بسم الله الرحمن الرحیم

مقالات

ASP.net

منبع:

www.Dev.ir

Namespace چیست؟

یک نکته مهم که در زمان استفاده از .NET Framework باید به آن توجه داشت آن است که فضانام (namespace یا نامکده) ها در ساختمان برنامه کاربردی قرار دارند. فضانام یک طرح نامگذاری منطقی برای گروه بندی کلاس های مرتبط است. این طرح مانع از آن می شود تا کلاس هایی که برای متدها و خصوصیات از یک شناسه یکسان استفاده می کنند تداخل داشته باشند.

مثلا .NET Framework برای گروه بندی تایپ ها به مقوله های منطقی عملکرد، از قبیل چارچوب برنامه کاربردی ASP.NET، از یک طرح نامگذاری سلسله مراتبی استفاده می کند. ابزارهای طراحی از فضانام ها با هدف تسهیل مرور و ارجاع تایپ ها در برنامه بهره برداری می کنند. مثلا فرض کنید در حال نوشتن کد زیر هستید:

```
Public Class NewClass
  [Procedures and Functions]
End Class

Public Class NewClass
  [Procedures and Functions]
End Class
```

این کد به خطا منجر می شود چون کامپایلر راهی برای تشخیص کلاس ها از یکدیگر ندارد. برای غلبه بر این مشکل می توان از یک فضانام استفاده کرد که اجازه می دهد دو کلاس همنام در صفحه با هم وجود داشته باشند. قطعه برنامه زیر تعریف این دو کلاس در فضانام های منحصر بفرد را نشان می دهد:

```
Namespace One
  Public Class NewClass
    [Procedures and Functions]
  End Class
End Namespace

Namespace Two
  Public Class NewClass
    [Procedures and Functions]
  End Class
End Namespace
```

در این کد برخوردی بین دو کلاس با نام NewClass وجود ندارد چون هر کدام در یک فضانام جداگانه قرار داده شده است. کلاس اول را می توان با استفاده از ترکیب One.NewClass صدا زد، حال آنکه کلاس دوم را می توان با استفاده از ترکیب Two.NewClass صدا زد.

شما می توانید در فضانام های خود از یک ساختار سلسله مراتبی استفاده کنید. قرار دادن اشیاء مشابه تحت زیرعنوانها در یک فضانام مشترک تشخیص هدف فضانام را آسانتر می کند، و در عین حال باعث می شود برنامه به مراتب شئ گراتر شود.

برای توضیح فضانام می توان ساختار فایل و دایرکتوری (کشو، فولدر) در یک کامپیوتر را در نظر گرفت. در این مثال کلاس ها به مثابه فایل ها و فضانام ها مانند دایرکتوری ها هستند. بدیهی

است همانگونه که می توانیم دایرکتوریهای تو در تو داشته باشیم، فضانام ها هم در حالیکه کلاس ها را در خود جای داده اند می توانند بصورت تو در تو باشند.

فضانام ها در ساخت برنامه های کاربردی ASP.NET نقش مهمی ایفا می کنند. خوشبختانه لازم نیست برای همه اشیایی که می توانند به وسیله صفحات ASP.NET مورد استفاده قرار بگیرند سیستم طبقه بندی فضانام ایجاد کنید. مایکروسافت این مساله را برای شما حل کرده است. دو فضانام ریشه، و فضانامهای فرزند آنها را می توان وارد صفحات ASP.NET خود کرد. اولی System نامیده می شود، و دومی Microsoft نام دارد. این فضانامها با جزئیات بیشتر در ادامه مورد بحث قرار گرفته اند.

فضانام System

فضانام System فضانام اصلی برای ساخت ASP.NET و همه برنامه های کاربردی دیگر مبتنی بر .NET Framework است. هر چیزی که در برنامه کاربردی شما قابل انجام باشد از طریق فضانام System کنترل می شود. به عنوان مثال کنترل آرایه، عملیات ریاضی، و تبدیل نوع داده ها از طریق فضانام System و فضانامهای فرزند آن اداره می شوند. ۹ فضانام پیش فرض (فضانام System و ۸ فرزند آن) وجود دارند که به صورت خودکار به صفحات ASP.NET اضافه می شوند:

- System
- System.ComponentModel.Design
- System.Data
- System.Drawing
- System.Web.SessionState
- System.Web
- System.Web.UI
- System.Web.UI.WebControls
- System.Web.UI.HTMLControls

هشت فضانام (بجز فضانام System) در زمان ساخت Visual Studio.NET یا VS.NET بطور خودکار به صفحات ASP.NET وارد می شوند. این فضانام ها در زیر به اختصار شرح داده شده اند.

• System.ComponentModel.Design: دربرگیرنده کلاس هایی است که می توان از آنها برای طراحی پشتیبانی سفارشی اجزا و زمان طراحی و دسترسی به سرویس های تامین شده توسط معماری .NET Framework استفاده کرد.

• System.Data: امکان دسترسی به کلاس ها و رابطهایی را فراهم می کند که معماری ADO.NET را برای دسترسی به داده های عمومی تشکیل می دهند.

• System.Drawing: دربرگیرنده کلاس ها و رابطهایی است که عملکرد گرافیکی اولیه را تامین می کنند. فضانام System.Drawing نیز از طریق فضانام System.Drawing.Drawing2D و System.Drawing.Imaging عملکرد پیشرفته تری فراهم می کند.

• System.Web: کلاس ها و رابطهایی تامین می کند که ارتباط مرورگر/سرویس دهنده را امکان پذیر می کنند. این فضانام دربرگیرنده کلاس HTTPRequest (فراهم کننده اطلاعات وسیعی درباره درخواست HTTP جاری)، کلاس HTTPResponse (فراهم آورنده امکان دسترسی به فرآیندها و یوتیلیتی های سمت سرویس دهنده) است.

• System.Web.SessionState: فراهم کننده کلاس ها و متدهایی برای مدیریت وضعیت جلسات کاری می باشد.

• System.Web.UI: فراهم کننده کلاس ها و رابطهایی برای رابط واسط کاربر برنامه کاربردی ASP.NET است که موجب می شوند برنامه کاربردی با سطوح مختلف صفحه، ارتباط برقرار کند. کلاس اصلی این فضانام، کلاس Page می باشد که دربرگیرنده همه خصوصیتها، متدها، و سازنده های صفحه است. اشیاء اصلی Active Server Page زیر خصوصیتهایی در کلاس Page هستند: Application، Response، Request،Server و Session.

• System.Web.UI.HTMLControls: کلاس هایی برای عناصر HTML استاندارد، شامل فرم ها، کنترل های ورودی، آنکور، جداول، قسمتهای متنی، و غیره فراهم می کند. این کنترلها همانند تگهای عادی HTML هستند با این تفاوت که داری دو صفت runat="server" و id="controlname" می باشند.

• System.Web.UI.HTMLControls: برای کنترلهای سرویس دهنده ای که شبیه کنترلهای HTML هستند ولی انعطاف پذیری بیشتر و عملکرد پیچیده تری دارند کلاس هایی را تامین می کند.

برخی فضانام های مهم و پرکاربرد دیگر به شرح زیر می باشند.

• System.IO: دربرگیرنده رابطها و کلاس هایی است که امکان خواندن و نوشتن همگام و غیرهمگام فایل ها و جریانهای داده را فراهم می کنند.

• System.Data.OleDb: امکان دسترسی به کلاس ها و رابطهای مخصوص دسترسی به یک منبع داده از طریق ADO را فراهم می کند.

• System.Data.SqlClient: امکان دسترسی به کلاس ها و رابطهای مخصوص دسترسی به داده های خاص Microsoft SQL Server از طریق ADO را فراهم می کند.

• System.Web.Security: امکان دسترسی به کلاسها و رابطهای مخصوص امنیت برنامه کاربردی ASP.NET را فراهم می کند. دستیابی به رمزنگاری، مجوزها، و تنظیمات خط مشی برنامه کاربردی در این فضانام قرار می گیرند.

• System.XML: امکان دسترسی به کلاسها و رابطهای مخصوص پردازش اسناد XML را فراهم می کند.

فضانام Microsoft

علاوه بر فضانام System که در چارچوب NET. یافت می شود، مایکروسافت چند فضانام اضافه کرده است که برای زبان برنامه سازی ای که می خواهید از آن در برنامه کاربردی خود استفاده کنید عملکرد لازم را تامین می کنند. ممکن است شما بصورت مستقیم با این فضانام کاری نداشته باشید.

• Microsoft.VisualBasic: این فضانام محتوی CLR یا زمان اجرای Visual Basic.NET است. از این زمان اجرا با زبان Visual Basic.NET استفاده می شود. این فضانام همچنین دربرگیرنده کلاس هایی است که از کامپایل و تولید کد با استفاده از زبان ویژوال بیسیک پشتیبانی می کنند.

• Microsoft.CSharp: این فضای نام دربرگیرنده کلاس هایی است که از کامپایل و تولید کد با استفاده از زبان #C پشتیبانی می کنند.

• Microsoft.JScript: این فضای نام دربرگیرنده کلاس هایی است که از کامپایل و تولید کد با استفاده از زبان JScript پشتیبانی می کنند.

• Microsoft.Win32: کلاسها و رابطهای مورد نیاز برای کار با کلیدها و hiveهای رجیستری را تامین می کند.

با وجود آنکه فضانام ها از قبل تامین می شوند، می توانید برای استفاده از برنامه کاربردی ASP.NET فضانام های خود را ایجاد کنید. برای هر کلاس ایجاد شده توسط سازنده یک فضانام تولید می شود.

استفاده از فضانام ها در صفحات ASP.NET

دو راه برای افزودن فضانام به برنامه کاربردی ASP.NET وجود دارد. از شبه دستور (Directive) صفحه Import@ برای صفحات ASPX و از کلمه کلیدی Imports برای افزودن فضانام به برنامه codebehind مربوطه در ویژوال بیسیک استفاده می شود و برای زبان #C از دستور using استفاده می گردد. قطعه برنامه زیر ترکیب نحوی برای افزودن فضانام System.Web.UI.WebControls به صفحه ASP.NET شما است.

```
<%@ Import namespace = "System.Web.UI.WebControls" %>
```

همین فضانام را در قسمت codebehind بصورت زیر به برنامه اضافه می کنیم.

```
Imports System.Web.UI.WebControls (vb.net)

using System.Web.UI.WebControls; (C#)
```

(به تفاوت Import و Imports دقت کنید)

در صورتیکه می خواهید چند فضانام را به صفحه ASP.NET خود و یا صفحه codebehind اضافه کنید باید هرکدام را جداگانه اضافه کنید. بعنوان مثال، برای افزودن فضانام System.Web.UI.HTMLControls به صفحات با فضانام های موجود، درست بعد از آخرین عبارت مهم به خط بعد بروید و Imports System.Web.UI.HTMLControls را اضافه کنید. به محض آنکه Imports System. را تایپ کنید، VS.NET فهرستی از فضانام ها را ظاهر می کند، و می توانید به سادگی فضانام مورد نظر را با ماوس برگزینید. امتیاز این فهرست آن است که مجبور نیستید همه فضانام های NET. را از حفظ بدانید، بلکه می توانید به آسانی از فهرست انتخاب کنید. این ویژگی با عنوان Intellisense شناخته می شود. برای صفحات ASP.NET از این ترکیب استفاده کنید:

```
<%@ Import namespace = "System.Web.UI.WebControls" %>
<%@ Import namespace = "System.Web.UI.HTMLControls" %>
<%@ Import namespace = "namespace name" %>
...
```

برای صفحات codebehind ویژوال بیسیک از این ترکیب استفاده کنید:

```
Imports System.Web.UI.WebControls
Imports System.Web.UI.HTMLControls
Imports namespace
...
```

Smart Navigation چیست؟

مفهوم Smart Navigation و فواید آن
Smart Navigation یکی از بهترین ابزارهای جدیدی است که ASP.NET آنرا عرضه کرده است. این ابزار جدید باعث شده ظاهر برنامه های وب و احساسی که نسبت به آن وجود دارد شباهت بیشتری با برنامه های عادی و نوشته شده برای ویندوز پیدا کند.

