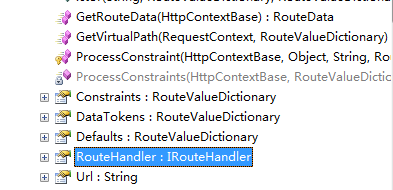
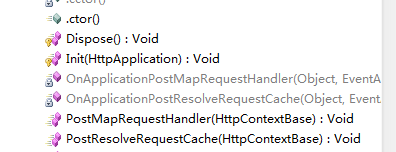
Da 几组重要类：

###### Route->RouteBase （路由）包含以下重要信息



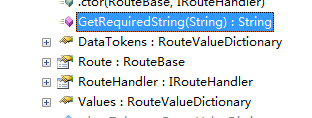
###### RouteCollection（上面路由集合） 会根据请求的路径匹配到一个Route

###### UrlRoutingModule（处理模块 用来获得匹配路由）



调用顺序 RouteCollection调用本身逐个遍历Route的GetRouteData方法 获取匹配的路由然后构造RouteData

###### RouteData信息如下



###### RequestContext

是在获取RouteData后 [RequestContext](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.RequestContext) **requestContext** = new [RequestContext](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.RequestContext/.ctor(System.Web.HttpContextBase,System.Web.Routing.RouteData))(context, routeData);

###### RequestData

被保存在HttpContext.item中 用来与处理程序进行数据的交互

[RequestData](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.UrlRoutingModule.RequestData) **data2** = new [RequestData](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.UrlRoutingModule.RequestData/.ctor()) {

[OriginalPath](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.UrlRoutingModule.RequestData/property:OriginalPath:String) = context.[Request](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Abstractions:3.5.0.0:31bf3856ad364e35/System.Web.HttpContextBase/property:Request:System.Web.HttpRequestBase).[Path](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Abstractions:3.5.0.0:31bf3856ad364e35/System.Web.HttpRequestBase/property:Path:String),

[HttpHandler](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.UrlRoutingModule.RequestData/property:HttpHandler:System.Web.IHttpHandler) = httpHandler

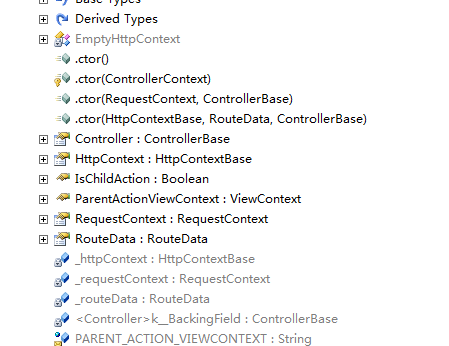
};

###### MvcRouteHandle

在UrlRoutingModule的[**PostResolveRequestCache**](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.UrlRoutingModule/PostResolveRequestCache(System.Web.HttpContextBase))中调用

[IHttpHandler](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web:2.0.0.0:b03f5f7f11d50a3a/System.Web.IHttpHandler) **httpHandler** =routeHandler.[GetHttpHandler](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.IRouteHandler/GetHttpHandler(System.Web.Routing.RequestContext):System.Web.IHttpHandler)(requestContext);用来获取具体的httpHandle默认是mvcHandle

###### [ControllerContext](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ControllerBase/property:ControllerContext:System.Web.Mvc.ControllerContext)



在ControlBase的Execute中的[Initialize](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ControllerBase/Initialize(System.Web.Routing.RequestContext))(requestContext)初始化;

this.[ControllerContext](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ControllerBase/property:ControllerContext:System.Web.Mvc.ControllerContext) = new [ControllerContext](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ControllerContext/.ctor(System.Web.Routing.RequestContext,System.Web.Mvc.ControllerBase))(requestContext, this);

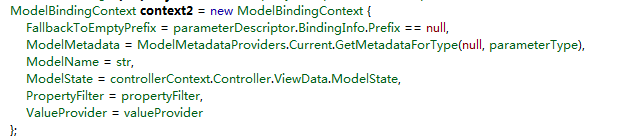
###### [ModelBinders](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ModelBinders)（静态类）

里面的静态属性binders属性存储了所有的默认绑定和自定义绑定。

注册ModelBinders.Binders.Add(typeof(Cart), new CartModelBinder());

这个binders会在整个网站共享

##### ModelBinderContext



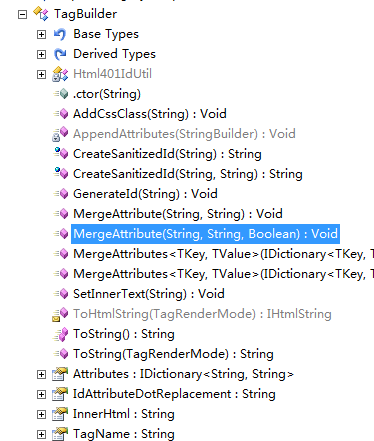
###### [ValueProviderFactories](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ValueProviderFactories)（静态类）

里面的静态属性[**Factories**](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ValueProviderFactories/property:Factories:System.Web.Mvc.ValueProviderFactoryCollection)**属性存储了所有值提供器的工厂类**

注册ValueProviderFactories.Factories.Insert(0, new CookieValueProviderFactory());

这个[**Factories**](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ValueProviderFactories/property:Factories:System.Web.Mvc.ValueProviderFactoryCollection)会在整个网站共享

###### TagBuilder用来生成html标签



###### MvcHtmlString返回不用编码的字符串

<http://www.entlib.net/?p=34>

# 路由

|  |  |  |  |
| --- | --- | --- | --- |
| Url | 网址匹配（必须） | string | "Browse/{category}/{pageIndex}" |
| RouteHandler |  | IRouteHandler | new MvcRouteHandler() |
| Defaults |  |  | new RouteValueDictionary(new {  controller = "Products",  action = "List",  category = "Fish",  pageIndex = 3 })  //control 和action一般情况是要的用来指定默认的控制器和动作  余下的参数是根据上面的URL来指定的，比如category对应上面的{ category } 当这个参数没有填时候用category = "Fish"  /Browse 就代表没有提供任何值所以采用下面的默认值  / Browse/”cat” 只提供了{ category }的值 其他采用默认值 |
| Constraints |  |  |  |
| DataTokens |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

请求过程

|  |
| --- |
| 当一个URL被请求时，系统调用，为每个申请注册的IHttpModules。 其中之一是UrlRoutingModule，这对于。NET 3.5的应用是从你直接引用 应用程序的Web.config文件，为。NET 4应用程序所引用的是计算机范围的ASP.NET Web.config文件和IIS 7.x中的applicationHost.config文件。这个模块做了三件事情： 1。它寻找RouteTable.Routes中与请求匹配的第一RouteBase对象。  标准的路由条目匹配三个条件得到满足：  •所请求的网址如下路由的URL模式。 •所有的花括号中的参数必须出现请求的URL，或在本 默认集中合。 •Constraints条件必须满足。  **注意：**  UrlRoutingModule只是简单的从RouteTable.Routes自上而下的按顺序去匹配。所以路由注册的顺序很重要。  2让匹配到RouteBase(也就是注册的路由)提供一个ROUTEDATA结构，包括以下4个方面  Route：本身的引用  RouteHandler：该对象实现了接口IRouteHandler，这将处理在ASP.NET MVC应用程序的请求（它通常是一实例MvcRouteHandler）  Values：URL指定的参数值，如果没有提供采用默认值。  DataTokens： 3。它调用RouteData的RouteHandler。它提供所有可用的RouteHandler 通过对参数的当前请求的信息称为requestContext。 requestContext包括RouteData信息和HttpContextBase对象所有信息（包括HTTP头，cookies，cookies,  authentication status, query string data, and form post data. ) |

