# 校验验证码

## 验证码生成一般处理程序

|  |
| --- |
| <%@ WebHandler Language="C#" Class="ValidateCode" %>  using System;  using System.Web;  using System.Drawing;  using System.Web.SessionState;  public class ValidateCode : IHttpHandler, IRequiresSessionState  {  HttpContext context;  public void ProcessRequest (HttpContext context1) {  this.context = context1;  CreateCheckCodeImage(GenerateCheckCode());  }  private string GenerateCheckCode()  {  int number;  char code;  string checkCode = String.Empty;  System.Random random = new Random();  for (int i = 0; i < 5; i++)  {  number = random.Next();  if (number % 2 == 0)  code = (char)('0' + (char)(number % 10));  else  code = (char)('0' + (char)(number % 10));  //code = (char)('A' + (char)(number % 26));  checkCode += code.ToString();  }  //Response.Cookies.Add(new HttpCookie("CheckCode", checkCode));  context.Session.Add("vCode", checkCode);  return checkCode;  }  private void CreateCheckCodeImage(string checkCode)  {  if (checkCode == null || checkCode.Trim() == String.Empty)  return;  System.Drawing.Bitmap image = new System.Drawing.Bitmap((int)Math.Ceiling((checkCode.Length \* 12.5)), 22);  Graphics g = Graphics.FromImage(image);  try  {  //生成随机生成器  Random random = new Random();  //清空图片背景色  g.Clear(Color.White);  //画图片的背景噪音线  for (int i = 0; i < 25; i++)  {  int x1 = random.Next(image.Width);  int x2 = random.Next(image.Width);  int y1 = random.Next(image.Height);  int y2 = random.Next(image.Height);  g.DrawLine(new Pen(Color.Silver), x1, y1, x2, y2);  }  Font font = new System.Drawing.Font("Arial", 12, (System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Italic));  System.Drawing.Drawing2D.LinearGradientBrush brush = new System.Drawing.Drawing2D.LinearGradientBrush(new Rectangle(0, 0, image.Width, image.Height), Color.Blue, Color.DarkRed, 1.2f, true);  g.DrawString(checkCode, font, brush, 2, 2);  //画图片的前景噪音点  for (int i = 0; i < 100; i++)  {  int x = random.Next(image.Width);  int y = random.Next(image.Height);  image.SetPixel(x, y, Color.FromArgb(random.Next()));  }  //画图片的边框线  g.DrawRectangle(new Pen(Color.Silver), 0, 0, image.Width - 1, image.Height - 1);  System.IO.MemoryStream ms = new System.IO.MemoryStream();  image.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);  context.Response.ClearContent();  context.Response.ContentType = "image/Gif";  context.Response.BinaryWrite(ms.ToArray());  }  finally  {  g.Dispose();  image.Dispose();  }  }    public bool IsReusable {  get {  return false;  }  }  } |

## 嵌入网页

|  |
| --- |
| <img src="/ashx/ValidateCode.ashx" /> |

## 校验

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  if (IsPostBack)//\_\_VIEWSTATE  {  if (CheckCode())  {  //服务端校验.  }  }  //判断验证码是否匹配  protected bool CheckCode()  {  if (Session["vCode"] != null)  {  string sysCode = Session["vCode"].ToString();  string txtCode = Request.Form["txtCode"];  //匹配时忽略大小写  if (sysCode.Equals(txtCode, StringComparison.InvariantCultureIgnoreCase))  {//匹配成功设置session为空，防止暴力破解  Session["vCode"] = null;  return true;  }  else  {  return false;  }  }  else  {  return false;  }  }  } |

## Session判断问题

如果每一个页面都要判断session是否为空，造成代码重复。有2种优化方法：

### 父类

在common层中，创建一个继承System.Web.UI.Page的类，注册事件，进行session判断

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  namespace BookShop.Web.Common  {  public class CheckSession:System.Web.UI.Page  {  protected override void OnPreInit(EventArgs e)  {  if(Session["Vcode"] ==null)  {  Response.Redirect("Login.aspx");  }  }  }  } |

其它需要用到session的web form都继承自创建的父类

|  |
| --- |
| public partial class WebForm1 : CheckSession  {  protected void Page\_Load(object sender, EventArgs e)  {  }  } |

如果Session["Vcode"]为空，则直接跳转到默认页。

### 使用IHttpModule接口

在common中，创建继承自IHttpModule的类，并注册事件

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  namespace BookShop.Web.Common  {  public class CheckModule : IHttpModule  {  public void Dispose()  {  throw new NotImplementedException();  }  public void Init(HttpApplication context)  {  //注册checkSession事件  context.AcquireRequestState += new EventHandler(checkSession);  }  private void checkSession(object sender, EventArgs e)  {  HttpApplication app = sender as HttpApplication;  HttpContext context = app.Context;  if (HttpContext.Current.Session["Vcode"] == null)  {  context.Response.Redirect("index.aspx");  }  }  }  } |

配置web.config

|  |
| --- |
| <system.webServer>  <modules>  <add name="CheckModule" type="BookShop.Web.CheckModule"/>  </modules>  </system.webServer> |

# 注册

验证码正确后，进行注册

## 在三层中添加注册方法

### DAL层

新建partial类型，根据用户名查询

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Maticsoft.DBUtility;  using System.Data;  using System.Data.SqlClient;  namespace BookShop.DAL  {  public partial class UserServices  {  public bool CheckUserName(string userName)  {  string sql = "select count(1) from Users where LoginId=@LoginId";  object obj=DbHelperSQL.GetSingle(sql, new SqlParameter("@LoginId",userName));  return Convert.ToInt32(obj) > 0;  }  }  } |

### Bll层

新建partial类，在其中判断用户是否存在。

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace BookShop.BLL  {  public partial class UserManager  {  /// <summary>  /// 增加一条数据  /// </summary>  public int Add(BookShop.Model.User model,out string msg)  {  //判断用户名是否存在.  if (CheckUserName(model.LoginId))  {  msg = "此用户已经存在";  return -1;  }  else  {  msg = "注册成功";  return dal.Add(model);  }  }  public bool CheckUserName(string userName)  {  return dal.CheckUserName(userName);  }  }  } |

### 验证码成功后，向数据库写入数据并跳转

|  |
| --- |
| if (CheckCode())  {  //服务端校验.  Model.User model = new Model.User();  model.LoginId = Request.Form["txtUserName"];  model.LoginPwd = Request.Form["txtUserPass"];  model.Mail = Request.Form["txtUserEmail"];  model.Name = Request.Form["txtRealName"];  model.Phone = Request.Form["txtUserPhone"];  model.Address = Request.Form["txtUserAddress"];  model.UserState.Id = 1;  BLL.UserManager bll = new BLL.UserManager();  string msg = string.Empty;  int i = bll.Add(model, out msg); //bll中判断用户名是否存在  if (i > 0)  {  //发送激活链接.    //跳转.  Response.Redirect("/ShowMsg.aspx?msg=" + Server.UrlEncode(msg) + "&txt=" + Server.UrlEncode("首页") + "&url=/Default.aspx");  }  } |

# 激活链接

|  |
| --- |
| MailMessage mailMsg = new MailMessage();//两个类，别混了，要引入System.Net这个Assembly  mailMsg.From = new MailAddress("wang\_itcast@126.com", "王承伟");//源邮件地址  mailMsg.To.Add(new MailAddress("526711443@qq.com", "王承伟"));//目的邮件地址。可以有多个收件人  mailMsg.Subject = "Hello,大家好!";//发送邮件的标题  mailMsg.Body = "Tai Xie E le！";//发送邮件的内容替换为下面的href  SmtpClient client = new SmtpClient("smtp.126.com");// 发件人所使用邮箱的SMTP服务器地址。 smtp.163.com，smtp.qq.com  client.Credentials = new NetworkCredential("wang\_itcast", "wangchengwei");//发件人邮箱的用户名和密码.  client.Send(mailMsg); |
| <a href="http://localhost/ashx/Active.ashx?userId=26&activeCode=sdfkjs66sd7f">请单击激活账号</a> |

# 无刷新上传图像

创建web form，使用swfupload插件

## 引入js脚本

|  |
| --- |
| <script src="SWFUpload/swfupload.js"></script>  <script src="SWFUpload/handlers.js"></script> |

## 创建html容器

|  |
| --- |
| <form id="form1" runat="server">  <div id="content">  <div id="swfu\_container" style="margin: 0px 10px;">  <div>  <span id="spanButtonPlaceholder"></span>  </div>  <div id="divFileProgressContainer" style="height: 75px;"></div>  <div id="thumbnails"></div>  </div>  </div>  </form> |

## 创建脚本

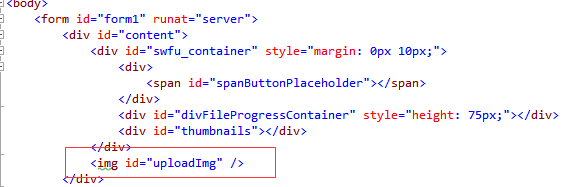
|  |
| --- |
| <script type="text/javascript">  var swfu;  window.onload = function () {  swfu = new SWFUpload({  // Backend Settings  //提交到的处理程序  upload\_url: "upload.ashx",  post\_params: {  "ASPSESSID": "<%=Session.SessionID %>"  },  // File Upload Settings  file\_size\_limit: "2 MB",  file\_types: "\*.jpg",  file\_types\_description: "JPG Images",  file\_upload\_limit: 0, // Zero means unlimited  // Event Handler Settings - these functions as defined in Handlers.js  // The handlers are not part of SWFUpload but are part of my website and control how  // my website reacts to the SWFUpload events.  swfupload\_preload\_handler: preLoad,  swfupload\_load\_failed\_handler: loadFailed,  file\_queue\_error\_handler: fileQueueError,  file\_dialog\_complete\_handler: fileDialogComplete,  upload\_progress\_handler: uploadProgress,  upload\_error\_handler: uploadError,  upload\_success\_handler: uploadSuccess,  //upload\_complete\_handler: uploadComplete,  // Button settings  button\_image\_url: "SWFUpload/images/XPButtonNoText\_160x22.png",  button\_placeholder\_id: "spanButtonPlaceholder",  button\_width: 160,  button\_height: 22,  //按钮上显示的文字  button\_text: '<span class="button">选择图片<span class="buttonSmall">(2 MB Max)</span></span>',  button\_text\_style: '.button { font-family: Helvetica, Arial, sans-serif; font-size: 14pt; } .buttonSmall { font-size: 10pt; }',  button\_text\_top\_padding: 1,  button\_text\_left\_padding: 5,  // Flash Settings  flash\_url: "SWFUpload/swfupload.swf", // Relative to this file  flash9\_url: "SWFUpload/swfupload\_FP9.swf", // Relative to this file  custom\_settings: {  upload\_target: "divFileProgressContainer"  },  // Debug Settings  debug: false  });    }  </script> |

## 一般处理程序

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Web;  namespace BookShop.Web  {  /// <summary>  /// Summary description for upload  /// </summary>  public class upload : IHttpHandler  {  public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  //接收文件  HttpPostedFile jpeg\_image\_upload = context.Request.Files["Filedata"];  string fileName = Path.GetFileName(jpeg\_image\_upload.FileName);  string fileExt = Path.GetExtension(fileName);  if(fileExt==".jpg")  {  jpeg\_image\_upload.SaveAs(context.Request.MapPath("Upload/Image/" + fileName));  //返回图片地址  context.Response.Write("Upload/Image/" + fileName);  }  }  public bool IsReusable  {  get  {  return false;  }  }  }  } |

## 上传成功显示图片

### 添加img标签



### 创建jquery脚本

|  |
| --- |
| function getUploadImg(file,serverData)  {  //serverData为返回的数据  $("#uploadImg").attr("src", serverData);  } |

