第2章 网页标记与网站配置元素

2.1 runat标记——服务器标记

代码声明块定义嵌入 ASP.NET 应用程序文件内使用 runat="server"属性标记的 <script>块中的服务器代码节，此属性标记对于服务器端代码块来说是必需的。

> 说明 代码声明块是使用<script>标记（包含设置为 server 的 runat 属性值）定义的。

语法

```
<script runat="server" language="codelanguage" Src="pathname">
   …//代码编写
</script>
```

示例 本示例包含一个文本框，用于接受用户输入，这是一个潜在的安全威胁。在默认情况下，ASP.NET 网页验证用户输入是否不包括脚本或 HTML 元素。示例的源代码如下：

```
<html>
  <script language="C#" runat="server">
    void Btn_Click(Object Src, EventArgs E)
  {
     Message.Text = "你好" + YourName.Text + ", welcome to GWH' S Blog!";
    }
    </script>
  <body>
     <form runat="server">
      Enter your name: <asp:textbox id="YourName" runat="server"/>
       <asp:button text="Enter" Onclick="Btn_Click" runat="server"/>
         <p>
          <asp:label id="Message" runat="server"/>
      </form>
  </body>
</html>
```

2.2 代码块<%=%>——定义内联代码或内联表达式

代码块呈现（<%%>）定义了当呈现页时执行的内联代码或内联表达式。

将代码添加到 ASP.NET 网页中的默认模型，要么创建一个代码隐藏类文件（代码隐藏页），要么将

页的代码写到具有 runat="server"属性的 script 块中（单文件页），编写的代码通常会与页上的控件进行交互。例如，通过从代码中设置控件的 Text（或其他）属性，可以在页上显示信息。另一种可能是使用嵌入式代码块将代码直接嵌入到页中。

语法

```
<%code%>//内联代码
<%=expression%>//内联表达式
```

使用内联代码可以定义独立的行或代码块。它是在呈现页面的过程中执行的服务器代码。

关于嵌入式代码块的使用说明，读者需要进一步理解，说明如下：

ASP.NET 网页中支持嵌入式代码块，主要用于保留与旧的 ASP 技术的向后兼容性。一般情况下，将嵌入式代码块用于复杂的编程逻辑并不是最佳做法，因为当页中的代码与标记混合时，很难进行调试和维护。此外，由于代码仅在呈现页的过程中执行，因此与将代码置于适当的页处理阶段以执行后台代码或脚本块代码相比，其灵活性大大降低。

嵌入式代码块的部分用途包括：

- 将控件元素或标记元素的值设置为函数返回的值。
- 将计算直接嵌入到标记或控件属性中。

示例 下面这个示例通过使用代码块在页面上输出 5 行标记为“ mrsoft,mrnxt ”的字符串，运行效果如图 2.1 所示。

图 2.1 内联代码的举例

实现本示例的源代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits=
"_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <%for (int i = 1; i <= 5; i++) %>
    <%{ %>
    <%Response.Write("mrsoft,mrnxt"); %><br />
    <%} %>
    </div>
```

```
    </form>
</body>
</html>
```

内联表达式是调用 Write 方法的快捷方式。<%=expression%>用于解析表达式，并将其值返回到块中。以下示例代码通过使用内联表达式显示 GetData 方法的值，运行效果如图 2.2 所示。

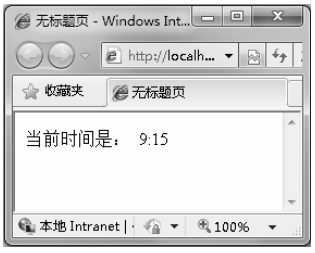

图 2.2　应用代码块<%=%>实现时间的绑定

实现本示例的源代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>无标题页</title>
    <script runat="server">
protected String GetTime()
{
    return DateTime.Now.ToString("t");
}
</script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
       当前时间是： <% =GetTime()%>
    </div>
    </form>
</body>
</html>
```

注意　代码块中的代码必须使用该页的默认语言进行编写。例如，如果该页的@Page 指令包含属性 language=“ C# ”,则该页将使用 Visual C#编译器对标有 runat=server 的所有脚本块中的代码，以及<%%>代码块中的所有内嵌代码进行编译。

典型应用　本示例应用<%%>代码块来实现 ASP.NET 中简单的属性绑定。

首先了解什么是数据绑定。数据绑定不仅可以直接输出一般输出语句可以输出的数据类型，还可以输出任何一种符合数据绑定要求的数据源，而且对于不同的数据显示控件对数据源的处理可以不一致，也可以根据情况的改变而动态地发生变化。也就是说，在数据绑定中，数据的显示是由数据源和数据显示控件共同决定的。数据源决定数据的内容，数据显示控件决定数据的显示方式。实际上，数据绑定的作用机制就是由数据显示控件调用数据源的方法得到数据的。

数据绑定的语法如下：

```
<语言标记 …属性’ <% 数据绑定表达式 %>’ runat="server">
```

如果将表达式的结果直接输出到网页上，那么数据绑定的语法如下：

```
字符串：<% 数据绑定表达式 %>
```

示例运行结果如图 2.3 所示。

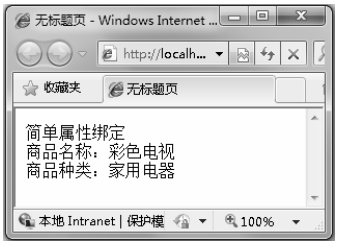

图 2.3 简单属性绑定

下面介绍程序实现的主要步骤。

（1）新建一个网站，默认主页为 Default.aspx。在 Default.aspx 前台页面代码中首先定义两个公共属性，这两个属性作为数据绑定时的数据源；之后应用<%%>代码块将它与显示控件之间建立绑定关系，完整示例代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits=
"_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
 <title>无标题页</title>
  <script runat="server">
      public string GoodsName
      {
          get
          {
             return "彩色电视";
          }
      }
      public string GoodsKind
      {
          get
          {
             return "家用电器";
          }
      }
 </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    简单属性绑定<br/>
    商品名称：<%# GoodsName %><br/>
    商品种类：<%# GoodsKind %>
    </div>
```

```
    </form>
</body>
</html>
```

说明 基于属性的数据绑定所涉及的属性必须包含 get 访问器，因为在数据绑定过程中，数据显示控件需要属性的 get 访问器从属性中读取数据。

（2）绑定完成后，只需要在页面的 Page_Load 事件中调用 Page 类的 DataBind 方法，来实现在页面加载时读取数据。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Page.DataBind();
}
```

2.3 <%!-- --%>——服务器端注释

<%!-- --%>代表服务器端注释。服务器端注释允许开发人员在 ASP.NET 应用程序文件的任何部分（除了<script>代码块内部）嵌入代码注释。服务器端注释元素的开始标记和结束标记之间的任何内容，不管是 ASP.NET 代码还是文本，都不会在服务器上进行处理或呈现在结果页上。


语法

```
<%!-- 注释--%>
```

ASP.NET 服务器端注释块与传统的语言特定注释块具有相同的用法（包括文档和测试）。例如，用户可以使用服务器端注释来描述文件的标记部分，或注释掉页面中声明的一个或多个服务器控件。这些注释可用于大多数类型的 ASP.NET 应用程序文件，包括网页、用户控件、Global.asax 文件、母版页和外观文件。


示例　下面的代码示例演示使用服务器端注释将 Button 控件注释掉，示例代码如下：

```
<%!--
<asp:button runat="server" id="MyButton"
  OnClick="MyButton_Click" />
--%>
```

注意 服务器端注释用于页面的主体，但不在服务器端代码块中使用。当在代码声明块（包含在<script runat="server"></script>标记中的代码）或代码呈现块（包含在<% %>标记中的代码）中使用特定语言时，应使用正用于编码语言的注释语法。如果在<% %>块中使用服务器端注释块，则会出现编译错误。开始和结束注释标记可以出现在同一行代码中，也可以由许多被注释掉的行隔开。服务器端注释块不能被嵌套。