## 路由顺序

一般把特殊的路由放在一般的路由前面 比如

|  |
| --- |
| routes.MapRoute(  "Default", // Route name  "{controller}/{action}/{id}", // URL with parameters  new { controller = "Home", action = "Index", // Parameter defaults  id = UrlParameter.Optional }  );  routes.MapRoute(  "Specials", // Route name  "DailySpecials/{date}", // URL with parameters  new { controller = "Catalog", action = "ShowSpecials" } // Parameter defaults  );  这个路由的顺序是错的 因为模块是自上而下寻找路由的 ，比如DailySpecials/2001-01也许会匹配成control=” DailySpecials” action=” 2001-01” 这样就会导致找不到控制器发生错误。 |

## 命名空间NameSpace

|  |
| --- |
| 通常，当一个请求匹配特定路由条目，MVC框架取得控制器参数（无论是从大括号中的URL模式{controler}参数控制器或者是路由项的默认值的集合），然后查找任何具有匹配名称的控制器类。对于 例如，如果输入Controller的值是Products，它会寻找所谓的ProductsController（大小写不敏感）的控制器类。必须有一个匹配的控制器所在的引用的程序集，如果有两个或更多，它就会失败，报告说有一个模糊的匹配。 如果你想使ASP.NET MVC的一些命名空间时优先选择控制器 处理一个请求，你可以传递一个额外的参数，称为命名空间。  例如  routes.MapRoute(null, "Articles/{id}",  new { controller = "Articles", action = "Show" },  new[] { "MyApp.Controllers", "AnotherAssembly.Controllers" }  );  会优先查找MyApp.Controllers命名空间下的Articles控制器 |

## 路由参数约束

|  |
| --- |
| public static void RegisterRoutes(RouteCollection routes) {    routes.MapRoute("MyRoute", "{controller}/{action}/{id}/{\*catchall}",  new { controller = "Home", action = "Index", id = UrlParameter.Optional },  new { controller = "^H.\*", action = "Index|About",  httpMethod = new HttpMethodConstraint("GET") },  new[] { "URLsAndRoutes.Controllers" });  } |

## 磁盘文件路由

|  |
| --- |
| 不是所有的MVC URL都应用于Control和action。我们还需要一种方式服务 的内容，如图像，静态的HTML文件，JavaScript库，等。  public static void RegisterRoutes(RouteCollection routes)  {  // Before or after adding route entries, you can set this:  routes.RouteExistingFiles = true;  } |

## 跳转ActionLink RouteLink

Html. ActionLink 位于System.Web.Mvc.Html.LinkExtensions

作用是返回包含指定操作的虚拟路径的定位点元素（a 元素）。

|  |
| --- |
| <%: Html.ActionLink("See all of our products", "List", "Products") %>  <a href="/Products/List">See all of our products</a>  注意：如果不指定controller那么默认controller就是当前执行的controller；  带querying参数 <%: Html.ActionLink("RSVP Now", "RsvpForm",new { color = "Red", page = 2 }) %>  <a href="/Home/RsvpForm?color=Red&amp;page=2">RSVP Now</a>  指定标签的属性   <%: Html.ActionLink("Red items", "List", "Products", new { color = "Red", page = 2 },new { name="许志伟",age=24})%>  <a age="24" href="/Products/List?color=Red&amp;page=2" name="许志伟">Red items</a> |

从 ROUTEDATA中指定URL跳转

Html.RouteLink位于System.Web.Mvc.Html.LinkExtensions

作用是返回包含指定操作的虚拟路径的定位点元素（a 元素）。

|  |
| --- |
| <%: Html.RouteLink("Click me", new { controller = "Products", action = "List" }) %>  <a href="/Products/List">Click me</a> |

用代码控制重定向操作 类似asp.net Redirect操作

|  |
| --- |
| <%: Html.RouteLink("Click me", new { action = "MyActionMethod" }) %>    public ActionResult MyActionMethod ()          {              return RedirectToAction("GetProducts", "Home");            }          public ViewResult GetProducts()          {              return View("Products");  }  上面RouteLink调用 MyActionMethod action 然后执行GetProducts 跳转到Products页面  或者采用  public ActionResult MyActionMethod()          {              return RedirectToRoute(new { action = "SomeAction", customerId = 456 });          }  实现一样的功能。  public ActionResult MyActionMethod()  {  string url = Url.Action("SomeAction", new { customerId = 456 });  // ... now do something with url  } |

## 使用区域 Working with Areas

当项目比较大

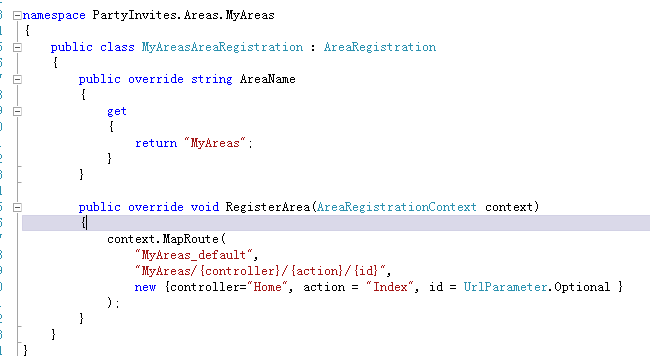
每个区域应该代表了你的应用程序的一个功能模块（例如，行政职能部门，报表或  
讨论论坛），每个区域并且是控制器，视图，路由，.NET类，JavaScirpt等等。使用区域有以下几个好处

1**组织结构清晰**：每个区域都有自己单独的目录结构，所以很容易看到  
如何将这些控制器，视图等，涉及到一个特定的应用功能。

2 **隔离**：

如果你有多个团队开发工作的同时，他们可以专注于自己的区域(Area)。这些小组将不互相干扰，他们每个人都能选择自己的控制器名称和路由配置而不用担心名称的冲突等等。

3**重用**。



### 区域工作原理：

|  |
| --- |
| 当您调用RegisterArea（）中调用context.MapRoute（），它的工作原理如同routes.MapRoute（）在Global.asax.cs，它增加了一个新的Route对象到全局RouteTable.Routes集合中。但它也引入一个DataTokens属性。  RegisterArea（）中调用context.MapRoute（）增加的过程如下  1.首先设置的URL Pattern是以Area名字开头的，这样做是必要的，毕竟这个URL Pattern最终是放在全局中的；  2.把该routebase添加到RouteTable.Routes中并设置该routebase DataTokens["Namespaces"]的值设置为为当前Area的命名空间，DataTokens["area"]的键值，并设置为当前Area名字，如上面图的MyAreas值  DataTokens["UseNamespaceFallback"]，并设置为false    3.当一个请求到来时候，URL会匹配一个路由，然后根据根据DataTokens["Namespaces"]命名空间下去寻找controller. |

