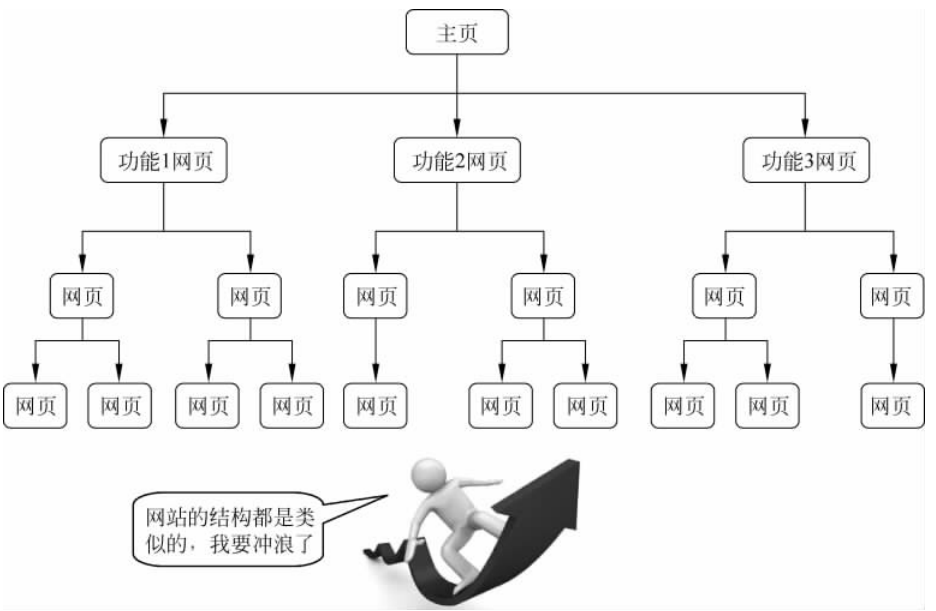
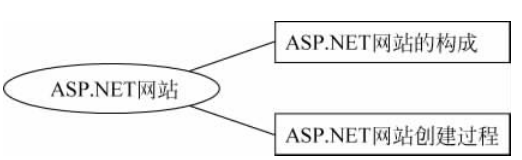
第 3 章　ASP .NET 网站结构



本章指南

3. 1　ASP .NET 网站

知识梳理



3.1.1 ASP .NET 网站的构成

图 3.1 展示了一个 ASP .NET 网站 OnRetS 的结构。从中看出一个典型的 ASP .NET 网站通常由多个网页文件组成，每个网页将共享许多通用的资源和配置设置。通常网页文件和相关的资源文件按内容存放在不同的目录中。

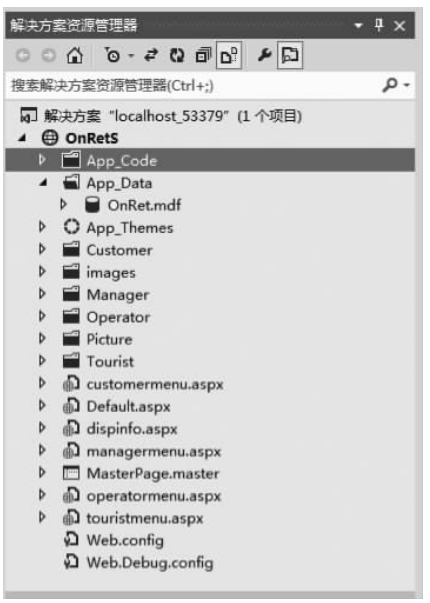


图 3.1 OnRets 网站的结构

1. 常见的文件类型

从文件组成看，网站由多个文件组成，包括 Web 网页、配置文件和数据库文件等。常见的文件类型如下。

- *.aspx：ASP .NET 网页文件，包含了用户界面的代码，如图 3.1 中 Default.aspx 是 OnRetS 网站的主页。
- *.cs：采用代码隐藏页模型设计网页时的代码隐藏文件，如选择 C# 作为开发语言就产生.cs 文件。
- *ascx：开发人员自己设计的 ASP .NET 用户控件。
- web.config：ASP .NET 应用程序的基于 XML 格式的配置文件，包含各种 ASP .NET 功能的配置信息，如数据库连接、安全设置、状态管理等。Web.Debug.config 是调试模式下的 web.config 文件。
- global.asax：全局应用程序文件，该文件驻留在 ASP .NET 网站的根目录下。
- *.asmx 文件：为其他计算机提供共享应用程序的 Web 服务。
- *.skin 文件：外观文件，用于设置主题。
- *.css 文件：样式表文件。
- *.master：母版页文件。例如，图 3.1 中 MasterPage.master 就是一个母版页文件，用作 Default.aspx 等网页的母版页。

2. 常用的目录

一个网站包含若干目录。为了更方便管理和使用，ASP .NET 保留了一些可用于特定内容的文件和目录名称。一个网站常用的目录如下。

- App_Data 目录：用于存放应用程序使用的数据库。它是一个集中存储应用程序所用数据库的地方。App_Data 目录可以包含 SQL Server 数据库文件(.mdf)、Access 数据库文件(.mdb)或 XML 文件等。例如，在图 3.1 中，App_Data 目录中存放 OnRet 数据库文件。
- App_Code 目录：用于存放所有应当作为应用程序的一部分动态编译的类文件。在开发网站时，对 App_Code 目录的更改会导致整个应用程序的重新编译。例如，在图 3.1 中，App_Code 目录存放网站的一些通用 C# 代码。
- Bin 目录：包含应用程序所需的，用于控件、组件或需要引用的任何其他代码的可部署程序集。该目录中存在的任何.dll 文件将自动地链接到应用程序。

- App_Themes 目录：用于存放主题文件和 CSS 文件等。例如，在图 3.1 中，App_Themes 目录存放 Blue 主题和 StyleSheet.css 文件。
- App_GlobalResources 目录：是全局资源文件目录，可以存放一些资源文件，这些资源文件中包含指向图像或其他数据的资源字符串，在网站的所有网页中都可以访问这些资源字符串。
- 其他目录：网站开发人员可以根据需要自定义目录。例如，在图 3.1 中，images 和 Picture 目录用于存放图片文件，Operator 用于存放操作员网页文件等。

