# 什么是Ajax

AJAX:”Asynchronous JavaScript and XML”

中文意思：异步JavaScript和XML。

指一种创建交互式网页应用的网页开发技术。

不是指一种单一的技术，而是有机地利用了一系列相关的技术：

简单理解为：JavaScript + XMLHttpRequest + CSS +服务器端 的集合.

## 案例1:异步获取服务器IP

### 创建后台处理程序

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  string ServerHost = context.Request.ServerVariables.Get("Local\_Addr").ToString();  context.Response.Write(ServerHost);  } |

### 创建HTML网页

添加一个按钮

|  |
| --- |
| <input type="button" value="获取服务器时间" id="GetServerHost" /> |

用Javascript注册一个点击事件，并实现异步

|  |
| --- |
| <script type="text/javascript">  window.onload = function () {  //获取按钮  var btn = document.getElementById("GetServerHost");  //注册点击事件  btn.onclick = function () {  var xhr;  //如果浏览器支持XMLHttpRequest  if (XMLHttpRequest) {  xhr = new XMLHttpRequest();//创建XMLHttpRequest对象  } else {//IE6  xhr = new ActiveXObject("Microsoft XMLHTTP");  }  //打开后台一个链接。第一个参数是请求方法，第二个请求网址，第三个是否异步（是否开辟一个线程）。  xhr.open("get", "Handler1.ashx?id=3", true);  //发送数据  xhr.send();  //监听异步请求状态改变的事件  xhr.onreadystatechange = function () {  if (xhr.readyState == 4) {//请求结束  if (xhr.status == 200) {//请求成功  alert(xhr.responseText);//获得返回的数据  }  }  };  };  };  </script> |

如果用post方法进行异步请求，需要用下面的方式：

|  |
| --- |
| xhr.open("post", "Handler1.ashx?", true);  //需要设置请求头的Content-type  xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  //在send中通过参数传递信息。  xhr.send("id=3"); |

这样就可以无刷新获得服务器IP了

## 案例2：登陆

### 创建后台处理程序

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  //收集数据  string name = context.Request.Form["name"];  string pwd = context.Request.Form["pwd"];  //判断用户名和密码  if (name == "admin" && pwd == "123")  {  context.Response.Write("ok");  }  else  {  context.Response.Write("登陆失败！");  }  } |

### 前台HTML

创建登陆用标签

|  |
| --- |
| <body>  <table>  <tr>  <td>用户名：</td>  <td>  <input type="text" name="txtName" id="txtName" />  </td>  </tr>  <tr>  <td>密码：</td>  <td>  <input type="text" name="txtPwd" id="txtPwd" />  </td>  </tr>  <tr>  <td colspan="2" style="text-align: right">  <input type="submit" id="btnLogin" value=" 提交" /></td>  </tr>  </table>  </body> |

效果：



点击提交触发javascript代码

|  |
| --- |
| <script type="text/javascript">  window.onload = function () {  var btn = document.getElementById("btnLogin");  btn.onclick = function () {  var txtName = document.getElementById("txtName").value;  var txtPwd = document.getElementById("txtPwd").value;  var xhr;  if (XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject("Microsoft.XMLHTTP");  }  var strUrl = "LoginHandler.ashx";  var params = "name=" + txtName + "&pwd=" + txtPwd;  xhr.open("post", strUrl, true);  xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  xhr.send(params);  xhr.onreadystatechange = function () {  if (xhr.readyState == 4 && xhr.status == 200) {  if (xhr.responseText == "ok") {  window.location.href = "Main.html";  } else {  alert(xhr.responseText);  }  }  };  };  };  </script> |

对异步进行封装

|  |
| --- |
| <script type="text/javascript">  window.onload = function () {  var btn = document.getElementById("btnLogin");  btn.onclick = function () {  var txtName = document.getElementById("txtName").value;  var txtPwd = document.getElementById("txtPwd").value;  var strUrl = "LoginHandler.ashx?name" + txtName + "&pwd=" + txtPwd;  //调用  MyAjax("get", strUrl,  function (data) {  if (data == "ok") {  window.location.href = "Main.html";  } else {  alert(data);  }  }  );  };  };  //封装方法  function MyAjax(httpMethod, url, callBack) {  var xhr;  if (XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject("Microsoft.XMLHTTP");  }  xhr.open(httpMethod, url, true);  xhr.send();  xhr.onreadystatechange = function () {  if (xhr.readyState == 4 && xhr.status == 200) {  callBack(xhr.responseText);  }  };  }  </script> |

## 案例：Jquery异步

### 一般处理程序

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  //收集数据  string name = context.Request["name"];  string pwd = context.Request["pwd"];  //判断用户名和密码  if (name == "admin" && pwd == "123")  {  context.Response.Write("登陆成功");  }  else  {  context.Response.Write("登陆失败！");  }  } |

### 前台HTML代码

添加提交标签

|  |
| --- |
| <input type="submit" id="btnLogin" value=" 提交" /> |

通过Jquery实现异步

#### $.get通过字符串传值

|  |
| --- |
| <script src="Scripts/jquery-2.1.4.min.js"></script>  <script type="text/javascript">  $(function () {  $("#btnLogin").click(function () {  $.get("LoginHandler.ashx", "name=admin&&pwd=123", function (data) {  alert(data);  });  });  });  </script> |

#### $.get通过json传值

|  |
| --- |
| <script type="text/javascript">  $(function () {  $("#btnLogin").click(function () {  $.get("LoginHandler.ashx", { name: "admin", pwd: "123" }, function (data) {  alert(data);  });  });  });  </script> |

#### Post方式发送数据

|  |
| --- |
| <script type="text/javascript">  $(function () {  $("#btnLogin").click(function () {  $.post("LoginHandler.ashx", { name: "admin", pwd: "123" }, function (data) {  alert(data);  });  });  });  </script> |

#### $.getJSON

返回类型需要时Json格式，处理程序如下：

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  string name = context.Request["name"];  string pwd = context.Request["pwd"];  //判断用户名和密码  if (name == "admin" && pwd == "123")  {  //返回类型为jason格式。key需要用双引号括起来。  context.Response.Write("{\"status\":\"登陆成功！\"}");  }  else  {  context.Response.Write("{\"status\":\"登陆失败！\"}");  }  } |

前台Jquery

|  |
| --- |
| $(function () {  $("#btnLogin").click(function () {  $.getJSON("LoginHandler.ashx", { name: "admin", pwd: "123" }, function (data) {  alert(data.status);  });  });  });  </script> |

