在controls 添加 控制器  hellowordcontrols

  在gloabl.asax.cs  里面有设置默认的home 和默认的index方法

新建storecontroller

  3个方法

  index()  return "这里是index()";

  browse()  return "..."

  details()   ....

传参数：

  browse(string gene)

  {

     string message=HttpUtility.HtmlEncode("store.browse,gene="+gene);

     return message;

  }

穿整数

 details(int id)

 {

   string message="store.browse,id="+id;

     return message;

 }

home/index  hemel控制器的index方法

store/details/5

HttpUtility.HtmlEncode用的是xx?xx=5

普通字符串是store/details/5

---------------------------------------视频2视图和模型-----------------------------------------------

return this.view();

添加视图名称默认和方法的名称是一样的

试图引擎始终选择 razor

不使用布局和模版页

增加了views 下的home 下的 index.cshtml 由homel控制器下的index方法使用

在shared文件夹下有viewstart.cshtml打开

里面设置了 布局为默认使用layout.cshtml的格式

运行 看到index 但是查看源代码就可以看到格式

可以删除layoyut.cshtml的内容 使用自定义的新的模板页

自定义的新的模板页的内容为

<!DOCTYPE html>

<html>

<head>

    <title>@ViewBag.Title</title>

    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet"

        type="text/css" />

    <script src="@Url.Content("~/Scripts/jquery-1.4.4.min.js")"

        type="text/javascript"></script>

</head>

<body>

    <div id="header">

        <h1>ASP.NET MVC MUSIC STORE</h1>

        <ul id="navlist">

            <li class="first"><a href="/" id="current">Home</a></li>

            <li><a href="/Store">Store</a></li>

        </ul>

    </div>

    <div id="main">

        @RenderBody()

    </div>

    <div id="footer">

        built with <a href="/">ASP.NET MVC 3</a>

    </div>

</body>

</html>

美工不好，修改美工

覆盖content文件夹  来存放自定义的图片和样式表

public string details(int id)

{

string message="store,details,id="+id;

}

public string browse(string genre)

{

string messsage=HttpUtility.HtmlEncode("store.browse,genre="+genre);

return message;

}

使用此方法预处理用户的输入，这些能阻止用户向视图中用连接注入js或者html标记

比如store/browse?genre=<scpipt>window.location="www."></>

添加新models  流派类 genre.cs

public string name{get;set;}

添加新类 专辑类 album

public string title{..}

 对象属性，表示专辑属于哪一类（摇滚。伤感，回忆）

 punlic      genre genre{..}

public ActionResult Details(int id)

        {

            //定义变量

            var album = new Album {title = "Album " + id };

            //转到视图

            return View(album);

        }

var表示定义变量，var可以不知道变量什么类型，编译器会把等号右边的类型

赋给var

添加视图 使用布局

@model 传过来的模型是什么类型的

@model musicstoreproject.models.album;

<h2>专辑的标题:@model.title</h2>

强类型  。。。models

details(string genre)

var model=new genre(){name=genre};

return this.view(model);

修改主方法为public actionresult inex()

{

vargenres=new list<genre>

{

 new genre{name="111"},

 new genre{name="111"},

 new genre{name="111"}

}

}

添加视图 显示全部的数据

集合对象 传递了许多个 genre对象

强类型  支持的框架模版 选择list 使用模版

@model IEnumer。。。。等等底下的以及edit 等都是自动生成的

@model.count 个数

《ul》

  @foreach(var genre in model)

  {

    //<li>@genre.name</li>读取列表的名字

    //增加页面之间的链接

    《li><a href='/store/browse?genre=@genre.name'>@genre.name</li>

  }

</ul>

在browse.html添加回到首页的超链接

@html.Actionlink("回到首页","index");回到首页 即index方法

和<a href='/store/browse?genre=@genre.name'>@genre.name</li>作用一样

@html.Actionlink("提示内容","action方法名",new {genre=genre.name});给方法传递的参数 匿名对象

-------------------------------数据访问---------------------------------------------------------

