1.1 ASP.NET 概述

ASP.NET 基于.NET Framework，使用.NET 语言调用.NET Framework 类库，实现 Web 应用程序开发。

1.1.1 静态页面和动态页面

- 静态页面只包含 HTML 元素和 CSS 样式，一般以扩展名.htm 或.html 存储。
- 静态页面显示的都是相同的内容。
- 解释执行静态页面完全由浏览器完成。
- 动态页面可以包含 HTML 元素和 CSS 样式，还可以包含 JavaScript 代码和需要在 Web 服务器端编译执行的代码。
- 开发技术：ASP.NET、ASP、JSP、PHP 等。
- 动态页面的内容存储于数据库。
- 所有动态页面都需要 Web 服务器转换成静态页面后，才能在用户浏览器中显示最终效果。
- 在同一个 ASP.NET 网站中，可同时存在静态页面和动态页面。
- 当页面内容可以直接通过页面设计而不需要通过改变数据库中数据进行更新时，常使用静态页面，反之，则使用动态页面。
- 静态页面的访问速度要快于动态页面。

1.1.2 .NET Framework

- .NET Framework 是一套 Microsoft 应用程序开发的框架，主要目的是要提供一个一致的开发模型。
- .NET Framework 具有两个主要组件：公共语言运行库（Common Language Runtime，CLR）和.NET Framework 类库。

1.1.3 ASP.NET 特性

- ASP.NET 不是一种编程语言，而是.NET Framework 提供的一个组件。
- 与.NET Framework 完美整合
- ASP.NET 属于编译型而非解释型

1.1.4 ASP.NET 的开发模式

- ASP.NET Web 窗体
- 普遍使用的开发模式。
- 包含 XHTML、ASP.NET 控件等用于页面呈现的标记，以及采用.NET 语言（如 C#）处理页面和控件事件的代码。

1.2 IIS

- IIS（Internet 信息服务）提供 Web 服务器功能。
- IIS 的版本与不同的操作系统有关，如 Windows 7 旗舰版对应 IIS 10。
- 注意：在 VS 2019 中进行网站设计与开发时，可以仅使用 IIS Express 运行网站，不需要额外安装操作系统中的 IIS。

1.2.2 IIS 10 中的网站、Web 应用程序和虚拟目录

- 网站是 Web 应用程序的容器。
- Web 应用程序是一种在应用程序池中运行并通过 HTTP 协议向用户提供 Web 内容的程序。
- 应用程序池用于工作进程的运行配置，并保证各工作进程的独立运行。
- 虚拟目录是映射到本地或远程 Web 服务器上的物理文件夹的别名。
- 网站、Web 应用程序和虚拟目录在组织结构上呈现出一种层次关系。
- 通过“Internet 信息服务(IIS)管理器”配置。
- 组织结构关系存储在%windir%\System32\inetsrv\config\applicationHost.config 文件的<sites>元素中。
- 注意：IIS 10 中的网站与 VS 2019 中的网站不是同一个概念。实际上，IIS 10 中的 Web 应用程序与 VS 2019 中的网站相对应。

1.3.3 发布 Web 应用

- 可以部署到 Microsoft Azure 或 IIS。
- Web 应用程序发布还包括“Web 部署”、“Web Deploy 包”、FTP、“文件系统”等方式。
- 预编译：将网站中 App_Code 文件夹下包含的.cs 文件、代码隐藏页等编译为系统随机命名的.dll 程序集文件。
- 动态编译：如果一个页面第一次被访问或被修改保存后再被访问时，.NET 环境会自动调用编译器进行编译，并缓存编译输出。
- 注意：必须以管理员身份启动 VSC 2019 才能通过“发布 Web 应用”命令正确地发布网站。

1.3.4 复制网站

- “复制网站”实质是在当前网站与另一网站之间复制文件，对当前网站不会预编译。
- 支持同步功能。
- 常用于将网站从“测试服务器”复制到“商业服务器”。
- 注意：为保护 C#源代码不被随意窃取，可组合使用“发布网站”和“复制网站”。即先将网站发布到本地某个文件夹，再利用“复制网站”同步服务器网站上的文件。

1.4 小结

- ASP.NET 网站通常包含静态页面和动态页面。
- .NET Framework 为建立 ASP.NET 网站提供了基础。
- ASP.NET 的开发模式包括 ASP.NET Web 窗体、ASP.NET MVC、ASP.NET Core 等。
- IIS 为 ASP.NET 提供了运行环境，通过建立不同的网站或应用程序使得在同一台 Web 服务器上运行不同的站点成为可能。
- 利用 VSC 2019，可以方便地实现 ASP.NET 网站开发。

2.1 .html 文件和 XHTML5

- .html 文件是一种静态页面文件，由 HTML 元素组成。当客户端浏览器访问.html 文件时，由浏览器解释执行。
- 在 VSC 2019 中默认使用 XHTML5 文件类型。
- XHTML5 与 HTML5 使用相同的元素，但 XHTML5 具有更严格的规则。
- 所有包含 ASP.NET 元素的动态页面最终都要转化为包含相应 XHTML 元素的静态页面才能被浏览器识别。

2.1.1 .html 文件结构

```
<!DOCTYPE html>
<html
xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>…</title>
</head>
<body>…</body>
</html>
```

2.1.2 常用 XHTML5 元素

- <!--…-->表示注释。
- <!DOCTYPE html>表示文档类型为 HTML5。
- <html>…</html>表示这是一个 HTML 文档。
- <head>…</head>表示文档头部信息。
- <meta>表示文档的元信息。
- <title>…</title>表示浏览器标题栏中显示的信息，应包含于<head>…</head>中。
- <style>…</style>表示 CSS 样式信息，应包含于<head>…</head>中。
- <body>…</body>表示文档主体部分。
- <header>…</header>表示整个显示页面的标题信息。
- <aside>…</aside>表示与旁边内容相关的标题信息。
- <section>…</section>表示显示页面的内容区域。
- <article>…</article>表示显示页面中与上下文不相关的独立内容。

2.2 .aspx 文件

- .aspx 文件（Web 窗体）在 ASP.NET 网站中占据主体部分。
- 直接或间接地继承自 System.Web.UI.Page 类。
- 每个 Web 窗体中的代码包括两部分：一部分是处于<body>元素之间的用于界面显示的代码；另一部分是包含事件处理等的 C#代码。

2.2.1 单文件页模型

- 界面显示代码和逻辑处理代码（事件、方法等）都放在同一个.aspx 文件中。
- 逻辑处理代码包含于<script>元素中。
- <script>元素位于<html>元素之上，且包含 runat="server"属性。

2.2.2 代码隐藏页模型

- 代码隐藏页模型适用于多个开发人员共同创建网站的情形。
- 显示界面的代码包含于.aspx 文件，而逻辑处理代码包含于对应的.aspx.cs 文件。
- .aspx 文件不包含<script>元素，但在@Page 指令中需包含引用的外部文件。

2.3 .css 文件和 CSS 常识

- 级联样式表 CSS 是应用于页面中元素的样式规则，现已被各类浏览器所接受。
- CSS 提供了精确定位和重新定义 XHTML 元素属性的功能。
- 一个 CSS 样式文件可以作用于多个 XHTML 文件。
- CSS 的版本有 CSS1、CSS2、CSS3。