2.4 <!--#include -->——包含标记

<!--#include -->称为服务器端文件包含，服务器端文件包含可以在 ASP.NET 页内的任意位置插入指定文件的原始内容。包括 Web 页面文件、用户控件文件和 Global.asax 文件。

语法

```
<!-- #include file|virtual="filename" -->
```

属性说明如表 2.1 所示。

表 2.1 属性说明

属 性	说 明
File	文件名是相对于包含带有#include 指令的文件目录的物理路径，此路径可以是相对的。需要注意的是，包括的文件可以位于同一目录或子目录中；但该文件不能位于带有#include 指令的文件的上级目录中
Virtual	文件名是网站中虚拟目录的虚拟路径，此路径可以是相对的。需要注意的是，由于文件的物理路径可能会更改，因此建议采用此方法

赋予 File 或 Virtual 属性的值必须用引号（""）括起来。在执行任何动态代码之前处理被包含的文件。从静态文本（如公共页标题或版权信息）到公共服务器端代码、控件或开发人员要插入到其他页中的 HTML 标记块，包含文件可用于包含任何内容。

注意 尽管用户仍然可以使用#include 标记（通过将公共服务器端代码、控件或 HTML 标记放入要包括在其他网页中的文件内）以实现代码重用，ASP.NET 常用的首选方法是使用 Web 用户控件。用户控件提供了一个面向对象的编程模型，并且提供了比#include 标记更多的功能。

示例 下面的代码示例演示如何使用服务器端包含指令语法来调用将在 ASP.NET 页上创建页眉和脚注的文件，两个文件都使用相对路径。示例代码如下：

```
<html>
  <body>
    <!-- #Include virtual="/include/header.inc" -->
      Here is the main body of the .aspx file.
    <!-- #Include virtual="/include/footer.inc" -->
  </body>
</html>
```

2.5 <authentication>元素——配置身份验证

<authentication>元素用于配置 ASP.NET 身份验证方案，该方案用于识别查看 ASP.NET 应用程序的用户。

语法

```
<authentication
  mode="[Windows|Forms|Passport|None]">
  <forms>...</forms>
  <passport/>
</authentication>
```

上述<authentication>元素的语法中涉及一个 mode 属性，该属性是必选的属性，用来指定应用程序的默认身份验证模式。此属性的值如表 2.2 所示。

表 2.2　< authentication > 元素的 mode 属性值

属　性　值	说　　明
Windows	将 Windows 验证指定为默认的身份验证模式。将它与以下任意形式的 Microsoft Internet 信息服务（IIS）身份验证结合起来使用：基本、摘要、集成 Windows 身份验证（NTLM/Kerberos）或证书。在这种情况下，您的应用程序将身份验证责任委托给基础 IIS。该属性值为默认值
Forms	将 ASP.NET 基于窗体的身份验证指定为默认身份验证模式
Passport	将 Microsoft Passport Network 身份验证指定为默认身份验证模式
None	指定任何身份验证。您的应用程序仅期待匿名用户，否则它将提供自己的身份验证

<authentication>元素的子元素的描述如表 2.3 所示。

表 2.3　< authentication >元素的子元素

子　元　素	说　　明
forms	为基于窗体的自定义身份验证配置 ASP.NET 应用程序
passport	指定要重定向到的页（如果该页要求身份验证，而用户尚未通过 Microsoft Passport Network 身份验证注册）

注意 authentication 元素为 ASP.NET 应用程序配置 ASP.NET 身份验证方案。身份验证方案确定如何识别要查看 ASP.NET 应用程序的用户。mode 属性指定身份验证方案。

<authentication>元素在 Machine.config 或根 Web.config 文件中未显示配置下面的默认 authentication 元素，但它是.NET Framework 中应用程序返回的默认配置。

```
<authentication mode="Windows">
   <forms>
      name=".ASPXAUTH"
      loginUrl="Login.aspx"
      defaultUrl="Default.aspx"
      protection="All"
      timeout="30"
      path="/"
      requireSSL="false"
      slidingExpiration="true"
      cookieless="UseDeviceProfile" domain=""
      enableCrossAppRedirects="false">
      <credentials passwordFormat="SHA1" />
   </forms>
   <passport redirectUrl="internal" />
</authentication>
```

示例　下面的代码示例演示如何为基于窗体的身份验证配置站点、指定传输来自客户端的登录信息的 Cookie 的名称，以及指定当初始身份验证失败时使用的登录页的名称。必须将 authorization 节包含在内，才能要求对所有用户进行 Forms 身份验证，并拒绝匿名用户访问站点，代码如下：

```
<configuration>
   <system.web>
      <authentication mode="Forms">
         <forms name="401kApp" loginUrl="/MRLogin.aspx"/>
      </authentication>
      <authorization>
```

```
            <deny users="?"/>
        </authorization>
    </system.web>
</configuration>
```

2.6 <authorization> 元素——授权

<authorization>元素配置 Web 应用程序的授权，以控制客户端对 URL 资源的访问。

语法

```
<authorization>
    <allow .../>
    <deny .../>
</authorization>
```

<authorization>元素子元素的描述如表 2.4 所示。

表 2.4 <authorization>元素的子元素

子 元 素	说 明
allow	向授权规则映射添加一个规则，该规则允许对资源进行访问
deny	向授权规则映射添加一条拒绝对资源进行访问的授权规则

<authorization>元素的父元素的描述，如表 2.5 所示。

表 2.5 <authorization>元素的父元素

父 元 素	说 明
configuration	指定公共语言运行库和.NET Framework 应用程序所使用的每个配置文件中均需要的根元素
system.web	指定配置文件中 ASP.NET 配置设置的根元素，并包含用于配置 ASP.NET Web 应用程序和控制应用程序行为方式的配置元素

注意 authorization 元素为 Web 应用程序配置授权，以控制对 URL 资源的客户端访问。运行时，授权模块从最本地的配置文件开始，循环访问 allow 和 deny 元素，直到它找到适合特定用户账户的第一个访问规则。然后，该授权模块根据找到的第一个访问规则是 allow 还是 deny 规则来允许或拒绝对 URL 资源的访问。默认的授权规则为<allow users="*"/>。因此，默认情况下允许访问，除非另外配置。

示例 本示例实现的是如何允许所有 Admins 角色成员进行访问，以及如何拒绝所有 users 角色成员进行访问，示例代码如下：

```
<configuration>
    <system.web>
        <authorization>
            <allow roles="Admins"/>
            <deny users="*"/>
        </authorization>
    </system.web>
</configuration>
```

2.7　<compilation>元素——编译设置

<compilation>元素配置 ASP.NET 用于编译应用程序的所有编译设置。

语法

```
<compilation>
   debug="[true|false]"
   batch="[true|false]"
   batchTimeout="number of seconds"
   defaultLanguage="language"
   explicit="[true|false]"
   maxBatchSize="maximim number of pages"
   maxBatchGeneratedFileSize="maximum combined size"
   numRecompilesBeforeAppRestart="number"
   strict="[true|false]"
   tempDirectory="temporary files directory"
   urlLinePragmas="[true|false]"
   assemblyPostProcessorType="assembly post processor, assembly"
   <assemblies>...</assemblies>
  <buildproviders>...</buildproviders>
   <codeSubDirectories>...</codeSubDirectories>
   <compilers>...</compilers>
   <expressionBuilders>...</expressionBuilders>
</compilation>
```