对性关系映射技术

安装ef 4.1

安装ef后会在引用文件中找到该ef程序集。。。.dll

我们将使用ef支持进行查询和更新数据库中的数据

ef是一个灵活的进行数据访问的对象关系映射api，允许开发人员使用面向对象的干事对数据库中的数据进行查询和更新

新建类 artist艺术家类

属性2个 artistId{..}

        name

ef约定1：类名加Id或者Id将会ef被自动设置为主键

增加专辑的album的内容

albumid  专辑id

genreid

artistid

title

price

albumarturl

genre

artist

ef约定2：有艺术家类的属性，则会自动创建主外键的关联

更新genre的内容

genreid

name

description

public list<album> albums{get;set;}

因为有集合 所以会有主外键的关联。

在webconfig的configuration中添加连接串

名称是自己起的 链接字符串  使用哪种数据库

<connectionStrings>

    <add name="MusicStoreEntities"

    connectionString="server=.;database=musicstore;uid=sa;pwd=123"

    providerName="System.Data.SqlClient"/>

  </connectionStrings>

添加新类的名字和字符串名字一样 MusicStoreEntities：DbContext

DbContext数据库上下文  using命名空间不报错

泛型的 对应数据库中的表名一样

public Dbset<album> albums{get;set;}

genre         genres

artist        artists

---------------------下面开始添加数据

复制sampledata.cs到models下面

添加新的类,往数据库中添加数据 sampledata:DropCreatedatabaseIfModelChanges<musicstoreentities>

发现模型变化了，则删除并重建数据库

seed（）重写基类的方法以提供样本数据

actionstart()当程序第一次被访问时，会执行的方法

添加进去一句话

system.data.entity.database.setInitializer(new mvcmusicstore.models.sampledata());

才可以每次都在检测模型变化后重建数据库

------------------查询数据库的的的话

在控制器store中

用于访问数据库的私有成员

private musicstoreentities storedb=new musicstoreentities();

更新indexaction查询数据库为

var genres=soredb.genres.tolist();

return this.view(genres);

查看数据库的话 多了一张额外的表来跟踪对表的修改

在数据库的 数据库关系图 确定  右键 新建数据库关系图

关系图  选择三张表 添加

----------修改storecontroller的browse(string genre)方法为

var genermodel=storedb.genres

.include("albums")  提前加载 分类里面的唱片的集合

.single(g=>g.name==genre);分类 找genres的genre=genre的分类

return this.view(genremodel)

---------修改browse.cshtml为：

添加

<h2>browseing genre:@model.name</h2>

<ul>@foreach(var album in model.albums)

{

 <li>@html.actionlin（album.title,"details",new{id=album.albumid} )@album.title</li>

}</ul>

------------修改details（int id）方法

找到指定id的  details

var album=storedb.albums.find(id)

return this..view(album);

至此，可以看到效果为：看到整个流派，每个流派的列表，每个列表的详情等

-------------------------通过支架创建编辑表单------------------------------------

按f12  把。。。文本修改为ie8标准

创建storemanger控制器  支架为：包含读写操作和视图的控制器使用ef

模型类用 album

数据上下文类为： musicstoreentities(mspm)

添加

-----------------下面开始解析storemanger控制器

create创建新专辑 post 和get 用【httppost】来区别

dispose回收资源 deleteconfirmed确认删除

index方法  找到视图 idnex。cshtml

删除<tr><td>albumurl</td></tr>

删除@html....