3.1.2　ASP .NET 网站创建过程

下面通过一个例子说明创建 ASP .NET 网站的过程。

【练一练】 创建一个文件系统网站 CH3 的操作步骤如下：

① 启动 Visual Studio 2012。

② 选择“文件|新建|网站”命令，出现“新建网站”对话框，单击模板列表中 Visual C# 项，列出已安装的网站模板。各网站模板的说明如下。

- ASP .NET 空网站：该模板只包含一个配置文件(web.config)。本书中主要使用该模板构建网站示例，并逐步添加文件和目录。
- ASP .NET 窗体网站：该模板用于配置一个基本的 ASP .NET 网站，包含许多文件和目录用于开始网站的开发。
- ASP .NET 网站(Razor v1 或 Razor v2)：通过微软的 Web Pages 框架，使用这些模板来创建网站。
- ASP .NET Dynamic Data 实体网站：该模板用于创建灵活且强大的 Web 网站来管理数据库中的数据，而不需要手工输入许多代码。
- WCF 服务：该模板用于创建包含一个或多个 WCF 服务的网站。
- ASP .NET 报表网站：该模板用于创建企业报表网站。

这里选择“ASP .NET 空网站”模板。

③ “Web 位置”选项有如下 3 种。

- 文件系统：如果主机没有安装 IIS，也不想设置服务器的位置等信息，可以使用这个设置，表示代码源文件放在选定本地文件目录中。Visual Studio 会把用户指定的路径视为该网站的根目录，并在预览时启动内置的网页服务器，根据这个位置来模拟执行，并可以很方便地进行程序源代码移植。
- HTTP：如果主机已经安装了 IIS，便可以使用这个设置。这个设置与 IIS 的设置相关，还必须设置网页服务器的预览网址，所设计的文件也会放置在 IIS 所设置网站的根目录中。在根目录下还可以设置若干虚拟目录。
- FTP：如果测试主机并不在本机上，可以使用这个设置，表示将代码源文件保存在远程的 FTP 服务器上。Visual Studio 通过文件传输协议 FTP 访问网站，这样更容易访问其他服务器上的网站。在第一次运行 FTP 网站时提示用户指定服务器 URL。

这里选择“Web 位置”为“文件系统”。

④ 单击“浏览”按钮，选择“D:\电子商务\CH3”目录，如图 3.2 所示，单击“确定”按钮。这样就创建了一个空网站 CH3，其中只有一个配置文件 web.config。

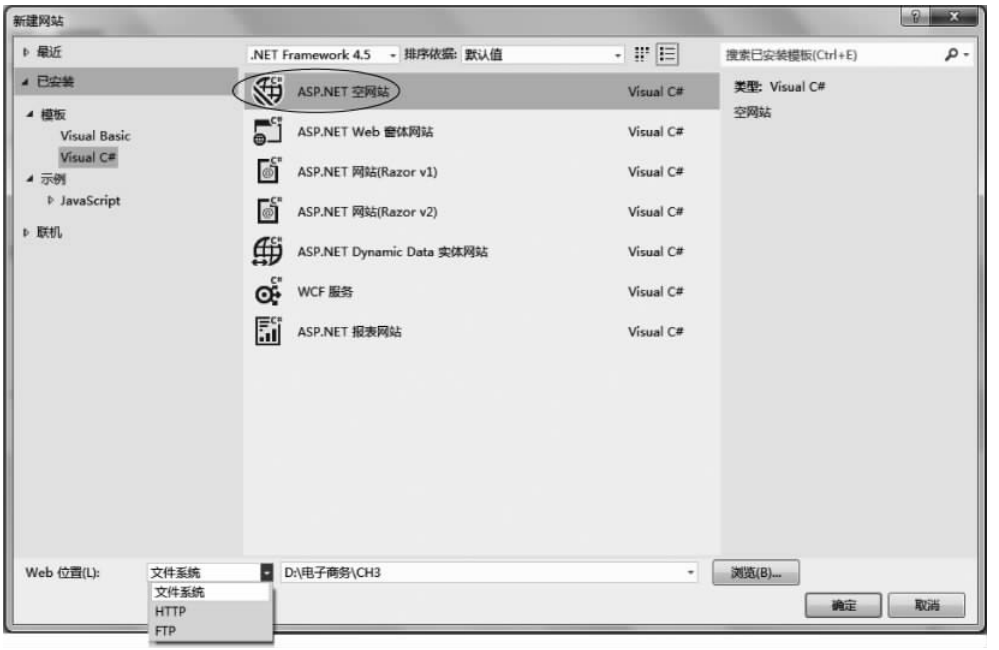
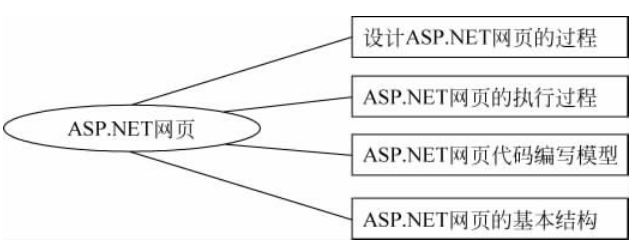


图 3.2　网站的模板

3.2　ASP .NET 网页

知识梳理



3.2.1　设计 ASP .NET 网页的过程

下面通过一个例子说明创建 ASP .NET 网页的过程。

【练一练】 在前面创建的 CH3 网站中设计一个 webform1 网页，其功能类似于 1.2.5 节介绍的静态网页 spage.html。

其设计步骤如下：

① 启动 Visual Studio 2012。

② 选择“文件|打开|网站”命令，出现如图 3.3 所示的“打开网站”对话框。

其中，左边的列表表示打开网站的类型。

- 文件系统：从磁盘位置打开网站。
- 本地 IIS：使用本地计算机上的 IIS 打开网站。

图 3.3　“打开网站”对话框

- FTP 站点：从 FTP 位置打开网站。
- 远程站点：打开 FrontPage 网站。
- 源代码管理：从源代码管理打开网站。

这里选择“文件系统”，并单击“D:\电子商务\CH3”文件，单击“打开”按钮。

③ 出现如图 3.4 所示的 Microsoft Visual Studio 对话框提示是否使用 IIS Express。因为在 Visual Studio 中开发网站时，需要 Web 服务器才能测试或运行它们。Visual Studio 2010 之前的版本内置了 Visual Studio Development Server(Visual Studio 开发服务器)用于测试或运行网站，Visual Studio 2010 版本开始有了 IIS Express。在 Visual Studio 2012 中可以使用 Visual Studio 开发服务器和 IIS Express 测试或运行网站，最好使用 IIS Express，因为

图 3.4　Microsoft Visual Studio 对话框

微软公司表示以后的 Visual Studio 版本可能不再支持 Visual Studio 开发服务器。

单击“是”按钮，表示使用 IIS Express。此时就打开 CH3 网站。

④ 在 CH3 网站中，选择“网站|添加新项”命令，出现“添加新项”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform1. aspx，其他保持默认项，如图 3. 5 所示，单击“添加”按钮，这样在 CH3 网站中就添加了一个名称为 webform1. aspx 的空网页。