یکی از موانع بزرگ برنامه های تحت وب به معماری و ساختار HTTP برمی گردد. جاییکه مجبوریم اطلاعات جمع آوری شده در سمت مشتری را به سرور بازگردانیم. به همین دلیل مجبور به رسم مجدد و کامل صفحه ای که قبلا دیده ایم می باشیم، که این نه تنها باعث می شود یك حالت فلش مانند در این رفت و برگشت و رسم مجدد رخ دهد، بلکه برای صفحه های بلند که برای دیدن تمام صفحه نیازمند بهscrolling هستیم، باعث می شود که دیدمان را به اول صفحه انتقال دهد، چیزی که هم شاید دلخواه ما نباشد و هم اینکه ممکن است باعث سردرگمی کاربر گردد. همچنین این فرآیند باعث تغییر فوکوس کنترل ها و بسیاری از اتفاقات دیگرنیز می شود.

در برنامه های عادی ویندوز ما به طور معمول فقط قسمت هایی از صفحه را به روز می کنیم که تغییری در آن ایجاد شده باشد یا تحت تاثیر چیزی قرار گیرند و این بدون نیاز به تغییر در کل برنامه می باشد (مثلا فقط یك عضو به listbox ما اضافه می شود. بدون تغییر و رسم مجدد فرم برنامه).

Smart Navigation یا به عبارتی هدایت هوشمندانه این توانایی موجود در برنامه های ویندوز را برای برنامه های تحت وب فراهم می کند! اما قبل از هر چیز باید بدانید که این ابزار فقط برای IE می باشد و آن هم نسخه های ۵ به بالاتر آن. با این وجود شما می توانید Smart Navigation را فعال یا غیرفعال سازید، بدون آنکه تاثیری در برنامه شما بگذارد. حتی اگر شما در پروژه تان مرورگرهای محتلفی را مدنظر قرار داده باشید، می توانید Smart Navigation را فعال سازید. در این صورت ASP.NET نوع مرورگر را تشخیص داده و Smart Navigation را فقط برای مرورگرهای پشتیبانی شده فعال می سازد.

چهار مورد برجسته ای که Smart Navigation فراهم می کند عبارتند از:

- صفحه در میان درخواست ها یك نمایش ممتد را داراست و به عبارتی حالت فلش زدن به خود نمی گیرد.
- موقعیت Scroll را حفظ می کند.
- فوکوس عضو دارنده فوکوس را نگه می دارد.
- آخرین صفحه درون تاریخچه (History) نگهداری می شود.

این ابزار در حالت واقع گرایانه برای برنامه هایی که ارسال به عقب(!) Postback فراوانی دارند طراحی شده است ولی با توجه به این نکته که محتوای صفحه نباید زیاد تغییر نکند. احتمالا بنا به دلایل کارایی و نه اینکه در تغییرات زیاد ایرادی بهم بزند - مترجم. شاید یك چیز شگفت آور در مورد این ابزار این باشد که شما در حقیقت نیاز به نوشتن هیچ کد و برنامه ای ندارید.

نحوه استفاده
Smart Navigation درون هدایت کننده صفحه (Page directive : <%@ %>)، برای تنظیم یك صفحه و درون web.config برای تنظیم کل برنامه استفاده می شود. برای تنظیم در Page Directive به صورت زیر عمل کنید:

```
<%@ Page SmartNavigation=true %>
```

و برای تنظیم در web.config از ساختار زیر استفاده نمایید:

```
<Configuration>
<System.web>
<Pages SmartNavigation=true />
</System.web>
</Configuration>
```

روش کار اینگونه است که کل صفحه بدرون يك فـريم دورنـی مخفـی (hidden IFrame) بارگـذاری (load) می شـود و سـپس فقط قسـمت های تغییر کرده دوباره رندر(render) می شـوند.

چرا به دات نت احتیاج داریم؟

به طور معمول نسل های جدید زبان های برنامه نویسی به این دلیل متولد می شوند که زبان های قدیمی تر دارای امکانات محدود بودند و یا قدرت استفاده از تکنولوژی های فعلی را به صورت مطلوب و ساده ندارند.

مهمترین نیازی که به عنوان آخرین تکنولوژی وجود دارد، برنامه نویسی در محیط اینترنت است. اینترنت در مدت تقریبا ۸ سال جای خود را به عنوان یکی از مهمترین وسایل ارتباطی برای کارهای روزمره و تجارت باز کرده است. سیستم های برنامه نویسی قدیمی تر امکان برنامه نویسی برای اینترنت را فراهم کرده بودند اما هر کدام دارای اشکالات بزرگی هستند، برای مثال تکنولوژی COM اولین بار در ویندوز به کار گرفته شد. در سال ۱۹۷۰ نیز سیستم هایی برای Unix نوشته شده بودند، جاوا نیز در اصل برای ابزارهای الکترونیکی بود و نه برای اینترنت.

سپس برای اولین بار یک سیستم جامع برای برنامه نویسی تحت اینترنت ایجاد شد. این سیستم -NET. از مراحل سطح پایین که به زبان ماشین می باشد تا بالاترین سطح که برنامه نویسی ویژوال آن می باشد برای استفاده در اینترنت طراحی شده است. البتهNET. فقط برای اینترنت نیست و با استفاده از آن می توان برنامه های کامل تحت Client نیز ایجاد کرد، اما بزرگترین مزیت آن دربرابر سیستم های دیگر امکانات اینترنت آن است.

برای اینکه مزایای استفاده ازNET. را بهتر متوجه بشویم بهتر است در ابتدا معایب سیستم های پیشین را ذکر کنیم. شرکت مایکروسافت تا قبل از سال ۱۹۹۵ به برنامه نویسی در محیط های Client و Server می پرداخت، اما از آن سال به بعد توجه بیشتری به مساله برنامه نویسی در اینترنت کرد. مایکروسافت COM و +COM را ایجاد کرد و آنها را در ویژوال استودیوی ۶ به کار گرفت. در سال ۱۹۹۹ حدود ۵۰ در صد از بزرگترین سایتهای تجارت الکترونیکی از محصولات مایکروسافت استفاده می کردند. اما هنوز هم مشکلات بزرگی در سیستم های مایکروسافت وجود داشت که یکی از آنها دشواری نوشتن برنامه در اینترنت با محصولات مایکروسافت بود. شرکت مایکروسافت برای راحتی کار برنامه نویس ها ASP یا Active Server Page را ایجاد کرد. با اینکه این یک قدم بزرگ بود و کارها را بسیار ساده کرد ولی هنوز از برنامه نویسی شی گرا پشتیبانی نمی کرد. همچنین در ویژوال استودیوی ۶ قسمتی برای Internet Application ایجاد شده بود و در آنها امکان ساختن Web Class وجود داشت ولی هیچ وقت به عنوان یک ابزار کار آمد برای برنامه نویسی وب درنظر گرفته نشد.

مدل برنامه نویسی DNA

مایکروسافت یک مدل برنامه نویسی به نام Distributed interNet Application دارد که بر پایه برنامه نویسی n-tier و COM بنا نهاده شده است. مدل DNA از سه بخش اساسی تشکیل شده است.

بخش اول به نام Presentation tire معروف است. در این بخش رابط تصویری کاربر وجود دارد و خود نیز به دو نوع Internet Browser و Win 32 GUI تقسیم می شود که هر کدام مشکلات خاص خود را دارند. در مدلی که از Win32 GUI یا همان نرم افزارهای معمولی استفاده می شود دو مشکل بزرگ وجود دارد ؛ دشواری بروز رسانی نرم افزار و دیگری DLL Hell که در ادامه توضیح داده خواهد شد. در نوع دوم مشکلاتی از قبیل نبود امکانات برنامه نویسی کافی در محیط مرورگر، نبود رابط قوی با کاربر، نبودن مرورگر های یکسان و... وجود دارد. همچنین همیشه یک اتصال به اینترنت یا اینترانت لازم است. در این نوع از برنامه نویسی می توان از Java Applet ها یا ActiveX استفاده کرد ولی مرورگر باید امکان استفاده از آن را داشته باشد، مخصوصاً هنگام استفاده از ActiveX که باید فقط از IE استفاده کرد.

بخش دوم که Middle tier نام دارد، مکانی است که اطلاعات و قوانین تجاری در آن وجود دارد. منظور از قوانین ، متد ها و اجزائی هستند که اعمال کاربران را کنترل می کنند. مهمترین و آسان ترین زبان برای نوشتن این اجزا از DNA ویژوال بیسیک است. برنامه نویسی که بخواهد در این رده برنامه بنویسد باید آشنایی کاملی با COM و پروتکل های رایج داشته، همچنین باید مهارت کافی در استفاده از ADO و ADSI داشته باشد. مشخص است که یک اشتباه در این لایه باعث بروز خطا و نقص در کل سیستم می شود.

بخش سوم یا Data tier مکانی است که اطلاعات سازمان در آن ذخیره می شود. معمولاً در این قسمت از بانکهای پیشرفته رابطه ای مانند SQL Server و Oracle استفاده می کنند.

محدودیت های COM

همانطور که دیدید مهمترین قسمت در DNA همان COM است که در جای جای آن استفاده می شود. در اینجا برخی معایب COM ذکر می شود : (در ابتدای متن ذکر شد که برای درک نیاز به.NET باید ابتدا معایب سیستم های قدیمی را بشناسیم)

DLL Hell: اگر کوچکترین تغییری در یک COM ایجاد شود، دیگر برنامه هایی که از ورژن قبلی استفاده می کردند قادر به فعال ساختن نسخه جدید نیستند. هنگامی که در ویندوز، یک COM نصب شود برایش در رجیستری یک GUID ثبت می شود که اطلاعات آن COM را در خود ذخیره می کند. اگر یک برنامه از نسخه اول یک COM استفاده کند و بعد از مدتی شما تغییراتی در نسخه اول بدهید و بخواهید آن را دوباره در سیستم نصب کنید ویندوز به شما پیغام خطا می دهد چون ورژن آن تکراری است، اگر هم آن را به ورژن دوم ارتقا دهید نرم افزار قبلی هنوز به دنبال نسخه اول می گردد. این امر باعث می شود که شما مجبور شوید یکبار دیگر کل برنامه را کامپایل کرده و در کامپیوترتان نصب کنید.

کمبود در وراثت: در نسخه های COM که در حال حاضرهستند چیزی به نام وراثتی که در C++ وجود دارد نمی باشد، بلکه وراثت تنها در واسط یک COM می باشد، استفاده از آن هم چندان کمکی به برنامه نویسی نمی کند.

برخی محدودیت های برنامه نویسی اینترنتی در مدل DNA

۱- وجود دو محیط برنامه نویسی برای اینترنت و Client

نقصان در نوشتن برنامه هایی با رابط گرافیکی خوب که در اینترنت کار می کردند کاملاً مشهود است، نمونه بارز آن اختلاف در برنامه نویسی در ویژوال بیسیک و ASP است. ویژوال بیسیک با رابط گرافیکی کاملاً سطح بالا و ASP تقریباً رابط گرافیکی ندارد. همین امر باعث می شد که یک برنامه نویس مجبور باشد طیف وسیعی از تکنیک ها و زبان ها را فرابگیرد تا بتواند برنامه ساده ای در اینترنت بنویسد.

۲- نبودن حالت های ذخیره اطلاعات رابط گرافیکی در صفحه های اینترنتی

نمونه این حالت زمانی است که در یک textbox متنی وجود داشته باشد. در برنامه های Win32 GUI متن داخل textbox تا زمانی که کاربر یا برنامه آن را تغییر نداده بر جای خود وجود دارد. اما در محیط اینترنت و نوع ASP با هر بار refresh کردن صفحه کل اطلاعات ازبین می رود. البته این مشکل با استفاده از شئ های Request و Response تقریباً قابل حل است ولی احتیاج به برنامه نویسی برای هر تکه از صفحه ASP دارد.