2.4 .js 文件和 JavaScript 常识

- JavaScript 是一种面向对象和事件驱动的客户端脚本语言，可以直接嵌入到页面中，不需要 Web 服务器端的解释执行即可由浏览器解释执行。
- 所有的浏览器均支持 JavaScript。
- 用途：在 XHTML 中创建动态文本；响应客户端事件；读取并改变 XHTML 元素的内容；验证客户端数据；检测客户端浏览器，并根据检测到的浏览器类型载入不同的页面；创建 Cookies；关闭浏览器窗口；在页面上显示时间等。

2.4.1 JavaScript 代码位置

位置形式通常有三种：在<head>元素中、在<body>元素中和独立的.js 文件中。

2.5 jQuery

- jQuery 由 John Resig 于 2006 年初创建。
- 一个优秀的 JavaScript 框架，提供 JavaScript 库。
- 访问和管理（包括插入、修改、删除等操作）XHTML 元素，设置 XHTML 元素的 CSS 样式，处理 XHTML 元素的事件，实现 XHTML 元素的动画特效，为网站提供 Ajax 交互。
- 支持 XHTML5 和 CSS3，提供的 jQuery Mobile 可以方便地用于智能手机和平板电脑的 Web 应用程序开发。
- 绝大多数浏览器均支持 jQuery。

2.6 .xml 文件和 XML 常识

- .xml 文件常用于解决跨平台交换数据的问题。
- XML 是一种可以扩展的标记语言，可以根据实际需要，定义相应的语义标记。
- 与 XHTML 相比，XHTML 用来显示数据，而 XML 旨在传输和存储数据。

2.7 Web.config

- XML 格式文件，用来存储配置信息。
- 形成一定的层次关系。
- 最高层是位于网站根文件夹中的 Web.config。
- 下一层是子文件夹中的 Web.config。
- 网站根文件夹下的 Web.config 作用于整个网站，而子文件夹中的 Web.config 常用于存储该子文件夹的授权信息。

Web.config 文件的基本结构

- 源程序：Web.config
- <connectionStrings>——用于数据库连接字符串的配置。
- <authentication>——用于身份验证的配置。
- <pages>——用于页面的特定配置。例如，当需要在页面中使用包含于 AjaxControlToolkit 程序包中的控件时，可通过子元素<controls>进行设置。
- <sessionState>——用于会话状态的配置。

2.8 Global.asax

- 全局应用程序类文件。
- 可选文件，用于包含响应应用程序级别和会话级别事件的代码。
- 必须存储于网站的根文件夹，且每个网站只能包含一个 Global.asax 文件。
- 包含在 Global.asax 文件中的代码将首先被执行。

2.10 小结

- 任何一个.aspx 文件的内容都要转化为 XHTML 才能在浏览器中查看。
- 软件公司在开发 Web 应用程序时大都采用代码隐藏页模型。
- CCS 样式能使网站保持统一风格。
- JavaScript 为静态页面提供动态功能。
- jQuery 能非常方便地控制和管理 XHTML 元素。
- XML 已成为 Internet 数据交换的标准格式。
- Web.config 用于存储 Web 应用程序的配置信息。
- Global.asax 文件用于包含响应应用程序级别和会话级别事件的代码。
- Bootstrap 以移动设备优先为设计理念，完全体现响应式设计思想，是目前用于 Web 应用程序前端设计与开发的主流框架。

4.1.1 ASP.NET 页面事件

事件	作用
Page.PreInit	通过IsPostBack属性确定是否第一次处理该页、创建动态控件、动态设置主题属性、读取配置文件属性等
Page.Init	初始化控件属性
Page.Load	读取和更新控件属性
控件事件	处理特定事件，如Button控件的Click事件

- 事件处理的先后顺序：Page.PreInit、Page.Init、Page.Load 和控件事件。
- 控件事件以 Click 和 Changed 事件为主。
- Click 事件被触发时会引起页面往返处理，即页面将被重新执行并触发 Page.Load 等事件。
- Changed 事件被触发时，先将事件的信息暂时保存在客户端的缓冲区中，等到下一次向服务器传递信息时，再和其他信息一起发送给服务器。
- 若要让控件的 Changed 事件立即得到服务器的响应，就需要将该控件的 AutoPostBack 属性值设为 True。
- 注意：当通过“属性”窗口设置值为逻辑值的控件属性时，值默认采用 Pascal 形式。实际上，在.aspx 文件中的逻辑值不区分大小写，但在.aspx.cs 文件中的逻辑值必须全部用小写字母表示。

4.1.2 IsPostBack 属性

- 如果想在执行控件的事件处理代码时不执行 Page_Load() 方法代码，可以通过判断 Page.IsPostBack 属性值实现。
- 注意：当 .aspx 文件中 @ Page 指令的 AutoEventWireup 属性值为 true 时，ASP.NET 能自动将页面事件绑定到名为“Page_事件名”的方法。而要把控件事件绑定到对应的方法，需要设置名为“On事件名”的属性。

4.2 ASP.NET 服务器控件概述

两种服务器控件：HTML 服务器控件和 Web 服务器控件。

4.2.2 Web 服务器控件简介

- 标准控件：除 Web 窗体中常用的按钮、文本框、下拉列表框等控件外，还包括一些特殊用途的控件，如日历等。
- 数据控件：用于连接访问数据库，显示数据库数据等。
- 验证控件：用于验证用户输入的信息，如输入的值要在指定的范围等。
- 导航控件：用于网站的导航。
- AJAX 扩展控件：用于只更新页面的局部信息而不往返整个页面。
- 动态数据控件：用于创建动态数据页面。
- 用户自定义控件：用于扩展系统功能，如保持网站的统一风格等。

4.3 常用 ASP.NET 标准控件

属性名	说明	属性名	说明
AccessKey	控件的键盘快捷键	Font	控件的字体属性
Attributes	控件的所有属性集合	Height	控件的高度
BackColor	控件的背景色	ID	控件的编程标识符
BoderWidth	控件的边框宽度	Text	控件上显示的文本
BoderStyle	控件的边框样式	ToolTip	当鼠标悬停在控件上时显示的文本
CssClass	控件的CSS类名	Visible	控件是否在页面上显示
Enabled	是否启用Web服务器控件	Width	控件的宽度

4.3.1 Label 控件

- Label 控件用于在页面上显示文本，通过 Text 属性指定控件显示的内容实现在服务器端动态地修改显示文本的作用。
- 一个很实用的属性是 AssociatedControlID，设置其值可把 Label 控件与窗体中另一个服务器控件关联起来。

4.3.2 TextBox 控件

属性、方法和事件	说明
TextMode属性	设置文本框类型。例如，值Password表示密码框，将显示特殊字符，如*；值MultiLine表示多行文本框
AutoPostBack属性	值true表示当文本框内容改变且把焦点移出文本框时触发TextChanged事件，引起页面往返处理
AutoCompleteType 属性	标注能自动完成的类型，如值Email表示能自动完成邮件列表
Focus()方法	设置文本框焦点
TextChanged事件	当改变文本框中内容且焦点离开文本框后被触发

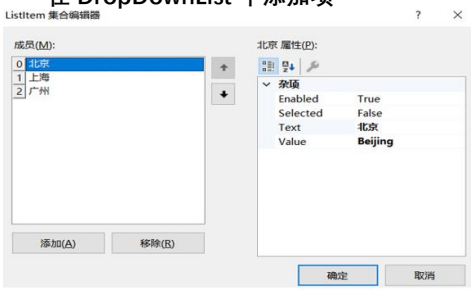
4.3.3 Button、LinkButton 和 ImageButton 控件