#### $.ajax最齐全的方式

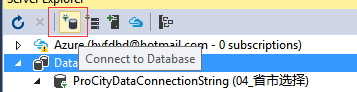
|  |
| --- |
| <script type="text/javascript">  $(function () {  $("#btnLogin").click(function () {  $.ajax({  url: "LoginHandler.ashx",  data: { name: "admin", pwd: 123 },  cach: false,//强迫当前请求去后台拿数据，不能用客户端缓存。  method:"post",  success: function (data) {  alert(data.status);  },  //返回类型  dataType:"json"  });  });  });  </script> |

## 案例：省市列表

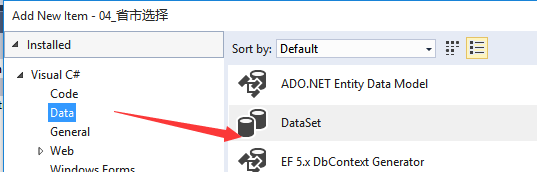
我们用强类型来查询数据库，当网站加载时，添加省份到select标签中，点击不同的省，第二个标签加载该省下的市区信息。

### 创建强类型Dataset

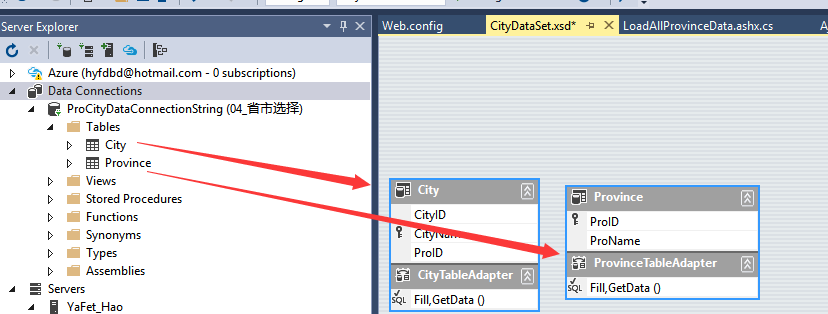
首先用Server Explorer连接数据库



之后，添加强类型dataset



然后，把city和provinces拖拽到xsd中



### 一般处理程序中查询数据

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  StringBuilder sb = new StringBuilder();  //用强类型查询数据  ProvinceTableAdapter adapter = new ProvinceTableAdapter();  CityDataSet.ProvinceDataTable provinces = adapter.GetData();  //遍历table，拼接字符串  foreach (var item in provinces)  {  sb.AppendFormat("<option value='{0}'>{1}</option>", item.ProID, item.ProName);  }  context.Response.Write(sb.ToString());  } |

### 前台异步显示

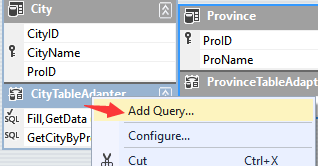
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title></title>  <meta charset="utf-8" />  <script src="Scripts/jquery-2.1.4.min.js"></script>  <script type="text/javascript">  $(function () {  initProvinceSelect();  });  //初始化省信息  function initProvinceSelect() {  $.get("LoadAllProvinceData.ashx", "", function (data) {  $("#province").html(data);  });  }  </script>  </head>  <body>  <select id="province"></select>  <select id="city"></select>  </body>  </html> |

### 根据Pid获得该省下的市信息

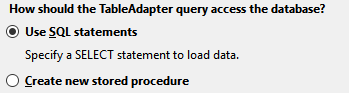
当选中某个省时，取得Pid传递到后台，根据pid查询该省下的市信息。

#### 在xsd中为city添加一个查询语句

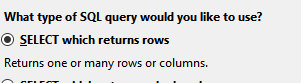
右击，选”Add Query“



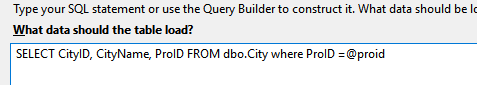
选中下面的选项



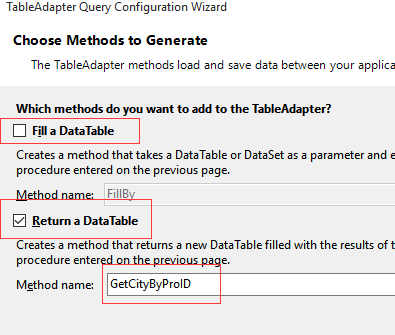
选中下面的返回类型



创建sql语句



方法名称：



#### 在一般处理程序中根据id查询数据并返回json对象

|  |
| --- |
| public class LoadCityByProID : IHttpHandler  {  public void ProcessRequest(HttpContext context)  {  //获得pid  int pId = int.Parse(context.Request["pId"] ?? "0");  //用强类型dataset查询数据  CityTableAdapter adapter = new CityTableAdapter();  CityDataSet.CityDataTable citys = adapter.GetCityByProID(pId);  //拼接json对象  StringBuilder sb = new StringBuilder();  sb.Append("[");  foreach (var item in citys)  {  sb.Append("{");  sb.AppendFormat("\"CityId\":\"{0}\",\"CityName\":\"{1}\"", item.CityID, item.CityName);  sb.Append("},");  }  string responseStr = sb.ToString().TrimEnd(',') + "]";  context.Response.Write(responseStr);  }  } |

### Select标签改变事件触发异步查询

|  |
| --- |
| <script src="Scripts/jquery-2.1.4.min.js"></script>  <script type="text/javascript">  $(function () {  initProvinceSelect();  bindCityByChangeEvt();  });  //下拉列表改变时加载市信息  function bindCityByChangeEvt() {  $("#province").change(function () {  //获取当前选中省的id  var pid = $(this).val();  //返回Json格式数据  $.getJSON("LoadCityByProID.ashx", { pId: pid }, function (data) {  $("#city").empty();  //遍历  for (var key in data) {  var cid = data[key].CityId;  var cname = data[key].CityName;  $("#city").append("<option value ='" + cid + "'>" + cname + "</option> ");  }  });  });  }  </script> |

### 首次请求时加载市信息

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title></title>  <meta charset="utf-8" />  <script src="Scripts/jquery-2.1.4.min.js"></script>  <script type="text/javascript">  $(function () {  initProvinceSelect();    });  //下拉列表改变时加载市信息  function bindCityByChangeEvt() {  $("#province").change(function () {  //获取当前选中省的id  var pid = $(this).val();  //返回Json格式数据  $.getJSON("LoadCityByProID.ashx", { pId: pid }, function (data) {  $("#city").empty();  //遍历  for (var key in data) {  var cid = data[key].CityId;  var cname = data[key].CityName;  $("#city").append("<option value ='" + cid + "'>" + cname + "</option> ");  }  });  });  }  //初始化省信息  function initProvinceSelect() {  $.get("LoadAllProvinceData.ashx", "", function (data) {  $("#province").html(data);  bindCityByChangeEvt();//省数据加载完后，再绑定市信息到select  $("#province").change();  });  }  </script>  </head>  <body>  <select id="province"></select>  <select id="city"></select>  </body>  </html> |