۳- نداشتن Event Handler در محیط برنامه نویسی اینترنت

یکی از مهمترین ابزاری که در برنامه نویسی Win32 GUI وجود دارد استفاده از Event ها است. با تکنولوژی که در حال حاضر وجود دارد تنها راه رسیدن به این مهم استفاده از ActiveX است که

به علت مسایل امنیتی در بیش از ۹۵ در صد مواقع توسط کاربر استفاده از آن رد می شود.

معایب استفاده از API

API ها توابعی هستند که از ویندوز نسخه ۱ تا امروز در برنامه نویسی کاربرد داشته و دارند. مهمترین کاری که این توابع انجام می دهند انجام کارهای سخت و سطح پایین سیستمی است که احتیاج به برنامه نویسی زیادی دارند و یا حتی امکان ایجاد آن با زبان هایی مثل ویژوال بیسیک نیست. اما هر API از هر نسخه ویندوز تا نسخه دیگر آن می تواند دچار تغییرات بشود. برای مثال برنامه ای که در ویندوز ۹۸ نوشته شده باشد می تواند در ویندوز ۹۵ اجرا نشود. همچنین هم اکنون ابزارهای جدیدی به بازار آمده است که برای آنها نیز می توان برنامه نویسی کرد، مانند تلفن های سیار، کیوسک تلفن، دستگاه های کامپیوتری جیبی و غیره. در این نوع دستگاه ها دیگر ویندوز به مفهومی که در حال حاضر وجود دارد قابل اجرا نیست و در نتیجه API هم وجود ندارد. لازم به ذکر است که ویندوز CE برای دستگاه های مذکور می باشد ولی قابلیت های آن با ویندوزهای دیگر تفاوت زیادی دارد.

نشان دادن قابلیتهای مرورگر در ASP.NET

اگرچه در حال حاضر جنگ مرورگرها تقریبا تمام شده است اما این موضوع دلیلی بر شناخته نشدن قابلیتهای مرورگرها نیست. در اینجا توانایی ASP.NET در نشان دادن قابلیتهای مرورگرها بحث شده است. بعنوان نمونه، مثال ۱ نوع مرورگر را به ما نشان می دهد.

```
<html><body>
You are using <% =Request.Browser.Type %>
</body></html>
```

برای نمونه اگر شما از IE 5 استفاده می کنید نتیجه خروجی چنین باید باشد:

You are using IE5

در مثال ۱ Request.Browser.Type یک رشته را که همان نام ونسخه‌ی مرورگر است را بر می گرداند. اما این موضوع چگونه صورت میگیرد ؟

شئ HTTPBrowserCapabilities

در حقیقت خاصیت Browser در شئ Request کلاسی از HTTPBrowserCapabilities است که در فضانام System.Web قراردارد. وقتی که این کلاس روی یک صفحه ASP.NET نمونه سازی می شود خواص صفحه سرویس گیرنده ای را نشان می دهد که از آن برای اجرا شدن کد استفاده شده است. شئ Request در برگیرنده این خاصیت مرورگر است که این کلاس را میتوان معادل کلاس MSWC. BrowserCapabilities در ASP کلاسیک در نظر گرفت.

در لیست زیر اکثر خاصیتهای شی HTTPBrowserCapabilities تشریح شده است:

ActiveXControls: نشان می دهد که مرورگر اکتیویکس را ساپورت می کند یا نه.
AOL: چک می کند که مرورگر از نوع AOL است یا نه.
Cookies: نشان می دهد که مرورگر کوکی ها را ساپورت می کند یا نه باید توجه داشت که این خاصیت وضعیت فعال بودن یا غیر فعال بودن کوکی ها را نشان نمی دهد.
Crawler: نشان میدهد که مرورگر سرویس گیرنده از موتورهای جستجو تاثیر می پذیرد یا نه.
Browser: نوع مرورگر را نشان می دهد.
Frames: نشان می دهد که مرورگر از قابلیت Frame برخورداراست یا نه.
MajorVersion: نسخه اصلی مرورگر را نشان می دهد بعنوان مثال در IE5 عدد ۵ نشانگر نسخه اصلی است.
MinorVersion: نسخه جزیی (کوچکتر) مرورگر را نشان می دهد بعنوان مثال در IE5.1 عدد ۱. نشانگر نسخه جزیی است.
Type: نوع ونسخه مرورگر را بصورت یک رشته بازمیگرداند..
VBScript: نشان می دهد که مرورگر VBScript را ساپورت می کند یا نه.
Version: نسخه اصلی و جزیی مرورگر را بعنوان یک رشته برمی گرداند.

در زیر نمونه کامل یک مثال آورده شده است.

```
<%@ page language="VB" %>
<%@ Import Namespace="System.Web" %>
<html>
<body>
```

```
<head><title>HTTPBrowserCapabilities Demo</title></head>

<%
Dim browserObj As HTTPBrowserCapabilities
browserObj = Request.Browser
%>
<font face="verdana, arial" size=2>

<p>Your browser supports ActiveX controls: <%=browserObj.ActiveXControls %>
</p>
<p>Your browser type: <%=browserObj.Type %> </p>
<p>Your browser version: <%=browserObj.Version%> </p>

... Add any other property that you would like to display

</font>
</body>
</html>
```

آشنائی با ASP.NET

NET نسل بعدی Active Server Pages یا ASP است که توسط شرکت میکروسافت ارائه شده است. این محصول توسط میکروسافت بعنوان شاخص اصلی فناوری در ساخت سایتهای وب در نظر گرفته شده است. با استفاده از ASP.NET می توان هم اینترانت کوچک یک شرکت را ساخت و هم یک سایت وب تجاری خیلی بزرگ را طراحی و پیاده سازی نمود. مهمترین نکاتی که در طراحی این محصول در نظر گرفته شده است راحتی استفاده و بالا بودن کارائی و قابلیت آن می باشد. در زیر برخی ویژگیهای ASP.NET را بررسی می کنیم.

• صفحات ASP.NET کامپایل می شوند.

هنگامی که یک صفحه ASP.NET برای اولین بارتوسط یک مراجعه کننده به سایت فراخوانی می شود، آن صفحه ابتدا کامپایل شده و بر روی سرور نگهداشته می شود و در فراخوانی های بعدی از آن استفاده می شود. این بدین معنی است که صفحات ASP.NET خیلی سریع اجرا می شوند.

• صفحات ASP.NET با ابزارهای روی سرور ساخته می شوند.

با ابزارهای موجود در ASP.NET می توان صفحات پیچیده وب را براحتی طراحی نمود. بعنوان مثال با استفاده از ابزار DataGrid می توان به آسانی داده های موجود در یک بانک اطلاعاتی را تحت وب نمایش داد.

• مجموعهASP.NET عضوی از بدنه .NET است.

بدنه .NET دارای بیش از ۴۵۰۰ کلاس آماده جهت استفاده در ASP.NET است. این کلاس ها تقریبا هر نیازی را در برنامه نویسی برآورده می کنند. بعنوان مثال از این کلاس ها می توان جهت تولید تصاویر بر حسب تقاضا، به رمز درآوردن یک فایل و یا ارسال یک نامه استفاده کرد.

مقایسه ASP.NET و ASP کلاسیک

ASP.NET نسل بعدی ASP یا ASP کلاسیک است. اما این یک پیشرفت تکاملی است بطوریکه این دو فناوری تقریبا از یکدیگر متفاوتند. صفحات ASP با زبان های دستورالعمل نویسی مانند VBScript یا JScript ایجاد می شوند اما در ASP.NET ما یک فرایند کامل برنامه نویسی با زبانهای Visual Basic یا C# (سي-شارپ تلفظ شود) داریم. همچنین در ASP کلاسیک تنها پنج کلاس استاندارد (Request, Response, Application Session, Server) وجود دارد حال آنکه در ASP.NET می توان از بیش از ۴۵۰۰ کلاس استاندارد موجود در بدنه .NET بهره جست.همچنین علیرغم قدرت و امکانات زیاد و متعدد ASP.NET، استفاده از آن در مقایسه با ASP کلاسیک بسیار آسانتر است. بعنوان مثال با استفاده از چند ابزار در یک صفحه ASP.NET می توان یک صفحه بسیار پیچیده HTML بدست آورد که ساخت آن در ASP کلاسیک ممکن است نیاز به چند روز کار داشته باشد.

زبانهای برنامه نویسی در ASP.NET

شما در ASP.NET می توانید از هر زبان برنامه نویسی که با بدنه .NET سازگار باشد استفاده کنید. این زبانها عبارتند از Visual Basic.NET و C# و JScript.NET . این بدین معنی است که شما جهت نوشتن برنامه در ASP.NET نیاز به فراگیری زبان جدیدی ندارید و اگر یکی از زبانهای ویژوال بیسیک یا C++ یا جاوا را می دانید هم اکنون می توانید در ASP.NET برنامه بنویسید.از طرف دیگر تعدادی زبانهای دیگر توسط بعضی از شرکتهای فعال در این زمینه به مجموعه زبانهای استاندارد ASP.NET افزوده شده است. بعنوان مثال اگر مایل باشید حتی می توانید از PERL و COBOL هم در ASP.NET استفاده کنید.

ابزارهای ASP.NET

سالهاست که برنامه نویسان ویژوال بیسیک جهت ساخت فرم های خود از ابزارهای ویژوال بیسیک مانند TextBox و ListBox استفاده کرده اند. در ASP.NET هم شما می توانید از ابزارهای فراوان موجود در آن برای ساخت فرم ها و صفحات خود استفاده نمائید. در ASP.NET چهار دسته عمده از ابزارها موجود است:

- ابزارهای اصلی مانند TextBox، RadioButton، ListBox و Button.
- ابزارهای اعتباری برای حصول اطمینان از ورود و تائید صحت اطلاعات ورودی فرم ها.
- ابزارهای داده ای برای ارتباط با بانک اطلاعاتی و دستکاری داده.
- ابزارهای پیشرفته جهت نمایش عناصر پیچیده در واسط کاربر مانند تقویم و آگهی های تبلیغاتي.

با استفاده از Visual Studio.NET شما براحتی می توانید با چیدن تصویری این ابزارها بر روی فرم مورد نظر، صفحه دلخواه خود را بسازید. در صورت تمایل حتی می توانید در یک ویرایشگر ساده متن مانند Notepad برنامه مورد نظر را نوشته و از این ابزارها استفاده کنید.

دریافت ASP.NET

جهت شروع برنامه نویسی در ASP.NET تنها کافی است که مجموعه ASP.NET را بهمراه بدنه .NET از سایت میکروسافت دریافت کنید.

http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/000/976/msdncompositedoc.xml

ASP.NET با سیستم عامل های Windows 2000 (نسخه Server و Professional) و Windows XP کاملا سازگار است.

