صفحه به بیش از یک Default Button نیاز داشتید باید کنترل‌های مرتبط به هم را در Panel‌های جداگانه قرار دهید این ویژگی را برای هر پنل مقدار دهی کنید.

## List Control

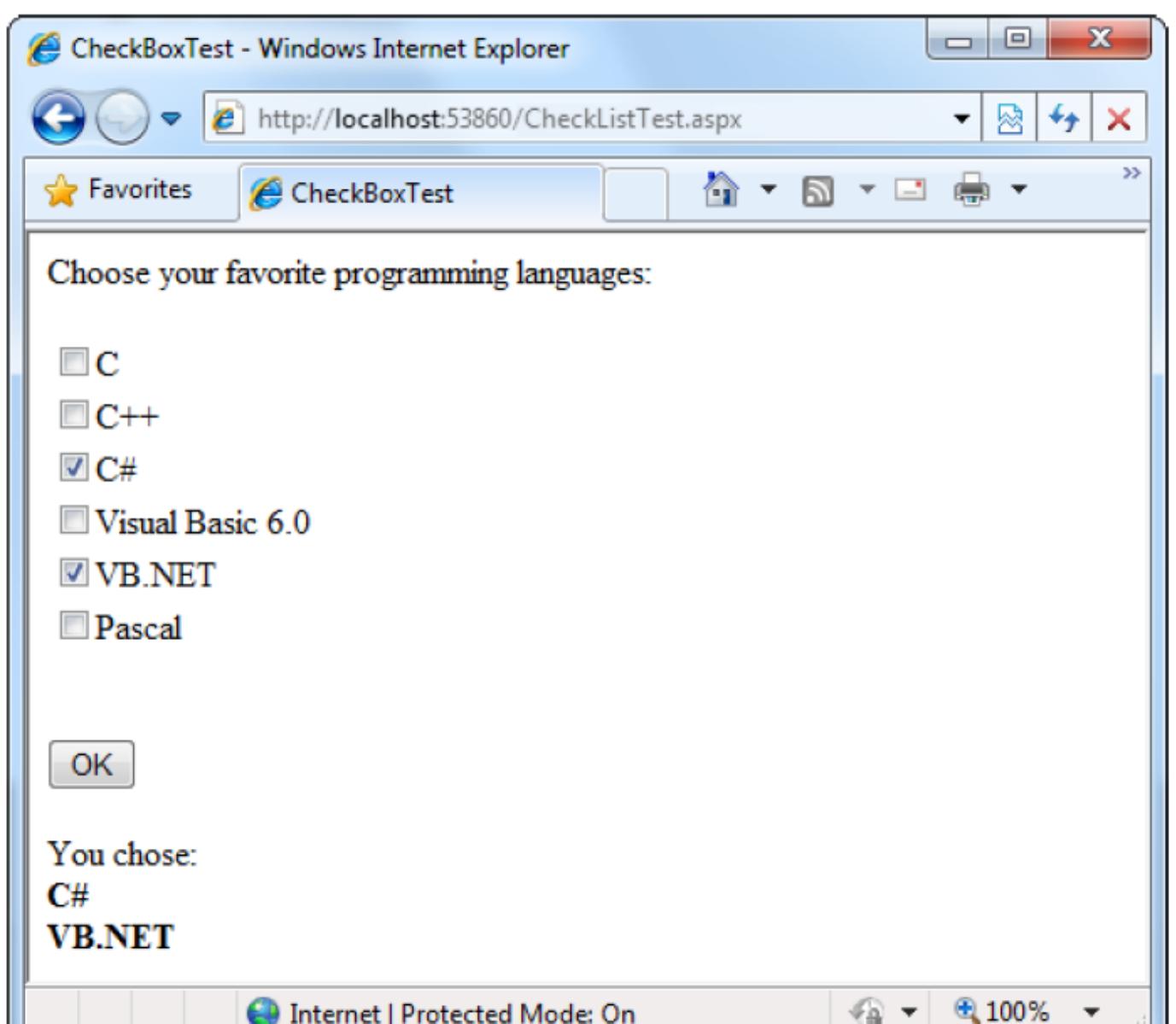
لیست کنترل‌ها شامل BulletedList, ListBox, DropDownList, CheckBoxList, RadioButtonList و SelectListItem هستند. آنها تقریباً شبیه هم کار می‌کنند ولی متفاوت رندر می‌شوند. همه کنترل‌های انتخابی، ویژگی SelectedIndex را که سطر انتخاب شده را ارائه می‌دهد، دارا می‌باشند. شی ListItem سه ویژگی مهم با نام‌های Text (متن نمایش داده شده)، Value (مقدار مخفی شده از Seleted (HTML) و true (Mقدار مشخص می‌کند آیتم انتخاب شده است یا نه)

```
ListItem item;
item = Currency.Items[Currency.SelectedIndex];
```

یا

```
ListItem item;
item = Currency.SelectedItem;
```

## Multiple-Select List Controls



```

<%@ page language="#c" autoeventwireup="true" codefile="CheckListTest.aspx.cs"
inherits="CheckListTest" %>

<!DOCTYPE html>

<html>
  <head id="Head1" runat="server">
    <title>CheckBoxTest</title>
  </head>
  <body>
    <form id="Form1" runat="server">
      <div>
        Choose your favorite programming languages:<br />
        <br />
        <asp:CheckBoxList ID="chklst" runat="server" />
        <br />
        <br />
        <asp:Button ID="cmdOK" Text="OK" OnClick="cmdOK_Click" runat="server" />
        <br />
        <br />
        <asp:Label ID="lblResult" runat="server" />
      </div>
    </form>
  </body>
</html>

```

```

protected void Page_Load(Object sender, EventArgs e)
{
  if (!this.IsPostBack)
  {
    chklist.Items.Add("C");
    chklist.Items.Add("C++");
    chklist.Items.Add("#c");
    chklist.Items.Add("Visual Basic 6.0");
    chklist.Items.Add("VB.NET");
  }
}

```

```

        chklst.Items.Add("Pascal");
    }

}

protected void cmdOK_Click(object sender, EventArgs e)
{
    lblResult.Text = "You chose:<b>";
    foreach (ListItem lstItem in chklst.Items)
    {
        if (lstItem.Selected == true)
        {
            // Add text to label.
            lblResult.Text += "<br />" + lstItem.Text;
        }
    }
    lblResult.Text += "</b>";
}

```

## BulletedList Control

این کنترل یک کنترل سروری معادل با المان <ul><li> می‌باشد.



توضیحات	ویژگی
نوع لیست را انتخاب می‌کند. عدد، حروف الفباء، حروف رومی، دایره، مربع و ...	BulletStyle
اگر نشانی یک تصویر را برای آن تعیین کنید، آن تصویر در سمت چپ هر آیتم به عنوان bullet قرار می‌گیرد.	BulletImageUrl
اولین مقدار برای شروع را تعیین می‌کند. برای نمونه اگر آن را با ۳ مقدار دهی کنید، برای اعداد لیست شما با ۳ و ۴ و ۵ و ... و برای حروف از c,d,e... آغاز می‌شود.	FirstBulletNumber
تعیین می‌کند آیا متن هر آیتم متن باشد یا یک هایپرلینک باشد ( hyperlink یا linkbutton ). تفاوت LB با HL در داشتن رویداد Click می‌باشد. HL این رویداد را ندارد.	DisplayMode

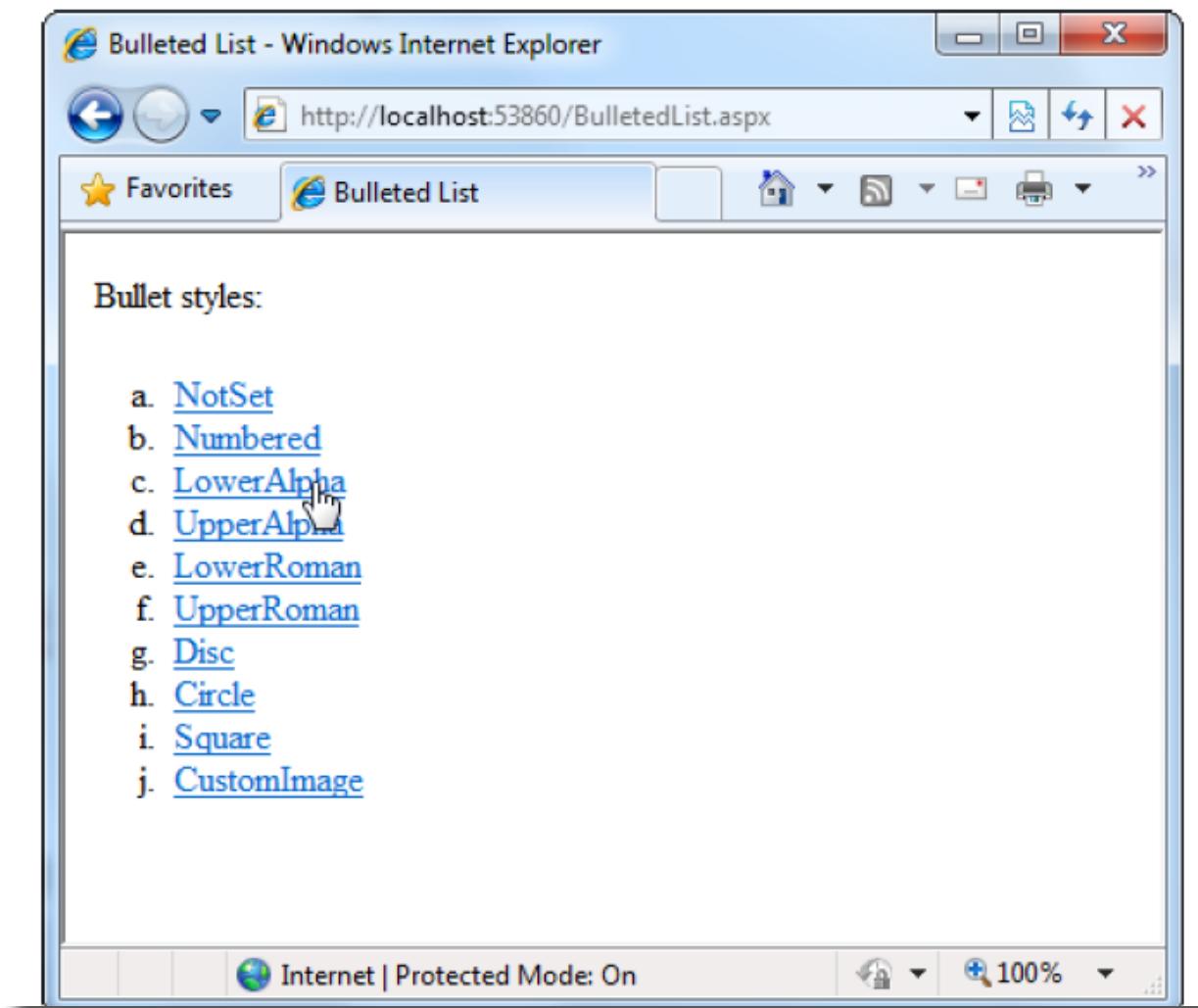
```

<asp:BulletedList BulletStyle="Numbered" DisplayMode="LinkButton"
ID="BulletedList1" OnClick="BulletedList1_Click" runat="server">
</asp:BulletedList>

protected void BulletedList1_Click(object sender, BulletedListEventArgs e)
{
    // Find the clicked item in the list.
    // (You can't use the SelectedIndex property here because static lists
    // don't support selection.)

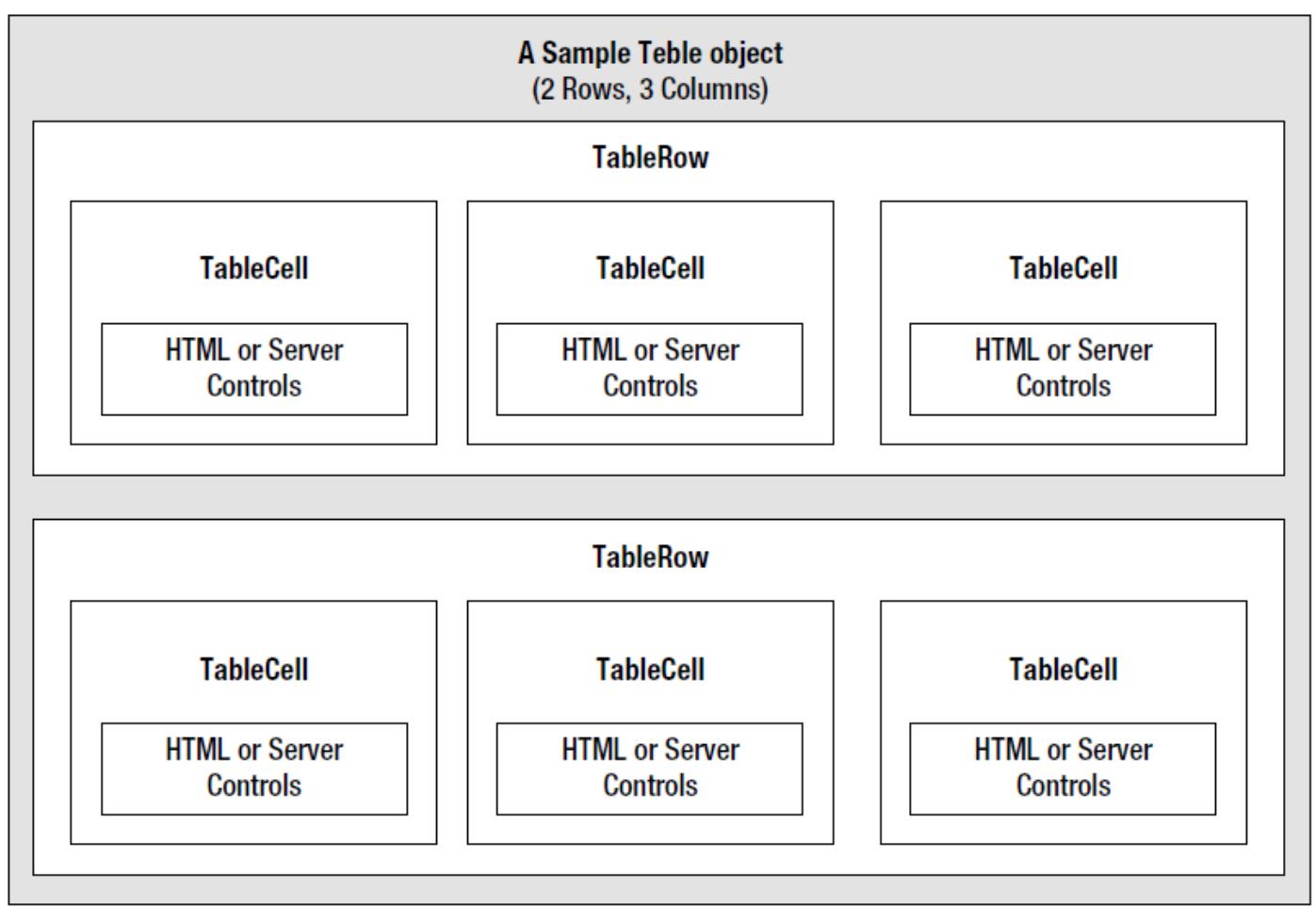
    string itemText = BulletedList1.Items[e.Index].Text;
    Label1.Text = "You choose item" + itemText;
}

```



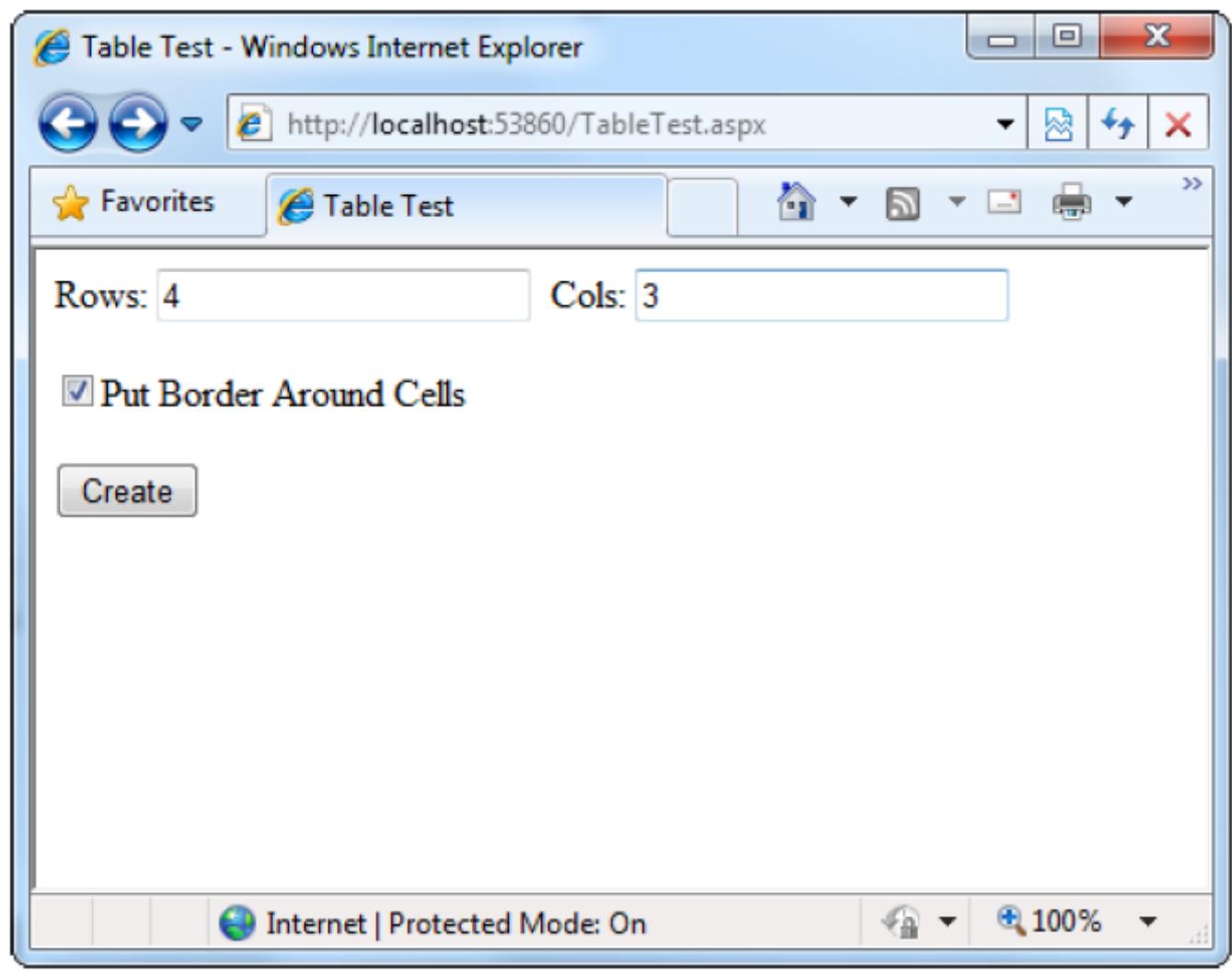
## Table Controls

هر شی Table، شامل یک یا چند شی TableRow می‌باشد. هر شی TableRow شامل یک یا چند TableCell می‌باشد که می‌تواند کنترل‌های HTML یا ASP.NET را در خود جای دهد. اگر با آشنایی موجود در table آشناسته باشید همین شباهت‌ها را خواهید دید.



برای ایجاد یک Table بصورت پویا باید از همان روش استفاده شده برای سایر کنترل‌های وب استفاده نمایید.

برای نمونه شما در زیر، تعداد ردیف و ستون را تعیین و پس از کلیک روی دکمه Create آنها در پایین صفحه ایجاد می‌شوند:



```

<%@ page language="#c" autoeventwireup="true" codefile="TableTest.aspx.cs"
inherits="TableTest" %>

<!DOCTYPE html>
<html>
<head id="Head1" runat="server">
    <title>Table Test</title>
</head>
<body>
    <form id="Form1" runat="server">
        <div>
            Rows:
            <asp:TextBox ID="txtRows" runat="server" />
            &nbsp; Cols:
            <asp:TextBox ID="txtCols" runat="server" />
            <br />
    </form>

```

```

<br />
<asp:CheckBox ID="chkBorder" runat="server" Text="Put Border Around Cells" />
<br />
<br />
<asp:Button ID="cmdCreate" OnClick="cmdCreate_Click" runat="server" Text="Create" />
<br />
<br />
<asp:Table ID="tbl" runat="server" />
</div>
</form>
</body>
</html>

```

همانطور که می بینید جدول دارای هیچ ردیف و ستون از پیش تعریف شده ای نمی باشد:

```

<asp:Table ID="tbl" runat="server">
    <asp:TableRow ID="row" runat="server">
        <asp:TableCell ID="cell" runat="server">A Sample Value</asp:TableCell>
    </asp:TableRow>
</asp:Table>

```

در زمان لود شدن صفحه، یک کادر دور آن اضافه می شود :

```

protected void Page_Load(Object sender, EventArgs e)
{
    // Configure the table's appearance.
    // This could also be performed in the aspx.file
    // or in the cmdCreate_Click event handler.

    tbl.BorderStyle = BorderStyle.Inset;
    tbl.BorderWidth = Unit.Pixel(1);
}

```

```

protected void cmdCreate_Click(object sender, EventArgs e)
{
    // Remove all the current rows and cells.
    // This is not necessary if EnableViewState is set to false.

    tbl.Controls.Clear();

    int rows = Int32.Parse(txtRows.Text);
    int cols = Int32.Parse(txtCols.Text);

    for (int row = 0; row < rows; row++)
    {
        // Create a new TableRow object.

        TableRow rowNew = new TableRow();

        // Put the TableRow in the Table.

        tbl.Controls.Add(rowNew);

        for (int col = 0; col < cols; col++)
        {
            // Create a new TableCell object.

            TableCell cellNew = new TableCell();

            cellNew.Text = "Example Cell (" + row.ToString() + ",";

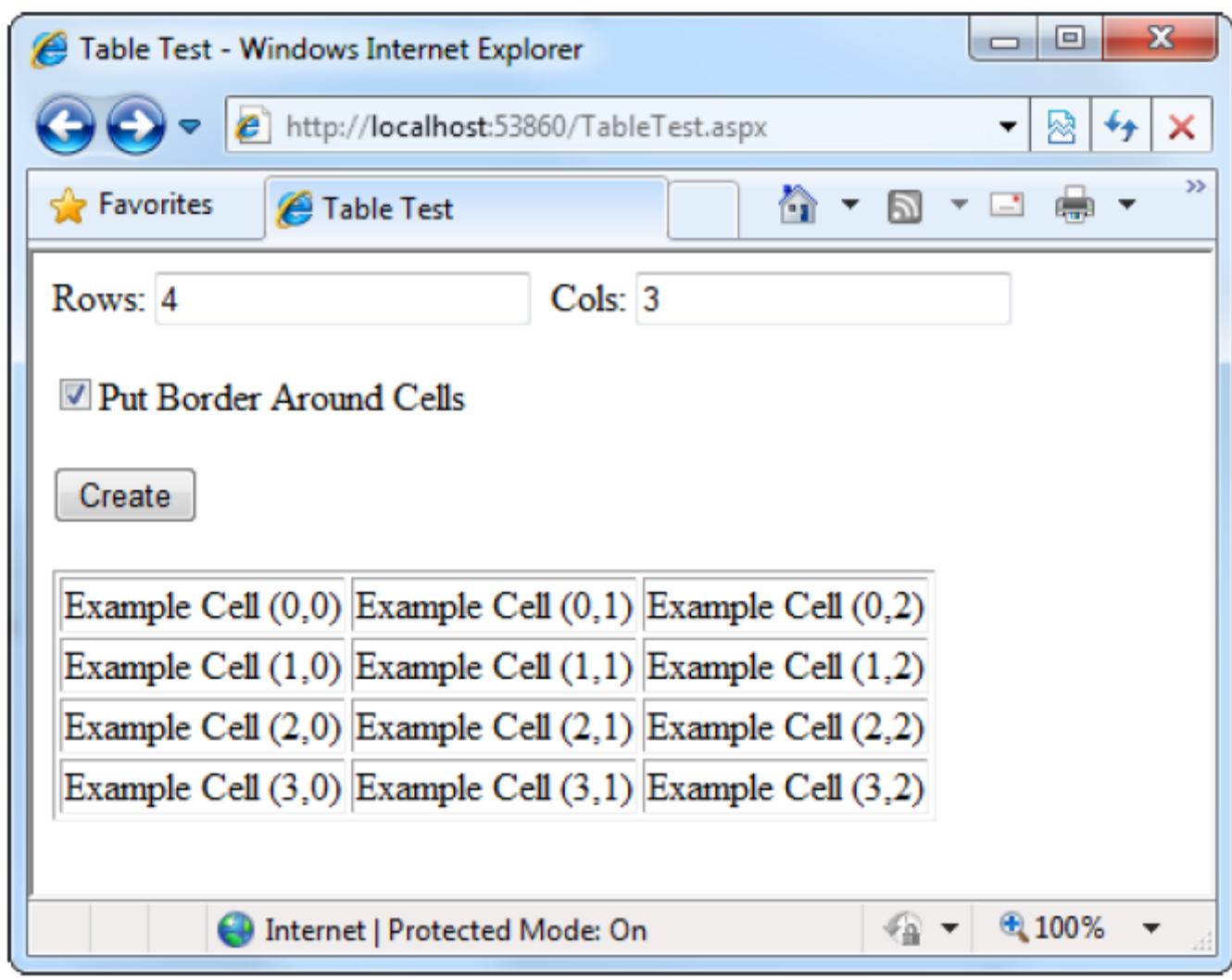
            cellNew.Text += col.ToString() + ")";

            if (chkBorder.Checked)
            {
                cellNew.BorderStyle = BorderStyle.Inset;
                cellNew.BorderWidth = Unit.Pixel(1);
            }

            // Put the TableCell in the TableRow.

            rowNew.Controls.Add(cellNew);
        }
    }
}

```



کد بالا از Controls Collection برای اضافه نمودن کنترل‌های فرزند (child) استفاده می‌کند. هر کنترل Container، این ویژگی را ارائه می‌دهد. همچنین می‌توانید برای اضافه کردن کنترل‌های وب به TableCell از TableCell Controls استفاده کنید.

همچنین می‌توانیم از کدهای زیر نیز استفاده نماییم :

```
// Create a new TableCell object.
cellNew = new TableCell();

// Create a new Label object.
Label lblNew = new Label();

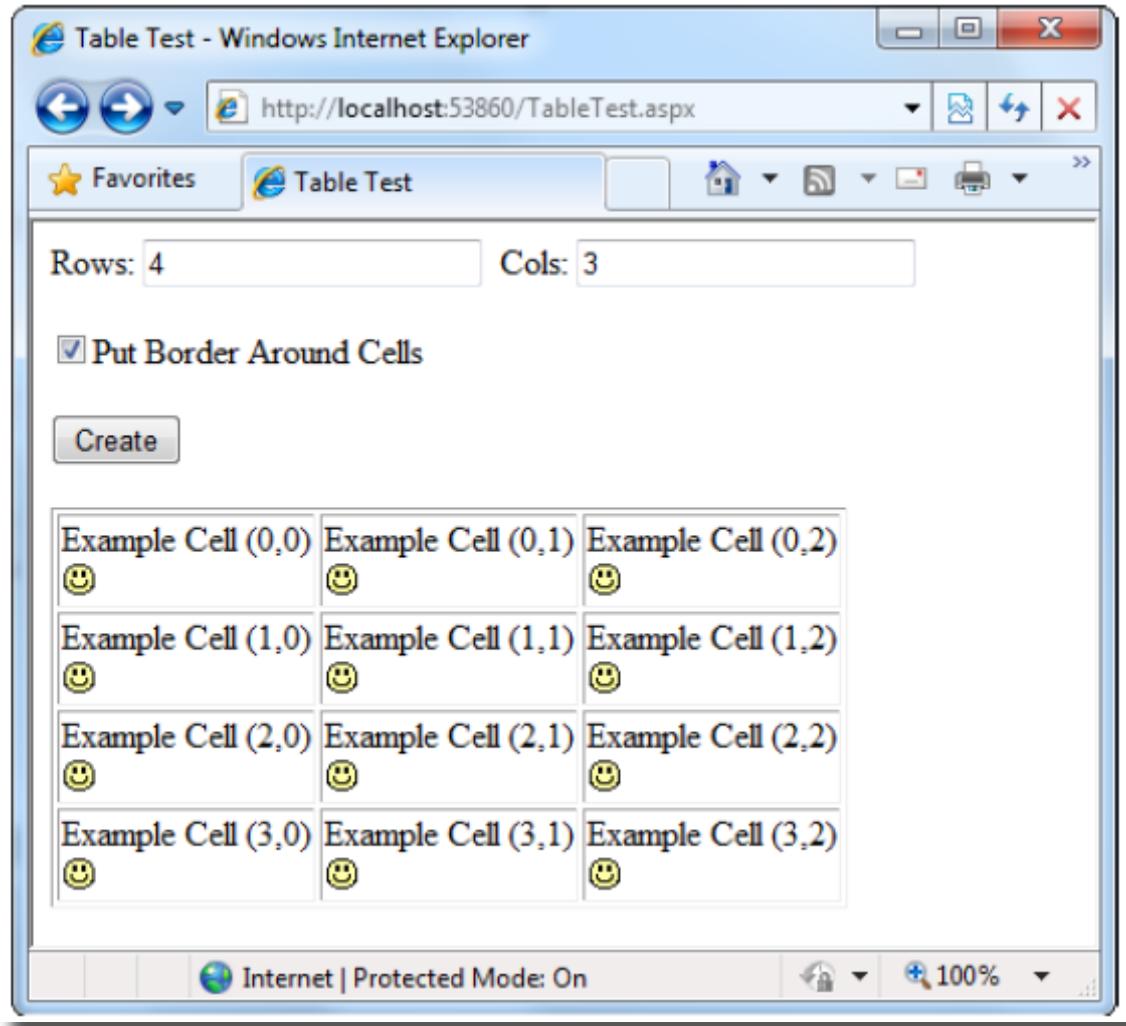
lblNew.Text = "Example Cell (" + row.ToString() + ", ";
lblNew.Text += col.ToString() + "<br />";

System.Web.UI.WebControls.Image imgNew = new System.Web.UI.WebControls.Image();
imgNew.ImageUrl = "cellpic.png";
// Put the label and picture in the cell.
```

```

cellNew.Controls.Add(lblNew);
cellNew.Controls.Add(imgNew);
// Put the TableCell in the TableRow.
rowNew.Controls.Add(cellNew);

```



## Web Control Events and AutoPostBack

در شکل زیر ترتیب پردازش رویدادهای یک صفحه وب را مشاهده خواهید کرد. پرسش این است که چگونه می‌توانید کدهای سمت سروری بنویسید که به رویدادهای سمت کلاینت عکس العمل انجام دهند.

بعضی از رویدادها مانند رویداد Click دکمه فوراً رخ می‌دهند زیرا زمانی که کلیک می‌شود، دکمه صفحه را (کل صفحه به سرور ارسال می‌شود) می‌کند. اما سایر کارهایی که باعث رخدادن یک رویداد می‌شود، صفحه را post back نمی‌کند. برای نمونه، زمانی که کاربر متن درون textbox را تغییر می‌دهد (که رویداد TextChanged را فعال می‌کند) یا زمانی که آیتمی را از یک لیست انتخاب می‌کند (که رویداد SelectedIndexChanged را فعال می‌کند). ممکن است بخواهید به این رویدادها پاسخ دهید اما بدون postback این کار امکان ندارد.

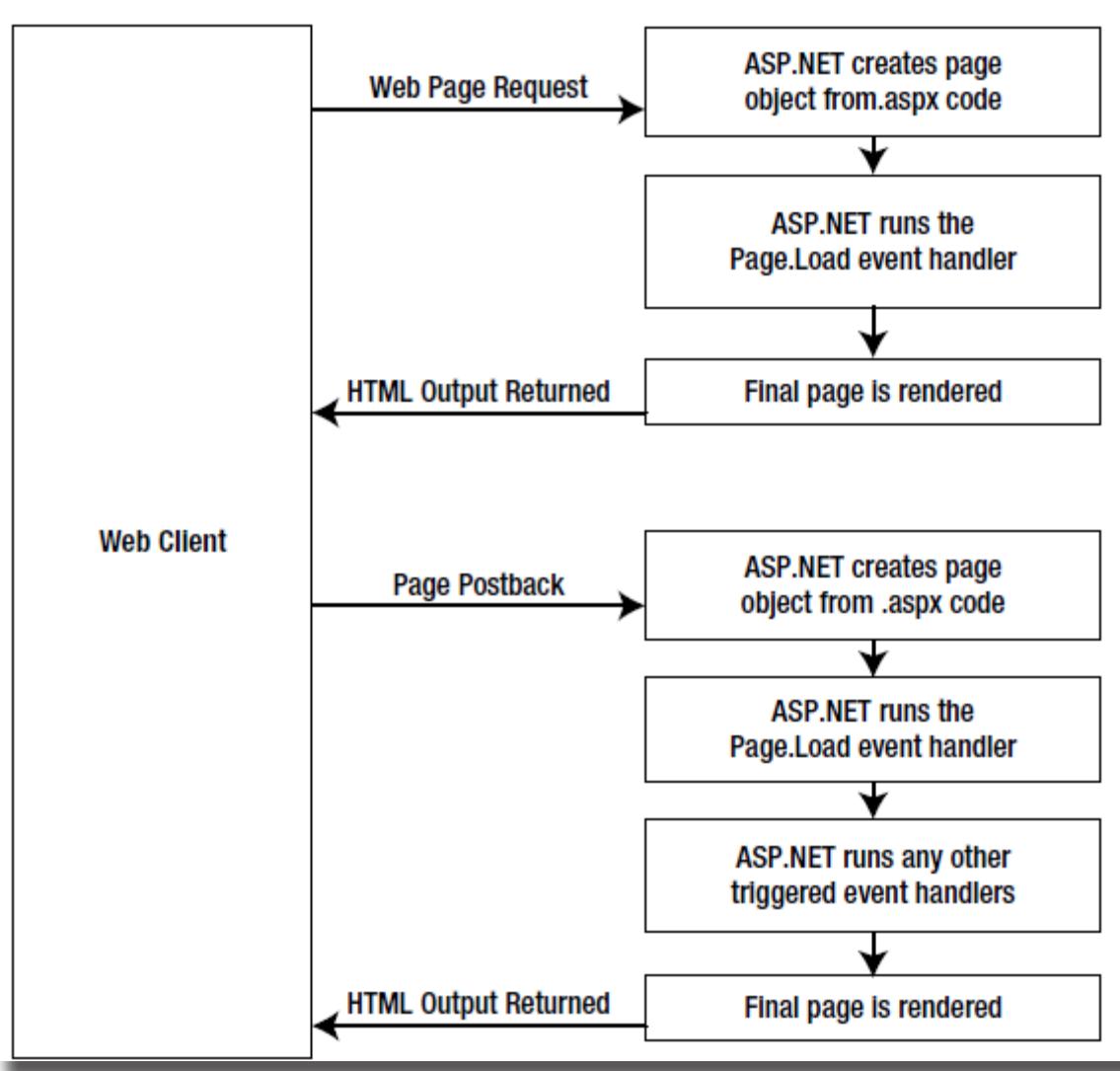
## ASP.NET، این کار را با این دو راه برای شما هندل می‌کند:

- می‌توانید تا postback بعدی منتظر بمانید. برای نمونه تصویر کنید می‌خواهید عکس العملی به رویداد SelectedIndexChanged نشان دهید. با انتخاب یک آیتم از لیست توسط کاربر هیچ اتفاقی نخواهد افتاد. اما زمانی که کاربر روی یک دکمه کلیک کند، صفحه postback شده و دو رویداد fire می‌شود.
- می‌توانید از ویژگی automatic postback استفاده کنید تا کنترل را مجبور کنید که اگر عمل خاصی از کاربر را تشخیص داد، صفحه را postback کند. در سناریوی ما، اگر کاربر یک آیتم از لیست را انتخاب کند، صفحه باید post back شود، که شما اجرا شود و نسخه جدید صفحه برگردانده شود.

نکته: تمامی کنترل‌های وب input از automatic postback پشتیبانی می‌کنند.

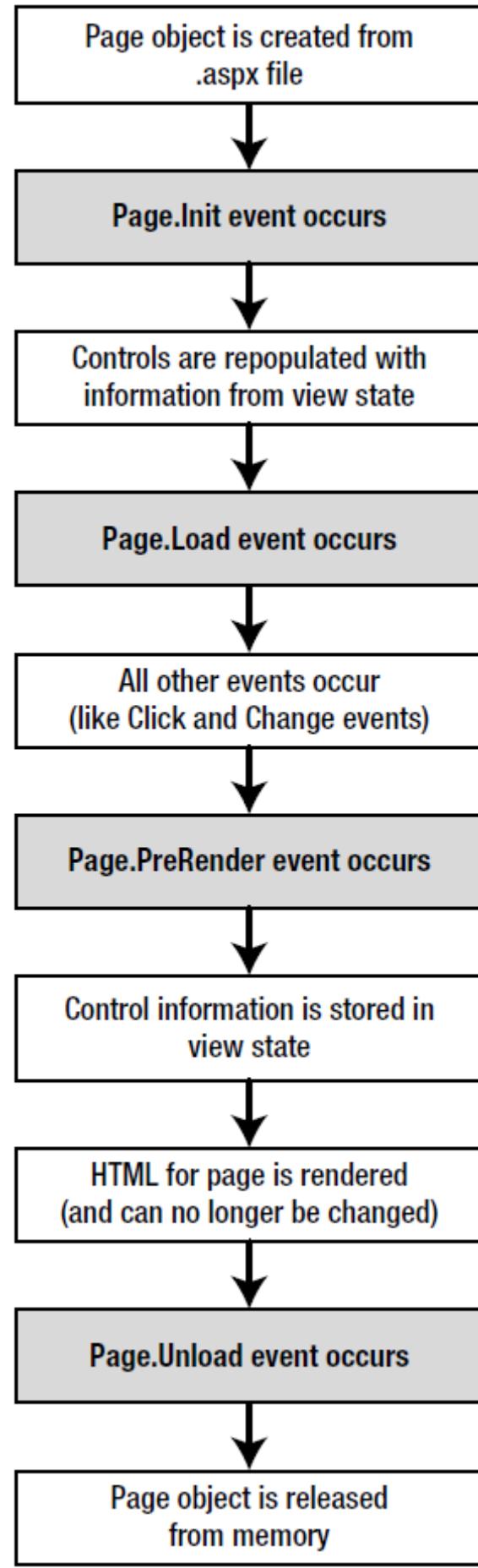


آیا همیشه postback می‌شود	وب کنترل فراهم کننده رویداد	رویداد
True	Button, ImageButton	Click
False	TextBox (fires only after the user)	TextChanged
False	changes the focus to another control)	CheckedChanged
False	CheckBox, RadioButton	SelectedIndexChanged



اگر می‌خواهید رویدادی فوراً انجام شود (مانند `TextChanged`, ..., `AutoPostBack` کنترل مربوطه را با `True` مقدار دهی کنید. زمانی که صفحه postback می‌شود، ASP.NET کلیه اطلاعات را لود می‌کند و به کد شما اجازه اجرای تعدادی پردازش اضافی در قبل از برگرداندن صفحه به کاربر را می‌دهد.

این سیستم PostBack، برای همه رویدادها مناسب نمی‌باشد. برای نمونه، تعدادی از رویدادهایی که ممکن است با آنها در برنامه‌های ویندوزی آشنایی داشته باشد، (مانند رویدادهای حرکتی یا فشردن دکمه) در برنامه‌های ASP.NET بکار نمی‌روند.



## چگونگی عملکرد رویداد PostBack

ASP.NET، از قابلیت‌های کدهای سمت کلاینت جاوا اسکریپت، به عنوان پلی بین کدهای client-side و server-side استفاده می‌کند.

زبان‌های دیگر اسکریپت نویسی مانند VBScript نیز وجود دارد، اما جاوا اسکریپت، تنها زبانی است که با همه مرورگرهای مدرن شامل Opera, IE, Chrome, FireFOx, Safari کار می‌کند.

اگر یک صفحه وب ایجاد کنید که شامل یک یا چند کنترل باشد که از AutoPostBack استفاده می‌کنند، ASP.NET یک متدهای HTML رندر شده اضافه می‌کند. این متدهای doPostBack نامیده می‌شود. زمانی که فراخوانی می‌شود، یک postback را ایجاد و داده‌ها را به سرور وب ارسال می‌کند.

ASP.NET، همچنین دو فیلد input hidden که برای ارسال اطلاعات به سرور استفاده می‌شود را نیز اضافه خواهد کرد. این اطلاعات، شامل ID کنترلی که رویداد را موجب می‌شود به همراه اطلاعات اضافی مربوطه می‌باشد.

این فیلدها بصورت پیش فرض خالی هستند:

```
<input type="hidden" name="__EVENTTARGET" ID="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" ID="__EVENTARGUMENT" value="" />
```

متدهای doPostBack مسئول تعیین این مقادیر با اطلاعات مناسب درباره رویداد و ارسال فرم می‌باشد. این متدهای بصورت خودکار توسط ASP.NET برای هر کنترل موجود در صفحه که از automatic postback استفاده می‌کند ایجاد می‌شود.

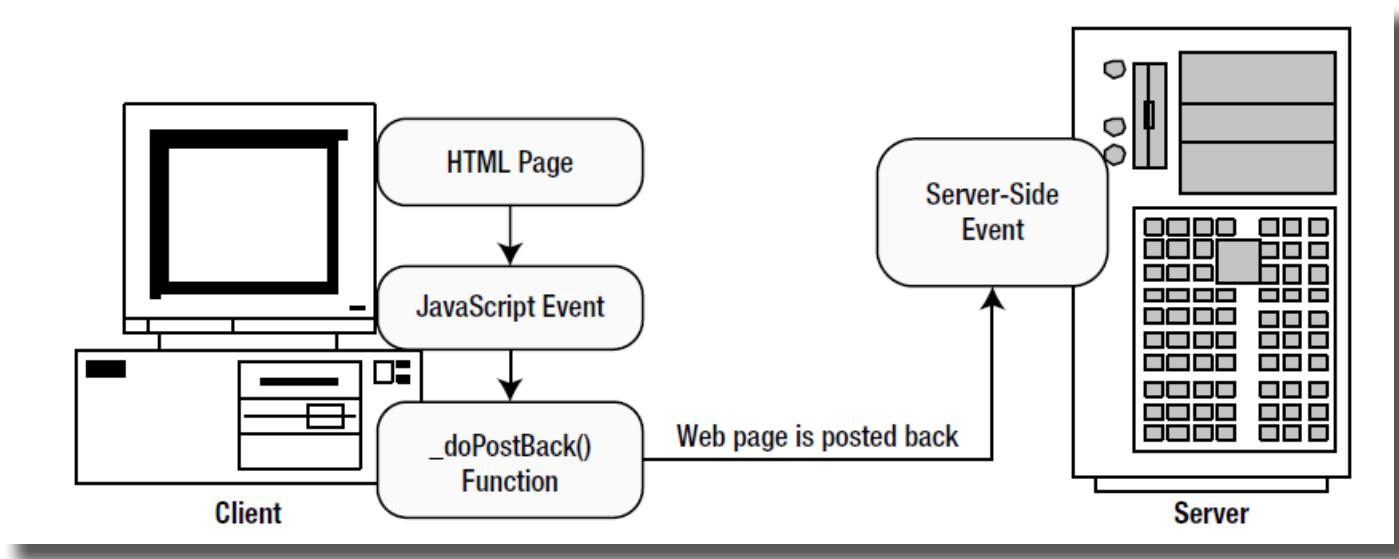
```
<script language="text/javascript">
function __doPostBack(eventTarget, eventArgument) {
    if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
        theForm.__EVENTTARGET.value = eventTarget;
        theForm.__EVENTARGUMENT.value = eventArgument;
        theForm.submit();
    }
    ...
}
</script>
```

در نهایت هر کنترلی که ویژگی AutoPostBack آن با True تعیین شده باشد با استفاده از خصیصه onclick یا onchange به متدهای doPostBack() متصل می‌شود. این خصیصه‌ها تعیین می‌کنند مرورگر چه عملی را باید در پاسخ به رویدادهای جاوا اسکریپتی سمت کلاینت (onclick یا onchange) انجام دهد.

کد زیر تگ listcontrol با نام lstBackColor را نشان می‌دهد که بصورت خودکار postback می‌شود. هرگاه که کاربر آیتم انتخابی در لیست را تغییر دهد، رویداد سمت کلاینت onchange (fire) می‌شود. سپس مرورگر متده \_doPostBack() را که صفحه را به سرور ارسال می‌کند را فراخوانی می‌کند.

```
<select ID="lstBackColor" onchange="__doPostBack('lstBackColor','")">
language="javascript">
```

به عبارت دیگر، ASP.NET با استفاده از تابع \_doPostBack، رویداد کلاینتی را به رویداد سروری تبدیل می‌کند.



## چرخه حیات صفحه (Page Lif Cycle)

برای درک چگونگی عملکرد رویدادها، باید درک درستی از چرخه حیات صفحه داشته باشید. در نظر بگیرید چه اتفاقی می‌افتد وقتی یک کاربر کنترلی را که ویژگی AutoPostBack آن با True مقداردهی شده است را تغییر می‌دهد :

- در سمت کلاینت تابع \_doPostBack فراخوانی می‌شود و صفحه مجدداً به سرور ارسال می‌شود.
- صفحه را مجدداً با فایل .aspx می‌سازد.
- اطلاعات state view را از فیلد hidden دریافت می‌کند و کنترل‌ها را بروز می‌کند.
- رویداد Page.Load اجرا (fire) می‌شود.
- تغییرات مناسب رویداد fire می‌شوند.
- رویداد Page.PreRender رخ می‌دهد و صفحه رندر می‌شود. (از مجموعه ای از اشیا به صفحه HTML تغییر می‌کند)
- در نهایت رویداد Page.Unload رخ می‌دهد.
- صفحه جدید به کلاینت ارسال می‌شود.

Event Tracker - Windows Internet Explorer

http://localhost:62605/EventTracker.aspx

Favorites Event Tracker

### Controls being monitored for change events:

New text

### List of events:

```
<< Page_Load >>
Page_PreRender
<< Page_Load >>
txt Changed
Page_PreRender
<< Page_Load >>
chk Changed
Page_PreRender
```

Internet | Protected Mode: On

100%

```
<%@ page language="#c#" autoeventwireup="true" codefile="EventTracker.aspx.cs"
inherits="EventTracker" %>

<!DOCTYPE html>

<html>
  <head id="Head1" runat="server">
    <title>Event Tracker</title>
  </head>
  <body>
    <form id="Form1" runat="server">
      <div>
        <h1>
          Controls being monitored for change events:</h1>
        <asp:TextBox ID="txt" runat="server" AutoPostBack="true" OnTextChanged="CtrlChanged"
        />
        <br />
        <br />
        <asp:CheckBox ID="chk" runat="server" AutoPostBack="true"
        OnCheckedChanged="CtrlChanged" />
        <br />
        <br />
        <asp:RadioButton ID="opt1" runat="server" GroupName="Sample" AutoPostBack="True"
        OnCheckedChanged="CtrlChanged" />
        <asp:RadioButton ID="opt2" runat="server" GroupName="Sample" AutoPostBack="True"
        OnCheckedChanged="CtrlChanged" />
        <h1>
          List of events:</h1>
        <asp:ListBox ID="lstEvents" runat="server" Width="355px" Height="150px" />
        <br />
        <br />
        <br />
        <br />
      </div>
    </form>
  </body>
</html>
```

```

public partial class EventTracker : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Log("<< Page_Load >>");
    }

    protected void Page_PreRender(object sender, EventArgs e)
    {
        // When the Page.PreRender event occurs, it is too late
        // to change the list.

        Log("Page_PreRender");
    }

    protected void CtrlChanged(Object sender, EventArgs e)
    {
        // Find the control ID of the sender.

        // This requires converting the Object type into a Control class.

        string ctrlName = ((Control)sender).ID;
        Log(ctrlName + " Changed");
    }

    private void Log(string entry)
    {
        lstEvents.Items.Add(entry);

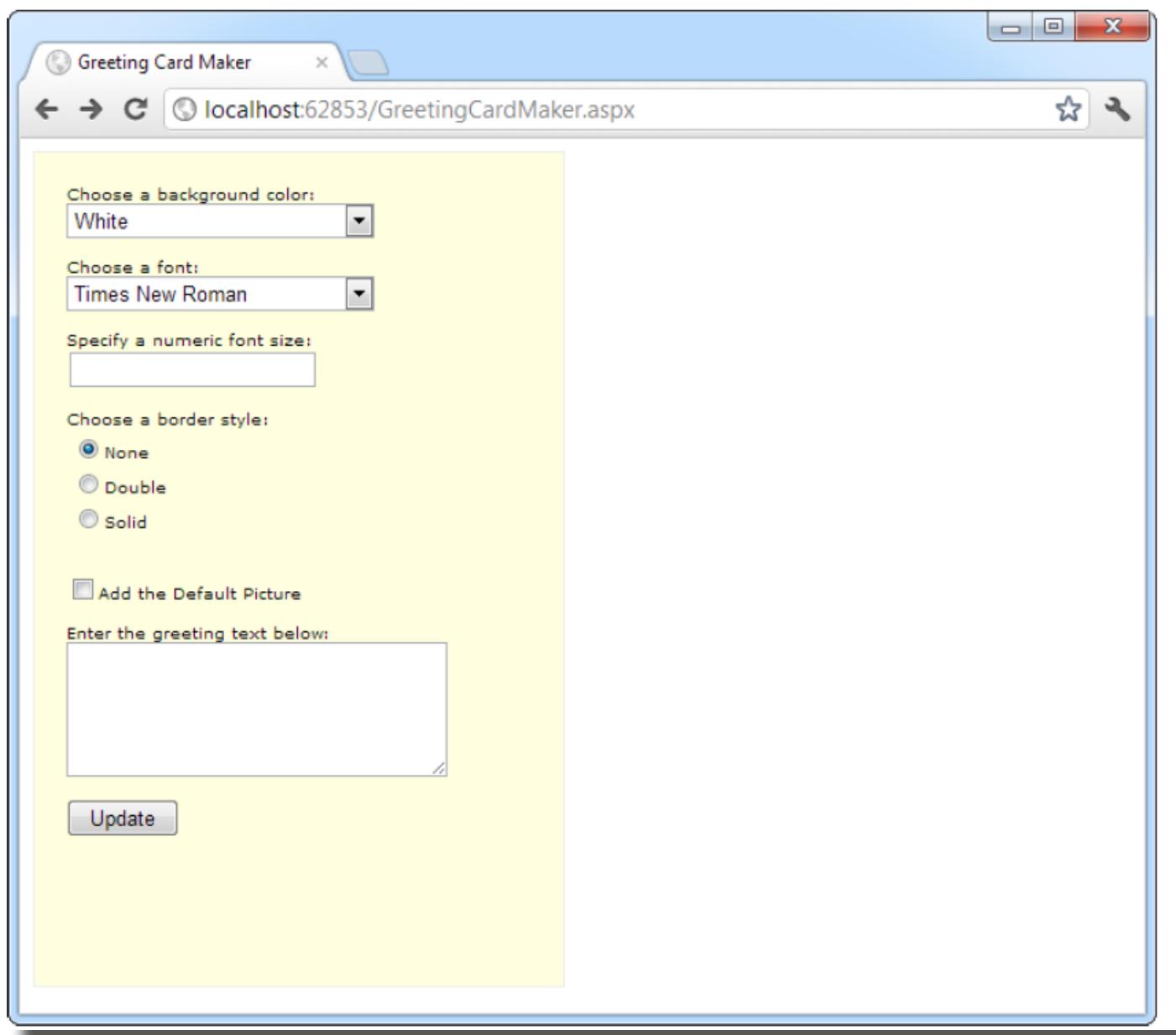
        // Select the last item to scroll the list so the most recent
        // entries are visible.

        lstEvents.SelectedIndex = lstEvents.Items.Count - 1;
    }
}

```

## صفحه وب تعاملی

یک مثال از e-card generator را می خواهیم با هم بررسی کنیم. صفحه وب به دو بخش تقسیم می شود. در سمت چپ، تگ <div> که شامل مجموعه ای از کنترل های وب برای تعیین گرینه های کارت می باشد، وجود دارد. در سمت راست یک کنترل پنل (با نام pnlCard) وجود دارد که شامل دو کنترل دیگر (lblGreeting و imgDefault) می باشد که برای نمایش متن قابل تعیین توسط کاربر و یک تصویر می باشد. این متن و تصویر، یک کارت پستال را ارائه دهد.



 نکته: المان <div>, زمانی مفید است که بخواهید متن و کنترل‌ها را گروه بندی کنید و مجموعه‌ای از ویژگی‌های فرمت بندی را بکار ببرید. المان <div> همچنین یک ابزار خاص برای قرار دادن بلوک‌هایی از محتوا در صفحه می‌باشد.

هرگاه روی دکمه Update کلیک کنید، صفحه postback می‌شود و کارت بروز می‌شود.



```

<%@ page language="#c#" autoeventwireup="true" codefile="GreetingCardMaker.aspx.cs"
    inherits="GreetingCardMaker" %>

<!DOCTYPE html>
<html>
<head id="Head1" runat="server">
    <title>Greeting Card Maker</title>
</head>
<body>
    <form id="Form1" runat="server">
        <div>
            <!-- Here are the controls: -->
            Choose a background color:<br />
            <asp:DropDownList ID="lstBackColor" runat="server" Width="194px" Height="22px" />

```

```

<br />
<br />

Choose a font:<br />

<asp:DropDownList ID="lstFontName" runat="server" Width="194px" Height="22px" />
<br />
<br />

Specify a numeric font size:<br />

<asp:TextBox ID="txtFontSize" runat="server" /><br />
<br />

Choose a border style:<br />

<asp:RadioButtonList ID="lstBorder" runat="server" Width="177px" Height="59px" />
<br />
<br />

<asp:CheckBox ID="chkPicture" runat="server" Text="Add the Default Picture' ''></
asp:CheckBox><br />
<br />

Enter the greeting text below:<br />

<asp:TextBox ID="txtGreeting" runat="server" Width="240px" Height="85px"
TextMode="MultiLine" /><br />
<br />

<asp:Button ID="cmdUpdate" OnClick="cmdUpdate_Click" runat="server" Width="71px"
Height="24px" Text="Update" />

</div>

<!-- Here is the card: -->

<asp:Panel ID=" pnlCard" runat="server" Width="339px" Height="481px"
HorizontalAlign="Center">

    Style="position: absolute; top: 16px; left: 313px;">
    <br />
    &nbsp;
    <asp:Label ID="lblGreeting" runat="server" Width="256px" Height="150px" /><br />
    <br />
    <br />
    <asp:Image ID="imgDefault" runat="server" Width="212px" Height="160px" />
</asp:Panel>
</form>
</body>
</html>

```

```

using System.Drawing;

public partial class GreetingCardMaker : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            // Set color options.

            lstBackColor.Items.Add("White");
            lstBackColor.Items.Add("Red");
            lstBackColor.Items.Add("Green");
            lstBackColor.Items.Add("Blue");
            lstBackColor.Items.Add("Yellow");

            // Set font options.

            lstFontName.Items.Add("Times New Roman");
            lstFontName.Items.Add("Arial");
            lstFontName.Items.Add("Verdana");
            lstFontName.Items.Add("Tahoma");

            // Set border style options by adding a series of
            // ListItem objects.

            ListItem item = new ListItem();
            // The item text indicates the name of the option.
            item.Text = BorderStyle.None.ToString();
            // The item value records the corresponding integer
            // from the enumeration. To obtain this value, you
            // must cast the enumeration value to an integer,
            // and then convert the number to a string so it
            // can be placed in the HTML page.
            item.Value = ((int)BorderStyle.None).ToString();
            // Add the item.

            lstBorder.Items.Add(item);
            // Now repeat the process for two other border styles.

            item = new ListItem();
        }
    }
}

```

```

        item.Text = BorderStyle.Double.ToString();

        item.Value = ((int)BorderStyle.Double).ToString();

        lstBorder.Items.Add(item);

        item = new ListItem();

        item.Text = BorderStyle.Solid.ToString();

        item.Value = ((int)BorderStyle.Solid).ToString();

        lstBorder.Items.Add(item);

        // Select the first border option.

        lstBorder.SelectedIndex = 0;

        // Set the picture.

        imgDefault.ImageUrl = "defaultpic.png";

    }

}

protected void cmdUpdate_Click(object sender, EventArgs e)
{
    // Update the color.

    pnlCard.BackColor = Color.FromName(lstBackColor.SelectedItem.Text);

    // Update the font.

    lblGreeting.Font.Name = lstFontName.SelectedItem.Text;

    if (Int32.Parse(txtFontSize.Text) > 0)
    {
        lblGreeting.Font.Size =
            FontUnit.Point(Int32.Parse(txtFontSize.Text));
    }

    // Update the border style. This requires two conversion steps.

    // First, the value of the list item is converted from a string
    // into an integer. Next, the integer is converted to a value in
    // the BorderStyle enumeration.

    int borderValue = Int32.Parse(lstBorder.SelectedItem.Value);

    pnlCard.BorderStyle = (BorderStyle)borderValue;

    // Update the picture.

    if (chkPicture.Checked)
    {
        imgDefault.Visible = true;
    }
}

```

```
else
{
    imgDefault.Visible = false;
}

// Set the text.

lblGreeting.Text = txtGreeting.Text;

}
}
```

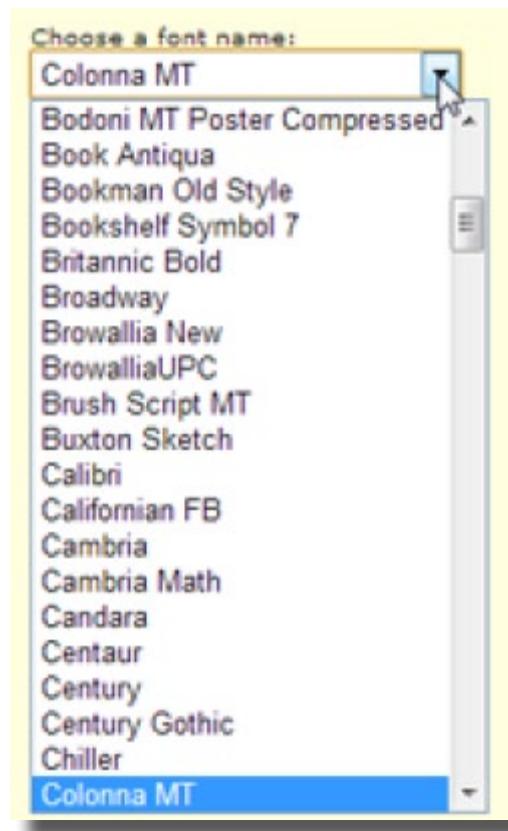
## توسعه Greeting Card Generator (ایجاد کننده کارت پستال)

با استفاده از کلاس InstalledFontCollection، می‌توانید کنترل lstFontName را با یک بیست از فونت‌ها پر کنید. از فضای نام System.Drawing.Text استفاده کنید:

```
using System.Drawing.Text;
```

با استفاده از کد زیر می‌توانید لیست را پر کنید:

```
InstalledFontCollection fonts = new InstalledFontCollection();
foreach (FontFamily family in fonts.Families)
{
    lstFontName.Items.Add(family.Name);
}
```



### ایجاد خودکار کارت‌ها

آخرین مرحله، استفاده از رویدادهای خودکار postback برای بروز رسانی اتوماتیک کارت‌ها در هر زمان که یک گزینه تغییر می‌کند می‌باشد. ویژگی AutoPostBack را با True مقدار دهی کنید.

Choose a background color:<br />

```
<asp:DropDownList ID="lstBackColor" AutoPostBack="True" runat="server" Width="194px"
    Height="22px" />
```

سپس تگهای کنترل را تغییر دهید و رویدادهای هر inputcontrol را به متدهای هندر مربوطه متصل نمایید.

Choose a background color:<br />

```
<asp:DropDownList ID="lstBackColor" AutoPostBack="True" runat="server" Width="194px"
    OnSelectedIndexChanged="ControlChanged" Height="22px" />
```

سپس متدهای هندر را بنویسید :

```
protected void ControlChanged(object sender, System.EventArgs e)
{
    // Refresh the greeting card (because a control was changed).
    UpdateCard();
}

protected void cmdUpdate_Click(object sender, EventArgs e)
{
    // Refresh the greeting card (because the button was clicked).
    UpdateCard();
}

private void UpdateCard()
{
    // (The code that draws the greeting card goes here.)
}
```



کنترل‌های مربوط به رنگ پس زمینه را نیز باید پر کرد که در اینجا توضیح نداده ایم.

# ASP.NET : توسعه برنامه های بخش اول

فصل اول آشنایی با ~~Visual Studio~~ فصل اول آشنایی با

فصل دوم: اصول فرم های وب فصل دوم: اصول فرم های وب

فصل سوم: کنترل های وب فصل سوم: کنترل های وب

Error Handling , Logging, Tracing فصل چهارم

فصل پنجم State Management فصل پنجم State Management



## Tracing و Error Handling, Logging

### مدیرت خطاهای، ثبت عملکرد و ردیابی اجرای برنامه

هیچ برنامه‌ای نمی‌تواند بدون خطا اجرا شود و برنامه exception برای آن ASP.NET ای نداشته باشد. در این فصل شما یاد خواهید گرفت چگونه برنامه را در برابر خطاهای رایج با استفاده از exception handling محافظت نمایید. همچنین نگاهی به page tracing که به شما اجازه مشاهده اطلاعات تشخیص خطای خواهیم انداشت.

### پیشگیری از خطاهای رایج

خطاهای می‌توانند در موقعیت‌های مختلفی رخ دهند. یک از رایج ترین آنها divide by zero (معمولًاً توسط ورودی نادرست و اطلاعات غلط رخ می‌دهد) و تلاش برای دسترسی به منابع محدود مانند یک فایل یا یک بانک اطلاعاتی (که اگر فایل موجود نباشد یا اتصال به بانک اطلاعاتی time out شود رخ می‌دهد) می‌باشد.

یکی از بدترین انواع خطاهای null reference exception می‌باشد که معمولًاً زمانی رخ می‌دهد که یک برنامه برای استفاده از یک شی که مقدار دهی اولیه نشده است تلاش می‌کند. نمونه کد زیر یک مشکل را با دو شی SqlConnection در عمل نشان می‌دهد.

```
// Define a variable named conOne and create the object.

private SqlConnection conOne = new SqlConnection();

// Define a variable named conTwo, but don't create the object.

private SqlConnection conTwo;

public void cmdDoSomething_Click(object sender, EventArgs e)
{
    // This works, because the object has been created
    // with the new keyword.

    conOne.ConnectionString = "...";

    ...

    // The following statement will fail and generate a
    // null reference exception.

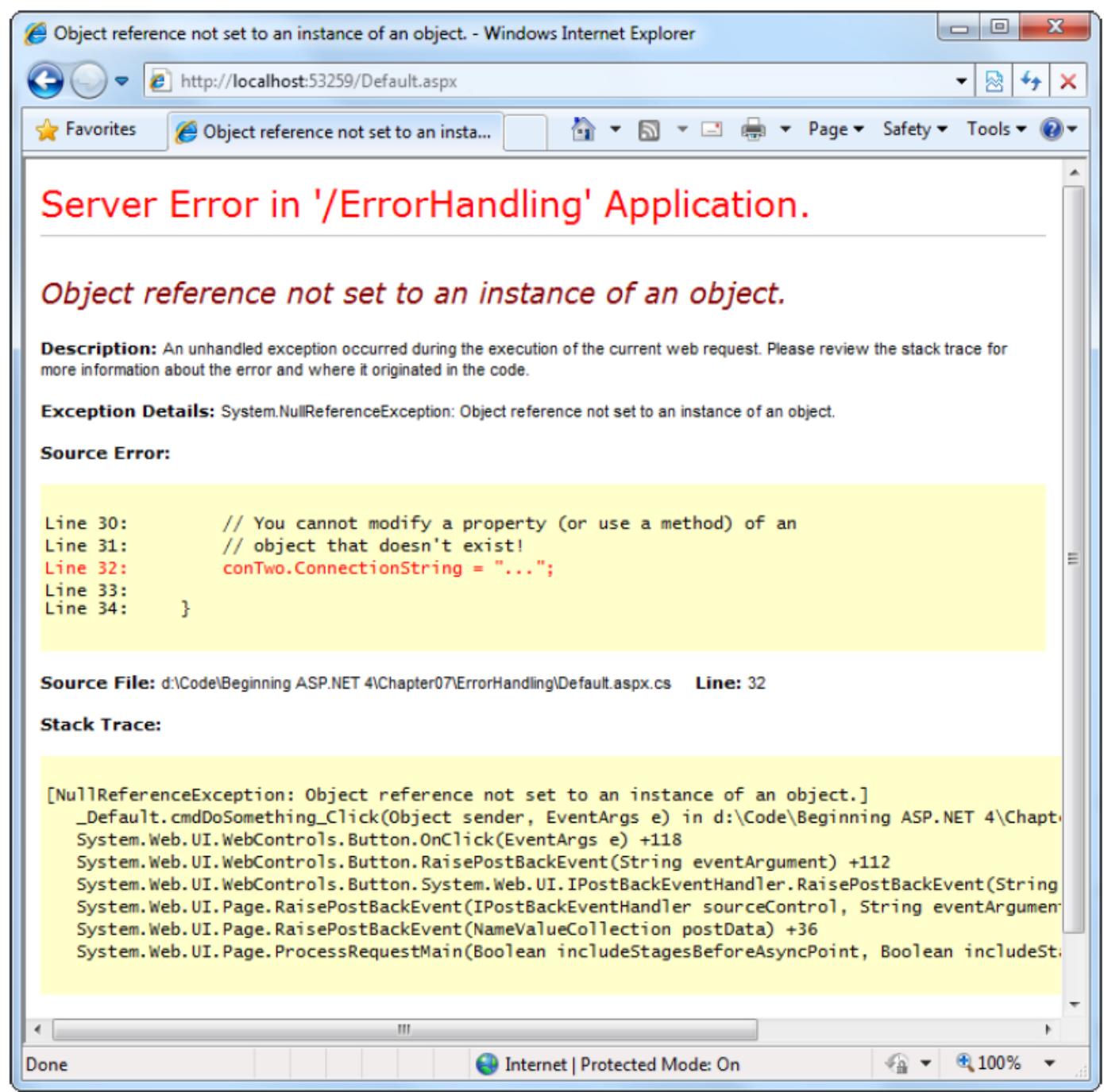
    // You cannot modify a property (or use a method) of an
    // object that doesn't exist!

    conTwo.ConnectionString = "...";

    ...
}
```

زمانی که یک خطا در کد شما رخ می‌دهد، دات نت چک می‌کند که آیا هیچ error handler ای در scope کنونی که خطا در آن رخ داده است وجود دارد یا نه. اگر خطا درون یک متود رخ داده باشد، دات نت به دنبال local error handler می‌گردد و سپس به جستجوی هر error handler ای در کد فراخوانی کننده متود خواهد گشت. اگر هیچ error handler ای پیدا نکند، پردازش صفحه متوقف می‌گردد. اگر برنامه را در VS اجرا کرده باشید، وارد مد debug خواهید شد. اگر برنامه را خارج از VS اجرا کرده باشید، یا یک صفحه خطا به شما نشان می‌دهد یا یک پیغام (بر اساس تنظیماتی که انجام داده اید) به شما نمایش خواهد داد.

اگر وارد مد debug شدید، می‌توانید مجدداً روی دکمه start کلیک کنید (یا F5 را فشار دهید) تا برنامه ادامه پیدا کند و صفحه جزئیات خطا به شما نمایش داده شود. البته این صفحه پس از deploy کردن برنامه با صفحه خطای عمومی‌تری که می‌توانید آن را در IIS پیکربندی کنید جایگزین خواهد شد.



## آشنایی با Exception Handling

با رخدادن یک خطا، فریمورک دات نت یک شی خطا که مشکل را ارائه می‌کند، ایجاد خواهد کرد. شما می‌توانید این شی را با استفاده از یک exception handler بگیرید (catch). اگر این کار را بکنید، می‌توانید به کاربر اطلاع دهید که خطا را رفع کند یا آن را نادیده بگیرد و اجازه دهد برنامه به اجرای خود ادامه بدهد.

**هندل کردن خطا بصورت ساخت یافته (Structured Exception Handling)** دارای چند ویژگی کلیدی می‌باشد :

- خطاها مبتنی بر شی هستند: هر خطا، اطلاعات تشخیص خطا را نمایش می‌دهد که درون یک شی پیچیده (Wrapped) شده است. این اشیای خطا یک ویژگی با نام InnerException نیز ارائه می‌دهند که به شما اجازه Wrap کردن یک خطا عمومی بر روی یک خطا خاص‌تر که باعث آن شده است را می‌دهد. حتی می‌توانید اشیای خطا خود را ایجاد و throw کنید.
- از یک ساختار بلوک بندی جدید استفاده می‌کند: این ویژگی، فعال و غیر فعال نمودن error handler های مختلف را برای بخش‌های مختلف کد و هندل کردن خطاهای آن‌ها را بصورت مستقل امکان پذیر می‌سازد.

 نکته: Exceptio Handler ها، یک تکنیک کلیدی برنامه نویسی هستند. آنها به ما اجازه عکس العمل در برابر مشکلاتی که در زمان اجرا رخ می‌دهد را خواهند داد. به هر حال شما قطعاً نباید از آنها برای مخفی کردن bug هایی که ممکن است محصول خود نرم افزار شما باشد استفاده نمایید! در عوض باید خطاهای برنامه نویسی را در زمان نوشتن برنامه کاهش داده و آنها را اصلاح نمایید. ویژگی موجود در VS به شما در انجام این کار کمک خواهد کرد.

## کلاس Exception

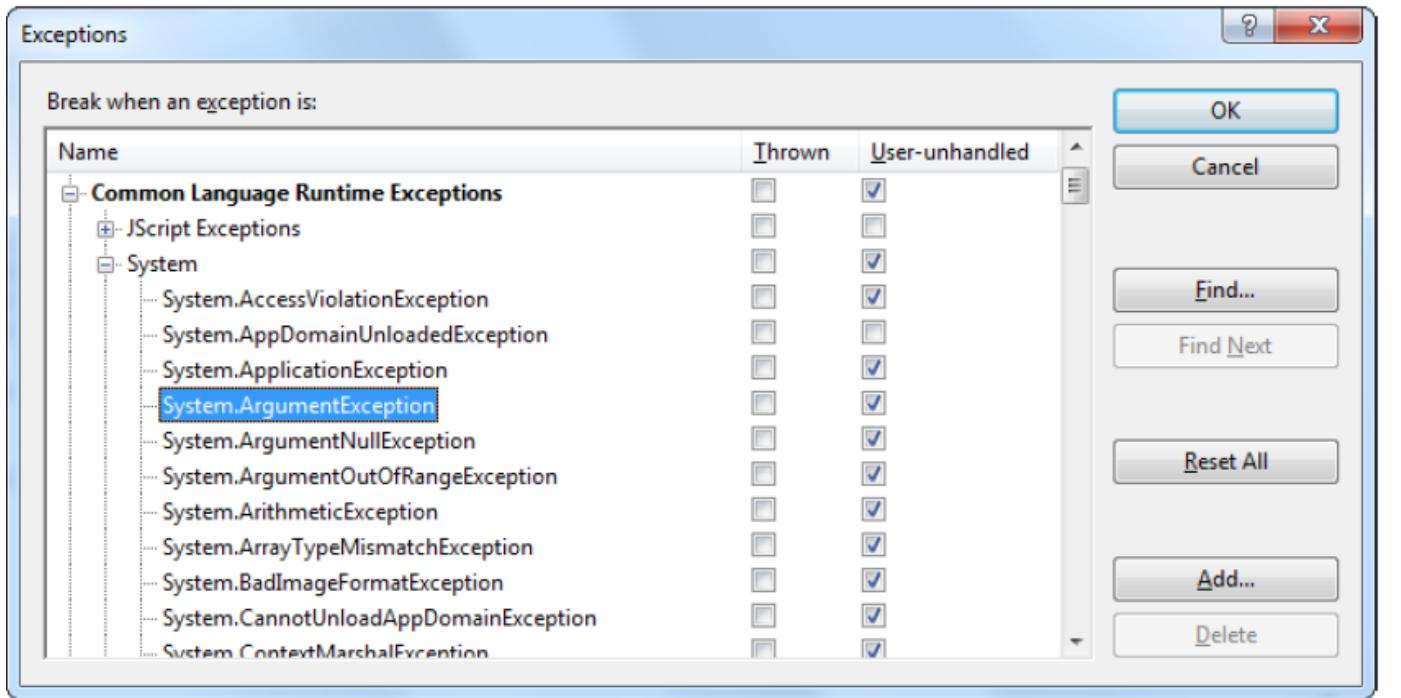
هر کلاس exception از کلاس پایه System.Exception ارث بری می‌کند. فریمورک دات نت پر از کلاس‌های exception مانند SQLException، IOException، NullReferenceException و... می‌باشد. این کلاس شامل عملکردهای خاص برای تعیین نوع خطا می‌باشد.



شرح	عضو
یک لینک به سند help که می‌تواند URL باشد. دات نت از ویژگی استفاده نمی‌کند ولی شما می‌توانید از آن در خطاهای سفارشی خودتان استفاده نمایید.	HelpLink
یک خطا داخلی می‌باشد. برای نمونه، یک متدهای ساده input/output catch و یک خطا سطح بالاتر "Operation Failed" ایجاد نماید. جزئیات خطا عمومی در ویژگی InnerException آن موجود می‌باشد.	InnerException
یک متن توصیفی برای شرح دادن خطا	Message
نام برنامه یا شی است که باعث رخداد خطا شده است.	Source

رشته string که شامل لیستی از متدهای فراخوانی شده در stack به ترتیب جدیدترین به قدیمی‌تر می‌باشد.	StackTrace
یک شی Reflection (یک نمونه از کلاس System.Reflection.MethodBase) که بعضی از اطلاعات درباره متده است را ارائه می‌دهد. این اطلاعات شامل جزئیات عمومی متده مانند نام آن، نوع داده ای پارامترهای آن و مقدار برگشتی توسط آن می‌باشد.	TargetSite
متده مورد استفاده در خطاهای تودرتو که بیش از یک لایه دارد. این متده خطای اصلی را توسط حرکت به پایه زنجیره InnerException بازیابی می‌کند.	GetBaseException()

ابزاری مفید برای جستجوی خطای در کتابخانه کلاس دات نت دارد. گزینه Debug> Exceptions را از منو انتخاب نمایید. آیتم Common Language Runtime Exceptions را باز کنید تا درخت سلسله مراتب خطاهای دات نت را بر اساس فضای نام آنها مشاهده کنید.



این قادر به شما کمک می‌کند تا تعیین کنید چه خطایی باید توسط کد شما هندل شود و چه خطایی VS را وارد به وارد شدن به مد خواهد کرد. بنابراین تنها خطاهایی که در این قادر تعیین کرده اید باعث توقف اجرای برنامه توسط VS می‌شود و خطاهای دیگر باید توسط کد شما هندل شود.

## هنдел کردن خطاهای

اولین مرحله از کنترل خطای چک کردن شروط خطای بالقوه قبل از اجرای یک عملیات خاص می‌باشد. برای نمونه برنامه می‌تواند بصورت واضح چک کند که آیا متغیر divisor قبل از اجرای محاسبات یا قبل از باز کردن یک فایل موجود، مقدار صفر را دارد یا نه.

```

if (divisor != 0)
{
    // It's safe to divide some number by divisor.
}

if (System.IO.File.Exists("myfile.txt"))
{
    // You can now open the myfile.txt file.

    // However, you should still use exception handling because a variety of
    // problems can intervene (insufficient rights, hardware failure, etc.).
}

```

با وجود همه این چک‌ها باز نمی‌توانید مطمئن باشید که مثلاً برنامه شما در برابر تمامی خطاهای ممکن (شامل خطای سخت افزار، خطای شبکه و ...)، محافظت شده است. همچنین شما هیچ راهی برای اعتبار سنجی کلمه کاربری و عبور برای اتصال به بانک اطلاعاتی را قبل از تلاش برای باز کردن یک Connection نخواهید داشت. بنابراین باید خطاهای تشخیص داده و در زمان رخ دادن با آنها روبرو شوید.

راه حل موجود، استفاده از روش Structured Exception Handling می‌باشد که می‌تواند خطاهای بالقوه را در بلوك خاصی از کد Wrap کند.

```

try
{
    // کدهای ریسک پذیر در اینجا قرار م یگیرند. مانند بازگردان یک فایل. اتصال به بانک اطلاعاتی و غیره

}

catch (Exception)
{
    // خطأ در اینجا تشخیص داده شده و شما با آن روبرو می شوید

}

finally
{
    // صرف نظر از اینکه آیا خطای رخ داده است یا نه کدهای این بخش اجرا می شوند

}

```

دستور Try هندل کردن خطا را ممکن می‌سازد. خطاهای رخ داده در این بخش می‌توانند پیگیری شوند. کدهای موجود در بلوک Catch در زمان تشخیص رخ دادن یک خطا اجرا خواهد شد. در نهایت بلوک Finally می‌باشد که تحت هر شرایطی اجرا خواهد شد و برای تمیز کردن کدهای شما مانند بستن اتصال بانک اطلاعاتی و ... استفاده می‌شود. این بخش از این نظر مهم است که در زمانی که خطای رخ داده است و برنامه امکان ادامه اجرا ندارد، شما هنوز فرصتی برای آزاد سازی منابع را خواهید داشت.

## گرفتن خطاهای خاص

```

try
{
    // Risky database code goes here.

}

catch (System.Data.SqlClient.SqlException err)
{
    // Catches common problems like connection errors.

}

catch (System.NullReferenceException err)
{
    // Catches problems resulting from an uninitialized object.

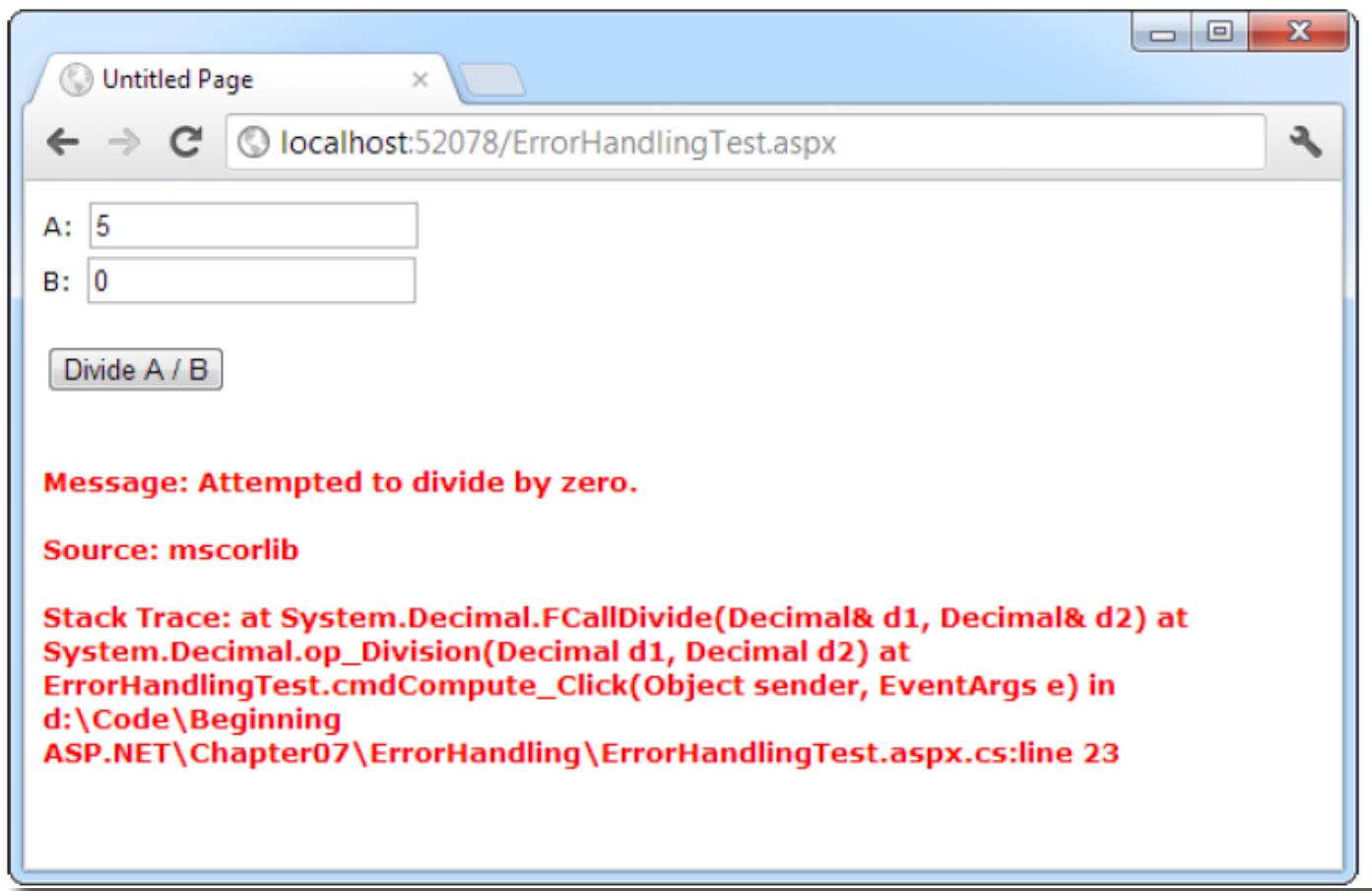
}

catch (System.Exception err)
{
    // Catches any other errors.
}

```

## در عمل Exception Handling

در اینجا یک مثال داریم که برنامه ای دو مقدار ورودی دریافت می‌کند و آنها را برهم تقسیم می‌کند. سپس کلیه خطاهای مرتبط با آنها نمایش داده خواهد شد.



البته قطعاً با استفاده از Validation Control یا Code-Safety Checks می‌توانید از رخ دادن این خطاهای جلوگیری کنید.

```
public partial class ErrorHandlingTest : System.Web.UI.Page
{
    protected void cmdCompute_Click(Object sender, EventArgs e)
    {
        try
        {
            decimal a, b, result;
            a = Decimal.Parse(txtA.Text);
            b = Decimal.Parse(txtB.Text);
            result = a / b;
            lblResult.Text = result.ToString();
            lblResult.ForeColor = System.Drawing.Color.Black;
        }
        catch (Exception err)
```

```

    {
        lblResult.Text = "<b>Message:</b> " + err.Message;
        lblResult.Text += "<br /><br />";
        lblResult.Text += "<b>Source:</b> " + err.Source;
        lblResult.Text += "<br /><br />";
        lblResult.Text += "<b>Stack Trace:</b> " + err.StackTrace;
        lblResult.ForeColor = System.Drawing.Color.Red;
    }
}
}

```

## Throw خطاهاي سفارشی خودتان

شما همچنین می‌توانید اشیای خطای خودتان را برای شرط‌های سفارشی سازی شده تعریف نمایید. همه آن چیزی که شما نیاز دارید، ایجاد یک نمونه از خطای مناسب و سپس استفاده از دستور throw می‌باشد.

 مثال زیر، متد DivideNumbers() را معرفی می‌کند که چک می‌کند آیا متغیر divisor صفر می‌باشد یا نه و سپس بصورت دستی یک نمونه از کلاس DivideByZeroException را ایجاد و throw خواهد کرد.

```

protected void Page_Load(Object sender, EventArgs e)
{
    try
    {
        DivideNumbers(5, 0);
    }
    catch (DivideByZeroException err)
    {
        // Report error here.
    }
}

private decimal DivideNumbers(decimal number, decimal divisor)
{
    if (divisor == 0)

```

```

{
    DivideByZeroException err = new DivideByZeroException();

    throw err;
}

else
{
    return number / divisor;
}

}

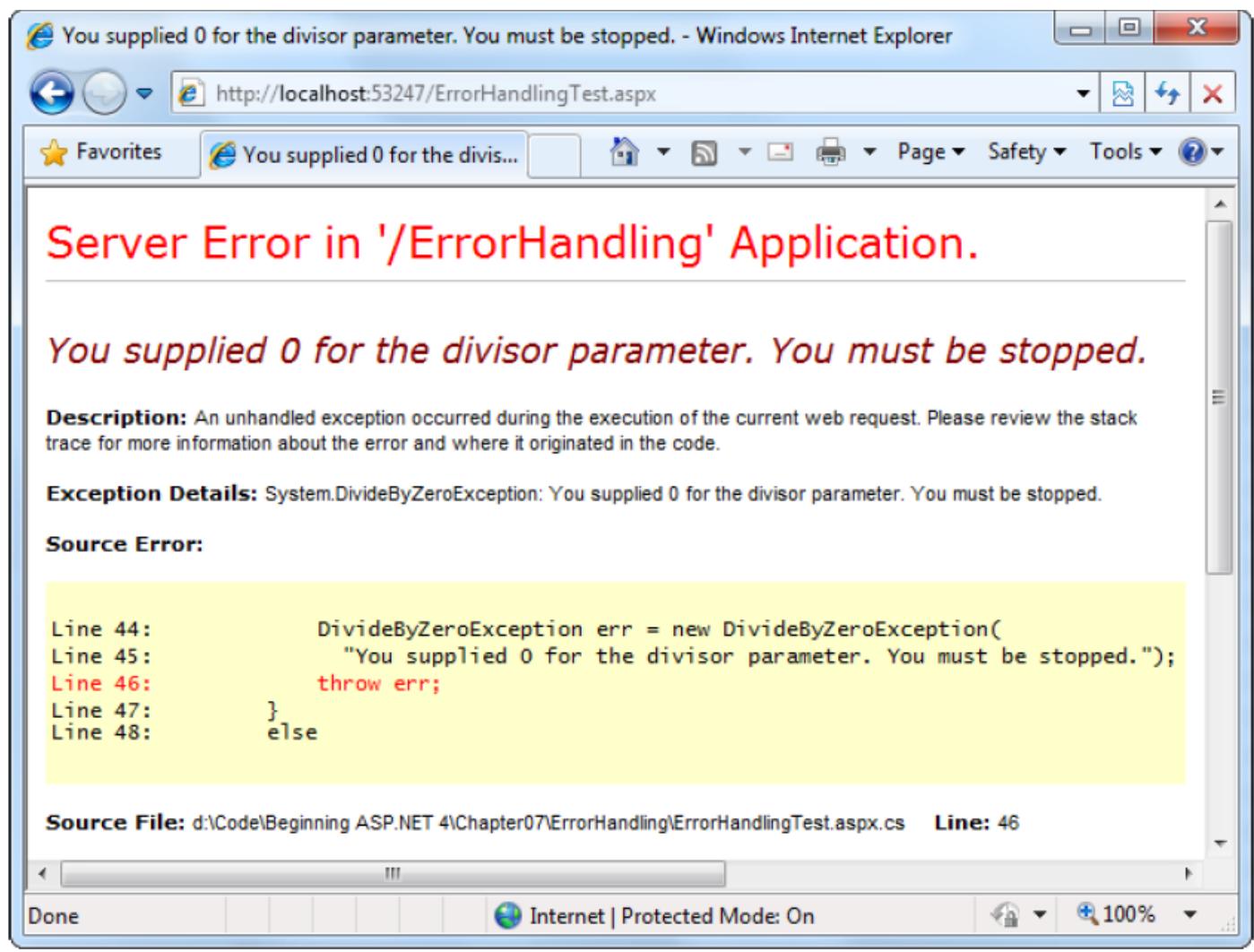
```

همچنین می‌توانید یک خطای دات نت ایجاد و با استفاده از constructor های دیگر آن، یک پیام سفارشی برای آن تعیین کنید.

```

private decimal DivideNumbers(decimal number, decimal divisor)
{
    if (divisor == 0)
    {
        DivideByZeroException err = new DivideByZeroException(
            "You supplied 0 for the divisor parameter. You must be stopped.");
        throw err;
    }
    else
    {
        return number / divisor;
    }
}

```



Throw کردن خطا، بیشتر در برنامه نویسی component-base (مثلاً در معماری لایه بندی یا ...) مفید خواهد بود. در این نوع از برنامه نویسی، صفحه ASP.NET شما، در حال ایجاد اشیا و فراخوانی متدها از کلاس تعریف شده در یک اسembly کامپایل شده جدآگانه می باشد.

کلاس موجود در کامپوننت، باید قادر به اطلاع رسانی به کد فراخوانی کننده خود (برنامه تحت وب) درباره ه خطایی که رخ داده باشد.

اگر نیاز به برگرداندن اطلاعات بیشتر یا خاص درباره خطای دارید، می توانید کلاس exception خاص خود را ایجاد نمایید.

کلاس سفارشی exception باید از کلاس System.ApplicationException ارث بری کند که خودش نیز از کلاس پایه Exception ارث بری کرده است.

زمانی که شما کلاس Exception را ایجاد کردید می توانید ویژگی هایی را برای نگهداری اطلاعات بیشتر به آن اضافه کنید.

```
public class CustomDivideByZeroException : ApplicationException
{
    // Add a variable to specify the "other" number.
    // This might help diagnose the problem.
```

```
public decimal DividingNumber;
}
```

نکته: این کلاس را هم می‌توانید در فایل aspx.cs. ایجاد کنی و هم یک فایل کلاس جدا برای آن ایجاد کنید.



این خطای سفارشی را می‌توانید مانند زیر throw کنید:

```
private decimal DivideNumbers(decimal number, decimal divisor)
{
    if (divisor == 0)
    {
        CustomDivideByZeroException err = new CustomDivideByZeroException();
        err.DividingNumber = number;
        throw err;
    }
    else
    {
        return number / divisor;
    }
}
```

برای اجرای خطای سفارشی، باید سه سازنده استاندارد برای آن ایجاد نمایید :

- بدون هیچ آرگومانی
- با یک پیام سفارشی
- با یک پیام سفارشی و یک شی exception که به عنوان inner exception استفاده می‌شود.

این سازنده‌ها نیاز قطعی به هیچ کدی مدارند. همه آن چیزی که این سازنده‌ها باید انجام دهنند، ارسال پارامترها به کلاس پایه با استفاده از کلمه کلیدی base می‌باشد.

```
public class CustomDivideByZeroException : ApplicationException
{
    // Add a variable to specify the "other" number.
    private decimal dividingNumber;
    public decimal DividingNumber
```

```

    {
        get { return dividingNumber; }
        set { dividingNumber = value; }
    }

    public CustomDivideByZeroException()
        : base()
    {
    }

    public CustomDivideByZeroException(string message)
        : base(message)
    {
    }

    public CustomDivideByZeroException(string message, Exception inner) :
        base(message, inner)
    {
    }
}

```

سازنده سوم مشخصاً برای برنامه نویسی کامپوننت‌ها مفید می‌باشد. این سازنده به شما اجازه ویژگی `InnerException` باشی که باعث رخدادن خطای اصلی می‌شود خواهد شد.

مثال بعدی نشان می‌دهد که چگونه می‌توانید این سازنده را با یک کلاس کامپوننت با نام `ArithmeticUtility` بکار ببرید:

```

public class ArithmeticUtilityException : ApplicationException
{
    public ArithmeticUtilityException()
        : base()
    {
    }

    public ArithmeticUtilityException(string message)
        : base(message)
    {
    }

    public ArithmeticUtilityException(string message, Exception inner) :
        base(message, inner)
    {
    }
}

public class ArithmeticUtility
{
    private decimal Divide(decimal number, decimal divisor)

```

```

    }

    try
    {
        return number / divisor;
    }

    catch (Exception err)
    {

        // Create an instance of the specialized exception class,
        // and place the original exception object (for example, a
        // DivideByZeroException) in the InnerException property.

        ArithmeticUtilityException errNew =
            new ArithmeticUtilityException("Calculation error", err);

        // Now throw the new exception.

        throw errNew;
    }

}

}

```

به یاد داشته باشید کلاس‌های سفارشی خطای ارتباط را استاندارد برای ارتباط یک خطاب با بخش‌های مختلفی از کد توسط یک کلاس می‌باشد.

## استفاده از Page Tracing

ابزار Debug موجود در VS و صفحات نمایش جزئیات خطای ASP.NET، زمانی مفید هستند که می‌خواهید برنامه را تست کنید. بعضی وقت‌ها نیاز دارید راهی برای تشخیص خطای انتشار نسخه نهایی Deploy (کردن برنامه و زمانی که VS نیز ندارید پیدا کنید).

با استفاده از ثبت اطلاعات تشخیص خطای event log، می‌توانید این خطاهای را تعیین کنید. ولی باید یک نفر مرتباً Log را بازبینی کند. می‌توانید بعضی از این اطلاعات را مستقیماً در صفحه وب نمایش دهید. مشکل این روش، نیاز به حذف همه این کدهای اضافی، قبل از Deploy کردن برنامه تحت وب می‌باشد. در غیر اینصورت کاربران وب سایت شما، می‌توانند پیام‌های اشکال زدایی (debugging) را مشاهده کنند.

خوب‌بختانه، یک راه ساده‌تر برای حل این مشکل وجود دارد. ASP.NET، یک امکان با نام tracing (پیگیری خطای ارائه داده است که یک راه مناسب برای گزارش اطلاعات تشخیص خطای می‌باشد.

## فعال سازی Tracing

برای استفاده از tracing، نیاز به فعال سازی آن دارد. چندین راه برای فعال کردن آن وجود دارد. راحت‌ترین راه، اضافه نمودن یک خصیصه مستقیماً به راهنمای صفحه (page directive) می‌باشد:

```
<%@ Page Trace="true" ... %>
```

همچنین می‌توانید با استفاده از شی System.Web.TraceContext Built-In Trace که می‌باشد (در فضای نام System.WebTraceContext) نیز آن را فعال نمایید.

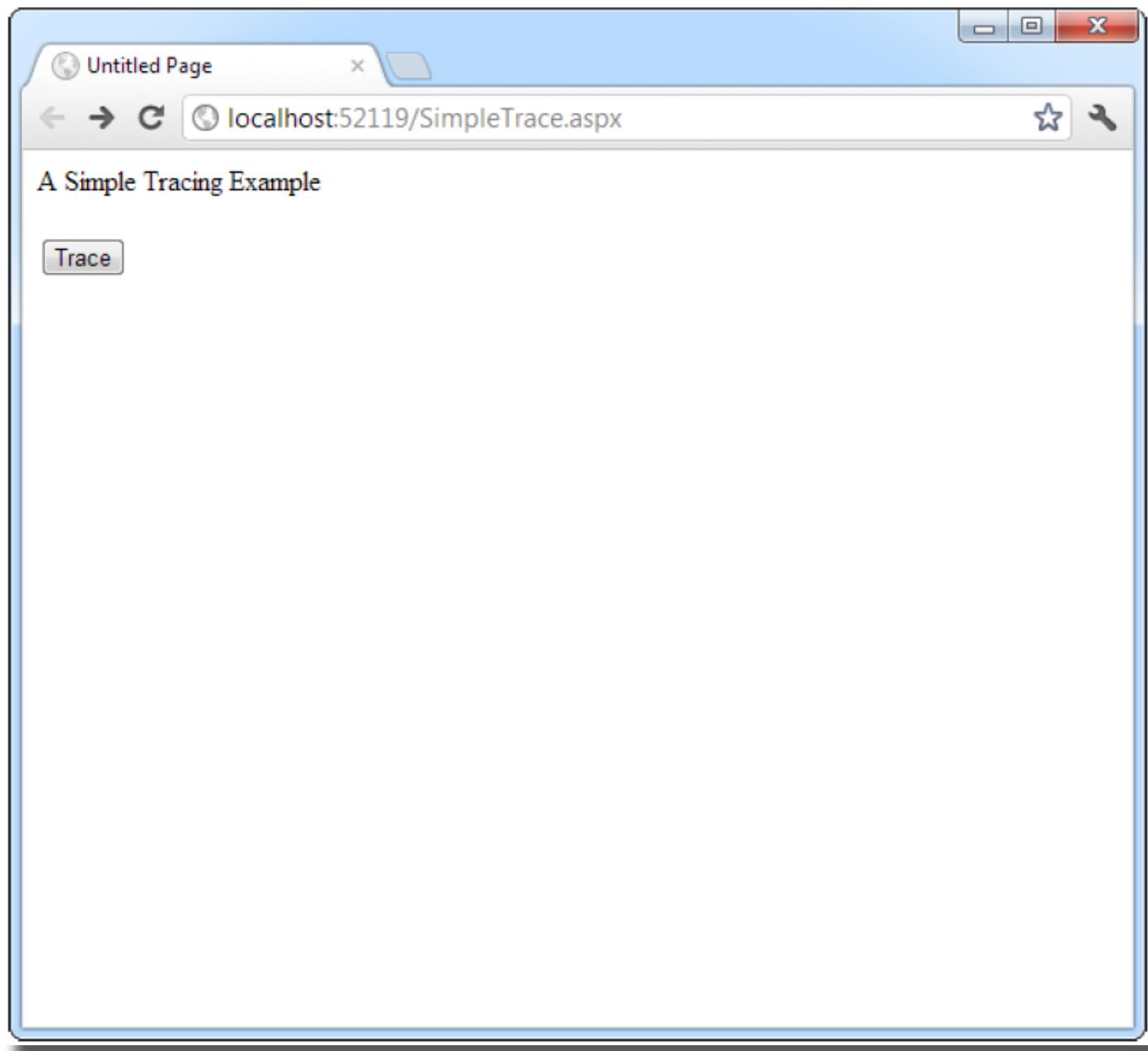
```
protected void Page_Load(Object sender, EventArgs e)
{
    Trace.IsEnabled = true;
}
```

به صورت پیش فرض پس از فعال سازی tracing، این ویژگی تنها برای درخواست‌های local بکار می‌رود. (به عبارت دیگر اگر با یک برنامه deploy شده کار می‌کنید، باید درخواست‌های خود را از یک مرورگر وب که روی کامپیوتر وب سرور می‌باشد بفرستید). این محدودیت از مشاهده اطلاعات tracing توسط کاربران نهایی جلوگیری می‌کند. برای مشاهده این اطلاعات از محل‌های دیگر باید remote tracing (ردیابی خطای راه دور) را با انجام تغییراتی در فایل web.config فعال کنید.

## اطلاعات ردیابی خطای راه دور

ASP.NET tracing بصورت خودکار مجموعه استاندارد طولانی از اطلاعات فرمت‌بندی شده ارائه می‌دهد. این اطلاعات به شما اجازه مشاهده و مانیتور کردن برنامه را از جنبه‌های مهم می‌دهد (مانند Session کنونی و زمان مورد نیاز برای اجرای بخشی از کد) در شکل زیر خواهید دید که این اطلاعات چطور نمایش داده می‌شوند.

اگر روی دکمه کلیک کنید، tracing با استفاده از Trace.IsEnabled فعال می‌شود. زمانی که صفحه مجدد رندر می‌شود شامل اطلاعات تشخیص خطای مناسب خواهد بود.



اطلاعات ردیابی در چندین گروه بندی ارائه می‌شوند. بسته به صفحه شما ممکن است همه این اطلاعات را نبینید. برای نمونه اگر درخواست صفحه هیچ پارامتر query string ای را پشتیبانی نکند شما بخش QueryString Vollection را نخواهید دید.

A Simple Tracing Example

**Trace**

### Request Details

<b>Session Id:</b>	v41000refugqeicb5zshish	<b>Request Type:</b>	POST
<b>Time of Request:</b>	6/27/2012 2:27:35 PM	<b>Status Code:</b>	200
<b>Request Encoding:</b>	Unicode (UTF-8)	<b>Response Encoding:</b>	Unicode (UTF-8)

### Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.000051	0.000051
aspx.page	Begin Init	0.000076	0.000025
aspx.page	End Init	0.000108	0.000032
aspx.page	Begin InitComplete	0.000129	0.000021
aspx.page	End InitComplete	0.000151	0.000022
aspx.page	Begin LoadState	0.000171	0.000021
aspx.page	End LoadState	0.000352	0.000180
aspx.page	Begin ProcessPostData	0.000383	0.000032
aspx.page	End ProcessPostData	0.000460	0.000076
aspx.page	Begin PreLoad	0.000483	0.000024
aspx.page	End PreLoad	0.000530	0.000047
aspx.page	Begin Load	0.000569	0.000039
aspx.page	End Load	0.000600	0.000031
aspx.page	Begin ProcessPostData Second Try	0.000623	0.000022

## درخواست جزئیات

این بخش شامل اطلاعات پایه‌ای مانند Session ID کنونی، زمان ایجاد درخواست وب سرور و نوع درخواست وب و نوع رمز کردن آنها می‌باشد (encoding).

### Request Details

<b>Session Id:</b>	v41000refugqeicb5zshish	<b>Request Type:</b>	POST
<b>Time of Request:</b>	6/27/2012 2:18:38 PM	<b>Status Code:</b>	200
<b>Request Encoding:</b>	Unicode (UTF-8)	<b>Response Encoding:</b>	Unicode (UTF-8)

## اطلاعات ردیابی

این اطلاعات، مراحل پردازش صفحه را قبل از ارسال به کلاینت نمایش می‌دهند. هر بخش اطلاعات اضافی درباره مدت زمانی که آن پردازش برای تکمیل شدن صرف کرده است را نشان می‌دهد.

### Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	End RaisePostBackEvent		
aspx.page	Begin LoadComplete	4.83301648673225E-05	0.000048
aspx.page	End LoadComplete	8.85587414042846E-05	0.000040
aspx.page	Begin PreRender	0.000119847634266366	0.000031
aspx.page	End PreRender	0.000163987322411089	0.000044
aspx.page	Begin PreRenderComplete	0.00019695240596221	0.000033
aspx.page	End PreRenderComplete	0.000227682568594612	0.000031
aspx.page	Begin SaveState	0.000686400087161916	0.000459
aspx.page	End SaveState	0.000843403281702004	0.000157
aspx.page	Begin SaveStateComplete	0.000880279476860886	0.000037
aspx.page	End SaveStateComplete	0.000917155672019768	0.000037
aspx.page	Begin Render	0.00094760646953733	0.000030
aspx.page	End Render	0.00136246366507475	0.000415

### درختواره کنترل‌ها

کنترل‌های موجود در صفحه، و کنترل‌های درون آنها نیز در این بخش نمایش داده می‌شوند. یکی از مفیدترین امکانات این بخش، نمایش ستون اندازه ViewState می‌باشد که به شما تعداد بایت‌های مورد نیاز برای نگهداری اطلاعات درون کنترل را خواهد گفت.

این کار به شما کمک می‌کند تصمیم بگیرید که آیا Control State را فعال نمایید یا نه.

### Control Tree

Control UniqueID	Type	Render Size Bytes (including children)	ViewState Size Bytes (excluding children)	ControlState Size Bytes (excluding children)
_Page	ASP.simpletrace_aspx	840	0	0
ctl02	System.Web.UI.LiteralControl	151	0	0
ctl00	System.Web.UI.HtmlControls.HtmlHead46	0	0	0
ctl01	System.Web.UI.HtmlControls.HtmlTitle	33	0	0
ctl03	System.Web.UI.LiteralControl	14	0	0
form1	System.Web.UI.HtmlControls.HtmlForm609	0	0	0
ctl04	System.Web.UI.LiteralControl	77	0	0
cmdTrace	System.Web.UI.WebControls.Button	67	0	0
ctl05	System.Web.UI.LiteralControl	12	0	0
ctl06	System.Web.UI.LiteralControl	20	0	0

## Application State و Session State

این بخش، هر آیتمی که در session state یا application state کنونی می‌باشد را نمایش می‌دهد.

### Session State

Session Key	Type	Value
TestString	System.String	This is just a string.
MyDataSet	System.Data.DataSet	System.Data.DataSet

## Response Cookies و Request Cookies

این بخش، کوکی هایی که توسط مرورگر ارسال شده اند را به همراه درخواست صفحه آن و کوکی هایی که توسط وب سرور به عنوان پاسخ برگردانده شده اند را نمایش خواهند داد.

در مثال زیر یک صفحه از کوکی با نام Preference استفاده کرده است که اطلاعات کلم کاربری را در خود نگه می‌دارد. علاوه بر آن، مرورگر وب، یک کوکی با نام ASP.NET\_SessionId را دریافت می‌کند که آن را برای نگهداری SessionID کنونی، بصورت خودکار ایجاد می‌کند.

## Headers Collection

همه HTTP header ها را نمایش می‌دهد. هدرها، اطلاعات کوچکی هستند که به عنوان بخشی از درخواست به سرور فرستاده می‌شوند. آنها شامل اطلاعاتی درباره مرورگر ایجاد کننده درخواست، انواع محتوای مورد پشتیبانی آن و زبانی که استفاده می‌کند، می‌باشند.

علاوه بر این، Response Header ها نیز، کلیه هدرهایی که به عنوان بخشی از پاسخ به کلاینت ارسال می‌شوند را در خورد لیست می‌کنند. این هدرها کوچک‌تر می‌باشند و شامل جزئیاتی درباره نسخه ASP.NET و نوع محتوایی (text/html) که به کلاینت ارسال شده است، هستند.

## Headers Collection

Name	Value
Cache-Control	max-age=0
Connection	keep-alive
Content-Length	214
Content-Type	application/x-www-form-urlencoded
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding	gzip,deflate,sdch
Accept-Language	en-GB,en-US;q=0.8,en;q=0.6
Cookie	ASP.NET_SessionId=v41000refugqeicb5zshislh;.ASPXAUTH=4D15116CAAFE32AE3F1E0517C7F50
Host	localhost:52078
Referer	http://localhost:52078/ErrorHandling/SimpleTrace.aspx
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.108
Origin	http://localhost:52078

## Response Headers Collection

Name	Value
X-AspNet-Version	4.0.30319
Cache-Control	private
Content-Type	text/html

## Form Collection

این بخش، اطلاعات Post-back (برگردانده شده از کلاینت) شده فرم را لیست می‌کند. این اطلاعات شامل، کلیه مقادیر ارسال شده توسط کنترل‌های وب مانند متن موجود در textbox و ... می‌باشد. در مثال زیر یک hiddenfiled و یک دکمه با نام cmdTrace : در صفحه وجود دارد :

## Form Collection

Name	Value
__VIEWSTATE	/wEPDwUKMTQ2OTkzNDMyMWRk6PdTEsrgNkgFP9+LyoOOXsjohnM=
cmdTrace	Trace
__EVENTVALIDATION	/wEWAgL9lt3/BQLqhI2yDuBna1E/STtAj+7NCZA4/ezpy4LZ

## Query String Collection

این بخش، کلیه متغیرها و مقادیر ارسال شده در query string (عبارت درخواستی که توسط URL بالای صفحه ارسال می‌شود) را لیست می‌کند. این اطلاعات را می‌توانید مستقیماً در URL صفحه وب (در کادر address در بالای مرورگر وب) مشاهده کنید.

مثال زیر اطلاعات یک صفحه که توسط دو query string با نام search و style درخواست شده است را نشان می‌دهد:

?search=cat&style=full

## Querystring Collection

Name	Value
search	cat
style	full

## متغیرهای سروری (Server Variables)

کلیه متغیرهای سروری به همراه محتوای آنها در این بخش لیست می‌شود. از طریق Request.ServerVariables نیز می‌توانید به آنها دسترسی داشته باشید.

## نوشتن اطلاعات ردیابی

علاوه بر اطلاعات استاندارد ردیابی، می‌توانید پیام‌های ردیابی خودتان را نیز ایجاد کنید. برای نمونه، ممکن است بخواهید، مقدار یک متغیر را در نقاط مختلف در زمان اجرا داشته باشید، تا آن را با یک مقدار معین مقایسه کنید. یا ممکن است پیامی را در هنگامی که کد به نقطه خاصی از اجرا رسید نمایش دهید.

برای نوشتن یک پیام ردیابی سفارشی، می‌توانید از متدهای Write() یا Trace() استفاده نمایید. این دو متدهای بهم برابرند. تنها تفاوت آنها این است که Write()، متن پیم را به رنگ قرمز نمایش می‌دهد که به راحت‌تر دیده شدن آن در بین سایر پیام‌ها کمک خواهد کرد.

```
protected void cmdWriteCategory_Click(Object sender, EventArgs e)
{
    Trace.Write("cmdWriteCategory_Click", "About to place an item in session state.");
    Session["Test"] = "Contents";
    Trace.Write("cmdWriteCategory_Click", "Placed item in session state.");
}
```

aspx.page Begin PreInit		
aspx.page End PreInit	0.000046	0.000046
aspx.page Begin Init	0.000070	0.000024
aspx.page End Init	0.000101	0.000031
aspx.page Begin InitComplete	0.000122	0.000021
aspx.page End InitComplete	0.000143	0.000021
aspx.page Begin LoadState	0.000163	0.000020
aspx.page End LoadState	0.000243	0.000080
aspx.page Begin ProcessPostData	0.000268	0.000025
aspx.page End ProcessPostData	0.000306	0.000039
aspx.page Begin PreLoad	0.000328	0.000022
aspx.page End PreLoad	0.000349	0.000021
aspx.page Begin Load	0.000369	0.000021
aspx.page End Load	0.000397	0.000028
aspx.page Begin ProcessPostData Second Try	0.000419	0.000021
aspx.page End ProcessPostData Second Try	0.000439	0.000020
aspx.page Begin Raise ChangedEvents	0.000460	0.000021
aspx.page End Raise ChangedEvents	0.000485	0.000025
aspx.page Begin Raise PostBackEvent	0.000506	0.000021
About to place an item in session state.	0.001081	0.000575
Placed item in session state.	0.001126	0.000045
aspx.page End Raise PostBackEvent	0.001155	0.000029
aspx.page Begin LoadComplete	0.001176	0.000021
aspx.page End LoadComplete	0.001197	0.000021
aspx.page Begin PreRender	0.001218	0.000021
aspx.page End PreRender	0.001255	0.000037

همچنین می‌توانید با استفاده از متدهای Write() یا Write() گروه بندی خاصی را تعیین کنید.

```
protected void cmdWriteCategory_Click(Object sender, EventArgs e)
{
    Trace.Write("cmdWriteCategory_Click",
    "About to place an item in session state.");
    Session["Test"] = "Contents";
    Trace.Write("cmdWriteCategory_Click",
    "Placed item in session state.");
}
```

Untitled Page			
localhost:52119/TraceExample.aspx			
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.000044	0.000044
aspx.page	Begin Init	0.000067	0.000023
aspx.page	End Init	0.000098	0.000031
aspx.page	Begin InitComplete	0.000119	0.000021
aspx.page	End InitComplete	0.000139	0.000021
aspx.page	Begin LoadState	0.000161	0.000022
aspx.page	End LoadState	0.000259	0.000098
aspx.page	Begin ProcessPostData	0.000284	0.000025
aspx.page	End ProcessPostData	0.000322	0.000038
aspx.page	Begin PreLoad	0.000344	0.000022
aspx.page	End PreLoad	0.000365	0.000021
aspx.page	Begin Load	0.000386	0.000021
aspx.page	End Load	0.000412	0.000026
aspx.page	Begin ProcessPostData Second Try	0.000434	0.000022
aspx.page	End ProcessPostData Second Try	0.000453	0.000020
aspx.page	Begin Raise ChangedEvents	0.000474	0.000021
aspx.page	End Raise ChangedEvents	0.000499	0.000025
aspx.page	Begin Raise PostBackEvent	0.000520	0.000021
cmdWriteCategory_Click	About to place an item in session state.	0.001173	0.000652
cmdWriteCategory_Click	Placed item in session state.	0.001216	0.000044
aspx.page	End Raise PostBackEvent	0.001245	0.000028
aspx.page	Begin LoadComplete	0.001266	0.000021
aspx.page	End LoadComplete	0.001287	0.000021
aspx.page	Begin PreRender	0.001307	0.000021
aspx.page	End PreRender	0.001338	0.000030

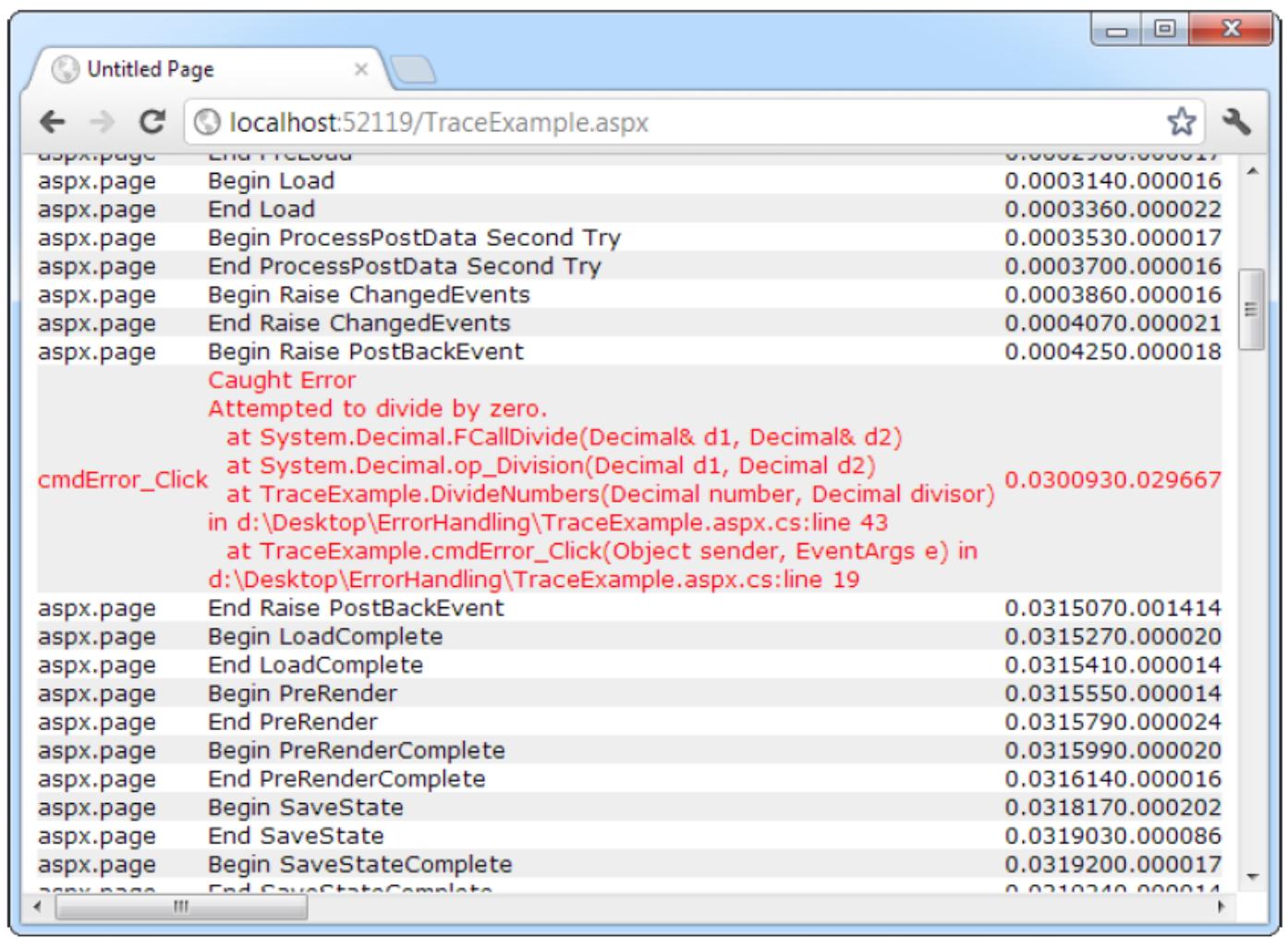
همچنین می‌توانید اطلاعات یک خط را نیز در trace نمایش دهید:

```
protected void cmdError_Click(Object sender, EventArgs e)
{
    try
    {
        DivideNumbers(5, 0);
    }
    catch (Exception err)
    {
        Trace.Warn("cmdError_Click", "Caught Error", err);
    }
}

private decimal DivideNumbers(decimal number, decimal divisor)
{
```

```
return number / divisor;
```

```
}
```



بصورت پیش فرض، پیام trace به ترتیبی که در کد نوشته می شوند، نمایش داده می شوند. به عنوان یک جایگزین با استفاده از خصیصه `TraceMode` در `Page Directive` می توانید تعیین کنید که پیام ها با چه ترتیبی باید ذخیره شوند.

```
<%@ Page Trace="true" TraceMode="SortByCategory" %>
```

```
Trace.TraceMode = TraceMode.SortByCategory;
```

یا

## Application-Level Tracing

این ویژگی به شما امکان فعال سازی ردیابی را برای کل برنامه می دهد. برای اطلاعات بیشتر باید منابع دیگری را مطالعه نمایید.

The screenshot shows a browser window titled "localhost:52119/trace.axd". The main content area is titled "Application Trace". It displays a table of "Requests to this Application" with the following data:

No.	Time of Request	File	Status	Code	Verb	Remaining:
1	6/27/2012 2:27:25 PM	SimpleTrace.aspx	200		GET	<a href="#">View Details</a>
2	6/27/2012 2:27:35 PM	SimpleTrace.aspx	200		POST	<a href="#">View Details</a>
3	6/27/2012 2:30:03 PM		200		GET	<a href="#">View Details</a>
4	6/27/2012 2:30:05 PM	TraceExample.aspx	200		GET	<a href="#">View Details</a>
5	6/27/2012 2:30:12 PM	ErrorTestLog.aspx	200		GET	<a href="#">View Details</a>
6	6/27/2012 2:30:14 PM	SimpleTrace.aspx	200		GET	<a href="#">View Details</a>
7	6/27/2012 2:30:16 PM	SimpleTrace.aspx	200		POST	<a href="#">View Details</a>
8	6/27/2012 2:30:20 PM	TraceExample.aspx	200		GET	<a href="#">View Details</a>

Below the table, a message reads: "Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.17626".

# ASP.NET : توسعه برنامه های بخش اول

Visual Studio  فصل اول آشنایی با

~~فصل دوم: اصول فرم مایع وب~~

~~فصل سوم: کنترل های وب~~  web control

~~فصل چهارم~~ Error Handling, Logging, Tracing

State Management  فصل پنجم



## State Management

### مدیریت وضعیت فرم

قابل توجه ترین تفاوت بین برنامه نویسی برنامه تحت وب و desktop state management (چگونگی ذخیره سازی اطلاعات در طول چرخه حیات برنامه) باشد. این اطلاعات می‌تواند نام کاربر یا ... باشد. در برنامه‌های قدیمی desktop، نیاز کمی به فکر درباره state management بود. حافظه همیشه در دسترس بود و شما تنها باید نگران یک کاربر بودید. در برنامه تحت وب داستان جور دیگری است. هزاران کاربر می‌توانند به طور همزمان یک برنامه را روی یک کامپیوتر اجرا کنند (وب سرور)، هر ارتباط از طریق اتصال HTTP که stateless نیز می‌باشد برقرار خواهد شد.

در این فصل با گزینه‌های مختلف ذخیره سازی این اطلاعات، شامل ViewState، Session State و Cookie آشنا خواهید شد. همچنین خواهید دید که چگونه اطلاعات را از صفحه‌ای به صفحه دیگر از طریق ارسال Query String و cross-page ارسال خواهیم کرد.

## آشنایی با مشکلی به نام State

در برنامه های desktop، کاربر با یک برنامه در حال اجرای مداوم سروکار داشت. بخشی از حافظه در کامپیوتر به ذخیره سازی مجموعه ای از اطلاعات کاری اختصاص می یافت.

در برنامه های تحت وب، ممکن است برنامه بصورت اجرای مداوم به نظر برسد، ولی در حقیقت اینطور نیست. در یک درخواست وب، کلاینت به وب سرور اتصال پیدا می کند و یک صفحه را درخواست خواهد کرد. زمانی که صفحه آماده و ارسال می شود، وب سرور همه اشیای صفحه را از درون حافظه پاک می کند. زمانی که کاربر صفحه را دریافت می کند، اجرای کدهای صفحه قبل از متوقف شده است و هیچ اطلاعاتی درون وب سرور باقی نمانده است.

این طراحی stateless (بدون حفظ وضعیت) یک فایده مهم دارد. به دلیل آنکه کلاینت ها تنها برای چند ثانیه نیاز به اتصال دارند، وب بروز می تواند تعدادی خیلی زیادی از درخواست ای همزمان را بدون مشکلی در کارایی اجرا کند. به هر حال اگر بخواهید اطلاعات را برای مدت بیشتری نگهدارید تا آنها بتوانند در طول چند postback یا در چند صفحه نیز استفاده شوند باید مراحل بیشتری را طی کنید.

## استفاده از View State

یکی از رایج ترین راه ها برای ذخیره اطلاعات، استفاده از View state می باشد. View state که hiddenfiled می باشد. بطری خودکار آن را در HTML رندر شده صفحه وب وارد می کند، استفاده می نماید. این محل، مکانی مناسب برای ذخیره اطلاعات مورد نیاز در چندین postback در یک صفحه می باشد.

View State تنها به کنترل های وب محدود نمی شود . صفحه وب شما نیز می تواند اطلاعات کوچکی را مستقیماً به view state خود صفحه اضافه و آنها را پس از postback شدن صفحه مجدداً بازیابی کند. می توانید انواع داده ای ساده و اشیای سفارشی خود را در آن ذخیره کنید.

## ViewState Collection

ویژگی view state صفحه، اطلاعات view-state کنونی را ارائه می دهد. ان ویژگی یک نمونه از کلاس StateBag collection می باشد. یک dictionary collection می باشد که هر آیتم را در یک بخش جداگانه با یک نام واحد (key) ذخیره می کند.

```
کلمه کلیدی this به شی صفحه کنونی اشاره می کند و اختیاری می باشد.
this.ViewState[“Counter”] = 1;
```

برای بازیابی مقدار آن باید از روش زیر استفاده کنیم :

```
int counter;
counter = (int)this.ViewState[“Counter”];
```

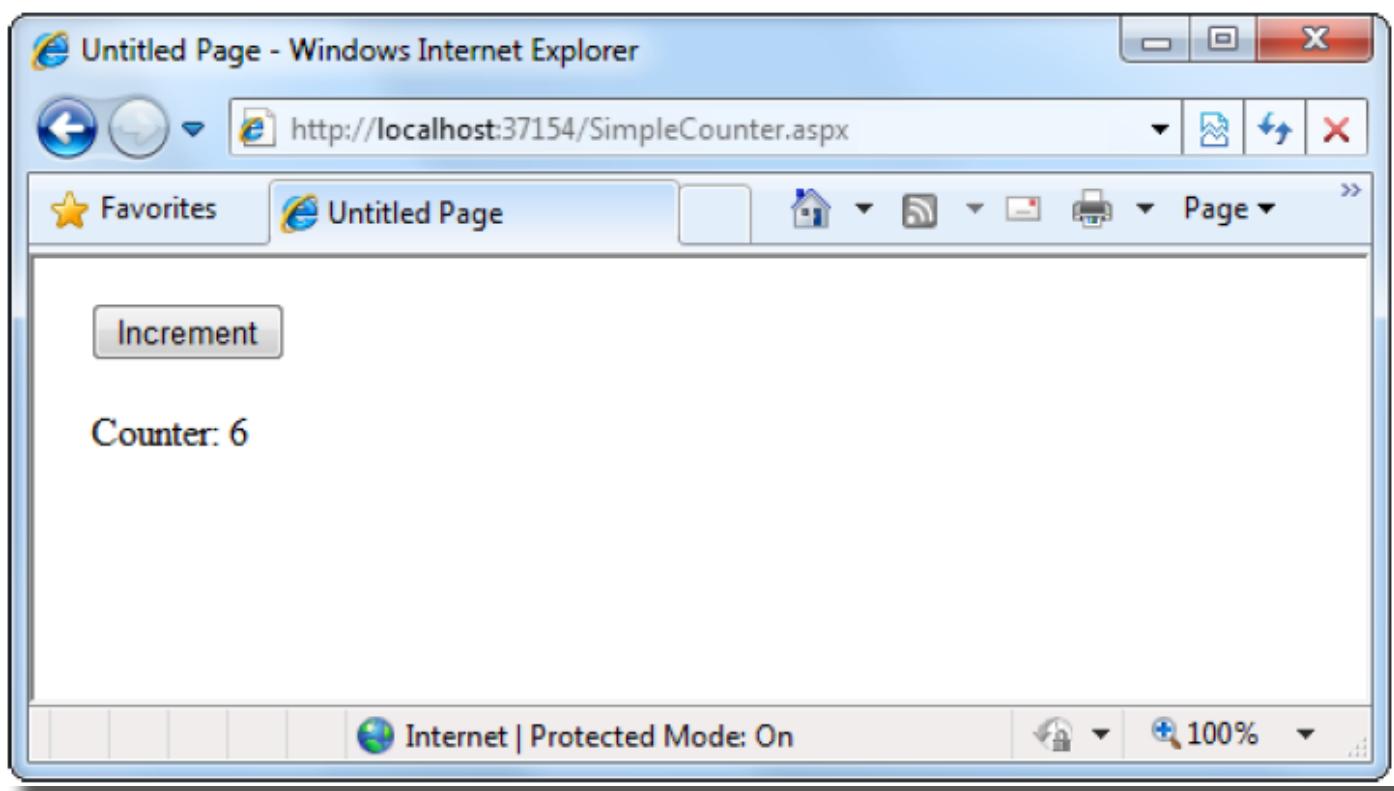
## یک مثال از View-State

مثال زیر یک برنامه ساده شمارنده می‌باشد که تعداد دفعاتی که یک دکمه کلیک شده است را ثبت می‌کند.

```
using System;

public partial class SimpleCounter : System.Web.UI.Page
{
    protected void cmdIncrement_Click(object sender, EventArgs e)
    {
        int counter;
        if (ViewState["Counter"] == null)
        {
            counter = 1;
        }
        else
        {
            counter = (int)ViewState["Counter"] + 1;
        }
        ViewState["Counter"] = counter;
        lblCount.Text = "Counter: " + counter.ToString();
    }
}
```

قبل از بازیابی View State، کد چک می‌کند تا مطمئن شود آیا آیتم در View State موجود است. در غیر اینصورت برنامه با خطای معروف null reference روبرو می‌شود.



## امن کردن View State

ASP.NET، دارای ابزاری است که می‌تواند View State را برای شما امن‌تر کند و آن را tamper-proof (در پایین با آن آشنا خواهید شد) سازد.

## Tamper-Proof View State

از hash code برای اطمینان از اینکه اطلاعات View-State، بدون اطلاع شما تغییر نخواهد کرد، استفاده می‌کند. تمامی داده‌های موجود در View State را قبل از رندر کردن صفحه نهایی، امتحان خواهد کرد و این داده‌ها را در الگوریتم hashing قرار می‌دهد. این الگوریتم یک بخش کوچک از داده ایجاد می‌کند که hash code نامیده می‌شود. این کد سپس به انتهای view state موجود در HTML نهایی ارسال شده به مرورگر اضافه خواهد شد.

زمانی که صفحه به سرور post back می‌شود، Hash code View state ASP.NET بسیار مفید است. در این زمان Hash code را امتحان کند و hash code را با استفاده از فرایند مشابه با ایجاد آن در دفعه اول، مجددًا محاسبه نماید. این کد جدید را با کد ذخیره شده مقایسه می‌کند. اگر کاربری، بخشی از این اطلاعات را تغییر داده باشد، hash code جدید ایجاد شده با قبلی برابر نخواهد بود، بنابراین کل postback را بر می‌گرداند و پیغام خطأ را نمایش می‌دهد. (کاربران به هیچ وجه نمی‌توانند hash code را ایجاد کنند زیرا ASP.NET مشابه با cryptograph key را ندارند)

Hash code ها بصورت پیشفرض فعال می‌باشند.

## Private View State

حتی زمانی که از hash code استفاده می‌کنید، داده view state هنوز توسط کاربر قابل خواندن خواهد بود. با استفاده از ویژگی encryption، می‌توانید Page Directive ViewStateEncryptionMode را برای صفحه خاصی فعال کنید.

```
<%@Page ViewStateEncryptionMode="Always" %>
```

یا این ویژگی را در فایل web.config برای کل صفحات برنامه فعال نمایید.

```
<configuration>
  ...
  <system.web>
    <pages ViewStateEncryptionMode="Always" />
  ...
</system.web>
</configuration>
```

سه گزینه برای این کار وجود دارد :

پیش فرض آن، گزینه Auto می‌باشد که یعنی صفحه view را encrypt نمی‌کند مگر آنکه یک کنترل در صفحه بصورت ویژه آن را درخواست کرده باشد. (از نظر تکنیکی یک کنترل این درخواست را توسط متده `Page.RegisterRequiresViewStateEncryption()` ایجاد می‌کند)

اگر هیچ کنترلی این متده را فراخوانی نکند تا بگوید که این اطلاعات حساس و مهم هستند، view state مربوطه کد نمی‌شود.

 نکته: در صورتیکه نیازی به کد کردن view state ندارید این کار را انجام ندهید، زیرا وب سرور مجبور می‌شود عملیات decryption و encryption را با هر postback انجام داده و کارایی پایین می‌آید.

## ذخیره سازی اشیای سفارشی

شما می‌توانید اشیای خود را به همان آسانی انواع عددی و رشته‌های کاراکتری در View state ذخیره کنید. به هر حال برای ذخیره سازی یک آیتم در view state ASP.NET قادر به تبدیل آن به جریانی از بایت‌ها باشد تا بتواند آن را به hidden input filed موجود در صفحه اضافه کند. این عملیات را serialization می‌نامند. اگر شی شما ( قادر به سریالیز کردن ) serializable نباشد، در هنگام قرار دادن آنها در view state، یک پیام خطأ دریافت می‌کنید.

برای serializable کردن اشیا، باید یک خصیصه Serializable، به قبل از تعریف کلاس اضافه کنید.

```
[Serializable]
public class Customer
{
    private string firstName;
    public string FirstName
    {
        get { return firstName; }
        set { firstName = value; }
    }
    private string lastName;
    public string LastName
    {
        get { return lastName; }
        set { lastName = value; }
    }
    public Customer(string firstName, string lastName)
    {
        FirstName = firstName;
        LastName = lastName;
    }
}
```

به دلیل آنکه کلاس Customer با خصیصه Serializable نشانه گذاری شده است، می‌توان آن را در view state ذخیره کرد :

```
// Store a customer in view state.

Customer cust = new Customer("Marsala", "Simons");
ViewState["CurrentCustomer"] = cust;
```

به یا داشته باشید زمانی که ز شی Custome استفاده می‌کنید، باید داده خود را در زمان بازیابی از view state نظر تبدیل کنید:

```
// Store a customer in view state.

Customer cust = new Customer("Marsala", "Simons");

ViewState["CurrentCustomer"] = cust;
```

برای تشخیص اینکه چه کلاس‌هایی از دات نت می‌توانند در view state ذخیره شوند می‌توانید از راهنمای موجود در استفاده کنید و تعریف کلاس مربوطه را در آن نگاه کنید:

```
[Serializable]
[ComVisible(true)]
public sealed class FileInfo : FileSystemInfo
```

## انتقال اطلاعات بین صفحات

یکی از محدودیت‌های view state، این است که فقط در یک صفحه می‌توان از آنها استفاده کرد. اگر کاربری به صفحات دیگری برود، به آن اطلاعات دسترسی نخواهد داشت.

در این بخش، شما دو تکنیک انتقال اطلاعات بین صفحات را خواهید آموخت: query string و cross-page posting.

## Cross-Page Posting

این تکنیک، مکانیزم postback را گسترش داده و بنابراین یک صفحه می‌تواند یک کاربر را به همراه همه اطلاعات صفحه به صفحات دیگر ارسال کند.

ویژگی PostBackUrl، که توسط اینترفیس IbuttonControl تعریف می‌شود و توسط LinkButton و ImageButton استفاده می‌شود، برای این تکنیک مناسب است. باید این ویژگی را با نام صفحه‌ای که می‌خواهید به آن منتقل شوید، مقداردهی کنید. زمانی که روی دکمه کلیک می‌شود، صفحه به همراه کلیه مقادیر درون کنترل‌های خود به URL جدید منتقل می‌شود.

```
<%@ page language="#c#" autoeventwireup="true" codefile="CrossPage1.aspx.cs" inherits="CrossPage1"%>
```

```
<html>
<head id="Head1" runat="server">
    <title>CrossPage1</title>
</head>
<body>
    <form id="form1" runat="server">
```

```

<div>

First Name:

<asp:TextBox ID="txtFirstName" runat="server"></asp:TextBox>

<br />

Last Name:

<asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>

<br />

<br />

<asp:Button runat="server" ID="cmdPost" PostBackUrl="CrossPage2.aspx" Text="Cross-
Page Postback" /><br />

</div>

</form>

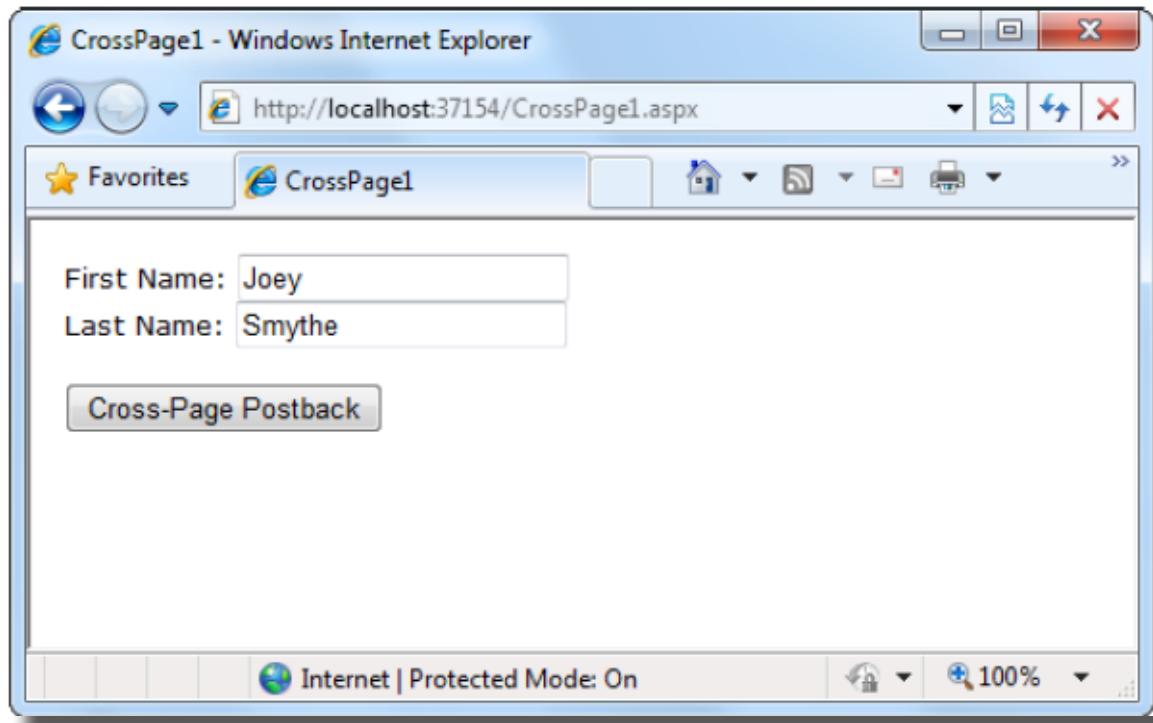
</body>

</html>

```

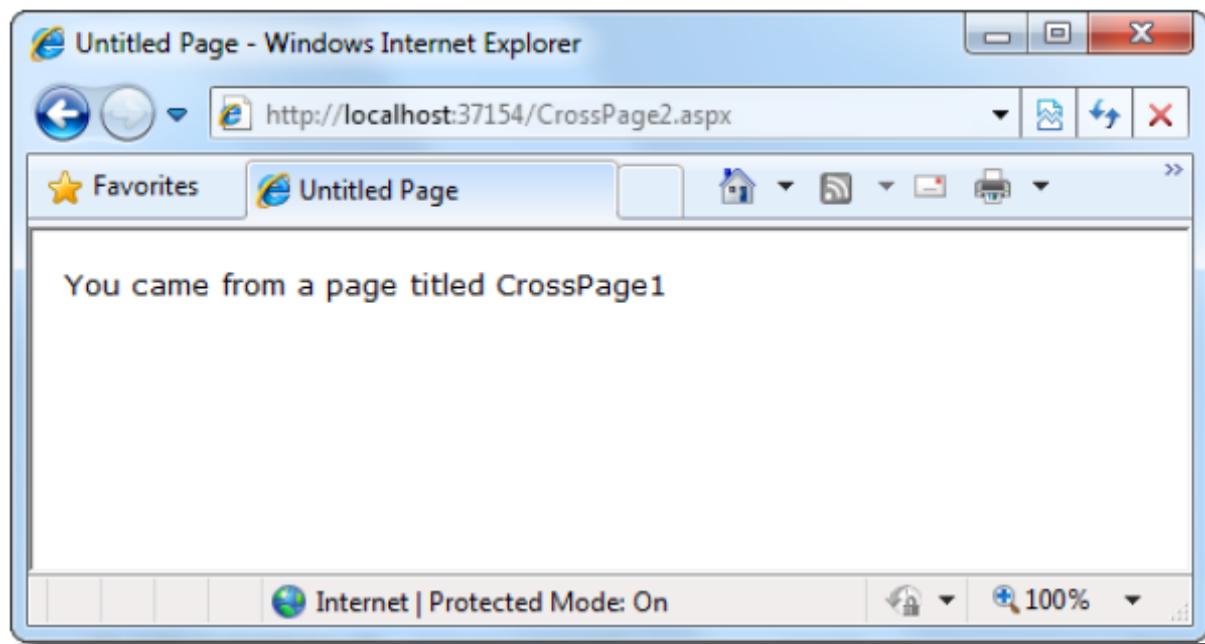
درون این صفحه هیچ کدی نیست.

اگر شما روی دکمه موجود در این صفحه کلیک کنید، به صفحه CrossPage2.aspx منتقل خواهید شد. در این صفحه می‌توانید با استفاده از ویژگی CrossPage1.aspx با Page.PreviousPage در تعامل باشید.



کد زیر، در صفحه CrossPage2.aspx وجود دارد که شامل یک event handler می‌باشد که عنوان صفحه قبل را گرفته و نمایش می‌دهد:

```
public partial class CrossPage2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (PreviousPage != null)
        {
            lblInfo.Text = "You came from a page titled " +
                PreviousPage.Title;
        }
    }
}
```



## دریافت اطلاعات بیشتر از صفحه مبدأ

برای دسترسی به جزئیات، مانند مقادیر موجود در کنترل‌ها باید صفحه قبل را به کلاس صفحه مورد نظر `cast` کنید (در مثال ما به کلاس `CrossPage1`).

```
public partial class CrossPage1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        CrossPage1 prevPage = PreviousPage as CrossPage1;
        if (prevPage != null)
        {
            // (Read some information from the previous page.)
        }
    }
}
```

بته می‌توانید به جای `cast` کردن دستی آن، راهنمای `PreviousPageType` را به بالای صفحه ای که دریافت کننده `cross page` است، اضافه کنید.

```
<%@ PreviousPageType VirtualPath="~/CrossPage1.aspx" %>
```

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

if (PreviousPage != null)
{
    // (Read some information from the previous page.)
}
```

**.CrossPage1.aspx** مثال : در صفحه 

```

public TextBox FirstNameTextBox
{
    get { return txtFirstName; }
}

public TextBox LastNameTextBox
{
    get { return txtLastName; }
}

public partial class CrossPage1 : System.Web.UI.Page
{
    public string FullName
    {
        get { return txtFirstName.Text + " " + txtLastName.Text; }
    }
}

```

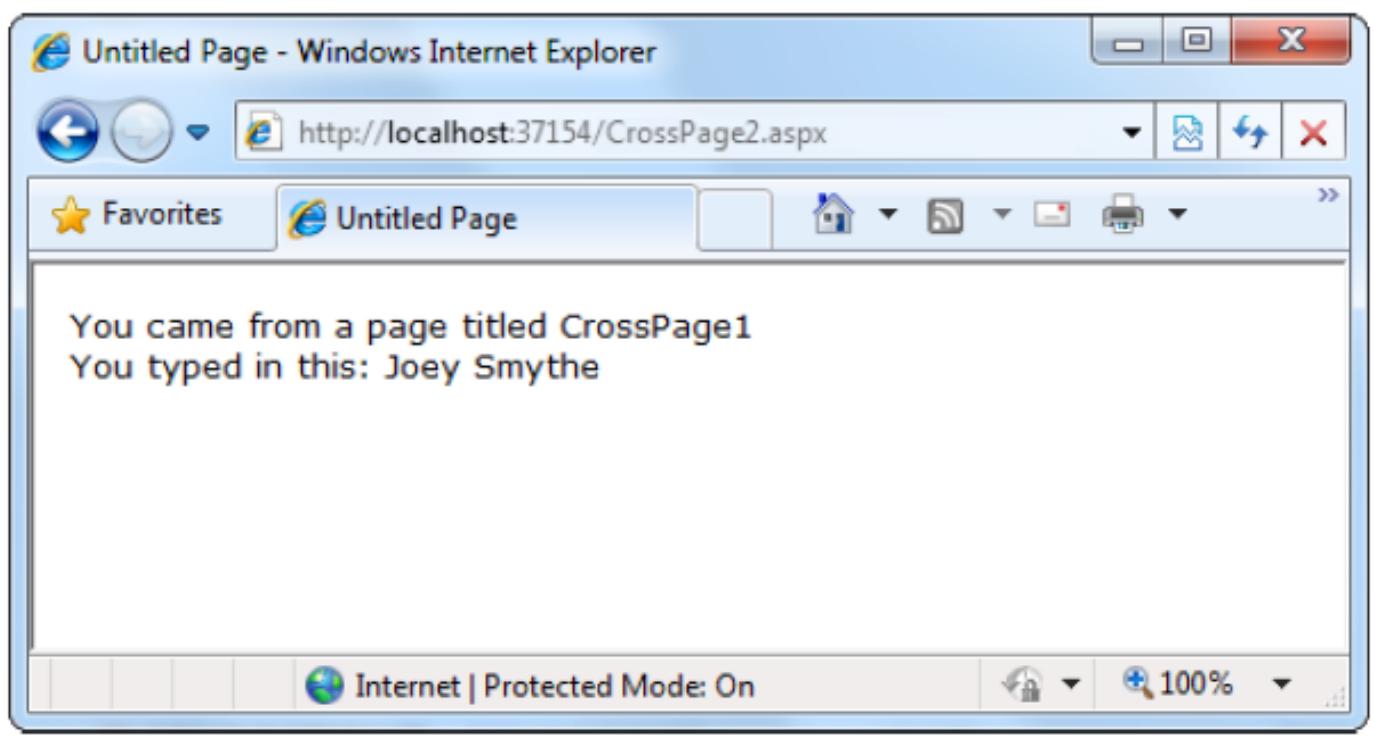
**.CrossPage2.aspx** مثال : در صفحه 

```

protected void Page_Load(object sender, EventArgs e)
{
    if (PreviousPage != null)
    {
        lblInfo.Text = "You came from a page titled " +
        PreviousPage.Title + "<br />";

        CrossPage1 prevPage = PreviousPage as CrossPage1;
        if (prevPage != null)
        {
            lblInfo.Text += "You typed in this: " + prevPage.FullName;
        }
    }
}

```



 نکته: تکنیک cross-page، روش مناسبی می‌باشد اما اگر چندین صفحه مبدأ به یک صفحه مقصد اشاره کنند، کار دشوار می‌شود. بنابراین بهتر هرگاه یک صفحه مبدأ و یک صفحه مقصد داشتیم از این روش استفاده کنیم.

## Query String

راه دیگر برای ارسال اطلاعات، استفاده از query string در URL می‌باشد. این روش بیشتر در موتورهای جستجو استفاده می‌شود. برای نمونه، اگر جستجویی در گوگل انجام دهید، شما به صفحه‌ای مانند زیر منتقل می‌شوید:

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

بخشی از URL می‌باشد که در مثال بالا `q` می‌باشد که شامل `organic+gradening` می‌باشد. فایده استفاده از این روش، سبکی و عدم سربار اضافی روی سرور می‌باشد. این روش محدودیت‌هایی نیز دارد :

- اطلاعات در یک string ساده محدود می‌شود که باید شامل کاراکترهای مجاز باشد.
- اطلاعات، بصورت واضح در اختیار کاربران می‌باشد.
- بسیاری از مرورگرها برای URL دارای محدودیت‌هایی می‌باشند (معمولًاً ۱ تا ۲ KB). بنابراین نمی‌توانید اطلاعات زیادی را در آن قرار دهید.
- ممکن است برخی از کاربران query string را تغییر داده و بنابراین برنامه شما قادر به پاسخگویی صحیح نخواهد بود و نمی‌تواند مانع انجام این کار شود.

برای نمونه فرض کنید دارای یک صفحه ای هستم که داخل خود لیستی دارد. کاربر یک آیتم از آن لیست را انتخاب و به صفحه بعد فرستاده می‌شود. در صفحه جدید جزئیات آیتم انتخاب شده از بانک اطلاعاتی دریافت و نمایش داده خواهد شد.

برای نگهداری اطلاعات در query string، ابتدا باید آن را به URL اضافه کنید. این کار توسط خودتان باید انجام گیرد.

```
// Go to newpage.aspx. Submit a single query string argument
// named recordID, and set to 10.