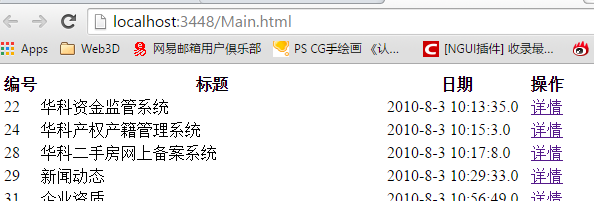
### 优化：序列号返回json对象

|  |
| --- |
| using \_04\_省市选择.CityDataSetTableAdapters;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Web;  namespace \_04\_省市选择  {  /// <summary>  /// Summary description for LoadCityByProID  /// </summary>  public class LoadCityByProID : IHttpHandler  {  public void ProcessRequest(HttpContext context)  {  //获得pid  int pId = int.Parse(context.Request["pId"] ?? "0");  //用强类型dataset查询数据  CityTableAdapter adapter = new CityTableAdapter();  CityDataSet.CityDataTable citys = adapter.GetCityByProID(pId);  List<CityData> listData = new List<CityData>();  foreach (var item in citys)  {  listData.Add(new CityData()  {  CityId = item.CityID,  CityName = item.CityName  });  }  //序列化为json对象  System.Web.Script.Serialization.JavaScriptSerializer jsSerializer = new System.Web.Script.Serialization.JavaScriptSerializer();  var responseStr = jsSerializer.Serialize(listData);  context.Response.Write(responseStr);  }  //内部类  public class CityData  {  public int CityId { get; set; }  public string CityName { get; set; }  }  public bool IsReusable  {  get  {  return false;  }  }  }  } |

# 异步增删改查

## 查询

实现类似下面的展示

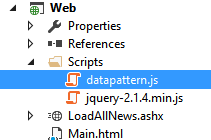


### 前台异步请求

### 设计html标签

|  |
| --- |
| body>  <table>  <thead>  <tr>  <th>编号</th>  <th>标题</th>  <th>日期</th>  <th>操作</th>  </tr>  </thead>  <tbody id="tbBody"></tbody>  </table>  </body> |

### 引入js文件



内容：

|  |
| --- |
| /\*\*  \* 对Date的扩展，将 Date 转化为指定格式的String  \* 月(M)、日(d)、12小时(h)、24小时(H)、分(m)、秒(s)、周(E)、季度(q) 可以用 1-2 个占位符  \* 年(y)可以用 1-4 个占位符，毫秒(S)只能用 1 个占位符(是 1-3 位的数字)  \* eg:  \* (new Date()).pattern("yyyy-MM-dd hh:mm:ss.S") ==> 2006-07-02 08:09:04.423  \* (new Date()).pattern("yyyy-MM-dd E HH:mm:ss") ==> 2009-03-10 二 20:09:04  \* (new Date()).pattern("yyyy-MM-dd EE hh:mm:ss") ==> 2009-03-10 周二 08:09:04  \* (new Date()).pattern("yyyy-MM-dd EEE hh:mm:ss") ==> 2009-03-10 星期二 08:09:04  \* (new Date()).pattern("yyyy-M-d h:m:s.S") ==> 2006-7-2 8:9:4.18  \*/  Date.prototype.pattern=function(fmt) {  var o = {  "M+" : this.getMonth()+1, //月份  "d+" : this.getDate(), //日  "h+" : this.getHours()%12 == 0 ? 12 : this.getHours()%12, //小时  "H+" : this.getHours(), //小时  "m+" : this.getMinutes(), //分  "s+" : this.getSeconds(), //秒  "q+" : Math.floor((this.getMonth()+3)/3), //季度  "S" : this.getMilliseconds() //毫秒  };  var week = {  "0" : "/u65e5",  "1" : "/u4e00",  "2" : "/u4e8c",  "3" : "/u4e09",  "4" : "/u56db",  "5" : "/u4e94",  "6" : "/u516d"  };  if(/(y+)/.test(fmt)){  fmt=fmt.replace(RegExp.$1, (this.getFullYear()+"").substr(4 - RegExp.$1.length));  }  if(/(E+)/.test(fmt)){  fmt=fmt.replace(RegExp.$1, ((RegExp.$1.length>1) ? (RegExp.$1.length>2 ? "/u661f/u671f" : "/u5468") : "")+week[this.getDay()+""]);  }  for(var k in o){  if(new RegExp("("+ k +")").test(fmt)){  fmt = fmt.replace(RegExp.$1, (RegExp.$1.length==1) ? (o[k]) : (("00"+ o[k]).substr((""+ o[k]).length)));  }  }  return fmt;  } |

### 实现异步请求

|  |
| --- |
| <script src="Scripts/jquery-2.1.4.min.js"></script>  <script src="Scripts/datapattern.js"></script>  <script type="text/javascript">  $(function () {  //初始化列表  initNewsList();  });  function initNewsList() {  $.getJSON("LoadAllNews.ashx", "", function (data) {  $("#tbBody").html("");  for (var key in data) {  //UTC时间格式化  var dateStr = (eval(data[key].Date.replace(/\/Date\((\d+)\)\//gi, "new Date($1)"))).pattern("yyyy-M-d h:m:s.S");  //拼接html  var str = "<tr>"  str += "<td>" + data[key].ID + "</td>";  str += "<td>" + data[key].title + "</td>";  str += "<td>" + dateStr + "</td>";  str += "<td>" + "<a href=#>详情</a>" + "</td>";  str += "</tr>";  $("#tbBody").append(str);  }  });  }  </script> |

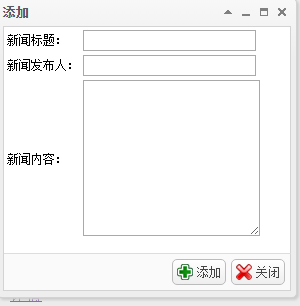
### 后台返回json对象

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  BLL.HKSJ\_Main mainService = new HKSJ\_Main();  List<Model.HKSJ\_Main> listModel = mainService.GetModelList("");  JavaScriptSerializer jsSerializer = new JavaScriptSerializer();  string jsonStr = jsSerializer.Serialize(listModel);  context.Response.Write(jsonStr);  } |