نحوه پردازش صفحات ASP.NET بر روي سرويس دهنده وب

برنامه هاي وب از معماري سرويس گيرنده - سرويس دهنده تبعيت نموده و بر روي سرويس دهنده وب مستقر و مسئوليت پاسخگوئي به درخواست هاي ارسالي توسط سرويس گيرندگان را برعهده خواهند داشت .در سمت سرويس گيرنده ، مرورگر و در سمت سرويس دهنده ، سرويس دهنده وب داراي جايگاهي خاص مي باشند . مرورگر ، ميزبان برنامه وب بوده و مهمترين وظيفه آن ارائه بخش رابط كاربر يک برنامه وب است . در اين راستا ، مرورگر داراي پتانسيل لازم به منظور تفسير و نمايش تگ هاي HTML مي باشد .در سمت سرويس دهنده ، برنامه هاي وب با نظارت و مديريت يک سرويس دهنده وب (مثلا" IIS) اجراء مي گردند . سرويس دهنده وب ، مسئوليت مديريت برنامه ، پردازش درخواست هاي ارسالي توسط سرويس گيرندگان و ارائه پاسخ لازم به سرويس گيرندگان را بر عهده دارد .به منظور قانونمند كردن ارسال درخواست سرويس گيرندگان و ارائه پاسخ سرويس دهنده ، مي بايست از يک پروتكل ارتباطي خاص استفاده گردد. پروتكل ، مجموعه اي از قوانين لازم بمنظور تشريح نحوه ارتباط دو و يا چندين آيتم از طريق يک محيط انتقال (زير ساخت انتقال داده) نظير اينترنت است . در برنامه هاي وب (ارسال درخواست توسط سرويس گيرنده و پاسخ به درخواست توسط سرويس دهنده) از پروتكل ارتباطي Hypertext Transport Protocol(HTTP) ، استفاده مي گردد.
ASP.NET پلات فرم مايكروسافت براي طراحي و پياده سازي برنامه هاي وب در دات نت مي باشد . پس از درخواست يک صفحه ASP.NET توسط مرورگر سرويس گيرنده ، پردازش هاي متعددي بر روي سرويس دهنده وب به منظور ارائه پاسخ لازم ، انجام خواهد شد.شايد تاكنون سوالات مختلفي در رابطه با نحوه پردازش صفحات ASP.NET بر روي سرويس دهنده ، براي شما مطرح شده باشد :

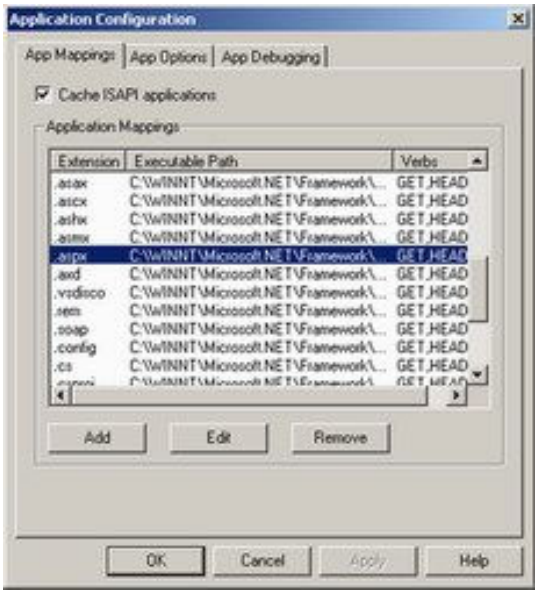
پس از درخواست يک صفحه ASP.NET ، بر روي سرويس دهنده وب چه اتفاقي مي افتد ؟
نحوه برخورد سرويس دهنده وب با درخواست ارسالي توسط سرويس گيرنده چگونه است ؟
تگ هاي HTML چگونه توليد و براي مرورگر ارسال مي گردد؟
و شايد سوالات ديگر!
در اين مقاله قصد داريم با نحوه پردازش صفحات ASP.NET بر روي سرويس دهنده بيشتر آشنا شويم . بديهي است تشريح تمامي مراحل با ذكر جزئيات از حوصله يک مقاله خارج بوده و هدف آشنائي با كليات موضوع با يک روند مشخص و سيستماتيک است .

مرحله اول : ايجاد يک درخواست HTTP براي يک صفحه ASP.NET توسط مرورگر
پردازش با درخواست يک صفحه ASP.NET كه توسط مرورگر ايجاد مي شود ، آغاز مي گردد .مثلا" يک كاربر ممكن است در بخش آدرس مرورگر كامپيوتر خود آدرس www.srco.ir//Articles/DocView.asp?ID=210 را به منظور دريافت اين مقاله وارد نمايد . مرورگر در ادامه يک درخواست HTTP را از سرويس دهنده وب محل استقرار سايت Srco.ir ايجاد و درخواست فايل حاوي مقاله را مي نمايد .

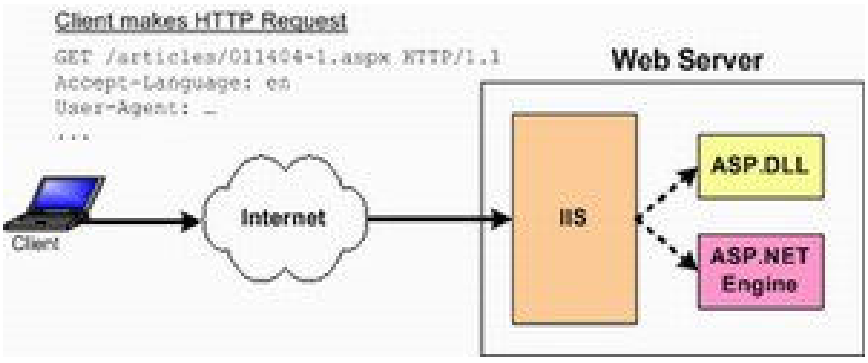
مرحله دوم : دريافت درخواست HTTP ، توسط سرويس دهنده وب
مهمترين وظيفه سرويس دهنده وب ، دريافت درخواست ارسالي HTTP و ارائه منبع درخواست شده درقالب يک پاسخ HTTP است . سرويس دهنده وب (مثلا" IIS) ، پس از دريافت درخواست ارسال شده توسط سرويس گيرنده ، تصميم لازم در رابطه با نحوه برخورد با آن را اتخاذ مي نمايد. محور تصميم گيري فوق بر پايه نوع انشعاب فايل درخواستي استوار مي باشد. مثلا" در صورتيكه فايل درخواستي داراي انشعاب asp. ، باشد ، IIS درخواست را به سمت asp.dll هدايت تا عمليات مرتبط با آن انجام شود . انشعابات فايل متعددي به موتور ASP.NET ، مپ مي گردند . برخي از آنان شامل موارد زير مي باشد :

انشعاب aspx . ، براي صفحات وب ASP.NET

انشعاب asmx . ، براي سرويس هاي وب ASP.NET
انشعاب config . ، براي فايل هاي پيكربندي ASP.NET
انشعاب ashx . ، براي هندلرهاي سفارشي ASP.NET HTTP
انشعاب rem . ، براي منابع راه دور
و ساير انشعابات ديگر



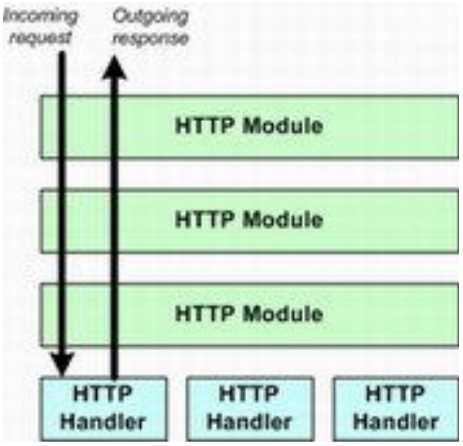
پس از دريافت درخواست ارسالي توسط سرويس گيرنده ، سرويس دهنده وب آن را در اختيار مسئول مربوطه قرار خواهد داد . مثلا" در صورتيكه درخواست دريافتي مربوط به يک صفحه ASP كلاسيک باشد ، درخواست در اختيار asp.dll گذاشته شده و يا در صورتيكه درخواست در ارتباط با يک صفحه ASP.NET باشد ، درخواست در اختيار موتور ASP.NET قرار داده مي شود . همانگونه كه اشاره گرديد ، معيار اصلي در اين تصميم گيري ، نوع انشعاب فايل درخواست شده توسط سرويس گيرنده مي باشد . شكل زير مراحل اول و دوم اشاره شده را نشان مي دهد .



منبع : http://www.4guysfromrolla.com

مرحله سوم : عملكرد موتور ASP.NET
پس از درخواست يک صفحه ASP.NET توسط سرويس گيرنده و دريافت آن توسط سرويس دهنده وب ، درخواست دريافتي در اختيار موتور ASP.NET قرار داده مي شود . از موتور ASP.NET ، اغلب با نام ASP.NET HTTP pipeline ياد مي گردد. علت نامگذاري فوق ، بدين دليل است كه درخواست دريافتي از بين تعداد متغيري از HTTP modules در بين مسير خود براي رسيدن به يک HTTP handler عبور مي نمايد . HTTP modules ، كلاس هائي مي باشند كه امكان دستيابي

به درخواست دريافتي را دارا مـي باشـند. ايـن مـاژول هـا قـادر بـه بـازبيني و بررسـي درخواسـت دريافتي و اتخاد تصميماتي مي باشند كـه مسـتقيما" بـر نحـوه گـردش داخلـي (رونـد برخـورد بـا درخواسـت) تاثير خواهد گذاشت . درخواسـت دريافتي پس از عبور از ماژول هـاي مسـخص شـده HTTP ، به يک HTTP Handler خواهد رسيد . HTTP Handler مسـئوليت ايجـاد خروجـي لازم بـه منظـور ارسـال بـراي مرورگـر متقاصـي (ارسـال كننـده درخواسـت) را برعهـده دارد. شـكل زيـر ، pipline يک درخواست ASP.NET را نشان مي دهد .



منبع : http://www.4guysfromrolla.com

تعداد زيادي از ماژول هاي HTTP از قبل ايجـاد شـده، بصـورت پـيش فـرض در HTTP pipline وجـود دارد:

OutputCache ، مسئوليت برگرداندن و Caching خروجي صفحات HTML در صـورت نيـاز ، برعهـده دارد .

Session ، ماژول فـوق ، مسـئوليت لـود Session state را بـر اسـاس درخواسـت دريـافتي كـاربر و روش Session كه در فايل Web.config مشخص شده است ، برعهده دارد .

FormsAuthentication ، ماژول فوق ، مسئوليت تائيد كاربران بر اساس مدل تعريف شـده Forms Authentication را در صورت ضرورت برعهده دارد .

و موارد ديگر

بـه منظورآشـنائي بـا مـاژول هـاي پـيش فـرض، مـي تـوان مقـادير نسـبت داده شـده بـه عنصـر <httpModules> در فايل machine.config را مشاهده نمود. جدول زيـر مقـدار پـيش فـرض عنصـر <httpModules> را نشان مي دهد .

machine.Config: httpModules Section
Path : $WINDOWS$\Microsoft.NET\Framework\$VERSION$\CONFIG

```
<httpModules>
<add name="OutputCache" type="System.Web.Caching.OutputCacheModule"/>
<add name="Session" type="System.Web.SessionState.SessionStateModule"/>
<add name="WindowsAuthentication"
type="System.Web.Security.WindowsAuthenticationModule"/>
```

```
<add name="FormsAuthentication"
type="System.Web.Security.FormsAuthenticationModule"/>
<add name="PassportAuthentication"
type="System.Web.Security.PassportAuthenticationModule"/>
<add name="UrlAuthorization" type="System.Web.Security.UrlAuthorizationModule"/>
<add name="FileAuthorization" type="System.Web.Security.FileAuthorizationModule"/>
<add name="ErrorHandlerModule" type="System.Web.Mobile.ErrorHandlerModule,
System.Web.Mobile,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
</httpModules>
```

هندلرهاي HTTP ، نقطه پايان در ASP.NET HTTP pipeline مي باشند . مسئوليت HTTP handler ، توليد خروجي براي منبع درخواست شده است . براي صفحات ASP.NET ، اين به معني Rendering ، كنترل هاي وب به HTML و برگرداندن HTML مي باشد. براي يک سرويس وب ، مسئوليت فوق ، شامل اجراي متد مشخص شده و Wrapping مقاير برگردانده شده به يک پاسخ مناسب و با فرمت SOAP مي باشد . منابع متفاوت ASP.NET از هندلرهاي متفاوت HTTP استفاده مي نمايند.هندلرهاي پيش فرض استفاده شده ، توسط بخش <httpHandlers> فايل machine.config مشخص شده اند. بخش فوق، شامل كلاس هائي است كه يا خود HTTP handler بوده و يا HTTP handler factories ، مي باشند. يک HTTP handler factory ، صرفا" يک نمونه از يک HTTP handler را پس از فراخواني ، برمي گرداند . جدول زير ، اطلاعات عنصر <httpHandlers> در فايل machine.config را نشان مي دهد .