关于上述<compilation>元素语法中常用属性和其子元素介绍分别如表 2.6 和表 2.7 所示。

表 2.6　<compilation>元素的常用属性

属　性	说　明
assemblyPostProcessorType	可选的 String 属性。通过引用程序集的处理器后功能为程序集指定后续处理编译步骤
batch	可选的 Boolean 属性。指定是否支持批处理，如果为 True，则清除在第一次访问文件时所需的编译导致的延迟。当将该属性设置为 True 时，ASP.NET 将以批处理模式预编译所有未编译的文件，第一次编译文件时，这将导致更长时间的延迟；但在这一初始延迟之后，对文件进行后续访问时，将消除编译延迟。默认值为 True
debug	可选的 Boolean 属性。指定是否应编译调试二进制文件（而非发布的二进制文件），默认值为 False
defaultLanguage	可选的 String 属性。指定要在动态编译文件中使用的默认编程语言，如“C#”或“PERL”。语言名是使用 system.codeDom 节的 compilers 元素或此元素的 compilers 子元素（已被否决）定义的，默认值为“vb”
numRecompilesBeforeAppRestart	可选的 Int32 属性。指定应用程序重新启动前可能对资源进行动态重新编译的次数，在全局和应用程序级别（而非目录级别）支持该属性
tempDirectory	可选的 String 属性。指定编译期间用于临时文件存储的目录，默认值为空字符串（""）
urlLinePragmas	可选的 Boolean 属性。指定编译器是否应使用 URL（而非物理路径），默认值为 False

<compilation>元素语法中的子元素介绍如表 2.7 所示。

表 2.7 <compilation>元素的子元素

子 元 素	说 明
assemblies	定义一个程序集名称的集合，这些程序集在 ASP.NET 资源编译期间使用
buildproviders	定义用于编译自定义资源文件的生成提供程序的集合
codeSubDirectories	定义一个有序子目录集合，这些子目录包含在运行时编译的文件
compilers	定义一个编译器选项的集合
expressionBuilders	定义一个要在编译期间使用的资源字符串的集合。资源字符串将前缀与表达式生成器关联起来

示例 下面的代码示例实现的是如何为应用程序配置编译设置，具体的配置代码如下：

```
<configuration>
  <system.web>
    <compilation defaultLanguage="VB"
      debug="true"
      numRecompilesBeforeAppRestart="15">
      <compilers>
        <compiler
          language="VB;VBScript"
          extension=".cls"
          type="Microsoft.VisualBasic.VBCodeProvider,system,
               Version=1.0.5000.0, Culture=neutral,
               PublicKeyToken=b77a5c561934e089"/>
        <compiler
          language="C#;Csharp"
          extension=".cs"
          type="Microsoft.CSharp.CSharpCodeProvider,system,
               Version=1.0.5000.0,  Culture=neutral,
               PublicKeyToken=b77a5c561934e089"/>
      </compilers>
      <assemblies>
        <add assembly="ADODB"/>
        <add assembly="*"/>
      </assemblies>
      <codeSubDirectories>
        <add directoryName="mySubDir1"/>
        <add directoryName="mySubDir2"/>
        <add directoryName="mySubDir3"/>
      </codeSubDirectories>
      <buildProviders>
        <buildProvider
          extension=".mafx" type="BuildProviderType,
           BuildProviderAssembly"/>
      </buildProviders>
    </compilation>
  </system.web>
</configuration>
```

2.8　<customErrors>元素——调试设置

<customErrors> 元素用于指定此应用程序域中的服务器信道是向本地或远程调用方返回经过筛选的异常信息，还是返回完整的异常信息。

 语法

```
<configuration>
  <system.runtime.remoting> 元素
    <customErrors> 元素
<customErrors
  mode="Off|On|RemoteOnly"
/>)
```

其中属性 mode 为必选的属性，指定此应用程序域中的服务器信道是向本地或远程调用方返回经过筛选的异常信息，还是返回完整的异常信息。默认值为 RemoteOnly，该值只将完整的异常信息（包括堆栈跟踪信息）返回给作为服务器的同一计算机上的调用方。

表 2.8 所示介绍了 3 个服务器通道属性值，这些值指定收到异常信息的调用方及其收到的信息类型。

表 2.8　<customErrors>元素中 mode 属性值

值	说　明
Off	所有调用方均收到完整的异常信息
On	所有调用方均收到筛选后的异常信息
RemoteOnly	本地调用方收到完整的异常信息，远程调用方收到筛选后的异常信息

 示例　本示例以配置文件示例指示.NET Framework 远程处理系统代表客户端应用程序域注册一个服务器通道，以便客户端域能够传递将用做回调的委托。如果指定<customErrors mode="Off"/>，那么在回调过程中客户端出现异常的情况下，可以使服务器收到完整的异常信息（包括堆栈跟踪信息）。示例代码如下：

```
<configuration>
<system.runtime.remoting>
<application>
<client url="http://computername:8080">
<activated type="ClientActivatedType, RemoteType"/>
</client>
<channels>
<channel ref="http" port="0">
<serverProviders>
<formatter ref="soap" typeFilterLevel="Full"/>
<formatter ref="binary" typeFilterLevel="Full"/>
</serverProviders>
</channel>
</channels>
</application>
<customErrors mode="Off"/>
</system.runtime.remoting>
</configuration>
```

2.9 <globalization>元素——全局设置

<globalization>元素用来配置应用程序的全球化设置。

语法

```
<globalization
   enableClientBasedCulture="true|false"
   requestEncoding="any valid encoding string"
   responseEncoding="any valid encoding string"
   fileEncoding="any valid encoding string"
   responseHeaderEncoding = "any valid encoding string"
   resourceProviderFactoryType = string
   enableBestFitResponseEncoding = "true|false"
   culture="any valid culture string"
   uiCulture="any valid culture string"/>
```

在上述语法结构中，主要属性介绍如表 2.9 所示。

表 2.9 <globalization>元素中的属性值

属 性	说 明
culture	可选属性。为处理传入的 Web 请求指定默认的区域性，此属性还可以设置为 auto
fileEncoding	可选属性。为.aspx、.asmx 和.asax 文件分析指定默认编码。无论此属性的值是什么，用 byte order mark 前缀保存的 Unicode 和 UTF-8 文件都将被自动识别
requestEncoding	可选的属性。指定每一传入请求的假定编码，包括已传递的数据和查询字符串
responseEncoding	可选的属性。指定响应的内容编码
uiCulture	可选的属性。为处理依赖于区域设置的资源搜索指定默认的区域性

注意 如果服务器或应用程序的 fileEncoding 属性设置已配置为使用 UTF-16，但 UTF-16 不是配置文件范围内的.aspx 页所使用的编码，则发送到客户端浏览器的输出将会损坏，并且可能会显示页的源代码。请确保已配置的 fileEncoding 值与该页中使用的编码相符。

下面代码中默认 globalization 元素不是在 Machine.config 文件或根 Web.config 文件中显式配置的，而是由应用程序返回的默认配置。

```
<globalization requestEncoding="utf-8"
   responseEncoding="utf-8"
   fileEncoding=""
   culture=""
   uiCulture=""
   enableClientBasedCulture="false"
   responseHeaderEncoding="utf-8"
   resourceProviderFactoryType=""
   enableBestFitResponseEncoding="false" />
```

示例 下面的代码示例实现的是如何为 ASP.NET 应用程序指定默认的请求和响应编码，示例代码如下：

```
<configuration>
  <system.web>
    <globalization
      requestEncoding="iso-8859-1"
      responseEncoding="iso-8859-1"/>
  </system.web>
```

典型应用　本示例实现的是在 Web.config 配置文件中设置<globalization> 元素，以防止上传中文名称文件时出现乱码的现象。运行结果如图 2.4 所示。

图 2.4　文件上传功能

当在 Web.config 配置文件中设置<globalization>元素的 requestEncoding 属性为 iso-8859-1，当单击上图中的“显示上传文件的路径及文件名称”按钮时，文件名称将会显示乱码，如图 2.5 所示。

