# ASP.Net服务端控件

ASP.Net服务端控件是ASP.Net对HTML的封装，在C#代码中就可以用txt1.Text=‘abc’这种方式来修改input的值，ASP.Net会将服务端控件转成HTML代码输出给浏览器。服务端控件是ASP.Net非常吸引初学者、非常容易上手的东西，也是最被人诟病的东西。物尽其用，服务端控件在内网系统、互联网系统的后台部分等访问频率不高的地方用的还是很适合的。）

所有的ASP.Net大部分都是从Control、WebControl类继承的，几乎都有的成员有：

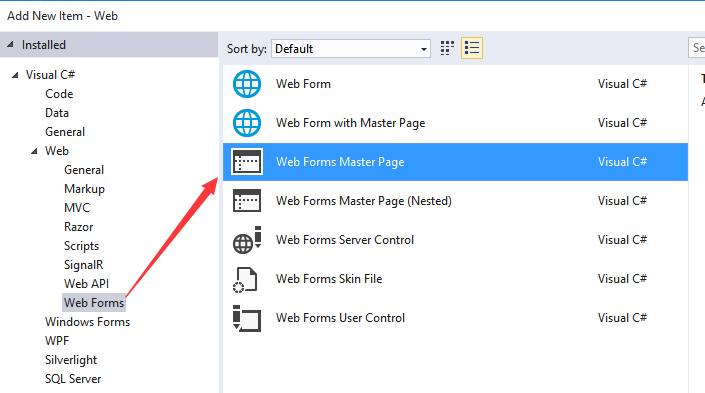
## 常用成员

### ClientID

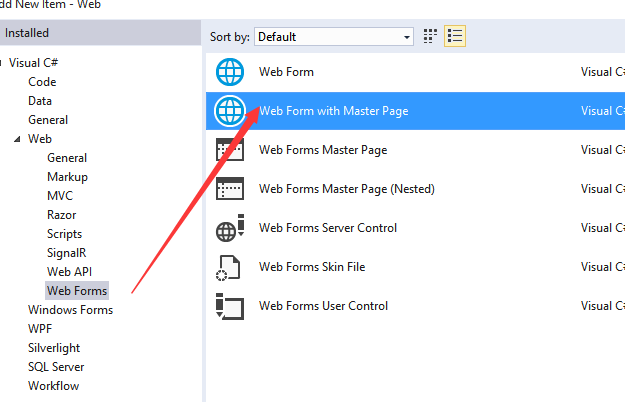
控件在客户端的Id，控件在服务端的Id不一定等于客户端HTML中的Id，比如说在ListView等控件的模板中。因此如果要在客户端通过JavaScript Dom、JQuery的getElementById、$(“#id”)来操作控件的话最好不要直接写服务端Id，而是$(‘#<%=txt1.ClientID%>’)。用JQuery事件设置鼠标移到控件上和从控件移开的不同样式。在用户控件中就可以看到ClientID和id的不同。UserControl、母版、ListView。推荐永远用ClientID

#### 案例

1. 创建模板页



1. 再创建带有模板页的页面



1. 添加两个控件



1. 在浏览器中查看源代码，可以看到id与服务端中不一样



1. 在服务端获得客户端id需要用ClientID

|  |
| --- |
| <asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">  <script type="text/javascript">  window.onload = function () {  var dom = document.getElementById('<%=this.Button1.ClientID %>');  dom.onclick = function () {  var lable = document.getElementById("<%=this.Label1.ClientID %>");  lable.innerHTML = "heelo";  return false;  };    };  </script>  </asp:Content> |

### Visible 属性

控件是否可见，如果Visible=False是不会转到HTML中的，这和在HTML中给元素style.display='none'效果是不一样的。

|  |
| --- |
| protected void Button1\_Click(object sender, EventArgs e)  {  Calendar1.Visible = !Calendar1.Visible;  } |

### CssClass

控件的样式名，就是HTML中控件的class属性。也可以单独修改BackColor、BorderStyle等属性，但是不建议这么做，因为会生成很多的内联样式，生成html尺寸大，不便于统一的修改。

### Attributes

用来设置获取控件的额外属性。和Dom中的setAttribute()、getAttribute()是一样的。Button1.Attributes[“a1”] = “2.jpg”; HTML 事件在服务端看来也是属性 Button1.Attributes[“onmouseover”] = “alert(‘hello’)”;

* 说明：所有的服务端控件不仅可以使用控件定义的属性，还可以使用额外的属性，这些属性包括控件没有封装的HTML属性（比如onmouseover等浏览器端事件页当作属性），ASP.Net会将它不识别的属性原封不动的渲染到客户端。在代码中也可以通过Attributes属性设置额外属性：CheckBox1.Attributes["onmouseover"] = "alert('hello')";

