1创建mvc项目

2020年1月29日

20:00

MVC简介：

web form 缺点：

1、前后端高耦合

2、由于视图状态的原因，会增加网络带宽占用

适用场景：适用控件，开发效率高，不在乎带宽，适用于局域网web应用程序开发，如企业内部管理系统

而MVC适用于基于因特网的高并发的大中型web应用程序开发，如电商，门户网站

web——asp.net web 应用程序(.net framework)——MVC

# 项目创建

# 1、选择MVC项目

# 2、空项目要勾选MVC

# 一、文档结构

/App\_Data:本地化的数据文件或数据库文件

/App\_Start:多个静态配置类。执行程序的初始化任务

/Content:静态内容，如CSS

/Controllers:控制器文件，后缀名.cs

/Models:数据模型对象的文件类。后缀名.cs

/Scripts:js、jquery等

/Views:视图文件

/Global.asax:全局应用程序类文件

/Web.config:应用程序配置文件

# 二、控制器基础

命名规范

1、以Controller结尾，如HomeController

2、控制器类的基类是Controller

3、至少包含一个返回值为ActionResult的public方法，称为操作方法，例如Index()方法就是一个操作方法。

4、HomeController对应视图Home，Index()\About()\Contact()方法是相应的.cshtml页面的数据操作方法

步骤

1、添加控制器Controllers文件夹-右键，添加-控制器-mvc5控制器

//也可以添加包含视图的mvc5控制器

2、添加视图，在创建的控制器相应的方法上右键-添加视图

路由格式为 <http://域名/控制器/视图> 首页可以省略home和index，控制器名不包含controller

路由配置在/App\_Start/RouteConfig.cs

提示：

1、在打开的页面文件，f5可以直接运行显示当前页

2、视图源码开头部分包含了@符号，是典型的Razor语法，允许C#与HTML混编

View()方法调用默认同名视图

# 三、视图基础

模板，可以选择模型，然后选择基础模板

基础模板分类

Create/Delete/Details/Edit/Empty/Empty(without model)/List

Empty(without model)常用 唯一不需要选择模型的基础模板

List数据管理常用，包含了指向操作的链接， 创建/编辑/删除

# 四、关于母版

1、\_ViewStart.cshtml

Views文件夹下\_ViewStart.cshtml文件，特点是预加载，任何页面加载前先检查\_ViewStart.cshtml页面是否存在，存在则先加载，它只有1句内容

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

//每个页面也可以设置这个属性单独指定使用的布局页或者不使用布局页

@{

Layout = null;

}

意思是Views目录下所有的视图，以~/Views/Shared/\_Layout.cshtml为布局模板页面

2、\_Layout.cshtml

Views/Shared文件夹下，母版页面

它有两个特别的Razor声明

@RenderBody()：子页面的内容会出现在@RenderBody()所在的位置。他只能在body中出现，且只能出现一次。不需要参数

@RenderSection()：块占位符，以指定名称的形式，将子页面中定义的代码块填入到@RenderSection()所在的位置。需要参数，可以出现多次，任何位置。

关于多个母版，可以再视图中从新指定母版路径，母版可以放在相应的视图文件夹下

ViewBag.Title = •Index•.• I 
Layout= • /Views/User/_ LayoutAd min.cshtmt 

页面中写链接

<li>@Html.ActionLink("联系方式", "Contact", "Home")</li>

显示时间

@DateTime.Now.Year

绑定样式和js

head中

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

页面后

@Scripts.Render("~/bundles/jquery") //绑定jquery

@Scripts.Render("~/bundles/bootstrap") //绑定bootstrap

@RenderSection("scripts", required: false) //表示在子页面写script脚本需要放在

//子页面

@section scripts{

<script>

</script>

}

2数据传递

2020年1月29日

21:23

# 一、从Controller向View传递数据

三种方式，ViewData\ViewBag和TempData

1、ViewData属性 字典类型(不可以跨请求)

ViewData["Message"]="信息"; //对象类型

页面调用@ViewData["Message"]

2、ViewBag属性 dynamic类型(不可以跨请求)

ViewBag.Message="信息"; //对象类型

页面调用@ViewBag.Message

特点ViewData/ViewBag只能在相对应的视图传值，并且，可以互相调用

3、TempData属性 字典类型

用于跨Action()传递数据。

特点：数据保存在Session中，但是控制器每次获取TempData后，会清除数据

注意：同一个页面的多次调用被认为是1次请求。

TempData的生命周期也结束，只能读取一次，可以在视图间传值。

public ActionResult Index(){

TempData["Message"]="信息";

return View();

}

public ActionResult About(){

return View();

}

在About.cshtml中调用

@TempData["Message"]

4、存储和遍历实体集

存储

MyDBEntities db=new MyDBEntities();

public ActionResult Index(){

ViewData["users"]=db.users.Tolist();

//ViewBag.users=db.users.Tolist();

return View();

}

页面调用//注意数据类型转换

@foreach (var item in ViewData["users"] as List<Models.Users>){

@item.username;

}

# 二、从View向Controller传递数据

使用Request对象来完成 //取得的为字符串类型

index.cshtml页面中

<form method="post" action="Save">

<input type="text" name="username">

<input type="submit" value="提交">

</form>

Save方法中

[HttpPost]

public ActionResult Save(){

string name=Request["username"]; //这种传值其实不需要使用HttpPost声明

ViewBag.Name=name;

}

Save.cshtml页面中调用

@ViewBag.Name

简单数据类型绑定

上述实例可以改写为

[HttpPost]

public ActionResult Save( string username){//参数与页面中name一致可获取页面绑定值

…

}

复杂数据类型绑定

[HttpPost]

public ActionResult Save( User user){//页面中input的字段name属性要和user的属性名称一致

…

}

razor语法注释 @\*\*@

视图View的参数

object ,任何数据类型

如果是字符串，会认为是某个view的名字

3 模型绑定

2020年1月29日

21:23

模型绑定使得控制器中可以直接获取视图或url传递来的数据，并且可以自动转换为模型对象。

# 一、手动获取表单数据

Request["Id"] Id为Form表单中input元素的name属性

# 二、模型绑定

1、模型（类）绑定

表单元素的name属性和对象属性的名称必须相同。

实例

[HttpPost]

public ActionResult Index(Product p){

ViewBag.Info=p.Name+p.Num

return View();

}

表单中的input的name属性必须对应模型的对应属性name,num

这种隐式类型的绑定，由框架默认的模型绑定器DefaultModelBinder类自动实现。其通过四种途径获取绑定的值，优先顺序如下

Request.Form

RouteData.Values

Request.QueryString

Request.Files

2、基本类型绑定

[HttpPost]

public ActionResult Index(string name,int? num){

ViewBag.Info=name+num

return View();

}

表单中的input的name属性必须对应参数名name,num

注意：使用int？的好处是，如果输入的不是数字，则num=null。

3、复杂类型绑定

集合，类都可以，类也可以直接定义在Controller里面

复杂类型绑定分为

1)复杂基本类型绑定（比如字符串集合、数字集合）

index页面

8<form method—"post" action—"Send"> 
@foreach (var item in ViewBag. emails)l 
< input class—"form—control" type—"text" name—"email" value—"@item 
< input class—"btn btn-primary" type—"submit" / > 
</form> 