图 2.5　上传的文件名称出现乱码

如何解决上述运行结果中出现的文件名称显示乱码的问题，需要在网站的 Web.config 配置文件中添加如下<globalization>元素代码：

```
<system.web>
   <globalization requestEncoding="gb2312" responseEncoding="gb2312" response
   HeaderEncoding="gb2312"/>
</system.web>
```

通过上述代码配置完成后，接下来编写实现显示上传文件路径及文件名称的代码：

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits=
"_Default" >
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>防止中文乱码</title>
<script runat="server">
   protected void btnWrite_Click(object sender, EventArgs e)
   {
```

```
        //获取上传文件路径及文件名称
        string FileName = fuTxt.PostedFile.FileName;
        //输出获取到的信息
        Response.Write(FileName);
    }
</script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        请选择文本文件: <asp:FileUpload ID="fuTxt" runat="server" /> <br />
        <br />
        <asp:Button ID="btnWrite" runat="server" onclick="btnWrite_Click"
        Text="显示上传文件的路径及文件名称" />
    </div>
    </form>
</body>
</html>
```

完成以上所有操作后，执行本示例，运行结果如图 2.6 所示。

图 2.6 设置<globalization> 元素防止出现乱码

2.10 <httpCookies>元素——配置Cookie

为 Web 应用程序使用的 Cookie 配置属性。

语法

```
<httpCookies domain="String" httpOnlyCookies="true|false" requireSSL="true|
false" />
```

上述语法中各属性说明如表 2.10 所示。

表 2.10 <httpCookies>元素属性

属 性	说 明
domain	可选的 String 属性，设置 Cookie 域名
httpOnlyCookies	可选的 Boolean 属性。默认值为 False
requireSSL	可选的 Boolean 属性，用来获取一个指定是否需要安全套接字层（SSL）通信的值。默认值为 False

下面代码中默认 httpCookies 元素不是在计算机配置文件或根 Web.config 文件中显式配置的，而是由.NET Framework 版本中的应用程序返回的默认配置。

```
<httpCookies httpOnlyCookies="false"  requireSSL="false"  domain="" />
```

说明 Internet Explorer 在 Internet Explorer 6 SP1 中新增了对名为 HttpOnlyCookies 的 Cookie 属性的支持，这样有助于减轻脚本跨站点时所导致的 Cookie 被盗取的威胁。如果兼容的浏览器接收某个 Cookie，而该 Cookie 的 HttpOnlyCookies 已经设置为 True，则客户端脚本无法访问该 Cookie。

示例　本示例为 ASP.NET 应用程序配置 Cookie。示例代码如下：

```
<httpCookies httpOnlyCookies="false"  requireSSL="false" />
```

2.11　<httpHandlers>元素——配置URL和HTTP谓词

根据请求中指定的 URL 和 HTTP 谓词将传入的请求映射到相应的处理程序。可以在配置层次中的任何级别声明此元素。

语法

```
<httpHandlers>
   <add... />
   <remove... />
   <clear/>
</httpHandlers>
```

上述语法中，包括 httpHandlers 元素的 3 个子元素，分别介绍如表 2.11 所示。

表 2.11　<httpHandlers> 元素的 3 个子元素

属　性	说　明
Add	可选的元素。指定映射到处理程序的谓词/路径
Clear	可选的元素。移除当前已配置或已继承的所有处理程序映射
remove	可选的元素。移除映射到处理程序的谓词/路径。remove 指令必须与前一个 add 指令的谓词/路径组合完全匹配。该指令不支持通配符

注意 Microsoft Internet 信息服务（IIS）有自己的 ISAPI 映射扩展模型。为使给定应用程序扩展与其处理程序之间的映射生效，该扩展必须在 IIS 中映射为 ASP.NET ISAPI。对于自定义扩展等非标准扩展，则必须相应地配置 IIS。

示例　本示例执行以下操作：

- 将文件扩展名为.New 的文件的所有 HTTP 请求映射到类 MyHandler.New。
- 将文件扩展名为.MyNewFileExtension 的文件的 HTTP GET 和 HEAD 请求映射到类 MyHandler.MNFEHandler。

两个类都位于 MyHandler.dll 文件中的程序集 MyHandler 中。示例代码如下：

```
<configuration>
   <system.web>
      <httpHandlers>
         <add verb="*"
```

```
                path="*.New"
                type="MyHandler.New,MyHandler"/>
            <add verb="GET,HEAD"
                path="*.MyNewFileExtension"
                type="MyHandler.MNFEHandler,MyHandler.dll"/>
        </httpHandlers>
    <system.web>
</configuration>
```

2.12 <httpModules>元素——配置HTTP模块

在一个应用程序内配置 HTTP 模块。可以在配置层次中的任何级别声明此元素。

 语法

```
<httpModules>
    <add... />
    <remove... />
    <clear/>
</httpModules>
```

上述语法中，包括<httpModules>元素的 3 个子元素，分别介绍如表 2.12 所示。

表 2.12 <httpModules>元素的 3 个子元素

属 性	说 明
Add	可选的元素。将 httpModules 元素添加到应用程序中
Clear	可选的元素。从应用程序中移除所有的 httpModules 元素
Remove	可选的元素。移除对 httpModules 元素的引用，该值必须与上一个 add 指令的值完全匹配

 示例 本示例演示如何添加 3 个对 ASP.NET 应用程序的 httpModules 引用，代码如下：

```
<configuration>
    <system.web>
      <httpModules>
        <add type="System.Web.Caching.OutputCacheModule"
            name="OutputCache"/>
        <add type="System.Web.SessionState.SessionStateModule"
           name="Session"/>
        <add type=Selector, selector.dll"
            name="Selector"/>
      </httpModules>
    </system.web>
</configuration>
```

2.13 <httpRuntime>元素——配置HTTP运行时设置

<httpRuntime>元素用来配置 ASP.NETHTTP 运行时的设置，以确定如何处理对 ASP.NET 应用程序的请求。

.NET Framework 提供了许多不同的运行时宿主，包括 ASP.NET 运行时宿主；当请求到达时，ASP.NET 将运行时库加载到要处理请求的进程中。ASP.NET 还为将在 Web 服务器上运行的每个 Web 应用程序创建一个应用程序域。

语法

```
<httpRuntime
   executionTimeout = "HH:MM:SS"
   maxRequestLength = "number"
   requestLengthDiskThreshold = "number"
   useFullyQualifiedRedirectUrl = "[True|False]"
   minFreeThreads = "number"
   minLocalRequestFreeThreads = "number"
   appRequestQueueLimit = "number"
   enableKernelOutputCache = "[True|False]"
   enableVersionHeader = "[True|False]"
   apartmentThreading = "[True|False]"
   requireRootedSaveAsPath = "[True|False]"
   enable = "[True|False]"
   sendCacheControlHeader = "[True|False]"
   shutdownTimeout = "HH:MM:SS"
   delayNotificationTimeout = "HH:MM:SS"
   waitChangeNotification = "number"
   maxWaitChangeNotification = "number"
   enableHeaderChecking = "[True|False]"
/>
```

关于上述<httpRuntime>元素语法中的主要属性如表 2.13 所示。

表 2.13　< httpRuntime >元素的属性

属　性	说　明
appRequestQueueLimit	可选的 Int32 属性。指定 ASP.NET 将为应用程序排队的请求的最大数目；当没有足够的自由线程来处理请求时，将对请求进行排队。当队列超出了该属性中指定的限制时，将通过“503-服务器太忙”错误信息拒绝传入的请求。默认值为 5000
enable	可选的 Boolean 属性。指定是否在当前的节点及子节点级别启用应用程序域（AppDomain），以接受传入的请求，如果为 False，则实际上关闭了该应用程序
enableHeaderChecking	可选的 Boolean 属性。指定 ASP.NET 是否应检查请求标头，以检测可能的注入式攻击；如果检测到攻击，ASP.NET 将返回错误作为响应。默认值为 True
executionTimeout	可选的 TimeSpan 属性。指定在被 ASP.NET 自动关闭前，允许执行请求的最大秒数，默认值为“00:01:50”（110 秒）
maxRequestLength	可选的 Int32 属性。指定输入流缓冲阈值限制（以 KB 为单位）。此限制可用于防止拒绝服务攻击；例如，因用户向服务器发送大型文件而导致的拒绝服务攻击。默认值为 4096（4MB）
requestLengthDiskThreshold	可选的 Int32 属性。指定输入流缓冲阈值限制（以字节为单位）。该值不应超过 maxRequestLength 属性，默认值为 256
useFullyQualifiedRedirectUrl	可选的 Boolean 属性。指定客户端重定向是否是完全限定的（采用"http://server/path"形式，这是某些移动控件所必需的），或者指定是否代之以将相对重定向发送到客户端；如果为 True，则所有不是完全限定的重定向都将自动转换为完全限定的格式

示例 本示例实现的是如何为 ASP.NET 应用程序指定 HTTP 运行时参数。示例代码如下：

