فصل دوم :
آشنایی با مقدمات زبان برنامه نویسی شی گرای C# و ایجاد اولین برنامه ASP.NET

مقدمه :

در این فصل با اصول پایه ای C# و برنامه نویسی آن برای ایجاد صفحات ASP.NET آشنا می شویم. اگر با زبان C آشنایی دارید ، فصل جاری فصلی ساده و بسیار روانی برای شما خواهد بود و در غیر اینصورت با کمی پشتکار مشکل حل خواهد شد. این مرور بسیار کاربردی و به دور از هرگونه فلسفه بافی می باشد و خیلی سریع کد نوشتن را شروع خواهیم کرد. بدیهی است که فقط برای آشنایی کامل با اساس و شالوده ی زبان سی شارپ به کتابی کامل نیاز می باشد و نه یک فصل چند صفحه ای .

آشنایی با فضاهای نام (NameSpaces) :

فضاهای نام روشی برای مدیریت کد نویسی هستند. برای مثال آنها ایجاد شده اند تا تداخلی بین نام های توابع در برنامه شما رخ ندهد. این مساله در پروژه های بزرگ خود را نشان می دهد و ممکن است دو آیتم در یک پروژه نام های یکسانی را پیدا کنند. بدین وسیله این شانس تصادم و تداخل کاهش پیدا می کند. برای ایجاد یک فضای نام به صورت زیر عمل می شود:

```
namespace anyName
{
.........
   Class anyClassName
{
  ........
}

.......
}
```

یکی از فضاهای نام پایه ای در دات نت فریم ورک ، فضای نام System می باشد. برای استفاده از آن می توان از کد زیر کمک گرفت :

```
using System;
```

تمام فضاهای نام به صورت پیش فرض public می باشند و در خارج از کد شما قابل دسترسی هستند. روش استفاده از آنها به صورت زیر است:

```
ProjectName.NameSpace.ClassName.MemberName
```

برای مثال اگر آرایه ای را در دات نت بخواهیم مرتب و سورت کنیم حداقل دو راه برای نوشتن وجود دارد:

```
System.Array.Sort(strArray);
```

ویا

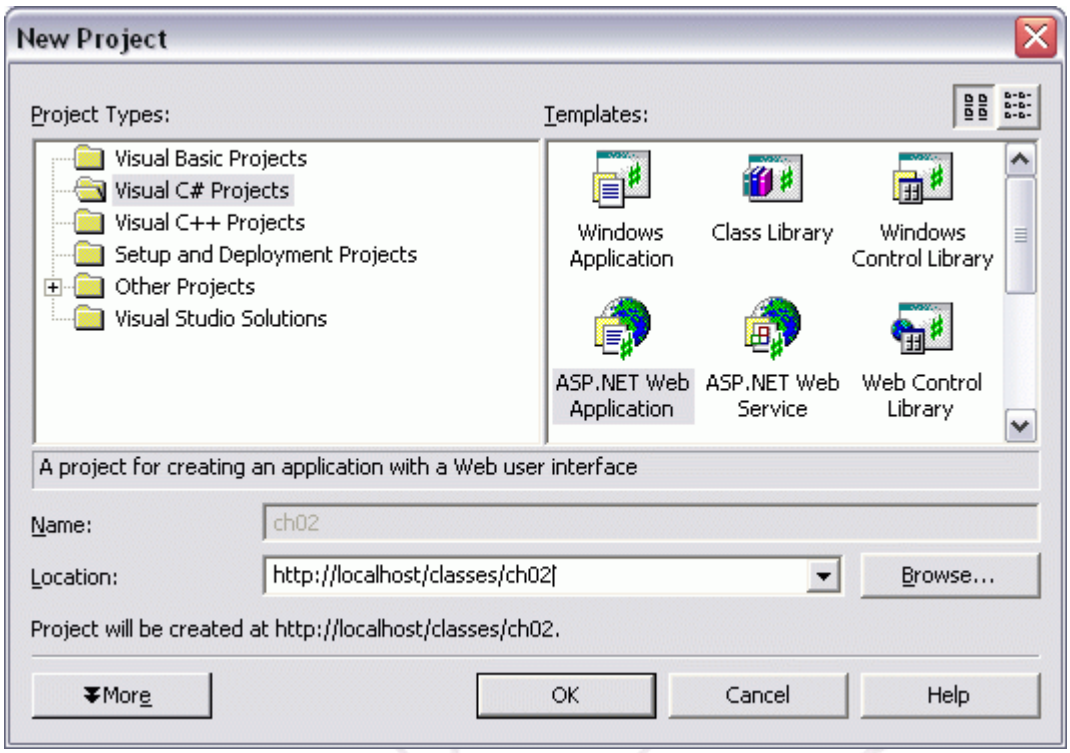
```
using System;
Array.Sort(strArray);
```