## 服务端基本控件

* 1、Label控件。Text属性为显示文本。AssociatedControlID属性用来关联一个控件，如果为空的话会展示为一个Span，如果指定为一个控件的id，则会展示为一个HTML中的<Label>并且将for属性设置为被关联控件的ClientId。
* 2、Literal控件也是展示一段文本，但是Literal控件不会渲染任何额外的标签，就是将Text属性的值展示出来而已。
* 3、TextBox控件，文本框控件。TextMode属性取值SingleLine、MultiLine、Password，分别渲染为input(type=text)、textarea和input(type=password)。当AutoPostBack属性为true的时候，用户焦点离开TextBox就会造成页面Post，实现原理就是讲ASP.Net原理时的AutoPostBack。TextChanged事件(需要AutoPostBack=true才会修改后触发，否则是页面提交才触发)，文本发生变化的时候事件触发。ASP.Net中要提交表单的时候最好调用\_\_doPostBack方法。
* 4、RadioButton控件，对应为input(type=radio)，通过GroupName属性进行分组
* 5、Button控件。OnClientClick属性，当用户点击按钮的时候在浏览器端执行的代码，注意OnClientClick是字符串属性，写的代码是JavaScript代码，运行在浏览器端。<asp:Button ID="btnDel" runat="server" onclientclick="return confirm('真的要删除吗？')" Text="删除" />
* 6、LinkButton，用法和Button差不多，区别就是Button控件渲染为按钮，而LinkButton渲染为超链接。不要用LinkButton来实现普通的超链接，因为LinkButton的href为一段javascript代码，进行的是表单的Post，无法“在新窗口中打开连接”。和讲“行删除”那个例子中href为javascript的超链接原理一样。一般用Button就行，只有用户要求“长成超链接好”采用LinkButton，就是一个在浏览器中渲染成超链接，服务端当成按钮用的东西。
* HyperLink：IsPostBack=False，没有提交表单内容，没有OnClick事件。
* 7、ImageButton控件也和Button差不多，只不过是显示为图片，渲染为input(type=image)
* 8、Button、LinkButton、ImageButton等控件都有CommandName、CommandArgument两个属性和Command事件，可以让多个按钮控件共享一个Command事件处理函数，通过读取事件对象e的CommandName、CommandArgument两个属性读取被点击按钮上设置的这两个参数来执行不同的操作。例子：编辑、删除多行数据。这种用法在ListView等控件中用的最多。
* 9、Panel控件用来盛放一些控件。如果设定GroupingText属性那么就渲染为含有<fieldset>的div标签，也就是GroupBox效果，否则渲染为<div>
* 10、HyperLink控件，超链接。和LinkButton不一样(常考)，不会向服务器端Post，就是一个超链接。NavigateURL：链接地址；Text：显示文本。如果设定ImageUrl属性则会显示图片超链接。
* 11、FileUpload控件，文件上传控件。渲染成input(type=file)。属性：FileContent以流形式获得上传的文件；FileName 上传文件名；HasFile Bool值，表示用户是否选择文件，SaveAs方法用于将文件保存到磁盘的指定位置。漏洞：文件上传漏洞（上传一个下载源代码的aspx、Process.Start启动格式化，创建管理员、开启远程桌面）。解决方法：只允许上传指定类型文件，上传文件夹不给执行权限。

## 三种控件

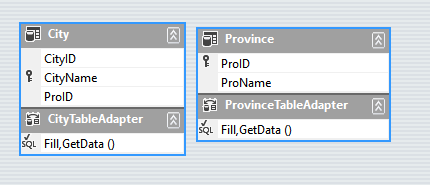
* HTML控件，ASP.Net把HTML控件当成普通字符串渲染到浏览器端，不去检查正确性、无法在服务器端进行处理。
* ASP.Net服务端控件，经过ASP.Net高度封装的控件，使用简单，运行在服务器端，可以在服务端使用C#代码进行操作，会渲染到客户端为HTML控件。
* runat=server的HTML控件。在HTML控件的基础上添加runat="server"，也是运行在服务器端的，也可以服务端使用C#代码进行操作，也会渲染到客户端，不像ASP.Net服务端控件那样高度封装，暴露的属性大部分是普通HTML属性。和ASP.Net服务端控件相比的好处是：当需要在服务器端要对控件进行操作的时候，如果控件没有被ASP.Net服务端控件封装的时候，用runat=server的HTML控件很方便，runat=server的HTML控件也会对虚拟路径(~/)、id→ClientID进行处理，所以在使用虚拟路径、UserControl中也可能会用到onmouseover="document.getElementById('<%=TextBox1.ClientID%>').value='哈哈';"

直接在属性中有问题，会把<%直接输出到浏览器端，因此不要在控件的属性值中写<%%>。

# 数据绑定

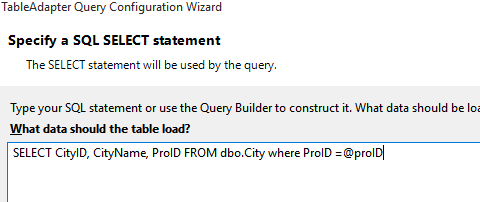
## Dropdownlist实现省市联动

### 创建强类型dataset

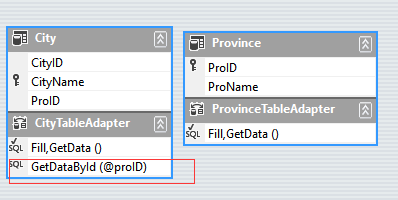


为city添加一个查询方法：GetDateById。

语句如下：



结果：



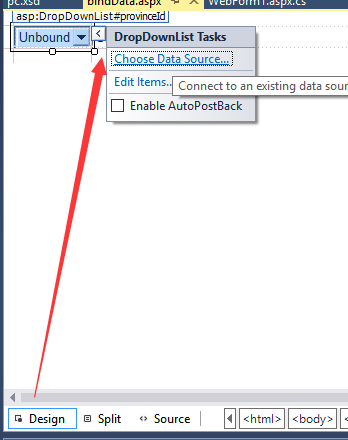
### 添加DropDownList

在aspx中添加服务端控件

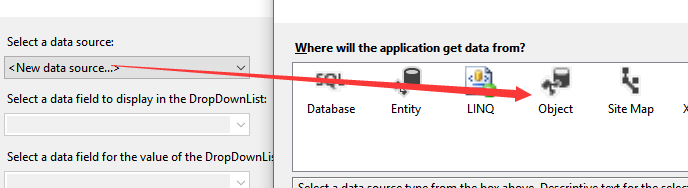
|  |
| --- |
| <asp:DropDownList ID="provinceId" runat="server"></asp:DropDownList>  <asp:DropDownList ID="cityId" runat="server"></asp:DropDownList> |