### 修改upload\_success\_handler

|  |
| --- |
| upload\_success\_handler: getUploadImg, |

预览



## 一般处理程序优化

### 创建common类

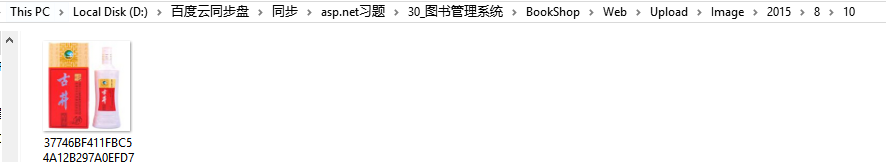
该类可以根据传入的文件流创建MD5，用来图片重命名

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  using System.IO;  namespace BookShop.Web.Common  {  public class WebCommon  {  /// <summary>  /// 计算文件的MD5值  /// </summary>  /// <param name="filepath"></param>  /// <returns></returns>  public static String GetStreamMD5(Stream stream)  {  string strResult = "";  string strHashData = "";  byte[] arrbytHashValue;  System.Security.Cryptography.MD5CryptoServiceProvider oMD5Hasher =  new System.Security.Cryptography.MD5CryptoServiceProvider();  arrbytHashValue = oMD5Hasher.ComputeHash(stream); //计算指定Stream 对象的哈希值  //由以连字符分隔的十六进制对构成的String，其中每一对表示value 中对应的元素；例如“F-2C-4A”  strHashData = System.BitConverter.ToString(arrbytHashValue);  //替换-  strHashData = strHashData.Replace("-", "");  strResult = strHashData;  return strResult;  }  }  } |

### 修改一般处理程序

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  //接收文件  HttpPostedFile jpeg\_image\_upload = context.Request.Files["Filedata"];  string fileName = Path.GetFileName(jpeg\_image\_upload.FileName);  string fileExt = Path.GetExtension(fileName);  if (fileExt == ".jpg")  {  //定义目录：形式为：Upload/Image/2015/8/7/  string dir = "/Upload/Image/" + DateTime.Now.Year + "/" + DateTime.Now.Month + "/" + DateTime.Now.Day + "/";  //创建文件夹。如果有该文件夹则不创建  Directory.CreateDirectory(Path.GetDirectoryName(context.Server.MapPath(dir)));  //GetStreamMD5传入文件流输出md5值  string fullDir = dir + Common.WebCommon.GetStreamMD5(jpeg\_image\_upload.InputStream) + fileExt;  jpeg\_image\_upload.SaveAs(context.Request.MapPath(fullDir));  //返回图片地址  context.Response.Write("Upload/Image/" + fileName);  }  } |

效果：



## 截图

### 原理

显示图片时，显示到div1中，在div2中再嵌套一个div作为截取的范围。

Div1的大小可以是固定的，也可以是不固定的；Div1固定时，传过来的图片要以缩略图技术与Div1匹配。Div1不固定时要根据图片大小调整。这里我们用不固定的方式实现。

1. 创建div

|  |
| --- |
| <div id="imgContent">  <div id="imgCut" style="width: 100px; height: 100px; border: 1px solid red"></div>  </div>  </div> |

1. 一般处理程序以json格式返回地址、宽度、高度

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  //接收文件  HttpPostedFile jpeg\_image\_upload = context.Request.Files["Filedata"];  string fileName = Path.GetFileName(jpeg\_image\_upload.FileName);  string fileExt = Path.GetExtension(fileName);  if (fileExt == ".jpg")  {  //定义目录：形式为：Upload/Image/2015/8/7/  string dir = "/Upload/Image/" + DateTime.Now.Year + "/" + DateTime.Now.Month + "/" + DateTime.Now.Day + "/";  //创建文件夹。如果有该文件夹则不创建  Directory.CreateDirectory(Path.GetDirectoryName(context.Server.MapPath(dir)));  //GetStreamMD5传入文件流输出md5值  string fullDir = dir + Common.WebCommon.GetStreamMD5(jpeg\_image\_upload.InputStream) + fileExt;  jpeg\_image\_upload.SaveAs(context.Request.MapPath(fullDir));  //返回图片大小  using (Image img = Image.FromStream(jpeg\_image\_upload.InputStream))  {  System.Web.Script.Serialization.JavaScriptSerializer jsSerializer = new System.Web.Script.Serialization.JavaScriptSerializer();  var data = new { imgUrl = fullDir, width = img.Width, height = img.Height };  string msg = jsSerializer.Serialize(data);  context.Response.Write(msg);  }  } |

1. 通过Jquery把图片作为div1的背景进行展示

|  |
| --- |
| function getUploadImg(file, serverData) {  //serverData为返回的数据  //转换为json对象  var obj = jQuery.parseJSON(serverData);  $("#imgContent").css("backgroundImage", "url(" + obj.imgUrl + ")").css("width", obj.width).css("height", obj.height);  } |

效果：



1. 用easyui实现拖拽与缩放

首先添加js脚本

|  |
| --- |
| <script src="js/jquery-easyui-1.4.3/jquery.min.js"></script>  <script src="js/jquery-easyui-1.4.3/jquery.easyui.min.js"></script> |

修改div

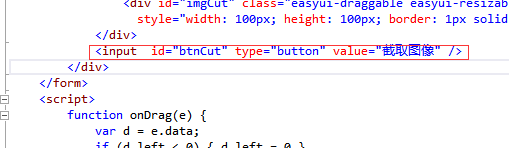
|  |
| --- |
| <div id="imgContent" class="easyui-panel" style="position: relative; overflow: hidden">  <div id="imgCut" class="easyui-draggable easyui-resizable" data-options="onDrag:onDrag" style="width: 100px; height: 100px; border: 1px solid red"></div>  </div> |

加载完div后加载脚本

|  |
| --- |
| <script>  function onDrag(e) {  var d = e.data;  if (d.left < 0) { d.left = 0 }  if (d.top < 0) { d.top = 0 }  if (d.left + $(d.target).outerWidth() > $(d.parent).width()) {  d.left = $(d.parent).width() - $(d.target).outerWidth();  }  if (d.top + $(d.target).outerHeight() > $(d.parent).height()) {  d.top = $(d.parent).height() - $(d.target).outerHeight();  }  }  </script>  </body> |

1. 截图

添加按钮



通过jquery向后台发送数据

|  |
| --- |
| $("#btnCut").click(function () {  var x = $("#imgCut").offset().left - $("#imgContent").offset().left;  var y = $("#imgCut").offset().top - $("#imgContent").offset().top;  var width = $("#imgCut").width();  var height = $("#imgCut").height();  var params = {  action: "cut",  imgUrl: obj.imgUrl,//展示的图片的地址，obj为全局变量  x: x,  y: y,  height: height,  width: width  };  $.post("upload.ashx", params, function (data) {  $("#cutImg").attr("src", data);//展示截图  });  }) |

保存截图

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  //接收文件  HttpPostedFile jpeg\_image\_upload = context.Request.Files["Filedata"];  string action = context.Request["action"];  if (action != "cut")  {  //上传图片  }  //保存图片  else  {  int x = (int)Convert.ToDouble(context.Request["x"]);  int y = (int)Convert.ToDouble(context.Request["y"]);  int width = (int)Convert.ToDouble(context.Request["width"]);  int height = (int)Convert.ToDouble(context.Request["height"]);  string imgUrl = context.Request["imgUrl"];  using (Bitmap map = new Bitmap(width, height))  {  using (Graphics g = Graphics.FromImage(map))  {  using (Image img = Image.FromFile(context.Server.MapPath(imgUrl)))  {  g.DrawImage(img, new Rectangle(0, 0, width, height), new Rectangle(x, y, width, height), GraphicsUnit.Pixel);  string fileName1 = Guid.NewGuid().ToString();  map.Save(context.Server.MapPath("upload/Image/" + fileName1 + ".jpg"));  context.Response.Write("upload/Image/" + fileName1 + ".jpg");  }  }  }  }  } |

添加img并展示

|  |
| --- |
| <img id="cutImg" /> |
| $("#cutImg").attr("src", data);//展示截图 |

1. 效果



### 使用imgAreaSelect插件实现

首先添加标签

|  |
| --- |
| <img id="imgUpload" src="#" />  <input id="btnCut" type="button" value="截取图像" /> |

在upload\_success\_handler中调用下面的方法，展示上传的图像

|  |
| --- |
| function getUploadImg(file, serverData) {  //serverData为返回的数据  //转换为json对象  obj = jQuery.parseJSON(serverData);  obj.i  $("#imgUpload").attr("src", obj.imgUrl);  } |

使用插件对展示的图片进行裁剪。首先添加js脚本

|  |
| --- |
| function getUploadImg(file, serverData) {  //serverData为返回的数据  //转换为json对象  obj = jQuery.parseJSON(serverData);  $("#imgUpload").attr("src", obj.imgUrl);  //设置img标签的选择范围  $('#imgUpload').imgAreaSelect({  selectionColor: 'blue', x1: 0, y1: 0, x2: 100, y2: 100,  maxWidth: 250, minWidth: 10, minHeight: 10, maxHeight: 250,  selectionOpacity: 0.2, onSelectEnd: preview  });  }  //设置选取框的选取信息  //利用jquery中的data方法来保存生成的数据  function preview(img, selection) {  $('#imgUpload').data('x', selection.x1);  $('#imgUpload').data('y', selection.y1);  $('#imgUpload').data('w', selection.width);  $('#imgUpload').data('h', selection.height);  } |

点击截取图像按钮，提交截取范围，并设置返回来的url

|  |
| --- |
| $("#btnCut").click(function () {  var pic = $('#imgUpload').attr('src');  var x,y,w,h;  $.post(  "upload.ashx",  {  action:"cut",  x:$('#imgUpload').data('x'),  y:$('#imgUpload').data('y'),  w:$('#imgUpload').data('w'),  h:$('#imgUpload').data('h'),  pic:pic  },  function(data){  //把裁剪后图片加载到原处  if(data){  $('#imgUpload').attr("src",data);  }  }  );  }); |

后台进行截图，并返回截取的图片的地址

|  |
| --- |
| int x = (int)Convert.ToDouble(context.Request["x"]);  int y = (int)Convert.ToDouble(context.Request["y"]);  int width = (int)Convert.ToDouble(context.Request["w"]);  int height = (int)Convert.ToDouble(context.Request["h"]);  string imgUrl = context.Request["pic"];  using (Bitmap map = new Bitmap(width, height))  {  using (Graphics g = Graphics.FromImage(map))  {  using (Image img = Image.FromFile(context.Server.MapPath(imgUrl)))  {  g.DrawImage(img, new Rectangle(0, 0, width, height), new Rectangle(x, y, width, height), GraphicsUnit.Pixel);  string fileName1 = Guid.NewGuid().ToString();  map.Save(context.Server.MapPath("upload/Image/" + fileName1 + ".jpg"));  context.Response.Write("upload/Image/" + fileName1 + ".jpg");  }  }  } |