## 添加

功能：点击添加按钮，弹出添加对话框（easyUI实现），在对话框中添加内容，点击添加按钮实现添加。

添加对话框的样式：



### 添加“增加”按钮和DIV

在html的body中添加标签

|  |
| --- |
| <input type='button' value='增加' id='btnAdd' />  <hr />  <div id="addDiv">  <form id="AddForm">  <table>  <tr>  <td>新闻标题：</td>  <td><input type="text" name="txtTitle" id="txtTitle" /></td>  </tr>  <tr>  <td>新闻发布人：</td>  <td><input type="text" name="txtPeople" id="txtPeople" /></td>  </tr>  <tr>  <td>新闻内容：</td>  <td><textarea name="txtConetnt" id="txtContent" cols="22" rows="10"></textarea></td>  </tr>  </table>  </form>  </div> |

### 隐藏DIV

加载页面时隐藏DIV

|  |
| --- |
| <script type="text/javascript">  $(function () {  //初始化列表  initNewsList();  $("#addDiv").css("display", "none");//隐藏DIV  }); |

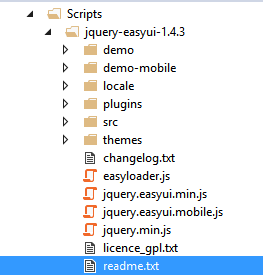
### 增加按钮注册事件

#### 点击增加显示DIV

|  |
| --- |
| function bindAddBtnClickEvent() {  $("#btnAdd").click(function () {  $("#addDiv").css("display", "block");    });  } |

#### EasyUI实现以对话框形式显示DIV

1. 添加easyUI插件



1. 在head中添加引用

|  |
| --- |
| <script src="Scripts/jquery-2.1.4.min.js"></script><!--1.引入jquery-->  <script src="Scripts/jquery-easyui-1.4.3/jquery.easyui.min.js"></script><!--2.引入easyui-->  <script src="Scripts/jquery-easyui-1.4.3/locale/easyui-lang-zh\_CN.js"></script><!--3.引入汉语-->  <link href="Scripts/jquery-easyui-1.4.3/themes/gray/easyui.css" rel="stylesheet" /><!--4.引入主题的样式-->  <link href="Scripts/jquery-easyui-1.4.3/themes/icon.css" rel="stylesheet" /><!--2.引入icon的样式--> |

1. 显示为对话框

|  |
| --- |
| function bindAddBtnClickEvent() {  $("#btnAdd").click(function () {  $("#addDiv").css("display", "block");  $("#addDiv").dialog({  title: "添加",  width: 300,  height: 300,  modal: true,  collapsible: true,  minimizable: true,  maximizable: true,  resizeable: true,  buttons: [{  text: "添加",  iconCls: "icon-add",  handler: function () {  alert("add");  }  }, {  text: "关闭",  iconCls: "icon-cancel",  handler: function () {  $("#addDiv").dialog("close");  }  }]  });  });  } |

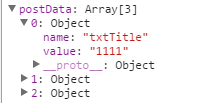
### 创建提交表单的方法

|  |
| --- |
| function submitAddForm() {  //收集表单中的数据  var postData = {  title: $("#txtTitle").val(),  people: $("#txtPeople").val(),  content: $("#txtContent").val(),  };  $.post("AddNews.ashx", postData, function (data) {  if (data == "ok") {  $("#addDiv").dialog("close");//关闭对话框  initNewsList();//刷新新闻列表  }  });  } |

在收集数据时，一个个表单手动获取效率比较差，jquery提供了一个serializeArray方法可以快速获取表单中的所有数据.

|  |
| --- |
| //收集表单中的数据  //var postData = {  // title: $("#txtTitle").val(),  // people: $("#txtPeople").val(),  // content: $("#txtContent").val(),  //};  //转换表单中的数据为json对象数组  var postData = $("#AddForm").serializeArray(); |

Jason对象的默认key的名字是input的标签的name。如下图：



### 点击对话框中的添加按钮，调用提交表单的方法

修改handler为方法名

|  |
| --- |
| buttons: [{  text: "添加",  iconCls: "icon-add",  handler: submitAddForm  } |

### 后台处理提交

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  Model.HKSJ\_Main news = new Model.HKSJ\_Main();  news.title = context.Request.Form["title"];  news.people = context.Request.Form["people"];  news.content = context.Request.Form["content"];  news.picUrl = string.Empty;  news.Date = DateTime.Now;  news.type = string.Empty;  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  mainService.Add(news);  context.Response.Write("ok");  } |

## 详情

### 修改详情标签

修改initNewsList下详情超链接：

|  |
| --- |
| function initNewsList() {  $.getJSON("LoadAllNews.ashx", "", function (data) {  $("#tbBody").html("");  for (var key in data) {  var dateStr = (eval(data[key].Date.replace(/\/Date\((\d+)\)\//gi, "new Date($1)"))).pattern("yyyy-M-d h:m:s.S");  var str = "<tr>"  str += "<td>" + data[key].ID + "</td>";  str += "<td>" + data[key].title + "</td>";  str += "<td>" + dateStr + "</td>";  //详情链接  str += "<td>" + "<a newsId='" + data[key].ID + "' href='javascipt:void(0)'>详情</a>" + "</td>";  str += "</tr>"  $("#tbBody").append(str);  }  bindDetailLinkClickEvent();//详情超链接的点击事件  });  } |

### 详情超链接的点击事件

|  |
| --- |
| function bindDetailLinkClickEvent() {  $("#tbBody a:contains('详情')").click(function () {  var newsId = $(this).attr("newsId");  alert(newsId);  });  } |

### 添加Div，并隐藏

#### 添加div

Html的Body中添加

|  |
| --- |
| <div id="detailDiv">  <h1>详情信息</h1>  <hr />  <table>  <tr>  <td>新闻标题：</td>  <td><span id="spTitle"></span></td>  </tr>  <tr>  <td>新闻内容：</td>  <td><span id="spContent"></span></td>  </tr>  </table>  </div> |

#### Jquery中初始化隐藏div

|  |
| --- |
| $("#detailDiv").css("display", "none"); |

### 点击超链接弹出对话框

|  |
| --- |
| function bindDetailLinkClickEvent() {  $("#tbBody a:contains('详情')").click(function () {  var newsId = $(this).attr("newsId");  // alert(newsId);  $("#detailDiv").css("display", "block");  $("#detailDiv").dialog({  title: "详情",  width: 300,  height: 300,  modal: true,  collapsible: true,  minimizable: true,  maximizable: true,  resizeable: true,  buttons: [{  text: "关闭",  iconCls: "icon-cancel",  handler: function () {  $("#detailDiv").dialog("close");  }  }]  });  });  } |