### 链接到不同的Area区域

|  |
| --- |
| <%: Html.ActionLink("跳转到MyAreas..:","Index","Home",new {area="MyAreas"},null)%>  <a href="/MyAreas">跳转到MyAreas..:</a>  因为上面home 和index是默认值 所以在URL可以省略 |

### 优先采用控制器名称

|  |
| --- |
|  |

# 控制器control和action

## Filter过滤器

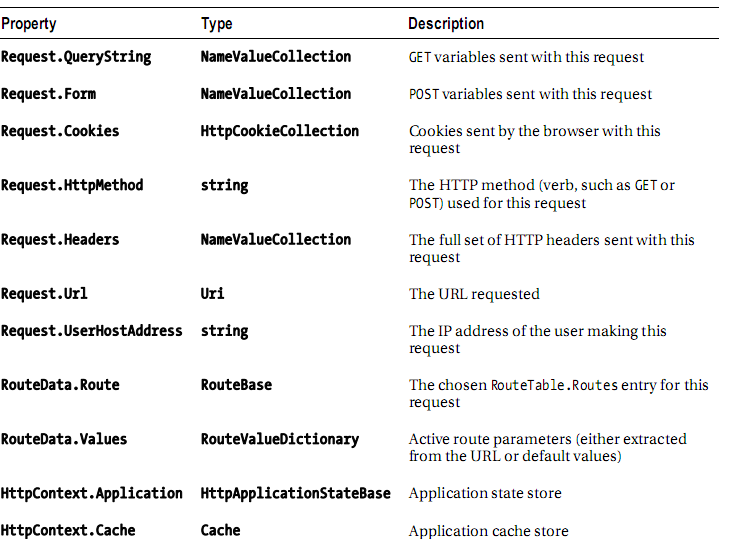
（具体参考英文书）

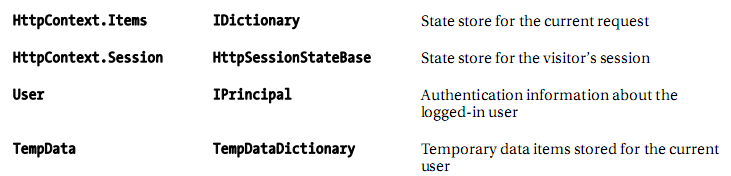
<http://blog.bandao.cn/archive/28359/blogs-659434.aspx>

过滤器可以向请求处理管道注入额外的逻辑。他们提供了一个简单而优雅的方式  
实现横切关注点。经典横切关注点的例子记录日志，授权，和缓存,异常处理等等。。

|  |
| --- |
| 在Asp.netMvc中当你有以下及类似以下需求时你可以使用Filter功能   1. 判断登录与否或用户权限 2. 决策输出缓存 3. 防盗链 4. 防蜘蛛 5. 本地化与国际化设置 6. 实现动态Action |

## 接受传入参数





### Getting Data from Context Objects

|  |
| --- |
| public ActionResult RenameProduct()          {              // Access various properties from context objects              string userName = User.Identity.Name;              string serverName = Server.MachineName;              string clientIP = Request.UserHostAddress;              DateTime dateStamp = HttpContext.Timestamp;              // Retrieve posted data from Request.Form              string oldProductName = Request.Form["OldName"];              string newProductName = Request.Form["NewName"];              ViewData["RenameResult"] = "sfsafas";              return View("ProductRenamed");          } |

还有以下参数可以用

|  |
| --- |
|  |

### 利用Action方法的参数接收传入值

|  |
| --- |
| public ActionResult ShowWeatherForecast()          {            string city = RouteData.Values["city"];             DateTime forDate = DateTime.Parse(Request.Form["forDate"]);              // ... implement weather forecast here ...          }  上面那种获取参数的方法 可以采用以下方法参数来获取   public ActionResult ShowWeatherForecast(string city, DateTime forDate)          {              // ... implement weather forecast here ...          }  为了上面的参数是利用MVC的model binders和valueProvider提供值（下面的篇章会介绍）MVC扫描了Request.QueryString, Request.Form, and RouteData.Values对象 寻找键跟方法参数一样的（大小写不敏感）。也就是说参数的名称要和form表单里面某个NAME元素对应等等。  获取值的顺序是Request.Form，RouteData.Value，Request.QueryString，Request.Files，再找不到就是Null了 |

### 可选参数和强制

|  |
| --- |
| 当遇到值类型的参数 当扫描不到改对象的值时候 会抛出异常 ，所以建议把他转为可空类型 比如int? datetime? |

### 指定默认值

|  |
| --- |
| public ActionResult Search(string query, [DefaultValue(1)] int page)      {              // ...       }  对于引用类型可以采用DefaultValue(new MyType(...)) |

在action方法传MODEL参数

|  |
| --- |
| public ViewResult RsvpForm(GuestResponse guestResponse)         {             if (ModelState.IsValid)             {                 guestResponse.Submit();                 return View("Thanks", guestResponse);             }             Else{ return View();         }  针对上面你可以采用普通表单 但是表单的NAME属性要跟MODEL的属性名称一样（大小写不敏感） 类似以下代码   <% using(Html.BeginForm()) { %>          <%: Html.ValidationSummary() %>          <p>Your name: <%: Html.TextBox("name") %></p>          <p>Your email: <%: Html.TextBox("email") %></p>          <p>Your phone: <%: Html.TextBox("phone") %></p>          <p>              Will you attend?              <%: Html.DropDownList("WillAttend", new[] {                  new SelectListItem { Text = "Yes, I'll be there",                                       Value = bool.TrueString },                  new SelectListItem { Text = "No, I can't come",                                       Value = bool.FalseString }                  }, "Choose an option") %>          </p>          <input type="submit" value="Submit RSVP" />  <% } %>  或者你可以采用强类型的表单  <%@ Page Language="C#" Inherits="System.Web.Mvc.ViewPage<PartyInvites.Models.GuestResponse>" %>     <% using(Html.BeginForm()) { %>          <%: Html.ValidationSummary() %>          <p>Your name: <%: Html.TextBoxFor(x => x.Name) %></p>          <p>Your email: <%: Html.TextBoxFor(x => x.Email) %></p>          <p>Your phone: <%: Html.TextBoxFor(x => x.Phone) %></p>          <p>              Will you attend?              <%: Html.DropDownListFor(x => x.WillAttend, new[] {                  new SelectListItem { Text = "Yes, I'll be there",                                       Value = bool.TrueString },                  new SelectListItem { Text = "No, I can't come",                                       Value = bool.FalseString }                  }, "Choose an option") %>          </p>          <input type="submit" value="Submit RSVP" />      <% } %> |

## 输出（也就是给view传值）

### ViewResult

return View("../Derived/MyView");

return View("Model”);

### ViewData（大小写不敏感）

|  |
| --- |
| public class Person         {             public string Name { get; set; }             public int Age { get; set; }         }           public ViewResult ShowPersonDetails()         {             Person someguy = new Person { Name = "Steve", Age = 108 };             ViewData["person"] = someguy;             ViewData["message"] = "Hello";             return View()      ； }  View页面  <div>          <%: ViewData["message"] %>, world! The person's name is          <%: ((PartyInvites.Models.Person)ViewData["person"]).Name %>          The person's age is          <%: ((PartyInvites.Models.Person)ViewData["person"]).Age %>  </div>  或者可以采用  <%:ViewData.Eval("person.Name")%>//不推荐使用 因为效率比较低-。- |