بدین صورت خلاصه نویسی در کد صورت می گیرد.

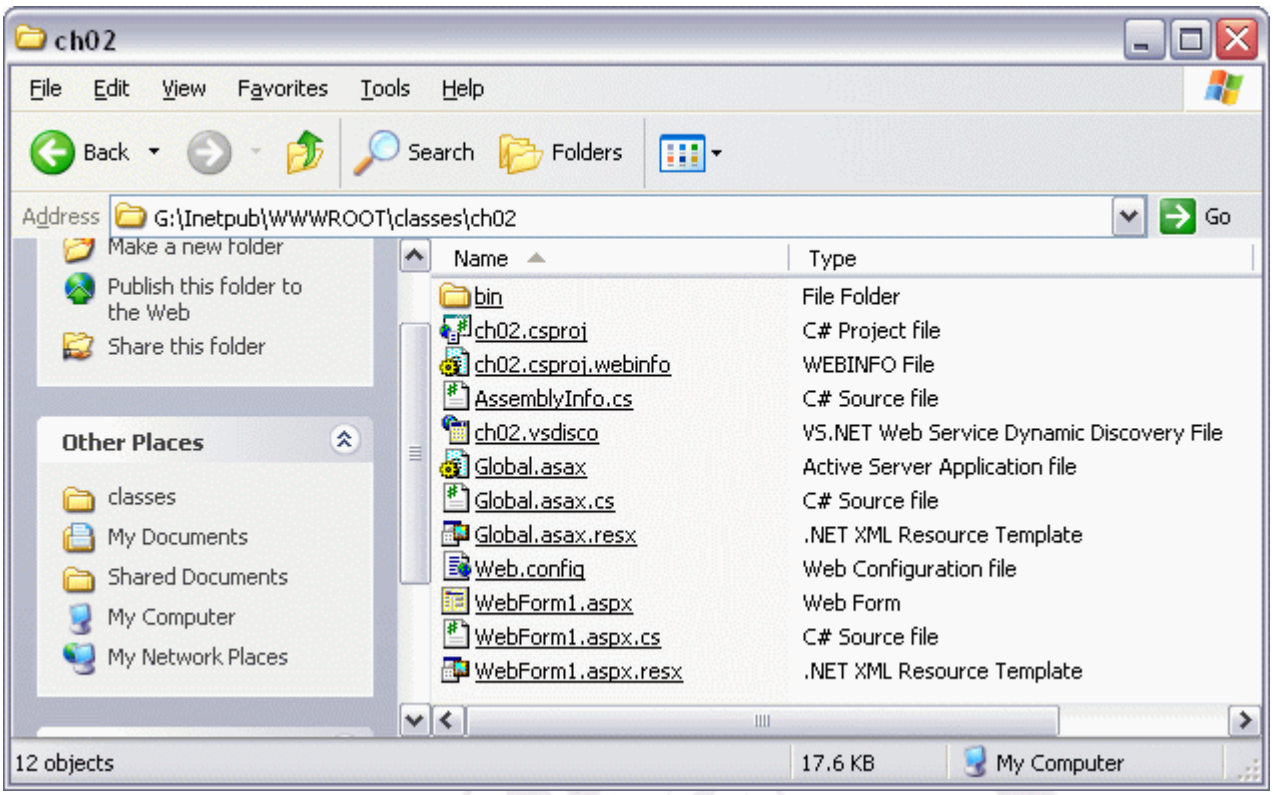
برنامه اول : مروری بر نحوه استفاده از NameSpace ها ، تعریف متغیر و مقدار دهی اولیه به آن ، توابع و خواص ها.

