1. Control负责把数据赋值给View:1)使用ViewBag来完成该任务。ViewBag是一个动态对象，可以给它的任意属性赋值。2)使用强类型视图。@model 权限定类.使用Model来在视图中引用强类型数据。
2. 加[HttpGet]或者[HttpPost]属性来使Action来只处理get或者post请求。
3. If语句： @if(){} else{} @switch(){} @foreach{}
4. 数据有效性验证，在System.ComponentModel.DataAnnotations命名空间，Required,RegularExpression注释。
5. Layout=null告诉MVC框架，视图是负责render全部内容。Layout负责给网站提供一致的界面，相当于MasterPage。在Views文件夹下以\_开头的文件，是不能被用户访问的。因此Layout文件应该是以\_开头的。@RenderBody()
6. Views\\_ViewStart.cshtml的内容就像它们被包含在视图的开始部分，（它先于视图执行,意味着可以在视图中覆盖——\_ViewStart的值）。即把视图的公共部分放到一个地方。
7. Razor表达式可以设置html属性。@:阻止Razor引擎将其翻译为C#语句。
8. @using在视图中引入命名空间。
9. Ninject是一个超轻量级，快速的DI以来注入容器。Entity是ORM tool.
10. 在Razor [view使用的类要么使用@using语句，要么在web.config](mailto:view使用的类要么使用@using语句，要么在web.config)声明。
11. Partial view是重用于嵌入其它view的。使用Html.Render Partial或者RenderAction来render partial view.
12. TempData temp data is deleted at the end of the HTTP request or it is readed. 重定向是要求浏览器重新发送一个请求。可以通过TempData传递数据给redirect的url.
13. routes.MapRoute(

name: "Default",

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

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

);

routes.MapRoute("ShopSchema2", "Shop/OldAction", new { controller = "Home", action = "Index" });

routes.MapRoute("ShopSchema", "Shop/{action}", new { controller = "Home" });

routes.MapRoute("", "X{controller}/{action}");

routes.MapRoute("MyRoute", "{controller}/{action}", new { controller = "Home", action = "Index" });

routes.MapRoute("", "Public/{controller}/{action}", new { controller = "Home", action = "Index" });

Route并没有特殊的含义，是后面的mvc request handle给予了controller和action含义。

1. mvc框架将url segment参数与action参数比较，如果名字一致，则将action参数赋值，并进行类型转换。

**routes.MapRoute("MyRoute", "{controller}/{action}/{id}/**{\*catchall}**",**

**new { controller = "Home", action = "Index", id = UrlParameter.Optional });**

\*catchall捕获所有的segment.catchall段表示后面所有的segment.

Route myRoute **= routes.MapRoute("AddContollerRoute",**

**"Home/{action}/{id}/{\*catchall}",**

**new { controller = "Home", action = "Index",**

**id = UrlParameter.Optional },**

**new[] { "URLsAndRoutes.AdditionalControllers" });**

myRoute.DataTokens["UseNamespaceFallback"] = false;

可以添加命名空间解决controller名字冲突问题。MVC框架首先查找指定的命名空间，再查其它的。设置usenamespacefallback=false之后，只搜索指定的命名空间。

**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" });**

**正则表达式限制。**Default values are used before constraints are checked.

1. 如果标准的限制不够，你可以通过实现IRouteConstraint定义自己的限制。
2. 路由系统首先检查url文件是否是磁盘文件。
3. Html.ActionLink用来生成outgoing url.生成外部url也是按照route的定义顺序来匹配的。如果没有匹配，生成空的url。 值对每个segment段都必须存在，为了查找值，mvc首先查看我们提供的值，当前请求中的值，最后是route定义中的默认值。我们提供的值与默认值有任何不匹配，则该route识别失败。
4. 在名字前面加@，可以使用保留字命名变量。
5. Url.Action显示url. RedirectToAction重定向url(浏览器url会变).
6. MVC把应用程序分为area, area代表一个功能段。Area的导航优先（取决于AreaRegistration.RegisterAllAreas();调用顺序）。
7. MVC框架给action方法参数赋值时，从Request.QuestString、Request.Form、RouteData.Values来获取值，并且参数名是大小写不敏感的。如果找不到可null类型的值，提供null,否则抛出异常，除非你指定默认值。
8. Post/redirect/get方法。Get只负责获取数据。
9. Filter:Filter是用于所有位置的具有相同功能的代码。Filter是.net属性增加额外的步骤到请求处理管道。
10. MVC提供的四种Filter. Authorize(在action和其它Filter之前运行), Action(在action之前或之后运行)，Result,Exception.
11. HandleErrorAttribute: when specified by ExceptionType is encountered, this filter will render the view specified by the View property (using the default layout or the one specified by the Master property).
12. GLOBAL filter应用于所有的controller的action。
13. Helper 方法:自定义helper方法，

@helper ListArrayItems(string[] items) {

foreach(string str in items) {

<b>@str </b>

}

}

1. 创建外部的helper方法：

public static MvcHtmlString ListArrayItems(this HtmlHelper html, string[] list)

{

TagBuilder tag = new TagBuilder("ul");

foreach (string s in list)

{

TagBuilder itemTag = new TagBuilder("li");

itemTag.SetInnerText(s);

tag.InnerHtml += itemTag.ToString();

}

return new MvcHtmlString(tag.ToString());

}