// GET: Email 
public ActionResult Index ( ) 
var emails = new string[l 
ViewBag. emails = 
emai Is ; 
return View() ; 
"mht@qq. com , 
public ActionResult Send (List<string> email) 
ViewBag. emails = 
emai l; 
return View() ; 

2)复杂模型类型绑定（比如学生集合）

前台form表单input name属性必须使用集合遍历的形式

ViewBag. Title = "Index" , 
8 (form method;" post" 
int index = 0:} 
Cforeach (var item in ViewBag. stus as IList<Dem01. Models. Student>) 
(input type= text" name= stus C@index). uname" 
<input class="form—control" type="text" 
(hr 
index++; 
var students 
@for (int i 0; 
(input class 
= ViewBag. stus as IList<Dem01. Models. Student>; 
i < students. Count; i++) 
name=" stus . uname" 
name= stus [ail. email" 
value= aitem. uname 
email " 
uname" 
email" 
(input 
form—control " 
(hr / > 
(input btn—primary 
type—text" 
placeholder=" type:" text" 
type;" submit" 
value— 

// GET: Email 
public ActionResult Index 
var stus = new Student 1] { 
new Student ( uname email 
new Student , 
new Student () , 
new Student , 
new Student 
ViewBag. stus 
stus; 
return View() ; 
"xbaqq. com 
public ActionResult Send stus) 
ViewBag. stus = stus; 
return View() , 

4、文件上传

使用HttpPostedFileBase对象

[HttpPost]

public ActionResult Upload(HttpPostedFileBase file){

if(file!=null){

if(file.ContentLength==0){retrun View();}

else{file.SaveAs(Server.MapPath("~/UploadFile/"+file.FileName));}

}

return View();

}

在Upload.cshtml中

<form action="/Home/Upload" method="post" enctype="multipart/form-data">

<input type="file" name="file">

….

<form>

必须使用enctype="multipart/form-data"否则不会传文件，只会传文件名

4表单和html辅助方法

2020年1月29日

21:23

MVC所有的HtmlHelper扩展方法都在System.Web.Mvc.Html命名空间中。

完全可以把它理解为一种语法，替代html的标签

# 超链接

@Html. ActionLink (")QY$2fCJ", "About") 
<br / > 
(a href=" 
@Html. ActionLink ("Åi%é2ffJ", "Contact", "Home") 
<br / > 
(a href—" / Home/ About" 
@Html. ActionLink "Contact", 
"Home", new { style — 
(a href—" / Home/ About/ 1" 
"color: red" } ) 
@Html. ActionLink ("Åi%ékfCJ", "Contact", "Home", new { id 
— I ) , new { style — 
</div> 
"color : red" 

CHtm1. ActionLink ( 
( 104) v grR)MvcHtrnlString HtmlHelper.ActionLink(string link Text, string actionName, string controllenName, object routeVaIues, object htmlAttributes) 
ae tionNa me: 

# 

# 

# 表单

@using (Html.BeginForm("Search", "Home", FormMethod.Post)) //表单

{

@Html.TextBox("text1",null, new { @class = "class1" }) //文本框

@Html.RadioButton("sex", "男", true) //单选

@Html.RadioButton("sex", "女")

@Html.CheckBox("hobby1", true, "读书") //多选

@Html.Hidden("pwd") //隐藏

@Html.Password("pwd")

@Html.TextArea("longtext1", "默认内容", 5, 30, null) //文本域

@Html.DropDownList("addr", new SelectList(new[] { "北京", "上海", "武汉" })) //下拉

@Html.DropDownList("addr",new SelectListItem[] { //下拉2

new SelectListItem { Text="北京",Value="beijing"},

new SelectListItem { Text="北京",Value="beijing"},

new SelectListItem { Text="北京",Value="beijing"}

})

}

@{Html.BeginForm()}

@{Html.EndForm()}

# Html.Partial()方法载入分部视图

1、创建分部视图

右键点击/Views/Shared选择添加-MVC5分部页(Razor)

2、页面调用

@Html.Partial("\_Top") //引用，\_Top为分布页名

输出纯文本可以用@:\

5表单验证

2020年1月29日

21:24

# 一、使用jquery进行表单验证

使用了html辅助方法的页面，同样可以使用jQuery进行表单验证

引入jquery-添加验证方法-调用验证方法

实例

<script>

$(function(){

//当表单提交时，绑定checkForm方法

$("form").bind("submit",checkForm);

});

function checkForm(){

if(checkUser()&&checkPwd())

return true;

return false;

}

function checkUser(){

…

return false;

}

function checkPwd(){

…

return false;

}

//页面中调用方法new {onblur="checkUser()"}

<script>

@using (Html.BeginForm("index","home",FromMethod.Post)){

<span>用户名</span>@Html.TextBox("uname","",new {onblur="checkUser()"})

}

//注意，给辅助方法写属性需要使用new

实例

Html.BeginForm("index","home",FromMethod.Post,new{id="",name="",onblur="",@class=""})

class前需要使用@

# 二、使用JQUERY Validate插件验证

先依次导入

jquery-3.4.1.min.js

jquery.validate.js

query.metadata.js

jquery.validate.messages\_cn.js 用以支持中文信息

<script src="~/Scripts/jquery-1.8.3.min.js"></script>

<script src="~/Scripts/jquery.validate.js"></script>

<script src="~/Scripts/jquery.metadata.js"></script>

<script src="~/Scripts/jquery.validate.messages\_cn.js"></script>

实际操作，以上也可以写为

**@Scripts.Render("~/bundles/jquery") //通常在布局页已经导入**

**@Scripts.Render("~/bundles/jqueryval")**

@section scripts{

@Scripts.Render("~/bundles/jqueryval")

<script>

$("#formstudent").validate({ //注意不是ta,是te结尾

rules: {

login\_name: { required: true },

login\_pwd: { required: true,rangelength:[3,12] }

},

messages:{

login\_name: { required: "账号不能为空"},

login\_pwd: { required: "密码不能为空",rangelength:"长度3-12位" }

}

})

</script>

}

\*\*\*在布局页有

@RenderSection("scripts", required: false)//false表示不是每个页面都需要包含这个代码块

然后在需要的页面

**@section scripts{**

**@Scripts.Render("~/bundles/jqueryval")**

**}**

**注意**

@RenderSection "scripts", equired: false)// false? 
@sectio scripts 
@Scripts.Re nder(" 'V/bundle s/jqueryval") 

**红色部分是代码块的名字，布局页和引用布局的页面要一致**

# 三、模型注解验证

引入命名空间System.ComponentModel.DataAnnotations

注解是描述性的语法，用来修饰类，方法或者属性，字段的特性 放在[]里面，写在要修饰的内容前面

（特性必须通过反射读取，建议学学反射）

之前的知识中我们已经用过[serializable]\[HttpPost]等

模型中添加注解

在模型字段前，通过注释，添加验证

[Required(ErrorMessage="错误信息")]