Response.Redirect("newpage.aspx?recordID=10");
```

می‌توانید چندین پارامتر را با یک جداکننده (separator) مانند & ارسال نمایید:

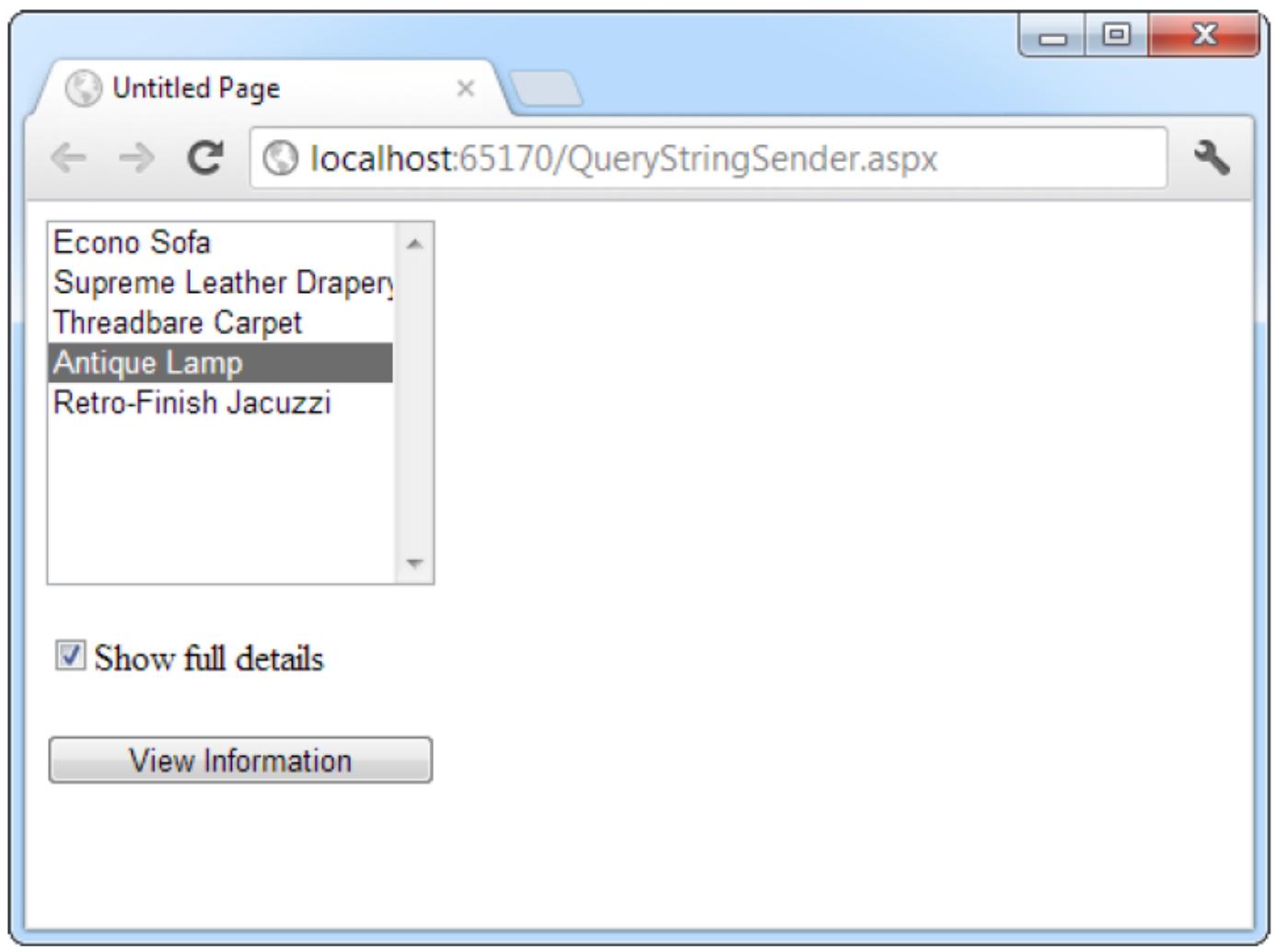
```
// Go to newpage.aspx. Submit two query string arguments:
// recordID (10) and mode (full).

Response.Redirect("newpage.aspx?recordID=10&mode=full");
```

برای دریافت مقادیر query string می‌توانید از یک دیکشنری با نام QueryString استفاده نمایید:

```
string ID = Request.QueryString["recordID"];
```

 نکته: برخلاف view state، اطلاعات ارسال شده توسط query string کاملاً در دسترس می‌باشند و کدگذاری نشده‌اند.  
بنابراین برای داده‌های حساس از آنها استفاده نکنید.



```

using System;

public partial class QueryStringSender : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            // Add sample values.

            lstItems.Items.Add("Econo Sofa");
            lstItems.Items.Add("Supreme Leather Drapery");
            lstItems.Items.Add("Threadbare Carpet");
            lstItems.Items.Add("Antique Lamp");
        }
    }
}

```

```

        lstItems.Items.Add("Retro-Finish Jacuzzi");

    }

}

protected void cmdGo_Click(Object sender, EventArgs e)
{
    if (lstItems.SelectedIndex == -1)
    {
        lblError.Text = "You must select an item.";
    }
    else
    {
        // Forward the user to the information page,
        // with the query string data.

        string url = "QueryStringRecipient.aspx?";
        url += "Item=" + lstItems.SelectedItem.Text + "&";
        url += "Mode=" + chkDetails.Checked.ToString();
        Response.Redirect(url);
    }
}
}

```



```

public partial class QueryStringRecipient : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        lblInfo.Text = "Item: " + Request.QueryString["Item"];
        lblInfo.Text += "<br />Show Full Record: ";
        lblInfo.Text += Request.QueryString["Mode"];
    }
}

```

## URL Encoding

یکی از مشکلات بالقوه query string، این است که برخی از کاراکترها اجازه قرار گرفتن در URL را ندارند. در حقیقت، لیستی از کاراکترهایی که امکان قرار گرفتن در URL را دارند، کوتاهتر از کاراکترهای مجاز سند HTML می‌باشد. همه کاراکترها باید الفبا یا یکی از اعضای مجموعه کوچک کاراکترهای خاص ( مانند \$-\_!@#\$%^& ) باشد. در برخی از مرورگرها این کاراکترها اندکی متفاوت خواهند بود. برخی از کاراکترها نیز معنای خاصی دارند. برای نمونه کاراکتر & برای جداسازی پارامترهای ارسالی بکار می‌رود و # برای اشاره به یک نقطه از صفحه با استفاده از bookmark استفاده می‌شود. اگر query string شما دارای این کاراکترهای خاص باشد، بخشی از داده شما از دست می‌رود. برای جلوگیری از این مشکل، کد کردن URL (URL Encoding) ایده مناسبی می‌باشد.

با استفاده از این روش، کاراکترهای خاص با کاراکتر % به همراه دو عدد هگزا دسیمال در ادامه جایگزین می‌شوند.

برای پیاده سازی این روش، از متدهای UrlEncode() و UrlDecode() از کلاس HttpServerUtility استفاده نمایید. این دو متدهای Page.Server طریق در دسترس خواهند بود.

```

string url = "QueryStringRecipient.aspx?";
url += "Item=" + Server.UrlEncode(lstItems.SelectedItem.Text) + "&";
url += "Mode=" + chkDetails.Checked.ToString();
Response.Redirect(url);

```

## استفاده از Cookie

کوکی ها راه دیگری برای ذخیره اطلاعات برای استفاده مجدد در دفعات بعد می‌باشند. کوکی ها فایل‌های کوچکی می‌باشند که در حافظه مرورگر وب (اگر آنها موقت باشند) یا در هارددرایو کلاینت (اگر دائمی باشند) ذخیره خواهند شد. این اطلاعات می‌توانند توسط همه صفحات برنامه شما استفاده شوند و حتی در بین مشاهده های مختلف باقی بمانند که به شما اجازه ذخیره سازی بلند مدت را می‌دهند. این فایل‌ها حاوی اطلاعات ساده string می‌باشند و اگر کاربر فایل‌های مربوطه را پیدا و باز کند به سادگی قابل خواندن هستند. این فاکتورها آنها را یک گزینه ضعیف برای ذخیره داده های پیچیده یا محرومانه قرار داده است.

بعضی از کاربران کوکی ها را بر روی مرورگر خود غیر فعال می کنند که ممکن است برای برنامه تحت وب شما که نیازمند آنها می باشد، مشکلاتی را به وجود آورد. همچنانی کاربران ممکن است بصورت دستی این فایل ها را پاک کنند .  
ابتدا کوکی را ایجاد و به مجموعه آن، موارد مورد نظر را اضافه و در پاسخ صفحه، به کلاینت می فرستیم.

```
// Create the cookie object.  
  
HttpCookie cookie = new HttpCookie("Preferences");  
  
// Set a value in it.  
  
cookie["LanguagePref"] = "English";  
  
// Add another value.  
  
cookie["Country"] = "US";  
  
// Add it to the current web response.  
  
Response.Cookies.Add(cookie);
```

یک کوکی که به روش بالا اضافه می شود، تا زمان که کاربر مرورگر را ببندد باقی می ماند و با هر درخواست فرستاده می شود.  
برای ایجاد کوکی هایی با طول عمر بیشتر می توانید تاریخ انقضا برای آنها تعیین کنید :

```
// This cookie lives for one year.  
  
cookie.Expires = DateTime.Now.AddYears(1);
```

می توانید کوکی را با استفاده از نام آن بازیابی کنید.  
HttpCookie cookie = Request.Cookies["Preferences"];

قبل از استفاده از کوکی بازیابی شده باید مطمئن شد که محتوای آن NULL نمی باشد :

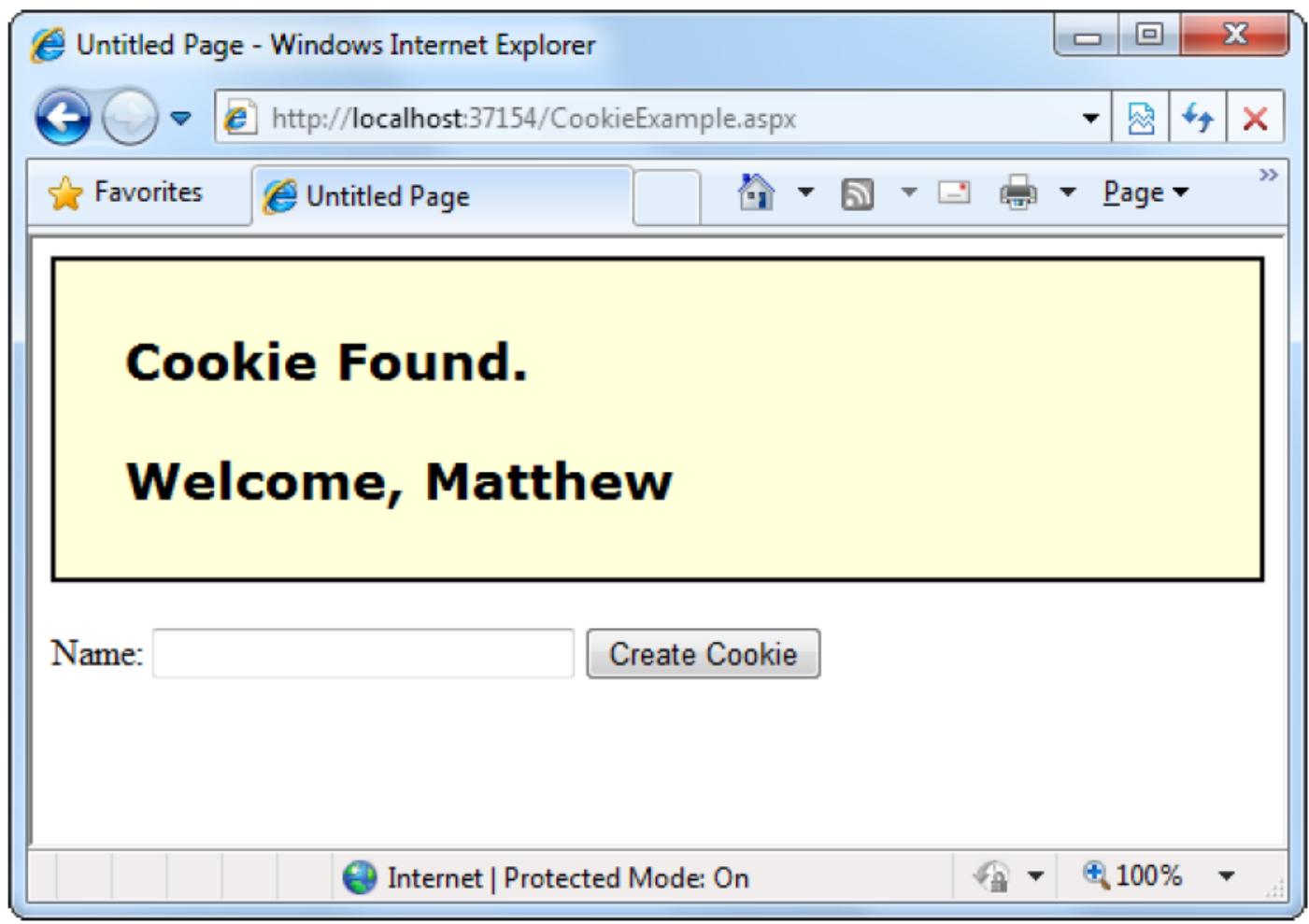
```
if (cookie != null)  
{  
    language = cookie["LanguagePref"];  
}
```

تنها راه برای حذف یک کوکی، جایگزینی آن با یک کوکی که تاریخ انقضا آن گذشته می باشد.  
HttpCookie cookie = new HttpCookie("Preferences");  
cookie.Expires = DateTime.Now.AddDays(-1);  
Response.Cookies.Add(cookie);

```

public partial class CookieExample : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        HttpCookie cookie = Request.Cookies["Preferences"];
        if (cookie == null)
        {
            lblWelcome.Text = "<b>Unknown Customer</b>";
        }
        else
        {
            lblWelcome.Text = "<b>Cookie Found.</b><br /><br />";
            lblWelcome.Text += "Welcome, " + cookie["Name"];
        }
    }
    protected void cmdStore_Click(Object sender, EventArgs e)
    {
        // Check for a cookie, and create a new one only if
        // one doesn't already exist.
        HttpCookie cookie = Request.Cookies["Preferences"];
        if (cookie == null)
        {
            cookie = new HttpCookie("Preferences");
        }
        cookie["Name"] = txtName.Text;
        cookie.Expires = DateTime.Now.AddYears(1);
        Response.Cookies.Add(cookie);
        lblWelcome.Text = "<b>Cookie Created.</b><br /><br />";
        lblWelcome.Text += "New Customer: " + cookie["Name"];
    }
}

```



```

public partial class CookieExample : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        HttpCookie cookie = Request.Cookies["Preferences"];
        if (cookie == null)
        {
            lblWelcome.Text = "<b>Unknown Customer</b>";
        }
        else
        {
            lblWelcome.Text = "<b>Cookie Found.</b><br /><br />";
            lblWelcome.Text += "Welcome, " + cookie["Name"];
        }
    }
}

```

```

protected void cmdStore_Click(Object sender, EventArgs e)
{
    // Check for a cookie, and create a new one only if
    // one doesn't already exist.

    HttpCookie cookie = Request.Cookies["Preferences"];
    if (cookie == null)
    {
        cookie = new HttpCookie("Preferences");
    }
    cookie["Name"] = txtName.Text;
    cookie.Expires = DateTime.Now.AddYears(1);
    Response.Cookies.Add(cookie);
    lblWelcome.Text = "<b>Cookie Created.</b><br /><br />";
    lblWelcome.Text += "New Customer: " + cookie["Name"];
}
}

```

## مدیریت Session State

ممکن است یک برنامه نیازمند ذخیره و دسترسی به اطلاعاتی مانند اشیای داده ای سفارشی باشد که نمی توانند به سادگی درون کوکی نگهداری یا از طریق Query String ارسال شوند. یا ممکن است یک برنامه از نظر امنیتی نتواند داده ها را در View state ذخیره کند. مدیریت وضعیت Session، یکی از امکانات خوب ASP.NET برای ذخیره هر نوع از داده ها در حافظه بر روی سرور می باشد. این اطلاعات محافظت شده (Protected) هستند زیرا هرگز به کلاینت ارسال نمی شوند و بصورت یکتا به یک Session bind خاص می شوند. هر کلاینت که به برنامه دسترسی دارد، دارای Session و مجموعه ای از اطلاعات مجازی متفاوتی می باشد. برای ذخیره اطلاعاتی مانند آیتم های موجود در سبد خرید کاربران در هنگام انتقال کاربر از یک صفحه به صفحه دیگر استفاده می شود.

## Session Tracking

را با استفاده از یک شناسه 120 بیتی پیگیری می کند. این شناسه یکتا می باشد و بصورت تصادفی ایجاد می شود. این شناسه تنها بخشی از اطلاعات مرتبط Session می باشد که بین وب سرور و کلاینت منتقل می شود.

زمانی که کلاینت Session ID را ارائه می دهد، به دنبال Session مربوطه ای می گردد که اشیایی که شما قبل از ذخیره کرده اید را بازیابی کند و آنها را در یک مجموعه خاص که از طریق کد شما قابل دسترس باشد قرار می دهد. این پردازش بصورت خودکار انجام می شود.

برای اینکه این سیستم کار کند، کلاینت باید با هر درخواست یک Session ID مناسب ارائه دهد. این کار از دو طریق قابل انجام می باشد :

## با استفاده از کوکی ها :

در این روش، ASP.NET Session ID به یک کوکی خاص منتقل می شود (با نام ASP.NET\_SessionId)، که آن را بصورت خودکار ایجاد می کند. این روش، پیش فرض می باشد.

## با استفاده از URL های تغییر داده شده :

در این روش، URL خاص تغییر داده شده منتقل می شود. این روش به شما اجازه ایجاد برنامه هایی که از استفاده می کنند را می دهد. (در کلاینت هایی که کوکی را پشتیبانی نمی کنند)

## استفاده از Session State

با استفاده از .Session state می توانید با System.Web.SessionState HttpSessionState در تعامل باشید. سیستمکس اضافه نمودن و بازیابی یک آیتم در Session مانند ViewState می باشد.

```
Session[“InfoDataSet”] = dsInfo;
```

```
dsInfo = (DataSet)Session[“InfoDataSet”];
```

در سراسر برنامه شما برای کاربر فعلی عمومی می باشد. session state به دلایل زیر می تواند گم شود یا از دست برود :

- اگر کاربر مرورگر را ببندد و دوباره باز کند.
- اگر کاربر به یک صفحه از طریق پنجره های مرورگر مختلفی دسترسی داشته باشد، البته در صورتی که صفحه از طریق پنجره اصلی مرورگر در دسترس باشد، session هنوز وجود دارد. مرورگرها در چگونگی هندل کردن این موقعیت، متفاوت عمل می کنند.
- اگر session time out شود. اطلاعات بیشتر در باره session time out را می توانید در بخش پیکربندی پیدا کنید.
- اگر کد صفحه وب شما session را توسط متده است abandon() Session.Abandon() از بین برده باشد.



توضیحات	عضو
تعداد آیتم های موجود در Session Collection فعلی.	Count
تعیین می کند آیا Session با کوکی یا modified url پیگیری شده است یا نه.	IsCookieless
تعیین می کند که آیا Session تنها برای درخواست کنونی ایجاد شده است یا نه. اگر اطلاعاتی در session state موجود نباشد، برای ایجاد یا پیگیری یک session cookie، کاری انجام نمی دهد. در عوض، Session، با هر درخواست مجددًا ایجاد می شود.	IsNewSession

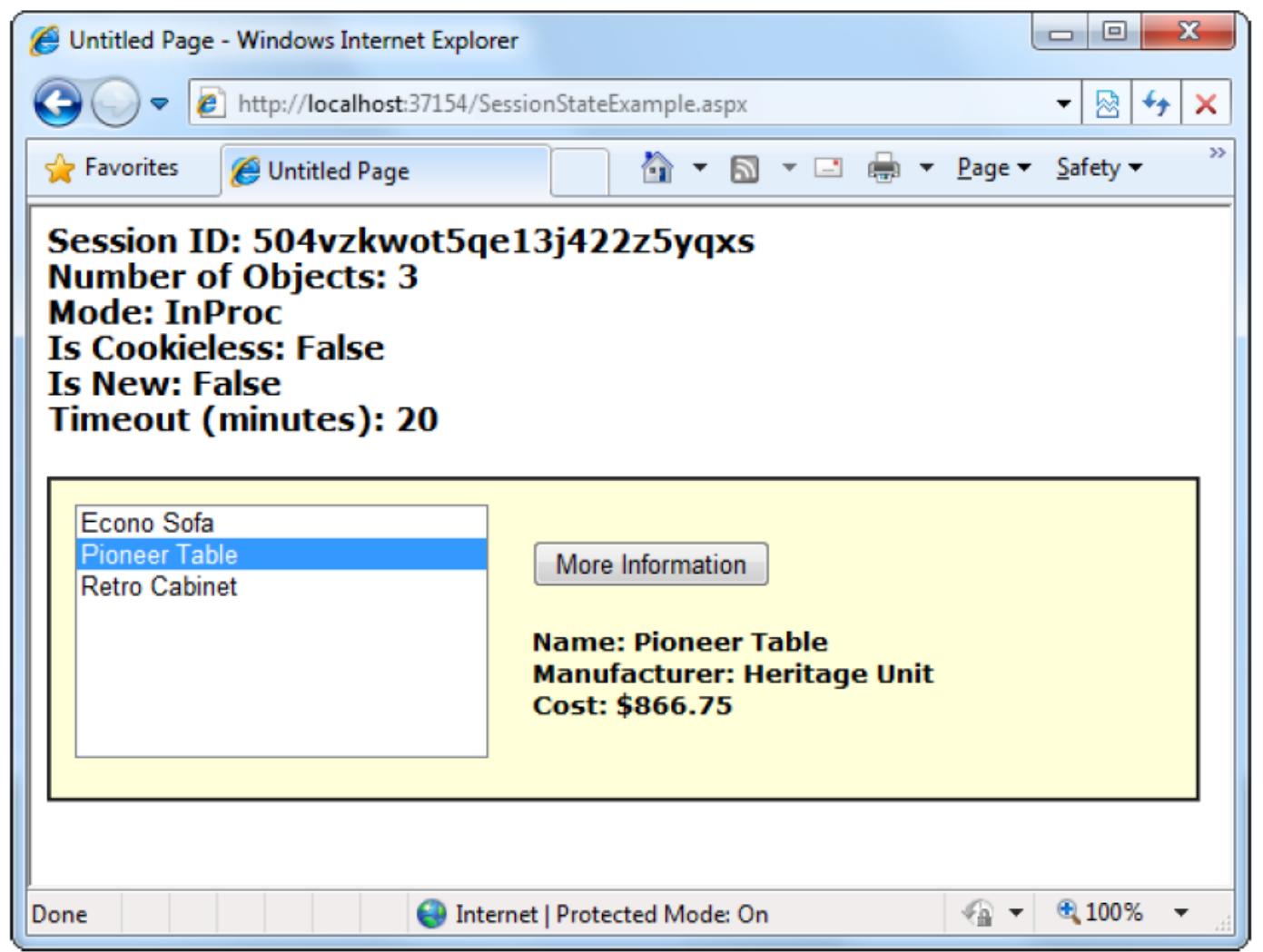
مجموعه‌ای از کلیدهای session را که برای نگهداری آیتم‌ها در session state استفاده می‌شود را، بر می‌گرداند.	Keys
یک نوع شمارشی که چگونگی ذخیره سازی اطلاعات session state را تعیین می‌کند، ارائه می‌دهد. ان نوع ذخیره سازی، بر اساس تنظیمات web.config تعیین می‌شود.	Mode
یک string با یک شناسه یکتا برای کلاینت فعلی ارائه می‌دهد.	SessionID
تعداد دقیقه‌هایی که session پس از آن از بین می‌رود را مشخص می‌کند.	Timeout
Session فعلی را فوراً کنسل کرده و کلیه منابع حافظه را آزاد می‌سازد. این تکنیک برای صحه logoff (خروج کاربر از برنامه) مناسب است زیرا شما را مطمئن می‌سازد که حافظه به سرعت آزاد گشته است.	Abandon()
کلیه آیتم‌های session را پاک می‌کند ولی شناسه session فعلی را تغییر نمی‌دهد.	Clear()

## یک مثال کاربردی از Session State

در مثال زیر از Session برای ذخیره چندین شی داده Furniture استفاده می‌شود. این شی داده، ترکیبی از چندین متغیر مرتبط و یک سازنده خاص می‌باشد. بنابراین می‌تواند به سادگی ایجاد و مقدار دهی شود.

```
public class Furniture
{
    public string Name;
    public string Description;
    public decimal Cost;
    public Furniture(string name, string description,
                    decimal cost)
    {
        Name = name;
        Description = description;
        Cost = cost;
    }
}
```

این اشیای Furniture اولین بار در زمان لود شدن صفحه ایجاد می‌شوند و سپس در Session state ذخیره می‌شوند. کاربر سپس می‌تواند از لیست furniture-Piece، موردی را انتخاب کند. زمانی که یک انتخاب ساخته شده، شی مرتبط با آن بازیابی می‌شود و اطلاعات آن نمایش داده می‌شوند.



```
public partial class SessionStateExample : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            // Create Furniture objects.

            Furniture piece1 = new Furniture("Econo Sofa",
                "Acme Inc.", 74.99M);

            Furniture piece2 = new Furniture("Pioneer Table",
                "Heritage Unit", 866.75M);

            Furniture piece3 = new Furniture("Retro Cabinet",
                "Sixties Ltd.", 300.11M);
        }
    }
}
```

```

// Add objects to session state.

Session["Furniture1"] = piece1;

Session["Furniture2"] = piece2;

Session["Furniture3"] = piece3;

// Add rows to list control.

lstItems.Items.Add(piece1.Name);

lstItems.Items.Add(piece2.Name);

lstItems.Items.Add(piece3.Name);

}

// Display some basic information about the session.

// This is useful for testing configuration settings.

lblSession.Text = "Session ID: " + Session.SessionID;

lblSession.Text += "<br />Number of Objects: ";

lblSession.Text += Session.Count.ToString();

lblSession.Text += "<br />Mode: " + Session.Mode.ToString();

lblSession.Text += "<br />Is Cookieless: ";

lblSession.Text += Session.IsCookieless.ToString();

lblSession.Text += "<br />Is New: ";

lblSession.Text += Session.IsNewSession.ToString();

lblSession.Text += "<br />Timeout (minutes): ";

lblSession.Text += Session.Timeout.ToString();

}

protected void cmdMoreInfo_Click(Object sender, EventArgs e)

{

if (lstItems.SelectedIndex == -1)

{

lblRecord.Text = "No item selected.';

}

else

{



// Construct the right key name based on the index.

```

```

        string key = "Furniture" +
    (lstItems.SelectedIndex + 1).ToString();

    // Retrieve the Furniture object from session state.

    Furniture piece = (Furniture)Session[key];

    // Display the information for this object.

    lblRecord.Text = "Name: " + piece.Name;

    lblRecord.Text += "<br />Manufacturer: ";

    lblRecord.Text += piece.Description;

    lblRecord.Text += "<br />Cost: " + piece.Cost.ToString("c");

}

}

}

```

## پیکربندی Session State

می‌توانید برای برنامه خود، Session State را از طریق web.Config پیکربندی نمایید. فایل پیکربندی، به شما اجازه تعیین گزینه های پیشرفتی مانند session-state mode و timeout را می‌دهد.

لیست زیر، مهم ترین گزینه هایی که می‌توانید برای المان <session State> تعیین کنید را نمایش می‌دهد. البته شما همه این گزینه ها را در یک زمان استفاده نخواهید کرد.

```

<configuration>
    ...
<system.web>
    ...
    <sessionState
        cookieless="UseCookies"
        cookieName="ASP.NET_SessionId"
        regenerateExpiredSessionId="false"
        timeout="20"
        mode="InProc"
        stateConnectionString="tcpip=127.0.0.1:42424">
    </sessionState>
</system.web>

```

```

stateNetworkTimeout="10"
sqlConnectionString="data source=127.0.0.1;Integrated Security=SSPI"
sqlCommandTimeout="30"
allowCustomSqlDatabase="false"
customProvider=""
compressionEnabled="false"
/>
</system.web>
</configuration>

```

## Timeout

تنظیمات timeout، تعداد دقیقه هایی که ASP.NET بدون دریافت درخواستی، قبل از بین بردن یا Session منتظر می ماند را نمایش می دهد.

اگر مدت این زمان زیاد باشد، حافظه مورد نیاز برنامه تحت وب افزایش پیدا می کند و کارایی آن کاهش پیدا خواهد کرد. این زمان را می توانید از طریق برنامه نویسی نیز تعیین کنید.

```
Session.Timeout = 10;
```

## Mode

این تنظیمات به شما اجازه پیکربندی ASP.NET را برای استفاده از سرویس Session-State مختلف بر اساس mode ای که شما انتخاب کرده اید می دهد.

 نکته : تغییر mode، یک عملیات پیکربندی پیشرفته می باشد. برای انجام موفقیت آمیز آن باید با محیطی که برنامه شما در آن توسعه خواهد یافت، آشنا باشید.

## InProc

مد پیش فرض inproc می باشد و بیشتر برای وب سایت های کوچک مناسب است. این حالت، اطلاعات را برای ذخیره سازی در همان پردازش ASP.NET worker thread، که بهترین کارایی را اما با حداقل دوام دارد، سازماندهی می کند.

اگر سرور را restart کنید، اطلاعات وضعیت آن از بین می رود. این مد، در حالتی که از webfarm استفاده می کنید، کاربردی ندارد.

## Off

این تنظیمات، مدیریت session state را برای هر صفحه در برنامه غیر فعال می کند. این کار اندکی در بهبود کارایی برای وب سایتهاست. که از session state استفاده نمی کنند، اثر خواهد داشت.

## StateServer

با این تنظیم، از یک پنجره خدمات جداگانه برای مدیریت state استفاده خواهد کرد. این سرویس در همان وب سرور اجرا خواهد شد اما در خارج از پردازش اصلی ASP.NET

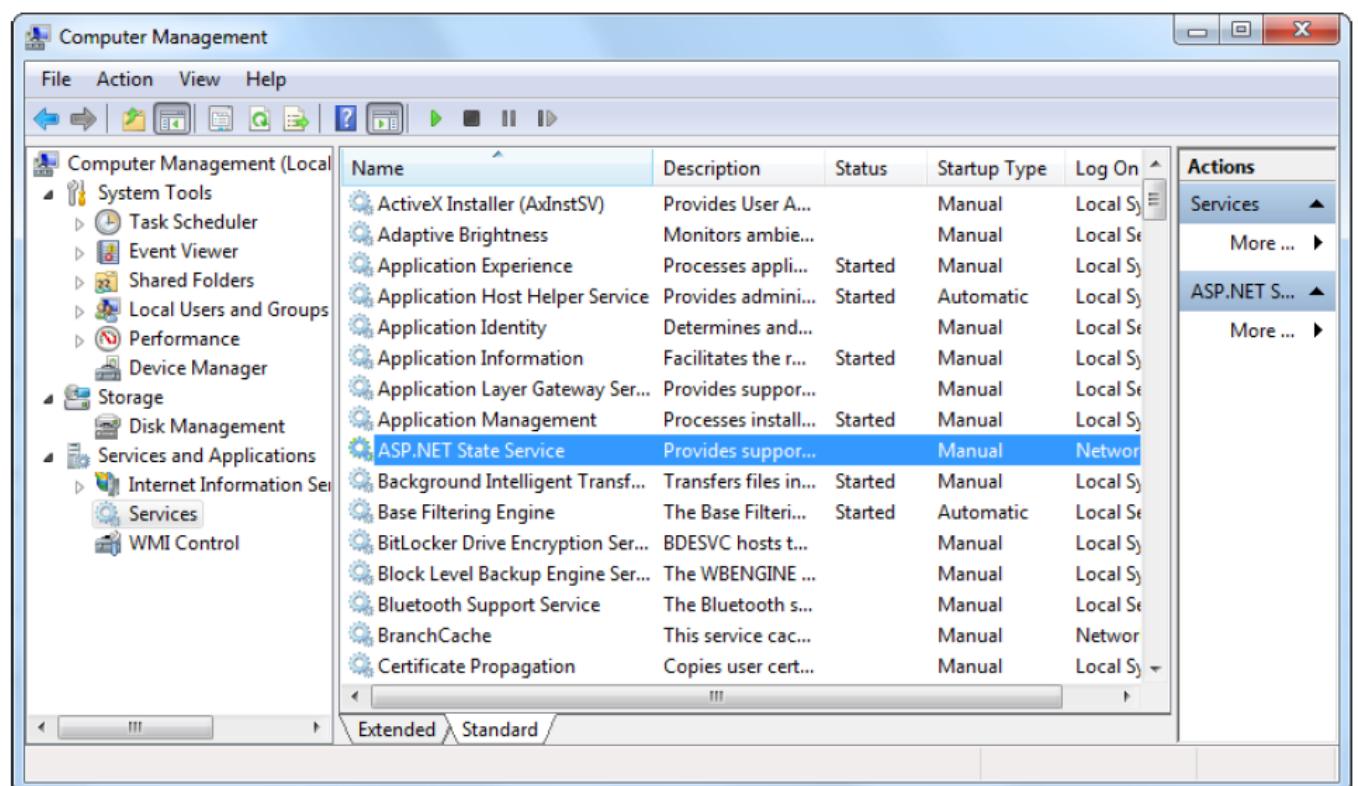
هزینه این روش، افزایش تأخیر در هنگام انتقال اطلاعات بین دو پردازش می باشد. اگر مرتباً به اطلاعات state دسترسی دارید و آن را تغییر می دهید، این روش می تواند اندکی کند باشد.

زمانی که از این روش استفاده می کنید، نیاز دارید یک مقدار برای StateConnectionString تعیین کنید. این رشتة، آدرس TCP/IP از کامپیوتری که در آن سرویس StateService در حال اجرا می باشد را به همراه شماره پورت آن مشخص می کند. این مد به شما امکان قرار دادن stateserver در کامپیوتر دیگر را می دهد. اگر این تنظیمات را تغییر ندهید، از local server استفاده می شود. برای start کردن این سرویس برای استفاده در برنامه، می توان از Microsoft Management Console (MMC) استفاده کرد.

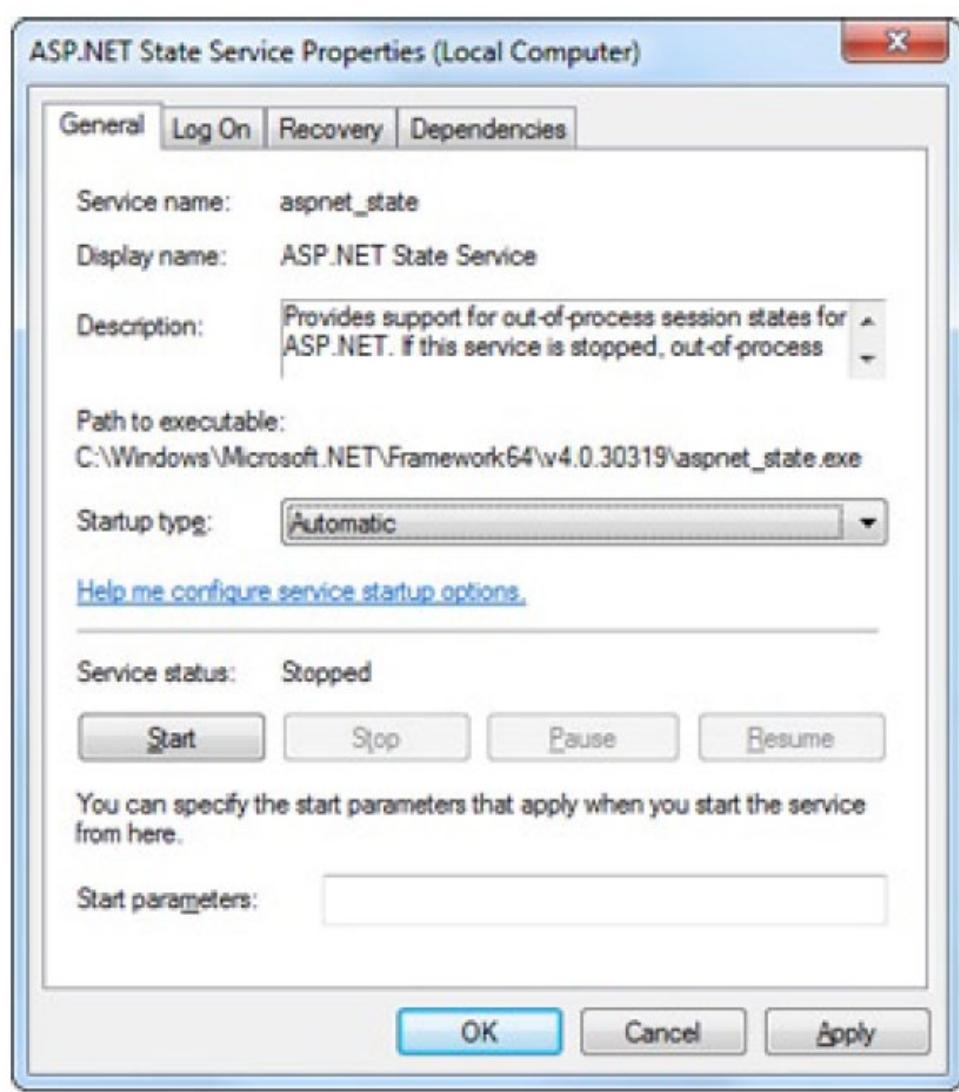
۱- روی Start کلیک و Computer Management را در کادر تایپ کنید.

۲- در پنجره ظاهر شده، روی Services>Applications>Services کلیک کنید.

۳- در لیست، سرویس ASP.NET State Service را پیدا کنید.



- زمانی که سرویس را در لیست پیدا کردید، می‌توانید بصورت دستی آن را start یا stop کنید.



## SQLServer

این تنظیمات، ASP.NET را برای استفاده از بانک اطلاعاتی SQL Server به منظور ذخیره اطلاعات session، آماده می‌کند. برای استفاده از این سرویس باید SQL Server را نصب کرده باشید. در هنگام مقدار دهی ویژگی SqlConnectionString، شما از ترتیب الگوی مشابه با دسترسی داده در ADO.NET پیروی خواهید کرد. در کل، شما باید یک منبع داده (آدرس سرور) و یک User ID و Password را تعیین کنید مگر اینکه از حالت SQL Integrated Security استفاده کرده باشید.

علاوه بر این، باید store procedure های خاص را نیز نصب کنید. این SP ها مراقب ذخیره و بازیابی اطلاعات session هستند. ASP.NET، دارای یک ابزار command line با نام aspnet\_regsql.exe می‌باشد که این کار را برای شما بصورت خودکار انجام می‌دهد. این ابزار در مسیر زیر پیدا می‌شود :

c:\Windows\Microsoft.NET\Framework64\v4.0.30319

برای اجرای این ابزار باید پنجره command prompt را در این مکان باز نمایید. (یک راه ساده برای این کار، کلیک روی پوشش شماره نسخه و نگه داشتن کلید shift و سپس کلیک راست روی آن و انتخاب گرینه aspnet\_regsql.exe“ می‌باشد) اگر در پوشش صحیح بودید، می‌توانید دستور “Open command prompt window here” را تایپ کنید.

با این دستور حتی می‌توانید، یک بانک اطلاعاتی session storage ایجاد نمایید :

aspnet\_regsql.exe -S localhost -E -ssadd

پس از ایجاد بانک اطلاعاتی session state ASP.NET بگویید که با دستکاری و تغییر در بخش <sessionState> در فایل web.config می‌تواند از آن استفاده کند.

```
<sessionState mode="SQLServer">
  <sqlConnectionString="data source=127.0.0.1;Integrated Security=SSPI">
    ...="" />
```

## Custom

در این مرحله، با استفاده از ویژگی customProvider، باید تعیین کنید کدام یک از session state store provider ها، استفاده می‌شود. این ویژگی نام کلاس را تعیین کنید. یک کلاس ممکن است بخشی از برنامه تحت وب شما یا درون اسمنبلی برنامه تحت وب باشد.

## Compression

زمانی که enableCompression را با مقدار true تعیین می‌کنید، داده session شما فشرده می‌شود. این ویژگی زمانی کار می‌کند که شما از حالت out-of-process برای ذخیره سازی session استفاده کرده باشید.

برای session state compress و decompress، وب سرور باید کارهای بیشتری را انجام دهد. فشرده سازی session state با دو سناریوی اصلی زیر انجام می‌شود :

- در هنگام ذخیره سازی مقدار حجمی از داده session در حافظه : حافظه وب سرور یک منبع با ارزش می‌باشد. معمولاً session state، برای بخش‌های کوچکی از داده استفاده می‌شود. اگر حجم این داده‌ها زیاد باشد از فشرده سازی استفاده می‌شود.

- زمانی که داده‌های session state را روی کامپیوتر دیگری نگهداری می‌کنیم. در برخی از برنامه‌های تحت وب بزرگ، session state در خارج پردازش اصلی (در SQL Server) روی یک دستگاه دیگر ذخیره می‌شود. در نتیجه، ASP.NET باید داده‌های session را از طریق شبکه منتقل کند. این طراحی، سرعت را به نسبت زمانی که session بر روی وب سرور ذخیره می‌شود، کاهش می‌دهد. به هر حال این روش هنوز بهترین تصمیم برای برخی از برنامه‌های تحت وب با ترافیک بالا که به ذخیره حجم زیادی از session state نیاز دارند، می‌باشد.

## Application State

Application state، به شما اجازه ذخیره اشیای عمومی که می‌توانند توسط هر کلاینتی ذخیره شوند را می‌دهد. بر اساس کلاس System.Web.HttpApplicationState، State می‌باشد.

Session State، شبیه Application State می‌باشد. همان اشیا را پشتیبانی می‌کند و داده‌ها را نیز سمت سرور نگهداری خواهد کرد. یک نمونه رایج از استفاده Application state، می‌تواند یک شمارنده کلی که تعداد دفعات اجرای یک عملیات (توضیح کلیه کلاینت‌های برنامه تحت وب) را می‌شمارد، باشد.

برای نمونه می‌توانید یک event handler در فایل global.asax بنویسید که تعداد session های ایجاد شده یا تعداد درخواست‌های دریافت شده در برنامه را بشمارد. می‌توانید همین منطق را در رویداد Page.Load برای شمارش تعداد درخواست‌های رسیده به این صفحه (از طریق کلاینت‌های مختلف) بکار ببرید.

```
protected void Page_Load(Object sender, EventArgs e)
{
    // Retrieve the current counter value.

    int count = 0;

    if (Application[“HitCounterForOrderPage”] != null)
    {
        count = (int)Application[“HitCounterForOrderPage”];
    }
}
```

```
// Increment the counter.

count++;

// Store the current counter value.

Application[“HitCounterForOrderPage”] = count;

lblCounter.Text = count.ToString();

}
```



```
protected void Page_Load(Object sender, EventArgs e)

{

    // Acquire exclusive access.

    Application.Lock();

    int count = 0;

    if (Application[“HitCounterForOrderPage”] != null)

    {

        count = (int)Application[“HitCounterForOrderPage”];

    }

    count++;

    Application[“HitCounterForOrderPage”] = count;

    // Release exclusive access.

    Application.Unlock();

    lblCounter.Text = count.ToString();

}
```

در حقیقت Application State، به ندرت در .NET استفاده می‌شود، زیرا دو کاربرد رایج آن، توسط متدهای ساده‌تر و مؤثرتری جایگزین شده است:

- در گذشته، application state، برای ذخیره مقادیر ثابت در برنامه، مانند رشته اتصال بانک اطلاعاتی استفاده می‌شد. همانطور که در بخش‌های قبلی مشاهده کردید، این نوع از مقادیر ثابت اکنون می‌توانند در فایل web.config ذخیره شوند که بسیار قابل انعطاف‌تر می‌باشد.

- همچنین می‌تواند برای ذخیره اطلاعاتی که به مراتب مورد استفاده قرار می‌گیرند (مانند کاتالوگ کامل

محصول که نیازمند جستجو در بانک اطلاعاتی می‌باشد) نیز استفاده شود. استفاده از application state مشکلاتی نظریه اینکه آیا این داده‌ها معتبر می‌باشند یا چگونگی جایگزین نمودن آنها در موقع لزوم، را به همراه دارد. اگر کاتالوگ محصولات بزرگ باشد، کارایی کاهش پیدا می‌کند. اگرچه این نوع از داده‌ها را در Cache ذخیره کنید.

 نکته: اگر می‌خواهید از application state استفاده کنید، می‌توانید محتوای آن را در هنگام اولین شروع برنامه، مقدار دهی کنید. این کدها را در متدهای Application\_OnStart() در فایل Global.asax قرار دهید.

## مقایسه گزینه‌های مدیریت State

هر کدام از گزینه‌های مطرح شده، دارای زمان حیات، کارایی، ناحیه و سطح پشتیبانی مختلفی می‌باشند.



Custom Cookies	QueryString	ViewState	
داده رشته‌ای	تعداد محدودی از داده‌های رشته‌ای	کلیه انواع داده‌ای قابل سریالیز شدن در دات نت	انواع داده‌ای مجاز
کامپیوتر کلاینت (در حافظه یا در یک فایل متنی کوچک، بر اساس تنظیمات چرخه حیات آن)	URL مرورگر	یک hidden filed در صفحه کنونی	محل ذخیره سازی
توسط برنامه نویس تعیین می‌شود. می‌تواند در چندین صفحه استفاده شود و در خلال مشاهده آنها باقی بماند.	اگر کاربر یک آدرس جدید وارد کند یا مرورگر را ببندد از بین می‌رود. اگرچه می‌تواند در bookmark باقی ماندن دائمی در postback های یک صفحه		مدت زمان حیات
کل برنامه ASP.NET	به صفحه مقصد محدود می‌شود.	به صفحه کنونی محدود می‌شود.	ناحیه
غیر امن است می‌تواند توسط کاربر تغییر کند.	به راحتی در دسترس کاربر برای انجام هرگونه تغییری می‌باشد.	به سادگی خوانده می‌شود. می‌توانید برای آن از encryption استفاده کنید.	امنیت
حجم داده اهمیتی ندارد	حجم داده اهمیتی ندارد	در صورتیکه مقدار زیادی از داده ذخیره شده باشد، کارایی آن پایین است ولی روی کارایی سرور تأثیری ندارد.	تأثیر روی کارایی
شخصی سازی تنظیمات برای یک وب سایت	ارسال شناسه یک محصول از صفحه کاتالوگ به صفحه جزئیات	تنظیمات خاص صفحه	یک نمونه کاربرد

Application State	Session State	
همه انواع داده‌ای دات نت	. in-process	انواع داده‌ای مجاز

Server memory	.SQL Server Server memoty, state server بر اساس مد انتخابی شما	محل ذخیره سازی
مدت زمان حیات برنامه (معمولًاً تا زمانی که سرور reboot شود)	پس از دوره از پیش تعريف شده، (معمولًاً ۲۰ دقیقه)، Time Out می‌شود.	مدت زمان حیات
کل برنامه ASP.NET . برخلاف سایر روش ها، داده برنامه برای همه کاربران در دسترس است.	کل برنامه ASP.NET	ناحیه
دارای امنیت بالا می‌باشد زیرا داده هرگز به کلاینت منتقل نمی‌شود.	دارای امنیت بالا می‌باشد زیرا داده هرگز به کلاینت منتقل نمی‌شود.	امنیت
در زمان ذخیره سازی حجم زیادی از داده ها، کارایی پایین داشته باشیم، زیرا هر کاربر دارای نسخه کپی شده خود از داده ها هرگز time out ندارند و حذف نمی‌شوند.	در زمان ذخیره سازی حجم زیادی از داده ها، کارایی پایین داشته باشند، مخصوصاً اگر چندین کاربر در یک زمان واحد داده ها هرگز time out ندارند و داده session می‌باشد.	تأثیر روی کارایی
ذخیره سازی عمومی هر نوع از داده.	ذخیره سازی آیتم ها درون یک سبد خرید.	یک نمونه کاربرد

 نکته : ASP.NET، دارای نوع دیگری از state management می‌باشد. پروفایل ها به شما اجازه ذخیره و بازیابی اطلاعات خاص کاربر از بانک اطلاعاتی را خواهد داشت.



همه آنها بیخ که عاشق سفر هستند، نمرد ها نند  
«جس. آر. آر. تاللین»

## بخش دوم: ساخت فرم‌های وب بیتتر

Validation – فحیل ششم: اعتبارسنجی

– فحیل هفتم: کنترل مانی در فرم‌ها

User Control – فحیل هشتم

Master Page , Styles, Themes – فحیل نهم



## ساخت صفحات وب بهتر

### تعیین اعتبار

در این بخش به معرفی برخی از مفیدترین کنترل‌های موجود در ASP.NET، با نام کنترل‌های اعتبارسنجی، خواهیم پرداخت. این کنترل‌ها، داده‌های ورودی کاربران را ارزیابی و خطاهای را گزارش می‌دهند و این کار را خودکار می‌سازند.

### آشنایی با اعتبار سنجی

در زیر، اشتباهات رایجی که یک کاربر می‌تواند در هنگام استفاده از فرم انجام دهد را مشاهده می‌کنید:

- کاربر ممکن است، یک فیلد مهم را نادیده بگیرد و آن را خالی بگذارد.
- اگر مقادیر خالی را مجاز نمی‌دانید، کاربر ممکن است یک متن الکترونیک اجباری باشد و کاربر یک پست غیر معتبر وارد نماید ممکن است برای برنامه ارسال خودکار email شما مشکل درست شود.
- کاربر ممکن است اشتباهات واضحی مانند ورود کاراکترهای غیر عددی در یک فیلد عددی بکند یا نوع غلطی از اطلاعات را وارد نماید.
- برخی از کاربران خرابکار ممکن است بخواهند خطای خاصی را برمی‌گرداند ایجاد نمایند. برای نمونه ممکن است که از SQL injection استفاده نمایند.

### کنترل‌های ارزیابی

ASP.NET، دارای پنج کنترل اعتبارسنجی که در جدول زیر می‌بینید، می‌باشد.



توضیحات	کلاس کنترل
تا زمانی که کنترل ورودی شامل هیچ رشته خالی نباشد، اعتبار سنجی با موفقیت روبرو می‌شود.	RequiredFieldValidator
اگر کنترل ورودی شامل یک مقدار عددی، الفبایی یا تاریخ باشد، اعتبارسنجی موفقیت آمیز است.	RangeValidator
اگر داده موجود در کنترل ورودی با یک عبارت منظم منطبق باشد، اعتبارسنجی موفقیت آمیز است.	RegularExpressionValidator
اگر داده ورودی با داده موجود در کنترل دیگر برابر باشد، اعتبارسنجی موفقیت آمیز است.	CompareValidator
اعتبار سنجی توسط متده تعريف شده توسط کاربر انجام می‌شود.	CustomValidator

هر کنترل اعتبار سنجی، می‌تواند به یک کنترل ورودی تنها متصل شود. علاوه بر این، می‌توانید بیش از یک کنترل اعتبارسنجی را برای یک کنترل تعیین کنید. (برای اعتبارسنجی‌های مختلف روی یک کنترل)

## اعتبار سنجی سمت سرور

از کنترل‌های validator، می‌توانید برای تعیین اعتبار خودکار یک صفحه در هنگامی که یک کاربر آن را submit می‌کند، استفاده کنید. هر دکمه در صفحه دارای یک ویژگی CausesValidation می‌باشد که می‌تواند با مقدار true یا false، مقدار دهی شود. زمانی که کاربر روی دکمه کلیک می‌کند بر اساس مقدار این ویژگی اتفاق‌های زیر رخ خواهد داد:

- اگر مقدار آن false باشد، ASP.NET کنترل اعتبارسنجی را نادیده می‌گیرد و صفحه Postback می‌شود و کدهای شما اجرا خواهند شد.

- اگر مقدار آن true باشد، در هنگامی که کاربر روی دکمه کلیک می‌کند، بصورت خودکار صفحه را اعتبارسنجی می‌کند و این کار را برای همه کنترل‌های روی صفحه انجام می‌دهد.

اگر کنترلی در صفحه با موفقیت تعیین اعتبار نشود، ASP.NET، صفحه‌ای را به همراه چند خطاب بر اساس تنظیمات شما برمی‌گرداند. کدهای صفحه شما بر اساس تعیین اعتبار صفحه، ممکن است اجرا شوند یا اجرا نشوند.

## اعتبار سنجی سمت کلاینت

در مرورگرهای جدید، بصورت خودکار کدهای جاوا اسکریپت برای اعتبارسنجی سمت کلاینت را اضافه می‌کند. زمانی که کاربر روی دکمه ای باعث اعتبارسنجی می‌شود کلیک می‌کند، همان خطاهای دیده می‌شود، بدون آنکه نیاز به submit شدن و برگرداندن صفحه از سرور باشد. این کار پاسخگویی صفحه وب شما را افزایش می‌دهد.

به هر حال حتی اگر اعتبارسنجی صفحه در سمت کلاینت موفقیت آمیز باشد، ASP.NET، هنوز در هنگام دریافت آن در سمت سرور، اعتبارسنجی خود را انجام می‌دهد. این کار به این دلیل است که کاربران حرفه‌ای ممکن است بتوانند اعتبارسنجی سمت کلاینت را دور بزنند. برای نمونه، یک کاربر خرابکار ممکن است کدهای اعتبارسنجی جاوا اسکریپت را از صفحه پاک و کار خود را با صفحه ادامه دهد.

## HTML5 Validation

HTML5، جدیدترین نسخه از زبان HTML می‌باشد و ویژگی‌های اعتبارسنجی سمت کلاینت خوبی که برای گرفتن اشتباهات و کم کردن خطاهای مناسب است را ارائه داده است. مشکل استفاده از HTML5 این است با همه مرورگرها سازگاری کاملی ندارد. اعتبارسنجی HTML5 همانند اعتبارسنجی بر اساس جاوا اسکریپت می‌باشد. برای نمونه اگر، کاربر متنی را درون یک فیلد عددی وارد کند، مرورگر مشکل را تشخیص داده و از submit شدن صفحه جلوگیری می‌کند. سپس در کنار فیلد مشکل دار، پیام خطاب را نمایش می‌دهد.

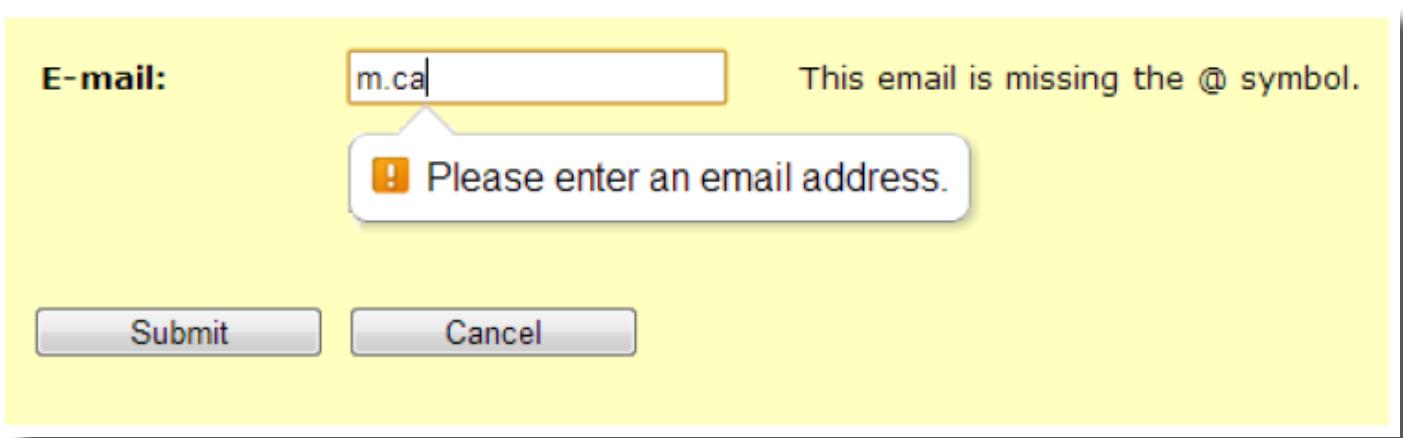
شما می‌توانید در صفحات وب ASP.NET از اعتبارسنجی با استفاده از HTML5 بهره‌مند شوید و به دو دلیل زیر این انتخاب بهترین گزینه نیست:

- اعتبارسنجی HTML5 یک راه حل کامل نیست. در مرورگرهایی که HTML5 را پشتیبانی نمی‌کنند مجبوریت از کد JS استفاده نمایید. در صورتیکه اعتبارسنجی کنترل‌های ASP.NET، با همه مرورگرها سازگاری دارد.
- اعتبارسنجی HTML5، تنها روی المان‌های <input> کار می‌کند و روی وب کنترل‌ها کار نمی‌کند.

ASP.NET یکی از ویژگی های اعتبارسنجی HTML5 را پشتیبانی می کند : خصیصه ای با نام type، که به شما اجازه ایجاد text box هایی که برای نوع خاصی از داده ها تعیین شده اند را می دهد. (مانند مقادیر عددی یا نشانی پست الکترونیک). این خصیصه را می توانید از طریق ویژگی TextBox.TextMode تعیین کنید.

```
<asp:TextBox id="txtAge" runat="server" TextMode="Number" />
```

البته شما می توانید از این ویژگی به همراه کنترل های validation در یک صفحه استفاده نمایید:



## استفاده از کنترل های اعتبارسنجی

این کنترل ها در فضای نام System.Web.UI.WebControls یافت می شوند :



ویژگی	توضیحات
ControlToValidate	کنترلی که اعتبارسنجی روی آن انجام می شود را تعیین می کند. هر validator، می تواند مقدار موجود در یک کنترل را چک کند. برای اجرای بیش از یک نوع error checking، باید از چندیم validator روی یک کنترل استفاده نمایید.
ErrorMessage ForeColor	اگر اعتبارسنجی با شکست روبرو شود، کنترل validator، می تواند یک متن (که با ویژگی ErrorCodear دارای دهی شده است) را نمایش دهد. با تغییر رنگ (ForeColor) می توانید پیام را مشخص تر نمایید.
Display	به شما اجازه می دهد پیام خطرا را یا بصورت پویا (در زمان نیاز) درون صفحه قرار دهید یا در محل ثابت مناسبی از صفحه نمایش دهید. روش پویا یا داینامیک زمانی مناسب است که شما چندین validator را در کنار هم قرار داده باشید. بنابراین لزومی به تعیین فضای مورد نیاز توسط شما نمی باشد. روش ثابت استاتیک زمانی مناسب است که درون جدول باشد و شما نمی خواهید در هنگامی که پیام خطایی وجود ندارد عرض سلول جدول را collapse کنید.

پس از اجرای اعتبارسنجی، این ویژگی بر اساس اینکه آیا نتیجه موفقیت آمیز بوده یا با شکست روبرو شده است، true یا false می‌باشد. بصورت کلی، به جای جستجوی اینکه آیا همه کنترل‌های اعتبارسنجی موفق بوده اند یا نه، می‌توان از وضعیت کل صفحه با این ویژگی آگاه شد.	IsValid
زمانی که با false مقدار دهی شود، اعتبارسنجی خودکار در زمان submit شدن صفحه انجام نخواهد شد.	Enabled
اگر با true مقدار دهی شود، اعتبارسنجی JS و DHTML به صفحه شما اضافه می‌کنند تا اعتبارسنجی سمت کلاینت در مرورگرهایی که آنها را پشتیبانی می‌کنند انجام شود.	EnableClientScript

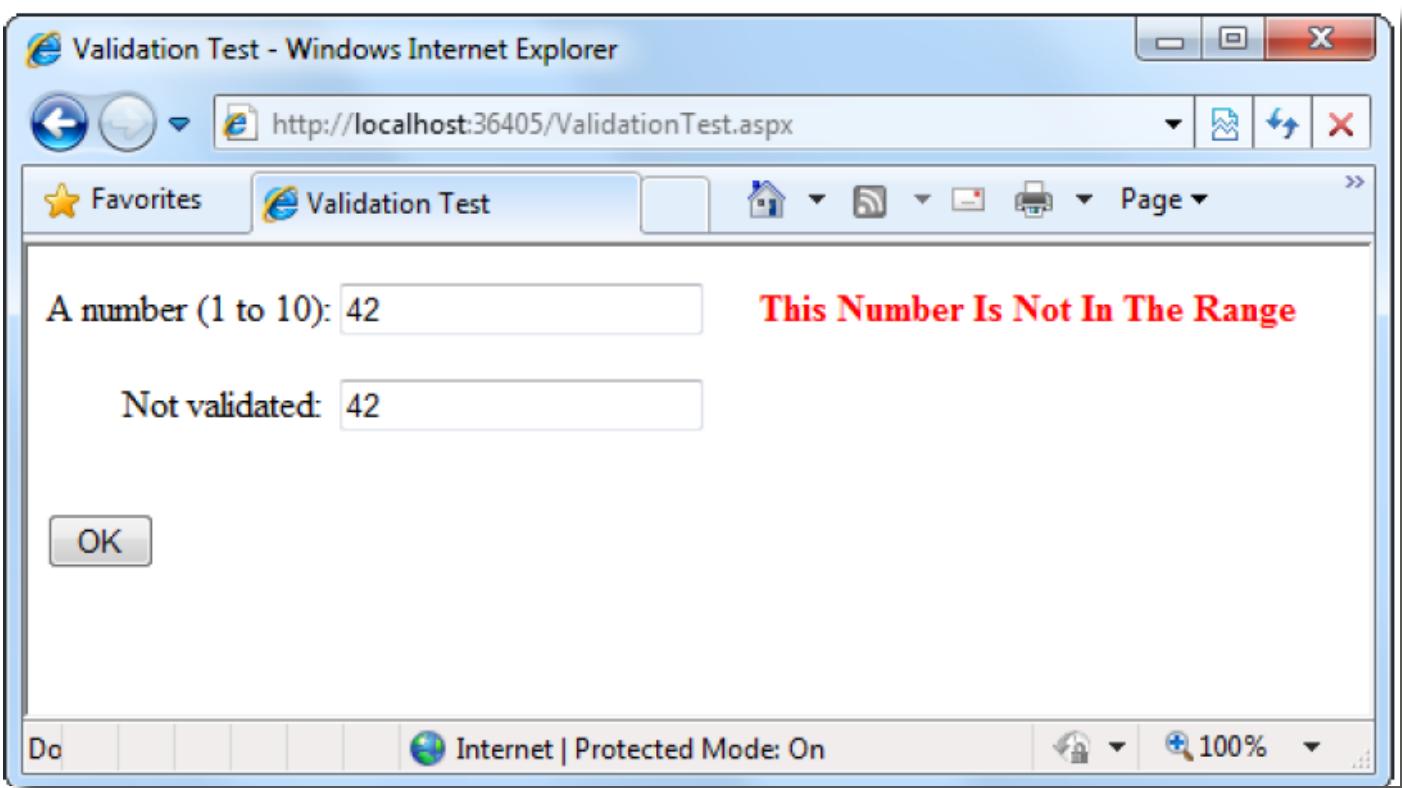
زمانی که با validation control ها کار می‌کنید، تنها ویژگی که نیاز دارید، ErrorMessage و ControlToValidate هستند. علاوه بر این، ممکن است نیاز به استفاده از ویژگی هایی که برای validator ها خاصی بکار می‌رود داشته باشید.



اعضای اضافه شده	Validator Control
هیچ	RequiredFieldValidator
MaximumValue, MinimumValue, Type	RangeValidator
ControlToCompare, Operator, Type, ValueToCompare	CompareValidator
ValidationExpression	RegularExpressionValidator
ClientValidationFunction, ValidateEmptyText, ServerValidate event	CustomValidator

## یک مثال کاربردی ساده

در این مثال از یک دکمه button، دو textbox و یک rangevalidator control استفاده شده است. کنترل first textbox اول را اعتبارسنجی خواهد کرد، استفاده از این ویژگی می‌کنیم. اگر اعتبارسنجی fail شود، کنترل second rangevalidator یک پیام خطای نمایش می‌دهد. بنابراین باید این کنترل را در کنار second textbox قرار دهید. دوم از هیچ کنترل اعتبارسنجی استفاده نمی‌کند.



علاوه بر این یک کنترل label در انتهای فرم قرار می دهیم که نشان می دهد آیا فرم postback شده و کدهای اجرا شده اند یا خیر. ویژگی EnableViewState آن را غیر فعال کنید تا مطمئن شوید که با هر بار با postback محتوای آن پاک می شود.

A number (1 to 10):

```
<asp:TextBox ID="txtValidated" runat="server" />

<asp:RangeValidator ID="RangeValidator" runat="server"
    ErrorMessage="This Number Is Not In The Range"
    ControlToValidate="txtValidated" MaximumValue="10" MinimumValue="1" ForeColor="Red"
    Font-Bold="true" Type="Integer" />

<br />
<br />
```

Not validated:

```
<asp:TextBox ID="txtNotValidated" runat="server" /><br />
```

```

<br />
<asp:Button ID="cmdOK" runat="server" Text="OK" OnClick="cmdOK_Click" />
<br />
<br />
<asp:Label ID="lblMessage" runat="server" EnableViewState="False" />

```

کد زیر نیز به رویداد کلیک دکمه پاسخ می‌دهد:

```

protected void cmdOK_Click(Object sender, EventArgs e)
{
    lblMessage.Text = "cmdOK_Click event handler executed.";
}

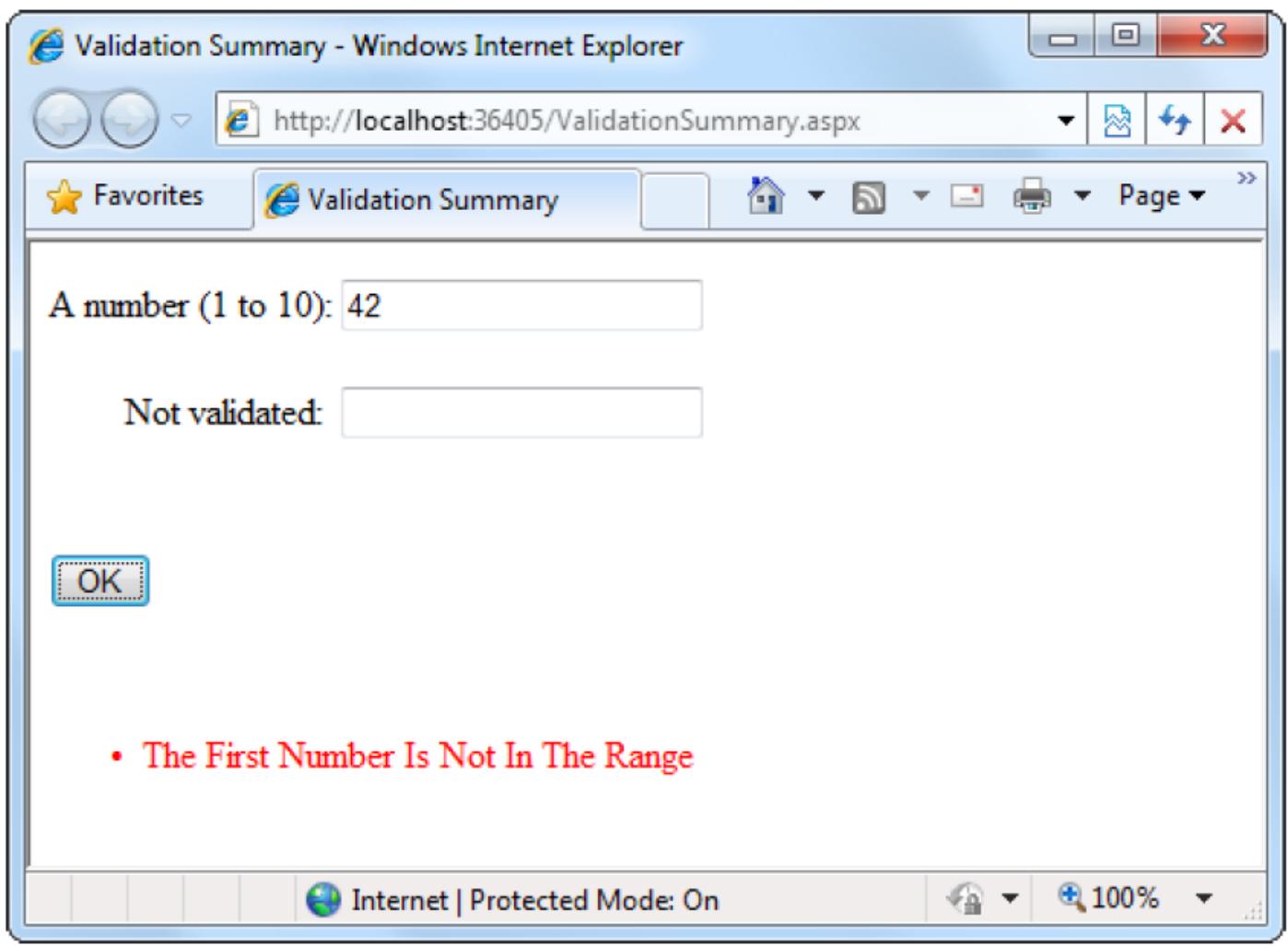
```

## سایر گزینه‌های نمایش پیام

با استفاده از کنترل ValidationSummary، می‌توانید نمایش پیام‌های خطای خطا را مدیریت کنید. برای این کار می‌توانید ویژگی Display از کنترل RangeValidator را با None مقداردهی کنید. بنابراین پیام خطای دیگر نمایش داده نخواهد شد. البته اعتبارسنجی کماکان انجام می‌شود و کاربر هنوز در صورت ورود اطلاعات نادرست، نمی‌تواند روی دکمه OK کلیک کند. سپس کنترل ValidationSummary را در مکان مناسبی قرار دهید (مانند انتهای صفحه).

```
<asp:ValidationSummary id="Errors" runat="server" ForeColor="Red" />
```

زمانی که صفحه را باز کنید، هیچ پیام داینامیکی در زمان ورود اطلاعات و یا پرسش به کنترل دیگر، مشاهده نخواهید کرد. هنگامی که روی OK کلیک می‌کنید، کنترل ValidationSummary ظاهر شده و لیستی از کلیه پیام‌های خطای خطا را نشان می‌دهید.



زمانی که لیست خطاها نمایش داده می‌شود، `validationsummary`، بصورت خودکار مقدار ویژگی `ErrorMessage` هر کنترل را بازیابی می‌کند. ممکن است بخواهید پیام کاملی در لیست به همراه یک نشانگر خطا در کنار کنترل مربوطه نمایش دهید. با استفاده از ویژگی `Text` کنترل `Validator`، می‌توانید این کار را انجام دهید. بصورت عادی این ویژگی خالی می‌باشد.

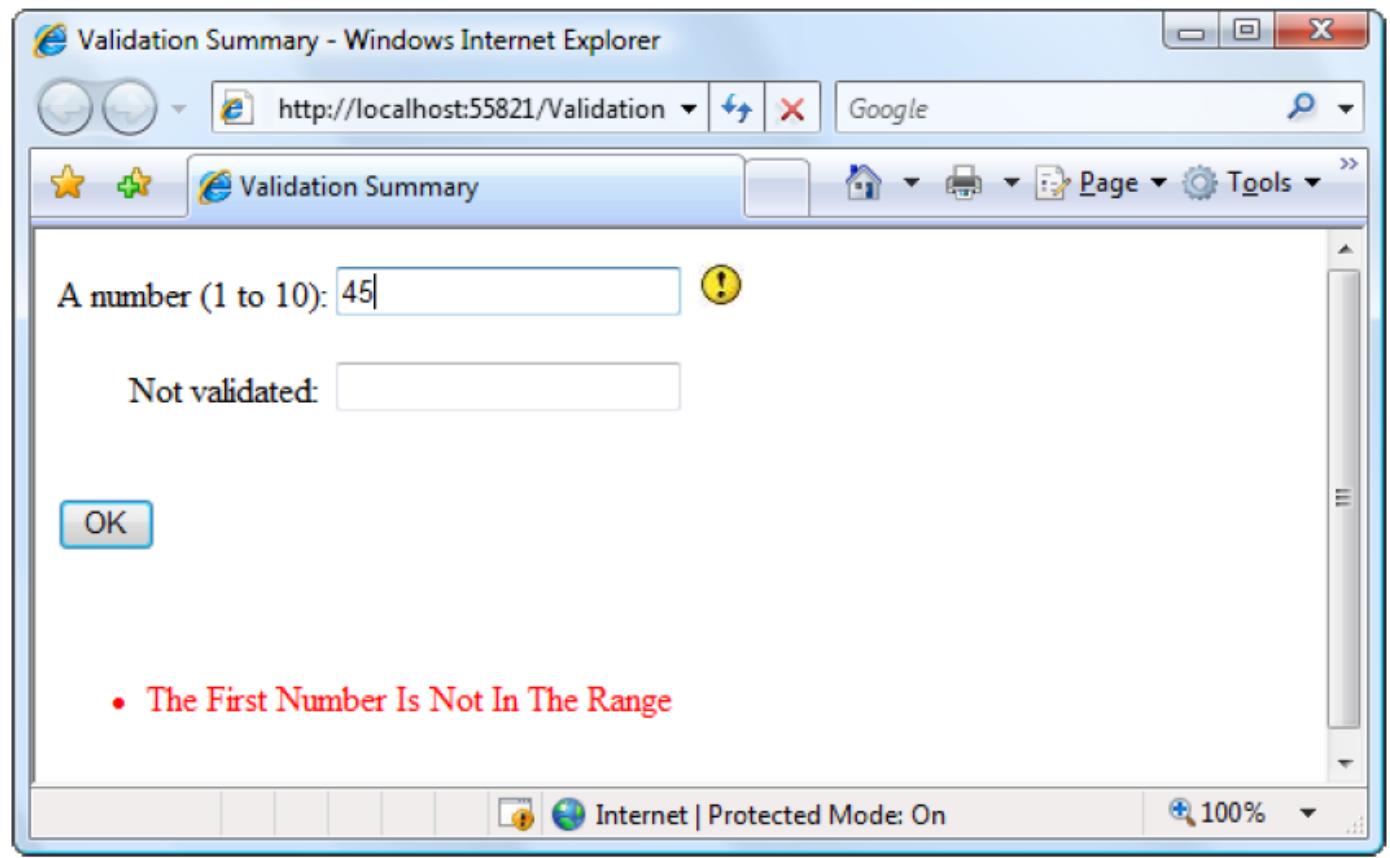
```
<asp:RangeValidator ID="RangeValidator" runat="server" Text="">
    ErrorMessage="The First Number Is Not In The Range"
    ControlToValidate="txtValidated" MaximumValue="10" MinimumValue="1" Type="Integer" />
```

اگر متن بیشتری برای ویژگی `Text` دارید، می‌توانید آن را درون تگ‌های این کنترل قرار دهید:

```
<asp:RangeValidator ID="RangeValidator" runat="server" ControlToValidate="txtValidated">
    MaximumValue="10" MinimumValue="1" ErrorMessage="The First Number Is Not In The Range"
    Type="Integer"><b>*** Error</b></asp:RangeValidator>
```

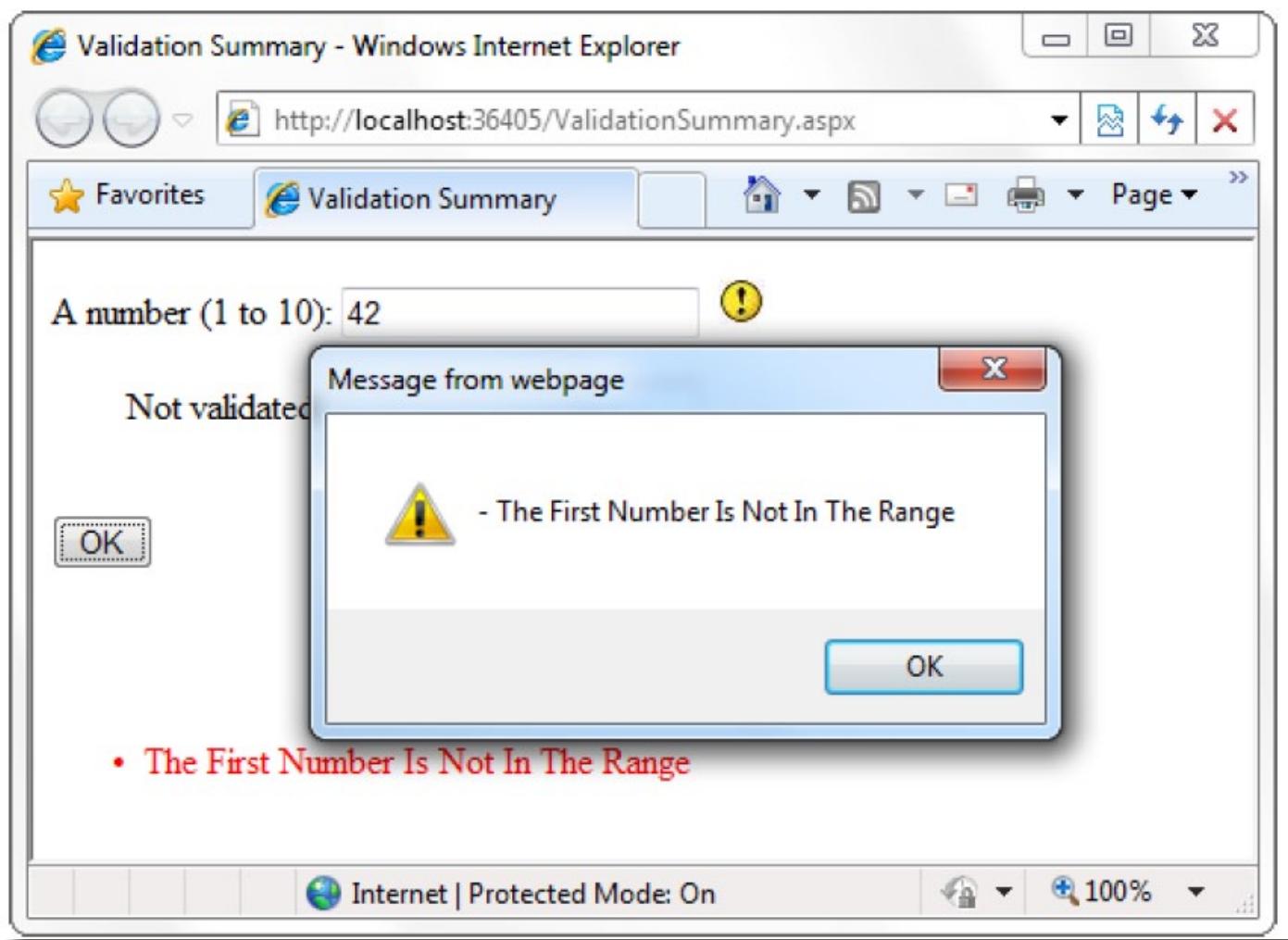
همچنین می‌توانید از آیکن‌های مختلف نیز استفاده کنید:

```
<asp:RangeValidator ID="RangeValidator" runat="server">
    
</asp:RangeValidator>
```



می‌توانید از ویژگی‌های `DisplayMode` و `ForeColor`، `HeaderText` موجود در این کنترل استفاده نمایید.

همچنین می‌توانید تعیین کنید این کنترل بصورت یک کادر pop-up نمایش داده شود. برای این منظور، ویژگی `ShowErrorMessage` را با `true` مقدار دهی نمایید.



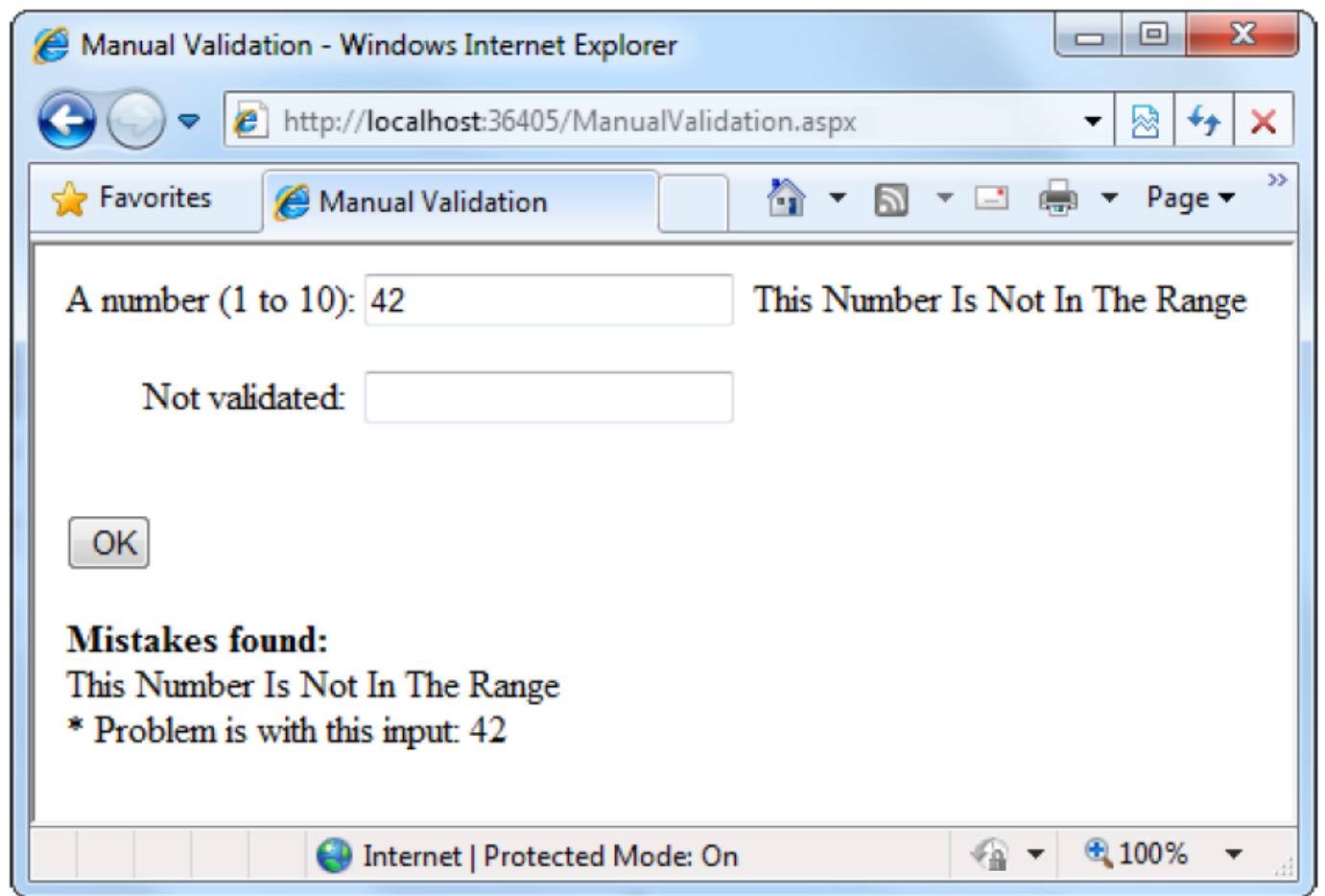
## اعتبارسنجی دستی

می‌توانید اعتبارسنجی را غیر فعال کنید و با کمک گرفتن از کنترل‌های Validation، خودتان آن را انجام دهید. این کار برای ایجاد پیام‌های خطاب برای سایر کنترل‌ها مورد استفاده قرار می‌گیرد.

با استفاده از سه روش زیر می‌توانید اعتبارسنجی دستی را انجام دهید :

- از کدهای خود برای تعیین اعتبار مقادیر استفاده کنید. در این روش به هیچ یک از کنترل‌های validation نیازی نخواهد داشت.
- ویژگی EnableClientScript را برای همه کنترل‌ها غیرفعال کنید. این کار اجازه submit شدن یک صفحه غیر معتبر را می‌دهد. می‌توانید تصمیم بگیرید با مشکل مربوطه چه کاری انجام دهید.
- یک دکمه که ویژگی CausesValidation آن برابر با false می‌باشد را اضافه نمایید. زمانی که این دکمه کلیک شود، با استفاده از متدهای Page.Validate()، صفحه را بصورت دستی اعتبارسنجی کنید. سپس تصمیم بگیرید که چه کاری باید انجام شود.

مثال زیر از روش دوم استفاده کرده است. پس از submit شدن صفحه، کلیه کنترل‌های Validation را امتحان می‌کند.



```

protected void cmdOK_Click(Object sender, EventArgs e)
{
    string errorMessage = "<b>Mistakes found:</b><br />";
    // Search through the validation controls.

    foreach (BaseValidator ctrl in thisValidators)
    {
        if (!ctrl.IsValid)
        {
            errorMessage += ctrl.ErrorMessage + "<br />";

            // Find the corresponding input control, and change the
            // generic Control variable into a TextBox variable.

            // This allows access to the Text property.

            TextBox ctrlInput =
                (TextBox)this.FindControl(ctrl.ControlToValidate);
        }
    }
}

```

```

        errorMessage += " * Problem is with this input: ";
        errorMessage += ctrlInput.Text + "<br />";
    }

}

lblMessage.Text = errorMessage;
}

```

این مثال از متده با نام `Page.FindControl()` استفاده کرده است. ویژگی `validator` یک `string` است که `ControlToValidate` هر کنترل فراهم می‌کند. برای پیدا کردن کنترلی که با این نام منطبق باشد (و بازیابی ویژگی `Text` آن)، باید از متده `FindControl` استفاده کنید. پس از بازیابی `textbox` منطبق با آن نام، کد می‌تواند کارهای دیگری مانند پاک کردن محتوا یا تغییر رنگ آن، انجام دهد.

## اعتبارسنجی با عبارت منظم (Regular Expression)

کنترل `RegularExpressionValidator`، متن را بر اساس اینکه آیا با الگوی خاصی تطبیق دارد یا نه ارزیابی می‌کند. برای نمونه، یک شماره تلفن باید مجموعه ای از اعداد باشد و یک پست الکترونیک باید شامل دقیقاً یک کاراکتر `@` باشد و نام فایل نمی‌تواند شامل کاراکترهای خاصی مانند `\` و `?` باشد.

تمامی `regular expression` ها دارای دو نوع از کاراکترها می‌باشند : `literal` ها و `metacharacter` ها. `Literal` ها، کاراکترهای تعریف شده خاصی را ارائه می‌دهند. برای نمونه اگر به دنبال `a` string literal ای مانند `"a"` باشید، تنها کاراکتر `a` را می‌باید و نه چیز دیگری.

Metacharacter ها، کاراکترهای خاصی مانند `( )`, `*`, `?` هستند :

`Del *.*`

دستور بالا در Dos بکار می‌رفت. عبارت `*.` شامل یک `literal` و دو `metacharacter` می‌باشد. ترجمه عبارت بالا می‌شود "کلیه فایل هایی که با هر تعداد از کاراکترها شروع شده و دارای پسوندی با هر تعداد از کاراکتر می‌باشند را حذف کن".

متاکاراکتر بعدی، علامت `?` می‌باشد که شامل تک کاراکترها می‌شود.

`Del hello.?`

معنی عبارت بالا : کلیه فایل‌ها که نام آنها `hello` می‌باشد و دارای پسوند تک کاراکتری می‌باشند را حذف کن.

\S : شامل هر کاراکتر whitespace (مانند space و tab) می‌شود.

\d : شامل هر کاراکتر عددی می‌شود.

مثال زیر با هر رشتہ ای که با عدد ۳۳۳ شروع می‌شود و سپس یک whitespace و سه عدد دیگر را در خود دارد، منطبق است.

+ : برای کاراکترهای تکراری از این علامت استفاده می‌شود.

برای مثال عبارت ۷۵+، با هر عبارتی که با یک یا چند عدد ۵ شروع شود و فقط به یک عدد ۷ ختم شود منطبق می‌باشد. بنابراین با عدد ۵۷ و ۵۵۵۵۷ مطابقت دارد.

]] : باز ها را می‌توانید با این علامت تعریف کنید.

برای نمونه [a-f]، با هر کاراکتر تنها از a تا f (با حروف کوچک) منطبق است.

برای نمونه عبارت زیر با پست‌های الکترونیک تطبیق دارد :

.+@.+

\D : هر کاراکتری به جز عدد را شامل می‌شود.

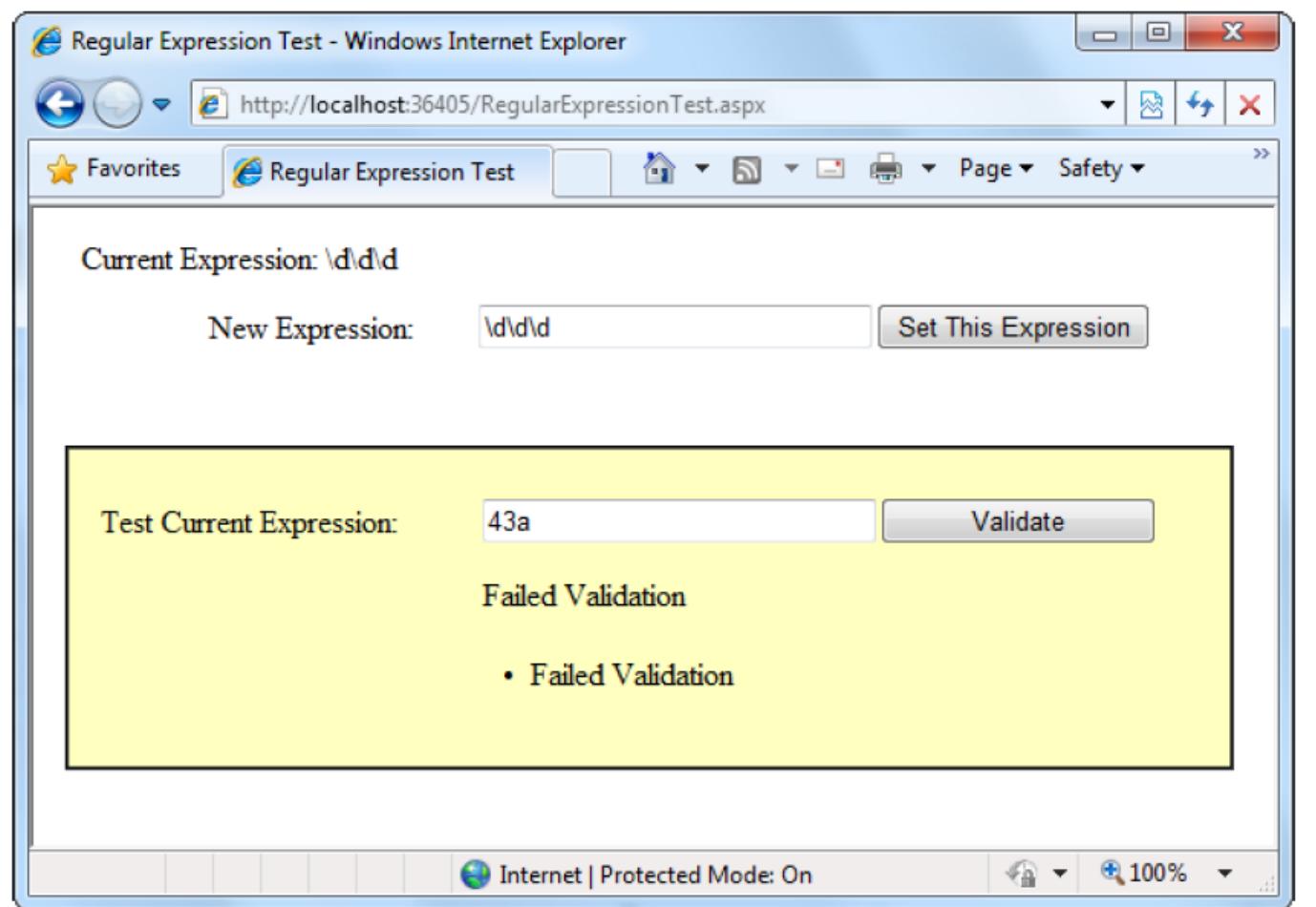
\S : هر کاراکتری به جز whitespace را شامل می‌شود.

\W : هر کاراکتر کلمه‌ای را شامل می‌شود (حرف، عدد و زیر خط)

\W : هر چیزی به غیر از کاراکترهای کلمه‌ای



توضیحات	Regular expression	محتوا
عبارت را باری تنها یک @ و نقطه چک می‌کند و تنها کاراکترهای به غیر از whitespace را مجاز می‌شمارد.	\S+@\S+\.\S+	پست الکترونیک
هر ترتیبی از یک یا چند کاراکتر	\w+	کلمه عبور
کلمه عبوری که حداقل ۴ کاراکتر و حداقل ۱۰ کاراکتر داشته باشد.	\w{4,10}	کلمه عبور با طول ثابت
مانند بالا، این عترت تنها شامل کلمات عبور ۴ تا ۱۰ کاراکتری می‌شود. کاراکتر اول باید در بازه a-z یا A-Z باشد.	[a-zA-Z]\w{3,9}	کلمه عبور پیشرفته



```

public partial class RegularExpressionTest : System.Web.UI.Page
{
    protected void cmdSetExpression_Click(Object sender, EventArgs e)
    {
        TestValidator.ValidationExpression = txtExpression.Text;
        lblExpression.Text = "Current Expression: ";
        lblExpression.Text += txtExpression.Text;
    }
}

```

## مثال کاربردی : فرم مشتری اعتبارسنجی شده

The screenshot shows a Microsoft Internet Explorer window titled "Customer Form - Windows Internet Explorer". The URL in the address bar is "http://localhost:36405/CustomerForm.aspx". The page displays a form with six fields: "User Name", "Password", "Password (retype)", "E-mail", "Age", and "Referrer Code". Each field has an associated validation message to its right:

- User Name:** You must enter a user name.
- Password:** Your password does not match.
- Password (retype):** This email is missing the @ symbol.
- E-mail:** This age is not between 0 and 120.
- Age:** Try a string that starts with 014.
- Referrer Code:**

At the bottom of the form are two buttons: "Submit" and "Cancel".

```

<form id="Form1" runat="server">

    <asp:RequiredFieldValidator ID="vldUserName" runat="server"
        ErrorMessage="You must enter a user name." ControlToValidate="txtUserName" />

    <asp:RequiredFieldValidator ID="vldPassword" runat="server"
        ErrorMessage="You must enter a password." ControlToValidate="txtPassword" />

    <asp:CompareValidator ID="vldRetype" runat="server" ErrorMessage="Your password does not
        match." ControlToCompare="txtPassword" ControlToValidate="txtRetype" />

    <asp:RequiredFieldValidator ID="vldRetypeRequired" runat="server"
        ErrorMessage="You must confirm your password." ControlToValidate="txtRetype" />

```

```

<asp:RegularExpressionValidator ID="vldEmail" runat="server"
    ErrorMessage="This email is missing the @ symbol."
    ValidationExpression=".+@.+ " ControlToValidate="txtEmail" />

<asp:RangeValidator ID="vldAge" runat="server" ErrorMessage="This age is not between 0 and
120." Type="Integer" MinimumValue="0" MaximumValue="120" ControlToValidate="txtAge" />

<asp:CustomValidator ID="vldCode" runat="server"
    ErrorMessage="Try a string that starts with 014."
    ValidateEmptyText="False" OnServerValidate="vldCode_ServerValidate"
    ControlToValidate="txtCode" />

<asp:Button ID="cmdSubmit" runat="server" OnClick="cmdSubmit_Click" Text="Submit">
</asp:Button>

<asp:Button ID="cmdCancel" runat="server" CausesValidation="False" OnClick="cmdCancel_
Click"
    Text="Cancel"></asp:Button>

protected void cmdSubmit_Click(Object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblMessage.Text = "This is a valid form.";
    }
}

protected void cmdCancel_Click(Object sender, EventArgs e)
{
}

```

```

lblMessage.Text = "No attempt was made to validate this form.";
}

protected void vldCode_ServerValidate(Object source, ServerValidateEventArgs e)
{
    try
    {
        // Check whether the first three digits are divisible by seven.

        int val = Int32.Parse(e.Value.Substring(0, 3));

        if (val % 7 == 0)

        {
            e.IsValid = true;
        }
        else

        {
            e.IsValid = false;
        }
    }
    catch
    {
        // An error occurred in the conversion.

        // The value is not valid.

        e.IsValid = false;
    }
}

```

همانطور که مشاهده می‌کنید، اعتبارسنجی سمت سرور شما تا زمانی که صفحه postback نشده باشد نمی‌تواند استفاده شود. این یعنی که اگر شما client script code را فعال نمایید می‌توانید در سمت کلاینت این اعتبارسنجی را انجام دهید.

برای این منظور می‌توانید از CustomValidator.ClientValidationFunction به بخش

صفحه وب خود اضافه نمایید. متدهای JS شما دو پارامتر می‌گیرد، (در استایل صحیح دات نت) که منبع رویداد و پارامترهای اضافی اعتبارسنجی را تعیین می‌کند.

کدهای JS زیر را می‌توانید در بخش `<head>` صفحه قرار دهید. (یا آنها را در یک فایل JS قرار داده و با استفاده از تگ `<script>` در بخش `<head>` صفحه به آن اشاره کنید)

سپس مقدار ClientValidationFunction را برابر با نام متدهای JS خود قرار دهید.

```

<script type="text/javascript">

    function MyCustomValidation(objSource, objArgs) {

        // Get value.

        var number = objArgs.Value;

        // Check value and return result.

        number = number.substr(0, 3);

        if (number % 7 == 0) {

            objArgs.IsValid = true;

        }

        else {

            objArgs.IsValid = false;

        }

    }

</script>

<asp:CustomValidator

    ID="vldCode"

    runat="server"

    ErrorMessage="Try a string that starts with 014."

    ControlToValidate="txtCode"

    OnServerValidate="vldCode_ServerValidate"

    ClientValidationFunction="MyCustomValidation" />

```

 نکته: حتی اگر از client-side validation استفاده می‌کنید، باید در کدهای خود از server side event handler استفاده نمایید، زیرا ممکن است که DHTML و JS توسط کلاینت پشتیبانی نشود.

## Validation Groups

در اکثر صفحات پیچیده، ممکن است چندین گروه از کنترل‌های مجزا در پنل‌های جداگانه را داشته باشید. در این موقعیت، ممکن است بخواهید اعتبارسنجی‌ها را بصورت جداگانه انجام دهید. برای نمونه، ممکن است بخواهید یک فرم که شامل یک کادر کنترل‌های login و یک کادر در زیر آن که با کنترل‌هایی برای ثبت یک کاربر جدید همراه می‌باشد، داشته باشید. هر کادر دارای دکمه submit خودش می‌باشد و می‌خواهید بر اساس اینکه کدام دکمه کلیک شده است، اعتبارسنجی آن کادر فقط انجام شود.

این سenario با استفاده از امکانی با نام validation groups، یک input، یک کنترل validator و یک Validation Group را در یک گروه قرار دهیم. ویژگی CausesValidation هر کنترل را با یک عبارت مناسب (در مثال ما مانند NewUserGroup یا LoginGroup) مقدار کنید. هر کنترل که ویژگی CausesValidation را دارا باشد، ویژگی ValidationGroup را نیز دارا خواهد بود.

برای مثال صفحه زیر دارای رو VG با نام‌های Group1 و Group2 می‌باشد.



```
<form id="form2" runat="server">

    <asp:Panel ID="Panel1" runat="server">
        <asp:TextBox ID="TextBox1" ValidationGroup="Group1" runat="server" />
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" ErrorMessage="*Required"
            ValidationGroup="Group1" runat="server" ControlToValidate="TextBox1" />
        <asp:Button ID="Button1" Text="Validate Group1" ValidationGroup="Group1"
            runat="server" />
    </asp:Panel>
    <br />
    <asp:Panel ID="Panel2" runat="server">
        <asp:TextBox ID="TextBox2" ValidationGroup="Group2" runat="server" />
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" ErrorMessage="*Required"
            ValidationGroup="Group2" ControlToValidate="TextBox2" runat="server" />
        <asp:Button ID="Button2" Text="Validate Group2" ValidationGroup="Group2"
            runat="server" />
    </asp:Panel>
</form>
```

```
</form>

Convert: &nbsp;

<input type="text" id="US" runat="server" />

&nbsp; U.S. dollars to &nbsp;

<select id="Currency" runat="server" />

<br />

<br />

<input type="submit" value="OK" id="Convert" onclick="Convert_ServerClick"
runat="server" />

<input type="submit" value="Show Graph" id="ShowGraph" runat="server" />

<br />

<br />

<img id="Graph" src="" alt="Currency Graph" runat="server" />

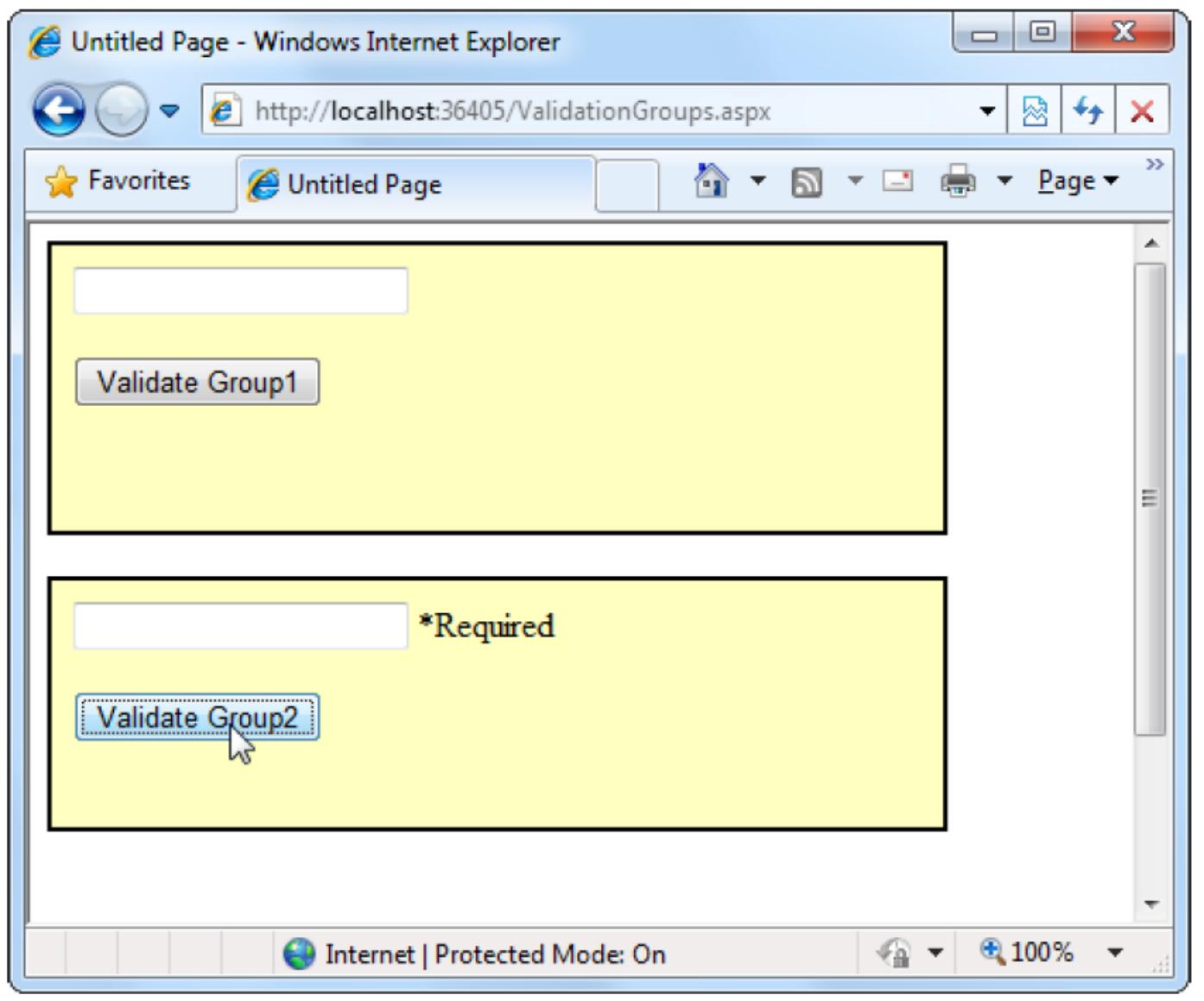
<br />

<br />

<p style="font-weight: bold" id="Result" runat="server">
</p>

</div>

</form>
```



اگر از این روش استفاده نکنیم، با فشردن یک دکمه کنترل‌هایی که در سایر بخش‌های صفحه نیز وجود دارند اعتبارسنجی می‌شوند و ممکن است بی‌دلیل جلوی submit شدن صفحه گرفته شود.

## بخش دوم: ساخت فرم های وب بیتتر

~~Validation~~ = فصل ششم: اعتبارسنجی

فصل هفتم: کنترل های حرفه ای

~~User Control~~ فصل هشتم:

~~Master Page , Styles, Themes~~ فصل نهم:



## Rich Controls

Rich control ها، وب کنترل هایی هستند که المان های واسط کاربری پیچیده ای را مدل می کنند. در حقیقت آنها وب کنترل هایی هستند که یک object model دارند که جدا از HTML ای می باشد که ایجاد کرده است. المان های این کنترل ها می توانند در برابر عمل های کاربرها عکس العمل نشان دهند و رویدادهایی دارند که کد شما می تواند به آنها پاسخ دهد. بنابراین rich control ها راهی برای ایجاد واسط کاربری پیشرفته در صفحه وب شما بدون نوشتگی زیادی از HTML فراهم می کند.

### کنترل AdRotator

اولین هدف این کنترل، فراهم نمودن یک گرافیک در صفحه که بصورت تصادفی از تصاویر موجود استفاده می کند، می باشد. در حقیقت هر بار که صفحه درخواست می شود، یک تصویر بصورت تصادفی انتخاب شده و نمایش داده می شود.

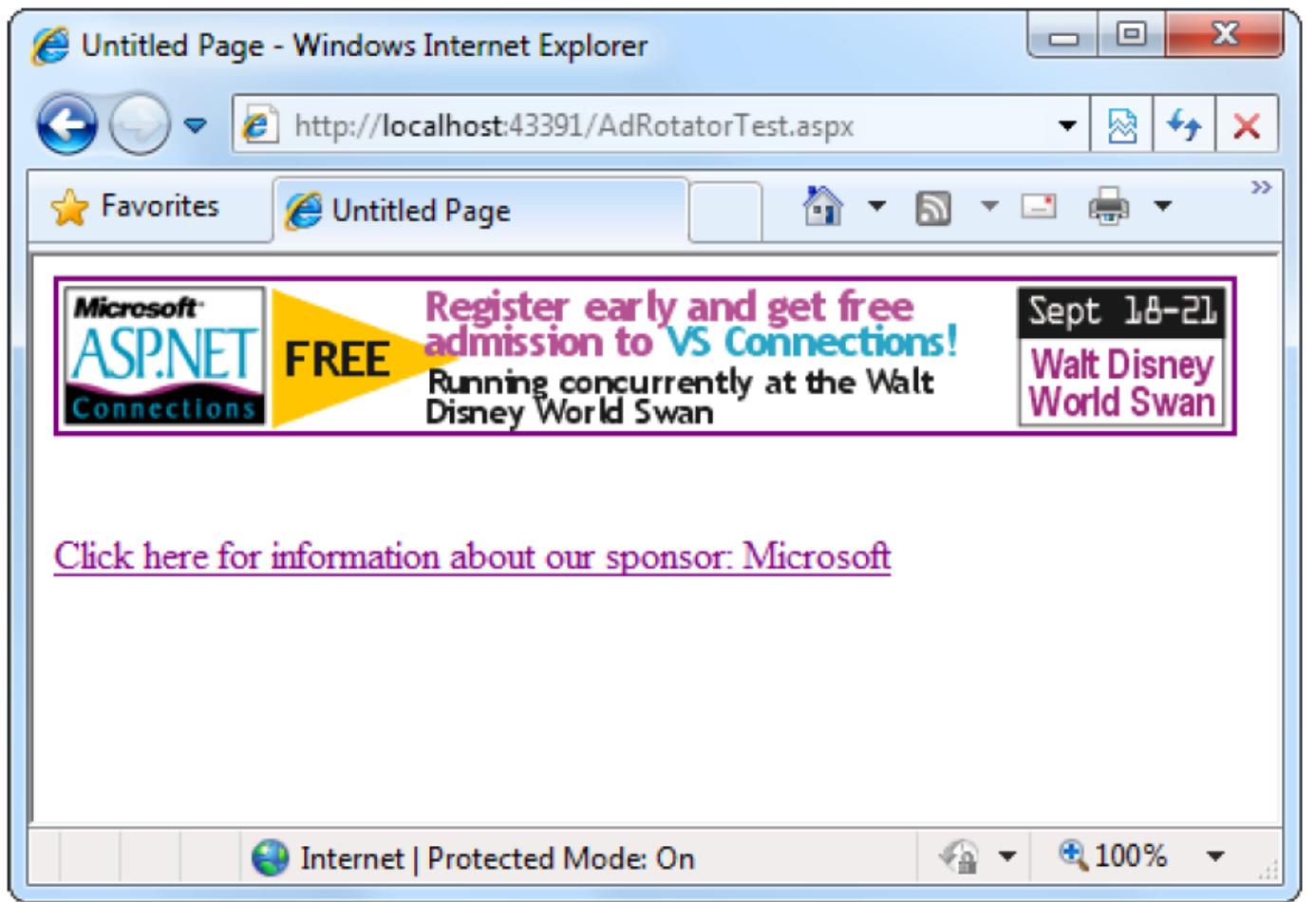
**آیا باید از ADROTATOR برای تبلیغات اینترنتی استفاده کرد؟**

به نظر من نام بهتر برای این کنترل RandomImage یا RotatingImage می باشد. امروزه این کنترل راه رایجی برای تبلیغات اینترنتی حتی در برنامه های ASP.NET نمی باشد. اکثراً از کدهای از پیش تولید شده JS استفاده می کنند.

این کنترل تصاویر خود را در یک فایل XML ذخیره می کند :

```
<advertisements>
<Ad>
<ImageUrl>prosetech.jpg</ImageUrl>
<NavigateUrl>http://www.prosetech.com</NavigateUrl>
<AlternateText>ProseTech Site</AlternateText>
<Impressions>1</Impressions>
<Keyword>Computer</Keyword>
</Ad>
</advertisements>

<asp:AdRotator ID="Ads" runat="server" AdvertisementFile="MainAds.xml"
Target="_blank" KeywordFilter="Computer" />
```



```

protected void Ads_AdCreated(Object sender, AdCreatedEventArgs e)
{
    // Synchronize the Hyperlink control.

    lnkBanner.NavigateUrl = e.NavigateUrl;

    // Syncronize the text of the link.

    lnkBanner.Text = "Click here for information about our sponsor: ";
    lnkBanner.Text += e.AlternateText;
}

```

## صفحه ای با چند View

عموماً در صفحات وب، شما با بخش های مختلفی کار دارید. برای نمونه، اگر بخواهید یک آیتم به سبد خرید خود اضافه نمایید و هزینه آن را پرداخت نمایید، باید به صفحه دیگر بروید. در بعضی موارد ایجاد یک صفحه تنها برای انجام کارهای مختلف لازم است. ممکن است شما بخواهید چندین View (نمایش) از یک داده (مانند chart-base view و grid-base view ) را فراهم کنید و به کاربر امکان سوییج کردن از یک view به دیگر view بدون ترک صفحه حاضر را بدهید. بنابراین نیاز به یک صفحه پویا برای ایجاد بیشتر از یک

view خواهد داشت. صفحه بر اساس اینکه کدام یک از view ها باید نمایش داده شود، کنترل‌های مختلفی را نمایش و مخفی می‌کند.

فرض کنید که صفحه‌ای با سه مرحله ویزارد دارید. بنابراین با اضافه نمودن سه پنل با نام‌های PanelStep1 و PanelStep2 و PanelStep3، به صفحه خود، هر یک برای یک مرحله، کار را شروع می‌کنیم.

کنترل‌های مناسب هر مرحله را می‌توانید در پنل آن مرحله قرار دهید. در ابتدا ویژگی Visible پنل ۲ و ۳ را false نمایید.

```
<asp:Panel ID="panelStep1" runat="server">...</asp:Panel>
<asp:Panel ID="panelStep2" Visible="False" runat="server">...</asp:Panel>
<asp:Panel ID="panelStep3" Visible="False" runat="server">...</asp:Panel>
```

 نکته: زمانی که ویژگی Visible یک کنترل را false قرار می‌دهید، آن کنترل در زمان اجرا در صفحه نمایش داده نمی‌شود و کنترل‌های درون آن نیز مخفی خواهند شد و در HTML رender شده در صفحه نیز نخواهند بود. البته در محیط VS Design می‌توانید آنها را مشاهده کنید.

```
protected void cmdNext_Click(object sender, EventArgs e)
{
    if (panelStep1.Visible)
    {
        // Move to step 2.

        panelStep1.Visible = false;
        panelStep2.Visible = true;

    }
    else if (panelStep2.Visible)
    {
        // Move to step 3.

        panelStep2.Visible = false;
        panelStep3.Visible = true;

        // Change text of button from Next to Finish.

        cmdNext.Text = "Finish";
    }
    else if (panelStep3.Visible)
```

{

```
// The wizard is finished.

panelStep3.Visible = false;

// Add code here to perform the appropriate task
// with the information you've collected.

}
```

}

## کنترل MultiView

خوببختانه ASP.NET کنترلی هایی با نام Wizard و MultiView ارائه داده است که کار را برای شما ساده تر کرده اند. کنترل MultiView، راهی برای تعریف چند View و نمایش تنها یکی از آنها در یک زمان را ارائه می‌دهد. می‌توانید تگ `<asp:MultiView>` را به فایل .aspx و درون آن، تگ `<asp:View>` را برای هر View اضافه نمایید.

```
<asp:MultiView ID="MultiView1" runat="server">

<asp:View ID="View1" runat="server">
    ...
</asp:View>

<asp:View ID="View2" runat="server">
    ...
</asp:View>

<asp:View ID="View3" runat="server">
    ...
</asp:View>

</asp:MultiView>
```

: مثال

```
<asp:MultiView ID="MultiView1" runat="server">

<asp:View ID="View1" runat="server">
    Choose a foreground (text) color:<br />
    <asp:DropDownList ID="lstForeColor" runat="server" AutoPostBack="True" OnSelectedIndexChanged="ControlChanged" />
    <br />
    <br />
```

Choose a background color:<br />

```
<asp:DropDownList ID="lstBackColor" runat="server" AutoPostBack="True" OnSelectedIndexChanged="ControlChanged" />
```

```
</asp:View>
```

```
<asp:View ID="View2" runat="server">
```

Choose a border style:<br />

```
<asp:RadioButtonList ID="lstBorder" runat="server" AutoPostBack="True" OnSelectedIndexChanged="ControlChanged" />
```

```
RepeatColumns="2" />
```

```
<br />
```

```
<asp:CheckBox ID="chkPicture" runat="server" AutoPostBack="True" OnCheckedChanged="ControlChanged" />
```

```
Text="Add the Default Picture" />
```

```
</asp:View>
```

```
<asp:View ID="View3" runat="server">
```

Choose a font name:<br />

```
<asp:DropDownList ID="lstFontName" runat="server" AutoPostBack="True" OnSelectedIndexChanged="ControlChanged" />
```

```
<br />
```

```
<br />
```

Specify a font size:<br />

```
<asp:TextBox ID="txtFontSize" runat="server" AutoPostBack="True" OnTextChanged="ControlChanged" />
```

```
<br />
```

```
<br />
```

Enter the greeting text below:<br />

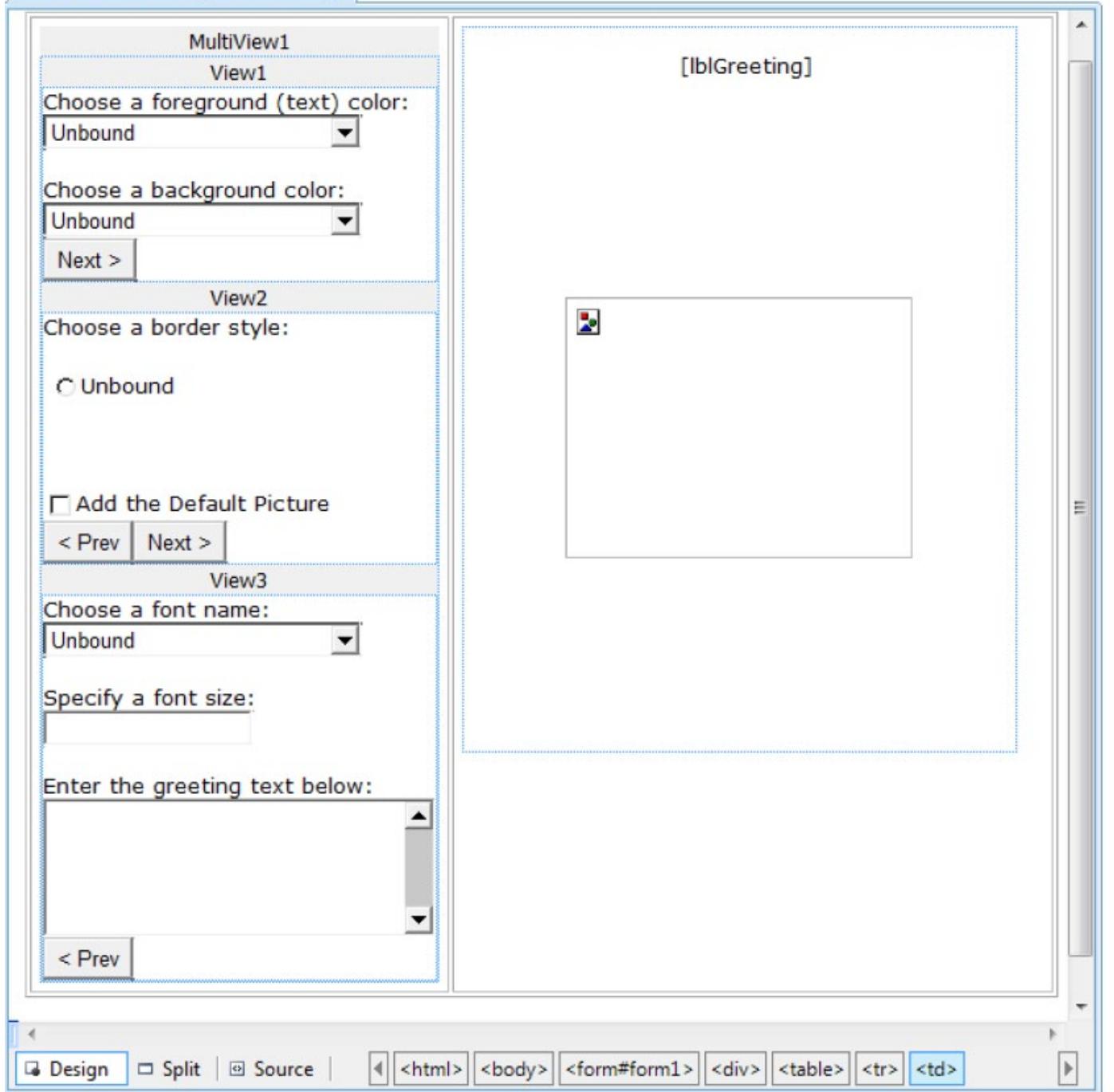
```
<asp:TextBox ID="txtGreeting" runat="server" AutoPostBack="True" OnTextChanged="ControlChanged" />
```

```
TextMode="MultiLine" />
```

```
</asp:View>
```

```
</asp:MultiView>
```

## MultiViewGreetingCardMaker.aspx



## نمایش یک View

اگر مثال بالا را اجرا کنید، چیزی در صفحه مشاهده نخواهد کرد. باید ویژگی MultiView.ActiveViewIndex را برای آن مقدار دهی کنید.

```
// Show the first view.

MultiView1.ActiveViewIndex = 0;
```

همچنین می‌توانید از متدهای static SetActiveView() نیز استفاده نمایید.

```
MultiView1.SetActiveView(View1);
```

در لیست زیر عناوین فرامین مختلف را که فیلدهای MultiView در کلاس static باشند مشاهده خواهید کرد.



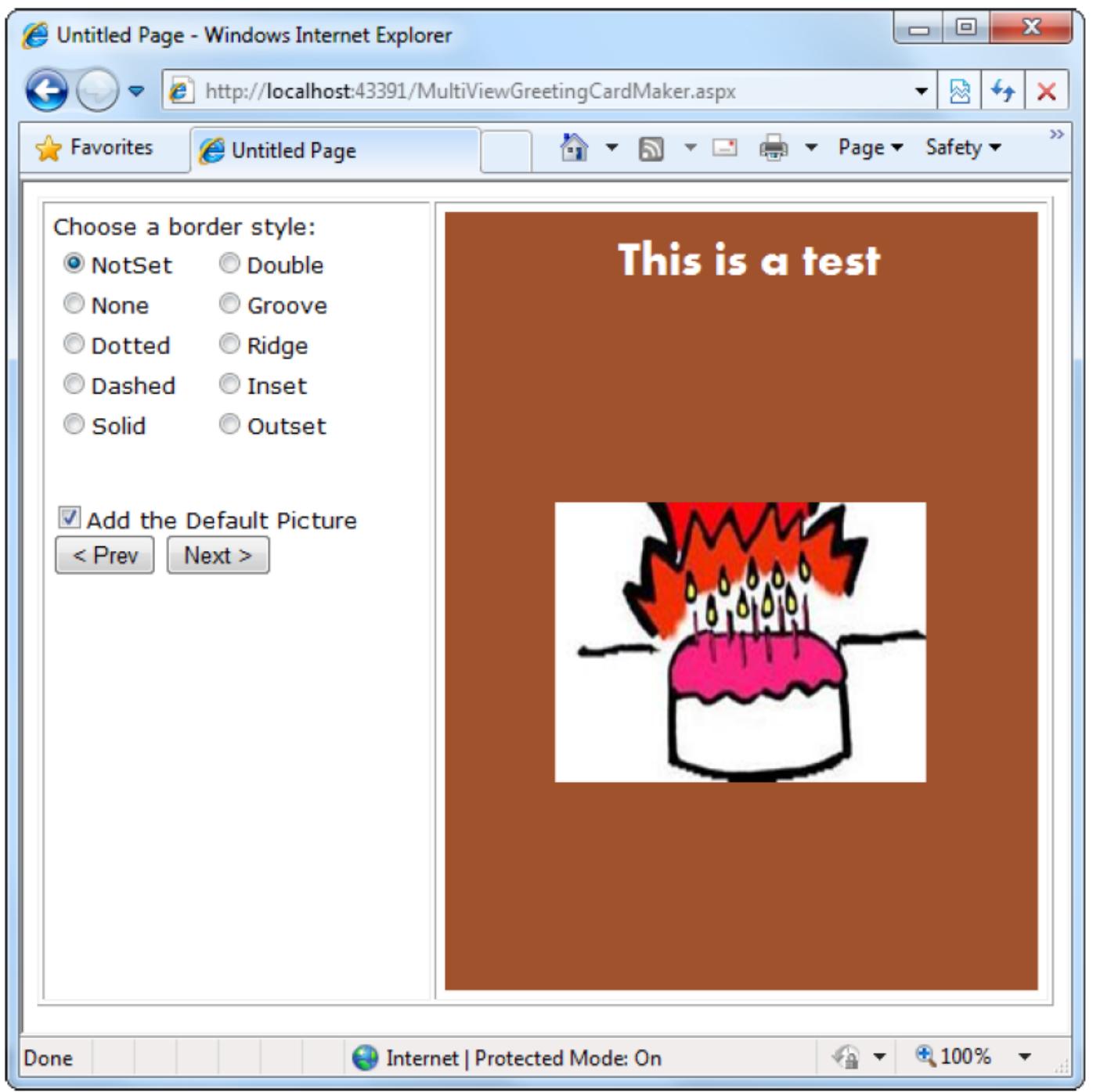
توضیحات	فیلد Multi View	نام دستور -Command Name
حرکت به View قبلی	PreviousViewCommandName	PrevView
حرکت به View بعدی	NextViewCommandName	NextView
حرکت به View با شناسه مشخص. CommandArgu- ID از ویژگی CommandArgument دکمه گرفته می‌شود.	SwitchViewByIDCommandName	SwitchViewByID
حرکت به View با ایندکس مشخص. این ایندکس از ویژگی CommandArgument دکمه گرفته می‌شود.	SwitchViewByIndexCommandName	SwitchViewByIndex

```
<asp:Button ID="Button1" runat="server" CommandArgument="View2"
CommandName="SwitchViewByID" Text="Go to View2" />
```

زمانی که روی این دکمه کلیک کنید، MultiView را برای نمایش View مشخص شده توسط ویژگی CommandArgument تنظیم می‌کند.

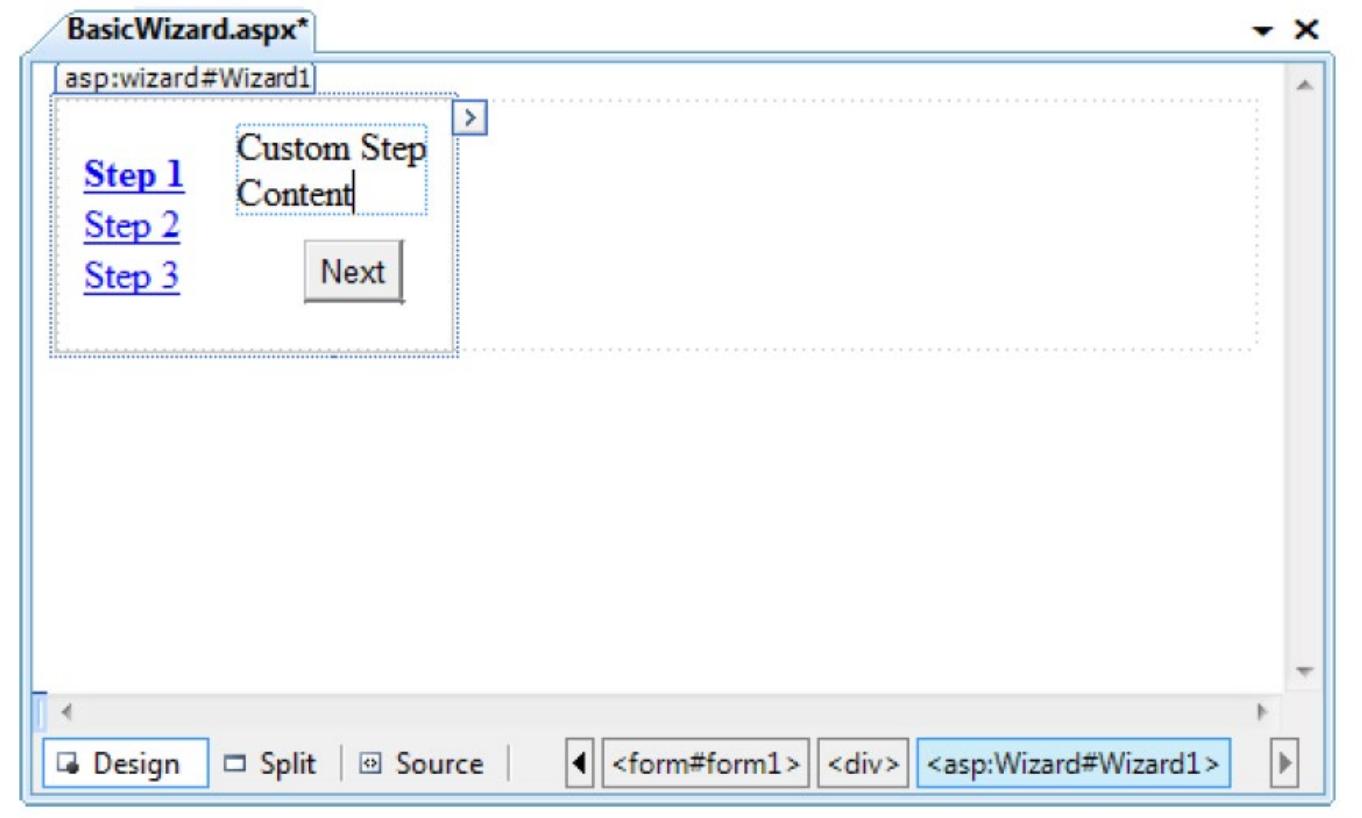
یا

```
<asp:Button ID="Button1" runat="server" Text="< Prev" CommandName="PrevView" />
<asp:Button ID="Button2" runat="server" Text="Next >" CommandName="NextView" />
```



## Wizard Control

این کنترل شبیه MultiView می‌باشد با این تفاوت که دارای دکمه‌های مرحله، استایل sidebar و template نیز هست. ویزارد ها معمولاً مرحله به مرحله و پشت سر هم هستند. کنترل ویزارد ASP.NET دارای لینک‌هایی برای پرش مستقیم به View مورد نظر می‌باشد. با استفاده از Wizard.DisplaySideBar می‌توانید نمایش sidebar را کنترل کنید.



## تعريف مراحل ویزارد

```
<asp:Wizard ID="Wizard1" runat="server" ...>

<WizardSteps>
    <asp:WizardStep runat="server" Title="Step 1">
        ...
    </asp:WizardStep>
    <asp:WizardStep runat="server" Title="Step 2">
        ...
    </asp:WizardStep>
    ...
</WizardSteps>

</asp:Wizard>
```

توضیحات	Property
نام مرحله است که در متن لینک sidebar استفاده می شود.	Title
نوع مرحله می باشد که مقدار از نوع شمارشی WizardStepType است. این ویژگی نوع کنترل های navigation را تعیین و برای آن مرحله نمایش می دهد.	StepType
تعیین می کند که آیا کاربر می تواند به این مرحله برگردد یا نه. اگر false بود، کاربر پس از عبور از این مرحله نمی تواند به آن برگردد. لینک موجود در sidebar برای این مرحله غیر فعال می شود و دکمه previous از این مرحله پریده یا مخفی می شود.	AllowReturn

```

<asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="0" BackColor="LemonChiffon"
    BorderStyle="Groove" BorderWidth="2px" CellPadding="10">

    <WizardSteps>

        <asp:WizardStep runat="server" Title="Step 1 - Colors">
            Choose a foreground (text) color:<br />
            <asp:DropDownList ID="lstForeColor" runat="server" />
            <br />
            Choose a background color:<br />
            <asp:DropDownList ID="lstBackColor" runat="server" />
        </asp:WizardStep>

        <asp:WizardStep runat="server" Title="Step 2 - Background">
            Choose a border style:<br />
            <asp:RadioButtonList ID="lstBorder" runat="server" RepeatColumns="2" />
            <br />
            <br />
            <asp:CheckBox ID="chkPicture" runat="server" Text="Add the Default Picture" />
        </asp:WizardStep>

        <asp:WizardStep runat="server" Title="Step 3 - Text">
            Choose a font name:<br />
            <asp:DropDownList ID="lstFontName" runat="server" />
            <br />
            <br />
        </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>

```

```

Specify a font size:<br />

<asp:TextBox ID="txtFontSize" runat="server" />

<br />

<br />

Enter the greeting text below:<br />

<asp:TextBox ID="txtGreeting" runat="server" TextMode="MultiLine" />

</asp:WizardStep>

<asp:WizardStep runat="server" StepType="Complete" Title="Greeting Card">

<asp:Panel ID="pnlCard" runat="server" HorizontalAlign="Center">

<br />

<asp:Label ID="lblGreeting" runat="server" />

<asp:Image ID="imgDefault" runat="server" Visible="False" />

</asp:Panel>

</asp:WizardStep>

</WizardSteps>

</asp:Wizard>

```

برخلاف MultiView تنها می‌توانید یک مرحله را در VS در هر زمان مشاهده کنید. برای تعیین اینکه روی کدام مرحله می‌خواهید کار کنید باید از لیست موجود در smarttag که در شکل زیر مشاهده خواهید کرد استفاده نمایید.

 نکته : به یاد آورید، زمانی که کنترل هایی را به مراحل جداگانه اضافه می‌کنید، آنها در ViewState خواهند ماند. اگر دارای ویژگی‌های پیچیده‌ای هستید، باید آنها را درون چند صفحه بشکنید و با استفاده از Server.Transfer() از یک صفحه به صفحه بعدی بروید.

می‌توانید مراحل را با مانند زیر اعتبار سنجی کنید :

```

protected void Wizard1_NextButtonClick(object sender, WizardEventArgs e)

{

    // Perform some sort of check.

    if (e.NextStepIndex == 1 && txtName.Text == "")

    {

        // Cancel navigation and display a message elsewhere on the page.

```

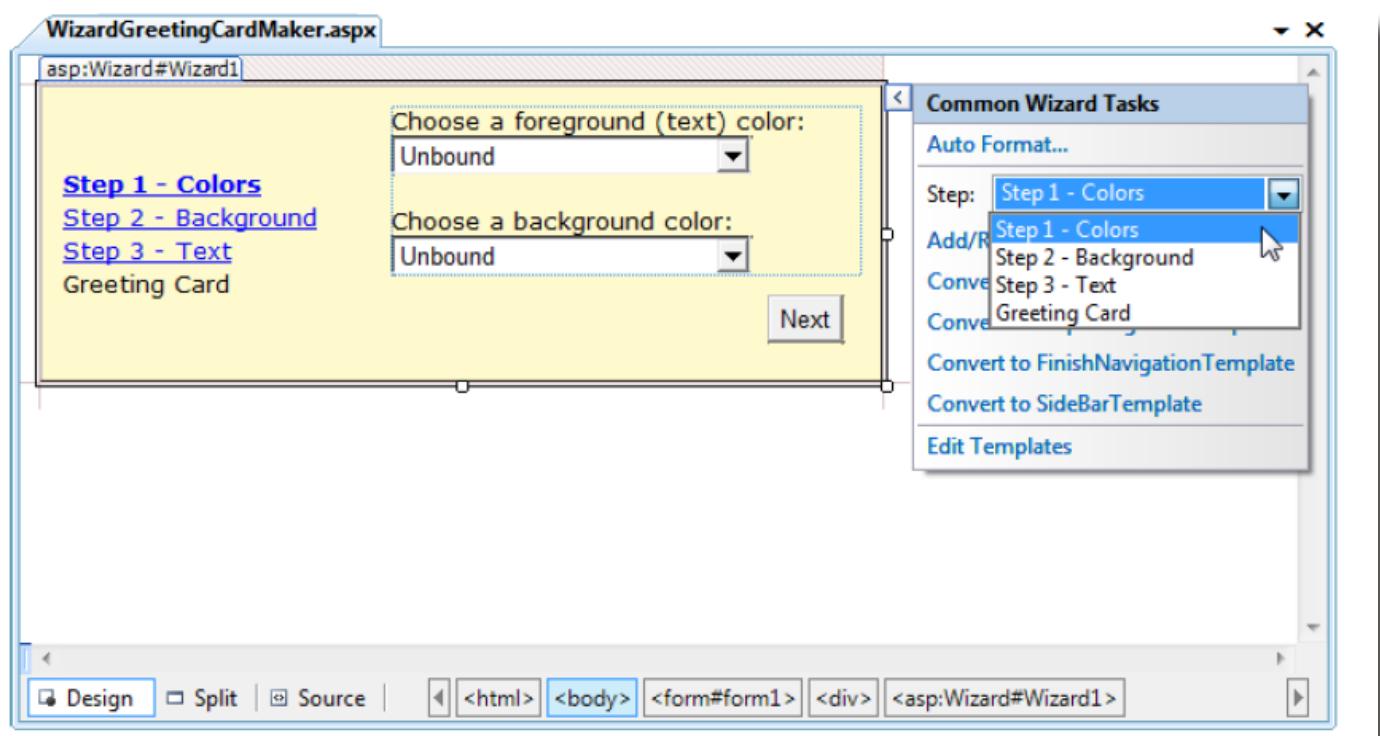
```

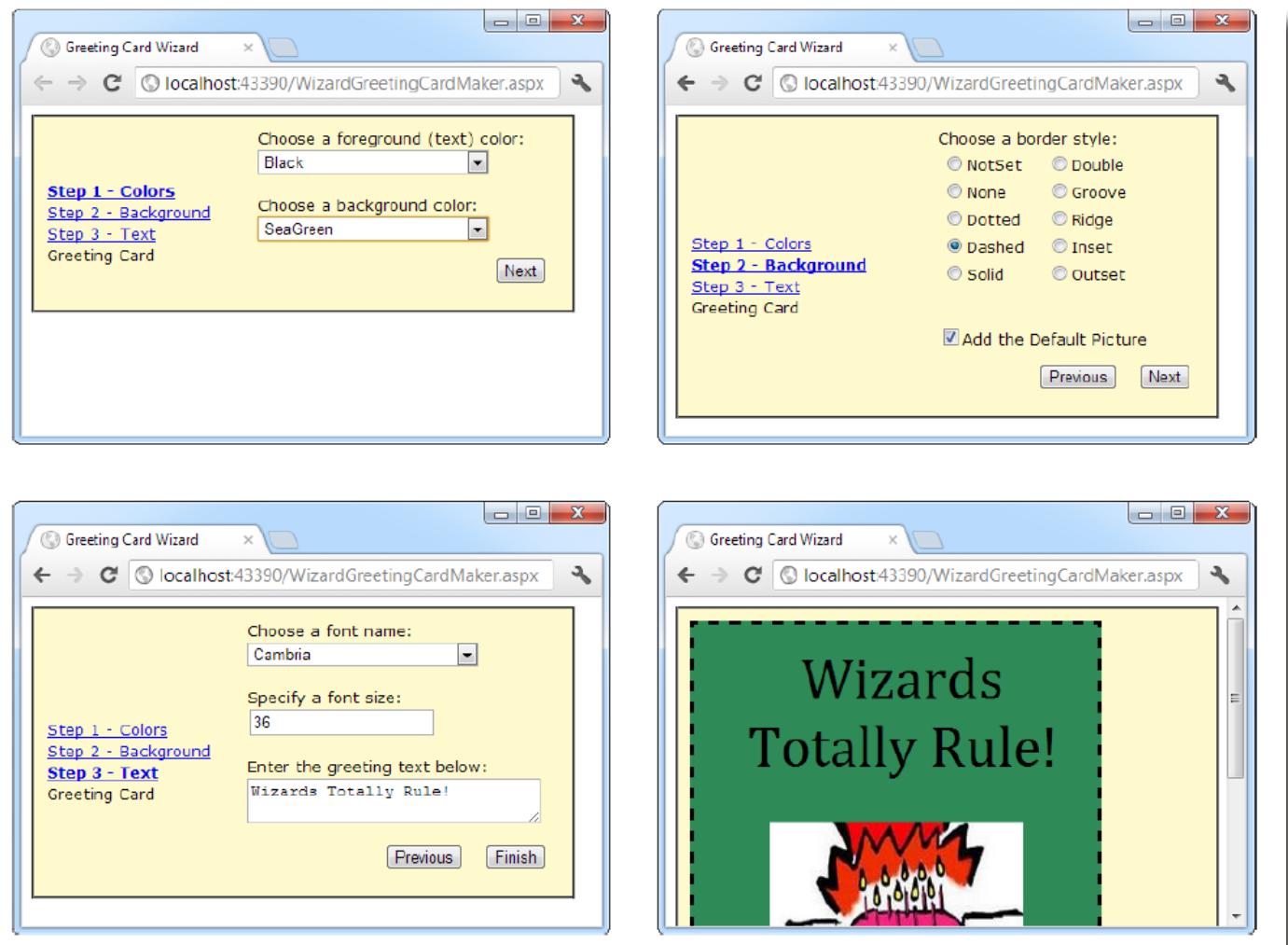
e.Cancel = true;

lblInfo.Text =
"You cannot move to the next step until you supply your name.";

}

```





نکته: شما همچنین می‌توانید از validator های موجود در ASP.NET بدون هیچ مشکلی در ویزارد ها استفاده کنید. اگر کنترل validation، داده غیرمجازی را شناسایی کند، از کلیک روی هر یک از لینک های sidebar جلوگیری می‌کند. البته کنترل ویزارد دارای رویدادهایی مانند SideBarButtonClick، CancelButtonClick، ActiveStepChanged و ... و همچنین استایل های ویزاردی مانند FinishPreviousButtonStyle، SideBarStyle، HeaderStyle، ControlStyle و ... می‌باشد که در این کتاب آنها را توضیح نمی‌دهیم.

# پیش دویم: ساخت فرم های وب بهتر

Validation = ~~فعال ششم: اعتبارسنجی~~

~~فعال هفتم: کنترل حایی در فایل~~

User Control ~~: فعال هشتم~~

Master Page , Styles, Themes ~~: فعال نهم~~



## Graphic و User Control

در این فصل، با سه روش گسترش صفحات وب آشنا می شویم.

در ابتداء، با usercontrol ها آشنا می شویم که به شما یک راه موثر برای استفاده مجدد از کدهای markup واسط کاربری و کدهای مربوط به آن ارائه می دهد. این کنترل ها یک ابزار کلیدی برای ساخت برنامه های تحت وب مازولار می باشد.

سپس با طراحی سفارشی توسط Microsoft Windows +GDI، آشنا خواهیم شد. خواهید دید که چگونه می توانید تصویری مورد نظر خود را نقاشی کرده و از یک تصویر پویا در وب فرم خود استفاده کنید.

در نهایت، نگاهی به کنترل ASP.NET Chart خواهیم انداخت که داده های شما را می گیرد و برای شما نمودارش را رسم می کند.

## User Controls

اگرچه استفاده مجدد از کد بسیار ساده می باشد (به سادگی کد را داخل محلی خارج از صفحه مانند کلاس دیگر قرار می دهید)، اما استفاده مجدد از markup صفحه به همین سادگی نمی باشد.

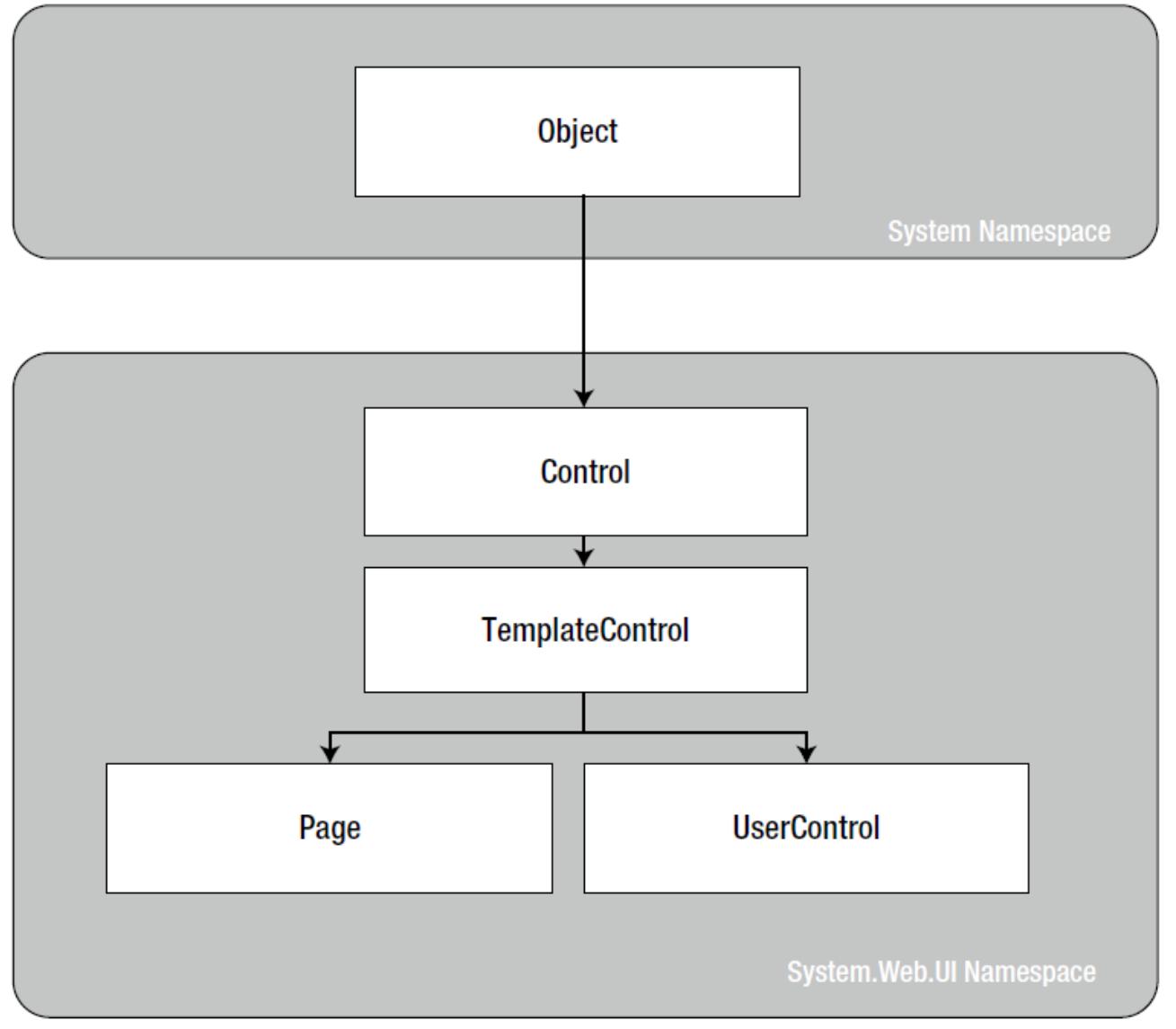
البته می توانید کدهای ASP.NET و HTML را cut و paste کنید. ولی در صورت تغییر markup باید هر جایی که آنها را کپی کرده اید را نیز تغییر دهید. در عوض شما نیاز به یک پوشش (wrap) برای پوشاندن کدهای markup صفحه وب در یک بسته قابل استفاده مجدد خواهید داشت.

Usercontrol ها بسیار شبیه فرم های وب ASP.NET می باشند. مانند وب فرم ها، دارای ترکیبی از کدهای HTML و تگ های کنترل (فایل .ascx) و بصورت انتخابی دارای یک فایل event-handling code-behind با منطق code-behind، می باشند. تنها تفاوت های بین user control ها و صفحات وب موارد زیر می باشد :

- UC ها دارای پسوند فایل .ascx می باشند و کدهای موجود در code-behind از کلاس System.Web.UI.USERControl ارث بری می کند. در حقیقت، کلاس UC و Page هر دو از یک کلاس پایه مشتق می شوند. بنابراین از متدها و رویدادهای زیاد مشترکی استفاده می کنند.

- فایل .ascx. با راهنمای صفحه <% Control @%> شروع می شود و .aspx با <% Page @%> شروع می شود.

- UC ها نمی توانند مستقیماً توسط یک مرورگر وب استفاده شوند. بنابراین باید درون یک صفحه وب گنجانده (embed) شوند.



## ایجاد یک User Control ساده

از منوی **Web User Control** website>add new item گزینه **website**>**add new item** نمایید.

UC موجود در زیر دارای تنها یک کنترل **label** می باشد:

```

<%@ Control Language="#c" AutoEventWireup="true" CodeFile="Footer.ascx.cs" Inherits="Footer" %>

<asp:Label ID="lblFooter" runat="server" />
  
```

در نمونه زیر از رویداد Load برای اضافه نمودن متن به label استفاده شده است :

```
public partial class Footer : System.Web.UI.UserControl
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        lblFooter.Text = "This page was served at ";
        lblFooter.Text += DateTime.Now.ToString();
    }
}
```

برای امتحان این UC، باید آن را درون یک صفحه وب وارد کنید. این کار دو مرحله دارد. ابتدا باید یک راهنمای صفحه به Register Directive که شامل UC می‌باشد. باید آن را بعد از Page Directive اضافه کنید.

این کار، کنترلی که می‌خواهد استفاده نمایید را مشخص و آن را با یک پیشوند کنترل مانند زیر نشان می‌دهد :

```
<%@ Register TagPrefix="apress" TagName="Footer" Src="Footer.ascx" %>
```

این کد، یک تگ پیشوند و یک نام را مشخص می‌کند. این پیشوندها، مجموعه‌ای از کنترل‌های مرتبط (برای نمونه، تمامی کنترل‌های وب از پیشوند asp استفاده می‌کنند) را در یک گروه قرار می‌دهند. آنها case-insensitive (نسبت به حروف کوچک و بزرگ حساس نیستند) می‌باشند و باید در کمپانی، سازمان و برنامه شما یونیک یا یکتا باشند.

مرحله دوم، اضافه نمودن UC در محل مورد نظرتان در صفحه می‌باشد.