machine.Config: httpHandlers Section
Path : $WINDOWS$\Microsoft.NET\Framework\$VERSION$\CONFIG

```
<httpHandlers>
<add verb="*" path="trace.axd" type="System.Web.Handlers.TraceHandler"/>
<add verb="*" path="*.aspx" type="System.Web.UI.PageHandlerFactory"/>
<add verb="*" path="*.ashx" type="System.Web.UI.SimpleHandlerFactory"/>
. ...
<add verb="*" path="*.resources" type="System.Web.HttpForbiddenHandler"/>
<add verb="GET,HEAD" path="*" type="System.Web.StaticFileHandler"/>
<add verb="*" path="*" type="System.Web.HttpMethodNotAllowedHandler"/>
</httpHandlers>
```

لازم است به اين نكته اشاره گردد كه امكان ايجاد HTTP modules و HTTP handler اختصاصي ، توسط طراحان وپياده كنندگان برنامه هاي وب ASP.NET نيز وجود دارد . پس از ايجاد ماژول ها و هندلرهاي HTTP ، مي توان آنان را به pipeline ملحق تا براي تمامي سايت هاي وب موجود بر سرويس دهنده وب ، قابل استفاده گردند. بدين منظور، مي توان تغييرات لازم را در فايل machine.config اعمال تا زمينه استفاده از آنان توسط تمامي برنامه هاي وب فراهم گردد . در اين رابطه مي توان تغييرات را در فايل Web.config نيز اعمال نمود، در چنين مواردي امكان استفاده از ماژول ها و هندلرهاي HTTP ايجاد شده ، صرفا" براي يک برنامه وب وجود خواهد داشت .

مرحله چهارم : توليد خروجي

آخرين مرحله درارتباط با پردازش يک صفحه ASP.NET بر روي سرويس دهنده وب ، شامل ايجاد خروجي مناسب است . خروجي فوق ، در ادامه از طريق ماژول هاي HTTP عبور داده شده تا

مجددا" به IIS برسد . در نهايت IIS ، خروجي توليد شده را براي سرويس گيرنده متقاضي ارسال مي نمايد .مراحل لازم به منظور توليد خروجي با توجه به HTTP handler متفاوت بوده و در ادامه صرفا" يک حالت خاص آن را بررسي مي نمائيم (هندلر HTTP که از آن به منظور rendering صفحات ASP.NET استفاده مي گردد).

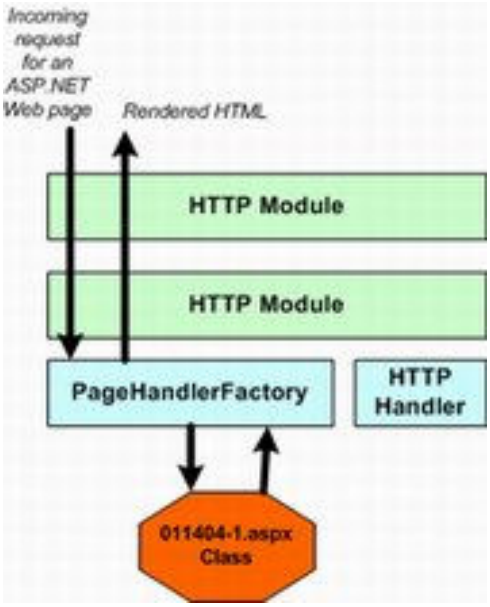
سرويس دهنده وب (IIS) پس از دريافت درخواستي براي يک صفحه ASP.NET (انشعاب فايل aspx.) ، آن را در اختيار موتور ASP.NET ، قرار خواهد داد. درخواست دريافتي در ادامه از بين ماژول ها عبور داده شده تا به PageHandlerFactory برسد (در بخش <httpHandlers> فايل machin.config که قبلا" mapping آن انجام شده است) .

machine.Config: httpHandlers Section :PageHandlerFactory
Path : $WINDOWS$\Microsoft.NET\Framework\$VERSION$\CONFIG

<httpHandlers>
...
<add verb="*" path="*.aspx" type="System.Web.UI.PageHandlerFactory"/>
. ...
</httpHandlers>

کلاس PageHandlerFactory ، يک HTTP handler factory است که وظيفه آن ارائه نمونه اي از يک HTTP handlerبوده که قادر به برخورد مناسب با درخواست ارسالي است.مهمترين رسالت PageHandlerFactory ، يافتن کلاس ترجمه شده اي است که نشاندهنده صفحه ASP.NET درخواستي مي باشد. در صورتيکه از ويژوال استوديو دات نت به منظور ايجاد صفحات ASP.NET استفاده مي گردد ، صفحات وب از دو فايل جداگانه (يک فايل با انشعاب aspx . ، شامل صرفا" کنترل هاي وب و تگ هاي HTML و يک فايل aspx.vb و يا aspx.cs شامل کلاس code-behind (کد سمت سرويس دهنده)) ، تشکيل مي گردند. در صورتيکه از ويژوال استوديو دات نت استفاده نمي گردد ، مي توان از يک بلاک سمت سرويس دهنده <Script> استفاده تا کد سمت سرويس دهنده را درخود نگهداري نمايد . صرفنظر از اينکه از کدام رويکرد استفاده مي گردد ، زمانيکه صفحه ASP.NET اولين مرتبه و پس از ايجاد تغيير در تگ هاي HTML و يا محتوي کنترل وب ، مشاهده مي گردد ، موتور ASP.NET يک کلاس که مشتق شده از کلاس System.Web.UI.Page مي باشد را ايجاد مي نمايد . کلاس فوق بصورت اتوماتيک ايجاد و کمپايل مي گردد .

Page Class ، عمليات پياده سازي IhttpHandler را انجام خواهد داد. PageHandlerFactory ، در ادامه بررسي لازم در خصوص وجود يک نسخه کمپايل شده از صفحه ASP.NET درخواستي را انجام خواهد داد. در صورتيکه صفحه ترجمه شده وجود نداشته باشد ، PageHandlerFactory آن را بصورت پويا ايجاد و ترجمه خواهد کرد . کلاس فوق ، در ادامه متد خاصي را به منظور توليد HTML ، فرا مي خواند . اطلاعات توليد شده به فرمت HTML ، در نهايت براي سرويس گيرنده ارسال مي گردد.وجود تاخير در مشاهده صفحات ASP.NET که بر روي آنان تغييراتي اعمال شده است (HTML و يا محتوي کنترل وب) ، بدين دليل است که موتور ASP.NET نيازمند ايجاد و ترجمه مجدد کلاس مرتبط با صفحه ASP.NET مي باشد.



منبع : http://www.4guysfromrolla.com

پس از ايجاد و ترجمه كلاس توسط PageHandlerFactory ، امكان فراخواني كلاس ايجاد شده به منظور توليد HTML ، فراهم مي گردد . فرآيند Rendering كه شامل بدست آوردن HTML لازم براي صفحه ASP.NET درخواست شده مي باشد از حوصله اين مقاله خارج بوده و مي توان در اين رابطه از مقاله The ASP.NET Page Object Model استفاده نمود .

ارسال ایمیل در ASP.NET با استفاده از HTML Template

آیا تاکنون سعی کرده اید برای سایت خود خبرنامه ایجاد کنید؟ آیا تاکنون وسوسه شده اید که سیستمی طراحی کنید که در صورتی که بینندگان سایت شما نظرات خود را در سایت شما وارد کنند برای آنها یک Email تشکرآمیز ارسال کنید؟ آیا می دانید ساختن یک HTML Template زیبا برای خبرنامه شما و استفاده همیشگی از آن جهت ارسال خبرنامه در ASP.NET بسیار ساده صورت می گیرد؟ چنانچه سوالات بالا شما را به دانستن بیشتر ترغیب کرده است به شما تبریک می گویم! چرا که در ادامه این مقاله شما روش بسیار ساده ارسال Emailهای HTML زیبا را بدون اینکه نگران چگونگی کدنویسی HTML متغیر رشته ای Body در MailMessage باشید را فرا خواهید گرفت.

برای شروع بیایید به روش کار نظری بیاندازیم: خوب احتمالا شما با روش ساختن یک نمونه از آبجکت MailMessage که در ASP.NET برای ارسال Email استفاده می شود آشنایی دارید (اگر این چنین نیست در ادامه مقاله به طور اجمالی توضیحاتی ارائه شده است) تنها قسمتی که باید مورد توجه قرار دهید این است که چگونه می توانیم Body نامه خود را به فرمت HTML و بدون نیاز به اینکه تمامی تگها را پشت سر هم در یک رشته طولانی و سردرگم کننده تایپ کنیم، درآوریم. یک ایده جالب این است که ما Template نامه خود را با استفاده از ادیتورهای WYSIWYG مثل FrontPage یا Dreamweaver طراحی کنیم و سپس تمامی کدهای HTML آن را درون یک متغیر رشته ای Import کرده و از آن استفاده کنیم. این کار بسیار آسان است. اما می توان این نامه را برای هر کاربر کمی سفارشی (Customize) کرد! برای مثال شما می توانید در ابتدای نامه کاربر را با نام وی مورد خطاب قرار دهید. بهتر است از این پس توضیحات را همراه با کدنویسی دنبال کنیم. (کلیه کدهای Server Side به زبان VB.NET نوشته شده است) سه گام اساسی برای این کار وجود دارد:

۱- ساختن Template مورد نظر شما جهت ارسال Email:

برای این کار کافی است که Template مورد نظر خود را به فرمت HTML طراحی کنید. این به خود شما بستگی دارد که کدهای HTML را به صورت دستی بنویسید و یا از ادیتورهای WYSIWYG مانند FrontPage یا Dreamweaver یا GoLive! استفاده کنید. تنها تفاوتی که در اینجا وجود دارد این است که شما بایستی قسمتهایی از متن نامه خود را که متغیر هستند (مانند نام گیرنده Email، آدرس پست الکترونیکی وی و ...) را به گونه ای از بقیه قسمتها متمایز کنید. برای مثال اگر شما می خواهید در ابتدای نامه، دریافت کننده نامه را با نام خود مورد خطاب قرار دهید عبارت را به صورت زیر وارد کنید: "سلام #NAME#" لطفا توجه کنید که هیچ محدودیتی در تکنیک به کار رفته وجود ندارد و قرار دادن کاراکترهای ## فقط جهت متمایز ساختن این قسمت از محتوای استاتیک صفحه است و شما می توانید به صور دیگر آن را مشخص کنید مثلا ?NAME? یا هر چیز مشابه دیگر. این قسمت بعدا با نام شخص مورد نظر ما جایگزین می شود. نکته قابل ذکر دیگر اینکه چنانچه قصد دارید نامه خود را فارسی ارسال کنید تگ زیر را فراموش نکنید:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

۲- خواندن فایل Template جهت قراردادن در Body نامه شما:

پس از اینکه Template را آماده نمودید بایستی این فایل را بخوانید، قسمتهای مورد نظر (متمایز شده با ##) را با عبارات مورد نظر خود (که می توانید از Database استخراج کنید) جایگزین کنید و این متن را به صورت یک متغیر رشته ای در Body نامه خود قرار دهید (گام ۳). این کار به وسیله کد زیر صورت می پذیرد: (قبل از هر چیز شما بایستی Namespace مورد نیاز که در اینجا System.IO می باشد را Import کنید)

```
Imports System.IO

Dim reader As StreamReader
Dim strFileName As String = Server.MapPath("templatel.htm")
Dim strFileText

reader = File.OpenText(strFileName)

While reader.Peek <> -1
        strFileText += reader.ReadLine()
End While

reader.Close()
strFileText = Replace(strFileText, "#Username#", Name.Text)
strFileText = Replace(strFileText, "#MsgID#", Request("MsgID"))
strFileText = Replace(strFileText, "#AuthorID#", Request("AuthorID"))
```

حال بیایید نگاهی به کد فوق بیاندازیم: در سطر اول شما یک شی از نوع StreamReader که جهت خواندن فایلهای متنی به کار می رود را می سازید و در سطر بعدی مسیر فیزیکی فایل Template خود را در متغیر رشته‌ای strFileName ذخیره می کنید.