- Button 呈现传统按钮外观。
- LinkButton 呈现超链接外观。
- ImageButton 呈现图形外观，其图像由 ImageUrl 属性设置。
- PostBackUrl 属性：设置跨页面提交时的目标页面路径。
- Click 事件：当单击按钮时被触发，执行服务器端代码。
- ClientClick 事件：当单击按钮时在 Click 事件之前被触发，执行客户端代码。
- <a>元素中通过 href 属性设置超链接形式，
- 如：链接到阿里云
- LinkButton 控件中需要设置 PostBackUrl 属性实现，或者在 Click 事件中输入代码，通过 Response 对象的重定向方法 Redirect() 实现超链接，如：Response.Redirect("http://www.aliyun.com");

4.3.4 DropDownList 控件

- DataSource 属性：设置数据源。
- DataTextField 属性：对应数据源中的一个字段，该字段所有内容将被显示于下拉列表中。
- DataValueField 属性：对应数据源中的一个字段，指定下拉列表中每个可选项的值。
- Items 属性：列表中所有选项的集合，常用 Add()方法添加项，Clear()方法删除所有项。
- SelectedItem 属性：当前选定项。
- SelectedValue 属性：当前选定项的 Value 属性值。
- SelectedIndexChanged 事件：当选择下拉列表中一项后被触发。
- DataBind()方法：绑定数据源。

----在 DropDownList 中添加项



- 利用 DropDownList 对象的 Items.Add()方法添加项，
- 如：ddlCity.Items.Add(new ListItem("北京", "beijing"));
- 通过 DataSource 属性设置数据源，再通过 DataBind()方法显示数据。

4.3.5 ListBox 控件

DropDownList 和 ListBox 控件都允许用户从列表中选择数据项，区别在于 DropDownList 的列表在用户选择数据项前处于隐藏状态，而 ListBox 的列表是可见的，并且可同时选择多项。SelectionMode 属性：其值为 Multiple 表示允许选择多项。

4.3.6 CheckBox 和 CheckBoxList 控件

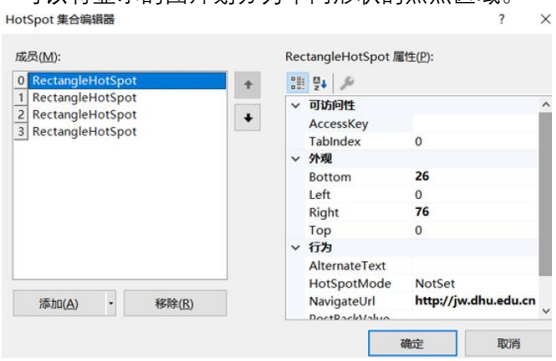
- 提供“真/假”、“是/否”或“开/关”选项之间进行选择的方法。
- 注意：判断 CheckBox 控件是否选中的属性是 Checked，而 CheckBoxList 控件作为集合控件，判断数据项是否选中的属性是成员的 Selected 属性。
- 在实际工程项目中，一般设置 CheckBoxList 控件的 AutoPostBack 属性值为 False。要提交数据到服务器，不采用 CheckBoxList 控件的自身事件，而是常配合 Button 控件实现。

4.3.7 RadioButton 和 RadioButtonList 控件

- 用于在多种选择中只能选择一项的场合。
- 单个的 RadioButton 只能提供单项选择，可以将多个 RadioButton 形成一组，方法是设置每个 RadioButton 的 GroupName 属性为同一名称。
- 注意：判断 RadioButton 控件是否选中使用 Checked 属性，而获取 RadioButtonList 控件的选中项使用 SelectedItem 属性。

4.3.8 Image 和 ImageMap 控件

- Image 控件用于在 Web 窗体上显示图片，可以使用 ImageUrl 属性在界面设计或编程时指定图片源文件。
- 在实际工程项目中常与数据源绑定，根据数据源中指定的字段显示图片。
- 注意：Image 控件不包含 Click 事件，如果需要 Click 事件处理流程，可使用 ImageButton 控件代替 Image 控件。
- 可以将显示的图片划分为不同形状的热点区域。



4.3.9 HyperLink 控件

- HyperLink 控件用于在页面上创建链接。
- 可以与数据源绑定。
- Target 属性：值_blank 决定了在一个新窗口中显示链接页，而值_self 决定了在原窗口中显示链接页。
- 注意：HyperLink 控件不包含 Click 事件，要使用 Click 事件可用 LinkButton 控件代替。
- 在同时设置 Text 和 ImageUrl 属性的情况下，ImageUrl 优先。

4.3.10 Table 控件

- Table 控件用于在 Web 窗体上动态地创建表格，是一种容器控件，而单击“表”→“插入表”命令产生的表格常用于页面布局且对应 XHTML 元素 < table > 。
- 由 Table 控件生成的 Table 对象由行（TableRow）对象组成，TableRow 对象由单元格（TableCell）对象组成。
- 注意：向 Table 对象添加行使用 Rows 属性；向 TableRow 对象添加单元格使用 Cells 属性；向 TableCell 对象添加控件使用 Controls 属性。

4.3.11 Panel 和 PlaceHolder 控件

- 都属于容器控件，常用于实现动态地建立控件和在同一个页面中根据不同情况显示不同内容。
- 使用 Panel 控件的好处是只需载入一个页面，即可呈现不同的内容。

4.4 小结

- 理解 ASP.NET 页面事件的处理流程需搞清常用事件 Page_PreInit、Page_Init、Page_Load 和控件事件的触发顺序。
- 常通过 IsPostBack 属性决定在页面往返时是否执行相应的代码。
- 可以通过添加“runat="Server""将 XHTML 元素转换为 HTML 服务器控件。
- Web 服务器标准控件是构建 Web 窗体的基础。

5.1 窗体验证概述

- 验证就是给所收集的数据制定一系列规则。验证不能保证输入数据的真实性，只能说是否满足了一些规则。
- 分为服务器端和客户端两种形式。服务器端验证是指将用户输入的信息全部发送到 Web 服务器进行验证；客户端验证是指利用 JavaScript 脚本，在数据发送到服务器之前进行验证。

5.2 ASP.NET 服务器验证控件

- ControlToValidate：指定要验证控件的 ID。
- Display：指定验证控件在页面上显示的方式。值 Static 表示验证控件始终占用页面空间；值 Dynamic 表示只有显示验证的错误信息时才占用页面空间；值 None 表示验证的错误信息都在 ValidationSummary 控件中显示。
- EnableClientScript：设置是否启用客户端验证，默认值 True。
- ErrorMessage：设置在 ValidationSummary 控件中显示的错误信息，若 Text 属性值为空会代替它。
- SetFocusOnError：当验证无效时，确定是否将焦点定位在被验证控件上。
- Text：设置验证控件显示的信息。
- ValidationGroup：设置验证控件的分组名。
- 一个很实用的 CausesValidation 属性：值 False 表示不执行验证过程。
- 若要对一个控件设置多个规则，可通过多个验证控件共同作用，此时各验证控件的 ControlToValidate 属性应为相同值。
- 若要对同一个页面上不同的控件提供分组验证功能，可以通过将同一组控件的 ValidationGroup 属性设置为相同的组名来实现。