[StringLength(50,ErrorMessage="错误信息")]

[StringLength(12,MinimumLength =3, ErrorMessage = "密码长度3-12位")]

[Range(1,50,ErrorMessage="错误信息")]

[Range(typeof(decimal),"1.00","50.00",ErrorMessage="错误信息")]

[Compare("表单字段",ErrorMessage="错误信息")]

[RegularExpression("pattern",ErrorMessage="错误信息")] 匹配正则表达式

**设置别名[Display(Name="")]**

引入命名空间using System.ComponentModel.DataAnnotations;

//模型注解可以使用服务器端或者脚本验证

1、服务器端控制器视图动作中

if(ModelState.IsValid){

ModelState.AddModelError("","用户名密码不匹配")；

}

2、视图中

@Html.ValidationSummary()方法集中显示错误提示信息

@Html.ValidationMessage("id") 显示相应字段的错误提示信息

如果是强类型（后面会讲也可以写成）

@Html.LabelFor(u => u.Account)：

@Html.TextBoxFor(u => u.Account, new { @class = "form-control", title = "英文账号" })

@Html.ValidationMessageFor(u => u.Account)

3、后台判断验证通过 Model.isValid

6 强类型

2020年1月29日

21:24

# 1、强类型

在视图中，使用@model声明具体类型。

语法，视图中

@model 模型对象（集合）

与普通类型的却别就是

@model 数据类型 定义好model的数据类型

用的时候通过Model进行调用，会有语法提示。

（注意大小写）

做法

控制器中

public ActionResult Index()

{

using (fashionshoppingDBEntities db = new fashionshoppingDBEntities())

{

//查询所有商品信息

IEnumerable<Product> plist = db.Product.ToList();

return View(plist);//注意参数

}

}

前面第三章的做法，并不使用参数。

视图中

@model IEnumerable<Demo.Models.Product>

@foreach (var product in Model)//强类型模型对象

{

<div class="product">

<div class="pic"><a href="#" target="\_blank">

<img id="Image3" src="/show\_imageurl/show\_@product.imageid" style="height:300px;width:225px;border-width:0px;" /></a>

</div>

<div style="text-align:center;"></div>

<div style="text-align:center;font-weight:bold"> @product.title </div>

<div style="text-align:center;"><a href="#" target="\_blank"></a></div>

<div style="text-align:center;color:red">￥ @product.marketprice 元</div>

</div>

}

# 二、强类型html辅助方法，在普通辅助方法后面+For

如Html.TextBox---Html.TextBoxFor()

实例，登录页

视图

@model Demo.Models.Users

@using (Html.BeginForm())

{

@Html.EditorFor(u => u.email, "", new { style = "width:124px;" })

@Html.PasswordFor(u => u.pwd, new { style = "width:124px;" })

}

@Html.ValidationMessageFor(u => u.email) <br/>

@Html.ValidationMessageFor(u => u.pwd)

@Html.ValidationSummary(true)

控制器

[HttpPost]

public ActionResult Login(Users user)

{

using (fashionshoppingDBEntities db = new fashionshoppingDBEntities())

{

if (ModelState.IsValid)

{

var u = db.Users.FirstOrDefault(x => x.email == user.email && x.pwd == user.pwd);

if (u == null)

{

ModelState.AddModelError("", "邮箱或密码错误！");

}

else

{

return RedirectToAction("Index");//登录成功，跳转到主页

}

}

}

return View();

}

强类型辅助方法中独有的

**Html.LableFor()**

**Html.DisplayNameFor() 用于显示字段名称，如果使用了模型的[Display(name="姓名")]则显示标注名称**

**Html.DisplayFor() 用于显示字段值**

**Html.EditorFor()相当于 Html.TexBoxFor(),不同的是，会根据数据类型，自己配置生成input type属性**

**另外：使用强类型的好处，在Edit页面，会自动将字段值添加到相应的输入框中，一个例外 PasswordFor()不会添加密码**

# 3、强类型视图

新建mvc项目，

Model添加实体数据模型

（右键项目，先生成一下，不然后面执行失败）

Controller右键-添加-控制器（或新搭建机架的项目）-包含视图的mvc5控制器(EF)

选择模型类，数据上下文，输入控制器名称

# 强类型下拉菜单的绑定

控制器

ViewBag. cateid new SelectList (db. Procate, 
return View(product) ; 
cateid", 
catename , 
product. cateid) ; 

前台页面

@Html.DropDownList("cateid", null, htmlAttributes: new { @class = "form-control" })

# [ValidateAntiForgeryToken]用于阻止伪造请求

控制器中，紧跟httppost后面

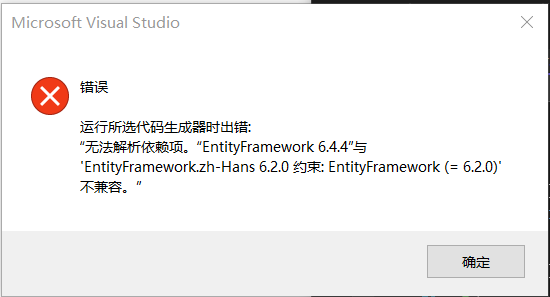


页面中紧跟表单

gusing (Html. BeginForm()) 
'Html. Ant iForgeryToken () 
<div 

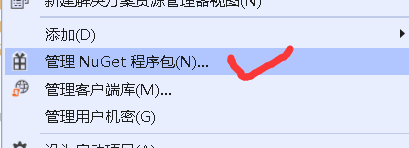
m. a 1 a Ion ummary rue 
ASPNET (Attributes) %JUMVC 
(add 
(add ker*UnobtrLEzveJavaScriptEnabled- 
(script 
"Login Was unsuccessful. Please Correct the errors and try 
HTML . 
true, false. 
(ModelState. IsVaIid)RÉ 

使用支架模型报错解决方法



把EF版本从6.4.4改成6.2.0即可，已经都是6.2.0还不行，就改成以前的版本如6.1.3或5.7.0的版本

项目右键



P 
CJ 
Sam Harwell, Terence Parr 
ANother Tool for Language Recognition, is a language tool that provides a framework for constructing recognizers, 
interpreters, compilers, and translators from grammatical descriptions containing actions in a variety of target langue... 
bootstrap Twitter, Inc. 
Bootstrap framework in CSS. Includes fonts and JavaScript 
EntityFramework EB Microsoft 
Entity Framework is Microsoft's recommended data access technology for new applications. 
EntityFramework.zh-Hans EB Microsoft 
v3.5.o.2 
v3.4.1 
"5.1.3 
v6.2.o 
M6.4.4 
v6.2.o 
•e EntityFramework 
e,-n-: 6.2.o 
6.4.4 
6.44 
Entity F 
for .NE 
6.44 
6.4. O 
63.0 
6.1.3 
6.1.2 
6.1.1 
NuGet Demol 
Package source • 
nugetorg 
lational mapper 
stabilization. 