则删除了艺术家的那一列

火狐浏览器 firebug

 jquery-1.4.4.min.js没有拿到

找到模板页lauout.cshtml将其修改为1.5.1.min.js即可正常edit

----------开始介绍create

create参数类型是专辑的类型

模型绑定 参数为：自动创建一个空的模型对象

isvalid  检查合格

     add添加到数据库

     savechanges()写入数据库

     redirecttoaction("方法名")重定向到主页面

若检查不合格 则返回原来的添加界面

SelectList(IEnumerable, String, String, Object) 使用列表的指定项、数据值字段、

数据文本字段和默认选定的值   来初始化 SelectList 类的新实例。

参数为：列表的数据来源 列表的名称 显示的内容  默认选中是谁，默认选中它上次选择的添加的字段值

viewbag 课添加任何属性，任何名称 向视图传递多个信息 可添加任何属性

@model.。。只能是一个 viewbag可传递多个值

向视图：传递2个下拉列表数据 +一个模型

使用htmlhelp 显示下拉列表

@html.dropdowmlist 参数 （从哪里获取显示用的列表， 哪一个值需要被预先选中）

第一个参数先从模型对象找值，找不到再从其他带的model带的找

创建时 默认没有选中的 所以下拉列表默认没有选中任何东西 string empty

修改时 默认选中了

----------------编辑edit

删除 delete  直接删除

【post，解决同名的问题的anctionname[delete]】

deleteconfirmed 删除的确认

dispose回收资源

通过[httppost]进行标注 是post还是get的

[httppost,actionname("name")]指定anction的实际名字 方法名字和action名字不一样

使用自定义lhelper截断长字符的方法

@helper truncate(string input,int length)

{

 if(input.lentgh<=length){@input}

 else {@input.substring(0,lentgh)<text>...</text>}

}

添加到indx.cshtml的第二行

在代码使用方法

修改<td>@html.displayfor(....)为  <td>@truncate(item.Artist.title,25)</td>

添加文件夹helpers  添加类 htmlhelps

此函数添加在里面可以方便很多页面都是用截取字符串的方法

using system.web.mvc;           this+ 被扩展的对象 +helper

public static string truncate(this.htmlhelper helper,string input,int length)

{

 if(input.lentgh<=length){@input}

 else {@input.Substring(0,lentgh)<text>...</text方法，

必须引入视图

防止每个视图都using 在配置文件页面使用此方法必须引入空间

@using musicstroe.helpers;

<td>@html.trucncate(viewbag.message as string ,8)</td>

或者 <td>@tuuncate(item.Artist.title,25)</td>

至此，对数据库的增删改查都已经做到了

------------------------------通过特性进行验证----------------------------------------------------

写的代码类去创建的数据库 所以是代码优先 code-first

在genre类里面写的类 public strign description与数据库的字段一样

下面了解什么是 代码优先 数据库优先  模型优先

---------------------------------数据库优先

新建项目

在数据库中添加表

vs2012 新建 数据 ado.net实体数据模型 musicstore.edmx

从数据库生成 数据库是新的 新建连接 使用sql server身份验证  测试连接

选择新建的数据库  确定  下一步  表 打钩 完成

-----------------------------模型优先

添加新建  ado.net实体模型   musicstore.edmx

空模型，而不是从数据库生成等

拖拽“实体”  邮件 添加属性等

为专辑表单增加模型验证

在命名空间 system.componentModel.DataAnnotation;

required 必须提供内容

displayname 显示名- 定义表单字段的提示名称

stringlength  字符串长度

range  范围

bind 绑定 列出请求半数绑定到模型时 包含和不包含的字段

scaffoldcolumn  支架咧 在 编辑表单的时候 需要隐藏起来的字符

原来的专辑页面是没有检查的,修改如下

using system.componentmodel.DataAnnotation;

[Bind(Exclude="albumid")]

public class album

{

  [scaffoldcolumn(face)]

  pulic int alubmid(get;set;)

  [displayname[artist]] public int artistid{..}

  [required(errormessage="错误信息提示")]

  [stringlengrh(16)]

   //文本字段可以不生成全部是varcharmax长度的了

   //可以检测用户输入的合法性

  public string title{get;set}

  [range(1,100,errormassage="ssss")]

  [displayname("显示在界面上的第一行的名称")]

  [scaffoldcolumn(false)]支架生成视图时，不考虑albumid列

  pulic int albumnid{set;get}

  [bind(exclude="alumid")]数据绑定 不考虑专辑的id的属性

 }

发现数据库被代码修改后 会删除并自动重建 速度比较慢 但是数据库界面不用使用及修改

labelfor(model=>model.albimurl)为它生成显示的标签，默认和属性同名

displayfor

使用jquery客户端脚本验证,有错误，重新改正之后，错误提示立马消失

web.config中有 不明显的javascript方式启用

               客户端验证 启用

-----------------------------------07成员管理和权---------------

任何人都可以进行增上改查，也不用接触到数据库

在framework v4.0中找到 asp.net\_regsql.exe

工具帮助我们在数据库创建会员管理表

C:\Windows\Microsoft.NET\Framework\v4.0.30319 找到 双击  启动

配置（第一个） 下一步 选择数据库 下一步 确认

刷新页面 可看到数据库中增多了许多表

用户表 角色表等

在web.config中配置 成员管理的使用

machine.config  web.config    你的项目里面的web.config优先级升高，可覆盖

C:\Windows\Microsoft.NET\Framework64\v4.0.30319 \ config文件夹下的web.config  网站用到的参数

machine.config  所有的配置参数适用于所有的.net

打开machine.config找到name="AspNetSqlMembershipProvider"