### 完整代码

#### 前端

|  |
| --- |
| <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="BookShop.Web.WebForm2" %>  <!DOCTYPE html>  <html xmlns="http://www.w3.org/1999/xhtml">  <head runat="server">  <title></title>  <script src="SWFUpload/swfupload.js"></script>  <script src="SWFUpload/handlers.js"></script>  <%-- // <script src="js/jquery-1.7.1.js"></script>--%>  <link href="js/css/imgareaselect-default.css" rel="stylesheet" />  <script src="js/jquery-easyui-1.4.3/jquery.min.js"></script>  <script src="js/scripts/jquery.imgareaselect.min.js"></script>  <script src="js/jquery-easyui-1.4.3/jquery.easyui.min.js"></script>  <script type="text/javascript">  var swfu;  window.onload = function () {  var obj;  swfu = new SWFUpload({  // Backend Settings  //提交到的处理程序  upload\_url: "upload.ashx",  post\_params: {  "ASPSESSID": "<%=Session.SessionID %>"  },  // File Upload Settings  file\_size\_limit: "2 MB",  file\_types: "\*.jpg",  file\_types\_description: "JPG Images",  file\_upload\_limit: 0, // Zero means unlimited  // Event Handler Settings - these functions as defined in Handlers.js  // The handlers are not part of SWFUpload but are part of my website and control how  // my website reacts to the SWFUpload events.  swfupload\_preload\_handler: preLoad,  swfupload\_load\_failed\_handler: loadFailed,  file\_queue\_error\_handler: fileQueueError,  file\_dialog\_complete\_handler: fileDialogComplete,  upload\_progress\_handler: uploadProgress,  upload\_error\_handler: uploadError,  upload\_success\_handler: getUploadImg,  upload\_complete\_handler: uploadComplete,  // Button settings  button\_image\_url: "SWFUpload/images/XPButtonNoText\_160x22.png",  button\_placeholder\_id: "spanButtonPlaceholder",  button\_width: 160,  button\_height: 22,  //按钮上显示的文字  button\_text: '<span class="button">选择图片<span class="buttonSmall">(2 MB Max)</span></span>',  button\_text\_style: '.button { font-family: Helvetica, Arial, sans-serif; font-size: 14pt; } .buttonSmall { font-size: 10pt; }',  button\_text\_top\_padding: 1,  button\_text\_left\_padding: 5,  // Flash Settings  flash\_url: "SWFUpload/swfupload.swf", // Relative to this file  flash9\_url: "SWFUpload/swfupload\_FP9.swf", // Relative to this file  custom\_settings: {  upload\_target: "divFileProgressContainer"  },  // Debug Settings  debug: false  });  function getUploadImg(file, serverData) {  $("#imgUpload").attr("src", serverData);  //设置img标签的选择范围  $('#imgUpload').imgAreaSelect({  aspectRatio: "1:1",//选取为正方形  selectionColor: 'blue', x1: 0, y1: 0, x2: 100, y2: 100,  maxWidth: 250, minWidth: 10, minHeight: 10, maxHeight: 250,  selectionOpacity: 0.2, onSelectEnd: preview  });  }  //设置选取框的选取信息  //利用jquery中的data方法来保存生成的数据  function preview(img, selection) {  $('#imgUpload').data('x', selection.x1);  $('#imgUpload').data('y', selection.y1);  $('#imgUpload').data('w', selection.width);  $('#imgUpload').data('h', selection.height);  }  $("#btnCut").click(function () {  var pic = $('#imgUpload').attr('src');  var x, y, w, h;  $.post(  "upload.ashx",  {  action: "cut",  x: $('#imgUpload').data('x'),  y: $('#imgUpload').data('y'),  w: $('#imgUpload').data('w'),  h: $('#imgUpload').data('h'),  pic: pic  },  function (data) {  //把裁剪后图片加载到原处  if (data) {  $('#imgUpload').attr("src", data);  }  }  );  });  }  </script>  <script type="text/javascript">  $(function () {  });  </script>  </head>  <body>  <form id="form1" runat="server">  <div id="content">  <div id="swfu\_container" style="margin: 0px 10px;">  <div>  <span id="spanButtonPlaceholder"></span>  </div>  <div id="divFileProgressContainer" style="height: 75px;"></div>  <div id="thumbnails"></div>  </div>  <img id="imgUpload" src="#" />  <input id="btnCut" type="button" value="截取图像" />  </div>  </form>  <script>  function onDrag(e) {  var d = e.data;  if (d.left < 0) { d.left = 0 }  if (d.top < 0) { d.top = 0 }  if (d.left + $(d.target).outerWidth() > $(d.parent).width()) {  d.left = $(d.parent).width() - $(d.target).outerWidth();  }  if (d.top + $(d.target).outerHeight() > $(d.parent).height()) {  d.top = $(d.parent).height() - $(d.target).outerHeight();  }  }  </script>  </body>  </html> |

#### 后端

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.IO;  using System.Linq;  using System.Web;  namespace BookShop.Web  {  /// <summary>  /// Summary description for upload  /// </summary>  public class upload : IHttpHandler  {  public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  //接收文件  HttpPostedFile jpeg\_image\_upload = context.Request.Files["Filedata"];  string action = context.Request["action"];  if (action != "cut")  {  string fileName = Path.GetFileName(jpeg\_image\_upload.FileName);  string fileExt = Path.GetExtension(fileName);  if (fileExt == ".jpg")  {  //定义目录：形式为：Upload/Image/2015/8/7/  string dir = "/Upload/Image/" + DateTime.Now.Year + "/" + DateTime.Now.Month + "/" + DateTime.Now.Day + "/";  //创建文件夹。如果有该文件夹则不创建  Directory.CreateDirectory(Path.GetDirectoryName(context.Server.MapPath(dir)));  //GetStreamMD5传入文件流输出md5值  string fullDir = dir + Common.WebCommon.GetStreamMD5(jpeg\_image\_upload.InputStream) + fileExt;  jpeg\_image\_upload.SaveAs(context.Request.MapPath(fullDir));  //返回图片    context.Response.Write(fullDir);    }  }  //保存图片  else  {  int x = (int)Convert.ToDouble(context.Request["x"]);  int y = (int)Convert.ToDouble(context.Request["y"]);  int width = (int)Convert.ToDouble(context.Request["w"]);  int height = (int)Convert.ToDouble(context.Request["h"]);  string imgUrl = context.Request["pic"];  using (Bitmap map = new Bitmap(width, height))  {  using (Graphics g = Graphics.FromImage(map))  {  using (Image img = Image.FromFile(context.Server.MapPath(imgUrl)))  {  g.DrawImage(img, new Rectangle(0, 0, width, height), new Rectangle(x, y, width, height), GraphicsUnit.Pixel);  string fileName1 = Guid.NewGuid().ToString();  map.Save(context.Server.MapPath("upload/Image/" + fileName1 + ".jpg"));  context.Response.Write("upload/Image/" + fileName1 + ".jpg");  }  }  }  }  }  public bool IsReusable  {  get  {  return false;  }  }  }  } |

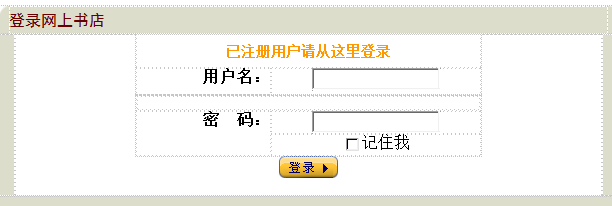
# 登陆

## 创建用户控件

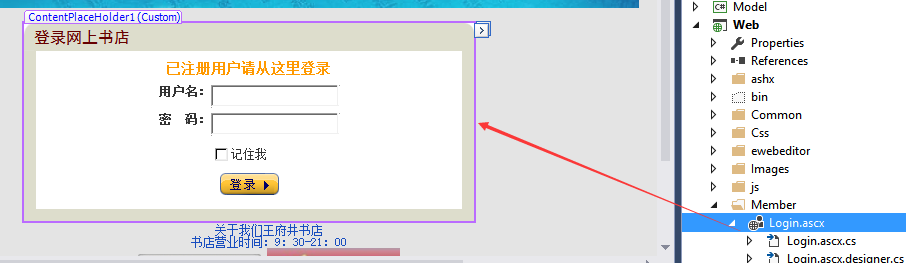
创建web forms 用户控件



**添加标签后效果如下：**



创建aspx，之后再design模式下，拖拽用户控件进去



## 校验用户

在ascx.cs文件的Page\_Load中进行校验

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  if (IsPostBack)  {  checkUserInfo();  }  }  private void checkUserInfo()  {  string Uname = Request["txtLoginId"];  string UPass = Request["txtLoginPassword"];  BLL.UserManager bll = new BLL.UserManager();  string msg = string.Empty;  Model.User model = null;  bool b = bll.UserLogin(Uname, UPass, out msg, out model);  if(b)  {  Session["userInfo"] = model;//把用户存到session中  Response.Redirect("/ShowMsg.aspx?msg="+Server.UrlEncode(msg)+"&txt="+Server.UrlEncode("首页")+ "&url=/Default.aspx");  }  } |

## 在bll中创建UserLogin方法

在bll的分部类中创建:

|  |
| --- |
| public bool UserLogin(string uname, string uPass, out string msg, out User model)  {  //根据名称获取实体  model = dal.GetModel(uname);  if (model != null)  {  if(model.UserState.Name=="正常")  {  if(model.LoginPwd == uPass)  {  msg = "登陆成功";  return true;  }else  {  msg = "密码错误";  return false;  }  }else  {  msg = "此用户已经被锁定了";  return false;  }  }  else  {  msg = "此用户为空";  return false;  }  } |

## 在dal中创建GetModel方法

在dal的分部类中创建

|  |
| --- |
| public User GetModel(string uname)  {  StringBuilder strSql = new StringBuilder();  strSql.Append("select top 1 Id,LoginId,LoginPwd,Name,Address,Phone,Mail,UserStateId from Users ");  strSql.Append(" where LoginId=@LoginId ");  SqlParameter[] parameters = {  new SqlParameter("@LoginId", SqlDbType.NVarChar,50) };  parameters[0].Value = uname;  BookShop.Model.User model = new BookShop.Model.User();  DataSet ds = DbHelperSQL.Query(strSql.ToString(), parameters);  if (ds.Tables[0].Rows.Count > 0)  {  if (ds.Tables[0].Rows[0]["Id"].ToString() != "")  {  model.Id = int.Parse(ds.Tables[0].Rows[0]["Id"].ToString());  }  model.LoginId = ds.Tables[0].Rows[0]["LoginId"].ToString();  model.LoginPwd = ds.Tables[0].Rows[0]["LoginPwd"].ToString();  model.Name = ds.Tables[0].Rows[0]["Name"].ToString();  model.Address = ds.Tables[0].Rows[0]["Address"].ToString();  model.Phone = ds.Tables[0].Rows[0]["Phone"].ToString();  model.Mail = ds.Tables[0].Rows[0]["Mail"].ToString();  if (ds.Tables[0].Rows[0]["UserStateId"].ToString() != "")  {  int UserStateId = int.Parse(ds.Tables[0].Rows[0]["UserStateId"].ToString());  model.UserState = userStateServices.GetModel(UserStateId);  }  return model;  }  else  {  return null;  }  } |

# 无登陆跳转到登陆界面

当输入网页没有登陆时，跳转到登陆页面，登陆后跳转回来。

## 设计检查是否登陆的类

因为登陆后，把用户信息写到session中了，所以只需判断session是否为空就可判断是否登陆。新建一个名为“CheckLogin.cs”文件

|  |
| --- |
| public class CheckLogin:System.Web.UI.Page//继承自page  {  protected override void OnPreInit(EventArgs e)  {  if(Session["userInfo"] ==null)  {  HttpContext.Current.Response.Redirect("/Member/Login.aspx?returnUrl="+HttpContext.Current.Request.Url.ToString());  }  }  } |

## 需要登录才能进入的页面继承自CheckLogin类

|  |
| --- |
| public partial class WebForm1 :Common.CheckLogin  {  protected void Page\_Load(object sender, EventArgs e)  {  }  } |

## 登陆页面中判断是否有get的参数

如果get过来网页则登陆后跳转到源网页。

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  if (IsPostBack)  {  checkUserInfo();  }  }  private void checkUserInfo()  {  string Uname = Request["txtLoginId"];  string UPass = Request["txtLoginPassword"];  BLL.UserManager bll = new BLL.UserManager();  Model.User model = null;  bool b = bll.UserLogin(Uname, UPass, out msg, out model);  if (b)  {  Session["userInfo"] = model;  //如果get过来网址，则跳转  if (!string.IsNullOrEmpty(Request.QueryString["returnUrl"]))  {  Response.Redirect(Request.QueryString["returnUrl"]);  }  Response.Redirect("/ShowMsg.aspx?msg=" + Server.UrlEncode(msg) + "&txt=" + Server.UrlEncode("首页") + "&url=/Default.aspx");  }  } |

## 登陆页面中注册按钮处理

如果在登陆页面中点击注册按钮，则跳转到注册页面。注册完毕也需要返回最初跳转过来的页面。所以需要把网页传递过来。

在后台中，创建一个参数接收传递过来的网页参数。因为没有点击登陆，登陆页面不是post过来的。所以非post中也需要接收参数。

|  |
| --- |
| public string returnUlr = string.Empty;  protected void Page\_Load(object sender, EventArgs e)  {  if (IsPostBack)  {  checkUserInfo();  }  else  {  returnUlr = Request.QueryString["returnUrl"];  }  } |