7 AJAX及其mvc辅助方法

2020年1月29日

21:24

# 第一部分 AJAX简介

AJAX是在后台请求服务器数据，而不必重载Web页面的一种技术。

MVC5支持AJAX

**jQuery ajax() 方法**



function On Insert (evt) 
var username . val () ; 
var pwd = val ; 
. val () 
var role = 
var data = ' ["username" : 
+ username + 
' ", "pwd":"' 
+pwd + ' "role" . 
+ role + ' 
get post put delete 
type: 'POST' , 
url: ' /api/Sysuser/', 
data : data, 
content Type: "application/ json; 
dataType: 
success: function (data) 
// console. info(JSON. stringify(data)) 
S ("StxtUserNa1ne") . val ' ) 
S ("gtxtPwd") . val '); 
$ ("StxtRole"). val ' ) ; 
S. getJSON (" /api/Sysuser" , 
alert(' 
error: function (xhr, error, 
alert(' 
LoadSysusers) ; 
ex) { 

# 留言板练习

(meta / > 
<title>GuestBook</tit1e> 
<script jquery-3. 3. 1. min. 
<script 
function GuestBookAjax (form, cal I back) 
S ("Sindi cator"). show() ; 
S. ajax({ 
url: form. action, 
type: form. method, 
dataType: "HTML", 
data: S(forrn). serialize(), 
completed: $ ("Sindicator") hide 
success: callback 
function update_sessions (result) { 
S ("Slastmessage") . html (result) ; 
</script> 
Cusing (Html. BeginForm 
(table> 
<tr> 
TextBox ("kTitle") 
TextBox ("kEmail 
</tr> 
TextBox ("kQQ") 
</tr> 

# 第二部分 MVC的AJAX辅助方法

1、AJAX辅助方法说明

当使用AJAX辅助方法时，不需要写任何脚本即可实现程序的异步执行。

使用前需要先引入 jQuery.unobtrusive-ajax.js或 jQuery.unobtrusive-ajax.min.js

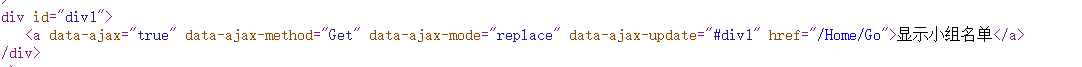
在Razor视图中使用@Ajax调用

2、@Ajax.ActionLink("连接文字","控制器操作方法名称",new AjaxOptions{})

实例



页面运行生成的a标签效果如下



**AjaxOptions对象能够配置对服务器的异步请求以及处理返回的数据**

|  |  |
| --- | --- |
| 强类型辅助方法 | 说明 |
| Confirm | 获取或设置提交请求之前，显示在确认窗口中的消息 |
| HttpMethod | 获取或设置HTTP请求方法（“GET”或“POST”） |
| InsertionMode | 获取或设置指定如何将响应结果插入到目标DOM元素的模式 |
| LoadingElementId | 获取或设置加载时要显示的HTML元素的id属性值 |
| OnBegin | 获取或设置更新页面之前调用的JavaScript函数的名称 |
| OnComplete | 获取或设置数据响应之后、更新页面之前，调用的JavaScript函数 |
| OnFailure | 获取或设置页面更新失败时调用的JavaScript函数 |
| OnSuccess | 获取或设置页面更新成功之后调用的JavaScript函数 |

3、@Ajax.BeginForm()

@Ajax.BeginForm("操作方法","控制器",new AjaxOptions{})



控制器写法，返回分布视图，并q绑定数据

public ActionResult Search (string pname) 
u'ing (fashionshoppingDBEntities db = new fashionshoppingDBEntities()) 
b' = db. Product. Where (p p. title. Contains (pname) ) . ToList () ; 
return ps) 

分部视图写法

'roducdnfos.cshtml •g X 
-lomoJ3nzoI er,cf 
In ox.csh Tm 
2 
4 
6 
8 
9 
10 
11 
12 
13 
Cmodel IEnumerable<Demo. Models. Product> 
S<div id="searchresults"> 
gif (Model. Count() O) { 
else 
foreach (var item in Model) 
< •item. title, iliYJffrf6: Citem. 
</div> 

8 ASPNET Web API

2020年1月29日

21:25

# 第一部分 Web API简介

Web API和SingalR共同构建了Services的框架。

Web API负责构建HTTP的常规服务

SignalR负责构建实时服务，如投票，聊天室，在线游戏等实时性要求比较高的服务

**Web API的主要功能**

**1、支持 基于HTTP verb(get,post,put,delete)的crud（create,retriece,update,delete）操作。**

**2、请求的回复格式支持JSON\XML**

**3、原生支持Odata**

**4、支持Self-host或者IIS host**

**5、支持MVC的大多数功能**

# 

# 第二部分、API实现步骤

新建mvc 项目选web api

步骤

1、使用EF添加实体，并且生成项目

2、生成controler步骤，在controller文件夹右键，添加-控制器

web API - 
Web API 2 EntiW Fra me work) 
web AP' 2 

选项1：包含模板

选项2：包含指定ef的增删改查操作

选项4：包含读写的模板

3、

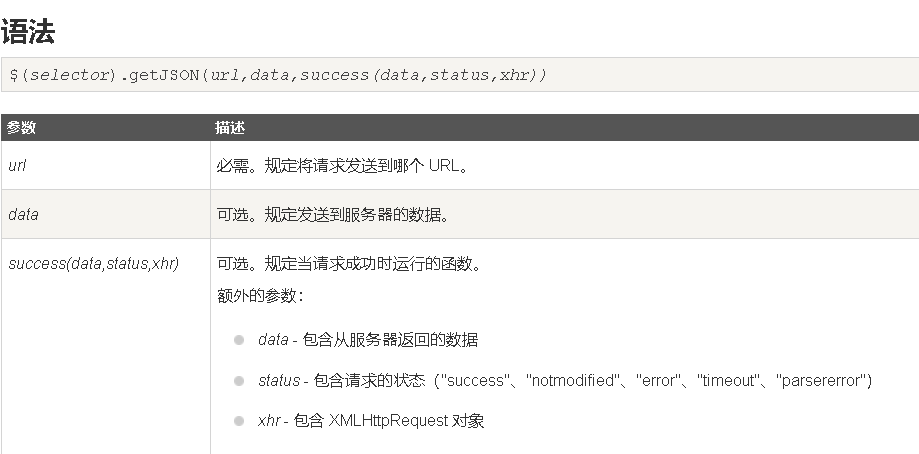
注意带导航的实体集合不能序列化会报错

解决方法：

a、实体模型中去掉导航属性

b、使用映射Select方法选择非导航属性生成新的实体

**\*关于 $.getJson()方法**



# 第三部分、使用ajax访问 api



**完成效果：**

用 户 列 表 
编 号 账 号 
状 态 
乍 改 改 改 改 
管 理 员 
704A8 囗 0g0 # 37620F61 
138 ” ” 34 
2 
704A8 囗 0g0 # 3762dF61 
135 ” ” 《 123 
9 
adsfafds 
10 222 
添 加 新 管 理 员 
提 交 