### ViewBag:

|  |
| --- |
| ViewBag是动态类型 比viewData更灵活  public ActionResult Index()   {       List<string> colors = new List<string>();       colors.Add("red");       colors.Add("green");       colors.Add("blue");        ViewBag.ListColors = colors; //colors is List       ViewBag.DateNow = DateTime.Now;       ViewBag.Name = "Hajan";       ViewBag.Age = 25;       return View();  }  public ActionResult Index()   {       List<string> colors = new List<string>();       colors.Add("red");       colors.Add("green");       colors.Add("blue");        ViewData["listColors"] = colors;       ViewData["dateNow"] = DateTime.Now;       ViewData["name"] = "Hajan";       ViewData["age"] = 25;        return View();   } |

### 采用ViewData.Model IEnumerable

|  |
| --- |
| public ViewResult ShowPersonDetails()         {             Person someguy = new Person { Name = "Steve", Age = 108 };             ViewData["person"] = someguy;             ViewData["message"] = "Hello";             //return View();             return View(someguy);         }  View页面<div>          The person's name is          <%: Model.Name %>          The person's age is          <%: Model.Age %>  </div>  -------------------------------------------------------------   public ViewResult ListModel()         {             IEnumerable<Person> viewModel = new List<Person> {          new Person { Name = "Archimedes", Age = 8 },          new Person { Name = "Aristotle", Age = 23 },          new Person { Name = "Annabelle", Age = 75 },      };             return View(viewModel);         }  View页面:  <ul>      <% foreach (var m in Model)         {  %>         <li>NAME IS<%: m.Name %>,AGE IS <%: m.Age %></li>      <%} %>      </ul> |

### 重定向（RedirectToAction，RedirectToRoute）

|  |
| --- |
| 第一种  public ViewResult RsvpForm(GuestResponse guestResponse)         {  guestResponse.Submit();              return GetProducts()           }  RsvpForm Action中执行GetProducts action ，GetProducts渲染他本身的view视图，但是RouteData.Values["action"]还是RsvpForm ，所以这个有点类似ASP.NET里面的transfer动作 返回状态值是200  第二种 和第三种  public ActionResult MyActionMethod()          {              var s = RouteData;                return RedirectToAction("GetProducts", "Home");            }    public ActionResult MyActionMethod()          {   return RedirectToRoute(new { action = "SomeAction", customerId = 456 });          }  这2种会导致302重定向 |

### RedirectResult

|  |
| --- |
| public RedirectResult ShowPersonDetails()         {             Person someguy = new Person { Name = "Steve", Age = 108 };             ViewData["person"] = someguy;             ViewData["message"] = "Hello";             //return View();             return Redirect("http://www.baidu.com");  或者return Redirect("/Example/Index");             //return View(someguy);         }  也是302状态  如果要返回301状态 可以采用  return RedirectPermanent("/Example/Index"); |

### ViewData与TempData属性来向View页传递对象

|  |
| --- |
| 但是TempData可以类似于Session一样到其它页面仍然存在,但只限一页的访问(类似于Monorail中的Flash)  TempData一般用于临时的缓存内容或抛出错误页面时传递错误信息。    public ViewResult Index()          {             //var t=   RouteTable.Routes;           //   var s = RouteData;                TempData["Name"] = "许志伟";              int hour = DateTime.Now.Hour;              ViewData["greeting"] = (hour < 12 ? "Good morning" : "Good afternoon");              return View();          }  Index.aspx    <%: Html.ActionLink("tempData测试", "GetTemplateData")%>  GetTemplateData.aspx  <div>      <%: TempData ["Name"].ToString()%>      </div>  默认情况TempData是储存在session下的 所以你不能禁用会话。  注意：<%: TempData ["Name"].ToString()%>这种写法，会让TempData["Name"] 在请求结束后提出TempData字典集合  如果想要TempData ["Name"]不被踢出TempData 字典集合，可以采取以下2种办法：  在后台代码中TempData.Keep("Name");  或者view读取时候采用<%:TempData.Peek("Name")%>这种方式 |

### ContentResult(xml,css,text等等)

|  |
| --- |
| 注意contentResult会清空页面的其他内容 然后输出  public ActionResult ContentResult()         {             return Content("名字：许志伟");         }  如果对content-type 不在乎的话可以用下面来代替  public string ContentResult()         {             return "名字：许志伟";         }  结合content-type  1: public ActionResult CssStyle()  2: {  3: string style = GetStyleFromDB(clientID);  4: if (!String.IsNullOrWhiteSpace(style))  5: return Content(style, "text/css"，Encoding.Default);  6: return View();  7: }  1: <link href="@Url.Action("CssStyle", "Home")" rel="stylesheet" type="text/css" />  返回XML实例  public ContentResult XMLData() {    StoryLink[] stories = GetAllStories();    XElement data = new XElement("StoryList", stories.Select(e => {  return new XElement("Story",  new XAttribute("title", e.Title),  new XAttribute("description", e.Description),  new XAttribute("link", e.Url));  }));    return Content(data.ToString(), "text/xml");  }  实体：  public class StoryLink{  public string Title {get; set;}  public string Description { get; set; }  public string Url { get; set; }  }  返回的XML：  <StoryList>  <Story title="First example story" description="This is the first example story"  link="/Story/1" />  <Story title="Second example story" description="This is the second example story"  link="/Story/2" />  <Story title="Third example story" description="This is the third example story"  link="/Story/3" />  </StoryList> |

### JsonResult

|  |
| --- |
| 用法将在后面章程的AJAX详谈  class CityData { public string city; public int temperature; }           [HttpPost]         public JsonResult JosnResultData()         {             var citiesArray = new[] {          new CityData { city = "London", temperature = 68 },          new CityData { city = "Hong Kong", temperature = 84 }             };               return Json(citiesArray);         }  默认情况下JsonResult 只能以POST方式获取的，这是为了方式第三方站点获取数据。  如果要允许GET形式 可以Json(citiesArray,JsonRequestBehavior.AllowGet); |

### JavaScriptResult

|  |
| --- |
| 用法将在后面章程的AJAX详谈   public JavaScriptResult JavaScriptResultSayHello()         {             return JavaScript("alert('Hello, world!');");         } |

### FileResult输出下载文件

|  |
| --- |
| 磁盘路径  public FilePathResult DownloadReport()         {             string filename = @"c:\files\somefile.pdf";             return File(filename, "application/pdf", "AnnualReport.pdf");         }  从比特流   public FileContentResult DownloadReport()  {      byte[] data = ... // Generate or fetch the file contents somehow      return File(data, "application/pdf", "AnnualReport.pdf");   }  Stream流  public FileStreamResult ProxyExampleDotCom()  {  WebClient wc = new WebClient();  Stream stream = wc.OpenRead("http://www.example.com/");  return File(stream, "text/html");  } |

### HttpStatusCodeResult

|  |
| --- |
| public HttpStatusCodeResult StatusCode() {  return new HttpStatusCodeResult(404, "URL cannot be serviced");  }  return new HttpNotFound();  return new HttpUnauthorizedResult(); |