نکته: در صورتیکه بخواهید در فایل کلاس خود از کد فوق استفاده کنید بایستی مسیر فیزیکی فایل را با استفاده از System.Web.HttpContext.Current.Server.MapPath("template.htm") به دست آورید. اما در Code Behind یک WebForm همان Server.MapPath() کافی است.

سپس در سطر بعدی ما با استفاده از متد OpenText فایل مورد نظر را باز کرده و متن آن را در Reader قرار می دهیم. حال در یک حلقه While...End While سطر به سطر فایل Template را (که در اینجا کدهای HTML ما هستند) می خوانیم و در متغیر strFileText ذخیره می کنیم. Reader.Peek <> -1 بررسی می کند که آیا به انتهای فایل رسیده ایم یا خیر. پس از اینکه کل فایل را خوانده و در متغیر ذخیره کردیم. شی reader را می بندیم.

در اینجا شما باید Template نامه خود را برای کاربر خاص Customize کنید. یعنی عبارات محصور شده با ## را با اطلاعات کاربر (که از بانک اطلاعاتی استخراج می شود و یا از یک Query String دریافت می شود) جایگزین کنید. در مثال بالا مقدار اول در Template با مقدار یک Textbox و دو مورد بعدی با Query String های انتهای یک URL جایگزین می شود.

حاصل کار یک متغیر رشته ای به نام strFileText است که حاوی کد HTML مورد نیاز شما برای ارسال یک HTML Email می باشد. که مثلا عبارت ابتدای آن به صورت زیر در آمده است: "سلام مهدی"

۳- استفاده از قالب Import شده فوق و ارسال Email:

حال به آسانی Email را ساخته و ارسال می کنیم. بدین ترتیب: (Imports System.Web.Mail را فراموش نکنید!)

```
'Creating and sending mail to user
Dim objMail As New MailMessage()
objMail.From = "You@YourDomain.com"
```

```
objMail.To = Email.Text.Trim
objMail.Subject = "YOUR SUBJECT GOES HERE..."
objMail.BodyFormat = MailFormat.Html
objMail.BodyEncoding = System.Text.Encoding.UTF8
objMail.Body = strFileText
SmtpMail.SmtpServer = "smtp.YOURSERVER.com"
SmtpMail.Send(objMail)
```

این قسمت نیاز به توضیح چندانی ندارد. جز اینکه objMail.To را بایستی به صورت داینامیک (از DB و ...) تغییر دهید (در اینجا مقدار از یک Textbox دریافت شده است)، SMTP Server خود را مشخص کنید و Email را ارسال کنید! به همین سادگی. بقیه موارد بسیار واضح هستند.

لطفا به این نکته توجه کنید که برخی SMTP Serverهایی که نیاز به Authentication دارند را نمی توان برای این منظور (ارسال Email به آدرس های خارج از SMTP Server فوق) مورد استفاده قرار داد. در این خصوص لطفا به Administrator سرور خود رجوع کنید.

اضافه کردن تصویر به پایگاه داده

در بسیاری مواقع ما نیاز به افزودن تصویر به پایگاه داده داریم. از آنجایی که دسترسی به فایلهای موجود در سرور جهت استفاده هکرها بسیار سهل الوصول است، برخی مواقع نمی‌توانیم اطلاعات حساس را در قالب فایلهای تصویری در سرور ذخیره کنیم. به همین جهت بهتر آن است که در قالب نوع داده image در بانک اطلاعاتی ذخیره شود.

در این قسمت موارد زیر بررسی می شوند:

بررسی پیش نیازهای اضافه نمودن فایل تصویری
کار با شیء Stream
پیدا کردن اندازه و نوع تصویری که آپلود می شود
چگونگی استفاده از متد InputStream

بررسی پیش نیازهای اضافه نمودن فایل تصویری

دو مورد اصلی که قبل از شروع آپلود نیاز داریم عبارتند از:

۱. - تنظیم خاصیت enctype فرم مربوطه (در برچسب Form) به enctype="multipart/form-data"
- داشتن <input type=file> که به کاربر اجازه انتخاب فایل مورد نظرش را می دهد. (فایلی که در پایگاه داده باید ذخیره شود) و یک Submit button جهت اجرای عملیات مربوط به آپلود کردن فای
- استفاده از فضانام System.IO جهت سروکار داشتن با شیء Stream

۲. سه پارامتر بالا در یک صفحه aspx بکار می روند.ما همچنین به پیش نیازهای زیر در SQL Server نیازمندیم:
- داشتن حداقل یک جدول با فیلدی از نوع image
- بهتر است که فیلد دیگری از نوع varchar جهت ذخیره تایپ تصویر داشته باشیم

روند کار بدین صورت است که در ابتدا محتویات فایل تصویر خوانده می‌شود و سپس تصویر به جدول افزوده می‌شود. در زیر کد مربوط به رویداد OnClick مربوط به Submit button که تصویر را خوانده و به جدول SQL اضافه می‌کند، می‌پردازیم:

```
Public Sub AddPerson(sender As Object, e As EventArgs)
  Dim intImageSize As Int64
  Dim strImageType As String
  Dim ImageStream As Stream

  ' Gets the Size of the Image
  intImageSize = PersonImage.PostedFile.ContentLength

  ' Gets the Image Type
  strImageType = PersonImage.PostedFile.ContentType

  ' Reads the Image
  ImageStream = PersonImage.PostedFile.InputStream
```

```
    Dim ImageContent(intImageSize) As Byte
    Dim intStatus As Integer
    intStatus = ImageStream.Read(ImageContent, 0, intImageSize)

    ' Create Instance of Connection and Command Object
    Dim myConnection As New SqlConnection(
ConfigurationSettings.AppSettings("ConnectionString"))
    Dim myCommand As New SqlCommand("sp_person_isp", myConnection)

    ' Mark the Command as a SPROC
    myCommand.CommandType = CommandType.StoredProcedure

    ' Add Parameters to SPROC
    Dim prmPersonImage As New SqlParameter("@PersonImage", SqlDbType.Image)
    prmPersonImage.Value = ImageContent
    myCommand.Parameters.Add(prmPersonImage)

    Dim prmPersonImageType As New SqlParameter("@PersonImageType",
SqlDbType.VarChar, 255)
    prmPersonImageType.Value = strImageType
    myCommand.Parameters.Add(prmPersonImageType)

    Try
      myConnection.Open()
      myCommand.ExecuteNonQuery()
      myConnection.Close()
      Response.Write("New person successfully added!")
    Catch SQLexc As SqlException
      Response.Write("Insert Failed. Error Details are: " & SQLexc.ToString())
    End Try
End Sub
```

شیء PersonImage نام کنترل HTMLInputFile است. در ابتدا ما نیاز داریم که اندازه تصویری که جهت اضافه کردن انتخاب شده را بدانیم که توسط کد زیر قابل اندازه گیری است:

```
intImageSize = PersonImage.PostedFile.ContentLength
```

سپس نوع تصویر را با استفاده از خاصیت ContentType دریافت می کنیم. حال باید یک استریم از تصویر مربوطه دریافت شود:

```
ImageStream = PersonImage.PostedFile.InputStream
```

در اینجا ما یک آرایه از بایتها بنام ImageContent که آماده نگهداری محتویات تصویر است، داریم. تصویر ورودی توسط متد Read از شیء Stream خوانده می شود. متد Read سه آرگومان می گیرد:

۱. موقعیت مکانی که محتویات تصویر در آن باید کپی شود
۲. مکان شروع جهت خواندن
۳. تعداد بایتهایی که باید خوانده شود

و دستور مربوط به خواندن هم به شکل زیر است:

```
intStatus = ImageStream.Read(ImageContent, 0, intImageSize)
```

حالا ما می خواهیم محتویات تصویر ورودی را که خوانده ایم به یک جدول SQL اضافه کنیم. به همین منظور از یک زیرروال ذخیره شده که تصویر و نوع آن را به یک جدول اضافه می کند، بهره می بریم. حالا ما می خواهیم محتویات تصویر ورودی را که خوانده ایم به یک جدول SQL اضافه کنیم. به همین منظور از یک زیرروال ذخیره شده که تصویر و نوع آن را به یک جدول اضافه می کند، بهره می بریم.

فایل aspx مربوطه:

```
<%@ Page Language="vb" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.IO" %>
<html>
  <head>
    <title>Inserting Image to a SqlServer</title>
    <script runat=server>
      Public Sub AddPerson(sender As Object, e As EventArgs)
        ' above code
      End Sub
    </script>
  </head>

  <body>

    <form enctype="multipart/form-data" runat="server">
      <input type="file" id="PersonImage" runat=server />
      <asp:Button Text="Add Person" OnClick="AddPerson" Runat=server />
   </form>

  </body>
</html>
```

جدول و زیرروال استفاده شده:

```
Drop Table Person
Go

Create Table Person
(
PersonID Int Identity,
PersonEmail Varchar(255),
PersonName Varchar(255),
```

```
PersonSex Char(1),
PersonDOB DateTime,
PersonImage Image,
PersonImageType Varchar(255)
)


Drop Proc sp_person_isp
Go

Create Proc sp_person_isp
@PersonEmail Varchar(255),
@PersonName Varchar(255),
@PersonSex Char(1),
@PersonDOB DateTime,
@PersonImage Image,
@PersonImageType Varchar(255)
As
Begin
  Insert into Person
    (PersonEmail, PersonName, PersonSex,
     PersonDOB, PersonImage, PersonImageType)
  Values
    (@PersonEmail, @PersonName, @PersonSex,
     @PersonDOB, @PersonImage, @PersonImageType)
End

Go
```

بازیابی تصاویر از SQL Server در ASP.NET:

در مقایسه با اضافه کردن تصویر، بازیابی آن بسیار ساده است. تنها چیز جدید برای این قسمت استفاده از متد BinaryWrite موجود در شیء Response است.

کدهای مورد استفاده جهت بازیابی تصاویر از SQL Server:

```
Public Sub Page_Load(sender As Object, e As EventArgs)

  Dim myConnection As New SqlConnection(
ConfigurationSettings.AppSettings("ConnectionString"))
  Dim myCommand As New SqlCommand("Select * from Person", myConnection)

  Try
    myConnection.Open()
    Dim myDataReader as SqlDataReader
    myDataReader =
myCommand.ExecuteReader(CommandBehavior.CloseConnection)

    Do While (myDataReader.Read())
```

```
        Response.ContentType = myDataReader.Item("PersonImageType")
        Response.BinaryWrite(myDataReader.Item("PersonImage"))
      Loop

      myConnection.Close()
      Response.Write("Person info successfully retrieved!")
    Catch SQLexc As SqlException
      Response.Write("Read Failed : " & SQLexc.ToString())
    End Try
End Sub
```

تمام کار این مرحله اجرای یک دستور SQL و تکرار آن برای همه ردیفهاست. قبل از نمایش تصویر، نوع آن هم مشخص می شود.

بررسی کوکی‌ها (Cookies) و جلسات (Sessions) در ASP.NET

در این مقاله ابتدا به بررسی کوکی‌ها (Cookies) پرداخته و سپس جلسات (Sessions) را بررسی خواهیم کرد. به دلیل نزدیکی بسیار زیاد این دو مفهوم تصمیم گرفتم همه آنها را در یک مقاله جمع‌آوری کنم. اگر مفاهیم فوق را به درستی درک کنید و طرز استفاده از آنها را یاد بگیرید، می توانید به قدرت برنامه‌های کاربردی وب خود بیفزایید. اما استفاده نابجا از این مفاهیم در وب سایتهای مختلف می تواند به شدت روی عملکرد سایت تاثیر گذاشته و از مخاطبان و کارایی آن بکاهد. در نتیجه با مطالعه این مقاله در مورد این دو مفهوم، باید بتوانید تصمیم بگیرید که کدامیک برای وب سایت شما مناسب‌تر می باشد. در هر حال مثال‌هایی را در این مقاله خواهید دید که به طور حتم به شما کمک زیادی خواهند کرد. لازم به ذکر است که بیشتر سعی در انتقال مفاهیمی کرده‌ام که پایه کار نمی باشند و سعی کردم به بیان نکاتی بپردازم که راحت پیدا نمی‌شوند.