**编码贴图**

Index.cshtml X 
—1 < div c 
LS : (table i d; 
< 'div) 
(div) 
: < form) 
(fieldset> 
(input 
(input 
type:" text" D 
(input 
type:" text" i d:" username' / > 
(input 
type: text" userstatus" D 
(input 
(input value: D 
< 'div) 
(script l. min. 
(script) 
$ (function O 
LoadUsers) : 
$ (%btnsub") . click (Onlnsert) : 
function LoadUsers (data) ( 
console. info (data) , 
(O) . remove() . 
$. each (data, function (key, Val) ( 
var neatr 
type-text" value"' + val. Loginld + ' + 
type—-text- 
Loginpvd ' 
type—-text- val. Telephone ' 
type* ext value— 
val. tserName + ' • + 
type* ext value— 
val. tserStatus ' 
S (%$userTabIe") . append (newtr) . 
$ ( • input (name:' btnUpdate' ' ) . click (OnCpdate) ; 
$ ('input (name:' btnDeIete' " ) . click (OnDeIete) ; 
+ Telephone e' " , "UserNaze" . 
+ Telephone*' "UserName- : 
function Onlnsert (evt) { 
var Loginld val() . 
Var 
var Telephone $ val 
var UserNaze = val() 
var UserStatus '$1, - *userstätus- ) Aral 
' {"Loginld" 
+ Loginld + ' 
var data = 
type: 'post' , 
url: ' /api/users' , 
data: data, 
'Loginpwd" 
Logi nPvd 
' "Telephone": " 
+ UserName + 
' " , 'UserStatus : 
contentlype appl 
datalype: ' json' , 
success: function (data) { 
//console. info (JON. stringify (data)) 
val ' ) : 
$ ( "*userphone") . val ' ) 
val ' ) 
val(' ' ) • 
S. getJSON<" / api/users", LoadUsers) 
alert(' ') 
function OnUpdate(evt) { 
$ (this) . parent . parent() 
var trthis 
var Userld trthis. children(). eQ(O). 
+ UserName 
+ UserStatus — 
' , "UserStatus . ' 
var Loginld trthis. children(). ea(l). find inmat' val 
var Login%d trthis. chi Idren(). eq(2). find( input' val : 
var Telephone trthis. children(). eq(3). find(' input' ) . val() 
var UserXaze trthis. children(). eq(4). find(' input' val 
var UserStatus = trthis. children . eq (5) . find(' input' ) . val ; 
' {"Userld" : +UserId+' 8, "Loginld" : 
' " , "Loginpvd" : 
var data = 
+ Loginld + 
type: 'put', 
url: ' /api/users/' +UserId, 
data: data, 
contentlype appl 
datalype: ' json' , 
success: function (data) 
//console. info (JON. stringify (data)) 
. val ' ) ; 
val(' ' ) 
$ ( "*userphone") . val . 
S ( " *username"). val ' ) 
val(' ' ) ; 
S. getJSON<" / api/users", LoadUsers) 
function OnDeIete(evt) 
var trthis $(this). parent parent() 
var id trthis. children() . first html : 
type: 'Delete' , 
url: ' /api/users/' +id, 
content Type : appl i cation/ j son: 
dataType: ' 'son' , 
success: function (data) 
//console. info (JSON. stringify (data)) ; 
j) 
+ LoginPwd 
' e, "Telephone" : " ' 
+ UserStatus — 

**代码拷贝（源于项目mytest）：**

<div class="jumbotron">。3

<h1>用户列表</h1>

<table id="userTable">

<tr>

<td>编号</td>

<td>账号</td>

<td>密码</td>

<td>电话</td>

<td>姓名</td>

<td>状态</td>

<td>操作</td>

</tr>

</table>

</div>

<div>

<form>

<fieldset>

<legend>添加新管理员</legend>

账号：<input type="text" id="loginid" /><br />

密码：<input type="text" id="loginpwd" /><br />

电话：<input type="text" id="userphone" /><br />

姓名：<input type="text" id="username" /><br />

状态：<input type="text" id="userstatus" /><br />

<input type="button" id="btnsub" value="提交" />

</fieldset>

</form>

</div>

<script src="~/Scripts/jquery-3.4.1.min.js"></script>

<script>

$(function () {

$.getJSON("/api/users", LoadUsers); //页面加载的时候加载数据,将连接读取到的json内容交给LoadUsers方法处理

$("#btnsub").click(OnInsert); //添加按钮点击事件

})

//加载数据的方法

function LoadUsers(data) {

console.info(data);

$("#userTable").find("tr:gt(0)").remove(); //先清空表格

$.each(data, function (key, val) {

var newtr = '<tr>' +

'<td>' +val.UserId+ '</td>' +

'<td><input type="text" value="' + val.LoginId + '"/></td>' +

'<td><input type="text" value="' + val.LoginPwd + '"/></td>' +

'<td><input type="text" value="' + val.Telephone + '"/></td>' +

'<td><input type="text" value="' + val.UserName + '"/></td>' +

'<td><input type="text" value="' + val.UserStatus + '"/></td>' +

'<td><input type="button" name="btnUpdate" value="修改" /><input type="button" name="btnDelete" value="删除" /></td>' +

'</tr>';

$("#userTable").append(newtr);

});

$("input[name='btnUpdate']").click(OnUpdate); //修改按钮点击事件 也可以直接写在input里面 onclick="OnUpdate()"

$("input[name='btnDelete']").click(OnDelete); //删除按钮点击事件

}

//添加

function OnInsert(evt) {

var LoginId = $("#loginid").val();

var LoginPwd = $("#loginpwd").val();

var Telephone = $("#userphone").val();

var UserName = $("#username").val();

var UserStatus = $("#userstatus").val();

//下面构造json数据

var data = '{"LoginId":"' + LoginId + '","LoginPwd":"' + LoginPwd + '","Telephone":"' + Telephone +'","UserName":"' + UserName + '","UserStatus":"' + UserStatus + '"}';

$.ajax({

type: 'Post',

url: '/api/users',

data: data,

contentType: "application/json;charset=utf-8",

dataType: 'json',

success: function (data) {

//console.info(JSON.stringify(data));

$("#loginid").val('');

$("#loginpwd").val('');

$("#userphone").val('');

$("#username").val('');

$("#userstatus").val('');

$.getJSON("/api/users", LoadUsers);//刷新页面

alert('添加成功！');

}

})

}

//修改

function OnUpdate(evt) {

var trthis = $(this).parent().parent();

var UserId = trthis.children().eq(0).html();

var LoginId = trthis.children().eq(1).find('input').val();

var LoginPwd = trthis.children().eq(2).find('input').val();

var Telephone = trthis.children().eq(3).find('input').val();

var UserName = trthis.children().eq(4).find('input').val();

var UserStatus = trthis.children().eq(5).find('input').val();