## 扩展控制器

|  |
| --- |
| 自定ControllerFactory  参考ControlExtendsibility中的CustomControllerFactory.cs  注册控制器工厂  ControllerBuilder.Current.SetControllerFactory(new CustomControllerFactory());  指定控制器的命名空间  protected void Application\_Start() {  AreaRegistration.RegisterAllAreas();  ControllerBuilder.Current.DefaultNamespaces.Add("ControllerExtensibility.Controllers");  ControllerBuilder.Current.DefaultNamespaces.Add("ControllerExtensibility.Controllers1");  DependencyResolver.SetResolver(new NinjectDependencyResolver());  RegisterGlobalFilters(GlobalFilters.Filters);  RegisterRoutes(RouteTable.Routes);  }  如果在路由里面也指定了命名空间 则路由的命名的空间会覆盖上面的命名空间。 |

### Dependency Resolver(依赖控制解析器)

|  |
| --- |
| 首先看下 怎么产生IControl类  IControl类是由IControlFactory.Cr[eateController](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.IControllerFactory/CreateController(System.Web.Routing.RequestContext,String):System.Web.Mvc.IController)([RequestContext](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.MvcHandler/property:RequestContext:System.Web.Routing.RequestContext), ControlName);  怎么获取IControlFactory的实现类呢？  在MVCHandle的[**ProcessRequestInit**](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.MvcHandler/ProcessRequestInit(System.Web.HttpContextBase,System.Web.Mvc.IController&,System.Web.Mvc.IControllerFactory&)) **里面有这样一句**  this.[ControllerBuilder](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.MvcHandler/property:ControllerBuilder:System.Web.Mvc.ControllerBuilder).[GetControllerFactory](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.ControllerBuilder/GetControllerFactory():System.Web.Mvc.IControllerFactory)();这句就是获取所有在  [ControllerBuilder](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.MvcHandler/property:ControllerBuilder:System.Web.Mvc.ControllerBuilder)里注册的ControlFactory.  怎么注册Controlfactory呢？  protected void Application\_Start()  {  。。。。。。。。。。  ControllerBuilder.Current.SetControllerFactory(new CustomControllerFactory());  。。。。。。。。。。。。    }  如果不注册 MVC框架提供一个默认的[**DefaultControllerFactory**](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory) :[IControllerFactory](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.IControllerFactory)  我们来看看它的实现吧    public virtual [IController](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.IController) [**CreateController**](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/CreateController(System.Web.Routing.RequestContext,String):System.Web.Mvc.IController)([RequestContext](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Routing:3.5.0.0:31bf3856ad364e35/System.Web.Routing.RequestContext) requestContext, [string](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.String) controllerName)  {  //根据路由参数和控制器名称 获取控制器的类型  [Type](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.Type) **controllerType** = this.[GetControllerType](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/GetControllerType(System.Web.Routing.RequestContext,String):System.Type)(requestContext, controllerName);  //创建控制器实例 注意下面用到了反转控制IOC依赖注入  return this.[GetControllerInstance](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/GetControllerInstance(System.Web.Routing.RequestContext,System.Type):System.Web.Mvc.IController)(requestContext, controllerType);  }  我们来看看[GetControllerInstance](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:2.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/GetControllerInstance(System.Web.Routing.RequestContext,System.Type):System.Web.Mvc.IController)的实现  protected internal virtual [IController](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IController) [**GetControllerInstance**](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/GetControllerInstance(System.Web.Routing.RequestContext,System.Type):System.Web.Mvc.IController)([RequestContext](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web:4.0.0.0:b03f5f7f11d50a3a/System.Web.Routing.RequestContext) requestContext, [Type](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:4.0.0.0:b77a5c561934e089/System.Type) controllerType)  {  //调用[ControllerActivator](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/property:ControllerActivator:System.Web.Mvc.IControllerActivator)来创建  return this.[ControllerActivator](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/property:ControllerActivator:System.Web.Mvc.IControllerActivator).[Create](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IControllerActivator/Create(System.Web.Routing.RequestContext,System.Type):System.Web.Mvc.IController)(requestContext, controllerType);  }  我们再来看看[ControllerActivator](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory/property:ControllerActivator:System.Web.Mvc.IControllerActivator).[Create](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IControllerActivator/Create(System.Web.Routing.RequestContext,System.Type):System.Web.Mvc.IController)方法实现  public [IController](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IController) [**Create**](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory.DefaultControllerActivator/Create(System.Web.Routing.RequestContext,System.Type):System.Web.Mvc.IController)([RequestContext](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web:4.0.0.0:b03f5f7f11d50a3a/System.Web.Routing.RequestContext) requestContext, [Type](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:4.0.0.0:b77a5c561934e089/System.Type) controllerType)  {  [IController](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IController) **controller**;  //这里就用反转控制器 只要实现继承[IDependencyResolver](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IDependencyResolver)就可以  controller = ([IController](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IController)) (this.[\_resolverThunk](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.DefaultControllerFactory.DefaultControllerActivator/_resolverThunk:System.Func%3cSystem.Web.Mvc.IDependencyResolver%3e)().[GetService](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IDependencyResolver/GetService(System.Type):Object)(controllerType) ?? [Activator](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:4.0.0.0:b77a5c561934e089/System.Activator).[CreateInstance](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:4.0.0.0:b77a5c561934e089/System.Activator/CreateInstance(System.Type):Object)(controllerType));  return controller;  }  注册[IDependencyResolver](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Web.Mvc:3.0.0.0:31bf3856ad364e35/System.Web.Mvc.IDependencyResolver)参考下面的例子  可以参考<http://www.cnblogs.com/n-pei/archive/2011/02/15/1955460.html>  具体例子 可以参考SportsStore源码中的NinjectDependencyResolver.cs 原理可以参考SportsStore中的控制台程序  全局注册  protected void Application\_Start()  {  AreaRegistration.RegisterAllAreas();  RegisterGlobalFilters(GlobalFilters.Filters);  RegisterRoutes(RouteTable.Routes);  DependencyResolver.SetResolver(new NinjectDependencyResolver());  ModelBinders.Binders.Add(typeof(Cart), new CartModelBinder());  } |

# 视图VIEW

### UrlHelper

<http://blog.bandao.cn/archive/28359/blogs-695685.aspx>

|  |
| --- |
| UrlHelper 类提供以下方法帮助您操作 URL：   * [Action](http://msdn.microsoft.com/zh-cn/library/system.web.mvc.urlhelper.action.aspx). 此方法生成一个映射到操作方法的 URL。 * [RouteUrl](http://msdn.microsoft.com/zh-cn/library/system.web.mvc.urlhelper.routeurl.aspx). 此方法生成一个映射到路由的 URL。 * [Content](http://msdn.microsoft.com/zh-cn/library/system.web.mvc.urlhelper.content.aspx). 此方法基于资源的虚拟（相对）路径来生成资源的 URL 路径。 * [Encode](http://msdn.microsoft.com/zh-cn/library/system.web.mvc.urlhelper.encode.aspx). 此方法会将指定 URL 中的特殊字符编码为字符-实体等效项。 |