修改超链接的href

|  |
| --- |
| <%if (!string.IsNullOrEmpty(returnUlr))  {%>  <a href="Register.aspx?url=<%=returnUlr%>">注册</a>  <%}  else  {  %>  <a href="Register.aspx">注册</a>  <%} %> |

如果是跳转过来的则传递网页，如果是从登陆页面过来的则不传递参数。

# 记住账号密码

## 创建Md5加密类

在用cookies保存密码时用md5进行加密，在common中创建

|  |
| --- |
| public static string GetstringMD5(string loginPwd)  {  MD5 md5 = MD5.Create();  Byte[] buffer = System.Text.Encoding.Unicode.GetBytes(loginPwd);  byte[] md5Byte = md5.ComputeHash(buffer);  StringBuilder sb = new StringBuilder();  foreach (byte b in md5Byte)  {  sb.Append(b.ToString("x2"));  }  return sb.ToString();  } |

在登陆页面中进行二次加密

|  |
| --- |
| //进行2此md5计算  private string MD5Pwd(string loginPwd)  {  return Web.Common.WebCommon.GetstringMD5(Web.Common.WebCommon.GetstringMD5(loginPwd));  } |

## 设置cookies

账号、密码校验成功后，判断是否点击了十天内免登陆checkbox。如果点击了该checkbox则设置cookies

|  |
| --- |
| Model.User model = null;  bool b = bll.UserLogin(Uname, UPass, out msg, out model);  if (b)  {  Session["userInfo"] = model;  //checkbox不为空表示点击了，则设置cookies  if (!string.IsNullOrEmpty(Request.Form["checkMe"]))  {  Response.Cookies["r1"].Value = model.LoginId;  Response.Cookies["r2"].Value = MD5Pwd(model.LoginPwd);//Md5加密  Response.Cookies["r1"].Expires = DateTime.Now.AddDays(10);  Response.Cookies["r2"].Expires = DateTime.Now.AddDays(10);  } |

## 自动登陆

当下一次进入登陆页面时，判断cookie中账号、密码是否正确，正确则跳转到首页。

因为进入页面是get过来的，所以是非IsPostBack

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  if (IsPostBack)  {  checkUserInfo();  }  else  {  returnUlr = Server.UrlPathEncode(Request.QueryString["returnUrl"]);  checkUserCookies();  }  }  //校验cookies是否正确  private void checkUserCookies()  {  if (Request.Cookies["r1"] != null && Request.Cookies["r2"] != null)  {  string uid = Request.Cookies["r1"].Value;  string pwd = Request.Cookies["r2"].Value;  BLL.UserManager bll = new BLL.UserManager();  Model.User model = bll.GetModel(uid);  if (model != null)  {  if (pwd == MD5Pwd(model.LoginPwd))  {  Session["userInfo"] = model;  Response.Redirect("/ShowMsg.aspx?msg=" + Server.UrlEncode(msg) + "&txt=" + Server.UrlEncode("首页") + "&url=/Default.aspx");  }else//因为有可能登陆后，修改密码了导致cookie与数据库中密码不相同。删除cookie即可  {  //删除cookie  Response.Cookies["r1"].Expires = DateTime.Now.AddDays(-1);  Response.Cookies["r2"].Expires = DateTime.Now.AddDays(-1);  }  }  }  } |

# 关于runat="server"

Aspx页面中的form默认是下面的类型

|  |
| --- |
| <form id="form1" runat="server"> |

这个会在生成的html页面中附加上viewstate。我们可以优化为纯html类型的，这样生成的就不带viewstate了

|  |
| --- |
| <form method="post" action=""> |

Action为空表示提交到当前页面。

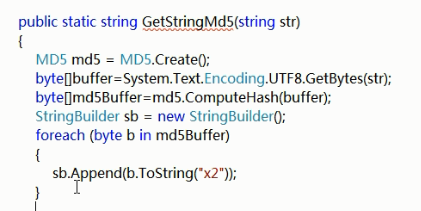
改为纯粹的html类型的form后，IsPostBack就不能用了。因为IsPostBack与viewstate有关。解决方法是传递一个隐藏域。

|  |
| --- |
| <input type="hidden" name="IsPostBack" value="1" /> |

在判断是否提交时

|  |
| --- |
| if (!string.IsNullOrEmpty(Request.Form["IsPostBack"]))  {  //提交过来的  } |

此外模板页中的form可以提到子页面中。需要提交数据的子页面内部可以添加form标签，而不需要提交数据的子页面则不需添加form标签。



# 图书列表页

## 查询进行展示

这里用reeater进行展示。

### 后置代码中绑定数据

|  |
| --- |
| public partial class WebForm1 : System.Web.UI.Page  {  protected void Page\_Load(object sender, EventArgs e)  {  BLL.BookManager bll = new BLL.BookManager();  List<Model.Book> books = bll.GetModelList("");  Repeater1.DataSource = books;  Repeater1.DataBind();  }  } |

### 前台展示数据

新知识：Eval格式化

占位符：# Eval("ISBN","../Images/BookCovers/{0}.jpg")

数字保留位数： <%# Eval("UnitPrice","{0:0.00}")

|  |
| --- |
| <body>  <form id="form1" runat="server">  <div style="width: 500px">    <asp:Repeater ID="Repeater1" runat="server">  <HeaderTemplate>    </HeaderTemplate>  <ItemTemplate>  <table>  <tbody>  <tr>  <td rowspan="2"><a onclick="window.location='BookDetail.aspx?bid=<%# Eval("Id") %>'">  <img style="cursor: hand" height="121" alt="<%# Eval("Title") %>" hspace="4" src="<%# Eval("ISBN","../Images/BookCovers/{0}.jpg") %>" width="95"></a>  </td>  <td style="font-size: small; color: red" width="650"><a class="booktitle" id="link\_prd\_name" href="BookDetail.aspx?bid=<%# Eval("Id") %>" target="\_blank" name="link\_prd\_name"><%# Eval("Title") %></a> </td>  </tr>  <tr>    <td align="left"><span style="font-size: 12px; line-height: 20px"><%# Eval("Author") %></span><br>  <br>  <span style="font-size: 12px; line-height: 20px"><%# Eval("ContentDescription") %></span>  </td>  </tr>  <tr>  <td align="right" colspan="2"><span style="font-weight: bold; font-size: 13px; line-height: 20px">¥  <%# Eval("UnitPrice","{0:0.00}") %></span> </td>  </tr>  </tbody>  </table>  <hr />  </ItemTemplate>  </asp:Repeater>  </div>  </form>  </body> |

### 截取描述信息

展示页中只显示100个字符，可以创建一个方法对字符串进行截取

|  |
| --- |
| public string TrimStringToXLength(string str, int length)  {  if (str.Length < length)  {  return str;  }  else  {  str = str.Substring(0, 100)+"...........";  return str;    }  } |

前台页面调整

|  |
| --- |
| <%# this.TrimStringToXLength( Eval("ContentDescription").ToString(),100) %> |

## 分页

### 根据类别查询条数

在dal中查询商品某类别下的总条数

|  |
| --- |
| public partial class BookServices  {  /// <summary>  /// 求指定类别下的总的记录数  /// </summary>  /// <param name="categoryId">类别编号.如果为0则表示全部数据</param>  /// <returns></returns>  public int GetRecordCount(int categoryId)  {  string sql = "select count(1) from Books";  if (categoryId != 0)  {  sql = sql + "where CategoryId=@CategoryID";  }  return Convert.ToInt32(DbHelperSQL.GetSingle(sql, new SqlParameter("@CategoryID", categoryId)));  }  } |

### 计算总的页数

在bll层计算总页数

|  |
| --- |
| public partial class BookManager  {  /// <summary>  /// 计算页的总数  /// </summary>  /// <param name="categoryId">类别</param>  /// <param name="pageSize">每页记录数</param>  /// <returns></returns>  public int GetPageCount(int categoryId, int pageSize)  {  int recordCound = dal.GetRecordCount(categoryId);  return Convert.ToInt32(Math.Ceiling((double)recordCound / pageSize));  }  } |

### 根据分页数据获取实体

在dal中

|  |
| --- |
| public DataSet GetPageList(int start, int end, int categoryId)  {  string sql = "select \* from(select ROW\_NUMBER()over(order by id) as num,\* from Books {0}) as t where num>=@start and num<=@end";  sql = string.Format(sql, categoryId != 0 ? "where CategoryId=@CategoryId" : "");  SqlParameter[] sps = { new SqlParameter("@start", start), new SqlParameter("@end", end), new SqlParameter("@CategoryId", categoryId) };  return DbHelperSQL.Query(sql, sps);  } |

### 获取分页数据并返回实体集合

在bll中

|  |
| --- |
| public List<Model.Book> GetPageList(int pageIndex, int pageSize, int categoryId)  {  int start = (pageIndex - 1) \* pageSize + 1;  int end = pageIndex \* pageSize;  DataSet ds = dal.GetPageList(start, end, categoryId);  return DataTableToList(ds.Tables[0]);  } |

### 前端展示

可以用html标签、服务端控件等实现分页。服务端控件会产生viewstate，禁用viewsate后，服务端控件需要用隐藏域来传值。Html则是通用的。

#### Html标签

|  |
| --- |
| <div>  第<%=CurrentPage %>页 共<%=PageCount %>页  <input type="hidden" name="currentPage" value="<%=CurrentPage %>" />  <input type="submit" name="prePage" value="上一页" />  <input type="submit" name="nextPage" value="下一页" />  </div> |

#### 后置代码

|  |
| --- |
| public partial class WebForm1 : System.Web.UI.Page  {  public int CurrentPage;  public int PageCount;  protected void Page\_Load(object sender, EventArgs e)  {  int pageSize = 5;  int category = 0;  BLL.BookManager bll = new BLL.BookManager();  PageCount = bll.GetPageCount(0, pageSize);  int pageIndex = Convert.ToInt32(Request["currentPage"] ?? "1");  //不为空表示按下了prepage按钮  if (!string.IsNullOrEmpty(Request["prePage"]))  {  pageIndex--;  if (pageIndex < 1)  {  pageIndex = 1;  }  }  if (!string.IsNullOrEmpty(Request["nextPage"]))  {  pageIndex++;  if (pageIndex > bll.GetPageCount(category, pageSize))  {  pageIndex = category;  }  }  CurrentPage = pageIndex;  List<Model.Book> books = bll.GetPageList(pageIndex, pageSize, category);  Repeater1.DataSource = books;  Repeater1.DataBind();  }  public string TrimStringToXLength(string str, int length)  {  if (str.Length < length)  {  return str;  }  else  {  str = str.Substring(0, 100) + "...........";  return str;  }  }  } |

### 数字分页

#### 编写类

|  |
| --- |
| public class commons  {  public static string htmlPaging(int pageSize, int PageIndex, int PageCount)  {  if (PageCount == 1)//只有1页，则不显示  {  return "";  }  int start = PageIndex - 4;  if (start < 0)//从1开始。  {  start = 1;  }  int end = start + 8;  if (end > PageCount)  {  end = PageCount;    start = end - 8 > 0 ? end - 8 : 1;  }  StringBuilder sb = new StringBuilder();  string ulr = HttpContext.Current.Request.Url.LocalPath;  sb.AppendFormat("<a href={0}>首页</a>", ulr);  for (int i = start; i <= end; i++)  {  if (i == PageIndex)  {  sb.Append(i);  }  else  {    sb.AppendFormat("<a href={0}?pageIndex={1}&pageSize={2}>{1}</a>", ulr, i, pageSize);  }  }  sb.AppendFormat("<a href={0}?pageIndex={1}&pageSize={2}>末页</a>", ulr,PageCount,pageSize);  return sb.ToString();  }  } |

