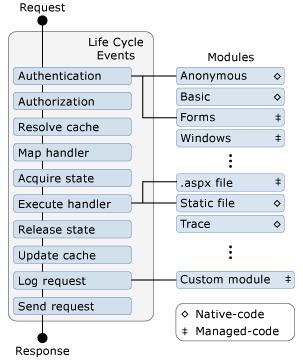
* **ASP.NET MVC生命周期介绍**
* <http://developer.51cto.com>  2009-07-23 18:14  me-sa  [cnblogs](http://www.cnblogs.com/me-sa/archive/2009/06/01/MVCLifecycle.html" \t "_blank)  [我要评论(0)](http://www.51cto.com/php/feedbackt.php?id=138884)

本文以IIS7中asp.net应用程序生命周期为例，介绍了asp.net mvc的生命周期。

asp.net应用程序管道处理用户请求时特别强调"时机"，对asp.net生命周期的了解多少直接影响我们写页面和控件的效率。对于asp.net mvc,我对它的生命周期兴趣很浓，于是对ASP.NET MVC生命周期提出两个问题：

一个HTTP请求从IIS移交到asp.net运行时,asp.net mvc是在什么时机获得了控制权并对请求进行处理呢?处理过程又是怎样的?

[](http://images.51cto.com/files/uploadimg/20090723/1839190.jpg)

[](http://images.51cto.com/files/uploadimg/20090723/1839191.jpg)

以IIS7中asp.net生命周期为例，上图是来自MSDN的一张HTTP请求处理过程发生事件的简图，后面我列出了一个完整的事件列表。既然 asp.net mvc还是以asp.net运行时为基础那么它必然要在asp.net应用程序的生命周期中对请求进行截获。第一反应当然是去web.config里面去 翻翻，我们可以看到UrlRoutingModule的配置节：

< add name="UrlRoutingModule" type="System.Web.Routing.UrlRoutingModule, System.Web.Routing, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

下面要做的就顺理成章了，用Reflector打开这个程序集,可以看到以下代码：

* 1. protected virtual void Init(HttpApplication application)
  2. {
  3. application.PostResolveRequestCache += new EventHandler(this.OnApplicationPostResolveRequestCache);
  4. application.PostMapRequestHandler += new EventHandler(this.OnApplicationPostMapRequestHandler);
  5. }

看到这里我们的第一个问题实际上已经有了答案：时机是在PostResolveRequestCache和PostMapRequestHandler.

我们使用VS2008中asp.net mvc模板创建一个Demo完成后续的讨论，当我们访问/Home的时候发生了什么呢？

1、Request 请求到来

2、IIS 根据请求特征将处理权移交给 asp.net

3、UrlRoutingModule将当前请求在 Route Table中进行匹配

4、UrlRoutingModule在RouteCollection中查找Request匹配的RouteHandler,默认是mvcRouteHandler mvcRouteHandler 创建 mvcHandler实例.

5、mvcHandler执行 ProcessRequest.

6、mvcHandler 使用 IControllerFactory 获得实现了IController接口的实例,找到对应的HomeController

7、根据Request触发HomeController的Index方法

8、Index将执行结果存放在ViewData

9、HomeController的Index方法返回 ActionResult

10、Views/Home/Index.aspx将 ViewData呈现在页面上

11、Index.aspx执行ProcessRequest方法

12、Index.aspx执行Render方法 输出到客户端

通过阅读asp.net mvc的源码,我们可以得到更为详细的处理过程,我尽可能的忽略掉枝节，强调请求处理的流程.我们从Global.asax.cs文件切入,下面是一段样例代码,这里初始化了路由表,请特别特别注意注释部分:

* 1. public class mvcApplication : System.Web.HttpApplication
  2. {
  3. public static void RegisterRoutes(RouteCollection routes)
  4. {
  5. routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");
  7. //The controller route value is a special value that the System.Web.mvc.mvcHandler class uses to call into the IControllerFactory interface.
  8. //The basic route handler is an instance of IRouteHandler named mvcRouteHandler.
  9. //We have complete control and could provide our own implementation of IRouteHandler if we wished.
  10. routes.MapRoute(
  11. "Default",                                              // Route name
  12. "{controller}/{action}/{id}",                           // URL with parameters
  13. new { controller = "Home", action = "Index", id = "" }  // Parameter defaults
  14. );

  17. }
  19. protected void Application\_Start()
  20. {
  21. RegisterRoutes(RouteTable.Routes);
  22. }

UrlRoutingMoudule在PostResolveRequestCache阶段从RouteCollection中获取当前请求的 RouteData.RouteData包含了一个请求处理对应的Controller和Action,RouteData这个作用贯穿请求的处理过 程.RouteData中提取RouteHandler,这里默认是mvcRouteHandler,mvcRouteHandler获取 HttpHandler,这里默认的是mvcHandler.