ویژوال استودیو دات نت را اجرا کنید و در صفحه ی باز شده روی دکمه New Project کلیک نمایید تا بتوان یک پروژه جدید ASP.NET را شروع کرد. از پنل Project گزینه Visual C# Project انتخاب کنید و از پنل سمت چپ گزینه ASP.NET Web Application را برگزینید. در قسمت Location می توانید

نامی دلخواه را در دایرکتوری Home مشخص کنید و یا اگر دایرکتوری مجازی درست کرده اید ، آنرا در اینجا انتخاب نمایید. پس از مشخص کردن کار، روی دکمه Ok کلیک کنید تا فایل های اولیه پروژه ساخته شوند (شکل های ۱و۲) .



شکل ۱- آغاز کردن یک پروژه جدید ASP.NET با استفاده از VC#



شکل ۲- فایل هایی که به صورت اتوماتیک توسط VS.NET ایجاد می شوند.

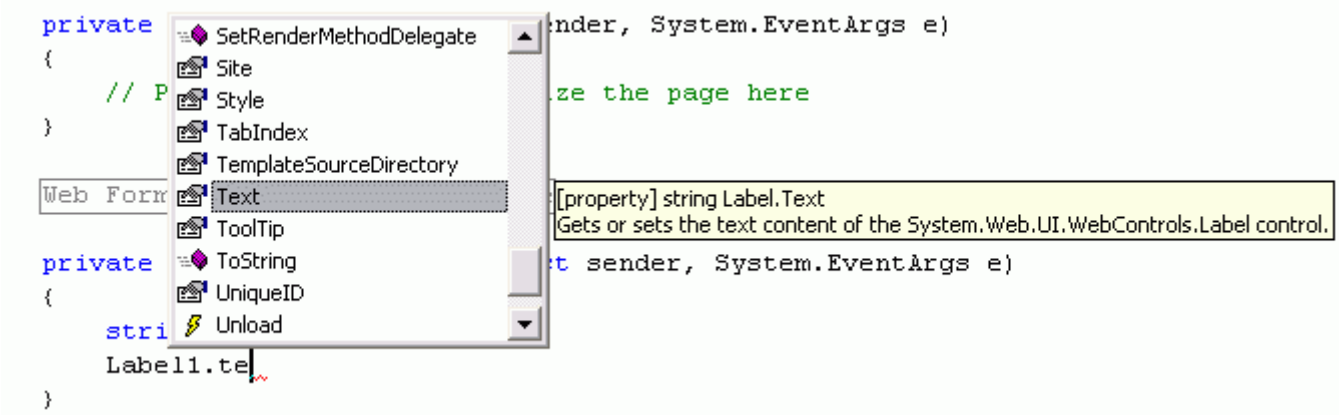
از ToolBox کنار صفحه یک Label و یک دکمه (Button) را روی فرم قرار دهید (شکل ۳).

شکل ۳- قرار دادن یک دکمه و یک لیبل بر روی فرم.

حالا روی دکمه دوبار کلیک کنید تا بتوانیم در تابعی که در هنگام رخ دادن رویداد کلیک شدن بر روی دکمه صدا زده می شود بتوانیم کد بنویسیم. اگر به صفحه ی باز شده که به آن Code Behind هم می گویند دقت کنید به صورت پیش فرض یک سری از فضاهای نام مفید و لازم در این سورس گنجانده شده است .