### 绑定省数据

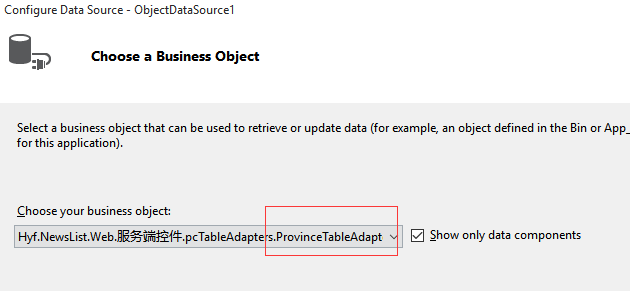
1. 在design中点击控件右上角的“<”按钮，选中”选中新数据源“。



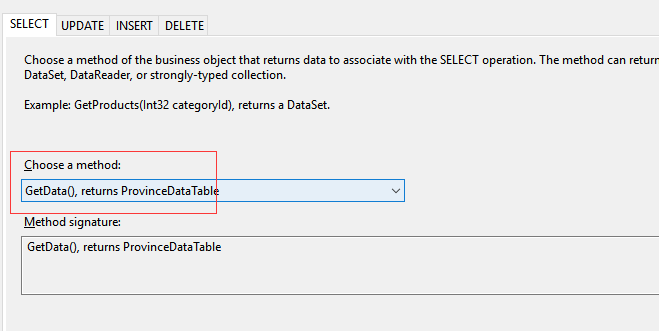
1. 选中object



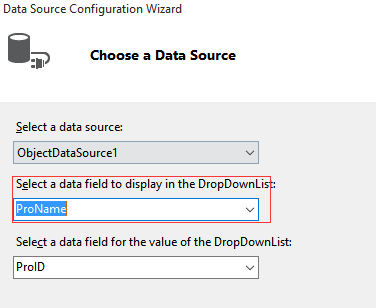
1. 选中省的适配器



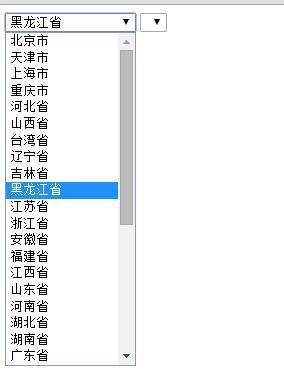
1. 选中方法



1. 选择要显示的属性

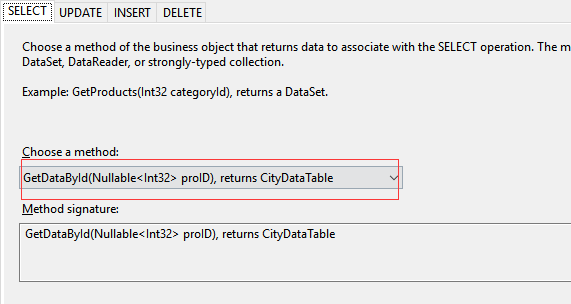


1. 预览

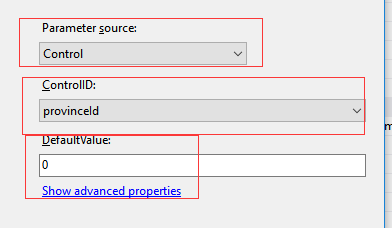


### 绑定市数据

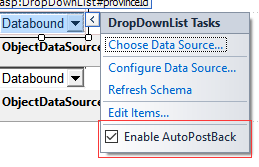
1. 步骤与省数据绑定类似，但是在选择方法时要选择我们自定义的那个方法



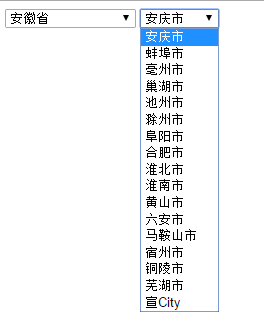
1. 参数选择



1. 打开省DropDownList的自动返回



1. 结果：



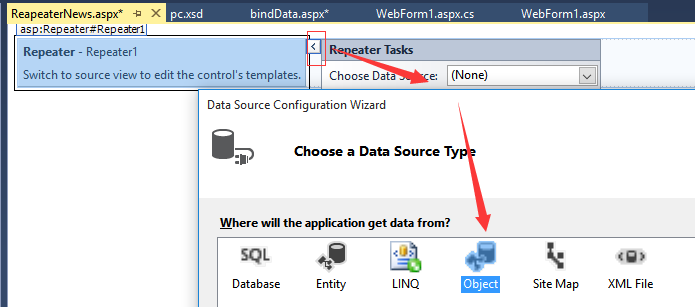
## Repeater实现新闻列表

### 添加repeater控件

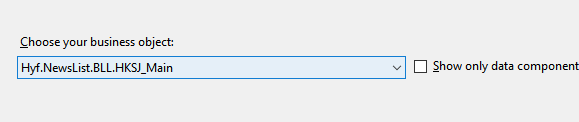
|  |
| --- |
| <asp:Repeater ID="Repeater1" runat="server">  </asp:Repeater> |