### 通过异步请求获取数据并赋值

|  |
| --- |
| function getNewsToDetail(newsId)  {  $.getJSON("getNewsById.ashx", { id: newsId }, function (data) {  $("#spTitle").text(data.title);  $("#spContent").html(data.content);  });  } |

点击详情按钮调用2.3.5的方法

|  |
| --- |
| function bindDetailLinkClickEvent() {  $("#tbBody a:contains('详情')").click(function () {  var newsId = $(this).attr("newsId");  getNewsToDetail(newsId); |

### 后台从数据库获取数据并返回json对象

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  int id = int.Parse(context.Request["id"] ?? "0");  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  Model.HKSJ\_Main model = mainService.GetModel(id);  System.Web.Script.Serialization.JavaScriptSerializer jsSerializer = new System.Web.Script.Serialization.JavaScriptSerializer();  string strJson = jsSerializer.Serialize(model);  context.Response.Write(strJson);  } |

## 修改

### 添加修改超链接

|  |
| --- |
| //详情链接  str += "<td>" + "<a newsId='" + data[key].ID + "' href='javascipt:void(0)'>详情</a>&nbsp;&nbsp;";  //修改超链接  str += "<a class='editLink' newsId='" + data[key].ID + "' href='javascipt:void(0)'>修改</a>" + "</td>";  str += "</tr>" |

### 添加DIv并隐藏

|  |
| --- |
| <div id="EditDiv">  <form id="ediFrm">  <input type="hidden" name="hiddenId" id="hiddenId" />  <table>  <tr>  <td>标题：</td>  <td><input type="text" id="txttitle" name="txttitle"/></td>  </tr>  <tr>  <td>内容：</td>  <td><input type="text" id="spcontent" name="spcontent"/></td>  </tr>  </table>  </form>  </div> |

隐藏

|  |
| --- |
| $("#EditDiv").css("display", "none"); |

### 添加点击事件的方法

|  |
| --- |
| function bindEditLinkClickEvent() {  $(".editLink").click(function () {  var newsId = $(this).attr("newsId");  getNewsToEditDIv(newsId);  $("#EditDiv").css("display", "block");  $("#EditDiv").dialog({  title: "修改",  width: 300,  height: 300,  modal: true,  collapsible: true,  minimizable: true,  maximizable: true,  resizeable: true,  buttons: [{  text: "修改",  iconCls: "icon-edit",  handler: submitEditFrm  }  , {  text: "关闭",  iconCls: "icon-cancel",  handler: function () {  $("#EditDiv").dialog("close");  }  }]  });    }); |

getNewsToEditDIv方法查询并写入标签中数据

|  |
| --- |
| function getNewsToEditDIv(newsId) {  $.getJSON("getNewsById.ashx", { id: newsId }, function (data) {  $("#txttitle").val(data.title);  $("#hiddenId").val(data.ID);  $("#spcontent").val(data.content);  });  } |

### 点击修改按钮调用submitEditFrm方法

|  |
| --- |
| function submitEditFrm() {  var postData = $("#ediFrm").serializeArray();  $.post("EditNews.ashx", postData, function (data) {  if(data=="ok")  {  $("#EditDiv").dialog("close");  }else  {  $.messager.alert("警告", "数据插入失败", 'error');  }  initNewsList();//初始化  });  } |

### 后台处理数据

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  int id = int.Parse(context.Request["hiddenId"]);  string title = context.Request["txttitle"] ?? string.Empty;  string content = context.Request["spcontent"] ?? string.Empty;  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  Model.HKSJ\_Main model = mainService.GetModel(id);  model.title = title;  model.content = content;  if (mainService.Update(model))  {  context.Response.Write("ok");  }  else  {  context.Response.Write("eroor");  }  } |

## 优化：Iframe

详情、添加、修改都需要创建div，代码变得复杂。可以用iframe来优化

设计知识点：

主窗体调用子网页中的方法

子网页调用住网页中的方法

现在我们对2.4节内容进行修改

### 修改div

在div中添加一个iframe，这样就不需要再div中写其他标签了

|  |
| --- |
| <div id="EditDiv">  <iframe src="javascript:void(0)" id="editIframe" width="100%" height="100%" frameborder="0"></iframe>  </div> |

### 弹出对话框前，设置iframe的src属性



### 创建EditNews.aspx并编辑

#### 在后置代码中接收数据

|  |
| --- |
| public Model.HKSJ\_Main News { get; set; }  protected void Page\_Load(object sender, EventArgs e)  {  if (!IsPostBack)  {  int id = int.Parse(Request.QueryString["id"] ?? "0");  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  Model.HKSJ\_Main model = mainService.GetModel(id);  News = model;  }  } |

#### 在html中展示数据

|  |
| --- |
| <body>  <form id="form1" runat="server">  <div>  <input type="hidden" value="<%=News.ID %>" name="hiddenId" />  <table>  <tr>  <td>新闻标题：</td>  <td>  <input type="text" name="txtTitle" value="<%=News.title %>" />  </td>  </tr>  <tr>  <td>新闻内容：</td>  <td>  <textarea name="areaContent" style="margin: 0px; width: 162px; height: 150px;"><%=News.content%></textarea>  </td>  </tr>  </table>  </div>  </form>  </body> |

### 点击对话框中的修改按钮提交iframe的网页中的内容

点击修改按钮，提交

|  |
| --- |
| buttons: [{  text: "修改",  iconCls: "icon-edit",  handler: function () {  //拿到子容器的window对象  var domFrame = $("#editIframe")[0];  //调用子容器中的方法  domFrame.contentWindow.submitFrm();  }  } |

### iframe中EditNews.aspx提交表单的方法

|  |
| --- |
| <script src="Scripts/jquery-2.1.4.min.js"></script>  <script type="text/javascript">  //提交表单  function submitFrm() {  var postData = $("#form1").serializeArray();  //提交到本页  $.post("EdiNews.aspx", postData, function (data) {  if (data == "ok") {  //提交成功后，主窗体执行afterEditSuccsess  window.parent.window.afterEditSuccsess();  }  });  }  </script> |

### 后置代码处理提交

|  |
| --- |
| int id = int.Parse(Request["hiddenId"]);  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  Model.HKSJ\_Main model = mainService.GetModel(id);  model.title = Request["txtTitle"];  model.content = Request["areaContent"];  if (mainService.Update(model))  {  Response.Clear();  Response.Write("ok");  Response.End();  } |

### 提交成功，调用主窗体中的方法

|  |
| --- |
| if (data == "ok") {  //提交成功后，主窗体执行afterEditSuccsess  window.parent.window.afterEditSuccsess();  } |