### MvcHtmlString

|  |
| --- |
| public ViewResult Index()          {       TempData["Name"] = "<p>许志伟</p>ss";  return View();          }  View页面代码    <%:  TempData["Name"].ToString()%> 这样页面输出的是<p>许志伟</p>ss 因为<%: 对它进行了编码 ;  如果你希望页面输出html代码 可以采用MvcHtmlString对象 比如  **<%:  MvcHtmlString.Create(TempData["Name"].ToString())%>**  这样页面就不会对TempData["Name"].ToString()进行HTML编码 |

### HtmlHelper

#### 内嵌模式

|  |
| --- |
| @helper CreateList(string[] items) {  <ul>  @foreach (string item in items) {  <li>@item</li>  }  </ul>  }  <h4>InlineHelper</h4>  Days of the week: <p/>  @CreateList(ViewBag.Days)  <p />  Fruit I like: <p />  @CreateList(ViewBag.Fruits) |

#### InputExtensions

##### ·CheckBox

|  |  |
| --- | --- |
| |  | | --- | | <%: Html.CheckBox("Name", true, new  {@class="className",myAtt="me1" })%>   <br />   <%: Html.CheckBoxFor(Model=>Model.WillAttend) %>  <br /> | |

##### ·RadioButton

##### ·TextBox

|  |
| --- |
| <%: Html.TextBox("TextboxName",ViewData.Model.Name , new {@class="className",myAtt="me1" })  %>      <br />  <%: Html.TextBoxFor(x=>x.Name) %>  如果Action中有返回 ViewData["Name"] = model.Name;或者view(model)中具有name属性   <%: Html.TextBox("Name"%>就可以取到值 |

##### ·Password

##### ·Hidden

#### SelectExtensions

##### DropDownList

参考htmlhepler文件夹源码

|  |
| --- |
| <%: Html.DropDownList("name",(IEnumerable<SelectListItem>)ViewData["selectlist"],"请选择") %>  在控制器的action里  SelectList selectList = new SelectList(list, "Age", "Name", 13);  ViewData["selectList"] = selectList;  或者  @Html.ListBox("MyListBox", new SelectList(new[] { "One", "Two", "Three" }))  @Html.DropDownList("sfsf", new List<SelectListItem>() { new SelectListItem() { Text = "safsa", Value = "safsa" },new SelectListItem() { Text = "safsa", Value = "safsa" } });  @Html.ListBox("sfsf", new SelectList(new[] { new SelectListItem() { Text = "safsa", Value = "safsa" } }, "Text", "Value"));  @Html.ListBox("sfsf", new List<SelectListItem>() { new SelectListItem() { Text = "safsa", Value = "safsa" }, new SelectListItem() { Text = "safsa", Value = "safsa" } }); |

#### FormExtensions

|  |
| --- |
| <% Html.BeginForm("MyAction", "MyController", new { param = "val" }); %>        <% Html.EndForm();%>  输出：<form action="/MyController/MyAction?param=val" method="post"></form> |

#### 自定义htmlhepler

|  |
| --- |
| namespace PartyInvites.Views.Helper  {      public static class VideoTagExtensions      {          public static MvcHtmlString Video(this HtmlHelper html, string src)          {              string url = UrlHelper.GenerateContentUrl(src, html.ViewContext.HttpContext);                TagBuilder tag = new TagBuilder("video");              tag.InnerHtml = "Your browser doesn't support video tags.";              tag.MergeAttribute("src", url);              tag.MergeAttribute("controls", "controls"); // Show Play/Pause buttons                return MvcHtmlString.Create(tag.ToString());          }      }  }  页面内  @helper CreateList(string[] items) {    <ul>  @foreach (string item in items) {  <li>@item</li>  }  </ul>  }  <h4>InlineHelper</h4>    Days of the week: <p/>  @CreateList(ViewBag.Days)    <p />  Fruit I like: <p />  @CreateList(ViewBag.Fruits) |

#### **LinkExtensions**

#### Creating Links and URLs

|  |
| --- |
| UrlHelper不会自动产生<a>标签 只会产生url链接字符串  **LinkExtensions**可以自动产生<a>  @Url.Content("~/Content/Site.css")  <br />  @Url.Action("UrlsSample", "Home", new { param = "sfsf", controller = "UrlsSample" }, "https")  <br />  @Url.RouteUrl("Default", new { controller = "Home", action = "UrlsSample", param = "sfsf" });  <br />  @Html.ActionLink("Hi", "UrlsSample", "Home")  <br />  @Html.RouteLink("routelinkTest", new { controller = "Home", action = "UrlsSample", param = "sfsf" }); |

### ParttialView

|  |
| --- |
| 创建部分视图  **部分视图必须在访问的页面所对应的Controller对应的文件夹下或者在/Views/Shared下**  **如果你想把视图放在自己定义的文件夹下面就必须采用虚拟全路径（如下面第二种方式）。**  右键添加-视图-勾选创建部分视图  ViewUserControl1.ascx  使用部分视图：  <%: Html.Partial("ViewUserControl1")%>  <%: Html.Partial("~/Views/Home/ViewUserControl1.ascx")%> <%-- 此处必须用这个路径--%>  <% Html.RenderPartial("ViewUserControl1");%>  Partial和RenderPartial的区别是 Partial会在内存中创建一个StringWriter,然后部分视图输出MvcHtmlString到StringWriter。  RenderPartial会直接输出到response流中。 |

#### 给Partview传值

##### Viewdata viewdata.model

PartialView可以访问它的容器页面（通常是aspx页面）的viewdata中的所有数据。用法跟通常的view页面一致。

还有一种特殊用法

|  |
| --- |
| public ViewResult ListModel()         {             IEnumerable<Person> viewModel = new List<Person> {          new Person { Name = "Archimedes", Age = 8 },          new Person { Name = "Aristotle", Age = 23 },          new Person { Name = "Annabelle", Age = 75 },      };             return View(viewModel);         }  容器aspx页面    <ul>      <% foreach(var person in Model) { %>          <li>              <% Html.RenderPartial("PersonInfo", (PartyInvites.Models.Person)person);%>          </li>      <% } %>  </ul>  Partial视图PersonInfo.ascx页面  <%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<PartyInvites.Models.Person>" %>  <%: Model.Name %>  <%: Model.Age %> |

### ChildAction

|  |
| --- |
| 参考 SectionsChildactionPatialView项目文件 |

# Model和数据录入

## Html.Display Html.DisplayFor

|  |
| --- |
| **以只读形式显示实体的某个属性，后面会介绍如何用Model metadata来改变显示**   <%: Html.Display("FirstName")%>    <%: Html.DisplayFor(x=>x.FirstName) %> |

## Html.DisplayForModel

|  |
| --- |
| **它呈现一个只读Model的视图对象，而不是一特定的属性。后面会介绍如何用Model metadata来改变显示**    <%: Html.DisplayForModel() %> |

## Html. DisplayText

|  |
| --- |
| 效果与Htm.DisplatText一样，不过他会忽略所有的模版信息 |

## Html.Editor Html.EditorFor

|  |
| --- |
| 以编辑控件的形式显示Model某一个属性   <%: Html.Editor("FirstName") %>   <%: Html.EditorFor(x=>x.FirstName) %>  显示效果如下图所示 |

## Html. EditorForModel

|  |
| --- |
| 为整个实体显示编辑控件，不是针对某一个属性   <%: Html.EditorForModel() %>  显示效果如下图所示： |

