虽然使用MVC已经不少年，相关技术的学习进行了多次，但是很多技术思路的理解其实都不够深入。其实就在MVC框架中有很多设计模式和设计思路的体现，例如DependencyResolver类就包含我们常见的DI依赖注入概念和注册表模式(GetService)等内容，ExceptionFilter等过滤器就体现AOP的概念，整个MVC内置了一个IOC容器，基本上所有的框架类的对象都是通过这种方式来创建的。此外，一直觉得很j2ee的spring很棒，其实如果大家很熟悉EHAB（微软企业开发库）的话，就会发现它很想Spring了，当然在一些细节上，还是不如Spring方便。

由于是概述，所以内容涉及知识点会很多，但不会很深入，就让我们来对MVC框架有个基本的认识吧，一些不太有特色的知识点不介绍了哈。

终于完成MVC系列的学习，想想挺有意思的，概述居然是最后一个完成的，其实想想也对，最后的总结往往比开头的概述要来的深刻，本系列从去年9月开始，也是我刚开始学着写技术博文，半年过去了，感觉自己还是有一些进步的，加油，熊二！鉴于这部分学习的很多章节的行文显得不够圆润和准确，就不放在首页了，不过可以通过目录访问。此外，俺也会不断的修订这部分内容，使其的实用价值更大。

快速入门系列--MVC--01概述

[快速入门系列--MVC--02路由](http://www.cnblogs.com/wanliwang01/p/ASP_NET_MVC_URLRoute.html)

[快速入门系列--MVC--03控制器和IOC应用](http://www.cnblogs.com/wanliwang01/p/ASP_NET_MVC_IOC.html)

[快速入门系列--MVC--04模型](http://www.cnblogs.com/wanliwang01/p/ASP_NET_MVC_Model.html)

[快速入门系列--MVC--05行为](http://www.cnblogs.com/wanliwang01/p/ASP_NET_MVC_Action.html)

[快速入门系列--MVC--06视图](http://www.cnblogs.com/wanliwang01/p/ASP_NET_MVC_View.html)

[快速入门系列--MVC--07与HTML5移动开发的结合](http://www.cnblogs.com/wanliwang01/p/MVC_HTML5_Mobile.html)

过去ASP.NET开发使用的是WebForm，其利用拖放服务器控件，有用的状态(semi-magical statefulness)来处理后台的复杂事务，大家应该都有被ViewState(将表单体进行base64编码，存储在hidden元素中)折磨的经历。而MVC是基于无状态的Web环境的，每一次请求都会有一个完整的生命周期，具体来讲就是每一个请求都是一个新的Controller对象来负责，其遵循约定优于配置(来自Ruby on rails的概念)、支持可插拔性(框架中类均能自定义扩展)等原则。

* 比较有趣的特性：

新的Razor视图引擎，用于生成HTML代码模板，和它类似的有JSP页面的JSTL, EL, Velocity等，其在IDE中的只能感知真心是很棒，编写代码起来非常的顺畅。

支持IValidatableObject接口便于扩展模型的验证，通过设置html元素的属性为data-xxx来实现验证，非常的具有非侵入式特性，此外很好的于JQuery兼容。

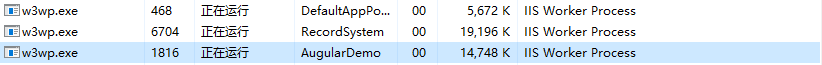
提供的新的MVC模板非常的棒棒哒，包括对多种不同客户端的支持，与JQuery Mobile的结合，以及一个Recipes包(Nuget)用于将现有页面转化为支持移动端的版本的组件。此外还有当前最为流行的单页应用程序的模板。

通过BundleConfig将CSS， JS打包压缩减少请求次数和流量，其所在的App\_Start目录包含了大部分的配置逻辑。

整合Json.NET、DotNetOpenAuth(包括一个OAuthWebSecurity的简化封装)等开源库。

* IIS（微软的web服务器）相关概念：

在IIS中，当检测到某个HTTP请求时，如果请求的是静态资源则以http回复的形式返回；如果是动态资源，则通过ISAPI动态连接库处理，它被加载到InetInfo.exe进程中，包括相关的元数据。在IIS为解决应用程序间的相互干扰，通过一个应用程序池的机制，为每一个应用程序池建立一个独立的工作线，提供进程的隔离级别的保护，如下图所示w3wp.exe就是相应的工作线程，需要注意的是，默认情况下w3wp.exe进程在一段时间不用后会休眠，在查看时需要注意唤醒。这部分概念在生产系统中，显得比较重要，一些奇葩的高并发、多线程问题都可能与这部分相关。