```
<configuration>
    <system.web>
    <httpRuntime maxRequestLength="4000"
      enable = "True"
      requestLengthDiskThreshold="512
      useFullyQualifiedRedirectUrl="True"
      executionTimeout="45"
      versionHeader="1.1.4128"/>
    </system.web>
</configuration>
```

典型应用 本示例实现的是在 Web.config 配置文件中设置<globalization> 元素，来防止上传中文名称文件时出现乱码的现象。

<httpRuntime>元素用来设置 ASP.NET HTTP 运行库。该元素可在计算机、站点、应用程序和子目录级别声明。本示例实现的是应用<httpRuntime>元素控制用户上传文件最大为 2MB，最长时间为 60s，最多请求数为 100。

为了实现上述功能要求，首需要在网站的 Web.config 配置文件中添加如下< httpRuntime >元素代码：

```
<system.web>
    <httpRuntime maxRequestLength="2048" executionTimeout="60" appRequest QueueLimit
    ="100"/>
</system.web>
```

通过上述代码配置完成后，接下来编写实现文件上传的功能代码，具体示例代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="FileUpload.aspx.cs"
Inherits="FileUpload" %>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>控制用户上传文件大小</title>
<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
    {
        string MyInfo = "";
        try
        {
            if (FileUpload1.PostedFile.FileName == "")
            {
                Label1.Text = "要上传的文件不允许为空！";
                return;
            }
            else
            {
                MyInfo += "<br/>文件大小（字节）：" + FileUpload1.PostedFile.ContentLength
                + "小于 2MB！";
                Response.Write(MyInfo);
                //获取上传文件的路径
                string filepath = FileUpload1.PostedFile.FileName;
                //获取文件上传名称
                string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
                //存储到服务器指定路径下
```

```
                string serverpath = Server.MapPath("File/") + filename;
                //确定上传文件
                FileUpload1.PostedFile.SaveAs(serverpath);
                Label1.Text = "上传成功！";
                Response.Write(filename);
                MyInfo += "<br/>文件大小（字节）：" + FileUpload1.PostedFile.ContentLength
                + "小于 2MB！";
            }
        }
        catch (Exception error)
        {
            Label1.Text = "处理发生错误！原因：" + error.ToString();
        }
    }
</script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        选择文件：<asp:FileUpload ID="FileUpload1" runat="server" /><br />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="单击上
        传" Width="102px" />
        <asp:Label ID="Label1" runat="server" Text="提示:上传文件不能大于 2M！"
Width="211px"></asp:Label></div>
    </form>
</body>
</html>
```

运行本示例代码，实现文件上传功能的结果如图 2.7 所示。

合法上传文件不大于 2MB 的运行结果如图 2.8 所示。

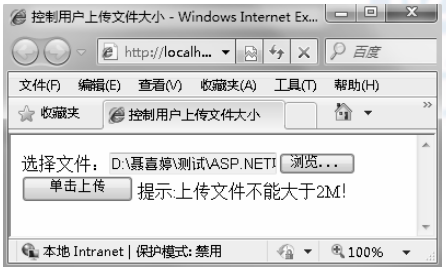

图 2.7　文件上传功能

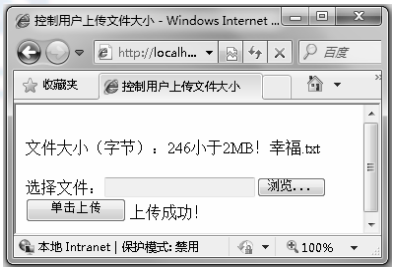

图 2.8　合法上传文件结果

说明 如果上传文件大于 2MB 该页将无法显示！

2.14　<identity>元素——配置Web应用程序的标识

配置 Web 应用程序的标识。此元素可以在配置文件层次结构中的任何级别进行声明。

<identity>元素中有两个重要的属性值：username 属性和 password 属性（在该元素语法中将做介绍），这两个属性值都是以明文形式存储在配置文件中的。虽然 Microsoft Internet 信息服务（IIS）不传输.config 文件来响应用户代理请求，但是可以通过其他途径读取.config 文件。例如，通过在包含服务器的域上具有

适当凭据的已经过身份验证的用户。由于安全原因，identity 属性支持在注册表中存储加密的 username 和 password 属性。凭据必须是 REG_BINARY 格式，并用 Microsoft Windows 2000 和 Windows XP 数据保护 API（DPAPI）加密函数加密。

 语法

```
<identity impersonate="true|false"
  userName="domain\username"
  password="<secure password>"/>
```

上述语法中各属性说明如表 2.14 所示。

表 2.14 <identity>元素属性

属 性	说 明
impersonate	必选项。指定是否对每一个请求使用客户端模拟。该属性为 False，则为指定使用客户端模拟；属性值为 True，则为指定使用客户端模拟
password	可选选项。如果 impersonate 属性为 True，则指定要使用的密码
username	可选选项。如果 impersonate 属性为 True，则指定要使用的用户名

要加密用户名和密码并将它们存储在注册表中，用户可按如下方式设置 username 和 password 属性。

```
userName="registry:HKLM\Software\AspNetProcess,Name"password="registry:HKLM\Softw
are\AspNetProcess,Pwd"
```

上述配置代码中，字符串中位于关键字 registry 之后和逗号之前的部分表示 ASP.NET 打开的注册表项的名称。逗号之后的部分包含一个字符串值的名称，ASP.NET 将从此名称中读取凭据。必须有逗号，并且凭据必须存储在 KLM 配置单元中。如果配置格式不正确，则 ASP.NET 不会启动辅助进程，然后将显示造成当前账户创建失败的代码路径。

 示例 本示例是应用程序返回的默认配置，代码如下：

```
<identity impersonate="false" userName="" password="" />
```

注意 默认 identity 元素不是在 Machine.config 文件或根 Web.config 文件中显式配置的。

2.15 <machineKey>元素——密钥配置

对密钥进行配置，以便将其用于对 Forms 身份验证 Cookie 数据和视图状态数据进行加密和解密，并将其用于对进程外会话状态标识进行验证。

 语法

```
<machineKey
   validationKey="AutoGenerate,IsolateApps" [String]
   decryptionKey="AutoGenerate,IsolateApps" [String]
   validation="SHA1" [SHA1 | MD5 | 3DES | AES]
   decryption="Auto" [Auto | DES | 3DES | AES]
/>
```