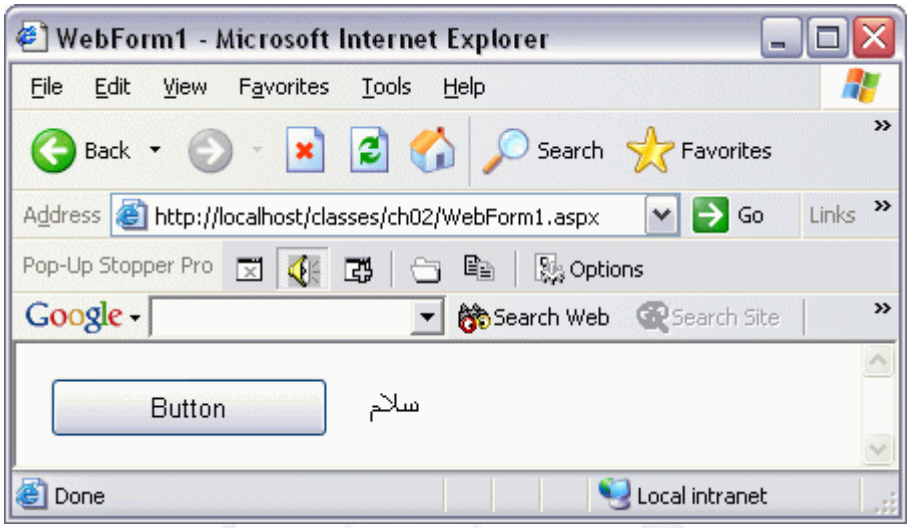
می خواهیم هر بار کاربر روی این دکمه کلیک کرد به او جمله ی " سلام ! این اولین برنامه ی من است! " را نشان دهد. برای اینکار ، فقط برای مرور یک سری از اصول ، طولانی ترین راه انتخاب می شود.

۱- یک متغیر از نوع string به نام strText تعریف کنید. بهتر است نوع متغیر به صورت خلاصه در ابتدای نام متغیر ذکر شود.
۲- آنرا مقدار دهی اولیه کنید (برای مثال " سلام " و یا جمله ی بالا) .
۳- به راحتی می توان داخل آن فارسی نوشت. در C# تا متغیری را مقدار دهی اولیه نکنید نمی توان از آن استفاده کرد.
٤- می خواهیم به خاصیت Text مربوط لیبلی که روی فرم گذاشته ایم این متغیر را نسبت دهیم. نام لیبل یعنی Label1 را بنویسید به همراه یک نقطه در جلوی آن یک منو که تمام توانایی های این کنترل را نمایش می دهد باز خواهد شد (شکل ٤) . گزینه Text آنرا انتخاب کنید و متغیر فوق را به آن نسبت دهید (اگر با کامپایلرهای ویژوال کار کرده باشید ملاحظه می کنید که همه چیز مانند آنها می باشد) .



شکل ٤- منوی autocomplete نمایش دهنده انواع متدها ، خواص و ... مربوط به کنترل لیبل.

۵- حالا بر روی دکمه F5 کلیک کنید تا برنامه در مرور گر وب اجرا شود. با کلیک کردن بر روی دکمه ، سلام ، نمایش داده می شود. به آدرسی که در Address Bar اینترنت اکسپلورر نوشته می شود نیز دقت کنید.



شکل ۵- خروجی برنامه پس از کلیک کردن روی دکمه در مرورگر وب مایکروسافت.