IIS6.0的基本架构形式



当HTTP.SYS监听到用户的HTTP请求后，分发到W3SVC，W3SVC解析出请求的URL，并根据从Metadata获取的URL与Web应用之间的映射关系得到目标应用，并进一步得到目标应用运行的AppicationPool或工作线程。如果不存在则创建，在其初始化过程中，相应的ISAPI动态连接库(aspnet\_isapi.dll)被加载，之后其负责CLR的加载、AppDomain的创建和应用的初始化。

IIS7.0添加了一个Windows进程激活服务(Windows Process Activation Servcie, WAS)，用于根据请求的不同协议类型来激活不同的windows服务（覆盖WCF的4中主要协议）。此外IIS7.0将IIS管道和ASP.NET管道有机的整合在了一起，允许本地代码和托管代码两种方式定义IIS Module，形成一个通用管道，例如可以将Form认证应用到静态文件的请求上等。

* ASP.NET管道

大家原来做过WebForm都应该有印象，不管是面试还是实践中，ASP.NET页面的生命的周期是一个非常常见的问题，其实这就是一个请求在管道中的一部分处理过程。接下来，对整个流程（初次请求，发布时也叫点火）做个简单的介绍。

第1步：当IIS接受到请求后，加载对应处理DLL后，会通过AppDomain创建一个应用程序域，随后一个特殊的运行时IsapiRuntime(System.Web.Hosting)被加载。

第2步：它首先会创建一个用于封装请求的IsapiWorkerRequest对象，之后将该对象传递给HttpRuntime，进入ASP.NET管道，HttpRuntime建立相应的HttpContext。

第3步：接着利用HttpApplicaitonFactory创建或获取新的HttpApplication对象（存在一个HttpApplication对象池），其初始化时会根据配置文件加载并初始化相应的HttpModule对象，该对象包含关于生命周期的大量方法，我们通过实现这些方法来完成很多类似面向切面的很多功能，如验证&授权、缓存、日志和统计信息等。

第4步：最终由HttpHandler对象完成请求处理。

HttpApplication对象是基础，由于其某一时刻只能处理一个请求，因而使用对象池的机制来进行管理，其相关事件列表如下。（为了这辈子再不被问这个，还是写了吧！）

|  |  |
| --- | --- |
| 名称 | 描述 |
| BeginRequest | 开始处理请求 |
| AuthenticationRequest,, PostAuthenticationRequest | 安全模块对请求进行身份验证 |
| AuthorizeRequest, PostAuthorizeRequest | 安全模块对请求授权 |
| ResolveRequestCache, PostResolveRequestCache | 使缓存模块利用缓存内容直接响应 |
| PostMapRequestHandler | 对于不同的访问资源，使用不同的HttpHandler进行处理，通关过扩展名选择匹配 |
| AcquireRequestState, PostAcquireRequestState | 获取当前请求状态，如SessionState |
| PreRequestHandlerExecute, PostRequestHandlerExecute | 请求处理的核心，HttpHandler的执行 |
| ReleaseRequestState, PostReleaseRequestState | 使状态管理模块释放当前请求的相应状态 |
| UpdateRequestCache, PostUpdateRequestCache | 使缓存模块将请求处理结果的内容保存到缓存(均是使用特性，进行声明式的控制) |
| LogRequest, PostLogRequest | 为当前请求纪录日志 |
| EndRequest | 请求完成 |

HttpModule负责将请求于请求资源类型相匹配的HttpHandler对象进行映射，其初始化时，会将一些功能注册到HttpApplication相应的事件中。典型的HttpModule包括：实现缓存的OutputCacheModule；在无状态HTTP协议上实现基于会话状态的SessionStateModule；实现Windows、Forms、Passport三种典型身份验证的相关模块；实现基于URI和文件ACL授权的UrlAuthorizationModule和FileAuthorizationModule。此外我们还可以通过配置将自定义的httpModule加入其中，例子为<modules><add name=”xx” type=”xx”>。

最终真正处理请求的HttpHandler对象通过ProcessRequest方法处理请求，其相关配置例子为<httpHandlers><add path=”xx.svc” verb=”\*” type=”xxxx” validate=”false”>。

之前介绍了很多关于MVC的基础，接下来通过一个简单的模拟示例来了解MVC框架的实现（参考蒋大神的示例）。该示例展示了请求的路由，Controller的构建与激活，Action的执行，同时穿插介绍了Model绑定器的实现。由于内容比较多，压缩包路径为：to do。

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.