5.2.1 RequiredFieldValidator 控件

- 用于对一些必须输入信息的控件进行验证，如用户名、密码等。
- 在页面上填写表单时，常常可看到有些文本框后跟着一个*，就是使用该验证控件产生的效果。
- 非常实用的 InitialValue 属性：用于指定被验证控件的初始文本。若设置了 InitialValue 属性值，则只有在被验证控件中输入值并与 InitialValue 值不同时，验证才通过。

5.2.2 CompareValidator 控件

- 用于比较一个控件的值和另一个控件的值，若相等则验证通过；也可用于比较一个控件的值和一个指定的值，若比较的结果为 true 则验证通过。
- ControlToCompare：指定与被验证控件比较的控件 ID。
- Operator：设置比较值时使用的操作符，包括 Equal、NotEqual、GreaterThan、GreaterThanEqual、LessThan、LessThanEqual 和 DataTypeCheck。
- Type：设置比较值时使用的数据类型。
- ValueToCompare：指定与被验证控件比较的值。
- 注意：ControlToCompare 和 ValueToCompare 属性在应用时只能选择一个。

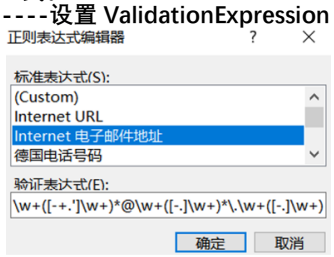
5.2.3 RangeValidator 控件

- 用来验证输入值是否在指定范围内。
- 提供了 MaximumValue 和 MinimumValue 属性，分别对应验证范围的最大值和最小值。

5.2.4 RegularExpressionValidator 控件

- 用来验证输入值是否和定义的正则表达式相匹配，常用来验证电话号码、邮政编码、Email 等。
- ValidationExpression 属性：用来确定验证所需的正则表达式。

----设置 ValidationExpression 属性



5.2.5 CustomValidator 控件

- 当 ASP.NET 提供的验证控件无法满足实际需要时，可以考虑先自定义验证函数，再通过 CustomValidator 控件调用它来满足需求。
- 若要使用客户端验证，则需要设置 ClientValidationFunction 属性值为客户端验证函数名，并且要设置 EnableClientScript 属性的值为 True；若使用服务器端的验证，则通过 ServerValidate 事件触发，此时，需要将完成验证功能的代码包含在事件处理代码中。
- IsValid 属性：用来确定是否通过验证。

5.2.6 ValidationSummary 控件

- 用于汇总其他验证控件错误信息的方式，即汇总其他验证控件的 ErrorMessage 属性值。
- DisplayMode 属性：指定了显示信息的格式，值分别为 BulletList、List 和 SingleParagraph。
- ShowMessageBox 属性：指定是否在一个弹出的消息框中显示错误信息。
- ShowSummary 属性：指定是否启用错误信息汇总。

5.3 小结

- 在窗体验证时常需同时使用客户端和服务器端验证。
- ASP.NET 提供了“必需输入”验证、比较验证、范围验证、正则表达式验证、自定义验证和汇总其它验证控件错误的功能。
- 为达到一定的验证效果，实际使用时对同一个控件可能使用多个验证控件。

6.1 HTTP 请求

- ASP.NET 通过 Page 类的 Request 属性能很好地控制请求数据，如访问客户端的浏览器信息、查询字符串、Cookie 等信息。
- 实际上，Page 类的 Request 属性值是 HttpRequest 类的一个实例对象，它封装了 HTTP 请求信息。

----HttpRequest 对象的数据集合对应表

数据集合	说明
QueryString	从查询字符串中读取用户提交的数据
Cookies	获得客户端的Cookie数据
ServerVariables	获得服务器端或客户端的环境变量信息
Browser	获得客户端浏览器信息

6.1 HTTP 请求（续）

- 在使用 HttpRequest 对象时，常通过 Page 类的 Request 属性调用，所以要获取 HttpRequest 对象的 Browser 数据集合的语法格式常写为：Request.Browser。
- 利用 QueryString 数据集合获得的查询字符串是指跟在 URL 后面的变量及值，它们以“?”与 URL 间隔，不同的变量之间以“&”间隔。
- 利用 ServerVariables 数据集合可以很方便地获取服务器端或客户端的环境变量信息，如客户端的 IP 地址等。
- 语法格式为：Request.ServerVariables["环境变量名"]。

-----常用的环境变量表

环境变量名	说明
LOCAL_ADDR	服务器端的IP地址
PATH_TRANSLATED	当前页面在服务器端的物理路径
REMOTE_ADDR	客户端IP地址
REMOTE_HOST	客户端计算机名
SERVER_NAME	服务器端计算机名
SERVER_PORT	服务器端网站的端口号

6.1 HTTP 请求（续）

- Browser 数据集合用于返回用户的浏览器类型、版本等信息，以便根据不同的浏览器编写不同的页面。
- 语法格式为：Request.Browser["浏览器特性名"]。

-------浏览器特性名对应表

名称	说明
Browser	浏览器类型
Version	浏览器版本号
MajorVersion	浏览器主版本号
MinorVersion	浏览器次版本号
Cookies	逻辑值，true表示支持Cookie
JavaScript	逻辑值，true表示支持JavaScript
ActiveXControls	逻辑值，true表示支持ActiveX控件

6.2 HTTP 响应

- ASP.NET 通过 Page 类的 Response 属性可以很好地控制输出的内容和方式，如页面重定向、保存 Cookie 等。
- 实际上，Page 类的 Response 属性值是 HttpResponse 类的一个实例对象。

-------HttpResponse 对象的常用属性和方法表

成员	说明
Cookies属性	添加或修改客户端的Cookie
AppendToLog()方法	将自定义日志信息添加到IIS日志文件中
End()方法	终止页面的执行
Redirect()方法	页面重定向，可通过URL附加查询字符串实现不同页面之间的数据传递
Write()方法	在页面上输出信息

6.3 HttpServerUtility

- 在 ASP.NET 中，Page 类的 Server 属性封装了服务器端的一些操作，如将 XHTML 元素标记转换为字符实体、获取页面的物理路径等。
- 实际上，Page 类的 Server 属性值是 HttpServerUtility 类的一个实例对象。

------HttpServerUtility 对象的常用属性和方法

- ScriptTimeOut 属性：设置页面执行的最长时间，单位为秒。
- Execute()方法：停止执行当前页面，转到并且执行新页面，执行完毕后返回原页面，继续执行后续语句。
- HtmlEncode()方法：将字符串中的 XHTML 元素标记转换为字符实体，如将“<”转换为&lt。
- MapPath()方法：获取页面的物理路径。
- Transfer()方法：停止执行当前页面，转到并且执行新页面，执行完毕后不再返回原页面。
- UrlEncode()方法：将字符串中某些特殊字符转换为 URL 编码，如将“/”转换为“%2f”，空格转换为“+”。