Main.html中afterEditSuccsess方法

|  |
| --- |
| function afterEditSuccsess()  {  $("#EditDiv").dialog("close");  initNewsList();  } |

## 删除

### 添加超链接

|  |
| --- |
| str += "<a class='deleteLink' newsId='" + data[key].ID + "' href='javascipt:void(0)'>删除</a>" + "</td>"; |

### 绑定删除超链接的事件

|  |
| --- |
| function bindDeleteLinkClickEvent() {  $(".deleteLink").click(function () {  var newsId = $(this).attr("newsId");  $.messager.confirm("提醒", "确定要删除吗?", function (r) {  if (r) {    $.post("deleteNews.ashx", { id: newsId }, function (data) {  if(data=="ok")  {  initNewsList();  }else  {  alert("删除失败");  }  });  }  });  });  } |

### 初始化列表后调用删除超链接的事件



### 后台处理

|  |
| --- |
| int id = int.Parse(context.Request["id"]);  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  if (mainService.Delete(id))  {  context.Response.Write("ok");  } |

## 分页

### 序列化实体对象和html

修改LoadAllNews（见2.1）中获取新闻时，还获取分页超链接信息

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  BLL.HKSJ\_Main mainService = new HKSJ\_Main();  int pageSize = int.Parse(context.Request["pageSize"]??"5");  int pageIndex = int.Parse(context.Request["pageIndex"] ?? "1");  int total = 0;  //把当前页数据拿到  List<Model.HKSJ\_Main> listModel = mainService.LoadPageData(pageIndex, pageSize, out total);  //返回分页的超级链接  string navHtml = Common.LaomaPager.ShowPageNavigate(pageSize,pageIndex,total);  //匿名方法：把实体对象和html标签封装到一个匿名类中  var data = new {NewsList = listModel,NavHtml= navHtml };  JavaScriptSerializer jsSerializer = new JavaScriptSerializer();  //序列化为json数组对象  string jsonStr = jsSerializer.Serialize(data);  context.Response.Write(jsonStr);  } |

### 网页加载时初始化新闻列表

新闻实体在data.NewsList对象数组中

下面代码位于initNewsList方法中。

|  |
| --- |
| $.getJSON("LoadAllNews.ashx", "", function (data) {  $("#tbBody").html("");  for (var key in data.NewsList) {  var dateStr = (eval(data.NewsList[key].Date.replace(/\/Date\((\d+)\)\//gi, "new Date($1)"))).pattern("yyyy-M-d h:m:s.S");  var str = "<tr>"  str += "<td>" + data.NewsList[key].ID + "</td>";  str += "<td>" + data.NewsList[key].title + "</td>";  str += "<td>" + dateStr + "</td>";  //详情链接  str += "<td>" + "<a newsId='" + data.NewsList[key].ID + "' href='javascipt:void(0)'>详情</a>&nbsp;&nbsp;";  //修改超链接  str += "<a class='editLink' newsId='" + data.NewsList[key].ID + "' href='javascipt:void(0)'>修改</a>" + "&nbsp;&nbsp;";  str += "<a class='deleteLink' newsId='" + data.NewsList[key].ID + "' href='javascipt:void(0)'>删除</a>" + "</td>";  str += "</tr>"  $("#tbBody").append(str);  } |

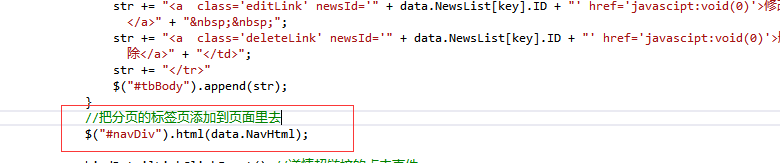
## 把分页的标签页添加到页面

### 添加div

在body中新闻列表的table下添加

|  |
| --- |
| <div id="navDiv">  </div> |

在初始化新闻列表完成后，把分页标签添加到页面



## 注册分页超链接的点击事件

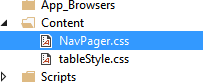
|  |
| --- |
| function bindNavLinkClickEvent() {  $(".pageLink").click(function () {  //获取href属性的值  var href = $(this).attr("href");  //去掉最开头的？  var strpostData = href.substr(href.lastIndexOf('?') + 1);  //根据传入的页码加载新闻  initNewsList(strpostData);  return false;  });  } |

修改initNewsList

|  |
| --- |
|  |

## 引用样式

引用NavPager样式



为分页div添加class

|  |
| --- |
| <div id="navDiv" class="paginator"> |

最后效果：



# Easyui实现crud

使用easyui的datagrid实现crud

## 新闻列表

### 创建table容器

在body中添加

|  |
| --- |
| <div>  <table id="tt" style="width: 700px;" title="标题，可以使用代码进行初始化，也可以使用这种属性的方式" iconcls="icon-edit"></table>  </div> |

### 引入样式与js文件

|  |
| --- |
| <meta charset="utf-8" />  <link href="../Scripts/jquery-easyui-1.4.3/themes/metro/easyui.css" rel="stylesheet" />  <link href="../Scripts/jquery-easyui-1.4.3/themes/icon.css" rel="stylesheet" />  <script src="../Scripts/jquery-2.1.4.min.js"></script>  <script src="../Scripts/jquery-easyui-1.4.3/jquery.easyui.min.js"></script>  <script src="../Scripts/jquery-easyui-1.4.3/locale/easyui-lang-zh\_CN.js"></script>  <script src="../Scripts/datapattern.js"></script> |

### 返回数据

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  //自动往后台发送：rows，page.要求返回的数据：{total:200,rows:[{},{}]}  int pageSize = int.Parse(context.Request["rows"] ?? "5");  int pageIndex = int.Parse(context.Request["page"] ?? "1");  int total = 0;  BLL.HKSJ\_Main mainService = new BLL.HKSJ\_Main();  List<Model.HKSJ\_Main> newsList = mainService.LoadPageData(pageIndex, pageSize, out total);  var data = new { total = total, rows = newsList };  System.Web.Script.Serialization.JavaScriptSerializer jsSerializer = new System.Web.Script.Serialization.JavaScriptSerializer();  string strJson = jsSerializer.Serialize(data);  context.Response.Write(strJson);  } |