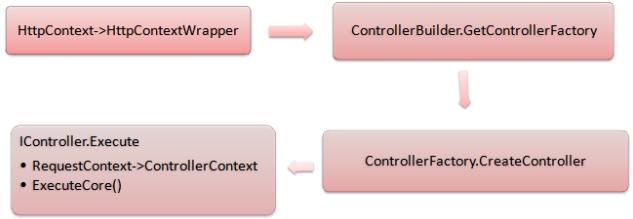
* 1. PostResolveRequestCache
  2. public virtual void PostResolveRequestCache(HttpContextBase context)
  3. {
  4. RouteData routeData = this.RouteCollection.GetRouteData(context);
  5. if (routeData != null)
  6. {
  7. IRouteHandler routeHandler = routeData.RouteHandler;
  8. if (routeHandler == null)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

[](http://images.51cto.com/files/uploadimg/20090723/1839192.jpg)

mvcHandler.ProcessRequest()中首先使用HttpContextWrapper对HttpContext进行封装,封装的目的是为了解耦以获得可测试性.然后从RequestContext.RouteData中提取Controller名称.

ControllerBuilder.GetControllerFactory --> ControllerFactory.CreateController --> IController.Execute

ControllerBase实现了IController接口,在Initialize时将RequestContext封装成为ControllerContext,Controller继承自ControllerBase并实现抽象方法ExecuteCore()

[](http://images.51cto.com/files/uploadimg/20090723/1839193.jpg)

在ExecuteCore中,Controller首先从RouteData中获得ActionName,然后执行ActionInvoker.InvokeAction.

在ActionInvoker中我们可以看到各种Filter,这是一种AOP实践:在Action方法执行的前后执行若干方法.这里有四种 Filter:ActionFilters,ResultFilters,AuthorizationFilters,ExceptionFilters. 这四种Filter并不是封闭的,都有对应的接口,这四个只是默认实现.Filter的执行顺序 是:AuthorizationFilter--->Action Filter.OnActionExecuting--->Action Method--->ActionFilter.OnActionExecuted.InvokeActionMethodWithFilters 返回的结果是ActionExecutedContext,接下来将Controller执行OnResultExecuting 方法.ActionResult执行的结果可以是 ViewResult,JsonResult,RedirectResult,ContentResult,或者是自定义的Result类型.

如果返回的类型是ViewResult,我们先看一下ViewReuslt的继承关 系:ViewResult-->ViewResultBase-->ActionResult,ViewResult包含两个属性View和 ViewEngineCollection,实际上是包含了两个接口的实现:IViewEngine定义了怎么定位View/Partial View.IView定义了如何RenderView.默认的实现时WebFormView和WebFormViewEngine.

Filter OnResultExecuted 最后一步了,可以这里捕获异常.上面我们说过还有ExceptionFilters,如果前面过程中的异常没有被捕获那么最终都会到冒泡到ExceptionFilters.

RouteData中获得ActionName

ActionInvoker.InvokeAction

通过ControllerContext获取ControllerDescriptor

FindAction-获取ActionDescriptor

GetFilters

ModelBinder把Request中的数据转换成Action方法需要的参数

AuthorizationFilter

Action Filter.OnActionExecuting

Action

ActionFilter.OnActionExecuted

ResultFilter.OnResultExecuting

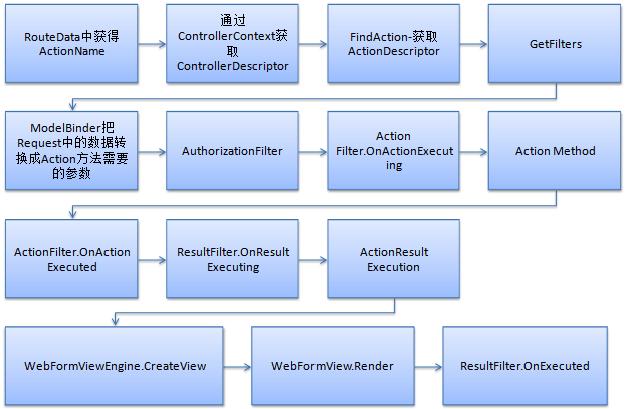
ActionResult Execution

ResultFilter.OnResultExecuted

WebFormViewEngine.CreateView

WebFormView.Render

ResultFilter.OnExecuted

[](http://images.51cto.com/files/uploadimg/20090723/1839194.jpg)

控制权归还到HttpApplication完成后续的asp.net mvc生命周期