-----Response.Redirect()、Server.Execute()和 Server.Transfer()的区别

- Redirect()方法尽管在服务器端执行，但重定向实际发生在客户端，可从浏览器地址栏中看到地址变化；而 Execute()和 Transfer()方法的重定向实际发生在服务器端，在浏览器的地址栏中看不到地址变化。
- Redirect()和 Transfer()方法执行完新页面后，并不返回原页面；而 Execute()方法执行完新页面后会返回原页面继续执行。
- Redirect()方法可重定向到同一网站的不同页面，也可重定向到其它网站的页面；而 Execute()和 Transfer()方法只能重定向到同一网站的不同页面。
- 利用 Redirect()方法在不同页面之间传递数据时，状态管理采用查询字符串形式；而 Execute()和 Transfer()方法的状态管理方式与 Button 类型控件的跨页面提交方式相同。

6.4 跨页面提交

- 利用 Button 类型控件实现跨页面提交是另一种实现页面重定向的方法。
- 需要将源页面上 Button 类型控件的 PostBackUrl 属性值设置为目标页面路径。而在目标页面上，需要在页面头部添加@ PreviousPageType 指令，并设置 VirtualPath 属性值为源页面路径。
- 在目标页面上访问源页面中数据的方法有两种：一是利用 PreviousPage.FindControl()方法访问源页面上的控件；二是先在源页面上定义公共属性，再在目标页面上利用“PreviousPage.属性名”获取源页面中数据。

------跨页面提交与调用 Server.Execute()或 Server.Transfer()方法的区别

- 如果是跨页面提交，那么 IsCrossPagePostBack 属性值为 true；如果是调用 Server.Execute()或 Server.Tranfer()方法，那么 IsCrossPagePostBack 属性值为 false。

6.5 状态管理

- 客户端状态管理：将状态数据保存在客户端计算机上，当客户端向服务器端发送请求时，状态数据会随之发送到服务器端。可选择 ViewState、ControlState、HiddenField、Cookie 和查询字符串。
- 服务器状态管理：将状态数据保存在服务器上。可选择 Session 状态、Application 状态或数据库形式。
- 客户端状态不消耗服务器内存资源，但容易泄露数据信息。而服务器端状态将消耗服务器端内存资源，但具有较高的安全性。

6.5.1 ViewState
- 又称为视图状态，用于维护 Web 窗体自身的状态。当用户请求 ASP.NET 页面时，ASP.NET 将 ViewState 封装为一个或几个隐藏的表单域传递到客户端。当用户再次提交页面时，ViewState 也将被提交到服务器端。这样后续的请求就可以获得上一次请求时的状态。
- 可在客户端浏览 ASP.NET 页面时，选择“查看”→“源文件”命令进行查看。
- EnableViewState 属性：值为 False 可实现禁用 ViewState 的目的。

6.5.2 HiddenField 控件
- 又称隐藏域，用于维护 Web 窗体自身的状态。
- 不会显示在用户的浏览器中，但可以像设置标准控件的属性那样设置其属性。
- 成员主要有 Value 属性和 ValueChanged 事件。
- 注意：要触发 ValueChanged 事件，需设置 HiddenField 控件的 EnableViewState 属性值为 False。

6.5.3 Cookie
- 保存在客户端硬盘或内存中的一小段文本信息，如网站、用户、会话等有关的信息。
- 与网站关联，而不是与特定的页面关联。
- 可以在客户端修改 Cookie 设置和禁用 Cookie。当用户的浏览器关闭了对 Cookie 的支持，但又要使用 Cookie 时，只需在 Web.config 文件的<system.web>元素中加入以下语句：
- <sessionState cookieless="AutoDetect">或
- <sessionState cookieless="UseUri">
- 在 Windows 7 操作系统中，Cookie 文本文件存储于“%userprofile%\AppData\Roaming\Microsoft\ Windows\Cookies”文件夹。
- 提供 System.Web.HttpCookie 类来处理 Cookie，常用的属性是 Value 和 Expires。

-------建立 Cookie

方法一：
```
Response.Cookies["Name"].Value="张三";
```
方法二：
```
HttpCookie cookie = new HttpCookie("Name");
cookie.Value = "张三";
cookie.Expires = DateTime.Now.AddDays(1);
Response.Cookies.Add(cookie);
```
--------获取 Cookie
```
Request.Cookies.["Name"].Value
```

6.5.4 Session
- 又称会话状态，常用于存储用户信息、多页面间的信息传递、购物车等。
- 产生在服务器端，只能为当前访问的用户服务。
- 以用户对网站的最后一次访问开始计时，当计时达到会话设定时间并且期间没有访问操作时，则会话自动结束。
- 如果同一个用户在浏览期间关闭浏览器后再访问同一个页面，服务器会为该用户产生新的 Session。
- 用一个唯一的 Session ID 来标识每一个会话。
- 若客户端支持 Cookie，则将 Session ID 保存到相应的 Cookie 中；若不支持，就将 Session ID 添加到 URL 中。
- 注意：不管 Session ID 保存在 Cookie 还是添加在 URL 中，都是明文。如果需要保护 Session ID，可考虑采用 HTTPS 通信。
- Session 由 System.Web.HttpSessionState 类实现。

-------HttpSessionState 常用的属性和方法表

属性和方法	说明
Contents属性	获取对当前会话状态对象的引用
Mode属性	获取当前会话状态的模式
SessionID属性	获取会话的唯一标识
TimeOut属性	获取或设置会话状态持续时间，单位为分钟，默认为20分钟
Abandon()方法	取消当前会话
Remove()方法	删除会话状态集合中的项

------Session_Start()和 Session_End()方法
- 包含于 Global.asax 文件中。
- Session_Start()方法中代码在新会话启动时会自动被执行。
- Session_End()方法中代码在会话结束时会自动被执行。
- 注意：只有 Web.config 文件中的 sessionState 模式设置为 InProc 时，才会执行 Session_End()方法代码。如果会话模式设置为 StateServer 或 SQLServer，则不会执行 Session_End()方法代码。

-------Session 的赋值

方法一：Session["Name"]="张三";
方法二：Session.Contents["Name"]="张三";
注意：Session 使用的名称不区分大小写，因此不要用大小写区分不同的 Session 变量。

--------Session 的存储方式
- 在 Web.config 中通过<sessionState>元素的 mode 属性来指定。
- 共有 Off、InProc、StateServer、SQLServer 和 Custom 五个枚举值供选择，分别代表禁用、进程内、独立的状态服务、SQLServer 和自定义数据存储。
- 在实际工程项目中，一般选择 StateServer，而对于大型网站常选用 SQLServer。

6.5.5 Application
- 又称应用程序状态，应用于所有的用户。
- Application 状态存在于网站运行过程中，当网站关闭时将被释放。
- 由 System.Web.HttpApplicationState 类来实现。
- 要修改 Application 状态值时，首先要调用 Application.Lock()方法锁定 Application 状态，值修改后再调用 Application.Unlock()方法解除锁定。
- Application_Start()、Application_End()、Application_Error()方法包含于 Global 类。

6.6 小结
- HttpRequest 提供了 QueryString、ServerVariables、Browser、Cookies 等数据集合来访问不同用途的数据。
- HttpResponse 提供了输出 XHTML 文本、JavaScript 脚本、Cookie 等功能。
- 为了有效防范 SQL 脚本注入，常会使用 HttpServerUtility 对象的 HtmlEncode()方法，该对象同时提供了 UrlEncode()、MapPath()等实用方法。
- 页面重定向可采用 <a> 元素、HyperLink、Response.Redirect()、Server.Execute()、Server.Transfer()和 Button 类型控件的跨页面提交等形式，在使用时要注意它们的区别。
- 状态管理分为客户端和服务器端两种管理形式。客户端形式使用较多的是 Cookie 和查询字符串，服务器端形式包含 Session、Application 和数据库等。其中，Session 对应单个用户，而 Application 对应所有用户。