membership>

<providers>

<add name="AspNetSqlMembershipProvider" type="System.Web.Security.SqlMembershipProvider, System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" connectionStringName="LocalSqlServer" enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="true" applicationName="/" requiresUniqueEmail="false" passwordFormat="Hashed" maxInvalidPasswordAttempts="5" minRequiredPasswordLength="7" minRequiredNonalphanumericCharacters="1" passwordAttemptWindow="10" passwordStrengthRegularExpression=""/>

</providers>

</membership>

复制到当前目录下的web.config

找到name="aspnetsqlroleprovider"

复制到当前目录下的web.config

不希望使用app\_data文件夹下的数据库 所以把connectionstringname=".."替换为""

musicstoreentities;再把前面的配置删除掉

 <clear>  在<add前面加

密码长度至少为7

非数字或者字母的其他字符至少有一个

以前的清除  加上enable=""true

打开项目 最后一项asp.net配置

打开安全 可以再此处创建用户 创建和管理角色等

添加之后可以再数据库中看到添加的结果

另一种配置方法 右键 重新建一个项目otherstore

internet应用程序 确定 生成了一个网站项目

在此web.config中已生成了许多配置 \复制没有的到web.config

没有的是

<profile>

在此处还生成了用户创建界面

accountctrooler.cs 复制到controllers里面 修改命名空间

accountmodels.cs也复制到models里面 修改命名空间

添加account文件夹到views文件夹下

先创建角色，再创建用户，并为用户选择角色adminstrator

在storemanger页面中加入【authorize(role="adminstrator")】

即可限制用户的访问后台数据管理界面

-------------------------------------------购物车-------------

--------------数据库05挂接上来搭建到08上

数据库，附加数据库

报错，拒绝访问

复制路径 到文件夹 点击 继续 找到文件

点击属性 安全 继续

添加 立即查找 users

可以访问 允许

按此方法也修改日志文件

手动删除数据库的时候要关闭现有链接，否则会等待操作完成后才删除

models下新建类cart

using system,compoentmodel.data...

【key】说明了 recordid为此表的主键

pulic int recordid{...}

cartid 区分不同的购物车

albumid专辑的id

count

datacreated

pulic virtual album album{}通过专辑对象 知道专辑多少钱 什么名

models添加orderdetail类

public int orderdetailid

 int orderid

int albumid

int quality

public decimal unitprice

pulic virtual album album{get;set;}

public virtual order order{..}

models添加order类

int orderid

string username

firstname

lastname

address

city

state

postalcode

country

phone

email

total

orderdate

pulic  list<oederdetail>orderdetails{get;set;}

下面把我们的musicstoreentities更新 来往数据库中添加表

添加 pulic Dbset<car>cars{get;set;}

publc DbSet<order>orders{}

orderdetail>orderdetails{setl;get;}

下面 我们在models文件夹创建shoppingcar类 处理cart类的数据访问

 另外 它还需要处理在购物车中增加或者删除项目的业务逻辑

shoppingcart提供如下方法：

addtocart 将专辑作为参数添加到购物车

removefromcart 通过专辑的表示从用户的购物车将这个专辑的数量减1

emptycart 删除用户的所有项目

getcartitems 获取购物项目的列表用来显示或者处理

getcount 获取用户中专辑的数量

gettotal总价

createorder 将购物车转化为结账过程中的订单

getcar 静态的方法 获取用户的购物车对象 它使用getcartid方法类读取保存当前session中的购物车标示

getcarid方法需要httpcontextbase以便获取当前的session

详细的介绍此类要看代码中

创建视图模型文件夹

创建视图模型类 shoppingcartviewmodel

创建shoppingcartremovemodel

---------创建index。cshtml

@model MvcMusicStore.ViewModels.ShoppingCartViewModel

@{

    ViewBag.Title = "Shopping Cart";