### 添加数据源

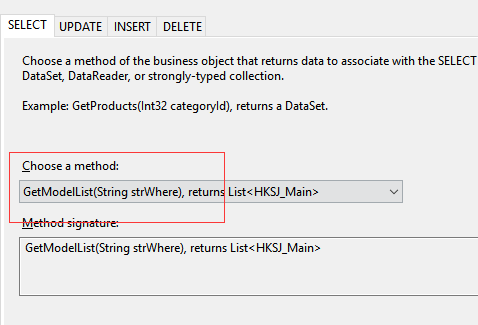
1. 新数据源



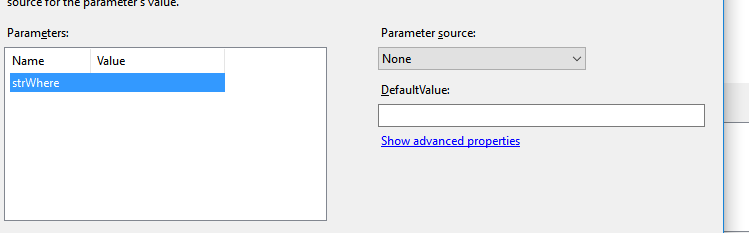
1. 选中业务对象



1. 选择方法



1. 参数为空。注意：defaultvalue中需要有一个空格。



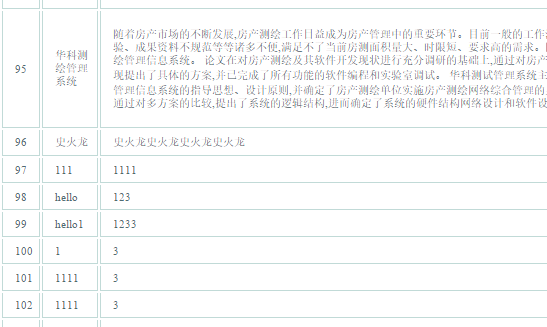
### 展示数据

在repeater外添加table。Repeater中添加一个项目模板（ItemTemplate）。这个项目模板会根据数据源中的对象列表创建。

|  |
| --- |
| <%# Eval("ID") %> |

上面的代码意思是在该代码所在位置输出数据源中属性为Id的值。

|  |
| --- |
| <table >  <asp:Repeater ID="Repeater1" runat="server" DataSourceID="ObjectDataSource1">  <ItemTemplate>  <tr>  <td><%# Eval("ID") %></td>  <td><%# Eval("title") %></td>  <td><%# Eval("content") %></td>  </tr>  </ItemTemplate>  </asp:Repeater>  </table> |

查看效果：  


### 丰富效果

#### 添加头模板与尾模板

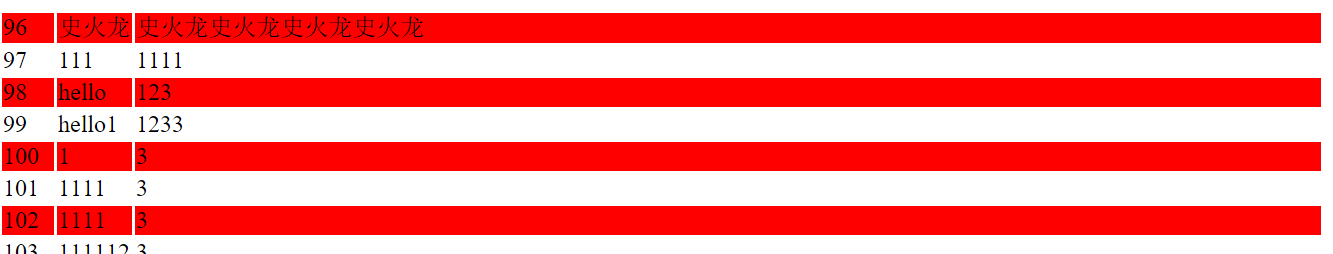
注意：<table>标签移动到头与尾模板中了。

|  |
| --- |
| <HeaderTemplate>  <table>  <tr>  <th>编号：</th>  <th>标题：</th>  <th>内容：</th>  </tr>  </HeaderTemplate>  <ItemTemplate>  <tr>  <td><%# Eval("ID") %></td>  <td><%# Eval("title") %></td>  <td><%# Eval("content") %></td>  </tr>  </ItemTemplate>  <FooterTemplate>  </table>  </FooterTemplate> |

#### 添加AlternatingItemTemplate

|  |
| --- |
| <tr style="background-color:red">  <td><%# Eval("ID") %></td>  <td><%# Eval("title") %></td>  <td><%# Eval("content") %></td>  </tr> |

这样每隔一行会呈现红色



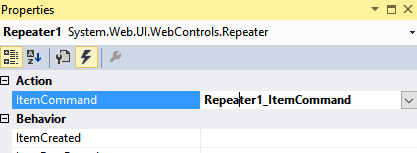
### 删除

#### 添加删除按钮

|  |
| --- |
| <ItemTemplate>  <tr>  <td><%# Eval("ID") %></td>  <td><%# Eval("title") %></td>  <td><%# Eval("content") %></td>  <td>  <asp:Button ID="btnDelete" CommandArgument='<%# Eval("ID") %>' CommandName="Delete" runat="server" Text="删除" OnClientClick="return confirm('确定要删除？')" />  </td>  </tr>  </ItemTemplate> |

#### ItemCommand事件

为ItemCommand事件添加方法



方法内容：

|  |
| --- |
| protected void Repeater1\_ItemCommand(object source, RepeaterCommandEventArgs e)  {  } |

#### 删除数据

|  |
| --- |
| protected void Repeater1\_ItemCommand(object source, RepeaterCommandEventArgs e)  {  //根据CommandName判断是否点击的是删除按钮  if (e.CommandName=="Delete")  {  //获取id  int id = int.Parse(e.CommandArgument.ToString());  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  mainService.Delete(id);  this.Repeater1.DataBind();//刷新  }  } |