上述语法中各属性说明如表 2.15 所示。

表 2.15　<machineKey>元素属性

属　性	说　明
decryption	可选的 String 属性，指定用于对数据进行解密的哈希算法的类型
decryptionKey	必选的 String 属性，指定用于验证加密数据的密钥。默认值为 AutoGenerate,IsolateApps。validationKey 值的 IsolateApps 修饰符指示 ASP.NET 使用应用程序的 ID，为每个应用程序生成唯一的加密密钥。IsolateApps 作为一部分包含在默认值中
validation	必选的 MachineKeyValidation 属性，指定用来验证数据的加密类型
validationKey	必选的 String 属性，指定用于验证加密数据的密钥

读者可以看到，上述语法中涉及对数据进行解密的哈希算法的类型，即“ decryption="Auto"[Auto | DES | 3DES | AES] ”，下面对其解密类型的列表值进行介绍，decryption 属性可选择其中任意一个值。具体介绍如表 2.16 所示。

表 2.16　decryption 属性的列表值

值	说　明
Auto	指定 ASP.NET 基于配置设置来确定使用哪个解密算法。这是默认的 machineKey 解密值
AES	指定 ASP.NET 使用 AES（Rijndael）算法来对数据进行解密。AES 是用于对数据进行解密的默认算法
3DES	指定 ASP.NET 使用 TripleDES 算法来对数据进行解密。TripleDES（3DES）算法使用 DES 算法的 3 次连续迭代
DES	指定 ASP.NET 使用数据加密标准（DES）算法来解密数据

在表 2.17 <machineKey>元素的 validation 属性中涉及用来验证数据的加密类型，即“ decryption="Auto" [Auto | DES | 3DES | AES] ”，具体介绍如表 2.17 所示。

表 2.17　validation 属性的列表值

值	说　明
AES	指定 ASP.NET 使用 AES（Rijndael）算法来验证数据
MD5	指定 ASP.NET 使用 MessageDigest5（MD5）哈希算法来验证数据。此算法的性能比 SHA1 好
SHA1	指定 ASP.NET 使用 SHA1 哈希算法来验证数据。使用此算法可增强安全性。这是默认值
3DES	指定 ASP.NET 使用 TripleDES 算法来验证数据。TripleDES 算法使用 DES 算法的 3 次连续迭代

示例　下面的代码示例演示如何将 validationKey 和 decryptionKey 属性都设置为 AutoGenerate。并且指定了 isolateApps 值，以便为服务器上的每个应用程序生成一个唯一的密钥。示例代码如下：

```
<machineKey
   validationKey="AutoGenerate,IsolateApps"
   decryptionKey="AutoGenerate,IsolateApps"
   validation="SHA1"
/>
```

2.16 <pages>元素——全局页配置

全局定义页特定配置设置，如配置文件范围内的页和控件的 ASP.NET 指令。

pages 元素定义页特定的配置设置。在全局范围内为配置文件范围内的所有页和控件设置某些 ASP.NET 页和控件指令。这包括以下页级别的指令，这些指令所指定的设置由页和用户控件编译器在处理 ASP.NET Web 窗体页（.aspx）和用户控件（.ascx）文件时使用：

- @Page 指令（@Page）。
- 通过 namespaces 子元素使用@Import 指令（@Import）。
- 通过 controls 子元素使用的@Register 指令（@Register）。

pages 元素还为以下操作提供支持：在运行时通过 tagMapping 元素将标记类型映射到其他标记类型。

> **注意** 将@Page 指令添加到母版页时，不能在依赖于母版页的页中使用相同的指令声明，而应该使用 pages 配置元素来全局定义页指令。

语法

```
<pages
    buffer="[True|False]"
    enableEventValidation="[True|False]"
    enableSessionState="[True|False|ReadOnly]"
    enableViewState="[True|False]"
    enableViewStateMac="[True|False]"
    smartNavigation="[True|False]"
    autoEventWireup="[True|False]"
    pageBaseType="typename, assembly"
    userControlBaseType="typename"
    validateRequest="[True|False]"
    masterPageFile="file path"
    theme="string"
    styleSheetTheme="string"
    maxPageStateFieldLength="number"
    compilationMode="[Always|Auto|Never]"
    pageParserFilterType="string"
    viewStateEncryptionMode="[Always|Auto|Never]"
    maintainScrollPositionOnPostBack="[True|False]"
    asyncTimeout="number">
    <controls>...</controls>
    <namespaces>...</namespaces>
    <tagMapping>...</tagMapping>
</pages>
```

关于上述<pages>元素语法中的主要属性及其子元素介绍分别如表 2.18 和表 2.19 所示。

表 2.18 <pages>元素的主要属性

属　性	说　明
autoEventWireup	可选的 Boolean 属性，指定是否自动启用页事件，当出现自动事件连接时就是如此，它表示系统将自动连接特定签名的方法（如 page_Load），默认值为 True

（续）

属　性	说　明
buffer	可选的 Boolean 属性，指定 URL 资源是否使用响应缓冲，默认值为 True
enableEventValidation	指定页和控件是否验证回发和回调事件。默认值为 True
enableSessionState	可选的 String 属性，为配置文件范围内的资源指定会话状态要求。有 3 种属性值：False（表明会话状态已禁用）、ReadOnly（表明会话状态不可写）和 True（表明会话状态已启用，此属性值是默认值）
maintainScrollPosition OnPostBack	可选的 Boolean 属性。指定在页回发到服务器上时，是否将用户返回到客户端浏览器中的同一位置。如果为 False，则用户将在页回发时返回到页首，默认值为 False
masterPageFile	可选的 String 属性。指定母版页相对于本地配置文件的路径。masterPageFile 属性设置为 True 的页必须包含一个 Content 控件作为顶级控件，默认值为空字符串（""）
theme	可选的 String 属性。指定用于配置文件范围内的页的主题名称。所指定的主题必须作为应用程序或全局主题存在。如果该主题不存在，将会引发 HttpException 异常，默认值为空字符串（""）
validateRequest	可选的 Boolean 属性。指示 ASP.NET 在从浏览器输入的所有内容中检查是否存在潜在的危险数据。如果是 True，则通过将所有输入数据与一个潜在危险值列表进行比较来执行请求验证。如果发生匹配，ASP.NET 将引发 HttpRequestValidationException 异常，默认值为 True

<pages>元素语法中的子元素介绍如表 2.19 所示。

表 2.19　<pages>元素的子元素

值	说　明
controls	定义标记前缀所在的 register 指令和命名空间的集合
namespaces	定义一个将在程序集预编译期间使用的导入指令的集合
tagMapping	定义一个标记类型的集合，这些标记类型在编译时重新映射为其他标记类型
controls	定义标记前缀所在的 register 指令和命名空间的集合

示例　下面的代码示例演示如何指定多个页配置设置，如其中设置了指定 URL 资源来响应缓冲、开启会话状态和指定母版页相对于本地配置文件的路径等。示例代码如下：

```
<configuration>
  <system.web>
     <pages buffer="true"
       enableSessionState="true"
       autoEventWireup="true"
       maintainScrollPositionOnPostBack="true"
       masterPageFile = "~/Masters/mrgwh.master" />
  </system.web>
</configuration>
```

典型应用

1．在客户端用户请求验证中检测潜在危险输入值

在开发 Web 项目时，有时会引入第三方组件如 FreeTextBox（即第三方文本控件），如果不进行相关配置就会出现如下错误信息：

从客户端（Content="<From language=javascript…/>"中检测到有潜在危险的 Request.From 值。

这里对上述错误进行说明，请求验证过程检测到有潜在危险的客户端输入值，对请求的处理已经中止。该值可能提示危及应用程序的尝试，如跨站点的脚本攻击，通过在 Page 指令或配置节中设置 ValidateRequest=false 可以禁用请求验证。但是，在这种情况下，强烈建议应用程序显式检查所有输入。

解决上述错误信息，可以在 Web.config 配置文件中的 System.Web 节中加入如下代码：

```
<configuration>
  <system.web>
    <Pages ValidateRequest="false" />
  </system.web>
</configuration>
```