```
<%@ Page Language="#c" AutoEventWireup="true" CodeFile="FooterHost.aspx.cs"
Inherits="FooterHost" %>
```

```
<%@ Register TagPrefix="KaraMoozesh" TagName="Footer" Src="Footer.ascx" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
```

```

<http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

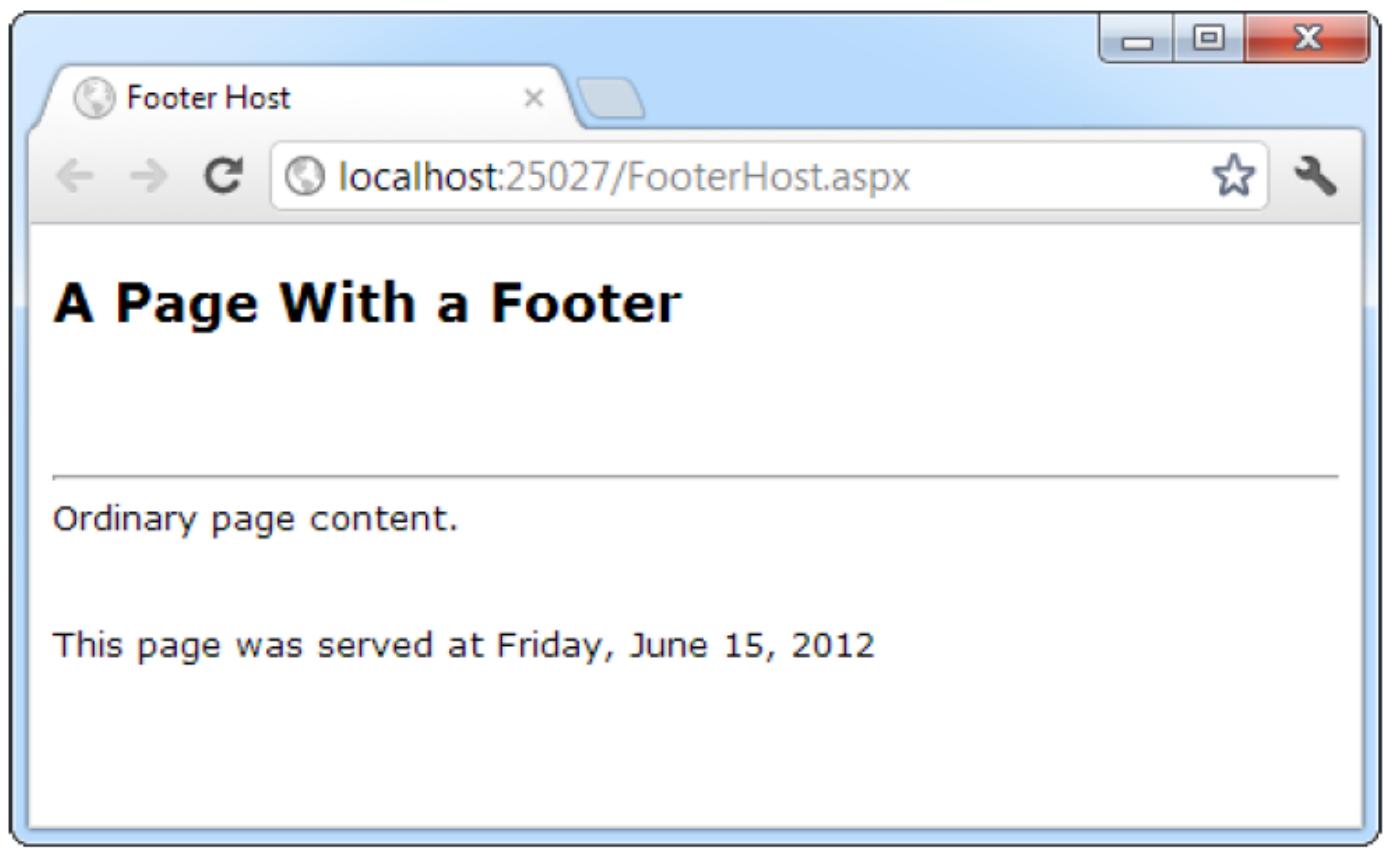
<head id="Head1" runat="server">
    <title>Footer Host</title>
</head>

<body>
    <form id="form1" runat="server">
        <div>
            <h1>
                A Page With a Footer</h1>
            <hr />
            Static Page Text
            <br />
            <br />
            <KaraMoozesh:Footer id="Footer1" runat="server" />
        </div>
    </form>
</body>
</html>

```

این مثال یک راه ساده برای ایجاد یک header یا footer و استفاده مجدد از آن در تمامی صفحات موجود در وب سایت شما را نشان می‌دهد.

 نکته: کلاس Page یک متد LoadControl که اجازه ایجاد یک UC بصورت داینامیک و پویا را در زمان اجرا در فایل ascx به شما می‌دهد را در اختیار شما قرار می‌دهد. UC به شما در قالب یک شی کنترل برگردانده می‌شود که می‌توانید آن را به یک کنترل Container (مانند یک PlaceHolder یا Panel) در صفحه وب اضافه کنید. این تکنیک یک جایگزین خوب برای تعریف یک UC نمی‌باشد زیرا پیچیده می‌باشد. البته برای ایجاد واسط کاربری پویا می‌توانید از این روش استفاده کنید.



در Visual Studio یک راه کوتاه برای اضافه نمودن یک UC در صفحه بدون تایپ راهنمای صفحه Register بصورت دستی دارد. صفحه وب را باز کنید. سپس فایل ascx مورد نظر را در solution Explorer پیدا کرده و آن را در محیط Design صفحه وب بکشید. VS، بصورت خودکار راهنمای صفحه Register را برای UC اضافه می‌کند.

گزینه دیگر، پیکربندی UC ر فایل web.config می‌باشد. در زیر نمونه ای برای کنترل Footer بالا را مشاهده می‌کنید:

```
<configuration>
  <system.web>
    <pages>
      <controls>
        <add tagPrefix="KaraMoozesh" src="~/Footer.ascx" tagName="Footer" />
      </controls>
    </pages>
  </system.web>
</configuration>
```

می توانید المان های **<add controls="UC">** برای ثبت UC های بیشتر اضافه نمایید. با این روش UC شما بدون ثبت در هر صفحه، در همه صفحات در دسترس می باشد.

## کار با User Control های مستقل

مفهوماً دو نوع از UC موجود می باشد: Integrated (مستقل) و Independent (یکپارچه).

Independent UC LinkMenu ها با بقیه کدهای درون فرم تعاملی ندارند. کنترل Footer یکی از این نمونه ها بود. نمونه دیگر یک که شامل لیستی از دکمه هایی که لینک هایی به صفحات دیگر دارند می باشد. این کنترل می تواند رویداد های همه دکمه ها را هندل کند و سپس متده ردایت Response.Redirect مناسب برای انتقال به صفحه دیگر را اجرا کند.

نمونه زیر، یک کنترل ساده که لیستی از لینک ها را نمایش می دهد تعریف می کند.

```
<%@ Control Language="#c" AutoEventWireup="true" CodeFile="LinkMenu.ascx.cs"
Inherits="LinkMenu" %>

<div>

Products:<br />

<asp:HyperLink ID="lnkBooks" runat="server" NavigateUrl="MenuHost.aspx?product=Books">
Books
</asp:HyperLink>
<br />

<asp:HyperLink ID="lnkToys" runat="server" NavigateUrl="MenuHost.aspx?product=Toys"> Toys
</asp:HyperLink>
<br />

<asp:HyperLink ID="lnkSports" runat="server" NavigateUrl="MenuHost.aspx?product=Sports">
Sports
</asp:HyperLink>
<br />

<asp:HyperLink ID="lnkFurniture" runat="server" NavigateUrl="MenuHost.aspx?product=Furniture"> Furniture
</asp:HyperLink>

</div>
```

لینک ها هیچ کد سمت سروری را ایجاد نمی کنند. در عوض آنها خودشان را مانند تگ های قدیمی anchor با URL های کدهای دستی نوشته شده در برنامه) رندر می کنند.

برای امتحان این منو، می توانید از صفحه وب MenuHost.aspx در زیر استفاده کنید. این صفحه شامل دو کنترل menu و label می باشد که Product querystring مربوط به پارامتر querystring را نمایش می دهند.

```
<%@ Page Language="#c" AutoEventWireup="true" CodeFile="MenuHost.aspx.cs" Inherits="MenuHost"
%>

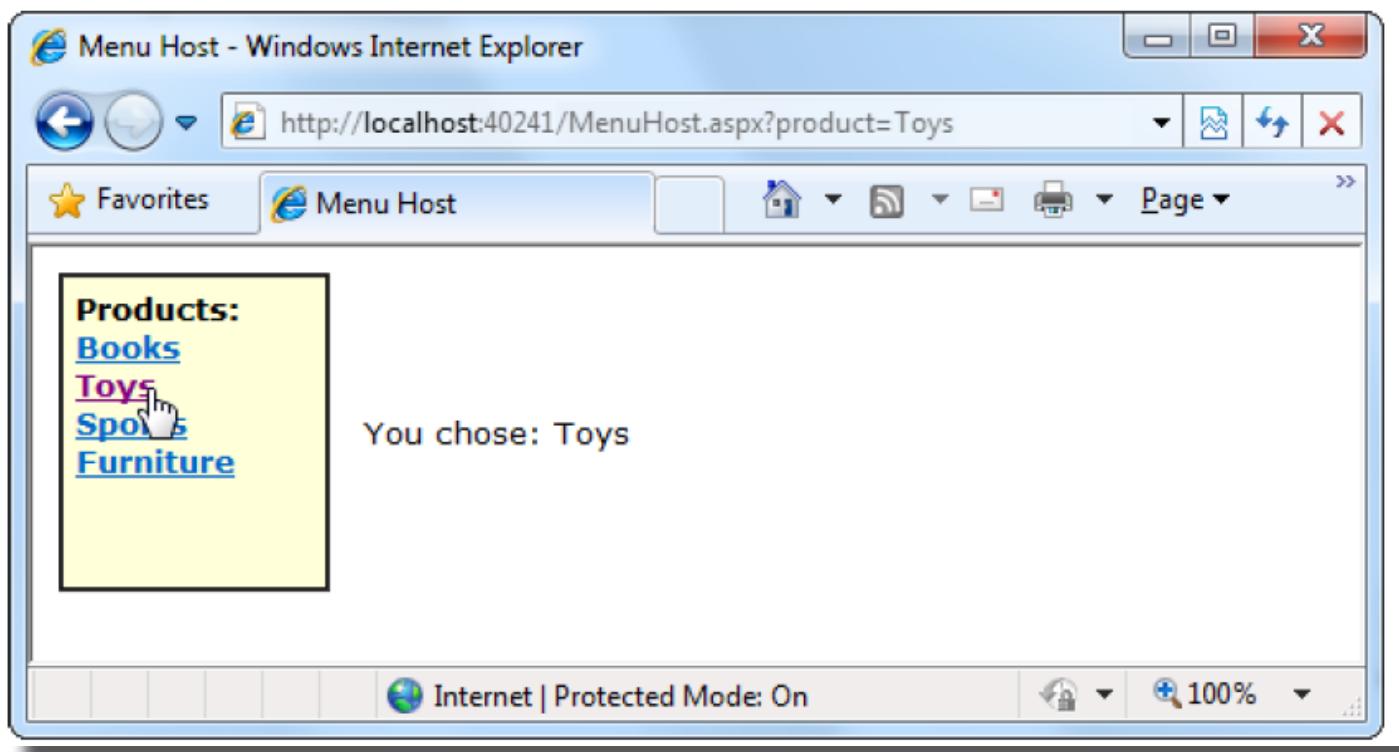
<%@ Register TagPrefix="KaraMoozesh" TagName="LinkMenu" Src="LinkMenu.ascx" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Menu Host</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <table>
                <tr>
                    <td>
                        <KaraMoozesh:LinkMenu id="Menu1" runat="server" />
                    </td>
                    <td>
                        <asp:Label ID="lblSelection" runat="server" />
                    </td>
                </tr>
            </table>
        </div>
    </form>
</body>
</html>
```

زمانی که صفحه لود می‌شود، اطلاعات مناسب به کنترل `lblSelection` اضافه می‌شود :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.Params["product"] != null)
    {
        lblSelection.Text = "You chose: ";
        lblSelection.Text += Request.Params["product"];
    }
}
```



## کار با Integrated User Control

این کنترل‌ها به یک طریقی با صفحه ای که در آن قرار دارند در تعامل می‌باشند. زمانی که این کنترل‌ها را طراحی می‌کنید نکات موجود در فصل ۴ درباره `clase-based design` بسیار مفید می‌باشد.

برای نمونه می‌توانید یک `footer` که دو فرمت نمایش را پشتیبانی می‌کند ایجاد کنید: `Short Time` و `Long date`. زمانی که یک صفحه وب از آن استفاده می‌کند، این کنترل تصمیم خواهد گرفت که کدام فرمت نمایش مورد نظر صفحه را باید ارائه دهد.

برای راحت تر شدن کار این UC از یک نوع شمارشی استفاده می‌کند که شامل دو فرمت ذکر شده باشد. این نوع شمارشی را می‌توانید درون کلاس Footer ایجاد نمایید.

```
public enum FooterFormat
{
    LongDate,
    ShortTime
}
```

به یاد بیاورید که یک نوع شمارشی تنها نوعی از `contant` ها می‌باشد که بصورت داخلی به عنوان یک عدد صحیح ذخیره می‌شود ولی در کد می‌توانید نام آنها را استفاده کنید.

مرحله بعدی اضافه کردن یک `Property` به کلاس `Footer` می‌باشد که به صفحه وب اجازه بازیابی و تعیین فرمت صحیح برای `footer` را می‌دهد. فرمت پیشفرض `long-date` می‌باشد.

```
private FooterFormat format = FooterFormat.LongDate;
```

```
public FooterFormat Format
{
    get { return format; }
    set { format = value; }
}
```

در نهایت، هندررویداد `UserControl.Load`، باید وضعیت کنونی `footer` را تعیین و خروجی را بر اساس آن فرمت دهی کند.

```
public partial class Footer : System.Web.UI.UserControl
{
    public enum FooterFormat
    {
        LongDate, ShortTime
    }

    private FooterFormat format = FooterFormat.LongDate;

    public FooterFormat Format
```

```
{  
    get { return format; }  
    set { format = value; }  
}  
  
  
protected void Page_Load(Object sender, EventArgs e)  
{  
    lblFooter.Text = "This page was served at ";  
    if (format == FooterFormat.LongDate)  
    {  
        lblFooter.Text += DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");  
    }  
    else if (format == FooterFormat.ShortTime)  
    {  
        lblFooter.Text += DateTime.Now.ToString("HH:mm:ss");  
    }  
}
```

}

برای آزمون این footer باید یک صفحه وب بسازید که ویژگی Format را کنترل Footer را تغییر دهد.



.cmdRefresh.Click توجه داشته باشید که Property موجود در UC در Page.Load تغییر می‌کند و نه در Click رویداد Load قبل از رندر شدن UC رخ می‌دهد. رویداد Click بعد از رندر شدن UC رخ می‌دهد و بنابراین property در کد در دسترس می‌باشد.

```
public partial class FooterHost : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        if (optLong.Checked)
        {
            Footer1.Format = Footer.FooterFormat.LongDate;
        }
        else if (optShort.Checked)
```

```

    }

    Footer1.Format = Footer.FooterFormat.ShortTime;

}

else

{

    // The default value in the Footer class will apply.

}

}

}

```

همچنین می‌توانید مقادیر مربوط به UC را در تگ کنترل مربوطه تعیین کنید:

```
<KaraMoozesh:Footer Format="ShortTime" id="Footer1" runat="server" />
```

عبارت "ShorTime" را به نوع شمارشی مرتبط با آن تبدیل می‌کند.

## استفاده از رویدادهای User Control

راه دیگر ارتباط بین UC و صفحه وب، رویدادها می‌باشند. با استفاده از متدها و property‌ها، UC می‌تواند به تغییرات ایجاد شده توسط کد صفحه وب عکس العمل نشان دهد. با استفاده از رویدادها ماجرا بر عکس می‌شود. UC، صفحه وب را درباره یک عمل آگاه می‌سازد و کد موجود در صفحه وب به آن پاسخ می‌دهد.

در نمونه زیر شما یک نسخه از کنترل LinkMenu را می‌بینید که از رویدادها استفاده می‌کند. در زمانی که کاربر روی یک دکمه کلیک می‌کند، بجای هدایت مستقیم به صفحه مناسب، این کنترل یک رویداد را منتشر می‌کند که صفحه وب می‌تواند آن را همانطور که می‌خواهد (توسط کد برای آن تعریف می‌شود) هندل کند.

مرحله اول برای ایجاد این کنترل، تعریف رویدادها می‌باشد. می‌دانید که برای تعریف یک رویداد ابتدا باید signature آن را انتخاب کرد. استاندارد دات نت برای رویدادها تعیین کرده است که هر رویداد باید از دو پارامتر استفاده کند. اولی یک ارجاع به کنترلی که رویداد را منتشر کرده است و دومی شامل اطلاعات اضافی در باره رویداد می‌باشد. این اطلاعات اضافی درون یک شی EventArgs پیچیده شده است که از کلاس System.EventArgs ارث بری می‌کند.

اگر رویداد شما به اطلاعات اضافی نیازی ندارد می‌توانید از کلاس از پیش تعریف شده EventArgs استفاده نمایید.

```
public partial class LinkMenu2 : System.Web.UI.UserControl
{
    public event EventHandler LinkClicked;
    ...
}
```

این کد یک رویداد با نام LinkClicked را تعریف می‌کند. این رویداد دارای signature ای می‌باشد که توسط یک delegate با نام تعیین شده است دارای دو پارامتر می‌باشد. System.EventHandler

```
protected void LinkMenu_LinkClicked(object sender, EventArgs e)
{ ... }
```

برای fire کردن رویداد، کنترل LinkMenu2 به سادگی آن را با نام رویداد فراخوانی و دو پارامتر مربوطه را به آن ارسال می‌کند.

```
// Raise the LinkClicked event, passing a reference to
// the current object (the sender) and an empty EventArgs object.

LinkClicked(this, EventArgs.Empty);
```

کنترل LinkMenu2، نیاز به چند تغییر دارد. نسخه اصلی از کنترل HyperLink استفاده می‌کرد. در اینجا نمی‌توان از آن استفاده کرد زیرا کنترل HyperLink رویدادی را در زمان کلیک شدن روی لینک fire نمی‌کند. در عوض از LinkButton استفاده می‌کنیم که رویداد Click را خواهد کرد.

```
public partial class LinkMenu2 : System.Web.UI.UserControl
{
    public event EventHandler LinkClicked;

    protected void lnk_Click(object sender, EventArgs e)
    {
        // One of the LinkButton controls has been clicked.

        // Raise an event to the page.

        if (LinkClicked != null)
    {
```

```
LinkClicked(this, EventArgs.Empty);
```

```
}
```

```
}
```

```
}
```

توجه داشته باشید که قبل از انتشار رویداد LinkClicked کنترل LinkMenu2 باید null reference بودن رویداد را تست کند. زمانی وجود خواهد داشت که، هیچ event-handler ای به رویداد متصل (attached) نشده باشد. در این حالت باید رویداد را منتشر کرد زیرا خطای خطا می‌دهد.

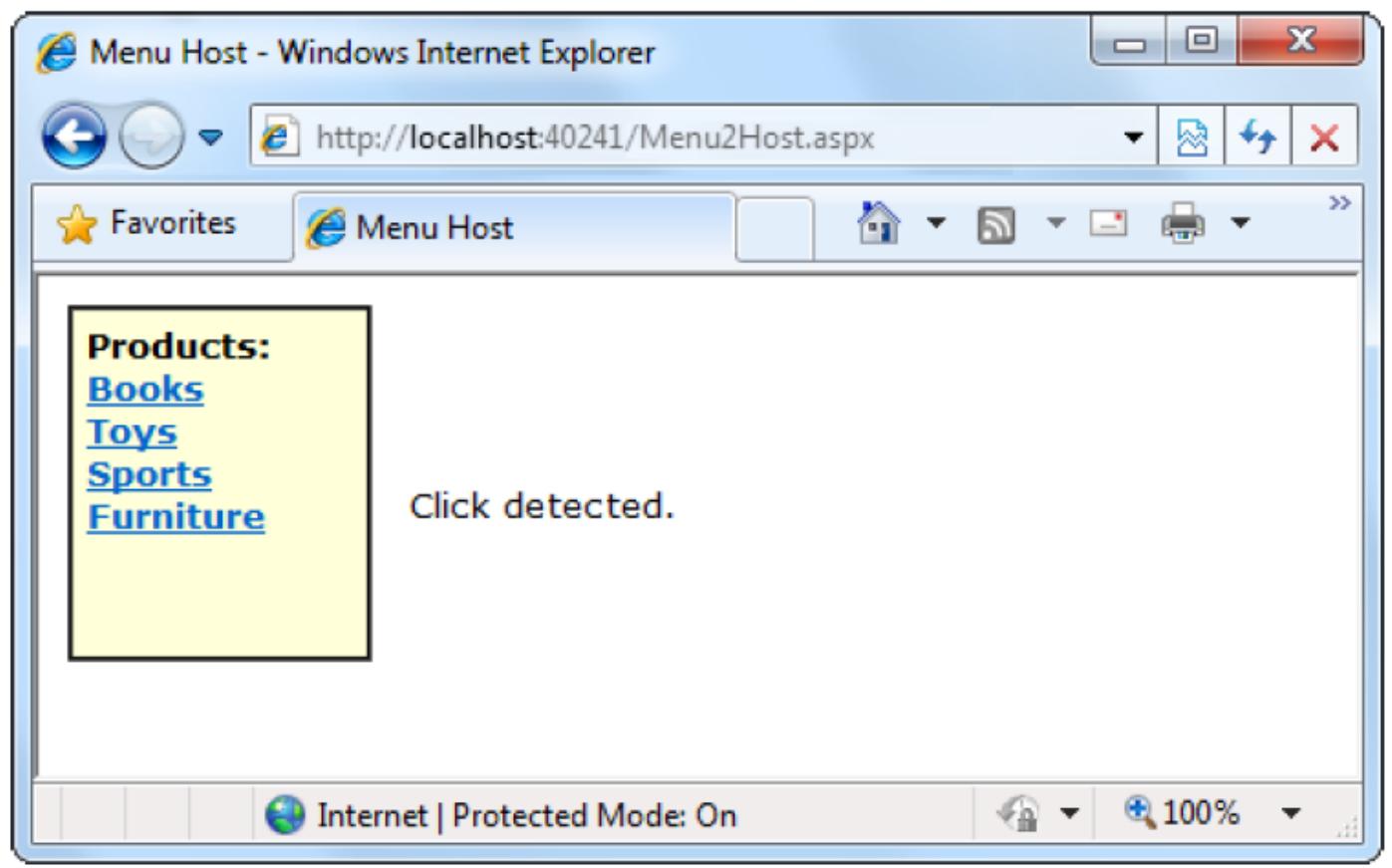
می‌توانید صفحه‌ای ایجاد کنید که از کنترل LinkMenu2 استفاده کند و سپس یک evenr-handler به آن اضافه نمایید. متأسفانه نمی‌توانید این event-handler ها را با استفاده از پنجره Properties موجود در VS به رویداد های تعریف شده توسط خودتان متصل کنید. بنابراین باید تگ LinkMenu2 را بصورت دستی تغییر دهید:

```
<KaraMoozesh:LinkMenu2 id="links" runat="server" OnLinkClicked="LinkClicked" />
```

در زیر event-handler ای که در صفحه وب موجود می‌باشد آمده است:

```
protected void LinkClicked(object sender, EventArgs e)
{
    lblClick.Text = "Click detected.";
}
```

 نکته: شما قادر به ایجاد event-handler ها در پنجره properties موجود در VS نیستید. بجای آن باید even-handler را بصورت دستی تایپ کنید.



## ارسال اطلاعات با استفاده از رویدادها

در نمونه بالا، هیچ اطلاعات سفارشی از طریق رویداد ارسال نشد. در بسیاری از موارد، شما می‌خواهید که اطلاعات بیشتری که مرتبط با رویداد می‌باشد را event-handler ارسال نمایید. برای این کار باید یک کلاس خاص که از EventArgs ارث بری می‌کند ایجاد کنید.

کلاس LinkClickedEventArgs، باید URL ای که کاربر از لیست Url ها انتخاب کرده است را ارسال کند. این کلاس همچنین یک property با نام Cancel فراهم می‌کند. اگر با true مقدار دهی شود، UC، پردازش خود را فوراً متوقف می‌کند. اما اگر false باشد، کار را به صفحه جدید می‌برد.

```
public class LinkClickedEventArgs : EventArgs
{
    public string Url { get; set; }

    public bool Cancel { get; set; }

    public LinkClickedEventArgs(string url)
    {
        Url = url;
    }
}
```

برای استفاده از کلاس سفارشی EventArgs، باید تعریف رویداد LinkClicked را تغییر دهید. بنابراین این رویداد از شی استفاده LinkClickedEventArgs خواهد نمود.

```
public event EventHandler<LinkClickedEventArgs> LinkClicked;
```

با استفاده از نوع generic شما می‌توانید متناسب با نوع نیاز خود در هر بار یک کلاس را معرفی کنید که ما در اینجا کلاس LinkClickedEventArgs را معرفی کردیم.

در مرحله بعد، UC شما برای انتشار رویداد نیاز به submit کردن اطلاعات مورد نیاز در زمان فراخوانی رویداد می‌باشد. اما چگونه متوجه می‌شود کدام لینک کلیک شده است؟ برای این کار باید بجای رویداد

استفاده LinkButton.Command از رویداد Command را مقدار Click بصورت خودکار CommandArgument که در تگ تعریف شده است را می‌گیرد:

```
<asp:LinkButton ID="lnkBooks" runat="server" CommandArgument="Menu2Host.aspx?product=Books"
    OnCommand="lnk_Command"> Books
</asp:LinkButton>
<br />
<asp:LinkButton ID="lnkToys" runat="server" CommandArgument="Menu2Host.aspx?product=Toys"
    OnCommand="lnk_Command"> Toys
</asp:LinkButton>
<br />
<asp:LinkButton ID="lnkSports" runat="server" CommandArgument="Menu2Host.
aspx?product=Sports"
    OnCommand="lnk_Command"> Sports
</asp:LinkButton>
<br />
<asp:LinkButton ID="lnkFurniture" runat="server" CommandArgument="Menu2Host.
aspx?product=Furniture"
    OnCommand="lnk_Command">
    Furniture</asp:LinkButton>
```

کنترل LinkMenu2 می‌تواند لینک را در طول صفحه مانند زیر ارسال کند :

```

LinkClickedEventArgs args = new LinkClickedEventArgs((string)e.CommandArgument);

LinkClicked(this, args);

public event EventHandler<LinkClickedEventArgs> LinkClicked;

protected void lnk_Command(object sender, CommandEventArgs e)

{

    // One of the LinkButton controls has been clicked.

    // Raise an event to the page.

    if (LinkClicked != null)

    {

        // Pass along the link information.

        LinkClickedEventArgs args =

            new LinkClickedEventArgs((string)e.CommandArgument);

        LinkClicked(this, args);

        // Perform the redirect.

        if (!args.Cancel)

        {

            // Notice we use the Url from the LinkClickedEventArgs

            // object, not the original link. That means the web page

            // can change the link if desired before the redirect.

            Response.Redirect(args.Url);

        }

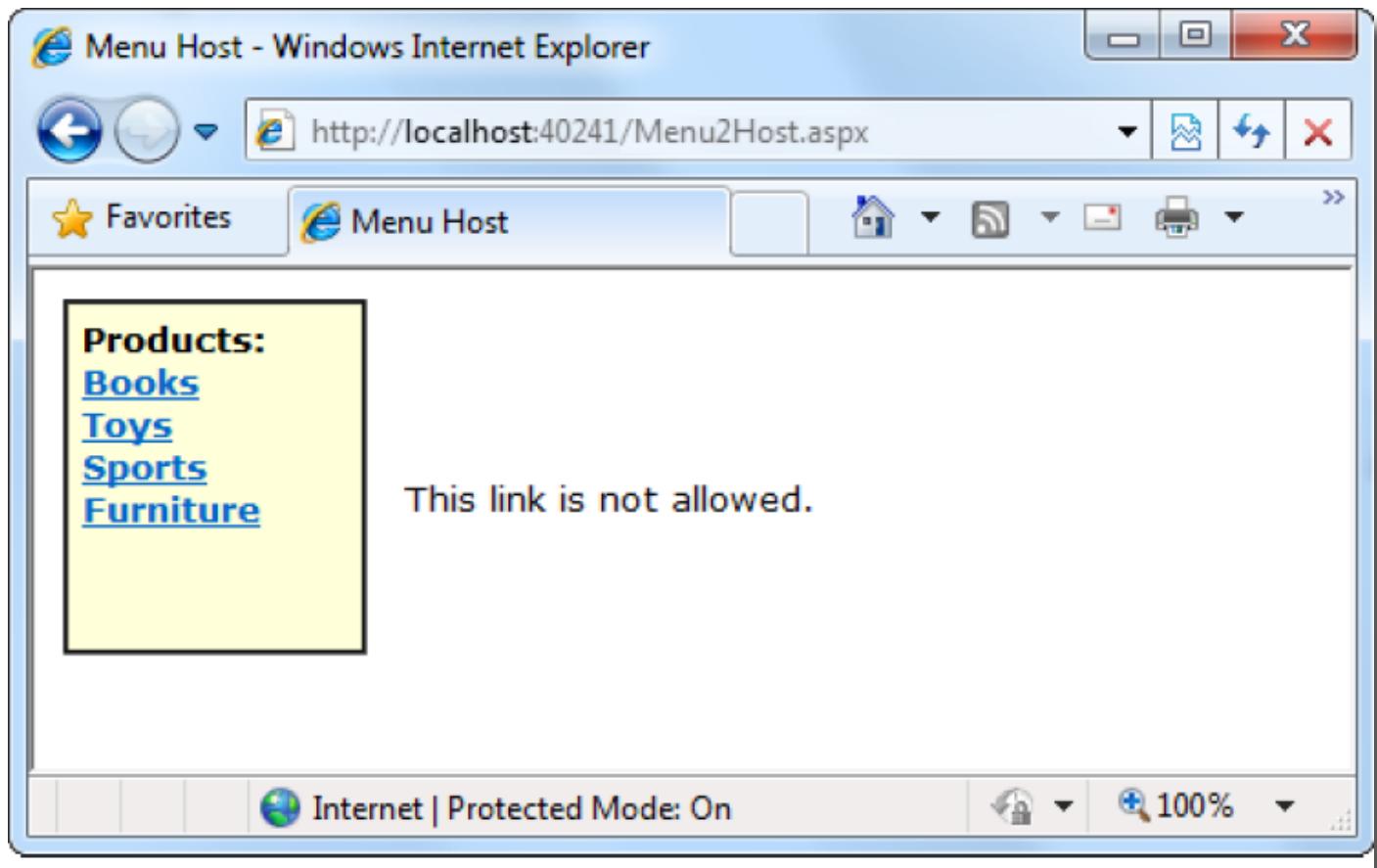
    }

}

```

در نهایت باید کد موجود در صفحه وب را تغییر دهید. Event-handler موجود در آن نیاز به signature جدید دارد.

```
protected void LinkClicked(object sender, LinkClickedEventArgs e)
{
    if (e.Url == "Menu2Host.aspx?product=Furniture")
    {
        lblClick.Text = "This link is not allowed.";
        e.Cancel = true;
    }
    else
    {
        // Allow the redirect, and don't make any changes to the URL.
    }
}
```



## گرافیک‌های پویا

یکی از ویژگی‌های دات نت فریمورک Windows می‌باشد که مجموعه‌ای از کلاس‌های طراحی شده برای رسم تصاویر می‌باشد. از GDI+ می‌توانید در برنامه‌های ویندوزی و یا تحت وب برای خلق گرافیک‌های پویا استفاده نمایید. در یک برنامه ویندوزی، گرافیکی که شما طراحی کرده اید در یک پنجره برای نمایش کپی می‌شود. در ASP.NET، کد شما گرافیک‌های مورد نظر شما را رندر و آنها را مستقیماً به مرورگر کلاینت می‌فرستد.

در حالت کلی، استفاده از GDI+ برای رسم یک گرافیک، کندر از استفاده از یک فایل تصویر آماده است. می‌توانید از ترکیب چند تصویر یک تصویر ترکیبی بدست آورید.

## HTML5 Canvas

اگر با HTML5 کار کرده باشید، می‌دانید که این زبان جدید یک امکان جدید با نام Canvas اضافه کرده است. یک ابزار رسم سمت کلاینت می‌باشد. این یک سطح رسم مستطیل شکل می‌باشد که شما در صفحه تعریف می‌کنید و سپس آن را توسط دستورات JS پرمی کنید.

به دلیل آنکه Canvas از JS استفاده می‌کند، رسم‌های خود را در سمت کلاینت انجام می‌دهد بنابراین ممکن است در همه مرورگرها به درستی پشتیبانی نشود و از نظر سرعت نیز از GDI+ کند تر می‌باشد.

البته به دلیل آنکه کار در کلاینت انجام می‌شود، برنامه‌نویسان می‌توانند تصاویر گرافیکی که به عمل‌های کاربرها واکنش نشان دهنده ایجاد کنند.

 نکته: GDI+، بهترین راه حل برای رندر کردن یک گرافیک ایستا و ثابت می‌باشد در حالیکه Canvas یک ابزار برای ایجاد اپلت‌های JS می‌باشد که در رسم تصاویر پویا استفاده می‌شود.

در زیر یک مثال را باهم می‌بینیم:

```
using System.Drawing;

protected void Page_Load(Object sender, EventArgs e)
{
    // Create an in-memory bitmap where you will draw the image.
    // The Bitmap is 300 pixels wide and 50 pixels high.
    Bitmap image = new Bitmap(300, 50);
```

```
// Get the graphics context for the bitmap.  
  
Graphics g = Graphics.FromImage(image);  
  
// Draw a solid yellow rectangle with a red border.  
  
g.FillRectangle(Brushes.LightYellow, 0, 0, 300, 50);  
  
g.DrawRectangle(Pens.Red, 0, 0, 299, 49);  
  
// Draw some text using a fancy font.  
  
Font font = new Font("Alba Super", 20, FontStyle.Regular);  
  
g.DrawString("This is a test.", font, Brushes.Blue, 10, 0);  
  
// Copy a smaller gif into the image from a file.  
  
// This code assumes smiley.gif is in the root website root.  
  
System.Drawing.Image icon =  
  
System.Drawing.Image.FromFile(Server.MapPath("smiley.gif"));  
  
g.DrawImageUnscaled(icon, 240, 0);  
  
// Render the entire bitmap to the HTML output stream.  
  
Response.ContentType = "image/gif";  
  
image.Save(Response.OutputStream,  
  
System.Drawing.Imaging.ImageFormat.Gif);  
  
// Clean up.  
  
g.Dispose();  
  
image.Dispose();  
}
```



آموزش کار با این تصاویر در حوزه آموزش این کتاب نمی‌باشد و برای کار با آن باید منابع دیگری را مطالعه نمایید.

## Chart Control

این کنترل‌ها، در پشت صحنه از توابع GDI+ برای ایجاد تصویری که شما در صفحه وب خواهید دید استفاده می‌کنند. بنابراین می‌توانید نمودارها را با داده صحیح پر کنید.

### ایجاد یک نمودار ابتدایی

با اضافه نمودن یک نمودار به صفحه VS آن را در صفحه ثبت می‌کند.

```
<%@ Register Assembly="System.Web.DataVisualization, Version=4.0.0.0, ..." TagPrefix="asp" %>
```

VS، همچنین می‌تواند یک نمونه جدید از کنترل chart را با استفاده از کدهای markup زیر به صفحه اضافه کند.

```

<asp:Chart ID="Chart1" runat="server">

    <series>
        <asp:Series Name="Series1">
        </asp:Series>
    </series>

    <chartareas>
        <asp:ChartArea Name="ChartArea1">
        </asp:ChartArea>
    </chartareas>

</asp:Chart>

```

می بینید که نمودار به دو بخش تقسیم شده است. بخش اول `<Series>` می باشد که داده هایی که می خواهید در نمودار قطعه بندی کنید و چگونگی نمایش آنها را پیکربندی کنید، ارائه می دهد. این جایی است که شما هر چیزی مانند نوع نمودار (خطی، دایره ای و ...)، متن محورها، استایل نشان دهنده های نقاط داده ای و ... را مشخص می کنید.

بخش دوم `<ChartAreas>` می باشد که به شما اجازه می دهد روی باقی نمودار اثر بگذارد. به عبارت دیگر، ناحیه ای که در پشت و اطراف داده شما ظاهر می شود در این بخش می باشد. از این بخش برای مواردی مانند افکت های سه بعدی، ابعاد محورها، سایه و ... می توان استفاده کرد.

در نمونه زیر یک نمودار دایره ای (Pie) را می بینید:

```

<h3>
    .NET Fans - Favorite Pastime Survey</h3>

<asp:Chart ID="passtimeChart" runat="server">

    <series>
        <asp:Series ChartType="Pie">
        <Points>
            <asp:DataPoint AxisLabel="Programming the Web" YValues="42" />
            <asp:DataPoint AxisLabel="Building Robots" YValues="18" />

```

```

<asp:DataPoint AxisLabel="Playing Video Games" YValues="17" />
<asp:DataPoint AxisLabel="Playing Foosball" YValues="5" />
</Points>
</asp:Series>
</series>

<chartareas>
<asp:ChartArea Name="chartArea">
</asp:ChartArea>
</chartareas>
</asp:Chart>

```

برای ساخت این نمودار، با ایجاد یک مجموعه (series) از داده ها شروع می کنیم. زمانی که series را تعریف کردید، نوع نمودار را با تعیین ویژگی ChartType از بین ۳۵ نوع نمودار انتخاب خواهید کرد.

```

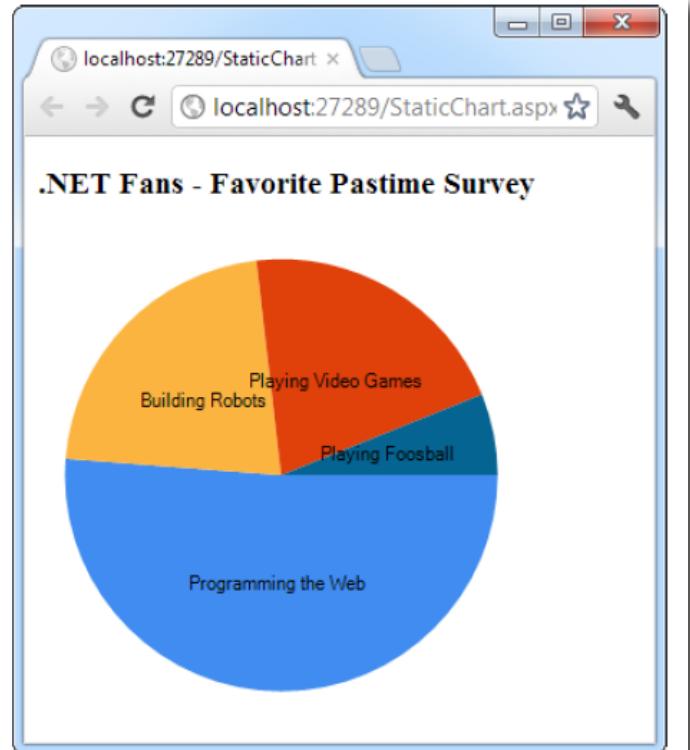
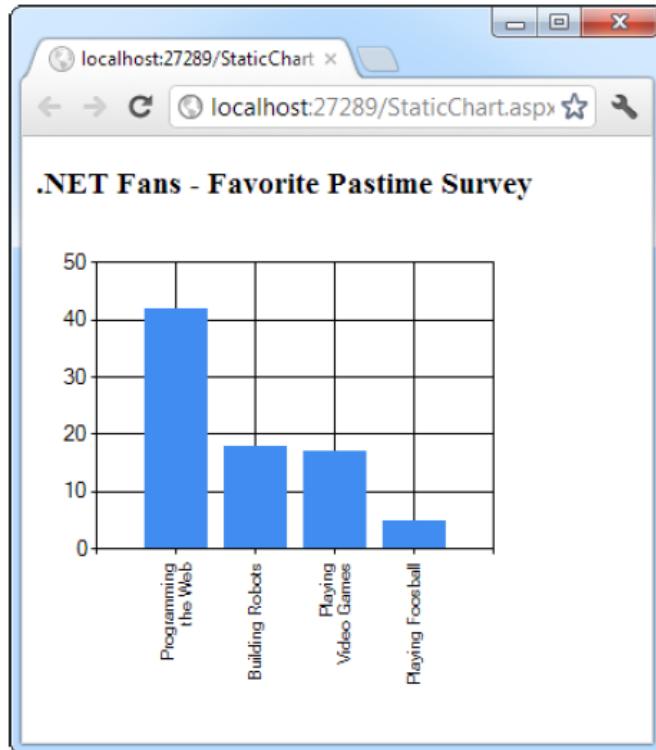
<asp:Series ChartType="Pie">
...
</asp:Series>

```

 نکته: اکثر نمودارهای پیچیده دارای چندین series می باشند. برای نمونه اگر یک نمودار خطی (Line) ایجاد کرده باشید که تعداد فروش ها را در چند فروشگاه در طول زمان نشان می دهد، هر فروشگاه دارای خط خودش می باشد و هر خط یک series جداگانه می باشد.

درون series، مجموعه ای از نقاط داده ای (data point) را ایجاد می کنید. برای ایجاد یک نمودار دایره ای ۴ بخشی (Four Slice)، نیاز به ۴ نقطه داده ای دارید. البته می توانید با تغییر یک ویژگی series، وضعیت نمایش نمودار را به نمودار ستونی تبدیل کنید.

```
<asp:Series ChartType="Column">
```



زمانی که این مثال را اجرا و به HTML تولید شده نگاه کنید، خواهید یافت که نمودار شامل تگ که مانند زیر به نظر می‌رسد، خواهد بود.

```

```

ASP.NET، یک تگ که اشاره به صفحه‌ای دارد که نمودار شما را ایجاد خواهد کرد، به صفحه اضافه می‌کند. در این روش ASP.NET، از یک صفحه مجازی بجای یک فرم واقعی که در وب سایت شما ذخیره شده است استفاده خواهد نمود. (به عبارت دیگر، فایلی با نام ChartIMG.axd، وجود ندارد. زمانی که ASP.NET درخواست را دریافت می‌کند، می‌داند که باید شروع به تقاضای اجرای نمودار کند، بنابراین کدهای رسم نمودار مربوطه را اجرا خواهد کرد)

### استفاده از سایر روش‌ها برای دریافت داده نمودار

با اختصاص یک نام برای series می‌توانید از درون کد به آن دسترسی داشته باشید و کلیه نقاط داده‌ای مورد نیاز را به آن اضافه کنید. برای نمونه اگر series شما شبیه زیر باشد :

```
<asp:Series Name="passtimes" ChartType="Pie" />
```

می توانید از طریق برنامه نویسی، نمودار مشابهی ایجاد کنید :

```
protected void Page_Load(Object sender, EventArgs e)
{
    // Find the right series.

    var passtimeSeries = passtimeChart.Series["passtimes"];
    // Add the points to the series.

    passtimeSeries.Points.AddXY("Programming the Web", 42);
    passtimeSeries.Points.AddXY("Building Robots ", 18);
    passtimeSeries.Points.AddXY("Playing Video Games", 17);
    passtimeSeries.Points.AddXY("Playing Foosball", 5);
}
```

در اغلب موارد، داده ای که می خواهد بر اساس آن نمودار رسم کنید از یک منبع خارجی مانند فایل یا بانک اطلاعاتی می آید.

در ادامه این کتاب، روش خواندن از هردو منبع را خواهید آموخت. پس از خواندن این داده ها از منبع، باید آنها را با استفاده از تکنیک `databinding` به `series` درون یک نمودار کپی کنید. اگر داده ها را از بانک اطلاعاتی بخوانید از روش هایی مانند `databinding` نیز می توانید برای کپی کردن داده ها در نمودار بدون نوشتگر بیش از یک یا دو خط کد، استفاده کنید.

## بخش دوم: ساخت فرم های وب بیتتر

Validation = فصل ششم: اعتبارسنجی

Master Page: کنترل صفحه مخفی

User Control  فصل هشتم

Master Page , Styles, Themes  فصل نهم:



## Master Pages و Styles, Themes

با استفاده از تکنیک هایی که تا کنون با آنها آشنا شده اید، می‌توانید یک صفحه ساده با امکان حرکت کاربر به صفحات دیگر تولید کنید. برای یکپارچه کردن صفحات وب خود درون یک وب سایت واحد، نیاز به چند ابزار دارید. که با سه تا از مهم ترین آنها یعنی آشنا خواهید شد. masterpage و sysle,theme

Style ها، بخشی از استاندارد CSS می‌باشند که مستقیماً به ASP.NET متصل نیستند. با استفاده از استایل‌ها می‌توانید مجموعه ای از گزینه‌های فرمت دهی را یکبار تعریف کنید و از آن‌ها برای فرمت دهی المان‌های مختلف در صفحات گوناگون استفاده نمایید. حتی می‌توانید استایل‌هایی را تعریف نمایید که کار خود را اتوماتیک انجام می‌دهند – برای نمونه استایلی که فونت همه متن‌های وب سایت شما را بدون نیاز به تغییر کد هیچ یک از صفحات، تغییر می‌دهد.

به دلیل آنکه استایل‌ها بر اساس استاندارد HTML بنا شده است، بنابراین هیچ درکی از مفاهیم ASP.NET مانند ویژگی‌های (Property) مربوط به کنترل‌ها ندارد. برای پر کردن این فاصله، ASP.NET، دارای امکانی به نام themes، می‌باشد که نقشی مانند استایل‌ها را اما فقط با کنترل‌های سروری بازی می‌کند.

امکان دیگر برای استانداردسازی وب سایت‌ها masterpage ها می‌باشد. بصورت خاص یک masterpage، یک صفحه آبی برای وب سایت شما می‌باشد که با استفاده از آن می‌توانید web page layout خود را تعریف کنید و آن را با جزئیاتی مانند .header، .Footer، .content page و بنرهای کامل کنید. هنگامی که دارای یک masterpage حرفه‌ای هستید، می‌توانید از آن برای ایجاد Menu (صفحات موجود در سایت) استفاده نمایید. هر content page layout مربوطه به masterpage ای که به آن متصل شده است را به خود می‌گیرد.

## استایل ها

استاندارد CSS کلیه مرورگرها را پشتیبانی می‌کند. (با اندکی تفاوت). CSS، بازه وسیعی از فرمت دهی سازگار با هر المان HTML را در اختیار قرار می‌دهد. استایل‌ها به شما اجازه اضافه نمودن border، تعیین جزئیات فونت، margin (حاشیه) و... را می‌دهند.

در ادامه، استانداردهای پایه‌ای CSS را خواهید آموخت و مشاهده خواهید کرد که چگونه کنترل‌های وب از CSS برای فرمت دهی خودشان بهره می‌برند.



**نکته:** CSS دارای نسخه‌های مختلفی می‌باشد (در حال حاضر 3 CSS) و دارای ویژگی‌ها و امکاناتی می‌باشد که توسط همه مرورگرها ممکن است اجرا نشوند. سایت ([Http://caniuse.com](http://caniuse.com))، یک منبع خوب برای تعیین اینکه کدام نسخه از مرورگرها، کدام یک از ویژگی‌های CSS را پشتیبانی می‌کنند، می‌باشد.

## أنواع استایل

صفحات وب می‌توانند از استایل‌ها به سه روش استفاده نمایند :

- **Inline style** : استایلی که مستقیماً درون یک تگ HTML نوشته می‌شود که فقط برای آن المان خاص بکار می‌رود. البته می‌توانید آن را پاک و درون sheet قرار دهید.

- **Internal Style Sheet** : مجموعه‌ای از استایل‌ها که در بخش <head> صفحه شما قرار می‌گیرند.

- **External Style Sheet** : شبیه روش قبلی می‌باشد ولی همه استایل‌ها درون یک فایل جدا قرار گرفته و در صفحه فراخوانی می‌شوند.

## ایجاد یک Inline Style

برای بگارگیری یک استایل روی یک المان HTML، می‌توانید خصیصه style آن را مقدار دهی کنید.

```
<p style="background: Blue">This text has a blue background.</p>
```

هر استایل شامل لیستی از یک یا چند ویژگی فرمت دهی می‌باشد. در مثال قبل استایل دارای یک ویژگی به نام background بود که با رنگ آبی مقدار دهی شد. برای اضافه کردن چندین ویژگی استایل، به سادگی می‌توانید آنها را با علامت (;) از هم جدا کنید:

```
<p style = "color:White; background:Blue; font-size:x-large; padding:10px">
```

```
This text has a blue background.</p>
```

این استایل یک متن بزرگ سفید در پس زمینه آبی با ۱۰ پیکسل فضا با لبه المان (کادر آبی) بوجود می‌آورد.

 نکته: لیست کامل ویژگی های فرمت دهی را می توانید در سایت زیر مشاهده کنید

<http://www.w3schools.com/css>

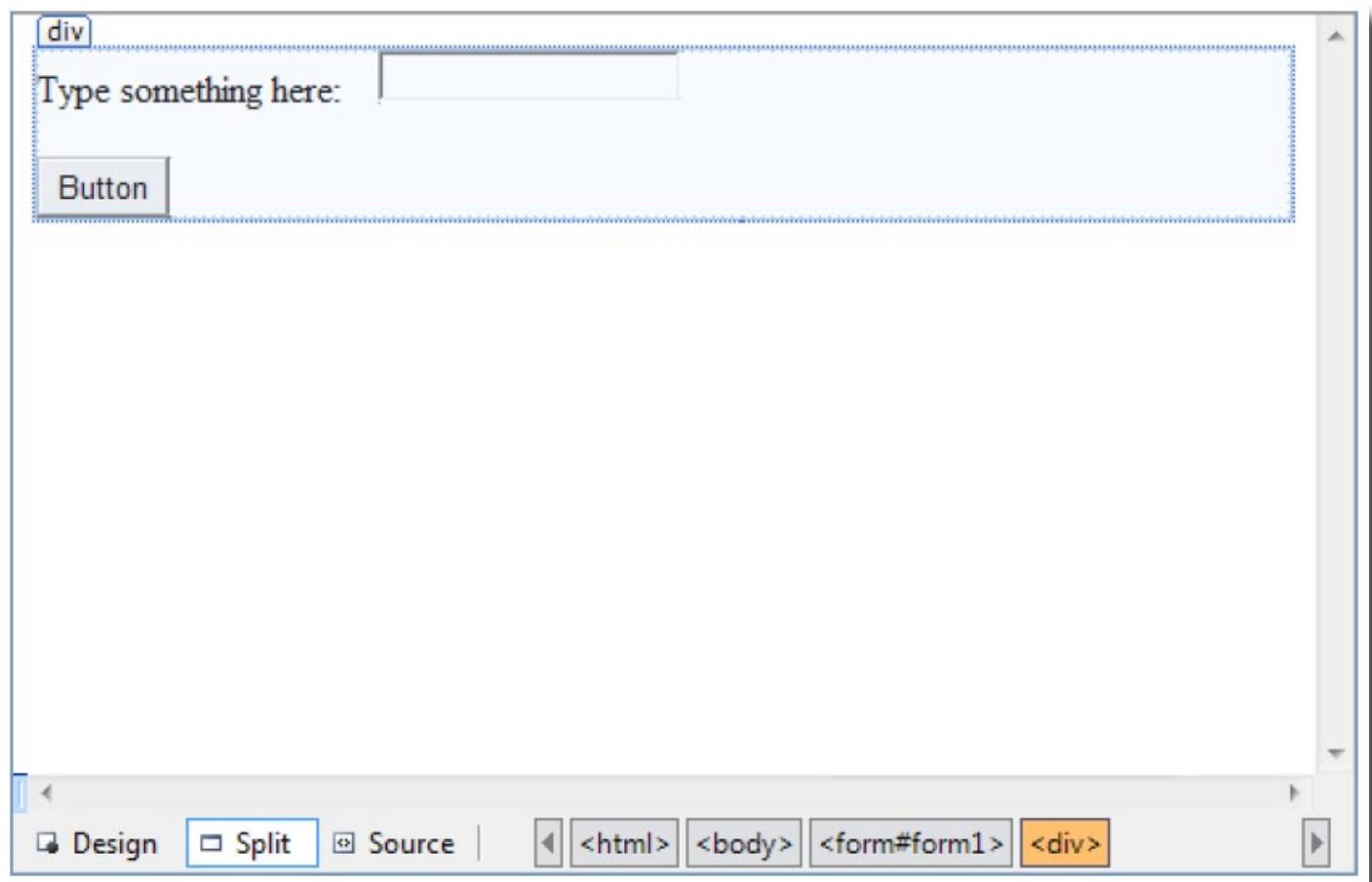
البته VS نیز، ابزارهایی برای کمک به شما در استفاده از این ویژگی ها دارد.

## Style Builder

VS، یک style builder که اجازه ایجاد استایل هایی توسط انتخاب آن در یک کادر جداگانه را خواهد داد، در اختیار شما قرار می دهد.

 نکته: CSS، یک ویژگی با نام inheritance، دارد که با استفاده از آن بعضی از ویژگی ها (مانند font family) از یک المان پدر به سایر المان های داخلی ارسال می شود. به عبارت دیگر، اگر font family را برای یک المان <div>، مقداردهی شود، همه المان های درون آن نیز با همان فونت مقداردهی خواهند شد (مگر آنکه بصورت واضح و جداگانه فونت آنها را تعیین کرده باشیم). سایر ویژگی ها مانند margin و padding از ارث بری استفاده نمی کنند.

فرض کنید می خواهید یک صفحه ساده که شامل چند کنترل مانند button, label, textbox و div می باشد را ایجاد کنید. قبل از فرمت دهی کردن صفحه، مطمئن شوید همه کنترل ها درون المان <div> باشند:



&lt;div&gt;

```

<asp:Label ID="Label1" runat="server"> Type something here:
</asp:Label>

<asp:TextBox ID="TextBox1" runat="server">
</asp:TextBox>

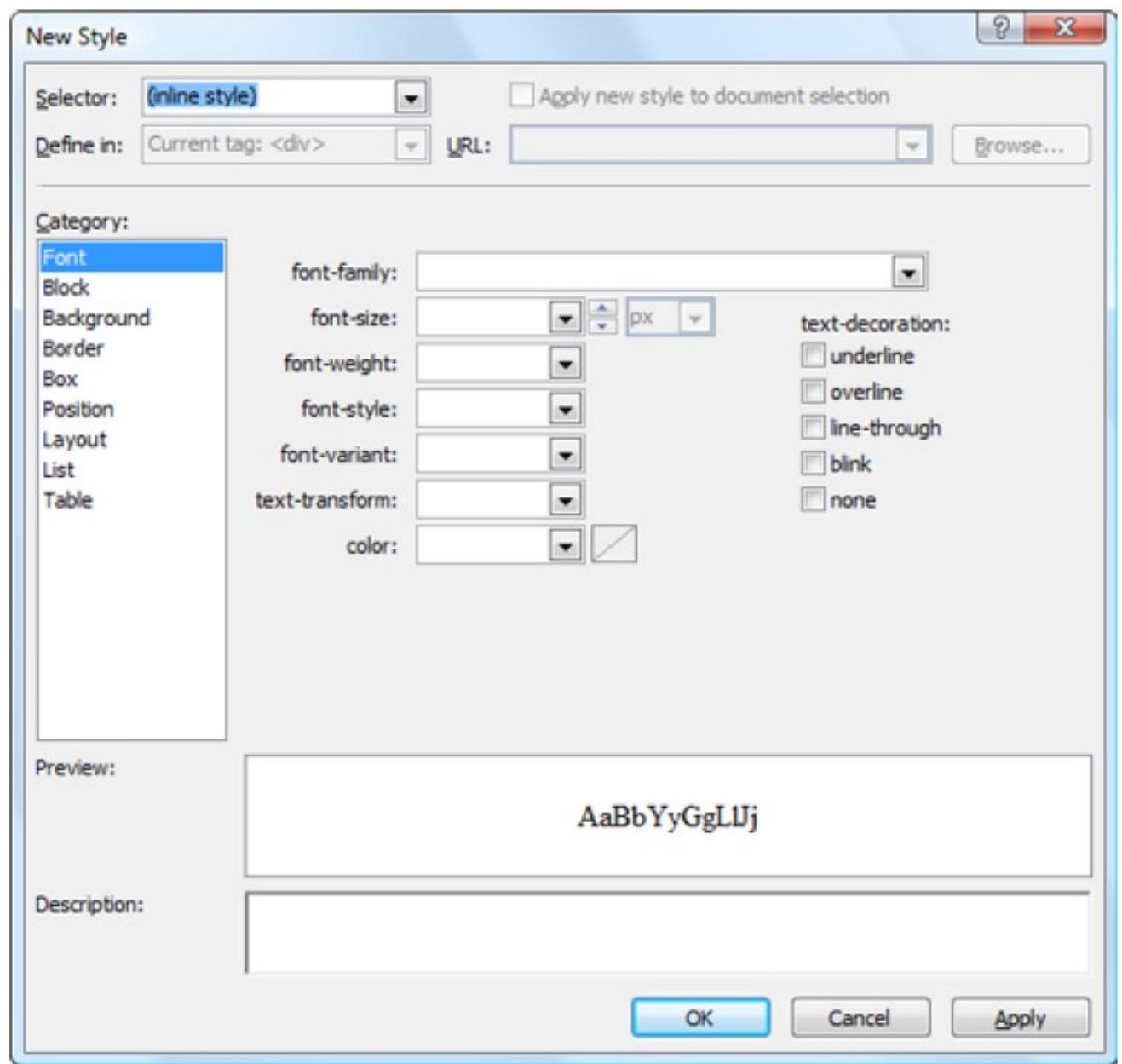
<br />
<br />

<asp:Button ID="Button1" runat="server" Text="Button"></asp:Button>

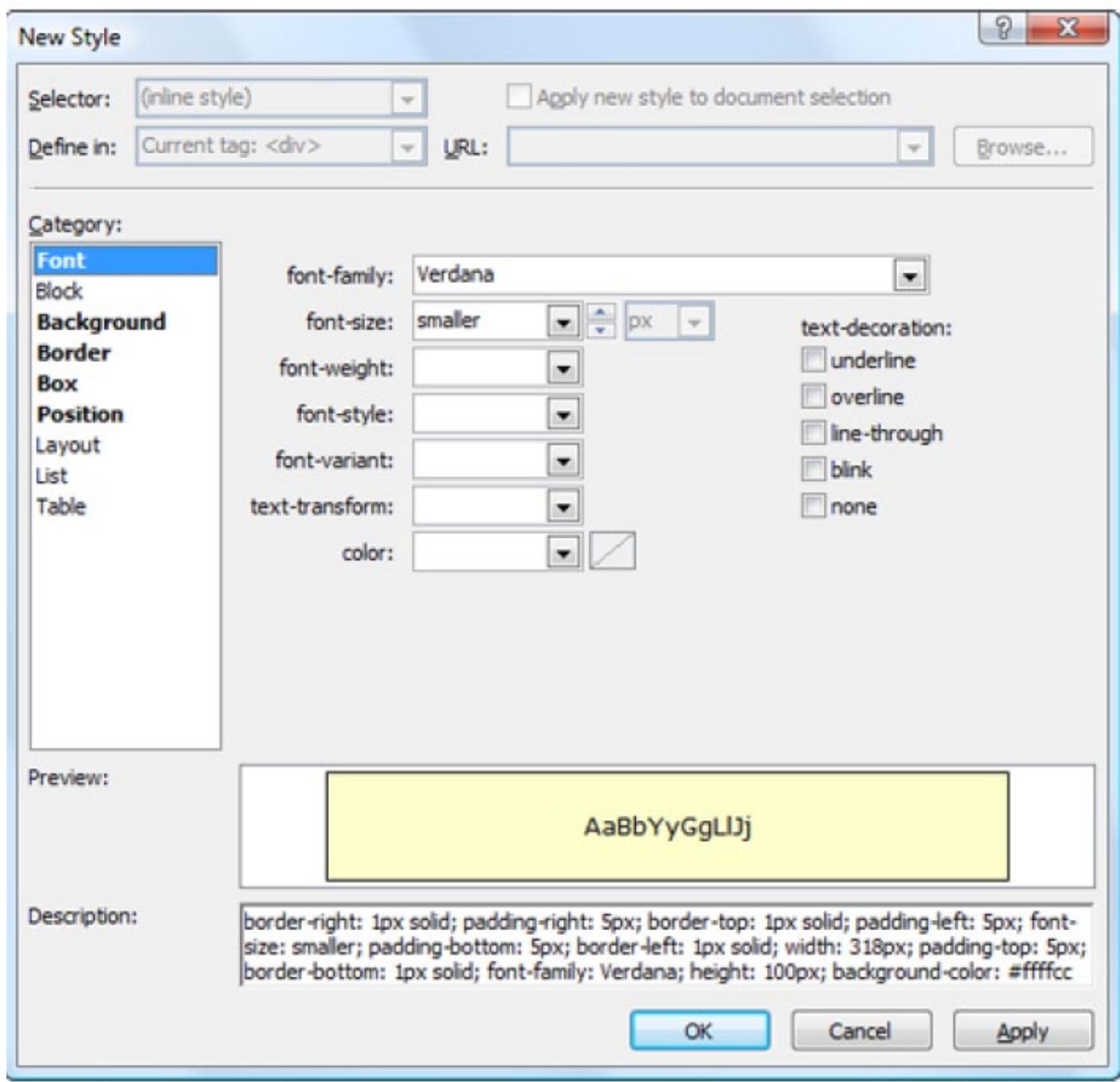
```

&lt;/div&gt;

در پنجره design، درون محلی در <div> (نه روی سایر کنترل‌ها) کلیک کنید. سپس از منو Formt > New Style را انتخاب کنید. کادر باز می‌شود. در کادر انتخابی بالای پنجره گزینه "inline style" را انتخاب کنید.



برای تنظیمات style، ابتدا باید یکی از گروه بندی ها را از category list انتخاب نمایید. برای نمونه اگر Font را انتخاب کنید، لیستی از تنظیمات فرمت دهنده مرتبط با فونت ها مانند font family, font size, text color و... را نشان می دهد.



زمانی که روی OK کلیک کردید، VS، اطلاعات استایل را به المان شما اضافه می کند.

```
<div style="border-style: solid; border-color: inherit; border-width: 1px; padding: 5px;
font-size: smaller; font-family: Verdana; background-color: #ffffcc">

<asp:Label ID="Label1" runat="server"> Type something here:
</asp:Label>

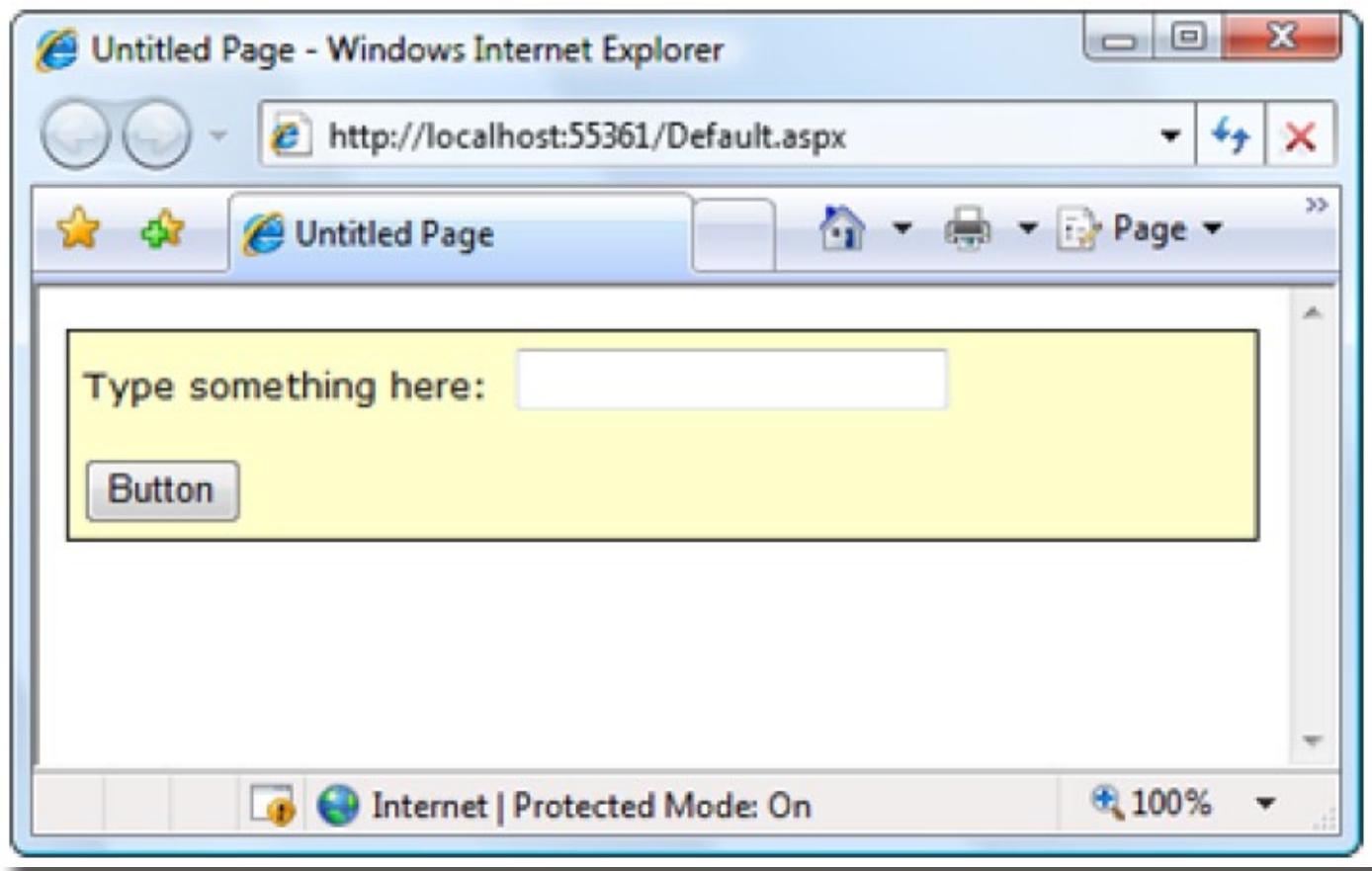
<asp:TextBox ID="TextBox1" runat="server">
</asp:TextBox>
```

```

<br />
<br />

<asp:Button ID="Button1" runat="server" Text="Button"></asp:Button>
</div>

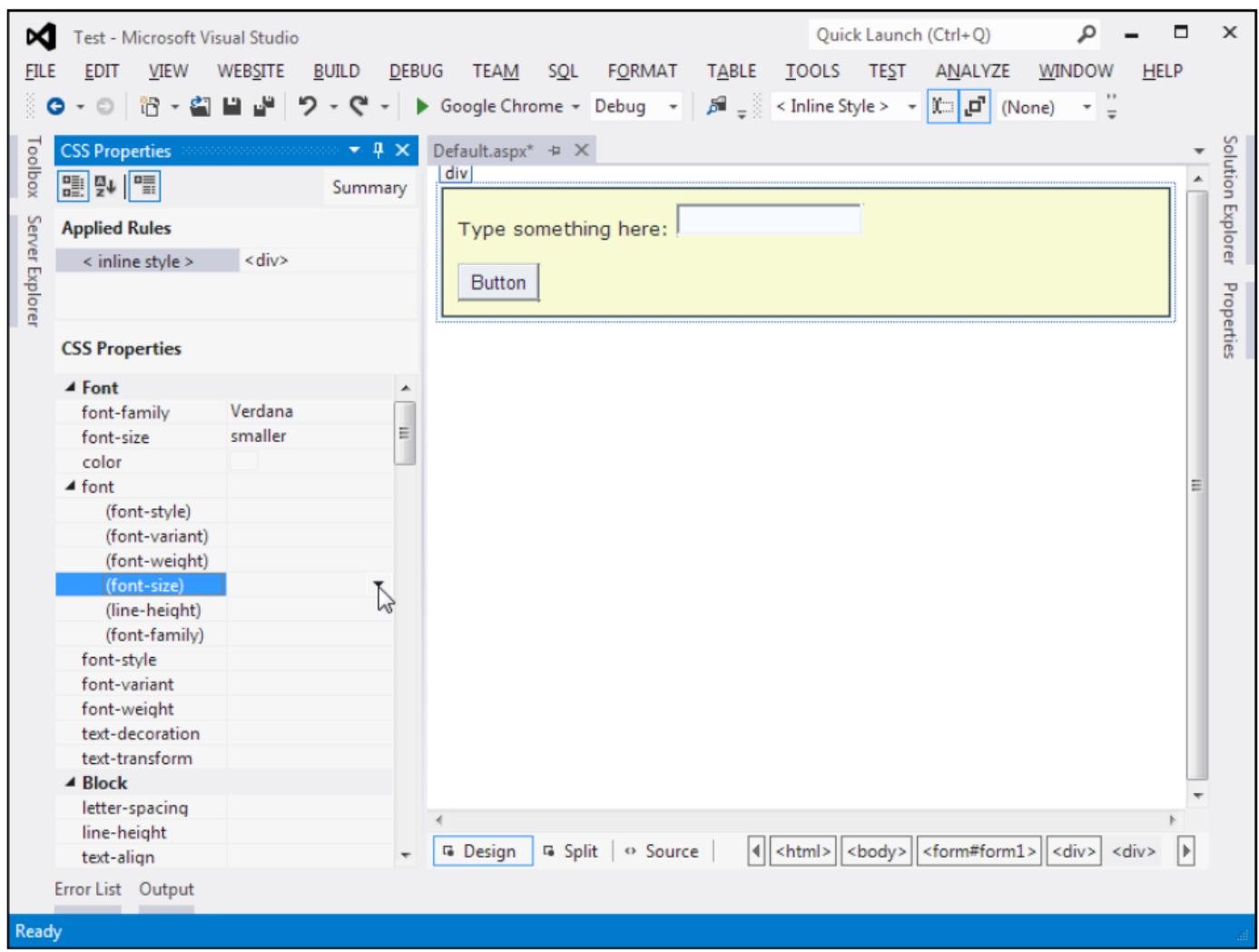
```



## پنجره CSS Properties

هنگامی که یک style را ایجاد می‌کنید، دو گزینه ساده برای تغییر آن در VS خواهد داشت.

برای نمایش پنجره CSS Properties، صفحه وب را در VS باز کنید و View > CSS Properties را انتخاب کنید.



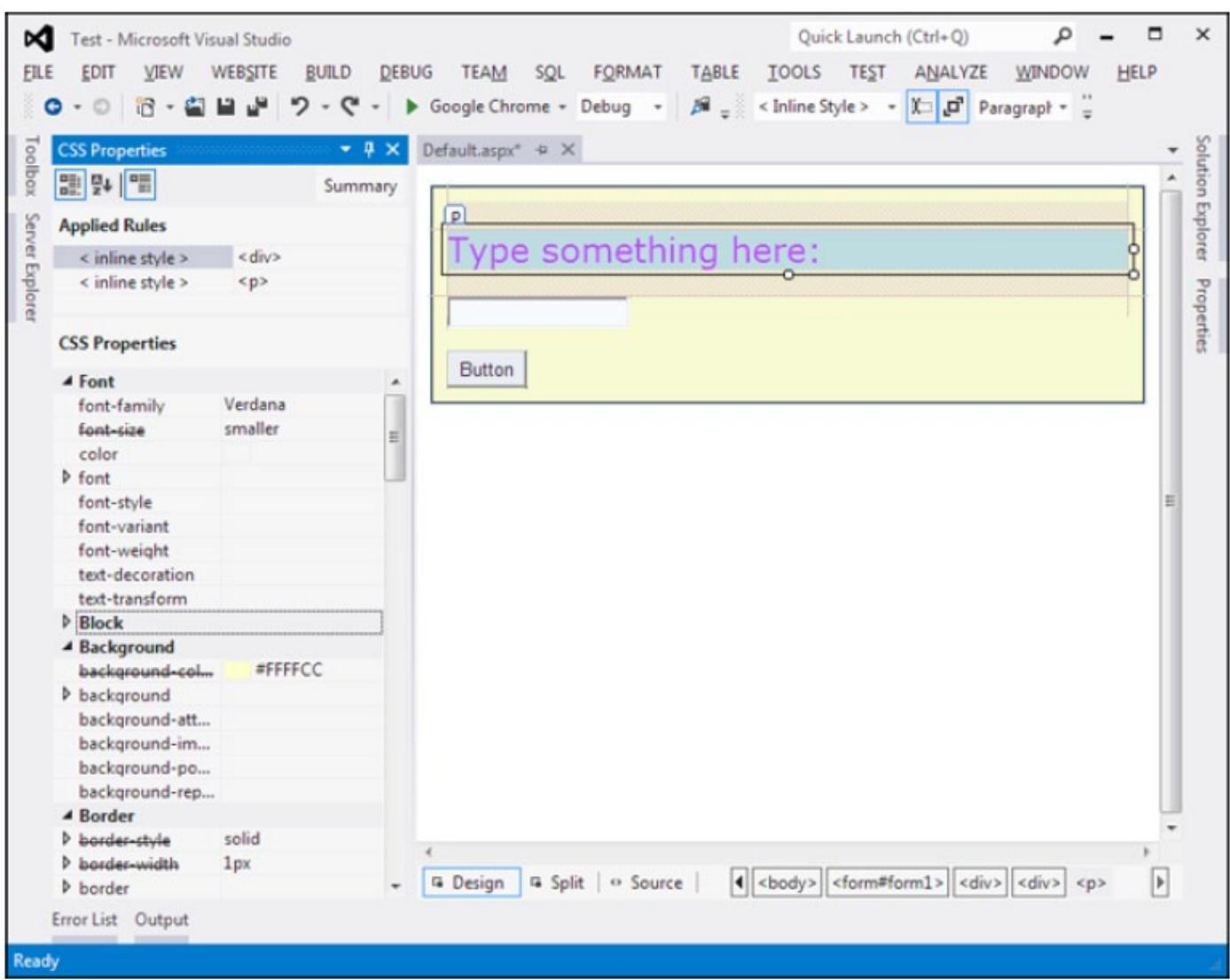
هنگامی که CSS Properties در دسترس باشد، می‌توانید از آن برای نمایش یکی از استایل‌های خود استفاده نمایید. ابتدا المان یا وب کنترلی که از خصیصه style استفاده می‌کند را پیدا کنید. سپس برای انتخاب آن در design view، روی آن کلیک کنید.

## Style Inheritance

پنجه CSS Properties، نه تنها استایل المان کنونی را نشان می‌دهد، بلکه هر استایلی که در المان‌های داخلی نیز استفاده شده است هم را نشان می‌دهد. برای نمونه، اگر به یک المان درون <div> نگاه کنید، استایل‌هایی که روی <div> بکار رفته اند را مشاهده خواهید کرد. اگر بیش از یک استایل در کار باشد (برای نمونه، المان درون <div> درون <div>) فرمت دهی شده دیگری قرار داشته باشد یا شما یک استایل به المان <body> اضافه کرده باشید)، همه این استایل‌ها را در لیست خواهید دید.

زمانی که استایل‌های ارث بری شده نمایش داده می‌شوند، پنجه CSS Properties بعضی اوقات یک خط قرمز روی نام property می‌کشد. این کار زمانی رخ می‌دهد که درون المان پدر مقداردهی شده باشد اما روی المان‌های داخلی بکار نرفته باشد.

برای نمونه، المان داخلی ممکن است آن ویژگی را با استایل خودش override کرده باشد. در زیر مثالی را می‌بینید که یک المان <p> استایل داده شده، درون یک <div> استایل داده شده، قرار گرفته است. استایلی که از <div> ارث بری کرده است، یک ویژگی font-family (که ارث بری شده است)، ویژگی font-size (که ارث بری شده ولی override شده است) و چند ویژگی border (که ارث بری نشده است) را تعریف می‌کند.



## ایجاد یک Style Sheet

برای ایجاد یک style sheet در VS Add new Item را از منو website>add new Item. می‌باید. یک نام برای آن تعیین و دکمه Add را کلیک کنید.

در چندین استایل را (که به عنوان rules شناخته می‌شوند) تعریف می‌کنیم. سپس می‌توانیم این قوانین را برای فرمت دهی کردن کنترل‌های HTML و ASP.NET اسفاده نماییم.

هر قانون مجموعه‌ای از پیش فرض‌های فرمت دهی، که چگونگی فرمت دهی کردن یک جز تنهای در صفحه وب را تعیین خواهد کرد. تعریف می‌کند.

برای نمونه، اگر بخواهیم یک قانون برای فرمت دهی کردن هدر تعریف کنیم، این کار را با تعریف قانون با یک نام واضح شروع می‌کنیم:

```
.heading1
```

```
{  
}
```

هر نام قانون، دارای دو بخش است. بخش اول که نشان دهنده المان HTML ای است که قانون روی آن پیاده می‌شود. در مثال بالا این بخش وجود ندارد بنابراین یعنی این قانون روی هر تگی می‌تواند بکار گرفته شود. بخش دوم، یک نام واحد می‌باشد (که css class نامیده می‌شود) که می‌توانید برای تعیین قانون آن را انتخاب نمایید. این نام case-sensitive name نامیده می‌شود.

هنگامی که قانون را تعریف کردید، می‌توانید فرمت دهی مناسب را به آن اضافه نمایید.

```
.heading1
{
    font-weight: bold;
    font-size: large;
    color: limegreen;
    font-family: Verdana, Arial, Sans-Serif;
}
```

همانطور که می‌بینید، سینتکس تعریف استایل در یک style sheet دقیقاً مشابه تعریف آن در یک استایل درون صفحه وب می‌باشد. همچنین می‌توانید قوانینی که در تگ‌های HTML بکار می‌روند را بصورت خودکار تعریف نمایید. برای انجام این کار، نام تگ را تعیین کنید. در زیر قانونی که روی همه تگ‌های `<h2>` اثر می‌گذارد را مشاهده می‌کنید:

```
h2
{
    ...
}
```

اگر می‌خواهید فرمتی را روی کل صفحه وب بکار بگیرید، می‌توانید یک قانون روی المان `<body>` تعریف کنید:

```
body
{
    ...
}
```

این کار یک راه مناسب برای تعیین فونت پیش فرض در اختیار شما قرار می‌دهد.

خوب‌بختانه، نیازی به نوشتن دستی این قوانین در style sheet نمی‌باشد. VS، به شما اجازه ساخت استایل‌ها را با استفاده از style builder که در قبل با آن آشنا شدید، می‌دهد.

یک استایل خالی با یک نام را اضافه کنید و بین {} کلیک راست و Build Style را انتخاب کنید.

```
body
{
    font-family: Verdana, Arial, Sans-Serif;
    font-size: small;
```

}

.heading1

{

font-weight: bold;

font-size: large;

color: lime;

}

.heading2

{

font-weight: bold;

font-size: medium;

font-style: italic;

color: #C0BA72;

}

.blockText

{

padding: 10px;

background-color: #FFFFD9;

border-style: solid;

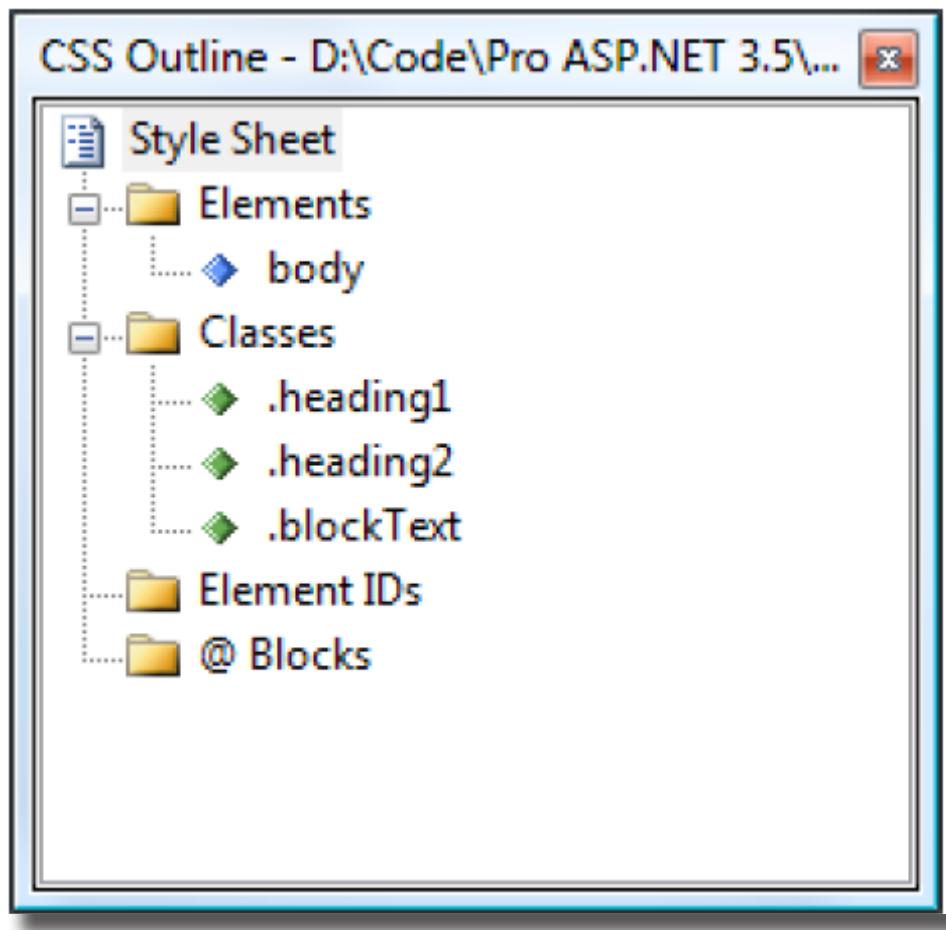
border-width: thin;

}

## CSS Outline پنجره

این پنجره یک دید کلی از قوانین موجود در style sheet به شما می‌دهد.

زمانی که style sheet را ویرایش می‌کنید، می‌توانید این پنجره را با انتخاب View>Other Windows>Document Outline مشاهده نمایید.



نام قوانین از نظر تکنیکی selector خوانده می‌شود، زیرا بخشی از سند HTML را که باید برای فرمت دهی انتخاب شود را تعیین می‌کند.

## بکارگیری قوانین Style Sheet

برای استفاده از یک قانون در یک صفحه وب، ابتدا باید آن صفحه را به style sheet مورد نظر لینک کنید. این کار را با اضافه کردن یک المان `<link>` به بخش `<head>` صفحه انجام می‌دهیم.

```

<><html>

<head id="Head1" runat="server">
    <title>...</title>
    <link href="StyleSheet.css" rel="stylesheet" />
</head>
<body>
    ...
</body>
</html>

```



نکته: برای اتصال یک صفحه وب به یک style sheet راه های زیادی موجود می باشد. می توانید در هنگام ویرایش یک صفحه وب، به مد رفته و Format>Attach Style Sheet را انتخاب کنید. این کار یک کادر محاوره ای که لیستی از style sheet های موجود در پروژه شما را نمایش می دهد و می توانید یکی را انتخاب و روی Ok کلیک کنید.

هنگامی که المان <link> را اضافه کردید، استایل شما در دسترس صفحه وب شما می باشد. می توانید هر کنترل ASP.NET یا HTML را به آن bind کنید. برای نمونه، اگر یک می خواهید یک کنترل heading1 از فرمت label استفاده نماید، باید ویژگی heading1 را با مقدار Label.CssClass برابر قرار دهید:

```
<asp:Label ID = "Label1" runat = "server" Text = "This Label Uses heading1"
    CssClass = "heading1"> </asp:Label>
```

همچنین می توانید ویژگی CssClass را از پنجره Properties مقدار دهی کنید.

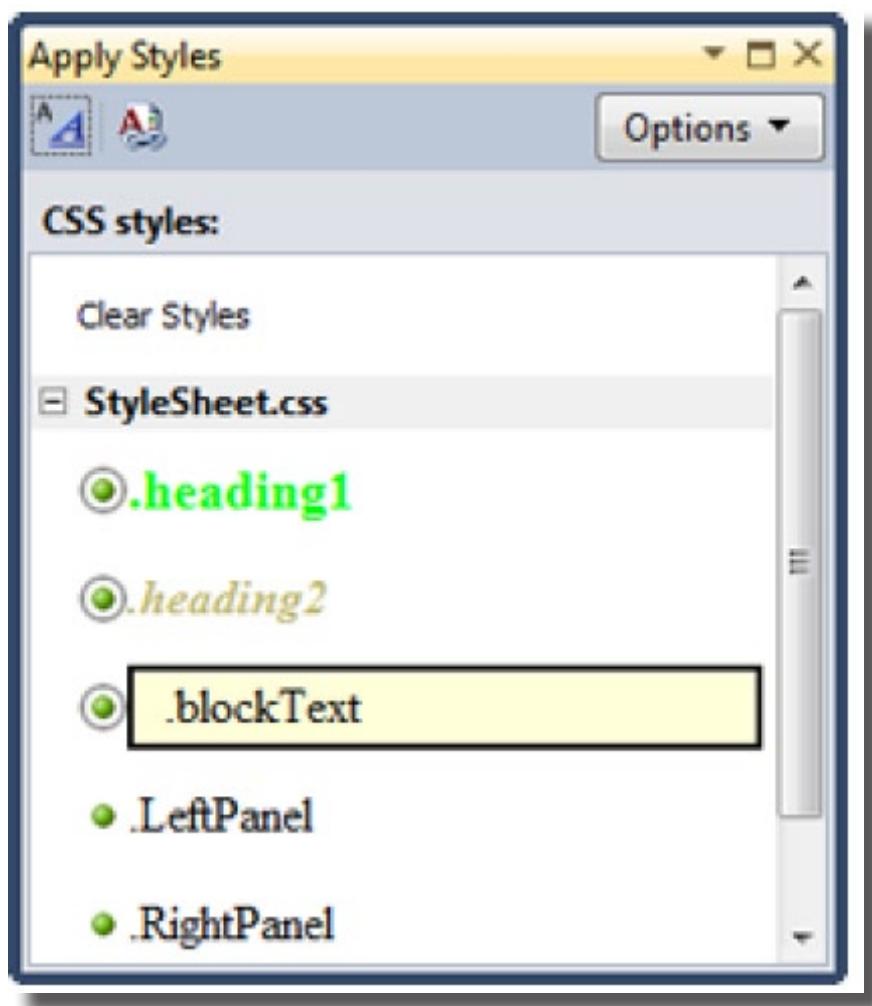
برای بکارگیری یک استایل روی بخشی از یک کنترل HTML باید ویژگی Class آن ر مقدار دهی کرد:

```
<div class="blockText" id="paragraph" runat="server">
    <p>
        This paragraph uses the blockText style.</p>
    </div>
```

## پنجره Apply Styles

برای اتصال استایل خود به المانی در صفحه وب، می‌توانید از این پنجره استفاده نمایید. برای نمایش این پنجره، صفحه وب خود را باز کنید و View > Apply Styles را انتخاب کنید. این پنجره در سمت چپ ToolBox نمایش داده می‌شود.

این پنجره، لیستی از همه استایل‌هایی که در style sheet مربوطه موجود می‌باشد را نمایش می‌دهد.



VS، به اندازه کافی باهوش می‌باشد تا راه مناسبی برای بکارگیری استایل بر اساس آن چیزی که در صفحه وب انتخاب کرده اید، تعیین کند:

- اگر یک کنترل وب را انتخاب کرده باشد، ویژگی CssClass را اضافه یا تغییر می‌دهد.
- اگر یک المان HTML را انتخاب کرده باشد، خصیصه Class را اضافه یا تغییر می‌دهد.
- اگر بخشی از یک محتوای HTML را انتخاب کرده باشد، یک المان <div> یا <span> (بر اساس نوع محتوای انتخاب شده) را اضافه می‌کند و سپس مقدار ویژگی Class آن را مقدار دهی می‌نماید.

## ایجاد استایل های بیشتر

در ابتدای این بخش، دیدید که چگونه می توانید از کادر New Style برای ایجاد یک inline style که کلیه فرمت دهی ها را برای المان مورد نظر (مانند یک `<div>`) درون تگ تعبیه می کند، استفاده نمایید.

برای استفاده از این پنجره در یک فایل style sheet، مراحل زیر را طی کنید:

- یک المان در design view و سپس گزینه Format>New Style را انتخاب کنید.
- در کادر "Define in"، گزینه "Editing Style Sheet" را انتخاب نمایید و فایل style sheet را از لیست URL ها انتخاب کنید.
- در کادر Selector، یک نام بر اساس استایلی که می خواهید ایجاد کنید وارد نمایید.
- در نهایت، را پیکربندی کنید و روی OK کلیک نمایید.

زمانی که این مراحل را طی کردید، اطلاعات استایل را به style sheet لینک شده اضافه می کند. شما حتی نیاز به باز کردن فایل CSS لینک شده نخواهید داشت.

## Themes

با توجه به سادگی CSS Style، شاید تعجب کنید که آیا به چیز بیشتری نیاز خواهید داشت یا نه. مشکل این است که قوانین CSS به مجموعه ای از خصوصیت های استایل محدود می شود. آنها به شما اجازه استفاده مجدد جزئیات فرمت دهی خاصی (font, border, foreground, color,...) ر می دهند اما قطعاً نمی توانند سایر جنبه های کنترل های ASP.NET را بکار ببرند.

برای نمونه، کنترل CheckBoxList، شامل ویژگی هایی می شود که نحوه قرار گرفتن آیتم ها در ردیف ها و ستون ها را مدیریت می کنند. این ویژگی ها روی جلوه های تصویری کنترل که خارج از ناحیه CSS هستند نیز تأثیر می گذارند، بنابراین باید آنها را دستی مقداردهی کنید. همچنین ممکن است بخواهید نحوه انتخاب کنترل تقویم را استاندارد کنید یا آن را در یک TextBox قرار دهید. این کار قطعاً با CSS امکان پذیر نمی باشد.

امکانی با نام themes، این فاصله را از بین می برد. مانند CSS theme ها به شما اجازه تعریف مجموعه ای از جزئیات استایل را که شما می توانید روی کنترل بکار بگیرید خواهند داد. اگرچه برخلاف CSS theme ها توسط مرورگرها پیاده سازی نمی شوند، ولی در عوض ASP.NET در هنگام ایجاد صفحه theme های شما را پردازش می کند.

 نکته: theme ها جایگزین استایل ها را نیستند. در عوض، آنها را تکمیل می کنند. استایل ها زمانی که شما می خواهید فرمت مشابهی را در کنترل های وب بکار گیرید مناسب می باشند. theme ها زمانی مناسب هستند که می خواهید ویژگی هایی از کنترل ها را که نمی توانید با CSS پیکربندی کنید را تنظیم کنید.

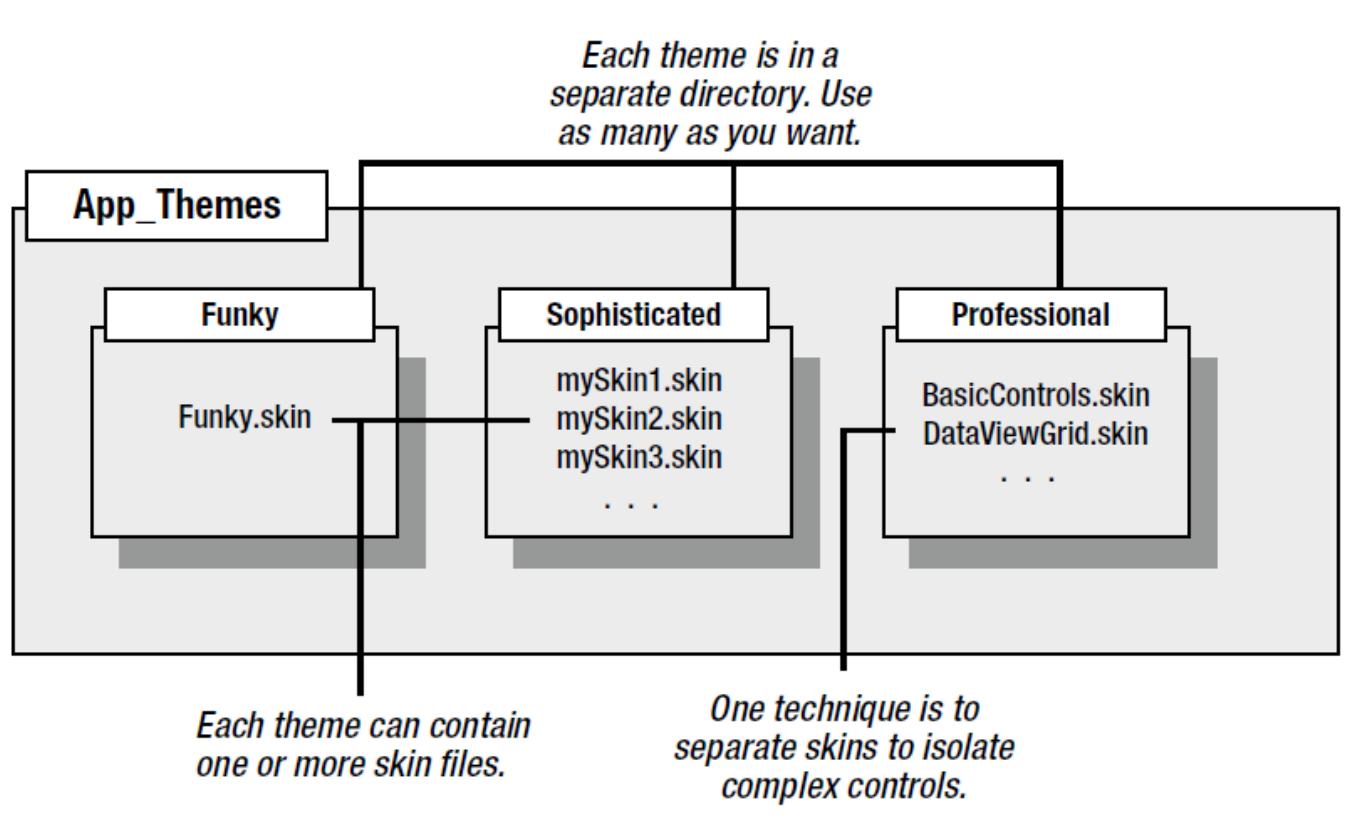
## ها چگونه کار می‌کنند : Theme

برای استفاده از theme ها درون یک برنامه تحت وب، باید یک پوشه برای آن ایجاد کنید. این پوشه باید در پوشه ای با نام App\_Themes قرار بگیرد که در پوشه ریشه پروژه وب می‌باشد. حداقل باید یک فایل skin در پوشه theme قرار دهید. این فایل یک فایل متنی با پسوند skin می‌باشد.

فایل skin شامل لیستی از تگ‌های کنترل می‌باشد. نیاز نیست این تگ‌ها کاملاً کنترل را تعریف کنند. آنها فقط باید ویژگی‌های مورد نظر شما را که می‌خواهید استاندارد شوند، مقداردهی کنند. برای نمونه، اگر بخواهید اگر بخواهید رنگ خاصی را تعیین کنید فقط باید BackColor و ForeColor را مقدار دهی کنید.

```
<asp:ListBox runat = "server" ForeColor = "White" BackColor = "Orange"/>
```

بخش runat="server" همیشه لازم است. بقیه موارد انتخابی می‌باشد. از اختصاص مقداری برای ID خودداری کنید زیرا صفحه ای که شامل ListBox می‌باشد باید یک نام واحد را برای کنترل تعریف کند. شکل زیر ارتباط بین skin و theme را به جزئیات نشان می‌دهد.

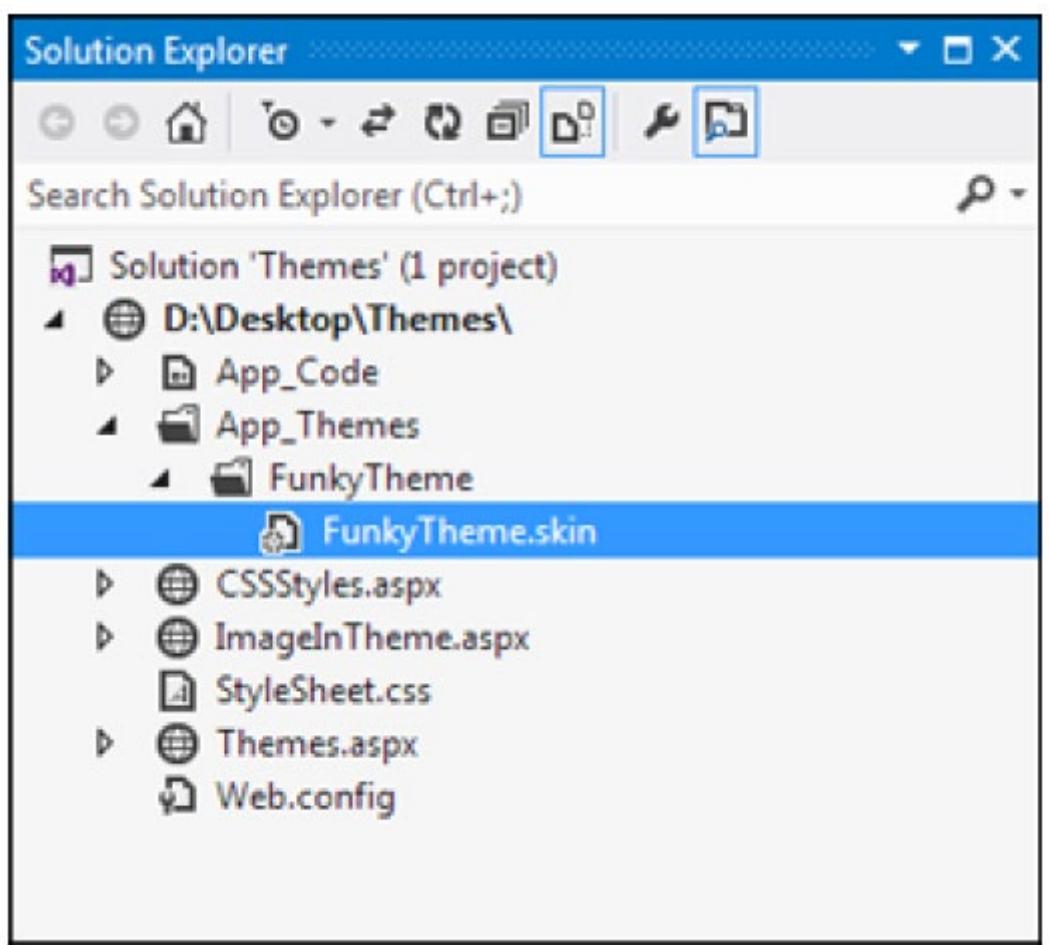


## بکارگیری یک Theme ساده

برای اضافه نمودن یک theme به پروژه، WebSite>Add New Item را انتخاب نمایید.

VS، به شما اخطار می‌دهد که فایل skin باید در زیرپوش‌های در پوشه App\_Themes قرار بگیرد. اگر Yes را انتخاب کنید، پوشه‌ای با نام مشابه با فایل theme شما درست می‌شود.

می‌توانید پوشه و فایل را rename کنید.



متاسفانه، VS، از ویژگی design-time theme برای ایجاد تگ‌های کنترل‌ها را از سایر صفحات وب به درون آن copy & paste کنید.

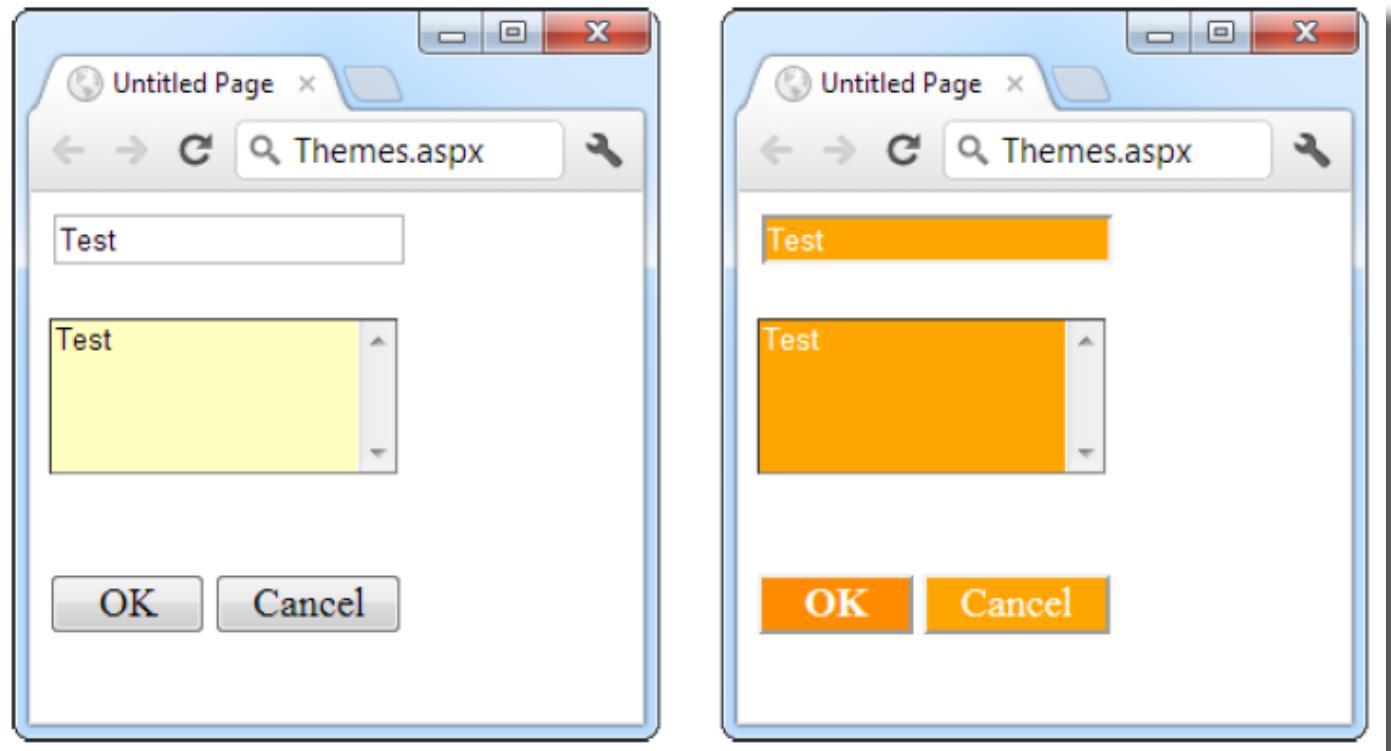
```
<asp:ListBox runat = "server" ForeColor = "White" BackColor = "Orange"/>
<asp:TextBox runat = "server" ForeColor = "White" BackColor = "Orange"/>
<asp:Button runat = "server" ForeColor = "White" BackColor = "Orange"/>
```

برای بکارگیری یک theme در صفحه وب، باید خصیصه theme از راهنمای صفحه را با نام پوشه theme خود مقدار دهی فرمایید.

```
<%@ Page Language = "#c" AutoEventWireup = "true" ... Theme = "FunkyTheme" %>
```

اکنون ASP.NET، بصورت خودکار، به دنبال پوشه ای با نام AppThemes\FunckyTheme می‌گردد و کلیه فایل‌های skin درون آن را اسکن می‌کند.

در تصویر زیر نتیجه بکارگیری FunkyTheme در یک صفحه ساده را مشاهده خواهید کرد. توجه فرمایید که تنظیماتی که تداخل دارند (مانند background listbox موجود برای overwrite بازنویسی) می‌شوند.

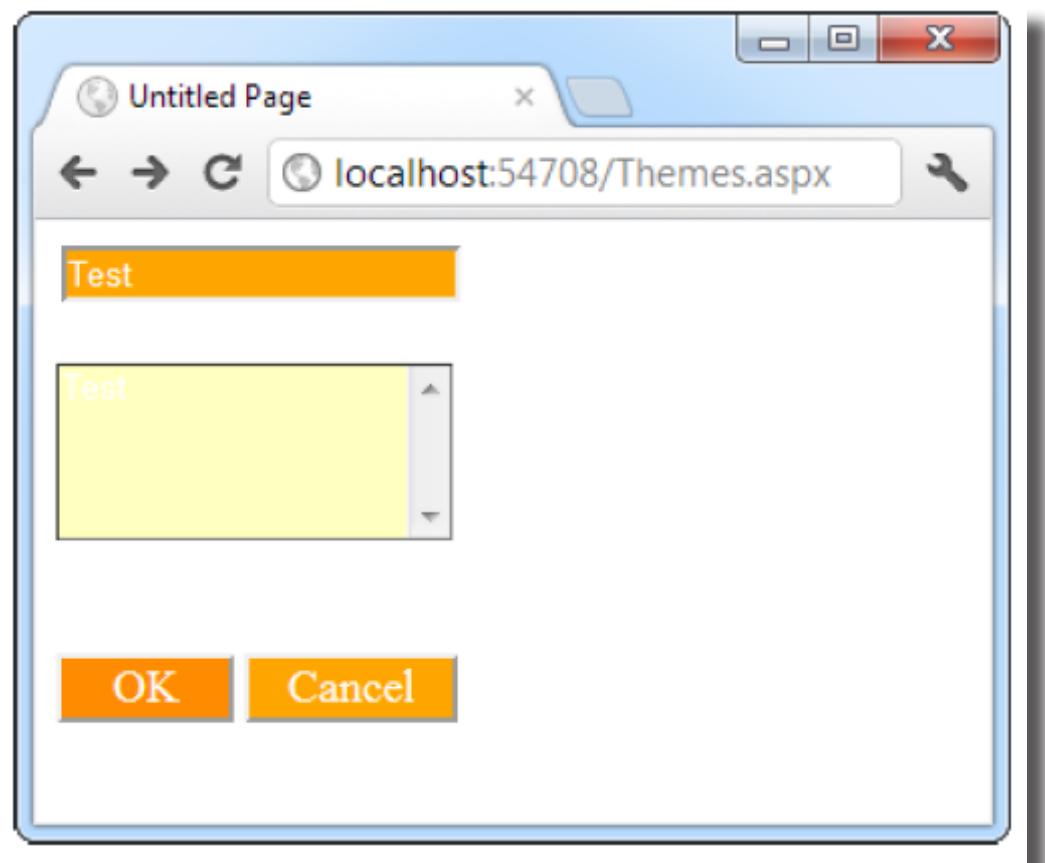


## هندل کردن تداخل های Theme

همانطور که می‌بینید، زمانی که تداخل بین ویژگی‌های کنترل‌ها و theme وجود دارد، برنده theme می‌باشد. برای اینکه این موضوع را بر عکس کنیم باید به جای خصیصه Theme در راهنمای صفحه از StyleSheetTheme استفاده کنیم.

```
<%@ Page Language = "#c" AutoEventWireup = "true" ... StyleSheetTheme = "FunkyTheme" %>
```

اکنون پس زمینه زرد رنگ کنترل listbox روی background color مشخص شده توسط theme قرار می‌گیرد.



 نکته: اگر هم از خصیصه theme و هم از خصیصه StyleSheetTheme استفاده نمایید، ویژگی های تعیین شده در theme بكارگیری شده و سایر ویژگی ها از کنترل خوانده می شود.

گزینه دیگر، پیکربندی کنترل های خاص می باشد. برای این منظور ویژگی EnableTheming را برابر با false قرار دهید.

```
<asp:Button ID = "Button1" runat = "server" ... EnableTheming = "false" />
```

## بكارگیری Theme در کل سایت

با استفاده از راهنمای صفحه، می توانید theme را به یک صفحه bind کنید. اگر بخواهید آن Theme را در کل صفحه استفاده کنید باید از فایل web.config استفاده نمایید:

```
<configuration>
<system.web>
<pages theme = "FunkyTheme">
```

```

...
</pages>
</system.web>
</configuration>
```

در بالا نیز می‌توانید به جای Theme از StyleSheetTheme استفاده نمایید.

```

<configuration>
<system.web>
<pages styleSheetTheme = "FunkyTheme">
...
</pages>
</system.web>
</configuration>
```

## ایجاد چندین Skin برای یک کنترل

ممکن است چندین نوع textbox با توجه به محلی که استفاده خواهد شد یا نوع محتوایی که در آنها قرار می‌گیرد داشته باشد. ASP.NET به شما اجازه ایجاد چندین تعریف برای یک کنترل را می‌دهد.

در حالت معمول اگر چند theme برای یک کنترل ایجاد کنید، ASP.NET به شما خطای می‌دهد که تنها یک skin پیش فرض برای هر کنترل باید وجود داشته باشد. برای رفع این مشکل باید یک skin دارای خصیصه SkinID ایجاد کنید:

```

<asp:listbox runat="server" forecolor="White" backcolor="Orange" />
<asp:textbox runat="server" forecolor="White" backcolor="Orange" />
<asp:button runat="server" forecolor="White" backcolor="Orange" />
<asp:textbox runat="server" forecolor="White" backcolor="DarkOrange" font-bold="True"
skinid="Dramatic" />
<asp:button runat="server" forecolor="White" backcolor="DarkOrange" font-bold="True"
skinid="Dramatic" />
```

برای استفاده از Skin باید مقدار SkinID کنترل درون صفحه وب را مقداردهی کنید.

```
<asp:Button ID = "Button1" runat = "server" ... SkinID = "Dramatic" />
```



نکته: نیازی نیست SkinID، نام واحدی داشته باشد. بلکه فقط باید برای هر کنترل unique باشد.

## های پیشرفته تر Skin

شما می‌توانید جزئیات بیشتری را از تگ‌های کنترل درون فایل theming Skin ایجاد کنید. بیشتر ویژگی‌های کنترل‌ها را پشتیبانی می‌کنند. اگر یک ویژگی نتواند در یک theme تعریف شود، build error دریافت می‌کنید.

مثال ۱:

می‌توانید یک کنترل calender را به همراه تعریف font,color و استایل‌های آن در skin تعریف کنید:

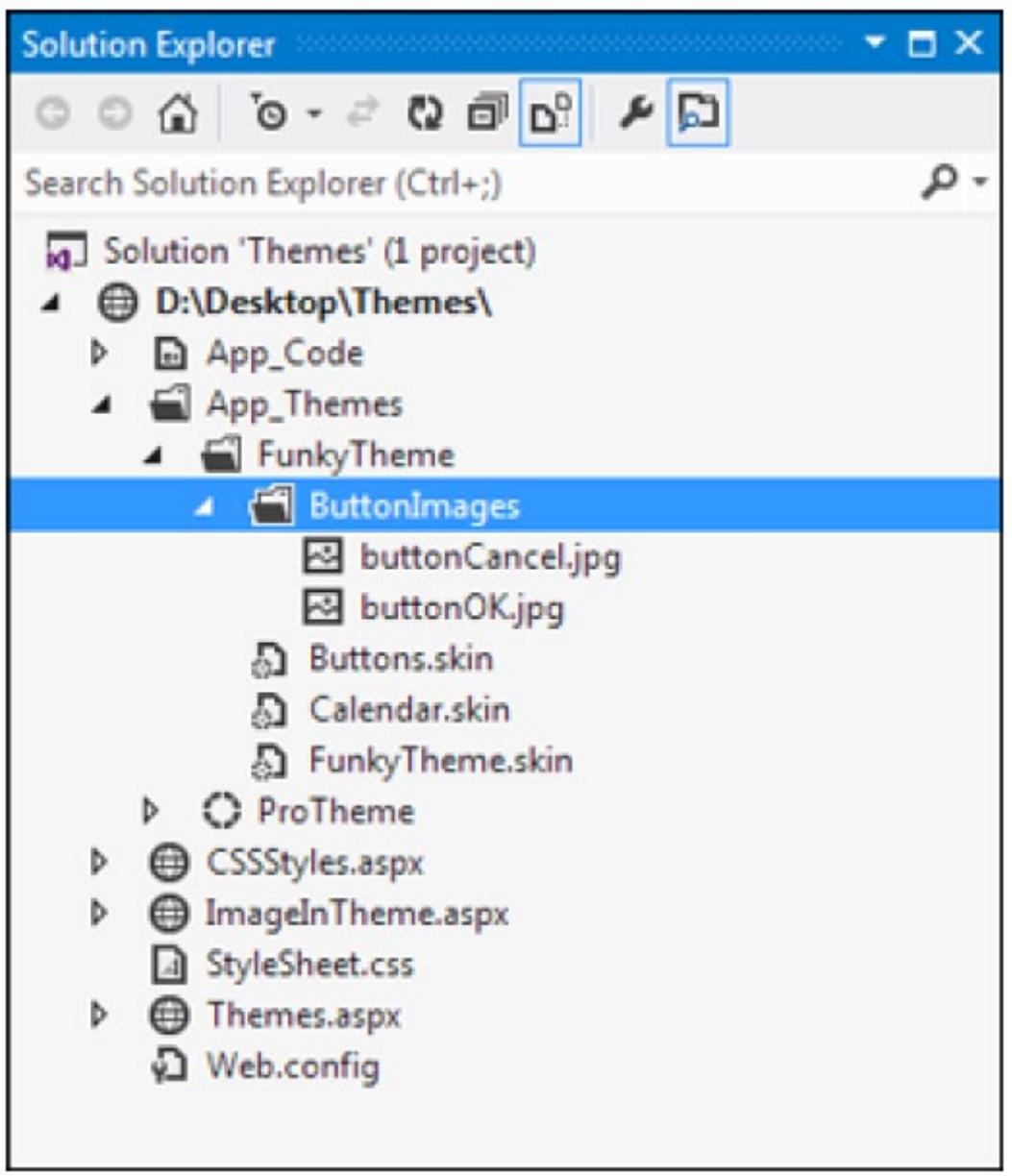
```
<asp:calendar runat="server" backcolor="White" forecolor="Black" bordercolor="Black"
borderstyle="Solid" cellspacing="1" font-names="Verdana" font-size="9 pt" height="250px"
width="500px" nextprevformat="ShortMonth" selectionmode="Day">
<SelectedDayStyle BackColor = "DarkOrange" ForeColor = "White" />
<DayStyle BackColor = "Orange" Font-Bold = "True" ForeColor = "White" />
<NextPrevStyle Font-Bold = "True" Font-Size = "8 pt" ForeColor = "White" />
<DayHeaderStyle Font-Bold = "True" Font-Size = "8 pt" ForeColor = "#333333"
Height = "8 pt" />
<TitleStyle BackColor = "Firebrick" BorderStyle = "None" Font-Bold = "True"
Font-Size = "12 pt" ForeColor = "White" Height = "12 pt" />
<OtherMonthDayStyle BackColor = "NavajoWhite" Font-Bold = "False"
ForeColor = "DarkGray" />
</asp:calendar>
```

و به سادگی آن را در صفحه استفاده نمایید.

```
<asp:Calendar ID = "Calendar1" runat = "server" />
```

**مثال ۲ :** 

ابتدا دو تصویر یکی برای دکمه cancel و دیگری را برای دکمه Ok در پوشه های زیر قرار دهید :



اکنون، شما باید skin هایی که از این تصاویر استفاده می کنند را ایجاد کنید:

```
<asp:imagebutton runat="server" skinid="OKButton" imageurl="ButtonImages/buttonOK.jpg" />
<asp:imagebutton runat="server" skinid="CancelButton" imageurl="ButtonImages/buttonCancel.jpg" />
```

در صفحه وب می‌توانید از کدهای زیر استفاده نمایید:

```
<asp:ImageButton ID = "ImageButton1" runat = "server" SkinID = "OKButton" />
<asp:ImageButton ID = "ImageButton2" runat = "server" SkinID = "CancelButton" />
```

## Master Page

سایت‌ها و برنامه‌های خوب توانند در تمامی صفحات خود یک واسطه کاربری دنباله دار با یک header و footer ثابت و مجموعه‌ای از لینک‌هایی در سمت چپ یا راست داشته باشند.

برای نمونه اگر بخواهید یک navigation bar در بالای هر صفحه‌ای داشته باشید، نه تنها نیاز دارید که HTML مربوط به واسطه کاربری خود را در هر صفحه کپی کنید، بلکه باید مطمئن شوید که آنها در نهایت در یک محل قرار گیرند. اگر بخواهید تغییری در navigation bar دهید باید در همه آنها آن را اعمال کنید.

بنابراین چگونه می‌توانید با یکپارچگی صفحات مختلف که باید به یک شکل مشاهده شوند و عمل کنند سروکار داشته باشید. یک گزینه، استفاده از تقسیم کردن صفحه به فریم‌هایی می‌باشد. Fframe ها، یک امکان HTML ای می‌باشد که به مرورگر اجازه نمایش بیش از یک صفحه را در کنار یکدیگر می‌دهند. متأسفانه، فریم‌ها مشکلات خودشان را دارند. برای نمونه هر فریم یک سند مجزا حساب شده و بصورت جداگانه توسط مرورگر فراخوانی می‌شود. این امر ایجاد کدی که بتواند بین فریم‌ها ارتباط برقرار کند را سخت می‌کند. گزینه بهتر برای این کار، استفاده از ASP.NET MasterPage می‌باشد که به شما امکان تعریف page template و استفاده مجدد آن در کل وب سایت را می‌دهد.

 نکته: فریم‌ها گزینه‌های صفحه‌آرایی و layout شما را محدود می‌کنند. زیرا هر فریم بصورت مجزا بخش ثابتی از صفحه را اشغال می‌کند. زمانی که یک فریم را Scroll می‌کنید، سایر فریم‌ها در محل خود ثابت می‌مانند. برای ایجاد فریم‌هایی که به خوبی کار کنند، باید اطلاعاتی درباره سایز صفحه و ... داشته باشید. فریم‌ها توسط سایت‌های بزرگ استفاده نمی‌شوند.

Master page، شبیه صفحات ASP.NET می‌باشد که می‌توانند شامل HTML، وب کنترل‌ها و کد باشند. دارای پسوند متفاوتی هستند (.aspx) به جای master. وب صورت مستقیم توسط مرورگرها قابل نمایش نمی‌باشند. آنها باید توسط سایر صفحات استفاده شوند که content page نامیده می‌شوند.

## یک Content Page و Master page ساده

برای ایجاد Master page در VS، از منو WebSite>Add New Item را انتخاب و در کادر باز شده، Master page را انتخاب نمایید و یک نام برای آن انتخاب کنید.

این صفحه ایجاد شده دارای یک کنترل ContentPlaceHolder می‌باشد. هر چیزی به غیر از محتوای درون این بخش، غیر قابل تغییر توسط content page می‌باشد. اگر یک content page در هر header درون آن اضافه کنید، این content page استفاده می‌شود.

SiteTemplate.master

Client Objects & Events (No Events)

```

</asp:ContentPlaceHolder>
</head>
<body>
<form id="form1" runat="server">
<div>
    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

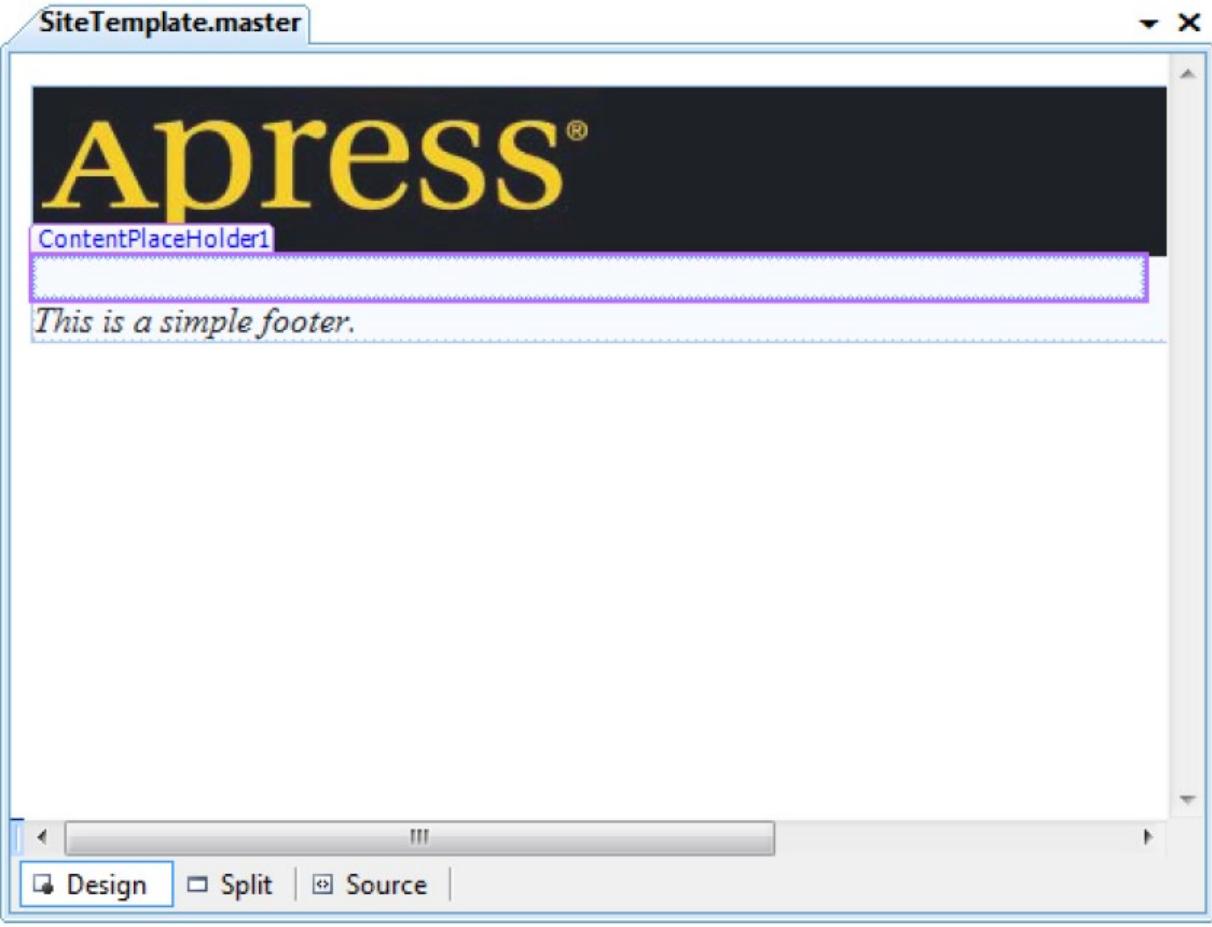
        </asp:ContentPlaceHolder>
    </div>
</form>
</body>

```

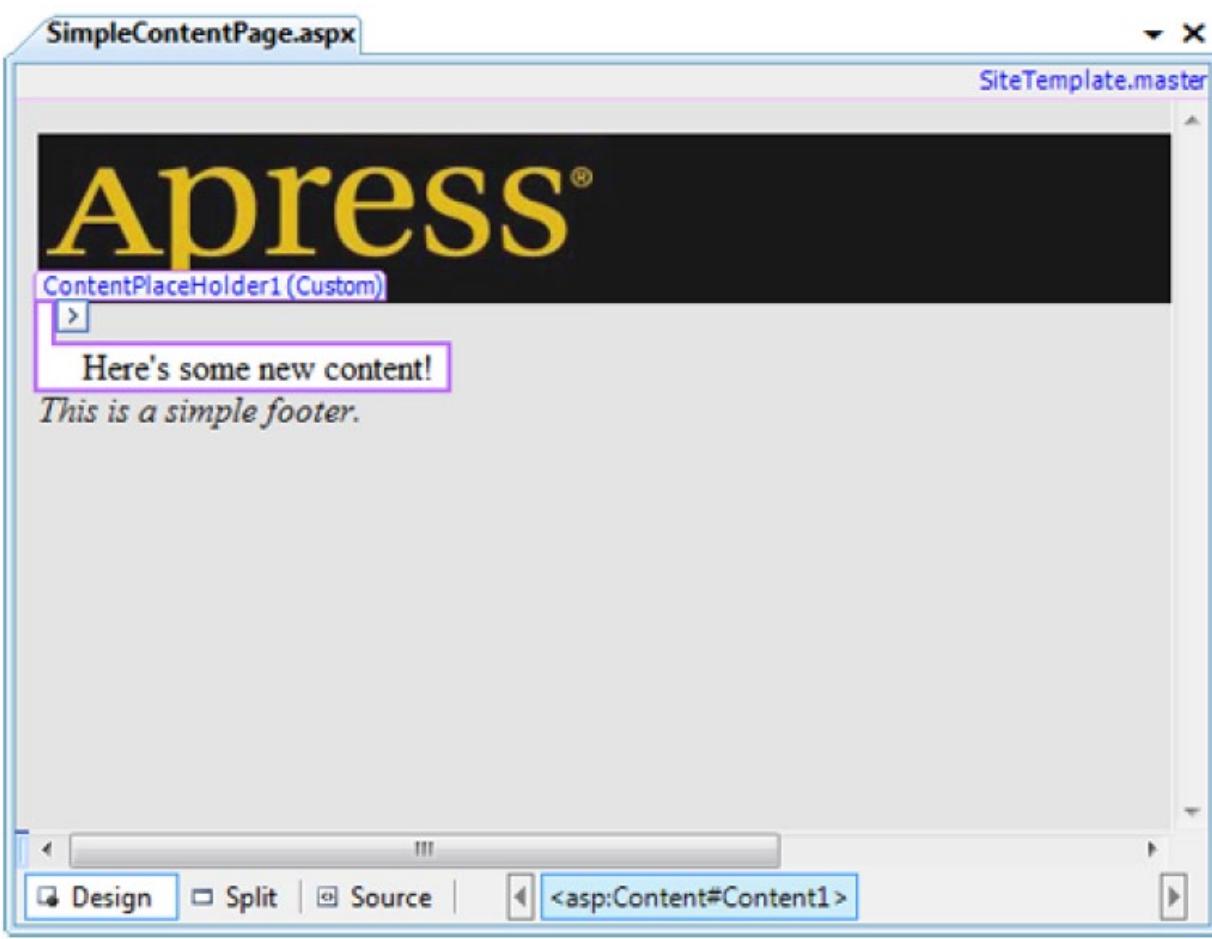
ContentPlaceholder1

Design Split Source | <body> <form#form1> <div> <asp:ContentPlaceHolder#C...>

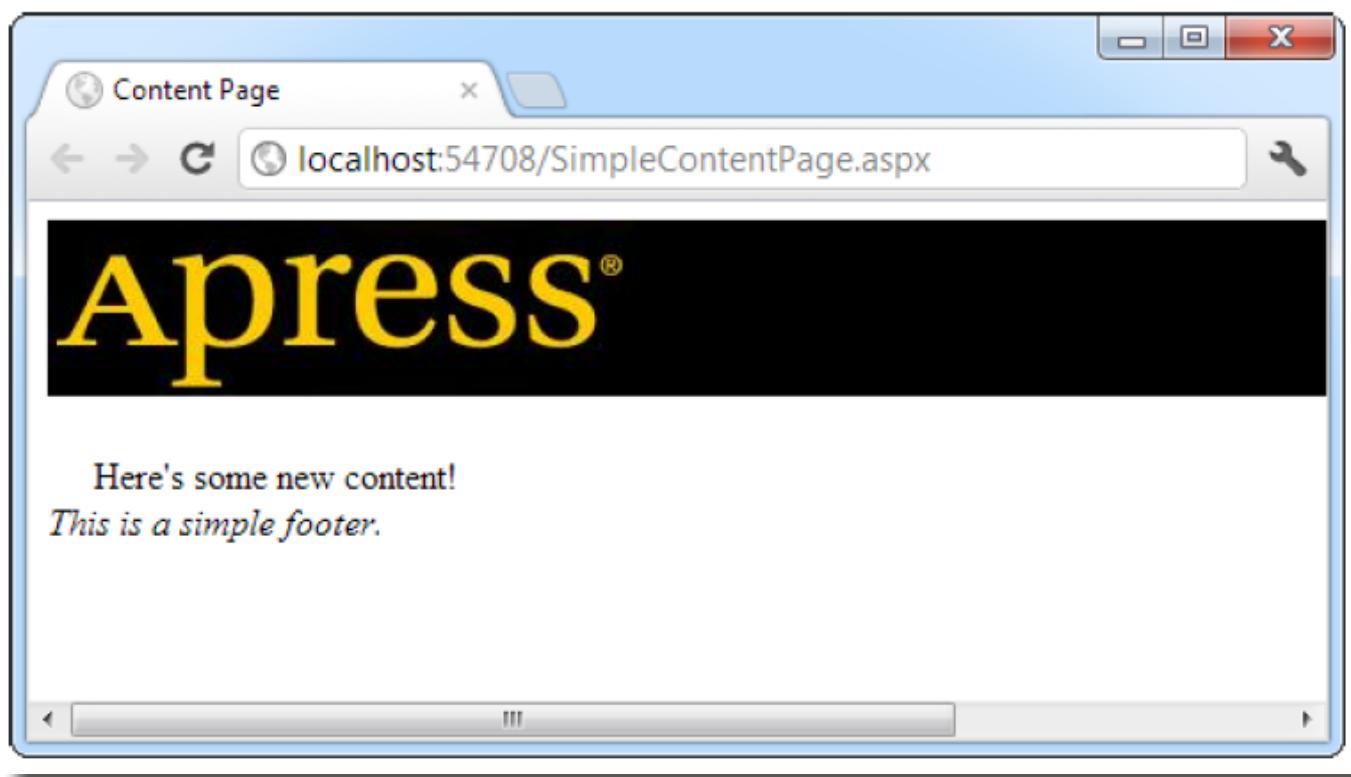
ابتدا بخش `<head>` را تعریف می‌کنیم که شامل metadata هایی مانند search keyword و لینک فایل‌های style sheet می‌باشد. سپس باید بخش `<body>` را تعریف کرد و محتوای صفحه را درون آن قرار داد. در نمونه زیر می‌خواهیم یک header قبل از `ContentPlaceHolder1` قرار دهیم.



اکنون برای ایجاد یک content page از منو WebSite> Add New Item و سپس از پنجره باز شده آیتم WebForm را انتخاب و سپس از پنجره باز شده آیتم Select Master Page را تیک بزنید و روی Add کلیک کنید.



بخش ContentPlaceHolder، برای اندازه شدن با محتوای درون خود، کوچک و بزرگ می‌شود.



چگونه ContentPage و MasterPage به هم متصل می‌شوند

تفاوت اصلی صفحات وب فرم با Master Page این است که آنها با راهنمای صفحه Page شروع می‌شوند و با راهنمای MasterPage شروع می‌شود.

```
<%@ Master Language = "#c" AutoEventWireup = "true" CodeFile = "SiteTemplate.master.cs"
Inherits = "SiteTemplate" %>
```

در زیر نمونه‌ای از یک Master Page ساده را مشاهده می‌کنید :

```
<%@ Master Language="#c" AutoEventWireup="true" CodeFile="SiteTemplate.master.cs"
Inherits="SiteTemplate" %>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
```

```

<form id="form1" runat="server">


```

زمانی که یک ContentPage ایجاد می‌کنید، صفحه شما را با استفاده از اضافه کردن خصیصه هایی به راهنمای صفحه، به Masterpage متصل می‌کند.

```

<%@ Page Language="#c" MasterPageFile("~/SiteTemplate.master" AutoEventWireup="true"
CodeFile="SimpleContentPage.aspx.cs" Inherits="SimpleContentPage" Title="Untitled Page" %>

```

توجه داشته باشید که خصیصه MasterPageFile، برای تعیین پوشه ریشه وب سایت، با مسیر ~/، شروع می‌شود. این مسیر را با (/) که ریشه وب سرور را نشان می‌دهد اشتباه نگیرید، زیرا ریشه وب سرور ممکن است محل برنامه شما باشد یا نباشد. استفاده از مسیر ریشه نسبی (root relative path)، روش بهتری است زیرا تعیین می‌کند ASP.NET (فارغ از محلی که برنامه را deploy کردید) کجا به دنبال Master Page شما بگردد.

در حقیقت مسیر (/) با توجه به صفحه ای که در آن حضور دارد آدرس دهی می‌کند و (~/) از ریشه برنامه شروع می‌کند.

 نکته: کاراکترهای ~/، مسیری است که همیشه با پوشه ریشه یک برنامه تحت وب شروع می‌شود. این سینتکس توسط کنترل‌های سروری و ASP.NET قابل درک می‌باشد و نمی‌توانید آنها را در کدهای HTML استفاده کنید مگر آنکه خصیصه "runat="server" را به آنها اضافه نمایید.

در زیر نمونه ای از یک content page ساده را مشاهده خواهید کرد: 

```

<%@ Page Language="#c" MasterPageFile "~/SiteTemplate.master" AutoEventWireup="true"
CodeFile="SimpleContentPage.aspx.cs" Inherits="SimpleContentPage" Title="Content Page" %>
<asp:content id="Content1" contentplaceholderid="ContentPlaceHolder1" runat="Server">

```

```
<br />
```

Here's some new content!

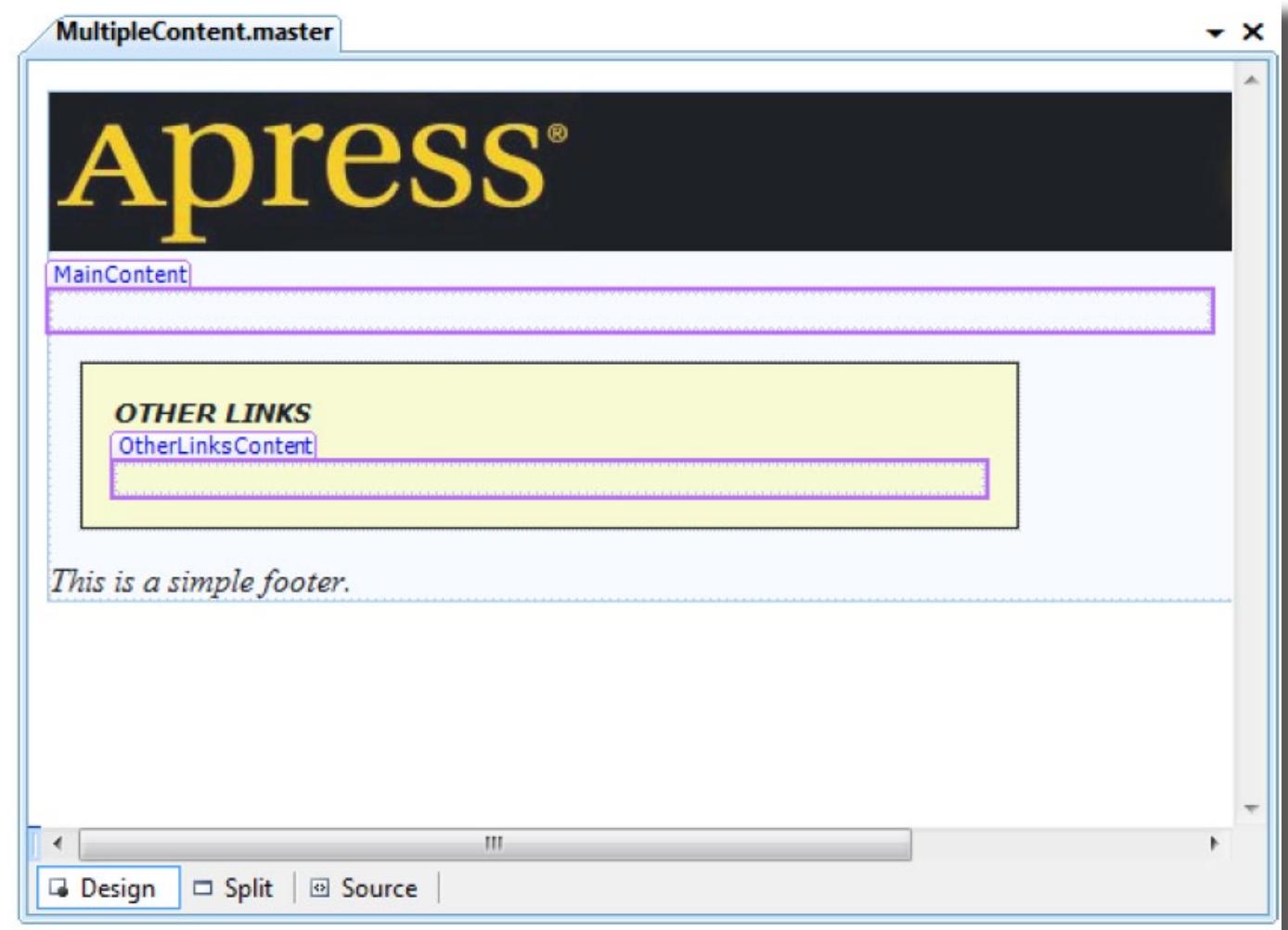
```
<br />
```

```
</asp:content>
```

## یک MasterPage با چند ناحیه محتوایی

Master page ContentPlaceHolder را اضافه محدود نمی‌شوند. بنابراین می‌توانید چندین `ContentPlaceHolder` را در نمونه زیر، یک CPH وجود دارد که کاربر می‌تواند محتوایی را در آن وارد کند و سپس در زیر آن یک `<div>` وجود دارد که دارای یک سری لینک و یک CPH دیگر می‌باشد.

صفحه باید به دو بخش تقسیم شود.



کد master page در زیر آمده است :

```
<%@ Master Language="#c" AutoEventWireup="true" CodeFile="MultipleContent.master.cs"
Inherits="MultipleContent" %>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">

<br />
<asp:contentplaceholder id="MainContent" runat="server">
</asp:contentplaceholder>
<i>
<div style="...">
<b>OTHER LINKS</b>
<br />
<asp:contentplaceholder id="OtherLinksContent" runat="server">
</asp:contentplaceholder>
</div>
This is a simple footer. </i>
</form>
</body>
</html>
```

زمانی که یک content control page بر اساس masterpage ایجاد می‌کنید، VS برنامه را با یک CPH برای هر درون masterpage آغاز می‌کند.

```
<%@ Page Language="#c" MasterPageFile("~/MultipleContent.master" AutoEventWireup="true"
CodeFile="MultipleContentPage.aspx.cs" Inherits="MultipleContentPage" Title="Content Page"
%>
<asp:content id="Content1" contentplaceholderid="MainContent" runat="Server">
```

This is the generic content for this page. Here you might provide some site

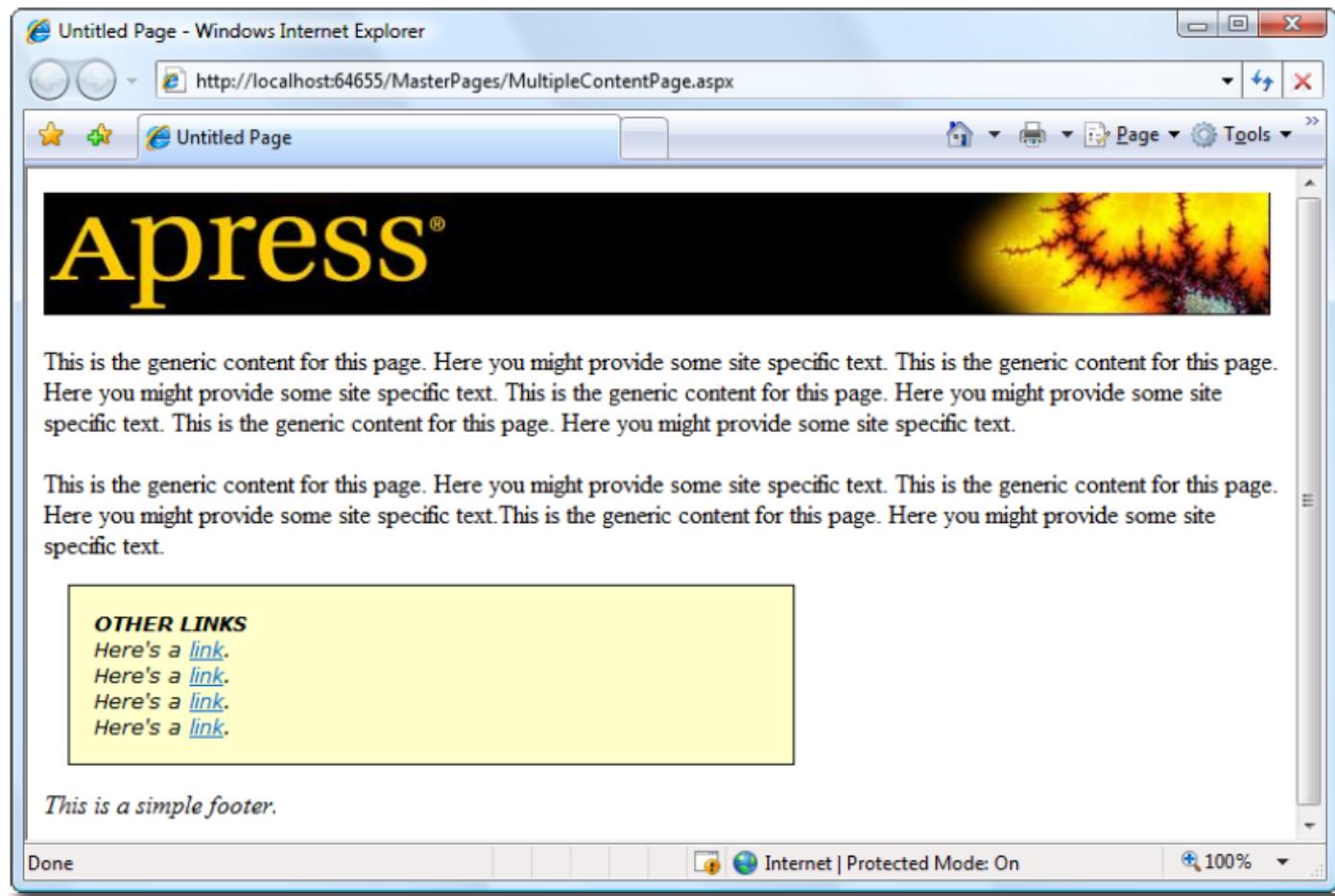
specific text ... </asp:content>

```
<asp:content id="Content2" contentplaceholderid="OtherLinksContent" runat="Server">
```

Here's a <a href = "http://www.prosetech.com">link</a>.<br />

...

```
</asp:content>
```



## محتوای پیش فرض

در صورتیکه بخواهید محتوای درون content page گاهی نمایش داده شود و گاهی نمایش داده نشود می‌توانید از محتوای پیش فرض مانند زیر استفاده کنید. در صورتیکه تگ <content> را از content page حذف کنید، متن پیش فرض درون master page نمایش داده می‌شود.

 نکته: نیازی به حذف دستی تگ Content نیست. می‌توانید از Visual Studio Smart Tag استفاده کنید. در view روی content کلیک کنید. سپس جهت نمای ظاهر شده در بالا-سمت راست را انتخاب و Default to Master Content (برای حذف content و استفاده از محتوای پیش فرض) یا Create Custom Content (برای ایجاد تگ content) را انتخاب نمایید.

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>

<body>
    <form id="form1" runat="server">
        
        <br />
        <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
            This is default content. <br />
        </asp:contentplaceholder>
        <i>This is a simple footer.</i>
    </form>
</body>
</html>
```

## و مسیر نسبی Master Page

اگر masterpage را درون پوشه ای متفاوت از پوشه contentpage ای که از آن استفاده می کند قرار دهید، تگ هایی مانند که از آن استفاده می کند قرار دهید، تگ هایی مانند را درون پوشه ای که به منابع دیگر اشاره دارند، با مشکل رو برو می شوند.

برای نمونه، فرض کنید، یک masterpage را درون زیر پوشه ای با نام MasterPages قرار داده اید و تگ زیر را در آن نوشه اید :

```
<img src = "banner.jpg" />
```

اگر فایل در مسیر \MasterPages\banner.jpg باشد، این کد به خوبی کار می کند. ولی اگر contentpage را درون زیر پوشه دیگری قرار دهید، مسیر تصویر، بصورت نسبی به آن پوشه تفسیر می شود. اگر فایل در آنجا نباشد، تصویر نمایش داده نمی شود.

زمانی که از تگ  یا  برای ارتباط با فایل های style sheet نیز استفاده می کنید، همین مشکل وجود دارد.

برای حل این مشکل، می توانید URL خود را بصورت نسبی به contentpage بنویسید. البته این روش کمی گیج کننده می باشد. روش بهتر این است که خصیصه runat="server" را درون اضافه کنید.