//下面构造json数据

var data = '{"UserId":"'+UserId+'","LoginId":"' + LoginId + '","LoginPwd":"' + LoginPwd + '","Telephone":"' + Telephone+'","UserName":"' + UserName + '","UserStatus":"' + UserStatus + '"}';

$.ajax({

type: 'Put',

url: '/api/users/'+UserId,

data: data,

contentType: "application/json;charset=utf-8",

dataType: 'json',

success: function (data) {

//console.info(JSON.stringify(data));

$("#loginid").val('');

$("#loginpwd").val('');

$("#userphone").val('');

$("#username").val('');

$("#userstatus").val('');

$.getJSON("/api/users", LoadUsers);//刷新页面

alert('修改成功！');

}

})

}

//删除

function OnDelete(evt) {

var trthis = $(this).parent().parent();

var id = trthis.children().first().html();

$.ajax({

type: 'Delete',

url: '/api/users/'+id,

contentType: "application/json;charset=utf-8",

dataType: 'json',

success: function (data) {

//console.info(JSON.stringify(data));

$.getJSON("/api/users", LoadUsers);//刷新页面

alert('删除成功！');

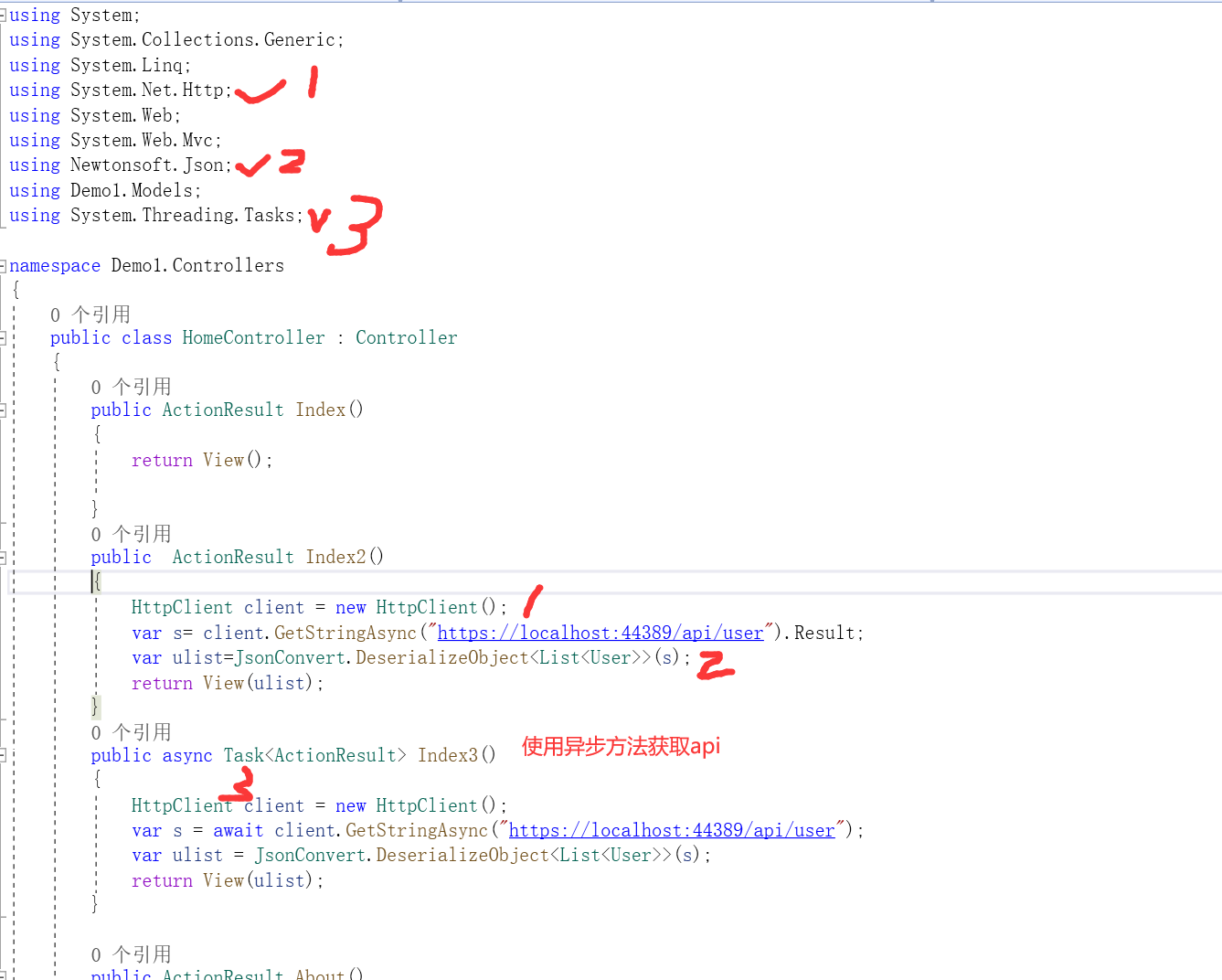
}

})

}

</script>

# \*第四部分 使用MVC获取api数据



9 过滤器和路由

2020年1月29日

21:25

# 

# 第一部分 ActionResult 对象（控制器方法的返回类型）

# 该类是一个抽象的基类

最常见的派生类包括

ViewResult 用于返回标准的视图页面 例

return View();

return View(Users); =》Model强类型绑定数据

return View("index");，

也可以 返回分布式图return ParticalView("plist")

跳转到其它视图 return RedirectToAction("index")

EmptyResult 不返回任何数据 例 return new EmptyResult();

ContentTesult 以字符串的形式指定响应的内容

FileResult 返回任意的文档内容

JavaScriptResult响应一段脚本内容

public ActionResult 
Test 
return Content ("body {color: red} ", "text/css") ; 
public ActionResult 
Img 
string path = Server. MapPath ("N/ Images/ 111. png") ; 
return File (path, "image/png") , 

# 

# index页面调用

Index.cshtml -g 
1 
2 
3 
4 
5 
6 
7 
8 
x 
Ham econ 
ViewBag. Title — 
"Home Page 
< link href—' @Url. Action ("Test") ' rel—"stylesheet" / > 
<img src=' @Url. Action img")' alt:" Alternate Text" / > 
Él{J 

# 

# @Url.Action()用户获取指定视图的路径

# 

# 第二部分 过滤器

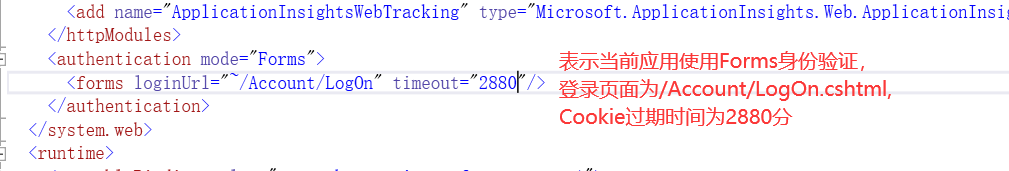
**过滤器类型**

过 滤 器 类 型 
Authorization 
Exceptio n 
Action 
Result 
接 囗 
IAuthorizationFilter 
IExceptionFilter 
IActionFilter 
IResultFilter 
描 述 
此 类 型 （ 或 过 滤 器 ） 用 于 限 制 执 行 控 制 器 或 控 制 器 的 
某 个 操 作 方 法 
于 指 定 一 个 操 作 ， 用 于 处 理 控 制 器 中 抛 出 的 异 常 
用 于 执 行 操 作 之 前 或 之 后 的 处 理 
用 于 返 回 结 果 之 前 或 之 后 的 处 理 
． 午 艹 圳 