7.1 数据访问概述
- 在 ASP.NET 1.x 中，主要使用 ADO.NET 访问数据，这种技术在基于 VS 2019 的 ASP.NET 中仍被支持。
- ADO.NET 提供了用于完成如数据库连接、查询数据、插入数据、更新数据和删除数据等操作的对象。
- 在 ASP.NET 2.0 中，增加了多种数据源控件和数据绑定控件。
- 在 ASP.NET 3.5 中，引入了一种新技术 LINQ。在具体使用 LINQ 技术时，需要使用对象关系映射（Object Relational Mapping，ORM）技术将数据库映射为类对象。而 Entity Framework（EF）作为 Microsoft 提供的 ORM 工具。

7.2 建立 SQL Server Express 数据库
- SQL Server Express（SSE）：免费版，适用于学习用途及中小型企业的数据库开发应用。
- SQL Server Express LocalDB：比 SSE 更轻量级的版本，目的是在 Web 应用程序开发时无需配置即可使用数据库。
- LocalDB 主要适用于开发环境，当包含 LocalDB 数据库的 Web 应用程序发布到 IIS 10 时，需要更改用于访问数据库的连接字符串。

--------连接字符串
- 存储于 Web.config 文件的<connectionStrings>元素中，并且需要根据不同类型的数据库实例和不同的身份验证形式进行配置。
- 数据库实例的类型：SQL Server、SQLEXPRESS、和 LocalDB 实例。
- 不同的实例类型将决定连接字符串中的 Data Source 属性值。例如，若要访问 SQLEXPRESS 实例，则需将 Data Source 属性值设置为“.\SQLEXPRESS”；若要访问 LocalDB 实例，则需将 Data Source 属性值设置为“(LocalDB)\MSSQLLocalDB”。
- SQL Server 数据库的身份验证有 Windows 验证和 SQL Server 验证。
- Windows 验证使用 Windows 操作系统用户连接 SQL Server，常用于局域网络。
- SQL Server 验证使用 SQL Server 中注册的用户连接 SQL Server，常用于 Internet 环境。

------MyPetShop 数据库
- 包含购物车、商品分类、用户、订单、商品、供应商等数据表。
- 通过 MyPetShop.sql 建立 MyPetShop 数据库。
- 源程序：MyPetShop.sql
- 注意：在 VSC 2019 中建立的数据库默认属于 LocalDB 数据库实例并且排序规则为 SQL_Latin1_General_CP1_CI_AS，为更好地支持中文信息处理，在利用 CREATE DATABASE 语句建立数据库时可指定参数 COLLATE Chinese_PRC_CS_AS，其中，CS 表示区分大小写，AS 表示区分重音。

-------访问 MyPetShop 数据库的连接字符串
```
<add
name="MyPetShopEntities"
connectionString="metadata=res://*/App_Code.MyPetShop.csdl|res://*/App_Code.MyPetShop.ssdl|res://*/App_Code.MyPetShop.msl;
    provider=System.Data.SqlClient;
    provider connection string=&quot;data
      source=(LocalDB)\MSSQLLocalDB;
    attachdbfilename=|DataDirectory|\MyPetShop.mdf;
    integrated security=True;
    MultipleActiveResultSets=True;
    App=EntityFramework&quot;"
providerName="System.Data.EntityClient" />
```

7.3 使用数据源控件实现数据访问
- EntityDataSource：用于访问基于实体数据模型的数据。
- LinqDataSource：利用 LINQ 技术访问数据库。
- ObjectDataSource：用于访问多层 Web 应用程序体系结构中的中间层业务对象数据。
- SiteMapDataSource：用于访问 XML 格式的网站地图文件 Web.sitemap。
- SqlDataSource：用于访问 Access、SQL Server、SQL Server Express、Oracle、ODBC 数据源和 OLEDB 数据源。
- XmlDataSource：用于访问具有“层次化数据”特性的 XML 数据源。
- 提供了统一的基本编程模型。
- 通过数据源控件中定义的各种事件，可以实现 Select、Update、Delete 和 Insert 等数据操作。
- 提供了数据操作前后的事件，可以编写相关事件代码实现更加灵活的功能。

7.4 使用 LINQ 实现数据访问
- LINQ 集成于.NET Framework 中，提供了统一的语法实现多种数据源的查询和管理。
- LINQ to Objects：用于处理 Array 和 List 等集合类型数据。
- LINQ to XML：用于处理 XML 类型数据。
- LINQ to DataSet：用于处理 DataSet 类型数据。
- LINQ to SQL：用于处理 SQL Server 数据库类型数据。
- LINQ to Entities 用于处理实体数据模型。

7.4.1 LINQ-查询语法
- from 子句——指定查询操作的数据源和范围变量。
- select 子句——指定查询结果的类型和表现形式。
- where 子句——指定筛选操作的逻辑条件。
- group 子句——对查询结果进行分组。
- orderby 子句——对查询结果进行排序。
- join 子句——连接多个查询操作的数据源。

7.4.1 LINQ-查询语法（续）

- let 子句——创建用于存储查询表达式中的子表达式结果的范围变量。
- into 子句——提供一个临时标识符，该标识符可以在 join、group 或 select 子句中被引用。
- 查询表达式必须以 from 子句开始，以 select 或 group 子句结束，中间可以包含一个或多个 from、where、orderby、group、join、let 等子句。
- Select()方法—对应 select 子句。
- Where()方法—对应 where 子句。
- GroupBy()方法—对应 group…by 或 group…by…into 子句。
- OrderBy()方法—对应 orderby 子句。
- OrderByDescending()方法—对应 orderby…descending 子句。
- Join()方法—对应 join…in…on…equals…子句。

7.4.2 Entity Framework 概述

- Entity Framework 的核心是实体数据模型，能将关系数据库映射为类对象，再借助 LINQ 技术使开发人员以操作对象的方式实现对数据的查询、修改、插入和删除等操作。

7.4.3 基于 Entity Framework 利用 LINQ 查询数据

基于 Entity Framework 利用 LINQ 查询数据时，有查询语法和方法语法 2 种方式。

------投影

- 投影实现了属性的选择。
- 投影后的结果将新生成一个对象，该对象通常是匿名的。

------选择

选择实现了记录的过滤，由 where 子句完成。

------分组

- 使用 group…by 子句。
- 分组后的结果集合将采用集合的集合形式。
- 外集合中的每个元素包括键值及根据该键值分组的元素集合。因此，要访问分组后的结果集合中的元素，必须使用嵌套的循环语句。外循环用于循环访问外集合中的每个元素（即每个组），内循环用于循环访问内集合中的元素（即每个组中的元素）。
- 若要引用分组操作的结果，可以使用 into 子句创建用于进一步查询的标识符。

------聚合

- 主要涉及 Count()、Max()、Min()、Average()等方法。
- 参数常使用 Lambda 表达式。格式如：（输入参数）=> {语句块}
- 当把 Lambda 表达式应用于 Max()、Min()、Average()等聚合方法时，编译器会自动推断输入参数的数据类型。