2．为应用程序指定和禁用主题

ASP.NET 版本中可以在不同的层级应用主题：应用程序级、文件夹级和网页级。这里主要介绍如何在应用程序级中设置主题。

设置应用程序级的主题，意味着整个应用程序中的所有页面和控件将使用相同的主题。实现这种主题设置可通过在 Web.Config 文件的<pages>配置节中配置如下内容：

```
<configuration>
  <system.web>
    <pages Theme="主题名"/>
  </system.web>
</configuration>
```

说明 以上代码中的 Theme 属性也可以改写成 StyleSheetTheme 属性。完成以上操作后，整个应用程序将自动应用所设置的主题，而不需要再为每个页头设置 Theme 或 StyleSheetTheme 属性了。

同样，一个主题也可以应用于一个给定文件夹和该文件夹下的所有页面。因此，可以将不同的 Web.Config 文件添加到不同的子文件夹中，并且可以给每个 Web 配置文件指定不同的主题。在文件夹中配置的 Web.Config 文件类似于在应用程序级中配置的 Web.Config 文件，配置完后该文件夹及该文件夹下的所有页面都将添加相应的主题。

如果需要禁用整个应用程序或文件夹中设置的主题，只需要将在 Web.Config 文件的<Pages>配置节中的 Theme 或 StyleSheetTheme 属性值设置为空（""）即可。示例代码如下：

```
<configuration>
  <system.web>
    <pages Theme=""/>
  </system.web>
</configuration>
```

2.17 <profile>元素——管理配置文件设置

使用 ASP.NET 配置文件来配置用于管理用户配置文件值的参数。

语法

```
<profile
   enabled="true|false"
   inherits="fully qualified type reference"
   automaticSaveEnabled="true|false"
   defaultProvider="provider name">
   <properties>...</properties>
   <providers>...</providers>
</profile>
```

上述语法中，涉及<profile>元素的属性和子元素，下面分别具体介绍。<profile>元素的属性如表 2.20 所示。

表 2.20　<profile>元素的属性

属　性	说　明
enabled	可选的 Boolean 属性。指定是否启用 ASP.NET 用户配置文件。如果为 True，则启用 ASP.NET 用户配置文件。默认值为 True
defaultProvider	可选的 String 属性。指定默认配置文件提供程序的名称，默认值为 AspNetSqlProfile-Provider
inherits	可选的 String 属性。包含从 ProfileBase 抽象类派生的自定义类型的类型引用。ASP.NET 动态地生成一个从该类型继承的 ProfileCommon 类，并将该类放在当前 HttpContext 的 Profile 属性中
automaticSaveEnabled	可选的 Boolean 属性，用来指定用户配置文件是否在 ASP.NET 页执行结束时自动保存。如果为 True，则用户配置文件在 ASP.NET 页执行结束时自动保存

<profile>元素的子元素介绍如表 2.21 所示。

表 2.21　<profile>元素的子元素

元　素	说　明
properties	必选的元素。定义用户配置文件属性和属性组的集合
providers	可选的元素。定义配置文件提供程序的集合

下面代码中默认 profile 元素在.NET Framework 的 Machine.config 文件中配置。

```
<profile>
  <providers>
       <add name="AspNetSqlProfileProvider" connectionStringName="LocalSqlServer"
        applicationName="/"
type="System.Web.Profile.SqlProfileProvider, System.Web, Version=2.0.0.0, Culture=
neutral,
PublicKeyToken=b03f5f7f11d50a3a" />
  </providers>
</profile>
```

示例　本示例默认 profile 元素在.NET Framework 的 Machine.config 文件中配置。示例代码如下：

```
<profile>
  <providers>
```

```
        <add name="AspNetSqlProfileProvider" connectionStringName="LocalSqlServer"
        applicationName="/"
type="System.Web.Profile.SqlProfileProvider, System.Web, Version=2.0.0.0, Culture=
neutral,
PublicKeyToken=b03f5f7f11d50a3a" />
  </providers>
</profile>
```

2.18 <securityPolicy>元素——安全策略集合配置

<securityPolicy>元素定义一个安全策略文件与其信任级别名称之间映射的集合。通过提供映射到 policyFile 属性所指定的文件，且由用户自己命名的 trustLevel 元素，可以扩展安全系统。

语法

```
<securityPolicy>
   <trustLevel />
</securityPolicy>
```

上述语法中，涉及 securityPolicy 元素的一个子元素 trustLevel，该元素是可选的元素。通过这个子元素可以向安全策略映射集合添加安全级别名称与关联策略文件之间的映射。然后，可以通过在 trust 元素的 level 属性中指定信任级别名称，将安全策略文件应用于 ASP.NET 应用程序。

示例 本示例实现的是如何指定处理不同信任级别的策略文件。示例代码如下：

```
<securityPolicy>
   <trustLevel name="Full"     policyFile="internal"/>
   <trustLevel name="High"     policyFile="web_hightrust.config"/>
   <trustLevel name="Medium"   policyFile="web_mediumtrust.config"/>
   <trustLevel name="Low"      policyFile="web_lowtrust.config"/>
   <trustLevel name="Minimal"  policyFile="web_minimaltrust.config"/>
   <trustLevel name="CustomTrustLevel"   policyFile="customtrust.config"/>
</securityPolicy>
```

2.19 <sessionState>元素——配置会话状态

<sessionState>元素为当前应用程序配置会话状态设置。新客户端在开始与 Web 应用程序交互时，会发出一个会话 ID，并且该 ID 将与会话的有效期间从同一客户端发出的所有后续请求关联。此 ID 用于在不同的请求中保持与客户端会话关联的服务器端状态。<sessionState>元素控制 ASP.NET 应用程序如何为每个客户端建立并保持这种关联。这种机制非常灵活，可以为开发人员提供许多功能，其中包括承载进程外的会话状态信息，以及在不使用 Cookie 的情况下跟踪状态。

语法

```
<sessionState
   mode="[Off|InProc|StateServer|SQLServer|Custom]"
   timeout="number of minutes"
   cookieName="session identifier cookie name"
```

```
    cookieless=
        "[true|false|AutoDetect|UseCookies|UseUri|UseDeviceProfile]"
    regenerateExpiredSessionId="[True|False]"
    sqlConnectionString="sql connection string"
    sqlCommandTimeout="number of seconds"
    allowCustomSqlDatabase="[True|False]"
    useHostingIdentity="[True|False]"
    stateConnectionString="tcpip=server:port"
    stateNetworkTimeout="number of seconds"
    customProvider="custom provider name">
    <providers>...</providers>
</sessionState>
```

<sessionState>元素的常用属性介绍如表 2.22 所示。

表 2.22　<sessionState>元素的常用属性

属　性	说　明
allowCustomSqlDatabase	可选的 Boolean 属性。指定会话状态 SQL 数据库是否可以是自定义数据库（而不是 ASP.NET 默认数据库），如果为 False，则不能指定初始目录或数据库作为 sqlConnectionString 属性的值。默认会话状态 SQL 数据库为 ASPState 数据库
cookieless	可选的 HttpCookieMode 属性。指定对于 Web 应用程序使用 Cookie 的方式。cookieless 属性的默认值为 UseCookies
mode	可选的 SessionStateMode 属性，指定存储会话状态值的位置，默认值为 InProc
sqlCommandTimeout	可选的 TimeSpan 属性。指定使用 SQLServer 会话状态模式的 SQL 命令的持续时间超时（秒），持续时间超时是 SQL 命令可以处于空闲状态的时间（秒），超过此时间之后，该命令将被取消。此属性是.NET Framework 中的新属性
sqlConnectionString	为运行 SQL Server 的计算机指定连接字符串，该属性在 mode 属性设置为 SQLServer 值时是必需的
timeout	可选的 TimeSpan 属性。指定在放弃一个会话前该会话可以处于空闲状态的分钟数。对于进程内和状态服务器模式，timeout 属性不能设置为大于 525 600 分钟（1 年）的值；会话 timeout 配置设置仅适用于 ASP.NET 页；更改会话 timeout 值不会影响 ASP 页的会话超时时间。同样，更改 ASP 页的会话超时时间不会影响 ASP.NET 页的会话超时时间

