到了View的呈现板块，感觉ASP.NET MVC的学习也进入了尾声，还是比较开心的，毕竟也有了不小收获。这部分内容相对比较简单，因为之前还专门学习过如何结合HTML5与MVC框架。前文中提到过，ActionResult的执行实际就是View呈现的入口。不同类型的ActionResult比较多，以下做个简要的介绍。

|  |  |
| --- | --- |
| 类型 | 简介 |
| EmptyResult | 最简单的Result，只是为了符合框架的流程。 |
| ContentResult | 可以设置ContentType为javascript, CSS等，默认将基本类型的返回值转化为ContentResult。 |
| FileResult | 包含FileContentResult, FilePathResult, FileStreamResult |
| JavascriptResult | 在服务端动态生成javascript并作为请求的相应。 |
| JsonResult | 提供对象和Json字符串的序列化与返序列化 |
| HttpStatusCodeResult | 方便返回Http状态码 |
| RedirectResult  RedirectToRouteResult | 前者是我们可以直接重定向到指定的目标地址，后者帮助我们根据注册的路由进行重定向，不提供URL参数，而提供RouteName和RouteValues。类似httpResponse的Redirect/RedirectPermanent，在SEO中，前者会使永久重定向目的地址更新为搜索引擎的索引，后则不会。 |

接下来介绍最重要的ViewResult以及VIewEngine的概念。View实现IView接口，其实就是一个Render方法，参数ViewContext除了常见属性外，还有两个与js相关属性(CleintValidationEnabled, UnobstrusiveJavascriptEnabled)，特别的是object类型的ViewBag用了[Dynamic]特性。ViewEngine也实现了相应的接口，注意View缓存，SearchedLocations属性包含了所有的查找路径，FindView方法的的返回名称比较奇怪，ViewEngineResult，但实际的意思应该是这个Result包含View和ViewEngine信息。在另一篇管自定义ViewEngine的博文中对这一块进行了更加深入的介绍。

再则简要的介绍下Razor引擎，我们知道.cshtml文件并不能直接执行，必须先动态编译后才能使用，那么问题来了，编译成什么文件的文件名是什么，所属的程序集为什么？并存放在哪呢？

文件名为：ASP\_Page\_views\_XXX\_ActionXXX\_cshtml

程序集名由代码可知：

|  |
| --- |
| public static MvcHtmlString ListViewAssemblies(this HtmlHelper helper)  {  TagBuilder ul = new TagBuilder("ul");  foreach (var assembly in AppDomain.CurrentDomain.GetAssemblies().Where(a => a.FullName.StartsWith("App\_Web\_")))  {  //omit  }  return new MvcHtmlString(ul.ToString());  } |

编译的临时文件的路径为：

”%WinDir%\Microsoft.NET\Framework\{VersionNo}\Temporary ASP.NET Files\”

可以通过如下配置，变更路径：

|  |
| --- |
| <system.web>  <compilation tempDirectory="C:\XXXXXXXX"/>  </system.web> |

补充一个获得编译后类型和程序集的方法，使用BuilderManager类型的静态方法GetCompiledType和GetCompiledAssembly。

View文件编译生成的类型为WebViewPage<TModel>，也就是说.cshtml编译后的类型均为以上类型。其基类WebPageExecutingBase包含Execute, Write, WriteLiteral等方法。该基类的另一个自雷WebPageRendingBase包含Layout, IsAjax, Server, Session, Profile, Cache等属性，以及输出整个页面的抽象方法ExecutePageHierarchy。

WebViewPage还有一个直接基类WebPageBase，它实现类抽象方法ExecutePageHierarchy，最终呈现的页面包含3部分内容：布局文件、开始页面View本身。蒋老师提供的一个不错的布局例子代码如下所示：

|  |
| --- |
| //布局文件定义  <!DOCTYPE html>  <html>  <head>  <title>@ViewBag.Title</title>  </head>  <body>  @{if (IsSectionDefined("Header"))  {RenderSection("Header");}  else  {<h2>@ViewBag.Title</h2>}  @RenderBody()  @{ if(IsSectionDefined("Footer")){  @RenderSection("Footer");  }  else{<hr/><p>@2012 Xixi. All rights reserved.</p>}  }}  </body>  </html>  //View的定义  <div>  <p>Bilibili</p>  </div>  @section Header  {<h2>Bili bili</h2>} |

看到这个我才真正明白了以前在代码中的RenderBody是如何使用的，布局文件和View是如何结合的。

最后用一张关于Razor引擎的图让大家有个比较清晰的印象。



注：本文主要供自己学习，不妥之处望见谅。

参考资料：

[1]蒋金楠. ASP.NET MVC4框架揭秘[M]. 上海:电子工业出版社, 2012. 390-444