}

<script src="/Scripts/jquery-1.7.1.min.js" type="text/javascript"></script>

<script type="text/javascript">

    $(function () {

        // Document.ready -> link up remove event handler

       -------点击了calss为这个的元素时

        $(".RemoveLink").click(function () {

            // Get the id from the link

            -------$(this).attr(key); 获取节点属性名的值，相当于getAttribute(key)方法

详细出处参考：http://www.jb51.net/article/41922.htm

            var recordToDelete = $(this).attr("data-id");

            if (recordToDelete != '') {

                // 去RemoveFromCart方法里面传参数id=recordToDelete

                $.post("/ShoppingCart/RemoveFromCart", { "id": recordToDelete },

                    function (data) {

                        // Successful requests get here

                        // Update the page elements

                        if (data.ItemCount == 0) {

=0  找到这一行 慢点 删除掉

                            $('#row-' + data.DeleteId).fadeOut('slow');

                        } else {

                            修改元素的text

                            $('#item-count-' + data.DeleteId).text(data.ItemCount);

                        }

                       修改页面中元素的总价 提示信息 数量等信息

                        $('#cart-total').text(data.CartTotal);

                        $('#update-message').text(data.Message);

                        $('#cart-status').text('Cart (' + data.CartCount + ')');

                    });

            }

        });

    });

    function handleUpdate() {

        // Load and deserialize the returned JSON data

        var json = context.get\_data();

        var data = Sys.Serialization.JavaScriptSerializer.deserialize(json);

        // Update the page elements

        if (data.ItemCount == 0) {

            $('#row-' + data.DeleteId).fadeOut('slow');

        } else {

            $('#item-count-' + data.DeleteId).text(data.ItemCount);

        }

        $('#cart-total').text(data.CartTotal);

        $('#update-message').text(data.Message);

        $('#cart-status').text('Cart (' + data.CartCount + ')');

    }

</script>

<h3>

    <em>Review</em> your cart:

</h3>

<p class="button">

    @Html.ActionLink("Checkout >>", "AddressAndPayment", "Checkout")

</p>

<div id="update-message">

</div>

<table>

    <tr>

        <th>

            Album Name

        </th>

        <th>

            Each Price

        </th>

        <th>

            Quantity

        </th>

        <th>Operating</th>

    </tr>

    @foreach (var item in Model.CartItems)

    {

        <tr id="row-@item.RecordId">

            <td>

参数为提示（显示）信息 方法名称 控制器名称 参数

                @Html.ActionLink(item.Album.Title, "Details", "Store", new { id = item.AlbumId }, null)

            </td>

            <td>

                @item.Album.Price

            </td>

            <td id="item-count-@item.RecordId">

                @item.Count

            </td>

            <td>

data-id 是用于取id的元素空间，可以自己自定义

                <a href="#" class="RemoveLink" data-id="@item.RecordId">Remove from cart</a>

            </td>

        </tr>

    }

    显示的最后一行

    <tr>

        <td>

            Total

        </td>

        <td>

        </td>

        <td>

        </td>

       这个模型就2个参数 1个是CartTotal 另一个是cartitems

        <td id="cart-total">

            @Model.CartTotal

        </td>

    </tr>

</table>

----------修改details页面

@model 音乐商店.Models.Album

@{

    ViewBag.Title = "Details";

}

<h2>Album: @Model.Title</h2>

<p>

<img alt="@Model.Title" src="@Model.AlbumArtUrl" />

</p>

<div id="album-details">

 <p><em>Genre:</em>@Model.Genre.Name</p>

 <p><em>Artist:</em>@Model.Artist.Name</p>

  <p><em>Price:</em>@String.Formate("{0:F}",model.Price)</p>

<p class="button">

//参数为提示信息 方法名称 控制器名称 参数

@Html.ActionLink("add to cart","AddTocart","ShoppingCart",new {id=model.AlbumId},"")</p>

</div>