#### 前端展示

样式文件

|  |
| --- |
| body {  }  .page\_nav{clear:both; padding:15px 0; color:#666; font:normal 12px/24px Arial; text-align:center;}  .page\_nav a{display:inline-block; height:22px; margin:0 2px; padding:0 8px; text-decoration:none; border:solid 1px #dbe5ee; -moz-border-radius:2px; -webkit-border-radius:2px; -khtml-border-radius:2px; border-radius:2px; background:#fff; color:#333; font:normal 12px/22px Arial, Helvetica, sans-serif; cursor:pointer;}  .page\_nav strong{display:inline-block; height:24px; margin:0 3px; padding:0 8px; border:none; -moz-border-radius:2px; -webkit-border-radius:2px; -khtml-border-radius:2px; border-radius:2px; background:#C00; color:#fff; font-weight:normal; line-height:24px; text-decoration:none;}  .page\_nav a:hover,  .page\_nav a.on{height:24px; margin:0 3px; border:none; background:#C00; color:#fff; line-height:24px; text-decoration:none;}  .page\_nav a.select{cursor:default;}  .page\_nav .view\_all{display:block; text-align:center;}  .page\_nav .view\_all a{height:auto; margin:0; padding:0; border:none; color:#06c; line-height:24px;}  .page\_nav .view\_all a:hover{height:auto; margin:0; padding:0; background:none;} |

前端代码

|  |
| --- |
| <div>  <p class="page\_nav">  <%=htmlpage1%>  </p>  </div> |

后置代码

|  |
| --- |
| public partial class WebForm1 : System.Web.UI.Page  {  public string htmlpage1;  protected void Page\_Load(object sender, EventArgs e)  {  int pageSize = 5;  int category = 0;  BLL.BookManager bll = new BLL.BookManager();  int PageCount = bll.GetPageCount(0, pageSize);  int pageIndex = Convert.ToInt32(Request["pageIndex"] ?? "1");  htmlpage1 = commons.htmlPaging(pageSize, pageIndex, PageCount);  List<Model.Book> books = bll.GetPageList(pageIndex, pageSize, category);  Repeater1.DataSource = books;  Repeater1.DataBind();  }  public string TrimStringToXLength(string str, int length)  {  if (str.Length < length)  {  return str;  }  else  {  str = str.Substring(0, 100) + "...........";  return str;  }  }  } |

## 静态页

可以减轻数据库压力。作用与缓存类似，但是不需要net 进行处理，速度更快。

像详情列表等不经常更改的页面可以生成静态页面。

### 创建静态页模板

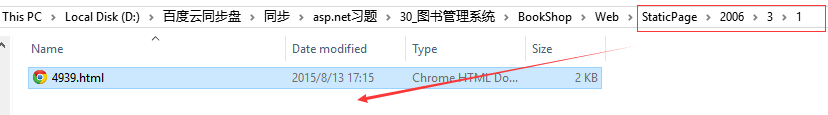
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>&title</title>  <meta charset="utf-8" />  </head>  <body>  <table>  <tr>  <td>$id</td>  </tr>  <tr>  <td>$content</td>  </tr>  </table>  </body>  </html> |

### 读取数据创建为静态页

创建aspx页面，点击按钮，进行处理生成html静态页面

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {    if (IsPostBack)  {  //读取数据  BLL.BookManager bll = new BLL.BookManager();  List<Model.Book> books = bll.GetModelList("Id<4945");  //读取模板文件  string temple = HttpContext.Current.Server.MapPath("StaticTemple.html");//指定模板路径  string htmltemple = File.ReadAllText(temple);  //遍历数据  foreach (var item in books)  {  string html = htmltemple;  //替换占位符  html = html.Replace("&title", item.Title).Replace("$id", item.Id.ToString()).Replace("$content", item.ContentDescription);  //创建文件夹  string dir = HttpContext.Current.Server.MapPath("/StaticPage/" + item.PublishDate.Year + "/" + item.PublishDate.Month + "/" + item.PublishDate.Day + "/");  Directory.CreateDirectory(Path.GetDirectoryName(dir));  //生成html文件  File.WriteAllText(dir + item.Id + ".html", html, Encoding.UTF8);  }  }  } |

结果：



## 详情页面替换为静态页

在9.1节展示的页面中，点击标题可以转到一个详情页，这个详情页我们用9.3的静态页实现。

主要难点：如何定位到该静态页面。

### 后置代码

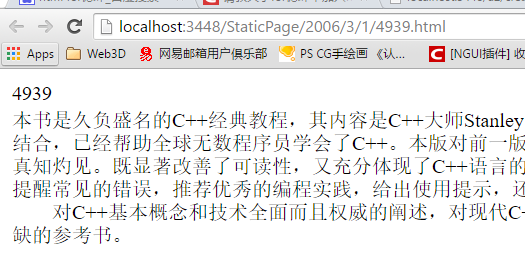
根据传入的出版日期定位到文件夹的方法。

|  |
| --- |
| public string GetDir(object obj)  {  DateTime d = Convert.ToDateTime(obj);  return "/StaticPage/" + d.Year + "/" + d.Month + "/" + d.Day + "/";  } |

### 前端绑定

|  |
| --- |
| <a href="<%#this.GetDir(Eval("PublishDate")) %><%# Eval("Id") %>.html"> |

点击超链接即可跳转



## 评论

在静态页面中进行评论，并加载评论

### 追加评论

#### 修改静态模板

添加评论用标签

|  |
| --- |
| ~~<table>~~  ~~<tr>~~  ~~<td>$id</td>~~  ~~</tr>~~  ~~<tr>~~  ~~<td>$content</td>~~  ~~</tr>~~  ~~</table>~~  <hr />  <textarea id="txtareaComment" rows="20" cols="50"></textarea>  <br />  <input type="button" value="发表评论" id="btn" /> |

异步提交数据

|  |
| --- |
| ~~<title>&title</title>~~  ~~<meta charset="utf-8" />~~  <script src="/js/jquery-1.7.1.js"></script><!--注意这个路径是相对生成的静态页面的，而不是模板文件-->  <script type="text/javascript">  $(function () {  $("#btn").click(function () {  var msg = $("#txtareaComment").val();  if (msg != null) {  $.post("/ashx/BookCommont.ashx", { action: "add", msg: msg, bookId: $id }, function (data) {  });  } else {  alert("评论不能为空");  }  });  });  </script> |

#### 一般处理程序接收数据并更新到数据库

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  string action = context.Request["action"];  if (action == "add")  {  Model.BookComment model = new Model.BookComment();  model.BookId = Convert.ToInt32(context.Request["bookId"]);  model.Msg = context.Request["msg"];  model.CreateDateTime = DateTime.Now;  BLL.BookCommentManager bll = new BLL.BookCommentManager();  if (bll.Add(model) > 0)  {  context.Response.Write("success");  }  else  {  context.Response.Write("fail");  }  }  } |

### 加载评论

#### 修改静态模板

修改标签用来容纳加载的评论

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

添加加载评论的方法

|  |
| --- |
| function loadCommont() {  $.post("/ashx/BookCommont.ashx", { action: "load", bookId: $id }, function (data) {  var comments = $.parseJSON(data);  for (var i = 0; i < comments.length; i++) {  $("<h3>" + comments[i].CreateDateTime + "</h3>" + "<p>" + comments[i].Msg + "</p><hr />").appendTo("#comment");  }  });  } |

添加刷新评论的方法

|  |
| --- |
| function RefreshComment() {  loadCommont();  $("#txtareaComment").val("");  } |

调用这两个方法：



#### 一般处理程序接收id并返回json数据

|  |
| --- |
| if (action == "add")  {  .........  }  else if(action=="load")  {  int bookId = Convert.ToInt32(context.Request["bookId"]);  BLL.BookCommentManager bll = new BLL.BookCommentManager();  List<Model.BookComment> lists = bll.GetModelList("BookId=" + bookId);  System.Web.Script.Serialization.JavaScriptSerializer jss = new System.Web.Script.Serialization.JavaScriptSerializer();  jss = new System.Web.Script.Serialization.JavaScriptSerializer();  string data = jss.Serialize(lists);  context.Response.Write(data);  } |

### 显示时间调整

我们将实现时间显示为几天前、几个小时前灯形式。

在net中两个时间差为TimeSpan格式，我们首先创建一个方法对当前时间与评论发布时间的差进行处理

|  |
| --- |
| public static string GetTimeSpan(TimeSpan ts)  {  if (ts.TotalDays > 365)  {  return Math.Floor(ts.TotalDays / 365) + "年前";  }  else if (ts.TotalDays > 30)  {  return Math.Floor(ts.TotalDays / 30) + "月前";  }  else if (ts.TotalHours >= 24)  {  return Math.Floor(ts.TotalDays) + "天前";  }  else if (ts.TotalHours >= 1)  {  return Math.Floor(ts.TotalHours) + "小时前";  }  else if (ts.TotalMinutes >= 1)  {  return Math.Floor(ts.TotalMinutes) + "分钟前";  }  else  {  return "刚刚";  }  } |

当从后台传值回来时，使用model会传递所有参数，可以创建一个viewmodel（MVC中的v）只有要显示的参数。把model中的参数的值赋值给viewmodel中的参数，传递回来即可。

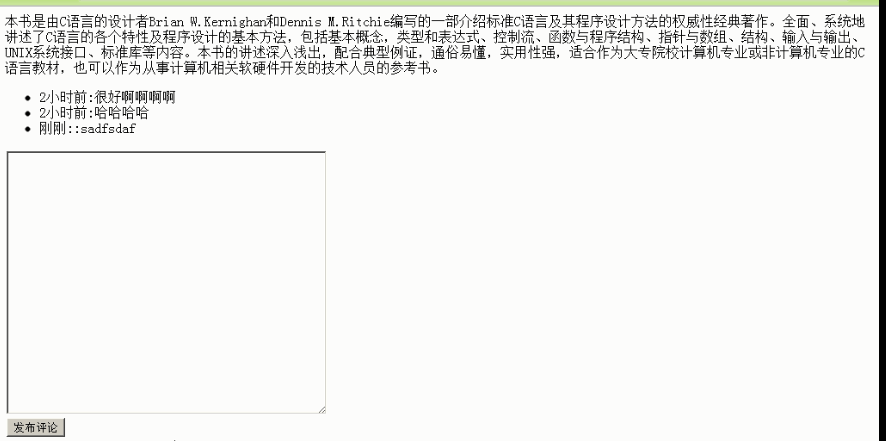
创建一个viewModel，只有要展示的值

|  |
| --- |
| public class CommentViewModel  {  public string CreateDateTime { get; set; }  public string Msg { get; set; }  } |

在处理程序中进行如下处理

|  |
| --- |
| else if (action == "load")  {  int bookId = Convert.ToInt32(context.Request["bookId"]);  BLL.BookCommentManager bll = new BLL.BookCommentManager();  List<Model.BookComment> lists = bll.GetModelList("BookId=" + bookId);  //创建viwmodel  List<CommentViewModel> newLists = new List<CommentViewModel>();  foreach (var item in lists)  {  CommentViewModel vm = new CommentViewModel();  //计算当前时间与评论时间之差  TimeSpan ts = DateTime.Now - item.CreateDateTime;  //把处理后的时间差赋值给viwmodel的参数  vm.CreateDateTime = Common.WebCommon.GetTimeSpan(ts);  vm.Msg = item.Msg;    newLists.Add(vm);  }  System.Web.Script.Serialization.JavaScriptSerializer jss = new System.Web.Script.Serialization.JavaScriptSerializer();  jss = new System.Web.Script.Serialization.JavaScriptSerializer();  //返回viewmodel  string data = jss.Serialize(newLists);  context.Response.Write(data);  } |