------连接

- 使用 join 子句。
- 对于具有外键约束的多表，可以直接通过引用对象的形式进行查询，也可以使用 join 子句实现。

------模糊查询

使用时需调用 System.Data.Linq.SqlClient.SqlMethods.Like()方法。

7.4.4 利用 Entity Framework 管理数据——插入数据

插入数据利用 AddRange()和 Add()方法实现，前者用于插入集合数据实体，后者用于插入单个实体。

------修改数据

修改数据时需要根据某种信息找到需要修改的数据，如个人信息的修改需先通过身份验证，再根据身份标识获取个人信息实现数据的修改。

------删除数据

删除数据利用 RemoveRange()和 Remove()方法实现，前者用于删除实体集合，后者用于删除单个实体。

------存储过程

- 要使用原来 SQL Server 中定义的存储过程，需要在建立 MyPetShop.dbml 时将存储过程拖入到对象关系设计器的右窗口中，然后，VSC 2019 会自动建立与存储过程对应的方法。在具体使用存储过程时，只要调用对象的方法就可以了。

7.4.5 LINQ to XML 概述

- XDocument 类——用于操作 XML 文档。调用其 Save()方法可建立 XML 文档。
- XDeclaration 类——用于操作 XML 文档中的声明，包括版本、编码等。
- XComment 类——用于操作 XML 文档中的注释。
- XAttribute 类——用于操作 XML 元素的属性，是一个名称/值对。
- XElement 类——用于操作 XML 文档中可包含任意多级别子元素的元素。其中，Name 属性用于获取元素名称；Value 属性用于获取元素的值；Load()方法用于导入 XML 文档到内存，并创建 XElement 实例；Save()方法用于保存 XElement 实例到 XML 文档；Attribute()方法用于获取元素的属性；Remove()方法用于删除一个元素；ReplaceNodes()方法用于替换元素的内容；SetAttributeValue()方法用于设置元素的属性值。

7.4.6 利用 LINQ to XML 管理 XML 文档——创建 XML 文档

- 利用 XDocument 对象。
- 按照 XML 文档的格式，分别把 XML 文档的声明、元素、注释等内容添加到 XDocument 对象中。
- 用 Save()方法保存到 Web 服务器硬盘。

------查询 XML 文档

使用 LINQ 查询表达式可方便地读取 XML 文档、查询根元素、查询指定名称的元素、查询指定属性的元素、查询指定元素的子元素等。

------插入元素

- 建立一个 XElement 实例。
- 添加相应内容。
- 利用 Add()方法添加到上一级元素中。
- 利用 Save()方法保存到 XML 文档。

------修改元素

- 根据关键字查找到该元素。
- 利用 SetAttributeValue()方法设置属性值，ReplaceNodes()方法修改元素的内容。
- 利用 Save()方法保存到 XML 文档。

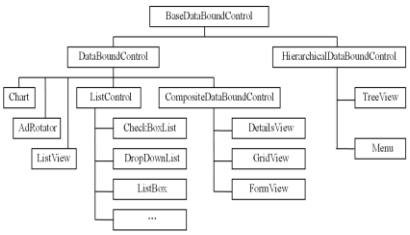
------删除元素

- 根据关键字查找到该元素。
- 利用 Remove()方法删除元素。
- 利用 Save()方法保存到 XML 文档。

7.5 小结

- Entity Framework 的核心是实体数据模型，支持 SQL Server、SQLite、MySQL、PostgreSQL 、Azure Cosmos DB 等数据库，代表 Microsoft 未来的数据访问技术。
- LINQ 技术完全与编程语言整合，将其中的数据作为对象处理，符合数据访问技术的发展。
- LINQ 有两种语法格式：查询语法和方法语法。
- 充分理解利用 Entity Framework 和 LINQ 技术，能满足任何数据访问的需求。

8.1 数据绑定概述



- 数据绑定控件若与数据源控件结合显示数据，则需设置 DataSourceID 属性值为数据源控件的 ID。
- 若与 LINQ 技术结合，则需设置 DataSource 属性值为 LINQ 查询结果值，并调用 DataBind()方法显示数据。

8.2 ListControl 类控件

- AppendDataBoundItems：用于将数据绑定项追加到静态声明的列表项上。
- DataTextField：绑定的字段用于显示列表项。
- DataValueField：绑定的字段用于设置列表项的值。

8.3 GridView 控件

- 用于显示二维表格式的数据.
- 可以方便地实现数据绑定、分页、排序、行选择、更新、删除等功能。

8.3.1 分页和排序

- 分页功能需要设置 AllowPaging 属性值为 True。
- 分页的效果可在 PagerSettings 属性集合中设置，例如，用于设置分页类型的 Mode 属性、用于设置“第一页”按钮图片 URL 的 FirstPageImageUrl 属性等。
- 排序功能需要设置 AllowSorting 属性值为 True。

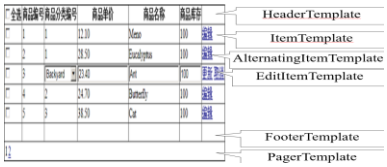
8.3.2 定制数据绑定列

- 需要设置 AutoGenerateColumns 属性值为 False。
- GridView 中的每一列都是一个 DataControlField 类，并从该类派生出不同类型的子类。

------GridView 中不同类型的数据绑定列

- BoundField：用于显示普通文本内容。
- CheckBoxField：用于显示布尔类型数据。
- CommandField：用于创建命令按钮列。
- DynamicField：用于绑定动态数据列。
- ImageField：用于显示图片列。
- HyperLinkField：用于显示超链接列。
- ButtonField：定义按钮列。
- TemplateField：以模板的形式自定义数据列。

8.3.3 使用模板列



-----TemplateField 中不同类型的模板

模板	说明
AlternatingItemTemplate	为交替项指定要显示的内容
EditItemTemplate	为处于编辑的项指定要显示的内容
FooterTemplate	为脚注项指定要显示的内容
HeaderTemplate	为标题项指定要显示的内容
ItemTemplate	为TemplateField列指定要显示的内容
PagerTemplate	为页码项指定要显示的内容

- AlternatingItemTemplate 需与 ItemTemplate 配合使用。
- 若未设置 AlternatingItemTemplate，则 GridView 的所有数据行都以 ItemTemplate 显示；
- 若已设置 AlternatingItemTemplate，则 GridView 中的奇数数据行以 ItemTemplate 显示，偶数数据行以 AlternatingItemTemplate 显示。

------数据绑定方法

- Eval()用于单向（只读）绑定.
- Bind()用于双向（可更新）绑定。
- 需要包含在<%#...%>中。
- 通过<%#...%>还可绑定变量、集合、表达式等。
- 例如，若 name 是在.aspx.cs 文件中定义的公共变量，则在.aspx 文件中使用<%# name %>并通过在.aspx.cs 文件中调用 Page.DataBind()方法后即能在浏览页面中显示 name 变量值。

8.3.4 利用 GridView 编辑、删除数据

- 单击 GridView 的智能标记，选择“启用编辑”和“启用删除”选项，可提供编辑和删除数据功能。当然，绑定至 GridView 的数据源控件也要提供更新、删除功能。

8.3.5 显示主从表

- 与数据库中的“一对多”联系对应。
- 可分为在同一页或不同页两种情况。

8.4 Repeater 控件

- Repeater 控件是一个数据显示控件，该控件允许通过为列表中显示的每一项重复使用指定的模板来自定义布局。
- 相对于 GridView 控件，该控件更加灵活方便，展示数据的形式更加丰富，同时对开发人员页面布局的要求更高。