## 用Model metadata 改变显示效果

### HiddenInput

|  |
| --- |
| **表示不显示该属性**  [HiddenInput(DisplayValue=false)]    public int PersonId { get; set; } |

### DisplayName

|  |
| --- |
| **表示属性显示的名称**  [DisplayName("First name")]     public string FirstName { get; set; }  **显示效果如下图所示** |

### UIHint

<http://www.dotblogs.com.tw/lastsecret/archive/2011/06/09/27568.aspx>

### 用parttialview做自定义模板和AdditionalViewData

<http://www.dotblogs.com.tw/lastsecret/archive/2010/07/11/16486.aspx>

#### 显示模板

|  |
| --- |
| 1．VIEW页面    <%:  Html.DisplayForModel("Person1")%> //person1是部分视图的名称  2．我们创建如下列模式的目录结构，记住如果是显示模板采用DisplayTemplates目录名称，如果是编辑模板则采用EditorTemplates目录名称。    3．然后我们创建一个名为Person1.ascx的部分视图模板，如果想要 Html.DisplayForModel()不指定模板名称则部分视图的名称应该跟Model名称一样。  4．在person1.ascx我们输入以下内容  <div class="person">    <img class="person-icon" src="/content/person-icon.png" />   <%: Html.ActionLink(Model.FirstName + " " + Model.LastName, "Edit", new { Model.PersonId }) %>      (born <%: Model.BirthDate.ToString("MMMM dd, yyyy") %>)    <div class="address">  // AddressSingleLine是模板名称       <%: Html.DisplayFor(x => x.HomeAddress, "AddressSingleLine") %>      </div>  </div>  5显示效果如下图所示 |

#### 编辑模板

|  |
| --- |
| 1．View页面：  <%: Html.EditorFor(x => x.FirstName, "FirstName", new { value = ViewData.Eval("FirstName") })%>  第二个参数指定模板的名称是指定模板名称（如果要指定模板名称，这个是不能省略的）  **第三个参数是要给模板传递的值，在模板里面可以用ViewData去获取。**  2．创建编辑部分视图    记住文件夹目录名称一定要是EditorTemplates。  3．<%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<System.String>" %>  自定义编辑模板：<%: Html.TextBox("", ViewData["value"])%>  或者可以采用  %: Html.TextBox("",  ViewData.TemplateInfo.FormattedModelValue)  %>    记住上面的类型一定要和View页面编辑的类型要一致。ViewData["value"]就是获取View页面传进来的值。  显示效果如下图所示 |

### ViewData.TemplateInfo

|  |
| --- |
| FormattedModelValue 一般用在模板页面中 表示实体的传过来的值 如上面的实例  GetFullHtmlFieldId 在模板中一般中如果指定了比如上面  <%: Html.TextBox("me", ViewData["value"])%>就会生成FirstName\_me的ID你调用ViewData.TemplateInfo.GetFullHtmlFieldId("me")就可以获取准确的ID了  GetFullHtmlFieldName  <%: Html.TextBox("me", ViewData["value"])%>就会生成FirstName.me的Name 你调用ViewData.TemplateInfo. GetFullHtmlFieldName ("me")就可以获取准确的Name了  HtmlFieldPrefix 获取input元素的前缀 例如上面就返回 FirstName  TemplateDepth 模板引用的深度    Visited |

### ModelMetadata

|  |  |  |
| --- | --- | --- |
| ModelMetadata这个类被设计来告诉你关于对象中你想用来显示或者编辑的信息.一般我们会在写模版是用到,这个元数据实际上那里都可以用的,即使不在模版中.  **属性和方法：** | | |
| 属性名称 | | 含义 | 用法 | |
| AdditionalValues | | 任意额外的元数据字典在自定义metadata provider中会使用到 | 用在自定义metadata provide和html helper中 | |
| ConvertEmptyStringToNull | | 标记forms中任何posted会服务端的空白字符串应该转换为空.默认:true | [DisplayFormat(ConvertEmptyStringToNull=true)]  public DateTime BirthDate { get; set; } | |
| DataTypeName | | 给字段熟悉提供进一步的含义信息 | [DataType(DataType.EmailAddress)]  public string Email { get; set; } | |
| **Description** | | 关于model描述 |  | |
| **DisplayFormatString** | | 用作在模版中显示model值的格式化的字符串. | [DisplayFormat(DataFormatString "{0:yyyy-MM-dd hh:mm:ss}")]  public DateTime BirthDate { get; set; } | |
| **DisplayName** | | 这个model值的显示名.在模版和Html.Label/LabelFor中用作label的text. | [DisplayName("姓?")]  public string FirstName { get; set; } | |
| EditFormatString | | 用作在模版中编辑model值的格式化的字符串 | [DisplayFormat(ApplyFormatInEditMode=true,DataFormatString = "{0:yyyy-MM-dd hh:mm:ss}")]  public DateTime BirthDate { get; set; } | |
| HideSurroundingHtml | | 用一个input hidden显示 | [HiddenInput(DisplayValue = false)]  public int PersonId { get; set; } | |
| IsReadOnly | | 只读 不接受任务新值 | [ReadOnly(true)]  public string LastName { get; set; } | |
| IsRequired | | 是否是必须的. | [Required] | |
| NullDisplayText | | 空值时候显示的文字，只在display中作用 | [DisplayFormat(NullDisplayText="邮®¨º件t为a空?")]  public string Email { get; set; } | |
| ShowForDisplay | | 是否显示 在编辑模版和显示模板中一样 | [ScaffoldColumn(false)]  public string LastName { get; set; }  如果为false就跳过显示 | |
| SimpleDisplayText | | 暂时没想到怎么用 |  | |
| TemplateHint | | 指定显示或者编辑该字段或者属性的模版名称 | [UIHint("Enum")]  public Role Role { get; set; } | |
| 内置的MVC 模板 | | | | |
| [AdditionalMetadata("RenderList", "true")]  public bool IsApproved { get; set; }  在模板中可以这么访问  @{  bool renderList = true;  if (ViewData.ModelMetadata.AdditionalValues.ContainsKey("RenderList")) {  renderList =  bool.Parse(ViewData.ModelMetadata.AdditionalValues["RenderList"].ToString());  }  } | | | | |

# Model绑定

## Model与传值

|  |
| --- |
| 前台view页面：    <% using (Html.BeginForm("RegisterMember", "Home"))         { %>      <div>          Name:          <%: Html.TextBox("myperson.FirstName")%></div>      <div>          Email address:          <%: Html.TextBox("myperson.Email") %></div>      <div>          Date of birth:          <%: Html.TextBox("myperson.BirthDate")%></div>      <input type="submit" value="提交" />      <% } %>  Action:  1.     public ActionResult RegisterMember()  {      var myperson = new Person();      myperson.FirstName = Request["FirstName"];      myperson.Email = Request["Email"];      myperson.BirthDate = DateTime.Parse(Request["BirthDate"]);      return View();  }  2.    public ActionResult RegisterMember(string FirstName, DateTime? BirthDate,string Email)          {            return View();          }  3.  public ActionResult RegisterMember(Person p)          {              return View();          }  第二种 action搜索值的顺序是以FirstName为例  1. Request.Form["FirstName"], if it exists  2. Otherwise, RouteData.Values["FirstName"], if it exists  3. Otherwise, Request.QueryString["FirstName"], if it exists  4. Otherwise, Request.Files["FirstName "]  5. Otherwise, null  第三种 注意前面表单的名字要跟实体类的属性名字一样，这样才能用反射得到值。 |