**一、授权过滤器 Authorization**

实例，控制index的访问权限，如果没有登录则使用过滤器跳转到登录页面，只有登录的才能访问

1、Web.config文件,即定义一个权限验证规则，只能有1条规则



**2、先引入命名空间**System.Web.Security; ，颁发通行证，验证通过就通行，不通过就返回到验证页

else 
FormsAuthentication. SetAuthCookie (emai l, 
return true; 
false) ; 
ffCl$IJcookie%gEurl 
NIJ, 

3、控制器方法前添加授权 在需要进行验证的操作或者类前调用验证

[Authorize] 
public ActionResult Index () 
return View() ; 

**二、自定义过滤器**

①新建文件夹filter ，添加类名TeacherLoginAttribute，必须加Attribute

②类中导入命名空间：using System.Web.Mvc，类继承接口ActionFilterAttribute

③登录过滤器，如果teacher的session为空的话，需要到首页中让其登录：

public override void OnActionExecuting(ActionExecutingContext filterContext)

{

//if (HttpContext.Current.Session["user"] == null)

if (filterContext.HttpContext.Session["user"] == null)

{

//返回首页进行登录

filterContext.Result = new RedirectResult("/Home/Login");

}

}

④控制器：using MVC\_Demo1.Filters;

public class TeacherController : Controller

{

// GET: Teacher

[TeacherLogin] //特性

}

注意调用的时候，如果 [TeacherLogin]写在方法前，则使用过滤器验证某个方法

如果用在类前，则整个类的所有方法都要调用该过滤器规则

注意，因为是使用特性调用控制器，特性是可以传参的，如下

public ActionResult IndexO 
public ActionResult Create() 

因此可以在过滤器里面添加属性或者方法

# 第三部分 路由

App\_Start文件夹下RouteConfig.cs 文件用于设置路由规则

在程序入口 Global.cs文件的Applicetion\_start()方法中会调用

RouteConfig.cs 文件的

public static void RegisterRoutes(RouteCollection routes)

**一、RouteConfig.cs 文件**

public static void RegisterRoutes(RouteCollection routes)

{

//忽略.axd结尾的网址，不匹配任何路由

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default", //定义路由名称

url: "{controller}/{action}/{id}", //路由参数名称，id为可选参数

//定义每个路由参数的默认值

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

}

自定义路由规则需要在默认的routes.MapRoute前增加新的定义，实例如下

routes.MapRoute(

name: "News",

url: "{controller}/{action}/{year}-{month}-{day}",

defaults: new

{

controller = "Home",

action = "News",

id = UrlParameter.Optional

}

);

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

路由的规则：

1、路由表中至少要包含一条规则

2、所有的路由参数是必须要赋值的（1内置参数 {controler},{action};2自定义参数）

赋值两种方式：1通过注册路由的时候给默认值 2通过URL

URL赋值优先级高

3、当一个路由，可以匹配路由表中多个规则时，匹配第一条

# 第四部分 路由分页实例

路由配置

routes.MapRoute(

name: "Pager",

url: "{controller}/{action}/{pagesize}/{pageindex}",

defaults: new { controller = "Teacher", action = "Index", pagesize = "2", pageindex = "1" }

);

DAL类中

public static List<Teacher> GetTeachers(int pageindex,int pagesize,out int pagecount)

{

using (ExamDBEntities db = new ExamDBEntities())

{

var teachers = db.Teacher;

pagecount = (int)Math.Ceiling(teachers.Count() \* 1.0 / pagesize);

return

teachers.OrderBy(t => t.TeacherID).Skip((pageindex - 1) \* pagesize).Take(pagesize).ToList();

}

}

BLL

public static IEnumerable<Teacher> GetTeachers(int pageindex,int pagesize,out int pagecount)

{

return TeacherService.GetTeachers(pageindex, pagesize,out pagecount);

}

Controler

public ActionResult Index(int pagesize=2, int pageindex=1)

{

if (Session["teacher"] ==null)

{

//return RedirectToAction("Login");//默认跳转到当前控制器的动作

return Content("<script>alert('请先登录');location.href='/Home/Index';</script>");//相当于ReponseWrite

}

var pagecount = 0;

//拿数据

var teachers = TeacherManager.GetTeachers(pageindex, pagesize,out pagecount).ToList();

ViewBag.PageIndex = pageindex;

ViewBag.Pagecount = pagecount;

ViewBag.Pagesize = pagesize;

return View(teachers);

}

//视图中分页

@{ var pageIndex = (int)ViewBag.PageIndex;

var pageCount = (int)ViewBag.Pagecount;

var pageSize = (int)ViewBag.Pagesize;

}

<div>

<span>第 @pageIndex 页</span>

<span>共 @pageCount 页</span>

</div>

@if (pageIndex <= 1)

{

@Html.ActionLink("上一页", "Index", null, new { disabled = "disabled",@class = "btn btn-primary" });

}else{

@Html.ActionLink("上一页", "Index", new { pagesize = pageSize, pageindex = pageIndex - 1 }, new { @class = "btn btn-primary" });

}

@if (pageIndex >= pageCount)

{

@Html.ActionLink("下一页", "Index", null, new { disabled = "disabled",@class = "btn btn-primary" });

}else{

@Html.ActionLink("下一页", "Index", new { pagesize = pageSize, pageindex = pageIndex + 1 }, new { @class = "btn btn-primary" });

}

10 测试和部署

2020年1月29日

21:25

# 第一部分 使用vs测试程序

**一、创建测试的两种方法**

1、在创建mvc项目的同时，勾选添加单元测试，并指定测试项目名称



2、在当前解决方案添加新建项目->选择单元测试项目->测试项目名右键->添加->单元测试

Windows 
ESNUnit EJE-Windows, Linux*0Macos .NET core±jÉF, 
Linux 
macos Windows Web 
Framework) 
MSTest 
Windows 

**二、MVC框架支持测试驱动开发，可以为控制器执行单元测试**

1.修改配置文件 将被测试项目的web.config中的连接字符串拷贝到测试项目的app.config

10 
11 
12 
13 
14 
(connectionStrings> 
(add name="fashionshoppingDBEntities 
(runtime > 
connectionString= 
met 

2.从demo bin文件夹拷贝 EntityFramework.SqlServer.dll库文件 到测试项目的bin debug文件夹下

( 也可管理nut包，添加EntityFramework选择一致的版本，另外如果是单独创建的测试单元项目，则页需要添加ASPNET.MVC并选择对应版本，必要时重启vs)