8.5 小结

- ListControl 类：提供了以列表显示数据的形式。
- GridView：提供了以二维表格显示数据的形式。
- Repeater 提供了自定义格式显示数据的形式。
- ListView：能显示多条记录。
- DataList：完全使用模板的数据列表控件。
- FormView：显示单条记录且显示完全通过模板实现。

------信息管理系统设计

- 系统设计应遵循以下几个原则：可靠性，可维护性，用户友好性，系统效率，管理可接受。
- 信息管理系统总体设计包括：硬件平台设计，软件平台设计，计算模式设计。
- 信息管理系统详细设计包括：功能结构设计，代码设计，数据库设计，界面设计。
- 用户界面（UI, User Interface）描述用户如何与计算机系统交互。
- 输入设计：对系统的质量起着决定性的影响。输入数据的正确性直接决定系统输出信息的正确性。
- 输出设计的目的是使系统能输出满足用户需要的有用信息。
- 系统设计报告，也称系统设计说明书。
- 系统设计报告是系统设计阶段的工作成果。
- 系统设计报告经审议成为有约束力的指导性文件。
- 系统设计报告是系统实现的依据。

------用户界面（UI，User Interface）

- 描述用户如何与计算机系统交互。
- 包括所有影响用户和计算机进行双向交流的硬件、软件、屏幕、菜单、功能和特征。

------用户界面的构成：

- 表单：如窗口、对话框。
- 菜单：用户可以通过菜单获得应用系统的全部功能，也是驱动系统运行的方式之一。
- 控件：构成了屏幕、表单和菜单（工具栏），在所有系统中通用。

------输入输出设计

- 数据采集：数据采集使用自动或手动装置来识别源数据并将其转换成计算机可读的形式
- 要注意：这些数据获取设备中使用的数据格式。
- 输出设计目的正确及时反映和组成用户需要的信息。
- 注意：先有输出设计，后有输入设计。

------输出设计的原则：

- 信息系统的输出应该简洁明了，易于阅读和理解。
- 信息系统输出应该是及时的。
- 访问信息系统输出信息的用户必须是经过授权的。
- 信息系统的输出必须是有效的。

------信息管理系统的一般类型

- 办公自动化系统：提供有效的方式处理个人和组织的业务数据，进行计算并生成文件。
- 通信系统：帮助人们协同工作，以多种不同形式交流并共享信息。
- 交易处理系统：收集和存储交易信息并对交易过程的一些方面进行控制。
- 管理信息系统和执行信息系统：将 TPS 数据转换成信息以监控绩效和管理组织，以可接收的形式向执行者提供信息。
- 决策支持系统：通过提供信息，模型和分析工具来帮助管理者制定决策。
- 企业系统：产生并维持一致的数据处理方法以及跨多种企业职能的集成数据库。
- 信息管理系统的基本功能
- 数据处理功能。包括数据收集和输入、数据传输、数据存储、数据加工和输出;
- 预测功能。运用现代数学方法、统计方法和模拟方法，根据过去的数据预测未来的情况;
- 计划功能。根据企业提供的约束条件，合理地安排各职能部门的计划，按照不同的管理层，提供不同的管理层，提供相应的计划报告;
- 控制功能。根据各职能部门提供的数据，对计划的执行情况进行检测、检测、比较执行与计划的差异，对差异情况分析其原因;
- 辅助决策功能。采用各种数学模型和所存储的大量数据，及时推倒出有关问题的最优解或满意解，辅助各级管理人员进行决策，以期合理利用人财物和信息资源，取得较大的经济效益。

------信息管理系统的开发原则

- 创新原则，体现先进性。计算机技术的发展十分迅速，要及时了解新技术，使用新技术，使目标系统较原系统有质的飞跃。
- 整体原则，体现完整性。企业管理可以理解为一个合理的'闭环'系统。目标系统应当是这个'闭环'系统的完善。企业完整的实现计算机管理不一定必须在企业的各个方面同时实现，但必须完整的设计系统的各个方面。
- 不断发展原则，体现超前性。为了提高使用率，有效的发挥 MIS 的作用，应当注意技术的发展和环境的变化。MIS 在开发过程中应注重不断发展和超前意识。
- 经济原则，体现实用性。大而全和高精尖并不是成功 MIS 的衡量标准。事实上许多失败的 MIS 正是由于盲目追求高新技术而忽视了其实用性。盲目追求完善的 MIS 而忽视了本单位的技术水平、管理水平和人员素质。

------信息管理系统的开发过程

- 规划阶段：系统规划阶段的任务是：在对原系统进行初步调查的基础上提出开发新系统的要求，根据需要和可能，给出新系统的总体方案，并对这些方案进行可行性分析，产生系统开发计划和可行性研究报告两份文档。
- 分析阶段：系统分析阶段的任务是根据系统开发计划所确定的范围，对现行系统进行详细调查，描述现行系统的业务流程，指出现行系统的局限性和不足之处，确定新系统的基本目标和逻辑模型，这个阶段又称为逻辑设计阶段。系统分析阶段的工作成果体现在“系统分析说明书”中，这是系统建设的必备文件。它是提交给用户的文档，也是下一阶段的工作依据，因此，系统分析说明书要通俗易懂，用户通过它可以了解新系统的功能，判断是否所需的系统。系统分析说明书一旦评审通过，就是系统设计的依据，也是系统最终验收的依据。
- 设计阶段：系统分析阶段回答了新系统“做什么”的问题，而系统设计阶段的任务就是回答“怎么做”的问题，即根据系统分析说明书中规定的功能要求，考虑实际条件，具体设计实现逻辑模型的技术方案，也即设计新系统的物理模型。所以这个阶段又称为物理设计阶段。它又分为总体设计和详细设计两个阶段，产生的技术文档是“系统设计说明书”。
- 实施阶段：系统实施阶段的任务包括计算机等硬件设备的购置、安装和调试，应用程序的编制和调试，人员培训，数据文件转换，系统调试与转换等。系统实施是按实施计划分阶段完成的，每个阶段应写出“实施进度报告”。系统测试之后写出“系统测试报告”。
- 维护与评价：系统投入运行后，需要经常进行维护，记录系统运行情况，根据一定的程序对系统进行必要的修改，评价系统的工作质量和经济效益。

------信息管理系统的划分方法

- 基于组织职能进行划分：MIS 按组织职能可以划分为办公系统、决策系统、生产系统和信息系统。
- 基于信息处理层次进行划分：MIS 基于信息处理层次进行划分为面向数量的执行系统、面向价值的核算系统、报告监控系统，分析信息系统、规划决策系统，自底向上形成信息金字塔。
- 基于历史发展进行划分：第一代 MIS 是由手工操作，使用工具是文件柜、笔记本等。第二代 MIS 增加了机械辅助办公设备，如打字机、收款机、自动记账机等。第三代 MIS 使用计算机、电传、电话、打印机等电子设备。
- 基于规模进行划分：随着电信技术和计算机技术的飞速发展，现代 MIS 从地域上划分已逐渐由局域范围走向广域范围。
- MIS 的综合结构：MIS 可以划分为横向综合结构和纵向综合结构，横向综合结构指同一管理层次各种职能部门的综合，如劳资、人事部门。纵向综合结构指具有某种职能的各管理层的业务组织在一起，如上下级的对口部门。