```
<img id="Img1" src = "banner.jpg" runat = "server"/>
```

این شی پس از مقداردهی شی Page برای masterpage ایجاد می‌شود، بنابراین تمامی مسیرها را بصورت نسبی با محل masterpage، تفسیر و ترجمه می‌کند.

برای آدرس دهی نسبی ریشه می‌توانید از روش زیر استفاده کنید :

```
<img id="Img1" src = "~/Images/banner.jpg" runat = "server"/>
```

## پیشرفته Master Page

در ادامه خواهید دید که استایل های CSS چگونه به شما در سازماندهی layout کمک خواهد کرد و چگونه content page ها می‌توانند با کلاس master page در کد تعامل داشته باشند.

## Style Based Layout

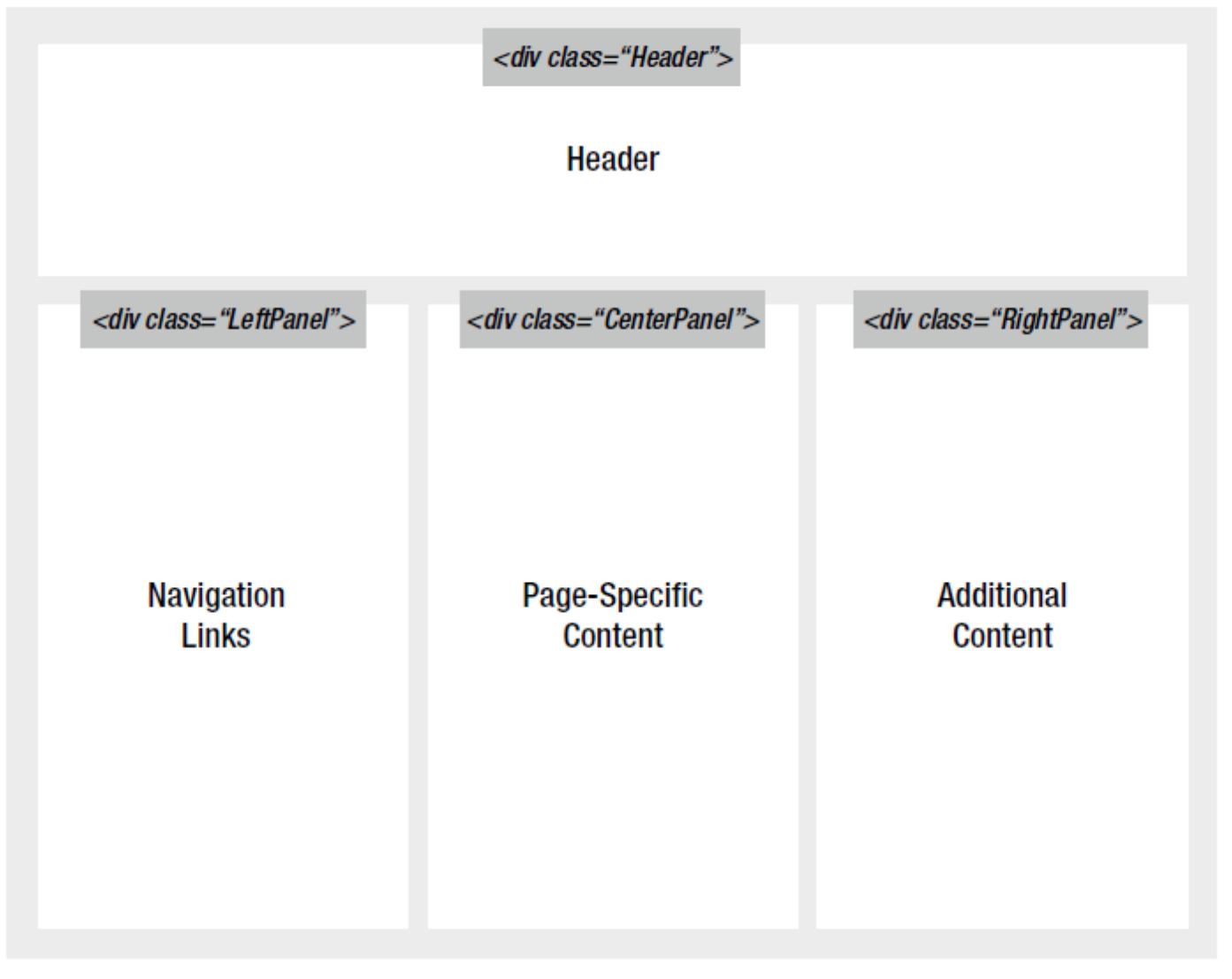
به منظور کنترل Layout صفحه وب دو گزینه برای کمک به شما وجود دارد:

**HTML tables** : با استفاده از جداول HTML ای، می‌توانید بخشی از صفحه خود را به ستون ها و ردیف هایی تقسیم کنید. سپس می‌توانید یک CPH درون یک سلول از جدول اضافه کنید.

**CSS Positioning** : با استفاده از استایل های CSS، خواهید توانست محتوای خود را درون ستون ها یا ناحیه هایی که هر یک از آنها درون یک المان <div> نگهداری می‌شوند، تقسیم کنید. سپس می‌توانید درون هر <div> یک CPH قرار دهید.

اگرچه هردو روش شبیه به هم هستند اما، استانداردتر و جدیدتر می‌باشد. در نمونه زیر ابتدا محتوای خود را به چند بخش تقسیم می‌کنید و هر بخش بصورت جداگانه درون یک ستون یا کادر قرار داده می‌شود. در markup خود می‌توانید هر بخش را درون یک المان <div> قرار دهید. در نهایت برای هر <div> می‌توانید از استایل های جداگانه استفاده نمایید و می‌توانید از ویژگی های CSS برای تعیین مکان آنها استفاده کنید.

به عنوان یک نمونه خوب، فرض کنید یک برنامه تحت وب با یک header و side panel دارد.



مطلوب مهم برای ایجاد این ترتیب از layout چند ستونی، استفاده از تعیین مکان absolute برای قرار دادن ستون ها می باشد. قدم اول برای پیاده سازی این طراحی، ایجاد ۴ استایل با نامهای Header,LeftPanel,RightPanel و CenterPanel (همه در یک فایل style sheet مجزا قرار دارند) می باشد.

.Header

```
{
  position: absolute;
  top: 10px;
  left: 10px;
  height: 60px;
  text-align: center;
}
```

کد بالا یک کادر با تعیین مکان absolute که نزدیک بالای صفحه قرار می‌گیرد، ایجاد می‌کند.

#### .LeftPanel

```
{
  position: absolute;
  top: 70px;
  left: 10px;
  width: 160px;
}
```

یک کادر با پهنای ۱۶۰ پیکسل ایجاد می‌کند. این ستون، ۷۰ پیکسل از بالای صفحه و ۱۰ پیکسل از سمت چپ فاصله دارد.

#### .RightPanel

```
{
  position: absolute;
  top: 70px;
  right: 10px;
  width: 160px;
}
```

مرحله نهایی، ایجاد ناحیه Content می‌باشد. این استایل نمی‌تواند از اندازه ثابت استفاده کند، زیرا عرض ثابت بستگی به این دارد که چه مقدار فضا در دو طرف دو پنل که در اطراف اضافه شده اند وجود دارد. این موضوع به نوبه خود بستگی به عرض پنجره مرورگر دارد. خوشبختانه، یک راه ساده وجود دارد. شما می‌توانید به سادگی مقدار margin های (حاشیه های) چپ و راست را بزرگ تر کنید تا با ستون های سمت راست و چپ تناسب پیدا کند و محتوا شما بصورت منظمی در فضای موجود قرار گیرد.

برای نمونه، پنل سمت چپ ۱۶۰ پیکسل عرض دارد و در ۱۰ پیکسلی سمت چپ لبه پنجره مرورگر قرار گرفته است که این یعنی شما نیاز به left margin به اندازه ۱۷۰ پیکسل هستید، به علاوه چند پیکسل بیشتر برای پیشگیری از لمس content و border آن در نظر بگیرید.

## .CenterPanel

```
{
    position: absolute;
    top: 70px;
    margin-left: 175px;
    margin-right: 180px;
}
```

اکنون شما از استایل ها درون master page استفاده می کنید:

```
<div class="Header">
</div>

<div class="LeftPanel">
</div>

<div class="CenterPanel">
</div>

<div class="RightPanel">
</div>
```

زمانی که از این تکنیک استفاده می کنید، ترتیب بخش های <div> مهم نیست زیرا هر کدام از آنها در محل دقیقی قرار خواهد گرفت. به دلیل آنکه این استایل ها درون فایل مجازی قرار دارند، می توانی در صفحات مختلفی از آنها استفاده کنید.

مرحله نهایی اضافه کردن CPH و content می باشد. در زیر Master Page ای که این کار را انجام داده است مشاهده خواهید کرد:

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">
    <title></title>
    <link href="LayoutStyles.css" rel="stylesheet" type="text/css" />
</head>

<body>
    <form id="form1" runat="server">
        <div class="Header">
```

```

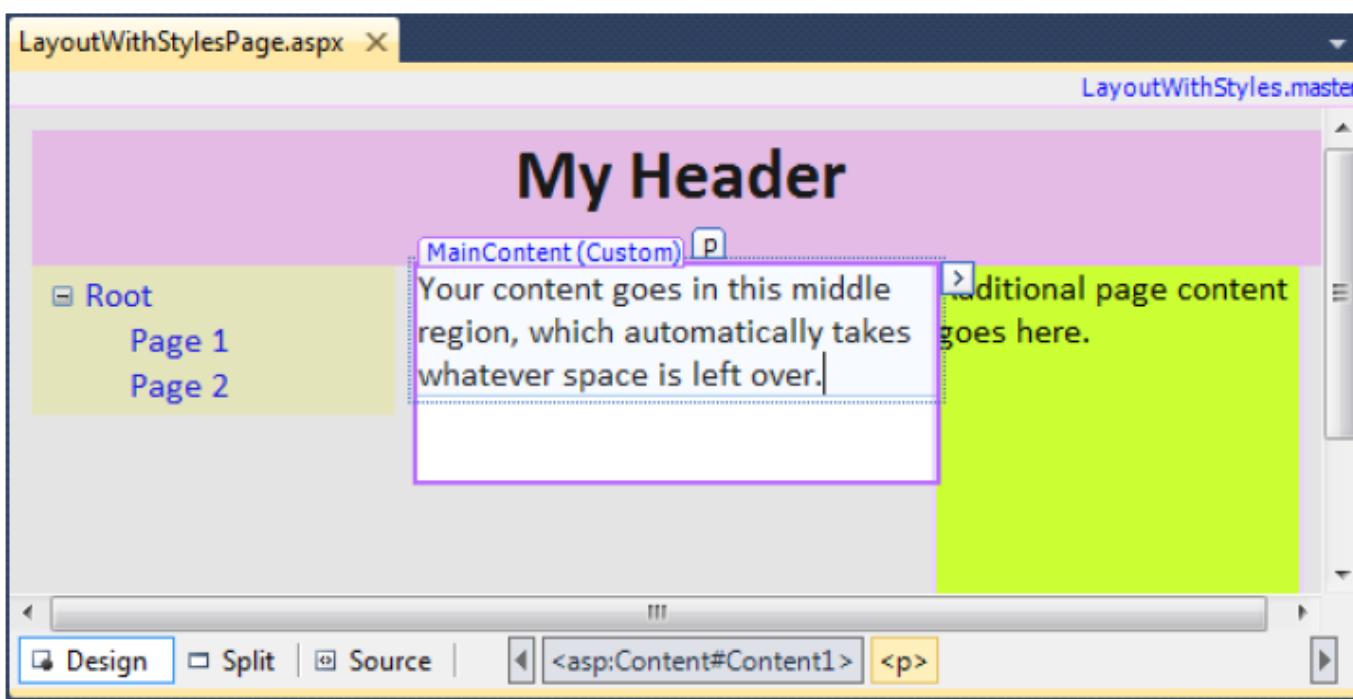
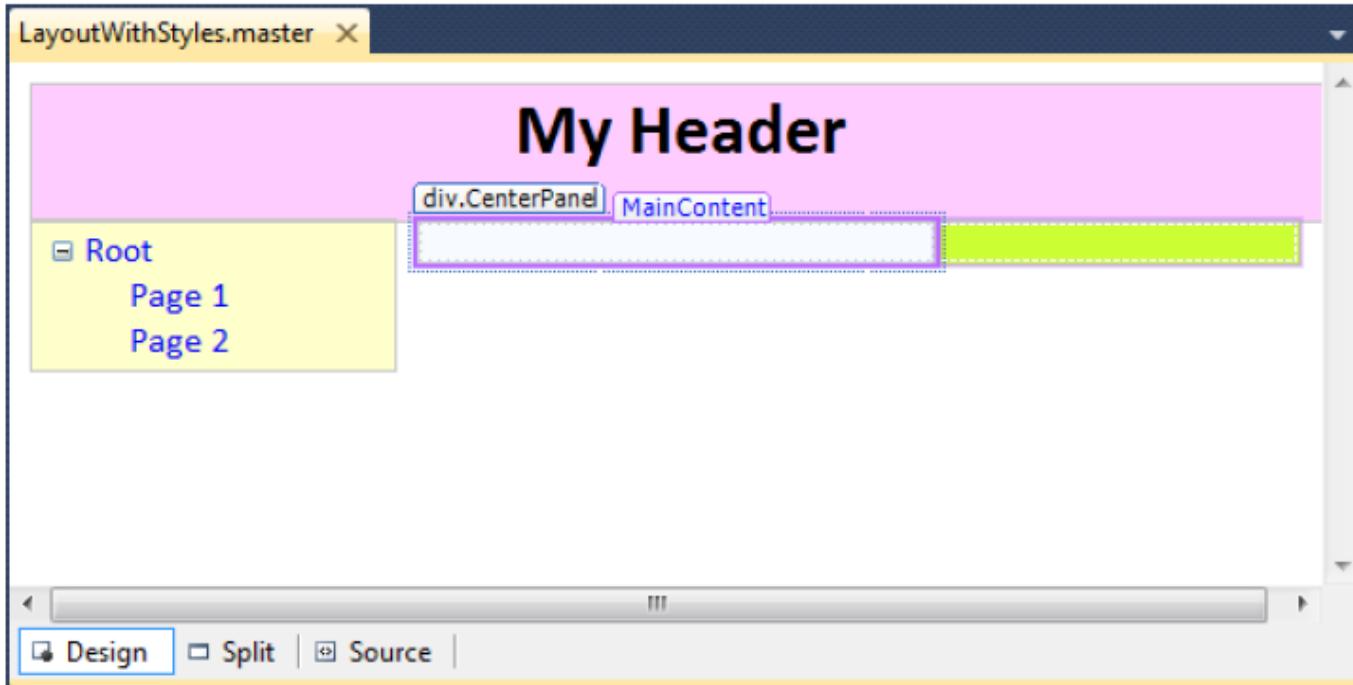
<h1>
    My Header</h1>
</div>

<div class="LeftPanel">
    <asp:treeview id="TreeView1" runat="server" width="150px">
        <Nodes>
            <asp:TreeNode Text = "Root" Value = "New Node">
                <asp:TreeNode Text = "Page 1" Value = "Page 1"> </asp:TreeNode>
                <asp:TreeNode Text = "Page 2" Value = "Page 2"> </asp:TreeNode>
            </asp:TreeNode>
        </Nodes>
    </asp:treeview>
</div>

<div class="CenterPanel">
    <asp:contentplaceholder id="MainContent" runat="server">
</asp:contentplaceholder>
</div>

<div class="RightPanel">
    <asp:contentplaceholder id="AdditionalContent" runat="server">
</asp:contentplaceholder>
</div>
</form>
</body>
</html>

```



**نکته:** برای بدست آوردن چند نمونه از layout چند ستونی ابتدایی، [www.karamoozesh.com/web/layouts](http://www.karamoozesh.com/web/layouts) را مشاهده کنید. برای نمونه های پیچیده تر می توانید سایت [www.csszengarden.com](http://www.csszengarden.com) را مشاهده کنید.



## تودر تو های Master Page

می توانید درون یک Master page از یک Master Page دیگر استفاده کنید. برای نمونه ممکن است شما دو بخش از وب سایت خود داشته باشید. هر بخش می تواند نیاز به کنترل های هدایتی خودش داشته باشد. اما هردو بخش باید دارای یک header باشند. در این مورد، می توانید یک Master Page سطح بالا ایجاد کنید که header را اضافه می کند. سپس می توانید Master Page دوم را اضافه نمایید که از Master Page اول استفاده خواهد کرد. (از طریق خصیصه MasterPageFile). در MP دوم می توانید header را دریافت کنید و کنترل های هدایتی خود را اضافه نمایید. می توانید دو نسخه از MP دوم ایجاد کنید و برای هر بخش از یکی از آن ها استفاده نمایید.

## کد درون یک Master Page

در همه مثال های زده شده در این بخش، layout ثابتی بوده اند. درست مانند صفحات وب، Master Page ها نیز شامل بخش کدی که می تواند به رویدادها پاسخ دهد می باشند. برای نمونه می توانید کلیک کردن کنترل های هدایتی را به منظور فرستادن کاربران به صفحه مورد نظرشان، هندل کنید.

## تعامل با یک Master Page از طریق برنامه نویسی

MP ها می توانند متدهایی را ارائه دهند که آنها را در موقع لزوم استفاده کند یا property هایی را ارائه دهند که content page بتوانند آنها را بر اساس نیازشان مقدار دهی کنند. این کار به content page اجازه تعامل با master page را می دهد.

برای نمونه تصور کنید که می خواهید به کاربر توانایی جمع کردن (collapse) سلوکی که شامل کنترل های هدایتی می شود را برای داشتن فضای بیشتر به منظور مشاهده محتوای صفحه، بدهید. شما نمی خواهید این کار را در Master Page پیاده سازی کنید زیرا می خواهید این امکان فقط در صفحه اصلی در دسترس باشد. content page نیز قطعاً نمی تواند این کار را انجام دهد زیرا باید بخش ثابتی از master page را تغییر دهد. راه حل موجود، ایجاد راهی برای content page می باشد تا با master page در تعامل باشد، بنابراین از MP خواهد خواست تا کنترل های هدایتی را جمع یا مخفی کند.

یک راه برای پیاده سازی این طرح، اضافه نمودن یک property جدید با نام ShowNavigationControls به کلاس Master Page می باشد. این ویژگی، زمانی که با false مقدار دهی شود می تواند بصورت خودکار کنترل های هدایتی را مخفی کند.

```
public bool ShowNavigationControls
{
    get
    {
        return TreeView1.Visible;
    }
    set
}
```

```

    {
        TreeView1.Visible = value;
    }
}
}

```

دقت داشته باشید که این public property می‌باشد بنابراین کلاس‌ها و صفحات دیگر می‌توانند به آن دسترسی داشته باشند.

برای دسترسی به این صفحه، از property با نام Page.Master استفاده می‌کند. برای دسترسی به property مورد نظر از content page استفاده کنید.

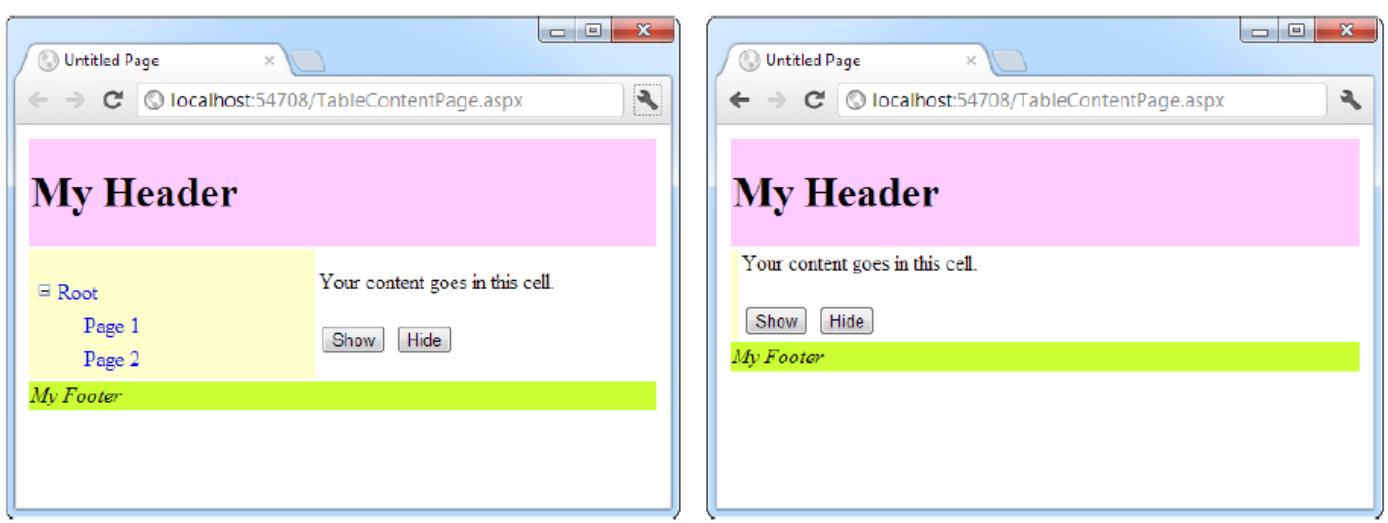
در زیر، کدهای هندل کننده درون content page که کنترل‌های هدایتی را مخفی می‌کنند یا نمایش می‌دهند، آمده است.

```

protected void cmdHide_Click(object sender, EventArgs e)
{
    TableMaster master = (TableMaster)this.Master;
    master.ShowNavigationControls = false;
}

protected void cmdShow_Click(object sender, EventArgs e)
{
    TableMaster master = (TableMaster)this.Master;
    master.ShowNavigationControls = true;
}

```





سفر کردن، فقط تماش کردن و عکس گرفتن نیست.  
شیوه زندگی شما پس از هر سفر تغییر می کند. «میریام برادری»

# پنجم سوچ: کار با داده ها

فصل دهم: اصول ADO.NET

فصل یازدهم: Data Binding

فصل دوازدهم: کنترل مای کروم



## ADO.NET

در ابتدای این کتاب، شما دید که ASP.NET تنها یکی از کامپوننت های موجود در پلت فرم دات نت می باشد. یکی دیگر از امکانات موجود در فریمورک دات نت، مدل دسترسی به داده ADO.NET می باشد.

به زبان ساده، ADO.NET، یک تکنولوژی می باشد که برنامه های دات نت از آن برای تعامل با بانک اطلاعاتی استفاده می کنند. در این فصل شما با ADO.NET و خانواده ای از اشیا که کاربردهای آن را فراهم می کنند آشنا خواهید شد.

 نکته : LINQ to Entities یک مدل سطح بالا می باشد. در رابطه با LINQ در فصل های بعدی توضیح داده خواهد شد.

### بانک اطلاعاتی

در بیشتر برنامه های ASP.NET، شما نیاز به یک بانک اطلاعاتی برای انجام بعضی از امور دارید. در زیر نمونه ای از این موارد را مشاهده خواهید کرد :

- سایت های E-Commerce (مانند Amazon.com)، از بانک های اطلاعاتی برای ذخیره کاتالوگ محصولات استفاده می کنند. همچنین سفارشات، مشتریان، و رکوردهای مربوط به حمل کالا را نیز می توانند ذخیره کنند.
- موتورهای جستجوگر (مانند Google) که از بانک های اطلاعاتی برای نگهداری ایندکس های URL صفحات، لینک ها و کلمات کلیدی استفاده می کنند.
- پایگاه های دانش (مانند Microsoft Support) که از بانک اطلاعاتی برای ذخیره لینک هایی به اسناد منابع گوناگون استفاده می کنند.
- سایت های رسانه ای (مانند New York Times) که از بانک اطلاعاتی برای ذخیره مقالات استفاده می کنند.

 نکته : در صورتی که آشنایی شما با مفاهیم بانک اطلاعاتی، طراحی و ایجاد بانک، انواع سرویس های ۲۰۱۲ و مدیریت بانک اطلاعاتی کم می باشد می توانید با مراجعه با سایت کارآموزش [www.com.karamoozesh.com](http://www.com.karamoozesh.com) در جریان کتاب های جدید و مفید قرار بگیرید.

### پیکربندی بانک اطلاعاتی خودتان

قبل از آنکه بتوانید هیچ کد دسترسی به داده ای را اجرا کنید، باید که database server برای گرفتن دستورات خود داشته باشد. البته گرینه های خوب زیادی وجود دارد که همگی به یک صورت با ADO.NET کار می کنند (و کدهای خاص مشابهی نیاز دارند). یکی از مناسب ترین آنها، Microsoft SQL Server می باشد.

کدهای استفاده شده در این فصل با SQL Server 7 و بعد از آن کار می کند.



نکته : این فصل و فصل های بعدی برای مثال های خود از بانک های Northwind و pubs استفاده می کنند. این بانک ها، در نسخه های جدید SQL Server از قبل نصب نمی شوند. شما می توانید به سادگی آنها را با استفاده از script فراهم شده توسط نمونه های آنلاین نصب کنید.

## استفاده از SQL Server Express

اگر بانک اطلاعات تست ندارید ممکن است بخواهید از SQL Server 2012 Express استفاده کنید. این نسخه یک نسخه با مقیاس کوچک از SQL Server می باشد که رایگان می باشد و محدودیت هایی دارد. برای نمونه تنها می تواند از یک CPU ۴ هسته ای و RAM یک گیگابایتی استفاده کند و نمی تواند حجمی بیش از 10GB داشته باشد. دو نسخه از SQL Server Express وجود دارد. نسخه اول که به تنها ی قابل دانلود و نصب می باشد) می باشد را می توانید با یا بدون ابزار بانک اطلاعاتی از [sql/express/com.microsoft.www](http://sql/express/com.microsoft.www) دانلود کنید. این نسخه ای است که شما می خواهید آن را روی یک وب سرور واقعی نصب کنید (و نسخه کامل SQL Server را ندارید).

نسخه دوم، SQL Server 2012 Express LocalDB نامیده می شود که در همه نسخه های Visual Studio وجود دارد. این نسخه نیز کاربردی مشابه دارد، اما بیشتر مناسب تست می باشد.

## نمایش و تغییر بانک اطلاعاتی در Visual Studio

به عنوان یک برنامه نویش ASP.NET، ممکن است مسئول ایجاد بانک اطلاعاتی مورد نیاز برنامه تحت وب شوید. همچنین ممکن است بانک اطلاعاتی از قبل ایجاد شده باشد. اگر از نسخه کامل SQL Server استفاده کرده باشید، ابزارها گرافیکی مانند Management Studio در اختیار شما می باشد.



نکته : SQL Server Express دارای Management Studio نمی باشد. البته می توانید آن را بصورت جداگانه از [www.microsoft.com/express/database](http://www.microsoft.com/express/database)

دانلود کنید. روی دکمه Sql Server 2012 Express کلیک کنید تا لیستی از بسته های قابل دانلود را نمایش دهد. سپس روی دکمه دانلود در کنار SQL Server Management Studio Express کلیک کنید.

اگر ابزار مناسبی برای مدیریت بانک اطلاعاتی ندارید، می توانید بسیاری از کارها را با استفاده از پنجره Server Explorer موجود در Visual Studio، انجام دهید. می توانید یک زبانه برای SQL Server، در سمت راست VS، بینید. این پنجره را می توانید از View>Server Explorer نیز بیاورید.

با استفاده از نود Data Connection در Server Explorer، می توانید به بانک های اطلاعاتی موجود متصل شوید یا یک بانک جدید ایجاد کنید. فرض کنید بانک pubs را نصب کرده اید. (فایل readme.txt را مطالعه کنید)، می توانید با استفاده از مراحل زیر به آن یک اتصال برقرار کنید :

۱- روی نود Data Connection کلیک راست کرده و Add Connections را انتخاب کنید.

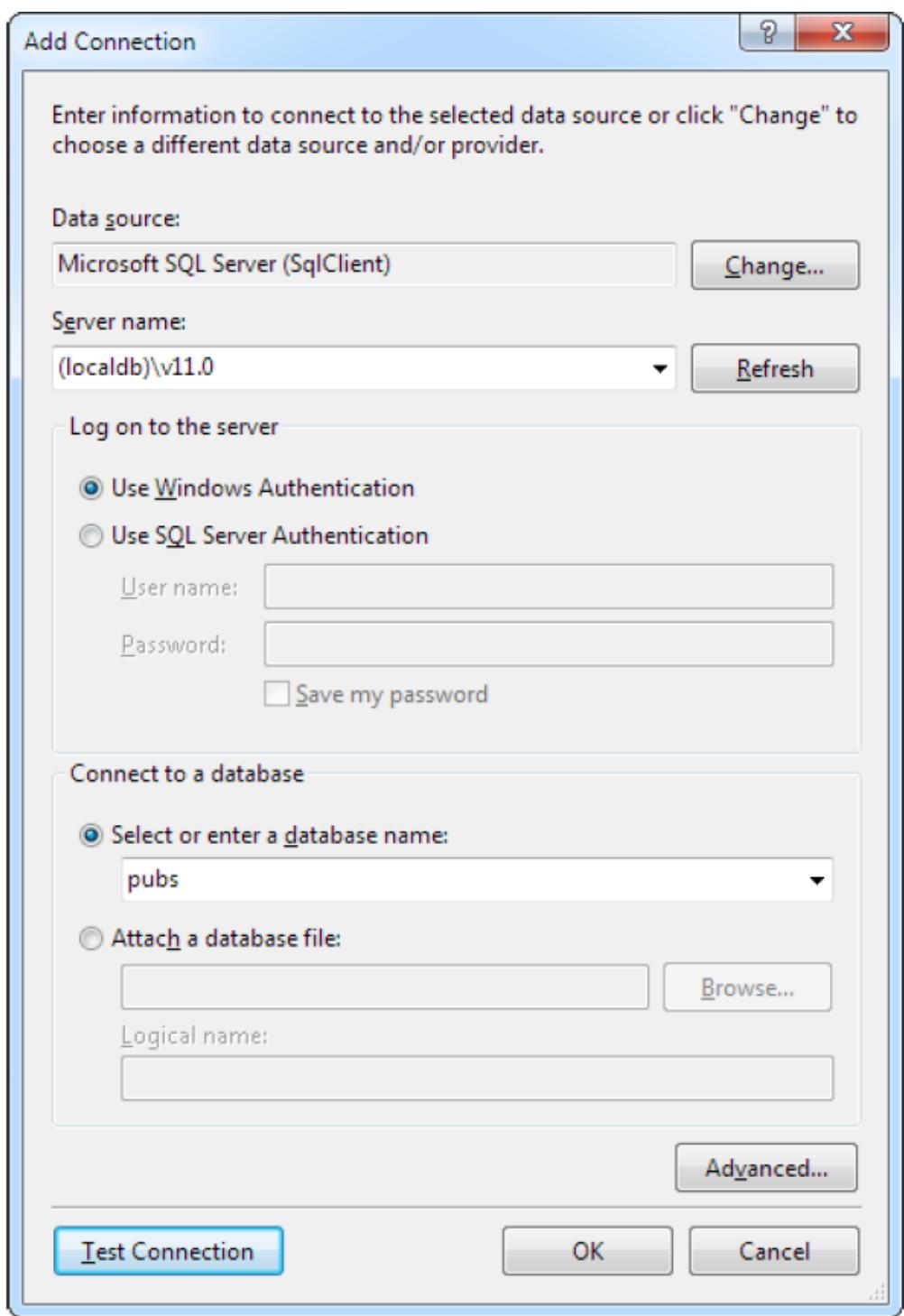
۲- زمانی که پنجره Choose Data Sources ظاهر می‌شود، گزینه Continue کلیک کنید.

۳- اگر از نسخه کامل SQL Server استفاده کرده اید، عبارت server name را به عنوان Localhost وارد کنید.

این کار تعیین می‌کند که database server (default instance) موجود در local computer (پیش فرض) نامه کامپیوتر remote (جایگزین کنید). در صورت نیاز این نام را با نام SQL Server Express اندکی متفاوت می‌باشد. اگر از SQL Server Express استفاده می‌کنید (که درون VS می‌باشد)، باید عبارت localdb\v11.0 را وارد نمایید و اگر نسخه جداگانه SQL Server Express را دانلود کرده اید باید عبارت localhost\SQLEXPRESS را وارد نمایید.

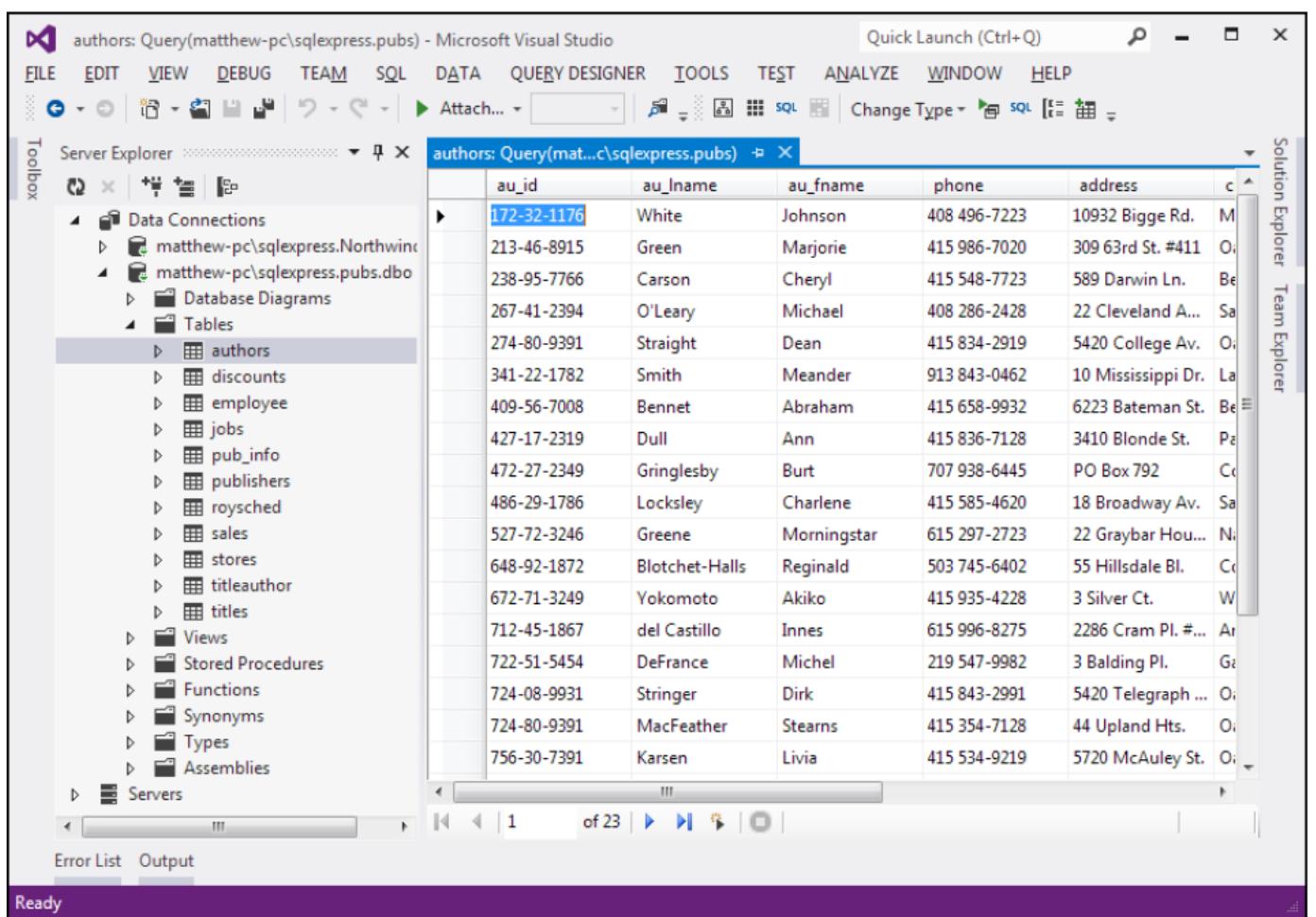
۴- روی دکمه Test Connection کلیک کنید تا محل بانک اطلاعاتی شما تعیین اعتبار شود.

۵- در لیست Select Or Enter a Database Name Pubs را وارد نمایید. (باید بانک Pubs را قبل نصب کرده باشید)



به عنوان جایگزین می‌توانید برای ایجاد یک بانک جدید روی نود DataConnections کلیک راست کرده و Create New SQL Server Database را انتخاب نمایید.

۶- روی OK کلیک کنید. Server Explorer در پنجره database connection نمایان می‌شود.



 نکته: پنجره Server Explorer در صورت استفاده از SQL Server Express بصورت دستی می‌باشد و به شما اجازه قرار دادن مستقیم بانک اطلاعاتی در پوشه App\_Data موجود در برنامه تحت وب را می‌دهد (به جای قرار دادن تمامی بانک‌ها در یک محل جدا). اگر VS، یک بانک را در App\_Data پیدا کنید، بصورت خودکار یک اتصال به آن در Data Connection group اضافه می‌کند.

## آشنایی با Data Provider Model

ADO.NET بر اساس عملکرد مجموعه کوچکی از کلاس‌های مرکزی کار می‌کند. می‌توانید این کلاس‌ها را به دو گروه تقسیم کنید: آنهایی که برای شامل شدن و مدیریت داده استفاده می‌شوند (مانند DataSet, DataTable, DataRow) و آنهایی که برای اتصال به یک منبع داده مشخص استفاده خواهند شد (مانند Command, Connection, DataReader).

های داده ای گوناگون سفارشی سازی شده اند. برای نمونه SQL Server provider، برای کار با SQL Server طراحی شده است. هر provider دارای پیشوند مخصوص خود می‌باشد. بنابراین OracleConnection کلاس‌های OracleProvider و SQLConnection، SQLCommand شامل SQLServerProvider می‌باشد.

کلاس‌های مربوط به SQL Server Data Provider در سه فضای نام وجود دارد که در زیر مشاهده می‌کنید :



فضای نام	هدف
System.Data.SqlClient	شامل کلاس‌هایی که برای اتصال به Microsoft SQL Server و اجرای دستورات استفاده می‌شود، می‌باشد ( SqlCommand و SqlConnection )
System.Data.SqlTypes	امل ساختارهایی برای انواع داده ای خاص SQL Server مانند SQLMoney و SQLDateTime. از این نوع‌ها می‌توانید برای کار با انواع داده ای SQL Server بدون نیاز به تبدیل آنها به نوع‌های داده ای استاندارد استفاده نمایید.
System.Data	شامل کلاس‌هایی مانند DataSet و DataRelation که به شما اجازه تغییرات در داده relational را می‌دهد.

## استفاده از دسترسی داده مستقیم (Direct Data Access)

در این روش یک SQL Command ایجاد و آن را اجرا می‌کنید. زمانی که داده‌ها را با استفاده از این روش درخواست می‌کنید، یک کپی از آنها را در حافظه نگه نمی‌دارید در عوض، در دوره‌های کوتاهی (تا آنجا که اتصال به بانک باز است) از زمان با آنها کار می‌کنید و سپس اتصال را در اولین فرصت ممکن می‌بندید. این روش با disconnected data access متفاوت است که در آن یک کپی از داده را در یک شی DataSet نگه می‌داشتید و پس از بستن اتصال نیز می‌توانستید با آن کار کنید.

روش دسترسی مستقیم، برای صفحات وب ASP.NET مناسب است. به یاد داشته باشید که یک صفحه وب زمانی لود می‌شود که صفحه درخواست شده باشد و زمانی کامل می‌شود که پاسخ به کاربر برگشته باشد؛ و این یعنی یک صفحه عموماً عمری در حدود چند ثانیه خواهد داشت.

 نکته: اگرچه صفحات وب ASP.NET، نیازی به ذخیره داده در حافظه ندارند، ولی راه‌هایی برای بالا بردن کارایی آنها وجود دارد. برای نمونه می‌توانید کاتالوگ محصولات را در بانک ذخیره کنید و داده‌های آن را در حافظه وب سرور نگهداری کنید تا زمانی که یک کاربر آن صفحه را درخواست کرد، از آن استفاده مجدد کنید. به این تکنیک Caching می‌گویند که در فصل‌های بعدی با آن آشنا می‌شوید.

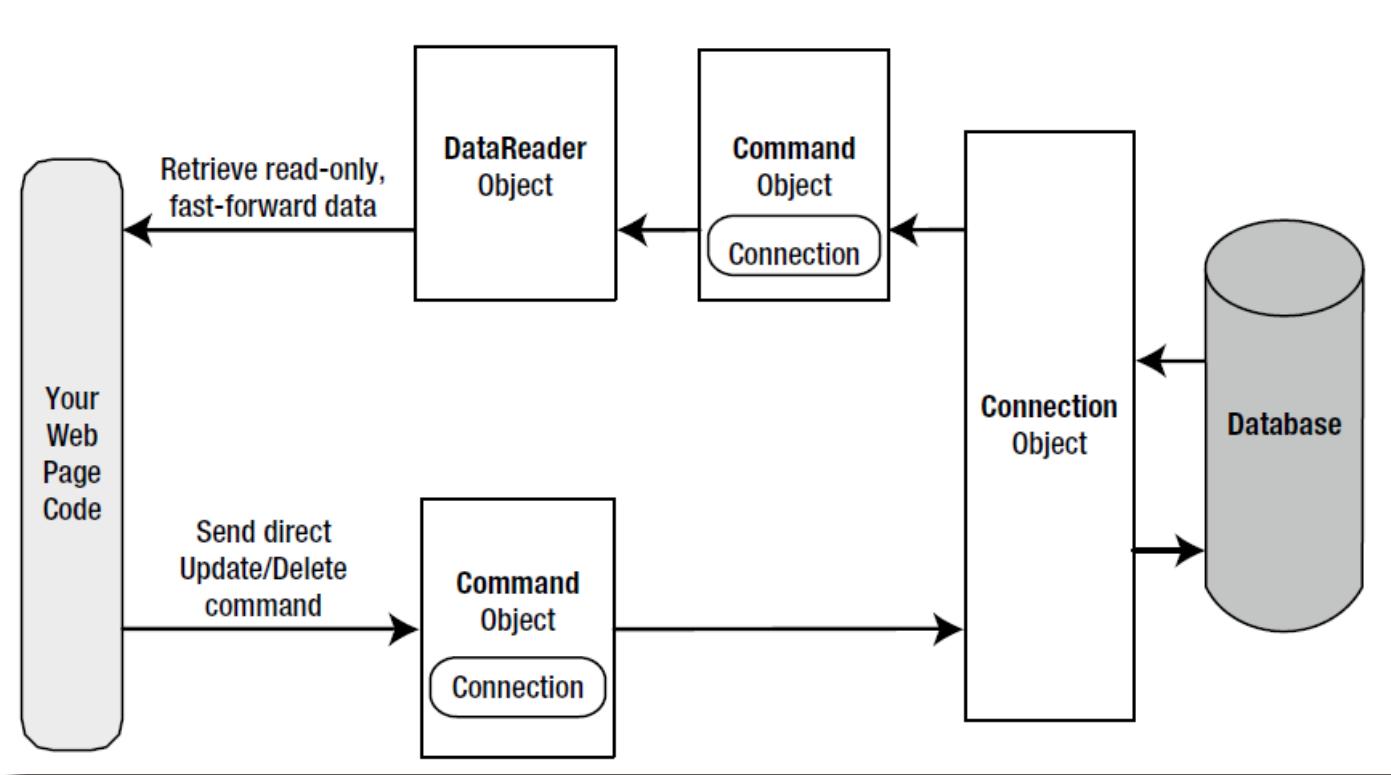
برای query زدن با این روش مراحل زیر را انجام دهید :

- اشیای Connection,Command و DataReader را ایجاد کنید.
- از DataReader برای بازیابی اطلاعات از بانک اطلاعاتی و نمایش آن درون یک کنترل در صفحه وب استفاده کنید.
- صفحه را به کاربر ارسال کنید. در این مرحله، بین اطلاعاتی که کاربر ما می بیند و اطلاعاتی که در بانک موجود است هیچ ارتباطی وجود ندارد و تمامی اشیای ADO.NET نابود شده اند.

برای ایجاد و ویرایش اطلاعات :

۱- یک Connection و Command جدید ایجاد کنید.

۲- SQL دستور (با مناسب) را اجرا کنید (Command).



## ایجاد Connection

قبل از آنکه بتوانید داده ای را بازیابی یا ویرایش کنید، باید یک اتصال به منبع داده ایجاد نمایید. بصورت کلی، اتصال ها به تعداد مشخصی می توانند ایجاد شوند و اگر بیش از آن باشد، تلاش شما برای ایجاد آن با شکست روبرو می شود. به همین دلیل، باید سعی کنید اتصال خود را برای زمان کوتاهی باز نگه دارید. همچنین بید کدهای بانک اطلاعاتی را درون بلوک های try-catch بنویسید.

زمانی که یک شی connection ایجاد کردید، باید یک مقدار برای ویژگیConnectionString آن قرار دهید. این ویژگی، کلیه اطلاعات مورد نیاز کامپیوتر برای یافتن منبع داده، ورود به آن (login) و انتخاب بانک مورد نظر را دارد.

```
SqlConnection myConnection = new SqlConnection();
myConnection.ConnectionString = "Data Source=localhost;" +
"Initial Catalog=Pubs;Integrated Security=SSPI";
```

اگر از نسخه Express استفاده می کنید عبارت شما باید دارای instance name باشد :

```
SqlConnection myConnection = new SqlConnection();
myConnection.ConnectionString = @"Data Source=(localdb)\v11.0;" +
"Initial Catalog=Pubs;Integrated Security=SSPI";
```

 نکته: از علامت @ در عبارت بالا به منظور جلوگیری از ایجاد اشتباه در متن توسط کامپایلر C# استفاده شده است. چون از \ استفاده شده یا باید از پیپ استفاده می شد یا در ابتدای رشته از @ استفاده می کردیم.

درون این رشته اتصال بخش هایی وجود دارد :

**Data source**: نام سروری که منبع داده در آن قرار دارد را مشخص می کند. اگر سرور در همان کامپیووتری باشد که سایت ASP.NET ما وجود دارد از عبارت Localhost استفاده می کنیم. همانطور که قبلًا هم گفتیم صورتیکه از نسخه Express استفاده شود، از localhost\SQLEXPRESS و localhost\v11.0 (یا در صورتیکه بانک روی یک سرور دیگر قرار داشته باشد : ServerName\SQLEXPRESS) استفاده می کنیم. همچنین می توان از \SQLEXPRESS نیز استفاده کرد. نقطه با localhost برابر است.

**Initial Catalog**: نام بانک اطلاعاتی می باشد که این اتصال به آن دسترسی خواهد داشت. این ویژگی را می توانید توسط متده Connection.ChangeDatabase() تغییر دهید.

**Integrated Security**: تعیین می کند که شما می خواهید با استفاده از حساب کاربری ویندوزی که کد صفحه وب روی آن اجرا شود به SQL Server متصل شوید و مقدار Security Support Provider Interface (SSPI) را می گیرد. البته می توانید به عنوان جایگزین از یک UserID و Password که برای تعیین اعتبار در SQL Server تعریف شده است استفاده کنید.

**Connection TimeOut**: تعداد ثانیه هایی که کد شما منتظر ایجاد یک خطأ (در صورتیکه نتواند یک ارتباط با بانک برقرار کند) می ماند را تعیین می کند. اگر آن را تعیین نکنید، پیش فرض آن 15 ثانیه می باشد. با استفاده از ۰ (صفر) می توانید آن را بدون محدودیت کنید.

## ذخیره Connection String (رشته اتصال)

معمولًا تمامی کدهای بانک اطلاعاتی در برنامه شما، از یک رشته اتصال مشابه استفاده می کنند. بنابراین نیاز به ذخیره رشته اتصال در یک متغیر عضو کلاس یا درون فایل پیکربندی احساس می شود.

همچنین می توانید یک شی Connection ایجاد و برای آن مقدار رشته اتصال را در سازنده کلاس مربوطه، مقدار دهی کنید.

```
SqlConnection myConnection = new SqlConnection(connectionString)
// myConnection.ConnectionString is now set to connectionString.
```

نیازی به hard-code (نوشتن دستی یک مقدار در کد) کردن رشته اتصال نمی‌باشد. بخش `<connectionStrings>` در فایل web.config، مکانی برای ذخیره رشته اتصال می‌باشد.

```
<configuration>
  <connectionStrings>
    <add name="Pubs" connectionString=
      "Data Source=localhost;Initial Catalog=Pubs;Integrated Security=SSPI"/>
  </connectionStrings>
  . . .
</configuration>
```

سپس می‌توانید این رشته اتصال را با استفاده از نام، درون برنامه بازیابی کنید. ابتدا System.Web.Configuration را Import کنید.

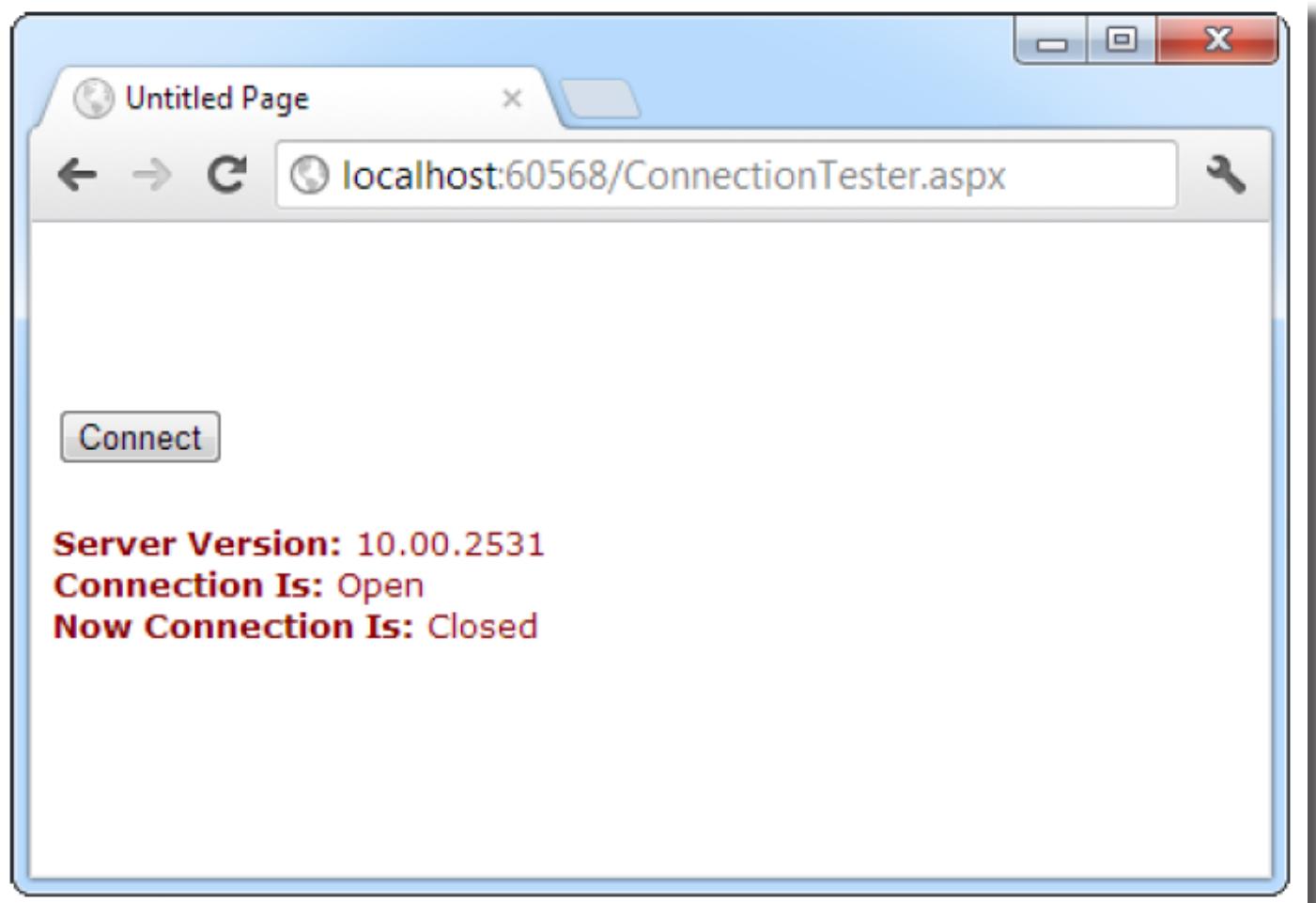
```
string connectionString =
  WebConfigurationManager.ConnectionStrings["Pubs"].ConnectionString;
```

با استفاده از این روش مطمئن می‌شوید همه صفحات وب شما از یک رشته اتصال استفاده می‌کنند و به سادگی می‌توانید آن را از یک محل تغییر دهید.

## ایجاد Connection

پس از ایجاد Connection، باید از آن استفاده کنید. پیش از اجرای هر دستوری روی بانک اطلاعاتی، باید Connection خود را باز کنید:

```
myConnection.Open();
```



```
// Define the ADO.NET Connection object.

string connectionString =
    WebConfigurationManager.ConnectionStrings["Pubs"].ConnectionString;
SqlConnection myConnection = new SqlConnection(connectionString);

try
{
    // Try to open the connection.

    myConnection.Open();

    lblInfo.Text = "<b>Server Version:</b> " + myConnection.ServerVersion;
```

```

lblInfo.Text += "<br /><b>Connection Is:</b> " +
myConnection.State.ToString();

}

catch (Exception err)

{

    // Handle an error by displaying the information.

    lblInfo.Text = "Error reading the database. ";

    lblInfo.Text += err.Message;

}

finally

{

    // Either way, make sure the connection is properly closed.

    // (Even if the connection wasn't opened successfully,

    // calling Close() won't cause an error.)

    myConnection.Close();

    lblInfo.Text += "<br /><b>Now Connection Is:</b> ";

    lblInfo.Text += myConnection.State.ToString();

}

```

پس از استفاده از متدها Open()، شما یک Connection زنده دارید. باید زمان باز نگه داشتن یک اتصال را کوتاه نمایید. بستن یک اتصال خیلی ساده است :

```
myConnection.Close();
```

روش دیگر برای این کار استفاده از دستور Using می‌باشد که می‌توانید اشیای Disposable را برای زمان کوتاهی در آن استفاده کنید. به محض اتمام بلوک Using، آن شی را با استفاده از فراخوانی متدها Dispose() آزاد می‌کند. بنابراین در مثال بالا، اتصال ما به بانک به صورت خودکار ایجاد و از بین می‌رود.

```

using (object)
{
}

```

در روش بالا شما نیازی به بلوک finally ندارد.

```
// Define the ADO.NET Connection object.

string connectionString =
WebConfigurationManager.ConnectionStrings[“Pubs”].ConnectionString;
SqlConnection myConnection = new SqlConnection(connectionString);

try
{
    using (myConnection)
    {
        // Try to open the connection.

        myConnection.Open();

        lblInfo.Text = “<b>Server Version:</b> “ + myConnection.ServerVersion;
        lblInfo.Text += “<br /><b>Connection Is:</b> “ +
        myConnection.State.ToString();
    }
}

catch (Exception err)
{
    // Handle an error by displaying the information.

    lblInfo.Text = “Error reading the database. “;
    lblInfo.Text += err.Message;
}

lblInfo.Text += “<br /><b>Now Connection Is:</b> “;
lblInfo.Text += myConnection.State.ToString();
```

## استفاده از DataReader

پس از تعریف دستور خود، باید تصمیم بگیرید که چگونه می‌خواهید از آن استفاده کنید. ساده‌ترین راه استفاده از DataReader می‌باشد که اجازه بازیابی سریع نتیجه را می‌دهد. DR. از یک live Connection استفاده می‌کند و باید به سرعت آن را بیندد. DR از قابلیتی به اسم read-only نیز می‌باشد استفاده می‌کند که در بازیابی اطلاعات به درد شما می‌خورد. fast-forward-only کارایی بهتری از DataSet را برای دسترسی مستقیم به داده می‌باشد.

قبل از آنکه از DR استفاده کنید، مطمئن شوید که Connection را باز کرده‌اید:

```
myConnection.Open();
```

برای ایجاد یک DR، از متده استفاده می‌کنیم :

```
// You don't need the new keyword, as the Command will create the DataReader.
```

```
SqlDataReader myReader;
```

```
myReader = myCommand.ExecuteReader();
```

پس از داشتن reader، می‌توانید یک ردیف تنها را با استفاده از متده Read() بازیابی کنید :

```
myReader.Read(); // The first row in the result set is now available.
```

سپس برای دسترسی به مقادیر در ردیف کنونی می‌توانید از نام فیلد مرتبط، استفاده نمایید.

```
lstNames.Items.Add(myReader["au_lname"] + ", " + myReader["au_fname"]);
```

برای حرکت به ردیف بعدی دوباره از متده Read() استفاده کنید. اگر حاصل True بود، یعنی اطلاعات سطر بعدی به درستی بازیابی شده است و اگر نه یعنی شما به انتهای Result Set (مجموعه رکوردهای بازیابی شده توسط دستور شما) رسیده اید.

به محض پایان خواندن رکوردها، DR و Connection را ببندید :

```
myReader.Close();
```

```
myConnection.Close();
```

## استفاده از Disconnected Data Access

در این روش از DataSet، برای نگهداری یک کپی از داده‌ها در حافظه استفاده می‌کنیم. باید به بانک اطلاعاتی متصل شوید و سپس داده‌خود را بازیابی کنید و درون Dataset قرار دهید و سپس فوراً disconnect شوید.

دلایل گوناگونی برای استفاده از DataSet و نگهداری داده‌ها در حافظه وجود دارد :

- بعضی وقت‌ها نیاز به کارهای زمان بر روی داده داریم. با قرار دادن آن در Dataset، مطمئن می‌شویم Connection شما برای زمان کوتاهی باز می‌باشد.

- اگر بخواهید از ASP.NET data binding (اتصال داده به کنترل‌های asp.net) برای پر کردن یک کنترل وبی (مانند GridView) استفاده کنید.

- هنگامی که می‌خواهید در زمان پردازش داده‌ها روی آنها به جلو و عقب حرکت کنید. این کار با DataReader که فقط روبه جلو حرکت می‌کند امکان پذیر نیست.

- اگر بخواهید از یک جدول به جدول دیگر بروید می‌توانید با استفاده از DS، اطلاعات چندین جدول را نگهداری کنید. حتی

می‌توانید ارتباط بین آنها را تعریف کنید.

- اگر بخواهید داده‌ها را درون یک فایل برای استفاده بعدی ذخیره کنید. DS دارای دو متده است `ReadXML()` و `WriteXML()`. می‌باید که اجازه اتصال به فایل و تبدیل آن به یک شی بانک اطلاعاتی زنده را خواهد داد.

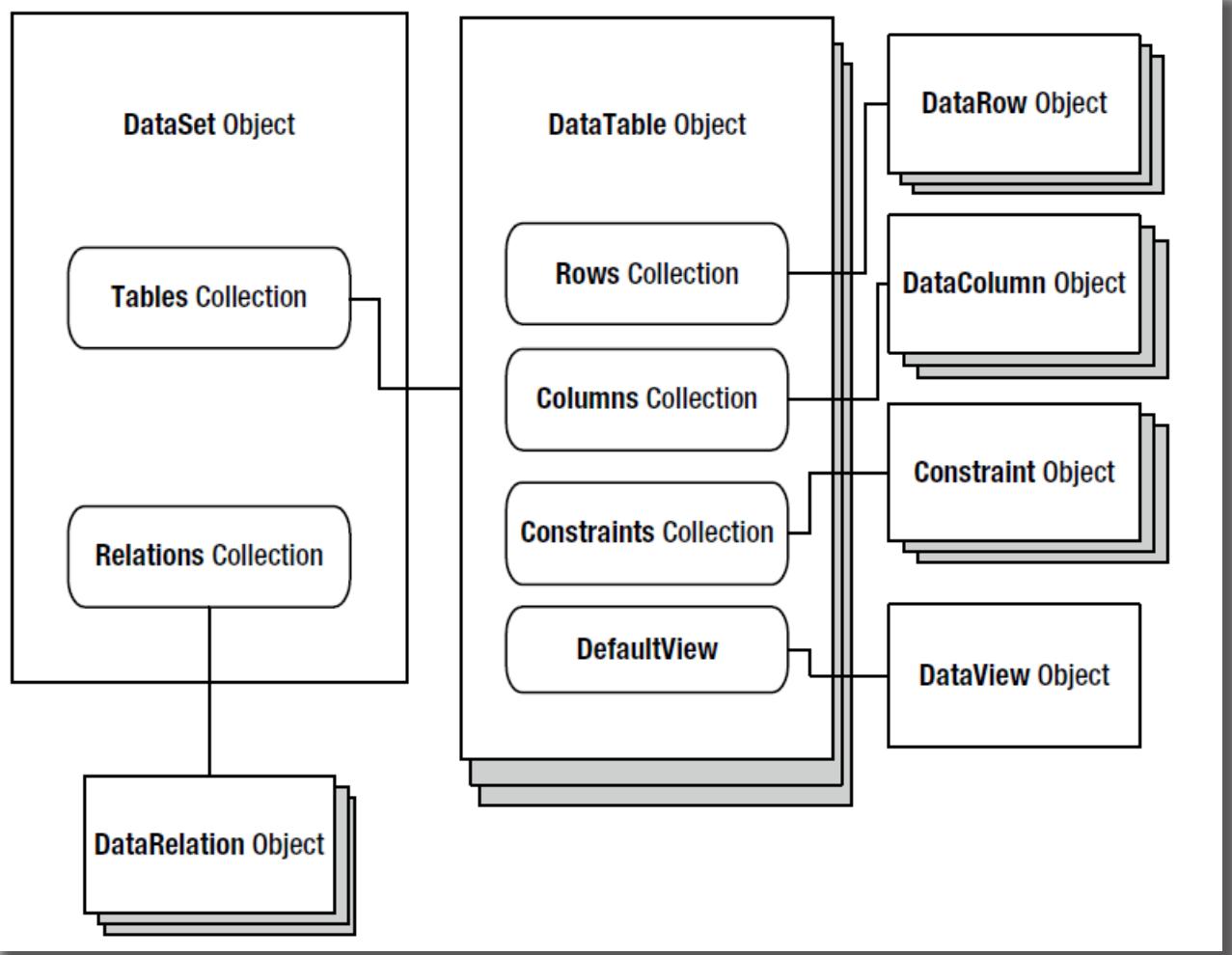
- اگر نیاز به یک بسته مناسب برای انتقال داده از یک کامپوننت دیگر داشته باشید. برای نمونه، می‌توان ای داشت که داده‌ها را با استفاده از DS، به یک صفحه ارائه می‌دهد.

- اگر بخواهید داده ای را ذخیره کنید تا در درخواست‌های بعدی مورد استفاده قرار بگیرد. می‌توانید از Dataset Caching استفاده کنید.

DS، تغییراتی که به داده‌ها اعمال می‌شود را درون خود نگهداری می‌کند. بنابراین می‌توان از آن برای ویرایش رکوردها استفاده کرد. این کار ساده است. یک یا چند رکورد را تغییر می‌دهید و سپس ویرایش خود را با استفاده از یک DataAdapter پیاده می‌کنید.

این روش برای برنامه‌های دسکتاپ مناسب‌تر است. زیرا آنها برای مدت طولانی در حال اجرا می‌باشند، بنابراین می‌توانند دسته‌ای 9batch) از تغییرات را ذخیره و در یک زمان همه آنها را اجرا نمایند. اما در برنامه‌های تحت وب شما باید تغییرات خود را همان لحظه که آنها رخ می‌دهند اعمال (Commit) کنید. علاوه براین، نقطه‌ای شما داده را بازیابی می‌کنید (دفعه اولی که صفحه درخواست می‌شود) و نقطه‌ای که آن داده تغییر می‌کند (در هنگام postback)، متفاوت می‌باشد. بنابراین بسیار سخت است که از یک شی DS مشابه برای نگهداری change-tracking (ردیابی تغییرات) اطلاعات در کل پردازش استفاده کنیم.

به دلایل ذکر شده، DS از ASP.NET برای ذخیره داده‌ها استفاده می‌کند ولی به ندرت از آن برای اعمال تغییرات و ویرایش استفاده خواهد کرد. در عوض دستورات مستقیمی برای اعمال تغییرات وجود دارد.



: مثال 

```

private void FillAuthorList()
{
    lstAuthor.Items.Clear();
    // Define ADO.NET objects.
    string selectSQL;
    selectSQL = "SELECT au_lname, au_fname, au_id FROM Authors";
    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand(selectSQL, con);
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataSet dsPubs = new DataSet();
    // Try to open database and read information.
    try
    {
        
```

```

con.Open();

// All the information is transferred with one command.

// This command creates a new DataTable (named Authors)

// inside the DataSet.

adapter.Fill(dsPubs, "Authors");

}

catch (Exception err)

{

    lblStatus.Text = "Error reading list of names. ";

    lblStatus.Text += err.Message;

}

finally

{

    con.Close();

}

foreach (DataRow row in dsPubs.Tables["Authors"].Rows)

{

    ListItem newItem = new ListItem();

    newItem.Text = row["au_lname"] + ", " +

    row["au_fname"];

    newItem.Value = row["au_id"].ToString();

    lstAuthor.Items.Add(newItem);

}

}

```

# پنجم سوم کار با داده ها

فصل یازدهم: ADO.NET

فصل یازدهم: Data Binding

فصل یازدهم: Data Controls



## Data Binding

در فصل قبل، یاد گرفتید که چگونه با استفاده از ADO.NET، داده ها از بانک اطلاعاتی بازیابی، در DataSet ذخیره و تغییراتی با استفاده از دستورات مستقیم روی آنها ایجاد کنید. این تکنیک ها قادر تمند هستند اما همیشه مناسب نمی باشند.

برای نمونه، می توانید از dataReader یا DataSet برای بازیابی ردیف های اطلاعات، فرمت دهی آنها و اضافه نمودن آنها به جداول HTML ای استفاده کنید. خوبی بختانه، ASP.NET، امکانی با نام data binding را ارائه داده است که به شما اجازه انتقال مستقیم داده به المان های HTML را می دهد. در این فصل، نحوه استفاده از data binding برای نمایش داده شرح خواهیم داد. همچنین یاد خواهید گرفت که چگونه از ASP.NET data source controls برای بازیابی داده از بانک اطلاعاتی بدون نوشتن یک خط از کد استفاده کنید.

## Data Binding معرفی

قوانين پایه ای databinding این است: به کنترل مورد نظر می گویید، کجا می تواند داده خود را پیدا کند و چگونه می خواهد آن را نمایش دهد. بقیه کارها را خود کنترل هندل می کند. در برنامه های دسکتاب، databinding، شامل ایجاد یک اتصال مستقیم بین منبع داده و کنترل می باشد. اگر کاربر یک مقدار را در کنترل تغییر دهد، داده درون بانک اطلاعاتی بصورت خودکار تغییر خواهد کرد. همچنین اگر بانک اطلاعاتی در هنگام کار کاربر تغییر کند، صفحه نمایش می تواند بصورت خودکار refresh شود.

این روش در ASP.NET کارایی ندارد زیرا نمی توانید به طرز موثری اتصال بانک اطلاعاتی را روی اینترنت نگهداری کنید.

ASP.NET در Databinding، یک طرفه است. اطلاعات از شی داده (dat object) به درون کنترل منتقل می شود. سپس این اشیا دور انداخته می شوند و صفحه به کلاینت فرستاده می شوند. اگر کاربر داده را در کنترل data-bound شده (کنترلی که داده های منبع داده را به آن وصل کرده اند)، تغییر دهد، برنامه شما می تواند رکوردهای مرتبط را در بانک اطلاعاتی ویرایش کند، البته نه بصورت خودکار.

## أنواع ASP.NET Data Binding

دو نوع از این data binding وجود دارد :

### Single - value binding -

برای اضافه نمودن اطلاعات در هر جای صفحه ASP.NET استفاده می شود. حتی می توانید اطلاعات را درون یک control property یا متن موجود در تگ HTML قرار دهید. این روش اجازه می دهد تا یک متغیر، property یا یک عبارت را گرفته و آن را بصورت خودکار به صفحه اضافه کنید.

### list binding یا Repeated – value binding -

این روش امکان نمایش کل جدول (یا تنها یک فیلد از یک جدول) را می دهد. برخلاف روش قبل، این روش نیاز به کنترل های خاصی برای پشتیبانی از آن دارد. عموماً کنترل هایی مانند CheckBoxList یا ListBox و بصورت کلی List Control ها هستند که از این روش استفاده می کنند. کنترلی از این روش استفاده می کند که دارای ویژگی DataSource باشد. در این روش نیازی نیست تا داده ها حتماً از بانک اطلاعاتی بیاید و می توانید داده ها را از یک collection یا array bind هم کنید.

## نحوه عملکرد Data Binding

اً توجه به دو روش بالا، نحوه عملکرد data binding اندکی فرق می‌کند. در روش single-value، باید یک عبارت data-bind می‌داند که درون markup صفحه (کد HTML) اضافه کنید. برای استفاده از repeated-value، باید یک یا چند property از data control را مقدار دهی کنید. عموماً این مقداردهی را در رویداد Page.Load انجام می‌دهید.

برای فعال سازی data binding، باید متدها DataBind() را فراخوانی کنید. این متدها بصورت خودکار کنترلی که شامل آن می‌شود را bind می‌کنند. می‌توانید این متدها را از list control استفاده کنید یا کل صفحه را با استفاده از متدها DataBind() موجود در شی data-binding کنید. پس از فراخوانی این متدها، تمامی عبارت‌ها موجود در صفحه ارزیابی شده و با مقادیر مشخص جایگزین می‌شوند.

### استفاده از Single-Value Data Binding

برای استفاده از این روش، باید عبارت data-binding خود را به فایل aspx اضافه کنید. این عبارت دارای فرمت زیر است :

```
<%# عبارت شما در اینجا قرار می‌گیرد %>
```

این عبارت ممکن است مانند script block به نظر برسد. اگر بخواهید درون آن کدی بنویسد با error روبرو می‌شوید. تنها چیزی که می‌توانید به آن اضافه کنید، یک عبارت data-binding معتبر می‌باشد. برای نمونه اگر یک متغیر protected public یا با نام Country در صفحه خود داشته باشید، می‌توانید عبارت زیر را بنویسید :

```
<%# Country %>
```

زمانی که متدها DataBind() را برای صفحه فراخوانی کنید، این متن با مقدار Country جایگزین می‌شود. همچنین می‌توانید از یک property یک شی ASP.NET مانند زیر استفاده کنید :

```
<%# Request.Browser %>
```

این عبارت، یک رشته را با نام مرورگر (مانند IE) جایگزین می‌کند. در حقیقت می‌توانید حتی یک متدها موجود در صفحه را فراخوانی کنید.

```
<%# GetUserName(ID) %>
<%# 1 + (2 * 20) %>
<%# "John " + "Smith" %>
```

به یاد داشته باشید عبارات بالا را باید در بخش markup فایل aspx خود قرار دهید و نه در code-behind.

## یک مثال ساده از Data-Binding



```
public partial class SimpleDataBinding : System.Web.UI.Page
{
    protected int TransactionCount;
    // (Additional code omitted.)
}
```

توجه کنید که متغیر شما می‌تواند Public, Protected یا Internal باشد ولی نمی‌تواند Private باشد. اکنون فرض کنید این متغیر در رویداد Page.Load، توسط کدهایی برای بازیابی داده از بانک اطلاعاتی مقدار دهی شده است. در زیر این مقدار را بصورت hard-code یا دستی به متغیر می‌دهیم:

```
public partial class SimpleDataBinding : System.Web.UI.Page
{
    protected int TransactionCount;
    // (Additional code omitted.)

    protected void Page_Load(object sender, EventArgs e)
    {
        // (You could use database code here
        // to look up a value for TransactionCount.)

        TransactionCount = 10;

        // Now convert all the data-binding expressions on the page.

        this.DataBind();
    }
}
```

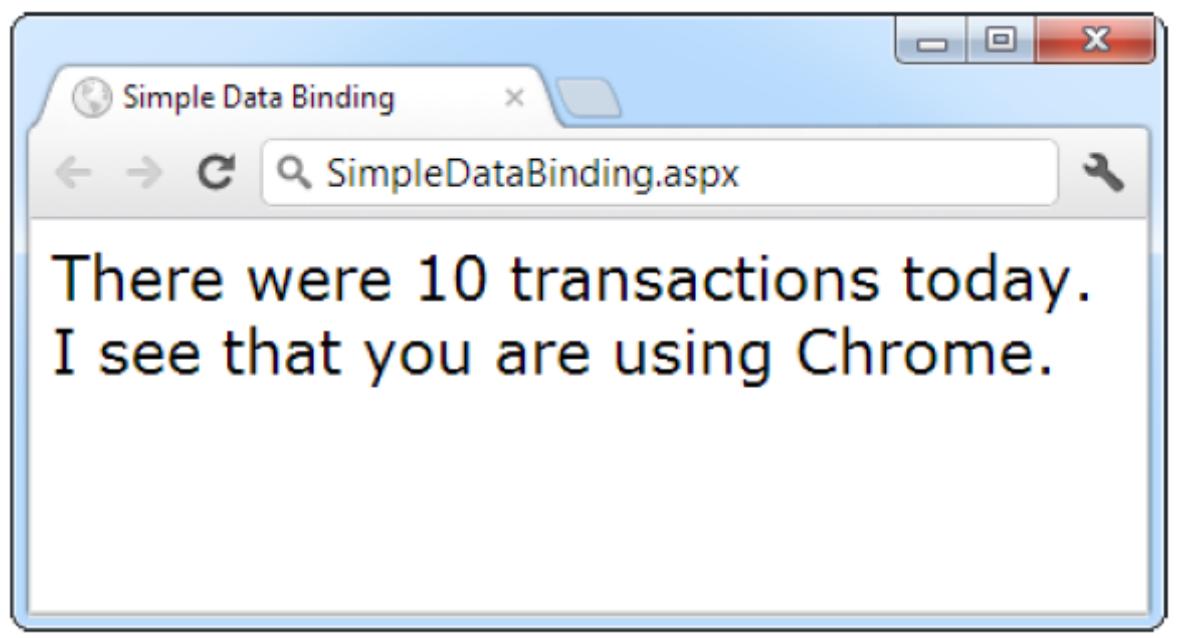
برای تکمیل data binding باید عبارات مخصوص به آن را اضافه کنیم.

## SimpleDataBinding.aspx\*

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="SimpleDataBinding.aspx.cs" %>
<!DOCTYPE html>
<html>
<head runat="server">
    <title>Simple Data Binding</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label id="lblDynamic" runat="server" Font-Size="X-Large" >
                There were <%# TransactionCount %> transactions today.
                I see that you are using <%# Request.Browser.Browser %>.
            </asp:Label>
        </div>
    </form>
</body>
</html>
```

100 %

```
<asp:label id="lblDynamic" runat="server" font-size="X-Large">
    There were <%# TransactionCount %> transactions today.
    I see that you are using <%# Request.Browser.Browser %>.
</asp:label>
```



 نکته: زمانی که از روش single-value data binding استفاده می‌کنید، در نظر بگیرید که باید متدها DataBind() را در چه زمانی فراخوانی کنید. برای نمونه اگر این اشتباه را بکنید که این متدها قبل از مقدار دهنده متغیر TransactionCount، فراخوانی کنید، عبارت مرتبط با آن مقدار ۰ بر می‌گرداند. به یاد داشته باشید که این روش یک طرفه می‌باشد و این یعنی، تغییرات متغیر TransactionCount پس از فراخوانی متدها DataBind()، تأثیری در خروجی ندارد.

## Property با استفاده از Data Binding

نمونه‌های قبلی، از عبارات data-binding برای مقداردهی یک متن ثابت درون یک تگ label استفاده می‌کرد. می‌توانید از value data binding برای مقداردهی انواع دیگری از اطلاعات درون صفحه شامل property‌های یک کنترل استفاده کنید.

برای نمونه، فرض کنید صفحه زیر یک متغیر با نام URL تعریف کرده است و از آن برای اشاره به یک تصویر در پوشش برنامه استفاده کرده است :

```
public partial class DataBindingUrl : System.Web.UI.Page
{
    protected string URL;
    protected void Page_Load(Object sender, EventArgs e)
    {
        URL = "Images/picture.jpg";
        this.DataBind();
    }
}
```

اکنون می‌توانید از این آدرس برای ایجاد یک Label استفاده کنید :

```
<asp:Label id="lblDynamic" runat="server"><%# URL %></asp:Label>
```

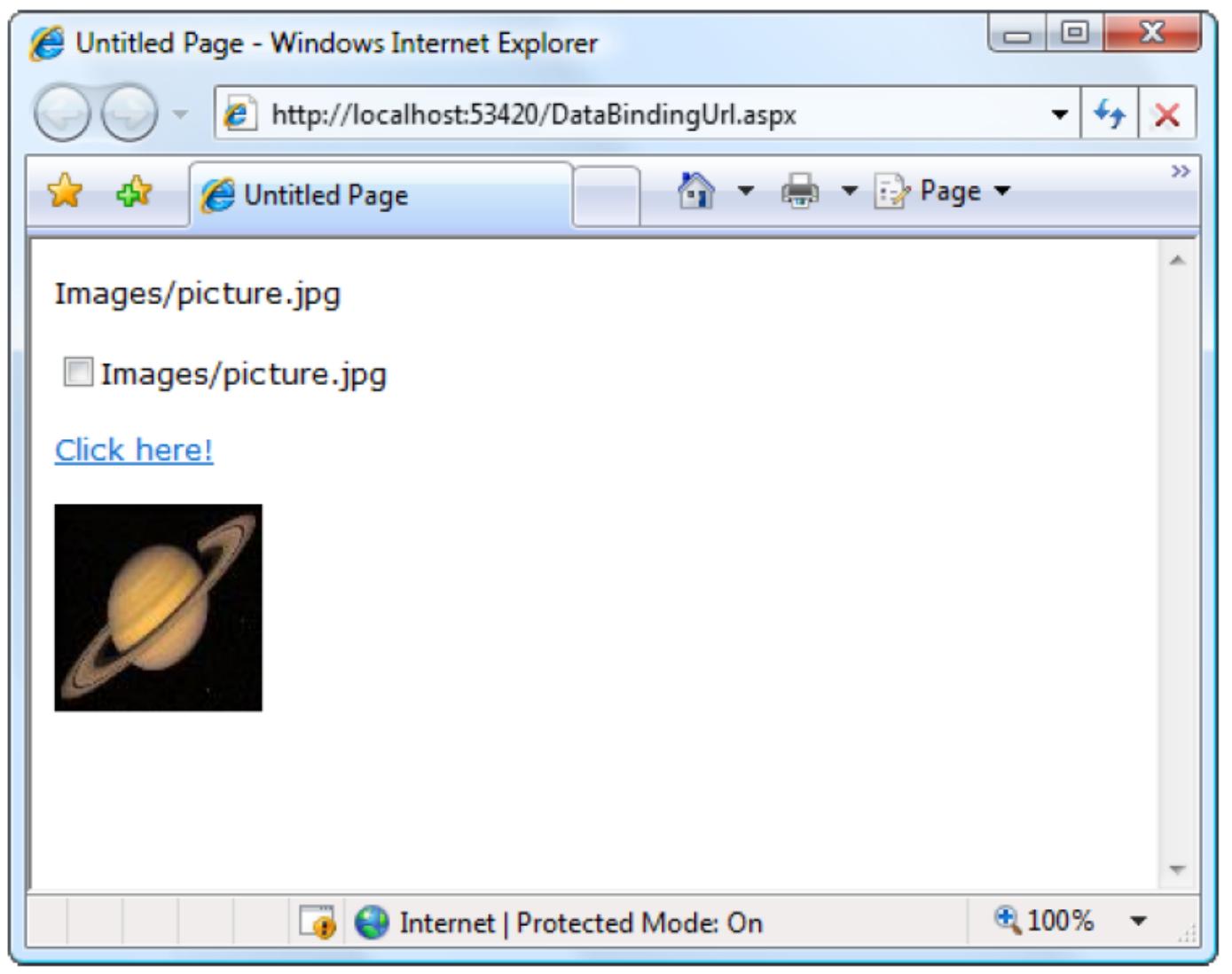
همچنین می‌توانید از آن برای ویژگی caption یک CheckBox استفاده کنید:

```
<asp:CheckBox id="chkDynamic" Text=" <%# URL %> " runat="server" />
```

یا

```
<asp:Hyperlink id="lnkDynamic" Text="Click here!" NavigateUrl="<%# URL %>" runat="server" />
```

```
<asp:Image id="imgDynamic" ImageUrl="<%# URL %>" runat="server" />
```



### مشکلات Single – Value Data Binding

- قرار دادن کد درون واسط کاربری : یکی از فواید ASP.NET این است که به برنامه نویس امکان جداسازی کدهای واسط کاربر از کدهای مربوط به data access و سایر کارهای انجام شده در code-behind می‌دهد. اگر حواستان نباشد ممکن است این روال را به هم بزنید.

- تکه کردن کدها: با استفاده از این روش نگهداری کد شما را سخت می‌کند. زیرا برای نمونه اگر کد صفحه شما تغییر کند یا نام متغیر یا متده عوض شود، عبارت data binding مربوطه می‌تواند ارائه اطلاعات را بدون هیچ اخطار واضحی قطع کند.

 نکته: روش single-value برای ساخت template ها مناسب است. template ها بلوک هایی از کدهای markup هستند که برای هر رکورد در یک جدول استفاده می‌شوند. آنها تنها با rich control هایی مانند GridView کار می‌کنند.

```
protected void Page_Load(Object sender, EventArgs e)
{
    TransactionCount = 10;

    lblDynamic.Text = "There were " + TransactionCount.ToString();

    lblDynamic.Text += " transactions today. ";

    lblDynamic.Text += "I see that you are using " + Request.Browser.Browser;

}
```

## استفاده از Repeated-Value Data Binding

اکثر برنامه های ASP.NET، به نحوی از این روش استفاده می‌کنند. این روش با کنترل های لیستی کار می‌کند. برای استفاده از این روش باید یکی از این کنترل ها را به یک منبع داده (مانند یک فیلد در یک جدول داده) وصل کنید. زمانی که () DataBind را فراخوانی می‌کنید، کنترل بصورت خودکار یک لیست کامل با استفاده از کلیه مقادیر مرتبط ایجاد می‌کند. این کار شما را از نوشتمن کدهایی که روی آرایه یا جدولی از داده ها، loop بزنند و بصورت دستی المان ها را به کنترل اضافه کند، نجات می‌دهد.

مثال هایی از کنترل های لیستی :

- **RadioButtons و DropDownList** : این کنترل های وب یک لیستی از اطلاعات تک فیلدی را فراهم می‌کند.

- **HTMLSelect** : این کنترل HTML ای سمت سرور، المان <select> را ارائه می‌دهد که مانند کنترل ListBox می‌باشد. از این کنترل برای تطبیق پذیری با المان های قدیمی تر استفاده می‌شود.

- **ListView، GridView، DetailsView، FormView** : این rich web control ها به شما اجازه تکرار لیست هایی که بیش از یک فیلد دارند را می‌دهند. برای نمونه می‌توانید آنها را به یک جدول موجود در DataSet خود Bind کنید تا مقادیر چندین فیلد را نمایش دهند. این کنترل ها گزینه های قدرتمند تر و قابل انعطاف تری برای data binding ارائه می‌دهند.

با استفاده از data binding repeated value، می‌توانید عبارت data-binding.aspx بنویسید یا را با استفاده از property های کنترل، پیاده کنید.

## با استفاده از کنترل های لیستی ساده Data Binding

۱- بعضی از انواع data object را ایجاد و پر کنید. گزینه های زیادی مانند ArrayList و arraye و Hashtable Collection و اشیای DataTable و DataSet دارید. بطور خاص از هر Collection ای که اینترفیس dictionary collection

را پیاده سازی می کند می توانید استفاده کنید.

۲- شی را به کنترل مورد نظر متصل کنید. برای انجام این کار، باید ای از `DataSource` مقدار دهی کنید. اگر آن را به یک `DataSet` کامل `Bind` کردید، نیاز به مقدار دهی ویژگی `DataMember` نیز برای تعیین جدول مورد نظرتان دارد.

۳- فراخوانی متدهای `data binding` به منظور فعال سازی `DataBind()`. این کار را با فراخوانی این متدهای برای همه کنترل های درون آن) انجام دهید.

## مثالی از Simple List- Binding

برای انجام این نوع از `data binding`, یک `listbox` به یک صفحه جدید اضافه کنید. سپس از رویداد `Page.Load` برای ایجاد `Strongly typed List Collection` به عنوان یک منبع داده استفاده کنید :

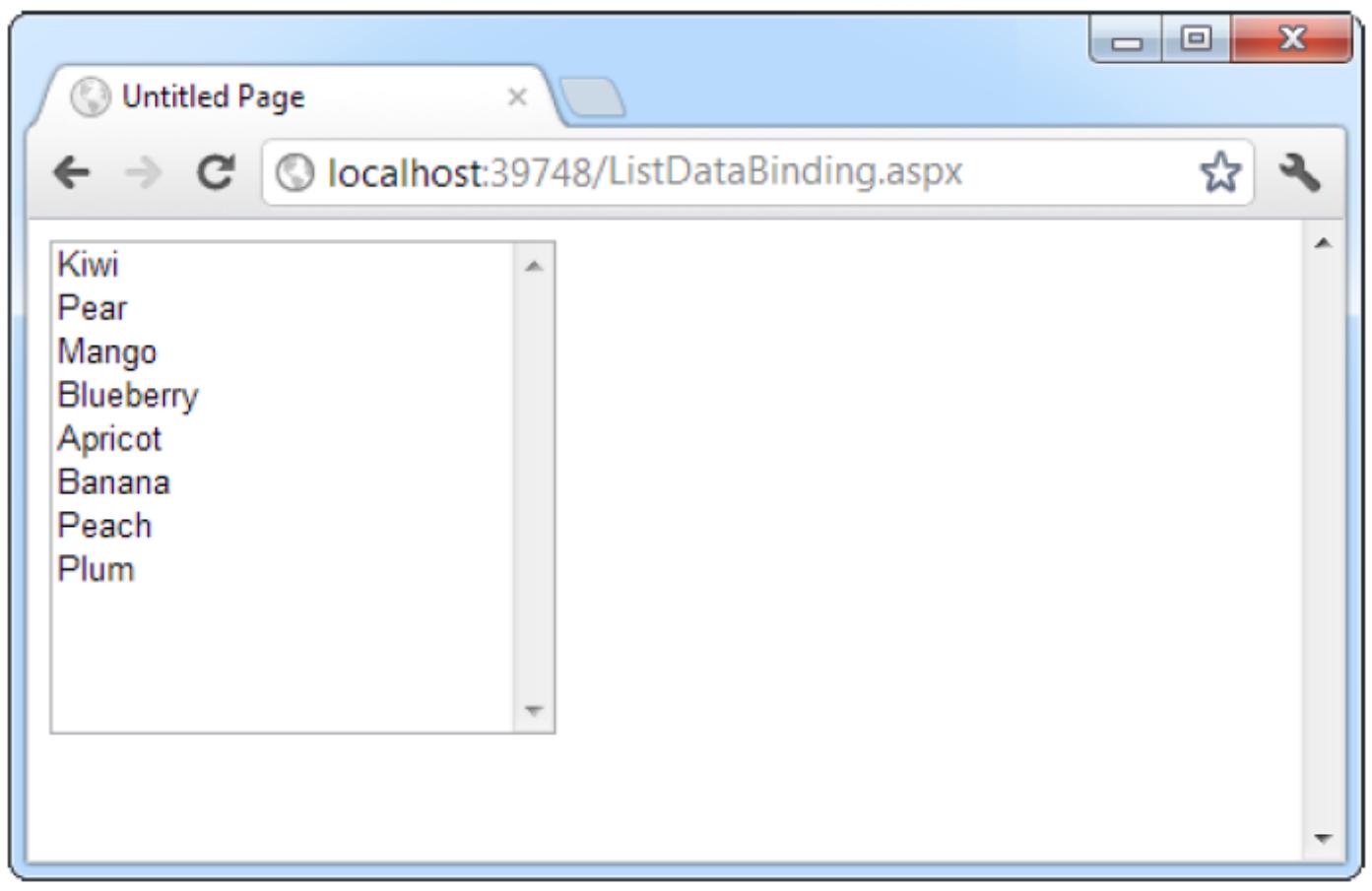
```
List<string> fruit = new List<string>();
fruit.Add("Kiwi");
fruit.Add("Pear");
fruit.Add("Mango");
fruit.Add("Blueberry");
fruit.Add("Apricot");
fruit.Add("Banana");
fruit.Add("Peach");
fruit.Add("Plum");
```

اکنون می توانید این `Collection` را به `listbox` متصل کنید :

```
lstItems.DataSource = fruit;
```

سپس `data binding` را فعال می کنیم :

```
this.DataBind();
```



## Multiple Binding

می‌توانید یک شی data list را به چندین کنترل متصل کنید.

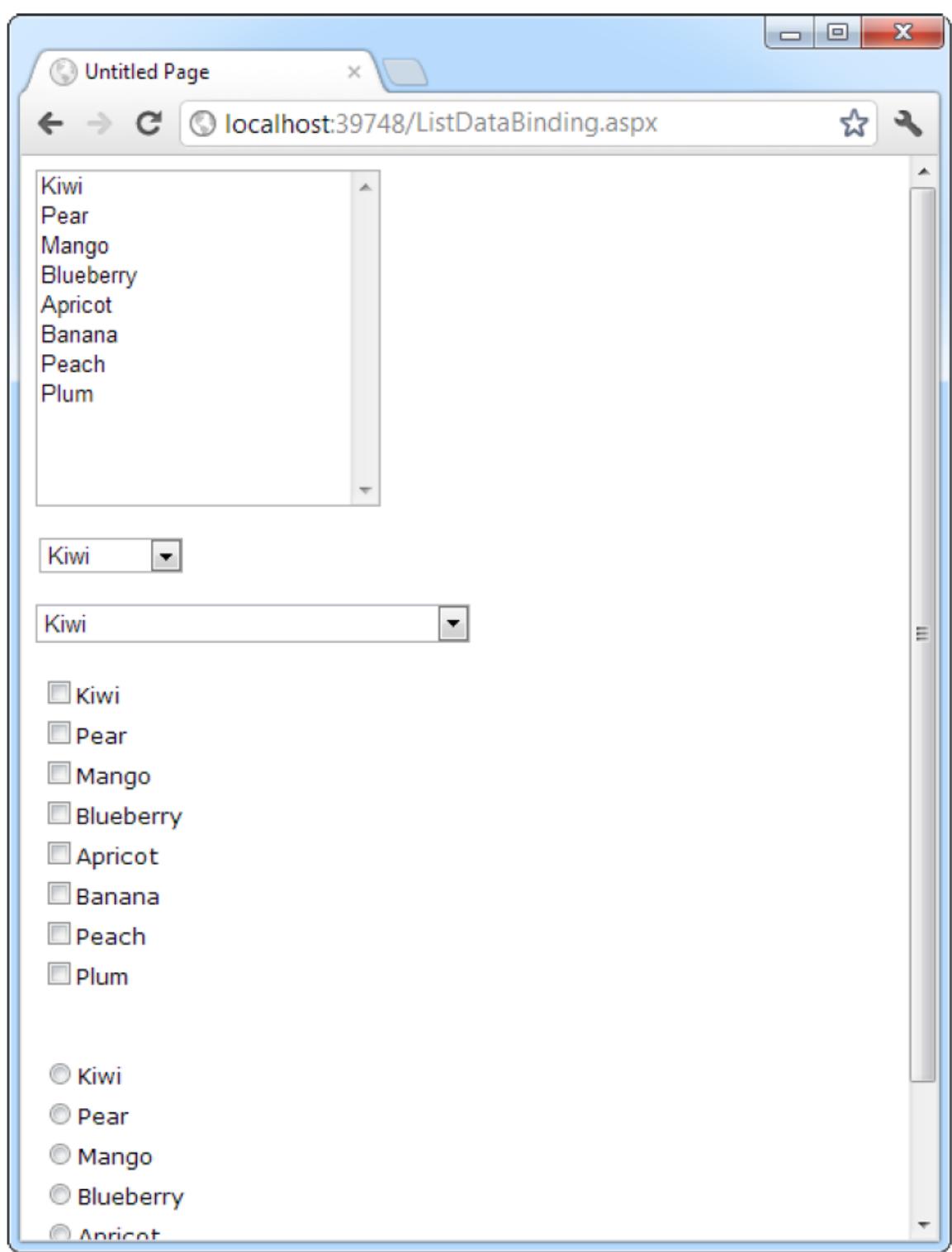
```
protected void Page_Load(Object sender, EventArgs e)
{
    // Create and fill the collection.

    List<string> fruit = new List<string>();

    fruit.Add("Kiwi");
    fruit.Add("Pear");
    fruit.Add("Mango");
    fruit.Add("Blueberry");
    fruit.Add("Apricot");
    fruit.Add("Banana");
    fruit.Add("Peach");
    fruit.Add("Plum");

    // Define the binding for the list controls.
```

```
MyListBox.DataSource = fruit;  
  
MyDropDownListBox.DataSource = fruit;  
  
MyHtmlSelect.DataSource = fruit;  
  
MyCheckBoxList.DataSource = fruit;  
  
MyRadioButtonList.DataSource = fruit;  
  
// Activate the binding.  
  
this.DataBind();  
  
}
```



## Dictionary Collection با یک Data Binding

یک dictionary collection نوع خاصی از collection ها می‌باشد که در هر آیتم خود با یک کلید مشخص ایندکس شده است. این همان چیزی است که collection های موجود در ASP.NET Application و Session Cache از آن استفاده می‌کنند.

DC، همیشه نیاز به کلید دارد. این کلید ها، بازیابی آیتم ها را ساده تر می‌کنند.

در دات نت می‌توانید از دو collection مبتنی بر استایل dictionary استفاده کنید. که اجازه ذخیره هر نوع از اشیا را می‌دهد و از هر نوع از اشیا برای مقادیر کلید استفاده می‌کند. Dictionary Collection از generic List Collection ها مشابه استفاده می‌کند.

مثال های زیر از Dictionary Collection استفاده می‌کنند.

```
protected void Page_Load(Object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        // Use integers to index each item. Each item is a string.
        Dictionary<int, string> fruit = new Dictionary<int, string>();
        fruit.Add(1, "Kiwi");
        fruit.Add(2, "Pear");
        fruit.Add(3, "Mango");
        fruit.Add(4, "Blueberry");
        fruit.Add(5, "Apricot");
        fruit.Add(6, "Banana");
        fruit.Add(7, "Peach");
        fruit.Add(8, "Plum");
        // Define the binding for the list controls.
        MyListBox.DataSource = fruit;
        // Choose what you want to display in the list.
        MyListBox.DataTextField = "Value";
        // Activate the binding.
        this.DataBind();
    }
}
```

هر dictionary دارای دو بخش key و value است که در بالا از استفاده شده است.

## استفاده از ویژگی DataValueField

این ویژگی اطلاعات مرتبط را به خصیصه value المان کنترل مورد نظر اضافه می‌کند. این کار اجازه ذخیره اطلاعات اضافی که به کاربر نمایش داده نمی‌شوند را و بعداً می‌توانید از آن استفاده نمایید را به شما می‌دهد.

```
MyListBox.DataTextField = "Value"
```

```
MyListBox.DataValueField = "Key";
```

کنترل شما با لیستی از نامهای میوه که در collection موجود است پر می‌شود. اگر به HTML رندر شده نگاهی بیندازید می‌بینید که خصیصه Value با مقادیر عددی مربوطه مقدار دهی شده است :

```
<select name="MyListBox" id="MyListBox">

    <option value="1">Kiwi</option>

    <option value="2">Pear</option>

    <option value="3">Mango</option>

    <option value="4">Blueberry</option>

    <option value="5">Apricot</option>

    <option value="6">Banana</option>

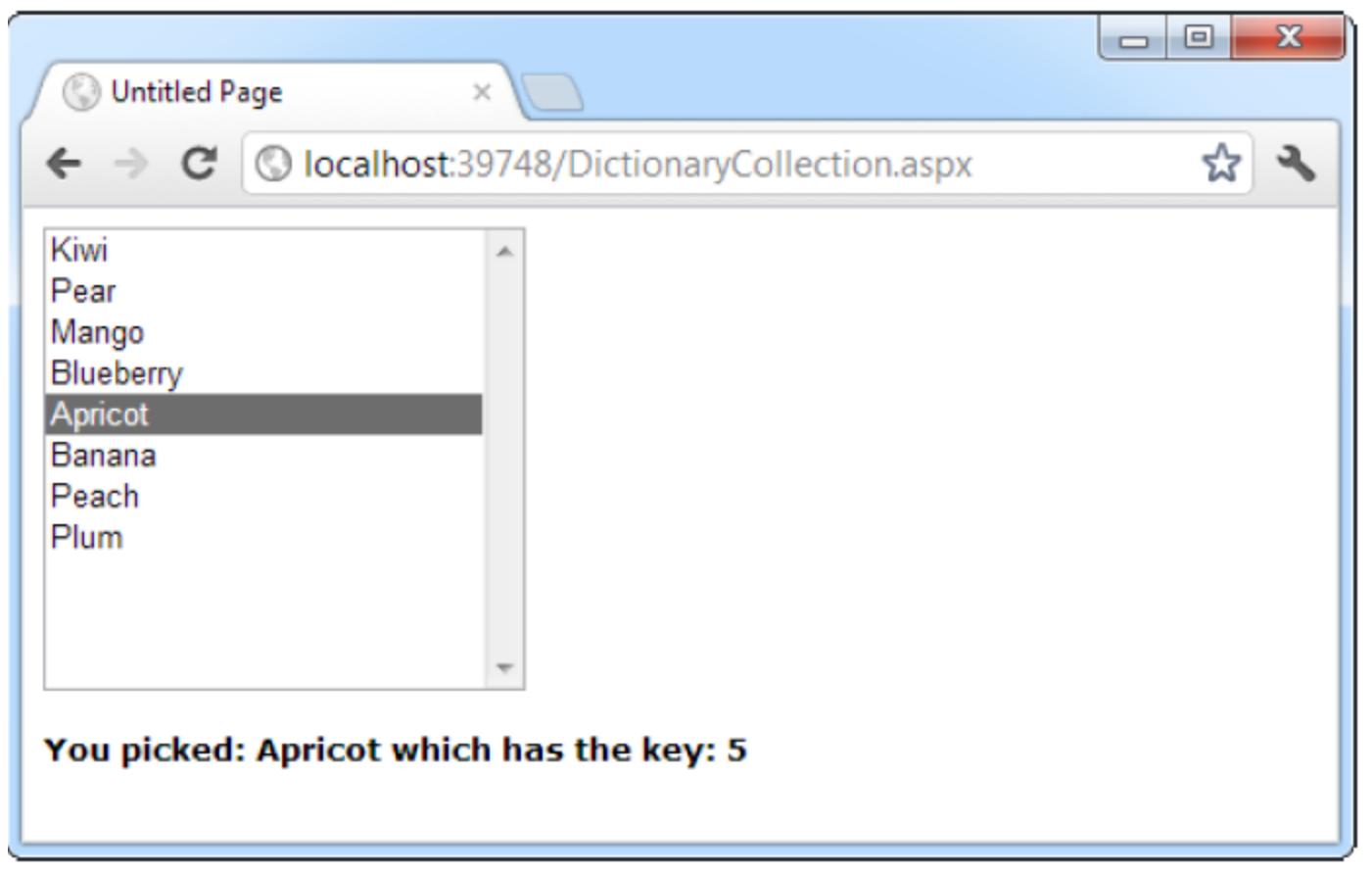
    <option value="7">Peach</option>

    <option value="8">Plum</option>

</select>
```

این مقادیر را می‌توانید بعداً توسط ویژگی SelectedItem بازیابی کنید. برای نمونه می‌توانید ویژگی AutoPostBack این کنترل را برابر با True قرار دهید و کدهای زیر را اضافه نمایید :

```
protected void MyListBox_SelectedIndexChanged(Object sender, EventArgs e)
{
    lblMessage.Text = "You picked: " + MyListBox.SelectedItem.Text;
    lblMessage.Text += " which has the key: " + MyListBox.SelectedItem.Value;
}
```



## ADO.NET با استفاده از Data Binding

تا کنون مثال های زده شده با ADO.NET هایی سروکار داشتند که شامل هیچ بانک اطلاعاتی یا هیچ بخشی از data binding نمی باشد. زمانی که از data binding با اطلاعاتی که از بانک اطلاعاتی آمده است استفاده می کنید، در سه مرحله انجام می شود.

ابتدا منبع ذخیره سازی خود را ایجاد می کنید که یک شی DataSet یا DataReader کارایی بهتری دارد و لی چون forward-only می باشد تنها به یک کنترل می توان آن را bind کرد و پس از آن نمی تواند به عقب برگردد.

برای پرکردن یک DataSet به صورت دستی باید مراحل زیر را طی کرد :

- ۱- یک DataSet ایجاد کنید.
- ۲- یک DataTable ایجاد و آن را به DataSet.Tables اضافه کنید.
- ۳- ساختار جدول را با اضافه کردن اشیای DataColumn (هر یک برای یک فیلد) به DataTable.Columns Collection (هر یک برای یک فیلد) تعريف کنید.
- ۴- با استفاده از DataTable.NewRow، می توانید یک رکورد خالی با ساختار تعريف شده خود داشته باشید. سپس باید داده ها را درون فیلدهای آن مقداردهی کنید و آن را به DataTable.Rows collection اضافه نمایید.

```
// Define a DataSet with a single DataTable.
```

```
DataSet dsInternal = new DataSet();
```

```

dsInternal.Tables.Add("Users");

// Define two columns for this table.

dsInternal.Tables["Users"].Columns.Add("Name");

dsInternal.Tables["Users"].Columns.Add("Country");

// Add some actual information into the table.

DataRow rowNew = dsInternal.Tables["Users"].NewRow();

rowNew["Name"] = "John";

rowNew["Country"] = "Uganda";

dsInternal.Tables["Users"].Rows.Add(rowNew);

rowNew = dsInternal.Tables["Users"].NewRow();

rowNew["Name"] = "Samantha";

rowNew["Country"] = "Belgium";

dsInternal.Tables["Users"].Rows.Add(rowNew);

rowNew = dsInternal.Tables["Users"].NewRow();

rowNew["Name"] = "Rico";

rowNew["Country"] = "Japan";

dsInternal.Tables["Users"].Rows.Add(rowNew);

```

سپس باید DataTable موجود در Dataset را به کنترل مناسب bind کنید. به دلیل اینکه list control ها تنها می‌توانند یک ستون را نمایش دهند، باید فیلدی که می‌خواهید نمایش داده شود را توسط ویژگی DataTextField مقدار دهی نمایید.

```

// Define the binding.

1stUser.DataSource = dsInternal.Tables["Users"];

1stUser.DataTextField = "Name";

```

همچنین می‌توانید به جای یک جدول از کل DataSet به عنوان منبع داده استفاده کنید. در این روش باید جدول خود را با استفاده از ویژگی DataMember تغییر دهید.

```

// Define the binding.

1stUser.DataSource = dsInternal;

1stUser.DataMember = "Users";

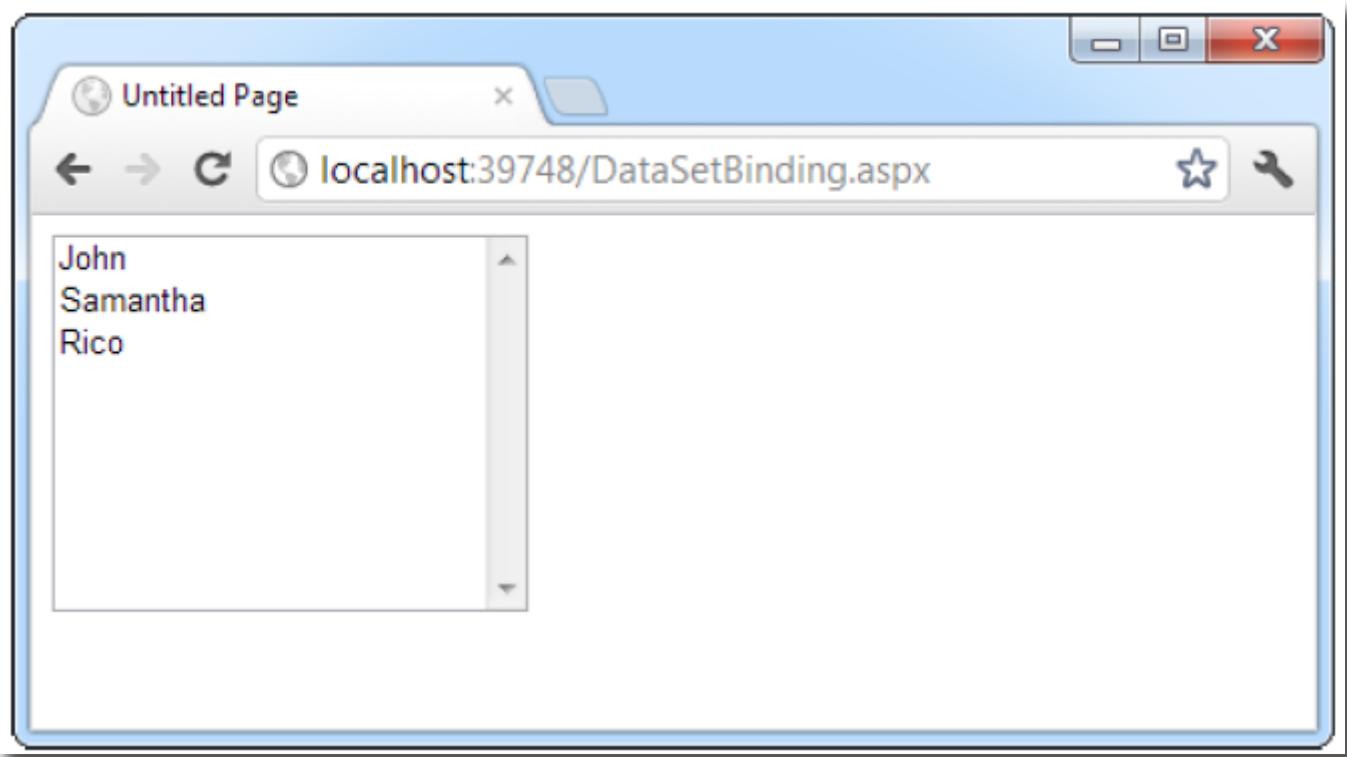
1stUser.DataTextField = "Name";

```

همیشه در انتهای binding خود را فعال سازید:

```
this.DataBind();
```

نتیجه نهایی لیستی از اطلاعات یک فیلد مشخص از بانک می‌باشد.



## ایجاد یک ویرایشگر رکورد

در نمونه طرح شده در این بخش، کاربر می‌تواند یک رکورد را انتخاب و بخشی از آن را با استفاده از کنترل‌های لیستی bind شده ویرایش نماید.

ابتدا Connection String را به فایل Web.Config اضافه نمایید.

این مثال از جدول Products از بانک NorthWind استفاده می‌کند.

```
<configuration>
<connectionStrings>
<add name="Northwind" connectionString=
      "Data Source=(localdb)\v11.0;Initial Catalog=Northwind;Integrated
      Security=SSPI" />
</connectionStrings>
...
</configuration>
```

مرحله بعدی بازیابی رشته اتصال و ذخیره آن در یک متغیر کلاس صفحه می‌باشد. بنابراین هر بخش از صفحه اکنون می‌تواند به آن دسترسی داشته باشد.

```
private string connectionString =
WebConfigurationManager.ConnectionStrings[“Northwind”].ConnectionString;
```

در مرحله بعد یک drop-down list که به کاربر اجازه انتخاب یک محصول برای ویرایش را می‌دهد ایجاد می‌کنیم.

```
protected void Page_Load(Object sender, EventArgs e)
{
    if (!this.IsPostBack)

    {
        // Define the ADO.NET objects for selecting products from the database.

        string selectSQL = “SELECT ProductName, ProductID FROM Products”;

        SqlConnection con = new SqlConnection(connectionString);

        SqlCommand cmd = new SqlCommand(selectSQL, con);

        // Open the connection.

        con.Open();

        // Define the binding.

        lstProduct.DataSource = cmd.ExecuteReader();

        lstProduct.DataTextField = “ProductName”;

        lstProduct.DataValueField = “ProductID”;

        // Activate the binding.

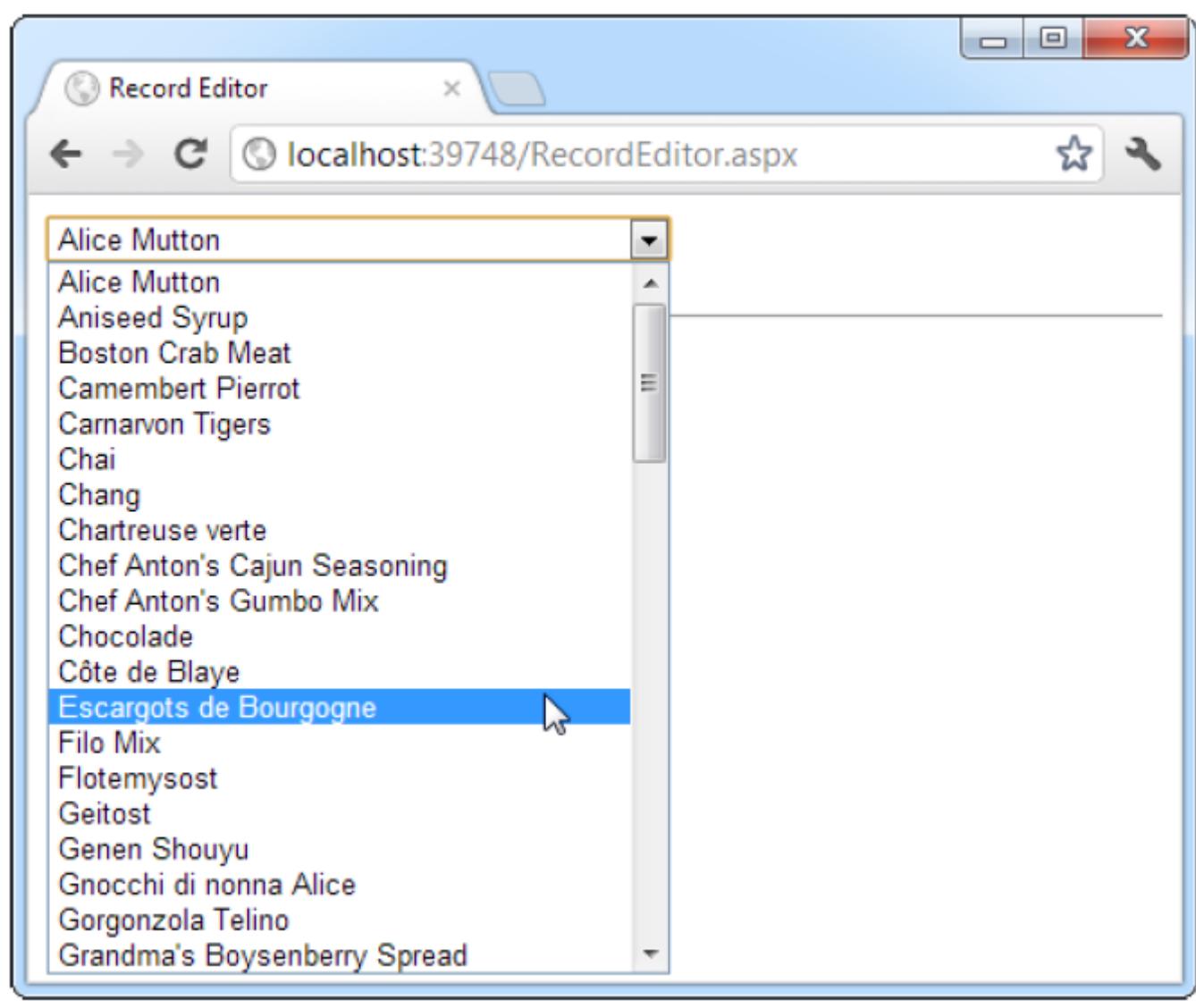
        this.DataBind();

        con.Close();

        // Make sure nothing is currently selected in the list box.

        lstProduct.SelectedIndex = -1;
    }
}
```

لیست تنها دفعه اولی که صفحه لود شود پر می‌شود(و بصورت خودکار در ViewState ذخیره می‌شود). اگر صفحه شود، لیست با همان موارد قبلی باقی می‌ماند. این روش کار بازک اطلاعاتی را کاهش می‌دهد و کار صفحه را سریعتر و موثر تر می‌کند.



SelectedItem ویژگی Drop-down list را فعال کرده است، بنابراین به محض انتخاب اولین محصول، رویداد-Changes (fire) در این مرحله کد شما کارهای زیر را انجام می‌دهد:

- رکوردهای مرتبط را از جدول product می‌خواند و اطلاعات اضافی درباره آن را در Label نمایش می‌دهد.
- لیست کامل گروه بندی محصولات را از جدول Categories خوانده و این اطلاعات را به یک کنترل لیستی دیگر Bind می‌کند.
- بصورت اولیه، این لیست در یک پنل با ویژگی Visible ای که برابر با False می‌باشد، پنهان شده است.
- ردیفی که در گروه مرتبط با محصول انتخابی می‌باشد را پررنگ کرده و انتخاب می‌کند.

رویداد زیر را هم می‌توانید بصورت دستی در code-behind بنویسید و هم آن را توسط VS با رفتن به پنجره Properties و سربرگ Events و انتخاب رویداد مربوطه و سپس کلیک مضاعف روی آن، ایجاد کنید. سپس کدهای خود را درون آن بنویسید.

```

protected void lstProduct_SelectedIndexChanged(object sender, EventArgs e)
{
    // Create a command for selecting the matching product record.

    string selectProduct = "SELECT ProductName, QuantityPerUnit, " +
        "CategoryName FROM Products INNER JOIN Categories ON " +
        "Categories.CategoryID=Products.CategoryID " +
        "WHERE ProductID=@ProductID";

    // Create the Connection and Command objects.

    SqlConnection con = new SqlConnection(connectionString);

    SqlCommand cmdProducts = new SqlCommand(selectProduct, con);

    cmdProducts.Parameters.AddWithValue("@ProductID",
        lstProduct.SelectedItem.Value);

    // Retrieve the information for the selected product.

    using (con)
    {
        con.Open();

        SqlDataReader reader = cmdProducts.ExecuteReader();

        reader.Read();

        // Update the display.

        lblRecordInfo.Text = "<b>Product:</b> " +
            reader["ProductName"] + "<br />";

        lblRecordInfo.Text += "<b>Quantity:</b> " +
            reader["QuantityPerUnit"] + "<br />";

        lblRecordInfo.Text += "<b>Category:</b> " + reader["CategoryName"];

        // Store the corresponding CategoryName for future reference.

        string matchCategory = reader["CategoryName"].ToString();

        // Close the reader.

        reader.Close();

        // Create a new Command for selecting categories.

        string selectCategory = "SELECT CategoryName, " +

```

```

“CategoryID FROM Categories”;

SqlCommand cmdCategories = new SqlCommand(selectCategory, con);

// Retrieve the category information and bind it.

lstCategory.DataSource = cmdCategories.ExecuteReader();

lstCategory.DataTextField = “CategoryName”;

lstCategory.DataValueField = “CategoryID”;

lstCategory.DataBind();

// Highlight the matching category in the list.

lstCategory.Items.FindByText(matchCategory).Selected = true;

}

pnlCategory.Visible = true;

}

```

اگر کاربر یک گروه متفاوت را انتخاب کرد و روی Update کلیک نمود، تغییرات در بانک ذخیره می‌شود.

```

protected void cmdUpdate_Click(object sender, EventArgs e)

{

    // Define the Command.

    string updateCommand = “UPDATE Products “ +

    “SET CategoryID=@CategoryID WHERE ProductID=@ProductID”;

    SqlConnection con = new SqlConnection(connectionString);

    SqlCommand cmd = new SqlCommand(updateCommand, con);

    cmd.Parameters.AddWithValue(“@CategoryID”, lstCategory.SelectedItem.Value);

    cmd.Parameters.AddWithValue(“@ProductID”, lstProduct.SelectedItem.Value);

    // Perform the update.

    using (con)

    {

        con.Open();

        cmd.ExecuteNonQuery();

    }

}

```

# پنجم سوم کار با داده ها

فصل دهم: اصول ADO.NET

فصل یازدهم: Data Binding

فصل دوازدهم: کنترل های داده ای - Data Controls



## Data Controls

### کنترل‌های داده ای

در این بخش با سه کنترل **GridView**, **DetailView**, **FormView** که به شما اجازه **bind** کردن کل جدول داده را می‌دهند، آشنا خواهید شد.

**GridView**: برای نمایش جداول بزرگ داده استفاده می‌شود. این کنترل یکی از سنگین ترین کنترل‌های ASP.NET می‌باشد.

**DetailsView**: برای نمایش یک رکورد در لحظه استفاده می‌شود. به صورت یک جدول که دارای یک رکورد برای هر فیلد می‌باشد نمایش داده می‌شود.

**FormView**: مانند **DetailView**, این کنترل نیز یک رکورد را در هر لحظه نمایش می‌دهد و امكان ویرایش آن را می‌دهد. تفاوت آنها در این است که **FormView** مبتنی بر **template** می‌باشد که امكان ترکیب فیلد‌ها درون یک **layout** منعطف را می‌دهد و نیازی به استفاده از **table** ندارد.

**ListView**: نقشی مانند **GridView** دارد. امكان نمایش چند رکورد را دارد. تفاوت آن با **GridView** این است که مبتنی بر **template** می‌باشد. بنابراین استفاده از آن کار بیشتری می‌برد و به شما انعطاف بیشتری می‌دهد. این کنترل در این کتاب آموزش داده نمی‌شود. می‌توانید این مبحث در کتاب بعدی گروه کلارا آموزش که مباحث پیشرفته و حرفه‌ای ASP.NET را بیان می‌کند خواهد آمد.

## GridView

این کنترل یک جدول چند ستونی می‌باشد. هر رکورد موجود در منبع داده به صورت یک ردیف جداگانه در گردید نمایش داده می‌شود؛ و هر فیلد آن رکورد به صورت یک ستون در گردید نشان داده خواهد شد.

### ایجاد خودکار ستون ها

GV، یک منبع داده برای شی داده ای که می‌خواهد نمایش دهید در اختیار قرار می‌دهد. البته این کنترل از ویژگی‌های DataTextField و DataTextValue پشتیبانی نمی‌کند زیرا در صورتیکه ویژگی AutoGenerateColumns برابر با True باشد، بصورت خودکار برای هر فیلد یک ستون ایجاد می‌کند.

```
<asp:GridView ID = "GridView1" runat = "server" />
```

کدهای زیر را در code-behind بنویسید :

```
protected void Page_Load(object sender, EventArgs e)
{
    // Define the ADO.NET objects.

    string connectionString =
        WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;

    string selectSQL = "SELECT ProductID, ProductName, UnitPrice FROM Products";

    SqlConnection con = new SqlConnection(connectionString);

    SqlCommand cmd = new SqlCommand(selectSQL, con);

    SqlDataAdapter adapter = new SqlDataAdapter(cmd);

    // Fill the DataSet.

    DataSet ds = new DataSet();

    adapter.Fill(ds, "Products");

    // Perform the binding.

    GridView1.DataSource = ds;

    GridView1.DataBind();
}
```

به یاد داشته باشید که برای کد بالا باید یک NorthWind در فایل Web.Config به بانک ConnectionString داشته باشد.

البته نیاز به نوشتن این کدهای دسترسی به داده بصورت دستی ندارید. می‌توانید از SQLDataSource برای تعریف خود

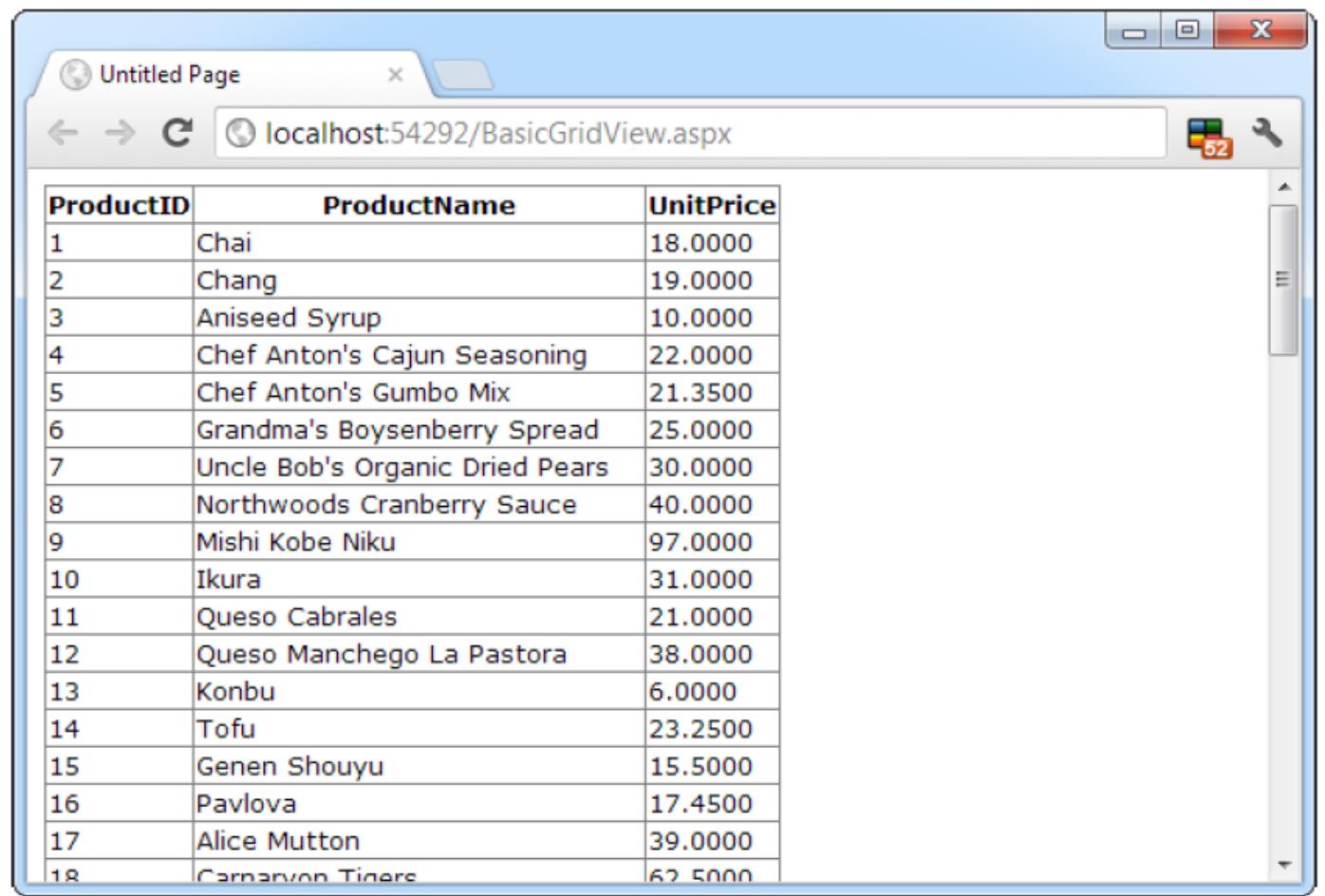
استفاده کنید. در کتابچه تکمیلی این کتاب خواهد آمد و در این کتاب آموزش داده نشده است)

```
<asp:SqlDataSource id="sourceProducts" runat="server"
connectionstring=" <%$ ConnectionStrings:Northwind % > "
selectcommand="SELECT ProductID, ProductName, UnitPrice FROM Products" />

<asp:GridView ID = "GridView1" runat = "server"
DataSourceID = "sourceProducts" />
```

## تعريف ستون ها

بصورت پیش فرض، ویژگی GridView.AutoGenerateColumns می باشد وGridView با True برابر با DataTable ایجاد می کند. این ایجاد ستون های اتوماتیک بیشتر برای ایجاد صفحات تستی مناسب است اما انعطاف پذیری مورد دلخواه شما را به شما نمی دهد. برای نمونه، اگر بخواهید ستونی را نمایش ندهید، ترتیب آن را تغییر دهید یا بعضی از جوانب نمایشی آن را پیکربندی کنید. برای همه این کارها باید ویژگی AutoGenerateColumns را با False برابر قرار دهید و بخش `<Columns>` از GridView را تعریف کنید.



The screenshot shows a web browser window with the title 'Untitled Page'. The address bar displays 'localhost:54292/BasicGridView.aspx'. The main content area contains a 'GridView' control with three columns: 'ProductID', 'ProductName', and 'UnitPrice'. The data is as follows:

ProductID	ProductName	UnitPrice
1	Chai	18.0000
2	Chang	19.0000
3	Aniseed Syrup	10.0000
4	Chef Anton's Cajun Seasoning	22.0000
5	Chef Anton's Gumbo Mix	21.3500
6	Grandma's Boysenberry Spread	25.0000
7	Uncle Bob's Organic Dried Pears	30.0000
8	Northwoods Cranberry Sauce	40.0000
9	Mishi Kobe Niku	97.0000
10	Ikura	31.0000
11	Queso Cabrales	21.0000
12	Queso Manchego La Pastora	38.0000
13	Konbu	6.0000
14	Tofu	23.2500
15	Genen Shouyu	15.5000
16	Pavlova	17.4500
17	Alice Mutton	39.0000
18	Carnarvon Tigers	62.5000



توضیحات	کلاس
این ستون متنی از فیلد موجود در منبع داده را نمایش می‌دهد.	BoundField
یک دکمه را در یک ستون grid نمایش می‌دهد.	ButtonField
این ستون یک checkbox را در ستون grid نمایش می‌دهد.	CheckBoxField
این ستون دکمه‌های انتخابی و ویرایشی را ارائه می‌دهد.	CommandField
این ستون محتوای خود را بصورت یک هایپرلینک نمایش می‌دهد.	HyperLinkFiled
این ستون داده‌های تصویری را از یک فیلد binary می‌خواند و نمایش می‌دهد.	ImageFiled
این ستون به شما اجازه تعیین چندین ستون، کنترل‌های سفارشی (custom controls) و HTML دلخواه‌تان را با استفاده از custom template می‌دهد.	TemplateField

پایه‌ای ترین نوع ستون BoundFiled می‌باشد که به یک فیلد منبع داده متصل می‌شود.

```
<asp:BoundField DataField = "ProductID" HeaderText = "ID" />
```

این تگ چگونگی تغییر متن هدر بالای ستون ProductID را به عبارت ID نشان می‌دهد. در اینجا یک کنترل GridView را به همراه تعریف ستون می‌بینید:

```
<asp:gridview id="GridView1" runat="server" datasourceid="sourceProducts"
autogeneratecolumns="False">

<Columns>

<asp:BoundField DataField = "ProductID" HeaderText = "ID" />
<asp:BoundField DataField = "ProductName" HeaderText = "Product Name" />
<asp:BoundField DataField = "UnitPrice" HeaderText = "Price" />

</Columns>

</asp:gridview>
```

تعریف explicit (واضح) ستون‌ها چند فایده دارد :

- به سادگی می‌توانید ترتیب ستون‌ها، هدر ستون‌ها و سایر جزئیات با استفاده از property‌های شی column تعیین کنید.
- می‌توانید ستون‌هایی را که نمی‌خواهید نمایش دهید را با حذف تگ آن ستون مخفی کنید.
- می‌توانید ستون‌ها را در محیط design ببینید (در VS). با ستون‌های ایجاد شده بصورت خودکار، GridView یک pldacegolder برای ستون‌ها نمایش می‌دهد.
- برای ترکیب ویرایش، انتخاب و سایر کارها می‌توانید ستون‌های بیشتری را اضافه نمایید.

## پیکربندی ستون ها

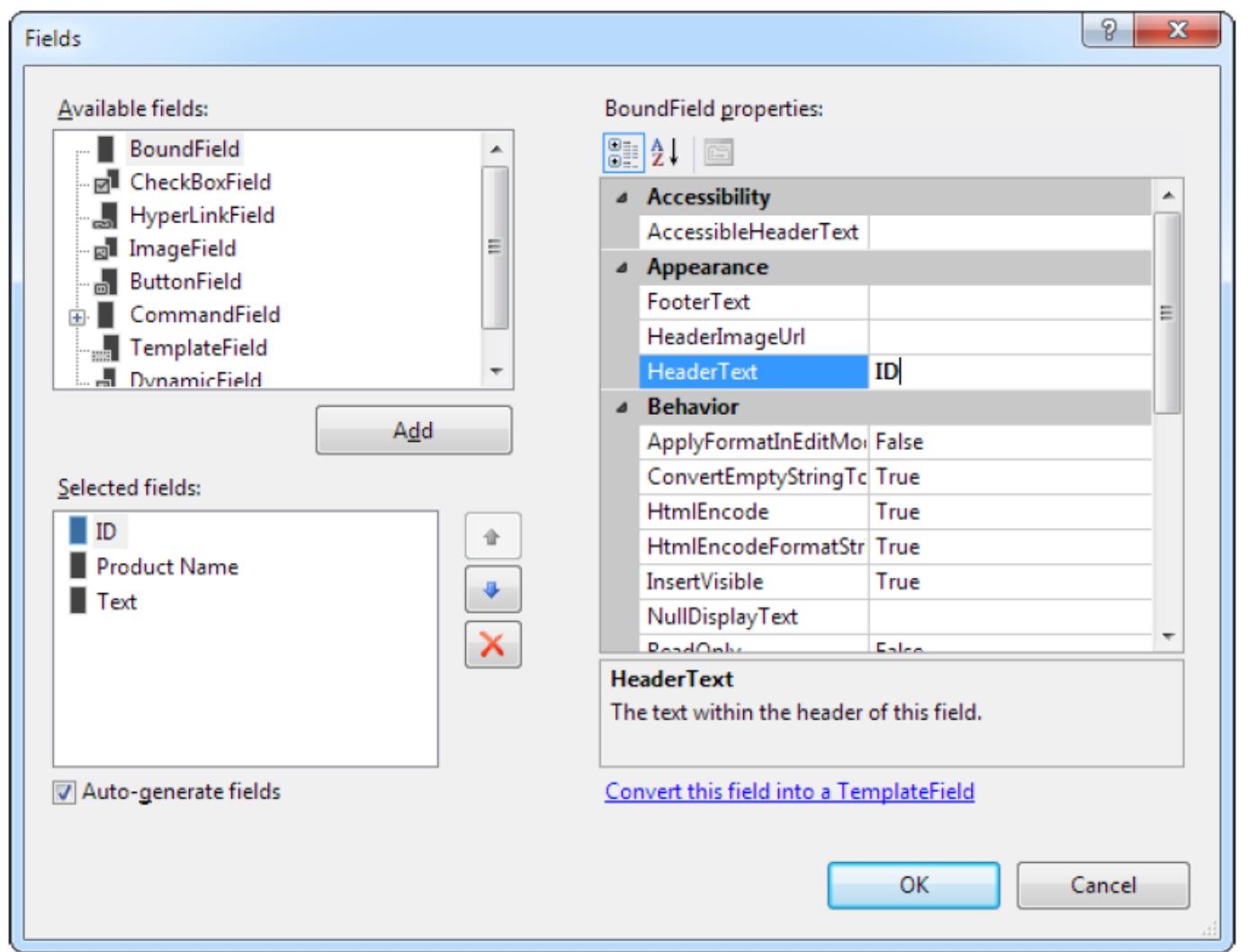


توضیحات	Property
فیلدی که می خواهید در ستون نمایش دهید را تعیین می کند (با نام) فیلد را فرمت دهی می کند. برای نمایش صحیح اعداد و تاریخها بکار می رود.	DataField DataFormatString
اگر true باشد، رشتہ DataFormat برای فرمت دهی مقدار حتی زمانیکه مقدار در textbox در حالت ویرایش باشد، استفاده می شود. پیش فرض آن false می باشد که این یعنی مقدار اولیه استفاده می شود. (مانند 1,143.02\$ به جای 1143.02)	ApplyFormatInEditMode
اگر grid دارای هدر (GridView.ShowHeader) برابر با true باشد) و فوتر (GridView.ShowFooter) برابر با true (باشد، متنی را در هدر و فوتر grid قرار می دهد. هدر برای نمایش یک برچسب توصیفی از محتوای ستون بکار می رود و فوتر می تواند شامل مقدار محاسبه شده داینامیک مانند مجموع (summary) باشد.	,FooterText, HeaderText and HeaderImageUrl
اگر true باشد، از پذیرش مقدار برای آن ستون در حالت ویرایش جلوگیری می شود. هیچ کنترل ویرایشی برای آن ایجاد نمی شود. کلید های اصلی جداول (PK) معمولاً read-only می باشند.	ReadOnly
اگر true باشد، از پذیرش مقدار برای آن ستون در حالت ایجاد (insert)، جلوگیری می شود. اگر می خواهید مقدار ستونی با کدنویسی مقدار دهی شود یا بر اساس مقدار پیش فرض بانک اطلاعاتی پر شود، می توانید از این ویژگی استفاده کنید.	InsertVisible
اگر false باشد، ستون visible نمی شود (و هیچ HTML ای برای آن ایجاد نمی شود). این کار به شما راه مناسبی برای مخفی کردن یا نمایش ستونی با کدنویسی را می دهد.	Visible
نتایج شما را بر اساس یک یا چند ستون مرتب می کند.	SortExpression
اگر true باشد(پیش فرض) همه متن ها برای پیشگیری از خراب شدن صفحه توسط کاراکترهای خاص encoded می شوند. بهترست همیشه از HTML Encoding برای همه مقادیر استفاده کنید.	HTMLEncode
متنی را برای مقادیر Null نشان می دهد. پیش فرض آن یک متن empty می باشد که می توانید آن را تغییر دهید.	NullDisplayText
اگر true باشد، تمامی رشتہ های empty را به مقدار Null تبدیل می کند و از ویژگی NullDisplayText برای نمایش آنها استفاده می کند.	ConvertEmptyString- ToNull
ظاهر نمایش این ستون را پیکربندی و استایل های ردیف را بازنویسی می کند.	ControlStyle, Header-Style, FooterStyle, and Item-Style

## ایجاد ستون ها با استفاده از Visual Studio

می توانید از ستون ها به راحتی استفاده کنید در حالیکه VS، تگ آنها را برای شما ایجاد کرده است. برای این کار، کنترل GridView را انتخاب و در بخش smart tag RefreshSchema روی tag کلیک کنید. اطلاعات نما (schema) را از درون منبع داده بازیابی کرده (برای نمونه، عنوانین و انواع داده ای هر ستون را) و سپس یک المان <BoundField> به ازای هر فیلد اضافه می کند.

هنگامی که ستون ها را ایجاد کردید، می توانید از بعضی از امکانات design-time برای پیکربندی ویژگی های هر ستون استفاده کنید. برای این کار GridView را انتخاب و روی (...) کلیک کنید. می بینید که یک کادر برای اضافه، حذف و تغییر ستون ها باز می شود.



## 弗مت دهی کردن GridView

برای فرمت دهی کردن ابتدا باید مطمئن شوید که داده ها، مقادیر اعدادی و ... به درستی ارائه شده اند. این کار را با ویژگی DataFormatString انجام می دهیم. سپس باید ترکیبی از رنگ ها، فونت ها، border ها و گزینه های alignment (چیدمان) را به هر بخش از grid از header تا footer GridView اضافه کنید. این ویژگی ها را از طریق استایل ها پشتیبانی می کند. در نهایت، می توانید همه اینکارها را از طریق کد نویسی انجام دهید.

## فرمت دهی کردن فیلد ها

هر ستون BoundField را ارائه می دهد که می تواند برای پیکربندی ظاهر اعداد و تاریخ ها استفاده شود. رشته فرمت دهی معمولاً شامل Placeholder و یک تعیین کننده فرمت که درون یک {} قرار گرفته است می شود.

{0:C}

در اینجا، مقداری است که فرمت دهی می‌شود و C یعنی currency format که یک عدد را به یک مقدار پولی تبدیل می‌کند.

3,400.34\$ به 3400.34 تبدیل فرمت می‌دهد)

```
<asp:BoundField DataField = "UnitPrice" HeaderText = "Price"
DataFormatString = "{0:C}" />
```

### رشته های فرمت دهی عددی



مثال	رشته فرمت	Type
	{C:0}	Currency
1.234500E+003	{E:0}	Scientific (Exponential) – نماد علمی
45.6%	{P:0}	Percentage – درصدی
بر اساس تعداد محلی که برای دسیمال آن در نظر بگیرید برای می‌تواند ۱۲۳.۴۰۰ یا (۱۲۳)۰۰۰ باشد.	{?F:0}	Fixed Decimal

برای یافتن سایر نمونه های رشته های فرمت دهی می‌توانید MSDN Help را مشاهده کنید.

```
<asp:BoundField DataField = "BirthDate" HeaderText = "Birth Date"
DataFormatString = "{0:MM/dd/yy}" />
```

### فرمت دهی تاریخ و زمان



مثال	سینتکس	رشته فرمت دهی	Type
10/30/2012	M/d/yyyy	{0:d}	Short Date
Monday, January	dddd, MMMM dd, yyyy	{0:D}	Long Date
2012	dddd, MMMM dd, yyyy	{0:f}	Long Date and
Monday, January	HH:mm aa	{0:F}	Short Time
2012 10:00 AM	dddd, MMMM dd, yyyy	{0:s}	Long Date and
Monday, January	HH:mm:ss aa	{0:M}	Long Time
2012 10:00:23 AM	yyyy-MM- ddTHH:mm:ss	{0:G}	ISO Sortable Standard
10/30/2012	MMMM dd	{0:M}	Month and Day
10:00:23 AM	M/d/yyyy HH:mm:ss aa (depends on locale-specific set- tings)	{0:G}	General

## استفاده از استایل ها

مدل های فرمت دهی قدرتمندی را بر اساس استایل ها ارائه می دهد.



استایل	توضیحات
HeaderStyle	ظاهر ردیف header که شامل عنوان ستون می باشد را پیکربندی می کند.
RowStyle	ظاهر هر ردیف داده ای را پیکربندی می کند.
AlternatingRowStyle	اگر مقدار دهی شود، فرمت دهی اضافی را برای هر ردیف دیگر بکار می گیرد. این فرمت دهی علاوه بر فرمت دهی RowStyle خواهد بود. برای نمونه اگر فونتی را با استفاده از RowStyle تعیین کرده باشید، آن فونت در ردیف های متناوب نیز بکار گرفته می شود، مگر اینکه شما بصورت واضح از طریق ویژگی AlternatingRowStyle یک فونت متفاوت برای آن تعیین کرده باشید.
SelectedRowStyle	ظاهر ردیف انتخاب شده را تعیین می کند. ظاهر ردیف را در حالت ویرایش تعیین می کند. این فرمت دهی علاوه بر فرمت RowStyle عمل می کند.
EditRowStyle	ظاهر ردیف را در حالت ویرایش تعیین می کند. این فرمت دهی علاوه بر فرمت RowStyle عمل می کند.
EmptyDataRowStyle	استایلی که برای یک ردیف خالی استفاده می شود را مشخص خواهد کرد.
FooterStyle	ظاهر ردیف footer در انتهای grid را مشخص می کند. (اگر خواسته باشید که آن را نمایش دهید)
PagerStyle	در صورتیکه ویژگی AllowPaging را برابر با true قرار داده باشید، ظاهر ردیفی که شامل لینک های صفحات grid می باشد را مشخص می کند.

```

<asp:gridview id="GridView1" runat="server" datasourceid="sourceProducts"
autogeneratecolumns="False">

    <RowStyle BackColor = "#E7E7FF" ForeColor = "#4A3C8C" />
    <HeaderStyle BackColor = "#4A3C8C" Font-Bold = "True" ForeColor = "#F7F7F7" />
    <Columns>
        <asp:BoundField DataField = "ProductID" HeaderText = "ID" />
        <asp:BoundField DataField = "ProductName" HeaderText = "Product Name" />
        <asp:BoundField DataField = "UnitPrice" HeaderText = "Price" />
    </Columns>
</asp:gridview>

```

در مثال بالا همه ردیف‌ها از یک استایل استفاده کردند. اگر بخواهید هر ستون بصورت مجزا دارای استایل خاصی باشد، می‌توانید استایل‌های ویژه‌ای برای آنها تعریف کنید.

```
<asp:gridview id="GridView2" runat="server" datasourceid="sourceProducts"
autogeneratecolumns="False">

<Columns>

    <asp:BoundField DataField = "ProductID" HeaderText = "ID" />

    <asp:BoundField DataField = "ProductName" HeaderText = "Product Name">
        <ItemStyle BackColor = "#E7E7FF" ForeColor = "#4A3C8C" />
        <HeaderStyle BackColor = "#4A3C8C" Font-Bold = "True" ForeColor = "#F7F7F7" />
    </asp:BoundField>

    <asp:BoundField DataField = "UnitPrice" HeaderText = "Price" />

</Columns>

</asp:gridview>
```

در تصویر زیر تفاوت این دو کد را خواهید دید:

Global style settings:		
ID	Product Name	Price
1	Chai	18.0000
2	Chang	19.0000
3	Aniseed Syrup	10.0000
4	Chef Anton's Cajun Seasoning	22.0000
5	Chef Anton's Gumbo Mix	21.3500
6	Grandma's Boysenberry Spread	25.0000
7	Uncle Bob's Organic Dried Pears	30.0000
8	Northwoods Cranberry Sauce	40.0000
9	Mishi Kobe Niku	97.0000
10	Ikura	31.0000
11	Queso Cabrales	21.0000
12	Queso Manchego La Pastora	38.0000
13	Konbu	6.0000
14	Tofu	23.2500
15	Genen Shouyu	15.5000
16	Pavlova	17.4500
17	Alice Mutton	39.0000
18	Carnarvon Tigers	62.5000
19	Teatime Chocolate Biscuits	9.2000
20	Sir Rodney's Marmalade	81.0000
21	Sir Rodney's Scones	10.0000

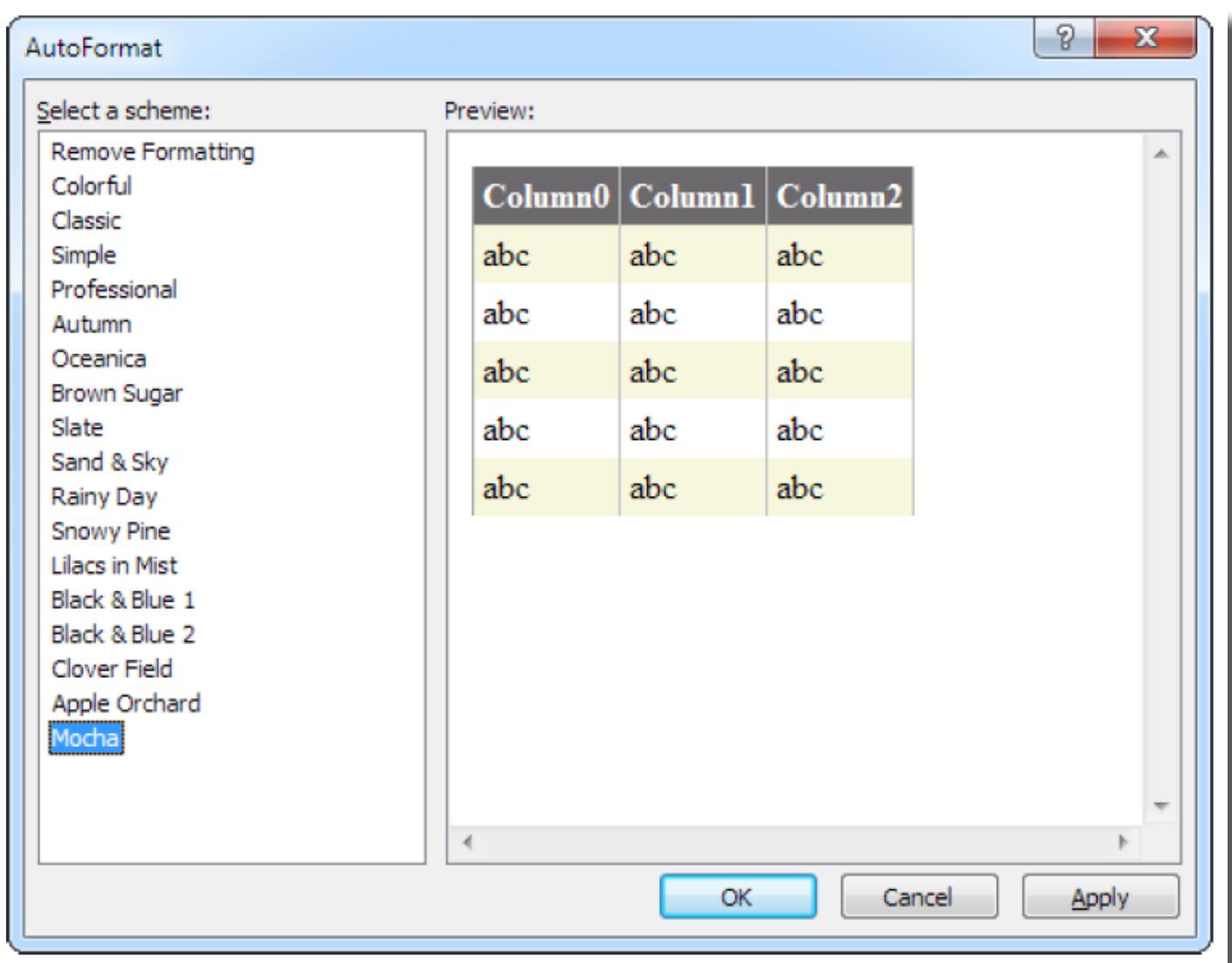
Column-specific styles:		
ID	Product Name	Price
1	Chai	18.0000
2	Chang	19.0000
3	Aniseed Syrup	10.0000
4	Chef Anton's Cajun Seasoning	22.0000
5	Chef Anton's Gumbo Mix	21.3500
6	Grandma's Boysenberry Spread	25.0000
7	Uncle Bob's Organic Dried Pears	30.0000
8	Northwoods Cranberry Sauce	40.0000
9	Mishi Kobe Niku	97.0000
10	Ikura	31.0000
11	Queso Cabrales	21.0000
12	Queso Manchego La Pastora	38.0000
13	Konbu	6.0000
14	Tofu	23.2500
15	Genen Shouyu	15.5000
16	Pavlova	17.4500
17	Alice Mutton	39.0000
18	Carnarvon Tigers	62.5000
19	Teatime Chocolate Biscuits	9.2000
20	Sir Rodney's Marmalade	81.0000
21	Sir Rodney's Scones	10.0000

## پیکربندی استایل ها با استفاده از Visual Studio

دلیلی برای نوشتن استایل ها بصورت دستی وجود ندارد، زیرا GridView این کار را با استفاده از ابزار قدرتمندی در زمان طراحی برای شما انجام می‌دهد. می‌توانید از پنجره Properties برای تغییر ویژگی استایل استفاده کنید.

همچنین می‌توانید ترکیبی از استایل ها را با استفاده از theme های از پیش تعریف شده، با کلیک روی لینک Auto Format در بخش smart tag موجود در GridView، بکار ببرید.

برای حذف کلیه استایل ها، گزینه Remove Formatting را انتخاب کنید.



### فرمت دهی مقادیر خاص

اگر بخواهید فرمت دهی ردیف خاص یا حتی یک ستون تنها را تغییر دهید چه می‌کنید؟

راه حل استفاده از رویداد GridView.RowDataBound می‌باشد. این رویداد برای هر ردیف درست بعد از پر شدن آن با داده (رخ می‌دهد) می‌شود. در این لحظه، شما می‌توانید به ردیف کنونی به عنوان یک شی GridViewRow دسترسی داشته باشید.

ویژگی GridViewRow.Cells، شی داده ای ردیف مربوطه را ارائه می‌دهد و GridViewRow.DataItem، اجازه بازیابی محتوای

```

protected void GridView1_RowDataBound(object sender, GridViewEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        // Get the price for this row.

        decimal price = (decimal) DataBinder.Eval(e.Row.DataItem, "UnitPrice");

        if (price > 50)
        {
            e.Row.BackColor = System.Drawing.Color.Maroon;
            e.Row.ForeColor = System.Drawing.Color.White;
            e.Row.Font.Bold = true;
        }
    }
}

```

ابتدا کد ما چک می‌کند که آیا آیتم ایجاد شده یک ردیف (row) می‌باشد یا نه. اگر نبود که یعنی این آیتم یک آیتم اینترفیس دیگر مانند header , footer ، pager می‌باشد.

برای بازیابی مقدار از شی داده ای bind قابل دسترسی می‌باشد،GridViewRowEventArgs.Row.DataItem (که توسط bind شده است) را به نوع صحیح cast کنید. اگر می‌خواهید کدی را برای این جزئیات ننویسید، تا کدهای دسترسی به داده شما سخت نشود، می‌توانید از متدهای DataBinder.Eval() با نام helpr استفاده کنید.

Untitled Page

localhost:54292/FormatHighPrices.aspx

ID	Product Name	Price
1	Chai	\$18.00
2	Chang	\$19.00
3	Aniseed Syrup	\$10.00
4	Chef Anton's Cajun Seasoning	\$22.00
5	Chef Anton's Gumbo Mix	\$21.35
6	Grandma's Boysenberry Spread	\$25.00
7	Uncle Bob's Organic Dried Pears	\$30.00
8	Northwoods Cranberry Sauce	\$40.00
<b>9</b>	<b>Mishi Kobe Niku</b>	<b>\$97.00</b>
10	Ikura	\$31.00
11	Queso Cabrales	\$21.00
12	Queso Manchego La Pastora	\$38.00
13	Konbu	\$6.00
14	Tofu	\$23.25
15	Genen Shouyu	\$15.50
16	Pavlova	\$17.45
17	Alice Mutton	\$39.00
<b>18</b>	<b>Carnarvon Tigers</b>	<b>\$62.50</b>
19	Teatime Chocolate Biscuits	\$9.20
<b>20</b>	<b>Sir Rodney's Marmalade</b>	<b>\$81.00</b>
21	Sir Rodney's Scones	\$10.00
22	Gustaf's Knäckebröd	\$21.00
23	Tunnbröd	\$9.00
24	Guaraná Fantástica	\$4.50
25	NuNuCa Nuß-Nougat-Creme	\$14.00

## Select اضافه نمودن دکمه

پشتیبانی از selection در GridView قرار داده شده است. فقط کافی است یک ستون CommandField با ویژگی ShowSelectButton برابر با True اضافه کنید. این ستون را به عنوان یک button، hyperlink یا تصویر ثابت رندر می‌کند. می‌توانید از طریق ویژگی SelectText، متنی را برای آن تعیین کنید یا لینکی به یک تصویر را از طریق ویژگی SelectImageUrl برای آن تعیین کنید.