### 分页

#### 修改绑定数据

|  |
| --- |
| List<Model.HKSJ\_Main> models;  protected void Page\_Load(object sender, EventArgs e)  {  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  int pageSize = int.Parse(Request["pageSize"] ?? "5");  int pageIndex = int.Parse(Request["pageIndex"] ?? "1");  int total = 0;  models = mainService.LoadPageData(pageIndex, pageSize, out total);  if (!IsPostBack)  {  this.Repeater1.DataSource = models;  this.Repeater1.DataBind();  }  } |

#### 删除按钮的重新绑定

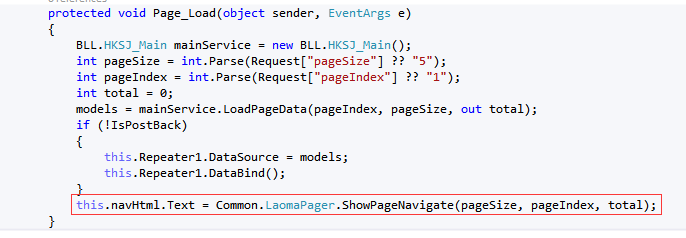
|  |
| --- |
| protected void Repeater1\_ItemCommand(object source, RepeaterCommandEventArgs e)  {  //根据CommandName判断是否点击的是删除按钮  if (e.CommandName == "Delete")  {  //获取id  int id = int.Parse(e.CommandArgument.ToString());  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  mainService.Delete(id);  this.Repeater1.DataSource = models;  this.Repeater1.DataBind();//刷新  }  } |

#### 插入分页div

在repeater下面插入div，内部添加一个literal

|  |
| --- |
| </asp:Repeater>  <div id="navDiv" class="paginator">  <asp:Literal ID="navHtml" runat="server"></asp:Literal>  </div> |

#### 设置literal内容



效果：

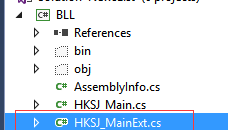


## ListView实现新闻列表

### 创建删除方法

ListView可以实现CRUD，但是在绑定时，方法传入的参数需要是实体。动软生成的代码没有通过实体删除的方法，首先需要扩展一个方法

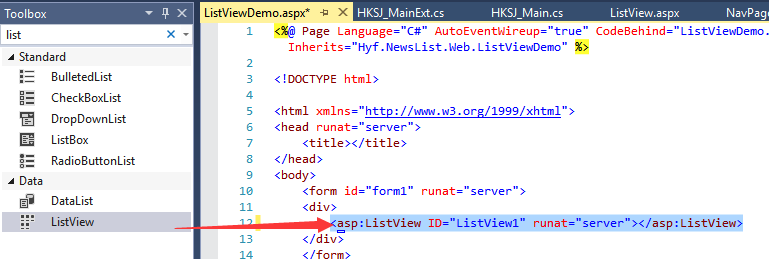
这里我们新建一个cs扩展文件（遵循可扩展不可修改的原则）。



之后编写代码如下：

|  |
| --- |
| namespace Hyf.NewsList.BLL  {  public partial class HKSJ\_Main  {  public void Delete(Model.HKSJ\_Main model)  {  this.Delete(model.ID);  }  }  } |

