Routing概念

站在应用的角度上看，这绝对是个非常简单的，因为应用程序中只需要寥寥几行代码就可以了！所以让我们从本质的角度上去了解，认清它的工作机制。  
  
从简单开始吧：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public static void RegisterRoutes(RouteCollection routes)  {  routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");  routes.MapRoute(  "Default", *// Route name*  "{controller}/{action}/{id}", *// URL with parameters*  new { controller = "Home", action = "Index", id = "" } *// Parameter defaults*  );  }    protected void Application\_Start()  {  RegisterRoutes(RouteTable.Routes);  } |

应用程序启动的时候，将自定义的路由信息加到RouteTable的路由集合中。

当编写完这些后，应用就结束了，但是从本质上只是一个开始，在此三个问题：

1、路由的集合中都是些什么数据？

asp.net mvc routing原理

这里最重要的是Route对象，因为我们设置的数据都是作为该对象的属性，例如：上面的Routename、URL…，然而它的作用就是根据这些属性和请求路径去构造RouteData对象。

构造Route对象的方式有两种：

1、new Route(…)，构造该对象，并使用RouteTable.Routes.Add(routeObj)来加入集合。

2、RouteCollectionExtensions.IgnoreRoute或MapRoute方式来构造Route对象，并加入集合。

大家可以注意到了，IgnoreRoute和MapRoute这两个扩展方法，有什么不同呢？！先看下Route的构造函数，构造的时候会有一个必须的参数IRouteHandler：

[](http://www.foxitjob.com/wp-content/uploads/2010/12/mvc1_2.jpg)

asp.net mvc routing原理

IgnoreRoute方法构造StopRoutingHandler作为参数，而MapRoute方法构造MvcRouteHandler作为参数．而这两个IRouteHandler之间的区别通过下面的代码就清晰的看到，

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | *//MvcRoutingHandler的实现*  protected virtual IHttpHandler GetHttpHandler(RequestContext requestContext)  {  return new MvcHandler(requestContext);  }    *//StopRoutingHandler的实现*  protected virtual IHttpHandler GetHttpHandler(RequestContext requestContext)  {  throw new NotSupportedException();  } |

在构造完Route对象后，对于它的利用主要就是GetRouteData方法，即根据HttpContextBase参数（下面阐述）和Route对象中的属性来构造并获取RouteData对象，

GetRouteData

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | public override RouteData GetRouteData(HttpContextBase httpContext)  {  string virtualPath = httpContext.Request.AppRelativeCurrentExecutionFilePath.Substring(2) + httpContext.Request.PathInfo;  RouteValueDictionary values = this.\_parsedRoute.Match(virtualPath, this.Defaults);  if (values == null)  {  return null;  }    RouteData data = new RouteData(this, this.RouteHandler);  if (!this.ProcessConstraints(httpContext, values, RouteDirection.IncomingRequest))  {  return null;  }    foreach (KeyValuePair pair in values)  {  data.Values.Add(pair.Key, pair.Value);  }    if (this.DataTokens != null)  {  foreach (KeyValuePair pair2 in this.DataTokens)  {  data.DataTokens[pair2.Key] = pair2.Value;  }  }  return data;  } |

2、路由数据在整个WEB生命周期中扮演着什么角色？

清楚了集合中的数据，那么就解决第二个问题，先看下WEB的生命周期：

[](http://www.foxitjob.com/wp-content/uploads/2010/12/mvc1_3.jpg)

asp.net mvc routing原理

然后，我们再去看下UrlRoutingModule这个类，该类就是扩展了PostResolveRequestCache和PostMapRequestHandler事件，即对于Route的用途就在这两个事件中．下面让我们从源代码上去了解该事件到底有什么．

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | private void OnApplicationPostMapRequestHandler(object sender, EventArgs e)  {  HttpContextBase context = new HttpContextWrapper(((HttpApplication) sender).Context);  this.PostMapRequestHandler(context);  }    private void OnApplicationPostResolveRequestCache(object sender, EventArgs e)  {  HttpContextBase context = new HttpContextWrapper(((HttpApplication) sender).Context);  this.PostResolveRequestCache(context);  } |

在这两个事件执行的时候都会去构建HttpContextBase对象，然后作为参数传入后面的方法．在处理IHttpHandler之前是执行 PostResolveRequestCache方法．该方法通过GetRouteData获取RouteData，并通过RouteData的 RouteHandler获取IRouteHandler，如果是StopRoutingHandler就执行完成，如果不是，则将执行 UrlRoutingHandler．

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public virtual void PostResolveRequestCache(HttpContextBase context)  {  RouteData routeData = this.RouteCollection.GetRouteData(context);  if (routeData != null)  {  IRouteHandler routeHandler = routeData.RouteHandler;  if (routeHandler == null)  {  throw new InvalidOperationException(string.Format(CultureInfo.CurrentUICulture, RoutingResources.UrlRoutingModule\_NoRouteHandler, new object[0]));  }    if (!(routeHandler is StopRoutingHandler))  {  RequestContext requestContext = new RequestContext(context, routeData);  IHttpHandler httpHandler = routeHandler.GetHttpHandler(requestContext);  if (httpHandler == null)  {  throw new InvalidOperationException(string.Format(CultureInfo.CurrentUICulture, RoutingResources.UrlRoutingModule\_NoHttpHandler, new object[] { routeHandler.GetType() }));  }  RequestData data2 = new RequestData();  data2.OriginalPath = context.Request.Path;  data2.HttpHandler = httpHandler;  context.Items[\_requestDataKey] = data2;  context.RewritePath("~/UrlRouting.axd");  }  }  } |

执行完成IHttpHandler后，就要执行PostMapRequestHandler方法，该方法做的事情很简单，就是重写下请求路径，让输出的路径和输入的路径相同，在这里用来记忆输入路径的是context.Items[]，从上下两段代码中可以看到。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public virtual void PostMapRequestHandler(HttpContextBase context)  {  RequestData data = (RequestData) context.Items[\_requestDataKey];  if (data != null)  {  context.RewritePath(data.OriginalPath);  context.Handler = data.HttpHandler;  }  } |

3、请求的Url和定制Routing中的Url之间在哪里做检测的，如何检测？

我们只要了解两个执行动作就可以了：

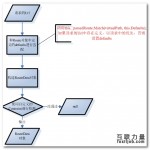
1、设置Route对象的Url，如下图，在设置Url的动作中作了如下动作，并将输出的ParseRoute对象设置到Route对象中的内部属性\_parsedRoute

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public string Url  {  get  {  return (this.\_url ?? string.Empty);  }    set  {  this.\_parsedRoute = RouteParser.Parse(value);  this.\_url = value;  }  } |

[](http://www.foxitjob.com/wp-content/uploads/2010/12/mvc1_4.jpg)

asp.net mvc routing原理

2、在PostResolveRequestCache方法中的RouteData routeData = this.RouteCollection.GetRouteData(context)，即根据请求的Url来和设定的Routing作比较，并获取 RouteData，这里可以看上面的GetRouteData代码，图示如下：

[](http://www.foxitjob.com/wp-content/uploads/2010/12/mvc1_5.jpg)

asp.net mvc routing原理