کوکی‌ها (Cookies)

استفان والتر در کتاب ASP.NET Unleashed در ابتدای بخش کوکی‌ها اینگونه می‌گوید: "پروتکل HTTP هیچ امکانی را در اختیار وب سرور قرار نمی‌دهد تا بتواند به کمک آن تشخیص دهد درخواست جدید از همان مرورگری صادر شده که در خواست قبلی را فرستاده یا از مرورگر دیگری آمده است. از این جهت به HTTP صفت ناپایداری (Stateless) را میدهند. از نقطه نظر وب سرور هر درخواستی که برای دریافت یک صفحه صادر شده است از طرف کاربری جدید ارسال شده است." این به طور قطع آن چیزی نیست که ما می خواهیم! وقتی می‌خواهیم اطلاعات کاربر را در هر صفحه به او نشان بدهیم (از قبیل شناسه کاربری، کلمه عبور، سبد خرید و...) باید بتوانیم وضعیت آن را حفظ کنیم یکی از راههای بسیار خوب در این زمینه استفاده از کوکی‌ها می‌باشد.

اولین بار Netscape کوکی‌ها را در مرورگر خود به کار برد و به تدریج کنسرسیوم وب (W3C) نیز آن را پذیرفت و امروزه اکثر مرورگرها از کوکی‌ها پشتیبانی می کنند. بر اساس مستندات اولیه Netscape، یک کوکی نمیتواند حجمی بیشتر از ۴ کیلوبایت داشته باشد و با بستن صفحه مرورگر کوکی‌ها نیز از بین می روند. البته نگران نباشید اینها کوکی‌هایی هستند که پارامتر Expires آنها تنظیم نشده است. اما اگر این پارامتر را تنظیم کنید، کوکی‌ها باقی مانده و دائمی می شوند. اما تا کی؟ تا آن تاریخی که در خاصیت Expires تنظیم کرده‌اید. مرورگرهایی که می توانند با کوکی‌ها کار کنند دارای چند فایل ویژه می‌باشند که در ویندوز به آنها فایل‌های کوکی و در مکینتاش فایل‌های جادویی می گویند. کوکی‌ها از طریق هدرهای HTTP بین مرورگر و سرور جابجا می شوند. سرور با استفاده از هدر Set Cookie یک کوکی جدید ایجاد کرده و در درخواست‌های بعدی این کوکی به سرور فرستاده می شود.

در مقاله‌ای از سایت ASPFREE در مورد خواندن و نوشتن کوکی‌ها اینگونه نوشته شده است: "برای نوشتن کوکی یک شیئ جدید HttpCookie بسازید و مقدار یک رشته را به آن اختصاص دهید (به خاصیت Value آن) و سپس متد ()Add را در Response.Cookies فرا بخوانید. شما همچنین می توانید مقدار Expires را به یک مقدار تاریخ تغییر دهید تا زمان انقضاء برای کوکی‌تان تایین کرده باشید."

باید توجه داشته باشید که کوکی‌ها فقط مقادیر رشته‌ای را ذخیره می کنند و برای نوشتن مقادیر دیگر در کوکی‌ها باید هر آنها را به یک رشته تبدیل کنید. این کد از سایت CodeToad برای یادگیری نحوه استفاده کوکی‌ها بسیار مناسب می باشد:

```
Using System.Web;
```

```
// نوشتن
Response.Cookies["BackgroundColor"].Value = "Red";

// خواندن
Response.Write(Request.Cookies["BackgroundColor"].Value);
```

به دلایل امنیتی شما می‌توانید فقط کوکی‌هایی را بخوانید که از یک دامنه آمده باشند. همچنین ممکن است شما نیاز به کوکی‌هایی داشته باشید که چند آیتم را در خود نگهداری کنند، یک مثال برای این کار در زیر می بینید:

```
HttpCookieCollection cookies = Request.Cookies;

for (int n = 0; n < cookies.Count; n++) {
    HttpCookie cookie = cookies[n];
    Response.Write("<hr/>Name: <b>" + cookie.Name + "</b><br />");
    Response.Write("Expiry: " + cookie.Expires + "<br />");
    Response.Write("Address1: " + cookie.Address1+ "<br />");
    Response.Write("Address2: " + cookie.Address2+ "<br />");
    Response.Write("City: " + cookie.City+ "<br />");
    Response.Write("Zip: " + cookie.Zip+ "<br />");
}
```

یک مثال درباره کوکی‌های تو در تو به زبان VB.NET:

```
If Request.Form("savecookie") = "Yes" and ValidLogin = "Yes" Then
   Response.Cookies("member")("username") = Request.Form("username")
   Response.Cookies("member")("password") = Request.Form("password")
   Response.Cookies("member").Expires = DATE + 365
End if
```

جدول زیر بعضی از خصوصیات پیشرفته کوکی‌ها را نمایش می‌دهد:

خاصیت	توضیحات
Domain	دامنه‌ای که محدوده کوکی را تعیین می کند.
Path	مسیر منتسب به کوکی.
Secure	مقدار بولینی که تعیین می کند آیا کوکی باید فقط روی یک اتصال رمز شده ارسال گردد یا نه؟
HasKeys	مقدار بولینی که تعیین می کند که آیا کوکی مربوط به یک کوکی دیکشنری است یا نه؟

بطور خلاصه، کوکی‌ها چه خوبیهایی دارند؟ اشغال فضا روی کلاینت که باعث کاهش ترافیک و اشغال فضا روی سرور می شود و اطمینان کار. کوکی‌ها چه بدیهایی دارند؟ بعضی از مرورگرها از کوکی‌ها پشتیبانی نمی کند، بعضی از کاربران کوکی‌ها را پاک می کنند یا نمی پذیرند و این که فقط داده نوع رشته‌ای را ذخیره می کنند.

جلسات (Sessions)

فریم ورک دات نت برای رد گیری حرکت کاربر ما را تنها نگذاشته و یک امکان خوب به نام Session State را در اختیار ما قرار داده است. به طور پیش فرض وقتی کاربر اولین بار صفحه‌ای را از یک وب سایت ساخته شده با ASP.NET درخواست می کند یک کوکی جلسه به نام ASP.NET_SessionID ساخته شده و به مرورگر او ارسال میشود. با این کار ASP.NET قادر به پیگیری کاربر شده و میتواند در درخواست‌های بعدی او را شناسایی کند.

بر این اساس در ASP.NET یک شیء به نام Session قرار داده شده است که میتوانید از آن برای نگهداری اطلاعات مربوط به هر کاربر استفاده کنید. برای مثال دستور زیر یک آیتم با نام MyItem ایجاد کرده و Hello را به آن نسبت میدهد:

```
Session("MyItem")="Hello!"
```

هنگام کار با Sessionها باید به نکات زیر توجه کنید:

۱. هر Session اگر کاربر مرورگر را ببندد یا ۲۰ دقیقه از سرور درخواست نکند از بین می رود.

۲. Session هر کاربر جدا از Session بقیه کاربران است.

۳. در Session بر خلاف کوکی‌ها می توان شیئ هم ذخیره کرد.

جدول زیر بعضی از خصوصیات و متدهای شیئ Session را نمایش میدهد:

خاصیت/متد	توضیحات
Remove	پاک کردنSession
RemoveAll	پاک کردن تمامSession ها
SessionID	IDمنحصر به فرد جلسه فعلی را برمیگرداند.
Abandon	Sessionفعلی را خاتمه میدهد. اگر کاربر پس از دستور فوق درخواست یک صفحه جدید کند به عنوان کاربر جدید در نظر گرفته می شود.
TimeOut	تغییر مهلت پیش فرض ختم جلسه. این خصوصیت هر عددی که باشد بعد از همان قدر دقیقه اگر کاربر درخواستی به سرور نفرستد Session ختم می شود.

نکته: از طریق فایل web.config نیز می‌توان مهلت ختم جلسه را تغییر داد:

```
<configuration>
  <system.web>
    <sessionstate timeout="60" />
  </system.web>
</configuration>
```

Eventها یا وقایع جلسه‌ها دو مورد هستند: Session_Start و Session_End. که Session_Start وقتی رخ می دهد که جلسه آغاز و Session_End وقتی رخ می دهد که جلسه خاتمه پیدا کند. این Eventها را باید در فایل Global.asax تعریف کرد.

در زیر یک مثال عملی از این رویدادها را خواهید دید:

```
<html>
  <head>
    <title>SessionCount.aspx</title>
    <Script Runat="Server">
      Sub Page_Load()
        lblSessionCount.Text = Application("SessionCount")
      End Sub
    </Script>
  </head>

  <body>

  Current Sessions:
  <asp:Label ID="lblSessionCount" Runat="Server" />

  </body>
</html>
```

Default.aspx

```
<Script Runat="Server">
  Sub Session_Start()
    If Application("SessionCount") Is Nothing Then
      Application("SessionCount") = 0
    End If

    Application("SessionCount") += 1
  End Sub

  Sub Session_End()
   Application("SessionCount") -= 1
  End Sub
</Script>
```

Global.asax

بطور کلی برای نگهداری مقادیر Sessionها در ASP.NET سه روش وجود دارد: درون پروسه (In Process)، ذخیره در سرویس ویندوز و ذخیره در SQL Server.

Sessionها به طور پیش فرض در داخل پروسه مدیریت می شود و تمام آیتمهایی که در Sessionها می‌سازیم در همان پروسه وب سرور ذخیره می شوند. مهمترین مشکل این روش این است که اگر به هر دلیل سرور از کار بیفتد و یا Web Applicationما دستکاری شود، تمام داده‌ها از بین میرود و از طرف دیگر بسط پذیری را در سایت محدود می کند و نمی توان آن را به اشتراک گذاشت.