## 使用Model binding前缀

|  |
| --- |
| public ActionResult RegisterMember([Bind(Prefix = "newuser")] Person p)     {            return View(p);    }  那么view页面必须提供   <%: Html.TextBox("newuser.FirstName")%></div>  也就是带newuser的前缀名称的name |

## Model-Binding to Arrays, Collections, and Dictionaries

|  |
| --- |
| 1. 简单的例子：     <div>      <% Html.BeginForm("listDemo");%>      <%: Html.TextBox("movies")%>      <%: Html.TextBox("movies")%>      <%: Html.TextBox("movies")%>      <%: Html.TextBox("movies")%>      <input type=submit />      <% Html.EndForm(); %>  </div>   public ViewResult ListDemo(IList<string> movies)     {              return View();     }   1. 传实体类集合     <% using (Html.BeginForm("ListDemoForModel","Home"))         { %>      <h2>First person</h2>      <div>Name: <%: Html.TextBox("people[0].FirstName")%></div>      <div>Email address: <%: Html.TextBox("people[0].Email")%></div>      <div>Date of birth: <%: Html.TextBox("people[0].BirthDate")%></div>        <h2>Second person</h2>      <div>Name: <%: Html.TextBox("people[1].FirstName")%></div>      <div>Email address: <%: Html.TextBox("people[1].Email")%></div>      <div>Date of birth: <%: Html.TextBox("people[1].BirthDate")%></div>      <input type="submit" />  <% } %>  上面这种是按索引来的，**记住上面索引必须是按顺序来，不能跳数**  **如果不想按这种顺序索引来的话，可以采用自定义的形式如下面的代码所示**   <% using (Html.BeginForm("ListDemoForModel","Home"))         { %>  <h2>First person</h2>  <input type="hidden" name="people.index" value="someKey" />  <div>Name: <%: Html.TextBox("people[someKey].FirstName")%></div>  <div>Email address: <%: Html.TextBox("people[someKey].Email")%></div>  <div>Date of birth: <%: Html.TextBox("people[someKey].BirthDate")%></div>  <h2>Second person</h2>  <input type="hidden" name="people.index" value="anotherKey" />  <div>Name: <%: Html.TextBox("people[anotherKey].FirstName")%></div>  <div>Email address: <%: Html.TextBox("people[anotherKey].Email")%></div>  <div>Date of birth: <%: Html.TextBox("people[anotherKey].BirthDate")%></div>   <input type="submit" />  <% } %>  每个前面加个input hidden 但是name必须为people.index 然后value值提供给下面的 textbox使用  Action代码：public ViewResult ListDemoForModel(IList<Person> people)          {              return View(people);          }  3传字典类  <% using (Html.BeginForm("DictionaryPeople","Home"))  { %>  <h2>First person</h2>  <input type="hidden" name="people[0].key" value="firstKey" />  <div>Name: <%: Html.TextBox("people[0].value.FirstName")%></div>  <div>Email address: <%: Html.TextBox("people[0].value.Email")%></div>  <div>Date of birth: <%: Html.TextBox("people[0].value.BirthDate")%></div>  <h2>Second person</h2>  <input type="hidden" name="people[1].key" value="secondKey" />  <div>Name: <%: Html.TextBox("people[1].value.FirstName")%></div>  <div>Email address: <%: Html.TextBox("people[1].value.Email")%></div>  <div>Date of birth: <%: Html.TextBox("people[1].value.BirthDate")%></div>  <input type="submit" />  <% } %>  注意上面的 key和value  Action代码如下：     public ViewResult DictionaryPeople(Dictionary<string, Person> people)     {       return View(people);   } |

# MVC中提供器

MVC提供了四种默认的值提供器：

FormValueProvider：表单数据，对应于ASP.NET的Request.Form集合

QueryStringValueProvider：查询字符串，对应于ASP.NET的Request.QueryString集合

HttpFileCollectionValueProvider：文件集合，数据来源于Request.Files集合

RouteDataValueProvider：路由信息，对应于RouteData.Values集合

   MVC为每一个值提供器提供了一个工厂：ValueProviderFactory，ValueProviderFactoryCollection是一个值提供器工厂集合，其中的GetValueProvider方法返回与当前控制器上下文匹配的值提供器。ValueProviderFactories是一个静态类，内部封装一个ValueProviderFactoryCollection集合(Factories属性)，此集合默认包含MVC中4个默认的值提供器，所以，如果我们要使用自己的值绑定器，可以通过此类的Factories属性，使用InsertItem增加新的值提供器，使用SetItem方法将某个默认值提供器替换为我们的自定义值提供器。 需要注意的是，ValueProviderFactoryCollection的GetValueProvider方法实际返回的是一个ValueProviderCollection集合，但由于此集合类同样实现了IValueProvider接口(其GetValue方法会从集合中所有的值提供器中查找符合条件的项)，所以GetValueProvider的返回类型仍然为IValueProvider

|  |
| --- |
| public class CookieValueProviderFactory : ValueProviderFactory      {          public override IValueProvider GetValueProvider(ControllerContext controllerContext)          {              return new CookieValueProvider(controllerContext.HttpContext.Request.Cookies);          }          private class CookieValueProvider : IValueProvider          {              private HttpCookieCollection Cookies;              public CookieValueProvider(HttpCookieCollection cs)              {                  Cookies = cs;              }              public bool ContainsPrefix(string prefix)              {                  return Cookies.AllKeys.Contains(prefix);              }              public ValueProviderResult GetValue(string key)              {                  if (!ContainsPrefix(key))                      return null;                  string value = Cookies[key].Value;                  return new ValueProviderResult(value, value, CultureInfo.CurrentCulture);              }          }  3、在Application\_Start中注册自定义值提供器工厂  ValueProviderFactories.Factories.Insert(0, new CookieValueProviderFactory());  控制器如下：   public ActionResult Index(DateTime? lastTime)          {                if (HttpContext.Request.Cookies.Get("lastTime") == null)              {                    Response.Cookies.Add(new HttpCookie("lastTime", DateTime.Now.ToString()));                }                return View();            }  当刷新的时候lastTime就可以获取到值 |

# 其他知识点

## LayOut

<http://weblogs.asp.net/scottgu/archive/2010/10/22/asp-net-mvc-3-layouts.aspx>

<http://weblogs.asp.net/scottgu/archive/2010/12/30/asp-net-mvc-3-layouts-and-sections-with-razor.aspx>

VIEWBAG

<http://www.cnblogs.com/damonlan/archive/2011/09/11/MVC3.html>

数据库脚本USE [safsafsaf]

GO

/\*\*\*\*\*\* 对象: Table [dbo].[Products] 脚本日期: 12/07/2011 17:23:10 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[Products](

[ProductID] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](100) NOT NULL,

[Description] [nvarchar](500) NOT NULL,

[Category] [nvarchar](50) NOT NULL,

[Price] [decimal](16, 2) NOT NULL,

[ImageData] [varbinary](max) NOT NULL,

[ImageMimeType] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Products] PRIMARY KEY CLUSTERED

(

[ProductID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO