什么是ASP？

**微软曾经的服务器端脚本技术ASP（Active Server Pages）**如今称为经典ASP。

ASP3.0是经典ASP的最后一个版本。

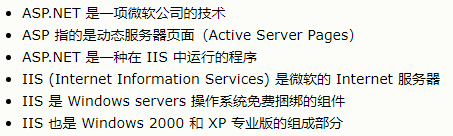
ASP.NET不是ASP

ASP.NET是下一代ASP，不是ASP的更新版本

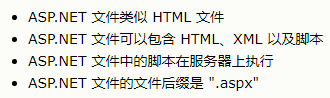
ASP.NET是服务器脚本编程的全新技术。

ASP.NET是微软.NET框架的主要部分

**ASP.NET是一种服务器端脚本技术，可以使（嵌入网页中的）脚本由Internet服务器执行**。



什么是ASP.NET文件？



**ASP.NET如何工作**？

* 当浏览器请求html文件时，服务器会返回该文件
* 当浏览器请求ASP.NET文件时，IIS会把该请求传递给服务器上的ASP.NET引擎。
* ASP.NET引擎会逐行地返回该文件，并执行文件中的脚本
* 最后，ASP.NET文件会以纯HTML的形式返回浏览器

.NET框架（.NET Framework）

.NET是.NET平台的基础

.NET框架是构建、开发及运行Web应用程序和Web服务的一种环境。

.NET框架包括3个主要的部分：

编程语言：

C#，J#等

服务器技术和客户端技术：

ASP.NET（Active Server Pages）

Windows Froms（Windows desktop solutions）

Compact Framework（PDA/Mobile solutions）

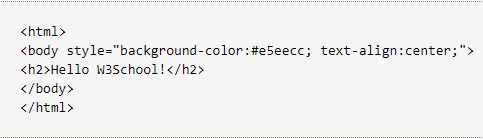
开发环境：

Vs 2017

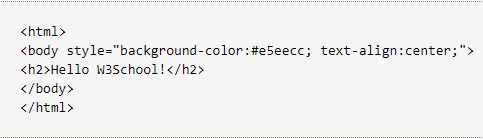
Web页面

* 简单的ASP.NET页面看上去类似普通的HTML页面
* 它如何工作：

从根本上讲，ASP.NET页面也HTML完全相同。



HTML页面



ASP.NET页面

* HTML页面的扩展名是.html。假如浏览器从服务器请求某张HTML页面，服务器不进行修改，就会把该页面发往浏览器。
* ASP.NET页面的扩展名是aspx。如果浏览器请求某张ASP.NET页面，那么在把结果发回浏览器之前，服务器首先会处理页面中的可执行代码。

上面的ASP.NET页面不包含任何可执行代码，因此也不会执行任何代码。在下面的例子中，我们将向页面添加一些可执行代码，以便向你演示静态HTML页面与动态ASP页面的不同之处。

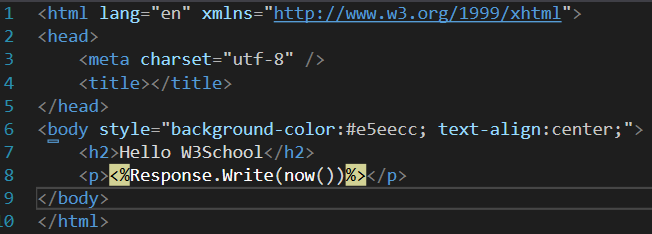
* 经典的ASP

Active Server Pages（ASP）已经流行很多年了。通过ASP，可以把可执行代码放置于HTML页面内部。

ASP.NET之前的ASP版本经常被称为经典ASP

* 用Classic ASP编写的动态页面

为了向你演示如何使用动态内容显示页面，我们向上面的例子添加一些可执行代码：



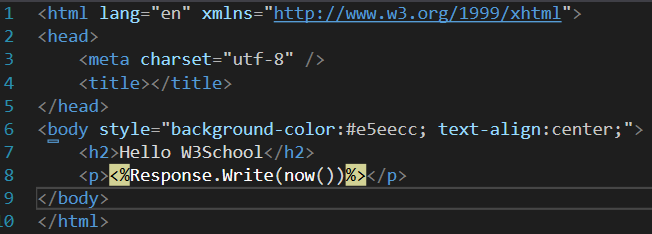
<%------%>标签内的代码在服务器上执行。

Response.Write是ASP代码，用来向HTML输出流文本

Now()是一个可返回服务器当前日期和时间的函数。

* 用ASP.NET编写的动态页面

下面的例子可把我们的例子显示为一个ASP.NET页面：



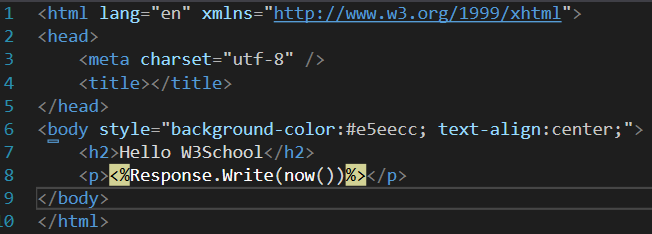
从上面的例子我们看不出ASP和ASP.NET的不同之处。

但是之后的学习，我们可以看到，**服务器控件**如何使ASP.NET比Classic ASP更加强大。

服务器控件

* 服务器控件是服务器可以理解的标签

Classic ASP的局限性



上面的代码能够反映出Classic ASP的局限性：**代码块必须放置在需要进行输出的位置**。

通过Classic ASP，我们是无法将可执行代码与HTML本身分离的。这使得页面难以阅读，也难以维护。

* 服务器控件

通过服务器控件，ASP.NET已经解决了上面所描述的问题。

服务器控件是可被理解的标签。

有三种类型的服务器控件：

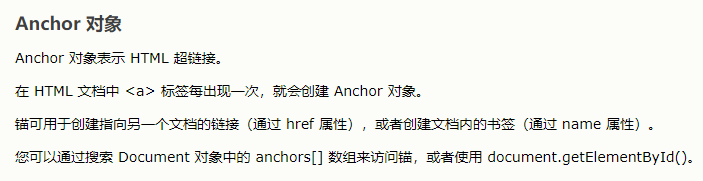
* + HTML服务器控件-传统的HTML标签
  + Web服务器控件-新的ASP.NET标签
  + Validation服务器控件-用于输入验证
* HTML服务器控件

HTML服务器控件是服务器可理解的HTML标签

ASP.NET中的HTML元素是作为文本来进行处理的。要想使这些元素可编程，就需要向这些HTML元素**添加runat=”server”属性**。该属性指示，此元素是一个服务器控件。同时要添加id属性来标识该服务器控件。该id引用可用于操作运行时的服务器控件。

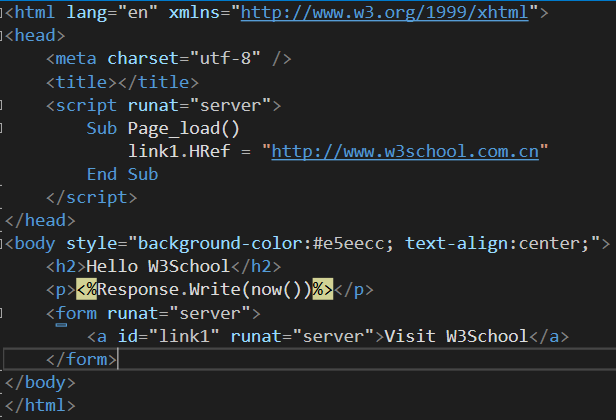
注释：所以HTML服务器控件**必须位于带有runat=“server”属性的<form>标签内**.runat=”server”属性指示该表单应在服务器进行处理.它同时执行其包括在内的空间可被服务器脚本访问.

在下面的例子中,我们在.aspx文件中声明了一个HTMLAnchor（该对象表示HTML超链接）服务器控件.然后我们在一个事件处理程序中操作该HTMLAnchor控件的href属性。



Page\_Load事件是众多ASP.NET可理解的事件中的一种类型：

**注释**：事件处理程序（event Handler）是一种针对给定事件执行代码的子例程。



* Web服务器控件

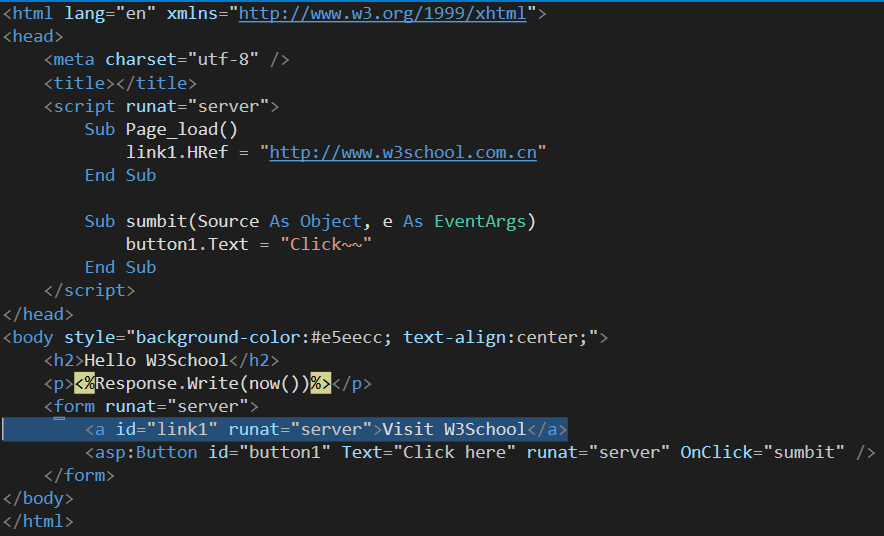
Web服务器控件是服务器可理解的特殊ASP.NET标签。

类似HTML服务器控件，Web服务器控件也在服务器上创建，它们同样需要runat=”server”属性以使其生效.不过,Web服务器控件没有必要映射任何已存在的HTML元素,它们代表更复杂的元素.

创建Web服务器控件的语法是:



在下面的例子中,我们在.aspx文件中声明了一个Button服务器控件.然后我们为Click事件创建一个事件处理程序,它可修改按钮上的文本:



Validation服务器控件

**Validation服务器控件用于验证用户输入**.如果用户输入没有通过验证,将向用户显示一条错误信息.

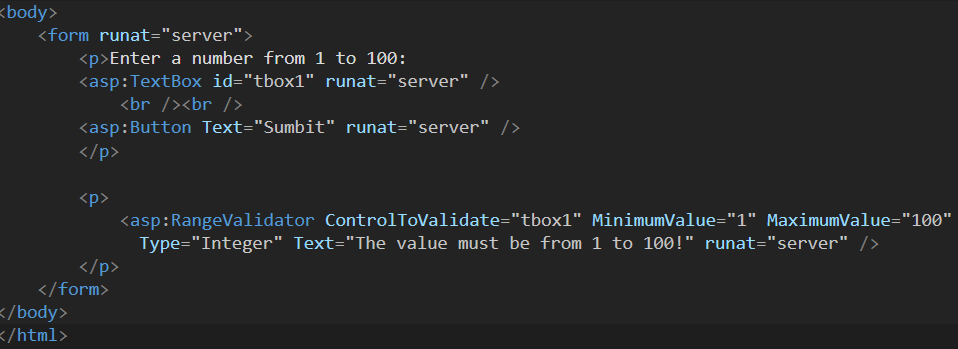
每种Validation控件执行一种特定的验证类型(比如验证某个具体的值或者某个范围的值).

默认地,当点击button,ImageButton或linkbutton时,就会执行页面认证.但是可以通过把CauseValidation属性设置为false,来阻止某个按钮控件被点击时进行验证.

创建Validation服务器控件的语法是:



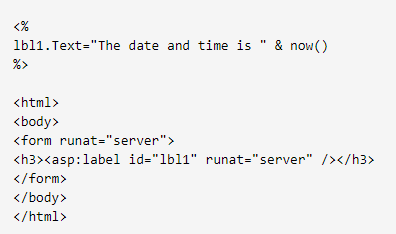
在下面的例子中,我们在.aspx文件中声明了一系列控件。如果验证失败，则失败文本将显示在RangeValidator控件中。



事件

事件句柄（event Handler）是一种针对给定事件来执行代码的子例程。

请查看如下代码：

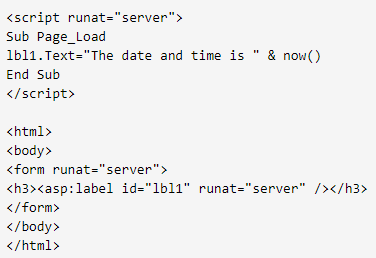


上面的代码什么时候会执行？答案是不知道

因为没有触发事件的代码

Page\_Load事件

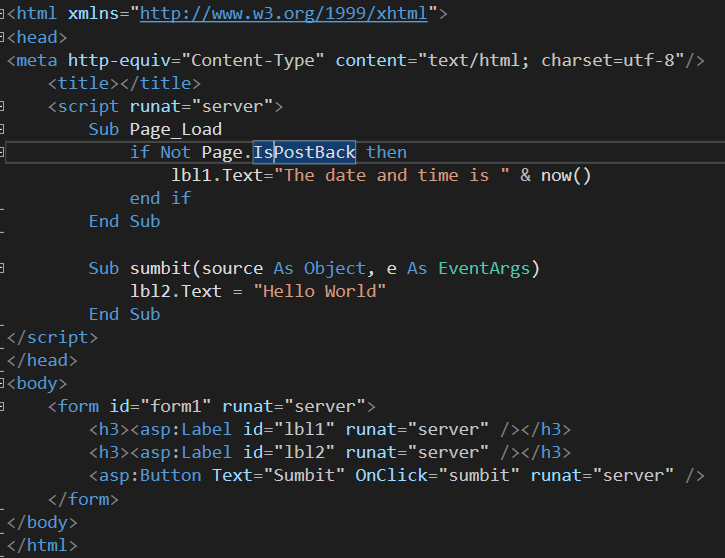
Page\_Load事件是众多ASP.NET可理解的事件之一。Page\_Load事件会在页面加载时触发，然后ASP.NET会自动调用子例程Page\_Load，并执行其中的代码：



注意：该Page\_Load事件不包含对象引用或事件参数！

Page.IsPostBack属性

Page\_Load子例程会在页面每次加载时运行。如果你仅希望在页面第一次加载时执行Page\_Load子例程中的代码，你可以使用Page.IsPostBack属性。如果Page.IsPostBack属性为false，则页面第一次被载入，如果为true，则页面传回服务器（例如，通过点击表单上的按钮）



上面的例子仅在页面初次加载时创建“The date and time is。。。”这条消息。当用户点击Submit按钮时，submit子例程将在第二个label创建“hello world”，但第一个label中的日期和时间不会改变。（IsPostBack属性只有在第一次页面加载时为false，只要加载一次后就变为true。）

Web窗体

* 所有服务器控件必须出现在<form>标签内,同时<form>标签必须包含runat=”server”属性.
* Web表单

所有服务器控件必须出现在<form>标签内,同时<form>标签必须包含runat=”server”属性.runat=”server”属性指示该表单必须在服务器上进行处理.它还指示装入其中的空间能够被服务器脚本访问：



**注释**：该表单总是向自身页面进行提交。如果你规定了一个action属性，它就会被忽略，提交到action指定的地址。如果你省略了method属性，它将被默认地设置为“**post**”。同时，如果你没有规定name和id属性，它们则由ASP.NET自动分配。

注释：一个.aspx仅能包含一个<form runat=”server”>控件

如果你查看一个.aspx页面的源代码,而其中包含的表单不带有name,method,action,id属性,那么你将看到ASP.NET已经把这些属性添加到该表单.

即系统默认值:

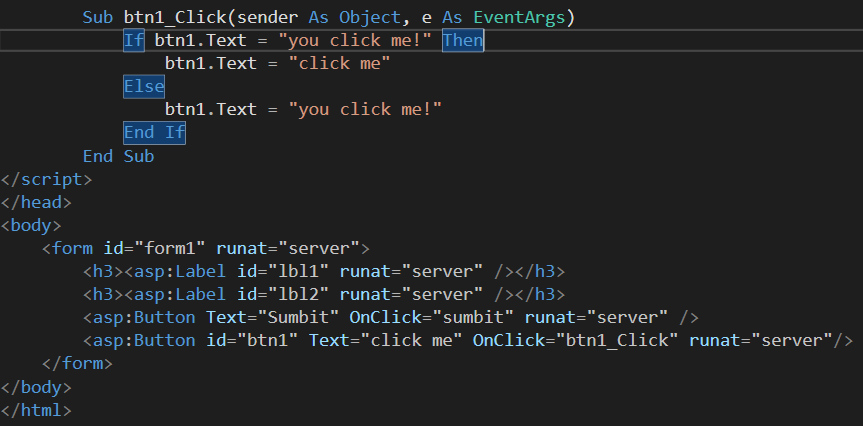
<form name="\_ctl0" method="post" action="page.aspx" id="\_ctl0">...some code</form>

* 提交表单

表单通常通过点击一个按钮来提交.ASP.NET中button服务器控件的格式是这样的:



Id属性为按钮定义了一个唯一的名称,而text属性则为按钮分配了一个label.onclick事件句柄规定了一个要执行的子例程.在下面的例子中,我们在一个.aspx文件中声明了一个按钮控件.一个鼠标点击就可以运行一个子例程.,可以更改该按钮上的文本.



维持ViewState

* 通过在Web表单中维持对象的Viewstate(试图状态),你可以省去大量的编码工作.
* 维持Viewstate

当classic ASP中的表单被提交时,所有的表单都会被清空.设想一下,你提交了一张带有大量信息的表单,而服务器返回了一个错误.你不得不返回表单,然后更正其中的信息.你点击后退按钮,然后会发生什么呢…所有的表单被清空了,而你不得不重新开始所有的一切.站点不会维持你的Viewstate.

当ASP.NET中的表单被提交时,表单会随所有表单值一同重新出现.如何做到的呢?这是由于ASP.NET维持了你的ViewState.ViewState会在页面被提交到服务器时指示其状态.通过在每张页面中的一个<form runat=”server”>控件中放置一个隐藏域，我们就可以定义页面的状态了。

<form name="\_ctl0" method="post" action="page.aspx" id="\_ctl0">

<input type="hidden" name="\_\_VIEWSTATE"

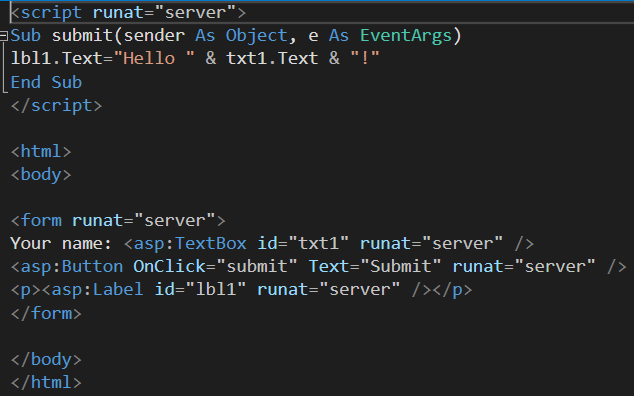
value="dDwtNTI0ODU5MDE1Ozs+ZBCF2ryjMpeVgUrY2eTj79HNl4Q=" />

// 这是一个隐藏域，用于存放Viewstate的值

.....some code

</form>

维持Viewstate是ASP.NET Web表单的默认设置。如果你不希望维持Viewstate，请在.aspx页面的顶部包含指令：<%@ Page EnableViewState=”false” %>，或为任意控件添加属性：EnableViewState=“false”。



TextBox控件

* TextBox控件用于创建用户可输入文本的文本框
* TextBox控件

TextBox控件用于创建用于可输入文本的文本框。

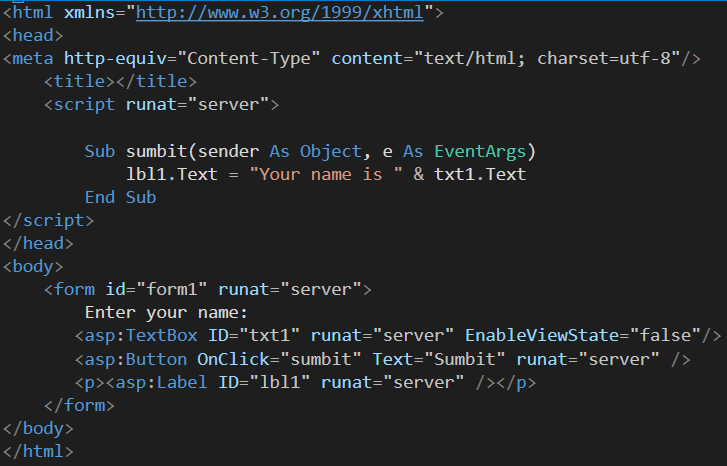
下面例子中演示了你可能使用到的一些属性



* 添加脚本

当表单被提交时，TextBox控件的内容和设置可通过服务器脚本进行修改。可通过点击一个按钮或当前用户更改TextBox控件中的值对表单进行提交。

在下面的例子中，我们在一个.aspx文件中声明了一个TextBox控件、一个label控件和一个button控件。当提交按钮被触发时，submit子例程就会被执行。Submit子例程会向label控件写一条文本：



例二：我们在.aspx页面中声明了一个TextBox控件和一个Label控件。当你修改了TextBox中的值，并且在TextBox外点击，change子例程就会被执行。



其中的**autopostback**是服务器控件设置是否自动提交属性。

当其设置为true时，比如dropDownList发生选择索引变化的时候，会使得网页自动提交。可以指定一个变动事件来使窗体被提交。

当AutoPostBack 设置为false的时候，n变动事件会被捕捉，但是不会立即引发提交。取而代之的是，它们会被控件缓存起来直到下一次发生提交为止。然后当网页在服务器上再次被处理时，所有搁置的事件会被引发并处理。  
在服务器网页的处理期间，所有的事件会优先处理，而不是依照特定的顺序处理。**当所有变动事件都处理完毕之后，引发窗体被提交的鼠标按键事件才会被处理**。

Button控件

* Button控件用于显示一个按钮
* Button控件

按钮可以是提交按钮或命令按钮，而该控件属于提交按钮。

Submit按钮没有命令名称，当它被点击时，它把页面传回服务。可以编写一些事件句柄，在提交按钮被点击时来控制动作的执行。

Command按钮拥有命名名称，且允许你在页面上创建多个button控件。可以编写一些事件句柄。在command按钮被点击时来控制动作的执行。

数据绑定

* 我们可以使用数据绑定（Date Binding）来完成带有可选项目的列表，这些可选项目来自某个导入的数据源，比如数据库、xml、或脚本。
* 数据绑定

下面的控件是支持数据绑定的列表控件：

* + asp: RadioButtonList
  + asp:CheckBoxList
  + asp:DropDownList
  + asp:ListBox

通常在一个或多个 asp:ListItem 控件中定义以上每个控件中的可选项目，类似这样：



不过，我们可以使用某种独立的源进行数据绑定，比如数据库、xml文件或脚本来完成带有可选项目的列表。

通过使用导入的源，数据从html分离了出来，并且对项目的任何改变都是在独立的数据源中完成的。

ArrayList对象

ArrayList对象是包含单一数据值的项目的集合



创建一个新的ArrayList对象，名为mycountries

默认的，一个ArrayList对象包含16个条目。可通过TrimToSize()方法把ArrayList调整为最终大小。

把数据绑定到ArrayList：

DataSource属性被设置为该ArrayList，它定义了这个控件的数据源

DateBind()方法把这个控件与数据源绑定在一起。

Hashtable对象

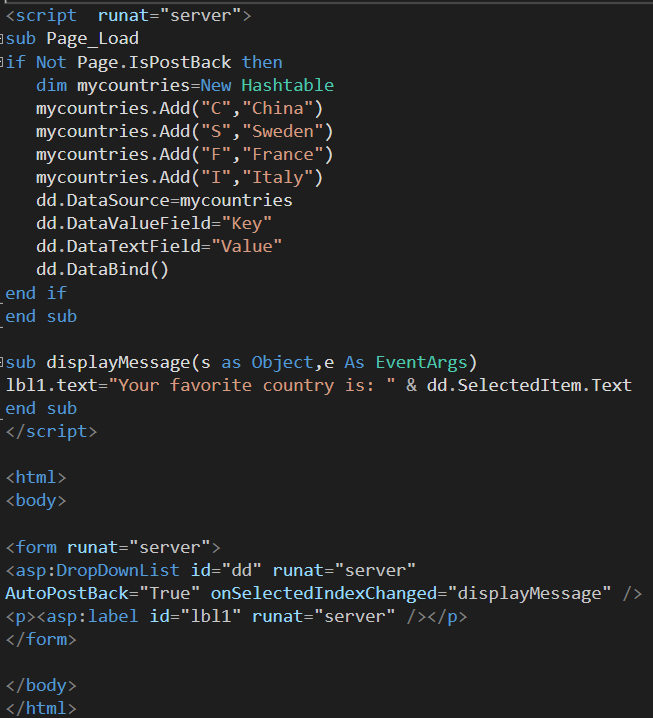
HashTable对象包含用键/值对表示的项目。

数据绑定

HashTable对象可为下面这些控件自动地生成文本和值：

* asp:RadioButtonList
* asp:CheckBoxList
* asp:DropDownList
* asp:Listbox

其中“C”为value值，“China”为text值，依次类推



Xml文件

* 我们可以把xml文件绑定到列表控件
* 一个xml文件

这里有一个名为countries.xml的xml文件：

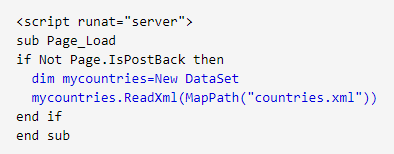


* 把**dataset**绑定到list控件：

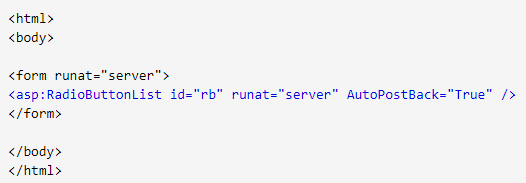
首先，导入system.Data命名空间。我们需要该命名空间与DataSet对象一起工作。把下面这条指令包含在.aspx页面的顶部：



接下来，为这个xml文件创建一个dataset，并在页面首先加载时把这个xml文件载入该dataset：



如需把该dataset绑定到radiobuttonlist，首先请在.aspx页面中创建一个radiobuttonlist控件（没有任何asp:LisetItem元素）

  
然后添加构建这个xml dataset的脚本：



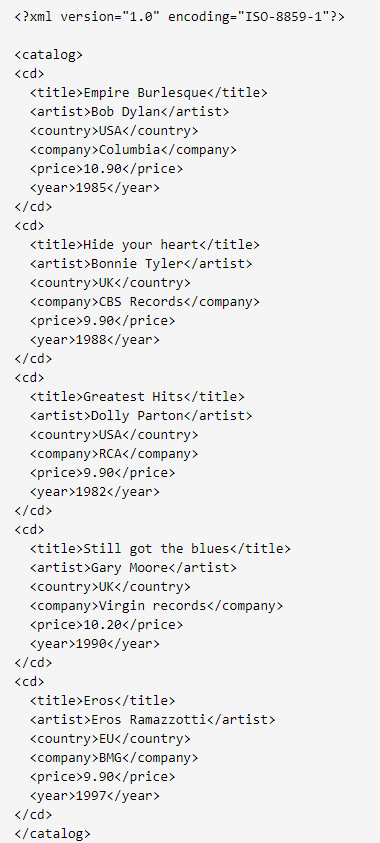
然后我们添加一个子例程，该例程会在用户点击radiobuttonlist控件中的项目时执行。当用户点击某个单选按钮时，label中会出现一条文本：

Repeater控件

* Repeater控件用于显示重复的项目列表，这些项目被限制在该控件
* 把dataset绑定到repeater控件

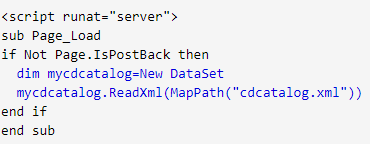
Repeater控件可被绑定到数据库表、xml文件或者其他项目列表里。这里，我们将展示如何把xml文件绑定到一个repeater控件。

Xml文件（cdcatalog.xml）

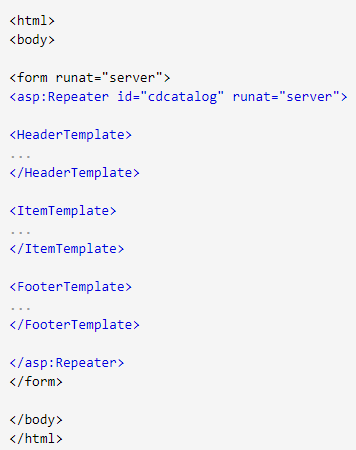


首先，导入“system.data”命名空间。我们需要此命名空间与dataset对象一同工作。

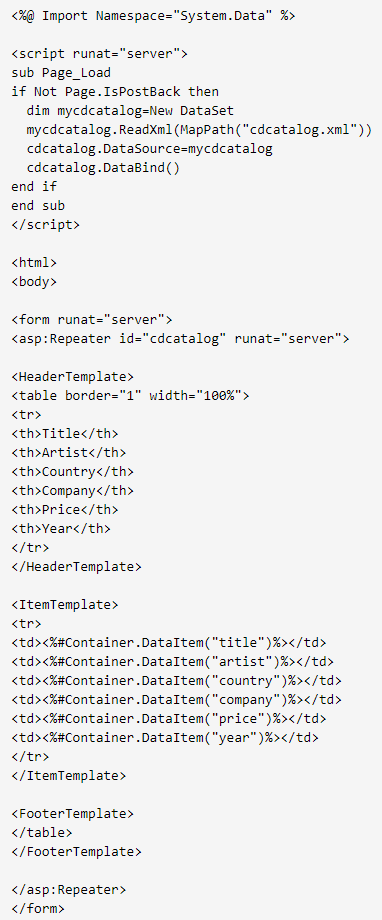
接下来，为xml文件创建一个dataset，并把此xml文件在页面首次加载时载入dataset



然后我们在.aspx页面中创建一个repeater控件。<HeaderTemplate>元素中的内容在输出中仅出现一次，而<ItemTemplate>元素的内容会对应dataset中的“record”重复出现，最后，<FooterTemplate>的内容在输出中仅出现一次：



然后我们添加可创建dataset的脚本，并把这个mycdcatalog dataset绑定到repeater控件。我们同样用HTML标签来填充这个repeater控件，并通过<%#Container.DataItem(“filename”)%>方法把数据项目绑定到<ItemTemplate>部分内的单元格：



用于显示表格

* 使用<AlternatingItemTemplate>

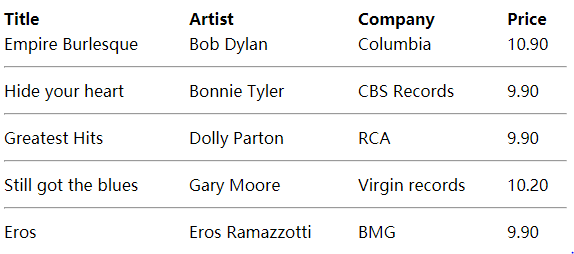
你可以在<ItemTemplate>元素后添加<AlternatingTemplate>元素，这样就可以描述交替行的外观了。在下面的例子中，该表格中每隔一行就会显示为浅灰色的背景：



使用<SeparatorTemplate>

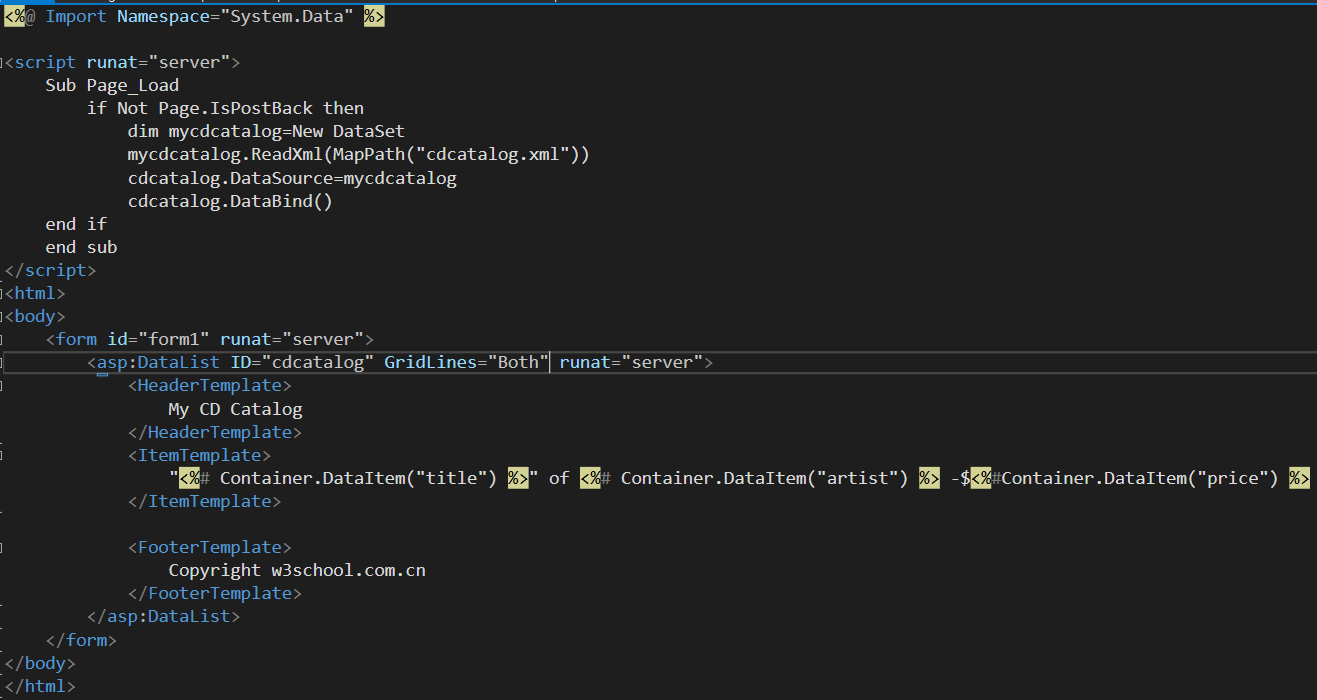
<SeparatorTemplate>元素能够描述每个记录之间的分隔符。下面的例子在每个表格之间插入了一条水平线。



效果如下：  


DataList控件

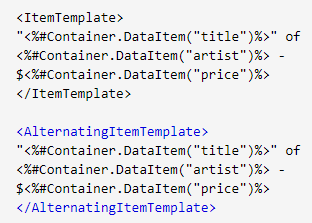
DataList控件，类似于Repeater控件，用于显示限制于该控件的项目的重复列表。不过，DataList控件会默认地在数据项目上添加表格。



DataList会根据具体的数据内容来重复填写表格列

可以在asp标签中添加样式来使表格更加美观

还可以使用<AlternatingItemTemplate>来描述交替行的外观。在<ItemTemplate>样式后面进行添加。



### 数据库连接

ADO.NET同样是.NET框架的组成部分

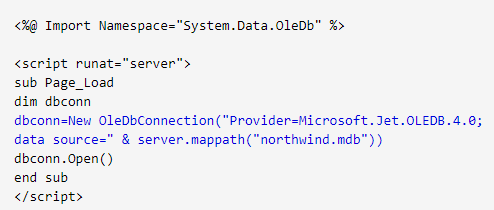
ADO.NET的职能是处理数据访问。通过ADO.NET，你就可以操作数据库了。

什么是ADO.NET？

* ADO.NET是.NET框架的组成部分
* ADO.NET由一系列用于操作数据访问的类组成
* ADO.NET完全基于XML
* 创建数据库连接

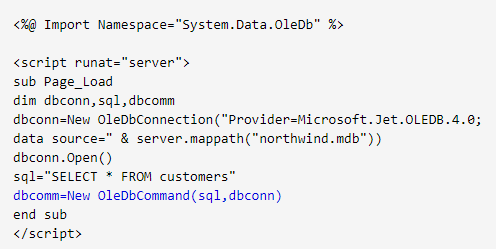
我们打算使用NorthWind（SqlServer和MySQL自带的模板数据库，用于学习）数据库

首先，导入命名空间“System.Data.OleDb”。我们需要该命名空间，以便可以操作Microsoft Access以及其他的OLEDB数据库提供者。我们将Page\_Load子例程中创建对此数据库的连接。我们**创建了一个dbConn变量**，**并把新的OLEDBConnection类赋值给这个变量**，这个**OLEDBConnection类带有一个连接字符串，可以指示OLEDB以及数据库的位置**。然后我们**打开此数据库的连接**：



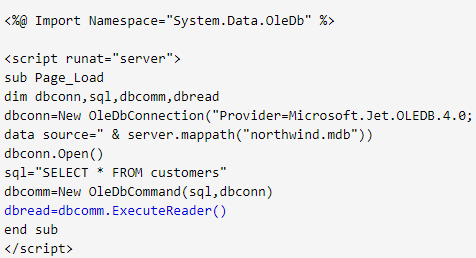
* 创建数据库命令

为了指定需从数据库取回的记录，我们将创建一个dbcomm变量，并为其赋值一个新的OleDbCommand类。这个类用于发出针对数据库表的SQL查询。



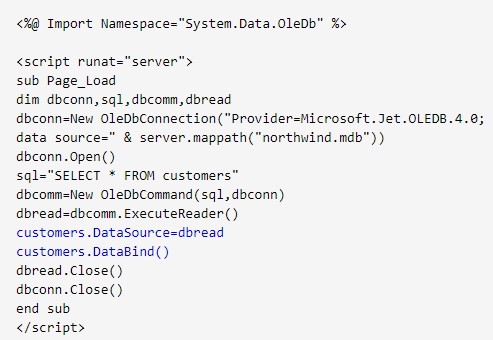
* 创建DataReader

OleDbDataReader类用于从数据源中读取记录流通过调用OleDbCommand对象的ExecuteReader方法，就可以创建DataReader：



* 绑定到Repeater控件

然后，我们把这个DataReader绑定到一个Repeater控件：



创建连接变量dbconn

建立连接后，打开连接：DleDbConnection（）；dbconn.open()

指明SQL语句，创建command对象。

执行command对象获得数据DataReader

将数据源绑定到控件

关闭数据库