```
<asp:CommandField ShowSelectButton = "True" ButtonType = "Button"
SelectText = "Select" />
```

در زیر نمونه ای وجود دارد که یک آیکن کوچک قابل کلیک را ارائه می‌دهد:

```
<asp:CommandField ShowSelectButton = "True" ButtonType = "Image"
SelectImageUrl = "select.gif" />
```



	ProductID	ProductName	UnitPrice
<a href="#">Select</a>	1	Chai	18.0000
<a href="#">Select</a>	2	Chang	19.0000
<a href="#">Select</a>	3	Aniseed Syrup	10.0000
<a href="#">Select</a>	4	Chef Anton's Cajun Seasoning	22.0000
<a href="#">Select</a>	5	Chef Anton's Gumbo Mix	21.3500
<a href="#">Select</a>	6	Grandma's Boysenberry Spread	25.0000
<a href="#">Select</a>	7	Uncle Bob's Organic Dried Pears	30.0000
<a href="#">Select</a>	8	Northwoods Cranberry Sauce	40.0000
<a href="#">Select</a>	9	Mishi Kobe Niku	97.0000
<a href="#">Select</a>	10	Ikura	31.0000
<a href="#">Select</a>	11	Queso Cabrales	21.0000
<a href="#">Select</a>	12	Queso Manchego La Pastora	38.0000
<a href="#">Select</a>	13	Konbu	6.0000
<a href="#">Select</a>	14	Tofu	23.2500
<a href="#">Select</a>	15	Genen Shouyu	15.5000
<a href="#">Select</a>	16	Pavlova	17.4500
<a href="#">Select</a>	17	Alice Mutton	39.0000
<a href="#">Select</a>	18	Carnarvon Tigers	62.5000
<a href="#">Select</a>	19	Teatime Chocolate Biscuits	9.2000

زمانی که روی یک دکمه select کلیک می‌کنید، صفحه postback شده و مجموعه ای از مراحل انجام می‌شود. ابتدا رویداد GridView.SelectedIndexChanged رخ می‌دهد که می‌توانید آن را کنسل کنید. سپسGridView به محلی که ردیف انتخاب شده است اشاره می‌کند. در نهایت رویداد GridView.SelectedIndexChanged می‌دهد که اگر می‌خواهید سایر کنترل‌ها را بر اساس انتخاب این ردیف از grid تغییر دهید، می‌توانید از آن استفاده کنید. زمانیکه صفحه رندر شد، ردیف انتخاب شده، استایل مورد نظر را می‌گیرد (selected row style).

## استفاده از Select به عنوان دکمه Data Field

نیازی به ایجاد ستون جدید برای پشتیبانی از row selection (انتخاب ردیف) نمی‌باشد. می‌توانید یکی از ستون‌های موجود را به link تبدیل کنید. این تکنیک به کاربران اجازه انتخاب ردیف‌ها در یک جدول را بر اساس مقدار ID می‌دهد. برای استفاده از این تکنیک، ستون CommandField را حذف و یک DataTextField اضافه کنید. سپس ButtonFiled را با نام ستون مورد نظر مقدار دهی کنید.

```
<asp:ButtonField ButtonType = "Button" DataTextField = "ProductID" />
```

این فیلد زمانی که کلیک شود صفحه را postbck GridView.RowCommnd کرده و رویداد Select برای CommandName می‌توانید باشد . با تعیین مقدار CommandName می‌توانید باعث رخداد SelectedIndexChanged رویداد شوید.

```
<asp:ButtonField CommandName = "Select" ButtonType = "Button"  
DataTextField = "ProductID" />
```

## استفاده از Selection برای ایجاد صفحات Master-Detail

برای ایجاد صفحات master-detail که به شما اجازه هدایت روابط در بانک اطلاعاتی را می‌دهد می‌توانید از روش بیان شده در این بخش استفاده کنید. برای این کار نیاز به دو GridView دارید. اولین GridView جدول اصلی یا پدر و در حقیقت همان master ما می‌باشد. زمانی که یک کاربر یک آیتم را در GV اول انتخاب می‌کند، GridView دوم با رکوردهای جدول جزئیات مرتبط با جدول اولی پر می‌شود.

برای نمونه می‌توانید رابطه بین مشتری و لیست سفارشاتش را در نظر بگیرید.

برای ایجاد یک صفحه master-detail، می‌توانید ویژگی SelectedIndex را از GridView اول بازیابی و از آن در query روی GridView دوم استفاده کنید. این روش یک مشکل دارد زیرا ایندکس‌ها عددی هستند که محل رکورد در grid را نمایش می‌دهند و این برای query شما به منظور بازیابی رکورد مرتبط مناسب نمی‌باشد. بنابراین باید یک فیلد unique از رکورد مربوطه داشته باشید. برای نمونه اگر جدول products داشته باشید، باید قادر به گرفتن ProductID از رکورد انتخاب شده باشید. برای گرفتن این اطلاعات، باید به GridView بگویید تا مقادیر فیلد کلید را نگهداری کند.

برای این کار باید ویژگی DataKeyNames را مقدار دهی کنید. اگر چند فیلد کلید دارید می‌توانید آنها را با علامت (,) جدا کنید.

```

<asp:GridView ID = "gridCategories" runat = "server"
DataKeyNames = "CategoryID" . . . >

    Categories:<br />

<asp:gridview id="gridCategories" runat="server" datasourceid="sourceCategories"
    datakeynames="CategoryID">

        <Columns>
            <asp:CommandField ShowSelectButton = "True" />
        </Columns>
        <SelectedRowStyle BackColor = "#FFCC66" Font-Bold = "True"
    ForeColor = "#663399" />
    </asp:gridview>
    <asp:sqldatasource id="sourceCategories" runat="server" connectionstring="< %$"
    ConnectionStrings:Northwind % > "
        selectcommand="SELECT * FROM Categories" > </asp:sqldatasource>
    <br />
    Products in this category:<br />

    <asp:gridview id="gridProducts" runat="server" datasourceid="sourceProducts">
        <SelectedRowStyle BackColor = "#FFCC66" Font-Bold = "True" ForeColor = "#663399" />
    </asp:gridview>
    <asp:sqldatasource id="sourceProducts" runat="server" connectionstring="< %$"
    ConnectionStrings:Northwind % > "
        selectcommand="SELECT ProductID, ProductName, UnitPrice FROM Products WHERE
    CategoryID = @CategoryID" >
        <SelectParameters>
            <asp:ControlParameter Name = "CategoryID" ControlID = "gridCategories"
                PropertyName = "SelectedDataKey.Value" />
        </SelectParameters>
    </asp:sqldatasource>

```

همانطور که می دانید، شما نیاز به دو منبع داده برای یک GridView دارید. منبع داده دوم از ControlParameter استفاده می کند که آن را به ویژگی SelectedDataKey از GridView اول وصل می کند. تا اکنون شما نیاز به نوشتن هیچ کدی برای هندل کردن رویداد SelectedIndexChanged ندارید.

	<b>CategoryID</b>	<b>CategoryName</b>	<b>Description</b>
<a href="#">Select</a>	1	Beverages	Soft drinks, coffees, teas, beers, and ales
<a href="#">Select</a>	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
<a href="#">Select</a>	3	Confections	Desserts, candies, and sweet breads
<a href="#">Select</a>	4	Dairy Products	Cheeses
<a href="#">Select</a>	5	Grains/Cereals	Breads, crackers, pasta, and cereal
<a href="#">Select</a>	6	<b>Meat/Poultry</b>	<b>Prepared meats</b>
<a href="#">Select</a>	7	Produce	Dried fruit and bean curd
<a href="#">Select</a>	8	Seafood	Seaweed and fish

<b>ProductID</b>	<b>ProductName</b>	<b>UnitPrice</b>
9	Mishi Kobe Niku	97.0000
17	Alice Mutton	39.0000
29	Thüringer Rostbratwurst	123.7900
53	Perth Pasties	32.8000
54	Tourtière	7.4500
55	Pâté chinois	24.0000

## ویرایش با استفاده از GridView

برای تبدیل یک ردیف به حالت ویرایش باید ویژگی EditIndex را مقدار دهی کنید. برای انتخاب رکورد باید از یک ستون CommandField استفاده کرد. ShowSelectButton با ویژگی true شده است، استفاده نمایید. برای اضافه کردن کنترل Edit باید ShowEditButton را با مقدار true دهی کنید.

```
<asp:gridview id="gridProducts" runat="server" datasourceid="sourceProducts"
autogeneratecolumns="False" datakeynames="ProductID">
```

```

<Columns>

    <asp:BoundField DataField = "ProductID" HeaderText = "ID" ReadOnly = "True" />
    <asp:BoundField DataField = "ProductName" HeaderText = "Product Name"/>
    <asp:BoundField DataField = "UnitPrice" HeaderText = "Price" />
    <asp:CommandField ShowEditButton = "True" />
</Columns>

</asp:gridview>

<asp:SqlDataSource id = "sourceProducts" runat = "server"
    ConnectionString = "<%$ ConnectionStrings:Northwind %>" 
    SelectCommand = "SELECT ProductID, ProductName, UnitPrice FROM Products"
    UpdateCommand = "UPDATE Products SET ProductName = @ProductName,
        UnitPrice = @UnitPrice WHERE ProductID = @ProductID" />

```

The screenshot shows a web browser window titled "Untitled Page" with the URL "localhost:54292/GridviewEdit.aspx". The page displays a GridView control containing 17 rows of product data from the Northwind database. The columns are labeled "ID", "Product Name", "Price", and an "Edit" link. The products listed are: Chai, Chang, Aniseed Syrup, Chef Anton's Cajun Seasoning, Chef Anton's Gumbo Mix, Grandma's Boysenberry Spread, Uncle Bob's Organic Dried Pears, Northwoods Cranberry Sauce, Mishi Kobe Niku, Ikura, Queso Cabrales, Queso Manchego La Pastora, Konbu, Tofu, Genen Shouyu, Pavlova, and Alice Mutton.

ID	Product Name	Price	
1	Chai	18.0000	<a href="#">Edit</a>
2	Chang	19.0000	<a href="#">Edit</a>
3	Aniseed Syrup	10.0000	<a href="#">Edit</a>
4	Chef Anton's Cajun Seasoning	22.0000	<a href="#">Edit</a>
5	Chef Anton's Gumbo Mix	21.3500	<a href="#">Edit</a>
6	Grandma's Boysenberry Spread	25.0000	<a href="#">Edit</a>
7	Uncle Bob's Organic Dried Pears	30.0000	<a href="#">Edit</a>
8	Northwoods Cranberry Sauce	40.0000	<a href="#">Edit</a>
9	Mishi Kobe Niku	97.0000	<a href="#">Edit</a>
10	Ikura	31.0000	<a href="#">Edit</a>
11	Queso Cabrales	21.0000	<a href="#">Edit</a>
12	Queso Manchego La Pastora	38.0000	<a href="#">Edit</a>
13	Konbu	6.0000	<a href="#">Edit</a>
14	Tofu	23.2500	<a href="#">Edit</a>
15	Genen Shouyu	15.5000	<a href="#">Edit</a>
16	Pavlova	17.4500	<a href="#">Edit</a>
17	Alice Mutton	39.0000	<a href="#">Edit</a>

زمانیکه کلیک کنید، این لینک به حالت Edit می‌باشد. همه فیلد‌ها به جز فیلد‌های textbox به read-only تبدیل می‌شوند.

ID	Product Name	Price	
1	Chai	18.0000	<a href="#">Edit</a>
2	Chang	19.0000	<a href="#">Edit</a>
3	Aniseed Syrup	10.0000	<a href="#">Update</a> <a href="#">Cancel</a>
4	Chef Anton's Cajun Seasoning	22.0000	<a href="#">Edit</a>
5	Chef Anton's Gumbo Mix	21.3500	<a href="#">Edit</a>
6	Grandma's Boysenberry Spread	25.0000	<a href="#">Edit</a>
7	Uncle Bob's Organic Dried Pears	30.0000	<a href="#">Edit</a>
8	Northwoods Cranberry Sauce	40.0000	<a href="#">Edit</a>
9	Mishi Kobe Niku	97.0000	<a href="#">Edit</a>
10	Ikura	31.0000	<a href="#">Edit</a>
11	Queso Cabrales	21.0000	<a href="#">Edit</a>
12	Queso Manchego La Pastora	38.0000	<a href="#">Edit</a>
13	Konbu	6.0000	<a href="#">Edit</a>
14	Tofu	23.2500	<a href="#">Edit</a>
15	Genen Shouyu	15.5000	<a href="#">Edit</a>
16	Pavlova	17.4500	<a href="#">Edit</a>
	All -	20.0000	<a href="#">Print</a>

لینک Cancel، ردیف را به وضعیت اولیه برمی‌گرداند. لینک Update مقادیر را به SQLDataSource ارسال می‌کند.

## مرتب سازی (Paging) و صفحه بندی (Sorting)

### Sorting

برای فعال سازی sorting، باید ویژگی gridView.AllowSorting را برابر با true قرار دهید. سپس باید برای هر ستونی که می‌تواند شود، SortExpression را وارد نمایید که همان نقش SortExpression sort را دارد.

```
<asp:BoundField DataField = "ProductName" HeaderText = "Product Name"
SortExpression = "ProductName" />
```

<b>ProductID</b>	<b>ProductName</b>	<b>UnitPrice</b>
17	Alice Mutton	39.0000
3	Aniseed Syrup	10.0000
40	Boston Crab Meat	18.4000
60	Camembert Pierrot	34.0000
18	Carnarvon Tigers	62.5000
1	Chai	18.0000
2	Chang	19.0000
39	Chartreuse verte	18.0000
4	Chef Anton's Cajun Seasoning	22.0000
5	Chef Anton's Gumbo Mix	21.3500

1 [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#)

## Paging

اگل جستجو در بانک اطلاعاتی تعداد ردیف های زیادی بر می گرداند. اگر کلاینت از یک اتصال کند، استفاده کند، یک GridView بزرگ، زمان زیادی برای لود شدن صرف خواهد کرد.

GridView یک سیستم Paging خودکار دارد. با استفاده از این سیستم، کل نتیجه query شما بازیابی شده و درون یک DataSet گرفته شود. مورد نظر bind به DataSet گروه های کوچکتر تقسیم می شوند (برای نمونه هر یک با ۲۰ ردیف) و تنها یک گروه به کاربر ارسال می شود. سایر گروه ها زمانی bind می شود که پردازش صفحه تمام شده باشد. زمانی که کاربر به صفحه بعد می رود، همین فرآیند تکرار می شود. (به عبارت دیگر Query یکبار دیگر اجرا می شود). تنها یک گروه از رکوردها را بازیابی می کند و صفحه رندر می شود.

## اعضای GridView موجود در Paging



توضیحات	Property
امکان فعال و غیر فعال کردن Paging را می دهد. بصورت پیش فرض غیر فعال است.	AllowPaging
تعداد آیتم های نمایش داده شده در یک صفحه از grid را تعیین می کند. پیش فرض آن ۱۰ می باشد.	PageSize
ایندکس صفحه نمایش داده شده در grid	PageIndex
گزینه های فرمت دهی برای کنترل Pager را ارائه می دهد که تعیین می کنند آیا کنترل های paging نمایش داده شوند یا نه و چه متن یا تصویری را شامل شوند.	PagerSetting
شی style که شما برای پیکربندی فونت ها، رنگ ها و چیدمان متن استفاده می کنید را ارائه می دهد.	PagerStyle
ررست قبل از تغییر PageIndex (PageIndexChanging) و پس از آن (PageIndexChanged) در زمانی رخ می دهد که یکی از المان های انتخابی صفحه، کلیک شده باشد	PageIndexChanging and PageIndexChanged events

```

<asp:GridView ID = “GridView1” runat = “server” DataSourceID = “sourceProducts”
    PageSize = “10” AllowPaging = “True” . . .
    .
    .
</asp:GridView>

```

The screenshot shows a web browser window titled "Untitled Page" with the URL "localhost:54292/GridviewSortPage.aspx". The page contains a GridView control with the following data:

<b>ProductID</b>	<b>ProductName</b>	<b>UnitPrice</b>
51	Manjimup Dried Apples	53.0000
52	Filo Mix	7.0000
53	Perth Pasties	32.8000
54	Tourtière	7.4500
55	Pâté chinois	24.0000
56	Gnocchi di nonna Alice	38.0000
57	Ravioli Angelo	19.5000
58	Escargots de Bourgogne	13.2500
59	Raclette Courdavault	55.0000
60	Camembert Pierrot	34.0000

Below the table, there is a navigation bar with links labeled 1, 2, 3, 4, 5, 6, 7, 8.

## كارايی Paging

زمانی که از paging استفاده می‌کنید، هر بار که یک صفحه جدید از grid درخواست شود، کل Dataset از بانک درخواست می‌شود. این یعنی لود بانک اطلاعاتی افزایش پیدا می‌کند. می‌توانید Dataset حاوی اطلاعات را در حافظه سرور Cache کنید. بنابراین هر بار که کاربر به یک صفحه مختلف رود، شما به سادگی داده را از حافظه بازیابی کرده و مجددآ آن را bind می‌کنید.

## استفاده از GridView Template

اگر بخواهید چند مقدار را درون یک سلول قرار دهید یا بخواهید محتوای درون سلول را با اضافه نمودن HTML و کنترل‌های سروری سفارشی سازی کنید نیاز به استفاده از TemplateField دارید.

به شما امکان تعریف یک template سفارشی برای یک ستون را می‌دهد. درون آن می‌توانید تگ‌های کنترل، TemplateField عبارات HTML و المان‌های data binding را اضافه کنید.

```

<asp:templatefield headertext="Status">
    <ItemTemplate>
        <b> In Stock:</b>
        <%# Eval("UnitsInStock") %> <br />
        <b> On Order:</b>
        <%# Eval("UnitsOnOrder") %> <br />
        <b> Reorder:</b>
        <%# Eval("ReorderLevel") %>
    </ItemTemplate>
</asp:templatefield>

```

برای ایجاد عبارات data binding، باید از متدهای Eval() درون template استفاده کرد که یک متدهای static از کلاس System.Web.UI.DataBinder می‌باشد. این متدهای خودکار داده‌ای که به ردیف کنونی bind شده است را بازیابی می‌کند. این کار را با استفاده از reflection برای پیدا کردن فیلد منطبق و بازیابی مقدار آن انجام می‌دهد.

البته از این متدهای توانید برای پذیرفتن فرمتهای اضافی روی پارامترها استفاده کنید :

```

<%# Eval("BirthDate", "{0:MM/dd/yy}") %>
<asp:sqlDataSource id="sourceProducts" runat="server"
    connectionstring=" <%$ConnectionString:Northwind %> " selectcommand="SELECT ProductID,
    ProductName, UnitPrice, UnitsInStock,
    UnitsOnOrder, ReorderLevel FROM Products" updatecommand="UPDATE Products SET ProductName =
    @ProductName,
    UnitPrice = @UnitPrice WHERE ProductID = @ProductID">
</asp:sqlDataSource>

```

## انواع TemplateField

اظاهار و محتوای سلول header را مشخص می‌کند	HeaderTemplate
اظاهار و محتوای سلول Footer را مشخص می‌کند	FooterTemplate
اظاهار و محتوای هر سلول داده را مشخص می‌کند	ItemTemplate
اظاهار و محتوای سلول‌های زوج را مشخص می‌کند.	AlternatingItemTemplate
اظاهار و محتوای کنترل‌های موجود در حالت ویرایش را مشخص می‌کند	EditItemTemplate
اظاهار و محتوای کنترل‌های موجود در حالت ویرایش را مشخص می‌کند.GridView این template را پشتیبانی نمی‌کند اما FormView و DetailsView از آن استفاده می‌کنند.	InsertItemTemplate

ID	Product Name	Price	Status
1	Chai	18.0000	In Stock: 39 On Order: 0 Reorder: 10
2	Chang	19.0000	In Stock: 17 On Order: 40 Reorder: 25
3	Aniseed Syrup	10.0000	In Stock: 13 On Order: 70 Reorder: 25
4	Chef Anton's Cajun Seasoning	22.0000	In Stock: 53 On Order: 0 Reorder: 0
5	Chef Anton's Gumbo Mix	21.3500	In Stock: 0 On Order: 0 Reorder: 0
6	Grandma's Boysenberry Spread	25.0000	In Stock: 120 On Order: 0 Reorder: 25
7	Uncle Bob's Organic Dried Pears	30.0000	In Stock: 15 On Order: 0 Reorder: 10

## ویرایش Template ها در Visual Studio

- ۱- یک GridView را با حداقل یک template column ایجاد کنید.
- ۲- انتخاب و روی smarttag Edit Template در GridView کلیک کنید. این کار محتواهای مختلف، کنترل های ثابت، ... را در آن قرار دهد.
- ۳- در smarttag Display از لیست template برای انتخاب مورد نظرتان برای ویرایش استفاده کنید.
- ۴- محتواهای خود را درون کنترل قرار دهید. می توانید محتواهای ثابت، کنترل های مختلف و ... را در آن قرار دهید.
- ۵- زمانی که کارتان تمام شد، از smarttag End Template Editing گزینه گزینه.

```

<asp:templatefield headertext="Status">

    <ItemTemplate>

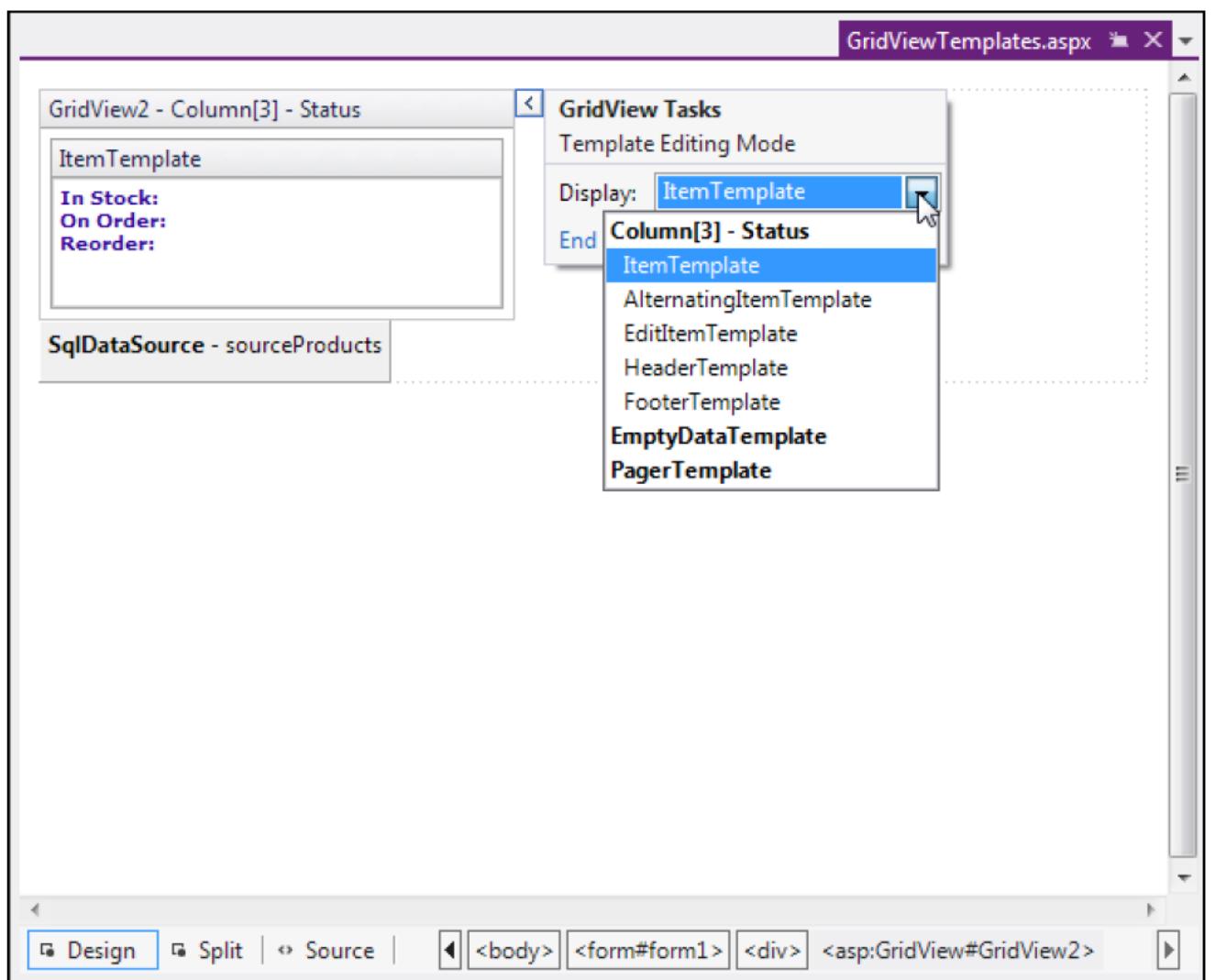
        <asp:ImageButton ID = "ImageButton1" runat = "server"
            ImageUrl = "statuspic.gif"
            CommandName = "StatusClick" CommandArgument = ' <%# Eval("ProductID") %> ' />

    </ItemTemplate>

</asp:templatefield>

protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "StatusClick")
        lblInfo.Text = "You clicked product #" + e.CommandArgument.ToString();
}

```



## ویرایش با template

```

<asp:templatefield headertext="Status">

    <ItemTemplate>
        <b> In Stock:</b> <%# Eval("UnitsInStock") %> <br />
        <b> On Order:</b> <%# Eval("UnitsOnOrder") %> <br />
        <b> Reorder:</b> <%# Eval("ReorderLevel") %>
    </ItemTemplate>

    <EditItemTemplate>
        <b> In Stock:</b> <%# Eval("UnitsInStock") %> <br />
        <b> On Order:</b> <%# Eval("UnitsOnOrder") %> <br /> <br />
        <b> Reorder:</b>
        <asp:TextBox Text = '<%# Bind("ReorderLevel") %>' Width = "25px"
            runat = "server" id = "txtReorder" />
    </EditItemTemplate>
</asp:templatefield>

```

	ID	Product Name	Price	Status
<a href="#">Update</a> <a href="#">Cancel</a>	1	Chai	18.0000	In Stock: 39 On Order: 0 Reorder: 10
<a href="#">Edit</a>	2	Chang	19.0000	In Stock: 17 On Order: 40 Reorder: 25
<a href="#">Edit</a>	3	Aniseed Syrup	10.0000	In Stock: 13 On Order: 70 Reorder: 25
<a href="#">Edit</a>	4	Chef Anton's Cajun Seasoning	22.0000	In Stock: 53 On Order: 0 Reorder: 0
<a href="#">Edit</a>	5	Chef Anton's Gumbo Mix	21.3500	In Stock: 0 On Order: 0 Reorder: 0

زمانی یک مقدار قابل ویرایش را به یک کنترل bind می کنید، باید از متده استفاده کنید. زیرا برخلاف متده Eval() که می توانست در هر جای صفحه قرار بگیرد، متده Bind() باید برای مقدار دهنده property کنترل مورد استفاده قرار گیرد. این متده یک اتصال دو طرفه برقرار می کند که مقادیر ویرایش شده را به سرور بر می گرداند.

```
<asp:sqldatasource id="sourceProducts" runat="server">

    connectionstring=" <%$ ConnectionStrings:Northwind %> " 

        selectcommand="SELECT ProductID, ProductName, UnitPrice, UnitsInStock,
UnitsOnOrder,ReorderLevel FROM Products" updatecommand="UPDATE Products SET ProductName =
@ProductName, UnitPrice = @UnitPrice,
ReorderLevel = @ReorderLevel WHERE ProductID = @ProductID">

</asp:sqldatasource>
```

## ویرایش به همراه تعیین اعتبار

برای اضافه کردن کنترل‌های اعتبار سنجی مانند زیر عمل کنید:

```
<asp:templatefield headertext="Status">

<ItemStyle Width = "100px" />

<ItemTemplate>

    <b> In Stock:</b> <%# Eval("UnitsInStock") %> <br />
    <b> On Order:</b> <%# Eval("UnitsOnOrder") %> <br />
    <b> Reorder:</b> <%# Eval("ReorderLevel") %>

</ItemTemplate>

<EditItemTemplate>

    <b> In Stock:</b> <%# Eval("UnitsInStock") %> <br />
    <b> On Order:</b> <%# Eval("UnitsOnOrder") %> <br /> <br />
    <b> Reorder:</b>
    <asp:TextBox Text = ' <%# Bind("ReorderLevel") %> ' Width = "25px"
runat = "server" id = "txtReorder" />
    <asp:RangeValidator id = "rngValidator" MinimumValue = "0" MaximumValue = "100"
ControlToValidate = "txtReorder" runat = "server"
ErrorMessage = "Value out of range." Type = "Integer"/>
</EditItemTemplate>

</asp:templatefield>
```

	ID	Product Name	Price	Status
<a href="#">Update</a> <a href="#">Cancel</a>	1	Chai	18.0000	<b>In Stock:</b> 39 <b>On Order:</b> 0 <b>Reorder:</b> -5 Value out of range.
<a href="#">Edit</a>	2	Chang	19.0000	<b>In Stock:</b> 17 <b>On Order:</b> 40 <b>Reorder:</b> 25
<a href="#">Edit</a>	3	Aniseed Syrup	10.0000	<b>In Stock:</b> 13 <b>On Order:</b> 70 <b>Reorder:</b> 25
<a href="#">Edit</a>	4	Chef Anton's Cajun Seasoning	22.0000	<b>In Stock:</b> 53 <b>On Order:</b> 0 <b>Reorder:</b> 0
				<b>In Stock:</b> 0

## ویرایش بدون استفاده از Command Column

تا کنون، تمامی نمونه هایی که دیدید از یک CommandFiled که بصورت خودکار کنترل های ویرایشی را ایجاد می کرد استفاده کردیم. اکنون می خواهیم کنترل های ویرایشی خودمان را اضافه کنیم.

```

<itemtemplate>

  <b> In Stock:</b> <%# Eval("UnitsInStock") %> <br />
  <b> On Order:</b> <%# Eval("UnitsOnOrder") %> <br />
  <b> Reorder:</b> <%# Eval("ReorderLevel") %>
  <br /> <br />

  <asp:LinkButton runat = "server" Text = "Edit"
    CommandName = "Edit" ID = "LinkButton1" />

</itemtemplate>
  
```

در حالت ویرایش، باید دو دکمه با CommandName برابر با Update و Cancel اضافه کنیم:

```
<edititemtemplate>

<b> In Stock:</b> <%# Eval("UnitsInStock") %> <br />
<b> On Order:</b> <%# Eval("UnitsOnOrder") %> <br /> <br />
<b> Reorder:</b>

<asp:TextBox Text = '<%# Bind("ReorderLevel") %>' Width = "25px"
runat = "server" id = "txtReorder" />

<br /> <br />
<asp:LinkButton runat = "server" Text = "Update"
CommandName = "Update" ID = "LinkButton1" />
<asp:LinkButton runat = "server" Text = "Cancel"
CommandName = "Cancel" ID = "LinkButton2" CausesValidation = "False" />

</edititemtemplate>
```

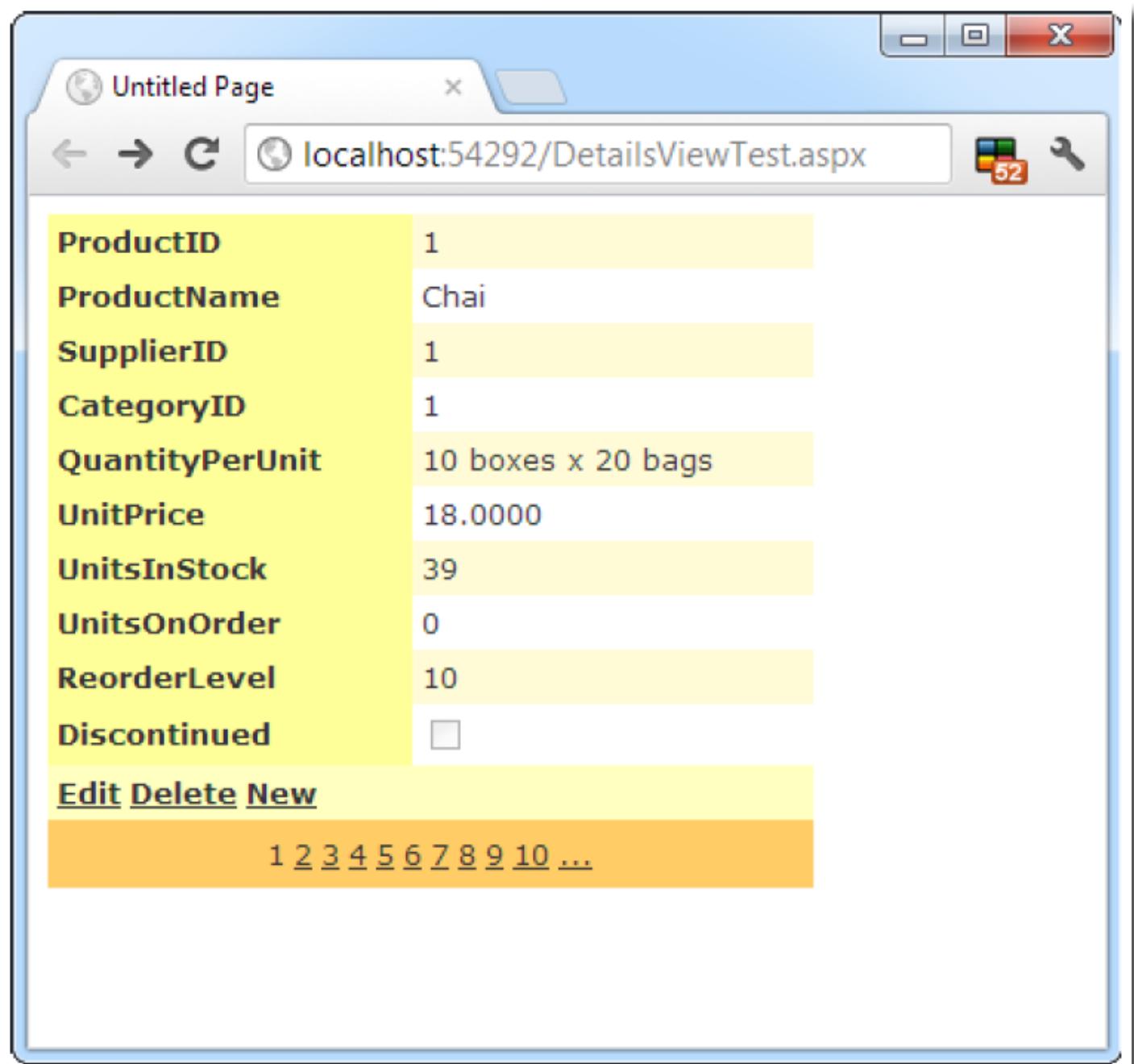
برای جلوگیری از تعیین اعتبار دکمه cancel باید ویژگی CausesValidation آن برابر با false باشد.

ID	Product Name	Price	Status
1	Chai	18.0000	In Stock: 39 On Order: 0  Reorder: 10  <a href="#">Update</a> <a href="#">Cancel</a>
2	Chang	19.0000	In Stock: 17 On Order: 40 Reorder: 25  <a href="#">Edit</a>
3	Aniseed Syrup	10.0000	In Stock: 13 On Order: 70 Reorder: 25

دو کنترل بعدی که معرفی خواهند شد، FormView و DetailsView می‌باشد. بسیاری از متدها و رویدادهای این دو کنترل شبیه GridView می‌باشد.

## DetailsView

این کنترل یک رکورد را در هر زمان نمایش می‌دهد و هر فیلد را در یک ردیف جدا از یک جدول قرار می‌دهد. این کنترل امکان حرکت از یک رکورد به رکورد دیگر را با استفاده از کنترل Paging می‌دهد. در هر بار حرکت به صفحات مختلف DetailsView، کل رکوردها بازیابی می‌شود. برای افزایش کارایی باید از Caching استفاده کنید.



```

<asp:detailsview id="DetailsView1" runat="server" autogeneraterows="False"
datasourceid="sourceProducts">

    <Fields>

        <asp:BoundField DataField = "ProductID" HeaderText = "ProductID"
ReadOnly = "True" />

        <asp:BoundField DataField = "ProductName" HeaderText = "ProductName" />

        <asp:BoundField DataField = "SupplierID" HeaderText = "SupplierID" />

        <asp:BoundField DataField = "CategoryID" HeaderText = "CategoryID" />

        <asp:BoundField DataField = "QuantityPerUnit" HeaderText = "QuantityPerUnit" />

        <asp:BoundField DataField = "UnitPrice" HeaderText = "UnitPrice" />

        <asp:BoundField DataField = "UnitsInStock" HeaderText = "UnitsInStock" />

        <asp:BoundField DataField = "UnitsOnOrder" HeaderText = "UnitsOnOrder" />

        <asp:BoundField DataField = "ReorderLevel" HeaderText = "ReorderLevel" />

        <asp:CheckBoxField DataField = "Discontinued" HeaderText = "Discontinued" />

    </Fields>

</asp:detailsview>

```

The image displays two side-by-side screenshots of a web browser window, both titled "Untitled Page" and showing the URL "localhost:54292/DetailsView".

**Left Screenshot (View Mode):**

- The page displays a table with the following data:

ProductID	1
ProductName	Chai
SupplierID	1
CategoryID	1
QuantityPerUnit	10 boxes x 20 bags
UnitPrice	18.0000
UnitsInStock	39
UnitsOnOrder	0
ReorderLevel	10
Discontinued	<input type="checkbox"/>

- Below the table are links: Edit, Delete, and New.
- A navigation bar at the bottom contains links: 1 2 3 4 5 6 7 8 9 10 ...

**Right Screenshot (Edit Mode):**

- The page displays a table with the same data as the left screenshot.
- Each field in the table has its input type changed to a text input field, allowing for modification.
- Below the table are links: Update and Cancel.
- A navigation bar at the bottom contains links: 1 2 3 4 5 6 7 8 9 10 ...

## FormView

اگر می خواهید انعطاف بیشتری در FormView ها داشته باشد، یک کنترل مبتنی بر template می باشد که برای ویرایش و نمایش یک رکورد بکار می رود.

در این کنترل با template های زیر کار می کنید :

ItemTemplate -

EditItemTemplate -

InsertItemTemplate -

FooterTemplate -

HeaderTemplate -

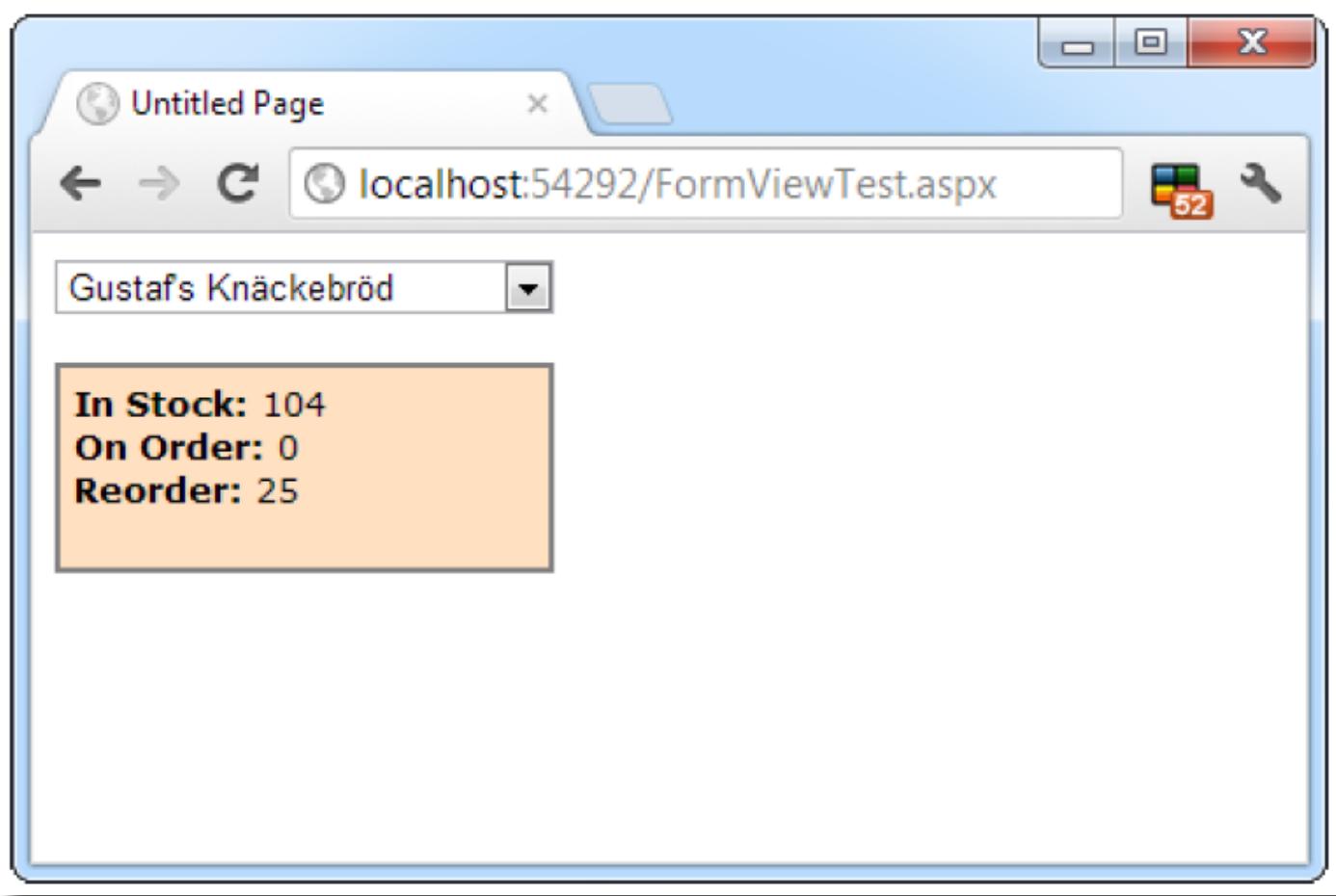
EmptyDataTemplate -

PagerTemplate -

```
<asp:formview id="FormView1" runat="server" datasourceid="sourceProducts">

    <ItemTemplate>
        <b> In Stock:</b>
        <%# Eval("UnitsInStock") %>
        <br />
        <b> On Order:</b>
        <%# Eval("UnitsOnOrder") %>
        <br />
        <b> Reorder:</b>
        <%# Eval("ReorderLevel") %>
        <br />
    </ItemTemplate>

</asp:formview>
```



```

<asp:SqlDataSource ID = “sourceProducts” runat = “server”
    ConnectionString = “ <%$ ConnectionStrings:Northwind % > ”
    SelectCommand = “SELECT ProductID, ProductName FROM Products”>
</asp:SqlDataSource>
<asp:DropDownList ID = “lstProducts” runat = “server”
    AutoPostBack = “True” DataSourceID = “sourceProducts”
    DataTextField = “ProductName” DataValueField = “ProductID” Width = “184px”>
</asp:DropDownList>
```



سەرەتىلەن از دواچىقىسىت، بىزىز تىرىپىن اشتىاهى دەركان،  
اينىسىت نەھەنلىنى خەلقىنەن ھەممىچىز تەختە كەشىلەن تۆستە. «جەن اشتىاهىن بىك»

# بخش چهارم : امنیت و ب سایت

فہل سیزدهم: اصول امنیت

~~فہل چھارم:~~ Membership

~~فہل پانزہم:~~ Profile



تصویرت معمول، وب سایت ASP.NET شما در دسترس هر کسی که به وب سرور شما از طریق local internet یا متعلق می‌شود قرار می‌گیرد.

ASP.NET، یک مدل امنیتی قدرتمند و منعطف به شما ارائه می‌دهد. بیشترین کار در امن کردن برنامه شما، نوشتن کدها نمی‌باشد، بلکه تشخیص محل های مناسب برای پیاده سازی استراتژی امنیتی شما می‌باشد. در این فصل شما دو روش تعیین اعتبار کاربران به نامهای Windows Authentication و Forms Authentication را یاد خواهید گرفت.

## آشنایی با نیازمندی‌های امنیتی

اولین مرحله در امن کردن برنامه شما، تصمیم گیری بر سر این است که کجا نیاز به امنیت دارد و چه چیزی باید مراقبت شود. برای نمونه، شما نیاز به بستن دسترسی به اطلاعات محروم‌خواهید داشت.

## تست و گسترش تنظیمات امنیتی

همانطور که می‌دانید، وب سایت‌های IIS web – hosting از نرم افزار ASP.NET استفاده می‌کند. اما در زمان ساخت و تست برنامه، شما از وب سرور موجود در VS استفاده می‌کنید.

## این تفاوت‌ها از نظر امنیتی دو اثر زیر را دارد:

حساب کاربری وب سایت زمانی که یک وب سایت را در VS، اجرا می‌کنید، وب سرور تست، از حساب کاربری ویندوزی شما استفاده می‌کند. در نتیجه، کد شما امکان اجرای کارهایی (خواندن فایل، نوشتن در log، اتصال به بانک اطلاعاتی و ...) که ممکن است نتوانید در یک وب سرور Live انجام دهید را می‌دهد. برای این که مطمئن شوید همه چیز پس از deploy کردن برنامه تحت وب کار می‌کند، باید اطلاعاتی در باره IIS و ASP.NET که از حساب کاربری ویندوزی استفاده می‌کنند داشته باشید.

پیکربندی امنیتی: شما تنظیمات امنیتی مختلفی را مانند نوع تعیین اعتبار (Authemtication) و قوانین تعیین مجوز (Authorization) کاربرانی که می‌توانند به صفحات مختلف دسترسی داشته باشند، مشخص می‌کنید. در یک سایت تست، می‌توانید این تنظیمات را به دو روش انجام دهید:

ویرایش فایل Web.Config یا اجرای WAT (website administration tool) (ابزار مدیریت سایت)

علاوه بر این، بعضی از تنظیمات امنیتی وجود دارد که پس از deploy کردن وب سایت می‌توانید آنها را انجام دهید. برای نمونه می‌توانید از IIS Manager برای تغییر وضعیت به SSL، که یک استاندارد کدینگ برای پیام‌های ارسالی بین کاربر و وب سرور می‌باشد استفاده کنید.

## Authorization و Authentication

**Authentication :** مراحل تعیین اینکه آیا کاربران معتبر می باشند یا نه را می گویند. معمولاً این کار شامل وارد کردن کلمه عبور و کلمه کاربری درون یک صفحه Login می باشد.

**Authorization :** زمانی که کاربر تعیین اعتبار شد، باید تعیین شود که آیا کاربر مجوز لازم برای اجرای یک کار خاص (مانند مشاهده یک صفحه یا بازیابی اطلاعات از بانک اطلاعاتی) را دارد یا نه.

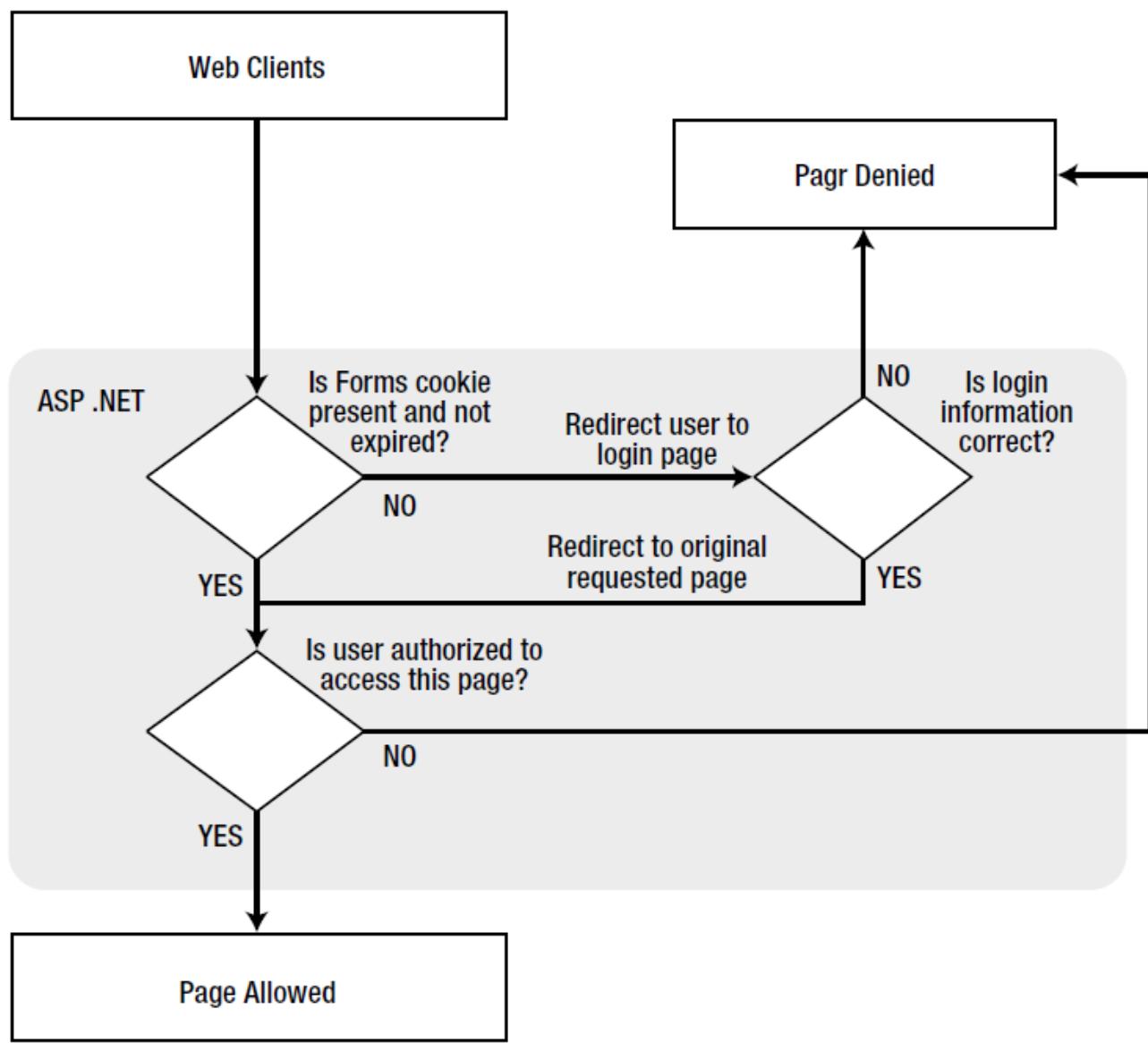
برای امن کردن برنامه یا وب سایت ASP.NET خود، می توانید از دو روش authentication زیر استفاده کنید :

**authorize** : با این روش، ASP.NET، مسئول تعیین اعتبار کاربران می باشد و هر درخواست را می کند. معمولاً این روش در تعامل با بانک اطلاعاتی که اطلاعات کاربر را (مانند کلمه عبور و کلمه کاربری) ذخیره کرده است بکار می رود. حتی می توانید اطلاعات کاربر را در یک فایل متن ساده ذخیره کنید یا کد یک Login سفارشی که با فراخوانی یک remote service کار می کند، را بنویسید.

**Windows Authentication** : با استفاده از این روش، وب سرور هر کاربر را مجبور به login کردن به عنوان یک کاربر ویندوزی می کند (بسته به پیکربندی که استفاده می کنید، این فرآیند login، می تواند خود کار انجام شود یا لازم به وارد کردن کلمه عبور و کاربری در یک کادر login می باشد). این سیستم نیاز دارد تا همه کاربران دارای حساب های کاربری ویندوزی روی سرور باشند.

## Forms Authentication

ASP.NET بصورت خود کار کوکی ها را می سازد و در هر بار login آنها را چک می کند.



برای بکارگیری forms-based security، مراحل زیر را انجام دهید :

- ۱- در فایل web.config، نوع تعیین اعتبار را با forms authentication برابر قرار دهید. (این کار را می‌توانید با ابزار گرافیکی WAT نیز انجام دهید)
- ۲- دسترسی کاربران ناشناس را به صفحه‌ای خاص در برنامه محدود کنید.
- ۳- صفحه Login را ایجاد کنید.

## تنظیمات Web.Config

با استفاده از تگ <authentication> می‌توانید نوع امنیت را در فایل web.config تعیین کنید. تنظیمات مناسب را با استفاده از تگ داخلی <forms> انجام دهید.

```
<configuration>
    ...
<system.web>
    ...
<authentication mode="Forms">
    <forms name="MyAppCookie"
        loginUrl("~/Login.aspx"
        protection="All"
        timeout="30" path="/" />
</authentication>
</system.web>
</configuration>
```



توضیحات	Attribute - خصیصه
نام HTTP cookie که برای اعتبارسنجی استفاده می‌شود. اگر چند برنامه روی یک وب سرور اجرا شوند، باید به کوکی امنیتی مربوط به هر برنامه یک نام واحد اختصاص دهید.	name
صفحه Login شما می‌باشد که اگر هیچ کوکی اعتبارسنجی معتبری پیدا نشود، کاربر به آن صفحه هدایت می‌گردد. مقدار پیش فرض آن Login.aspx می‌باشد.	loginURL
نوع validation و encryption استفاده شده برای کوکی های امنیتی می‌باشد.	protection
تعداد دقیقه هایی که طول می‌کشد تا کوکی expire شود. ASP.NET، هر زمان که یک درخواست را دریافت کند، کوکی را refresh خواهد کرد. مقدار پیش فرض آن ۳۰ می‌باشد.	timeout
مسیر کوکی را مشخص می‌کند. مقدار پیش فرض آن (/) می‌باشد.	path

## فواینین Authorization

اگر تغییرات بیان شده را روی فایل web.config برنامه اعمال کرده و یک صفحه را درخواست کنید، متوجه می‌شوید که هیچ اتفاق خاصی نمی‌افتد. زیرا شما تنها authentication را فعال کرده اید و کاربران ناشناس را محدود نکرده اید.

برای کنترل اینکه چه کسی می‌تواند به وب سایت شما دسترسی داشته باشد، باید فواینین کنترل دسترسی (access control rule) را در بخش <authorization> از فایل web.config تعریف کنید.

```
<configuration>
```

```
  . . .
```

```
<system.web>
```

```
  . . .
```

```
<authentication mode="Forms">
```

```
  <forms loginUrl="~/Login.aspx" />
```

```
</authentication>
```

```
<authorization>
```

```
  <allow users="*" />
```

```
</authorization>
```

```
</system.web>
```

```
</configuration>
```

کاراکتر (\*) یک کاراکتر wildcard می‌باشد که به همه کاربران اجازه استفاده از برنامه را می‌دهد، حتی اگر آنها authenticate نشده باشند. اما اگر این خط را در برنامه قرار ندهید، کماکان برنامه کار می‌کند زیرا پیش فرض ASP.NET اجازه به همه کاربران می‌باشد.

اگر از علامت (?)، استفاده کنید، تعیین کرده اید که کاربران ناشناس نمی‌توانند از برنامه استفاده کنند. هر کاربر باید شود و هر درخواست کاربر نیازمند کوکی امنیتی می‌باشد. اگر یک صفحه موجود در پوشش برنامه را درخواست کنید، ASP.NET تشخیص می‌دهد که درخواست شما authenticate نمی‌باشد و شما را به صفحه Login هدایت می‌کند.

```
<authorization>
```

```
  <deny users="?" />
```

```
</authorization>
```

اکنون بیش از یک قانون را به این بخش اضافه می‌کنیم :

```
<authorization>

<allow users="*" />
<deny users="?" />

</authorization>
```

## کنترل دسترسی به پوشه‌های خاص

بیشتر برنامه‌ها، فایل‌های مورد نیاز برای تعیین اعتبار را در یک پوشه جداگانه قرار می‌دهند. با استفاده از فایل‌های پیکربندی ASP.NET، این کار ساده است. تنظیمات `<authorization>` را در یک فایل Web.Config اضافه و آن را در پوشه مورد نظر قرار دهید:

```
<!-- This web.config file is in a subfolder. -->
```

```
<configuration>

<system.web>

<authorization>
<deny users="?" />
</authorization>

</system.web>

</configuration>
```

 نکته: نمی‌توانید تگ `<authentication>` در فایل web.config از هر زیر پوشه را تغییر دهید. در عوض، تمامی پوشه‌های درون برنامه باید از یک سیستم authentication مشابه استفاده کنند حتی اگر هر پوشه دارای قوانین authorization خودش باشد.

## کنترل دسترسی به فایل‌های خاص

عموماً، تنظیمات مجوز دسترسی فایل بسیار آسان می‌باشد. با اضافه نمودن تگ `<location>` به فایل web.config، می‌توانید دسترسی به فایل‌های خاص را محدود نمایید. تگ location خارج از تگ `<system.web>` می‌باشد و مستقیماً درون تگ `<configuration>` قرار می‌گیرد.

```
<configuration>

. . .

<system.web>

. . .

<authentication mode="Forms">
```

```

<forms loginUrl="~/Login.aspx" />
</authentication>
<authorization>
<allow users="*" />
</authorization>
</system.web>
<location path="SecuredPage.aspx">
<system.web>
<authorization>
<deny users="?" />
</authorization>
</system.web>
</location>
<location path="AnotherSecuredPage.aspx">
<system.web>
<authorization>
<deny users="?" />
</authorization>
</system.web>
</location>
</configuration>

```

در نمونه بالا، تمامی فایل‌های موجود در برنامه، به جز SecredPage.aspx و AnotherSecuredPage.aspx مجاز می‌باشند. توجه کنید که حتی اگر از چند بخش <location> برای پشتیبانی از مجموعه‌ای از قوانین authorization استفاده کنید، شما تنها یک بخش <authentication> دارید.

## کنترل دسترسی برای کاربران خاص

با قوانین <allow> و <deny> می‌توانید یک یا لیستی از کلمات کاربری (جدا شده با کاما) را تعیین کنید. برای نمونه لیست زیر، بصورت خاص دسترسی سه کاربر را محدود می‌کند. این کاربران نمی‌توانند به صفحات موجود در این پوشه دسترسی داشته باشند. سایر کاربران تعیین اعتبار شده (authenticated user)، این امکان را دارند.

## &lt;authorization&gt;

```

<deny users="?" />
<deny users="matthew,sarah" />
<deny users="john" />
<allow users="*" />

```

## &lt;/authorization&gt;

قانون اول در نمونه بالا، تمامی کاربران ناشناس را deny می‌کند. قوانین بعدی به تنها یکی اثری ندارند زیرا کاربران نمی‌توانند خودشان را authenticate کنند.

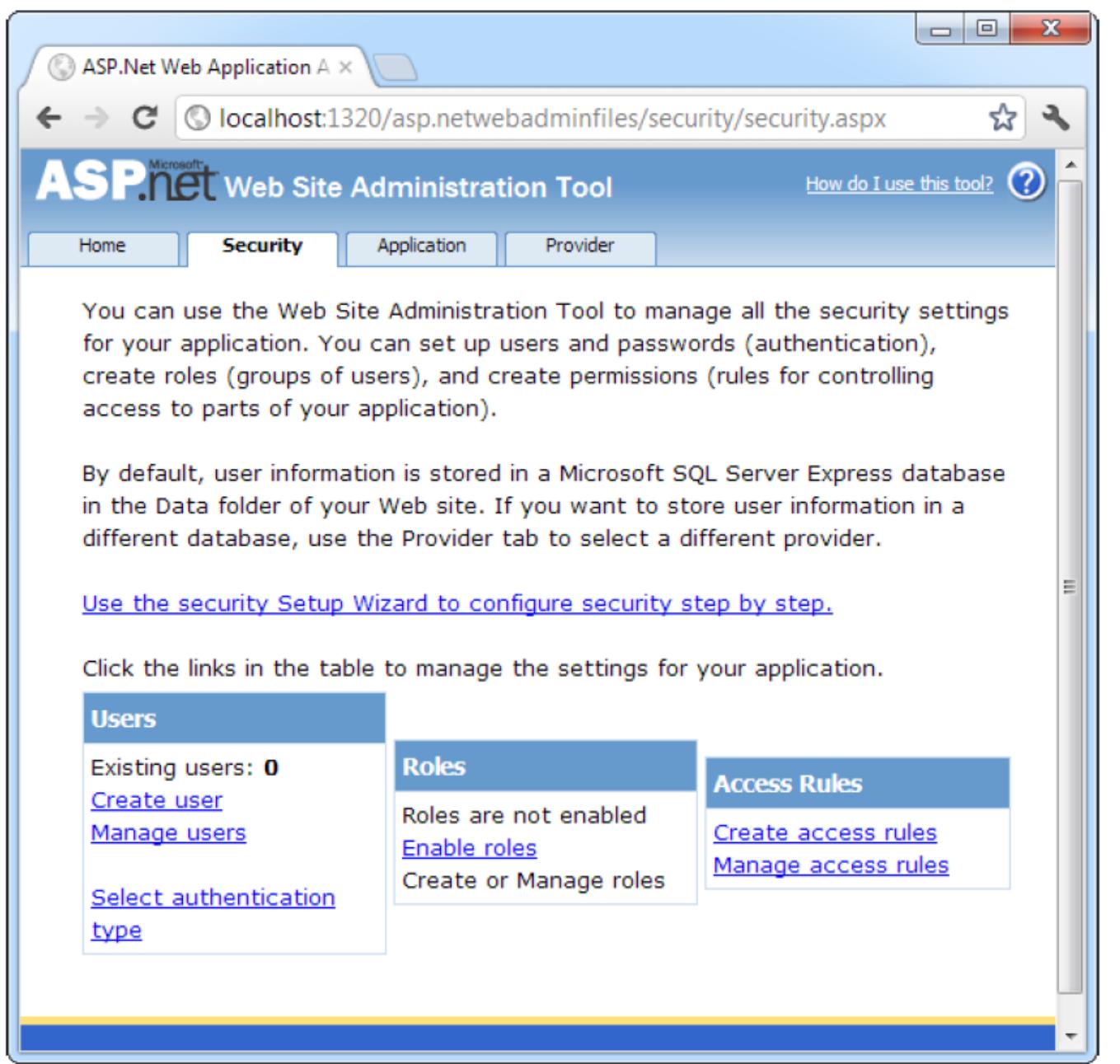
در نمونه زیر، دو کاربر ذکر شده مجاز می‌باشند. دسترسی سایر کاربران حتی اگر authenticate شده باشند، deny می‌باشد.

این کلمات کاربری را با حساب‌های کاربری ویندوز که روی وب سرور شما (دستگاهی که برنامه شما را روی آن قرار می‌گیرد) اشتباه نگیرید. زمانی که از forms authentication استفاده می‌کنید، مدل امنیتی برنامه شما از سیستم حساب کاربری ویندوز مجزا می‌شود. برنامه شما کلمه کاربری شما باید در بانک اطلاعاتی unique باشد.

## WAT

شما راه‌های دیگری برای قوانین authentication و authorization، به جز ویرایش مستقیم فایل web.config دارید. می‌توانید از ابزار WAT درون VS استفاده کنید.

برای استفاده از آن باید Website>ASP.NET Configuration را از منو انتخاب کنید. سپس روی زبانه Security کلیک نمایید. پنجره‌ای برای تعیین authentication type (the Access Rules section) و define authorization rules (enable role-based security)، باز می‌شود.



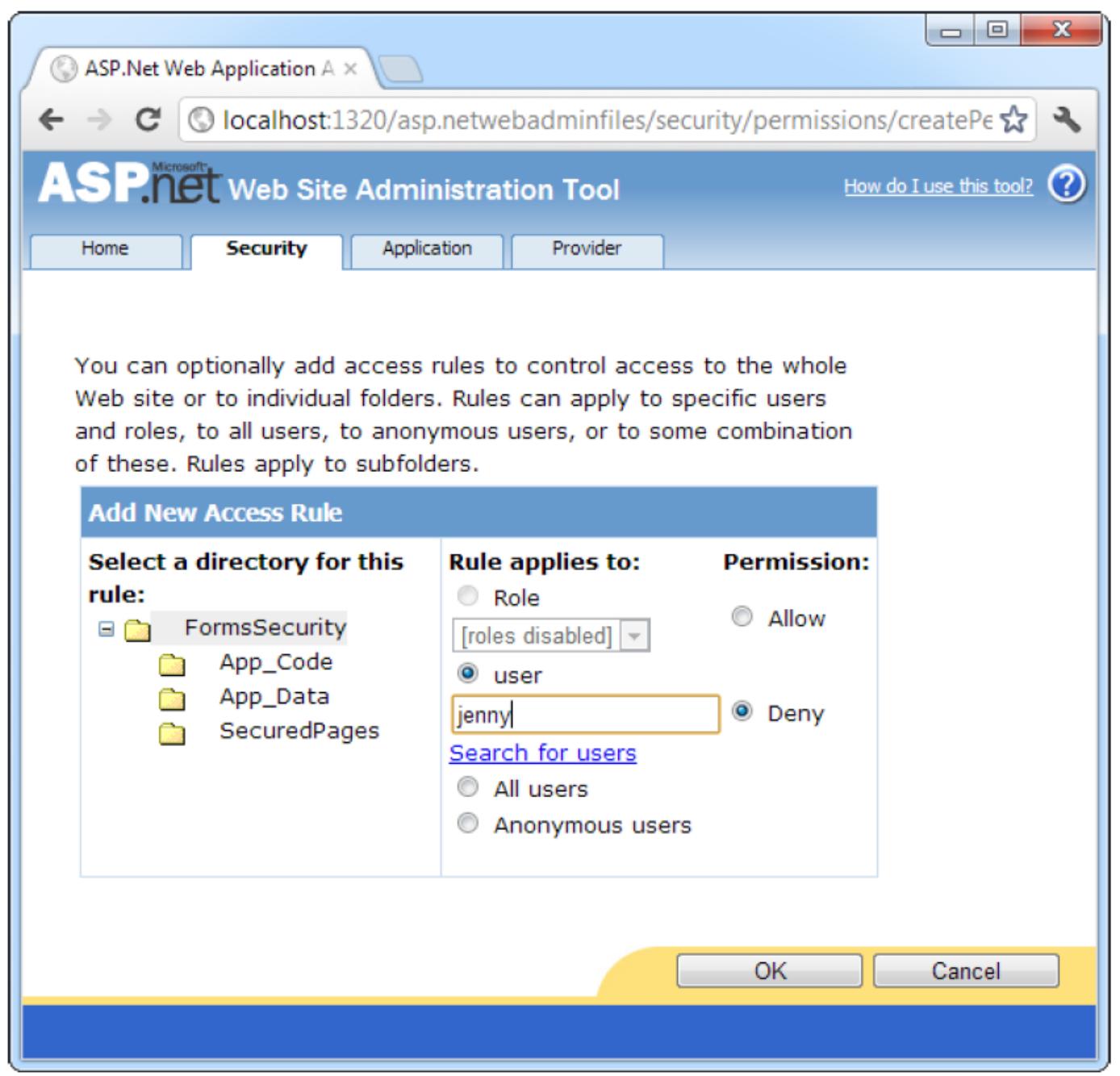
برای تنظیم یک برنامه به منظور استفاده از forms authentication مراحل زیر را انجام دهید :

- ۱- روی Select Authentication Type کلیک کنید.

- ۲- گزینه From a Local Network را انتخاب نمایید. اگر گزینه From the Internet Option را انتخاب کنید، از روش اعتبارسنجی ویندوز استفاده می‌رود.

- ۳- روی Done کلیک کنید. تگ <authorization> در فایل web.config ایجاد می‌شود.

سپس باید قوانین authorization را تعریف کنید. برای این کار روی Create Access Rules کلیک کرده (البته می‌توانید قوانین موجود را نیز با کلیک روی Manage Access Rules تغییر دهید) و قوانین جدیدی برای محدود کردن کاربران خاص از کل وب سایت یا یک صفحه یا زیر پوشه ایجاد کنید.



زبانه Security، دارای بخش های مختلفی است که امکانات زیادی به شما خواهد داد. این جزئیات بخشی از ارکان گسترده ای به نام membership می باشد که بعداً با آن آشنا می شوید.

Use this page to manage access rules for your Web site. Rules are applied in order. The first rule that matches applies, and the permission in each rule overrides the permissions in all following rules. Use the **Move Up** and **Move Down** buttons to change the order of the selected rule.

Rules that appear dimmed are inherited from the parent and cannot be changed at this level.

Permission	Users and Roles	Delete
Deny	[anonymous]	<a href="#">Delete</a>
Deny	jenny	<a href="#">Delete</a>
Allow	[all]	<a href="#">Delete</a>

[Add new access rule](#)

## صفحه Login

هنگامی که روش authentication و قوانین authorization را تعیین کردید، باید یک صفحه Login بسازید که یک صفحه .aspx باشد که اطلاعاتی درباره کاربر را می‌گیرد و تصمیم می‌گیرد که آیا کاربر باید تعیین اعتبار شود یا نه.

یک کلاس مخصوص با نام System.Web.Security در فضای نام FormsAuthentication ارائه داده است که از متدهای static برای کمک به این فرآیند استفاده خواهد کرد.



توضیحات	عضو
یک ویژگی <b>read-only</b> می‌باشد که نام کوکی تعیین اعتبار فرم را ارائه می‌دهد.	FormsCookieName
یک ویژگی <b>read-only</b> می‌باشد که مسیر کوکی تعیین اعتبار فرم را ارائه می‌دهد.	FormsCookiePath
کلمه کاربری و عبور را در لیستی از account ها ای فایل config که می‌توانند وارد شوند چک می‌کند.	Authenticate()
کاربر را پس از ورود موفق به برنامه، به صفحه اصلی مورد درخواستش هدایت می‌کند. یک کوکی برای ثبت کاربر در برنامه ASP.NET ایجاد و آن را به پاسخ کنونی پیوست می‌کند.	RedirectFromLoginPage()
کاربر را از حالت ثبت شده در برنامه خارج کرده و کوکی encrypt شده را حذف می‌کند.	SignOut()
کاربر را با استفاده از ایجاد و پیوست کوکی اعتبارسنجی فرم در برنامه ثبت می‌کند. برخلاف متدهای بالا، این متدها، کاربر را به صفحه مورد درخواست نمی‌فرستند.	SetAuthCookie()
URL، صفحه درخواست شده را بر می‌گرداند. می‌توان از این متدها به همراه متدهای SetAuthCookie() برای ثبت یک کاربر در برنامه و تصمیم گیری درباره هدایت کاربر به صفحه مورد درخواستش یا استفاده از یک صفحه پیش فرض، استفاده کرد.	GetRedirectUrl()
یک کوکی تعیین اعتبار ایجاد می‌کند، اما آن را به پاسخ کنونی پیوست نمی‌کند. می‌توانید کوکی را سفارشی سازی کرده و سپس آن را بصورت دستی به پاسخ ارسالی پیوست کنید.	GetAuthCookie()
متن یک رشته را با استفاده از الگوریتم معین شده، encrypt می‌کند. (SHA1 یا MD5). این مقادیر hashed شده، یک راه امن تر برای ذخیره یک کلمه عبور کد شده در فایل یا بانک را فراهم می‌کند.	HashPasswordForStoringInConfigFile()

یک صفحه Login ساده، می‌تواند این متدها را درون خود داشته باشد.

<configuration>

...

<system.web>

...

<authentication mode="Forms">

<forms loginUrl="~/Login.aspx" />

</authentication>

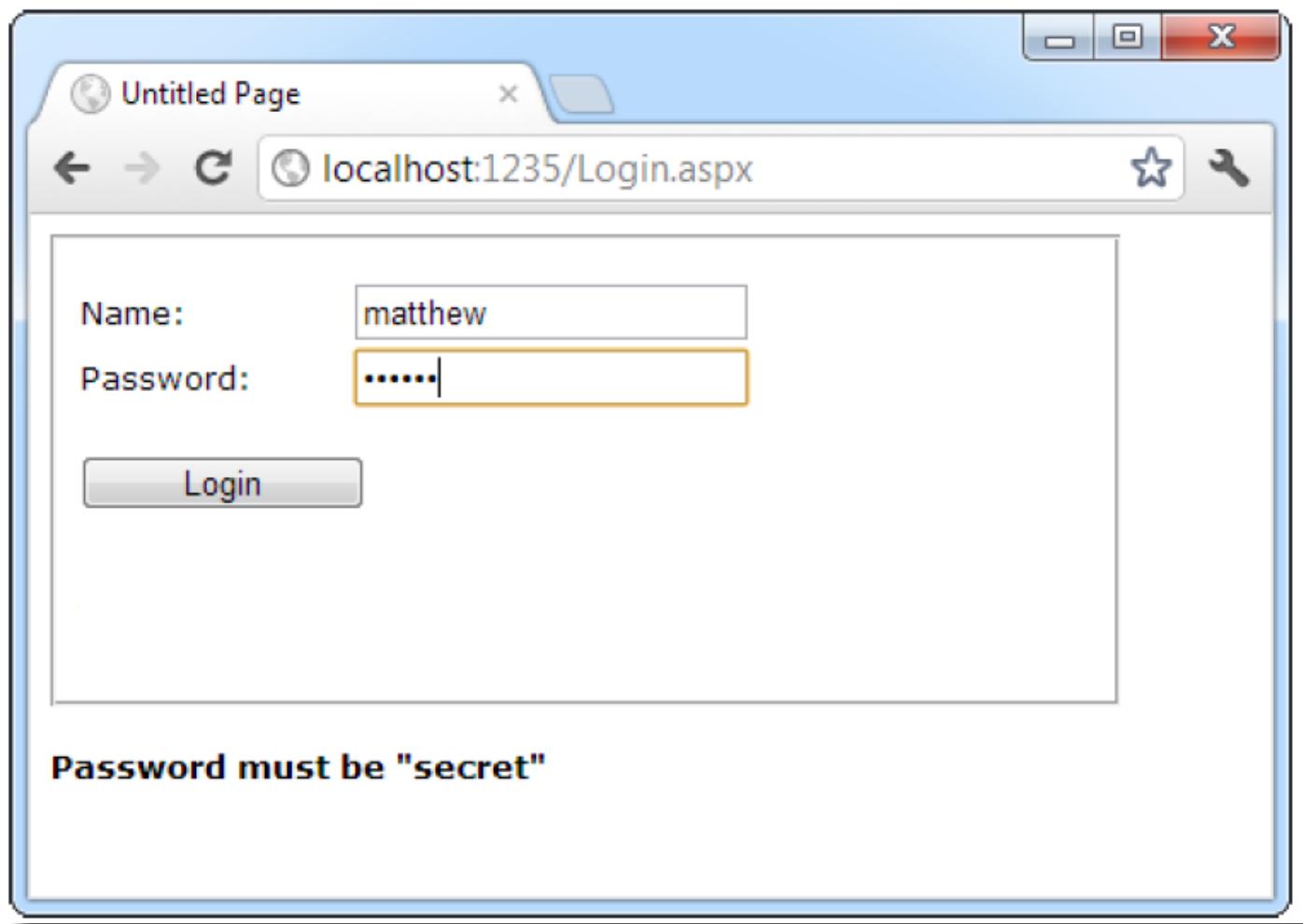
<authorization>

<deny users="?" />

<allow users="\*" />

```
</authorization>
</system.web>
</configuration>
```

اکنون کاربر محدود به صفحه Login.aspx می‌باشد.



زمانی که کاربر روی دکمه Login کلیک کند، صفحه چک می‌کند آیا کاربر کلمه "secret" را تایپ کرده یا نه و سپس از متد RedirectFromLoginPage کاربر را وارد نرم افزا می‌کند.

```
public partial class Login : System.Web.UI.Page
{
    protected void cmdLogin_Click(Object sender, EventArgs e)
    {
        if (txtPassword.Text == "secret")
        {
            FormsAuthentication.RedirectFromLoginPage(
```

```

txtName.Text, false);

}

else

{

lblStatus.Text = "Try again.';

}

}

}

```

متده RedirecFromLoginPage()، به دو پارامتر نیاز دارد. پارامتر اول نام کاربر را تعیین می‌کند. پارامتر دوم، یک متغیر Boolean می‌اشد که تعیین می‌کند آیا شما می‌خواهید که یک کوکی ثابت ایجاد کنید(یک کوکی که از هارد درایو کاربر برای زمان طولانی استفاده کند). البته شما در برنامه‌ها معمولاً کلمه عبور و کاربری را با مقادیر موجود در بانک اطلاعاتی چک می‌کنید.

## بازیابی هویت کاربر

زمانی که کاربر login می‌کند، می‌توانید هویت کاربر را از طریق ویژگی User بازیابی کنید.

```

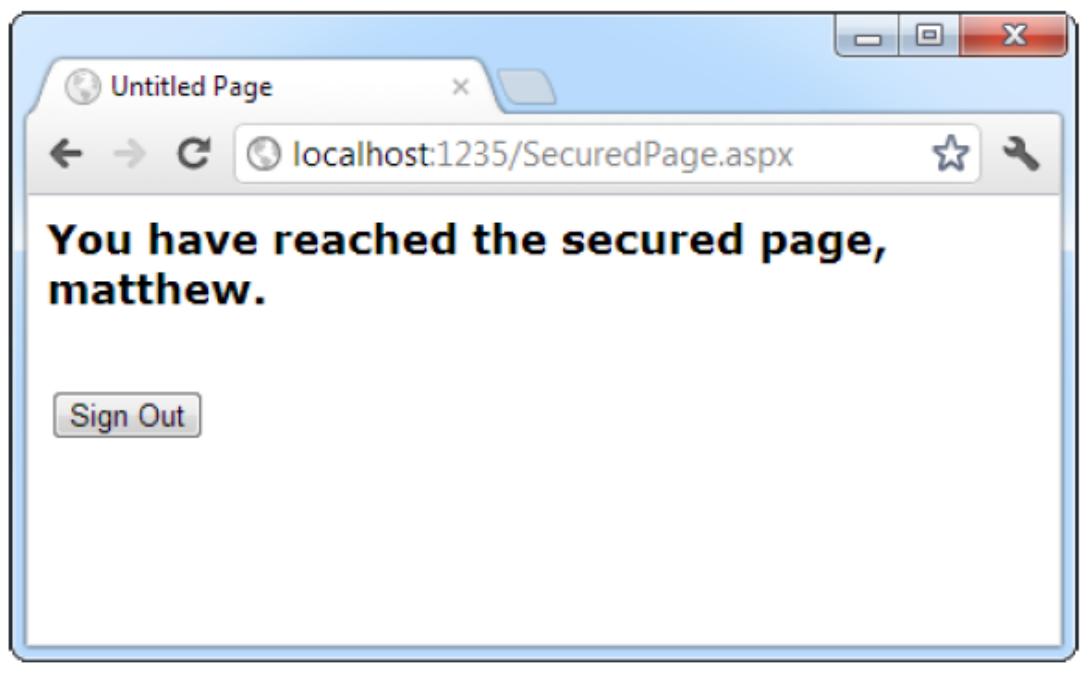
protected void Page_Load(Object sender, EventArgs e)

{
    lblMessage.Text = "You have reached the secured page, ";

    lblMessage.Text += User.Identity.Name + ".";
}

```

از کد بالا می‌توانید برای تعیین هویت کاربر وارد شده به سیستم در هر زمان استفاده کنید.



شی User، اطلاعاتی درباره کاربر وارد شده به سیستم می‌دهد:

- ویژگی Identity، به شما امکان بازیابی نام کاربر Login کرده به سیستم و نوع authentication مورد استفاده شما را می‌دهد.
- متدهای IsInRole()، به شما امکان تعیین اینکه آیا کاربر عضوی از role (نقش) مشخص شده می‌باشد یا نه.

## Sign Out

برای از بین بردن کوکی تعیین اعتبار فرم، و خروج کاربر از سیستم باید یک دکمه sign out ایجاد کنیم.

```
private void cmdSignOut_Click(Object sender, EventArgs e)
{
    FormsAuthentication.SignOut();
    Response.Redirect("~/Login.aspx");
}
```

## کوکی های ماندگار (Persistent Cookies)

در بعضی از موارد، ممکن است از forms authentication login، به جای امنیت برای شخصی سازی استفاده کنید. در این موارد ممکن است تصمیم به ایجاد کوکی های ماندگار بگیرید. این کوکی ها در هارددرایو کاربر مانده و کاربر را حتی اگر مرورگر را ببندد و دوباره باز کند، ساعت ها، روزها و هفته ها در حالت signed in می‌دارند.

ایجاد این کوکی ها، نیازمند کدهای بیشتری به نسبت یک کوکی استاندارد forms authentication است. همچنین به جای استفاده از متدهای RedirectFormLoginPage()، باید بصورت دستی یک authentication ticket ایجاد کرده و زمان منقضی شدن

(Expiration time) آن را تعیین کنید. اگر از کوکی های استاندارد و متده استفاده کنید همه این کارها خودکار انجام می شود)

کوکی های ماندگار، دارای ریسک امنیتی هستند، زیرا یک کاربر دیگر می تواند بدون اجباری به Login، از همان کامپیوتر برای دسترسی به صفحات محافظت شده استفاده کند. برای در اختیار قرار دادن این امکان به کاربر باید آن را انتخابی بگذارد. می توانید این کار را با یک checkBox با نام "Keep me Logged In" یا "مرا در برنامه نگه دار" انجام دهید.

```
// Perform the authentication.

if (txtPassword.Text == "secret")
{
    if (chkPersist.Checked)
    {
        // Calculate the total number of minutes in 20 days,
        // and use that for the timeout.

        int timeout = (int)TimeSpan.FromDays(20).TotalMinutes;

        // Create an authentication ticket.

        FormsAuthenticationTicket ticket = new
        FormsAuthenticationTicket(txtName.Text, true, cookietimeout);

        // Encrypt the ticket (so people can't steal it as it travels over
        // the Internet).

        string encryptedTicket = FormsAuthentication.Encrypt(ticket);

        // Create the cookie for the ticket, and put the ticket inside.

        HttpCookie cookie = new HttpCookie(FormsAuthentication.FormsCookieName,
        encryptedTicket);

        // Give the cookie and the authentication ticket the same expiration.

        cookie.Expires = ticket.Expiration;

        // Attach the cookie to the current response. It will now travel back to
        // the client, and then back to the web server with every new request.

        HttpContext.Current.Response.Cookies.Set(cookie);

        // Send the user to the originally requested page.

        string requestedPage = FormsAuthentication.GetRedirectUrl(txtName.text,
        false);

        Response.Redirect(requestedPage, true);
    }
}
```

```

    }

    else

    {

        // Use the standard authentication method.

        FormsAuthentication.RedirectFromLoginPage(
            txtName.Text, false);

    }

}

```

شایان ذکر است که متد `FormsAuthentication.SignOut()` را صرف نظر از اینکه کوکی نرمال یا ماندگار است، حذف می‌کند.

## Windows Authentication

با این نوع از اعتبارسنجی، وب سرور، مسئول فرآیند تعیین اعتبار می‌باشد. زمانی که از windows authentication استفاده می‌کنید، کاربران را قبل از اینکه امکان دسترسی به محتوای حفاظت شده در وب سایت شما را داشته باشند، وادار به login به IIS می‌کنید. اطلاعات login کاربر، می‌تواند از چند روش منتقل شود(بر اساس network enviroment، مرورگر درخواست کننده و روشی که IIS پیکربندی شده است)، اما نتیجه نهایی، کاربری می‌باشد که از طریق یک حساب کاربری local اعتبارسنجی شده است.

این روش بیشتر برای اینترنت مناسب است که دارای کاربران محدود مشخصی می‌باشند که قبلاً در سرور شبکه ثبت شده اند.

برای پیاده سازی windows – based security باید مراحل زیر را طی کنید :

۱- در فایل web.config، حالت windows authentication را انتخاب کنید. (البته می‌توانید از WAT نیز می‌توانید استفاده کنید)

۲- با استفاده از قوانین authorization، دسترسی افراد ناشناس را به یک پوشه غیر فعال کنید.

۳- حساب های کاربری ویندوزی را روی وب سرور پیکربندی کنید.

## تنظیمات Web.Config

```
<configuration>
  <system.web>
    ...
    <authentication mode="Windows" />
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</configuration>
```

می‌توانید المان `<allow>` و `<deny>` را برای دادن مجوز یا محدود کردن کاربران به فایل‌ها یا پوشه‌های خاص، اضافه کنید. برخلاف `forms authentication`، شما نیاز به تعیین نام سرور یا دامینی که حساب‌های کاربری (accounts) در آن موجود است، دارید. برای نمونه قانون زیر به حساب کاربری با نام matthew، در کامپیوترا با نام WebServer، اجازه دسترسی می‌دهد.

```
<allow users = "WebServer\matthew" />
```

به عنوان یک میانبر، می‌توانید از localhost استفاده کنید:

```
<allow users = ".\matthew" />
```

همچنین می‌توانید انواع مشخصی از کاربران که حساب‌های آنها عضو یک گروه ویندوزی مشابه هستند را محدود کنید :

```
<authorization>
  <deny users="?" />
  <allow roles=".\\SalesAdministrator,.\\SalesStaff" />
  <deny users=".\\matthew" />
</authorization>
```

در این مثال، کلیه کاربرانی که عضو گروه SalesAdministraor یا SalesStaff هستند بصورت خودکار برای دسترسی به صفحات موجود در این پوشه مجاز می‌باشند. درخواست‌های ارسالی از کاربر matthew مجاز نمی‌باشند مگر اینکه او عضو یکی از دو گروه نامبرده باشد. توجه داشته باشید که قانون منطبقی با شرایط کاربر پیدا کند آنها را امتحان می‌کند. همچنین می‌توانید این کار را درون کد انجام دهید. تا زمانی که عبارت شما شامل (\) می‌باشد، باید آن را در #C دوبار بنویسید مگر آنکه از @ قبل از عبارت استفاده کنید:

```

protected void Page_Load(Object sender, EventArgs e)
{
    if (User.IsInRole(@"MyDomainName\SalesAdministrators"))
    {
        // Do nothing; the page should be accessed as normal because
        // the user has administrator privileges.

    }
    else
    {
        // Don't allow this page. Instead, redirect to the home page.

        Response.Redirect("Default.aspx");
    }
}

```

در این نمونه، کد اعضای یک گروه ویندوزی مشخص با نام SalesAdministrators را چک می‌کند. اگر می‌خواهید چک کنید که آیا کاربری عضو یکی از گروه‌های built-in (از قبل ساخته شده) در ویندوز می‌باشد یا نه، نیازی به ذکر نام کامپیوتر یا دامین نیست :

```

if (User.IsInRole(@"BUILTIN\Administrators"))
{
    // (Code goes here.)
}

```

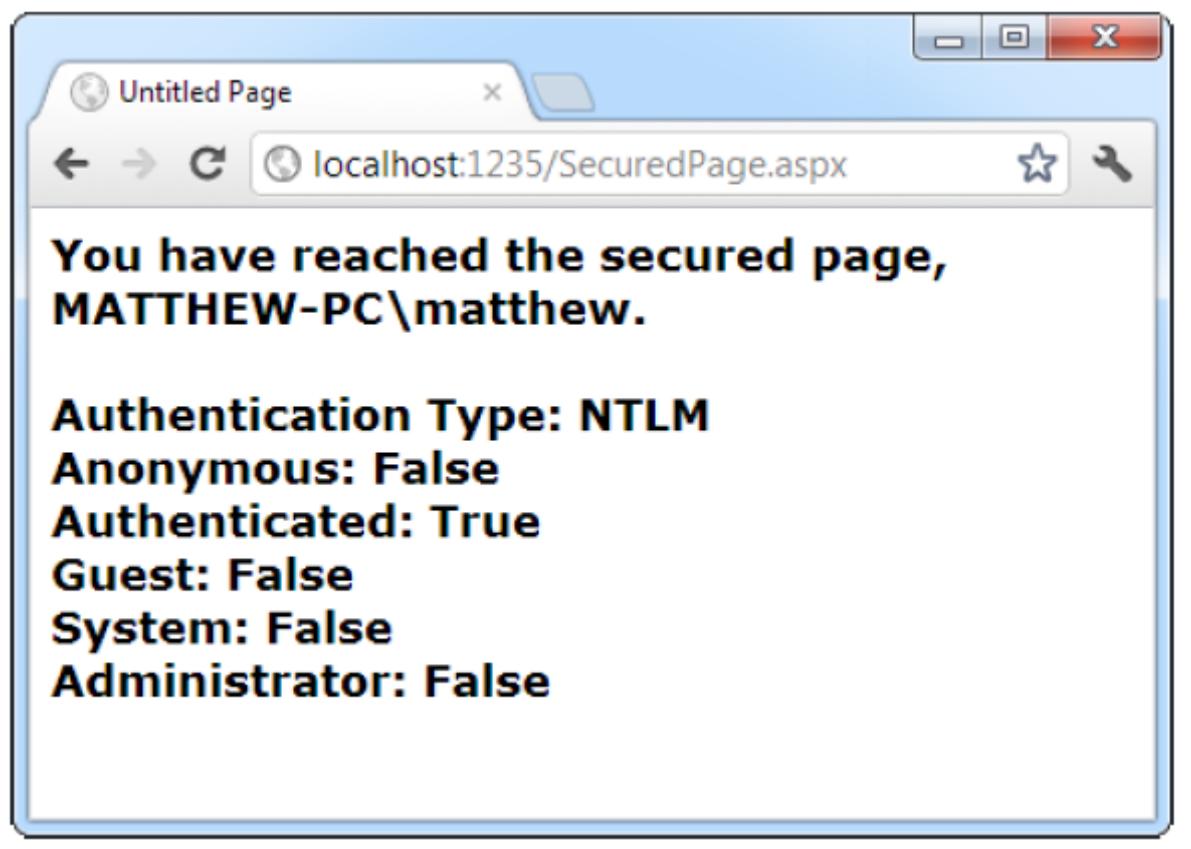
## نقش‌ها یا گروه‌های پیش‌فرض ویندوزی

نقش – Role	توضیحات
AccountOperator	کاربرانی برای مدیریت حساب‌های کاربری روی یک کامپیوتر یا دامین
Administrator	کاربرانی با دسترسی کامل و نامحدود به کامپیوتر و دامین.
BackupOperator	کاربرانی که مسئول backup و restore هستند.
Guest	یک نقش کاربری بسیار محدود می‌باشد.
PowerUser	شبی نقش administraor اما با اندکی محدودیت می‌باشد.
PrintOperator	مانند User معمولی ولی با مجوزهای بیشتر برای کنترل چاپگر.
Replicator	مانند User اما با امتیازات اضافی برای پشتیبانی از تکرار فایل در یک دامنه
SystemOperator	مانند administrator با اندکی محدودیت می‌باشد. معمولاً این نقش کامپیوتر را مدیریت می‌کند.
User	تنها برنامه‌های تعیین شده را اجرا می‌کند.

```

public partial class SecuredPage : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        StringBuilder displayText = new System.Text.StringBuilder();
        displayText.Append("You have reached the secured page, ");
        displayText.Append(User.Identity.Name);
        WindowsIdentity winIdentity = (WindowsIdentity)User.Identity;
        displayText.Append("<br /><br />Authentication Type: ");
        displayText.Append(winIdentity.AuthenticationType);
        displayText.Append("<br />Anonymous: ");
        displayText.Append(winIdentity.IsAnonymous);
        displayText.Append("<br />Authenticated: ");
        displayText.Append(winIdentity.IsAuthenticated);
        displayText.Append("<br />Guest: ");
        displayText.Append(winIdentity.IsGuest);
        displayText.Append("<br />System: ");
        displayText.Append(winIdentity.IsSystem);
        displayText.Append("<br />Administrator: ");
        displayText.Append(User.IsInRole(@"BUILTIN\Administrators"));
        lblMessage.Text = displayText.ToString();
    }
}

```



# بخش چهارم : امنیت و ب سایت

\_\_\_\_\_ فصل سیزدهم: اصول امنیت

Membership فصل چهاردهم:

Profile فصل پانزدهم



در فصل قبل آموختید که چگونه می‌توانید از اعتبار سنج فرم ASP.NET به عنوان امنیت وب سایت خود استفاده کنید. با استفاده از forms authentication می‌توانید کاربران را تعیین اعتبار کرده و آنها را از دسترسی به صفحاتی که نباید محدود نمایید. بهتر از همه اینکه ASP.NET، کل فرآیند را با استفاده از ایجاد و چک کردن کوکی اعتبارسنجی فرم، برای شما مدیریت می‌کند.

البته هنوز باید برخی از کارها را انجام دهید. باید صفحه Login ایجاد کرده و تصمیم بگیرید که چگونه محتوای public را از private جدا سازید و تعیین کنید هر کاربر چه کاری باید انجام دهد. مایکروسافت یک لایه از امکانات را به فریمورک اعتبارسنجی فرم اضافه کرده است که با نام membership شناخته می‌شود.

## امكان membership به سه گروه تقسیم می‌شود :

### User record Management

به جای ایجاد بانک اطلاعاتی خودتان، اگر از ویژگی membership استفاده کرده باشد، می‌تواند این اطلاعات کاربر را نگهداری کند. حتی می‌تواند قوانین پیشرفتی مانند نیاز به نشانی email، سوالهای امنیتی و Strong Password را پیاده سازی کند.

### کنترل‌های امنیتی:

هر وب سایت امن به یک صفحه Login نیاز دارد. با کنترل‌های امنیتی ASP.NET، نیازی به طراحی آنها توسط شما وجود ندارد، در عوض می‌توانید از بخش Login نوار ابزار استفاده کنید. با استفاده از Property ها و رویدادهای کنترل‌ها، می‌توانید کنترل‌های امنیتی را سفارشی سازی کنید.

### امنیت مبتنی بر نقش (Role-Based)

در بسیاری از سایت‌ها شما باید مجوزهای مختلفی به کاربران مختص دهید. می‌توانید کاربران را در گروه‌هایی که مجوزهای قطعی را دارند قرار دهید. این گروه‌ها role نامیده می‌شوند و امكان membership موجود در ASP.NET، شامل ابزارهایی برای ایجاد خودکار بانک اطلاعاتی با اطلاعات role، می‌باشد.

## MemberShip Data Store

ویژگی کلیدی membership، توانایی ASP.NET در ذخیره اطلاعات اعتباری در بانک اطلاعاتی می‌باشد. می‌توانید یک وب سایت امن را با کد و کار کمتر بنویسید.

برخی از دلایلی که ممکن است نخواهید از منبع ذخیره سازی membership استفاده کنید :

- اگر نخواهید داده‌های کاربر را در یک بانک اطلاعاتی ذخیره کنید : در تئوری، می‌توانید لیست کاربران را در هر منبع ذخیره سازی از فایل XML گرفته تا بانک اطلاعاتی Oracle نگهداری کنید. بصورت فنی، هر منبع ذخیره سازی، نیازمند یک ارائه provider membership (membership provider) شامل دو ASP.NET (می‌باشد. البته Active Directory و membership های مرتبط را پیدا کنید.

- نیاز به سازگاری: اگر قبلاً یک جدول برای ذخیره سازی اطلاعات کاربر ایجاد کرده باشد، ممکن است تغییر وضعیت به منبع ذخیره سازی مشکل باشد. این به آن دلیل است که SQL Server Membership Provider، دارای ساختار جدولی خاصی می‌باشد که با جداول موجود کار نمی‌کند زیرا دارای ترکیبی از فیلدها و datatype ها مختلف می‌باشد. حتی اگر نیازی به نگهداری ساختار کنونی جدول ندارید، بازسازی تمامی رکوردهای کاربران در منبع ذخیره سازی membership کار سختی می‌باشد.

- اگر بخواهید اطلاعات کاربر را در برنامه‌های غیر ASP.NET مدیریت کنید: همانطور که دیدید، به شما مجموعه قدرتمندی از اشیا برای تعامل با داده membership خواهد داد. برای نمونه، می‌توانید رکوردهای کاربران را ویرایش، حذف، بازیابی و... نمایید. می‌توانید از membership در سایر انواع برنامه‌های دات نتی استفاده کنید(برای نمونه می‌توانید یک برنامه ویندوزی برای مدیریت حساب‌های کاربری ایجاد نمایید). به هر حال اگر بخواهید برنامه دیگری خارج از دات نت ایجاد کنید که نیاز به انجام این کارها داشته باشد، باید با ساختار جداول membership آشنا شوید. در این مورد، مدیریت کاربر با دستورات SQL بسیار آسان تر از آشنایی با ساختار کامل این جداول می‌باشد.

اگر تصمیم بگیرید که از منبع ذخیره سازی ADO.NET استفاده نکنید، می‌توانید کدهای membership را برای بازیابی رکوردهای کاربران نوشته و اطلاعات اعتباری کاربر را چک کنید. با استفاده از این تکنیک، ایجاد صفحه Login شما، کمی سخت تر از آنچه در قبل توضیح داده شد، می‌باشد.

## SQL Server Express با استفاده از Membership

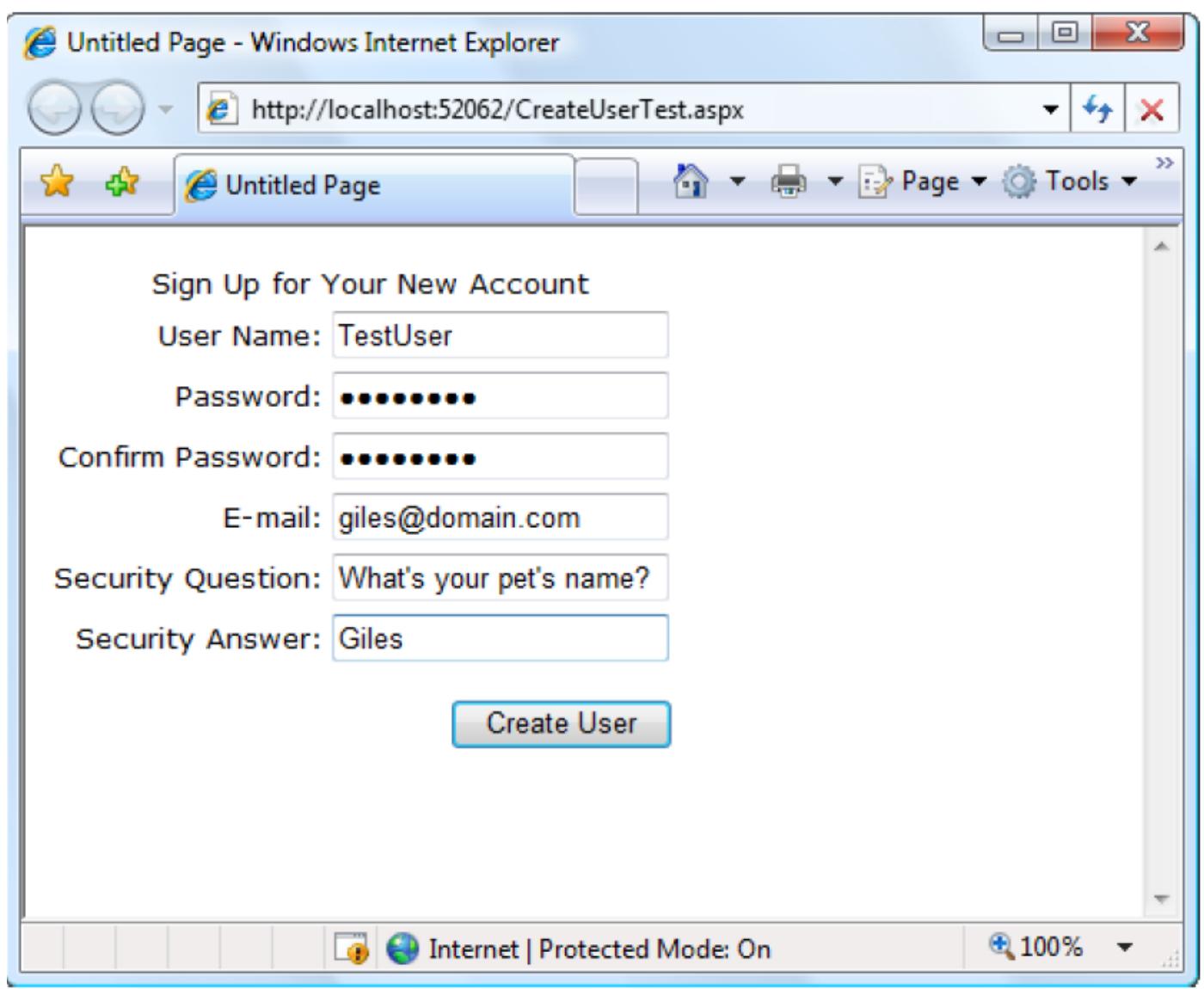
بصورت پیش فرض، روی هر وب سایت ایجاد شده، فعال می‌باشد. membership provider پیش فرض، باعث فرض‌های زیر می‌شود :

- می‌خواهید بانک اطلاعاتی membership را با استفاده از SQL Server Express ذخیره کنید.
- می‌خواهید از SQL Server LocalDB (که درون Visual Studio قرار گرفته است) استفاده کنید.
- منبع ذخیره سازی membership شما، یک فایل با نام aspnetdb.mdf می‌باشد که در پوشه App\_Data در برنامه تحت وب شما ذخیره می‌شود.

فرض‌های بالا به شما امکان ایجاد ساخت چند برنامه تحت وب که دارای بانک اطلاعاتی جداگانه‌ای می‌باشند، خواهد داد. زیرا هر وب سایت، دارای فایل aspnetdb.mdf مربوط به خودش می‌باشد. این فایل‌ها هرگز در SQL Server ثبت نشده‌اند و این یعنی که زمانی که یک connection در یک برنامه دیگر باز می‌کنید، تعداد زیادی از بانک‌های اطلاعاتی کاربر را نمی‌بینید. تنها راه اتصال به آن بانک، تعیین مسیر فایل در connection string می‌اشد.

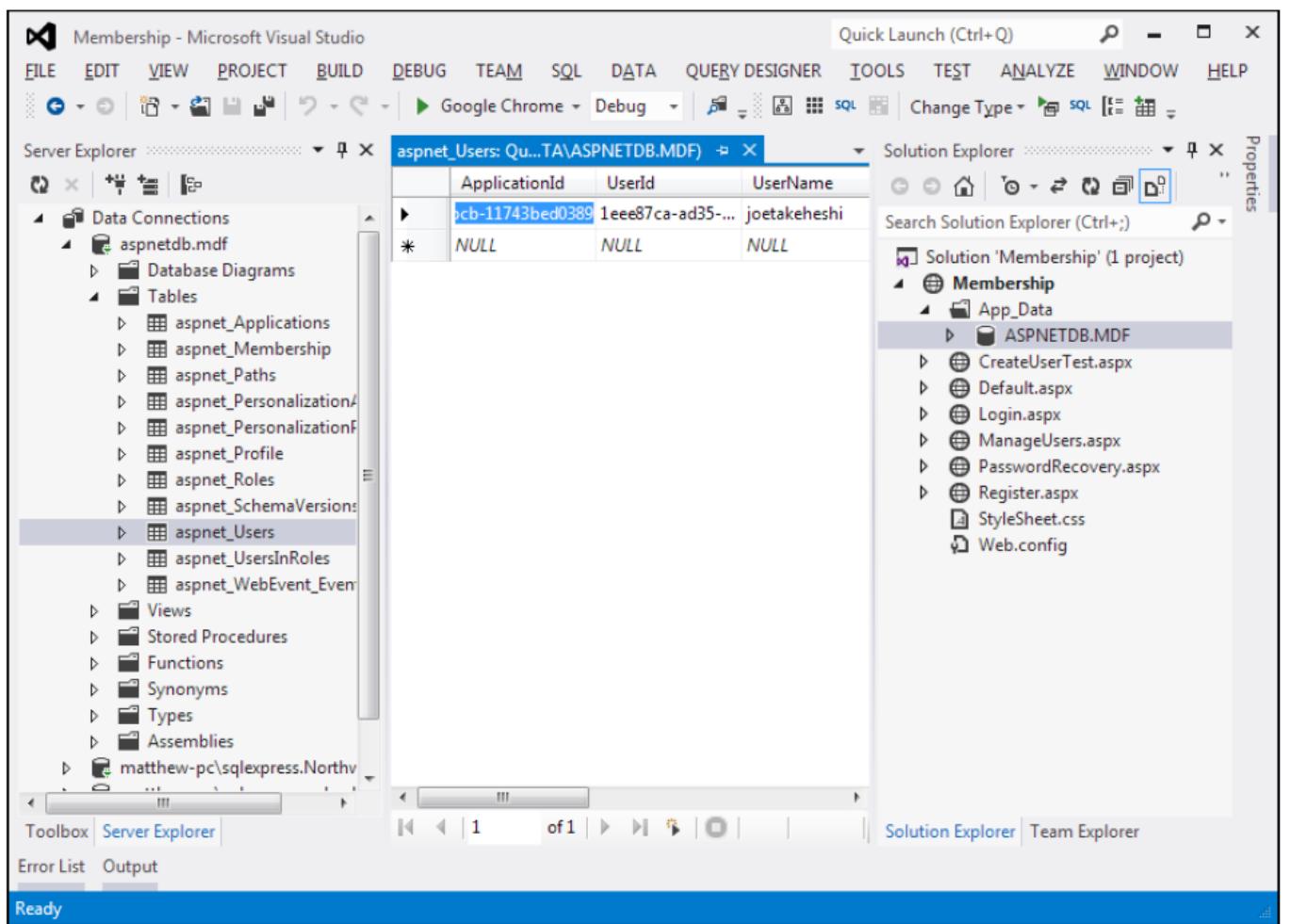
فایده دیگر این روش، deploy کردن آسان وب سایت شما می‌باشد. فرض کنید وب سروری که می‌خواهید برنامه در آن Deploy می‌کنید، دارای SQL Server Express باشد، بنابراین نیازی به تغییر connection string نیست. همچنین نیاز به اجرای مراحل اضافی برای نصب بانک اطلاعاتی نخواهید داشت. به سادگی فایل aspnetdb.mdf را با وب سایت خود کپی می‌کنید، بنابراین پشتیبانی از چند وب سایت که هر کدام دارای بانک مربوط به خود می‌باشد کار ساده‌ای است.

برای اینکه ببینید، چگونه کار می‌کند، کنترل CreateUserWizard را به داخل صفحه خود بیندازید. اکنون صفحه خود را بدون نوشتگر هیچ کد اضافه‌ای اجرا کنید.



اطلاعات همه textbox‌ها را پر کرده و توجه داشته باشید که کلمه عبور بصورت پیش فرض، حداقل ۷ کاراکتر می‌باشد که یکی از آنها باید عدد یا حرف باشد (undercore یا ...). سپس روی Create User کلیک کنید. سپس روی membership provider را ایجاد (اگر قبلًا ایجاد نشده باشد) و سپس کاربر جدید را وارد آن می‌کند. در نهایت یک پیام برای ایجاد کاربر نشان می‌دهد.

می‌توانید جدول aspnet\_Users را باز کنید تا ببینید که کاربر جدید در آن اضافه شده است. در VS ابتدا به پوشه App\_Data رفته و با دوبار کلیک روی فایل نظر باعث ایجاد یک connection شوید. سپس روی جدول کلیک راست کرده و گزینه Show Table Data را انتخاب کنید.



## استفاده از نسخه کامل SQL Server

گر از بانک اطلاعاتی ایجاد شده بصورت خودکار برای SQL Server Express استفاده کرده باشد، نیازی به تغییر فایل web.config نمی‌باشد. اکنون اگر از نسخه کامل SQL Server نیز استفاده کنید، هنوز از تنظیمات پیش فرض membership استفاده خواهد شد. Connection String پیش فرض مورد استفاده برای membership دارای نام LocalSQLServer می‌باشد که می‌توانید تنظیمات آن را در فایل machine.config مستقیماً تغییر دهید. اگر می‌خواهید این تنظیمات را برای یک برنامه تغییر دهید بهتر است این تغییرات را درون فایل web.config مربوطه به آن برنامه قرار دهید. ابتدا با استفاده از المان <Clear> تمامی connection string های موجود را حذف کنید. سپس رشته اتصال LocalSQLServer را اضافه کنید.

```

<configuration>

  <connectionStrings>

    <clear />

    <add name="LocalSqlServer" providerName="System.Data.SqlClient"
      connectionString="Data Source=localhost;Integrated Security=SSPI;
      Initial Catalog=aspnetdb" />

  </connectionStrings>

  . . .

</configuration>
```

 نکته: تنظیمات پیش فرض membership در یک فایل با نام Local Machine.Config و رشتہ اتصال قرار می‌گیرد. این فایل در نشانی زیر قرار دارد و تنظیمات کلی که روی هر برنامه ای صدق می‌کند در آن قرار می‌گیرد، برخلاف فایل web.config که مخصوص یک برنامه می‌باشد.

c:\Windows\Microsoft.NET\Framework\[Version]\Config

c:\Windows\Microsoft.NET\Framework64\[Version]\Config

در این روش، بانک اطلاعاتی بصورت خودکار ایجاد نمی‌شود. باید آن را با استفاده از دستور aspnet\_regsql در ابزار command-line ایجاد کنید. برای این کار، Visual Studio Command Prompt را باز کنید.

A::: Programs>Micorosoft Visual Studio 2010>Visual Studio Tools>Visual Studio Command Propmt

دستور aspnet\_regsql را در آن تایپ کنید. ویندوز یک ویزارد به شما نشان می‌دهد که مراحل ساخت بانک را با آن طی می‌کنید. البته می‌توانید با استفاده از دستورات تکمیلی تغییراتی در ایجاد بانک اعمال کنید. دستور بیان شده بانک را کامل می‌سازد ولی اگر نیازی به بخش‌هایی از آن مانند تعیین مجوز profile یا webpart، می‌توانید جداول مورد نیاز خود را تعیین کنید.

aspnet\_regsql -S (local) -E -A all

البته لیست کامل این دستورات را می‌توانید از msdn دریافت کنید.

## پیکربندی Membership Provider

می‌توانید تغییراتی در تنظیمات membership انجام دهید، مثلاً می‌توانید Password Policy پیش فرض را تغییر دهید.

 نکته: می‌توانید فایل machine.config را تغییر دهید تا این تغییرات برای همه برنامه اعمال شود ولی در هنگام deploy با مشکل روبرو می‌شوید. بنابراین بهتر است یک membership provider جدید در فایل web.config پیکربندی کنید.

<configuration>

...

<system.web>

...

<membership defaultProvider="MyMembershipProvider">

<providers>

```

<!-- Clear any existing providers. -->
<clear />

<!-- Define your provider, with custom settings. -->
<add name="MyMembershipProvider" .../>

</providers>
</membership>
</system.web>
</configuration>

```

در زیر می بینید که با استفاده از attribute provider هایی، خود را سفارشی کرده ایم :

```

<membership defaultProvider="MyMembershipProvider">
<providers>
  <clear/>
  <add
    name="MyMembershipProvider"
    Type="System.Web.Security.SqlMembershipProvider"
    connectionStringName="LocalSqlServer"
    requiresQuestionAndAnswer="false"
    minRequiredPasswordLength="1"
    minRequiredNonalphanumericCharacters="0" />
</providers>
</membership>

```

## تنظیمات رایج membership



توضیحات	خصیصه—Attribute
نامی را برای membership provider تعیین نمی‌کند. می‌توانید از آن برای بازیابی اطلاعات provider از طریق کدنویسی اقدام کنید.	name*
نوع provider را مشخص می‌کند. در این فصل شما فقط از استفاده system.Web.Security.SqlMembershipProvider بررديدة. البته ASP.NET شامل ActiveDirectoryMembership- membership Provider نیز می‌باشد که به شما امکان استفاده از امکانات Active Directory windows authentication را می‌دهد. در نهایت می‌توانید از provider سفارشی که شما یا یک شرکت دیگر ایجاد کرده است استفاده نمایید.	type*
نام برنامه تحت وب را تعیین می‌کند. اگر چند وب سایت داشته باشید که از یک بانک اطلاعاتی membership استفاده می‌کنند، این نام بکار می‌آید. بنابراین با تخصیص نام جداگانه برای هر برنامه، کلیه موارد مانند profile، user ها و ... جدا می‌شود و تنها توسط برنامه مربوطه قابل استفاده می‌باشد.	applicationName
توضیحات اختیاری درباره provider	connectionStringName*
روش ذخیره کلمه عبور در بانک اطلاعاتی را تعیین می‌کند. می‌توانید از Clear (ذخیره کلمه عبور بدون هیچ encryption)، Encrypted (کلمه عبور توسط کلید خاص کامپیوتر encrypt می‌شود)، یا Hashed (کلمه عبور hash می‌شود و مقادیر hash شده درون بانک اطلاعاتی ذخیره می‌شوند). Hash کردن کلمه عبور، حفاظت مشابهی با encryption باشد. اگر کاربر encrypt hash با hash این است گه زمانی که کلمه عبور hash می‌شود، نمی‌توان آن را بازیابی کرد و تنها می‌توان آن را reset کرد.	passwordFormat
حداقل طول کلمه عبور را تعیین می‌کند.	minRequiredPasswordLength
تعداد کاراکترهای غیر الفبایی (کاراکترهای به غیر از عدد و حرف) که کلمه عبور باید آن را داشته باشد. اگر کاربر کمتر از این تعداد را وارد کند تلاش آن با شکست روبرو می‌شود.	minRequiredNonalphanumericCharacters
تعداد دفعاتی که کاربر می‌تواند کلمه عبور را اشتباه وارد کند. بیشتر از آن حساب کاربری او قفل می‌شود و غیر قابل دسترس خواهد بود. پیش فرض آن ۵ است.	maxInvalidPasswordAttempts
زمانی را که در آن maxInvalidPasswordAttempts، اندازه گیری می‌شود را مشخص می‌کند. اگر مقدار آن را ۳۰ در نظر بگیرید، در صورتی که کاربر سقف تعداد مجاز برای ثبت کلمه عبور اشتباه را رد بازه ۳۰ دقیقه رد کند، حساب کاربری وی قفل می‌شود.	passwordAttemptWindow
زمانی که کاربر کلمه عبور را فراموش کند می‌تواند آن را reset کند.	enablePasswordReset

تعیین می‌کند آیا کلمه عبور را می‌توان بازیابی کرد یا نه (به یک e-mail به کاربر). این ویژگی در صورتیکه PasswordFormat Hashed باشد باشد.	enablePasswordRetrieval
تعیین می‌کند آیا پاسخ امنیتی شما در هنگام بازیابی یا reset کردن کلمه عبور لازم است یا نه.	requiresQuestionAndAnswer
اگر false باشد اجازه می‌دهد بیش از یک کاربر دارای یک email مشابه باند.	requiresUniqueEmail

&lt;membership&gt;

&lt;providers&gt;

```

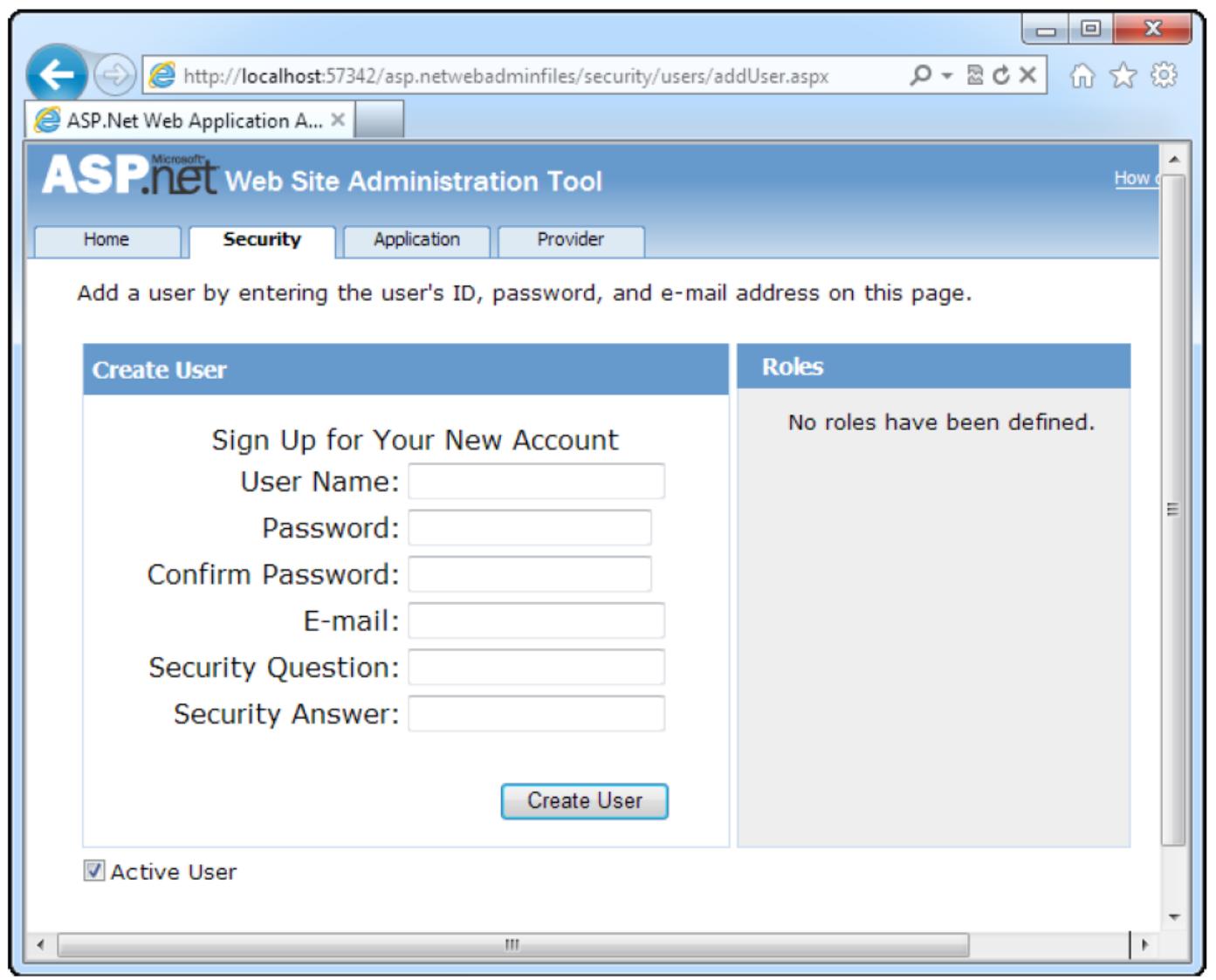
<add name="AspNetSqlMembershipProvider"
      Type="System.Web.Security.SqlMembershipProvider . . ."
      connectionStringName="LocalSqlServer"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="true"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="7"
      minRequiredNonalphanumericCharacters="1"
      passwordAttemptWindow="10" />
</providers>
</membership>

```

## ایجاد کاربر با استفاده از WAT

برای ایجاد کاربر از طریق WAT، منوی WebSite>ASP.NET Configuration را انتخاب کنید. روی زبانه Security کلیک کنید. در پایین سمت چپ صفحه، تعداد کاربران موجود را نشان می‌دهد و لینک هایی برای ایجاد یا ویرایش کاربران موجود در اختیار شما قرار می‌دهد.

The screenshot shows the Microsoft ASP.NET Web Site Administration Tool interface. The browser address bar shows `http://localhost:57342/asp.netwebadminfiles/security/security.aspx`. The main title is "ASP.net Microsoft Web Site Administration Tool". The navigation tabs at the top are Home, **Security**, Application, and Provider. The "Security" tab is selected. Below the tabs, there is descriptive text about managing security settings for the application, including user authentication, roles, and permissions. A link "Use the security Setup Wizard to configure security step by step." is provided. At the bottom, there is a table with three columns: "Users", "Roles", and "Access Rules". The "Users" column shows 1 existing user, a "Create user" link, a "Manage users" link, and a "Select authentication type" link. The "Roles" column shows 0 existing roles, a "Disable Roles" link, and a "Create or Manage roles" link. The "Access Rules" column contains links for "Create access rules" and "Manage access rules".



البته به غیر از این ابزار می‌توانید از کلاس Membership برای ایجاد کاربر نیز استفاده کنید :

```
// Create a user record based with user name, password, and e-mail information.
```

```
Membership.CreateUser(userName, password, email);

// Here's an example with hard-coded values:

Membership.CreateUser("joes", "ignreto12__", "joes@domains.com");
```

البته می‌توانید از سایر Overload های متدهای CreateUser نیز برای تعیین جزئیاتی مانند سؤال و پاسخ امنیتی کلمه عبور استفاده کنید.

```
MembershipCreateStatus createStatus;

Membership.CreateUser("joes", "ignreto12__", "joes@domains.com",
"What is your favorite restaurant?", "Saigon", true, out createStatus);
```

پارامتر اول کلمه عبور، نام کاربری و ... را دریافت می‌کند. پارامتر دوم تعیین می‌کند که حساب کاربری IsApproved هست یا نه که اگر false باشد حساب کاربری غیر فعال می‌گردد و باید آن را با متدهای Membership.UpdateUser() ویرایش کنید.

پارامتر آخر یک مقدار از نوع شمارشی MembershipCreateStatus بر می‌گرداند اگر مقدار آن برابر با MembershipCreateStatus.Success نباشد، در هنگام ایجاد رکورد خطأ خطا رخ داده است. (به عنوان مثال اگر email تکراری قبول نکند و شما یک مقدار تکراری وارد کرده اید).

## کلاس‌های Membership و Memebership

یک کلاس مفید با متدهای کارآمد static مانند CreateUser() می‌باشد که در فضای System.WebSecurity قرار دارد.

### متدهای Membership

توضیحات	متدها
	CreateUser()
کلمه کاربری را به این متده می‌دهد و آن کاربر را برای شما از بانک حذف می‌کند. همچنین می‌توانید تعیین کنید که آیا کلیه داده‌های مرتبط با کاربر را نیز حذف کند یا نه. بصورت پیش‌فرض این کار را انجام می‌دهد.	DeleteUser()
	UpdateUser()
یک کاربر را بر اساس نام کاربری از بانک بازیابی می‌کند	GetUser()
اگر دو کاربر با یک email داشته باشیم این متده اولین کاربر را می‌آورد.	GetUserNameByEmail()
کاربرانی که با نام کاربری تعیین شده سازگاری داشته باشند را می‌آورد. اگر بخشی از نام کاربری آنها نیز با نام کاربری تعیین شده منطبق باشد نیز می‌تواند بازیابی شود.	FindUsersByName()
	FindUsersByEmail()
	GetAllUsers()
	GetNumberOfUsersOnline()
ایجاد یک کلمه عبور تصادفی با طول مشخص. زمانی که از طریق کدنویسی یک کاربر ایجاد می‌کنید می‌توانید از این متده استفاده نمایید.	GeneratePassword()
چک می‌کند که آیا کاربری با نام کاربری و کلمه عور مشخص شده در بانک موجود هست یا نه. برای نوشتن منطق اعتبارسنجی کاربر می‌توان از این متده استفاده کرد.	ValidateUser()

کلاس Memebrship همچنین static property های read-only نیز ارائه می‌دهد که امکان بازیابی اطلاعات درباره پیکربندی membership provider شما را در اختیار قرار می‌دهد. به عنوان نمونه می‌توانید حداکثر طول کلمه عبور و ... را بازیابی کنید.



توضیحات	متدها
کاربری که بر اثر تعداد login های نامعتبر قفل شده است را از حالت قفل خارج می کند.	UnlockUser()
کلمه عبور کاربر را بازیابی می کند. اگر ویژگی requiresQuestionAndAnswer با مقدار true برابر باشد، باید پاسخ مربوط به پرسش امنیتی را ارائه دهید تا کلمه عبور بازیابی شود. توجه کنید که این متدها در صورتی که مقدار enablePasswordRetrieval برابر با false باشد یا کلمه عبور Hash شده باشد کار نمی کند.	GetPassword()
کلمه عبور کاربر را با ک کلمه عبور جدید که بصورت تصادفی توسط این متدها ایجاد شده است جایگزین می کند. اگر ویژگی requiresQuestionAndAnswer برابر با true باشد باید پاسخ صحیحی را ارائه دهید. می توانید کلمه عبور جدید را به کاربر نشان دهید یا برای ارسال کنید.	ResetPassword()
	ChangePassword()
	ChangePasswordQuestionAndAnswer()

User Name	Email	
joes	joes@domains.com	Select
liu_liu	iu@company.com	Select
test	testuser@nodomain.com	Select

برای ایجاد این صفحه باید یک GridView که لیستی از اشیای MembershipUser را ارائه می دهد داشته باشد. برای هر کاربر مقدار کلمه کاربری و Email نمایش داده می شود.

```

<asp:gridview id="gridUsers" runat="server" onselectedindexchanged="gridUsers_SelectedIndexChanged">
    <autogeneratecolumns=False datakeynames="UserName">
        <Columns>
            <asp:BoundField DataField="UserName" HeaderText="User Name" />
            <asp:BoundField DataField="Email" HeaderText="Email" />
            <asp:CommandField ShowSelectButton="True" />
        </Columns>
    </asp:gridview>

protected void Page_Load(object sender, EventArgs e)
{
    gridUsers.DataSource = Membership.GetAllUsers();
    gridUsers.DataBind();
}

```

زمانی که کاربر یک رکورد را انتخاب می‌کند، شی `MembershipUser` مربوطه بازیابی می‌شود. این شی سپس به یک `collection` اضافه می‌شود که می‌تواند به `DetailsView` دیگر bind شود.

```

protected void gridUsers_SelectedIndexChanged(object sender, EventArgs e)
{
    List<MembershipUser> list = new List<MembershipUser>();
    list.Add(Membership.GetUser(gridUsers.SelectedValue.ToString()));
    detailsUser.DataSource = list;
    detailsUser.DataBind();
}

```

در زیر `detailsview` را اضافه می‌کنیم :

```
<asp:DetailsView ID="detailsUser" runat="server"></asp:DetailsView>
```

از ایجاد ردیف خودکار استفاده می‌کند، زیرا `AutoGenerateRows` بصورت پیش فرض برابر با `true` می‌باشد.

Untitled Page

localhost:57193/ManageUsers.aspx

User Name	Email	
joes	joes@domains.com	<a href="#">Select</a>
liu_liu	iu@company.com	<a href="#">Select</a>
test	testuser@nodomain.com	<a href="#">Select</a>

<b>UserName</b>	liu_liu
<b>Email</b>	iu@company.com
<b>PasswordQuestion</b>	What's your favorite pet's name?
<b>Comment</b>	
<b>IsApproved</b>	<input checked="" type="checkbox"/>
<b>IsLockedOut</b>	<input type="checkbox"/>
<b>LastLockoutDate</b>	12/31/1753 7:00:00 PM
<b>CreationDate</b>	6/28/2012 12:26:30 AM
<b>LastLoginDate</b>	6/28/2012 12:26:30 AM
<b>LastActivityDate</b>	6/28/2012 12:26:30 AM
<b>LastPasswordChangedDate</b>	6/28/2012 12:26:30 AM
<b>IsOnline</b>	<input checked="" type="checkbox"/>
<b>ProviderName</b>	AspNetSqlMembershipProvider

## تائید اعتبار با Membership

درای تائید کاربر نیازی به استفاده از ADO.NET و بازیابی اطلاعات او نمی‌باشد. می‌توانید با استفاده از کلاس MemberShip بسیاری از این کارها را انجام دهید:

```
protected void cmdLogin_Click(object sender, EventArgs e)
{
    if (Membership.ValidateUser(txtName.Text, txtPassword.Text))
    {
        FormsAuthentication.RedirectFromLoginPage(txtName.Text, false);
    }
    else
    {
        lblStatus.Text = "Invalid username or password.";
    }
}
```

### کنترل‌های امنیتی

توضیحات	کنترل
کادر کلمه عبور و کلمه کاربری را با یک دکمه Login نمایش می‌دهد	Login
یک دکمه Login را نمایش می‌دهد که در صورتیکه کاربر قبلًا وارد سایت نشده باشد، کاربر را به صفحه Login هدایت می‌کند. در غیر اینصورت این دکمه عبارت sign-out را نمایش می‌دهد.	LoginStatus
کلمه کاربری، کاربر وارد شده به سایت را نمایش می‌دهد.	LoginName
محتوای متفاوتی را بر اساس اینکه کاربر login کرده است یا خیر به او نشان می‌دهد. همچنین می‌توانید از این کنترل به منظور نمایش محتوای مختلف برای گروه و نقش‌های کاربری استفاده کنید.	LoginView
	PasswordRecovery
	ChangePassword

### Login کنترل

این کنترل دارای امکانات زیر می‌باشد:

- دارای validator به منظور جلوگیری از postback شدن صفحه در صورت عدم ورود نام کاربری و کلمه عبور توسط کاربر می‌باشد. این validator ها از client-side validation و همچنین server-side validation استفاده می‌کنند.

- فرآیند sign-in در صورت ورود موفق کاربر را هندل می‌کند. در صورت نامعتبر بودن اطلاعات اعتباری، یک پیام خطا نشان می‌دهد.

- Check box با نام remember me یا "مرا به خاطر بسپار" را ارائه می‌دهد که در صورت انتخاب، یک کوکی ماندگار را روی کامپیوتر کاربر ذخیره می‌کند.

برای استفاده از این کنترل نیاز به نوشتن هیچ کدی نمی‌باشد. البته می‌توانید کدهای سفارشی خود را نیز در آن بنویسید. باید با رویدادهای کنترل Login در تعامل باشید.



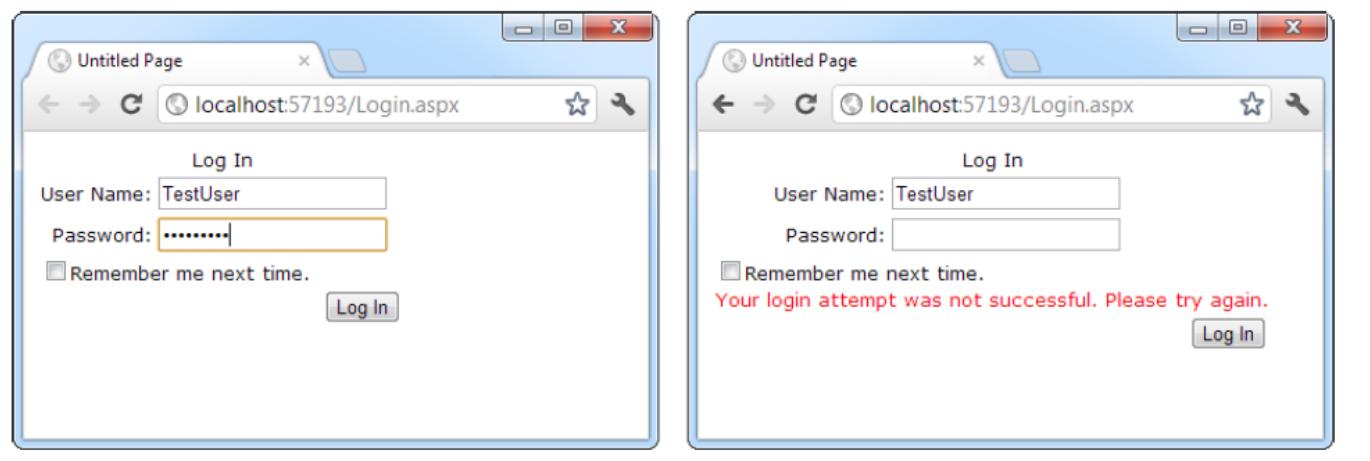
رویداد	توضیحات
LoggingIn	قبل از شدن کاربر authenticate raise می‌شود.
LoggedIn	پس از شدن کاربر authenticate raise می‌شود.
LoginError	زمانی که تلاش برای ورود با شکست روبرو شود raise می‌شود.
Authenticate	برای کردن کاربر authenticate raise می‌شود. اگر این رویداد را هندل کنید، باید ورود کاربر را خودتان هندل کنید و کنترل Login هیچ کاری انجام نمی‌دهد.

: مثال

```
protected void Login1_LoginError(object sender, EventArgs e)
{
    lblStatus.Text = "Have you forgotten your password?";
    lnkRedirectToPasswordRetrieval.Visible = true;
}
```

: مثال

```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
{
    if (Membership.ValidateUser(Login1.UserName, Login1.Password))
    {
        e.Authenticated = true;
    }
    else
    {
        e.Authenticated = false;
    }
}
```



رایج ترین property های فرمت دهی برای کنترل Login ها هستند که امکان تغییر font,color و چیدمان بخش های مشخصی از کنترل را فراهم می کنند.



استایل	توضیحات
TitleTextStyle	استایل متن عنوان کنترل را تعیین می کند.
LabelStyle	استایل label های مربوط به کلمه عبور و کاربری را تعیین می کند.
TextBoxStyle	استایل کنترل های username و password
LoginButtonStyle	استایل متنی که در صورت شکست در login نمایش داده خواهد شد.
FailureTextStyle	استایل کنترل با متن Remember me را تعیین می کند.
CheckBoxStyle	
ValidatorTextStyle	
HyperLinkStyle	استایل کلیه لینک های موجود در کنترل را تعیین می کند. شامل لینک هایی که به شما امکان ایجاد رکورد جدید، بازیابی کلمه عبور و... را می دهد، می باشد. این لینک ها زمانی ظاهر می شوند که شما ویژگی های PasswordRecoveryUrl و CreateUserUrl را تعیین کرده باشید.
InstructionTextStyle	متن کمکی که می توانید زیر کنترل Login اضافه کنید. بصورت پیش فرض کنترل Login هیچ متن راهنمایی ندارد.

```
<asp:login id="Login1" runat="server" backcolor="#EFF3FB" bordercolor="#B5C7DE" borderpadding="4" borderstyle="Solid" borderwidth="1px" font-names="Verdana" forecolor="#333333" height="256px" width="368px"

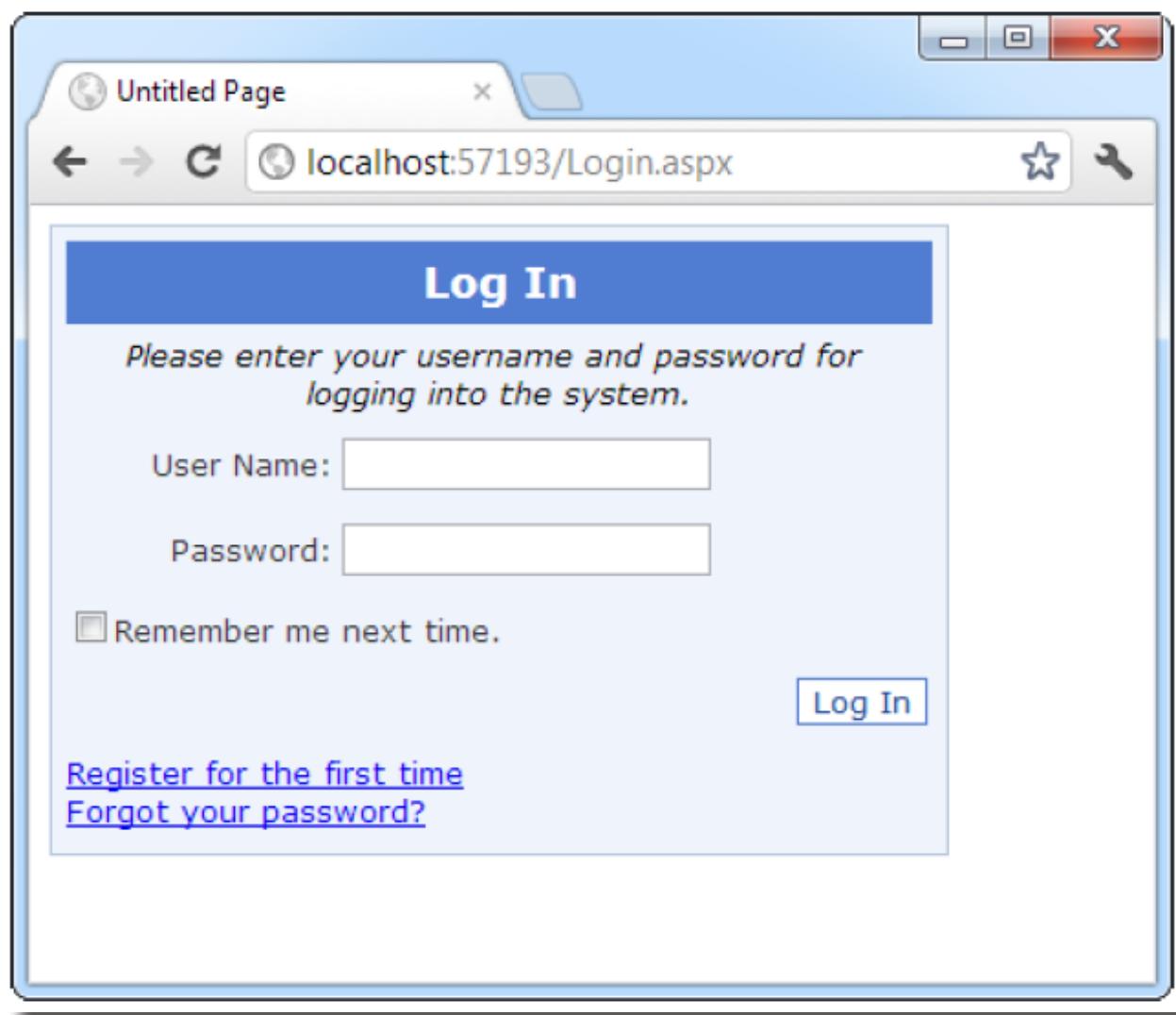
createusertext="Register for the first time" createuserurl="Register.aspx"

passwordrecoverytext="Forgot your password?" passwordrecoveryurl="PasswordRecovery.aspx"
```

```

instructiontext="Please enter your username and password for logging into the system.">
<TitleTextStyle BackColor="#507CD1" Font-Bold="True" Font-Size="Large">
ForeColor="White" Height="35px" />
<InstructionTextStyle Font-Italic="True" ForeColor="Black" />
<LoginButtonStyle BackColor="White" BorderColor="#507CD1">
BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
ForeColor="#284E98" />
</asp:login>

```



البته property های کاربردی دیگری نیز وجود دارد:

- **UsernameRequiredErrorMessage** و **PasswordRequiredErrorMessage** : متنی که رای کنترل validator در نظر گرفته می شود. می توان از علامت (\*) نیز استفاده کرد.

- آدرسی که کاربر پس از ورود موفق به آن هدایت می‌شود. بصورت پیش فرض این **DestinationPageUrl** خالی می‌باشد و این یعنی که کاربر به صفحه اصلی مورد درخواست خود یا صفحه‌ای که در فایل **web.config** تعیین شده است هدایت می‌شود.

- اگر برابر با **false** باشد، اگر کاربر قبل از **Login** کرده باشد، کنترل **Login** بصورت خودکار **VisibleWhenLoggedIn** مخفی می‌شود و اگر **true** باشد حتی در صورت **Login** بودن کاربر از قبل، این کنترل نمایش داده خواهد شد.

## کنترل **CreateUserWizard**

این کنترل که در بخش‌های قبلی اندکی با آن آشنا شدید دارای سه نوع از **property**‌ها می‌باشد:

- **TitleTextStyle** properties، که تنها بخشی از کنترل را فرمت دهی می‌کنند: برای نمونه **TitleTextStyle** تعیین می‌کند که متن هدر چگونه فرمت دهی شود.
- **Property**‌هایی که متنی را برای کنترل تعیین می‌کنند: برای نمونه توانید **label**‌ها، متن موفقیت یا شکست عملیات را تعیین کنید.
- **Property**‌هایی که بخشی از کنترل را مخفی می‌کند یا نمایش می‌دهد.

 نکته: بصورت پیش فرض، کاربر ثبت شده، **Login** نیز می‌شود که می‌توانید با تغییر مقدار **CreateUserWizard.Login** به این کار جلوگیری کنید.

می‌توانید مراحل بیشتری را به این کنترل اضافه نمایید. این مراحل می‌توانند کارهای دیگری را انجام دهنند مانند ثبت کاربر برای دریافت خبرنامه.

کد **Markup** مربوط به این کنترل بصورت ابتدایی در زیر آمده است:

```
<asp:createuserwizard id="CreateUserWizard1" runat="server">

<WizardSteps>
    <asp:CreateUserWizardStep runat="server">
    </asp:CreateUserWizardStep>
    <asp:CompleteWizardStep runat="server">
    </asp:CompleteWizardStep>
</WizardSteps>

</asp:createuserwizard>
```

یک کنترل ویزارد می‌باشد که از دو مرحله خاص پشتیبانی می‌کند: CreateUserWizard

- که محلی است که اطلاعات کاربر جمع آوری می‌شود و رکورد کاربر ایجاد می‌شود.

- محلی که پیام تأیید نمایش داده می‌شود.

در نمونه زیر، می‌بیند که چگونه می‌توانید یک WizardStep به این مراحل اضافه کنید. در اینجا مرحله اضافی، برخی از گزینه‌ها را برای کاربر تازه ثبت شده (ارسال خبرنامه و ...)، فراهم می‌کند.

```
<asp:createuserwizard id="CreateUserWizard1" runat="server" displaysidebar="True">

<WizardSteps>

<asp:CreateUserWizardStep runat="server" Title="Create User">
</asp:CreateUserWizardStep>

<asp:WizardStep runat="server" Title="Subscribe">

Would you like to sign up for the following newsletters?<br />
<br />

<asp:CheckBoxList ID="chkSubscription" runat="server">
<asp:ListItem>MSN Today</asp:ListItem>
<asp:ListItem>VB Planet</asp:ListItem>
<asp:ListItem>The High-Tech Herald</asp:ListItem>
</asp:CheckBoxList>
</asp:WizardStep>

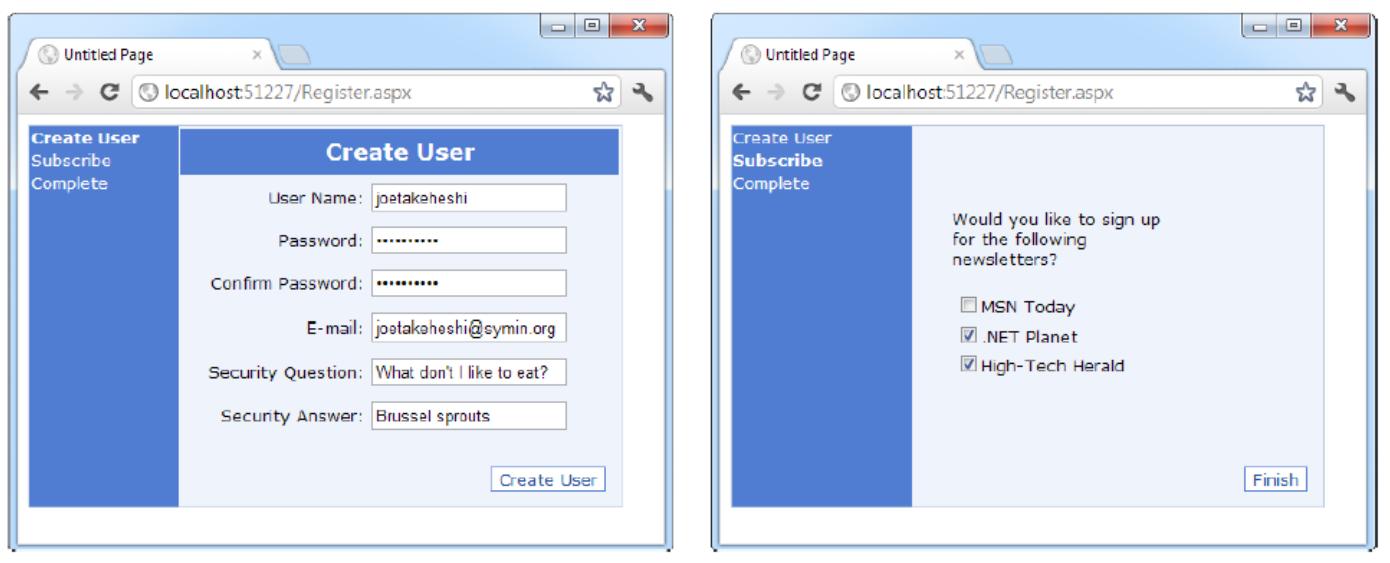
<asp:CompleteWizardStep runat="server">
</asp:CompleteWizardStep>

</WizardSteps>

</asp:createuserwizard>
```

با توجه به اینکه property با نام DisplaySidebar را با true برابر قرار دادیم، یک نوار کناری نمایش داده خواهد شد.

اید با پاسخ به یکی از رویدادهای CreateUserWizard، عمل مناسبی انجام دهیم. در اینجا از رویداد FinishButton- Click استفاده می‌کنیم که در مرحله آخر قبل از نمایش پیام تکمیل عملیات نشان داده خواهد شد.



ی توانید عکس العملی مناسب در کد به یکی از رویداد های CreateUserWizard نشان دهید. در این مورد، شما از رویداد Finish-ButtonClick استفاده می کنید، زیرا در آخرین مرحله قبل از پیام تکمیل عملیات رخ می دهد.

```
<asp:CompleteWizardStep ID="CompleteWizardStep1" runat="server">

    <ContentTemplate>

        <table border="0" style="">
            <tr>
                <td align="center" colspan="2" style="">
                    Complete
                </td>
            </tr>
            <tr>
                <td>
                    Your account has been successfully created.<br />
                    <br />
                    You subscribed to:
                    <asp:Label ID="lblSubscriptionList" runat="server">
                    </asp:Label>
                </td>
            </tr>
        </table>
    </ContentTemplate>
</asp:CompleteWizardStep>
```

```

<tr>

    <td align="right" colspan="2">
        <asp:Button ID="ContinueButton" runat="server"
            BackColor="White" BorderColor="#507CD1"
            BorderStyle="Solid" BorderWidth="1px"
            CausesValidation="False" CommandName="Continue"
            Font-Names="Verdana" ForeColor="#284E98" Text="Continue" ValidationGroup="CreateUserWizard1" />
    </td>
</tr>
</table>
</ContentTemplate>
</asp:CompleteWizardStep>

```

اکنون، زمانی که کاربر به مرحله آخر می‌رود، می‌توانید Label را با اطلاعات کنترل CheckBoxList پر کنید. به دلیل آنکه درون یک template قرار گرفته است، نمی‌توانید با استفاده از نام، مستقیماً به آنها دسترسی داشته باشید. در عوض می‌توانید آنها را از درون کنترل CreateUserWizard، پیدا کنید. برای گرفتن و دسترسی به Label، باید به مرحله دسترسی داشته باشیم، اولین کنترلی که شامل می‌شود را بدست آورده (content template) و سپس با متده Complete FindControl برای جستجوی Label استفاده کنید.

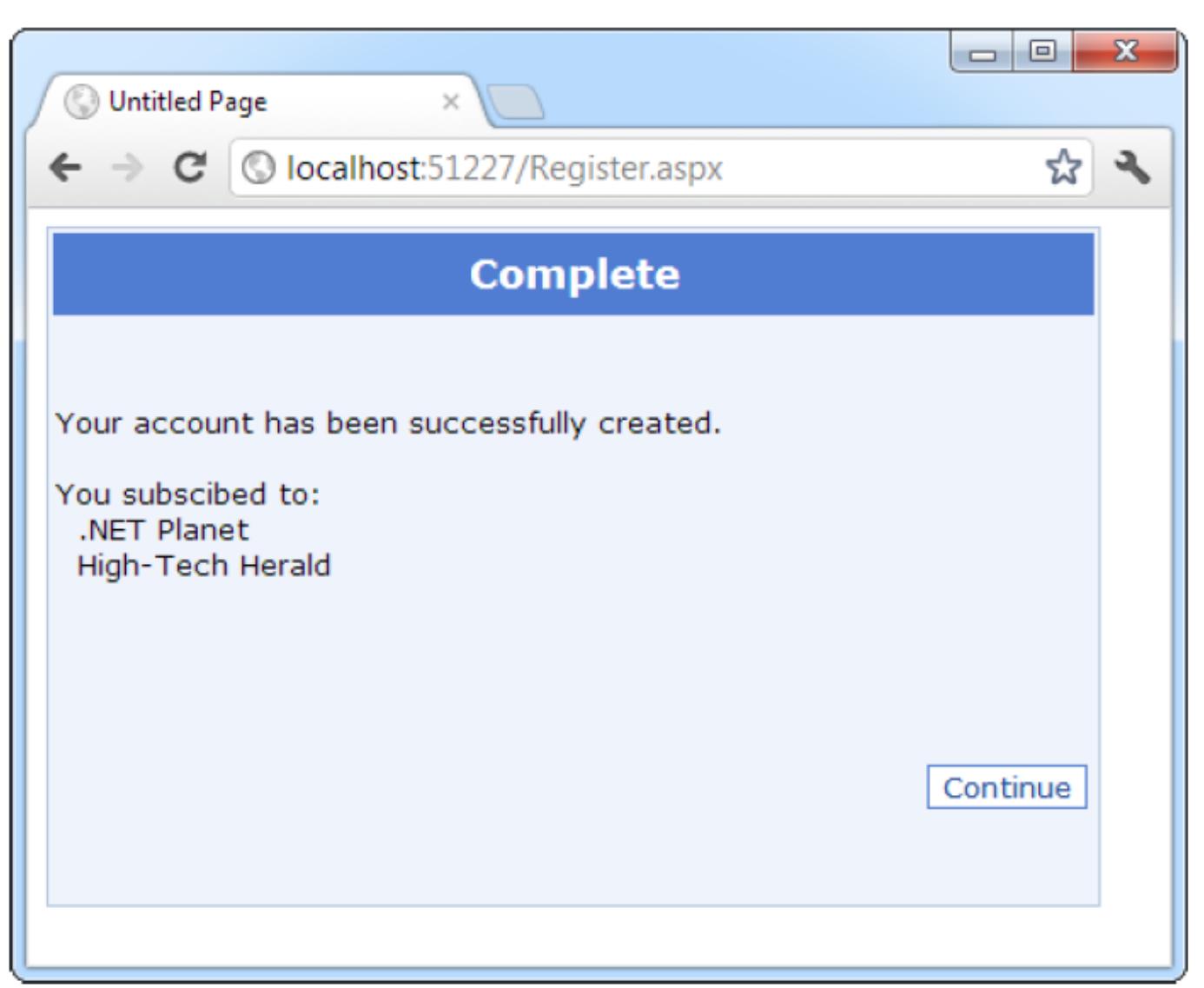
```

protected void CreateUserWizard1_FinishButtonClick(object sender, WizardEventArgs e)
{
    Label lbl = (Label)CreateUserWizard1.CompleteStep.Controls[0].FindControl(
        "lblSubscriptionList");

    CheckBoxList chk = (CheckBoxList)CreateUserWizard1.FindControl(
        "chkSubscription");

    string selection = "";
    foreach (ListItem item in chkSubscription.Items)
    {
        if (item.Selected)
            selection += "<br />" + item.Text;
    }
    lbl.Text = selection;
}

```



## کنترل PasswordRecovery

این کنترل، عمل بازیابی کلمه عبور کاربر را در سه مرحله انجام می‌کند. سپس سؤال امنیتی را نشان می‌دهد و پاسخ را از شما می‌خواهد. در نهایت یک E-Mail، به کاربر می‌زند. اگر شما از فرمت کلمه عبور Clear Encrypt یا استفاده کرده باشید، E-Mail، شامل کلمه عبور اصلی می‌باشد. اگر از فرمت کلمه عبور hashed استفاده کنید، یک کلمه عبور جدید ایجاد می‌شود و سپس E-Mail به کاربر ارسال می‌شود. در نهایت یک پیام که به شما اطلاع می‌دهد که E-Mail ارسال شده است، به شما نشان داده خواهد شد.

Untitled Page

localhost:57193/PasswordRecovery.as

## Forgot Your Password?

Enter your User Name to receive your password.

User Name:

Untitled Page

localhost:57193/PasswordRecovery.as

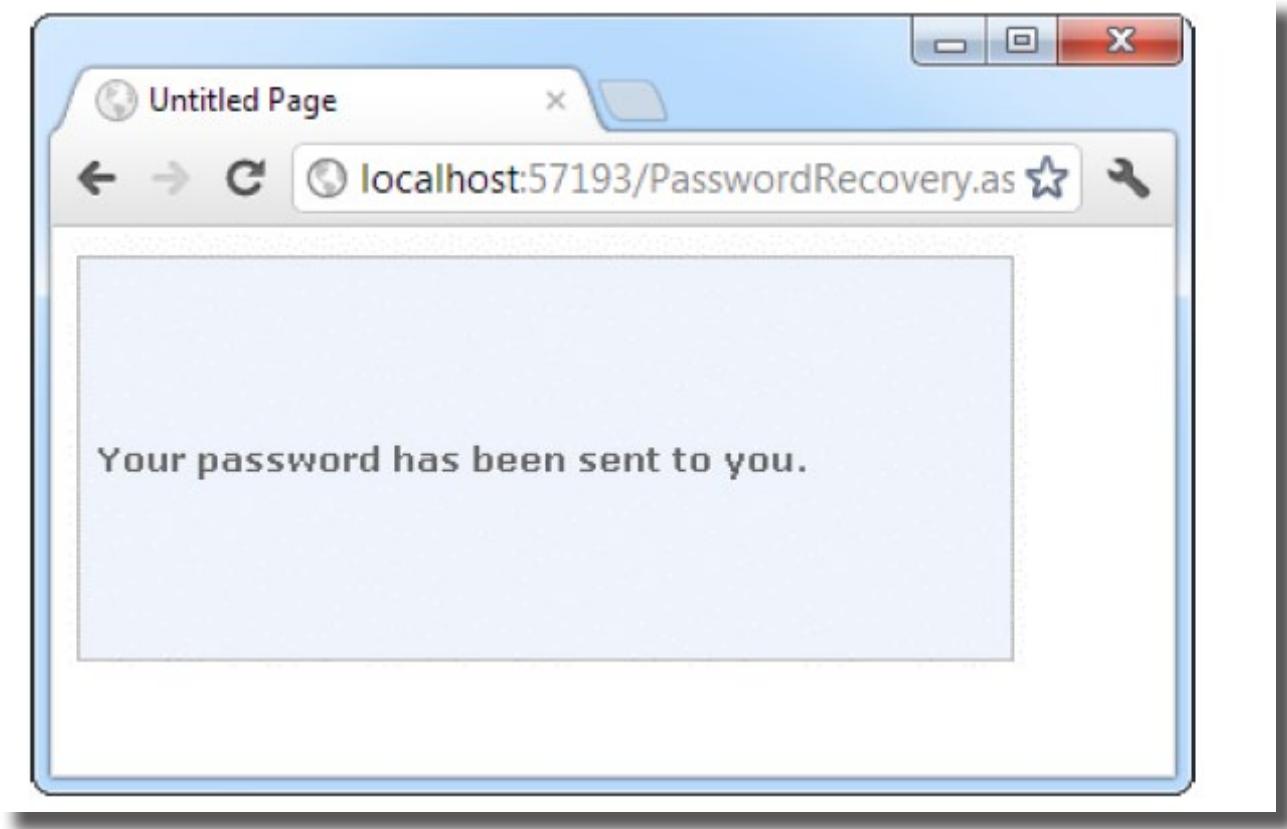
## Identity Confirmation

Answer the following question to receive your password.

User Name: liu\_liu

Question: What's your favorite pet's name?

Answer:

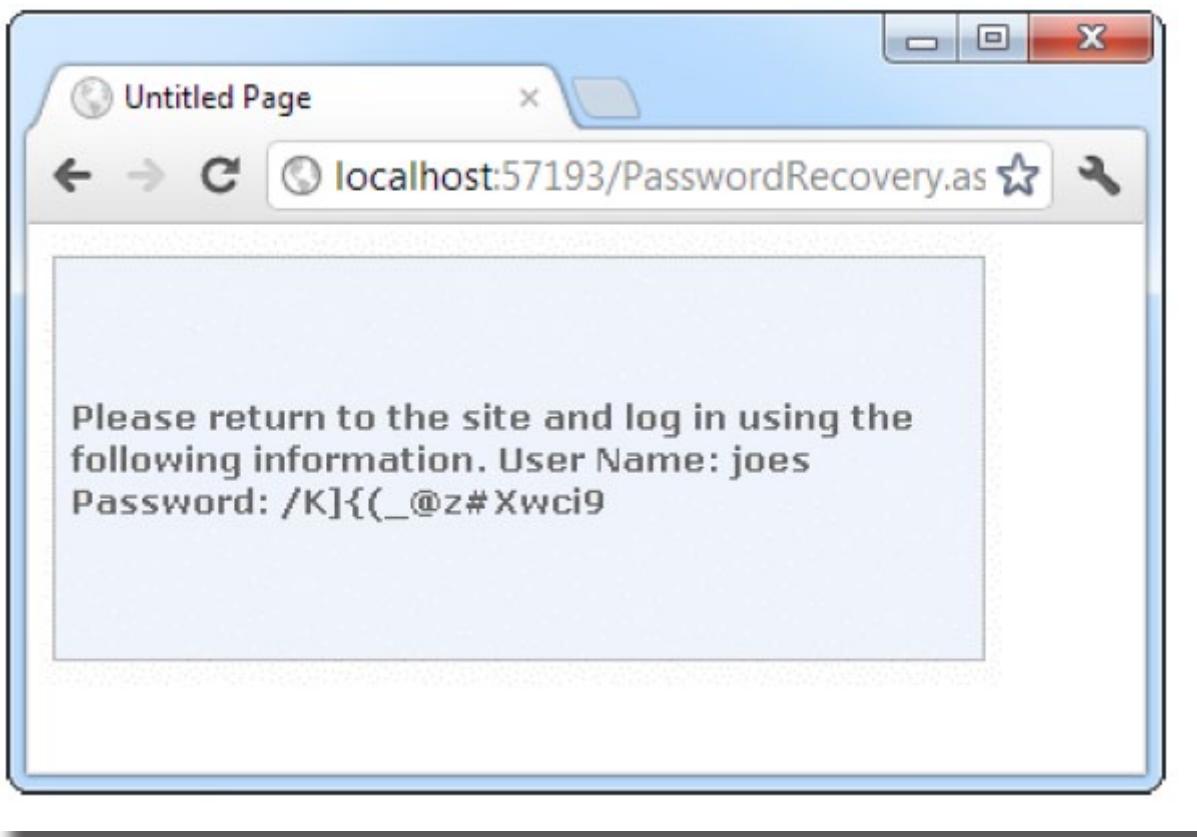


برای اینکه این کنترل کار خودش را انجام دهد، کامپیوتر شما باید دارای سرور SMTP با پیکربندی صحیح باشد و کاربر باید حتماً نشانی E-Mail داشته باشد.

 نکته: می‌توانید SMTP Server را با انتخاب کنترل PasswordRecovery و سپس انتخاب گزینه Administer Web-Configurable SMTP E-Mail را انتخاب و روی لینک Configure SMTP E-Mail کلیک کنید. smart tag از Site Application، پیکربندی کنید. زبانه smart tag از Site Application را انتخاب و روی لینک Configure SMTP E-Mail کلیک کنید.

اگر برنامه شما این دو مورد را ندارد، نمی‌توانید e-mail e-mail بزنید. می‌توانید کلمه عبور را مستقیماً درون صفحه نشان دهید. برای این کار باید جلوی ارسال e-mail را بگیرید.

```
protected void PasswordRecovery1_SendingMail(object sender, MailMessageEventArgs e)
{
    e.Cancel = true;
    PasswordRecovery1.SuccessText = e.Message.Body;
}
```



## امنیت مبتنی بر نقش (Role – Based Security)

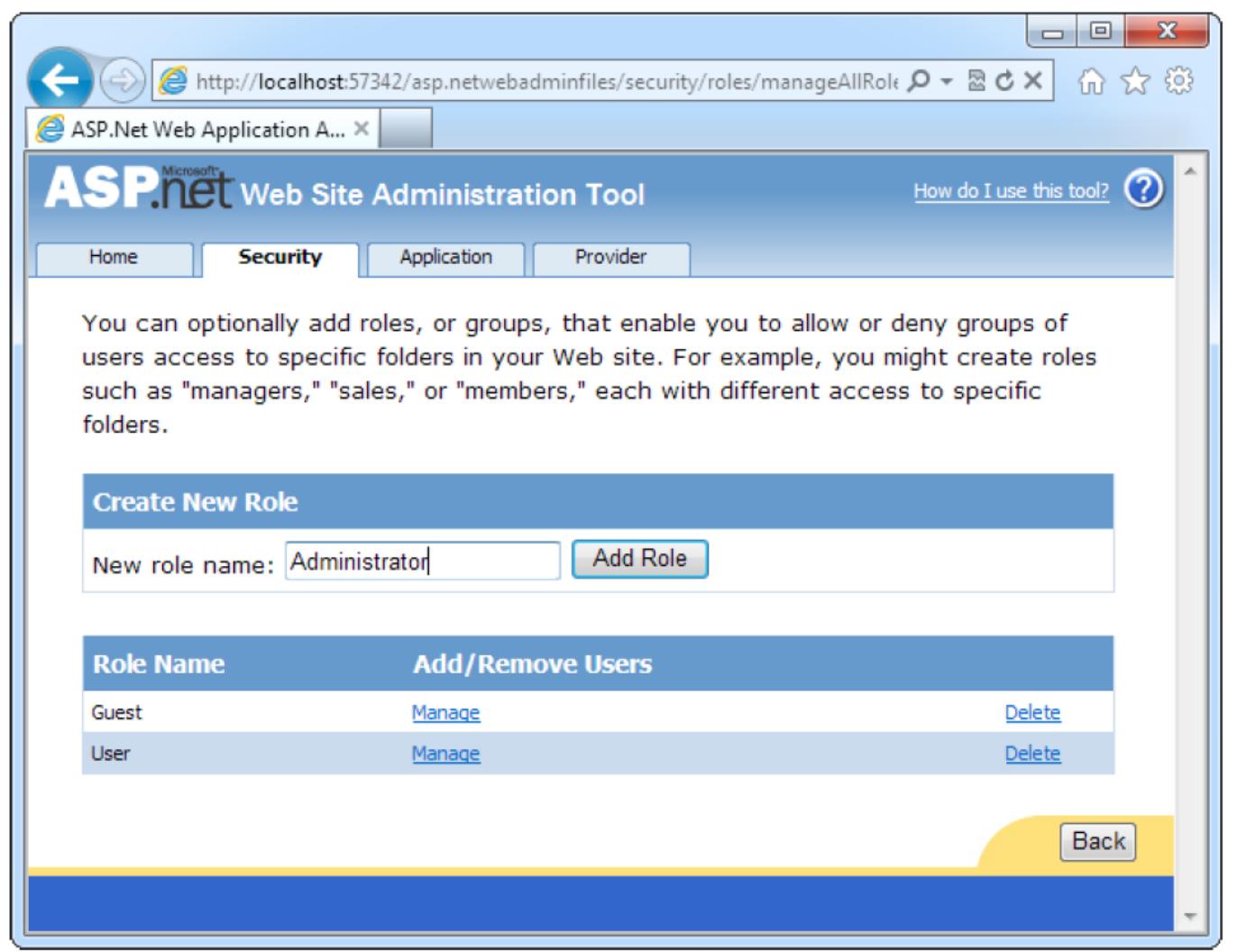
در بسیاری از موارد نیاز است تا برنامه سطوح مختلف کاربران را تشخیص دهد. برخی از کاربران ممکن است مجموعه ای از توانمندی های محدودتری داشته و برخی دیگر بتوانند تغییرات خطرناکی در سایت دهند یا برای اداره بخش هایی از سایت استفاده شوند.

برای ایجاد این دسترسی های چند سطحی، باید از ویژگی role-based authorization موجود در ASP.NET استفاده کنید. مانند membership، در اینجا نیز Mراقب ذخیره اطلاعات نقش ها و در دسترس قرار دادن آنها در کد می باشد. تمام کاری که شما باید انجام دهید این است که role را ایجاد و به یک کاربر نسبت دهید. قبل از استفاده از role-based authorization، باید آن را فعال کنید. این کار را با استفاده از WAT انجام دهید (روی لینک Enable Roles کلیک کنید)، یا در فایل web.config چند خط اضافه نمایید.

```
<configuration>
  ...
<system.web>
  ...
<roleManager enabled="true" />
</system.web>
</configuration>
```

## ایجاد و تخصیص نقش

۱. WAT را باز کنید.
۲. روی زبانه Security کلیک کنید.
۳. روی لینک Create or Manage Roles کلیک کنید.
۴. برای ایجاد نقش جدید، نام آن را وارد و روی Add Role کلیک نمایید.



- برای قرار دادن یک کاربر درون یک نقش روی دکمه back کلیک و در صفحه باز شده مراحل زیر را انجام دهید :
۱. از زبانه Security را انتخاب کنید. لیست تمامی کاربران سایت را می بینید.
  ۲. روی لینک Edit Role در کنار کاربر مورد نظر کلیک کنید.
  ۳. هر نقشی که می خواهید به این کاربر اختصاص دهید را انتخاب کنید.



البته می‌توانید بدون استفاده از Wat و از طریق کدنویسی نیز این کار را انجام دهید. کلاس Role همان کاری را برای انجام می‌دهد که کلاس membership management برای role management انجام می‌داد.

توضیحات	متدها
یک نقش جدید در بانک ایجاد می‌کند.	CreateRole()
نقش موجود در بانک را حذف می‌کند. اگر نقشی که کاربرانی به آن اختصاص داده شده اند را حذف می‌کنید، یا باید ابتدا کاربران را از درون نقش حذف کنید یا از overload دیگری از متدهای DeleteRole() استفاده کنید که پارامتر throwOnPopulatedRole (باید برابر با false شود) را می‌پذیرد.	DeleteRole()
چک می‌کند آیا نقشی با این نام وجود دارد یا نه.	RoleExists()
لیستی از تمامی نقش‌های این برنامه را برمی‌گرداند.	GetAllRoles()

متدهایی مانند زیر نیز وجود دارند :

AddUserToRole()	-
AddUserToRoles()	-
AddUsersToRole()	-
and AddUsersToRoles()	-
RemoveUserFromRole()	-
RemoveUserFromRoles()	-
RemoveUsersFromRole()	-
and RemoveUsersFromRoles()	-
IsUserInRole()	-
GetRolesForUser()	-
GetUsersInRole()	-

در این متدهای شبیه GetUsersInRole() و FindUsersInRole() می‌باشد، حتی اگر مواردی مشابه بخشه از متنی که به عنوان نام role نیز تایپ کرده اید پیدا شود، برمی‌گردد.

```

protected void CreateUserWizard1_CreatedUser(object sender, EventArgs e)
{
    Roles.AddUserToRole(CreateUserWizard1.UserName, "User");
}

```

The screenshot shows the Microsoft ASP.NET Web Site Administration Tool. At the top, there's a browser-like header with the URL <http://localhost:57342/asp.netwebadminfiles/security/users/manageUsers.aspx>. Below it, a title bar says "ASP.NET Web Application A...". The main content area is titled "ASP.NET Web Site Administration Tool". It has tabs for Home, Security (which is selected), Application, and Provider. A message at the top says: "Click a row to select a user and then click **Edit user** to view or change the user's password or other properties. To assign roles to the selected user, select the appropriate check boxes on the right." Another message below says: "To prevent a user from logging into your application but retain his or her information in your database, set the status to inactive by clearing the check box." A "Search for Users" section has a search bar with dropdowns for "Search by: User name" and "for:" and a "Find User" button. Below it, it says "Wildcard characters \* and ? are permitted." and lists letters A through Z with an "All" option. The main table lists users with columns for Active, User name, and Roles. The first user, "joes", has checkboxes for "Administrator" and "Guest", with "User" checked. The second user, "liu\_liu", has checkboxes for "Administrator" and "Guest", both unchecked. The third user, "test", has checkboxes for "Administrator" and "Guest", both unchecked. There are "Edit user", "Delete user", and "Edit roles" links for each user. A "Create new user" link is at the bottom left, and a "Back" button is at the bottom right.

## محدودیت دسترسی بر اساس نقش ها

- می توانید قوانین authorization ای بنویسید که دسترسی به صفحات یا زیر پوشه های تعیین شده را منوع می کنند.
- می توانید این قوانین را با اضافه کردن <authorization> به فایل web.config انجام دهید یا با استفاده از WAT آنها را تعریف کنید.
- می توانید از متد User.IsInRole() برای تعیین اینکه کاربر درون نقش خاصی قرار دارد یا نه استفاده کنید.
- می توانید از کنترل LoginView برای تعیین محتوای مختلف برای هر نقش استفاده کنید.

```
<authorization>
<deny users="?" />
<deny roles="Guest" />
<allow users="*" />
</authorization>
```

```
private void Page_Load(Object sender, EventArgs e)
{
    lblMessage.Text = "You have reached the secured page, ";
    lblMessage.Text += User.Identity.Name + ".";
    if (User.IsInRole("Administrator"))
    {
        lblMessage.Text += "<br /><br />Congratulations!";
        lblMessage.Text += "you are an administrator.";
    }
}
```

## کنترل LoginView

این کنترل یک کنترل نمایشی مانند MultiView یا Panel می‌باشد با این تفاوت که کاربر انتخاب نمی‌کند که View نمایش داده شود. view ها بر اساس وضعیت تأیید اعتبار کاربر تنظیم می‌شوند. ساده‌ترین استفاده از این کنترل، نمایش محتوای جداگانه برای کاربران تأیید اعتبار شده و ناشناس می‌باشد.

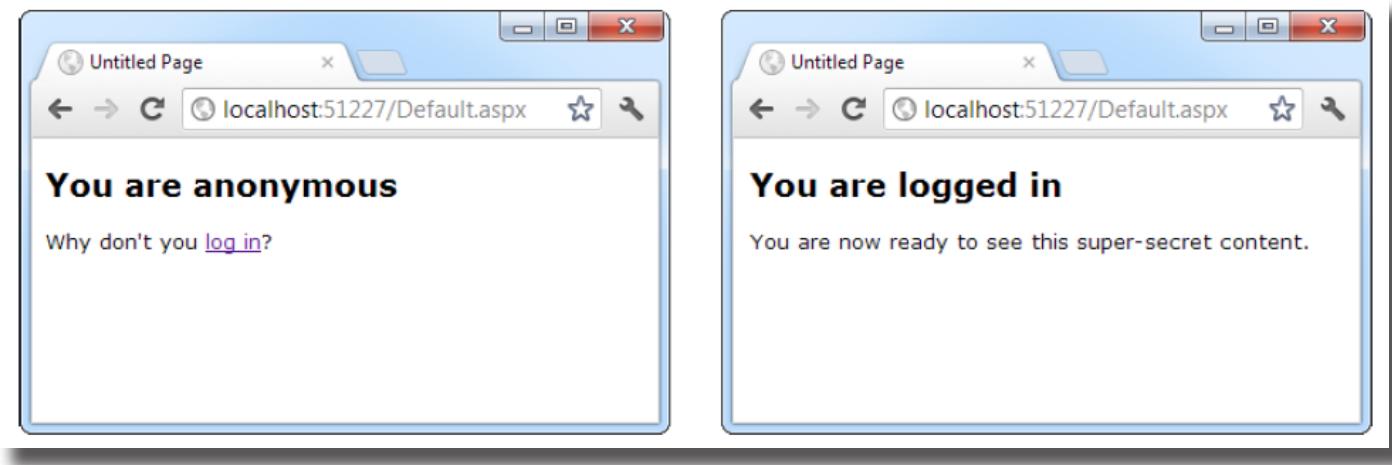
```
<asp:LoginView ID="LoginView1" runat="server">
    <AnonymousTemplate>
        <h1>
            You are anonymous</h1>
        Why don't you <a href="Login.aspx">log in</a>?
    </AnonymousTemplate>
    <LoggedInTemplate>
        <h1>
            You are logged in</h1>
```

&lt;p&gt;

You are now ready to see this super-secret content.&lt;/p&gt;

&lt;/LoggedInTemplate&gt;

&lt;/asp:LoginView&gt;



این کنترل، همچنین یک تگ دیگر با نام RoleGroups را پشتیبانی می‌کند. درون این تگ می‌توانید یک یا چند کنترل RoleGroup اضافه کنید. هر role group به یک یا چند role اختصاص داده شده است. در حقیقت شما template های گوناگونی برای کاربران مختلف بر اساس اینکه عضو کدام role هستند نمایش می‌دهد.

```
<asp:LoginView ID="LoginView1" runat="server">

    <AnonymousTemplate>

        <h1>
            You are anonymous</h1>
        Why don't you <a href="Login.aspx">log in</a>?

    </AnonymousTemplate>

    <RoleGroups>

        <asp:RoleGroup Roles="User, Guest">
            <ContentTemplate>
                <p>
                    If you can see this, you are a member of the User or Guest roles.</p>
            </ContentTemplate>
        </asp:RoleGroup>
    </RoleGroups>
</asp:LoginView>
```



```
<asp:RoleGroup Roles="Administrator">  
    <ContentTemplate>  
        <p>  
            Congratulations, you are an administrator.</p>  
    </ContentTemplate>  
</asp:RoleGroup>  
</RoleGroups>  
</asp:LoginView>
```

توجه داشته باشید که یک کاربر می‌تواند دارای بیش از یک role داشته باشد. به هر حال تنها یک template می‌تواند در یک زمان نمایش داده شود.

# بخش چهارم : امنیت و ب سایت

\_\_\_\_\_ فصل سیزدهم: اصول امنیت

\_\_\_\_\_ فصل پانزدهم: Membership

\_\_\_\_\_ فصل پانزدهم: Profile



## Profile

می‌توانید اطلاعات کاربران وب سایت را به روش‌های مختلف ذخیره نمایید. در فصل‌های قبل، تکنیک‌هایی مانند ViewState و cookie را ای نگهداری کوتاه مدت اطلاعات مشاهده کردید. اما اگر بخواهید اطلاعات را در طول مشاهده‌های مختلف نگهداری کنید، باید آن را در بانک اطلاعاتی ذخیره نمایید.

مشکل ذخیره در بانک این است که باید همه کدهای مورد نیاز بازیابی اطلاعات و ویرایش رکوردها را بنویسید. ASP.NET، دارای یک امکان می‌باشد که به شما اجازه کار با این روش ره به سادگی می‌دهد. این امکان Profiles نام دارد و برای ردیابی اطلاعات خاص کاربر بصورت خودکار، طراحی شده است.

زمانی که از profile‌ها استفاده می‌کنید، ASP.NET کارهایی مانند بازیابی اطلاعات و ویرایش بانک اطلاعاتی را در زمان تغییر آن، هندل خواهد کرد. نیازی به نوشتمن هیچ کد ADO.NET یا حتی طراحی جداول مربوطه نیست، زیرا ASP.NET مراقب همه جزئیات می‌باشد. ASP.NET Authentication Profile‌ها با Login نیز یکپارچه می‌باشند، بنابراین اطلاعات کاربر در دسترس کد صفحه وب شما می‌باشد.

یکی از محدودیت‌های Profile‌ها این است که در قبل برای نگهداری جزئیات خاص کاربر طراحی کرده اید جلوگیری می‌کند و اگر بخواهید از این اطلاعات در برنامه دیگر یا ابزار‌های گزارش گیری استفاده کنید، با مشکل روبرو می‌شوید.

## آشنایی با Profile

یکی از تفاوت‌های بین profile‌ها و سایر انواع مدیریت state، این است که آنها برای ذخیره دائمی اطلاعات با استفاده از یک بانک اطلاعاتی طراحی شده‌اند.

## کارایی Profile

هدف استفاده از Profile‌ها، ذخیره اطلاعات کاربر بدون اجبار به نوشتمن کد‌های دسترسی داده با استفاده از ADO.NET می‌باشد.

Profile‌ها به دو روش درون صفحات و قرار می‌گیرند:

- اولین باری به شی Profile در کد دسترسی پیدا می‌کنید، کل کاربر کنونی را از بانک اطلاعاتی بازیابی می‌کند. اگر اطلاعات Profile را بیش از یک مرتبه در یک درخواست خوانید، ASP.NET آن را یک مرتبه خوانده و سپس استفاده مجدد می‌کند، بنابراین از انجام کارهای اضافی روی بانک اطلاعاتی جلوگیری می‌شود.
- اگر داده‌ای از Profile را تغییر دهید، عملیات ویرایش تا زمانی که فرآیند صفحه به اتمام برسد به عقب می‌افتد. در این زمان (پس از اجرای رویدادهای PrerenderComplete و Unload)، درون بانک نوشته می‌شود.

Profile‌ها با ویژگی Caching یکپارچه نمی‌شوند، بنابراین هر درخواست که از داده Profile استفاده کند، نیازمند اتصال بانک اطلاعاتی می‌باشد.

از نظر کارایی، Profile ها زمانی بهتر کار می‌کنند که :

- تعداد کمی از صفحات دسترسی به داده های Profile داشته باشند.
- مقدار کمی از داده ها را ذخیره کنید.

اگر شرایط زیر برقرار باشد آنها کار کمتری می‌کنند:

- تعداد صفات زیادی دارید که نیاز به اطلاعات Profile داند.
- مقدار زیادی از داده را ذخیره می‌کنید.

البته می‌توانید Profile ها را با انواع دیگر state management ترکیب کنید. برای نمونه ابتدا داده ها را از Profile لود کنید و سپس آنها را در Session ذخیره نمایید.

## چگونه داده را ذخیره می‌کند Profile

Profile Provider موجود در ASP.NET، اطلاعات پروفایل را درون بلوکی از داده که در فیلدی در بانک اطلاعاتی وارد شده است را سریالیز می‌کند. برای نمونه اگر شما اطلاعات آدرس را سریالیز کنید، چیزی شبیه زیر می‌شود :

Marty Soren  
315 Southpart Drive  
Lompoc California 93436 U.S.A.

فیلد دیگر با فرمتی شبیه زیر تعیین می‌کند که هر مقدار د کجا شروع و پایان می‌یابد.

Name:S:0:11:Street:S:11:19:City:S:30:6:State:S:36:10:ZipCode:S:46:5:Country:S:51:6

این رشته، مقدار (name, street, city) و...، روش ذخیره سازی (String برای S) براي name, street, city است.

بنابراین اولین بخش این رشته تعیین می‌کند که اولین Property موجود در Profile برابر با name است که به صورت String ذخیره شده و از مکان • شروع می‌شود و ۱۱ کاراکتر طول دارد.

می‌توانید کدی بنویسید که داده Profile را تجزیه (Parse) کند و اطلاعات مورد نیاز شما را پیدا کند، اما بر اساس حجم داده و نوع داده مورد استفاده، این کار بسیار خسته کننده می‌باشد. حتی اگر هم این کار را انجام دهید، هنوز برای بازیابی آن محدودیت دارد.

به جای روش بالا می‌توانید برنامه ای ایجاد کنید که از serialization استاندارد Profile با SQLProfileProvider استفاده می‌کند و سپس آن را با Provider سفارشی خود تعویض کنید.

برای این تغییر می‌توانید تنظیماتی در web.config انجام دهید.

## استفاده از SQL ProfileProvider

به شما امکان ذخیره اطلاعات Profile را در بانک SQL Server می‌دهد. می‌توانید جداول Profile را در هر بانکی ایجاد کنید ولی نمی‌توانید جزئیات database schema مانند نام جداول، ستون‌ها و فرمت سریالیز کردن را تغییر دهید.

برای استفاده از profile، مراحل زیر را انجام دهید :

۱- Authentication را برای بخشی از وب سایت خود فعال سازید.

۲- SQL Server Express را پیکربندی کنید. (اگر از Profile Provider استفاده می‌کنید، این مرحله انتخابی می‌باشد)

۳- جداول profile را ایجاد کنید. (اگر از SQL Server Express استفاده می‌کنید، این مرحله نیاز نمی‌باشد)

۴- برخی از property‌های profile را تعریف کنید.

۵- از property‌ها درون کد صفحه وب خود استفاده نمایید.

## فعال سازی Authentication

به دلیل آنکه profile‌ها درون رکورد خاص کاربر ذخیره می‌شوند، باید کاربر کنونی را قبل از خواندن و نوشتمن اطلاعات تأیید اعتبار کنید. می‌توانید از windows-based authentication یا forms-based authentication برای این کار استفاده کنید.

```
<configuration>
  ...
<system.web>
  ...
<authentication mode="Windows"/>
<authorization>
  <deny users="?"/>
</authorization>
</system.web>
</configuration>
```

به دلیل اینکه در این مثال از windows authentication استفاده می‌کنید، نیازی به ایجاد رکورد برای هر کاربر ندارید. در عوض، از حساب کاربری ویندوزی موجود که در وب سرور تعریف شده است استفاده می‌کنید. این کار شما را از ایجاد صفحه Login نجات می‌دهد زیرا مرورگر خودش فرآیند Login را هندل می‌کند.

اگر بخواهید از forms authentication استفاده کنید، باید تعیین کنید که آیا می‌خواهید تأیید اعتبار را با استفاده از لیست کاربران سفارشی خود انجام دهید یا با استفاده از Membership.

در بیشتر موارد، membership و profile در رابطه با یکدیگر استفاده می‌شوند.

## استفاده از SQL Server Express

زمانی که از این نسخه استفاده می‌کنید، ASP.NET، اطلاعات profile را در یک فایل بانک اطلاعاتی با نام aspnetd.mdf که بصورت خودکار ایجاد شده است، ذخیره می‌کند. اگر این فایل ایجاد نشده باشد، در اولین باری که از membership profile یا استفاده می‌کنید، ایجاد می‌شود.

## استفاده از نسخه کامل SQL Server

اگر از نسخه‌ای به جز Express استفاده کنید، ایجاد بانک اطلاعاتی را باید خودتان بر عهده بگیرید و پیکربندی profile‌ها را در web.config انجام دهید.

تصویر پیش فرض، رشته اتصالی که توسط Profile استفاده می‌شود، LocalSqlServer نام دارد که می‌توانید مستقیماً در config آن را تغییر دهید یا در فایل Web.Config برنامه خود آن را پیکربندی کنید.

```
<configuration>
  <connectionStrings>
    <clear />
    <add name="LocalSqlServer" providerName="System.Data.SqlClient"
      connectionString="Data Source=localhost;Integrated Security=SSPI;
Initial Catalog=aspnetdb" />
  </connectionStrings>
  . . .
</configuration>
```

## بانک‌های اطلاعاتی Profile

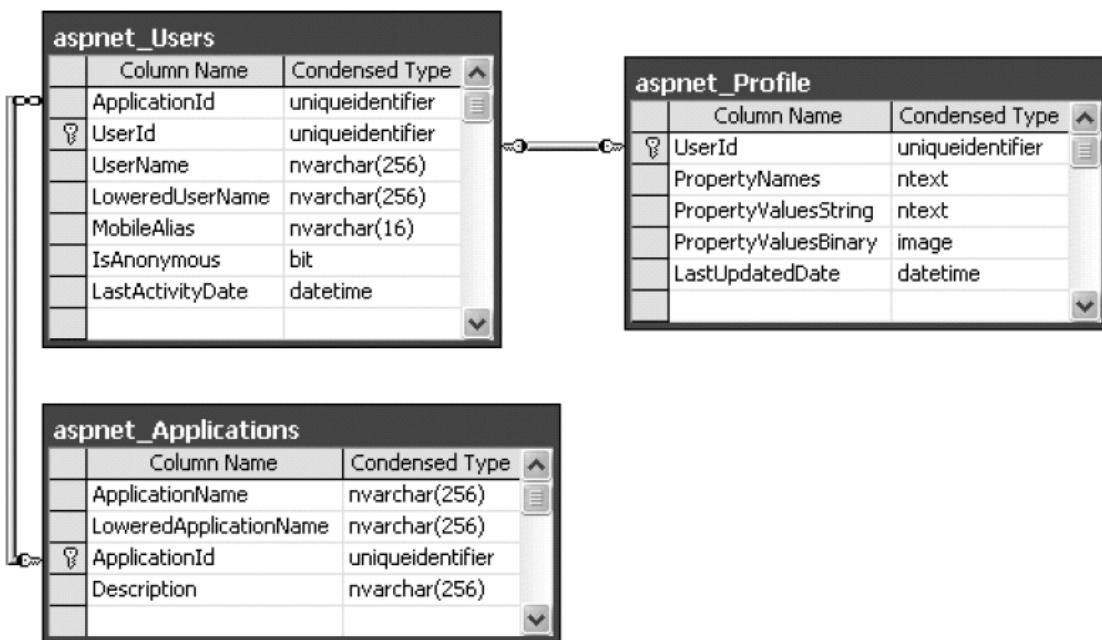
چه از ASP.NET برای ایجاد بانک اطلاعاتی استفاده کرده باشد و چه از نسخه Express استفاده کرده باشد و بانک را خودکار برای شما تولید کرده باشد، دارای جداول یکسانی خواهد بود.

همه اطلاعاتی که در یک profile ذخیره می‌کنید درون یک رکورد ترکیب می‌شوند و کاملاً درون فیلدی با نام PropertyValueString در جدولی با نام aspnet\_Profile قرار خواهند گرفت.

## جداول مورد استفاده برای profile



توضیحات	نام جدول
کلیه برنامه های تحت وب که در این بانک اطلاعاتی رکورد دارند را لیست می کند.	aspnet_Applications
اطلاعات profile کاربر را نمایش می دهد. هر رکود شامل اطلاعات کامل profile برای یک کاربر تنها می باشد. فیلد PropertyNames نام property ها را لیست می کند و فیلد propertyValuesString و propertyValuesBinary را لیست می کند. هر رکورد شامل تاریخ و زمان آخرین تغییر می باشد. البته استفاده از این اطلاعات در یک برنامه غیر ASP.NET ای کار سختی می باشد.	aspnet_Profile
های پشتیبانی شده برای ذخیره profile را ارائه می دهد.	aspnet_SchemaVersions
کلیه کلمات کاربری را لیست می کند و آنها را به یکی از برنامه های موجود در جدول aspnet_Applications نگاشت می کند.	aspnet_Users



## تعريف Profile Properties

قبل از آنکه بتوانید هیچ پروفایلی را ذخیره کنید، باید تعیین کنید که چه چیزی را می خواهید ذخیره کنید. اینکا را با اضافه کردن المان `<profile>` درون بخش `<properties>` از فایل `web.config` انجام دهید. درون المان `<properties>` می توانید تگ `<add>` را برای هر بخش خاص از اطلاعات کاربر که می خواهید ذخیره کنید، استفاده نمایید. المان `<add>` حداقل دارای نام property می باشد.

```

<configuration>
    ...
    <system.web>
        ...
        <profile>
            <properties>
                <add name="FirstName"/>
                <add name="LastName"/>
            </properties>
        </profile>
    </system.web>
</configuration>

```

معمولًا، انواع دادهای را نیز پشتیبانی می‌کنید. می‌توانید انواع داده ای قابل سریالیز شدن دات نت را در اینجا مشخص کنید:

```

<add name="FirstName" type="System.String"/>
<add name="LastName" type="System.String"/>
<add name="DateOfBirth" type="System.DateTime"/>

```

برای ایجاد property های پیشرفته تر از جدول زیر استفاده کنید:



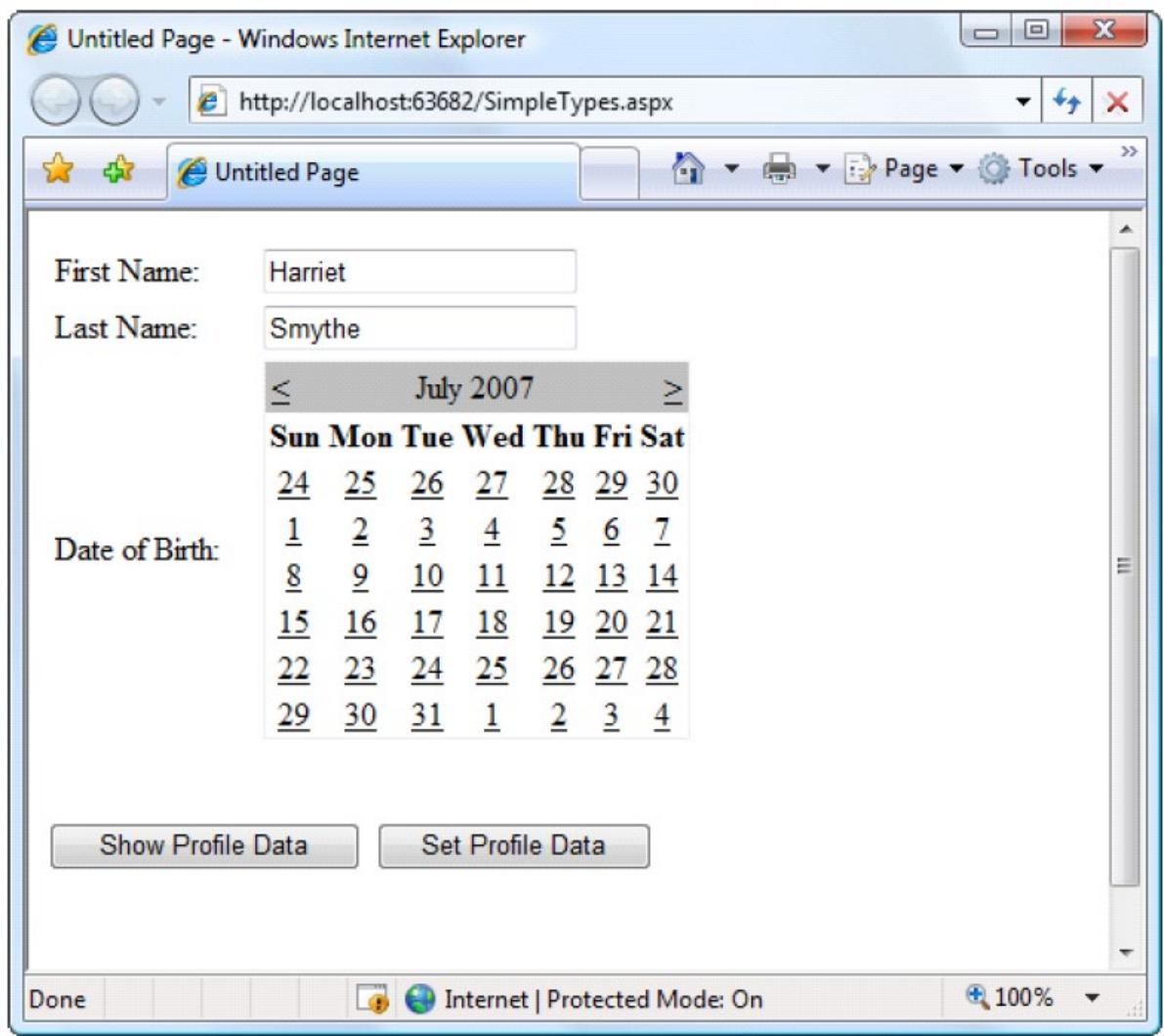
توضیحات	<add name="مان">
نام Property	name
نوع property را مشخص می‌کند که بصورت پیش فرض String است.	type
فرمت مورد نیاز برای سریالیز کردن این مقدار String, Binary, XML یا ProviderSpecific (اطلاعات بیشتر را در بخش سریالیز کردن پروفایل مشاهده نمایید).	serializeAs
یک مقدار Boolean، که تعیین می‌کند آیا یک مقدار قابل تغییر است یا نه. اگر true باشد، property خوانده می‌شود و قابل تغییر نیست. پیش فرض آن false می‌باشد.	readOnly
مقدار پیش فرضی که در صورت عدم وجود profile یا شامل نبودن این بخش از اطلاعات در profile، استفاده می‌شود.	defaultValue
یک مقدار boolean که تعیین می‌کند آیا این مقدار می‌تواند توسط پروفایل‌های ناشناس مورد استفاده قرار بگیرد یا نه. پیش فرض آن false است.	allowAnonymous
فرض، همه property ها با استفاده از provider تعیین شده در المان <profile> مدیریت می‌شوند ولی می‌توانید property های مختلفی را به provider های مختلف نسبت دهید.	provider

## استفاده از Profile Properties

با استفاده از این جزئیات، آماده دسترسی به اطلاعات Profile با بکارگیری ویژگی موجود در صفحه هستید. زمانی که برنامه خود را اجرا می‌کنید، یک کلاس که از System.Web.ProfileBase مشتق می‌شود را ایجاد می‌کند که مجموعه‌ای از تنظیمات Profile را شامل می‌شود.

اگر یک string property با نام FirstName تعریف کرده باشید، می‌توانید آن را در صفحه خود مقدار دهی کنید:

```
Profile.FirstName = "Henry";
```



اولین باری که صفحه اجرا می‌شود، هیچ اطلاعات profile ای بازیابی نمی‌شود و هیچ اتصال بانکی استفاده نمی‌شود. اگر روی دکمه کلیک کنید، اطلاعات Profile بازیابی می‌شود و درون صفحه نمایش داده می‌شود.

```
(protected void cmdShow_Click(object sender, EventArgs e
```

```

{
    lbl.Text = "First Name: " + Profile.FirstName + "<br />" +
    "Last Name: " + Profile.LastName + "<br />" +
    "Date of Birth: " + Profile.DateOfBirth.ToString("D");
}

```

اگر پروفایل نابود شده باشد یا اتصال به بانک باز نشود، خطأ رخ می‌دهد.

 نکته: property های پروفایل، مانند متغیرهای عضو کلاس عمل می‌کنند. این یعنی، اگر مقدار یک profile یک مقداردهی نشده بود را خواندید، مقدار پیش فرض آن را دریافت می‌کنید. (مانند رشته خالی یا صفر)

```

protected void cmdSet_Click(object sender, EventArgs e)
{
    Profile.FirstName = txtFirst.Text;
    Profile.LastName = txtLast.Text;
    Profile.DateOfBirth = Calendar1.SelectedDate;
}

```

زمانی که درخواست صفحه پایان یابد، اطلاعات پروفایل در بانک اطلاعاتی قرار گرفته است.

## Profile Serialization

اگر مقدار PropertyValueString برابر با Harriet FirstName مقدار Smyth LastName باشد، هردو مقدار درون فیلد az جدول aspnet\_Profile قرار خواهند گرفت.

HarrietSmyth

فیلد PropertyNames، اطلاعاتی درباره تجزیه هر مقدار از فیلد PropertyValueString در اختیار قرار می‌دهد. در زیر مقداری که در فیلد PropertyNames قرار می‌گیرد آورده شده است:

FirstName:S:0:7:LastName:S:7:6:

علامت (:) به عنوان جداکننده استفاده می‌شود.

PropertyName:StringOrBinarySerialization:StartingCharacterIndex:Length:

اگر شما یک پروفایل با یک نوع داده DateTime مانند زیر ایجاد کنید :

```
<add name="DateOfBirth" type="System.DateTime" serializeAs="String"/>
<add name="FirstName" type="System.String" serializeAs="Xml"/>
<add name="LastName" type="System.String" serializeAs="Xml"/>
```

درون فیلد PropertyValuesString مقدار زیر قرار می‌گیرد:

```
<?xml version="1.0" encoding="utf-16"?><dateTime>2007-07-12T00:00:00-04:00
</dateTime>HarrietSmythe
```

## گزینه های Serialization



تبدیل نوع به یک رشته. نیاز به یک تبدیل کننده نوع برای هندل کردن این کار دارد.	String
تایپ یا نوع مورد نظر را با استفاده از XML به System.XML.XMLSerializer تبدیل می‌کند که در یک رشته String ذخیره می‌شود.	Xml
ووع مورد نظر را با استفاده از System.Runtime.Serialization.Formatters.BinaryFormatter، به باینری تبدیل می‌کند. این نوع فشرده ترین و البته کم انعطاف ترین گزینه می‌باشد. این نوع از داده به جای فیلد PropertyValuesBinary در فیلد PropertyValues ذخیره می‌شود.	Binary
یک سفارشی را که در Serialization provider موجود است را پیاده می‌کند.	ProviderSpecific

برای نمونه در زیر می‌بینید که چگونه نوع Profile را Serializaation یک تغییر دهید :

```
<add name="DateOfBirth" type="System.DateTime" serializeAs="String"/>
<add name="FirstName" type="System.String" serializeAs="Xml"/>
<add name="LastName" type="System.String" serializeAs="Xml"/>
```

مقدار ذخیره چیزی شبیه زیر است :

```
2007-06-27<?xml version="1.0" encoding="utf-16"?><string>Harriet</string>
<?xml version="1.0" encoding="utf-16"?><string>Smythe</string>
```

اگر از BinarySerialization استفاده کنید، مقدار PropertyValuesString به جای فیلد PropertyValuesString درون فیلد PropertyValueBinary قرار می‌گیرد.

```
<add name="DateOfBirth" type="System.DateTime" serializeAs="String"/>
<add name="FirstName" type="System.String" serializeAs="Binary"/>
<add name="LastName" type="System.String" serializeAs="String"/>
```

تنها تفاوت، تبدیل S به B در فیلد PropertyNamesField می‌باشد(بایت‌های کمتری مورد نیاز است):

```
:DateOfBirth:S:0:9:FirstName:B:0:31:LastName:S:9:64
```

اگر Profile‌های یک Property را تغییر دهید چه اتفاقی می‌افتد؟ روشن‌سازی نمی‌کنند. می‌توانید Property‌هایی را اضافه یا کم کنید. برای نمونه، ASP.NET Property‌هایی که در جدول web.config تعریف شده اند ولی در فایل web.config تعریف نشده اند را نادیده می‌گیرد. همین طور اگر در فایل web.config تعریف شده باشد که در اطلاعات سریالیز شده پروفایل موجود نباشد، ASP.NET، تنها از مقدار پیش فرض استفاده خواهد کرد. بنابراین تغییر نام یا نوع Property یا سایر موارد می‌تواند در هنگام خواندن اطلاعات پروفایل موجب بروز خطا شود.

 نکته: تمامی انواع یا تایپ‌ها قابل سریالیز شدن نیستند. برای نمونه، کلاس‌ها، نمی‌توانند سازنده‌های بدون پارامتر که قادر به سریالیز شدن با فرمت XML باشند را ارائه دهند. کلاس‌هایی که Serializable attribute (خصوصیه) ندارند، می‌توانند با فرمت Binary سریالیز شوند.

## Profile Groups

اگر عدد زیادی از تنظیمات profile داشته باشید، و برخی از آنها به یکدیگر مرتبط باشند، می‌توانید برای سازماندهی بهتر آنها از Profile Groups استفاده کنید.

برای نمونه، در زیر اطلاعات تنظیمات واسط کاربری و اطلاعات سکونتی کاربر را در دو گروه تقسیم کرده ایم :

```
<profile>
  <properties>
    <group name="Preferences">
      <add name="LongDisplayMode" defaultValue="true" type="Boolean" />
      <add name="ShowSummary" defaultValue="true" type="Boolean" />
    </group>
    <group name="Address">
      <add name="Name" type="String" />
      <add name="Street" type="String" />
      <add name="City" type="String" />

      <add name="ZipCode" type="String" />
      <add name="State" type="String" />
      <add name="Country" type="String" />
    </group>
  </properties>
</profile>
```

&lt;/profile&gt;

اکنون از طریق نام گروه می‌توانیم به مقدار Property دسترسی داشته باشیم :

lblCountry.Text = Profile.Address.Country;

## پروفایل ها و انواع داده ای سفارشی

استفاده از یک کلاس سفارشی با پروفایل ساده است. باید یک کلاس که اطلاعات مورد نیاز را نگهداری می‌کند ایجاد کنید. می‌توانید از عضای Public استفاده کنید.

```
[Serializable()]
public class Address
{
    public string Name { get; set; }
    public string Street { get; set; }
    public string City { get; set; }
    public string ZipCode { get; set; }
    public string State { get; set; }
    public string Country { get; set; }

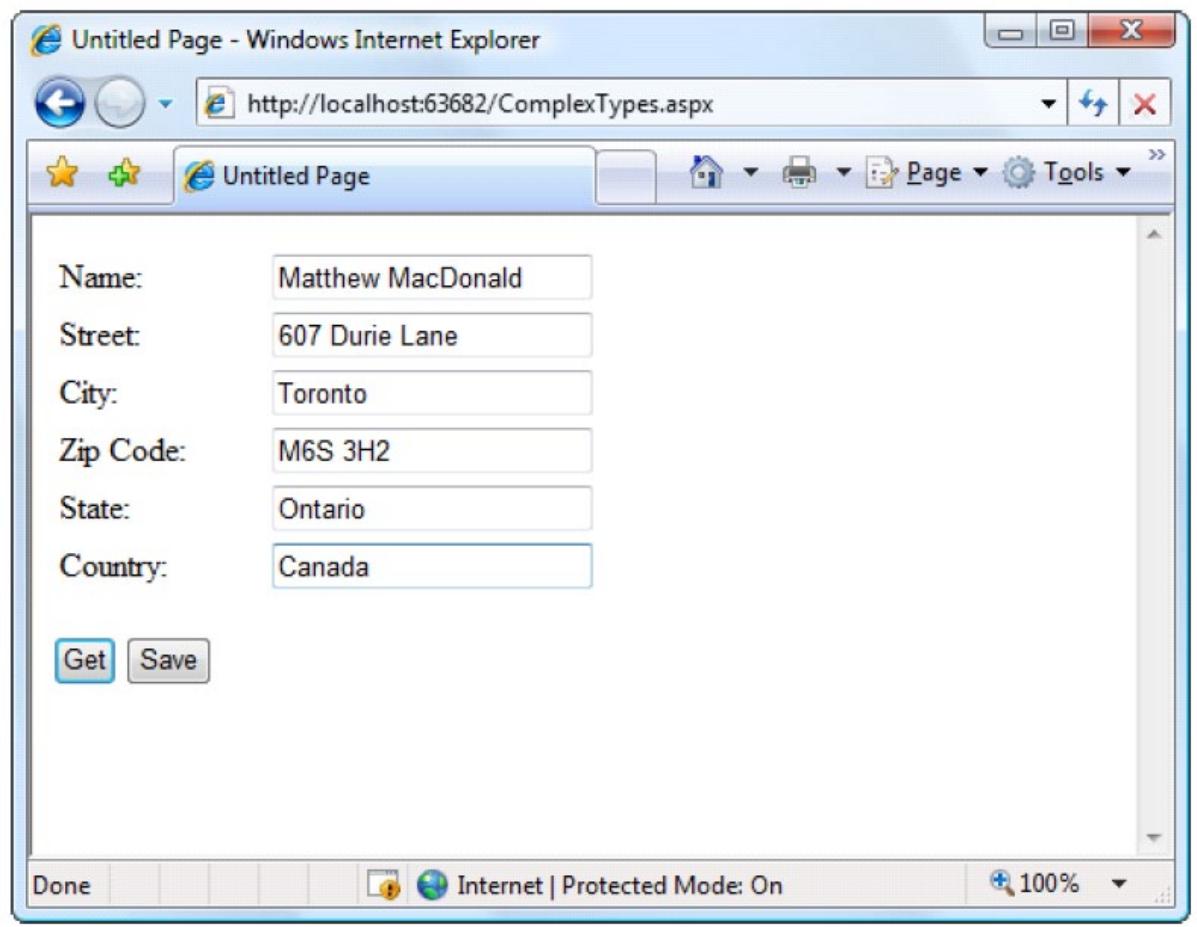
    public Address(string name, string street, string city,
        string zipCode, string state, string country)
    {
        Name = name;
        Street = street;
        City = city;
        ZipCode = zipCode;
        State = state;
        Country = country;
    }

    public Address()
    { }
}
```

می‌توانید این کلاس را در پوشه App\_Code قرار دهید. مرحله نهایی اضافه کردن یک Property که از آن استفاده می‌کند می‌باشد:

```
<properties>
  <add name="Address" type="Address" />
</properties>
```

اکنون می‌توانید یک صفحه تست که از کلاس Address استفاده می‌کند، ایجاد نمایید.



```
public partial class ComplexTypes : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
            LoadProfile();
    }

    protected void cmdGet_Click(object sender, EventArgs e)
    {
```

```

LoadProfile();

}

private void LoadProfile()
{
    txtName.Text = Profile.Address.Name;
    txtStreet.Text = Profile.Address.Street;
    txtCity.Text = Profile.Address.City;
    txtZip.Text = Profile.Address.ZipCode;
    txtState.Text = Profile.Address.State;
    txtCountry.Text = Profile.Address.Country;
}

protected void cmdSave_Click(object sender, EventArgs e)
{
    Profile.Address = new Address(txtName.Text,
        txtStreet.Text, txtCity.Text, txtZip.Text,
        txtState.Text, txtCountry.Text);
}
}

```

- زمانی که صفحه لود می شود، اطلاعات profile های مختلف کپی می شود. متدهای LoadProfile() این کار را انجام می دهد.
- کاربر می تواند مقدار address textbox را درون Save کلیک روی دکمه تغییرات تا زمانی که کاربر روی دکمه کلیک نکند اعمال نمی شود.
- زمانی که دکمه Save کلیک شد، یک شی جدید Address با استفاده از سازنده کلاس ایجاد می شود. شی مورد نظر به داده های Profile.Address نسبت داده می شود.
- زمانی که درخواست به پایان برسد، شی Profile، بصورت خودکار درون بانک اطلاعاتی ذخیره می شود.

## Custome Type Serialization

بصورت پیش فرض، تمامی انواع داده ای سفارشی، از XML Serializer با بکارگیری XML Serialization استفاده می کنند.

```

<Address>
<Name> . . . </Name>
<Street> . . . </Street>

```

```

<City> . . . </City>
<ZipCode> . . . </ZipCode>
<State> . . . </State>
<Country> . . . </Country>
</Address>

```

در زمان Deserialize کردن کلاس، XMLSerializer باید قادر به یافتن پارامترهای public سازنده باشد. هیچ از Property های شما نمی‌تواند read-only باشد.

اگر بخواهید از فرمت باینری برای سریالیز کردن استفاده کنید، دات نت روش کاملاً متفاوتی را خواهد رفت :

```
<add name="Address" type="Address" serializeAs="Binary"/>
```

ASP.NET، با استفاده از property تمامی public های و private را سریالیز می‌کند.

## ذخیره خودکار

Profile، امکان تشخیص تغییر در انواع داده ای ترکیبی (هر چیزی به غیر از boolean، string، انواع داده ای عددی ساده، آنها) را ندارد و این یعنی اگر پروفایل شما شامل complex data type باشد، ASP.NET، کل پروفایل را در انتهای هر درخواست دسترسی به شی profile، ذخیره می‌کند.

این کار سربارهای غیرضروری دارد. برای بهینه کردن کارایی در هنگام کار با Complex type ها، چندین انتخاب دارید. گزینه اول، تعیین property مرتبط با خاصیت read-only در فایل web.config در فایل می‌باشد. (اگر می‌دانید که property، هرگز تغییر نخواهد کرد، گزینه دیگر، غیرفعال کردن autosave، با اضافه کردن خصیصه automaticSaveEnabled در المان profIrlr و تنظیم مقدار آن با false می‌باشد.

```
<profile defaultProvider="SqlProvider" automaticSaveEnabled="false"> . . . </profile>
```

اگر از این روش استفاده کنید، باید متده Save را برای اعمال تغییرات فراخوانی نمایید. در حالت کلی، این روش مناسب تر است زیرا به سادگی هر جا که لازم باشد متده Save را فراخوانی می‌کنید.

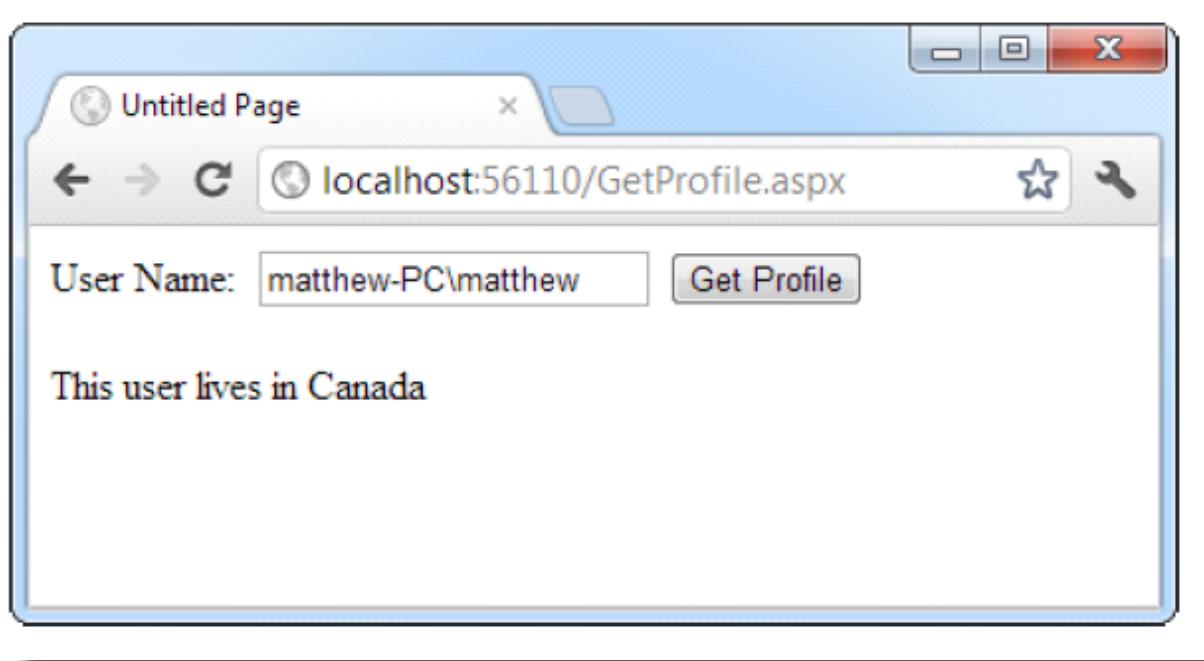
```
Profile.Address = new Address(txtName.Text, txtStreet.Text, txtCity.Text,
txtZip.Text, txtState.Text, txtCountry.Text);
Profile.Save();
```

برای نمونه، می‌توانید بگویید که تنها زمانی که اطلاعات address تغییر کرد، آن را ذخیره کن. بنابراین باید ذخیره خودکار را غیرفعال کنید و متده Save را در زمان کلیک روی دکمه TextBox.TextChanged فراخوانی کنید. همچنین می‌توانید رویداد profile.Save را برای تعیین مانی که تغییرات ایجاد می‌شود هندل کنید و اطلاعات profile را در آن زمان ذخیره کنید.

## Profile API

اگرچه صفحه شما بصورت خودکار اطلاعات profile را برای کاربر فعلی دریافت می‌کند، اما این کار باعث جلوگیری از شما برای بازیابی و تغییر پروفایل های سایر کاربران نمی‌شود. شما دو کلاس برای انجام این کار دارید: کلاس ProfileBase و کلاس ProfileManager.

شی GetProfile() شامل متده است که اطلاعات پروفایل یک کاربر را با استفاده از کلمه کاربری او برمی‌گرداند. مثال زیر یک کاربری که با حالت ویندوزی تائید اعتبار شده است را نشان می‌دهد.



```
protected void cmdGet_Click(object sender, EventArgs e)
{
    ProfileCommon profile = Profile.GetProfile(txtUserName.Text);
    lbl.Text = "This user lives in " + profile.Address.Country;
}
```

متده GetProfile() بر می‌گرداند. البته نمی‌توانید آن را در کتابخانه کلاس دات نت پیدا کنید، زیرا یک کلاس بصورت داینامیک ایجاد شده می‌باشد که اطلاعات پروفایل برنامه وب شما را نگهداری می‌کند.

```
protected void cmdGet_Click(object sender, EventArgs e)
{
    ProfileCommon profile = Profile.GetProfile(txtUserName.Text);
    if (profile.LastUpdatedDate == DateTime.MinValue)
```

```

{
    lbl.Text = "No user match found.";
}
else
{
    lbl.Text = "This user lives in " + profile.Address.Country;
}
}

```

اگر می خواهید سایر کارها را با پروفایل انجام دهید باید از ProfileManager در system.Web.Profile استفاده کنید. بسیاری از این متدها با کلاس ProfileInfo کار می کنند. این کلاس شامل Username, last update و last activity date می باشد.

**ProfileManager** متدهای

DeleteProfile() -

DeleteProfiles() -

DeleteInactiveProfiles() -

GetNumberOfProfiles() -

GetNumberOfInactiveProfiles() -

لیست پروفایل هایی که از مانی که شما تعیین کرده اید مورد استفاده قرار نگرفته اند.

GetAllProfiles() -

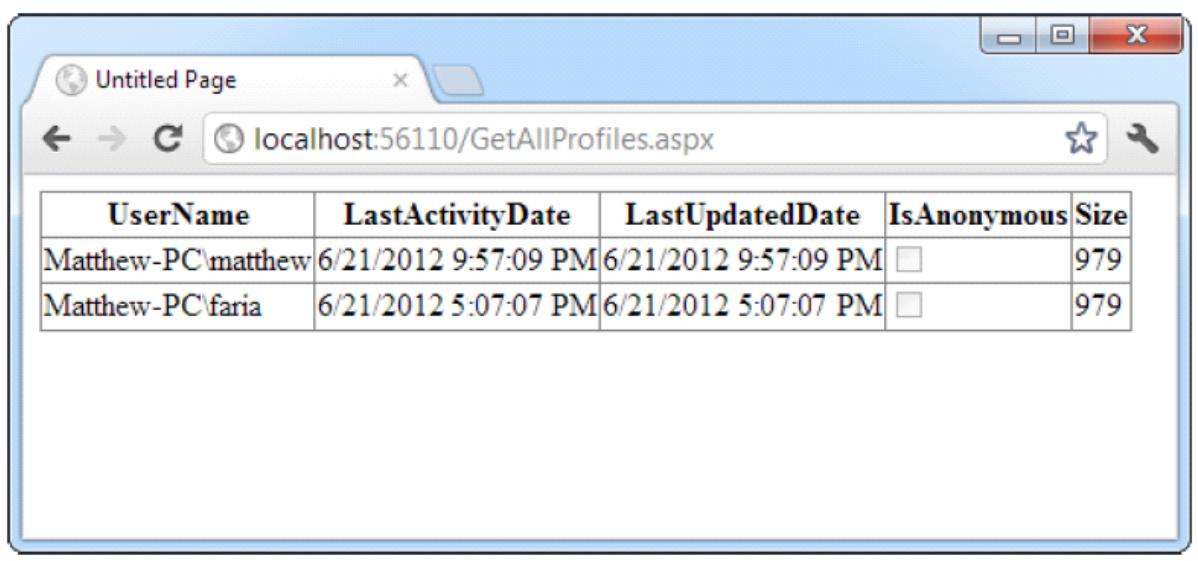
FindProfilesByUserName() -

FindInactiveProfilesByUserName() -

```
ProfileManager.DeleteProfile(User.Identity.Name);
```

برای نمایش کل اطلاعات پروفایل :

```
protected void Page_Load(object sender, EventArgs e)
{
    gridProfiles.DataSource = ProfileManager.GetAllProfiles(
        ProfileAuthenticationOption.Authenticated);
    gridProfiles.DataBind();
}
```



The screenshot shows a Microsoft Internet Explorer window titled "Untitled Page". The address bar displays "localhost:56110/GetAllProfiles.aspx". The main content area contains a grid of user profiles with the following data:

UserName	LastActivityDate	LastUpdatedDate	IsAnonymous	Size
Matthew-PC\matthew	6/21/2012 9:57:09 PM	6/21/2012 9:57:09 PM	<input type="checkbox"/>	979
Matthew-PC\faria	6/21/2012 5:07:07 PM	6/21/2012 5:07:07 PM	<input type="checkbox"/>	979

## Anonymous Profiles

معمولًاً قبل از دسترسی به اطلاعات کاربر، باید کاربر تائید شود. بعضی اوقات، بهترست یک پروفایل موقت برای کاربر جدید ناشناس ایجاد کنید. برای نمونه، بیشتر سایت‌های e-commerce، به کاربران جدید اجازه اضافه نمودن آیتم‌هایی به سبد خرید قبل از ثبت آنها در سایت را می‌دهد.

امکان تعیین هویت کاربر ناشناس را ارائه می‌دهد که یک شناسه تصادفی برای هر کاربر ناشناس ایجاد می‌کند که حتی اگر هیچ کلمه کاربری در دسترس نباشد، اطلاعات پروفایل را در بانک اطلاعاتی ذخیره می‌کند. کلمه عبور با استفاده از کوکی در سمت کلاینت دنبال می‌شود. (با اگر از حالت cookiless استفاده می‌کنید در URL پیگیری می‌شود). زمانیکه این کوکی ناپدید شود (برای نمونه زمانی که کاربر مرورگر را می‌بندد و دوباره باز می‌کند)، session ناشناس، گم می‌شود و یک anonymous session جدید ایجاد می‌شود.

```
<configuration>
  ...
<system.web>
  <anonymousIdentification enabled="true" />
  ...
</system.web>
<configuration/>
```

همچنین می‌توانید هر profile property که برای کاربران ناشناس باقی می‌ماند را با استفاده از یک فلگ مشخص کنید. این کار امکان ذخیره برخی از اطلاعات مورد نظرتان را به شما می‌دهد.

```
<properties>
  <add name="Address" Type="Address" allowAnonymous="true" />
  ...
<properties/>
```

## جابجایی Anonymous Profiles

یک چالش که توسط anonymous profile ها ایجاد می‌شود، نحوه برخورد با اطلاعات پروفایل، در زمانی که کاربر ناشناس قبلی login کرده است، می‌باشد. برای نمونه دیک وب سایت e-commerce، یک کاربر ممکن است، چندین آیتم را انتخاب و سپس برای تکمیل فرآیند خود ثبت نام یا login کند. در این زمان، باید مطمئن شوید که اطلاعات سبد خرید از پروفایل کاربر ناشناس به پروفایل کاربر تائید اعتبار شده منتقل خواهد شد.

خوب‌بختانه، ASP.NET، یک راه حل از طریق رویداد ProfileModule.MigrateAnonymous در اختیار قرار می‌دهد. این

رویداد زمانی fire می‌شود که یک شناسه anonymous در دسترس باشد(از طریق یک کوکی یا URL) و کاربر فعلی authenticate شده باشد. برای استفاده از رویداد MigrateAnonymous، باید یک هندر رویداد (متد) به فایل Global.asax اضافه کنیم.

یک تکنیک ابتدایی این است که پروفایل کاربر ناشناس را با استفاده از Profile.GetProfile() لود کنیم و شناسه AnonymousID را به آن پاس دهیم.

زمانی که این داده را لود کردید، می‌توانید بصورت دستی این تنظیمات را به پروفایل جدید منتقل کنید. در نهایت باید پروفایل anonymous را از بانک حذف و شناسه آن را از بین ببرید.

```
void Profile_MigrateAnonymous(Object sender, ProfileMigrateEventArgs pe)
{
    // Get the anonymous profile.

    ProfileCommon anonProfile = Profile.GetProfile(pe.AnonymousID);

    // Copy information to the authenticated profile
    // (but only if there's information there).

    if (!anonProfile.IsNullOrEmpty())
    {
        Profile.Address = anonProfile.Address;
    }

    // Delete the anonymous profile from the database.
    // (You could decide to skip this step to increase performance
    // if you have a dedicated job scheduled on the database server
    // to remove old anonymous profiles.)

    System.Web.Profile.ProfileManager.DeleteProfile(pe.AnonymousID);

    // Remove the anonymous identifier.

    AnonymousIdentificationModule.ClearAnonymousIdentifier();
}
```



در نقشه نیست، زبانش نلند.  
جالب ترین جاها در نقشه نیستند. « هرمان ملویل »

# بخش پنجم : ASP.NET پیشرفته

□ فصل شانزدهم: برنامه نویسی مبتنی بر کامپونت

□ ~~فصل هفدهم: Caching~~

□ ~~فصل هجدهم: Entity Framework , LINQ~~

□ ~~فصل نوزدهم: ASP.NET AJAX~~

□ ~~فصل بیست و یکم: Deploy کردن برنامه های~~



## برنامه نویسی Component-Based

این شیوه از کد نویسی به شما امکان سازماندهی، سازگاری و قابل استفاده مجدد بودن بیشتری می‌دهد. در دات نت پیاده سازی این روش بسیار ساده می‌باشد زیرا هرگز نیازی به استفاده از رجیستری ویندوز یا اجرای هرگونه پیکربندی خاص نمی‌باشد.

یک کامپوننت، در ساده ترین شکل خود، یک یا چند کلاس می‌باشد که درون یک فایل اسمبلی DLL قرار داده شده است. این کلاس‌ها برخی از کاربردهای مرتبط از نظر منطقی را پیاده سازی می‌کنند. می‌توانید از یک کامپوننت در یک برنامه تنها استفاده کنید یا آن را در اختیار چند برنامه قرار دهید. برنامه وب شما (یا هر برنامه تحت دات نت)، می‌تواند از کلاس‌های درون کامپوننت شما همانطور که کلاس‌های معمولی را بکار می‌گیرد، استفاده نماید. کامپوننت شما کپسوله شده است که این یعنی، امکاناتی که کد شما نیاز دارد را فراهم می‌کند اما کلیه جزئیات پیاده سازی را در خود مخفی خواهد کرد.

### چرا از کامپوننت‌ها استفاده می‌کنیم؟

برای پیشرفت در برنامه ASP.NET، باید توانایی استفاده از .NET Cass Library را بدست آورید.

تا کنون اگر می‌خواستید اطلاعاتی را از بانک اطلاعاتی مانند SQL Server، بازیابی کنید، باید جزئیات بانک اطلاعاتی (مانند SQL Query ها) را مستقیماً درون برنامه وب خود می‌آوردید. حتی آنها را در کلاس‌های code-behind یا درون صفحه کدهای صفحه markup (در صورت استفاده از aspx) قرار می‌دادید.

اکنون اگر ساختار بانک اطلاعاتی تغییر می‌کرد باید تمامی صفحات مرتبط را تغییر می‌دادید. برای حل این مشکل نیاز به یک لایه اضافی بین صفحات وب و بانک اطلاعاتی دارید که نوعی از کامپوننت سفارشی می‌باشد.

مورد بیان شده در بالا تنها یکی از دلایل استفاده از کامپوننت‌های سفارشی می‌باشد. سایر دلایل را در زیر می‌بینید:

**امنیت :** به دلیل آنکه source code درون صفحه وب نمی‌باشد، نمی‌توانید آن را تغییر دهید. برای نمونه می‌توانید کامپوننت بانک اطلاعاتی را طوری پیکربندی کنید که تنها با table, field و ردیف‌های خاصی کار کند. این کار اغلب آسان‌تر از تنظیم مجوزهای پیچیده روی بانک می‌باشد. به دلیل آنکه برنامه از طریق کامپوننت اجرا می‌شود، باید قوانین آن را اجرا کند.

**سازماندهی بهتر:** کامپوننت، آشفتگی و بی نظمی کدها را از برنامه شما درو می‌کند. همچنین با شکستن برنامه درون کامپوننت‌های جدا، آشنایی سایر برنامه نویس‌ها را با منطق برنامه شما ساده‌تر خواهد کرد. بدون کامپوننت‌ها کدهایی که مورد استفاده زیاد دارند، ممکن است در کل برنامه copy & paste شده باشند که تغییر و یکسان سازی آنها را مشکل می‌کند.

**رفع مشکل آسان‌تر :** برنامه‌هایی که از کامپوننت‌ها استفاده می‌کنند، به بلوک‌های کوچکتری از کد تقسیم می‌شوند و تشخیص محل دقیق خطأ در آنها کار ساده‌تری می‌باشد. همچنین تست یک کامپوننت بصورت مستقل از باقی برنامه وب شما، امکان پذیر می‌باشد.

**مدیریت پذیری بیشتر:** این نوع از برنامه‌ها ساده‌تر تغییر می‌کنند زیرا کامپوننت و کدهای برنامه وب را می‌توان جداگانه تغییر داد.

این روش به شما امکان قرار دادن یک تیم توسعه روی کامپوننت و یک تیم دیگر روی کدنویسی وب سایتی که از آن کامپوننت استفاده می‌کند را خواهد داد.

**استفاده مجدد کد:** کامپوننت‌ها می‌توانند با هر برنامه ASP.NET که به قابلیت‌های آنها نیاز دارند به اشتراک گذاشته می‌شوند. حتی هر برنامه تحت دات نت نیز می‌تواند از یک کامپوننت استفاده کند، یعنی می‌توانید یک کامپوننت ایجاد کنید که هم در یک برنامه تحت وب و هم در یک برنامه ویندوزی استفاده شود.

**سادگی :** کامپوننت ها می توانند چندین کار مرتبه را برای یک درخواست کلاینت انجام دهند (نوشتن چند رکورد در بانک، باز کردن و خواندن فایل در یک مرحله و حتی شروع و مدیریت transaction با نکه اطلاعاتی). همچنین به دلیل آنکه کامپوننت، جزئیات را مخفی می کند، یک برنامه نویس می تواند از database component استفاده کند، بدون آنکه نگران نام بانک، مکان قرار گیری سرور و استفاده از حساب کاربری برای اتصال به بانک، باشد. حتی می توانید با استفاده از یک criteria، یک موتور جستجوگر را درون کامپوننت پیاده سازی کنید و کامپوننت خودش تصمیم بگیرد که آیا از یک ایجاد کننده داینامیک دستورات SQL استفاده کند یا از . storeprocedure

## طراحی بر اساس کامپوننت

### طراحی سه لایه (three-tier)

لایه اول، user interface (واسط کاربری) یا presentation می باشد که کنترل ها را نمایش می دهد و ورودی های کاربر را دریافت و اعتبارسنجی می کند. تمامی رویدادهای هندل شده در صفحات وب در این لایه قرار دارند. سطح یا لایه دوم، business tier می باشد که منطق اصلی برنامه در آن قرار می گیرد. برای یک سایت e-commerce، منطق خاص برنامه شامل قوانینی مانند اینکه چه عمل هایی از کاربر باید در سیستم ثبت (log) شود و ... می باشد. این لایه امل جزئیات عمومی دات نت مانند اینکه چگونه یک فایل را باز کنیم یا به بانک اطلاعاتی متصل شویم، نمی باشد. سطح سوم، data tier می باشد که منطق ذخیره سازی اطلاعات شما در فایل، بانک اطلاعاتی یا سایر منابع ذخیره سازی را در خود نگهداری می کند. همچنین شامل منطق چگونگی بازیابی و ویرایش داده مانند SQL Queries یا store Procedure می باشد.

 نکته مهم درباره طراحی سه لایه، این است که اطلاعات تنها از یک سطح به سطح مجاور می تواند حرکت کند. به عبارت دیگر، کد صفحه وب شما نمی تواند برای بازیابی اطلاعات مستقیماً به بانک اطلاعاتی وصل شود. در عوض، از طریق کامپوننت در business tier، می توانید به بانک اطلاعاتی متصل و داده را بازیابی کنید.

## کپسوله سازی (Encapsulation)

شما فقط کافی است متدى را از درون یک کامپوننت فراخوانی کنید. سایر کارها از دید شما مخفی می ماند و نتیجه به شما بر می گردد.

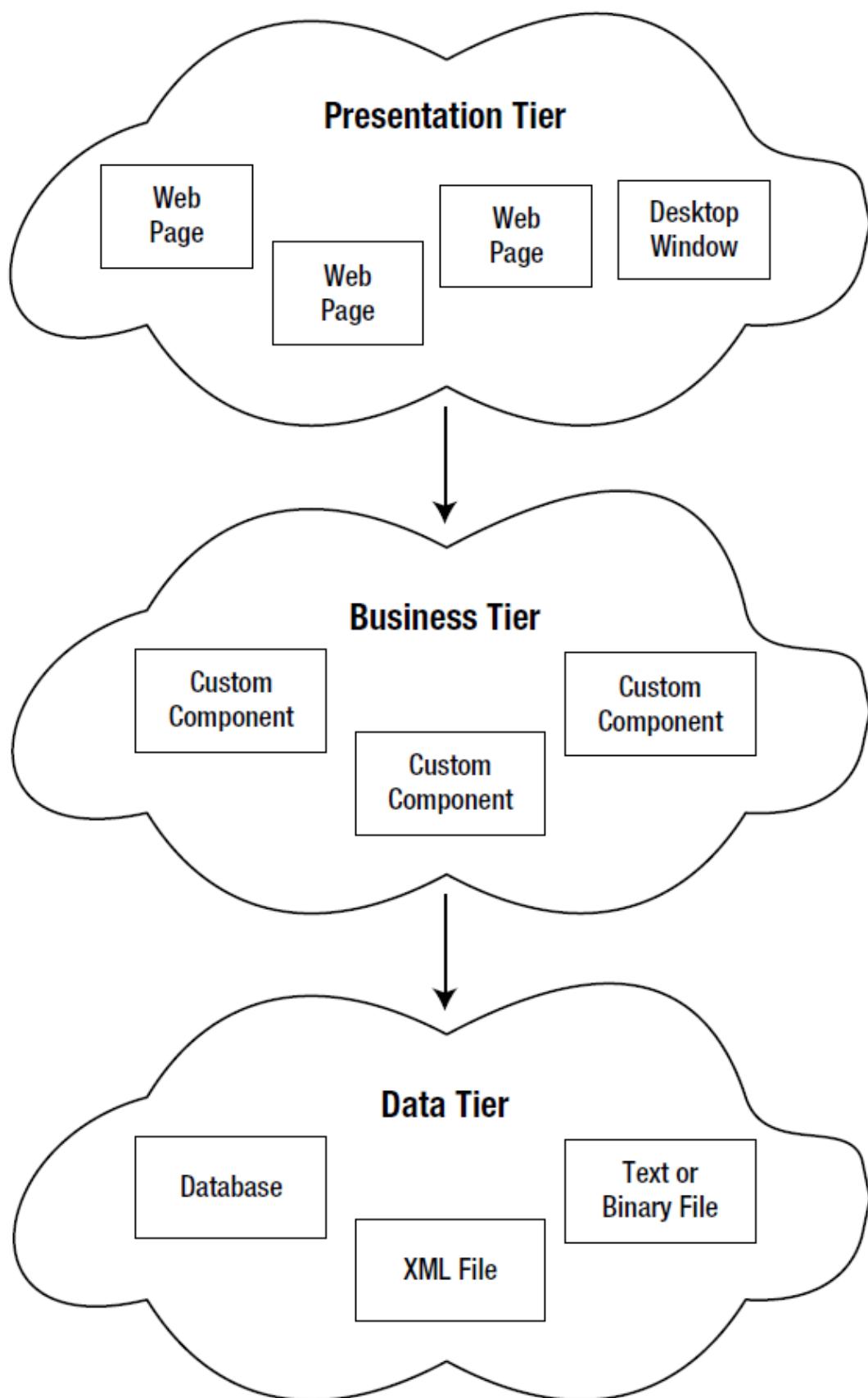
## Business Objects

Business object ها، کامپوننت هایی در لایه دوم برنامه شما هستند که یک لایه اضافی بین منبع داده و کد شما فراهم می کنند. به این دلیل که قوانین business ای را پیاده می کنند به آنها business object می گویند. برای نمونه، اگر بخواهید یک درخواست خرید که شامل هیچ آیتمی نمی باشد را بفرستید، BO مرتبط، یک خطای exception را throw می کند و ادامه کار را متوقف خواهد کرد.

## Data Objects

واژه data object، برای موارد مختلف بکار می رود ولی در این کتاب، بسته داده ای می باشد که برای ارسال اطلاعات بین صفحات وب

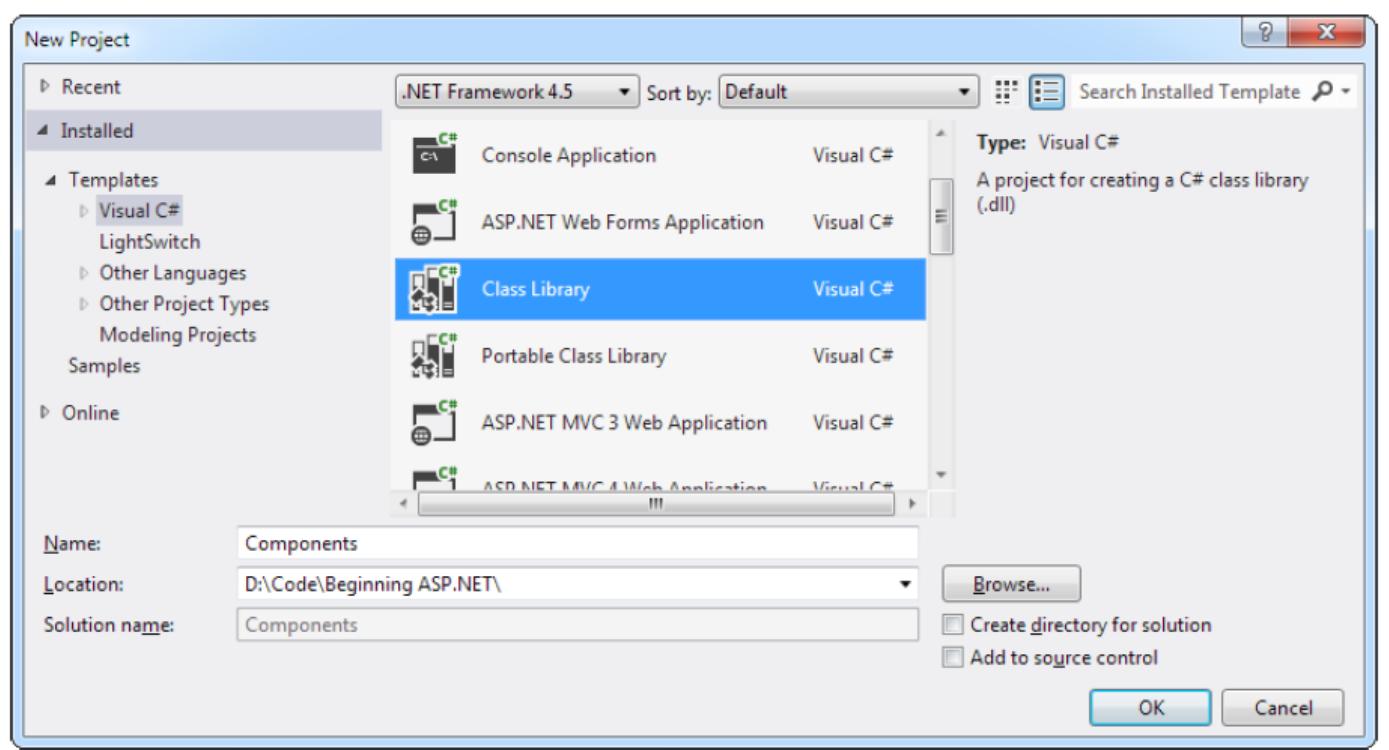
business object ها استفاده می‌شود. برای نمونه، می‌توانید یک کلاس Employee با نام data که اطلاعات یک رکورد در جدول Employee را ارائه می‌دهد، ایجاد کنید و آن را با property هایی مانند FirstName, LastName و DateofBirth پر کنید. معمولاً data object دارای property هایی می‌باشد ولی شامل هیچ متادی نیست.



## کلاس‌ها و کامپوننت‌ها

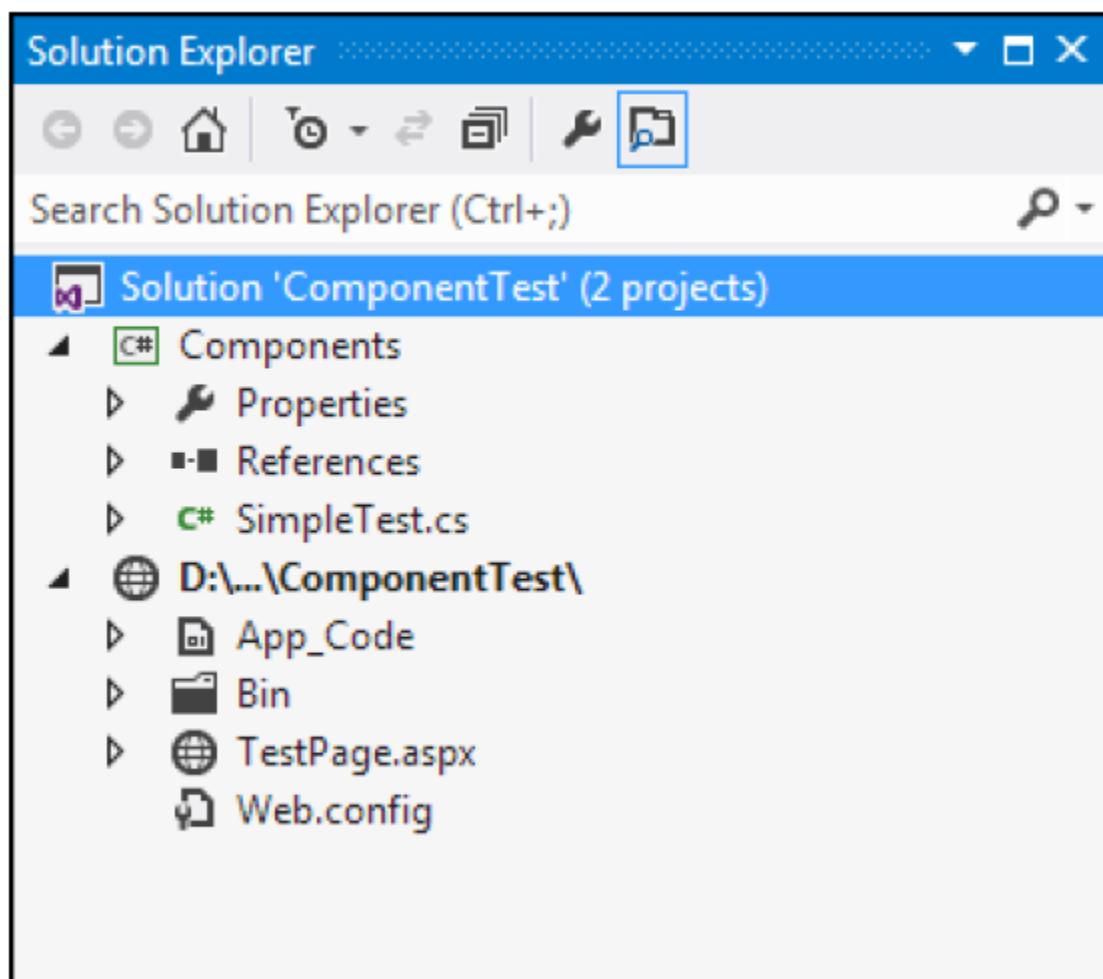
کامپوننت‌ها یک یا چند کلاس می‌باشند که درون یک DLL کامپایل شده‌اند. برای نمونه System.Web.DLL یک کامپوننت دات‌نوت می‌باشد.

برای ایجاد یک کامپوننت، یک پروژه Class Library ایجاد کنید. File>New>Project را انتخاب کنید و سپس در پنجره باز شده، در سمت چپ را انتخاب کنید و در کادر سمت راست Class Library را انتخاب کنید.



برای ایجاد یک Class Library جدید درون یک web solution موجود، گزینه New Project را انتخاب کنید. در شکل زیر، یک Solution، یک website و یک class library مشاهده می‌کنید.

می‌توانید class library را با کلیک راست روی پروژه و انتخاب گزینه Build، کامپایل کنید. این کار یک فایل اسembly DLL را ایجاد می‌کند.



## کلاس‌ها و فضای نام‌ها

زمانی که شما پروژه class library را ایجاد کردید، می‌توانید کلاس‌ها را درون فایل‌های .vb. یا .cs به آن اضافه کنید.

منوی Class > Add را با راست کلیک روی نام پروژه انتخاب کنید.

```
public class SimpleTest
{
    // (Code goes here, inside one or more methods.)
}
```

می‌توانید چندین کلاس را درون یک فایل ایجاد کنید یا فایل‌های جداگانه برای هر کلاس در نظر بگیرید. البته روش دوم بهتر است.

کلاس‌های شما درون یک فضای نام قرار می‌گیرند.

**namespace Components**

{

```

public class SimpleTest
{
    // (Class code omitted.)
}

public class SimpleTest2
{
    // (Class code omitted.)
}

```

زمانی که یک کلاس جدید به class library اضافه می‌کنید، C#، بصورت خودکار، یک بلوک فضای نام با استفاده از فضای نام پیش فرض پرورژه شما، ایجاد می‌کند. این فضای نام root namespace نامیده می‌شود. برای نمونه، اگر یک پرورژه با نام Component ایجاد کرده باشید، کلاس‌های SimpleTest و SimpleTest2 در فضای نام Componenet قرار می‌گیرند و نام کامل آنها Component. SimpleTest و SimpleTest2 خواهد شد. نام کامل کلاس‌ها را باید بدانید زیرا در سایر برنامه‌ها این فضای نام‌ها به اشتراک گذاشته نمی‌شوند.

اگر فضای نام پیش فرض را دوشت ندارید، می‌توانید نام آن را ویرایش کنید. روی پرورژه کلیک راست کنید و Properties را انتخاب نمایید. اگر می‌خواستید نام root namespace را تغییر دهید، زبانه Application را انتخاب و سپس نام آن را در کادر Default NameSpace تغییر دهید. همچنین می‌توانید نام جدیدی در کادر AssemblyName وارد کرده تا نام فایل کامپایل شده اسembly را تغییر دهید.

اگر دارای یک کامپوننت پیچیده هستید، ممکن است آن را به فضای نام‌های تودر تو تقسیم کرده باشد. برای نمونه، ممکن است که یک فضای نام با نام Component.Database Validation داشته باشد. برای ایجاد فضای نام‌های تودر تو، درون فضای نام پیش فرض پرورژه، از بلوک فضای نام مانند زیر استفاده کنید:

**namespace Components**

{

```

namespace Database
{
    public class SimpleDatabaseTest
    {
        // (Class code omitted.)
    }
}

```

 نکته: قانون کلی برای نام‌گذاری فضای نام‌ها، استفاده از نام کمپانی به همراه نام تکنولوژی و بصورت انتخابی، نام امکانات Microsoft در اختیار قرار داده شده می‌باشد. ساختار کلی CompanyName.TechologyName.Feature می‌باشد. مثلاً: Microsoft.Media.Audio و Microsoft.Media

## اعضای کلاس

برای کاربردی نمودن کلاس، متدهای Public یا Property‌هایی را به آن اضافه کنید. کدهای درون صفحات وب، این اعضای درون کلاس را برای بازیابی اطلاعات و انجام کارهای خاص، فراخوانی می‌کنند.

```
public class SimpleTest
{
    public string GetInfo(string param)
    {
        return "You invoked SimpleTest.GetInfo() with " +
            param + "";
    }
}

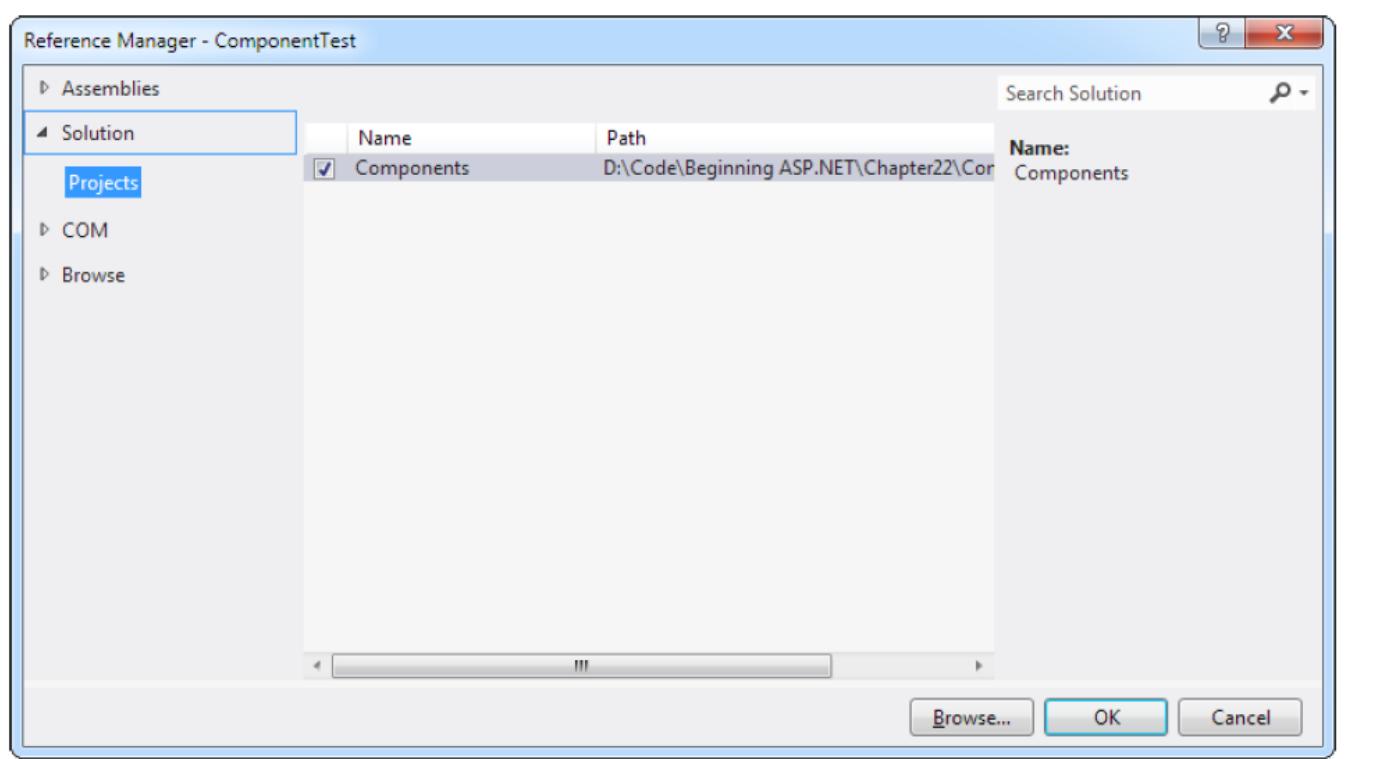
public class SimpleTest2
{
    public string GetInfo(string param)
    {
        return "You invoked SimpleTest2.GetInfo() with " +
            param + "";
    }
}
```

## اضافه نمودن Reference به کامپونت

استفاده از یک کامپونت درون صفحه ASP.NET بسیار ساده است. وب سایت شما، به یک کپی از کامپونت شما درون پوشه Bin خود نیاز دارد. ASP.NET بصورت خودکار، این پوشه را مانیتور می‌کند و تمامی کلاس‌های آن را در دسترس کلیه صفحات وب برنامه قرار می‌دهد. برای ایجاد این کپی، از یک امکان در VS به نام references استفاده می‌کنیم.

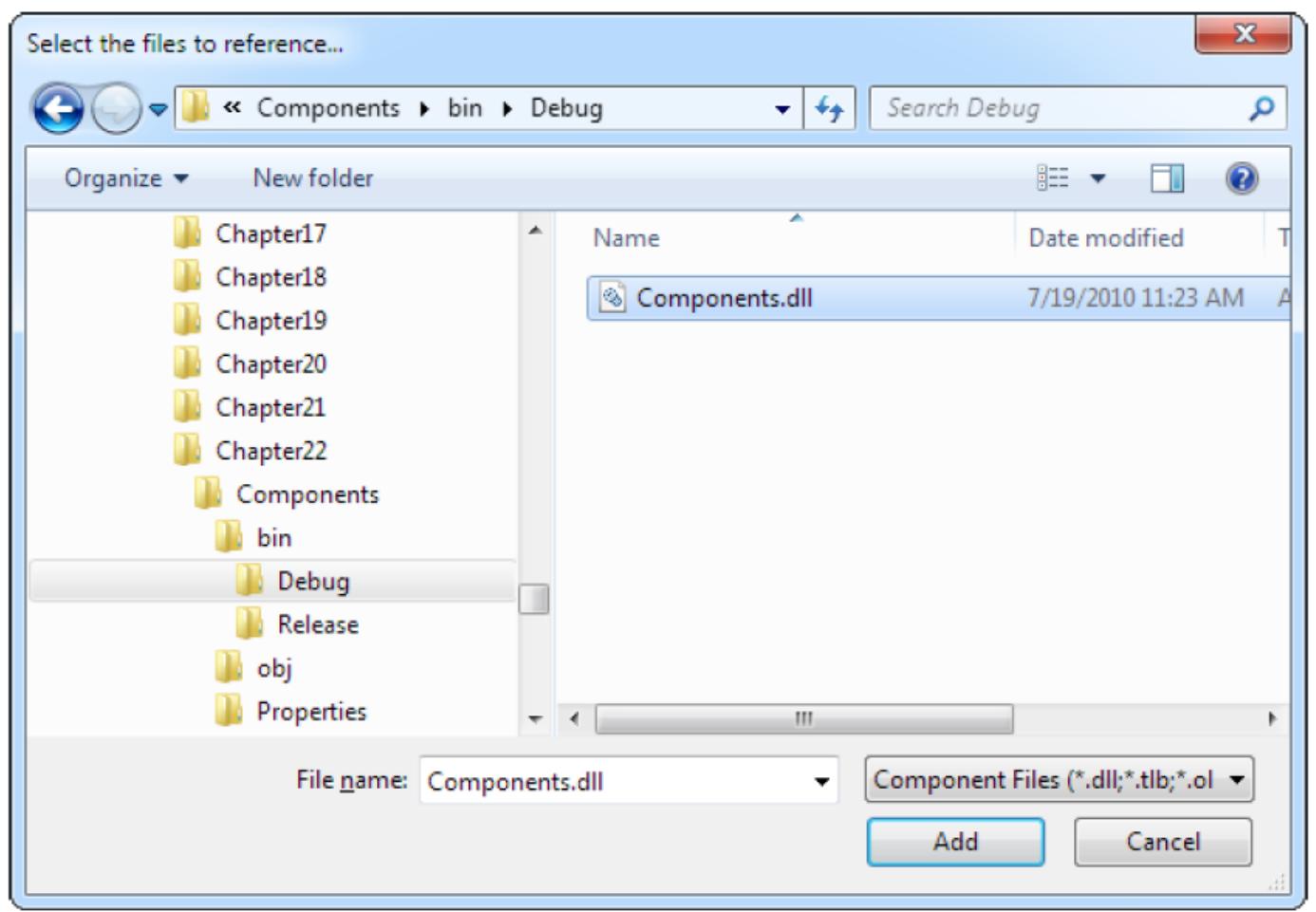
ابتدا، پروژه وب سایت خود را در VS و سپس منوی WebSite>Add Reference را انتخاب کنید. در کادر می‌توانید کارهای زیر را انجام دهید :

**Add a Project reference** - اگر پروژه Class Library شما درون همین solution ای که برنامه تحت وب شما قرار دارد می‌باشد، از سمت چپ، Solution>Projects را انتخاب تا لیستی از classs library ها را مشاهده کنید.

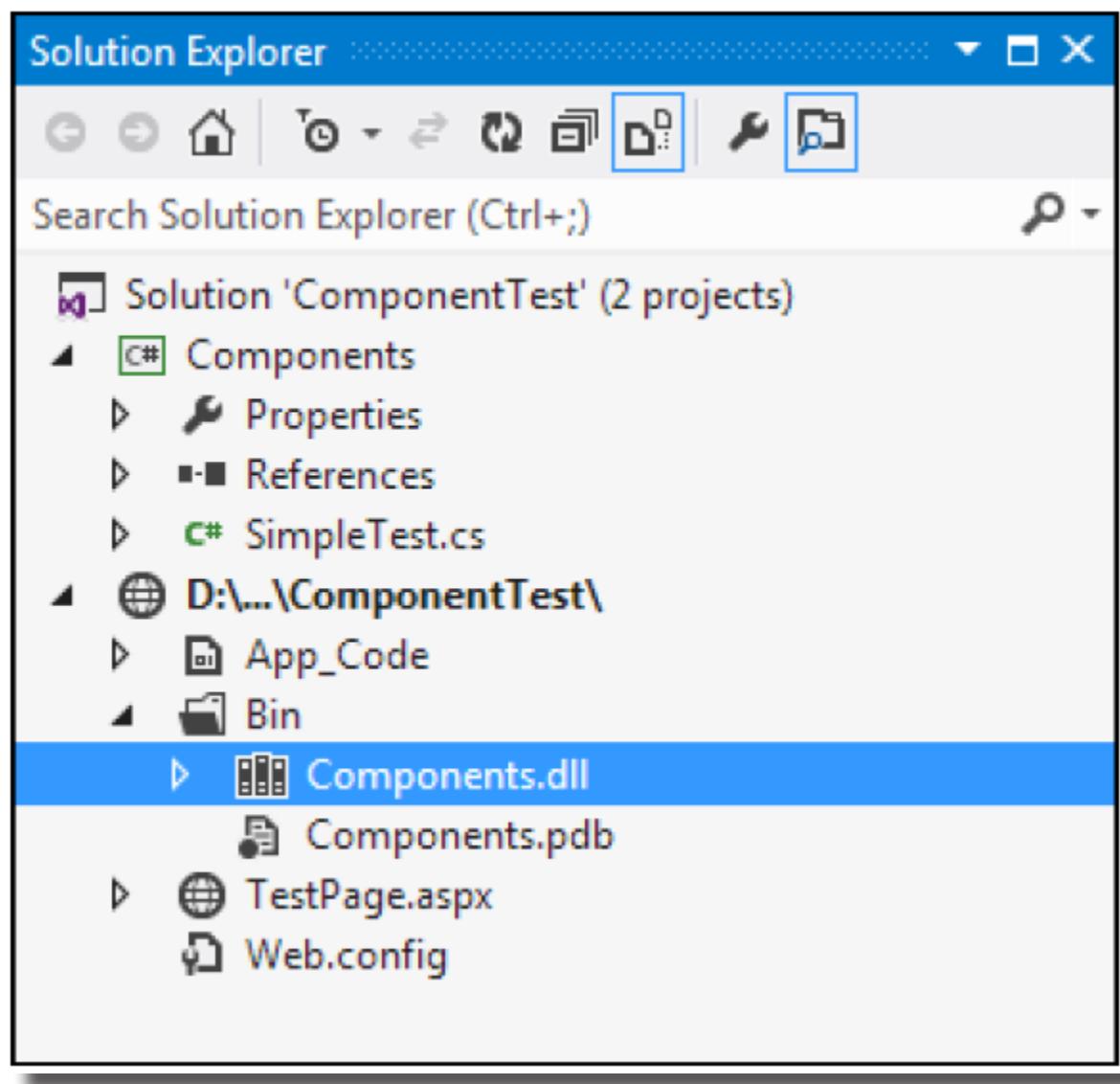


**Add an assembly reference** - اگر class library شما درون یک solution دیگری می‌باشد، یا شما تنها یک فایل DLL دارید (دسترسی به پروژه class library ندارید و این کامپوننت توسط شخص دیگری ایجاد شده است)، روی اسembly brows در انتهای پنجره کلیک کنید تا پنجره جدیدی باز شود. مسیر فایل DLL را تعیین و روی OK کلیک کنید.

**نکته:** در صورت استفاده از assembly reference، باید ابتدا کامپوننت را قبل از add reference، کامپایل کنید(گزینه (Build>Build Solution



دات نت، DLL های کامپایل شده را به پوشه Bin موجود در پوشه برنامه و ب شما کپی می کند. همچنین یک فایل .PDB نیز مشاهده می کنید که شامل اطلاعاتی برای debugging برای استفاده VS می باشد.



VS، مراقب این است که شما از آخرین نسخه کامپوننت استفاده کنید. اگر کامپوننت را تغییر دهید و آن را مجدد کامپایل کنید، VS تغییر را متوجه می‌شود و دفعه بعدی که برنامه را اجرا کنید، بصورت خودکار کامپوننت جدید را درون پوشه Bin کپی می‌کند.

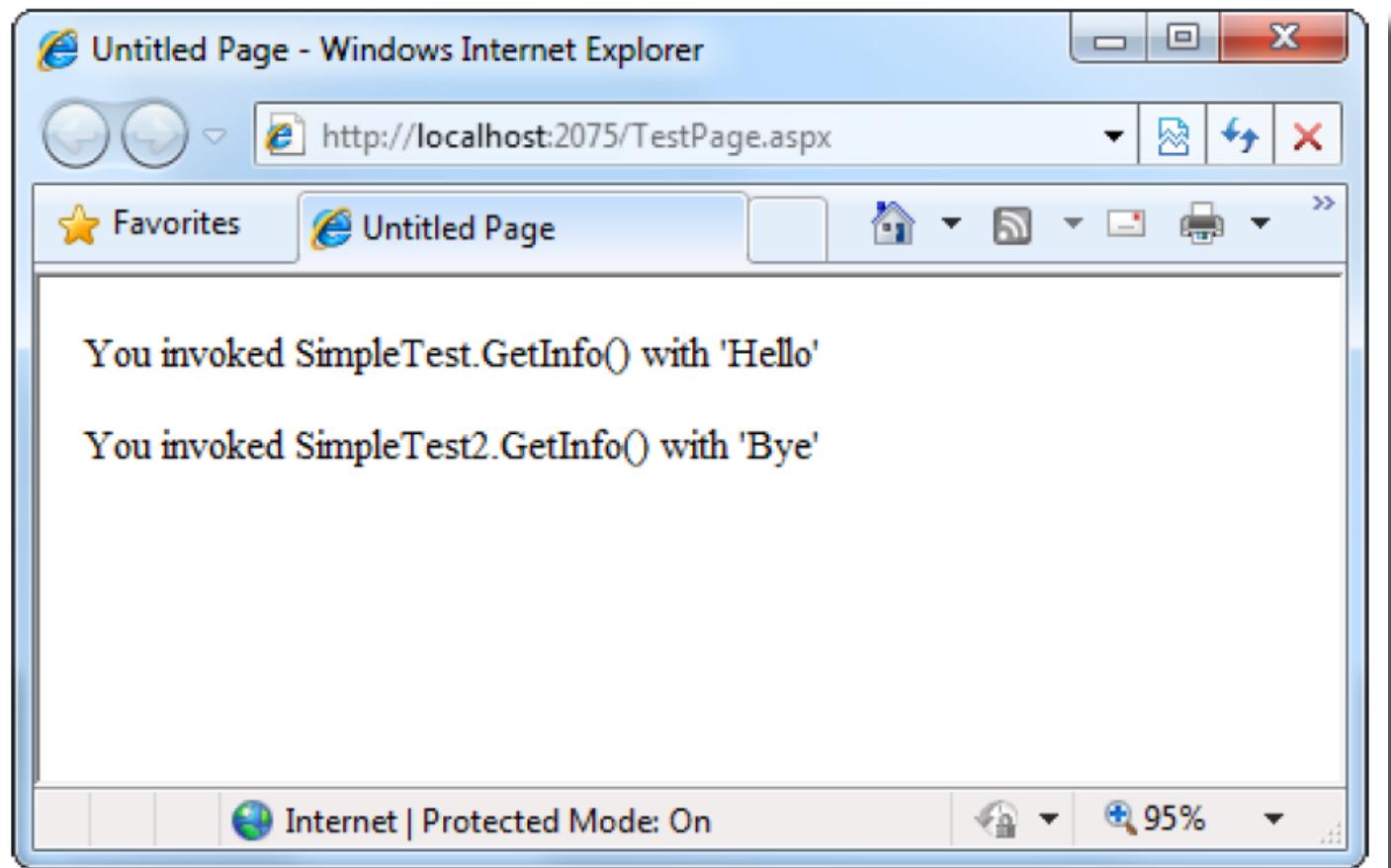
اگر از Project Reference استفاده کرده باشید، VS، یک کار اضافه تر هم انجام می‌دهد. هر زمان که پروژه وب سایت خود را اجرا کنید، VS، کلیه تغییرات انجام شده در فایل‌های سورس کد کامپوننت را چک می‌کند. اگر تغییری در یکی از این فایل‌ها رخ داده باشد، VS، بصورت خودکار کامپوننت را مجدداً کامپایل کرده و نسخه جدید را درون پوشه Bin از برنامه تحت وب قرار می‌دهد.

## استفاده از کامپونت

زمانی که یک reference را add کردید، می‌توانید با استفاده از ایجاد نمونه هایی از کلاس SimpleTest یا SimpleTest2 از آن کامپونت استفاده کنید :

```
using Components;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        SimpleTest testComponent = new SimpleTest();
        SimpleTest2 testComponent2 = new SimpleTest2();
        lblResult.Text = testComponent.GetInfo("Hello") + "<br><br>";
        lblResult.Text += testComponent2.GetInfo("Bye");
    }
}
```



برای آنکه این کد را ساده تر کنیم می‌توانیم از متدهای static در کامپوننت استفاده کنیم.

```
public class SimpleTest
{
    public static string GetInfo(string param)
    {
        return "You invoked SimpleTest.GetInfo() with " +
               param + "!";
    }
}
```

درون صفحه وب می‌توانید این متدر از طریق نام کلاس و بدون نیاز به ایجاد شی از آن فراخوانی کنید:

```
protected void Page_Load(Object sender, EventArgs e)
{
    lblResult.Text = SimpleTest.GetInfo("Hello");
}
```

 نکته: به یاد داشته باشید که اگر از assembly reference ها استفاده می‌کنید و کامپوننت و برنامه تحت وب شما درون یک sloution نیستند، حاصل تغییرات خود را مستقیماً نخواهید دید. باید کامپوننت خود را مجدد کامپایل کنید، VS (Build> Build Solution) و سپس برنامه تحت وب خود را Rebuild کنید. اگر از Project Reference استفاده کنید، خودکار متوجه تغییرات شما در کامپوننت می‌شود و با اجرای برنامه آن را کامپایل و درون پوشه Bin قرار می‌دهد.

تشخیص اینکه چه موقع از static method و چه موقع از instance method استفاده کنید با شما است. برای نمونه اتصال به بانک های مختلف برای انجام یک عملیات، باز و بسته کنید. آنها بهترین گزینه برای یکبار پیکربندی کردن شی و چندین بار استفاده از آن می‌باشند. برای نمونه،SqlConnection به شما اجازه تعیین connection string و سپس باز و بسته کردن اتصال را خواهد داد. در سمت دیگر static method هاستند که اگر متدر شما یک کار تنها که نیازی به هیچ مقداردهی اولیه ندارد را انجام می‌دهد، می‌توانید از آنها استفاده کنید. مانند محاسبات در کلاس math و کارهای business ای (مانند ثبت یک مشتری جدید) در یک کامپوننت سطح بالای business ای.

## State و Property

کلاس SimpleTest از طریق متدهای Public خود، کاربردی هایی را ارائه می دهد. زمانی که از Property ها استفاده می کنید و اطلاعات را درون متغیرها قرار می دهید، از طراحی stateful، بهره می برید. در این طراحی، کلاس، مسئول نگهداری بخش هایی از اطلاعات می باشد. در طراحی stateless، هیچ اطلاعاتی در بین فراخوانی متدها حفظ نمی شود.

کلاس SimpleTest زیر، برخلاف کلاس statefull قبلی SimpleTest می باشد.

```
public class SimpleTest
{
    private string data;
    public string Data
    {
        get
        {
            return data;
        }
        set
        {
            data = value;
        }
    }
    public string GetInfo()
    {
        return "You invoked SimpleTest.GetInfo()," +
               "and data is " + data + "";
    }
}
```

برای انجام یک کار باید چندین property را قبل از فراخوانی متدهایی کنید. هر یک از این مراحل، یک سربار اضافی غیر ضروری به برنامه تحمیل خواهد کرد. در طراحی stateless، همه این کارها تقریباً درون متدهای اجرا می شود. به دلیل آنکه هیچ اطلاعاتی درون وجود ندارد، باید چندین پارامتر را مقداردهی کنید که کمی خسته کننده می باشد. یک نمونه خوب برای مقایسه اشیای statefull و stateless در برابر FileInfo و File نشان داده شده است.

کامپوننت هایی با کارایی بالا که اغلب از transaction ها یا منابع محدود مانند اتصال های بانک اطلاعاتی استفاده می کنند، از طراحی stateless استفاده می کنند.

فرض کنید یک کلاس CustomerAccount دارد. اطلاعات از بانک خوانده شده و توسط متدهای Update() و Insert() می شوند.

```

public class CustomerAccount
{
    private int accountNumber;
    private decimal balance;
    public decimal Balance
    {
        get
        { return balance; }
        set
        { balance = value; }
    }
    public CustomerAccount(int accountNumber)
    {
        // (Code to read account record from database goes here.)
    }
    public void Update()
    {
        // (Code to update database record goes here.)
    }
}

```

## کلاس StateFull با طراحی CustomerAccount

اگر دو شی CustomerAccount داشته باشد که دارای ویژگی Balance باشند، باید دو مرحله جداگانه برای انتقال پول از یک حساب به حساب دیگر اجرا کنید.

```

// Create an account object for each account,
// using the account number.

CustomerAccount accountOne = new CustomerAccount(122415);
CustomerAccount accountTwo = new CustomerAccount(123447);
decimal amount = 1000;
// Withdraw money from one account.

```

```

accountOne.Balance -= amount;

// Deposit money in the other account.

accountTwo.Balance += amount;

// Update the underlying database records using an Update method.

accountOne.Update();

accountTwo.Update();

```

## کلاس AccountUtility با طراحی Stateless :

```

public class AccountUtility
{
    public static void FundTransfer(int accountOne, int accountTwo, decimal amount)
    {
        // (The code here retrieves the two database records,
        // changes them, and updates them.)

    }
}

```

با دستورات زیر می‌توانید از این متده استفاده نمایید :

```

// Set the account and transfer details.

decimal amount = 1000;

int accountIDOne = 122415;

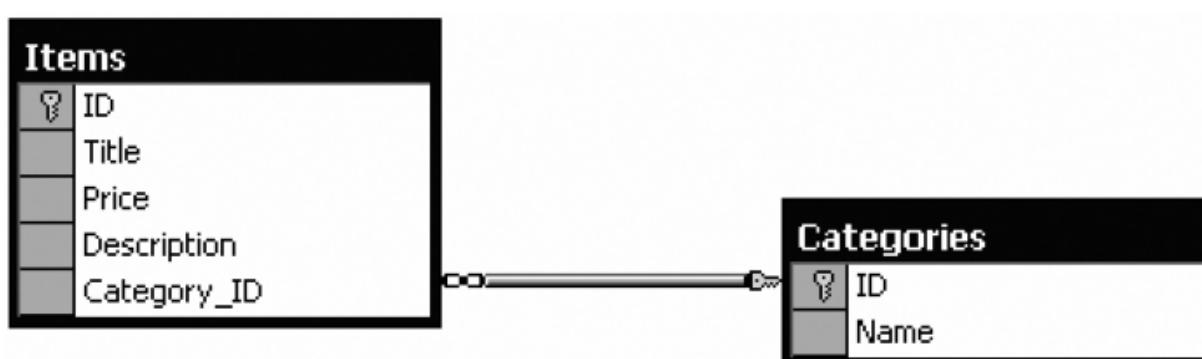
int accountIDTwo = 123447;

AccountUtility.FundTransfer(accountIDOne, accountIDTwo, amount);

```

## Data-Access Component

در مثال زیر بانک اطلاعاتی دارای دو جدول می‌باشد. یکی Items، که توضیحات و قیمت آیتم را مشخص می‌کند و دیگری Categories که لیست گروه‌های مختلف را نشان می‌دهد.



```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Web.Configuration;
namespace DatabaseComponent
{
    public class DBUtil
    {
        private string connectionString;
        public DBUtil()
        {
            connectionString =
                WebConfigurationManager.ConnectionStrings[
                    "AdBoard"].ConnectionString;
        }
        public DataSet GetCategories()
        {
            string query = "SELECT * FROM Categories";
    
```

```

    SqlCommand cmd = new SqlCommand(query);

    return FillDataSet(cmd, "Categories");

}

public DataSet GetItems()

{

    string query = "SELECT * FROM Items";

    SqlCommand cmd = new SqlCommand(query);

    return FillDataSet(cmd, "Items");

}

public DataSet GetItems(int categoryID)

{

    // Create the Command.

    string query = "SELECT * FROM Items WHERE Category_ID=@CategoryID";

    SqlCommand cmd = new SqlCommand(query);

    cmd.Parameters.AddWithValue("@CategoryID", categoryID);

    // Fill the DataSet.

    return FillDataSet(cmd, "Items");

}

public void AddCategory(string name)

{

    SqlConnection con = new SqlConnection(connectionString);

    // Create the Command.

    string insertSQL = "INSERT INTO Categories ";

    insertSQL += "(Name) VALUES @Name";

    SqlCommand cmd = new SqlCommand(insertSQL, con);

    cmd.Parameters.AddWithValue("@Name", name);

    try

    {

        con.Open();

        cmd.ExecuteNonQuery();
    }
}

```

```

        }

    finally

    {

        con.Close();

    }

}

public void AddItem(string title, string description,
decimal price, int categoryID)

{

    SqlConnection con = new SqlConnection(connectionString);

    // Create the Command.

    string insertSQL = "INSERT INTO Items ";

    insertSQL += "(Title, Description, Price, Category_ID)";

    insertSQL += "VALUES (@Title, @Description, @Price, @CategoryID)";

    SqlCommand cmd = new SqlCommand(insertSQL, con);

    cmd.Parameters.AddWithValue("@Title", title);

    cmd.Parameters.AddWithValue("@Description", description);

    cmd.Parameters.AddWithValue("@Price", price);

    cmd.Parameters.AddWithValue("@CategoryID", categoryID);

    try

    {

        con.Open();

        cmd.ExecuteNonQuery();

    }

    finally

    {

        con.Close();

    }

}

private DataSet FillDataSet(SqlCommand cmd, string tableName)

```

```

{
    SqlConnection con = new SqlConnection(connectionString);

    cmd.Connection = con;

    SqlDataAdapter adapter = new SqlDataAdapter(cmd);

    DataSet ds = new DataSet();

    try
    {
        con.Open();

        adapter.Fill(ds, tableName);

    }
    finally
    {
        con.Close();
    }

    return ds;
}

}
}

```

### شرح کد بالا :

- زمانی که کد یک شی DBUtil، ایجاد می‌کند، سازنده آن بصورت خودکار، از فایل web.config connection string را بازیابی می‌کند.
- کد شامل متدهایی برای بازیابی (متدهایی که با Get شروع می‌شوند) و ویرایش داده (متدهایی که با Add شروع می‌شوند) می‌باشد.
- این کلاس شامل یک متد overload با نام GetItems() می‌باشد که کلاینت می‌تواند آن را بدون هیچ پارامتری برای برگرداندن گروه‌ها فراخوانی کند.
- Connection را در بازه زمانی زنده بودن کلاس lifetime (باید باز نگه داشت).



نکته: وب سرور شما می‌تواند بصورت متوالی و بدون کاهش سرعت، اتصال به بانک را باز و بسته کند. زیرا ADO.NET از connection pooling برای نگهداری یک مجموعه کوچک از connection های آماده استفاده باز استفاده می‌کند. تا زمانی که SqlConnection.Open() وجود دارد، اگر connection string را تغییر نداده‌اید و تا زمانی که اتصال های قابل دسترسی در pool وجود نداشته باشد، این اتفاق را فراخوانی کنید، یکی از این اتصال‌ها را بازیابی می‌کنید، بنابراین از سربارهای اضافی پیکربندی یک connection جلوگیری می‌شود.

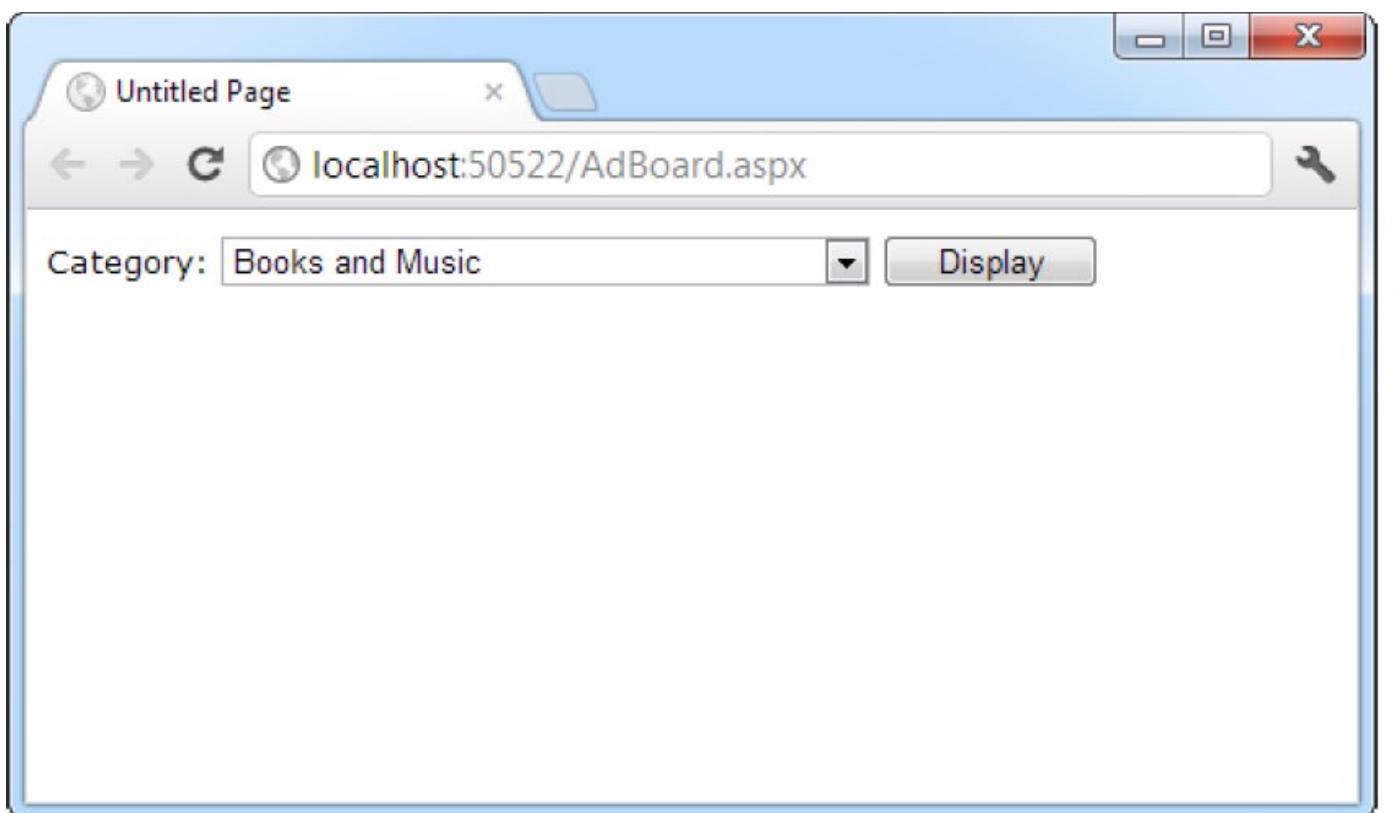
- در کد از متده FillDataSet() برای مختصر کردن کدنویسی استفاده شده است.

## استفاده از Data-Access Componenet

برای استفاده از کامپوننت در برنامه وب، ابتدا باید مطمئن شوید که web.config در فایل connection string پیکربندی شده است.

```
<configuration>
  <connectionStrings>
    <add name="AdBoard" connectionString=
      "Data Source=(localdb)\v11.0;Initial Catalog=AdBoard;Integrated
      Security=SSPI" />
  </connectionStrings>
```

سپس، فایل DLL را کامپایل و کپی کنید. تنها کار باقی مانده، اضافه نمودن واسط کاربری برای صفحه وبی که از کامپوننت استفاده می‌کند، می‌باشد.



زمانی که یک گروه انتخاب می‌شود، آیتم‌های منطبق با آن نمایش داده می‌شوند.

The screenshot shows a web browser window titled 'Untitled Page' with the URL 'localhost:50522/AdBoard.aspx'. A dropdown menu labeled 'Category' is set to 'Books and Music', and a 'Display' button is visible. Below the header, the title 'Add Item To Current Category' is displayed. The form contains three input fields: 'Title' (empty), 'Price' (empty), and 'Description' (empty). An 'Add' button is located below the form. At the bottom, a grid displays three items:

ID	Title	Price	Description	Category_ID
1	Learning Thermodynamics	7.9900	A great place to start learning about entropy.	6
2	ASP.NET: The Complete Reference	17.9900	A steal at this price!	6
3	The Sound And the Fury	2.9900	Mild wear and coffee stains throughout. Still readable.	6

برای دسترسی آسان به کامپوننت، فضای نام زیر را در صفحه import کنید:

```
using DatabaseComponent;
```

دد صفحه، یک کامپوننت برای بازیابی اطلاعات بانک و نمایش آن با استفاده از dataset bind کردن یا drop-down list به View ایجاد می‌کند.

```
public partial class AdBoard : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            DBUtil DB = new DBUtil();

            lstCategories.DataSource = DB.GetCategories();
            lstCategories.DataTextField = "Name";

            lstCategories.DataValueField = "ID";
            lstCategories.DataBind();
            pnlNew.Visible = false;
        }
    }

    protected void cmdDisplay_Click(Object sender, EventArgs e)
    {
        DBUtil DB = new DBUtil();

        gridItems.DataSource = DB.GetItems(
            Int32.Parse(lstCategories.SelectedItem.Value));
        gridItems.DataBind();
        pnlNew.Visible = true;
    }

    protected void cmdAdd_Click(Object sender, EventArgs e)
    {
        DBUtil DB = new DBUtil();

        try
        {
            DB.AddItem(txtTitle.Text, txtDescription.Text,
                Decimal.Parse(txtPrice.Text),
                Int32.Parse(lstCategories.SelectedItem.Value));
        }
    }
}
```

```

gridItems.DataSource = DB.GetItems(
    Int32.Parse(lstCategories.SelectedItem.Value));
gridItems.DataBind();
}

catch (FormatException err)
{
    // An error occurs if the user has entered an
    // invalid price (non-numeric characters).

    // In this case, take no action.

    // Another option is to add a validator control
    // for the price text box to prevent invalid input.
}
}
}

```

## در کامپونت Error Handling

هر خطای بانک اطلاعاتی که رخ می‌دهد، یک کد برمی‌گرداند.

```

public class DBUtil
{
    private string connectionString;

    public DBUtil()
    {
        if (WebConfigurationManager.ConnectionStrings[“AdBoard”] == null)
        {
            throw new ApplicationException(
                “Missing ConnectionString variable in web.config.”);
        }
        else
        {
            connectionString =
                WebConfigurationManager.ConnectionStrings[

```

```

        “AdBoard”].ConnectionString;

    }

}

// (Other class code omitted.)

}

```

## کامپوننت با توابع aggregate

```

public class DBUtil

{
    // (Other class code omitted.)

    public decimal GetAveragePrice()

    {
        string query = “SELECT AVG(Price) FROM Items”;

        SqlConnection con = new SqlConnection(connectionString);

        SqlCommand cmd = new SqlCommand(query, con);

        con.Open();

        decimal average = (decimal)cmd.ExecuteScalar();

        con.Close();

        return average;
    }

    public int GetTotalItems()

    {
        string query = “SELECT Count(*) FROM Items”;

        SqlConnection con = new SqlConnection(connectionString);

        SqlCommand cmd = new SqlCommand(query, con);

        con.Open();

        int count = (int)cmd.ExecuteScalar();

        con.Close();

        return count;
    }
}

```

## ObjectDataSource

با استفاده از کامپوننت هایی مانند SQLDataSource مجبور بودید همه کارها را در لایه واسط کاربر انجام دهید. در طراحی لایه بندی نمی توان از این روش استفاده کرد. در عوض یک کامپوننت دیگر با نام ObjectDataSource در دات نت وجود دارد که می تواند به لایه مورد نظر شما وصل شده و متدهای آن را فراخوانی کند. بنابراین متدهای خود را درون آن لایه نوشه و در لایه واسط کاربر از آنها توسط این کنترل استفاده می کنید.

برای آنکه بتوانید data component ای داشته باشید که توسط ObjectDataSource قابل استفاده باشد باید این قوانین را رعایت کنید :

- کلاس شما باید stateless باشد. زیرا ODS یک نمونه از آن را در زمان نیاز ایجاد کرده و آن را در پایان هر درخواست از بین می برد.
- کلاس شما باید دارای سازنده پیش فرض بدون پارامتر باشد.
- کل منطق باید درون یک کلاس باشد. (اگر می خواهید کلاس های مختلفی برای انتخاب و ویرایش داده ها باشید، باید آنها را در سایر کلاس های سطح بالاتر قرار دهید)
- نتیجه Query باید، یک collection یا اشیای DataSet ,DataTable را برگرداند. (در صورت برگرداندن public property data object باید تمامی فیلد ها را بصورت public property در خود داشته باشد)

```
<asp:ObjectDataSource ID="sourceItems" runat="server" SelectMethod="GetItems"
    TypeName="DatabaseComponent.DBUtil">

    <SelectParameters>

        <asp:ControlParameter ControlID="lstCategories" Name="categoryID"
            PropertyName="SelectedValue"
            Type="Int32" />

    </SelectParameters>

</asp:ObjectDataSource>
```

# بخش پنجم : پیشرفته ASP.NET

فیل شانزدهم: برنامه نویسی مبتنی بر کامپونت

Caching فیل هفدهم:

Entity Framework , LINQ فیل هجدهم:

ASP.NET AJAX فیل نوزدهم:

ASP.NET مرکز بروزرسانی Deploy فیل بیست و یکم:



## Caching

برنامه های تحت وب نیاز دارند تا بتوانند به صدھا کاربر سریع و راحت سرویس دهنند. می توانید با رعایت چند نکته، برنامه های تحت وب را به همان روشی که برنامه های ویندوزی را می نویسید، تهیه کنید. یکی از ساده ترین راه ها برای بهبود کارایی سایت شما، استفاده از caching می باشد که تکنیکی برای ذخیره اطلاعات با ارزش در حافظه سرور می باشد تا مورد استفاده مجدد قرار بگیرند. برخلاف سایر انواع caching دارای ویژگی های تعییه شده در دات نت می باشد که کارایی آن را تضمین می کند.

### آشنایی با Caching

Caching، اغلب برای ذخیره اطلاعاتی که از بانک اطلاعاتی بازیابی می شود، مورد استفاده قرار می گیرد. بازیابی اطلاعات از بانک، کار زمان بری می باشد. با یک بهینه سازی (optimization) هوشمندانه، می توانید این زمان را کاهش دهید ولی نمی توانید آن را از بین ببرید. اما با استفاده از سیستمی که از Caching استفاده می کند، برخی از درخواست های داده، نیازی به اتصال بانک اطلاعاتی و query ندارد. در عوض، آنها اطلاعات را مستقیماً از حافظه سرور که سریعتر می باشد، می گیرند.

البته ذخیره کردن اطلاعات، در حافظه، همیشه ایده خوبی نیست. حافظه سرور یک منبع محدود می باشد. اگر سعی کنید زیادی اطلاعات زیادی را در آن ذخیره کنید، بخشی از آن در دیسک صفحه بندی (Paged) می شود که همین کار کل سیستم را کند می کند.

طول عمر اطلاعات در این روش بستگی به سرور دارد. اگر Cache پر شده باشد، یا سایر برنامه ها از آن بصورت متوالی استفاده کنند، داده ها بصورت انتخابی از Cache حذف می شوند.

### چه زمانی از Caching استفاده کنیم

 نکته مهم در استفاده از Caching، زمان صحیح بکار گیری آن می باشد. دو دستورالعمل مورد نیاز برای Caching در زیر آمده:

- Cache کردن داده (یا صفحات وب) هایی که با ارزش هستند:

Cache کردن داده ای که ایجاد آن زمان بر می باشد. محتواهای Query یا فایل، مثال های خوبی از این مورد هستند. نه تنها باز کردن یک اتصال به بانک اطلاعاتی یا باز کردن فایل زمان بر هست، بلکه می تواند دسترسی سایر افرادی که می خواهند به یک چیز دسترسی داشته باشند را با تأخیر رو برو کرده یا منع می کند.

- Cache کردن داده (یا صفحات وب) ای که مرتبأً استفاده می شوند :

داده هایی که هرگز مجددًا استفاده نمی شوند را نباید Cache کرد. بری نمونه، ممکن است جزئیات یک محصول را Cache کنید، زیرا صدھا هزار محصول مختلف با صفحات خودشان وجود دارند. اما Cache کردن لیست گروه های محصولات که در درخواست های متعدد مختلفی استفاده می شوند، مناسب می باشد.

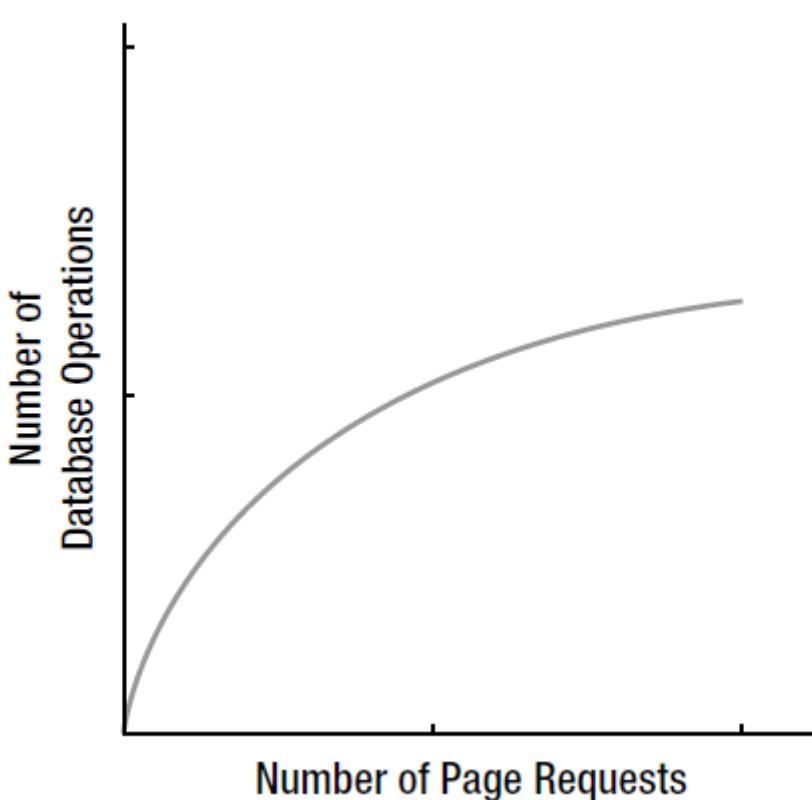
اگر دو نکته بالا را در نظر بگیرید می توانید هم کارایی (Performance) و هم مقیاس پذیری (Scalability) را بهبود دهید.

کارایی، سرعت کار یک صفحه برای یک کاربر تنها می باشد. Caching، کارایی را با پشت سر گذاشتن نقاط محدودیتی مانند بانک

اطلاعاتی افزایش می‌دهد. در نتیجه، صفحات وب بسیار سریعتر پردازش شده و به کلاینت برگردانده می‌شوند.

مقیاس پذیری، چگونگی کاهش کارایی صفحه وب برنامه شما را با افزایش استفاده کاربران همزمان از آن، اندازه گیری می‌کند. Caching، مقیاس پذیری را با استفاده مجدد از اطلاعات بهبود می‌بخشد. با استفاده از آن، افراد بیشتری می‌توانند از وب سایت شما استفاده کنند، بدون آنکه تعداد رفت و برگشت‌ها به بانک اطلاعاتی تغییر زیادی کند.

تأثیر یک Caching خوب را در شکل زیر مشاهده می‌کنید :



## ASP.NET در Caching

ASP.NET، دو نوع از Caching را ارائه می‌دهد. برنامه شما باید از هر دوی آنها استفاده کند زیرا آنها مکمل یکدیگر هستند:

**Output Caching** : ساده‌ترین نوع Caching می‌باشد. یک کپی از HTML رندر شده نهایی صفحه که به کلاینت ارسال می‌شود را نگهداری می‌کند. کلاینت بعدی، که یک درخواست را برای این صفحه ارسال می‌کند، واقعاً آن را اجرا نخواهد کرد. در عوض، HTML نهایی بصورت خود کار برای آن ارسال می‌شود.

**DataCaching** : این روش بصورت دستی در کد انجام می‌شود. باید بخش‌های مهمی از اطلاعاتی که ساخت مجدد آنها زمان بر می‌باشد (مانند DataSet ای که از بانک بازیابی شده است) را در Cache نگهداری کنید. صفحات دیگر می‌توانند وجود این اطلاعات

را چک کنند و از آن استفاده نمایند. بنابراین مرحله بازیابی مجدد اطلاعات، نادیده گرفته می‌شود. Data Caching، مفهوماً شبیه استفاده از application state می‌باشد، اما به دلیل آنکه در زمانی که آیتم‌ها حجم شوند روی کارایی تأثیر می‌گذارند و بنابراین بصورت خودکار از Cache حذف می‌شوند، می‌توان گفت که server-friendly تر می‌باشند.

همچنین بر اساس این مدل‌ها، دو نوع خاص از Caching ساخته شده است :

### **: Fragment Caching**

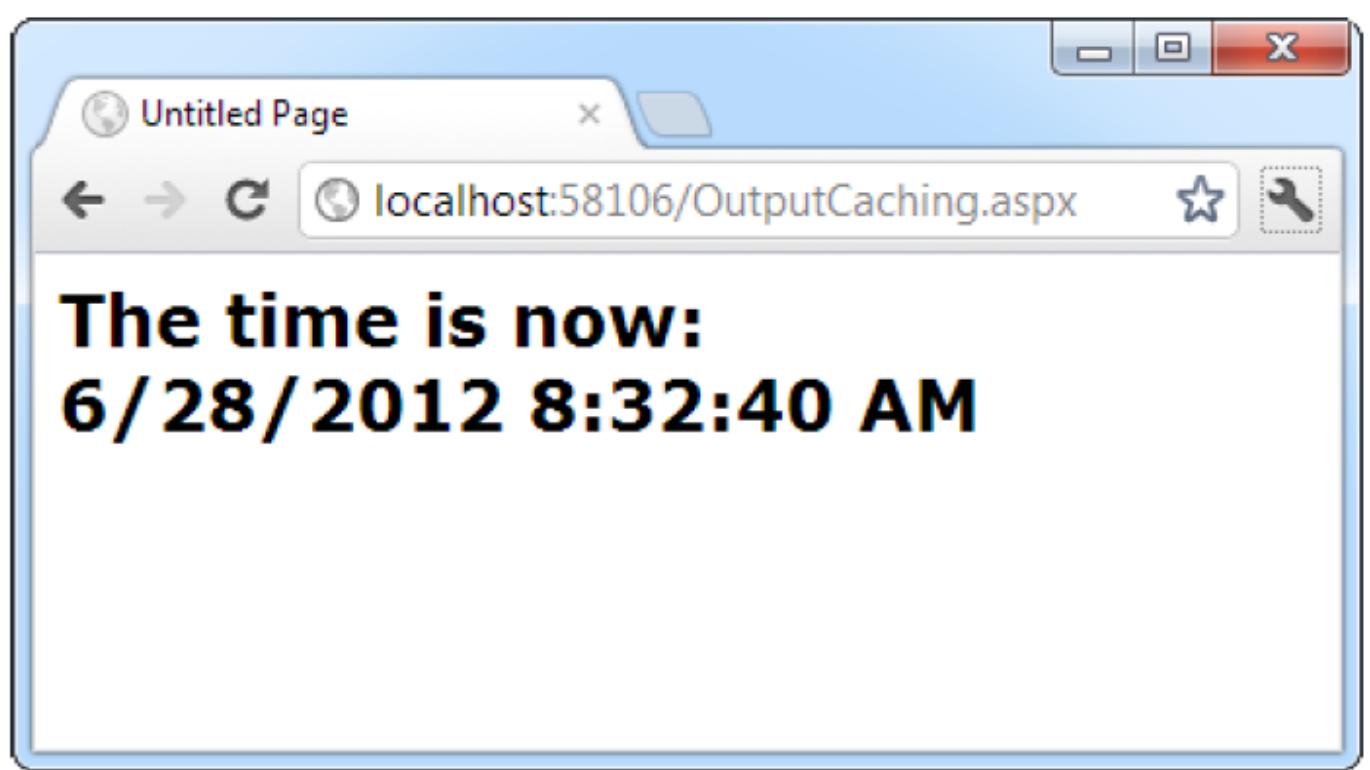
نوع خاصی از output caching می‌باشد. به جای آنکه HTML کل صفحه را Cache کند، اجازه Cache کردن بخشی از آن را به شما می‌دهد. این نوع از caching برای نگهداری HTML رندر شده یک user control استفاده می‌شود. در زمان بعدی که صفحه درخواست شود، رویدادهای همان صفحه رخ خواهد داد ولی کد مربوط به آن user control، اجرا نخواهد شد.

### **: Data Source Caching**

این نوع از Caching درون کنترل‌هایی مانند ObjectDataSource و SQLDataSource استفاده شده است. استفاده می‌کند و این نیاز به هندل کردن آن توسط شما نمی‌باشد.

### **: Output Caching**

در این روش کل صفحه Cache می‌شود و در دفعات بعدی که صفحه درخواست می‌شود، اشیای کنترل ایجاد نمی‌شوند و چرخه حیات (life cycle) صفحه شروع نمی‌شود و هیچ کدی اجرا نخواهد شد. در عوض HTML‌ای Cache شده است در اختیار قرار می‌گیرد.



```

public partial class OutputCaching : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        lblDate.Text = "The time is now:<br />";
        lblDate.Text += DateTime.Now.ToString();
    }
}

```

شما می‌توانید یک صفحه ASP.NET را به دو روش Cache کنید. رایج‌ترین روش اضافه کردن راهنمای OuputCache به بالای فایل.aspx می‌باشد.

```
<%@ OutputCache Duration="20" VaryByParam="None" %>
```

خصیصه Duration، باعث می‌شود صفحه برای ۲۰ ثانیه Cache شود. با خصیصه VarByParam در ادامه آشنا خواهد شد. دفعه اولی که صفحه را اجرا می‌کنید یک زمان را به شما نشان می‌دهد. اگر مجدد آن را refresh کنید، صفحه بروز نخواهد شد. اگر ASP.NET درخواستی را پس از منقضی شدن Cache دریافت کند، آن صفحه را از ابتدا اجرا کرده و یک کپی Cache شده از HTML آن نیز ایجاد می‌کند و برای ۲۰ ثانیه بعدی از آن استفاده خواهد کرد.

## Query String و Caching

یکی از نکات مهم در Caching، تصمیم‌گیری درباره این موضوع است که چه زمانی یک صفحه می‌تواند استفاده مجدد شود و کی اطلاعات باید دقیقاً بروز شوند.

در مثال بالا، خصیصه VarByParam با مقدار None برابر گشته بود. این مقدار به ASP.NET می‌گوید که شما نیاز به ذخیره تنها یک کپی از صفحه Cache شده دارید. اگر درخواستی برای این صفحه، دارای آرگومان‌های query string در URL خود باشد، ASP.NET تا زمانی که خروجی ما expire شود، کماکان از همان خروجی Cache شده استفاده خواهد کرد. بنابراین اگر لازم باشد برای هر Query String مختلف یک صفحه جداگانه Cache شده باشد، باید VarbyParam را برابر با \* قرار دهید.

```
<%@ OutputCache Duration="20" VaryByParam="*" %>
```

اگر صفحه را با اطلاعات query string اضافی، درخواست کنید، آن را امتحان خواهد کرد و اگر عبارت با یکی از درخواست‌های قبلی منطبق باشد و یک کپی از صفحه نیز موجود باشد، از آن مجدد استفاده خواهد کرد. در غیر این صورت، یک کپی

جدید از صفحه ایجاد و Cache خواهد شد.

برای درک بهتر، موارد زیر را در نظر بگیرید :

- 1 یک صفحه را بدون هیچ Query String ای درخواست می کنید و صفحه کپی شده A را دریافت می کنید.
- 2 صفحه را با پارامتر ProductID=1 درخواست کرده و صفحه کپی شده B را دریافت می کنید.
- 3 کاربر دیگری، صفحه را با پارامتر ProductID=2 درخواست می کند و صفحه کپی شده C را دریافت می کند.
- 4 کاربر دیگری صفحه را با پارامتر ProductID=1 درخواست می کند. اگر خروجی Cache شده B، منقضی نشده باشد، به کاربر ارسال می شود.
- 5 کاربر سپس صفحه را بدون هیچ پارامتر query string ای درخواست می کند. اگر کپی A هنوز expire نشده باشد، به کاربر ارسال می شود.

برای مشاهده تأثیر Caching بهتر است اندکی سرعت صفحه خود را کند کنید :

```
protected void Page_Load(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(TimeSpan.FromSeconds(15));
}
```

### Caching با پارامترهای Query String مشخص

```
<%@ OutputCache Duration="20" VaryByParam="ProductID" %>
```

در این مثال، در ASP.NET query string به دنبال پارامتر ProductID می گردد. درخواست هایی با ProductID های مختلف، جداگانه Cache می شوند ولی سایر پارامتر ها نادیده گرفته می شوند. اگر صفحه با پارامترهایی همراه باشد که استفاده نمی شوند، این روش مناسب خواهد بود.

می توانید چندین پارامتر را با (;) جدا کنید :

```
<%@ OutputCache Duration="20" VaryByParam="ProductID;CurrencyType" %>
```

برای مثل، URL های زیر دارای یک نسخه Cache شده می باشند:

<http://localhost:56315/OutputCaching.aspx>

<http://localhost:56315/OutputCaching.aspx?CustomerID=12>

<http://localhost:56315/OutputCaching.aspx?key=98534>

اما URL های زیر، نسخه های Cache شده مختلفی از صفحه را دریافت می کنند :

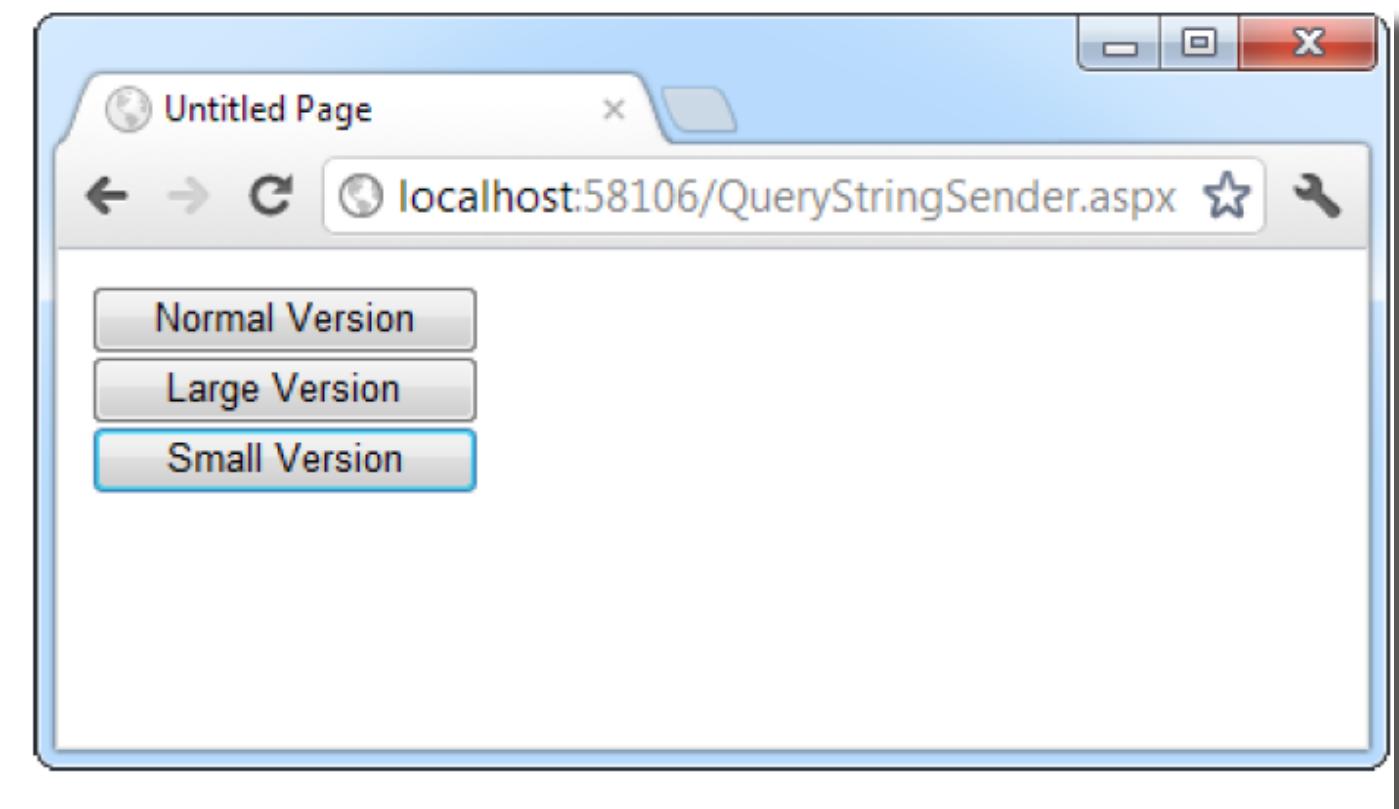
<http://localhost:56315/OutputCaching.aspx?ProductID=5>

<http://localhost:56315/OutputCaching.aspx?ProductID=452>

<http://localhost:56315/OutputCaching.aspx?CustomerID=12&CurrencyType=CDN>

## Caching هایی از

مثال زیر دارای دو صفحه برای تعیین چگونگی Cache شدن جداگانه چندین نسخه از صفحه می باشد. اولین صفحه می باشد که Cache نشده است.QueryStringSender.aspx



یک event handler برای هندل کردن هر سه دکمه در نظر گرفته شده است.

```
protected void cmdVersion_Click(Object sender, EventArgs e)
{
    Response.Redirect("QueryStringRecipient.aspx" + "?Version=" +
        ((Control)sender).ID);
}
```

صفحه QueryStringRecipient.aspx، یک پیام آشنا را به شما نشان می‌دهد و از کد زیر استفاده می‌کند:

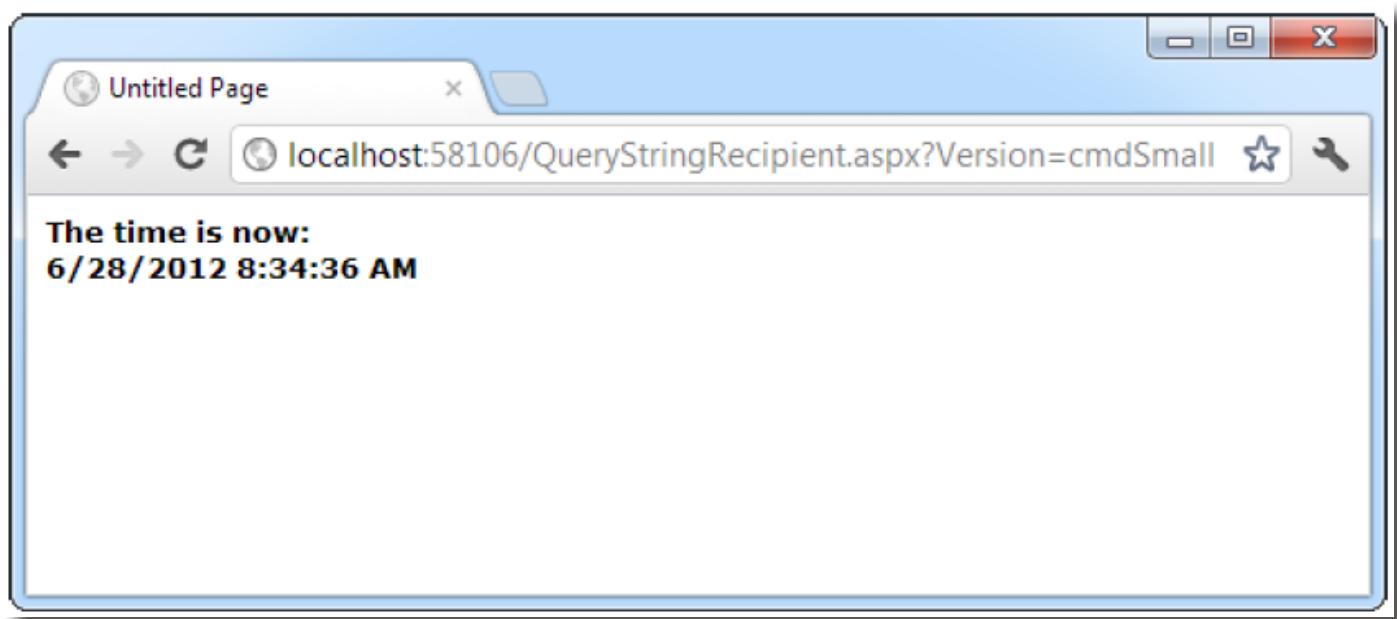
```
<%@ OutputCache Duration="60" VaryByParam="Version" %>
```

کد زیر را درون code-behind این صفحه قرار می‌دهیم:

```
protected void Page_Load(Object sender, EventArgs e)
{
    lblDate.Text = "The time is now:<br />" + DateTime.Now.ToString();
    switch (Request.QueryString["Version"])
    {
        case "cmdLarge":
            lblDate.Font.Size = FontUnit.XLarge;
            break;

        case "cmdNormal":
            lblDate.Font.Size = FontUnit.Large;
            break;

        case "cmdSmall":
            lblDate.Font.Size = FontUnit.Small;
            break;
    }
}
```



## Fragment Caching

اگر بخواهید بخشی از صفحه را Cache کنید، یک راه برای انجام آن استفاده از data caching برای ذخیره اطلاعات مورد استفاده صفحه می‌باشد. گزینه دیگر، ایجاد یک user control برای آن بخشی از صفحه می‌باشد که می‌خواهد cache کنید. سپس باید راهنمای صفحه OutputCache را به آن اضافه کنید.

## Profile Cache ها

یکی از مشکلات output caching، قرار دادن دستوراتی در کد code-behind یا markup صفحه می‌باشد. بنابراین تغییراتی مانند تغییر مدت زمان cache شدن صفحه، باید در کلیه صفحات مورد نیاز انجام شود و ASP.NET نیز آن صفحات را مجدد Cache نماید. ASP.NET، دارای یک ویژگی با نام cache profile می‌باشد که بکارگیری تنظیمات مشابهی را روی گروهی از صفحات آسان می‌کند. بنابراین می‌توانید کلیه صفحات متصل به این گروه را با تغییر پروفایل cache آنها در فایل web.config تغییر دهید.

```

<configuration>
  <system.web>
    <caching>
      <outputCacheSettings>
        <outputCacheProfiles>
          <add name="ProductItemCacheProfile" duration="60" />
        </outputCacheProfiles>
      </outputCacheSettings>
    </system.web>

```

```
</caching>
...
</system.web>
</configuration>
```

اکنون می‌توانید از این پروفایل از طریق خصیصه CacheProfile استفاده نمایید :

```
<%@ OutputCache CacheProfile="ProductItemCacheProfile" VaryByParam="None" %>
```

## Data Caching

Data caching، منعطف ترین نوع caching می‌باشد . در این روش باید، اطلاعاتی که ایجاد دوباره آنها هزینه بردار است را درون شی system collection ای با نام Cache ذخیره کنید. یکی از property های کلاس Page می‌باشد و یک نمونه از کلاس Web.Caching.Cache گرداند.

آیتم های موجود در شی Cache بصورت خودکار حذف می‌شوند. ASP.Net، یک آیتم را در صورتیکه expire شده باشد، یا سرور low-memory داده باشد، حذف می‌کند. بنابراین می‌توانید آزادانه از Cache استفاده کنید و نگران از دست رفتن حافظه با ارزش سرور نباشید، زیرا ASP.NET آیتم ها را در صورت لزوم حذف می‌کند. البته به همین دلیل نیز باید همیشه قبا از استفاده، چک کنید که آیا شی Cache موجود می‌باشد یا نه.

می‌توانید یک شی cache را به یک فایل، جدول بانک اطلاعاتی یا سایر انواع منابع وصل کنید. در صورتیکه منابع تغییر کنند، شی Cache شما نیز بصورت خودکار نامعتبر شده و release می‌شود.

## اضافه نمودن یک آیتم به Cache

از راه های مختلف می‌توان یک شی را درون cache قرار داد. می‌توانید به سادگی به آن یک key name اختصاص دهید(همانطور که برای application یا session استفاده می‌کردید) :

```
Cache[“KeyName”] = objectToCache;
```

البته با استفاده از این روش، هیچ کنترلی روی مدت زمانی که می‌خواهید شی درون cache باقی بماند ندارید. راه بهتر استفاده از متدهای Insert() می‌باشد که دارای چهار overload می‌باشد.

یکی از پرکاربردترین overload ها در زیر می‌بینید:

```
Cache.Insert(key, item, dependencies, absoluteExpiration, slidingExpiration);
```

Key : رشته ای که نامی را به این آیتم cache شده در collection اختصاص می‌دهد و می‌توانید با استفاده از آن در زمان های بعدی، این آیتم را پیدا کنید.

**Item** : شی ای که می خواهد cache کنید.

یک شی CacheDependency می باشد که امکان ایجاد یک وابستگی را برای یک آیتم در Cache بوجود می آورد. اگر نمی خواهد از آن استفاده کنید، می توانید آن را Null رد کنید.

یک شی DateTime که تاریخ و زمانی که آیتم از cache حذف خواهد شد را تعیین می کند.

یک شی TimeSpan، که تعیین می کند TimeSpan، چه مدت بین درخواست ها باید منتظر بماند و سپس آیتم cache شده را حذف کند. برای نمونه اگر مقدار آن ۲۰ دقیقه باشد، آیتم را در صورتیکه در این ۲۰ دقیقه توسط هیچ کدی استفاده نشده باشد، حذف می کند.

```
Cache.Insert("MyItem", obj, null, DateTime.Now.AddMinutes(60), TimeSpan.Zero);
```

```
Cache.Insert("MyItem", obj, null, DateTime.MaxValue, TimeSpan.FromMinutes(10));
```

 نکته: در صورت نیاز، از مدت زمان زیاد برای cache نترسید. در مثال های مایکروسافت حتی تا ۱۰۰ دقیقه هم برای آن در نظر گرفته شده است.

## یک Cache ساده

```
public partial class SimpleDataCache : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        if (this.IsPostBack)
        {
            lblInfo.Text += "Page posted back.<br />";
        }
        else
        {
            // Your code here
        }
    }
}
```

```

lblInfo.Text += "Page created.<br />";

}

if (Cache["TestItem"] == null)
{
    lblInfo.Text += "Creating TestItem...<br />";

    DateTime testItem = DateTime.Now;

    lblInfo.Text += "Storing TestItem in cache ";

    lblInfo.Text += "for 30 seconds.<br />";

    Cache.Insert("TestItem", testItem, null,
        DateTime.Now.AddSeconds(30), TimeSpan.Zero);

}
else
{
    lblInfo.Text += "Retrieving TestItem...<br />";

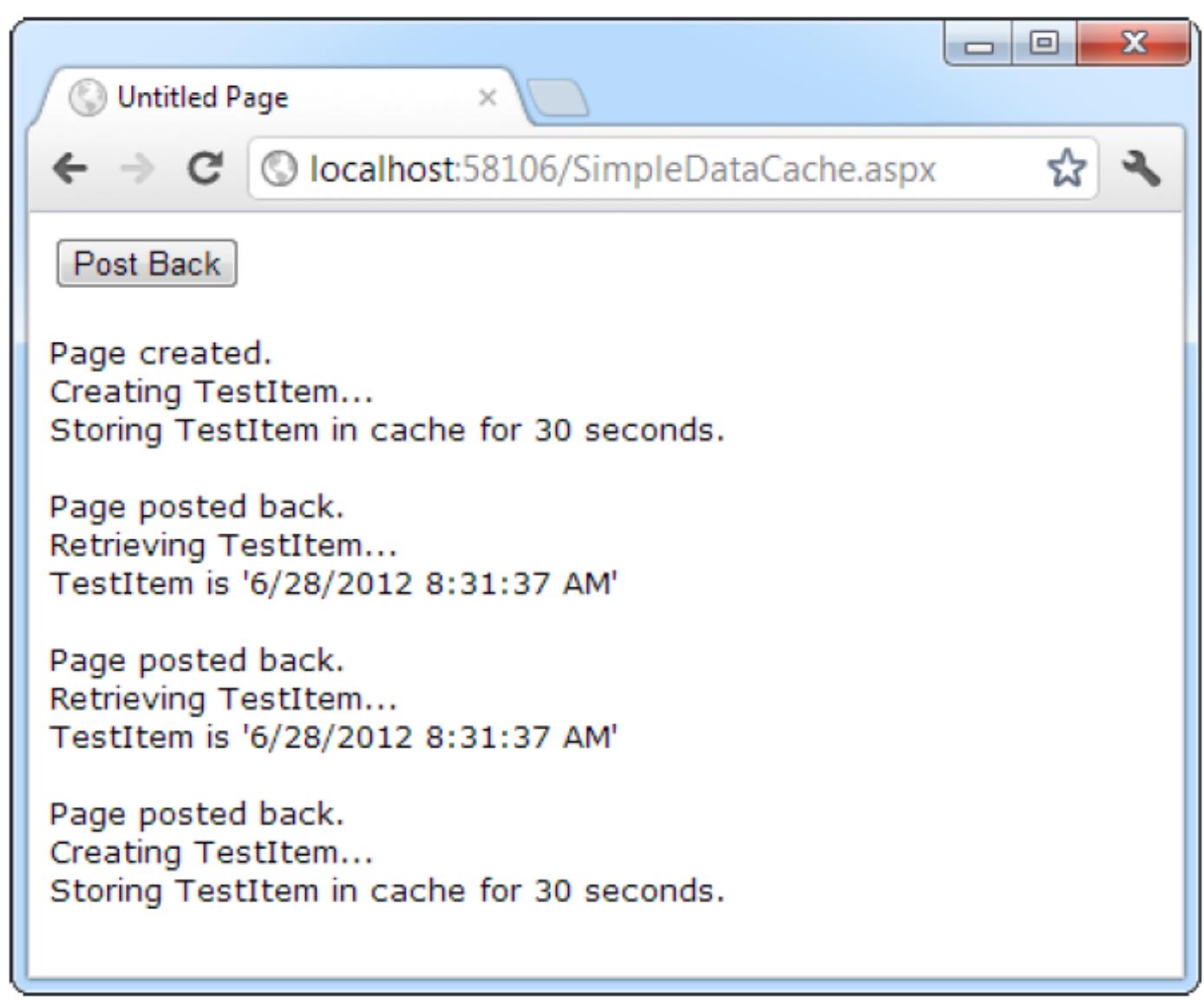
    DateTime testItem = (DateTime)Cache["TestItem"];

    lblInfo.Text += "TestItem is " + testItem.ToString();

    lblInfo.Text += "<br />";

}
lblInfo.Text += "<br />";
}
}

```



## View برای ایجاد چند Caching

در مثال زیر اطلاعاتی را از بانک اطلاعاتی بازیابی و درون یک DataSet ذخیره می‌کنیم. این اطلاعات توسط یک GridView نمایش داده می‌شوند. خروجی صفحه نمی‌تواند cache شود، زیرا ستون های به نمایش در آمده در GridView توسط کاربر قابل تغییر است. شما که نمی‌توانید برای هر یک از حالت‌ها خروجی را Cache کنید. تنها برای ۱۰ ستون باید هزاران حالت را در نظر بگیرید.

بنابراین به جای تلاش برای استفاده از output caching، این صفحه، شی Cache را dataSet می‌کند.

Hide Columns:

<input type="checkbox"/> CustomerID	<input type="checkbox"/> Region
<input checked="" type="checkbox"/> CompanyName	<input type="checkbox"/> PostalCode
<input type="checkbox"/> ContactName	<input checked="" type="checkbox"/> Country
<input checked="" type="checkbox"/> ContactTitle	<input checked="" type="checkbox"/> Phone
<input type="checkbox"/> Address	<input checked="" type="checkbox"/> Fax
<input type="checkbox"/> City	

**Filter and Show**

Retrieved from cache.

CustomerID	ContactName	Address	City	Region	PostalCode
ALFKI	Maria Anders	Obere Str. 57	Berlin		12209
ANATR	Ana Trujillo	Avda. de la Constitución 2222	México D.F.		05021
ANTON	Antonio Moreno	Mataderos 2312	México D.F.		05023
AROUT	Thomas Hardy	120 Hanover Sq.	London		WA1 1DP
BERGS	Christina Berglund	Berguvsvägen 8	Luleå		S-958 22
BLAUS	Hanna Moos	Forsterstr. 57	Mannheim		68306
BLONP	Frédérique Citeaux	24, place Kléber	Strasbourg		67000
POLTD	Martín Sommer	C/ Araquil, 67	Madrid		28022

```
private DataSet RetrieveData()
{
    string connectionString =
        WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
    string SQLSelect = "SELECT * FROM Customers";
    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand(SQLSelect, con);
```

```

SqlDataAdapter adapter = new SqlDataAdapter(cmd);

DataSet ds = new DataSet();

try
{
    con.Open();

    adapter.Fill(ds, "Customers");

}
finally
{
    con.Close();
}

return ds;
}

```

اکنون باید چک کرد که آیا DataSet درون Cache موجود می‌باشد یا نه.

```

private DataSet GetDataSet()
{
    DataSet ds = (DataSet)Cache["Customers"];
    // Contact the database if necessary.

    if (ds == null)
    {
        ds = RetrieveData();

        Cache.Insert("Customers", ds, null, DateTime.MaxValue,
        TimeSpan.FromMinutes(2));

        lblCacheStatus.Text = "Created and added to cache.";
    }
    else
    {
        lblCacheStatus.Text = "Retrieved from cache.";
    }

    return ds;
}

```

}

دفعه اولی که صفحه لود می‌شود، متده DataSet() برای بازیابی فراخوانی می‌شود.

```
if (!this.IsPostBack)
{
    DataSet ds = GetDataSet();

    chkColumns.DataSource = ds.Tables["Customers"].Columns;

    chkColumns.DataTextField = "ColumnName";

    chkColumns.DataBind();
}
```

برای فیلتر کردن ستون ها می‌توانید از کد زیر استفاده کنید :

```
protected void cmdFilter_Click(Object sender, EventArgs e)
{
    DataSet ds = GetDataSet();

    // Copy the DataSet so you can remove columns without
    // changing the cached item.

    ds = ds.Copy();

    foreach (ListItem item in chkColumns.Items)
    {
        if (item.Selected)
        {
            ds.Tables[0].Columns.Remove(item.Text);
        }
    }

    gridCustomers.DataSource = ds.Tables[0];

    gridCustomers.DataBind();
}
```

## Caching با وابستگی ها

با گذشت زمان، اطلاعات موجود در data source تغییر می‌کند. اگر کد شما از cache استفاده کند، ممکن است شما از این تغییرات اطلاع پیدا نکنید و از اطلاعات منسخه شده در cache استفاده کنید. برای حل این مشکل از ویژگی cache dependencies موجود در ASP.NET استفاده می‌کنیم. این ویژگی به شما امکان ایجاد یک آیتم cache شده که وابسته به سایر منابع می‌باشد را خواهد داد. بنابراین اگر آن منبع تغییر کند، آیتم cache شده بصورت خودکار حذف می‌شود.

ASP.NET، سه نوع از dependency را ارائه می‌دهد:

- وابستگی به فایل یا فolder
- وابستگی به سایر آیتم های cache شده
- وابستگی به query بانک اطلاعاتی

### وابستگی به فایل

ابتدا باید یک شی CacheDependency ایجاد کنید.

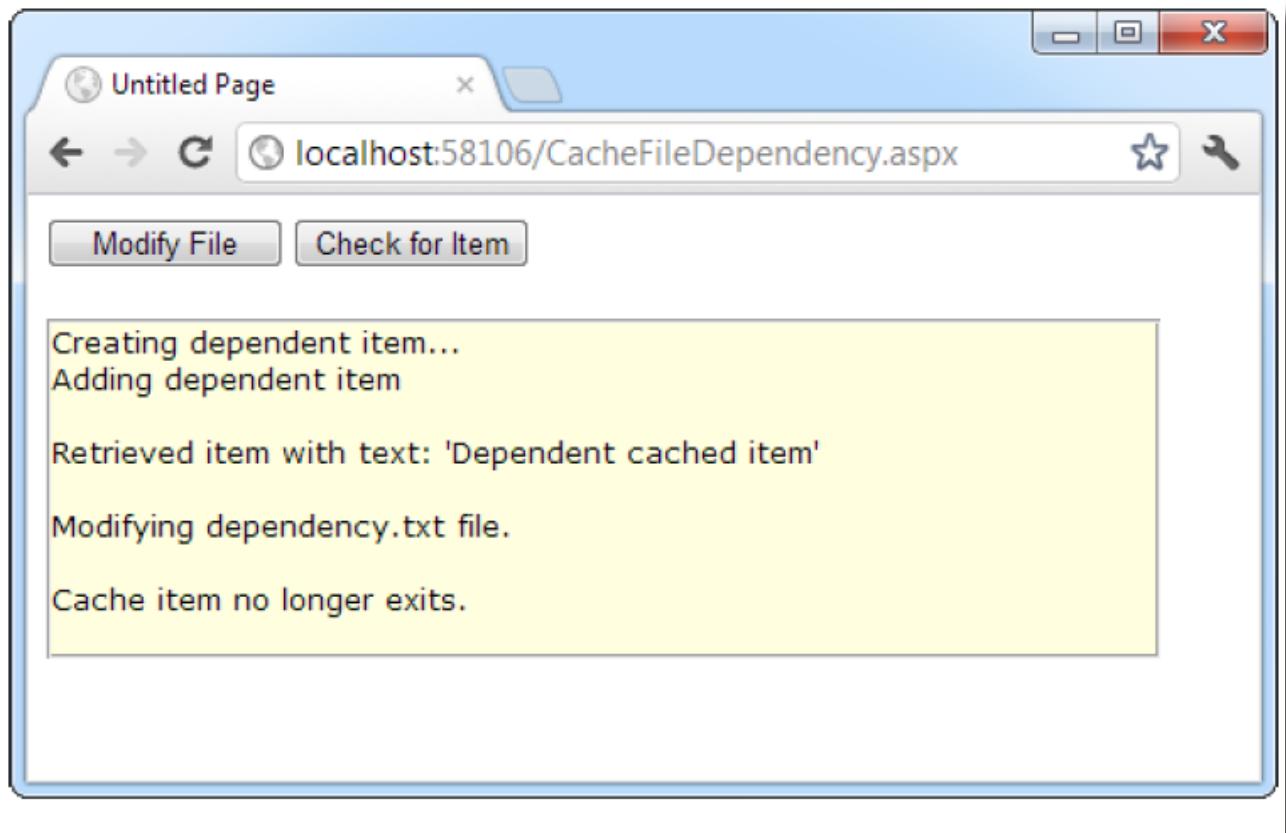
کد زیر یک CacheDependency ایجاد می‌کند که به یک فایل XML با نام ProductList.xml وابسته می‌باشد. زمانی که فایل CacheDependency XML تغییر کند، نامعتبر می‌شود و آیتم cache شده وابسته به آن نیز حذف می‌شود.

```
// Create a dependency for the ProductList.xml file.

CacheDependency prodDependency = new CacheDependency(
    Server.MapPath("ProductList.xml"));

// Add a cache item that will be dependent on this file.

Cache.Insert("ProductInfo", prodInfo, prodDependency);
```



## SQL Server Cache Dependency

این ویژگی، امکان نامعتبر کردن خودکار یک **DataSet** (مانند **data object**) را در هنگامی که داده های مرتبط با آن در بانک اطلاعاتی تغییر می کنند را دارد.

برای درک اینکه **database dependency** ها چگونه کار می کنند، ابتدا باید اندکی درباره سیستم پیام دهی موجود در **Service Broker** با نام **Service Broker** آشنا شوید.

**Service Broker**، **صف هایی** (Queues) را که شامل اشیای بانک اطلاعاتی می باشند که دارای جداول، **Store Procedure** ها یا **View** های یکسان هستند، مدیریت می کند.

با استفاده از **Queue**، می توانید به **SQL Server** دستور دهید که چه پیام های اطلاع رسانی را برای رویدادهای خاص با استفاده از دستور **Create Command Notification** ارسال کند. اما، **ASP.NET**، یک مدل مناسب تر سطح بالا را پیشنهاد می دهد. می توانید یک **query** را ثبت کنید و **ASP.NET** بصورت خودکار به **SQL Server** دستور می دهد تا پیام های اطلاع رسانی را برای هر عملیاتی که روی نتیجه **query** تأثیر می گذارد، ارسال نماید. هر زمان که یک عملیات را اجرا کنید، **SQL Server**، تعیین می کند که آیا عملیات شما روی دستور ثبت شده اثر می گذارد یا نه.

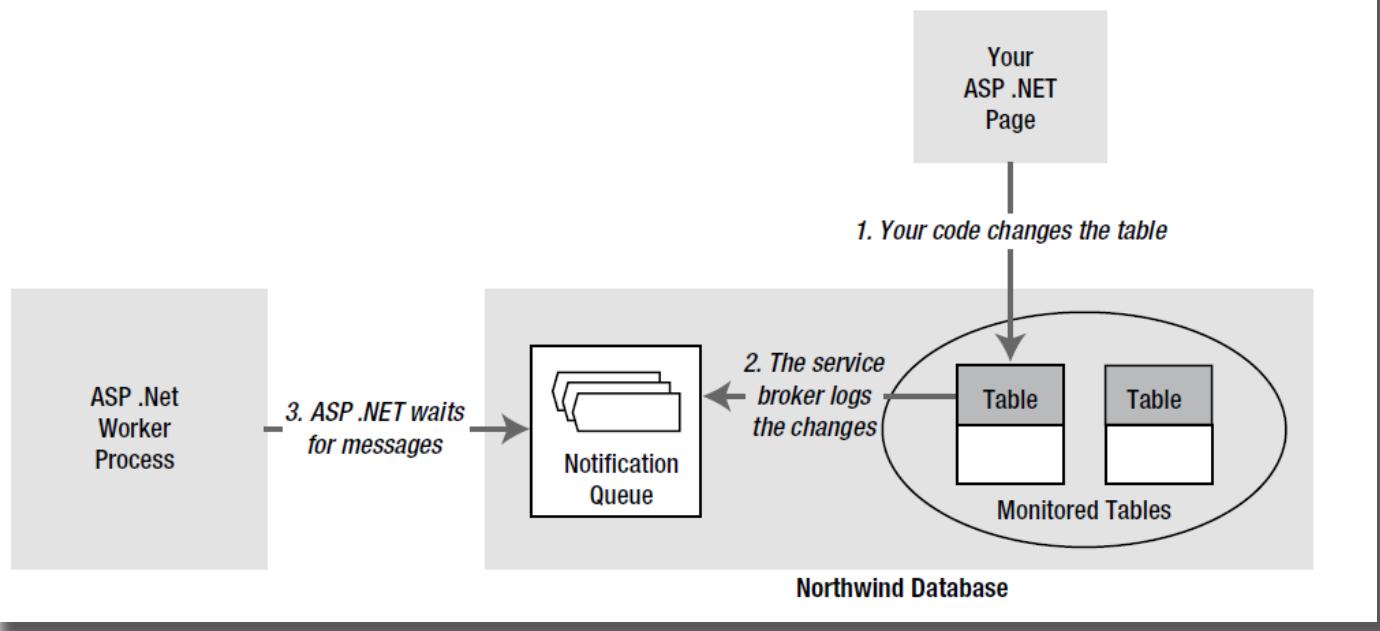
**Notification** ها (پیام های اطلاع رسانی)، با دستورات **Select** و **Store Procedure** کار می کنند. برای پشتیبانی از **notification**، دستورات شما باید قوانین زیر را رعایت کنید:

- باید نام کامل جداول را با فرم **[dbo.Employee Owner].table** مانند **dbo.Employee** وارد کنید.

- **Query** شما نمی توانند از توابع **aggregate** استفاده کند.

- نمی توانید همه ستون ها را با \* انتخاب کنید. باید نام هر ستون را ذکر کنید.

```
SELECT EmployeeID, FirstName, LastName, City FROM dbo.Employees
```



## فعال سازی Service Broker

این ویژگی بصورت پیش فرض در SQL Server غیر فعال می باشد. باید برای مشاهده تغییرات در بانک اطلاعاتی و تحويل پیام های اطلاع رسانی به Queue های مناسب، این سرویس را فعال کنید.

ابتدا پنجره C:\Program Files\Microsoft SQL Server\110\Tools\Binn را از پوشش command prompt سپس sqlcmd.exe را اجرا کنید.

```
sqlcmd -S (localdb)\v11.0
```

دد بالا به SQL Server Express وصل می شود. اگر بانک شما روی سرور دیگری نصب شده است، نام آن کامپیوتر را به جای loc- قرار دهید.

ابزار sqlcmd.exe یک command propmt SQL را در آن وارد نمایید.

```
USE Northwind
```

```
ALTER DATABASE Northwind SET ENABLE_BROKER
```

```
GO
```

## Caching Service اولیه

قبل از استفاده از Cache Dependency با `SqlDependency.Start()` SEL Server را فراخوانی کنید. این کار را روی وب سرور، مقدار دهی اولیه می کند.

```
string connectionString = WebConfigurationManager.ConnectionStrings[
    "Northwind"].ConnectionString;
SqlDependency.Start(connectionString);
```

تنها باید یکبار متده استارت Application\_Start() از فایل Global.asax قرار دهید. بنابراین بصورت خودکار trigger می شود.

## ایجاد Cache Dependency

اگر باید دستوری را که داده شما را بازیابی می کند، مشخص کنید. بنابراین SQL Server، بازه ای از رکوردها که می خواهد مانیتور کنید را تشخیص می دهد.

برای تعیین این دستور، با استفاده از سازنده شی `SQLCacheDependency` که یک پارامتر می گیرد، `SQLCommand` را ایجاد کنید.

```
//Create the ADO.NET Objects
SqlConnection con = new SqlConnection(connectionString);
string query = "Select EmployeeID, FirstName, LastName, City From dbo.Employees";
SqlCommand cmd = new SqlCommand(query, con);
SqlDataAdapter adapter = new SqlDataAdapter(cmd);

//Fill the DataSet
DataSet ds = new DataSet();
adapter.Fill(ds, "Employees");

// Create the dependency.
SqlCacheDependency empDependency = new SqlCacheDependency(cmd);
// Add a cache item that will be invalidated if one of its records changes
// (or a new record is added in the same range).
Cache.Insert("Employees", ds, empDependency);
```

اکنون، زمانی که شما داده درون جدول را تغییر دهید، notification تحويل داده می‌شود و آیتم از Cache حذف می‌شود. دفعه بعدی که این را ایجاد کنید، باید آن را به همراه یک DataSet اضافه کنید.

## Failed Notification

اگر آیتم شما هرگز Expire نشده باشد، پیام invalidation ASP.NET Pooling Service را دریافت نمی‌کند زیرا ممکن است در بانک اطلاعاتی شما، CLR، فعال نباشد. Procedure هایی که پیام‌ها را ارسال می‌کنند.NET Procedure می‌باشند، بنابراین باید CLR فعال باشد.

برای فعال نمودنCLR سازی، پنجه‌های Command Prompt باز کرده و sqlcmd.exe را اجرا کنید و سپس دستورات زیر را در آن بنویسید:

```
sqlcmd -S (localdb)\v11.0
```

```
EXEC sp_configure 'show advanced options', '1'
```

```
GO
```

```
RECONFIGURE
```

```
GO
```

```
EXEC sp_configure 'clr enabled', 1
```

```
GO
```

```
RECONFIGURE
```

```
GO
```

# بخش پنجم : ASP.NET پیشرفته

فصل شانزدهم : برنامه نویسی مبتنی بر کامپونت

~~Caching~~ فصل هفدهم

Entity Framework , LINQ : فصل هجدهم

~~ASP.NET AJAX~~ فصل نوزدهم

ASP.NET ~~کردن برنامه های~~ Deploy : فصل بیست و یکم



## Entity Framework و LINQ

LINQ (Language Integrated Query) یک extention (افزونه) از زبان برای query روی داده می‌باشد. به کاربران، امکان جستجوی یک شی در یک Collection، را می‌دهد. امتیاز LINQ، پتانسیل استفاده از منابع داده مختلف مانند فایل‌های XML و بانک‌های اطلاعاتی می‌باشد. علاوه بر جستجو و مرتب‌سازی مجموعه ای از داده‌های in-memory، این موارد روی بانک اطلاعاتی نیز اجرا می‌کند.

### آشنایی با LINQ

LINQ، کلمات کلیدی را تعریف می‌کند که می‌توانید از آن برای انتخاب، فیلتر، مرتب‌سازی، گروه‌بندی و انتقال داده‌ها استفاده کنید. LINQ، اجازه می‌دهد این کارها را با انواع مختلفی از داده انجام دهید. در زیر برخی از LINQ Provider های موجود در .NET 4.5 را مشاهده می‌کنید :

**LINQ to Object** : ساده‌ترن فرم Query زدن روی LINQ می‌باشد. اجازه Query را می‌دهد. (مانند array, arraylist, dictionary).

**LINQ to Dataset** : شبیه مورد بالا می‌باشد با این تفاوت که اشیای DataTable را از DataRow بیرون می‌کشد.

**LINQ to XML** : اجازه جستجوی المان‌های موجود در XElement و XDocument را خواهد داد. در حقیقت جستجوهای قدرتمند تری را روی داده in-memory xml انجام می‌دهد.

**LINQ to Entities** : اجازه اجرای query های بانک اطلاعاتی با استفاده از عبارات LINQ را می‌دهد.

### اصول LINQ

در مثال زیر را LINQ to Object آشنا می‌شوید:

```
public class Employee
{
    public int EmployeeID { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }

    public string TitleOfCourtesy { get; set; }

    public Employee(int employeeID, string firstName, string lastName,
        string titleOfCourtesy)
    {
    }
}
```

```

EmployeeID = employeeID;

FirstName = firstName;

LastName = lastName;

TitleOfCourtesy = titleOfCourtesy;

}

}

.

// Create the collection.

List<Employee> employees = new List<Employee>();

// Fill the collection.

employees.Add(new Employee(1, "Nancy", "Davolio", "Ms."));

employees.Add(new Employee(2, "Andrew", "Fuller", "Dr."));

employees.Add(new Employee(3, "Janet", "Leverling", "Ms."));

```

در این مثال داده های هر شی Employee، بصورت Hard-Code مقدار دهی شده است. البته می توانید به خواست خود این مثال را با خواندن داده از بانک اطلاعاتی، فایل XML یا سایر منابع انجام دهید. برای گرفتن داده از collection، می توانید از LINQ to Object استفاده کنید.

فرض کنید، می خواهید لیستی از کلیه employee هایی که دارای یک lastname که با حرف D شروع می شوند باشید. روش قبلی، استفاده از یک حلقه روی employee collection و اضافه نمودن موارد منطبق با شرط به یک collection دوم می باشد:

```

// Create the source collection.

List<Employee> employees = new List<Employee>();

// (Code for filling the collection omitted to save space.)

// Find the matching employees.

List<Employee> matches = new List<Employee>();

foreach (Employee employee in employees)

{
    if (employee.LastName.StartsWith("D"))

    {
        matches.Add(employee);
    }
}

```

نتیجه را می‌توانید در یک gridview نمایش دهید:

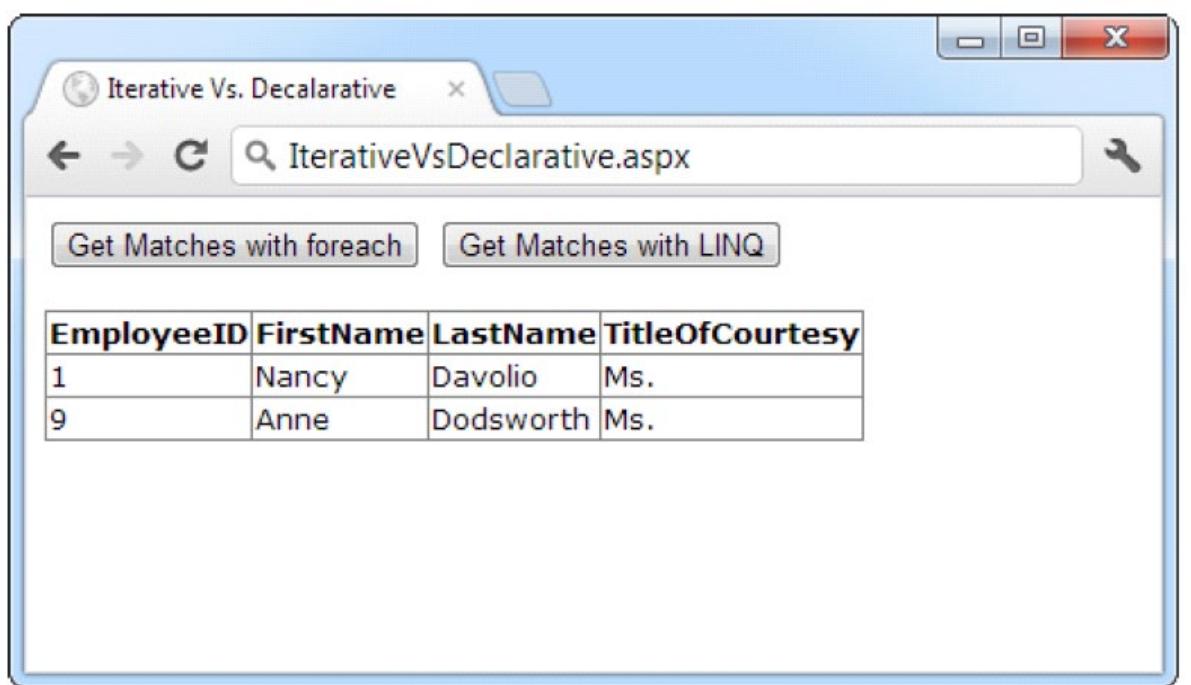
```
gridEmployees.DataSource = matches;
gridEmployees.DataBind();
```

امکان جایگزینی دستورات حلقه تکرار (مانند foreach) را با یک عبارت می‌دهد.

```
// Create the source collection.
List<Employee> employees = new List<Employee>();
// (Code for filling the collection omitted to save space.)
var matches = from employee in employees
              where employee.LastName.StartsWith("D")
              select employee;
gridEmployees.DataSource = matches;
gridEmployees.DataBind();
```

نتیجه حاصل شده با کد قبل برابر است.

LINQ می‌تواند روی هر collection که اینترفیس IEnumerable را پیاده می‌کند، اجرا شود. بنابراین می‌توانید var را با <IEnumerable<Employee> جایگزین کنید.



## LINQ عبارات

باید با نحوه ساخت عبارت LINQ آشنا شوید. تمامی عبارات LINQ باید از واژه from که منبع داده را تعیین می‌کند و select که داده مورد نظر شما را برای بازیابی تعیین می‌کند، تشکیل شوند.

```
var matches = from employee in employees
...

```

عبارة from، دو بخش از اطلاعات را تعیین می‌کند. کلمه ای بعد از in می‌آید، منبع داده را مشخص می‌کند. که در این مثال یک collection با نام employee می‌باشد. کلمه ای که بعدی از from می‌آید، یک نام مستعار می‌باشد که آیتم‌های مستقل در منبع داده را تعیین می‌کند. بنابراین هر EmployeeDetails را در این مثال employee می‌نامیم.

```
var matches = from employee in employees
    select employee;
```

برای آشنایی بیشتر با سایر عملیات‌های LINQ، به آدرس زیر مراجعه نمایید :

<http://msdn.microsoft.com/vcsharp/aa336746.aspx>

## Projection

توانایی تبدیل داده‌ای که حاصل query می‌باشد به نتیجه‌ای با ساختار متفاوت را projection گویند. می‌توانید عبارت select را برای گرفتن زیر مجموعه‌ای از داده‌ها تغییر دهید. برای نمونه، می‌توانید لیستی از firstname را بیرون بکشید :

```
var matches = from employee in employees
    select employee.FirstName;
...
var matches = from employee in employees
    select employee.FirstName + " " + employee.LastName;
```

یا

همانطور که در زیر می‌بینید، می‌توانید از استاندارد اپراتورهای C# روی داده‌های عددی یا رشته‌ای برای تغییر اطلاعاتی که می‌خواهید انتخاب کنید استفاده نمایید.

حتی می‌توانید بصورت پویا یک کلاس جدید که این اطلاعات(اطلاعاتی که می‌خواهید برگردانید) را در خود قرار می‌دهد، ایجاد کنید. برای نمونه اگر بخواهید `firstname` و `lastname` را بازیابی کنید اما آنها را در یک `string` جداگانه قرار دهید، می‌توانید یک نسخه از کلاس `EmployeeDetails` که شامل یک `property` با نام `FirstName` و `LastName` می‌باشد را ایجاد کنید. برای انجام این کار از یکی از ویژگی‌های `c#` با نام `anonymous types` استفاده می‌کنیم. ابتدا باید یک کلمه `new` به دستور `select` اضافه کنید و سپس یک `{}` را در ادامه بیاورید. درون `{}` می‌توانید هر `property` که می‌خواهید در نتیجه قرار بگیرد را انتخاب نمایید.

```
var matches = from employee in employees
              select new { First = employee.FirstName,
                           Last = employee.LastName };
```

این عبارت پس از اجرا، مجموعه‌ای از اشیا که از کلاس ایجاد شده استفاده می‌کنند را برمی‌گرداند. هر شی دارای دو `property` می‌باشد. `first` ، `last` . تعریفی از کلاس را نمی‌بینید زیرا این کلاس توسط کامپایلر در زمان اجرا تولید می‌شود.

First	Last
Nancy	Davolio
Andrew	Fuller
Janet	Leverling
Margaret	Peacock
Steven	Buchanan
Michael	Suyama
Robert	King
Laura	Callahan
Anne	Dodsworth

برای استفاده از projection باید از anonymous type استفاده کنید. البته می‌توانید نوع مورد نظر خود را تعریف و سپس در عبارت از آن استفاده کنید.

```
public class EmployeeName
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
}

var matches = from employee in employees
    select new EmployeeName
    {
        FirstName = employee.FirstName,
        LastName = employee.LastName
    };

```

البته می‌توانید از روش زیر نیز برای مقدار دهی این property‌ها در عبارت استفاده نمایید:

```
var matches = from employee in employees
    select new EmployeeName(firstName, lastName);
```

## فیلتر کردن و مرتب سازی

با استفاده از کلمه where می‌توانید نتیجه را فیلتر کنید. برای نمونه می‌توانید employee‌هایی را که دارای lastname می‌باشند که با حرف D شروع می‌شوند را فیلتر کنید:

```
var matches = from employee in employees
    where employee.LastName.StartsWith("D")
    select employee;
```

می‌توانید چندین شرط را با استفاده از (&&) و (||) ترکیب کنید و یا از <,=> استفاده نمایید.

```
var matches = from product in products
    where product.UnitsInStock > 0 && product.UnitPrice > 3.00
    select product;
```

یکی از ویژگی های عبارات LINQ این است که می توانید به سادگی از متدهای خود در آنها استفاده کنید. برای نمونه می توانید متدهای `TestEmployee()` را نوشه و در عبارت استفاده کنید:

```
private bool TestEmployee(Employee employee)

{
    return employee.LastName.StartsWith("D");
}

var matches = from employee in employees
    where TestEmployee(employee)
    select employee;
```

اپراتور `orderby`, نیز برای مرتب سازی استفاده می شود.

```
var matches = from employee in employees
    orderby employee.LastName, employee.FirstName
    select employee;
```

همچنین می توانید کلمه `descending` را بعد از نام فیلد برای مرتب سازی معکوس اضافه نمایید:

```
var matches = from employee in employees
    orderby employee.LastName descending,
    employee.FirstName descending
    select employee;
```

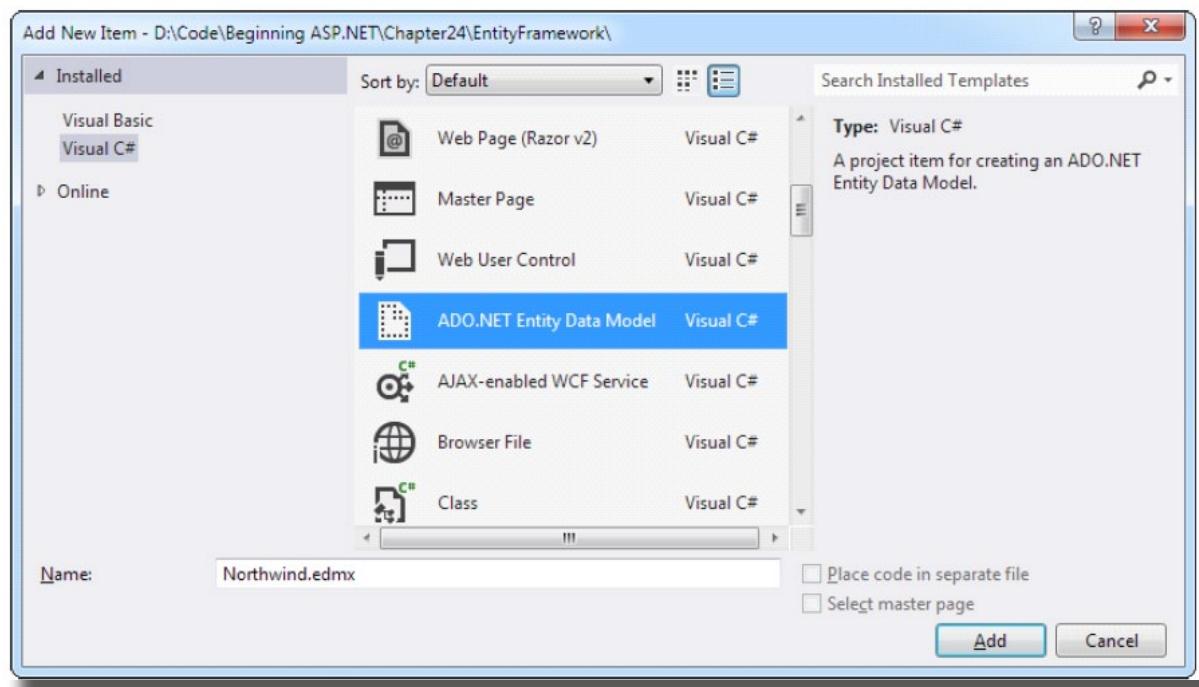
دوسستان عزیز، این فصل صرفاً جهت آشنایی و اطلاع شما از وجود LINQ و نحوه استفاده ابتدایی آن می باشد. مسلماً فراگیری موارد بیشتر و حرفه ای تر نیازمند مطالعه کتاب ویژه ای در زمینه LINQ می باشد.

## استفاده از Entity Framework

به شما اجازه ایجاد کد بر اساس ساختار بانک اطلاعاتی را می دهد. با استفاده از EF، شما نگران کlassenها و کدهای ADO.NET نیستید بلکه از دستورات LINQ to Entity برای کار با بانک اطلاعاتی استفاده می کنید. البته استفاده از آن مقداری سربار به همراه خواهد آورد و اندکی در سرعت شما تأثیر گذار خواهد بود.

## ایجاد Entity Data Model

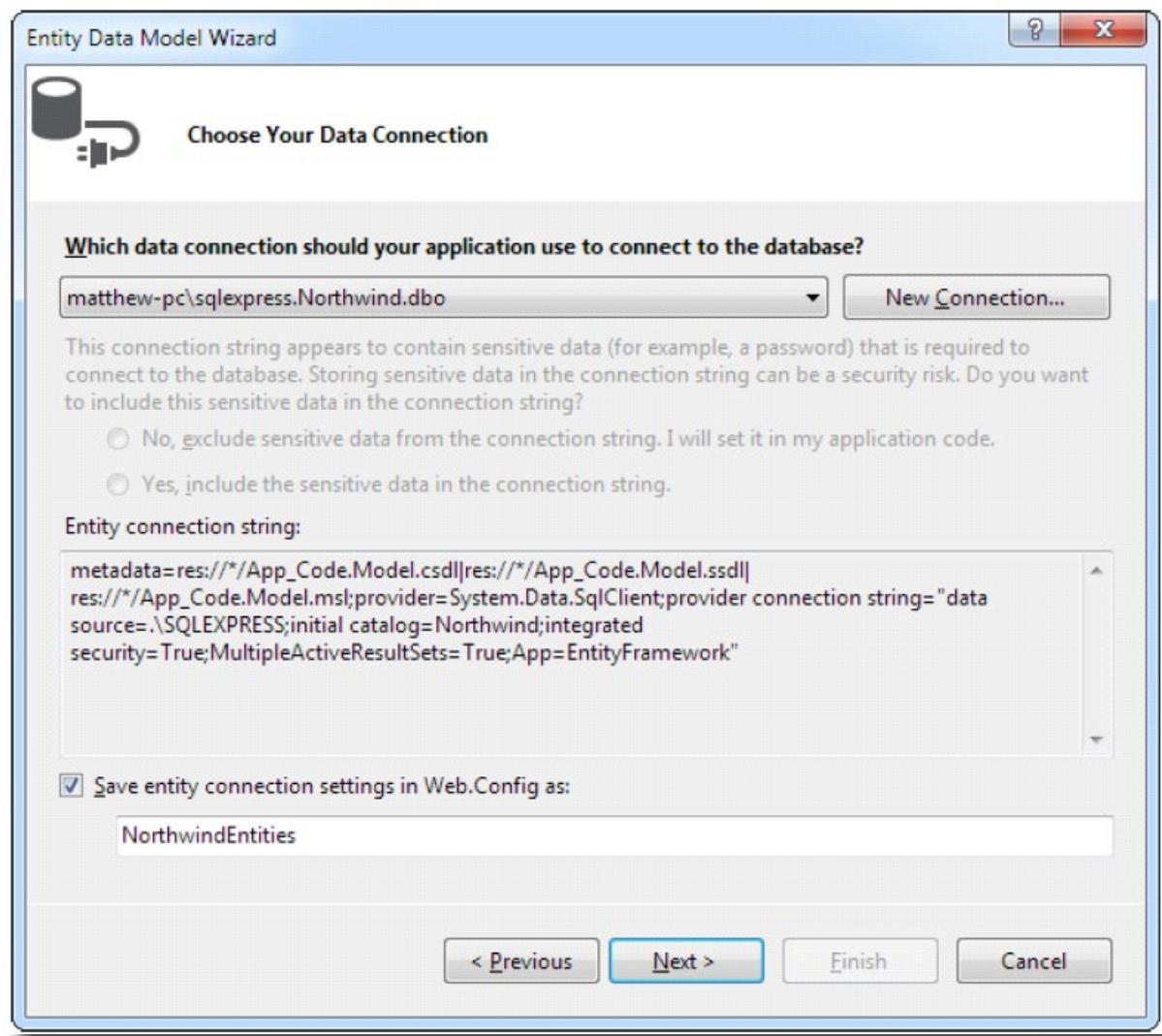
ابتدا باید entity data mode را به برنامه وب خود اضافه نمایید. روی پروژه کلیک راست کرده و Add New Item را انتخاب نمایید. Northwind.edmx را انتخاب کنید و نامی را برای انتخاب نمایید. به عنوان مثال گزینه ADO.NET Entity Data Model



سپس، VS Entity Data Model Wizard را نمایش می‌دهد. در مرحله اول باید تعیین کنید که می‌خواهید مدل شما از یک بانک اطلاعاتی موجود درست شود یا می‌خواهید آن را بصورت دستی تعریف نمایید.

 نکته: entity، واژه دیگری برای یک data object می‌باشد. هر entity اطلاعاتی درباره یک آیتم داده مشخص را نگهداری می‌کند. (به زبان ساده هر entity با یک رکورد جدول مرتبط می‌باشد)

در مرحله بعد، باید connection به بانک اطلاعاتی را انتخاب کنید. اگر قبلاً این اتصال را در server explorer تعريف کرده باشید، بصورت خودکار آن را در drop-down list مشاهده خواهید کرد. اگر نه، باید روی New Connection کلیک کنید.



VS، رشته اتصال را در بخش `<connectionStrings>` از فایل `web.config` قرار می‌دهد که می‌توانید آن را در صورت لزوم بعداً تغییر دهید. بصورت پیش فرض نام آن به همراه `Entities` در انتهای آن تشکیل می‌شود. البته می‌توانید نام دیگری برای آن تعريف کنید. در مرحله سوم، VS، کاتالوگ تمامی جداول، `view`‌ها و `storeprocedure`‌ها را به شما نمایش می‌دهد تا موارد مورد نیاز را انتخاب نمایید.

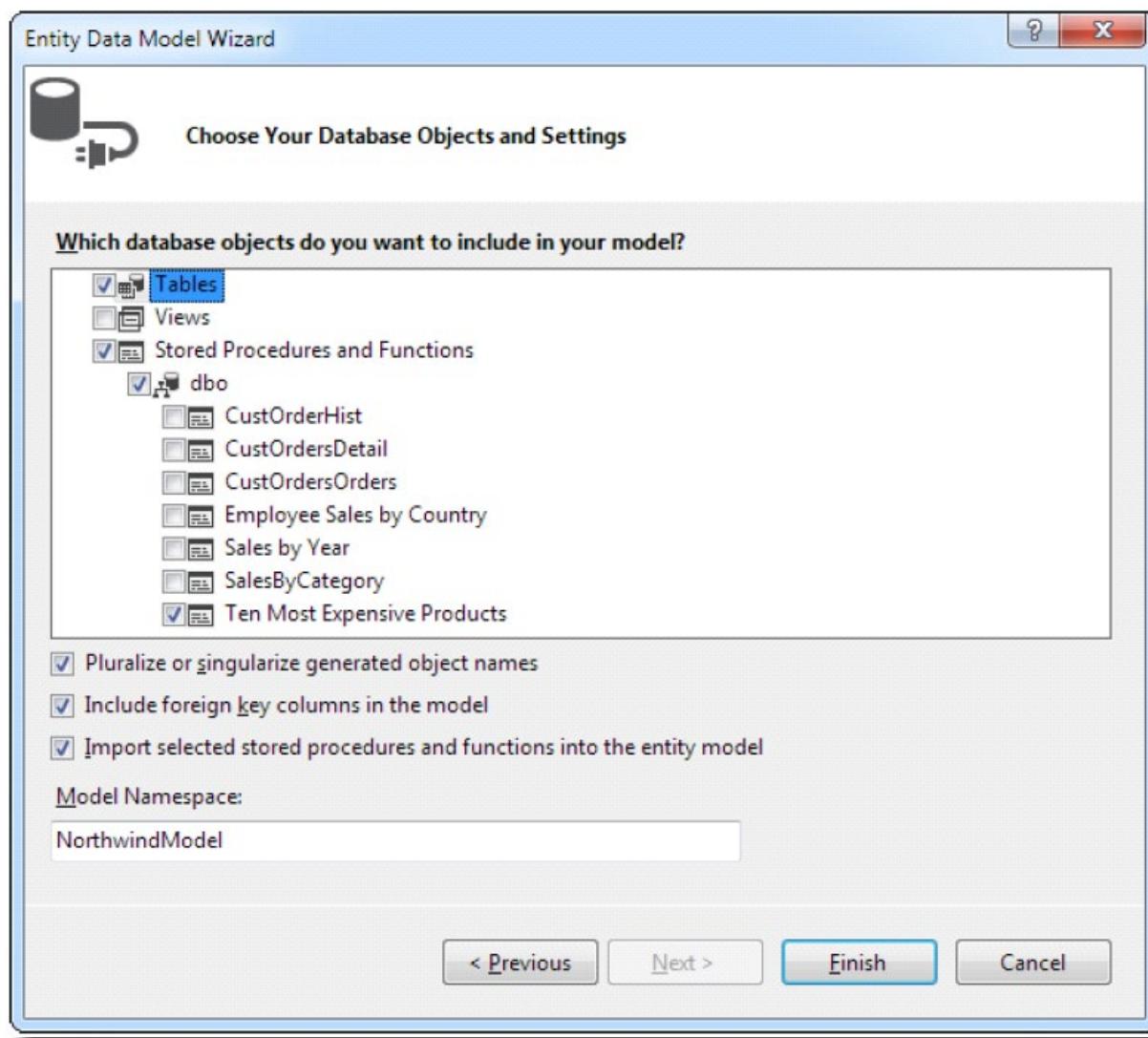
در این مرحله دو گزینه دارید:

**Pluralize or singularize object names**: اگر از این گزینه استفاده کنید، VS، یک آیتم `Product` برای ارائه هر رکورد در جدول `Products` ایجاد می‌کند. یک ویژگی `Products` برای ارائه مجموعه‌ای از محصولاتی که با رکورد `ProductCategory` در ارتباط هستند نیز اضافه می‌شود. بطور خلاصه، VS، بصورت خودکار، از نام‌های واضح و رایج استفاده می‌کند. حرف `s` را به نامی اضافه یا کم می‌کند. برای نمونه برای کلماتی مانند `Address` و `Person` استفاده می‌کند. در حقیقت اگر نام جدول شما مفرد باشد در مدل بصورت جمع می‌آید و اگر جمع باشد بصورت مفرد می‌آید.

**Include foreign-key columns**: این گزینه تعیین می‌کند که آیا VS، کلید‌های خارجی را در `data model` ایجاد شده قرار دهد یا نه. اگر آن را انتخاب نکنید، هر `entity`، یک شی کاملاً مجزا و بدون هیچ ارتباطی با سایر `entity`‌ها خواهد بود. اگر آن را انتخاب کنید، می‌توانید از طریق `relationship`‌ها، بین `property`‌ها حرکت کرد. برای نمونه، قادر خواهید بود کلیه محصولات موجود

در یک گروه مشخص را با استفاده از navigation property ای Products مانند که بصورت خودکار ایجاد شده است، پیدا کنید.

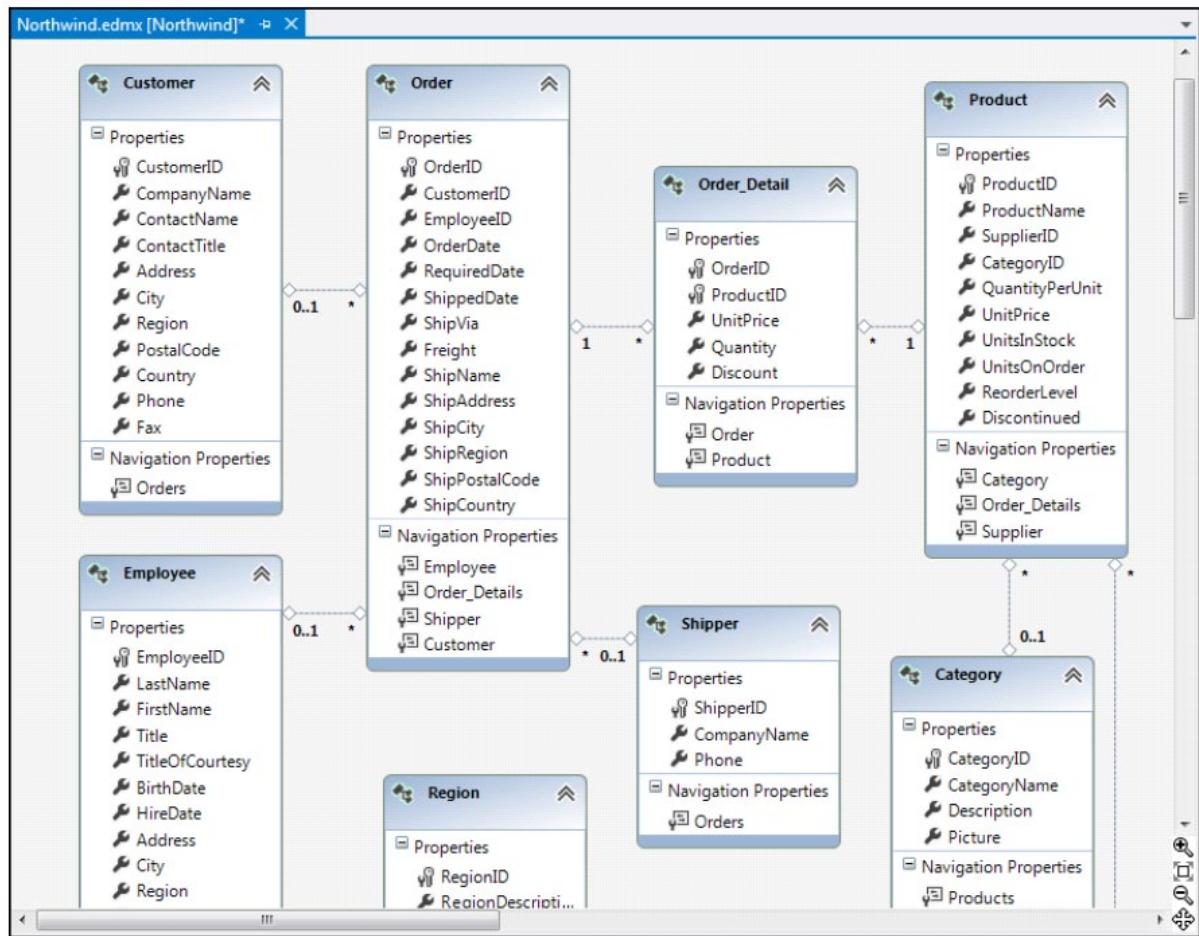
در نهایت روی Finish کلیک کنید. VS، بصورت خودکار، مدل را ایجاد می‌کند. درون فolder App\_Code، دارای دو فایل خواهد بود: یکی فایل مد شما (برای مثال، Northwind.edmx) و دیگری فایل که شامل کدهای C# بصورت خودکار تولید شده (. Northwind.Designer.cs )



## Data Model Diagram

فایل .edmx، یک فایل xml می‌باشد که جزئیات ساختار بانک اطلاعاتی شما را نگهداری می‌کند. با نمونه، همه جداولی که انتخاب کردید را به همراه relationship ها و data type های آن شامل می‌شود.

VS، یک محیط طراحی گرافیکی که محتوای موجود در فایل .edmx را نمایش می‌دهد در اختیار شما قرار داده است.



- هر کادر در بالا معادل یک entity میباشد که با یک جدول در بانک مرتبط است.

- هر entity، شامل دو نوع از property میباشد. نوع اول data property که با فیلد های موجود در جدول مرتبط میباشد. نوع دوم navigation property میباشد که امکان پرسش از یک جدول به رکورد مرتبط در جدول دیگر را میدهد.

- خطوط نقطه ای بین جداول، ارتباط ها را مشخص میکنند. (ارتباطات 0..1 و...)

در زیر برخی از تغییراتی که میتوانید ایجاد کنید آمده است :

- بازچینی entity ها : میتوانید برای مشاهده بهتر مدل، محل entity ها را مطابق میل خود تغییر دهید.

- حذف entity ای که مورد نیاز شما نیست : entity مورد نظر را انتخاب و دکمه delete را فشار دهید. میتوانید فیلد ها را نیز به همین روش حذف نمایید.

با حذف آیتمی از مدل، این تغییرات به مدل اعمال میشود و نه به بانک اطلاعاتی.

- تغییر نام entity

- تغییر نام فیلد : برای مشاهده نام واقعی فیلد در جدول میتوانید روی یک جدول کلیک راست کرده و انتخاب کنید تا پنجره زیر باز شود:

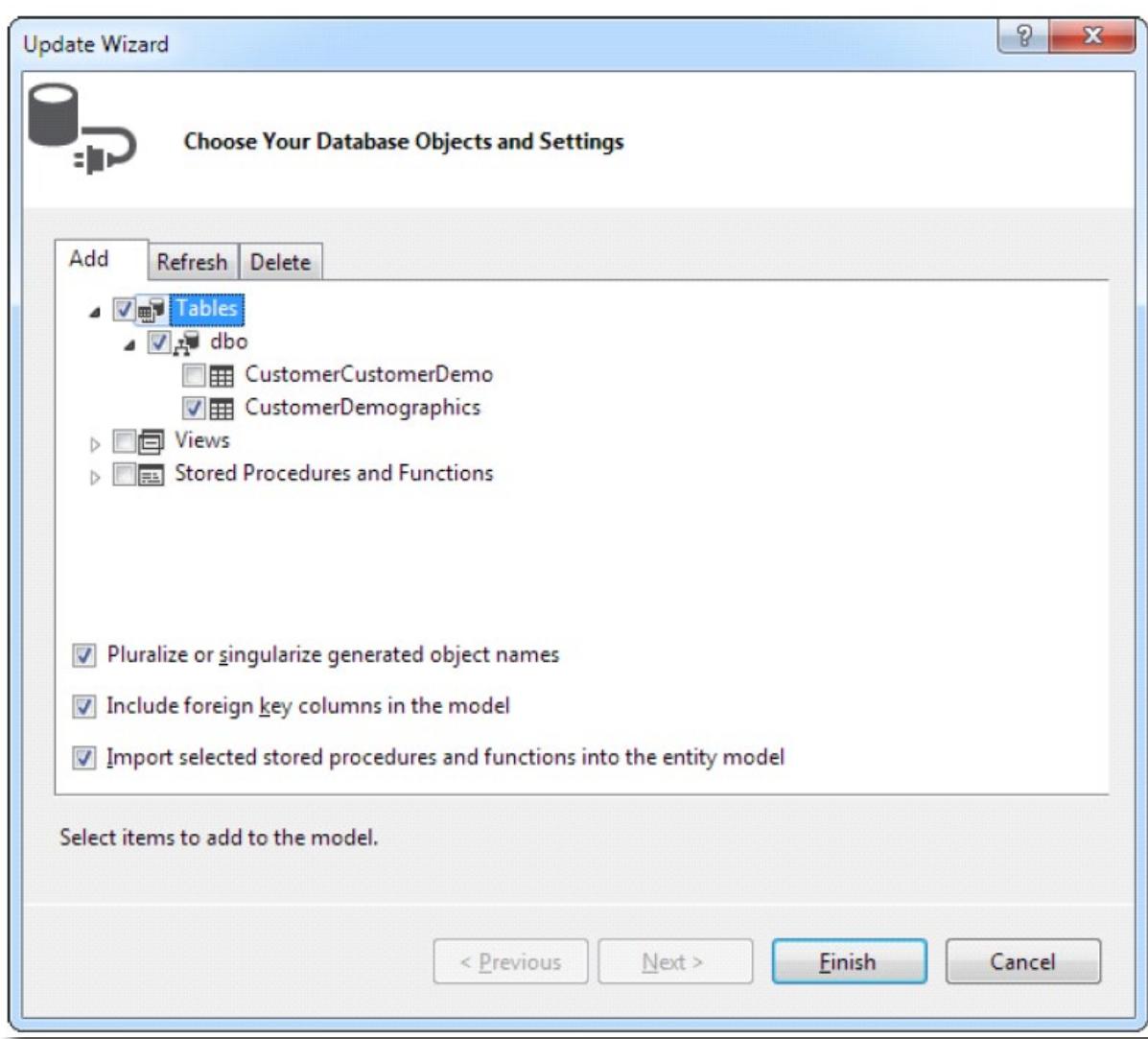
Mapping Details - Customer

Column	Oper...	Value / Property
<b>Tables</b>		
<ul style="list-style-type: none"> <li><b>Maps to Customers</b></li> <li>&lt;Add a Condition&gt;</li> <li><b>Column Mappings</b></li> </ul>		
CustomerID : nchar	⇄	CustomerID : String
CompanyName : nvarchar	⇄	CompanyName : String
ContactName : nvarchar	⇄	ContactName : String
ContactTitle : nvarchar	⇄	ContactTitle : String
Address : nvarchar	⇄	Address : String
City : nvarchar	⇄	City : String
Region : nvarchar	⇄	Region : String
PostalCode : nvarchar	⇄	PostalCode : String
Country : nvarchar	⇄	Country : String
Phone : nvarchar	⇄	Phone : String
Fax : nvarchar	⇄	Fax : String
<Add a Table or View>		

تغییر ویژگی های فیلد : می توانید طول nullable بودن و ... را با کلیک راست روی یک فیلد و انتخاب Properties تغییر دهید.

## ویرایش یک Data Model

برای اینکه تغییرات اعمال شده روی بانک اطلاعاتی را در مدل خود داشته باشید می توانید روی یک نقطه از مدل کلیک راست کنید و گزینه "Update Model from Database" را انتخاب کنید.



در ویزارد Update سه سربرگ زیر وجود دارد :

- **Add** : اجازه اضافه کردن اشیای جدید بانک اطلاعاتی را به مدل شما می‌دهد. می‌توانید انتخاب کنید کدام شی باید به مدل اضافه شود.
- **Refresh** : کلیه اشیا بانک اطلاعاتی که اکنون در مدل شما می‌باشد را نمایش می‌دهد. VS، این اشیا را برای تغییرات چک کرده و مجدداً آنها را ایجاد می‌کند.
- **Delete** : این لیست، اشیایی که در مدل شما وجود دارد ولی در بانک اطلاعاتی موجود نیست را نمایش می‌دهد.

## Entities

از Entity Framework ها، همان کلاس های data object ای هستند که رکوردی از بانک شما را نمایش می‌دهند. entity یک کلاس جدایگانه برای هر جدول استفاده می‌کند. هر کلاس entity دارای ساختار پایه ای مشابهی می‌باشد که شامل navigation property ها و data property ها می‌باشد.

همچنین هر کلاس entity، همچنین شامل یک متد static CreateEmployee() می‌باشد که اجازه ایجاد یک شی entity را به شما می‌دهد. برای نمونه می‌توانید متد Employee.CreateEmployee() را فراخوانی کنید و پارامترهای مناسب را برای ایجاد یک شی entity

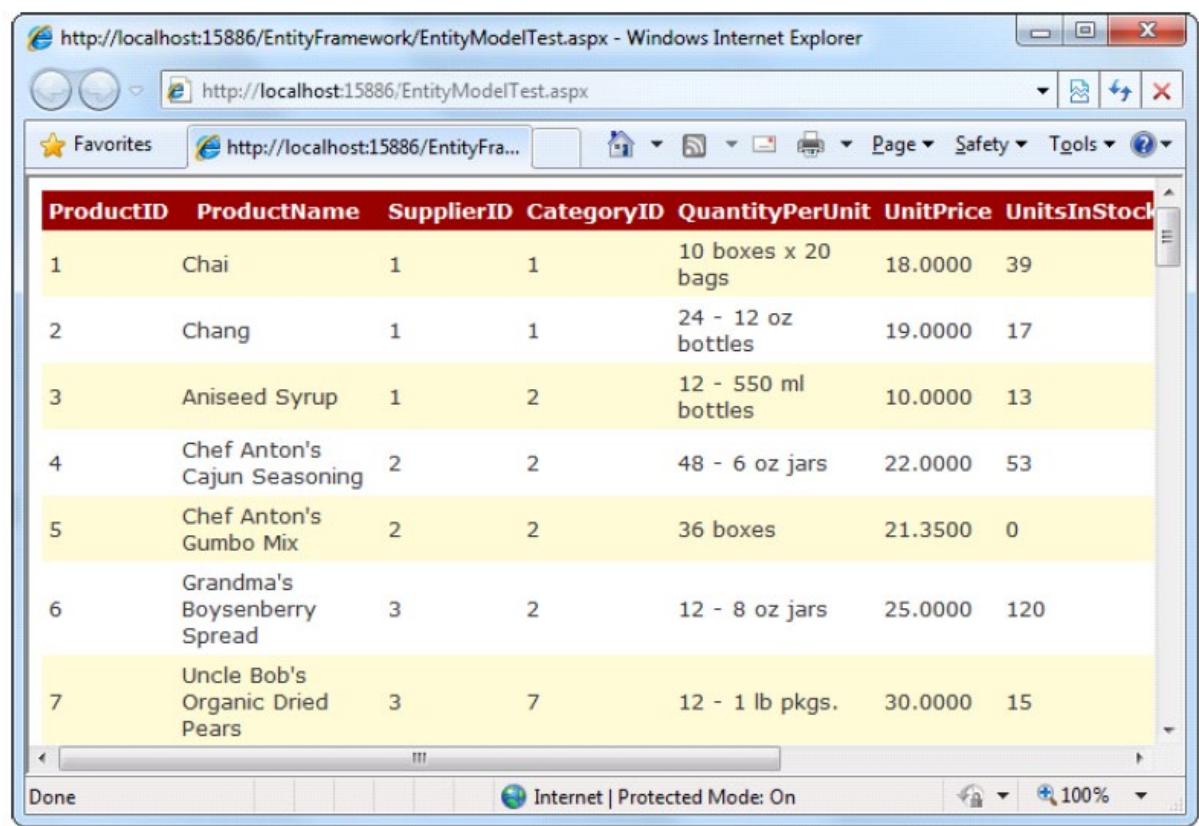
به آن ارسال کنید.

## Contexts

Entity مراقب ارائه داده می‌باشد و Context، مراقب دسترسی داده و کدی که رکورد های مورد نظر شما را از بانک fetch و تغییرات data model جدآگانه می‌باشد. بنابراین برای ما با نام NorthwindEntities مورد نیاز است که از کلاس context با نام Northwind یک ObjectContext باشد. مراقب ارائه داده می‌باشد. هر بانک اطلاعاتی نیازمند یک کلاس Context می‌باشد. بنابراین برای کلاس NorthwindEntities، شما Context می‌باشد. برای نمونه در کلاس NorthwindEntities شامل یک مجموعه برای هر جدول در بانک اطلاعاتی می‌باشد. برای Employee Entity با نام Employee با نام collection property یک گرداند.

## Data Model زدن روی Query

فرض کنید می‌خواهید لیستی از محصولات را نمایش دهید.



The screenshot shows a Microsoft Internet Explorer window displaying a table of product data from the Northwind database. The table has columns: ProductID, ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice, and UnitsInStock. The data is as follows:

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock
1	Chai	1	1	10 boxes x 20 bags	18.0000	39
2	Chang	1	1	24 - 12 oz bottles	19.0000	17
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.0000	13
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.0000	53
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.3500	0
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.0000	120
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.0000	15

```
using NorthwindModel;
```

مجموعه ای از محصولات را از ویژگی Propuct را فراخوانی کنید:

```
NorthwindEntities entities = new NorthwindEntities();
GridView1.DataSource = entities.Products;
GridView1.DataBind();
```

LINQ دارای یک ویژگی با نام context میباشد که زمانی که شما یک deferred execution ایجاد کرده یا به proper ty های آن دسترسی پیدا میکنید رخ نخواهد داد. بلکه هنگامی که روی مجموعه ای از entity ها حلقه ای را تکرار میکنید (iterate)، یا به یک شی تنهای entity دسترسی دارید، رخ نخواهد داد. در مثال قبل، زمانی روی Products Collection کار اطلاعاتی کار انجام میشود که GridView.DataBind() فراخوانی شود. زیرا این دستور به GridView میگوید که روی Products Collection حلقه تکرار را انجام دهد و Entity FrameWork برای بازیابی اطلاعات شما به کار اندازد. بنابراین برای error handling میتوانید از دستورات زیر استفاده کنید :

```
NorthwindEntities entities = new NorthwindEntities();
GridView1.DataSource = entities.Products;
try
{
    GridView1.DataBind();
}
catch (Exception err)
{
    // (Do something here, like displaying an error message.)
    ...
}
```

در مثال زیر از دستورToList() استفاده شده است که میتوانید کارایی را کاهش دهد:

```
NorthwindEntities entities = new NorthwindEntities();
GridView1.DataSource = entities.Products.ToList();
GridView1.DataBind();
```

زیرا با استفاده از این دستور، deferred execution نا دیده گرفته میشود.

اگر نمی خواهید نگران محل دقیق جایی که خط رخ می دهد باشید، می توانید هر چیزی که احتمال رخ دادن خط رخ دارد را درون یک exception block قرار دهید. می توانید کد دسترسی داده را درون یک متدهیگر قرا دهید سپس آن متدهی را درون exception block فراخوانی کنید.

```
try
```

```
{
```

```
// Call the method that does all the data processing and display.
```

```
.DataBindProductGrid();
```

```
}
```

```
catch (SqlException err)
```

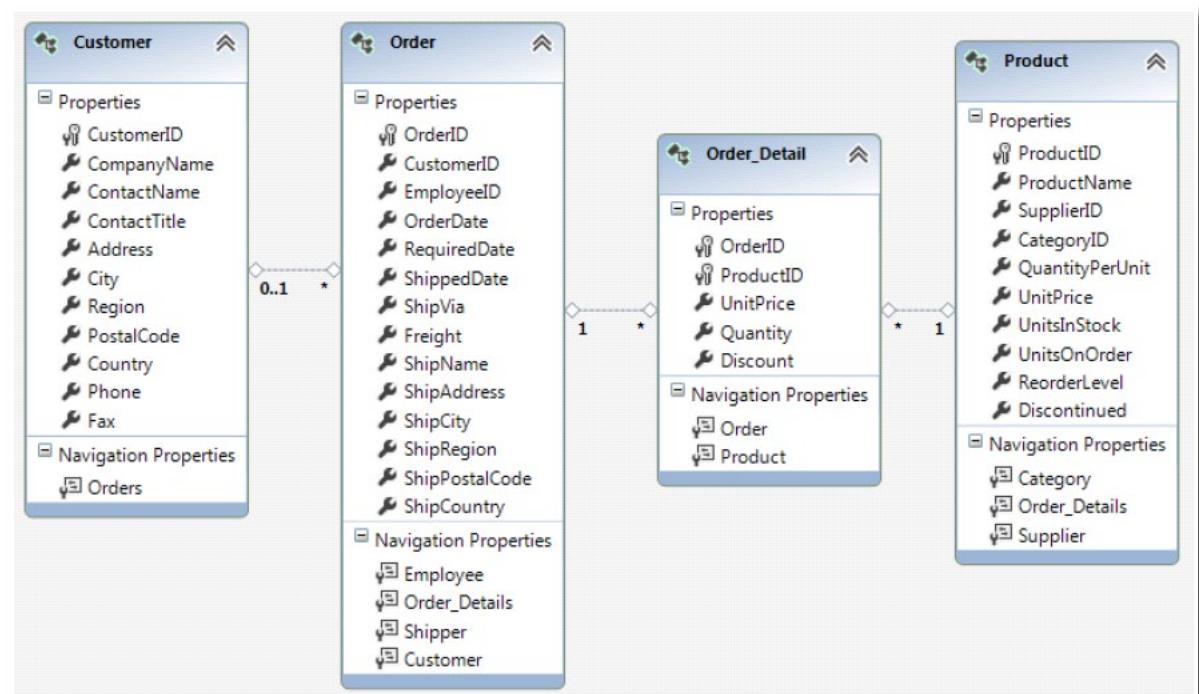
```
{
```

```
...
```

```
}
```

## Navigating Relationships

یک راه مناسب برای کار با روابط (Relationships) دارد. صرف نظر از استفاده از متدهای query های Entity FrameWork جداگانه، می توانید با استفاده از navigational properties به سادگی از یک رکورد به داده مرتبط با آن حرکت کنید و برسید.



یک خط ارتباطی که از ۱ به \* (برای نمونه بین Order و Order\_Detail) می رسد، یک ارتباط یک به چند را مشخص می کند. هر رکورد پدر در جدول اول (Order)، دارای یک یا چند مورد منطبق در جدول فرزند (Order\_Detail) می باشد.

یک خط ارتباطی که از **1..0** به **\*** می‌رسد، یک ارتباط یک به چند را که امکان وجود مقادیر **NULL** دارد را مشخص می‌کند. یعنی احتمال اینکه هیچ مورد منطبقی با این رکورد پیدا نشود وجود دارد.

یک خط ارتباطی که از **\*** به **\*** می‌رسد، نشان دهنده یک ارتباط چند به چند می‌باشد که توسط یک جدول واسط ایجاد می‌شود. برای نمونه در بانک NorthWind، هر رکورد Customer می‌تواند با یک یا چند رکورد از جدول CustomerDemographic مرتبط باشد و هر رکورد از جدول CustomerDemographic نیز می‌تواند با یک یا چند رکورد از جدول Customer مرتبط باشد. برای پیاده سازی این طرح، بانک اطلاعاتی از یک جدول واسط با نام CustomerCustomerDemo استفاده می‌کند که دارای رو فیلد با نام‌های CustomerTypeID و CustomerID می‌باشد.

EF، این مدل جدول را دقیقاً به کلاس‌های entity تبدیل نمی‌کند. فقط entity با نام‌های CustomerDemographic و Customer را ایجاد می‌کند و آنها را توسط Collection‌ها به یکدیگر متصل می‌کند.

جدول CustomerCustomerDemo، را در پشت صحنه مدیریت و استفاده می‌کند. برای نمونه اگر یک رکورد CustomerCustomerDemo، EF را ایجاد می‌کند و آنها را توسط CustomerDemographic مرتبط با یک Customer موجود اضافه کنید، EF، بصورت خودکار رکورد CustomerCustomerDemo، مرتبط با آنها را ایجاد می‌کند.

ارتباط بین Entity‌ها از طریق Navigational Property Order.Customer، لیست Order‌هایی که دارای رکورد Order (لیست مشتری‌هایی که چیزی سفارش داده اند) را بر می‌گرداند. ویژگی Oder\_Details Collection از رکوردهای فرزند مرتبط (با نام جمع مشخص شده) بر می‌گرداند. یک Order.Order\_Details Collection از اشیای order که با تعیین شده مرتبط هستند را بر می‌گرداند.

```
NorthwindEntities entities = new NorthwindEntities();

// Build a string full of HTML markup.

StringBuilder sb = new StringBuilder();

foreach (Customer customer in entities.Customers)

{

    // Write out the customer information in bold.

    sb.Append("< b > ");

    sb.Append(customer.CompanyName);

    sb.Append("</b > <br />");

    // List this customer's orders.

    foreach (Order order in customer.Orders)

    {

        sb.Append(order.OrderID.ToString());

        sb.Append(" - made on date: ");
    }
}
```

```

sb.Append(order.OrderDate.Value.ToShortDateString()));

sb.Append(" < br />");

}

// Add a horizontal line.

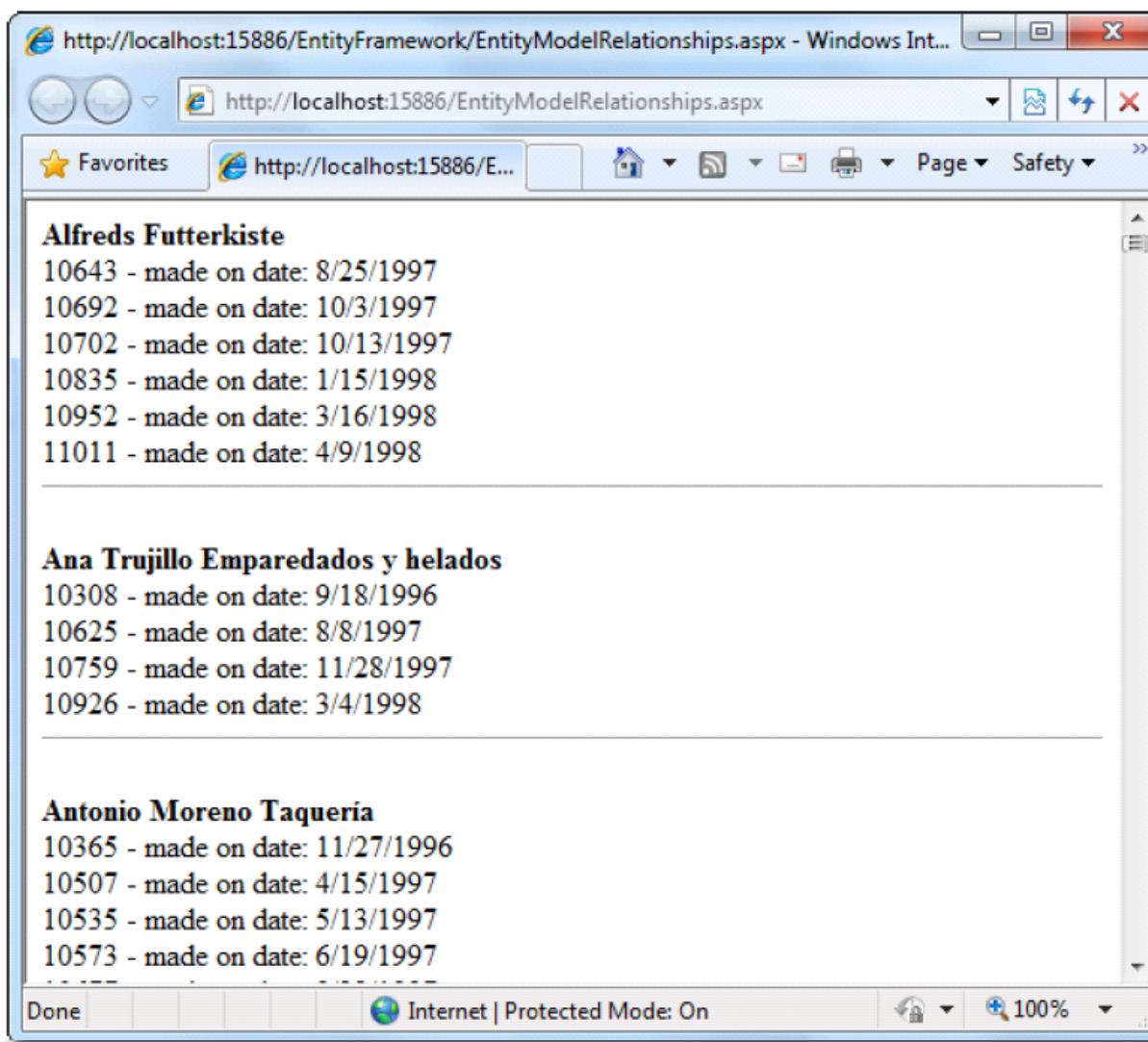
sb.Append(" < hr /> < br />");

}

// Show the HTML in a Label control.

lblData.Text = sb.ToString();

```



## عملیات های پیشرفته با Entity Framework

اگر از LINQ برای کار با EF استفاده کنید در حقیقت از LINQ to Entity استفاده کرده اید. با استفاده از query می توانید LTE هایی بنویسید که روی بانک اطلاعاتی کار می کنند. برای نمونه کد زیر، سفارشاتی را می گیرد که توسط یک مشتری خاص انجام شده اند:

```
NorthwindEntities entities = new NorthwindEntities();
gridProducts.DataSource = from order in entities.Orders
                         where order.CustomerID == "ALFKI"
                         select order;
GridView1.DataBind();
```

در مثال زیر کاربر لیستی از گروه های محصول را می بینید و می تواند یکی را انتخاب کند. سپس محصولات مرتبط با گروه انتخابی در GridView نمایش داده می شود.

The screenshot shows a Windows Internet Explorer window with the URL <http://localhost:15886/EntityFramework/LinqToEntities.aspx>. On the left, there is a dropdown menu labeled 'Beverages' with the sub-option '(Select a Category)' highlighted. To the right of the dropdown is a table displaying product information:

	Quantity	Stock
10 boxes x 20 bags	39	
24 - 12 oz bottles	17	
12 - 355 ml cans	20	
24 - 12 oz bottles	111	
24 - 12 oz bottles	20	
12 - 75 cl bottles	17	
750 cc per bottle	69	
16 - 500 g tins	17	
24 - 12 oz bottles	52	
24 - 355 ml bottles	15	
24 - 0.5 l bottles	125	
500 ml	57	

The browser status bar at the bottom indicates 'Internet | Protected Mode: On' and '100%'. The title bar of the browser window also displays the page URL.

```

<asp:DropDownList ID="lstCategories" runat="server" AutoPostBack="True" OnSelectedIndexChanged="lstCategories_SelectedIndexChanged"
    AppendDataBoundItems="True">

    <asp:ListItem Text="(Select a Category)" Value="-1"> </asp:ListItem>
</asp:DropDownList>

<asp:GridView ID="gridProducts" runat="server" CellPadding="4"
    GridLines="None" Font-Size="X-Small"
    ForeColor="#333333" AutoGenerateColumns="True">
    ...
</asp:GridView>

```

## Code-Behind

```

private NorthwindEntities entities = new NorthwindEntities();

protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)

    {
        lstCategories.DataTextField = "CategoryName";
        lstCategories.DataValueField = "CategoryID";
        lstCategories.DataSource = entities.Categories;
        lstCategories.DataBind();
    }
}

```

ویژگی Autopostback dropdownlist برابر با true می باشد.

```

protected void lstCategories_SelectedIndexChanged(object sender, EventArgs e)
{
    int selectedID = Int32.Parse(lstCategories.SelectedValue);
    if (selectedID == -1)

```

{

```

    // The "(Select a Category)" item was picked.

    // Don't show anything.

    gridProducts.DataSource = null;

}

else

{

    // Query the products in the selected category.

    gridProducts.DataSource = from product in entities.Products

        where product.CategoryID == selectedID

        select new

    {

        Name = product.ProductName,

        Quantity = product.QuantityPerUnit,

        Stock = product.UnitsInStock

    };

}

gridProducts.DataBind();

}

```

این مثال از دو ویژگی LINQ استفاده کرده است. ابتدا، عبارت Where یک فیلترینگ برای یافتن محصولاتی که در یک گروه خاص موجود است را بکار می‌گیرد. سپس عبارت new از Projection برای ایجاد یک شی جدید با زیر مجموعه‌ای از داده‌های محصول، استفاده می‌کند. بنابراین GridView تنها سه فیلد را نمایش می‌دهد.

NorthwindEntities.Products می‌باشد که روی LINQ to Object query های Query اجرا شد. البته اندکی با هم تفاوت دارند. در LTO، تمامی اطلاعات در حافظه نگهداری می‌شوند و LINQ به سادگی روی آنها حرکت و جستجو می‌کند. اما در LTE، اطلاعات با استفاده از deferred execution از بانک اطلاعاتی خوانده می‌شوند و تا زمانی که داده را bind نکنید و یا روی آن حلقه تکرار ایجاد نکنید، query اجرا نمی‌شود. در نتیجه، LINQ قادر به تبدیل عبارت شما به query بانک اطلاعاتی بهینه شده می‌باشد.

تفاوت آنها در کارایی می‌باشد. اگر به سادگی از LTO استفاده می‌کنید، باید query را روی همه محصولات اجرا کنید و آنها را در حافظه نگه دارید و سپس رکوردها و فیلدهایی که نمی‌خواهید را اجرا کنید. این کار یک سربار اضافی روی سرور بانک اطلاعاتی تحمیل می‌کند که سرعت صفحه وب شما را نیز کاهش می‌دهد و از حافظه بیشتری روی وب سرور استفاده خواهد کرد. اما اگر از LTE استفاده کنید، شما تنها رکوردها و فیلدهایی که مورد نیازتان است را بازیابی می‌کنید. بنابراین به جای استفاده از دستور SQL زیر:

```
SELECT * FROM Products
```

LTE از دستور زیر استفاده می‌کند:

```
SELECT ProductName, QuantityPerUnit, UnitsInStock FROM Products WHERE CategoryID = 1
```

البته می‌توانید EF را وادار کنید تمامی اطلاعات را در حافظه قرار دهد و سپس از LTO استفاده کنید. برای این کار از متدهایی نظیر `ToList()` استفاده کنید:

```
gridProducts.DataSource = from product in entities.Products.ToList();
```

اگرچه، این کد شبیه مثال قبلی می‌باشد ولی کار کرد آن متفاوت است. اکنون کل جدول products در حافظه قرار می‌گیرد و سپس با استفاده از LTO روی آن لینک زده می‌شود.

## کنترل زمان بارگذاری داده

با توجه به نکات ذکر شده در مباحث قبلی، EF تصمیم می‌گیرد که چه زمان database query را اجرا کند. این کار به EF کردن query را می‌دهد که در بخش قبل دیدید. البته در بعضی موارد، به دلیل پیچیدگی های موجود، کارایی حتی کاهش هم می‌یابد. نمونه آشکار این موضوع، زمانی است که به داده های مرتبط دسترسی داریم. یکبار دیگر، EF، منظر می‌ماند تا زمان مناسب برای اجرای این query پیدا شود. در مثال نمایش محصولات بر اساس گروه انتخابی، زمانی که شروع به حرکت روی EF می‌کنید، یک query را برای انتخاب کلیه رکوردهای customer اجرا می‌کند. سپس زمانی که حلقه داخلی Customer Collection می‌گردد، دومین query را برای انتخاب سفارشات آن مشتری اجرا می‌کند. زمانی که شما به مجموعه Customer.Orders دنبال از مشتری دوم دسترسی پیدا کنید، EF، سومین query را برای انتخاب سفارشات مشتری اجرا می‌کند و این فرایند با query جدایگانه سفارشات برای هر مشتری ادامه پیدا می‌کند.

این روش را lazy loading می‌نامند و زمانی مناسب است که نیاز به دریافت تعداد کمی از رکوردهای مرتبط به هم (برای نمونه، سفارشات یک یا دو مشتری) دارد. اگر بخواهید سفارشات همه مشتری‌ها را داشته باشید (مانند این مثال)، این روش خیلی مناسب نمی‌باشد. در حقیقت کاری که سرور بانک اطلاعاتی باید انجام دهید را چند برابر می‌کند.

در این موقعیت، بهترین راه حل، پیش بارگذاری (PreLoad) داده با استفاده از متدهای `Include()` و `Configure()` تعیین جدول با داده مرتبط می‌باشد. برای نمونه، می‌توانید مثال بالا را از:

```
NorthwindEntities entities = new NorthwindEntities();
foreach (Customer customer in entities.Customers)
{
    customer.Orders = customer.Orders.Include("OrderDetails");
}
```

: ۴

```

foreach (Order order in customer.Orders)

{
    ...
}

```

```

NorthwindEntities entities = new NorthwindEntities();

foreach (Customer customer in entities.Customers.Include("Orders"))

{
    foreach (Order order in customer.Orders)
    {
        ...
    }
}

```

تغییر دهید.

جدول Customers شامل یک property Orders به نام باشد. زمانی که query اول foreach می‌باشد، EF فوراً یک foreach loop را بازیابی می‌کند، اجرا خواهد کرد. هنگامی که حلقه foreach دوم اجرا می‌شود، هیچ عملیاتی روی بانک اجرا نمی‌شود. در عوض، EF از entity order شده استفاده خواهد کرد.

اگر می‌خواهید چند سطح بین جداول حرکت کنید، می‌توانید بعد از include() از نقطه استفاده کنید. برای نمونه، برای بازیابی رکوردهای Order\_Detail برای هر سفارش، باید متدهای Order\_Detail را مانند زیر پیاده کنید.

```

NorthwindEntities entities = new NorthwindEntities();

foreach (Customer customer in entities.Customers.Include(
    "Orders.Order_Details.Product"))

```

{

...

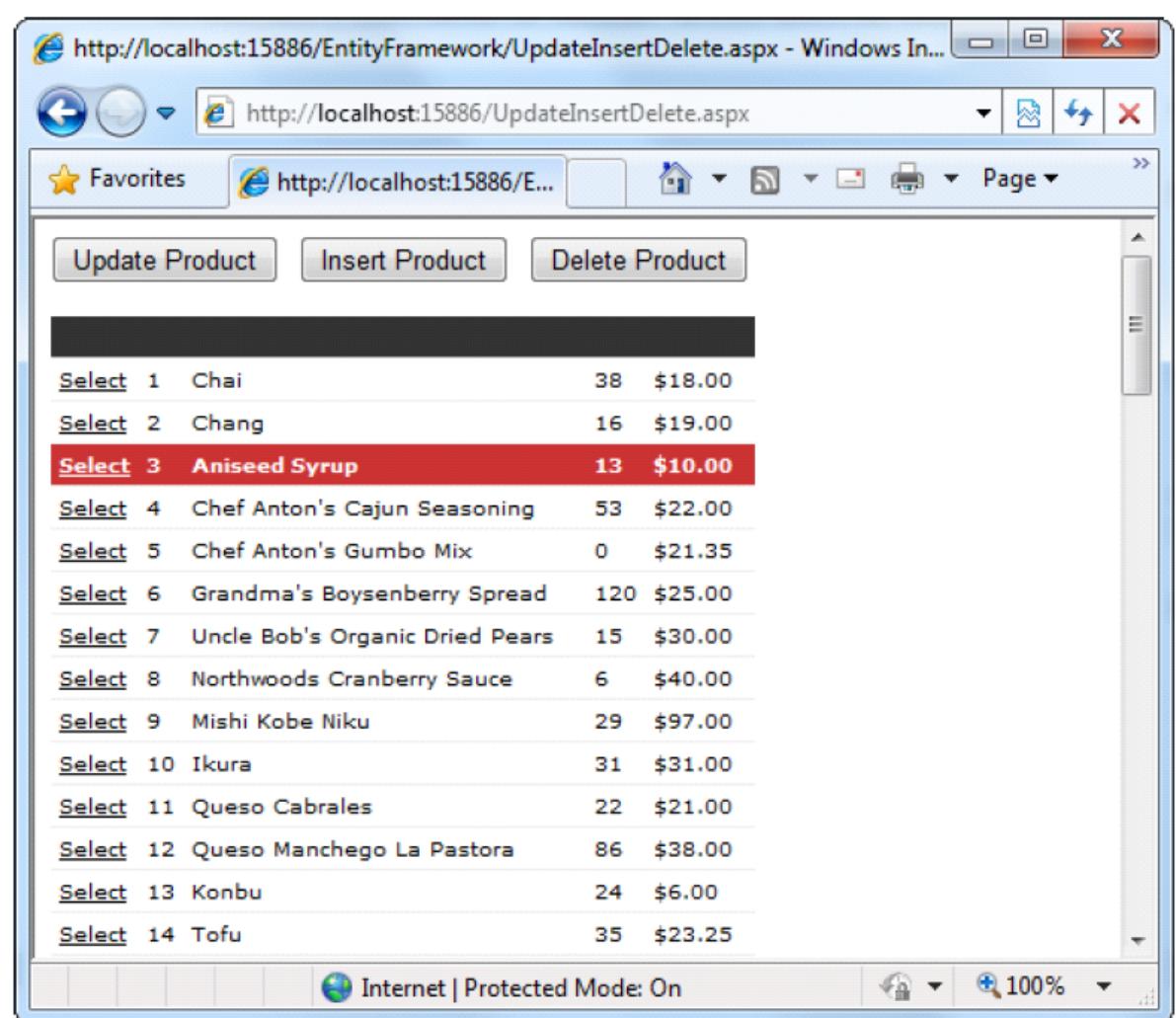
```

foreach (Order order in customer.Orders)
{
    ...
}
}

```

## Update,Insert,Delete اجرای

کدهای c# شما را به دستورات SQL تبدیل می‌کند. برای این کار باید فقط متدهای SaveChanges() و NorthwindEntities.SaveChanges() را از شی Context خود را فراخوانی کنید (مثلاً).



برای ویرایش یک رکورد ابتدا نیاز است تا شی entity مورد نظر را بازیابی کنید.

کد زیر یک محصول خاص را (با استفاده از LINQ) انتخاب و سپس Stock Number آن را ویرایش می‌کند.

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

{
    // Only allow updating if a record is currently selected.

    if (gridProducts.SelectedIndex != -1)

    {
        // Use a LINQ expression to find the selected product.

        int selectedProductID = (int)gridProducts.SelectedDataKey.Value;

        var matches = from p in entities.Products

                      where p.ProductID == selectedProductID

                      select p;

        // Execute the query and return the entity object.

        Product product = matches.Single();

        // Change the entity object.

        product.UnitsInStock -= 1;

        // Commit the changes back to the database.

        entities.SaveChanges();
    }
}
```

نکته: متدهای SaveChanges() و همچنین یک عدد صحیح که تعداد رکوردهای ویرایش شده را مشخص می‌کند، بر می‌گرداند.



برای ایجاد یک رکورد، باید از متدهای خاص این کار در کلاس entity مورد نظر خود استفاده نمایید. بنابراین اگر می‌خواهید یک product ایجاد کنید، باید Product.CreateProduct() را فراخوانی کنید. سپس شی جدید را با استفاده از متدهای AddObject() و NorhtwindEntities.Products اضافه کنید.

```

protected void cmdInsert_Click(object sender, EventArgs e)
{
    // The CreateProduct() method requires the three non-nullable
    // Product fields
    // as arguments: ProductID, ProductName, and Discontinued. However, the
    // ProductName isn't actually used--when the update is finished,
    // it's replaced
    // by the automatically generated ID that the database creates.

    Product newProduct = Product.CreateProduct(0, "Thick-As-Cement Milkshake",
        false);

    // You can now set additional properties that aren't required.

    newProduct.CategoryID = 1;
    newProduct.UnitsInStock = 10;
    newProduct.UnitPrice = 15.99M;

    // Finally, commit the changes and insert the record in the database.

    entities.Products.AddObject(newProduct);
    entities.SaveChanges();

}

```

در نهایت، برای حذف باید متد DeleteObject() را فراخوانی کنید.

```

protected void cmdDelete_Click(object sender, EventArgs e)
{
    // Check if a row is selected.

    if (gridProducts.SelectedIndex != -1)

    {
        // Use a LINQ expression to find the selected product.

        int selectedProductID = (int)gridProducts.SelectedDataKey.Value;

        var matches = from p in entities.Products
                     where p.ProductID == selectedProductID
                     select p;

        // Execute the query and return the entity object.

```

```

Product product = matches.Single();

// Delete the record from the database.

entities.Products.DeleteObject(product);

entities.SaveChanges();

// Clear the selection (which may now be pointing to a different row.)

gridProducts.SelectedIndex = -1;

}

}

```

برای انجام این عملیات ها از یک سیستم EF change-tracking استفاده می‌کند. زمانی که متده است () SaeChanges() می‌کنید، شی Context تلاش می‌کند تا تمامی تغییرات در داده ای که رديابی شده است (مانند update, delete, insert) را اعمال کند. اگر دارای رکوردهای مرتبط باشد، EF، تغییرات را روی یک order اعمال می‌کند. (برای نمونه، قبل از اضافه کردن رکورد پدر، رکورد فرزند را اضافه نمی‌کند).

## مدیریت هم زمانی

از راهکار ویرایشی "last-in-win" (آخرین ورودی برنده است)، استفاده می‌کند. این یعنی که ویرایش های جدید همیشه موفق هستند اما آنها ممکن است تغییرات سایر کاربران را از بین برنده زیرا EF، از مقادیر قدیمی استفاده نمی‌کند. اگر یک فیلد تنها را تغییر دهید، و SaveChanges() را تغییر دهید، EF، از دستور update که فقط آن فیلد را تغییر می‌دهد استفاده می‌کند. اگر دو کاربر همزمان یک رکورد و یک فیلد را ویرایش کنند EF، دچار مشکل می‌شود. برای حل این مشکل باید EF را برای استفاده از stricter concurrency checking

## (کنترل هم زمانی سخت گیرانه تر) را پیکربندی کنید.

با استفاده از Data Model Designer، فیلدهای مورد نظر که یک ویرایش موفق را انتخاب کنید. برای نمونه، ممکن است تصمیم بگیرید، ویرایش یک رکورد Customer باید تنها اگر فیلد های CompanyName و ContactName آنها توسط فرد دیگری از زمان بازیابی این رکورد تغییر نکرده بود، بتواند انجام شود.

برای این کار، روی فیلد مورد نظر در مدل کلیک و سپس از پنجره Properties گزینه ConcurrencyMode از None به Fixed تغییر دهید.

اگر بخواهید تغییراتی را اعمال کنید که کاربر دیگر تغییر داده است هم پوشانی داشته باشد، ویرایش با شکست روبرو خواهد شد و یک OptimisticConcurrencyException رخ خواهد داد.

البته لازم به ذکر است که کنترلی مشابه SqlDataSource وجود دارد که در این کتاب به توضیح آن نخواهیم پرداخت

# بخش پنجم : پیشرفته ASP.NET

فصل شانزدهم: برنامه نویسی مبتنی بر کامپوننت

Caching فصل هفدهم

Entity Framework , LINQ فصل هجدهم

ASP.NET AJAX فصل نوزدهم

ASP.NET مرکز بزرگ آموزشی Deploy فصل بیست و یکم



تا کنون ساخت صفحات وب با استفاده از مدل postback را یاد گرفته اید. با این مدل زمانی صرف انتقال کل صفحه به سرور، اجرای کدها، رندر کردن مجدد صفحه و برگرداندن HTML تولید شده می شد. بنابراین باید کل صفحه refresh شود و بخش هایی از صفحه که نیازی به تغییر نداشته اند نیز refresh شده و مراحل بالا برای کل صفحه انجام می شود. حال فرض کنید کاربری در حال پرکردن فرمی باشد. اگر چندین بخش از فرم نیازمند چک کردن قوانین خاصی روی موارد ورودی از طرف کاربر باشد می توان اجازه داد کاربر کل فرم را پر کند و سپس وقتی آن را Submit کرد کل آن موارد را بررسی و نتیجه را به او اعلام کرد. در ضمن نمی شود بایت هر بخش از فرم یکبار فرم را refresh کنیم تا کاربر از نتیجه اطلاعات ورودی خود با خبر شود. بنابراین بهترین روش این است که فرم را طوری طراحی کنیم که به عنوان مثال اگر کاربر کلمه عبور حساب کاربری جدیدی که می خواهد ایجاد کند را وارد کرده و فوکوس به داخل textbox کلمه عبور رفته است، فرم بصورت اتوماتیک کلمه عبور کاربر را از نظر تکراری بودن چک کند و کاربر نیز بدون مشکل در حال پرکردن کلمه عبور خود باشد. این کار توسط یک استاندارد برنامه نویسی تحت وب با نام AJAX قابل انجام است.

برای بسیاری از فرم ها مهم است که وقتی کاربر بخشی از فرم را پر کرده که نیازمند یک عملیات سروری می باشد و اکنون می خواهد ادامه کار با فرم را در بخش دیگری از آن انجام دهد کل صفحه refresh نشود و فقط اطلاعات همان بخش از یک صفحه بروز شود.

مثالاً فرض کنید نتایج یک مسابقه فوتbal به صورت آنلاین در یک صفحه نمایش داده می شود. این نتیجه درون یک Label نیز می تواند به نمایش درآید. اکنون نیازی نیست با تغییر نتیجه بازی کل صفحه refresh شود بلکه کافی است فقط با استفاده از Ajax اطلاعات مربوط به آن Label تغییر کند. این کار سرعت صفحه را نیز بسیار بالا می برد.

## آشنایی با AJAX

AJAX، یک تکنولوژی کاملاً جدید نیست. بلکه ترکیبی از تکنیک‌ها می‌باشد. حتماً کنترل‌های ASP.NET را دیده اید که از JS برای ارائه کنترل‌های قوی‌تر استفاده کرده‌اند (مانند validation control ها و Menu control). صفحات AJAX، استفاده فراتری از JS می‌کنند، آنها اغلب نیازمند تعامل بین کنترل‌ها می‌باشند و اطلاعات اضافی را از وب سرور با استفاده از یک شی خاص مروگر با نام XMLHttpRequest درخواست می‌کنند که در دسترس کد JS در سمت کلاینت می‌باشد.

اگر مروگری AJAX را پشتیبانی نکند (مانند بعضی از مروگرهای موبایل)، ASP.NET خودکار به حالت PostBack برای آن صفحه تغییر وضعیت می‌دهد.

## ASP.NET در AJAX

در این فصل شما از JS مستقیماً استفاده نمی‌کنید. در عوض از یک مدل سطح بالا به نام ASP.NET AJAX که مجموعه‌ای از کامپوننت‌های سمت سرور و کنترل‌هایی که می‌توانید در هنگام طراحی یک صفحه استفاده کنید، در اختیار شما قرار می‌دهد. این کامپوننت‌ها بصورت خودکار، تمامی JS مورد نیاز را تولید می‌کنند.

## Script Manager

برای استفاده از ASP.NET AJAX، شما باید یک کنترل وبی جدید با نام ScriptManager را درون صفحه قرار دهید که همان معز ASP.NET AJAX می‌باشد.

```
<asp:ScriptManager ID = "ScriptManager1" runat = "server" > </asp:ScriptManager>
```

در حالت design، این کنترل شبیه یک کادر خاکستری رنگ می‌باشد و در زمان اجرا چیزی نخواهد دید زیرا HTML خاصی برای آن تولید نمی‌شود. این کنترل، لینک‌هایی را به کتابخانه جاوا اسکریپت ASP.NET AJAX در صفحه قرار می‌دهد.

```
<script src = "/ScriptResource.axd?d = RUSU1mI ..." type = "text/javascript">
</script>
```

بلوک اسکریپت اضافه شده به صفحه توسط این کنترل دارای کدی نیست. بلکه از یک خصیصه src برای بیرون کشیدن کد JS از یک فایل جداگانه استفاده می‌کند. فایل ScriptResource.axd، یک فایل واقعی نمی‌باشد. در حقیقت یک منبع می‌باشد که به ASP.NET می‌گوید که فایل JS درون یکی از اسembly‌های کامپایل شده دات نت قرار دارد. Query string URL طولانی در انتهای مشخص می‌کند که کدام فایل به مروگر ارسال شود. این کدهای JS یکبار دانلود شده و سپس در مروگر cache می‌شود.

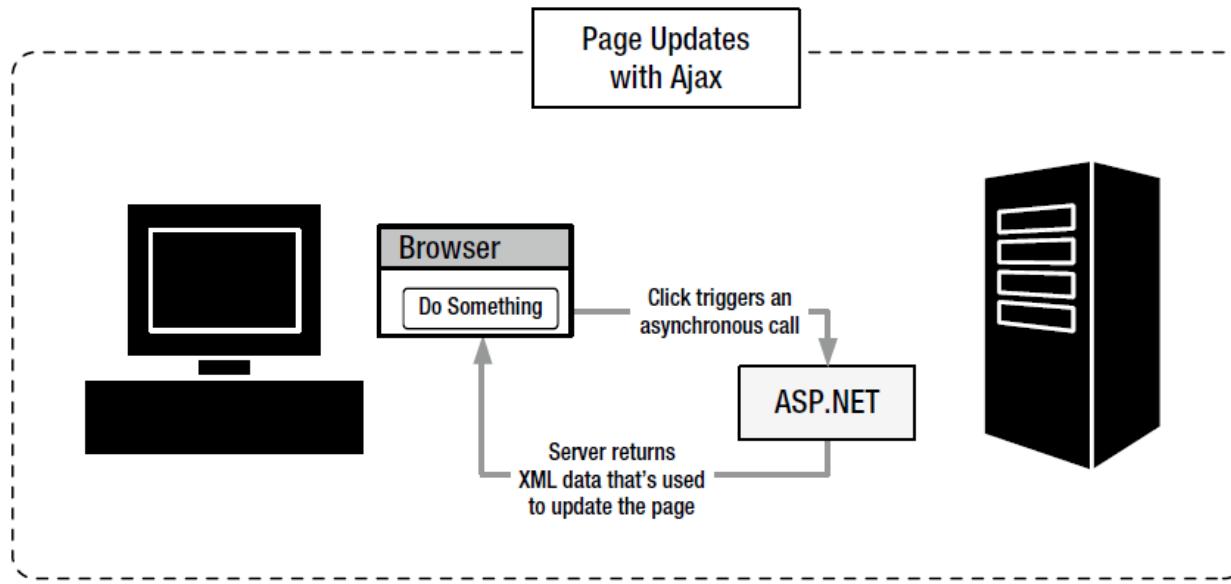
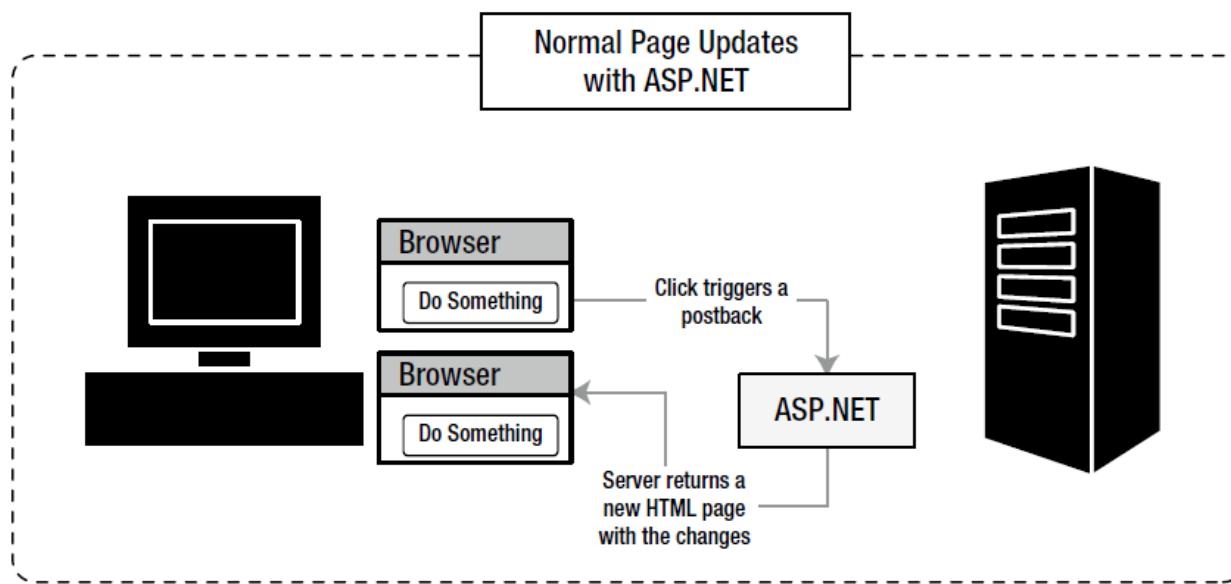
برای آنکه در هر صفحه از ScriptManager استفاده نکنید می‌توانید آن را در MasterPage قرار دهید. در صورتیکه هر صفحه بخواهد های خاص خود را روی ScriptManager پیکربندی کند می‌توانید یک property MasterPage در ScriptManager و یک (Content Page) که خواهان پیکربندی خاص خود می‌باشند، قرار دهید. ScriptManagerProxy

در ادامه روش های استفاده از ASP.NET AJAX را خواهیم آموخت.

## استفاده از Partial Refresh

تکنیک کلیدی در یک برنامه وب ASHX، استفاده از partial refresh می باشد که کل صفحه نیازی به refresh نخواهد داشت و انجام نمی شود.

CHAPTER



برای انجام این روش از کنترلی به نام Update Panel استفاده می‌کنیم. در این روش شما صفحه را به نواحی مختلف تقسیم می‌کنید و هر یک از آنها را درون یک Update Panel قرار می‌دهید. زمانی که رویدادی در کنترلی که درون UP قرار دارد رخ می‌دهد و آن رویداد بصورت نرمال کل صفحه را متوقف کرده و یک asynchronous callback post back می‌کند، رویداد را متوقف کرده و آن فرآخوانی می‌کند.

۱- کاربر روی یک دکمه درون UP کلیک می‌کند.

۲- UP، کلیک سمت کلاینت را متوقف می‌کند. ASP.NET AJAX، به جای یک callback postback کامل، یک سرور

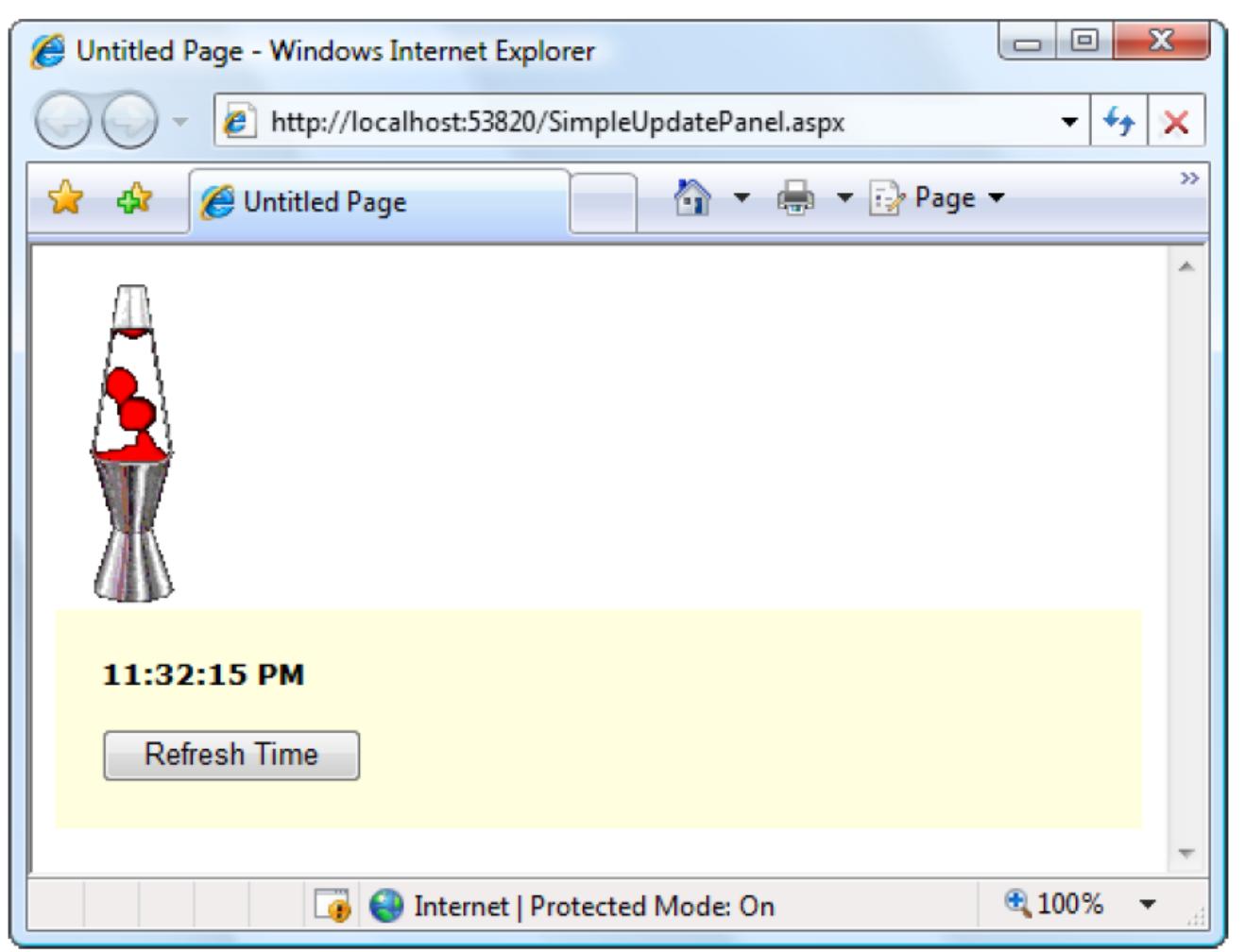
ارسال می‌کند.

۳- در سمت سرور، چرخه حیات نرمال صفحه با تمامی رویدادهای آن اجرا می‌شود. در نهایت صفحه رندر می‌شود و به مرورگر برگردانده می‌شود.

۴- HTML برای هر UP موجود در صفحه دریافت می‌کند. کد اسکریپت سمت کلاینت، درون هر UP را با محتوای جدید آن جایگزین می‌کند.

### یک مثال ساده

زمانی که روی دکمه کلیک کنید، صفحه، زمان حال حاضر را از سرور گرفته و Labael را بروز می‌کند. تصویر gif موجود در صفحه به شما کمک می‌کند که ببینید زمانی که روی دکمه کلیک می‌کنید، لامپ موجود در تصویر بدون توقف به حباب سازی خود ادامه می‌دهد.



```

<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>

<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
<ContentTemplate>
<div style="background-color: LightYellow; padding: 20px">
<asp:Label ID="lblTime" runat="server" Font-Bold="True"> </asp:Label>
<br />
<br />
<asp:Button ID="cmdRefreshTime" runat="server"
    OnClick="cmdRefreshTime_Click" Text="Refresh Time" />
</div>
</ContentTemplate>
</asp:UpdatePanel>

```

کد زیر را در code-behind قرار دهید :

```

protected void cmdRefreshTime_Click(object sender, EventArgs e)
{
    lblTime.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss");
}

```

### مراحل انجام کار:

۱- زمانی که HTML در حال رندر شدن است، UP، به محتوای آن نگاه کرده و متوجه یک دکمه می‌شود که می‌تواند یک JS ای را اضافه می‌کند که جلوی رویداد کلیک دکمه را در سمت کلاینت می‌گیرد و از javascript برای هندل کردن آن استفاده می‌کند.

۲- زمانی که روی دکمه کلیک می‌کنید، javascript routin را اجرا کرده اید.

۳- این routin را کامل اجرا نمی‌کند. در عوض، یک درخواست بصورت پس زمینه به وب سرور ارسال می‌کند. این درخواست asynchronous (غیر همزمان) می‌باشد که یعنی صفحه شما در طول انجام درخواست کاملاً پاسخگو باقی می‌ماند.

 نکته: به دلیل آنکه UP از درخواست async استفاده می‌کند، امکان چندین بار کلیک روی دکمه Refresh در مثال بالا قبل از برگشتن نتیجه وجود دارد. در این موارد، پاسخ مربوط به درخواست‌های ابتدایی نادیده گرفته می‌شود و پاسخ آخرین درخواست استفاده می‌شود.

۴- Backgroun request، به همان روشی که یک postback معمولی پردازش می‌شد، پردازش خواهد شد. کلیه داده‌ها از همه کنترل‌ها به وب سرور برگردانده می‌شوند. در وب سرور، چرخه حیات صفحه انجام می‌شود. ابتدا PageLoad و سپس رویداد‌هایی که باعث postback شده‌اند (در این مثال button click) رخ می‌دهد.

۵- زمانی که مرورگر HTML رندر شده را دریافت می‌کند، View State مرتبط را بروز کرده و کوکی‌های برگردانده شده می‌گیرد.

۶- بخش‌هایی که در UP قرار دارد را جایگزین می‌کند. باقی HTML صفحه، تغییر نمی‌کند.

 نکته: زمانی که از UP استفاده می‌کنید، مقدار bandwith استفاده شده یا زمان صرف شده برای دریافت پاسخ از سرور کاهش نمی‌یابد، زیرا هنوز کل صفحه ارسال می‌شود. تنها تفاوت این است که صفحه بدون refresh شدن، بروز می‌شود.

## Error Handling

برای روشن شدن مطلب یک خطأ را درون PageLoad صفحه throw کنید:

```
if (this.IsPostBack)
{
    throw new ApplicationException("This operation failed.");
}
```

زمانی که این خطأ رخ می‌دهد، توسط ScriptManager گرفته می‌شود و به کلاینت ارسال می‌شود. JS سمت کلاینت، یک خطای جاوا اسکریپتی throw می‌کند. اتفاقات بعدی به تنظیمات مرورگر شما دارد. معمولاً مرورگرها، خطأ را نمایش می‌دهند. در IE، خطای درون صفحه، در statusbar نمایش داده می‌شود. با کلیک روی آن صفحه نمایش پیام باز می‌شود.

البته گزینه دیگری نیز برای خطاهایی که در هنگام asynchronous postback رخ می‌دهد نیز وجود دارد. می‌توانید از یک صفحه نمایش خطای سفارشی استفاده کنید.

&lt;configuration&gt;

...

&lt;system.web&gt;

...

&lt;customErrors defaultRedirect = “ErrorPage.aspx” mode = “On” &gt; &lt;/customErrors&gt;

&lt;/system.web&gt;

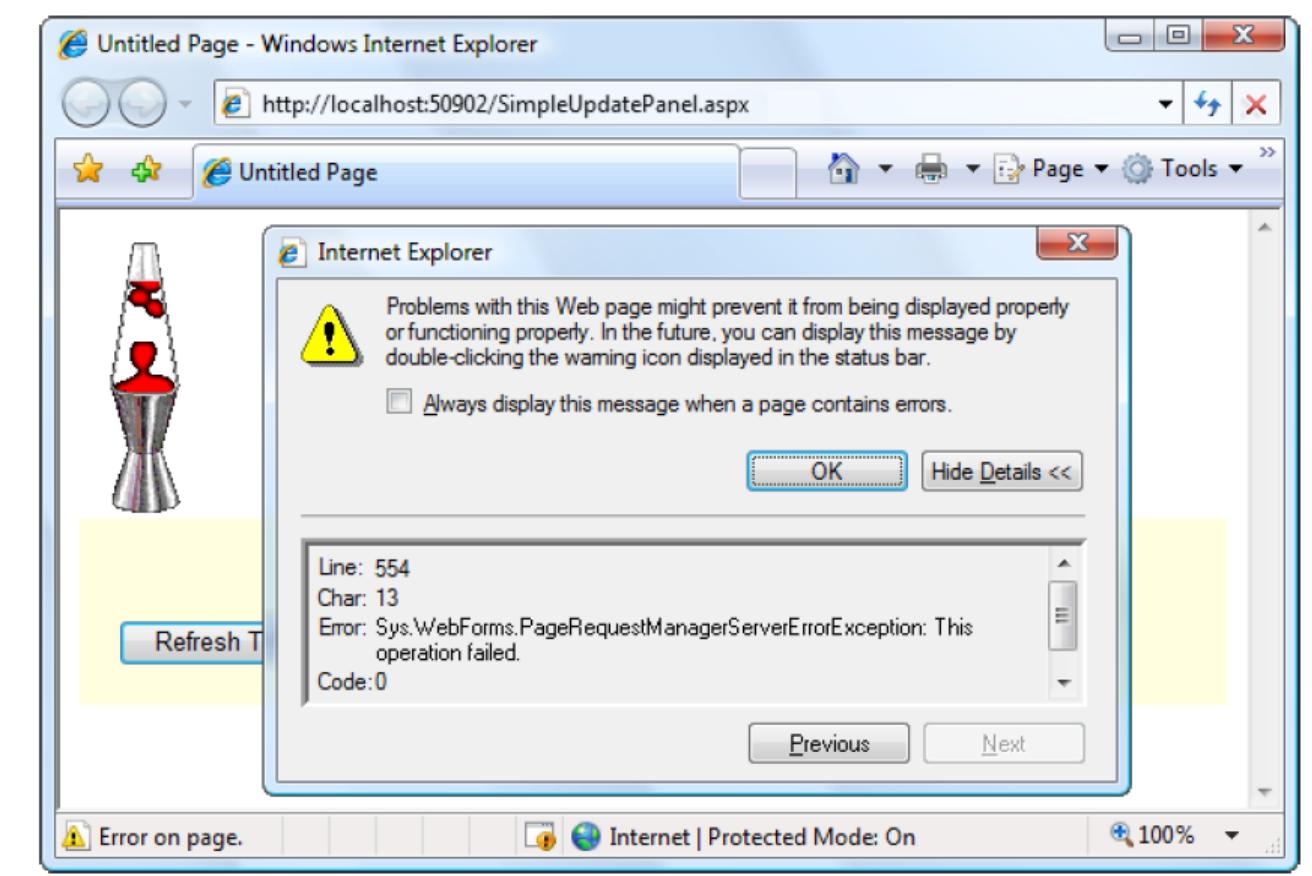
&lt;/configuration&gt;

اگر زمانی که PageRequestManager از وجود خطا آگاه می‌شود، مرورگر را به صفحه ErrorPage.aspx هدایت می‌کند. همچنین یک URL نیز اضافه می‌کند که خطا در آن رخ داده است را مشخص می‌کند.

<http://localhost/Ajax/ErrorPage.aspx?aspxerrorpath=/Ajax/UpdatePanels.aspx>

کد شما در صفحه خطا می‌تواند شبیه زیر می‌باشد :

```
string url = Request.QueryString[“aspxerrorpath”];
if (url != null) Response.Redirect(url);
```



## بروز رسانی مشروط

در صفحات پیچیده، ممکن است بیش از یک UP استفاده کنید. در این موارد، زمانی که یک UP یک بروز رسانی را موجب می‌شود، تمامی نواحی refresh، UP می‌شوند.

اگر بخواهید هر UP، جداگانه بروز رسانی شود باید ویژگی Conditional Always را از UpdatePanel.UpdateMethod به تغییر دهید. اکنون UP، زمانی refresh می‌شود که تنها رویدادی مربوط به یکی از کنترل‌های درون آن رخ داده باشد.

اگر بخواهید در زمان اجرا تصمیم بگیرید که آیا یک UP باید refresh شود یا نه می‌توانید از UpdatePanel.Update() استفاده کنید. البته متد () Update را روی پنلی استفاده کنید که آن برابر با Always نباشد. همچنین نباید آن را بعد از رندر شدن صفحه فراخوانی کنید.

## Trigger

اکنون می‌خواهیم بروز شدن کنترل‌ها را بر اساس trigger‌ها یاد بگیریم.

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>

<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
    <!-- These are the controls for creating the greeting card. -->
    <div>
        Choose a background color:<br />
        <asp:DropDownList ID="lstBackColor" runat="server" Width="194px"
AutoPostBack="True">
        </asp:DropDownList>
        <br />
        <br /> Choose a foreground (text) color:<br />
        <asp:DropDownList ID="lstForeColor" runat="server" Height="22px"
Width="194px" AutoPostBack="True">
        </asp:DropDownList>
        <br />
        <br /> Choose a font name:<br />
        <asp:DropDownList ID="lstFontName" runat="server" Height="22px"
Width="194px" AutoPostBack="True">
    </ContentTemplate>
</asp:UpdatePanel>
```

```

</asp:DropDownList>

<br />

< br /> Specify a font size:<br />

<asp:TextBox ID="txtFontSize" runat="server" AutoPostBack="True">

</asp:TextBox>

<br />

< br /> Choose a border style:<br />

<asp:RadioButtonList ID="lstBorder" runat="server" Height="59px"
Width="177px" Font-Size="X-Small"

AutoPostBack="True" RepeatColumns="2">

</asp:RadioButtonList>

<br />

<br />

<asp:CheckBox ID="chkPicture" runat="server" Text="Add the Default Picture"
AutoPostBack="True">

</asp:CheckBox>

<br />

< br /> Enter the greeting text below:<br />

<asp:TextBox ID="txtGreeting" runat="server" Height="85px" Width="240px"
TextMode="MultiLine"

AutoPostBack="True">

</asp:TextBox>

</div>

<!-- This is the panel that shows the greeting card. -->

<asp:Panel ID=" pnlCard" runat="server">

<asp:Label ID="lblGreeting" runat="server" Width="272px" Height="150px" ></
asp:Label>

<br />

<asp:Image ID="imgDefault" runat="server" Width="212px" Height="160px"
Visible="False">

</asp:Image>

</asp:Panel>

</ContentTemplate>

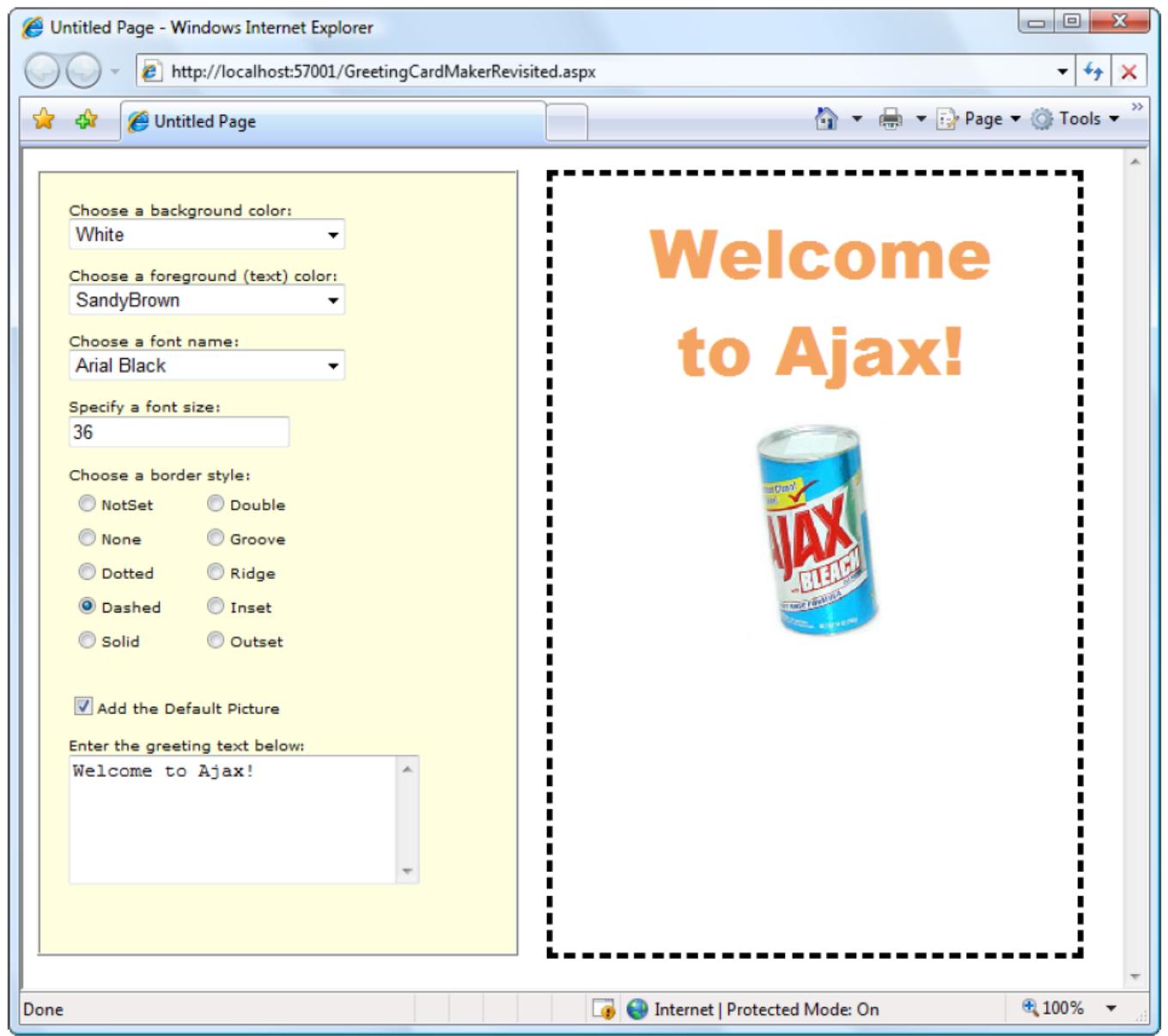
```

</asp:UpdatePanel>

## : Code-behind

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        // (Initialize all the controls here.)
    }
    else
    {
        // Refresh the greeting card.
        UpdateCard();
    }
}
```

UP، کنترل‌های درون خود را می‌بینید و هر رویدادی که یک trigger postback را کند، متوقف می‌کند. کنترلی مانند Listbox می‌باشد که موجب رخداد SelectedIndexChanged می‌شود. UP، این رویداد را نیز متوقف می‌کند.



در این مثال، کل صفحه درون یک UP قرار گرفته است و HTML کل صفحه Refresh می‌شود. گزینه بهتر، قرار دادن فقط label و تصویر در UP می‌باشد. البته این روش کار نمی‌کند. اگر سایر کنترل‌ها را از UP خارج کنید، رویدادهای آنها دیگر متوقف نمی‌شود و بنابراین کل صفحه را postback می‌کنند.

راه حل موجود این است که به UP بگویید که کنترل‌های را مانیتور کند(حتی آنهایی که خارج از UP قرار دارند). این کار را می‌توانید توسط اضافه نمودن trigger به UP انجام دهید. می‌توانید یک trigger برای هر دکمه اضافه نمایید.

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<!-- The controls for creating the greeting card go here. -->
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
```

```

<!-- This is the panel that shows the greeting card. -->

<asp:Panel ID="pnlCard" runat="server">
    <asp:Label ID="lblGreeting" runat="server"
        Width="272px" Height="150px" > </asp:Label>
    <asp:Image ID="imgDefault" runat="server" Width="212px"
        Height="160px" Visible="False">
    </asp:Image>
</asp:Panel>
</ContentTemplate>

<Triggers>
    <asp:AsyncPostBackTrigger ControlID="lstBackColor" />
    <asp:AsyncPostBackTrigger ControlID="lstForeColor" />
    <asp:AsyncPostBackTrigger ControlID="lstFontName" />
    <asp:AsyncPostBackTrigger ControlID="txtFontSize" />
    <asp:AsyncPostBackTrigger ControlID="lstBorder" />
    <asp:AsyncPostBackTrigger ControlID="chkPicture" />
    <asp:AsyncPostBackTrigger ControlID="txtGreeting" />
</Triggers>
</asp:UpdatePanel>

```

 نکته: نیازی به نوشتتن دستی trigger نیست. می‌توانید از پنجره Properties موجود در VS استفاده کنید. UP را انتخاب و روی ویژگی Triggers در Properties کلیک کنید. در کادر باز شده، می‌توانید برای هر کنترل یک trigger انتخاب کنید.

به UP، می‌گوید که رویداد پیش فرض را متوقف کند. در نتیجه بخش کوچکتری از صفحه پس از هر async request باشد refresh شود.

می‌توانید از trigger ها به روش دیگری نیز استفاده کنید. به جای اینکه بگویید، کنترل‌ها را مانیتور کنند، می‌توانید بگویید برخی از کنترل‌ها را نادیده بگیرند. برای نمونه فرض کنید دکمه ای درون UP دارید. با کلیک روی آن یک async request رخ می‌دهد و آن ناحیه refresh می‌شود. اگر بخواهید این کنترل، کل صفحه را postback کند باید یک PostBackTrigger اضافه کنید. (به جای .(AsynchronousPostBackTrigger

```

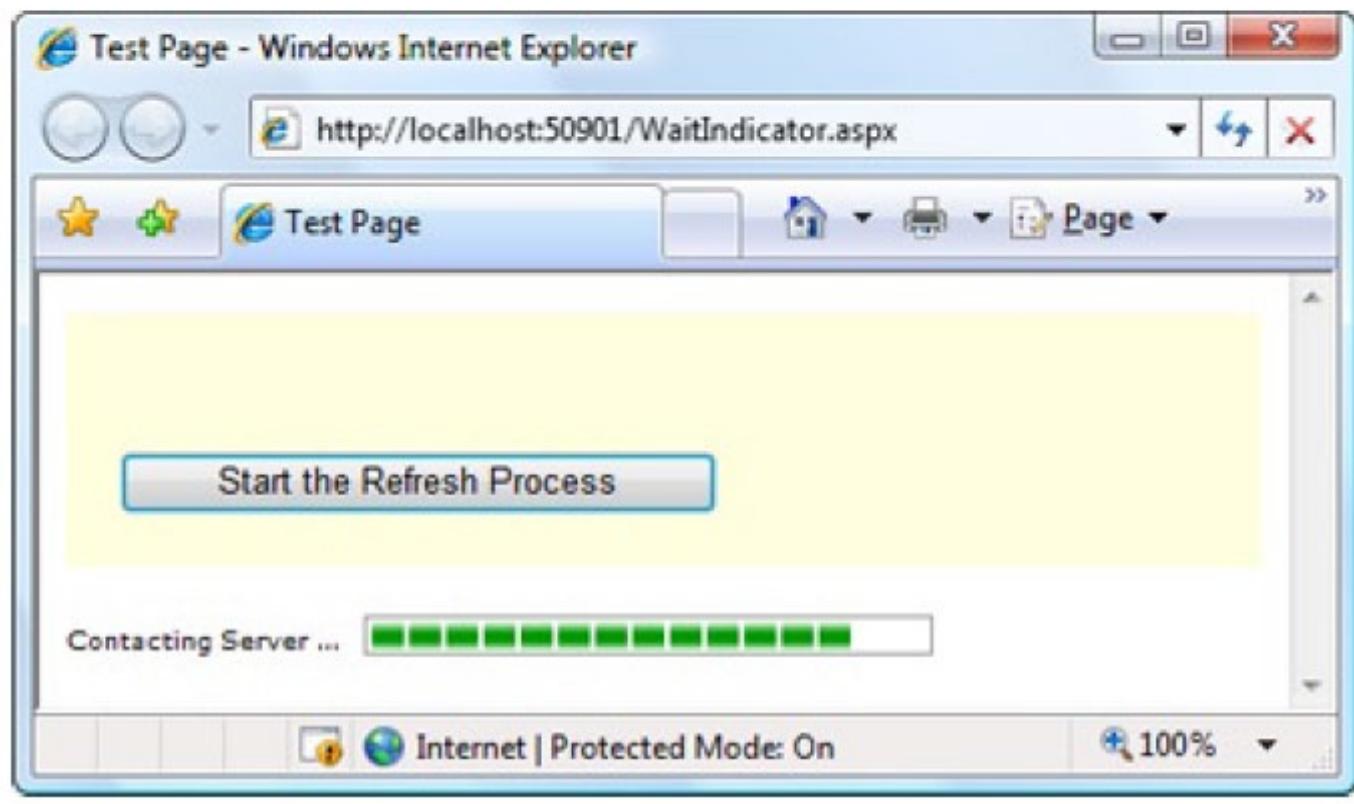
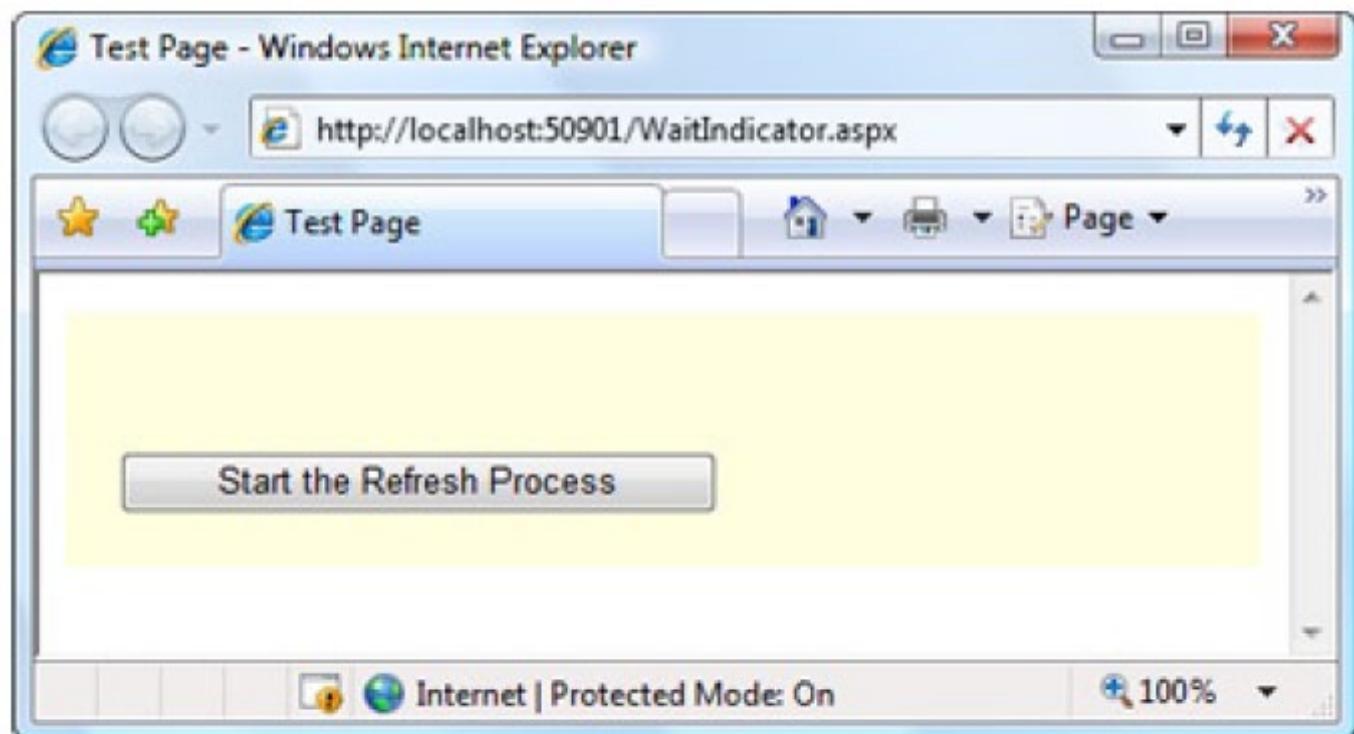
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Font-Bold="True"> </asp:Label>
        <br />
        <br />
        <asp:Button ID="cmdPostback" runat="server" Text="Refresh Full Page" />
    </ContentTemplate>
    <Triggers>
        <asp:PostBackTrigger ControlID="cmdPostback" />
    </Triggers>
</asp:UpdatePanel>

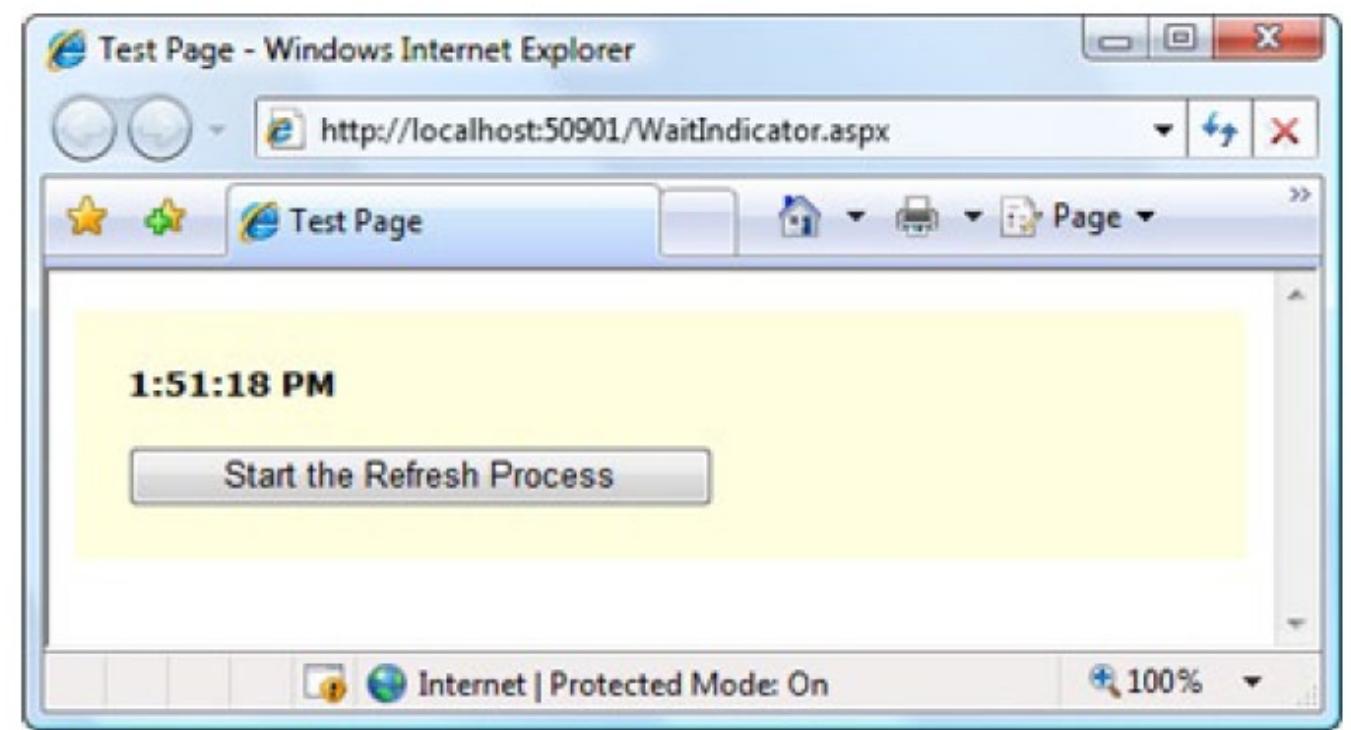
```

### استفاده از Progress Bar (نوار پیشرفت کار)

به دلیل اینکه UP بصورت async کار می‌کند، بنابراین اگر کاربر روی دکمه‌ای کلیک کند، تا زمانی که پاسخی از سرور برقراردد، اتفاق خاصی در صفحه نخواهد افتاد. بنابراین اگر عملیات زمان بر باشد، کاربر ممکن است فکر کند صفحه کار نمی‌کند و چندین بار روی دکمه کلیک کند.

ASP.NET، دارای کنترلی به نام UpdateProgress می‌باشد که با UpdatePanel می‌باشد. این کنترل به شما اجازه می‌دهد تا پیامی را در هنگام یک بروز رسانی رمان بر نمایش دهید.





```

<asp:UpdatePanel ID="UpdatePanel1" runat="server">

    <ContentTemplate>

        <div style="background-color: #FFFFE0; padding: 20px">

            <asp:Label ID="lblTime" runat="server" Font-Bold="True"> </asp:Label>
            <br />
            <br />
            <asp:Button ID="cmdRefreshTime" runat="server"
                OnClick="cmdRefreshTime_Click" Text="Start the Refresh Process" />
        </div>

    </ContentTemplate>

</asp:UpdatePanel>

<br />

<asp:UpdateProgress ID="updateProgress1" runat="server">

    <ProgressTemplate>

        <div style="font-size: xx-small">
            Contacting Server ... < img src = "wait.gif" alt = "Waiting..." />
        </div>
    </ProgressTemplate>

</asp:UpdateProgress>

```

## Code-behind

```
protected void cmdRefreshTime_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(TimeSpan.FromSeconds(10));
    lblTime.Text = DateTime.Now.ToString();
}
```

نیازی به اتصال مستقیم UpdateProgress به UpdatePanel نیست. کنترل UpdateProgress بصورت خودکار هر زمان که یک callback را شروع کند، نمایش داده می‌شود. اگر دارای یک صفحه پیچیده باشد که بیش از یک UpdatePanel دارد، می‌توانید با استفاده از ویژگی AssociatedPanelID آنها برای یکی از آنها به نمایش درآید.

## فعال سازی لغو عملیات

کنترل UpdateProgress، یک دکمه Cancel دارد که زمانی که کاربر روی آن کلیک کند، فوراً متوقف می‌شود. کد زیر را در انتهای صفحه قبل از <body>/> قرار دهید.

```
<%@ Page Language="#c" AutoEventWireup="true" CodeFile="WaitIndicator.aspx.cs"
Inherits="WaitIndicator" %>
```

```
<!DOCTYPE html>

<html>
<head id="Head1" runat="server">
    ...
</head>
<body>
    <form id="form1" runat="server">
        ...
    </form>
    <script type="text/javascript">
        var prm = Sys.WebForms.PageRequestManager.getInstance();
        prm.add_initializeRequest(InitializeRequest);
        function InitializeRequest(sender, args) {
```

```

if (prm.get_isInAsyncPostBack()) {
    args.set_cancel(true);
}

}

function AbortPostBack() {
    if (prm.get_isInAsyncPostBack()) {
        prm.abortPostBack();
    }
}

</script>

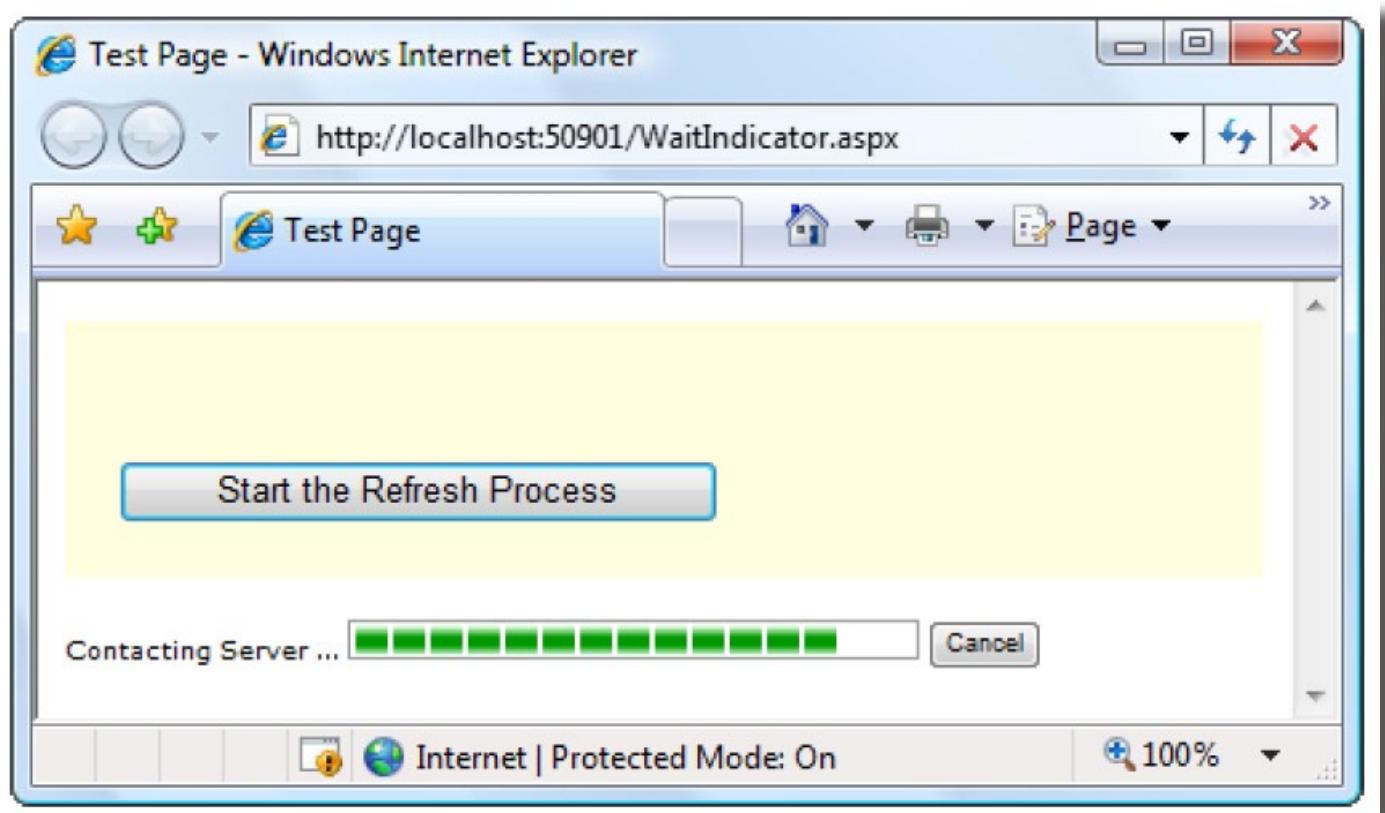
</body>
</html>

```

زمانی که این کد را اضافه نمودید، می‌توانید از کد JS برای فراخوانی متده AbortPostBack و کنسل کردن callback استفاده کنید.

```
<input ID = "cmdCancel" onclick = "AbortPostBack()" type = "button" value = "Cancel" />
```

اگر روی دکمه Cancel کلیک کنید، متده AbortPostBack() فوراً متوقف می‌شود. می‌توانید این دکمه را درون ProgressTemplate قرار دهید.



## زمان بندی شده Refresh

اگر نمی خواهید refresh صفحه وابسته به عملی از کاربر باشد و می خواهید در بازه های زمانی معینی خود کار انجام شود، باید از کنترل Timer استفاده کنید.

کد زیر صفحه را پس از یک دقیقه refresh می کند :

```
<asp:Timer ID = "Timer1" runat = "server" Interval = "60000" />
```

البته این کنترل سربار صفحه شما را بالا می برد و scalability را کاهش می دهد. پس برای استفاده از آن خوب فکر کرده باشید.  
این کنترل، کدهای JS سمت کلاینتی تولید کرده که یک timer جاوا اسکریپتی را شروع می کنند. زمانی که این javascript timer شروع بکار می کند (در بازه ای که شما تعیین کرده اید)، اسکریپت سمت کلاینت، یک postback را ایجاد می کند و یک رویداد سروری با نام Tick را fire خواهد کرد.

اگر شما این کنترل را درون UP قرار دهید، نتیجه آن بهتر می شود. UP، همه postback ها را به callback تبدیل کرده و از partial rendering برای بروز رسانی آن بخش خاص از صفحه استفاده می کند.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
    <ContentTemplate>
        ...
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
    </Triggers>
</asp:UpdatePanel>
```

برای کنترل Timer باید حتماً از trigger ها استفاده کنید.

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    // Update the tick count and store it in view state.
    int tickCount = 0;
    if (ViewState["TickCount"] != null)
    {
        tickCount = (int)ViewState["TickCount"];
    }
}
```

```
}

tickCount++;

ViewState[“TickCount”] = tickCount;

// Decide whether to disable the timer.

if (tickCount > 10)

{

    Timer1.Enabled = false;

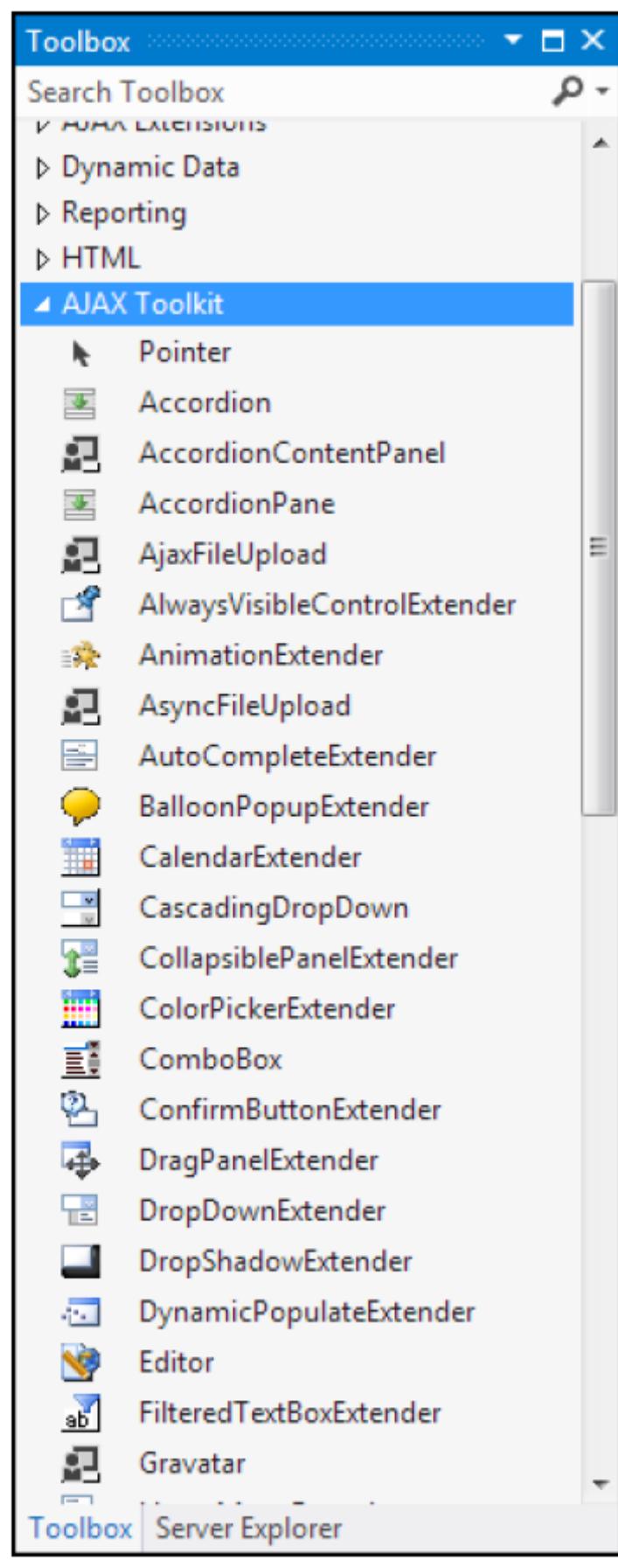
}

}
```

## ASP.NET AJAX Control ToolKit

این ابزار مجموعه‌ای از کنترل‌هایی که از کتابخانه‌های ASP.NET Ajax استفاده می‌کنند را دارد و رایگان می‌باشد. برای نصب آن می‌توانید به آدرس زیر رفته و در زبانه Download فایل آن را پیدا کنید:

<http://ajaxcontroltoolkit.codeplex.com>.



# بخش پنجم : ASP.NET پیشرفته

فیل شانزدهم: برنامه نویسی مبتنی بر کامپونت

~~Caching~~ فیل هفتم

~~Entity Framework , LINQ~~ فیل هشتم

~~ASP.NET AJAX~~ فیل نوزدهم

ASP.NET Deploy فیل بیستم: کردن برنامه های



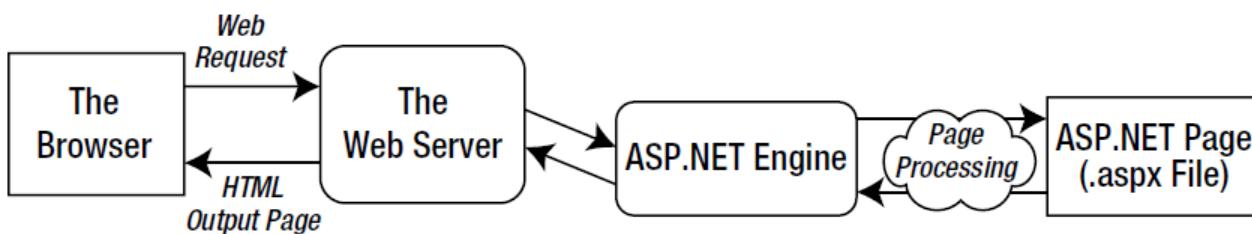
## Deploying ASP.NET Application

برنامه های ASP.NET، همیشه در تعامل با یک وب سرور می باشند که درخواست ها را از طریق HTTP دریافت و محتوای مورد نظر را در اختیار قرار دهد. زمانی که برنامه را در VS Express اجرا می کنید، از یک test web server با نام IIS استفاده می کنید. اما زمانیکه برنامه را deploy می کنید، نیاز به یک وب سرور واقعی دارید.

### چگونگی عملکرد وب سرور

ساده ترین کار وب سرور، فراهم نمودن صفحات HTML می باشد. زمانی که یک فایل را درخواست می کنید، وب سرور به سادگی آن را از هارد رایو خوانده (یا آن را از in-memory cache می خواند) و سند کامل را به مرورگر می فرستد.

زمانی که از وب سرور در تعامل با صفحات پویا استفاده می کنید، وب سرور هیچ ایده ای برای پردازش تگ های ASP.NET یا اجرای کد های #C# ندارد. بلکه از ASP.NET Engine، برای اجرای آنها بهره می برد. در شکل زیر می بینید که زمانی که صفحه Default.aspx را درخواست می کنید، وب سرور درخواست شما را به ASP.NET Engine می فرستد و آن نیز صفحه موردنظر خواست را لود کرده و محتوای درون آن را اجرا می کند. سپس کل سند HTML را ایجاد نموده و آن را به IIS بر می گرداند. IIS نیز آن را به کلاینت می فرستد.



در این فصل شما چگونگی کلیه کارهای deployment و پیکربندی وب سایت را یاد خواهید گرفت.

### Virtual Directory

زمانی که یک برنامه را روی وب سرور deploy کردید، آن برنامه از طریق چیزی با نام virtual directory در دسترس می باشد.

برای نمونه، وب سایت شما ممکن است درون پوشه ای در سرور با نام C:\MySite قرار داشته باشد. برای اینکه کاربر راه دور (کاربری که برنامه را از روی دستگاه دیگری اجرا می کند)، به این وب سایت از طریق مرورگر دسترسی داشته باشد، می توانید آن را از طریق [virtual directory در اختیار قرار دهید. زمانی که کاربر صفحه ای که در VD می باشد را درخواست می کند \(<http://WebServer/My->\), وب سرور به دنبال فایل مرتبط در پوشه فیزیکی مرتبط \(\[Site/Checkout.aspx\]\(http://WebServer/My-Checkout.aspx\)\)، می گردد.](http://WebServer/My-)

## Web Application URLs

برنامه وب را می‌توانید روی Internet یا LAN استفاده کنید.

یک backbone (شبکه)، از طریق گروهی از تجهیزات متصل به هم ایجاد می‌شود. در حقیقت اینترنت نیز چیزی بیش از یک backbone پرسرعت که میلیون‌ها Lan را به یکدیگر متصل می‌کند نمی‌باشد.

سنگ بنای اینترنت IP می‌باشد که هر کامپیوتر می‌تواند یک عدد واحد با نام IP Address در شبکه داشته باشد. IP از ۴ عدد بین ۰ تا ۲۵۵ تشکیل می‌شود که با نقطه از هم جدا می‌شوند. (۱۹۲,۱۶۸,۰,۱). به یاد سپردن این اعداد کار سختی می‌باشد. بنابراین وب سرورها عموماً روی اینترنت، یک domain name واحد مانند [com.karamoozesh.www](http://com.karamoozesh.www) ثبت می‌کنند. این نام از طریق یک کاتالوگ ای (DNS) Domain Name Service که در شبکه ای از سرورها نگهداری می‌شود، به یک IP Address نگاشت می‌شود. این شبکه DNS نام دارد. زمانی که [com.karamoozesh.www](http://com.karamoozesh.www) را تایپ می‌کنید، مرورگر با DNS تماس گرفته و IP Address نگاشت شده به این نشانی را جویا می‌شود و سپس با آن تماس می‌گیرد.

برای در دسترس بودن در اینترنت، وب سرور مورد استفاده شما باید در DNS registry باشد. برای قرار دادن آن در DNS registry باید یک IP ثابت داشته باشید.

برنامه‌های تحت وب ASP.NET، نیازی به اجرا از طریق اینترنت ندارند. بیشتر آنها درون شبکه کار می‌کنند.

اگر برنامه را درون یک virtual directory با نام MyWebApp قرار دهید، می‌توانید از طریق آدرس زیر در وب سرور به آن دسترسی داشته باشید:

<http://localhost/MyWebApp>

 نکته: به یاد داشته باشید که Loaljost URL می‌باشد که به آن loopback می‌گویند و همیشه به کامپیوتر کنونی اشاره دارد. نشانی IP آن نیز ۱۲۷,۰,۰,۱ می‌باشد.

اگر نام کامپیوتر شما MyWebServer باشد :

MyWebApp/ <http:// MyWebServer>

 نکته: اگر نام کامپیوتر را نمی‌دانید، روی آیکن کامپیوتر در دسکتاپ راست کلیک کنید.

اکنون فرض کنید، MyWebServer در DNS ثبت شده است ([www.MyDomain.com](http://www.MyDomain.com))، و در اینترنت قرار گرفته است. شما می‌توانید از آدرس زیر استفاده کنید:

<http://www.MyDomain.com/MyWebApp>

در نهایت، می‌توانید، همیشه از IP address کامپیوتر استفاده کنید.

<http://123.5.123.4/MyWebApp>

به دلیل آنکه شبکه های داخلی اغلب از IP address های داینامیک استفاده می کنند و آنها تغییر می کند، استفاده از نام کامپیوتر یا دامین، راه بهتری به نظر می رسد.

URL هایی که توسط VS استفاده می شود اندکی متفاوت می باشد. آنها دارای شماره پورت می باشند:

[aspx.Default/MyWebApp/localhost//:http](http://aspx.Default/MyWebApp/localhost//:http)

<http://localhost/۱۰۰:MyWebApp/Default.aspx>

وب سرور های حقیقی همیشه پورت ۸۰ را مانیتور می کنند و نیازی به نوشتن این پورت در URL نمی باشد.

## Web Farms

برخی از برنامه ها روی یک web farm اجرا می شوند که یک گروه از کامپیوترهای سرور می باشد که مسئول هندل کردن درخواست ها می باشد. معمولاً web farm ها برای برنامه های قدرتمند وبی رزو می شوند که نیاز به هندل کردن بار سنگین دارند، زیرا چند کامپیوتر راحت تر از یک وب سرور می توانند درخواست ها را هندل کنند.

Web farm ها به جای آنکه فایل های یک برنامه وب را روی یک وب سرور قرار دهند، یک کپی از آن را روی چندین وب سرور قرار خواهند داد. زمانی که یک درخواست دریافت می شود، به سمت یکی از این وب سرورها هدایت می شود. (بر اساس اینکه کدامیک از آنها بر سبکتری داشته باشند). البته برای ویرایش برنامه، باید مطمئن شوید که همه وب سرورها را با یک نسخه روز رسانی کرده اید.

کمپانی های وب هاستینگ، از web farm ها برای هاست چندین وب سایت استفاده می کنند. برای نمونه، وب سایت شما ممکن است روی بیش از یک وب سرور اجرا شود، اما هر یک از این وب سرورها، ممکن است چند وب سایت را روی خود داشته باشند. این موضوع مدل آنها را کمی پیچیده می کند، که به برنامه های مختلف وب امکان به اشتراک گذاری منابع را می دهد.

## (Internet Information Services (IIS

همانطور که ممکن است حدس زده باشد، deploy کردن یک برنامه وب، تنها فایل های برنامه وب شما روی یک وب سرور می باشد.

با استفاده از IIS موارد زیر را بدست می آورید :

- مطمئن می شوید که برنامه شما حتی بدون VS، نیز اجرا می شود.

- به کاربران روی سایر کامپیوترها اجازه اجرای برنامه های تحت وب را می دهید. (VS، تنها درخواست های روی کامپیوتر local را پاسخ می دهد).

- URL های برنامه وب، نیاز به پورت ندارد.

IIS، بر اساس سیستم عامل مورد استفاده شما دارای نسخه های متفاوتی می باشد :

- Windows Vista and Windows Server 2008 use IIS 7.
- Windows 7 and Windows Server 2008 R2 use IIS 7.5
- . Windows 8 and Windows Server 2012 use IIS 8 -

البته به جز یکسری امکانات خاص و تفاوت در کارایی، نسخه های مختلف بسیار شبیه هم هستند.

## نصب IIS روی نسخه ویندوز دسکتاپ

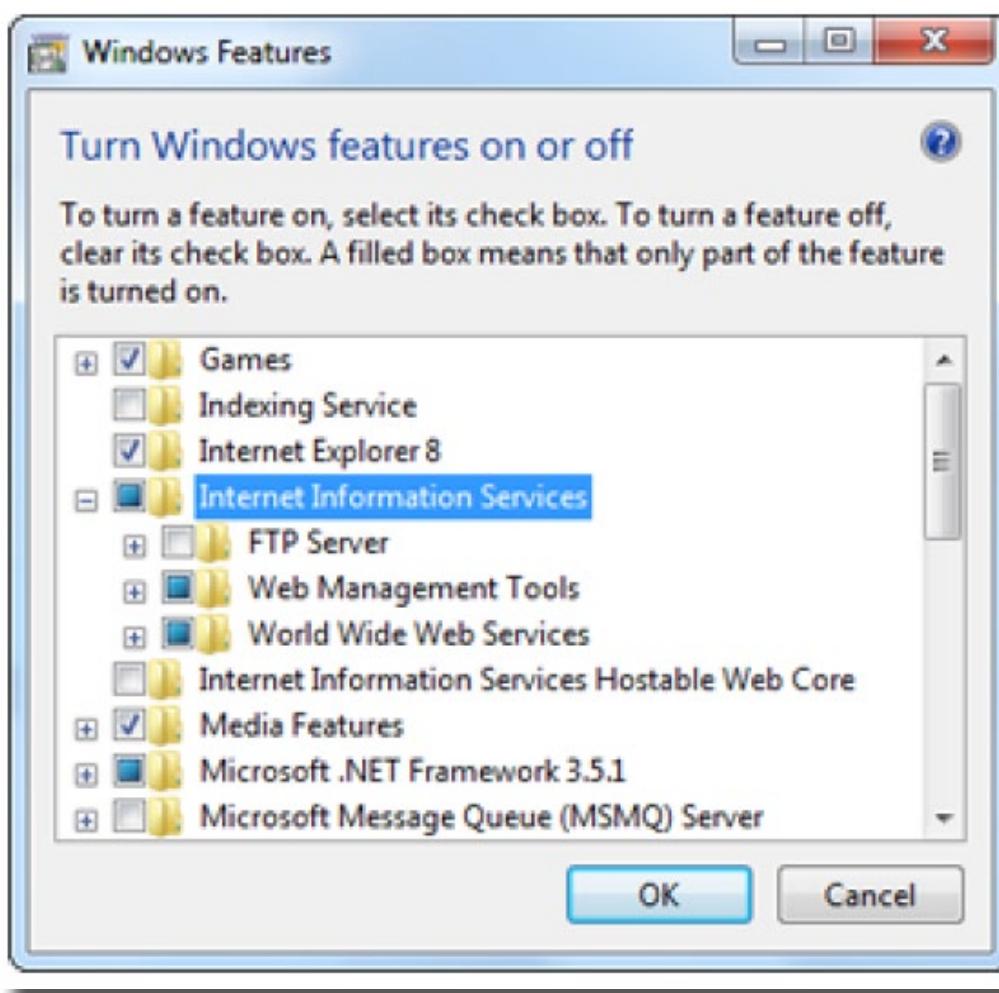
مراحل فعال سازی IIS 8 (در ویندوز ۸)، 7.5 (در ویندوز ویستا) یا 7 IIS (در ویندوز ۷) شبیه هم می باشد :

control panel را باز کنید. در ویندوز ویستا، ساده ترین روش، کلیک روی start و سپس کلیک روی Control Panel می باشد. در ویندوز ۸، بهترین راه، تغییر وضعیت به desktop mode و سپس کلیک راست روی start و انتخاب Control Panel از منو می باشد. (کتاب آموزش تصویری ویندوز ۸ را از سایت گروه کارآموزش تهیه کنید)

روی Programs کلیک کنید. -

روی "Turn Windows features on or off" کلیک کنید. -

آیتم Internet Information Services را پیدا و روی آن کلیک کنید. -



۱- ویندوز به شما اجازه فعال سازی بسیاری از امکانات IIS را بصورت جداگانه می‌دهد. برای مشاهده آنها، روی علامت بعلاوه در سمت چپ عبارت کلیک کنید تا باز شود.

۲- مطمئن شوید که ASP.NET را در زیر application development features~asp.net انتخاب کرده اید.

 نکته: اگر می‌خواهید از امکان پشتیبانی IIS در VS استفاده کنید که به شما اجازه IIS virtual directory را مستقیماً از new web site می‌دهد، باید آیتم IIS 6 Management Compatibility را انتخاب کنید. می‌توانید آن را در کادر

Internet Information Services → Web Management Tools → IIS 6 Management Compatibility

II S Metabase and IIS 6 Configuration Compatibility ~

پیدا کنید.

۳- اگر گزینه های IIS را انتخاب کردید، روی OK برای تکمیل تغییرات پیکربندی کلیک کنید.

 نکته: برای اینکه تست کنید آیا IIS نصب شده است یا نه، عبارت [localhost//:http](http://localhost//:http) را در مرورگر همان کامپیوتر تایپ کنید. اگر صفحه welcome را دیدید به درستی نصب شده است.

ویندوز ۸، شامل .NET 4.5 runtime می‌باشد که با نصب IIS می‌توانید کار با آن را شروع کنید. در نسخه های قبلی ویندوز، شما باید .NET 4.5 را نصب می‌کردید. برای نتیجه بهتر ابتدا IIS را نصب کنید و سپس .NET 4.5 را اضافه کنید. می‌توانید آن را با VS 2012 نصب کنید یا آن را جداگانه دانلود و نصب کنید.

<http://msdn.microsoft.com/en-us/netframework>

اگر ترتیب این مراحل را برعکس رعایت کنید، ابتدا .NET را نصب و سپس IIS را نصب کنید، مجبور می‌شوید کارهای بیشتری انجام دهید، زیرا .NET، قادر به اضافه نمودن Application Pool که در ادامه با آن آشنا می‌شوید نمی‌باشد. سریع‌ترین روش برای حل این مشکل، استفاده از ابزار aspnet\_regiis.exe می‌باشد. باید از نسخه ای از aspnet\_regiis.exe استفاده کنید که شامل .NET 4.5 باشد.

.(c:\Windows\Microsoft.NET\Framework\v4.0.30319)

اگر VS را نصب کرده اید می‌توانید از مسیر زیر استفاده کنید :

All Programs → Microsoft Visual Studio 2012 → Visual

Studio Tools → Developer Command Prompt

سپس عبارت زیر را تایپ کنید :

```
aspnet_regiis -ir
```

## نصب IIS، روی ویندوز سرور ۲۰۰۸

۱- Server Manager را باز کنید. برای این کار، روی

start>All Programs>Administrative Tools>Server Manager کلیک کنید.

۲- روی نود Roles کلیک کنید.

۳- روی Add Roles در بخش سمت راست پنجره کلیک نمایید.

۴- مراحل ویزارد را تا زمان رسیدن به مرحله Select Server Role کلیک کنید. سپس در مرحله server role را از لیست انتخاب کنید و next را بزنید.

۵- اکنون IIS به همراه .NET 3.5. runtime نصب شده است.

۶- اکنون باید .NET 4.5. را نصب کنید. ساده ترین روش استفاده از Web Platform Installer می‌باشد که می‌توانید آن را از <http://msdn.microsoft.com/netframework> دانلود نمایید.

## نصب IIS روی Windows Server 2012

مراحل این نصب نیز شبیه روش بالا اما با اندکی تفاوت می‌باشد.

۱- روی کاشی (tile) start در صفحه Server Manager کلیک کنید.

۲- به دنبال منو در گوشه سمت راست بالا از صفحه Server Manager بگردید و Manage>Add Roles را انتخاب نمایید.

۳- گزینه "Role-based or feature-based installation" را انتخاب و next را کلیک کنید.

۴- سرور خود را انتخاب و رو next کلیک کنید.

۵- Web Server Role را از لیست role ها انتخاب و روی next کلیک کنید. البته می‌توانید امکانات و service های بیشتری را نیز نصب کنید. روی confirmation کلیک کنید تا به صفحه confirmation برسید.

۶- روی Install کلیک کنید.

۷- زمانی که به پایان رسید، IIS به همراه .NET 4.5 نصب شده است.

## IIS Manager با استفاده از مدیریت وب سایت

زمانی که IIS نصب شود بصورت خودکار یک پوشه در مسیر c:\inetpub\wwwroot\، ایجاد می‌کند که وب سایت شما را ارائه می‌دهد. هر فایلی در این پوشه، طوری به نظر می‌رسد که گویا در ریشه وب سایت شما هستند.

برای اضافه نمودن صفحات بیشتر به وب سرور خود، می‌توانید فایل‌های HTML, ASP, ASP.NET یا HTML را مستقیماً درون این پوشه قرار دهید. برای نمونه، فایل TestFile.html را به این پوشه اضافه کنید. می‌توانید به آن از طریق آدرس

<http://localhost/TestFile.html>

دسترسی داشته باشید. حتی می‌توانید زیر پوشه‌هایی برای گروه بندی منابع اضافه نمایید. برای نمونه می‌توانید فایل c:\inetpub\wwwroot\Mysite\MyFile.html را از آدرس

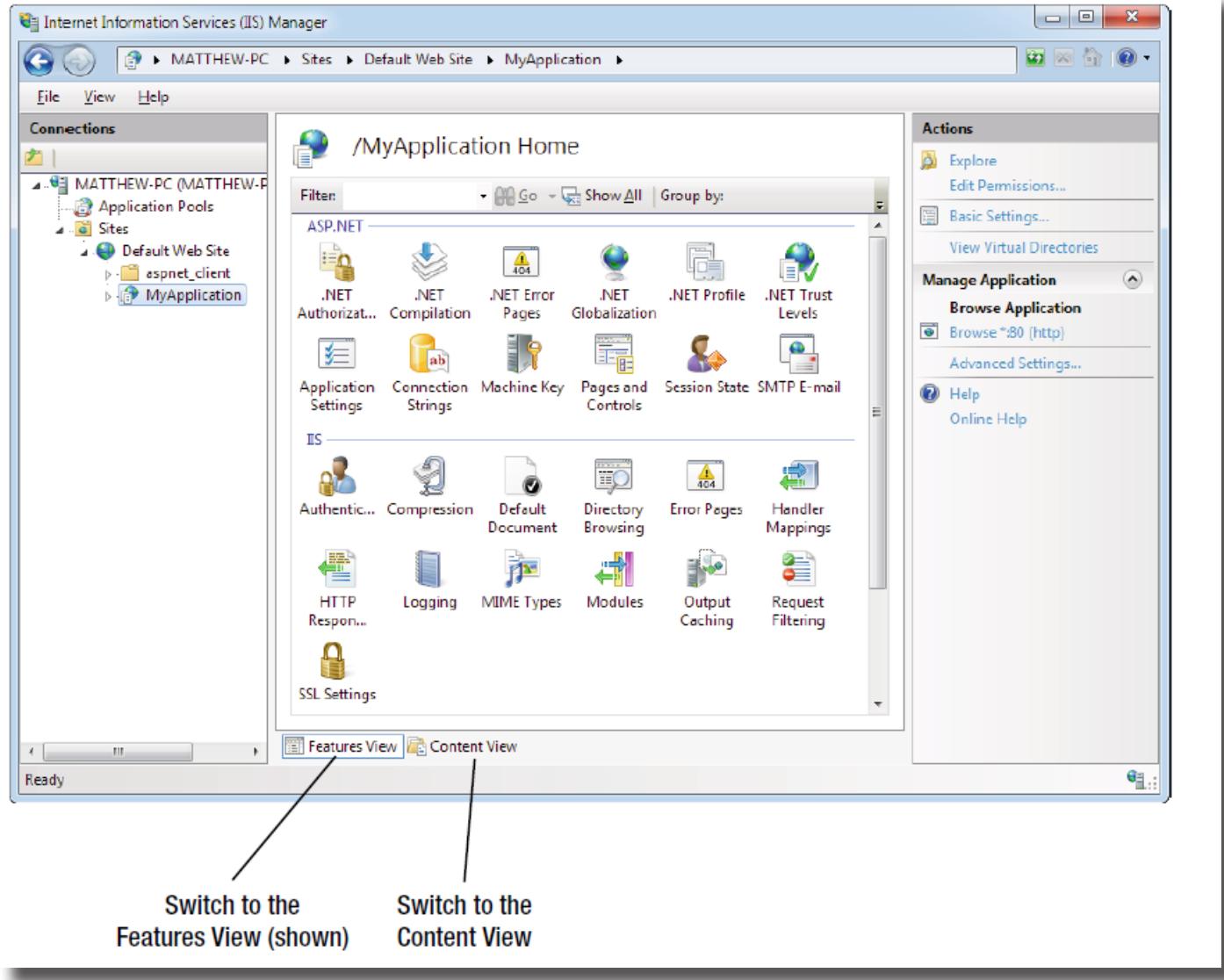
<http://localhost/Mysite/MyFile.html>

مشاهده کنید.

استفاده از پوشه wwwroot ساده است ولی برای سازماندهی‌های ضعیف مناسب می‌باشد. برای استفاده از ASP یا ASP.NET، باید خود را برای هر برنامه وب ایجاد کنید. با یک virtual directory، می‌توانید پوشه‌های فیزیکی (روی هر درایو از کامپیوتر) روی وب سرور را همان گونه که به پوشه wwwroot دسترسی داشتید، در اختیار قرار دهید.

قبل از شروع کار باید IIS Manager را باز کنید. یک راه سریع برای این کار، جستجوی inetmgr (در ویندوز ۸ یا ویندوز سرور ۲۰۱۲ در App search و در ویندوز ۷ تایپ IIS Manager در searchbox می‌باشد).

از پنجره باز شده، از سمت چپ، گزینه Default Web Site را باز کرده تا کلیه virtual directory‌های درون آن را ببینید. هر کدام یک برنامه وب جداگانه را ارائه می‌دهد.

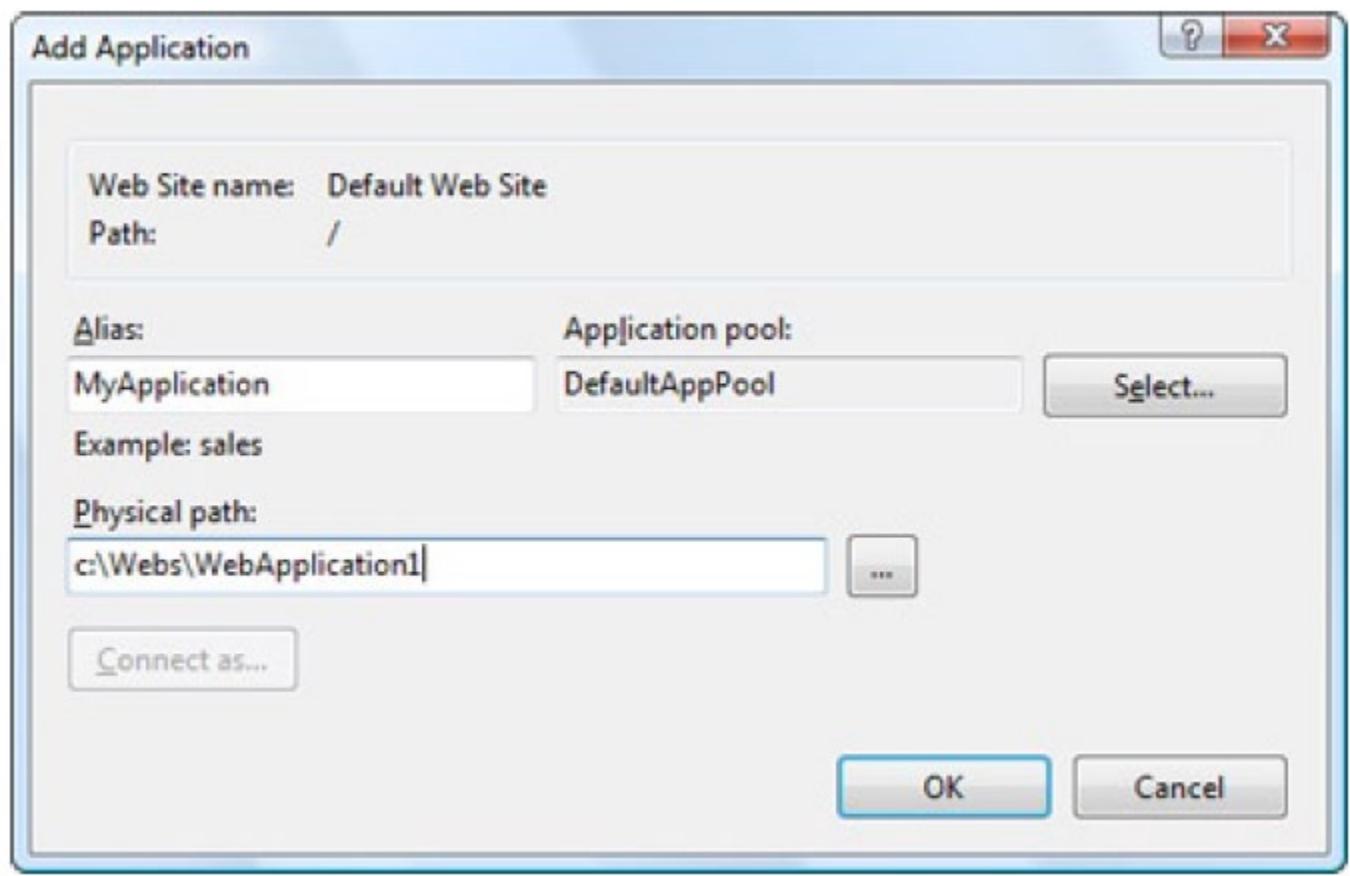


در کادر وسط، مجموعه ای از آیکن ها را می بینید که برای انجام پیکربندی های مختلف استفاده می شوند. دو زبانه FeatureView و Content View وجود دارد که دومی برای مشاهده محتوا درون هر پوشه انتخابی در کادر سمت چپ می باشد.

## ایجاد یک Virtual Directory

اولین مرحله که معمولاً باید در هنگ `hl deploy` انجام دهد، ایجاد پوشه فیزیکی برای ذخیره صفحات می باشد. مرحله دوم، در اختیار قرار دادن این پوشه به عنوان یک VD، از طریق IIS می باشد. این یعنی که وب سایت بصورت عمومی در دسترس سایر کامپیوترهای متصل به کامپیوتر شما می باشد. کامپیوتر remote قادر به دسترسی به پوشه فیزیکی شما (مثلًا `c:\mysite`) نمی باشد. برای اینکه دستگاه `remote` بتواند به این پوشه از طریق IIS دسترسی داشته باشد باید آن را به یک VD نگاشت (map).

- ۱- برای ایجاد یک VD جدید برای یک پوشه فیزیکی موجود، نود کامپیوتر کنونی را و نود Site در زیر آن را نیز باز نمایید.
- ۲- روی Add Application کلیک راست کرده و Default Web Site را انتخاب نمایید.



۳- ابتدا باید alias (یک نام می‌باشد که کامپیوتر remote برای دسترسی به فایل‌های درون VD از آن استفاده می‌کند) را وارد کرد. برای نمونه اگر alias شما MyApp باشد، و کامپیوتر شما MyServer باشد، می‌توانید صفحه را از نشانی مانند

<http://MyServer/Mapp/MyPage.aspx>,

درخواست کنید.

۴- سپس، می‌توانید physical path را انتخاب کنید. این یک پوشه روی هارددرایو می‌باشد که به عنوان VD ارائه خواهد شد. برای نمونه پوشه wwwroot یک پوشه فیزیکی می‌باشد که به عنوان VD ریشه وب سرور شما استفاده می‌شود.

۵- سپس باید application pool را انتخاب نمایید. یک AP، گروهی از تنظیمات بکار رفته در یک یا چند برنامه وب می‌باشد. گزینه DefaultAppPool به نظر کامل می‌رسد ولی بر اساس ASP.NET 2.0 می‌باشد. برای اینکه به یک VD قابلیت اینکه هاست یک برنامه ASP.NET باشد را اعطا کنید، باید AP با نام ASP.NET v4.5 را انتخاب نمایید. روی OK کلیک کنید.

۶- سپس روی OK در کادر Add Virtual Directory کلیک نمایید.

فرض کنید، یک VD، ایجاد کرده اید که پوشه MyApp روی کامپیوتری با نام MyServer را فراخوانی می‌کند. (c:\MyApp). اگر یک زیر پوشه مانند c:\MyApp\MyFiles قرار می‌گیرد و کلاینت از طریق <http://myserver/MyApp/MoreFiles/Somefile.html>.

به آن دسترسی دارد.

تصویرت پیش فرض، زیر پوشه ها نیز مجوزهای VD را ارث می‌برند. البته می‌توانید این تنظیمات را با استفاده از IIS Manager تغییر دهید.

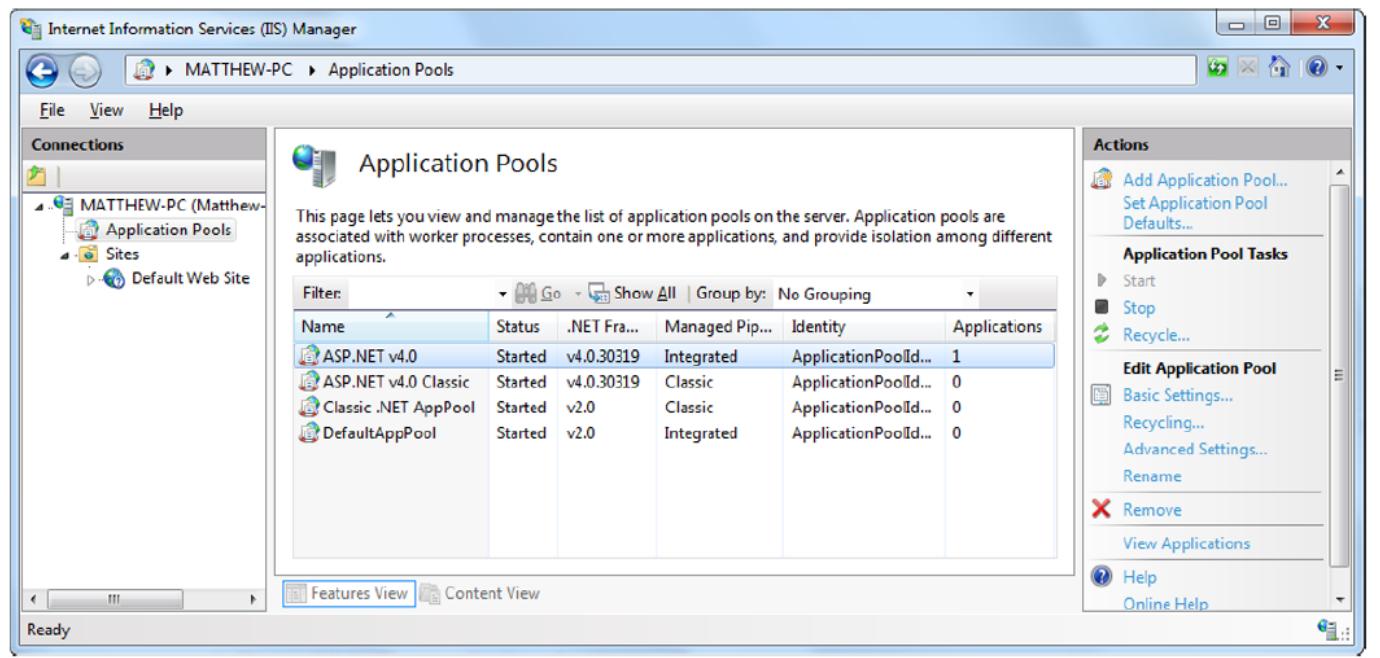
یک اشتباه رایج در ASP.NET وجود دارد. فرض کنید یک وب سایت در پوشه ای به نام c:\Application\WebApps\Site1، دارید. این پوشه باید برای ایجاد VD استفاده شود. اگر تصادفاً، یک VD، برای پوشه پدر ایجاد کنید (c:\Applications\WebApps)، ممکن است خطای مشاهده نکنید. زیرا هنوز قادر به دسترسی به فایل‌های موجود در Site1 هستید. زمانیکه برای دسترسی به یکی از صفحات موجود در Site1 تلاش می‌کنید، یک صفحه خطا مشاهده خواهد کرد به شما اعلام می‌کند بعضی از تنظیمات در web.config معتبر نمی‌باشد. مشکل این است که تنظیمات معتبر هستند ولی تنها در سطح برنامه و نه در سطح زیر پوشه. برای حل این مشکل، VD اشتباه را حذف کنید و مجدداً به درستی آن را ایجاد نمایید.

## آشنایی با Application Pool

برخی از تنظیمات مستقیماً درون برنامه انجام می‌شود و برخی دیگر از طریق امکانی به نام application pool بیشتر پیکربندی‌های برنامه‌های تحت وب، از طریق VD، انجام می‌شود. Web Application Pool، یک گروه کوچک از تنظیمات سطح پایین مانند، تعداد درخواست‌های مجاز برای قرار گرفتن در صف انتظار، قبل از ارسال service unavailable (پیش فرض ۱۰۰۰)، که تنها روی برنامه‌های ASP.NET بکار می‌رond را تعیین می‌کند.

### دو تنظیم مهم در application pool :

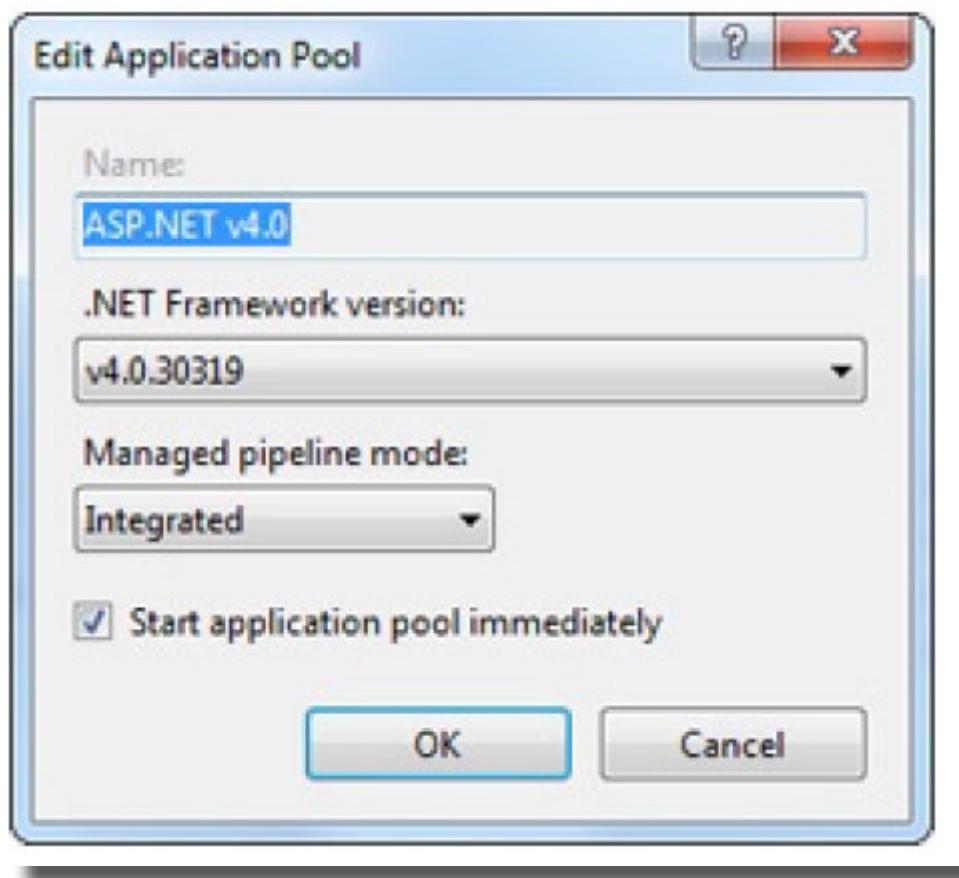
- نسخه ASP.NET که آن را برای پردازش درخواست‌های وب سایت شما اجرا می‌کند.
- Windows account که IIS برای اجرای برنامه شما از آن استفاده می‌کند.



 نکته: واژه Application Pool در نام Classic نشان دهنده این است که برنامه از روش متفاوتی برای تعامل با IIS (که با 6 IIS و قبل تر تطبیق دارد) استفاده می‌کند. عموماً تنها زمانی از مد classic استفاده می‌شود که کامپوننت ایجاد کرده اید که درخواست‌های ASP.NET را روی یک IIS قدیمی پردازش می‌کند و شما می‌خواهید مطمئن شوید روی نسخه‌های جدیدتر نیز کار می‌کند.

Basic Setting pool را انتخاب و روی کلیک کنید. تنظیمات ابتدایی به شما اجازه تغییر جزئیات را می‌دهد. این جزئیات، نسخه .NET، استفاده از مد integrated یا classic و امکان اینکه آیا pool بصورت خودکار با شروع به کار کامپیوتر آغاز شود (بنابراین از تأخیر غیر ضروری در اولین باری که وب سرویس درخواستی را برای وب سایت شما دریافت می‌کند، جلوگیری می‌شود) یا نه.

Advanced Setting pool را انتخاب و روی کلیک کنید.



اگر می‌خواهید یک pool با تنظیمات سفارشی (مانند حساب کاربری ویندوز متفاوت) داشته باشید می‌توانید از این امکان استفاده نمایید.

Actions Pane در View Applications را انتخاب و روی See What applications are in a pool کلیک کنید. سایت‌هایی که ممکن است از تغییر تنظیمات این application pool تأثیر بپذیرند را نمایش می‌دهد.

## ASP.NET Account

برخی از مشکلات موجود در ASP.NET deploy کردن، امنیتی می‌باشد. زمانی که وب سرور برنامه شما را اجرا کند، همه کارهای خود را تحت عنوان یک windows account خاص انجام می‌دهد که دارای یک مجموعه از امتیازهای محدود می‌باشد. account مورد استفاده، به وب سروری که استفاده می‌شود بستگی دارد:

- اگر از test server موجود در VS، استفاده کنید، سرور بر اساس account شما اجرا می‌شود. این یعنی، سرور همه مجوزهای شما را دارد و در نتیجه، شما در هنگام تست برنامه با مشکل روبرو نخواهید شد.
- اگر از 7 IIS، استفاده می‌کنید، سرور از network service استفاده می‌کند. این یک account خاص می‌باشد که ویندوز در هنگام نصب آن را ایجاد می‌کند. این کار بیشتری انجام نمی‌دهد اما می‌تواند به منابع شبکه دسترسی داشته باشد.
- اگر از 7.5 IIS یا 8 استفاده می‌کنید، account مورد استفاده، بستگی به application pool دارد. برای نمونه، یک AP با نام IISAppPool\ASP.NET v4.5، از account با نام ASP.NET استفاده می‌کند که آن را بصورت خودکار ایجاد کرده است.

 نکته: Network Service application pool identity، دارای مجوزها و محدودیت‌های مشابه هستند. IIS 7.5 با اضافه نمودن application pool identity، پیکربندی وب سرورهایی که هاست چند برنامه وب هستند را ساده تر کرده است. اگر همه این برنامه‌ها از network service استفاده می‌کردند، دقیقاً دارای مجوزهای مشابهی می‌بودند که این یعنی مجوز اضافه شده به یک برنامه در دسترس سایر برنامه‌ها نیز قرار می‌گیرد. البته این مشکل با ایجاد application domain های جداگانه که دارای account های مختلفی می‌بودند قابل حل بوده است. همه این کارها بصورت خودکار انجام می‌دهند.

شاید بگویید چرا کد ASP.NET، با سایر account ها اجرا نمی‌شود . مثلاً با account کاربری که از برنامه استفاده می‌کند. پاسخ این است که کاربر نهایی دارای حساب کاربری ویندوزی روی وب سرور نمی‌باشد. حتی اگر هم باشد، آن account، دارای حقوق مشابه ASP.NET enging نمی‌باشد.

باید از حسابی که به اندازه کافی محدود می‌باشد استفاده نمایید که نتواند توسط کاربران غیرمجاز استفاده شود ولی دارای مجوزهای مورد نیاز برای اجرای کدهای شما باشد. هم network account و هم application pool account دارای مجوزهای محدود مناسبی می‌باشد.

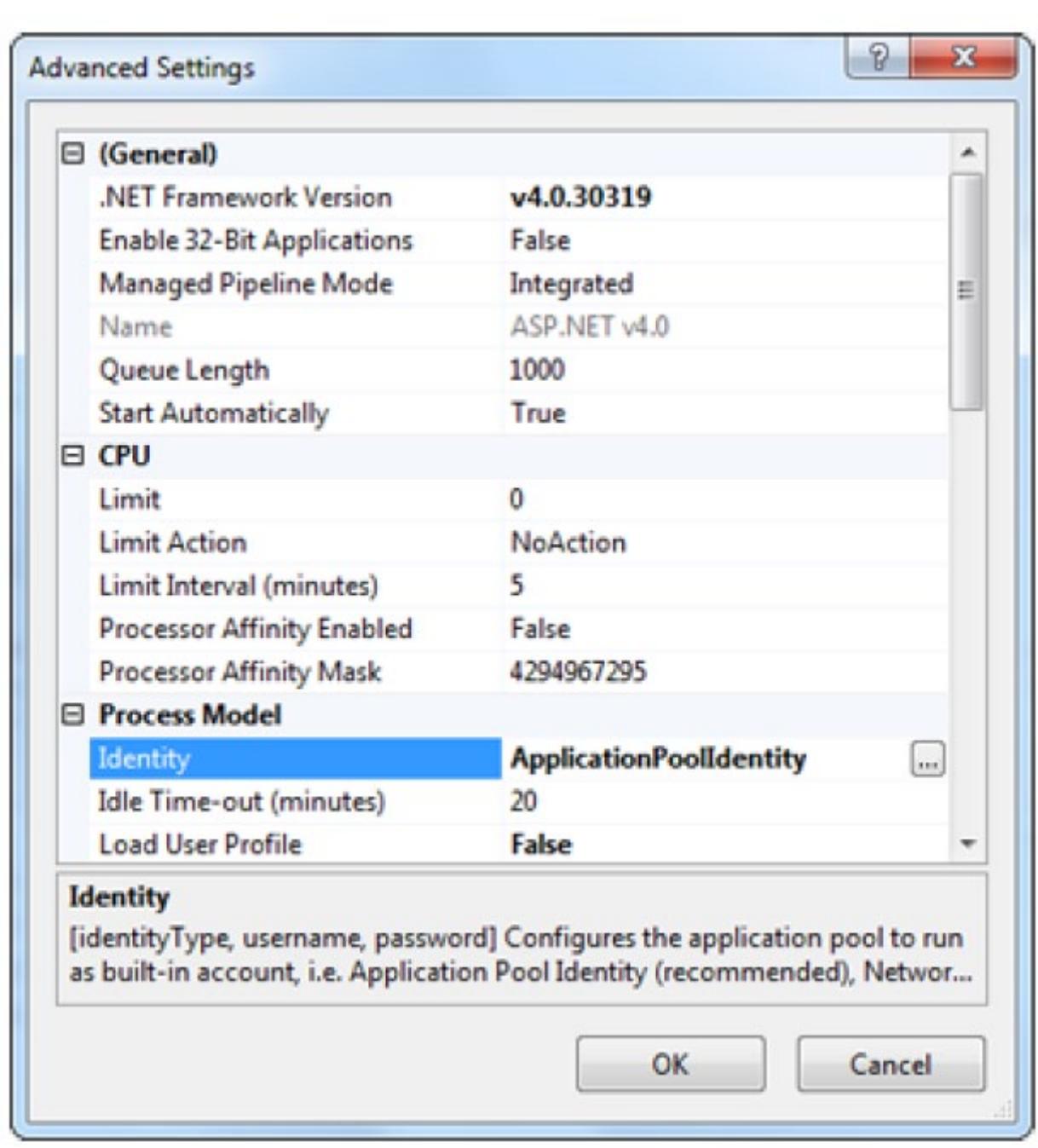
بصورت پیش فرض، ASP.NET account، اجازه اجرای کارهایی نظیر خواندن windows registry، بازیابی اطلاعات از بانک اطلاعاتی، یا نوشتن در محل‌های مختلف در local hard drive، را ندارد. این حساب اجازه انجام کارهای معمولی مانند دسترسی به پوشش، c:\Windows\Microsoft.NET\Framework64\v4.0.30319\Temporary ASP.NET Files برای کامپایل و cache کردن صفحات وب را دارد.

تنظیمات امنیتی ASP.NET، و network service account ها برای جلوگیری از حمله به وب سرور طراحی شده اند .

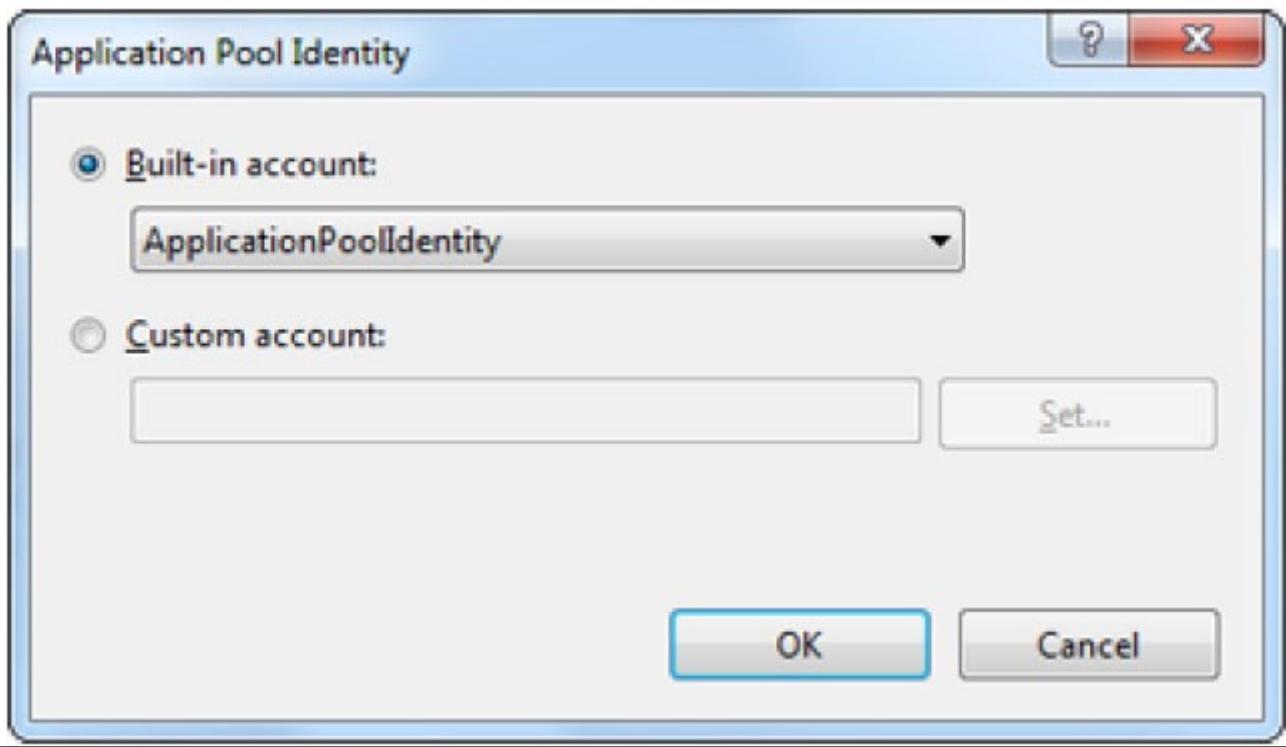
 نکته : قبل از تغییر حساب کاربری مورد استفاده ASP.NET، مطمئن شوید تأثیرات آن را می‌دانید. اگر از یک حساب با چند مجوز بیش از نیاز خود استفاده کرده اید، در را برای هکرهای باز کرده اید.

## ASP.NET Account تغییر

حساب ویندوزی، یکی از تنظیمات پیشرفتی می‌باشد. برای تغییر آن application pool مرتبط می‌باشد. برای انتخاب و روی Advanced Setting کلیک کنید.



برای تغییر Identity setting آن را انتخاب و روی (...) کلیک نمایید.



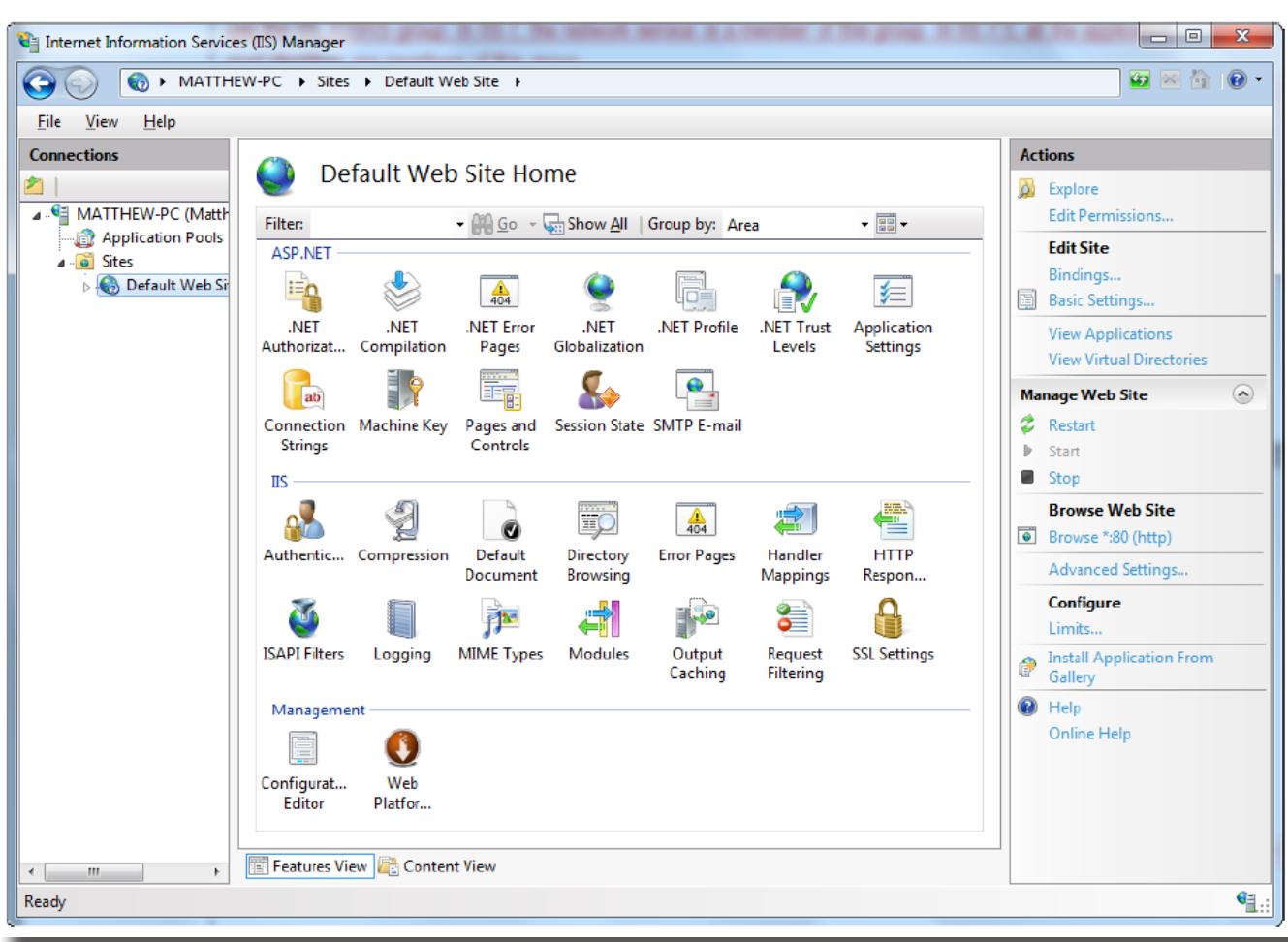
و IIS7.5 (در ApplicationPoolIdentity از پیش تعریف شده شامل، account یکی از انواع است) Network Service (شبیه Local Service در IIS7)، Network Service (در IIS8)، Local System (که به عنوان administrator با مجوزهای توسعه یافته می‌باشد) را انتخاب نمایید. های شبکه) یا

روش اطلاعات شما روی وب سرور کد می‌شود بنابراین توسط هکرها مورد استفاده قرار نمی‌گیرد.

### اختیارات بیشتر به ASP.NET account

تغییر Account ای که استفاده می‌کند، یک مرحله دارای ریسک می‌باشد. برای نمونه، تصور کنید که یک صفحه وب ایجاد کرده اید که به کاربران اجازه آپلود فایل را می‌دهد. اگر این صفحه را با دقت طراحی نکرده باشید، ممکن است کاربر برنامه را وادار به آپلود فایل در مکانی که مجاز نیست (مانند c:\windows directory) بکند. بهتر است اغلب مجوزهایی را به یک گروه اختصاص دهید نه به یک کاربر.

## پیکربندی یک وب سایت



پیکربندی وب سایت به سه گروه تقسیم می‌شود. IIS، ASP.NET و مدیریت.

### آیکن های پیکربندی ASP.NET

شما از این آیکن های گرافیکی استفاده می‌کنید و IIS فایل web.config را برای شما ویرایش می‌کند. این کار شبیه کاری است که WAT انجام می‌داد. در گروه ASP.NET موارد زیر را می‌بینید:

#### - NET Authorization.

- نحوه کامپایل کدهای ASP.NET، قبل از اجرای آنها را مشخص می‌کند.

- یک صفحه خطای سفارشی که در زمان رخدادن خطا نمایش داده خواهد شد را مشخص می‌کند. (زمانی که صفحه ای درخواست می‌شود که موجود نیست یا سرور خیلی مشغول است و نمی‌تواند درخواست را هندل کند).

- تنها زمانی تأثیر دارد که از اطلاعات culture مانند زبان های مختلف برای متن موجود در صفحه استفاده کرده باشد. بنابراین می‌توانید آن را برای گرفتن اطلاعات از مرورگر درخواست کننده پیکربندی کنید.

**.NET Profile**

NET Trust Level : توسط شرکت های وب هاستینگ استفاده می شود. می توانید سطح اطمینان پایین تری برای وب سایت مشخص کنید. بنابراین کد نمی تواند کارهایی به غیر از مجوزهایی که شما داده اید را انجام دهد.

**Application Setting**

Connection String : رشته اتصال موجود در فایل web.config را تغییر می دهد.

**Machine Key**

Pages and Controls : المان `<pages>` را پیکربندی می کند. می توانید view master page, theme را تعیین و state را خاموش کنید.

**Session state**

IIS SMTP E-Mail : به IIS می گوید که پیام های e-mail را که شما از طریق کد نویسی از درون برنامه ارسال می کنید، هندل کنند. این پیام ها می توانند درون پوشه تعیین شده ذخیره شوند یا از طریق SMTP Server ارسال شوند.

Request Filtering : انواع فایل هایی که IIS قبول نمی کند را مشخص خواهد کرد. برای نمونه اگر وب سرور درخواستی برای فایل `.config`, `.vb`, `.cd`, `.mdf`, `.bas` پسوند آنها، درخواست ها را رد می کند.

SSL Settings : وب سایت شما را برای نیاز به SSL Connection, پیکربندی می کند.

**Default Page**

VD، با آدرس `http://localhost/MySite` را در نظر بگیرید. یک کاربر می تواند یک صفحه در این پوشه را با استفاده از `http://localhost/MySite/MyPage.aspx`,

مشاهده کند. اما اگر کاربر چیزی شبیه

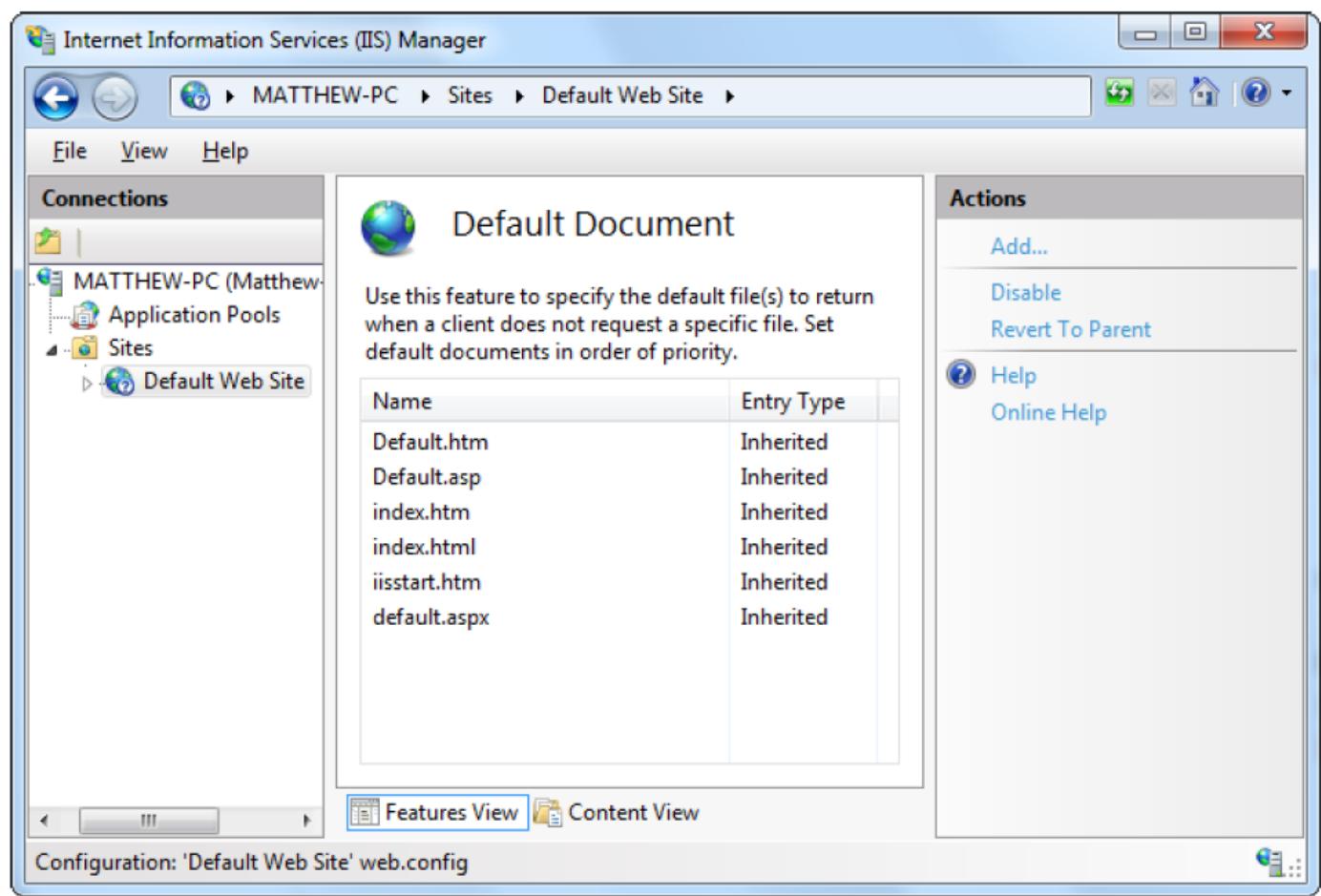
`http://localhost/MySite`

را در مرورگر تایپ کند چه پیش خواهد آمد.

در این مورد، IIS، لیستی از default document ها را که برای VD، تعریف شده است، امتحان می کند. برای مشاهده documents، VD، را انتخاب و روی آیکن Default Document کلیک کنید.

IIS، لیست default document را از بالا به پایین اسکن کرده و اولین صفحه منطبق را برمی گرداند. با استفاده از لیست موجود در شکل زیر، IIS ابتدا فایل Default.asp و پس Default.htm و... را چک می کند.

شما به سادگی می‌توانید لیست default document را تغییر دهید. برای اضافه کردن یک سند پیش فرض جدید روی Add کلیک کنید. می‌توانید ترتیب آنها را تغییر دهید.



## Custom Error Page

در درخواست یک صفحه وب، خطاهای مختلفی می‌تواند رخ دهد. کاربر ممکن است تلاش کند به یک فایل غیر مجاز دسترسی داشته باشد، فایل مورد نظرش موجود نباشد یا سرور خیلی مشغول باشد یا صفحه خودش یک خطای هندل نشده ایجاد کند. IIS به دو روش عمل می‌کند. اگر یک صفحه را بصورت locally درخواست کرده باشید (از مرورگری که روی وب سرور اجرا می‌شود)، یک خطای دسترسی صفحه مانند شکل زیر داده می‌شود. اگر صفحه را از کامپیوتر دیگر درخواست کنید، IIS، امنیت خوبی را نگهداری کرده و توضیحات بیشتری از خطابرمی گرداند.

## صفحه خطا برای درخواست local

**IIS 7.5 Detailed Error - 404.0 - Not Found - Windows Internet Explorer**

http://localhost/does\_not\_exist

Favorites IIS 7.5 Detailed Error - 404.0 - Not Found Page Safety Tools ?

# Server Error in Application "DEFAULT WEB SITE"

Internet Information Services 7.5

**Error Summary**

**HTTP Error 404.0 - Not Found**

**The resource you are looking for has been removed, had its name changed, or is temporarily unavailable.**

**Detailed Error Information**

Module	IIS Web Core	Requested URL	http://localhost:80/does_not_exist
Notification	MapRequestHandler	Physical Path	C:\inetpub\wwwroot\does_not_exist
Handler	StaticFile	Logon Method	Anonymous
Error Code	0x80070002	Logon User	Anonymous

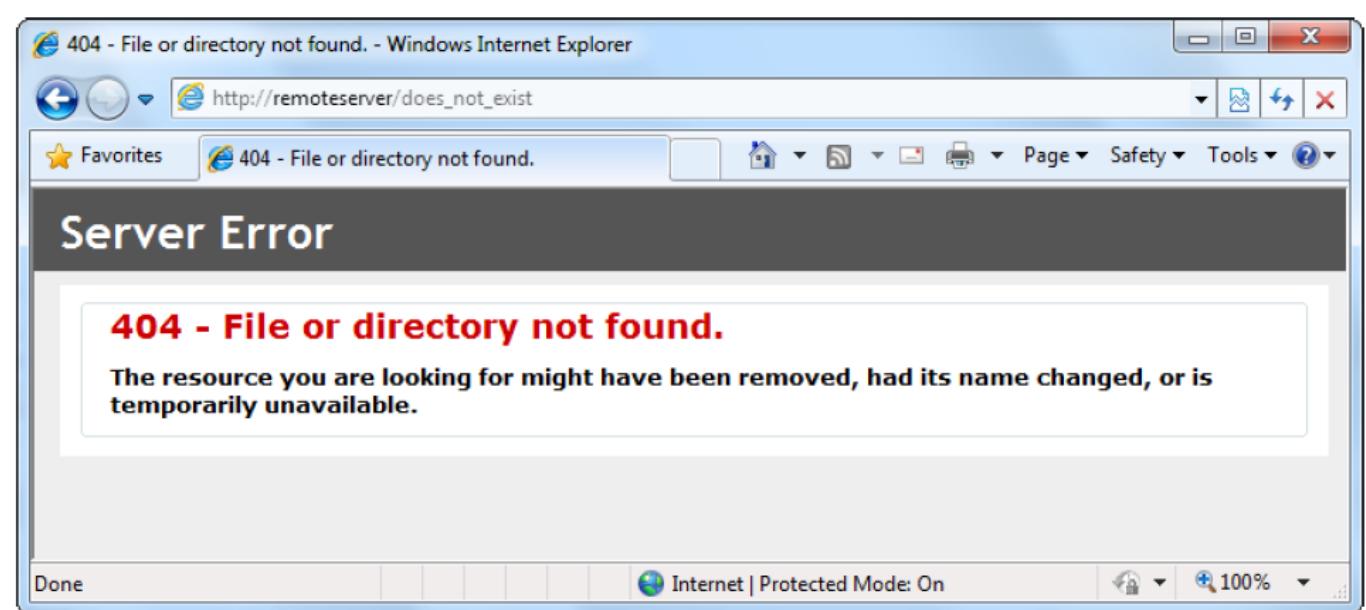
**Most likely causes:**

- The directory or file specified does not exist on the Web server.
- The URL contains a typographical error.
- A custom filter or module, such as URLScan, restricts access to the file.

**Things you can try:**

- Create the content on the Web server.
- Review the browser URL.
- Create a tracing rule to track failed requests for this HTTP status code and see which module is calling SetStatus. For more information about creating a tracing rule for failed requests, click [here](#).

Done | Internet | Protected Mode: On | 100% |



برای پیکربندی صفحه خطای سفارشی، VD را انتخاب و روی آیکن Error Pages دوبار کلیک کنید. لیستی از خطاهای HTTP مشاهده خواهید کرد که به صفحات خطای HTML نگاشت شده است. می‌توانید آیتم‌ها را اضافه یا حذف کنید یا روی یک خطا برای انتخاب فایل HTML متفاوت کلیک کنید. برای تعیین یک صفحه خطای پیش‌فرض، روی Edit Feature Setting کلیک کنید.

Status Code	Path	Type	Entry Type
401	%SystemDrive%\inetpub\cu...	File	Inherited
403	%SystemDrive%\inetpub\cu...	File	Inherited
404	%SystemDrive%\inetpub\cu...	File	Inherited
405	%SystemDrive%\inetpub\cu...	File	Inherited
406	%SystemDrive%\inetpub\cu...	File	Inherited
412	%SystemDrive%\inetpub\cu...	File	Inherited
500	%SystemDrive%\inetpub\cu...	File	Inherited
501	%SystemDrive%\inetpub\cu...	File	Inherited
502	%SystemDrive%\inetpub\cu...	File	Inherited

## Windows Authentication

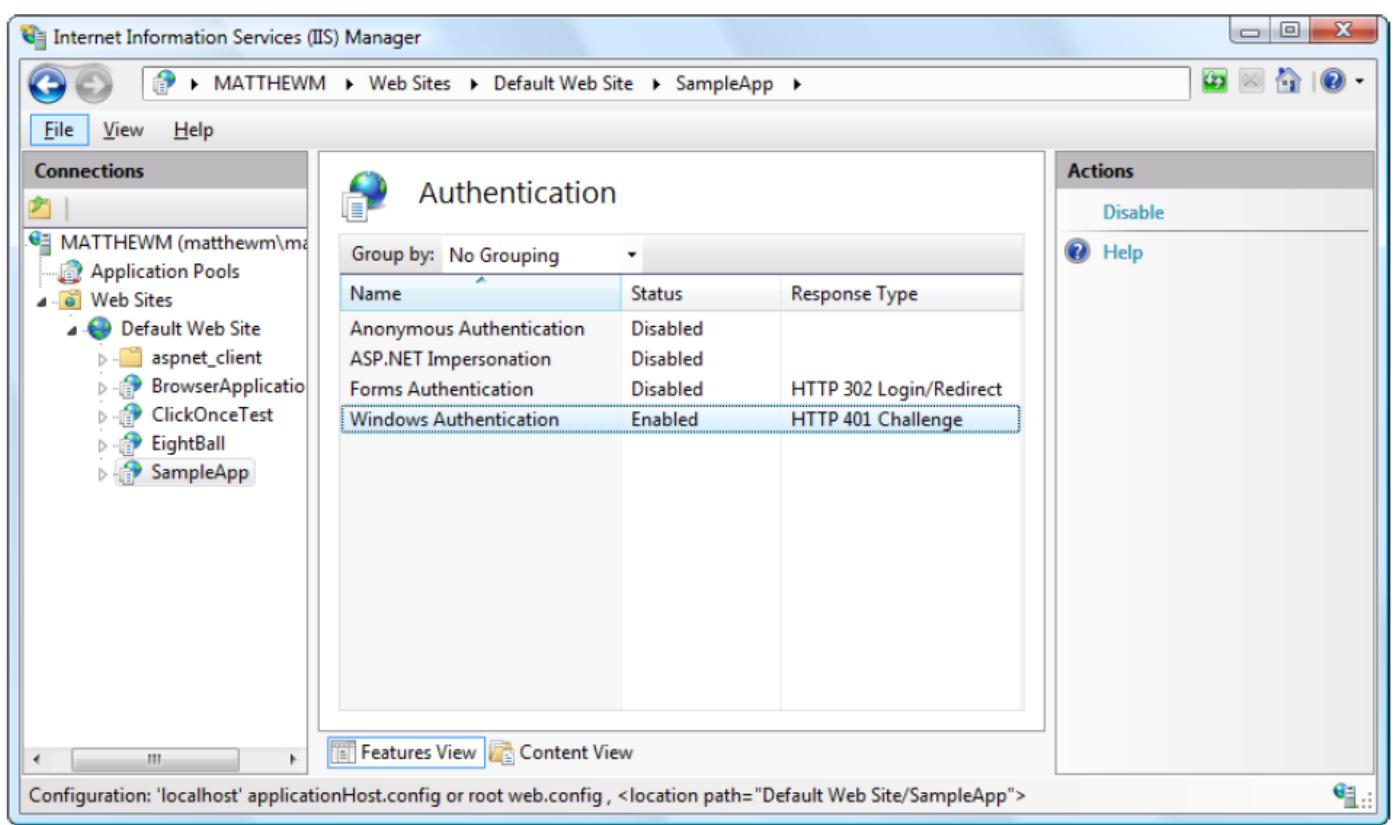
در فصل های قبلی روش های مختلفی برای محدود کردن کاربران ناشناس از دسترسی به وب سایت آموختید. اولین راه، استفاده از ASP.NET built-in forms می باشد که یک فرم login برای تعیین اعتبار کاربر به همراه یک کوکی اعتبارسنجی برای نگهداری مشخصات حساب کاربری تعریف شده روی وب سرور مشخص می کند.

با استفاده از VS، تست یک وب سایت که از windows authentication استفاده می کند کار ساده ای می باشد. اما در یک وب سرور واقعی، باید IIS را پیکربندی کنید.

### روش های windows authentication



روش	توضیحات
Anonymous	اعتبار سنجی anonymous، یک اعتبارسنجی صحیح نمی باشد زیرا کلاینت ها نیاز به ارسال اطلاعات نمی باشند. در عوض کاربران دسترسی آزادی به وب سایت از طریق حساب کاربری خاصی دارند.
Basic	بخشی از استاندارد HTTP می باشد و همه مرورگرها و وب سرورها از آن پشتیبانی می کنند. مرورگر به کاربر یک کادر login با کلمه عبور و کلمه کاربری نمایش می دهد. این اطلاعات سپس به IIS ارسال می شود. عیب این روش این است که کلمه عبور بدون رمزگشایی ارسال می شود و روی اینترنت در دسترس است (مگر آنکه آن را با تکنولوژی SSL ترکیب کنید)
Digest	یک روش ضعیفتر از basic می باشد. به جای کلمه عبور، digest را (یا همان hash) ارسال می کند. یکی از معایب آن، نیاز به استفاده از Active Directory یا دسترسی به server دارد.
Integrated	بهترین گزینه برای سناریوی اینترنت می باشد. در این روش IE، اطلاعاتی که بصورت خودکار توسط login، windows account شده به سیستم استفاده خواهد شد، ارسال می شود. این نوع از اعتبارسنجی روی proxy server ها کار نمی کند.



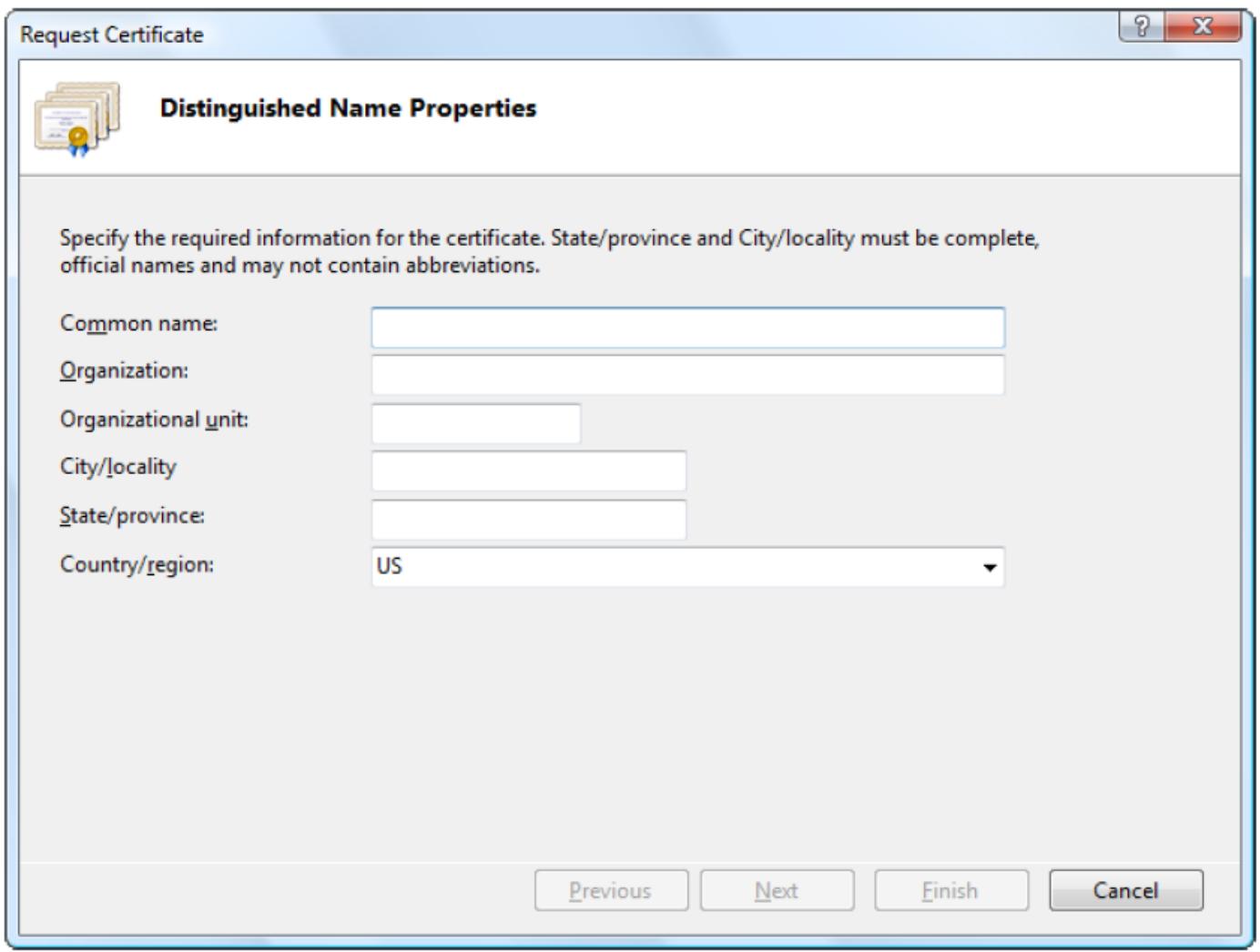
## محرومانه بودن با استفاده از گواهینامه ها و SSL

یک گواهینامه به شما اجازه می‌دهد تا تعیین کنید که سایت و اطلاعات سازمان شما ثبت شده است و توسط certificate authority تعیین اعتبار شده است. این گواهینامه‌ها اندکی شبیه گواهینامه رانندگی می‌باشند که نشان می‌دهند شما توسط سازمانی مورد تائید هستید. وب سرور شما نیز نیاز به یک گواهینامه برای استفاده از SSL دارد، که خودکار کلیه اطلاعات ارسالی بین کلاینت و سرور را کد می‌کند.

برای اضافه کردن یک گواهینامه به سایت، ابتدا باید یک certificate authority کنید که انواع آن در زیر آمده است :

- VeriSign (<http://www.verisign.com>)
- GeoTrust (<http://www.geotrust.com>)
- GlobalSign (<http://www.globalsign.com>)
- Thawte (<http://www.thawte.com>)

برای ایجاد درخواست گواهینامه، ارسال یک certificate request email برای وب سرور شما می‌باشد. IIS Manager، اجازه ایجاد یک (Certificate Signing request)CSR را بصورت خودکار به شما می‌دهد. برای انجام این کار، کامپیوتر خود را از درخت Actions pane، Create Certificate Request روی آیکن Server Certificate دوبار کلیک کنید. سپس روی کلیک کنید.



در انتهای این پردازش، شما یک کلید درخواست ایجاد کرده اید. می‌توانید فایل ایجاد شده را به عنوان فایل text ذخیره کنید اما باید در نهایت آن را به یک certificate authority email کنید.

## بکارگیری SSL

تکنولوژی SSL ارتباطات بین یک کلاینت و یک وب سایت را encrypt می‌کند. معمولاً زمانی که نیاز به انتقال اطلاعات حساس و محترمانه بین یک کاربر تعیین اعتبار شده و برنامه تحت وب می‌باشد، از این روش استفاده خواهد شد. بدون SSL، هر اطلاعاتی که در اینترنت ارسال خواهد شد، شامل کلمه های عبور، شماره کارت اعتباری و ...، به سادگی قابل نمایش توسط تجهیزات شبکه ای می‌باشد.

حتی بهترین encryption ها نیز نمی‌توانند امنیت مورد نیاز را تعیین کنند، زیرا یک وب سرور آلوهه می‌تواند در انتهای رابطه، کلیه اطلاعات را decrypt کند. برای جلوگیری از این نوع خطرات، SSL از گواهینامه ها استفاده می‌کند.

برای استفاده از SSL، باید یک گواهینامه معتبر نصب کنید. سپس می‌توانید تنظیمات پوشش IIS را که پوشش های جداگانه ای برای یک مشخص می‌کند را تعیین نمایید. برای این کار، وب سایت را در IIS Manager، روی SSL Settings انتخاب، دوبار کلیک کرده و گزینه Require SSL را انتخاب نمایید.

برای دسترسی به یک صفحه از طریق SSL، کلاینت URL را با https تایپ می‌کند. در کد ASP.NET نیز می‌توانید چک کنید که آیا یک کاربر از طریق یک اتصال امن در حال اتصال است یا نه :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.IsSecureConnection)
    {
        lblStatus.Text = "This page is running under SSL.";
    }
    else
    {
        lblStatus.Text = "This page isn't secure. <br />";
        lblStatus.Text += "Please request it with the ";
        lblStatus.Text += "prefix https:// instead of http://";
    }
}
```

## نحوه کار : SSL

- ۱- کلاینتی اتصال SSL را درخواست می‌کند.
- ۲- سرور گواهینامه دیجیتالی آن را امضا و به کلاینت ارسال می‌کند.
- ۳- کلاینت، گواهینامه ای که توسط certificate authority ارسال شده است را تعیین اعتبار کرده و می‌بیند که منقضی شده است یا نه. اگر گواهینامه معتبر بود، کلاینت به مرحله بعد می‌رود.
- ۴- کلاینت به سرور می‌گوید که سرور چه کلید اعتبارسنجی را پشتیبانی می‌کند.
- ۵- سرور قوی ترین طول کلید را توسط کلاینت و سرور پشتیبانی می‌شود، انتخاب کرده و سپس سایز آن را به کلاینت خبر می‌دهد.
- ۶- کلاینت یک session key (رشته تصادفی از بایت‌ها) را ایجاد می‌کند و آن را توسط public key (که توسط گواهینامه دیجیتالی سرور ایجاد شده است)، encrypt می‌کند و آن را به سرور می‌فرستد.
- ۷- سرور این session key را با استفاده از private key decrypt کرده تا سرور و کلاینت دارای یک session key تصادفی یکسان باشند که برای encrypt کردن ارتباط بین خود استفاده می‌کنند.

## Deploy ساده یک سایت کردن

۱- یک VD، روی وب سرور ایجاد کنید.

۲- کل سایت را درون آن VD کپی کنید.

معمولًا برای انتقال فایل‌ها نیاز به برنامه FTP دارید. اگر وب سرور و کامپیوتر شما در یک شبکه داخلی باشند، تنها کافی است از windows explorer برای این کار استفاده کنید.

قبل از انتقال فایل‌ها، باید مطمئن شوید که مد Debug در نسخه Deploy شده غیر فعال است. برای این کار، خصیصه debug را در تگ compilation، پیدا و آن را برابر با false قرار دهید.

```
<configuration>
```

```
<system.web>
```

```
  <compilation debug = "false" targetFramework = "4.5">
```

```
  </compilation>
```

```
  <!-- Other settings omitted. -->
```

```
</system.web>
```

```
<configuration><?xml version="1.0"?>
```

اگر debugging فعال باشد، کد کامپایل شده صفحه ASP.NET، بزرگ‌تر شده و آهسته تر کار می‌کند. به همین دلیل باید فقط در تست و توسعه نرم افزار، از مد debugging استفاده کرد.

### کامپایل کد

بصورت پیش فرض، زمانیکه یک برنامه را deploy می‌کنید، فایل‌های منبع کامپایل نشده را deploy می‌کند. اولین باری که صفحه درخواست می‌شود، بصورت خودکار کامپایل شده و درون یک پوشه موقت برای استفاده مجدد cache می‌شود.

### معایب این روش :

- اولین درخواست صفحه آهسته می‌باشد. البته پس از اینکه صفحه بیش از یکبار درخواست شد، این مشکل رفع می‌شود.
- وب سرور شامل کلیه کدهای شما خواهد بود که در دسترس هر شخصی که دسترسی به سرور دارد قرار خواهد گرفت. اگرچه کاربران نمی‌توانند کدهای شما را ببینند اما مدیران می‌توانند آنها را دیده و تغییر دهند.

برای بهبود کارایی و عدم دسترسی سایر افراد به کدهای شما، می‌توانید از امکانی با نام ASP.NET PreCompilation استفاده نمایید. تا قبل از deploy کردن برنامه، کل وب سایت شما به فایل‌های باینری تبدیل شود.

این کار را هم می‌توانید از طریق command-line انجام دهید :

```
aspnet_compiler -m metabasePath targetDirectory
```

که البته این روش در اینجا توضیح داده نخواهد شد و هم از طریق VS :

۱- می‌توانید در هنگام ایجاد یک پروژه جدید، یک VD، ایجاد کنید.

۲- می‌توانید از امکان Copy Web Site برای انتقال یک وب سایت موجود به یک VD در محل دیگری اقدام کنید.

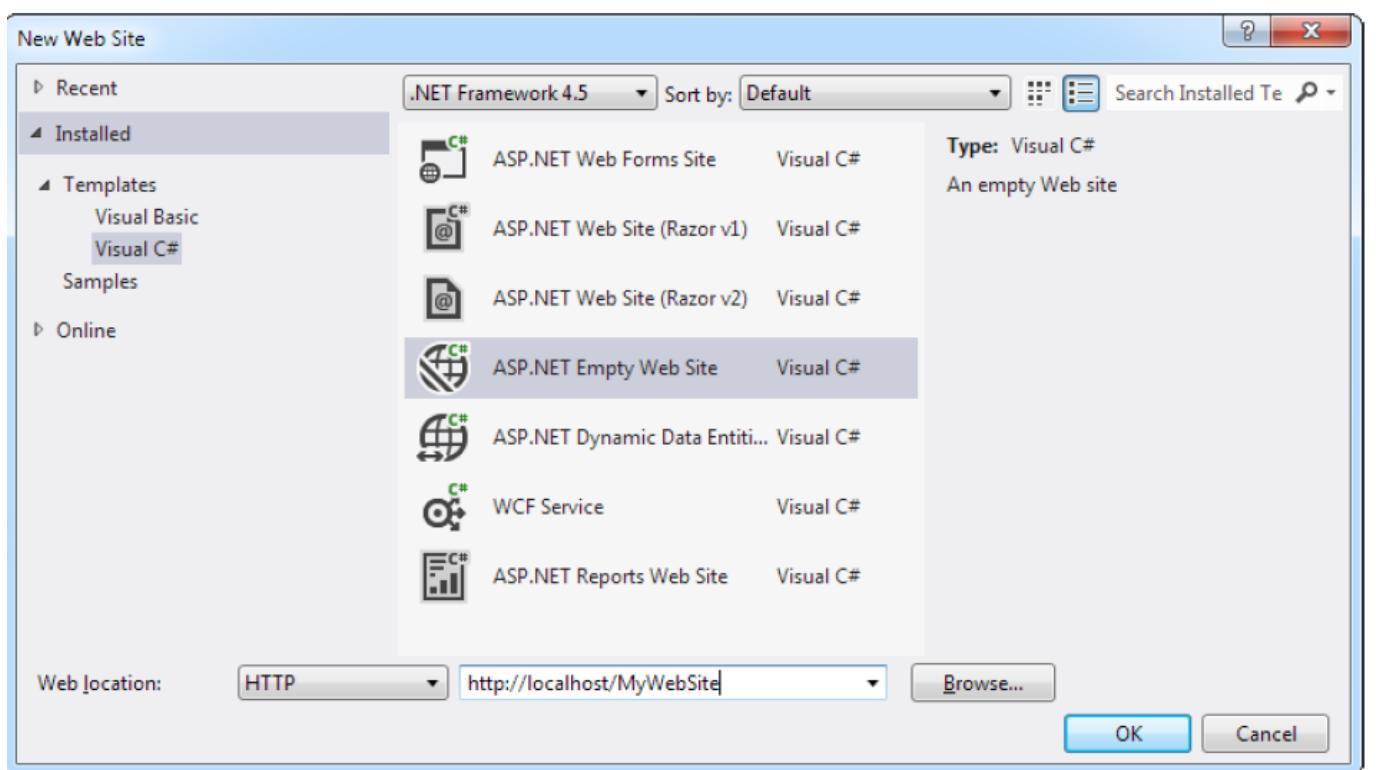
۳- اگر از website (و نه از web project) استفاده می‌کنید، می‌توانید از امکان web package برای بسته بندی کردن تنظیمات IIS، گواهینامه های امنیتی و SQL script برای برنامه خود استفاده نمایید.

### ایجاد یک Virtual Directory برای یک پروژه جدید

زمانی که یک وب سایت در VS ایجاد کردید، می‌توانید همزمان، یک VD، نیز برای آن ایجاد نمایید. بنابراین VS، از استفاده نخواهد کرد. در عوض، همه درخواست‌های شما از طریق نسخه کامل IIS انجام می‌شود.

برای انجام این کار، ابتدا VS را با حالت as an administrator باز کنید. (روی کلید میانبر VS کلیک راست کرده و Run as Administrator را اجرا کنید.)

سپس File > New Web Site را از منو انتخاب کنید. در پنجره باز شده، گزینه HTTP را به جای File System برای انتخاب محل استفاده کنید. یک آدرس وارد کنید (http://localhost/MyWebSite) در این کامپیوتر در مسیر c:\inetpub\wwwroot ایجاد خواهد کرد.



این روش معمولاً بهترین راه برای ایجاد VD، نمی باشد، زیرا دارای محدودیت های زیر است :

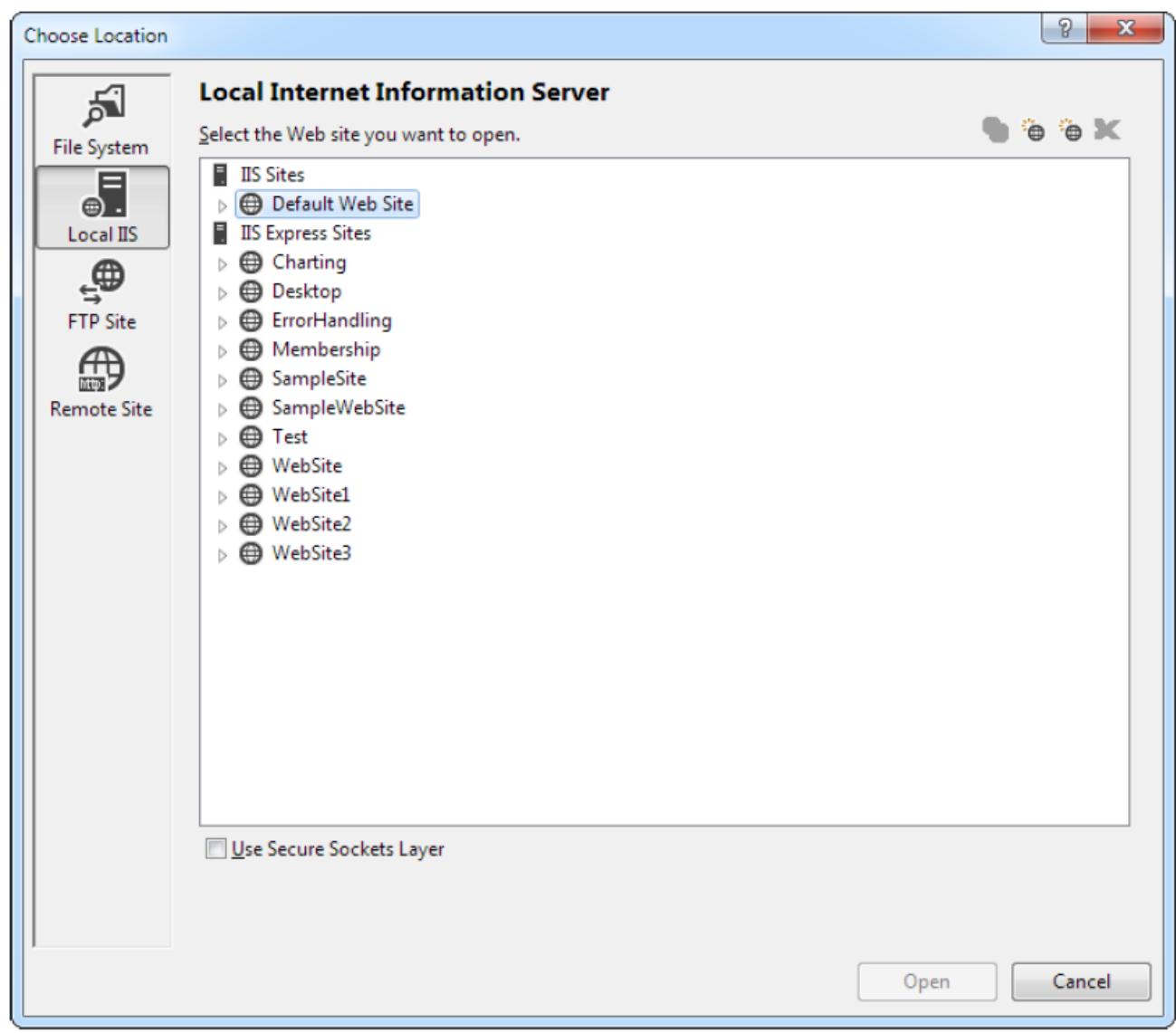
- شما را مجبور به ایجاد VD، در زمام ایجاد برنامه می کند.
- نمی توانید پیکربندی به جز این تنظیمات (مانند صفحه پیش فرض، خطاهای سفارشی و مجوزهای VD) در نظر بگیرید.

بنابراین باید ابتدا برنامه را در VS ایجاد و سپس یک VD بصورت دستی در هنگام deploy کردن آن روی وب سرور تست یا نهایی، ایجاد کنید.

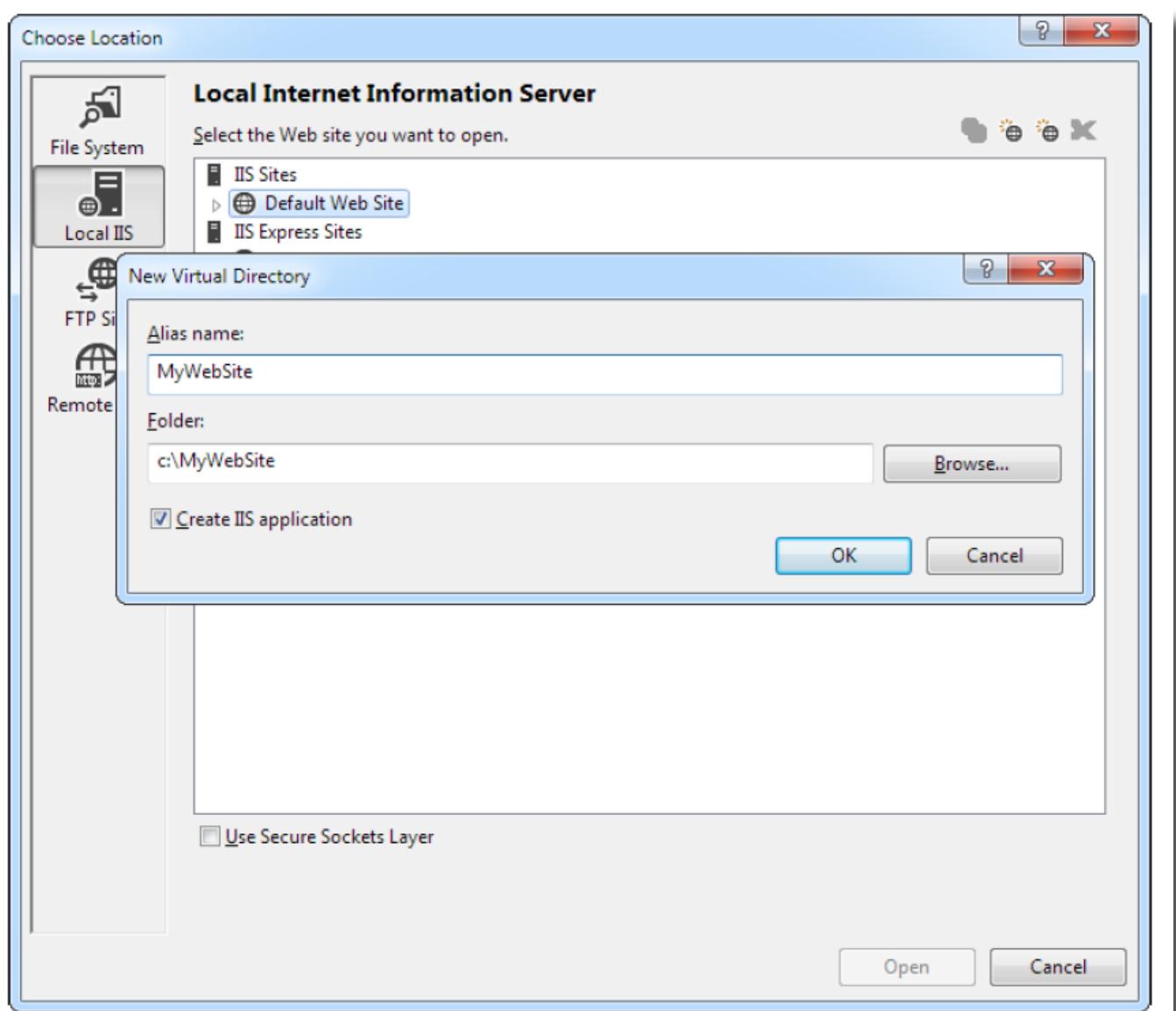
به شما گزینه های کاملی از IIS Manager ارائه نمی دهد . در کادر New Web Site، عبارت

<http://localhost>

را تایپ و سپس دکمه browse را کلیک کنید. لیست VD هایی که در IIS تعريف شده است را مشاهده خواهید کرد.

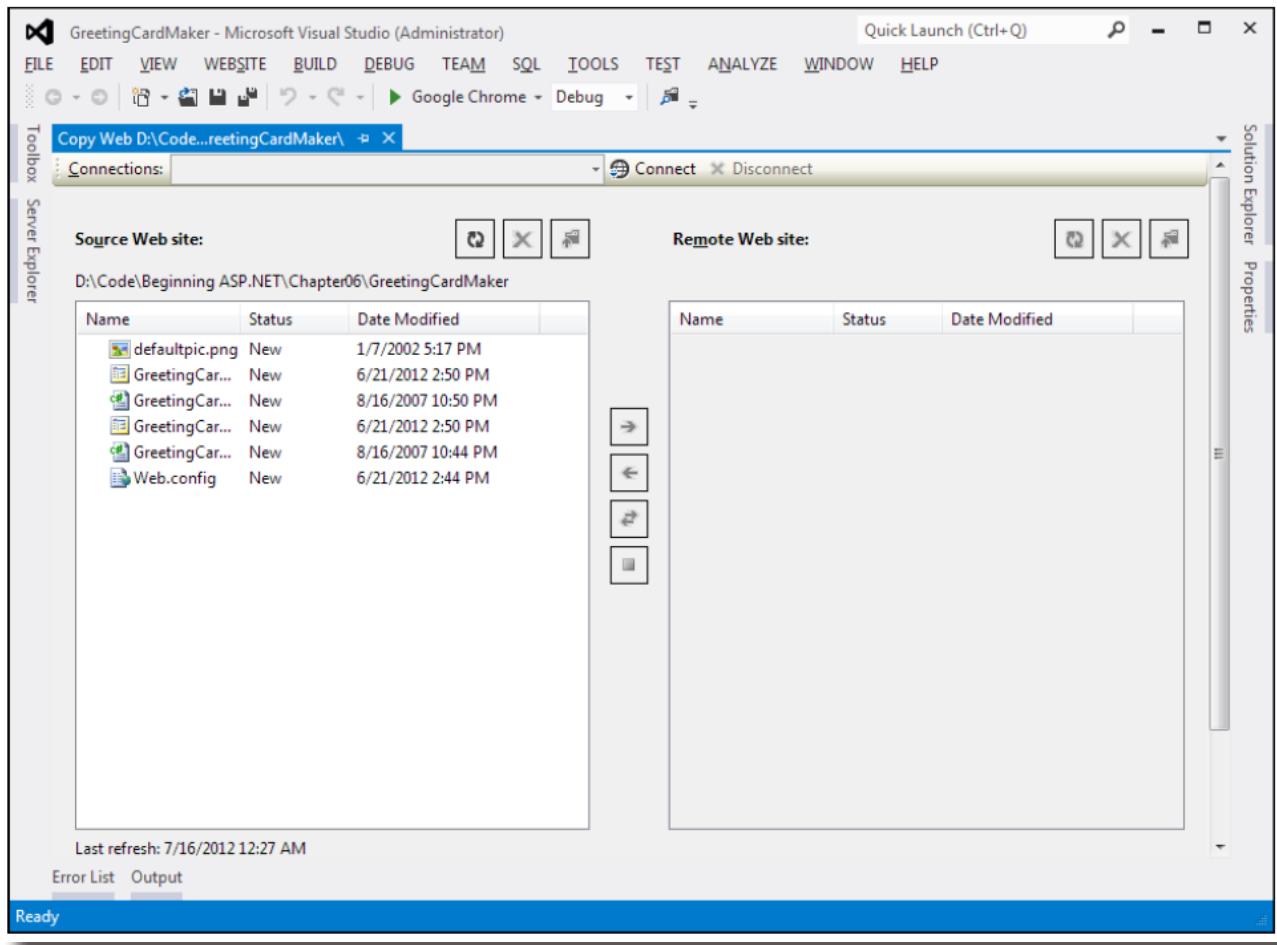


نمی توانید Property ها را تغییر یا نمایش دهید اما می توانید یک VD، موجود را انتخاب کنید یا یک VD جدید ایجاد نمایید.



## کپی کردن یک وب سایت

VS، همچنین شامل یک راه ساده و سریع برای انتقال برنامه وب شما بدون استفاده از یک برنامه جداگانه می باشد. گزینه WebSite>Copy Web Site را انتخاب کنید. این کار یک کادر جدید باز می کند.



این پنجره شامل دو لیست فایل می باشد. در سمت چپ فایل های موجود در پروژه کنونی (local hard drive) و در سمت راست فایل های موجود در مکان مقصد را مشخص می کند. زمانی که روی Connect کلیک می کنید، VS، یک کادر جدید نمایش می دهد که می توانید با استفاده از آن یکی از انواع مکان ها را مشخص کنید:

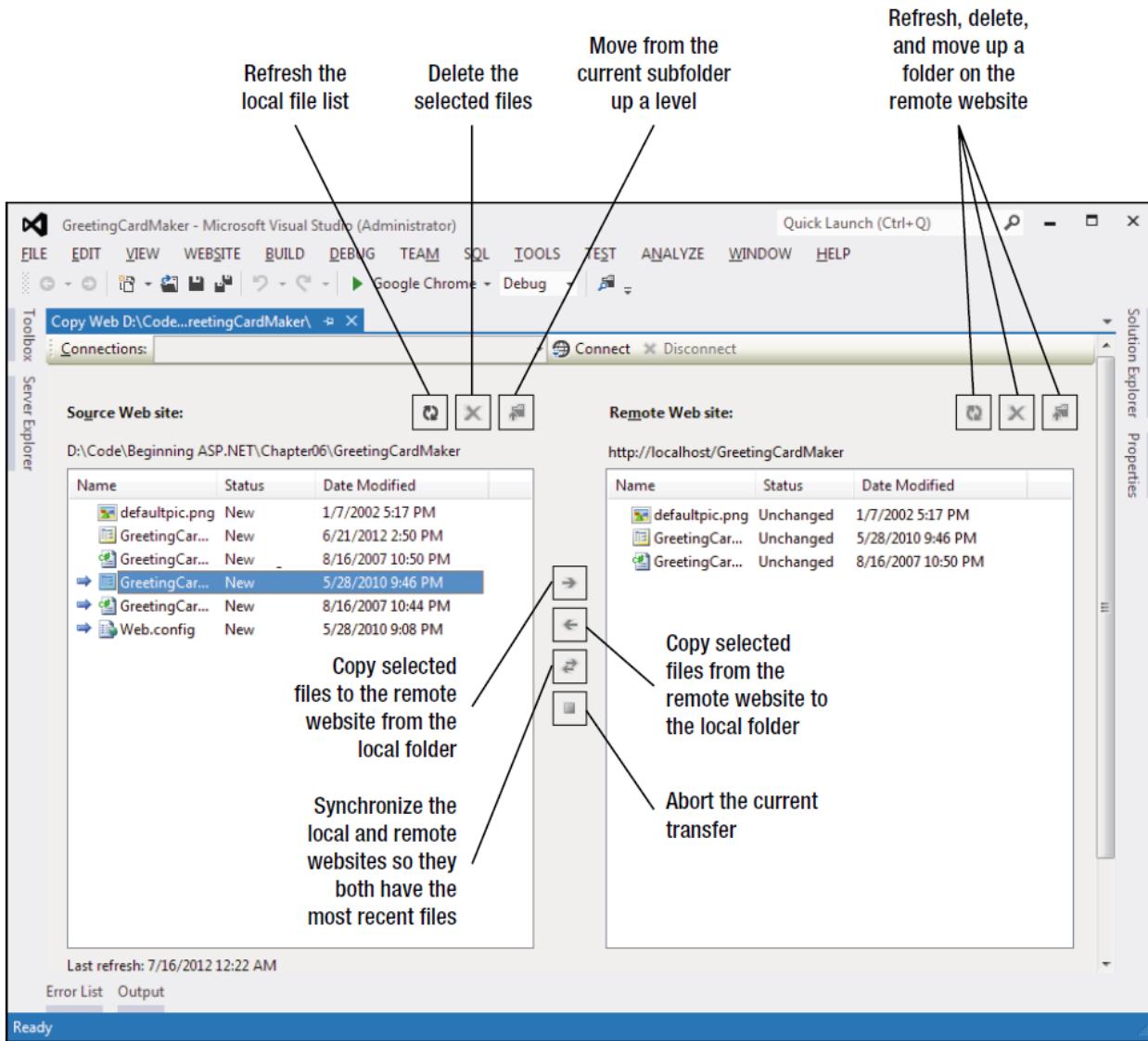
**File System**: این ساده ترین انتخاب می باشد. به راحتی می توانید در بین پوشه ها و درایو ها روی کامپیوتر خود یا در پوشه های share روی کامپیوترهای موجود در شبکه حرکت کنید.

**Local IIS**: این گزینه به شما امکان دسترسی به VD هایی که در کامپیوتر local IIS ایجاد شده است را می دهد. برای ایجاد یک VD جدید برای برنامه وب شما، روی آیکن Create New Virtual Directory یا Create New Application کلیک کنید.

**FTP Site**: این گزینه روش کاملاً مناسبی برای نمایش یک پوشه نمی باشد. در عوض، باید تمامی اطلاعات اتصال شامل سایت FTP پورت، پوشه و کلمه کاربری و عبور را قبل از برقراری ارتباط وارد نمایید.

:این گزینه از طریق یک URL مشخص با استفاده از HTTP به وب سایت دسترسی دارد. برای اینکه این گزینه کار کند، باید FrontPage Extension روی وب سرور نصب شده باشد. در زمان اتصال، کلمه کاربری و عبور را از شما می خواهد.

**روش Copy Web Site**, برای بروز رسانی یک وب سرور مناسب است. زیرا VS، لیست فایل‌ها روی local و remote را مقایسه کرده و فایل‌هایی که تنها در یک مکان موجود است را علامت گذاری می‌کند. (با وضعیت NEW و Changed). فایل مورد نظر را انتخاب و روی دکمه جهت نما برای انتقال از یک مکان به مکان دیگر کلیک کنید.

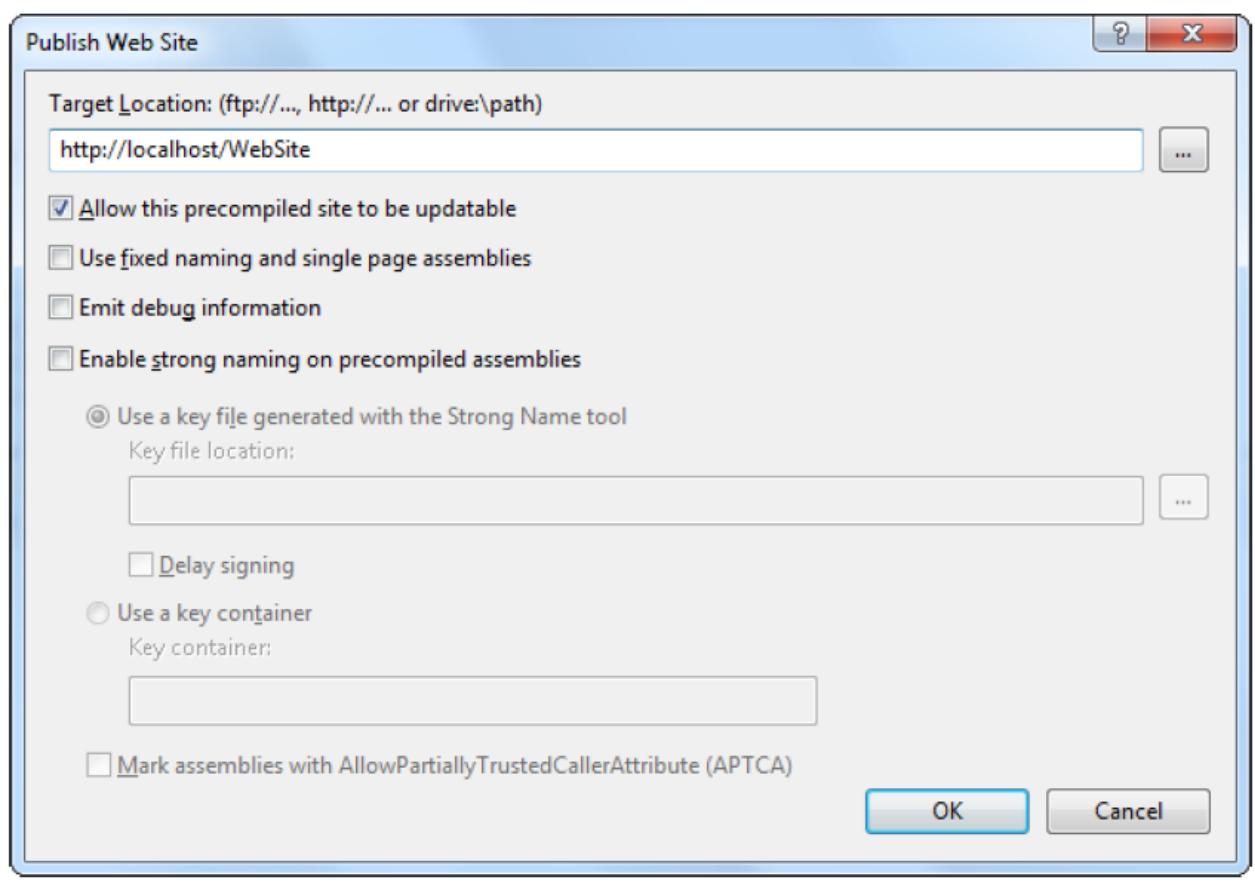


## Publish کردن وب سایت

روش کپی کردن وب سایت که در بالا آموختید مناسب انتقال فایل به یک سرور نست می‌باشد. اما امکان PreCompile کردن کدها را به شما نمی‌دهد. اگر برنامه خود را روی یک وب سرور حقیقی deploy کنید و بخواهید کدهای خود را محافظت کنید باید از روش دیگری استفاده کنید.

می‌توانید از ابزار (Coomand-Line) aspnet\_compiler برای کامپایل کدهای ASP.NET استفاده کنید. البته می‌توانید از طریق VS، به این ابزار دسترسی داشته باشید.

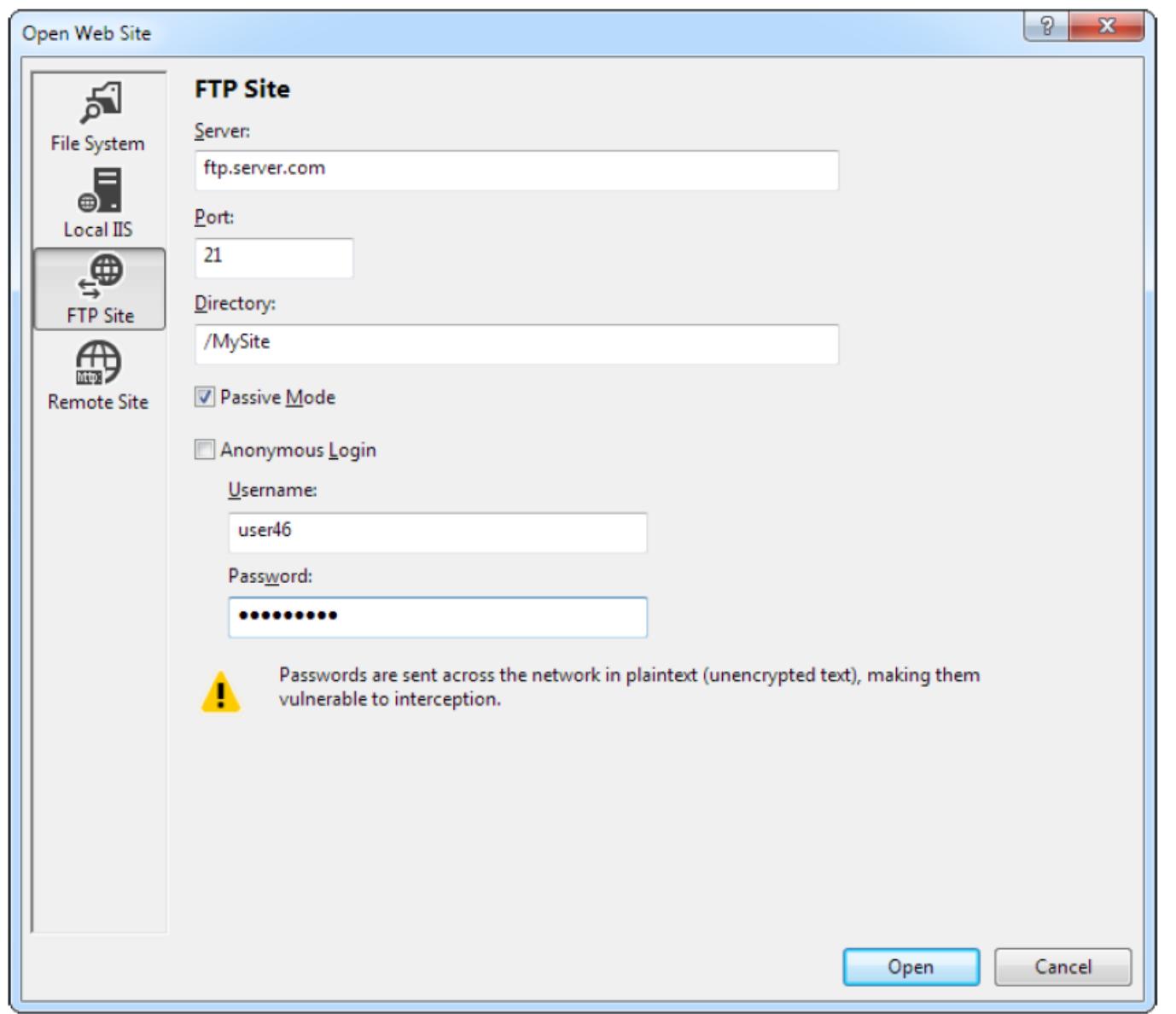
Build>Publish Web Site -۱



۲- مسیر فایل URL یک سایت FTP یا یک سایت که FrontPage روی آن فعال می‌باشد را در کادر Location را وارد کنید.

۳- در کادر باز شده، می‌توانید اجازه update کدهای code-behind در حالتی که کدهای HTML و فایل aspx، کامپایل نشده است، را بدهید.

۴- روی OK کلیک کنید. فایل‌های شما توسط aspnet\_compiler کامپایل شده و به مکان مقصد منتقل می‌شود.

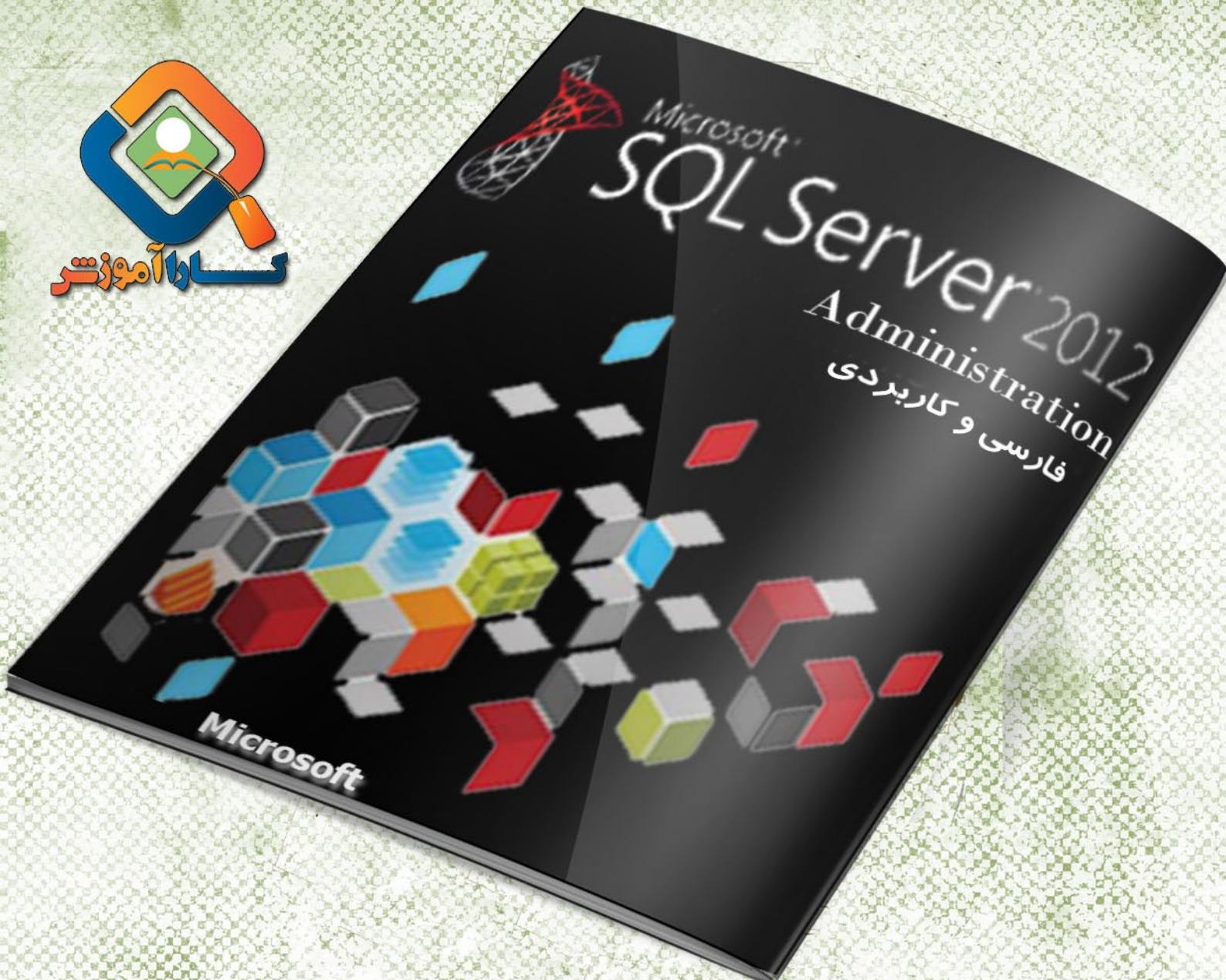




ویندوز ۸ (به انگلیسی: Windows 8) آخرین نسخه سیستم عامل ویندوز شرکت مایکروسافت می‌باشد که در ۲۶ اکتبر ۲۰۱۲ به بازار عرضه شد. ویندوز ۸ برای استفاده در رایانه‌های شخصی، رایانه‌های همراه و تبلتها تولید شده است. این سیستم عامل هشتمین نسل از سیستم عامل‌های ویندوز می‌باشد و به همین دلیل نام آن را ویندوز ۸ گذاشته‌اند.

### برخی از ویژگی‌های جدید ویندوز ۸

رابط کاربری جدید (Metro) برای صفحه نمایش لمسی - حذف دکمه استارت - مرورگر اینترنت اکسپلورر ۱۰ - ورود کاربری با استفاده از تصویر - بوت سریع هیبریدی - Task Manager (برنامه مدیریت وظیفه) جدید - راه اندازی ویندوز با استفاده از فلاش مموری - پشتیبانی از چندین نمایشگر - پشتیبانی از USB ۳ - صفحه شروع جدید - قابلیت بازگردانی سیستم عامل به تنظیمات کارخانه - سیستم پرونده جدید - پشتیبانی از ارتباط حوزه نزدیک (NFC) - بوت امن - پشتیبانی از فناوری RAID - اضافه شدن نوار ریبون به ویندوز اکسپلورر مانند مایکروسافت آفیس ۲۰۱۰ - توانایی بازکردن پرونده‌های ISO، IMG و VHD - نرم افزار ویندوز دیفندر در این ویندوز دارای قابلیت آنتی ویروس است، که مشابه نرم افزار مایکروسافت سکیوریتی عمل می‌کند - اضافه شدن ویژگی ایمنی خانواده (Family safety) و ...



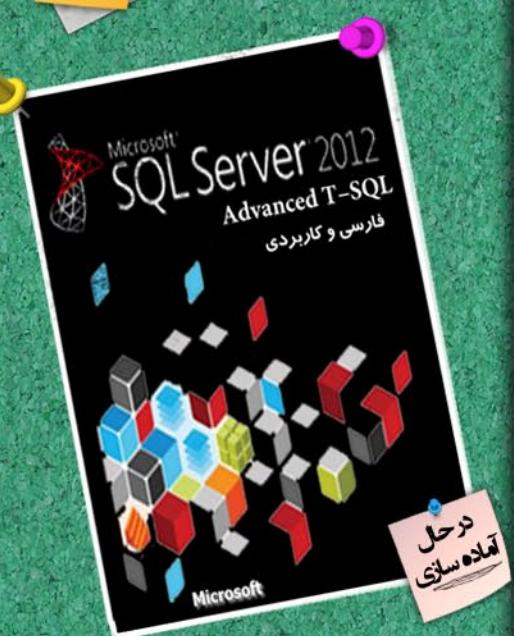
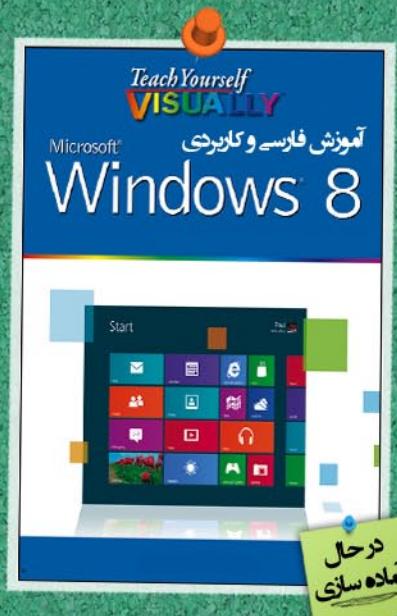
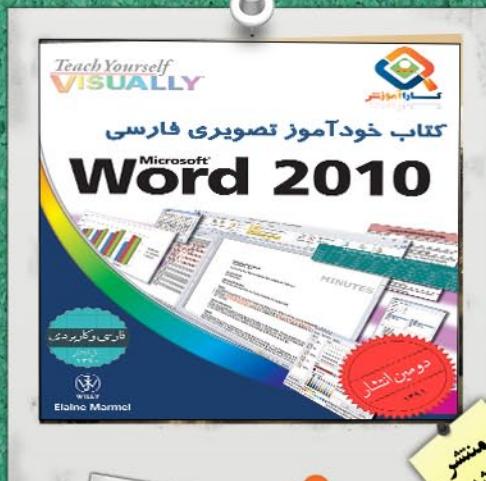
به دلیل آنکه در کشور ما در حوزه نگهداری از داده‌ها آنگونه که باید مدیران پایگاه داده متخصص تربیت نیافته اند و نیاز به آموختن صحیح و کاربردی این موضوع، امری ضروری به نظر می‌رسد، گروه کارآموزش به منظور ایجاد و ارتقا سطح دانش تخصصی در حوزه مدیریت پایگاه داده، در حال آماده سازی کتاب تخصصی مدیریت پایگاه داده بر روی آخرین و جدیدترین نسخه از نرم افزار Microsoft Sql Server نسخه ۲۰۱۲ می‌باشد.

مدیر پایگاه داده یکی از تخصصی‌ترین مشاغل در حوزه کامپیوتر می‌باشد که علاوه بر جذابیت‌های بسیار و حساسیت‌های فراوان، درآمد بالای شغلی را نیز برای فرد متخصص خود به همراه می‌آورد.

مدیر پایگاه داده یا به اختصار DBA چه کسی است؟

فردي که مسئول نصب، پیکربندی، ارتقا، مدیریت، نظارت و نگهداری از پایگاه داده در یک سازمان می‌باشد. این نقش شامل توسعه و طراحی استراتژی‌های پایگاه داده، نظارت و بهبود عملکرد و ظرفیت پایگاه داده و برنامه ریزی برای توسعه ملزمات در آینده می‌باشد. آن‌ها همچنین ممکن است برنامه ریزی، همکاری هماهنگی و پیاده سازی اقدامات امنیتی لازم برای محافظت از پایگاه داده را به عهده داشته باشند.

# انتشارات مجازی گروه کارا آموزش





هیچ کس نباید به دلیل مشکلات مالی، آموزش فود را به تغییر بیندازد.  
ما نمیگذاریم.



کارآموزتر