效果：



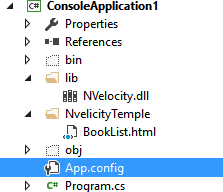
## Nvelocity模板引擎

在9.3.1创建静态模板时，当需要替换的对象的属性比较多时，需要一个个定义占位符；如果想替换的是集合对象的属性时，就需要借助模板引擎了。

Nvelocity模板引擎支持诸如foreach循环等操作，可以快速实现html生成。

### 案例:生成图书详情列表

#### 创建控制台程序



把插件放到lib文件夹下，并通过路径引用该程序集

把html模板文件放到NvelocityTemple文件夹下。

配置app.config链接数据库。

#### 静态模板文件

|  |
| --- |
| <!DOCTYPE html>  <html lang="en" xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta charset="utf-8" />  <!--占位符要$开始-->  <title>$PageTitle</title>  </head>  <body>  <!--循环以#foreach开始，#end结束-->  <!--book变量也以#开始，变量不替换-->  #foreach($book in $books)  <table>  <!--设置Id属性-->  <tr><td>$book.Id</td></tr>  <tr><td>$book.Title</td></tr>  <tr><td>$book.Author</td></tr>  <tr><td>$book.AurhorDescription</td></tr>  <tr><td>$book.PublishDate</td></tr>  <tr><td>$book.ISBN</td></tr>  <tr><td>$book.WordsCount</td></tr>  <tr><td>$book.UnitPrice</td></tr>  <tr><td>$book.ContentDescription</td></tr>  </table>  <hr />  #end  </body>  </html> |

#### 从数据中读取对象、替换与生产静态文件

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using BookShop.BLL;  using BookShop.Model;  using NVelocity.App;  using NVelocity.Runtime;  using NVelocity;  using System.IO;  namespace ConsoleApplication1  {  class Program  {  static void Main(string[] args)  {  VelocityEngine vltEngine = new VelocityEngine();//声明Velocity模板引擎。  //通过Velocity操作文件。  vltEngine.SetProperty(RuntimeConstants.RESOURCE\_LOADER, "file");  vltEngine.SetProperty(RuntimeConstants.FILE\_RESOURCE\_LOADER\_PATH, @"D:\百度云同步盘\同步\asp.net习题\30\_图书管理系统\BookShop\ConsoleApplication1\NvelicityTemple\");//指定读取模板文件的路径.  vltEngine.Init();//初始化。  //向模板中输出的内容（替换模板中的占位符）  VelocityContext vltContext = new VelocityContext();  vltContext.Put("PageTitle", "图书列表");//put类似replace  BookManager bll = new BookManager();  List<Book> books = bll.GetModelList("Id<4945");  vltContext.Put("books", books);  Template vltTemplate = vltEngine.GetTemplate("BookList.html");//获取具体模板页面  System.IO.StringWriter vltWriter = new System.IO.StringWriter();  vltTemplate.Merge(vltContext, vltWriter);  File.WriteAllText(@"list.html", vltWriter.GetStringBuilder().ToString());  Console.WriteLine(vltWriter.GetStringBuilder().ToString());  Console.ReadKey();  }  }  } |

# 购物车页面

在图书详情页面中点击购买添加该商品到购物车中。

功能：

1. 持久化
2. 区分用户商品
3. 匿名用户也可以商品放到购物车中

实现方式：cookie。

当实现不同电脑上购物车商品信息同步时，需要保存信息到数据库中。匿名时保存信息到cookie，登陆后保存信息到数据库即可。

## 静态页面中添加超链接

|  |
| --- |
| <a href="Cart.aspx?bookId=$id"><img src="/Images/Common/buy.gif" /></a> |

点击图片超链接跳转到购物车页面

## 图书、用户信息保存到数据库

在cart.aspx中，先把图书id、用户id、购买数量保存到数据库中。

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  addCart();  }  private void addCart()  {  int bookId;  if (int.TryParse(Request["bookId"], out bookId))  {  //根据id从数据库中查询该商品  BLL.BookManager bookBll = new BLL.BookManager();  Model.Book bookModel = bookBll.GetModel(bookId);    //如果为空，表示没有该商品  if (bookModel != null)  {  int userId = ((Model.User)Session["userInfo"]).Id;//从session中获取userid  BLL.CartManager cartBll = new BLL.CartManager();  Model.Cart cartModel = cartBll.GetModel(bookId,userId);  //如果数据库中信息为空，表示购物车没有该商品  if(cartModel !=null)  {  //商品数量+1  cartModel.Count = cartModel.Count + 1;  cartBll.Update(cartModel);  }  else  {  //添加商品到购物车  Model.Cart model = new Model.Cart();  model.User = (Model.User)Session["userInfo"];  model.Count = 1;  model.Book = bookModel;  cartBll.Add(model);  }  }  else  {  Response.Redirect("/ShowMsg.aspx?msg=" + Server.UrlEncode("没有此商品") + "&txt=" + Server.UrlEncode("商品页面") + "&url=/BookList.aspx");  }  }  } |

### 补充：bll与dal添加方法

bll中

|  |
| --- |
| public Model.Cart GetModel(int bookId, int userId)  {  return dal.GetModel(bookId, userId);  } |

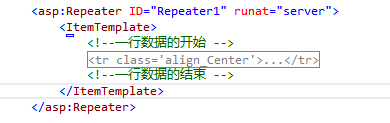
Dal中

|  |
| --- |
| public Model.Cart GetModel(int bookId, int userId)  {  StringBuilder strSql = new StringBuilder();  strSql.Append("select top 1 Id,UserId,BookId,Count from Cart ");  strSql.Append(" where UserId=@userId and BookId=@bookId ");  SqlParameter[] parameters = {  new SqlParameter("@userId", SqlDbType.Int,4),  new SqlParameter("@bookId", SqlDbType.Int, 4)};  parameters[0].Value = userId;  parameters[1].Value = bookId;  BookShop.Model.Cart model = new BookShop.Model.Cart();  DataSet ds = DbHelperSQL.Query(strSql.ToString(), parameters);  if (ds.Tables[0].Rows.Count > 0)  {  if (ds.Tables[0].Rows[0]["Id"].ToString() != "")  {  model.Id = int.Parse(ds.Tables[0].Rows[0]["Id"].ToString());  }  if (ds.Tables[0].Rows[0]["UserId"].ToString() != "")  {  int UserId = int.Parse(ds.Tables[0].Rows[0]["UserId"].ToString());  model.User = userServices.GetModel(UserId);  }  if (ds.Tables[0].Rows[0]["BookId"].ToString() != "")  {  int BookId = int.Parse(ds.Tables[0].Rows[0]["BookId"].ToString());  model.Book = bookServices.GetModel(BookId);  }  if (ds.Tables[0].Rows[0]["Count"].ToString() != "")  {  model.Count = int.Parse(ds.Tables[0].Rows[0]["Count"].ToString());  }  return model;  }  else  {  return null;  }  } |

## 显示信息

显示添加到数据库中书籍的信息到页面上

### 添加repeater用来实现列表



### 后台代码

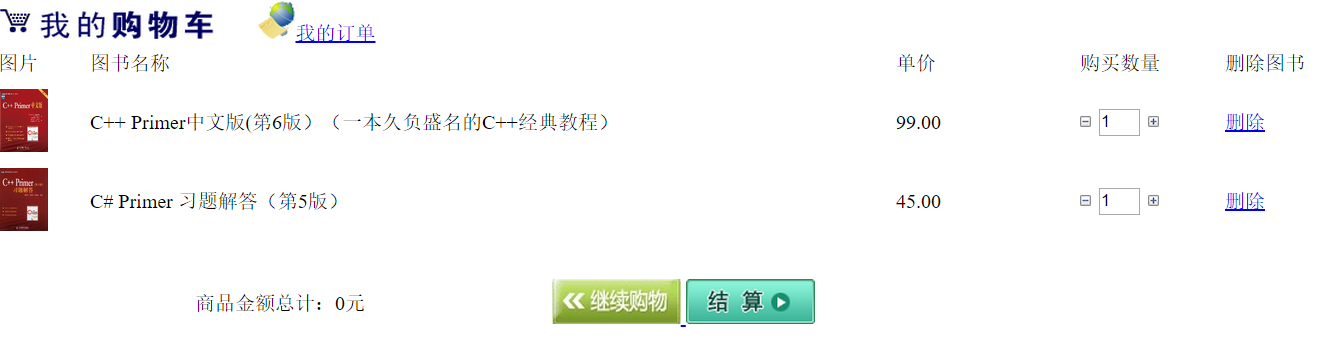
从数据库获取添加的书籍并绑定到repeater上

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  addCart();  bindCartList();//绑定信息到控件上  }  private void bindCartList()  {  BLL.CartManager bll = new BLL.CartManager();  List<Model.Cart> carts = bll.GetModelList("UserId=" + ((Model.User)Session["userInfo"]).Id);  this.Repeater1.DataSource = carts;  this.Repeater1.DataBind();  } |

### Repeater中绑定信息

|  |
| --- |
| <asp:Repeater ID="Repeater1" runat="server">  <ItemTemplate>  <!--一行数据的开始 -->  <tr class='align\_Center'>  <td style='padding: 5px 0 5px 0;'>  <img src='<%# Eval("Book.ISBN","/images/bookcovers/{0}.jpg") %>' width="40" height="50" border="0" /></td>  <td class='align\_Left'><%# Eval("Book.Title") %></td>  <td>  <span class='price'><%# Eval("Book.UnitPrice","{0:0.00}") %></span>  </td>  ~~<td><a href='#none' title='减一' onclick="changeBar('-',<%# Eval("Book.Id") %>,<%# Eval("Id") %>)" style='margin-right: 2px;'>~~  ~~<img src="Images/bag\_close.gif" width="9" height="9" border='none' style='display: inline' /></a>~~  ~~<input type='text' id='txtCount<%# Eval("Book.Id") %>' name='txtCount<%# Eval("Book.Id") %>' maxlength='3' style='width: 30px' onkeydown='if(event.keyCode == 13) event.returnValue = false' value='<%# Eval("Count") %>' onfocus='changeTxtOnFocus();' onblur="changeTextOnBlur();" />~~  ~~<a href='#none' title='加一' onclick="changeBar('+',<%# Eval("Book.Id") %>,<%# Eval("Id") %>)" style='margin-left: 2px;'>~~  ~~<img src='/images/bag\_open.gif' width="9" height="9" border='none' style='display: inline' /></a> </td>~~  ~~<td>~~  ~~<a href='#none' id='btn\_del\_1000357315' onclick="removeProductOnShoppingCart()">删除</a></td>~~  ~~</tr>~~  ~~<!--一行数据的结束 -->~~  ~~</ItemTemplate>~~  ~~</asp:Repeater>~~ |

效果：



## 增加购买数量

### 点击+、-按钮调用js方法

|  |
| --- |
| <td><a href='#none' title='减一' onclick="changeBar('-',<%# Eval("Book.Id") %>,<%# Eval("Id") %>)" style='margin-right: 2px;'>  <img src="Images/bag\_close.gif" width="9" height="9" border='none' style='display: inline' /></a>  <input type='text' id='txtCount<%# Eval("Book.Id") %>' name='txtCount<%# Eval("Book.Id") %>' maxlength='3' style='width: 30px' onkeydown='if(event.keyCode == 13) event.returnValue = false' value='<%# Eval("Count") %>' onfocus='changeTxtOnFocus();' onblur="changeTextOnBlur();" />  <a href='#none' title='加一' onclick="changeBar('+',<%# Eval("Book.Id") %>,<%# Eval("Id") %>)" style='margin-left: 2px;'>  <img src='/images/bag\_open.gif' width="9" height="9" border='none' style='display: inline' /></a> </td> |

### changeBar方法

|  |
| --- |
| <script src="js/jquery-1.7.1.js"></script>  <script type="text/javascript">  function changeBar(op, bookId, pkId) {  //获取文本框中的数量  var $control = $("#txtCount" + bookId);  var count = $control.val();  if (op == '-') {  count = parseInt(count) - 1;  if (count < 1) {  alert("数量不能为0");  return;  }  } else {  count = parseInt(count) + 1;  }  $.post("/ashx/ProcessCart.ashx", { action: "edit", count: count, pkId: pkId }, function (data) {  if (data == "yes") {  //更新价格  $control.val(count);  }  });  }  </script> |

### 一般处理程序（ProcessCart.asxh）

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  string action = context.Request["action"];  if (action == "edit")  {  int pkId = Convert.ToInt32(context.Request["pkId"]);  BLL.CartManager bll = new BLL.CartManager();  Model.Cart modle = bll.GetModel(pkId);  modle.Count = Convert.ToInt32(context.Request["count"]);  bll.Update(modle);  context.Response.Write("yes");  }  } |

### 弹出提示框

用easyui实现当数量少于1时弹出提示



## 删除

### 点击删除调用异步方法

|  |
| --- |
| <td>  <a href='#none' id='btn\_del\_1000357315' onclick="removeProductOnShoppingCart(<%# Eval("Id"),this %>)">删除</a></td>  </tr> |

异步方法

|  |
| --- |
| function removeProductOnShoppingCart(pkid,control) {  if (confirm("确定要删除吗？")) {  $.post("/ashx/ProcessCart.ashx", { action: "delete", pkId: pkid }, function (data) {  if (data == "yes") {  //从页面移除该项  $(control).parent().parent().remove();  //更新商品的价格  }  });  }  } |

### 一般处理程序

|  |
| --- |
| public void ProcessRequest(HttpContext context)  {  context.Response.ContentType = "text/plain";  string action = context.Request["action"];  if (action == "edit")  {    }else if(action== "delete")  {  int pkId = Convert.ToInt32(context.Request["pkId"]);  BLL.CartManager bll = new BLL.CartManager();  bll.Delete(pkId);  context.Response.Write("yes");  }  } |

## 计算总价

在前台计算，因为仅仅是让用户预览

|  |
| --- |
| function getTotalMoney() {  var totalMoney = 0;  $(".align\_Center:gt(0)").each(function () {  var price = $(this).find(".price").text();  var count = $(this).find("input").val();  totalMoney = totalMoney + (parseFloat(price) \* parseFloat(count));  });  $("#totleMoney").text(totalMoney);  } |

在加载、调整数量以及删除时调用。效果如：



# 订单

## 存储过程

生成订单涉及多个表，可以用存储过程

|  |
| --- |
| create proc createCustomerOrder  @orderId nvarchar(50),--订单号  @userId int,--用户编号  @address nvarchar(255),--收货人地址  @totalMoney money output--总金额  as  begin  declare @error int  set @error = 0--记录错误编号  begin transaction  --计算当前用户的订单需要支付的总价。[count]在cart表中，UnitPrice在Books表中  select @totalMoney= SUM([count]\*UnitPrice)from Cart inner join Books on Cart.BookId=Books.Id where Cart.UserId=@userId  set @error=@error+@@ERROR --记录上一条语句是否错误  if(@totalMoney is null)  begin  rollback transaction  raiserror('购物车中没有书',18,18)  return  end  --订单主表  insert into Orders(OrderId, OrderDate, UserId, TotalPrice, PostAddress, [state]) values(@orderId,GETDATE(),@userId,@totalMoney,@address,0)  set @error=@error+@@ERROR --记录上一条语句是否错误  --订单明细表  insert into OrderBook(OrderID, BookID, Quantity, UnitPrice) select @orderId, BookId,[count],UnitPrice from Cart inner join Books on Cart.BookId=Books.Id where Cart.UserId=@userId  set @error=@error+@@ERROR  --清楚购物车表  delete from Cart where Cart.UserId=@userId  set @error=@error +@@ERROR  --执行存储过程  if(@error>0)  begin  rollback tran--回滚  Raiserror ('出错',18,18)  return  end  else  begin  commit transaction  end  end |

## 订单确认页面(OrderConfirm)



### 收货人信息

收货人信息默认是用户注册时的信息。所以需要绑定数据库信息到输入框中。

#### 后置代码

|  |
| --- |
| public partial class OrderConfirm : System.Web.UI.Page  {  protected Model.User currUser;  protected void Page\_Load(object sender, EventArgs e)  {  currUser = Session["userInfo"] as Model.User;  }  } |

#### 前台绑定

|  |
| --- |
| <div style="background:#AACDED; text-align:left"><h2>填写核对订单信息</h2></div>  <div align="left">  <h2>收货人信息:</h2>  <div>  <table datasrc="" style="width:100%" id="userinfo">  <tbody>  <tr>  <td style="WIDTH: 100px" align="right">姓名：</td>  <td style="text-align:left"><input type="text" name="txtName" size="50" value="<%=currUser.Name %>" /><img src="images/cha.ico" style="display:none" width="15" height="15" /></td>  </tr>  <tr>  <td align="right">地址：</td>  <td style="text-align:left"><input type="text" name="txtAddress" size="50" value="<%=currUser.Address %>" /><img src="images/cha.ico" style="display:none" width="15" height="15" /> </td>  </tr>  <tr>  <td align="right">电话：</td>  <td style="text-align:left"><input type="text" name="txtPhone" size="50" value="<%=currUser.Phone %>" /><img src="images/cha.ico" style="display:none" width="15" height="15" /></td>  </tr>  <tr>  <td align="right">邮编：</td>  <td style="text-align:left"><input type="text" name="txtPostCode" size="50" value="<%=currUser.Mail %>" /><img src="images/cha.ico" style="display:none" width="15" height="15" /></td>  </tr>  </tbody>  </table>  </div> |

### 商品清单

将用户放在购物车中的商品再次展示

#### 后置代码中查询数据

这里我们用字符串拼接html列表。首先复制模板中的html代码，用记事本代替 “"” 为“\"”进行转义

|  |
| --- |
| <tr class=\"align\_Center\">  <td style=\"PADDING-BOTTOM: 5px; PADDING-LEFT: 0px; PADDING-RIGHT: 0px; PADDING-TOP: 5px\"><%#Eval(\"Book.Id\") %></td>  <td class=align\_Left><a onmouseover=\"\" onmouseout=\"\" onclick=\"\" href='<%#Eval(\"Book.Id\",\"book.aspx?id={0}\") %>' target=\"\_blank\" ><%#Eval(\"Book.Title\") %></a>  </td>  <td><span class=\"price\">￥<%#Eval(\"Book.UnitPrice\",\"{0:0.00}\") %></span></td>  <td><%#Eval(\"count\") %></td>  </tr> |

之后用StringBuilder进行拼接。

|  |
| --- |
| sb.Append("< tr class=\"align\_Center\">");  sb.Append(" <td style =\"PADDING-BOTTOM: 5px; PADDING-LEFT: 0px; PADDING-RIGHT: 0px; PADDING-TOP: 5px\"><%#Eval(\"Book.Id\") %></td>");  sb.Append(" <td class=align\_Left><a onmouseover =\"\" onmouseout=\"\" onclick=\"\" href='<%#Eval(\"Book.Id\",\"book.aspx?id={0}\") %>' target=\"\_blank\" ><%#Eval(\"Book.Title\") %></a></ td > ");  sb.Append("<td><span class=\"price\">￥<%#Eval(\"Book.UnitPrice\",\"{0:0.00}\") %></span></td>");  sb.Append("<td><%#Eval(\"count\") %></td> </tr>"); |

完整的加载方法见下表：

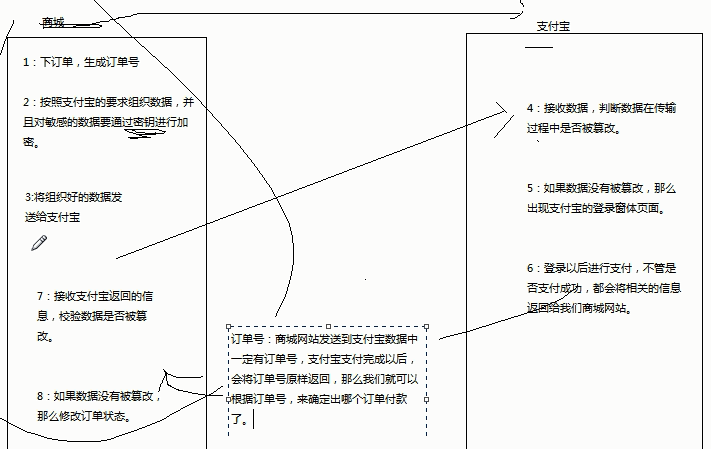
|  |
| --- |
| public partial class OrderConfirm : System.Web.UI.Page  {  ~~protected Model.User currUser;~~  protected string strHtml = string.Empty;  protected decimal totalfee = 0;  protected void Page\_Load(object sender, EventArgs e)  {  ~~currUser = Session["userInfo"] as Model.User;~~  if (!IsPostBack)  {  BindCartList();//绑定购物车中的数据到商品清单  }  }  private void BindCartList()  {  BLL.CartManager bll = new BLL.CartManager();  List<Model.Cart> list = bll.GetModelList("UserId=" + currUser.Id);  if (list.Count < 0)  {  Response.Redirect("/ShowMsg.aspx?msg=" + Server.UrlEncode("购物车中没有商品" + "&url=/BookList.aspx" + "@txt=" + Server.UrlEncode("返回商品列表页面")));  }  else  {  StringBuilder sb = new StringBuilder();  foreach (Model.Cart model in list)  {  sb.Append("<tr>");  sb.Append(" <td style =\"PADDING-BOTTOM: 5px; PADDING-LEFT: 0px; PADDING-RIGHT: 0px; PADDING-TOP: 5px\">" + model.Book.Id + "%></td>");  sb.Append(" <td class=align\_Left><a onmouseover =\"\" onmouseout=\"\" onclick=\"\" href='<%#Eval(\"" + model.Book.Id + "\",\"book.aspx?id={0}\") %>' target=\"\_blank\" >" + model.Book.Title + "</a></ td > ");  sb.Append("<td><span class=\"price\">￥" + model.Book.UnitPrice.ToString("0.00") + "</span></td>");  sb.Append("<td>" + model.Count + "</td> ");  sb.Append("</tr>");  totalfee += model.Count \* model.Book.UnitPrice;  }  strHtml = sb.ToString();  }  }  } |

#### 前台绑定

|  |
| --- |
| <hr style="border-style:dashed; width:100%; border-color:#ccc" />  <div align="left"><!--订单确定-->  <h2>商品清单:</h2>  <div>  <table datasrc="" cellspacing="0" cellpadding="1" width="98%" border="1">  <tbody>  <tr class="align\_Center Thead">  <td width="10%">图书编号</td>  <td>商品名称</td>  <td width="10%">单价</td>  <td width="8%">数量</td>  </tr>  <%=this.strHtml %>  </tbody>  </table>  </div>  </div><!--订单确定-->  <div align="right" style=" margin-right:20px"><!--总价格显示-->  <h2>  你需要支付的总价格为:<span class="price">￥<%=this.totalfee.ToString("0.00") %></span  </h2>  <br /> |

### 支付

#### 支付流程



#### 生成订单

Dal中

|  |
| --- |
| public decimal CreateOrder(int userId, string orderId, string address)  {  SqlParameter[] pars = {  new SqlParameter("@orderId",SqlDbType.NVarChar,50),  new SqlParameter("@userId",SqlDbType.Int,4),  new SqlParameter("@address",SqlDbType.NVarChar,255),  new SqlParameter("@totalMoney",SqlDbType.Money,8)  };  pars[0].Value = orderId;  pars[1].Value = userId;  pars[2].Value = address;  pars[3].Direction = ParameterDirection.Output;  DbHelperSQL.RunProcedure("createOrder", pars);  return Convert.ToDecimal(pars[3].Value);  } |

Bll中

|  |
| --- |
| public decimal CreateOrder(int userId, string orderId, string address)  {  return dal.CreateOrder(userId, orderId, address);  } |

后置代码中

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  ~~currUser = (Model.User)Session["userInfo"];~~  ~~if (!IsPostBack)~~  ~~{~~  ~~BindCartList();//将用户放在购物车中的商品再次展示~~  ~~}~~  else  {  GoPay();  }  }  /// <summary>  /// 去支付  /// </summary>  protected void GoPay()  {  //1:调用存储过程，完成下订单  int userId = currUser.Id;  string orderId = DateTime.Now.ToString("yyyyMMddHHmmssfff") + userId;  //收货人:itcast,地址:ddd,电话:123,邮编:111111  string address = string.Format("收货人:{0},地址{1},电话:{2},邮编:{3}", Request.Form["txtName"], Request.Form["txtAddress"], Request.Form["txtPhone"], Request.Form["txtPostCode"]);  BLL.OrdersManager bll = new BLL.OrdersManager();  decimal totalMoney = bll.CreateOrder(userId, orderId, address);  //2：向支付宝发送数据 |

#### 支付实现

首先，配置web.config，设置支付参数

|  |
| --- |
| <configuration>  <appSettings>  <!-- 支付参数 -->  <add key="partner" value="1"/><!—商户编号 -->  <add key="return\_url" value="http://localhost:3448/ashx/AliPay.ashx"/><!—回调商户地址（通过商户网站的哪个页面来通知支付成功！） -->  <add key="seller\_email" value="zhangsan@126.com"/><!—卖家邮箱 -->  <add key="key" value="mk@2$z"/><!—密钥 -->  <add key="payGateUrl" value="http://localhost:8899/AliPay/PayGate.ashx"/>  </appSettings><!—支付地址 --> |

创建Pay.cs文件

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  using System.Configuration;  namespace BookShop.Web.AliPay  {  public class Pay  {  //构造函数为属性赋值  public Pay(string subject, string body, string out\_trade\_no, decimal total\_fee)  {  //从web.confg中获取支付参数  partner = ConfigurationManager.AppSettings["partner"];  return\_url = ConfigurationManager.AppSettings["return\_url"];//返回  seller\_email = ConfigurationManager.AppSettings["seller\_email"];  key = ConfigurationManager.AppSettings["key"];  payGateUrl = ConfigurationManager.AppSettings["payGateUrl"];  this.body = body;  this.subject = subject;  this.out\_trade\_no = out\_trade\_no;  this.total\_fee = total\_fee;  //sign：数字签名。为按顺序连接 总金额、 商户编号、订单号、商品名称、商户密钥的MD5值。(小写值)  this.sign = Common.WebCommon.GetStringMd5(total\_fee + partner + out\_trade\_no + subject + key).ToLower();  }  //支付的方法  public string GoPayUrl()  {  return string.Format("{0}?partner={1}&return\_url={2}&subject={3}&body={4}&out\_trade\_no={5}&total\_fee={6}&seller\_email={7}&sign={8}",payGateUrl,partner,return\_url,subject,body,out\_trade\_no,total\_fee,seller\_email,sign);  }    private string partner;//商户编号 1 --  public string Partner  {  get { return partner; }  set { partner = value; }  }  private string return\_url;//回调商户地址（通过商户网站的哪个页面来通知支付成功！）1 --  public string Return\_url  {  get { return return\_url; }  set { return\_url = value; }  }  private string subject;//商品名称  public string Subject  {  get { return subject; }  set { subject = value; }  }  private string body;//商品描述  public string Body  {  get { return body; }  set { body = value; }  }  private string out\_trade\_no;//订单号！！！(由商户网站生成，支付宝不确保正确性，只负责转发。)  public string Out\_trade\_no  {  get { return out\_trade\_no; }  set { out\_trade\_no = value; }  }  private decimal total\_fee;//总金额  public decimal Total\_fee  {  get { return total\_fee; }  set { total\_fee = value; }  }  private string seller\_email;//卖家邮箱1--  public string Seller\_email  {  get { return seller\_email; }  set { seller\_email = value; }  }  private string sign;//数字签名。为按顺序连接 总金额、 商户编号、订单号、商品名称、商户密钥的MD5值。(小写值)支付时,请将上述参数以get形式传给接入地址.  public string Sign  {  get { return sign; }  set { sign = value; }  }  private string key;//密钥 --1  public string Key  {  get { return key; }  set { key = value; }  }  private string payGateUrl;//支付地址 1  public string PayGateUrl  {  get { return payGateUrl; }  set { payGateUrl = value; }  }  }  } |

完成订单后向支付宝发送数据

|  |
| --- |
| //1:调用存储过程，完成下订单  //2：向支付宝发送数据  AliPay.Pay pay = new AliPay.Pay("图书", "图书商城", orderId, totalMoney);  string url = pay.GoPayUrl();  Response.Redirect(url);  } |

#### 支付的返回参数

|  |
| --- |
| 回调商户接口地址参数说明：  out\_trade\_no：订单号。给PayGate.ashx传过去的out\_trade\_no再传回来  returncode：返回码，字符串。ok为支付成功，error为支付失败。  total\_fee：支付金额  sign：数字签名。为按顺序连接 订单号、返回码、支付金额、商户密钥为新字符串的MD5值。 |

可以在return\_url，在本文中即AliPay.ashx通过returncode判断是否支付成功，通过返回的sing与我们自己计算出的md5进行匹配判罚是否被篡改了。如果一切ok，那么就可以修改订单的状态了。

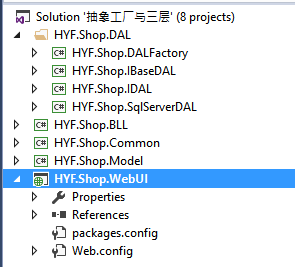
# Web.config

在文件夹下的web.config只对该文件夹下的文件起作用。例如想要单独对某个网页进行设置时，可以新建一个web.config进行设置。

# 三层与抽象工厂

抽象工厂用来解耦，业务层与数据访问层通过接口分离，改变数据库不会影响业务层

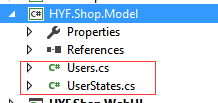
## 搭建架构



上面的HYF.Shop.IBaseDAL后来删掉了。IBaseDAL放到IDAL中了。

## Model层

### 创建类文件



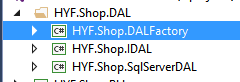
Users.cs内容

|  |
| --- |
| namespace HYF.Shop.Model  {  public class Users  {  //Id, LoginId, LoginPwd, Name, Address, Phone, Mail, UserStateId  public int Id { get; set; }  public string LoginId { get; set; }  public string LoginPwd { get; set; }  public string Name { get; set; }  public string Address { get; set; }  public string Phone { get; set; }  public string Mail { get; set; }  public UserStates userStates { get; set; }//类型变量  }  } |

UserStates.cs内容

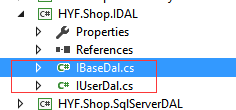
|  |
| --- |
| namespace HYF.Shop.Model  {  public class UserStates  {  public int Id { get; set; }  public string Name { get; set; }  }  } |

## DAL层



### IDAl

IDAl层是接口层



IBaseDal中定义了公共方法的泛型接口。该公共方法是所有表类型都具备的。

|  |
| --- |
| namespace HYF.Shop.IDAL  {  //where约束表示T为类，而且new创建  public interface IBaseDal<T>where T:class,new()  {  List<T> GetModelList();  }  } |

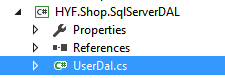
IUserDal继承了IBaseDal的接口，是users实体的接口。

|  |
| --- |
| namespace HYF.Shop.IDAL  {  public interface IUserDal:IBaseDal<Model.Users>  {    }  } |

如果说还有shop的实体那么新建一个类文件，内容如下：

|  |
| --- |
| namespace HYF.Shop.IDAL  {  public interface IShopDal:IBaseDal<Model.Users>  {    } |

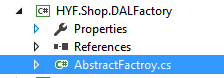
### SqlServerDAL层



UserDal类实现了IUserDal接口

|  |
| --- |
| namespace HYF.Shop.SqlServerDAL  {  public class UserDal : IDAL.IUserDal  {  string conStr = ConfigurationManager.ConnectionStrings["conStr"].ConnectionString;  public List<Model.Users> models = new List<Model.Users>();  public List<Shop.Model.Users> GetModelList()  {  string sql = "select Id, LoginId, LoginPwd,Name, Users.Address, Phone, Mail, UserStateId from Users where Id<20 ";  using (SqlConnection conn = new SqlConnection(conStr))  {  SqlCommand cmd = new SqlCommand(sql, conn);  conn.Open();  SqlDataReader reader = cmd.ExecuteReader();  if (reader.HasRows)  {  while (reader.Read())  {  Model.Users model = new Model.Users();  model.Id = reader.GetInt32(reader.GetOrdinal("Id"));  model.LoginId = reader.GetString(reader.GetOrdinal("LoginId"));  model.LoginPwd = reader.GetString(reader.GetOrdinal("LoginPwd"));  model.Name = reader.GetString(reader.GetOrdinal("Name"));  model.Address = reader.GetString(reader.GetOrdinal("Address"));  model.Phone = reader.GetString(reader.GetOrdinal("Phone"));  model.Mail = reader.GetString(reader.GetOrdinal("Mail"));  Model.UserStates userState = new Model.UserStates();  userState.Id = reader.GetInt32(reader.GetOrdinal("UserStateId"));  userState.Name = "hyf";  model.userStates = userState;  models.Add(model);  }  }  }  return models;  }  }  } |

### DALFactory层



抽象工厂根据反射创建实例

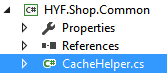
### 设置web.config

|  |
| --- |
| <configuration>  <connectionStrings>  <add name="conStr" connectionString="server=.;uid=sa;pwd=123;database=book\_shop"/>  </connectionStrings>  <appSettings>  <add key="DalAssemblyPath" value="HYF.Shop.SqlServerDAL"/>  <add key="NameSpace" value="HYF.Shop.SqlServerDAL"/>  </appSettings> |

### 创建cache类。

根据反射创建对象时，效率比较慢，每一个人访问都要根据反射创建对象，效率更加低下，可以利用cache类优化。

在common中创建CacheHelper类



在代码中编写获取和设置的方法：

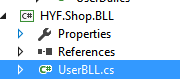
|  |
| --- |
| namespace HYF.Shop.Common  {  public class CacheHelper  {  /// <summary>  /// 根据键从缓存中取出数据  /// </summary>  /// <param name="key"></param>  /// <returns></returns>  public static object Get(string key)  {  System.Web.Caching.Cache cache = HttpRuntime.Cache;  return cache[key];  }  public static void Insert(string key, object value)  {  System.Web.Caching.Cache cache = HttpRuntime.Cache;  cache.Insert(key, value);  }  }  } |

### 实例化对象

根据web.config中的appsetings获取类型与命名空间，用反射创建对象。

|  |
| --- |
| namespace HYF.Shop.DALFactory  {  /// <summary>  /// 抽象工厂类，完成数据层中对象的创建（反射创建对象）  /// 程序集，命名空间  /// </summary>  public class AbstractFactroy  {  private static string DalAssemblyPath = ConfigurationManager.AppSettings["DalAssemblyPath"];  private static string NameSpace = ConfigurationManager.AppSettings["NameSpace"];  public static IDAL.IUserDal CreateUserDal()  {  string fullClassName = NameSpace + ".UserDal";  return CreateInstance(DalAssemblyPath, fullClassName) as IDAL.IUserDal;  }  /// <summary>  /// 通过反射创建出类实例  /// </summary>  /// <param name="dalAssemblyPath"></param>  /// <param name="fullClassName"></param>  /// <returns></returns>  private static object CreateInstance(string dalAssemblyPath, string fullClassName)  {  //根据key获取缓存，如果不为空则直接返回类型对象。否则用反射创建对象。  object instance = Common.CacheHelper.Get(fullClassName);  if (instance == null)  {  var assembly = Assembly.Load(dalAssemblyPath);//加载程序集  instance = assembly.CreateInstance(fullClassName);//创建出指定程序集下类的实例.  Common.CacheHelper.Insert(fullClassName, instance);  }  return instance;  }  }  } |

## Bll层



|  |
| --- |
| namespace HYF.Shop.BLL  {  public class UserBLL:IDAL.IUserDal  {    public List<Model.Users> GetModelList()  {  //用工程类创建对象。类型为接口类型。  IDAL.IUserDal dal = DALFactory.AbstractFactroy.CreateUserDal();  return dal.GetModelList();  }  }  } |

## UI层

|  |
| --- |
| namespace HYF.Shop.WebUI  {  public partial class test : System.Web.UI.Page  {  protected void Page\_Load(object sender, EventArgs e)  {  BLL.UserBLL userbll = new BLL.UserBLL();  var models = userbll.GetModelList();  this.GridView1.DataSource = models;  this.GridView1.DataBind();  }  }  } |

## 总结

* 根据数据库只需要更改web.config，添加类似OracleDAL.cs的实现了IUserDal的类即可。
* BLL中申明的是接口类型，用工厂类创建对象。
* 业务层和数据访问层实现了低耦合，高内聚。