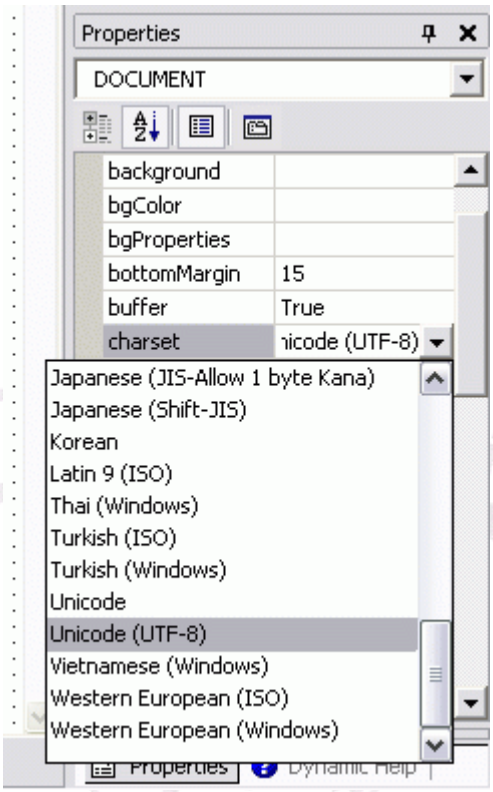
عمدا این مثال را انتخاب کرده ام . برای فارسی نویسی به نکات زیر باید توجه کرد:

۱- در محیط ویژوال استودیو بر روی Tab مربوط به WebForm1.aspx کلیک کنید. سپس در یک جای خالی روی صفحه کلیک نمایید تا صفحه ی خواص Document در سمت چپ محیط ویژوال استودیو نمایش داده شود. حالا روی منوی پایین افتادنی charset کلیک کنید و گزینه ی Unicode **(UTF-8)** را انتخاب کنید (شکل ٦) .

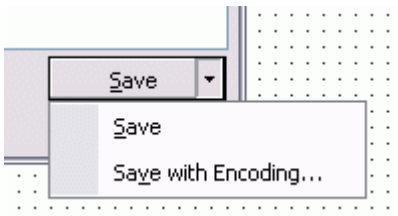
۲- باز هم کافی نیست! حالا از منوی فایل گزینه ی save as را انتخاب کنید. روی دکمه save یک علامت مثلث قرار دارد. روی آن کلیک نموده تا یک منوی جدید باز شود (شکل ۷). حالا روی گزینه Save with encoding کلیک نمایید و صفحه ی اخطار باز شده را تایید کنید و از صفحه ی ظاهر شده بعدی از آیتم Encoding گزینه ی**(** UTF-8 with signature **)** Unicode را برگزینید. حالا یک نفس راحت بکشید!

از این پس با خیال راحت و بدون هیچ نگرانی در مورد به هم ریختن فرمت فارسی برنامه می توانید برنامه ها را اجرا نمایید. (در منوی فایل گزینه advanced save options هم همین کار را انجام می دهد و این گزینه فقط در مورد سورس ها مهیا است)

کد کامل شده این قسمت به همراه فصل ارائه می شود و روال کار از این پس نیز به همین صورت خواهد بود.



شکل ٦- انتخاب یونیکد برای مشخص کردن charset صفحه خروجی .



شکل ٧- فرمت فایل نیز باید یونیکد انتخاب شود.

مروی بر مفاهیم بکار گرفته شده در کد ارائه شده :

فرمت کردن کد :

هر چقدر فرمت نوشتن کد شما بهتر باشد ، خواندن ، نگهداری و استفاده مجدد از آن ساده تر خواهد بود. دو مورد مهم دندانه دار نویسی و نوشتن توضیحات و یا کامنت ها می باشد. نوشتن توضیحات خصوصا در برنامه نویسی تیمی بسیار مهم و کار ساز است. در C# از // برای نوشتن کامنت استفاده می شود (مانند C++) و همانند C هنوز /* ..... */ نیز معتبر است.

نکته :

اگر دقت کرده باشید هنگامی که کرسر ماوس را روی هر آیتمی در منوی autocomplete نگه می دارید و یا آنرا انتخاب می کنید یک راهنمای کوچک نمایش داده می شود که در حقیقت کامنت مربوط به آن تابع می باشد. روش نوشتن چنین کامنت حرفه ای که در منوهای ویژوال استودیو ظاهر شود به صورت زیر است که بهتر است (!) قبل از هر تابع یا خاصیت یا کلاس و .... نوشته شود

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

تعریف متغیر و مقدار دهی به آن :

در هنگام تغیر یک متغیر ، ناحیه ای از حافظه برای ذخیره سازی داده ، اختصاص داده می شود. در C# برخلاف بعضی از زبان ها که نیازی به تعریف صریح متغیرها ندارند ، هم باید نوع متغیر را تعریف کنید و هم آنرا مقدار دهی اولیه نمایید.

البته اگر فراموش کردید که متغیری را مقدار دهی اولیه کنید مهم نیست! کامپایلر حتما آنرا به شما با یک خطا گوشزد خواهد کرد!

مقدار دهی اولیه یک متغیر از بسیاری از خطاهای زمان اجرا مانند جمع زدن دو متغیر بدون مقدار جلوگیری خواهد کرد.

استفاده از خواص :

شما به ویژگی های یک شیء با استفاده از خواص آن می توانید دسترسی پیدا کنید. یک property عضوی است که امکان دسترسی به ویژگی شیء یا کلاس را فراهم می کند. برای مثال طول یک رشته (string) ، سایز یک فونت ، عنوان یک فرم و نام یک مصرف کننده ، خاصیت هستند .

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

برنامه دوم : مروری بر آرایه ها و حلقه ها در C#

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

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

برای تعریف آرایه چندین راه مختلف وجود دارد

۱- تعریف آرایه ای از رشته ها و مقدار دهی اولیه آن.

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

۲- تعریف و مقدار دهی اولیه

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

که آرایه ای از نوع رشته ای به طول ۲ عضو با مقدار دهی اولیه ایجاد شده است. در این حالت نیازی به تعیین طول آن نمی باشد.

۳- روشی دیگر برای مقدار دهی اولیه

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

در دات نت کلاسی به نام Array وجود دارد که امکانات جالبی را برای کار با آرایه ها ارائه می دهد. برای مثال تابع Sort آن به سادگی یک آرایه را مرتب می کند.

برای حرکت بین اعضای یک آرایه با تعداد بالا به سادگی می توان از حلقه ی for ویا foreach استفاده کرد.

برای مثال

```
for( int i=0 ; i< strData.Length ; i++)
do some things!
```

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

برای نوشتن برنامه دوم یک آرایه با اعضایی دلخواه به طول ۱۰ تعریف کنید و سپس با استفاده از یک حلقه اعضای آنرا تک تک به مقادیر یونیکد معادل تبدیل نمایید و در یک آرایه دیگر ذخیره نمایید. برای تبدیل به یونیکد از کد زیر استفاده کنید:

سپس با استفاده از کلاس Array آنرا سورت کرده و سپس خروجی آنرا در یک TextBox نمایش دهید. برای اینکه تکست باکس از حالت یک خطی بیرون بیاید و چند خطی شود خاصیت TextMode آنرا به

MultiLine تغییر دهید. این مثال را به عنوان تمرین خودتان می توانید تکمیل کنید و سپس کد نوشته شده اتان را با source همراه فصل مقایسه کنید. نظرات و قسمت های نامفهوم احتمالی را می تـوان در forum مطرح کرد.

آشنایی بیشتر با کلاس ها ، متدها

متدها یا همان توابع در زبان C ، اعضای یک شیء یا کلاس هستند و مجموعه ای از یک سری از کارهـا را انجام می دهند. با خواص هم که در قسمت های قبل آشنا شـدید. بسـیاری از کـلاس هـای دات نـت فریم ورک متدها و یا توابع مفید حاضر و آماده ای را دارند. برای مثال کلاس DateTime ، متدی بـه نـام ToLongDatastring دارد که تاریخ را به صورت یک رشته طولانی بر می گرداند. برای تعریف یک کلاس همانطور که گفته شد به صورت زیر عمل می شود:

```
class myClassName
{
.........
}
```

برای تعریف یک متد یا تابع ابتدا سطح دسترسی به آن مانند public و private سـپس نـوع خروجـی تابع مانند void (هیچی) ذکر می گردد کـه داخـل ایـن پرانتزهـا مـی تـوان ورودی هـای تـابع یـا بقـولی آرگومان های ورودی را معرفی کرد. سپس تابع باید با } شروع و با یک { خاتمه یابد.

برای مثال :

```
public int myFunc( int x )
{
.......
}
```

هر تابعی می تواند صفر تا تعداد بیشماری آرگومان ورودی و صفر تـا تعـداد بیشـماری خروجـی داشـته باشد. بوسیله یک تابع می توان پیچیدگی کار را مخفی کرد و صـرفا بـا صـدا زدن نـام آن ، یـک سـری از

عملیات را انجام داد. گاهی از اوقات لازم می شود دو یا چند تابع با یک نام داشته باشیم بطوریکه پارامترهای ورودی یا مقادیر خروجی و یا نوع آرگومان های ورودی آنها با هم متفاوت باشد به این کار overloading می گویند.

برای تعریف خواص قاعده کلی به صورت زیر است :

```
AnyType propertyName
{

get;
set;

}
```

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

در این برنامه می خواهیم کلاسی را تعریف کنیم که در آن با استفاده از خواص ، دورشته را دریافت و توسط یک متد ساده ، این دو رشته به هم متصل گردیده و تعدادی کاراکتر از آن مطابق خاصیتی دیگر که آن طول این رشته جدا شده را از انتهای رشته مشخص می کند ، نمایش دهیم (بحث های مربوط به کنترل خطا در فصول دیگر مرور می شوند).

با توجه به توضیحات ارائه شده در مورد تعریف کلاس ها ، توابع و خاصیت ، این برنامه ساده بوده و می توان به سورس همراه فصل مراجعه کرد.

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

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

مبحث شیء گرایی در C# آنقدر مفصل است که می توان یک کتاب ۷۰۰ صفحه ای راجع بـه آن نوشـت! اگر باور ندارید یک سری به آدرسهای زیر بزنید (کتاب Thinking in C#) :

www.thinkingin.net
www.BruceEckel.com

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

نکته ای در مورد نحوه ی اجرای برنامه های همراه فصل:

برای اینکه بتوانید برنامه های همراه فصل را اجرا کنید باید ابتدا فصل اول را کامل مرور کرده باشـید و در ایجاد دایرکتوری مجازی مشکلی نداشته باشید و یا در WWWRoot یک دایرکتوری بـه نـام classes ایجاد کنید و به ازای هر فصل یک دایرکتوری به نام Chxx ایجاد نمایید که در آن xx شماره فصـل اسـت. سپس دایرکتوری های مثالهای همراه را در آن کپی نمایید. در هر حال هر روشی که برای شـما سـاده تـر است به آن صورت عمل کنید. بدیهی است که در غیر اینصورت هیچکدام از مثالهای همـراه را نمـی توانیـد اجرا نمایید.

تمرین

۱- یک پروژه ASP.NET جدید در VS.NET ایجاد کنید . می خواهیم برنامه ای بنویسیم که مساحت یک دایره را با دریافت شعاع آن بر روی صفحه نمایش دهد. برای اینکار یک کلاس باید به برنامه اضافه شود که شعاع را به صورت خاصیت دریافت نموده و مساحت را بصورت یک متد و یا تابع پیاده کند. از یک TextBox برای دریافت شعاع دایره و از یک لیبل برای نمایش مساحت استفاده می شود. یک دکمه هم برای دریافت رخداد مربوط به انجام عملیات لازم است.