3.引入命名空间，新建测试单元

using System.Collections.Generic;

using System.Web.Mvc;

using Demo.Models;

using Demo.Controllers;

**实例**

被测试

public class ProductContr011er 
// GET: Product 
1 +}lÆIe1/1 
Controller 
public ActionResult Index 
using (fashionshoppingDBEntities db new fashionshoppingDBEntities ()) 
return View (db. Products. Incl ude ("Procate"). ToList ) ; 

**测试代码**

Index.cshtml 
ProductController.cs 
UnitTest1 .cs* X 
•t: Demo. Tests.Con troll ers.ProductControIIerTest 
t? Index0 
public class ProductContr011erTest 
[TestMethod] 
public void Index() 
ProductContr01 ler col = new ProductController () ; 
col. Index() as ViewResult; 
var result = 
Assert. Islnstance0fType (result. Model, typeof , 

**Assert的常用方法**

|  |  |
| --- | --- |
| **方法** | **作用** |
| **AreEqual/AreNotEqual** | **等值判断** |
| **IsTrue/IsFalse** | **bool判断** |
| **IsNotNull/IsNull** | **空值判断** |
| **IsInstanceOfType(sum,typeof(int))** | **类型判断** |
| **AreSame** | **判断是否引用同一对象** |

# 第二部分 安装iis

# 控制面板，程序，启动或关闭windows功能

internet information service勾选

1、Web管理工具-IIS管理控制台

2、万维网服务-应用程序和开发功能-

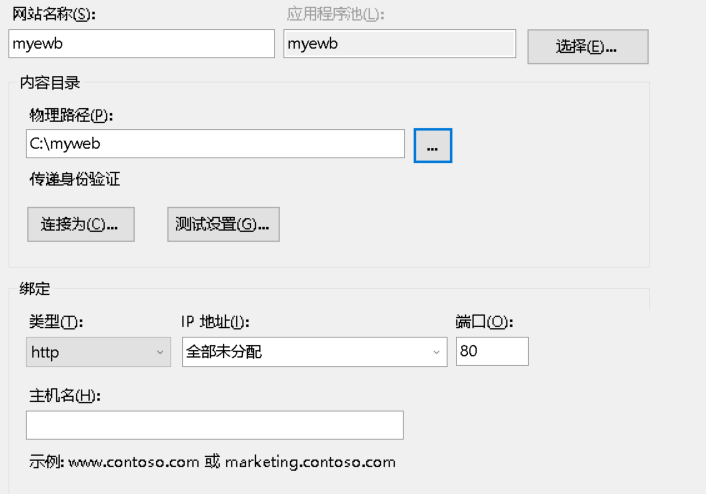
1、勾选web管理工具

2、勾选万维网服务，并展开，勾选asp.net对应版本ASP.NET 4.7（相应版本）

# 第三部分 程序发布

先创建一个网站 ->我的电脑-管理-网站添加一个网站

DESKTOP-HMMSM75 myweb 
DESKTOP-HMMSM75 (CEN Ticent) 
myvveb 
Internet Information Services (IIS)E— 
WMI 
SQL 
SQL 



1、发布时，以管理员方式启动vs打开项目

2、项目->右键->发布->IIS\FTP等->发布方法 Web部署或者WebDeploy包->服务器loaclhost->

项目名称 Default Web Site/App1 （Default Web Site网站需要已经存在，否则失败）

->验证连接绿色对号成功->保存->发布

3、网站有更新的时候，点击发布就会自动更新

4、开发时，数据库通常是本地的，发布，配置时可以编辑数据库连接字符串，勾选在运行时使用此链接字符串（更新目标web.config）

注意：不以管理员方式启动vs会报错提示 运行msdeploy提升，解决办法是以管理员方式启动vs

如果浏览网站提示以下错误



# 可以从新生成一下项目，再发布

# 第四部分 管理网站

安装好后配置网站

电脑右键->管理->服务

DESKTOP-HMMSM75 (C 
Default Web Site 
•9 appv 
Internet Information 
WMI 
SQL servergemEH* 
SQL Server 
ASP_NET 
.NET 
.NET 
.NET 
.NET 

Default Web Site是已经存在的网站，可以设置其基本路径，可以绑定IP等操作

\*11 Cache缓存的使用

2022年2月14日

14:04

## 特点

1、指的是服务器的缓存。与Application类似，会话间共享

2、可以灵活设置数据的销毁时间

## 数据cache有两种

1、固定期缓存

2、可调期缓存



## 缓存依赖：当数据变化而缓存数据没有更新就成为脏数据，因此要用到缓存依赖

步骤：

进入目录



1、启用数据库监视

aspnet\_regsql -S . -E -d ExamsDB -ed

如果数据库没有启用混合登陆

aspnet_regsql -S . -E -d ExamsDB -ed 
(aspnet regsql -S . -U sa -P xxxx -d ExamsDB -ed) 

2、启用表监视

aspnet\_regsql -S . -E -d ExamsDB -t RightsInfo -et

打开“开始”|“所有程序”|“Microsoft Visual Studio 2010”|“Visual Studio Tools”|“Visual Studio 2010命名提示”菜单命令。

 在命令框内输入：aspnet\_regsql.exe -S  SqlServer服务器  -U <Username> -P <Password> -ed -d 数据库名称 -et -t 表名

这个也可以

3、配置Web.config(根目录的)

<connectionStrings>

<add name="Exams4Cache" connectionString="data source=.;initial catalog=ExamsDB;integrated security=True;"/>

</connectionStrings>

<system.web>

<caching>

<sqlCacheDependency enabled="true" pollTime="2000">

<databases>

<add name="RightsCache" connectionStringName="Exams4Cache"/>

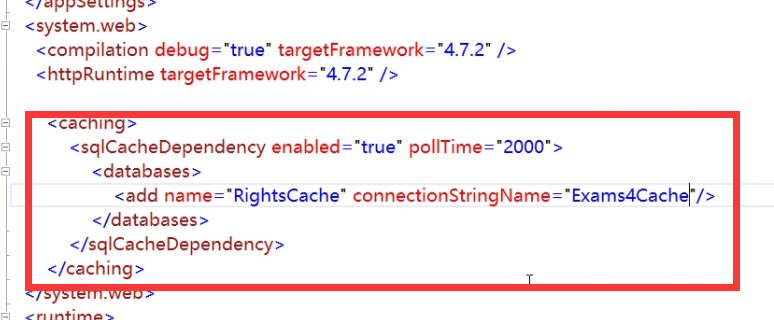
</databases>

</sqlCacheDependency>

</caching>

</system.web>

< connectionStrings> 
<add name= "ExamsDBEntities" connectionString= Modell .csdllres://*/Models.Mode11 
Models.Mode11 connection 
catalog = ExamsDB;integrated security= True;MultipleActiveResultSets = EntityFramework&quot; 
<add connectionString="data source=.;initial catalog=ExamsDB;integrated 
<entitvFramework> 



4、设置缓存依赖



也就是上面代码null的位置



rightsCache是配置文件中的名字，RightsInfo是步骤2中监视表的名字