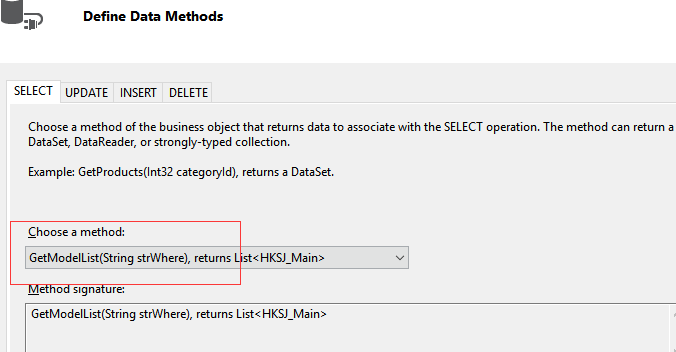
### 创建listview



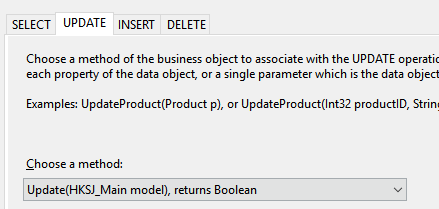
### 绑定数据

绑定步骤与前2节相似，只是在选择方法时需要设置

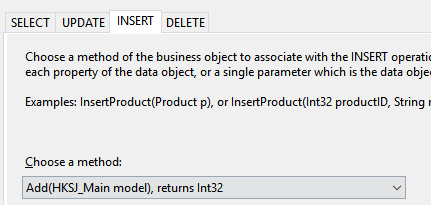
选择select方法



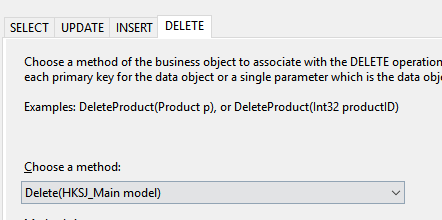
选择更新方法



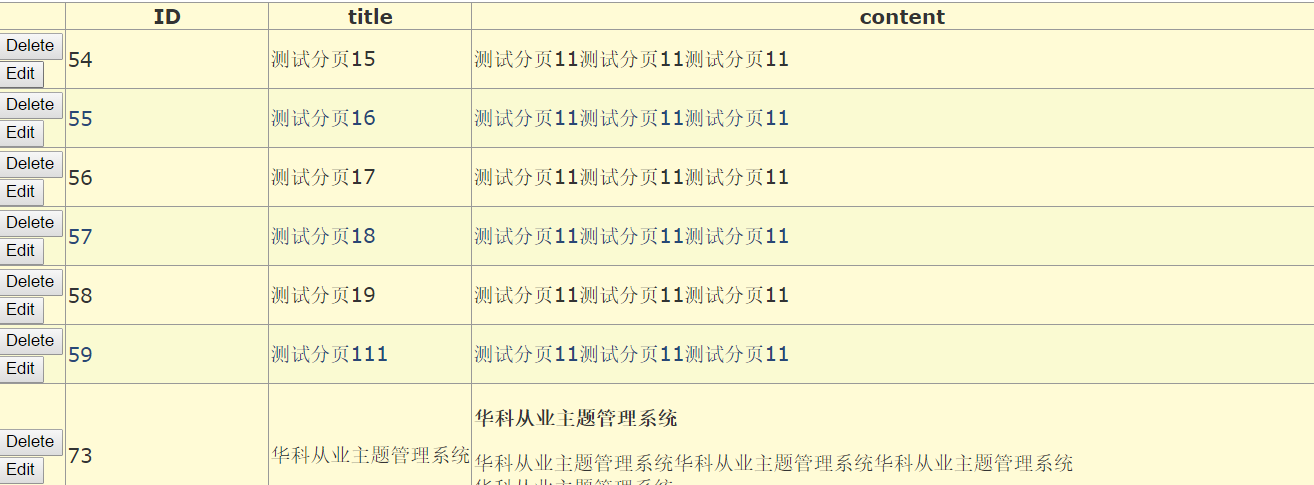
选择插入方法



选择删除方法

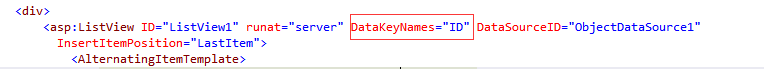


预览



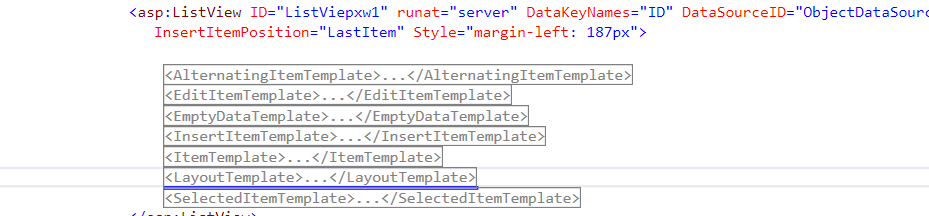
### 删除需要设置属性

在ListView中设置DataKeyNames为主键的字段



### 修改表格标题为中文

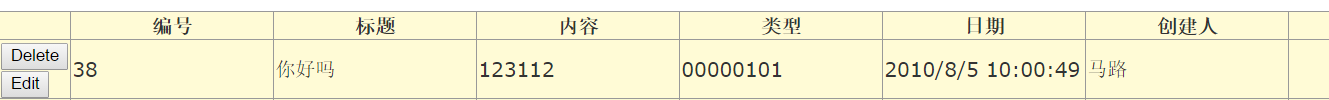
Listview有多个模板



在LayoutTemplate修改



效果：

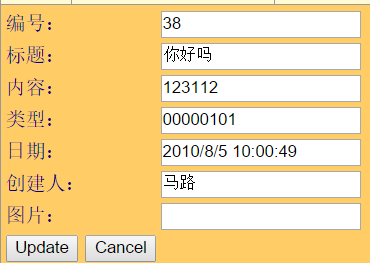


### 竖行进行编辑

编辑EditItemTemplate

|  |
| --- |
| <EditItemTemplate>  <tr style="background-color: #FFCC66; color: #000080;">  <!—添加一个td设置其宽度-->  <td colspan="8">  <!—套一个table-->  <table>  <tr>  <td>编号：</td>  <td>  <asp:TextBox ID="IDTextBox" Enabled="false" runat="server" Text='<%# Bind("ID") %>' />  </td>  </tr>  <tr>  <td>标题：</td>  <td>  <asp:TextBox ID="titleTextBox" runat="server" Text='<%# Bind("title") %>' />  </td>  </tr>  <tr>  <td>内容：</td>  <td>  <asp:TextBox ID="contentTextBox" runat="server" Text='<%# Bind("content") %>' />  </td>  </tr>  <tr>  <td>类型：</td>  <td>  <asp:TextBox ID="typeTextBox" runat="server" Text='<%# Bind("type") %>' />  </td>  </tr>  <tr>  <td>日期：</td>  <td>  <asp:TextBox ID="DateTextBox" runat="server" Text='<%# Bind("Date") %>' />  </td>  </tr>  <tr>  <td>创建人：</td>  <td>  <asp:TextBox ID="peopleTextBox" runat="server" Text='<%# Bind("people") %>' />  </td>  </tr>  <tr>  <td>图片：</td>  <td>  <asp:TextBox ID="picUrlTextBox" runat="server" Text='<%# Bind("picUrl") %>' />  </td>  </tr>  <tr>  <td>  <!—更新与放弃按钮放到最后面-->  <asp:Button ID="UpdateButton" runat="server" CommandName="Update" Text="Update" />  <asp:Button ID="CancelButton" runat="server" CommandName="Cancel" Text="Cancel" />  </table>  </td>  </tr>  </EditItemTemplate> |

效果：



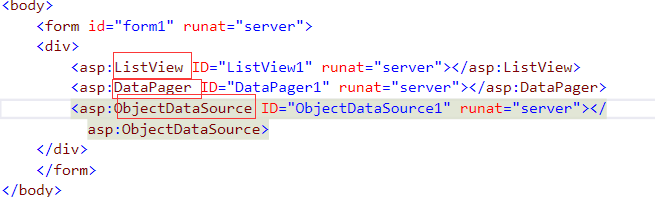
### 分页

Listview自带的分页，是从数据库中拿出所有数据，再判断展示的是那些数据。当数据量大时，效率低下。

可以用DataPaper实现“真分页”，也就是仅索取该页的数据进行展示

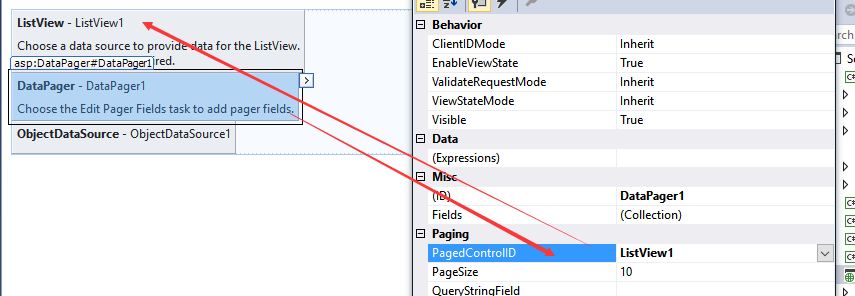
#### 添加控件

添加以下三个控件



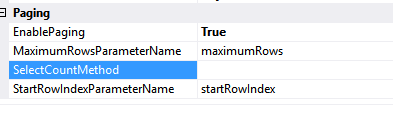
#### 设置DataPager的PagedControlID

设置分页控件给谁用。



#### 设置objectDataSource的Paging

EnablePaging属性设为true，selectCountMethod需要数据源连接的实体中包含的方法，该方法用来获取数据总条数。maximumRows和startRowIndex为获取数据时需要传入的参数。



所以，需要扩展两个方法来实现分页。现在我们在HKSJ\_MainExt类文件添加2个方法。

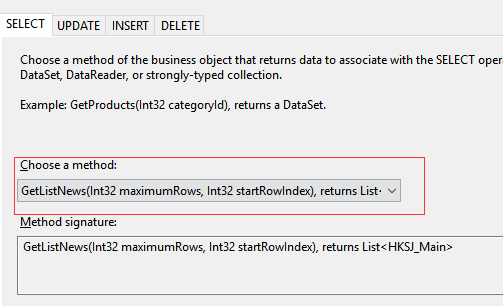
|  |
| --- |
| public int GetCount()  {  return this.GetRecordCount(string.Empty);  }  //参数名称需要与objectDataSource的paging中的那两个参数一致。  public List<Model.HKSJ\_Main> GetListNews(int maximumRows, int startRowIndex)  {  var ds = this.GetListByPage(string.Empty, "id", startRowIndex + 1, startRowIndex + maximumRows);  return this.DataTableToList(ds.Tables[0]);  } |

SelectCountMethod设置为GetCount。

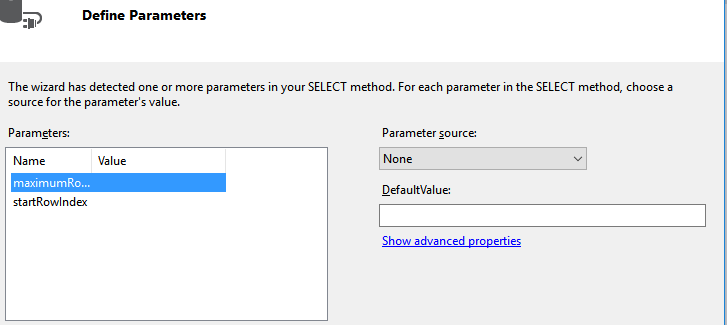


#### 设置数据源

在绑定select方法时选择在上面创建的GetListNews方法。



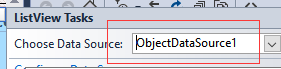
在设置参数时留空即可



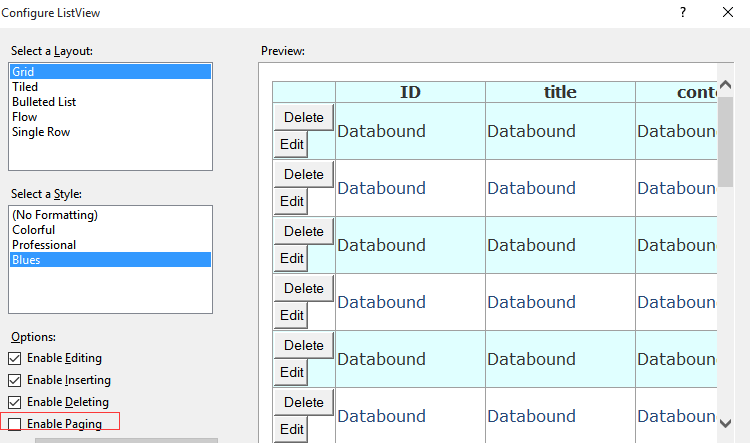
注释掉下面的代码

|  |
| --- |
| <%-- 这两个数据是由分页控件提供的，这里需要注释或删掉--%>  <%-- <SelectParameters>  <asp:Parameter Name="maximumRows" Type="Int32" />  <asp:Parameter Name="startRowIndex" Type="Int32" />  </SelectParameters>--%> |