### 创建datagrid并绑定数据

|  |
| --- |
| <script type="text/javascript">  $(function () {  initTable();  });  function initTable() {  $('#tt').datagrid({  url: 'LoadNewsForEasyUiTable.ashx',//表格加载完成后，自动请求此地址。自动往后台发送：rows，page.要求返回的数据：{total:200,rows:[{},{}]}  title: '新闻列表',  width: 700,  height: 400,  fitColumns: true,  idField: 'ID',//后台返还数据的主键列，一定注意大小写  loadMsg: '正在加载用户的信息...',  pagination: true,//启用分页  singleSelect: true,  pageSize: 10,  pageNumber: 1,//默认当前页  pageList: [10, 20, 30],//每页多少条  queryParams: {},//异步请求，额外传送的数据  columns: [[  { field: 'ck', checkbox: true, align: 'left', width: 50 },  { field: 'ID', title: '编号', width: 80 },  { field: 'title', title: '新闻标题', width: 120 },  {  field: 'Date', title: '提交时间', width: 80, align: 'right',  formatter: function (value, row, index) {  return (eval(value.replace(/\/Date\((\d+)\)\//gi, "new Date($1)"))).pattern("yyyy-M-d h:m:s.S");  }  }  ]],  toolbar: [{  id: 'btnDownShelf',  text: '添加新闻',  iconCls: 'icon-add',  handler: function () {  alert("add");  }  }],  onHeaderContextMenu: function (e, field) {  }  });  }  </script> |

效果：



## 修改

### 在toolbar中添加修改新闻按钮

|  |
| --- |
| toolbar: [{  id: 'btnDownShelf',  text: '添加新闻',  iconCls: 'icon-add',  handler: function () {  alert("add");  }  }, {  id: 'btnDownShelf',  text: '修改新闻',  iconCls: 'icon-edit',  handler: function () {  editNewsBtnClickEvent();  }  }], |

### 点击事件

|  |
| --- |
| function editNewsBtnClickEvent() {  //获取选中的行  var rows = $("#tt").datagrid("getSelections");  if (rows.length != 1) {  $.messager.alert("提醒", "一次只能修改一行", "warning");  return;  }  //获取选中行的id属性  var id = rows[0].ID;  //加载信息到对话框  loadNewsToDialog(id);  //显示对话框  showDialog("#editDiv", "修改新闻", "编辑", editDialogEvent, "icon-edit");  } |

### 加载信息到对话框

|  |
| --- |
| //加载信息信息到对话框中  function loadNewsToDialog(id) {  $.getJSON("GetNewsById.ashx", { id: id }, function (data) {  $("#txtTitle").val(data.title);  $("#spContent").html(data.content);  $("#hiddenId").val(data.ID);  });  } |

### 显示对话框

|  |
| --- |
| //显示对话框  function showDialog(vId, vTitle, vText, vhandler, vIcon) {  $(vId).css("display", "block");  $(vId).dialog({  title: vTitle,  width: 400,  height: 400,  modal: true,  collapsible: true,  minimizable: true,  maximizable: true,  resizeable: true,  buttons: [{  text: vText,  iconCls: vIcon,  handler: vhandler  }  , {  text: "关闭",  iconCls: "icon-cancel",  handler: function () {  $(vId).dialog("close");  }  }]  });  } |

### 点击对话框中编辑按钮事件（editDialogEvent）

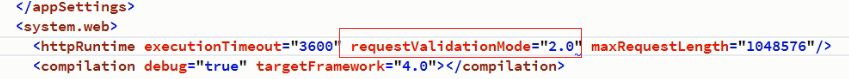
|  |
| --- |
| //编辑按钮事件  function editDialogEvent() {  var postData = $("#editFrm").serializeArray();  $.post("EditNews.ashx", postData, function (data) {  if (data == "ok") {  $("#editDiv").dialog("close");  $('#tt').datagrid("reload");//刷新  }  });  } |

## 总结

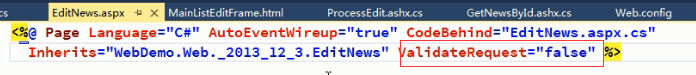
|  |
| --- |
| JQuery EasyUI的datagrid的使用方式总结  第一步：添加样式和js脚本在前台添加展示数据表格的table元素  例如：  <div>  <table id="tt" style="width: 700px;" title="标题，可以使用代码进行初始化，也可以使用这种属性的方式" iconcls="icon-edit">  </table>  </div>  注：表格的属性可以在table中设置（Unobtrusive），也可以直接使用js脚本进行控制。建议使用js脚本控制      属性的定义：  请参见Jquery easyui API    第二步：在doucment.ready中初始化表格的属性以及数据获取的方式。  例如：  $('#tt').datagrid({  url: '/UserInfo/GetAllUserInfos',  title: '演示表格使用',  width: 700,  height: 400,  fitColumns: true,  idField: 'ID',  loadMsg: '正在加载用户的信息...',  pagination: true,  singleSelect: false,  pageSize:10,  pageNumber:1,  pageList: [10, 20, 30],  queryParams: {},  columns: [[  { field: 'ck', checkbox: true, align: 'left', width: 50 },  { field: 'ID', title: '主键', width: 80 },  { field: 'UserName', title: '用户名', width: 120 },  { field: 'SubTime', title: '提交时间', width: 80, align: 'right',  formatter:function(value,row,index){  return (eval(value.replace(/\/Date\((\d+)\)\//gi, "new Date($1)"))).pattern("yyyy-M-d h:m:s.S");  }  },  {field:'showprice',title:'商品价格',width:80,align:'right',  styler:function(value,row,index){  if (value < 20){  return 'background-color:#ffee00;color:red;';  }  },  formatter:function(value,row,index){  return "<a href='#' onclick='editGoodsPrice("+row.goodsid+");return false;'>"+value+"</a>";  }  }  ]],  toolbar: [{  id: 'btnDownShelf',  text: '上架',  iconCls: 'icon-add',  handler: function () {  var rows = $('#goodGrid').datagrid('getSelections');  if (!rows || rows.length == 0) {  //alert("请选择要修改的商品！");  $.messager.alert("选择商品提醒", "请选择要修改的商品!", "error");  return;  }  $.messager.confirm("上架提醒", "您是否要真的要将此商品上架？", function (r) {  if (r) {  updateGoodsNewHot(rows, "onshelf");  }  });  }  }],  onHeaderContextMenu: function (e, field) {  }  });      第三步：后台设置加载的数据：  注意：表格Post或者get回来的请求中  page：3 代表page为key，然后选择的当前页码为3  rows：10 代表一页的大小为10  后台返回的数据的格式为：{total：''，rows：[{},{}]}  只要包含了总数tatol字段，rows是具体的行数  例如：  Asp.Net MVC 例子：  public JsonResult GetAllUserInfos()  {  int pageSize = 5;  int pageIndex = 1;  int.TryParse(this.Request["page"], out pageIndex);  int.TryParse(this.Request["rows"], out pageSize);  pageSize = pageSize <= 0 ? 5 : pageSize;  pageIndex = pageIndex < 1 ? 1 : pageIndex;  var temp = db.UserInfo  .OrderBy(u=>u.Sort)  .Skip<UserInfo>((pageIndex-1)\*pageSize)  .Take<UserInfo>(pageSize)  .ToList<UserInfo>();  Hashtable ht = new Hashtable();  ht["total"] = db.UserInfo.Count();  ht["rows"] = temp;  return Json(ht);  }    Asp.Net WebForm 例子：  public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  var strWebName = context.Request["WebName"] ?? string.Empty;  var GoodsNo = context.Request["GoodsNo"] ?? string.Empty;  int categoryId = 0;  int pageIndex = 1;  int pageSize = 10;  int.TryParse(context.Request["rows"], out pageSize);  int.TryParse(context.Request["page"], out pageIndex);  decimal priceLeft = 0;  decimal priceRight = 1000000;  int goodsStatus = 0;  decimal.TryParse(context.Request["PriceLeft"], out priceLeft);  decimal.TryParse(context.Request["PriceRight"], out priceRight);  int.TryParse(context.Request["CategoryId"], out categoryId);  int.TryParse(context.Request["GoodsStatus"], out goodsStatus);  var goodsQueryParamter = new GoodsQueryParamter();      goodsQueryParamter.GoodsStatus = (Model.GoodsModel.GoodsStatusEnum)goodsStatus;  var ds = goodsService.GetGoodsList(goodsQueryParamter);  string json = string.Empty;  if (ds != null && ds.Tables.Count > 0)  {  System.Text.StringBuilder rowJson = new System.Text.StringBuilder();  int colLen = ds.Tables[0].Columns.Count;  DataColumnCollection col = ds.Tables[0].Columns;  foreach (DataRow row in ds.Tables[0].Rows)  {  System.Text.StringBuilder colJson = new System.Text.StringBuilder();  rowJson.Append("{");  for (int i = 0; i < colLen; i++)  {  colJson.Append("\"" + col[i].ColumnName + "\":\"" + row[i].ToString() + "\",");  }  rowJson.Append(colJson.ToString().TrimEnd(','));  rowJson.Append("},");  }  json = "{\"total\":" + ds.Tables[0].Rows[0]["sumGoods"] + ",\"rows\":[" + rowJson.ToString().TrimEnd(',') + "]}";  }  context.Response.Write(json);  } |

# 补充：跨站脚本攻击

在web.config中



在页面中



# 富文本编辑器

## 创建form

Textarea用于富文本编辑器，submit用于提交数据

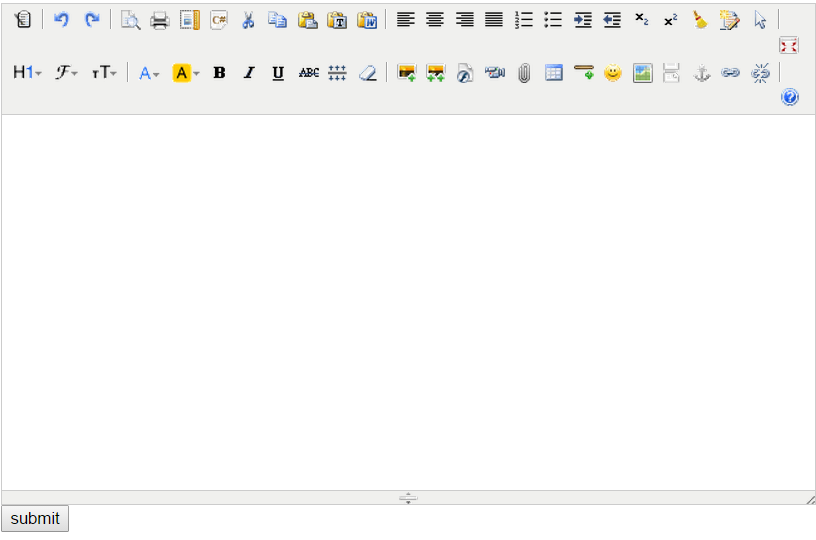
|  |
| --- |
| <form action="edit.aspx" method="post">  <textarea name="content" style="width: 500px; height: 400px; display: none;"></textarea>  <input type="submit" value="submit" />  </form> |

## 使用Kindeditor

引入js脚本

|  |
| --- |
| <link href="kindeditor-4.1.10/themes/qq/qq.css" rel="stylesheet" />  <script src="kindeditor-4.1.10/kindeditor-min.js"></script>  <script src="kindeditor-4.1.10/lang/zh\_CN.js"></script>  <script>  var editor;  KindEditor.ready(function (K) {  editor = K.create('textarea[name="content"]', {  allowFileManager: true  });  });  </script> |

效果：



## 获取提交的内容

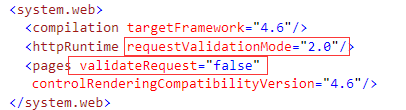
|  |
| --- |
| public partial class edit : System.Web.UI.Page  {  public string Content { get; set; }  protected void Page\_Load(object sender, EventArgs e)  {  string content = Request.Form["content"];  Content = content;  }  } |

## 展示提交的内容

|  |
| --- |
| <body>  <form id="form1" runat="server">  <div>  <%=Content %>  </div>  </form>  </body> |

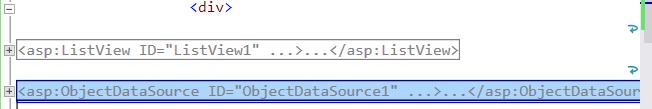
## 关闭错误提示

提交的html文本，默认会被屏蔽，需要在web.config中进行调试

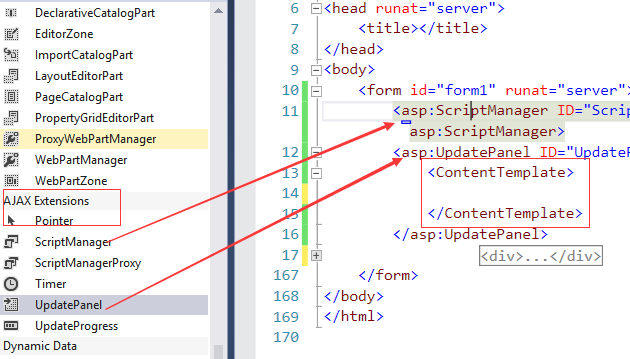


# Ajax组件

## 添加listview并绑定数据



## 添加ajax扩展组件



ScriptManager只能有1个而updatePanel可以有多个

## 把listview所在的div放到contentTemplate中