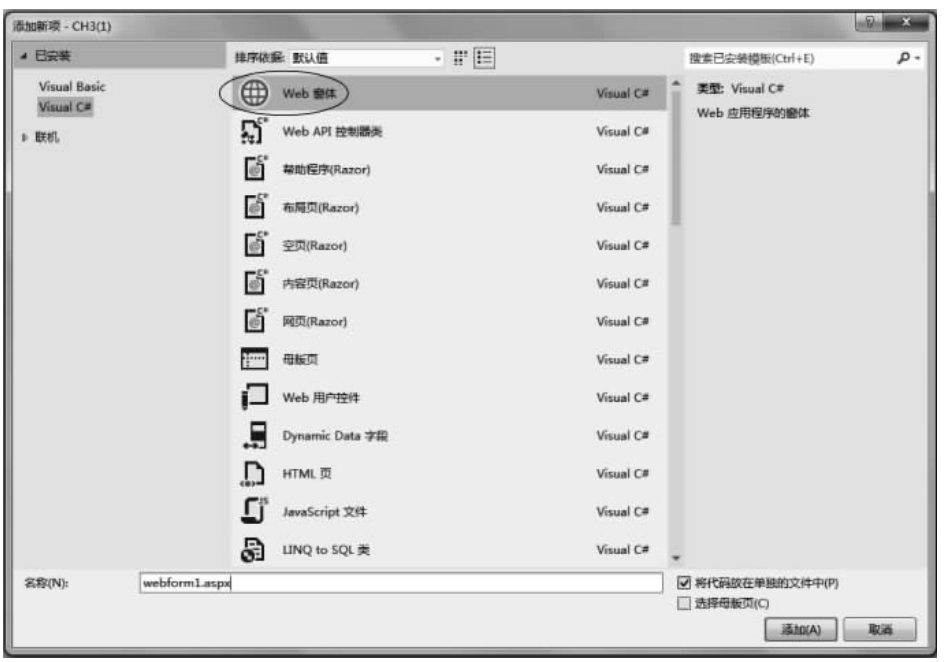


图 3.5 “添加新项”对话框

⑤ 在系统集成界面的中间部分出现 webform1. aspx 网页的源视图代码，单击下方的“设计”选项卡，进入网页的可视化设计界面，从工具箱中拖曳两个标签到网页中，名称分别为 Label1 和 Label2，将它们的 Text 属性分别修改为“-电子商务网站-”和“欢迎光临”，再通过属性窗口修改它们的字体，如图 3. 6 所示。

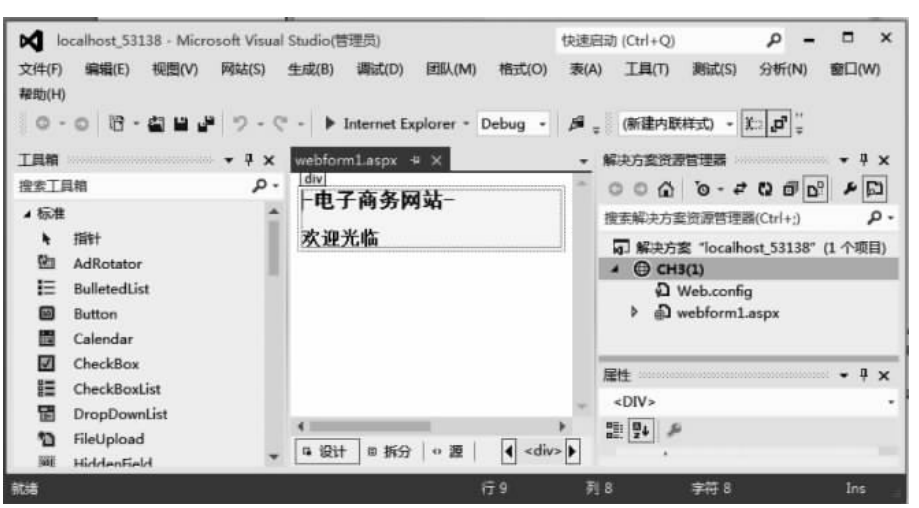


图 3.6 “添加新项”后的显示

说明：网页源视图代码编辑窗口下方有 3 个选项卡，其中 设计 选项卡用于可视化设计网页；源 选项卡用于显示网页的源视图代码；拆分 选项卡用于显示网页元素与对应的源视图代码关联。

⑥ 单击 源 选项卡，看到 webform1 网页的源视图代码（称为 ASP .NET 网页代码）如下：

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="webform1.aspx.cs" Inherits="webform1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
   <head runat="server">
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
      <title>webform1 网页</title>
   </head>
   <body>
     <form id="form1" runat="server">
       <div>
        <asp:Label ID="Label1" runat="server" Text="-电子商务网站-"
            style="font-size: large; font-weight: 700; font-family: 黑体">
        </asp:Label>
        <br /><br />
        <asp:Label ID="Label2" runat="server" Text="欢迎光临"
            style="font-size: medium; font-weight: 700">
        </asp:Label>
       </div>
    </form>
   </body>
</html>
```

从中看到，Visual Studio 将开发人员的可视化设计操作转换为相应的网页代码。例如，在网页中拖曳一个标签 Label1 并设置其属性的代码如下：

```
<asp:Label ID="Label1" runat="server" Text="-电子商务网站-"
    style="font-size: large; font-weight: 700; font-family: 黑体"></asp:Label>
```

其中，“asp:”是 ASP .NET 控件标识符；ID 属性指定标识号（网页中每个元素的标识号是唯一的）；style 定义该标签在浏览器中的显示样式，包括字体和字体大小等。

⑦ 单击工具栏的 Internet Explorer（将 IE 浏览器设置为默认的浏览器）浏览该网页。在第一次浏览时，系统会询问是否要在＜Web.config＞文件中添加调试功能，如图 3.7 所示，单击“确定”按钮即可。在以后浏览时就不再出现询问对话框。如果按 Ctrl＋F5 键，表示在非调试状态浏览网页，会立即出现 webform1 网页的浏览界面。webform1 网页执行界面如图 3.8 所示。

图 3.7　“未启用调试”对话框

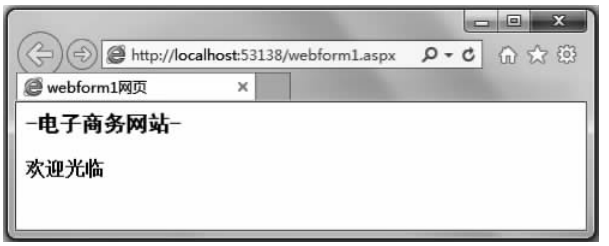


图 3.8　webform1 网页执行界面

3.2.2　ASP .NET 网页的执行过程

在图 3.8 的浏览界面中右击，在出现的快捷菜单中选择“查看源”命令，显示的纯 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head><meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>webform1 网页</title>
    </head>
    <body>
        <form method="post" action="webform1.aspx" id="form1">
          <div class="aspNetHidden">
                <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
                    value="0XDZ3/n/lKuptIJQ8PIKqKZR1iC40XFJWTjE/jCLRI91ZJ
                    u3WyJ+jV+ZaEXUE2xpmDiqEliCNQmH8Vq4jcXRpPwjmW4oJxqb/e0HwxuyDuA=" />
          </div>
          <div class="aspNetHidden">
              <input type="hidden" name="__VIEWSTATEGENERATOR"
                  id="__VIEWSTATEGENERATOR" value="C687F31A" />
          </div>
          <div>
              <span id="Label1" style="font-size: large; font-weight: 700; font-family:
                  黑体">-电子商务网站-</span>
              <br />
              <br />
              <span id="Label2" style="font-size: medium; font-weight: 700">
                  欢迎光临</span>
          </div>
        </form>
    </body>
</html>
```

上述代码是纯 HTML。其中有两个隐藏域，name 为_VIEWSTATE 的隐藏域，只有 form 表单(窗体)标识 runat 为 server 时才会自动生成。当请求某个网页时，ASP .NET 把所有控件的状态序列化成一个字符串，然后作为网页的隐藏属性送到客户端。当客户端把网页回传时，ASP .NET 分析回传的网页窗体属性，并赋给控件对应的值。name 为_VIEWSTATEGENERATOR 的隐藏域，用于在 ASP .NET 运行时帮助确定是回传到相同的网页还是跨页。

从中看到，ASP .NET 网页代码与浏览器中最终显示的纯 HTML 代码是不同的，这种转换是由 ASP .NET 引擎完成的。实际上，像 IE 这样的浏览器并不认识 ASP .NET 网页代码。

一个典型的 ASP .NET 网页执行过程如图 3.9 所示。图中的各个步骤说明如下：

① 客户机通过浏览器发出 Web 请求(请求访问某个 ASP .NET 网页)。

② Web 服务器收到 ASP .NET 动态网页请求。

③ Web 服务器从硬盘的指定位置查找相应的 ASP .NET 动态网页文件。

④ 将硬盘中找到的 ASP .NET 动态网页文件返回给 Web 服务器。

⑤ Web 服务器将 ASP .NET 动态网页发给 ASP .NET 引擎。

⑥ ASP .NET 引擎会逐行地读取该文件,并执行文件中的程序代码(脚本),如果需要访问数据库,则将这部分代码发给数据库服务器;如果不需要访问数据库,则直接转到步骤⑧。

⑦ 数据库服务器执行数据库访问,并将结果返回给 ASP .NET 引擎。

⑧ ASP .NET 引擎生成最终的纯 HTML 文件并返回给 Web 服务器。

⑨ Web 服务器将该纯 HTML 文件发送给客户机。

⑩ 客户机收到请求的纯 HTML 文件,并在浏览器中以图形方式将 HTML 标记显示在计算机屏幕上。

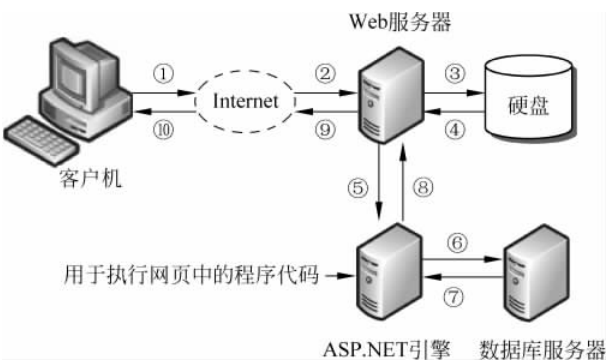


图 3.9　ASP .NET 网页的执行由 ASP .NET 引擎处理

因此,ASP .NET 网页是在服务器上执行的,通过 ASP .NET 引擎处理将其转换为纯 HTML 代码,然后由浏览器执行。图 3.10 所示是 Label1 控件的转换过程。

.aspx 网页代码

```
<asp:Label ID="Label1" runat="server" Text="-电子商务网站-"
  style="font-size: large; font-weight: 700; font-family: 黑体">
</asp:Label>
```

ASP.NET 引擎转换

```
<span id="Label1" style="font-size: large; font-weight: 700;
font-family:黑体">-电子商务网站-</span>
```

纯 HTML 代码

图 3.10　ASP .NET 进行的代码转换

3.2.3　ASP .NET 网页代码编写模型

在 ASP .NET 网页中,开发人员的编程工作分为两个部分:可视元素和编程逻辑。可视元素部分包括标记、服务器控件和静态文本的文件组成。编程逻辑部分包括事件处理程序和其他代码,这些代码由用户创建与网页进行交互。程序代码可以驻留在网页的 script 块或单

独的类中。如果代码在单独的类文件中，则该文件称为“代码隐藏”文件。在本书中编程逻辑代码使用C#语言编写。

因此，ASP .NET提供两种代码编写模型：单文件页模型和代码隐藏页模型。这两个模型功能相同，在两种模型中可以使用相同的控件和代码。

1. 单文件页模型

在单文件页模型中，网页的标记及其程序代码位于同一个.aspx文件中，也称为内联模型。程序代码位于＜script＞元素中，它包含runat＝"server"属性，此属性将其标记为ASP .NET引擎应执行的代码。

【练一练】 在CH3网站中添加一个单文件页模型的网页webform2，其功能是实现用户输入的两个整数相加运算。设计步骤如下：

① 进入CH3网站，选择“网站|添加新项”命令，出现的“添加新项”对话框，在中间列表中选择“Web窗体”，将文件名称改为webform2.aspx，去掉“将代码放在单独的文件中”的勾选，如图3.11所示，单击“添加”按钮，这样在CH3网站中添加一个名称为webform2.aspx的单文件页模型网页。

图3.11　添加webform2网页

② 单击 设计 选项卡，进入网页的可视化设计界面，从工具箱中拖曳3个标签Label1～Label3、两个文本框TextBox1和TextBox2、一个命令按钮Button1到设计界面中。通过属性窗口将Label1和Label2的Text属性分别更改为“整数1”和“整数2”。按下Ctrl键，选择Label1和Label2，选择“格式|A字体”命令，设计其字体如图3.12所示。采用同样的操作设置其他控件的Text属性和字体。最终的设计界面如图3.13所示。

③ 双击“相加”按钮，在＜script runat＝"server"＞和＜/script＞之间添加如下代码：

```
protected void Button1_Click(object sender, EventArgs e)
{    int a = int.Parse(TextBox1.Text.Trim());
     int b = int.Parse(TextBox2.Text.Trim());
     int c = a + b;
     Label3.Text = "两个整数相加结果: " + c.ToString();
```

}

图 3.12　“字体”对话框

图 3.13　webform2 网页设计界面

④ 单击 源 选项卡，看到 webform2 网页的源视图代码如下：

```
<%@ Page Language="C#" %>
<!DOCTYPE html>
<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
    {   int a = int.Parse(TextBox1.Text.Trim());
        int b = int.Parse(TextBox2.Text.Trim());
        int c = a + b;
        Label3.Text = "两个整数相加结果: " + c.ToString();
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
   <head runat="server">
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
      <title></title>
      <style type="text/css">
         .auto-style1 {
            font-family: 楷体;
            font-weight: bold;
            font-size: medium;
            color: #0000FF;
         }
      </style>
   </head>
   <body>
      <form id="form1" runat="server">
        <div>
          <asp:Label ID="Label1" runat="server" Text="整数 1"
              CssClass="auto-style1"></asp:Label>
          <asp:TextBox ID="TextBox1" runat="server" Width="74px"></asp:TextBox>
            
```

```
            <asp:Label ID = "Label2" runat = "server" Text = "整数 2"
                CssClass = "auto - style1"></asp:Label >
            <asp:TextBox ID = "TextBox2" runat = "server" Width = "80px"></asp:TextBox >
          </div >
        <div >
          <br />
                   
          <asp:Button ID = "Button1" runat = "server" Text = "相加" OnClick = "Button1_Click"
             style = "color: #FF0000; font - size: medium; font - weight: 700;
                  font - family: 黑体" />
        </div >
        <div >
          <br />
          <asp:Label ID = "Label3" runat = "server"
              style = "color: #FF00FF; font - size: medium; font - weight: 700;
                   font - family: 仿宋"></asp:Label >
        </div >
      </form >
    </body >
  </html >
```

说明：＜style＞元素中的.auto-style1 样式是由 Visual Studio 自动生成的，将开发人员的可视化设计操作转换为样式并自动通过 CssClass 属性作用于相关的可视元素。

⑤ 单击工具栏的 Internet Explorer 浏览该网页。在两个文本框中分别输入 8 和 15，如图 3.14 所示，单击“相加”按钮，其执行结果如图 3.15 所示。

图 3.14　webform2 网页执行界面一

图 3.15　webform2 网页执行界面二

从本例看出，单文件页模型只对应一个文件，即 webform2.aspx。其中，包含可视元素，如 Label1、TextBox1 和 Button1 等；又包含编程逻辑，如 Button1 的单击事件处理方法 Button1_Click，这些编程逻辑代码放置在＜script＞元素中。

2. 代码隐藏页模型

代码隐藏模型的特点是，在一个文件(*.aspx)中存放可视元素代码，在另一个文件(*.aspx.cs)中存放编程逻辑代码。代码文件的名称会根据所使用的编程语言而有所变化。

这种模型的优点是：可以清楚地分隔标记(HTML 代码)和代码(编程逻辑)，适用于包含大量代码或多个开发人员共同创建网站的 Web 应用程序；便于程序的维护和升级；程序代码可在多个网页中重用等。

【练一练】 在 CH3 网站中添加一个代码隐藏模型的网页 webform3，其功能与 webform2 网页相同。设计步骤如下：

① 进入 CH3 网站，选择“网站|添加新项”命令，出现的“添加新项”对话框，在中间列表中

选择“Web 窗体”，将文件名称改为 webform3.aspx，保留“将代码放在单独的文件中”的勾选，单击“添加”按钮，这样在 CH3 网站中添加一个名称为 webform3.aspx 的代码隐藏模型网页。

② 其他操作与 webform2 网页设计相同。

③ 单击 源 选项卡，看到 webform3 网页的源视图代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="webform3.aspx.cs"
    Inherits="webform3" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <style type="text/css">
        .auto-style1 {
            font-family: 楷体;
            font-weight: bold;
            font-size: medium;
            color: #0000FF;
        }
    </style>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <asp:Label ID="Label1" runat="server" Text="整数 1"
            CssClass="auto-style1"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server" Width="74px"></asp:TextBox>
          
        <asp:Label ID="Label2" runat="server" Text="整数 2"
            CssClass="auto-style1"></asp:Label>
        <asp:TextBox ID="TextBox2" runat="server" Width="80px"></asp:TextBox>
      </div>
      <div>
        <br />
             
        <asp:Button ID="Button1" runat="server" Text="相加" OnClick="Button1_Click"
        style="color:#FF0000;font-size:medium;font-weight:700;font-family:黑体" />
      </div>
      <div>
       <br />
       <asp:Label ID="Label3" runat="server"
        style="color:#FF00FF;font-size:medium;font-weight:700;font-family:仿宋" />
      </div>
    </form>
  </body>
</html>
```

④ 在“解决方案资源管理器”中双击 webform3.aspx.cs 文件，看到其代码如下：

```
using System;
public partial class webform3 : System.Web.UI.Page
{    protected void Page_Load(object sender, EventArgs e)
```

```
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {   int a = int.Parse(TextBox1.Text.Trim());
            int b = int.Parse(TextBox2.Text.Trim());
            int c = a + b;
            Label3.Text = "两个整数相加结果: " + c.ToString();
        }
}
```

webform3 网页的执行与 webform2 完全相同。

从中看到，代码隐藏模型的 webform3 网页的文件结构如图 3.16 所示，网页的可视元素和编程逻辑分离存放。在 webform3.aspx 文件的头部包含如下页面指令：

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform3.aspx.cs"
    Inherits = "webform3" %>
```

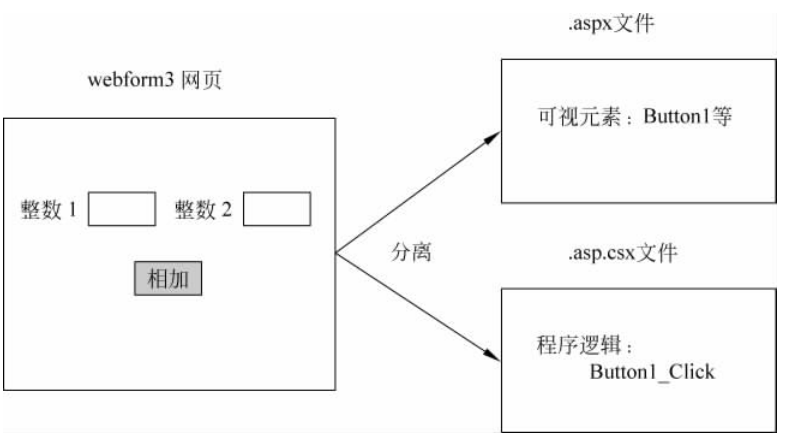


图 3.16　webform3 网页的文件结构

其作用是，指定 ASP .NET 网页编译器使用 C# 作为网页的服务器端代码语言，自动绑定网页的事件，该网页代码隐藏文件为 webform3.aspx.cs，供页面继承的代码隐藏类为 webForm3。也就是说，通过 CodeFile 和 Inherits 属性将分离存放的两个文件关联。

说明：对于代码隐藏模型的网页，即使不设计任何编程逻辑代码，如前面的 webform1 网页，也是采用两个文件存储的。

3.2.4　ASP .NET 网页的基本结构

ASP .NET 网页的基本结构是模块化的，每个 aspx 网页文件一般包含 3 个独立部分的部分：页面指令、代码脚本块和页面内容。

1. 页面指令

页面指令用于设置该页面的运行环境，规定 ASP .NET 如何处理该页面，控制 ASP .NET 页面的行为，一个页面可包含多条页面指令。页面指令不作为发送到浏览器标记的一部分呈现。当使用页面指令时，虽然标准的做法是将指令写在文件的开头，但是它们可以位于.aspx 或.ascx 文件中的任何位置。每个指令都可以包含一个或多个特定于该指令的属性(与值成对出现)。常见的 ASP .NET 页面指令如表 3.1 所示。

表 3.1　常见的页面指令

页 面 指 令	说　　明
@Page	定义 ASP .NET 页分析器和编译器使用的页(.aspx 文件)特定属性
@Import	将命名空间显式导入网页中,使所导入的命名空间的所有类和接口可用于该网页页。导入的命名空间可以是 .NET Framework 类库或用户自定义的命名空间的一部分
@OutputCache	以声明的方式控制 ASP .NET 页或页中包含的用户控件的输出缓存策略
@Implements	指示当前或用户实现指定的 .NET Framework 接口
@Register	将别名与命名空间及类名关联,以便在自定义服务器控件语法中使用简明的表示法
@Assembly	在编译过程中将程序集链接到当前页,以使程序集的所有类和接口都可用在该页上
@Control	定义 ASP .NET 页分析器和编译器使用的用户控件(.ascx 文件)特定属性。该指令只能用于用户控件
@Master	标识 ASP .NET 母版页
@MasterType	为 ASP .NET 网页的 Master 属性分配类名,使该页可以获取对母版页成员的强类型引用
@PreviousPageType	提供用于获得上一网页的强类型的方法,可通过 PreviousPage 属性访问上一网页
@Reference	以声明的方式指示,应该根据在其中声明此指令的网页对另一个用户控件或页源文件进行动态编译和链接

其中最常用的是@ Page 指令。该指令允许开发人员为网页指定多个配置选项,并且该指令只能在 Web 窗体网页中使用。每个.aspx 文件只能包含一条@Page 指令。该指令可以指定页面中代码的服务器编程语言、页面是将服务器代码直接包含在其中(即单文件页面),还是将代码包含在单独的类文件中(即代码隐藏页面)、调试和跟踪选项、页面是否为某母版页的内容页等。

其基本格式如下:

```
<%@ Page 属性 = "值" [属性 = "值" …] %>
```

@Page 指令的属性及其说明如表 3.2 所示。若要定义@Page 指令的多个属性,需要使用一个空格分隔每个属性/值对。对于特定属性,不要在该属性与其值相连的等号(=)两侧加空格。

表 3.2　@Page 指令的属性及其说明

属　　性	说　　明
AutoEventWireup	指示页面的事件是否自动绑定。如果启用了事件自动绑定,则为 true(默认值),否则为 false
Buffer	确定是否启用了 HTTP 响应缓冲。如果启用了页面缓冲,则为 true(默认值),否则为 false
ClassName	一个字符串,指定在请求页面时将自动进行动态编译的页面的类名。此值可以是任何有效的类名,并且可以包括类的完整命名空间(完全限定的类名)。如果未指定该属性的值,则已编译页面的类名将基于页面的文件名
CodeBehind	指定包含页面关联类的已编译文件的名称,该属性不能在运行时使用
CodeFile	指定指向页面引用的代码隐藏文件的路径

续表

属　性	说　明
CodePage	指示用于响应的编码方案的值，该值是一个用作编码方案 ID 的整数
Description	提供该页面的文本说明，ASP .NET 分析器忽略该值
ErrorPage	定义在出现未处理页面异常时用于重定向的目标 URL
Inherits	定义供页面继承的代码隐藏类，与 CodeFile 属性(包含指向代码隐藏类的源文件的路径)一起使用
Language	指定在对页面中的所有内联呈现和代码声明块进行编译时使用的语言。只可以表示任何 .NETFramework 支持的语言，如 C#
MasterPageFile	设置内容页面的母版页面或嵌套母版页面的路径，支持相对路径和绝对路径
Title	指定在响应的 HTML<title>标记中呈现的页面的标题，也可以通过编程方式将标题作为页面的属性来访问
Trace	指示是否启用跟踪。如果启用了跟踪，则为 true，否则为 false(默认值)

例如，在前面的 webform3.aspx 网页文件中就看到了@ Page 页面指令的作用。

2. 代码脚本块

代码脚本块是由“<script runat=server></script>”标记对括起来的程序代码。在代码脚本块中可以定义页面的全局变量及控件事件处理程序等，这些程序代码要先编译后执行。代码脚本块采用 C# 和 Javascript 等编写。

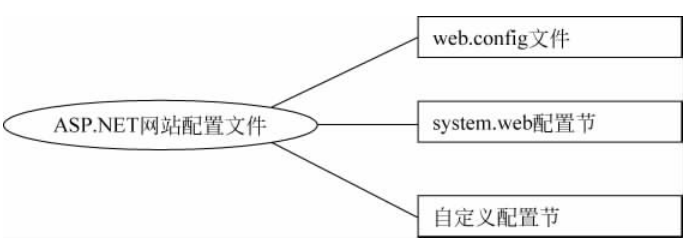
如果采用单文件页模型，代码脚本块放置在“<script runat=server>…</script>”中。如果采用代码隐藏页模型，代码脚本块放置在单独的文件中。

3. 页面内容

ASP .NET 网页的页面内容是由 ASP .NET 控件构成的，有关 ASP .NET 控件的内容在第 7 章介绍。在一个网页中合理地布局 ASP .NET 控件是十分重要的。

3.3 ASP .NET 网站配置文件

知识梳理



3.3.1 web.config 文件

在创建一个网站时，总会在网站根目录中自动建立一个 web.config 文件，它就是网站的配置文件。

web.config 文件是一个 XML(可扩展标记语言)文本文件。XML 与 HTML 相似，同属

于标记语言，但和 HTML 不同，XML 被设计用来传输和存储数据，其焦点是数据的内容；而 HTML 被设计用来显示数据，其焦点是数据的外观。另外，在 HTML 中，所有标记是预先定义的，每个标记都有特定的功能；而 XML 中标记并没有被预先定义，需要自行定义标记。

web.config 文件用来存储 ASP .NET 应用程序的配置信息（如最常用的设置 ASP .NET 应用程序的身份验证方式）。实际上，web.config 文件可以出现在应用程序的每一个目录中，根目录中的 web.config 文件包括默认的配置设置，所有的子目录都继承它的配置设置。

如果想修改子目录的配置设置，可以在该子目录下新建一个 web.config 文件，它可以提供除从父目录继承的配置信息以外的配置信息，也可以重写或修改父目录中定义的设置。

web.config 文件的根节点是＜configuration＞，在＜configuration＞节点下的常见子节点有＜system.web＞、＜appSettings＞和＜connectionStrings＞。这是一种嵌套结构，每个子节点中又可以包含若干子节点。

例如，最基本的 web.config 文件的内容如下：

```
<?xml version = "1.0"?>
<! --
  有关如何配置 ASP .NET 应用程序的详细信息，请访问
  http://go.microsoft.com/fwlink/?LinkId = 169433
  --->
<configuration>
  <system.web>
    <compilation debug = "true" targetFramework = "4.5"/>
    <httpRuntime targetFramework = "4.5"/>
  </system.web>
</configuration>
```

3.3.2　system.web 配置节

＜system.web＞节主要是网站运行时的一些配置，它的常见节点如下：

1. ＜compilation＞节

＜compilation＞节用于配置 ASP .NET 使用的编译设置。默认的 debug 属性为 true，即允许调试，在这种情况下会影响网站的性能，所以在程序编译完成交付使用之后应将其设为 false。例如，若要禁止调试，设置＜compilation＞节如下：

```
<compilation
    debug = "false" >
</compilation>
```

2. ＜authentication＞节

＜authentication＞节用于为 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。其 mode 属性指定身份验证方案，它是必选的属性，其取值如表 3.3 所示。

其子元素有，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）。

表 3.3 mode 属性的取值及其说明

取 值	说 明
Windows(默认值)	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。在这种情况下的应用程序将身份验证责任委托给基础 IIS
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	不指定任何身份验证。应用程序仅期待匿名用户，否则它将提供自己的身份验证

例如，以下示例为基于窗体(Forms)的身份验证配置站点，当没有登录的用户访问需要身份验证的网页，网页自动跳转到登录网页：

```
<authentication mode = "Forms">
    <forms loginUrl = "login.aspx" name = "FormsAuthCookie"/>
</authentication>
```

其中，元素 loginUrl 表示登录网页的名称；name 表示 Cookie 名称。

3. <authorization>节

<authorization>节用于控制对 URL 资源的客户端访问(如允许匿名用户访问)。此元素可以在任何级别(计算机、站点、应用程序、子目录或网页)上声明，与<authentication>节配合使用。其子元素有，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访问的授权规则。

例如，以下示例禁止匿名用户的访问：

```
<authorization>
    <deny users = "?"/>
</authorization>
```

其中，users 属性指出可访问的账号；? 表示以匿名方式访问；* 表示多有用户。

4. <customErrors>节

<customErrors>节用于为 ASP.NET 应用程序提供有关自定义错误信息的信息，不适用于 Web 服务中发生的错误。其子元素为 error(可选)，用于指定给定 HTTP 状态代码的自定义错误页。其属性及其说明如表 3.4 所示。

表 3.4 <customErrors>节的属性及其说明

属 性	说 明
defaultRedirect	可选的属性。指定出错时将浏览器定向到的默认 URL。如果未指定该属性，则显示一般性错误。URL 可以是绝对的(如 www.contoso.com/ErrorPage.htm)或相对的。相对 URL(如/ErrorPage.htm)是相对于为该属性指定 URL 的 web.config 文件，而不是相对于发生错误的网页。以波形符(~)开头的 URL(如~/ErrorPage.htm)表示指定的 URL 是相对于应用程序的根路径
mode	必选的属性。指定是启用或禁用自定义错误，还是仅向远程客户端显示自定义错误。此属性可以为下列值之一。On：指定启用自定义错误，如果未指定 defaultRedirect，用户将看到一般性错误，会向远程客户端和本地主机显示自定义错误。Off：指定禁用自定义错误，会向远程客户端和本地主机显示详细的 ASP.NET 错误。RemoteOnly(默认值)：指定仅向远程客户端显示自定义错误并且向本地主机显示 ASP.NET 错误

例如，在以下配置中，当网站运行发生错误时将自动跳转到自定义的错误网页 ErrorPage.aspx：

```
<customErrors defaultRedirect = "ErrorPage.aspx" mode = "RemoteOnly"> </customErrors >
```

3.3.3　自定义配置节

可以在＜appSettings＞节和＜connectionStrings＞节设置一些应用程序的设置项，即自定义配置。

1. ＜appSettings＞节

此节用于定义应用程序设置项。对一些不确定设置，还可以让用户根据实际情况设置。例如，在其中添加用于存储数据库连接字符串的子节点。当然，如果程序需要其他自定义的全局配置信息，也可以在此添加相应的子节点。

appSettings 元素的子元素有：add(可选的子元素)向应用程序设置集合添加名称/值对形式的自定义应用程序设置；clear(可选的子元素)移除所有对继承自定义应用程序设置的引用，仅允许由当前 add 属性添加的引用；remove(可选的子元素)从应用程序设置集合中移除对继承自定义应用程序设置的引用。

例如，在 web.config 文件中的＜appSettings＞节中，采用＜add＞添加了一个与 SQL Server 数据库 Stud 连接的子节点和一个 Web 服务子节点：

```
<appSettings >
    <remove key = "mydata" />
    <add key = "mydata" value = "我的数据" />
</appSettings >
```

对于＜appSettings＞节中的子节点，可以使用 ConfigurationManager.AppSettings["key 值"]来读取这些子节点值。例如：

```
mystr = System.Configuration.ConfigurationManager.
   AppSettings["myconnstring"].ToString();
```

其中，ConfigurationManager 类位于 System.Configuration 命名空间。这样，等价于如下语句：

```
mystr = "我的数据";
```

2. ＜connectionStrings＞节

在 ASP .NET 中，如会话、成员资格、个性化设置和角色管理器等功能均依赖于存储在 connectionStrings 元素中的连接字符串，还可以使用 connectionStrings 元素来存储应用程序的连接字符串。

connectionStrings 元素的子元素有：add 子元素向连接字符串集合添加名称/值对形式的连接字符串；clear 子元素移除所有对继承的连接字符串的引用，仅允许那些由当前 add 元素添加的连接字符串；remove 子元素从连接字符串集合中移除对继承连接字符串的引用。

例如，在 web.config 文件中的＜connectionStrings＞节中，采用＜add＞添加了一个与 SQL Server 数据库 school 连接的子节点：

```
connectionStrings>
    <add name = "myconnstring"
        connectionString = "Data Source = LCB - PC; Initial Catalog = school;
            Integrated Security = True"
     providerName = "System.Data.SqlClient" />
  </connectionStrings>
```

对于＜connectionStrings＞节中的子节点的 Web 应用程序配置信息，可以使用 ConfigurationManager.ConnectionSettings["key 值"].ToString()来读取这些子节点值。例如：

```
mystr = System.Configuration.ConfigurationManager.
          ConnectionStrings["myconnstring"].ToString();
```

这样，等价于如下语句：

```
mystr = "Data Source = LCB - PC; Initial Catalog = school; Integrated Security = True";
```

如果整个网站中只有一个网页需要使用 myconnstring，这样做的意义不大，但若有多个网页需要，这样设计就十分必要。当 myconnstring 发生更改时，只需要更改＜connectionStrings＞节中相应的定义即可。

3.4 练　习　题

1. 单项选择题

(1) 创建 ASP .NET 网站时，“新建网站”对话框中“Web 位置”选项不能是(　　)。

A. 文件系统　　B. HTTP　　C. FTP　　D. ASP .NET

(2) 文件系统网站适合于学习使用，因为(　　)。

A. 不需要安装 IIS　　B. 网站允许放置在任意目录下

C. 能够进行单独测试　　D. A 和 B

(3) 以下叙述中正确的是(　　)。

A. 一个 ASP .NET 网站至少包含一个网页文件

B. 一个 ASP .NET 网站至少包含一个数据库文件

C. 一个 ASP .NET 网站至少包含一个应用程序类文件

D. 一个 ASP .NET 网站至少包含一个图像文件

(4) 采用“ASP .NET 空网站”模板创建一个空网站时，该网站包含(　　)。

A. 一个 Default.aspx 网页文件

B. 一个配置文件(web.config)

C. 一个 SheetStyle.css 样式文件

D. 一个 SkinFile.skin 外观文件

(5) 一个 ASP .NET 网站包含若干目录，通常数据库文件放在(　　)目录中。

A. App_Data　　B. App_Code　　C. App_Themes　　D. Bin

(6) 一个 ASP .NET 网站包含若干目录，通常应用程序的类文件放在(　　)目录中。

A. App_Data　　B. App_Code　　C. App_Themes　　D. Bin

(7) ASP .NET 网页开发有单文件页模型和代码隐藏页模型两种,以下叙述正确的是(　　)。

A. 代码隐藏页模型的网页开发功能更强,而单文件页模型的网页开发功能较弱

B. 单文件页模型的网页开发功能更强,而代码隐藏页模型的网页开发功能较弱

C. 两种在网页开发上功能是等价的

D. 以上都不对

(8) ASP .NET 网页开发有单文件页模型和代码隐藏页模型两种,以下叙述错误的是(　　)。

A. 单文件页模型中,C#代码必须包含于<script>…</script>之间

B. 代码隐藏页模型中,可视元素和编程逻辑代码放在一个文件中

C. 代码隐藏页模型中,可视元素和编程逻辑代码放在不同文件中

D. 单文件页模型中,可视元素和编程逻辑代码放在不同文件中

(9) 采用 Visual Studio 设计的网页代码称为 ASP .NET 网页代码,最终浏览器运行的网页代码称为纯 HTML 代码。以下叙述正确的是(　　)。

A. 在用户请求 ASP .NET 网页时,需要 ASP .NET 引擎将 ASP .NET 网页代码转换为纯 HTML 代码

B. 不需要 ASP .NET 引擎,Web 服务器就会将 ASP .NET 网页代码转换为纯 HTML 代码

C. ASP .NET 网页代码和纯 HTML 代码是一样的

D. 以上都不对

(10) 用户通过 Internet 请求 ASP .NET 网页,对应的 Web 服务器(　　)。

A. 只需要配置 IIS 即可

B. 只需要配置 ASP .NET 引擎即可

C. 必须配置 IIS 和 ASP .NET 引擎

D. 以上都不对

(11) ASP .NET 的@ Page 指令的作用是(　　)。

A. 定义 ASP .NET 页分析器和编译器使用的页的特定属性

B. 将命名空间显式导入到网页中

C. 以声明的方式控制 ASP .NET 页或页中包含的用户控件的输出缓存策略

D. 指示当前或用户实现指定的 .NET Framework 接口

(12) ASP .NET 的@ Import 指令的作用是(　　)。

A. 定义 ASP .NET 页分析器和编译器使用的页的特定属性

B. 将命名空间显式导入到网页中

C. 以声明的方式控制 ASP .NET 页或页中包含用户控件的输出缓存策略

D. 指示当前或用户实现指定的 .NET Framework 接口

(13) 下列选项中,只有(　　)不是@ Page 指令的属性。

A. CodeBehind　　B. Buffer　　C. NameSpace　　D. Language

(14) ASP .NET 的@ Page 指令中 CodeFile 属性的含义是(　　)。

A. 指定包含与页面关联的类的已编译文件的名称

B. 指定指向页面引用的代码隐藏文件的路径

C. 指示用于响应的编码方案的值,该值是一个用做编码方案 ID 的整数

D. 定义供页面继承的代码隐藏类

(15) ASP .NET 的@ Page 指令中 Inherits 属性的含义是(　　)。

A. 指定包含与页面关联的类的已编译文件的名称

B. 指定指向页面引用的代码隐藏文件的路径

C. 指示用于响应的编码方案的值,该值是一个用做编码方案 ID 的整数

D. 定义供页面继承的代码隐藏类

(16) web.config 文件不能用于(　　)。

A. Application 事件定义　　B. 数据库连接字符串定义

C. 对文件夹访问授权　　D. 基于角色的安全性控制

2. 问答题

(1) 简述创建一个 ASP .NET 空网站的步骤。

(2) 简述开发 ASP .NET 网页的单文件页模型和代码隐藏页模型的区别。

(3) 网页设计窗口下方有 设计 、源 和 拆分 3 个选项卡,说明它们的作用。

(4) 在一个 ASP .NET 网页中有如下源视图代码:

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

给出对应的纯 HTML 代码。

(5) 简述页面指令@ Page 的作用。

(6) 简述 web.config 文件中<connectionStrings>节的作用。