#### ListView绑定数据



listView设置中不能选中“enable paging”。

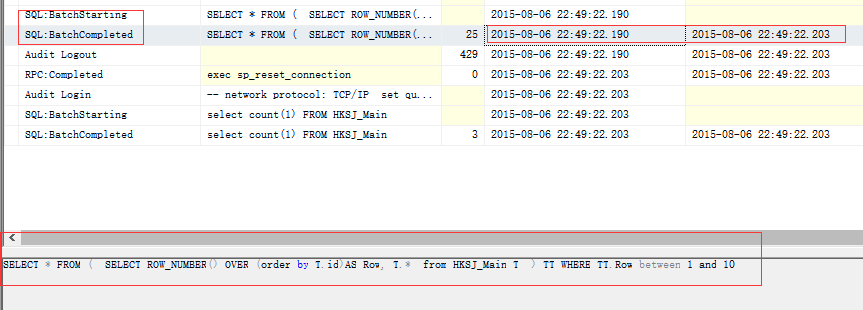


最后还需要设置主键才能删除



#### 用SQL Server Profiler进行监视

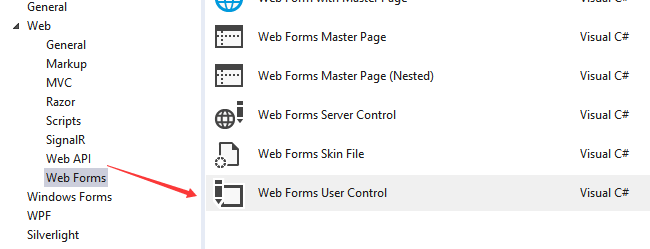
可以发现检索用的时间不到1毫秒。



# 模板页

## 用户控件

### 创建用户控件



### 添加内容

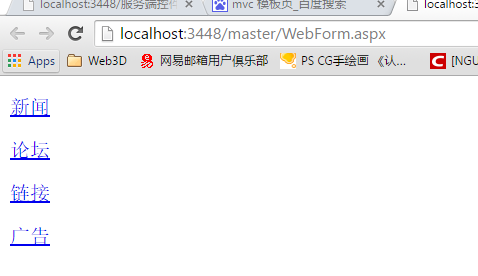
|  |
| --- |
| <div>  <p><a href="#">新闻</a></p>  <p><a href="#">论坛</a></p>  <p><a href="#">链接</a></p>  <p><a href="#">广告</a></p>  </div> |

### 在页面中使用空间

只需要拖拽控件页面到要使用的页面中即可。



### 预览

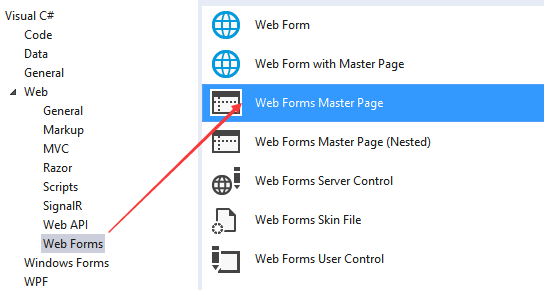


## 模板页

当大量网页具有相同的结构的时候，每一次都需要重新布局，即使用控件，每个网页也需要拖拽控件将来，还需要布局。

微软提供了模板技术，相同的地方，都放到模板文件中，为不同的地方留个“洞”。当使用时只需要把“洞”添上即可。

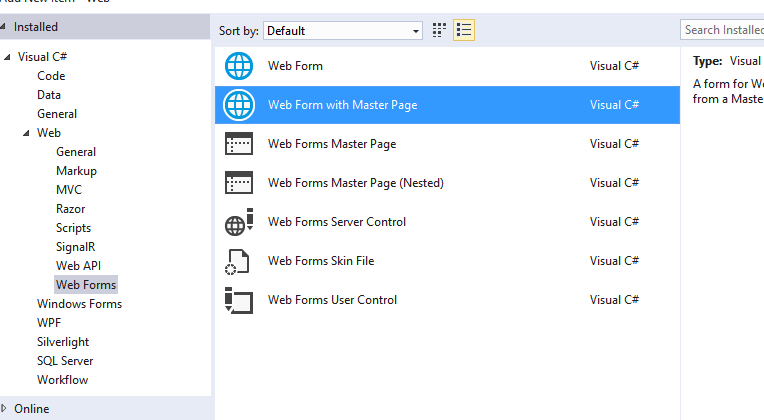
### 创建模板页



### 编辑内容

|  |
| --- |
| <%@ Master Language="C#" AutoEventWireup="true" CodeBehind="MainFrameSite.master.cs" Inherits="WebDemo.Web.\_2013\_12\_5.MainFrameSite" %>  <!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>  <%--留下来的洞--%>  <asp:ContentPlaceHolder ID="head" runat="server">  </asp:ContentPlaceHolder>    <style type="text/css">  #header {  height: 100px;  width: 100%;  }  #leftBar {  float: left;  margin-left: 0px;  width: 200px;  border: 1px solid red;  height: 500px;  }  #mainWork {  margin-left: 210px;  border: 1px solid red;  height: 500px;  }  #footer {  clear: both;  height: 20px;  width: 100%;  }  body {  padding: 0px;  margin: 0px;  }  </style>  </head>  <body>  <form id="form1" runat="server">  <div id="header">    </div>  <div id="leftBar">  <p><a href="#">修改</a></p>  <p><a href="#">更新</a></p>  <p><a href="#">删除</a></p>  <p><a href="#">刷新</a></p>  </div>  <div id="mainWork">  <%--留下来的洞--%>  <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">    </asp:ContentPlaceHolder>  </div>  <div id="footer">  版权所有 @......xx  </div>  </form>  </body>  </html> |

### 新建带模板的页面



### 填充内容

|  |
| --- |
| <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">  <p>段落</p>  <h1>hello</h1>  </asp:Content> |

效果：