اما با استفاده از تکنیک ذخیره در پایگاه داده SQL Server می‌توان حتی در صورت از کار افتادن سرور نیز اطلاعات را حفظ کرد. تعریف اشیای ضروری در SQL Server به منظور مدیریت داده‌های جلسه با اجرای بچ فایل InstallSqlState.sql صورت می گیرد. بعد از این کار باید فایل web.config را نیز به شکل زیر تغییر داد:

```
<configuration>
  <system.web>
    <sessionstate
      mode="SqlServer"
      sqlConnectionString="Server=127.0.0.1;UID=sa;Pwd=YourPassword" />
  </system.web>
</configuration>
```

باز کردن پنجره Popup در ASP.NET با استفاده از Code Behind

قطعا در بسیاری از Applicationهای شما مواردی پیش آمده است که شما نیاز داشتید صفحه‌ای خاص را در یک پنجره Popup باز کنید. این کار به سادگی با استفاده از یک خط کدنویسی ساده جاوا اسکریپت امکان پذیر است. حتی در صورتی که فایلی که کد جاوا اسکریپت شما در آن قرار می گیرد یک فایل Server Side باشد نیز به علت اینکه در تمامی زبانهای Server Side پیش از ASP.NET کلیه کدها به صورت Block Script و لابلای کدهای HTML قرار می گرفت منعی در استفاده از بلاکهای جاوا اسکریپت نیز وجود نداشت. با ظهور ASP.NET و قابلیت قدرتمند آن یعنی Code Behind برنامه نویسان از کلنجار رفتن با کدهای تو در توی طولانی رها شده و کلیه کدهای Server Side خود را در Code Behind می نویسند و حاصل کار آنها یک فایل aspx. است که در آن اثری از کدنویسی غیر از تگهای HTML (و تگهای توسعه یافته ASP.NET) دیده نمی‌شود. حال اگر شما بخواهید یک Popup باز کنید چه می‌کنید؟ آیا تابع جاوا اسکریپت بازکننده Popup را لابلای کدهای HTML صفحه خود می‌نویسید؟ این روش به نظر صحیح نمی آید. چرا که ما خود را از قابلیت خوانانویسی کدهای HTML محروم کرده و باز قسمتی از برنامه نویسی خود را به درون فایل اصلی خود انتقال داده‌ایم. چه می توان کرد؟ آیا راهی برای اینکه ما با استفاده از کدنویسی در Code Behind یک Popup باز کنیم هست؟ پاسخ مثبت است. در ادامه مقاله به این موضوع می پردازیم: در این روش در واقع ما کدهای جاوا اسکریپت خود را به صورت یک attribute به کنترل مورد نظر خود Add می کنیم. اما برای اینکه برای هر کنترل کدنویسی را تکرار نکنیم این قسمت را درون یک ساب روتین قرار می دهیم:

```
using System.Web.UI.WebControls;

public void OpenPopUp(WebControl opener, string PagePath) {

   string clientScript;
   clientScript = "window.open('" + PagePath + "')";
   opener.Attributes.Add("OnClick", clientScript);
}
```

طرز کار بسیار ساده است! مقادیر ورودی یک کنترل و صفحه‌ای است که باید باز شود. کد جاوا اسکریپت در یک متغیر رشته‌ای به نام clinetString قرار داده می‌شود و با استفاده از متد Add به کنترل اضافه می‌شود. متد Add دو مقدار دریافت می‌کند: ابتدا نام property و یا event و سپس مقدار آن که پس از Generate شدن صفحه به تگ HTML تولید شده اضافه می‌شود.

حال شما می‌توانید به راحتی به صورت زیر از این روتین استفاده کنید:

```
OpenPopup(Button1, "http://www.google.com");
```

که در آن Butoon1 یک کنترل از نوع asp:button است. طرز کار بدین صورت است که پنجره بلافاصله پس از فراخوانی روتین باز نمی‌شود بلکه کنترل شما را برای رویداد مورد نظر تنظیم می‌کند و در هنگام وقوع رویداد (مثلا کلیک کردن روی دکمه) روتین اجرا می‌شود.

با کمی کدنویسی اضافه می‌توان ظاهر پنجره باز شده را نیز کنترل کرد. برای مثال شما می توانید طول و عرض پنجره، عنوان آن، اسکرول بار و... را تنظیم کنید:

```
public shared void OpenPopUp(WebControl opener, string PagePath,
string windowName, int width, int height) {
```

```
    string windowAttribs;
    string clientScript ;

    windowAttribs = "width=" + width + "px," + "height=" + height + "px,"
      "left='+((screen.width -" & width & ") / 2)+'," +
      "top='+ (screen.height - " & height & ") / 2+'";";

    clientScript = "window.open('" + PagePath + "','" + windowName + "','" +
      windowAttribs + "'); return false;";

    opener.Attributes.Add("OnClick", clientScript);
}
```

مزیت استفاده از این روش این است که به علت اینکه شما این تابع را یک بار و آن هم در کلاس Code Behind و یا قرار می‌دهید می‌توانید به راحتی خطاهای برنامه نویسی خود را قبل از اینکه اجرای کد خود را آغاز نمایید، پیدا کنید.

آشنائی با فرمهای وب در ASP.NET

به عنوان یک برنامه نویس ویژوال بیسیک شما می توانید برنامه های تحت اینترنت نیز بنویسید. به طور معمول برنامه نویسان ویژوال بیسیک به سمت ASP که یک تکنولوژی از مایکروسافت است متمایل هستند. دلیل این امر هم شباهت میان VB و VBScript می باشد. بزرگترین ایرادی که ASP کلاسیک دارد نداشتن یک محیط ویژوال مانند فرمهای معمولی بیسیک است. مایکروسافت با Visual InterDev سعی کرد این کمبود را جبران کند اما چندان موفق نبود. بالاخره در ویژوال بیسیک دات نت ترکیبی از InterDev و ویژوال بیسیک وجود دارد و امکانات فرمهای ویژوال بیسیک را برای اینترنت نیز فراهم می کند.

Web Form ها یکی از اجزای تکنولوژی ASP.NET است که به برنامه نویس های اکثر زبان ها این امکان را می دهد که یک قالب ویژوال با HTML و یک محیط برنامه‌نویسی تحت سرور با کدهای پیشرفته داشته باشند.

Web From ها در عمل

بهترین راه برای فراگیری این تکنولوژی یک مثال عملی از آن است. پس از مثال معروف Hello World برای شروع استفاده می کنیم.

آماده سازی محیط

قبل از شروع ابتدا باید نرم‌افزارهای مورد نیاز را از روی لیست زیر نصب کنید. اگر ویژوال استودیو دات نت را به شکل کامل و بر روی ویندوز ۲۰۰۰ یا اکس پی نصب کردید احتیاج به مراحل زیر ندارید.

سیستم عامل شما باید حتماً از نوع ان تی باشد، ویندوز ۲۰۰۰ (سرور یا Professional)، ویندوز اکس پی Professional و یا ان تی سرور ۴.
باید .NET Framework بر روی سروری که می خواهید با آن کار انجام دهید یا برنامه شما بر روی آن اجرا خواهد شد نصب شده باشد. اگر ویژوال استودیو دات نت را نصب کرده اید مشکلی در این مرحله ندارید.
مایکروسافت توصیه کرده که سیستم فایل هارد دیسک سرور شما بهتر است NTFS باشد، هم به دلیل مسائل امنیتی و هم سرعت بیشتر.

Hello World

در فرم مخصوص ایجاد یک پروژه جدید ASP.NET Web Application را انتخاب کنید و نام آن را HelloWorld قرار دهید. دقت کنید که مکان ذخیره پروژه http://localhost باشد.

سپس بر روی کلید OK کلیک کنید تا یک Solution جدید ایجاد گردد. به طور قراردادی ویژوال استودیو یک Web Form با نام WebForm1.aspx ایجاد می کند. دقت کنید که پسوند فایل چه تغییری کرده است.

وقتی بر روی کلید OK کلیک می کنید چند عمل در پشت صحنه انجام می شود. به غیر از ایجاد کردن یک شاخه در دایرکتوری Visual Studio Projects، ویژوال استودیو یک Web Application نیز در سروری که انتخاب کرده اید ایجاد می کند. بر روی سرور، ویژوال استودیو دات نت:

یک دایرکتوری با نام پروژه در شاخه inetpub/wwwroot ایجاد می کند.
این دایرکتوری را به عنوان یک IIS Application معرفی کرده و اجازه اجرای Script را بر روی آن می دهد.

اگر FrontPage Server Extensions را نصب کرده باشید یک FrontPage Web ایجاد می کند تا با FrontPage هم بتوانید به آن دسترسی داشته باشید.
می توانید همانگونه که با فرم های معمولی ویژوال بیسیک کار می کردید از Web Formای که جلوی شما است استفاده کنید، یعنی به شما امکان استفاده از Toolbox و استفاده از کامپوننت های درون آن بر روی Web Form داده شده است. یک Label را از Toolbox برداشته و بر روی قسمت بالای فرم قرار دهید و خاصیت Text آن را به Hello World تغییر دهید.

برای این مثال تمام کاری که لازم بود انجام شود را انجام دادیم. حالا می توانیم برنامه را اجرا کنیم. قبل از اینکار از Toolbar و در قسمت Solution Configuration به جای Release،Debug را انتخاب کنید. حالا بوسیله کلید F5 برنامه را اجرا کنید. اگر هیچ مشکلی در سیستم نباشد باید صفحه مرورگر باز شود و فایل WebForm1.aspx نمایش داده وبر روی آن Hello World نوشته شود.

بر روی صفحه مرورگر کلید سمت راست موس را بزنید و View Source را انتخاب کنید تا ببینید چه مطالبی در سورس این صفحه آمده است. همانطور که می بینید کدهایی به HTML است که بوسیله فایل aspx در زمان اجرا ایجاد شده است.

همانطور که می بینید یک HTML Form در این متن دیده می شود در حالی که ما چنین چیزی را اضافه نکرده بودیم، درباره این مساله در ادامه توضیح خواهم داد. Labelای که اضافه کرده بودیم در تگ Span قرار دارد. تگ Span مانند یک Container برای Label ما است و اطلاعات آن را در خود نگهداری می کند. به ویژوال استودیو دات نت باز می گردیم.

همانطور که دیدید Web Form ها خیلی شبیه فرم های معمولی ویندوز هستند. در Web From جدیدی که می سازیم این خاصیت را بیشتر امتحان می کنیم. در برنامه Hello World که ایجاد کردیم تنها یک Web Form داشتیم: WebForm1.aspx. یک Web Form دیگر می سازیم تا کارهای بیشتری با آن انجام دهیم.

منوی Project | Add Web Form را انتخاب کنید. در فرمی که باز می شود Web Form را انتخاب کنید و مطمئن شوید که نام آن WebForm2.aspx است. (قبل از این کار ویژوال استودیو را از حالت اجرای برنامه خارج کنید)

بر روی Open کلیک کنید تا WebForm2.aspx در Solution ایجاد شود. بر روی WebForm2.aspx در Solution Explorer دو بار کلیک کنید تا مطمئن باشید که فرمی که تازه ایجاد کرده اید فعال است. مانند مثال قبلی یک Label بر روی فرم قرار دهید، سپس یک Button در زیر آن قرار دهید و اندازه هر دو را یکسان کنید. بر روی Label کلیک کنید و از پنجره Properties خاصیت ID را انتخاب کنید و آن را به lblText تغییر دهید. سپس بر روی کلید کلیک کنید و ID آن را به btnSubmit تغییر دهید. بر روی کلید یک بار کلیک کنید، سپس کلید Enter را بزنید تا به قسمت نوشتن کد برای این کلید وارد شوید.

در ASP.NET هر کدام از کنترل ها، کدی در پشت صحنه برای خود دارند. همانطور که مشاهده می کنید یک روال با نام btnSubmit_Click وجود دارد که هنگامی که بر روی کلید کلیک می شود اجرا می شود. کدی که در این روال نوشته شده باشد در سرور اجرا می شود و نه در مرورگر کامپیوتر کاربر. کد زیر را در روال مورد بحث بنویسید:

```
lblText.Text = "Hello World"
```

همانطور که مشاهده کردید IntelliSense وارد عمل شده و وقتی بعد از lblText، نقطه را تایپ کردید لیستی از خواص و متد های مربوط به Label را به شما نمایش داد. این خاصیت در

InterDev هم وجود دارد ولی در ویژوال استودیو دات نت از امکانات بیشتر و لیست پرمحتواتری برخوردار است.

پنجره کد را ببندید و به قسمت طراحی Web Form بروید، خاصیت Text کلید را به Submit تغییر دهید. حالا برنامه را امتحان می کنیم. اگر سعی کنید تا برنامه را بوسیله کلید F5 اجرا کنید دوباره WebForm1.aspx نمایش داده خواهد شد، زیرا که این فرم، فرم ابتدایی در پروژه ما است. برای اینکه WebForm2.aspx به فرم ابتدایی تبدیل شود در پنجره Solution Explorer بر روی WebForm2.aspx کلید سمت راست موس را بزنید و سپس Set As Start Page را انتخاب کنید. حالا می توانید برنامه اجرا کنید.

Web Form جدید، WebForm2.aspx در مرورگر اینترنت نمایش داده می شود در حالی که بر روی آن یک Label و یک کلید وجود دارد. بر روی کلید کلیک کنید تا متنی که تایپ کرده بودید در Label نمایش داده شود. همانطور که می بینید برنامه مانند فرم های معمولی ویندوز اجرا می شود.