上述<sessionState>元素属性列表中，mode 属性的默认值为 InProc，除此之外，该元素中 mode 的属性值还可以选择以下列表值之一，如表 2.23 所示。

表 2.23　<sessionState>元素的 mode 属性值列表值

属　性	说　明
Custom	会话状态将使用自定义数据存储区来存储会话状态信息
InProc	会话处于正在处理 ASP.NET 辅助进程的状态
Off	会话状态被禁用
SQLServer	会话状态将使用进程外 SQL Server 数据库来存储状态信息
StateServer	会话状态将使用进程外 ASP.NET 状态服务来存储状态信息

下面对表 2.23（<sessionState>元素的 mode 属性值列表值）中两个重要的属性列表值 SQLServer 和 StateServer 进行详细讲解。

（1）使用 StateServer 模式。

- 在将要存储会话状态信息的远程服务器上，确保 ASP.NET 状态服务正在运行。ASP.NET 状态服务随 ASP.NET 一起安装，默认情况下位于%windir%\Microsoft.NET\Framework\version\aspnet_state.exe 中。
- 在应用程序的 Web.config 文件中，将 mode 设置为"StateServer"，并将 stateConnectionString 设置为诸如"tcpip=dataserver:42424"这样的值。

（2）使用 SQLServer 模式。

- 在运行将要存储会话状态的 SQL Server 的计算机上运行 InstallSqlState.sql。
- 在应用程序的 Web.config 文件中，将 mode 设置为"SQLServer"，并将 sqlConnectionString 设置为诸如"data source=localhost;Integrated Security=SSPI;" 这样的值。

以上主要对<sessionState>元素的主要属性进行了讲解，该元素语法中还包含它的一个子元素 providers，该子元素主要是用来包含自定义会话状态存储区提供程序的集合。

示例 本示例实现的是如何指定会话状态配置设置，其中使用了 SQLServer 模式，并指定使用 SQL Server 会话状态模式的 SQL 命令的持续时间超时为 10 秒，代码如下：

```
<sessionState
   mode="SQLServer"
   cookieless="true"
   sqlConnectionString=" Integrated Security=SSPI;data source=MySqlServer;"
   sqlCommandTimeout="10" />
```

2.20 <siteMap>元素——站点地图配置

<siteMap>元素定义配置设置以支持用于配置、存储和呈现站点导航的导航基础结构。

> **注意** 此元素是.NET Framework 中的新元素。

语法

```
<siteMap
  defaultProvider="provider name">
  <providers>...</providers>
</siteMap>
```

上述语法中，涉及<siteMap>元素的一个属性 defaultProvider 和一个子元素 providers，下面分别进行详细介绍。

- DefaultProvider 属性：必选的属性，指定提供程序的名称。默认值为 XmlSiteMapProvider 提供程序。
- 子元素 providers：定义导航提供程序的集合。

示例 本示例实现的是如何指定处理不同信任级别的策略文件。示例代码如下：

```
<configuration>
  <system.web>
    <siteMap defaultProvider="XmlSiteMapReader">
      <providers>
```

```
        <add
          name=" XmlSiteMapReader"
          type="XmlSiteMapProvider, System.Web, Version=1.1.3300.0,
               Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
          siteMapFile="my.sitemap"
          description="XmlSiteMapProvider that loads my.sitemap"/>
      </providers>
    </siteMap>
  </system.web>
</configuration>
```

2.21　<webControls>元素——客户端脚本文件的共享位置

<webControls>元素用来指定客户端脚本文件的共享位置。

注意　此元素是.NET Framework 中的新元素。

语法

```
<webControls clientScriptsLocation="String" />
```

其中 clientScriptsLocation 属性是该元素必需的 String 属性，用于设置客户端脚本位置，默认值为“ /aspnet_client/{0}/{1}/ ”。

示例　下面的代码示例为设置 ASP.NET 应用程序的客户端脚本文件的共享位置。

```
<webControls clientScriptsLocation="/aspnet_client/{0}/{1}/" />
```

注意　上面示例中的“ {0} ”将替换为命名空间名称（如 system_web），“ {1} ”将替换为 Framework 版本号。

2.22　<webServices>元素——Web服务客户端配置

<webServices>元素控制使用 ASP.NET 部署的 Web 服务的设置，以及运行在.NET Framework 上的 Web 服务客户端的设置。

语法

```
<webServices>
    <protocols>
        <add name="protocol name"/>
        <remove name="protocol name"/>
        <clear/>
    </protocols>
    <serviceDescriptionFormatExtensionTypes>
        <add type="type"/>
        <remove type="type"/>
        <clear/>
    </serviceDescriptionFormatExtensionTypes>
```

```
    <soapExtensionTypes>
        <add type="type" priority="number" group="0|1"/>/>
        <remove type="type"/>
        <clear/>
    </soapExtensionTypes>
    <soapExtensionReflectorTypes>
        <add type="type" priority="number" group="0|1"/>/>
        <remove type="type"/>
        <clear/>
    </soapExtensionReflectorTypes>
    <soapExtensionImporterTypes>
        <add type="type" priority="number" group="0|1"/>/>
        <remove type="type"/>
        <clear/>
    </soapExtensionImporterTypes>
    <wsdlHelpGenerator href="help generator file"/>
    <diagnostics suppressReturningExceptions="true|false" />
</webServices>
```

<webServices>元素的子元素介绍如表 2.24 所示。

表 2.24 <webServices>元素的子元素

元 素	说 明
diagnostics	指定是否要将异常返回客户端以用于调试目的
protocols	指定 ASP.NET Web 服务可用来接收从客户端发送来的请求数据和返回响应数据的协议。协议可用来将请求数据与方法及其参数关联起来，以及将响应数据与方法及其返回值关联起来
serviceDescriptionFormatExtensionTypes	指定要在配置文件的范围内运行的服务描述格式扩展
soapExtensionTypes	指定在 Web 服务或客户端上进行处理的过程中用来检查或修改 SOAP 消息的 SOAP 扩展。SOAP 扩展扩充了 Web 服务的功能
soapExtensionReflectorTypes	指定 SOAP 扩展发送程序类，这些类用于扩展服务说明（WSDL 文档）的生成过程。适用于服务说明格式扩展（SDFE）
soapExtensionImporterTypes	指定 SOAP 扩展导入程序类，这些类用于扩展客户端代理的生成过程。适用于服务说明格式扩展（SDFE）
wsdlHelpGenerator	指定 Web 服务帮助页（.aspx 文件），该帮助页在浏览器直接定位到 ASMX Web 服务页时显示给该浏览器

示例 下面的示例指定 XML Web Services 配置设置，具体的配置代码如下：

```
<configuration>
    <system.web>
      <webServices>
        <diagnostics suppressReturningExceptions="false"/>
        <protocols>
          <add name="HttpGet"/>
          <add name="HttpPost"/>
          <add name="Documentation"/>
        </protocols>
```

```
            <serviceDescriptionFormatExtensionTypes>
            </serviceDescriptionFormatExtensionTypes>
            <soapExtensionTypes>
            </soapExtensionTypes>
            <soapExtensionReflectorTypes>
            </soapExtensionReflectorTypes>
            <soapExtensionImporterTypes>
            </soapExtensionImporterTypes>
            <wsdlHelpGenerator href="DefaultSdlHelpGenerator.aspx"/>
        </webServices>
    </system.web>
</configuration>
```