第一图书网, tushu007.com

<<ASP.NET MVC 4框架揭秘>>

图书基本信息

书名：<<ASP.NET MVC 4框架揭秘>>

13位ISBN编号：9787121190490

10位ISBN编号：7121190494

出版时间：2013-1

出版时间：电子工业出版社

作者：蒋金楠

页数：578

字数：855000

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

前言

ASP.NET MVC是一个建立在ASP.NET平台上基于MVC模式的Web开发框架，它提供了一种与传统Web Forms完全不同的Web应用开发方式。
ASP.NET Web Forms借鉴了Windows Forms基于控件和事件注册的编程模式，使Web应用的开发变得简单而快捷，但是它却使开发人员与Web的本质渐行渐远。
ASP.NET MVC是一种回归，它使开发人员可以真正地面向Web进行编程，我们面对的不再是拖拉到Web页面的控件，而是整个HTTP请求和响应的流程。
这不是一本ASP.NET MVC入门书籍我个人觉得掌握ASP.NET MVC具有三个层次。
了解基本的编程模式，掌握Controller和View的定义方式，知道路由如何注册，以及验证规则如何定义，此为第一层次。
第二层次要求我们对ASP.NET MVC框架本身从请求接收到响应回复的整个流程具有一个清晰的认识，包括请求如何被路由、目标Controller如何被激活、Model元数据如何被解析、Action方法如何被执行、View如何呈现等。
ASP.NET MVC本身是一个极具可扩展的开发框架，合理利用其扩展性可以解决很多开发中的实际问题，而掌握ASP.NET MVC的最高层次就是凭着对框架本身的运行机制的了解准确地找到相应的扩展点，并创建相应的扩展来解决我们遇到的问题。
本书不是一本ASP.NET MVC入门书籍，而是让处于第一层次的读者快速进入第二和第三层次的书。
这是一本讲述ASP.NET MVC框架本质的书很多.NET开发人员都在抱怨微软开发技术过快的更新频率让他们无所适从。
其实他们看到的只是单纯的版本升级而已，一些本质的东西一直是“稳定”的。
微软推出.NET战略已经十多年了，CLR却只有四个版本而已。
最新版本的ASP.NET虽然表面上已经看不到太多最初的影子，但是整个请求处理的管道一直未曾改变。
对于一项开发技术，只要我们了解了它最根本的一些东西，就不应该惧怕其高频率的版本更替，而应该热烈拥抱它。
本书力求将关于ASP.NET MVC框架最根本的东西带给大家，而不是罗列一些简单的编程技巧。
这是一本实用的书可能有人觉得这本剖析ASP.NET MVC框架运行原理的书没有什么“实际”的意义，因为我们每天的日常工作就是编程，知道了ASP.NET MVC从请求接收到响应回复之间整个处理流程并不会对我们的工作造成实质性的改变。
其实这种想法是极端错误的，因为我们编写的程序最终是在ASP.NET MVC框架上运行的，程序的高效性决定于它是否能够最大限度地“迎合”框架的运行机制，所以了解ASP.NET MVC框架的运行原理有利于我们写出高质量的程序。
我个人将基于ASP.NET MVC的编程分为两类，即“面向业务编程“和“面向框架编程”。
前者根据具体的业务逻辑定义Controller和设计View，这是大部分Web开发人员的主要工作；后者则是为整个Web应用搭建一个框架，让最终的开发人员只需要关注具体的业务逻辑，而让框架来完成所有与业务无关的部分。
对于后者，我们可以充分利用ASP.NET MVC的扩展性，通过自定义的扩展将非业务的功能自动“注入”到业务逻辑的处理流程中，这样不仅可以提高开发效率，而且还能提高开发质量。
本书在剖析ASP.NET MVC框架运行机制过程中几乎列出了其所有的扩展点，并且通过实例演示的形式提供了很多实用的扩展。
可以将本书视为一本“架构设计”的书在我的周围存在这样的一些人，他们以刚毕业一两年的毕业生为主，他们大都工作勤奋、聪明好学，手中经常捧着GoF的《设计模式》，总是希望将书中的设计模式应用到具体项目之中，或者希望通过项目的实践来印证他们在书本上的设计模式，但是理论和实践之间的距离总让他们感到困惑。
要从真实的项目或者产品中学习“实用”的软件架构设计知识，先得确定目标项目或者产品中采用的架构思想和设计模式是正确的，而我们参与的很多项目其实被“架构”得一塌糊涂。

对于像ASP.NET这样的产品，其基础架构能够在很长一段时间内保持不变，本身就证明了应用在上面的架构设计的正确性，它们不正是我们学习架构设计最好的素材吗？
本书对ASP.NET MVC框架的运行机制进行了深入剖析，实际上是将ASP.NET MVC的整个设计展示在读者面前，读者朋友们也许可以将本书作为一本“架构设计”的书来读。
本书的写作特点我想本书的读者可能很多都读过我的《WCF全面解析》，虽然内容不同，本书却可能看成是它的延续，因为它们基本上采用了相同的写作手法。
总地来讲，我基本上采用“原理讲述、代码分析和实例证明”这个模式来介绍某个技术要点，对于一个具体的知识点，我不仅仅会告诉读者“是什么”，还会告诉读者“为什么”，以及“如何证明是这样”。
除此之外，如果某个知识点在真实的项目开发中具有“实用”价值，我一般会给出一个相关的实例演示。
本书具有一个与其他中文原创或者翻译书籍截然不同的特点，那就是几乎所有的术语都采用英文，比如Controller、Model和View在本书中并没有翻译成中文“控制器”、“模型”和“视图”。
因为我认为很多术语其实很难找到一个语义完全等同的中文词组或短语与之对应，对于习惯了英文作为“开发语言”的读者来说，强行翻译其实是不必要的。
这不是一本纯理论的书，而是一本“实证型”的书，在书中提供了110个可供单独下载的实例演示。
这些实例在本书中具有不同的作用，有的是为了探测和证明对应的论点，有的是为了演示某种使用的编程技巧，有的直接就是一个完整的案例。
本书读者我们说《ASP.NET MVC 4框架揭秘》不是一本ASP.NET MVC入门书籍，主要是因为本书在第1章并没有提供一个“Hello World”实例，关注重点主要落在ASP.NET MVC框架本身的运行机制上面，但是并不是说本书的读者需要预先对ASP.NET MVC具有多深入的认识才行。
如果读者对ASP.NET MVC基本的编程模式具有一定的了解，读懂这本书是完全没有问题的。
对于从未接触过ASP.NET MVC的.NET开发人员，可以通过官方网站来学习ASP.NET MVC。
本书结构第1章 ASP.NET + MVCASP.NET和MVC，分别代表了ASP.NET MVC的技术平台和设计思想。
本章对MVC模式及其变体比如MVP和Model 2等作了概括性介绍，同时对ASP.NET的管道式设计，以及与各种版本的IIS之间的交互机制进行了全面讲述。
为了让读者对ASP.NET MVC框架的运行机制具有一个大概的了解，本章按照其原理创建了一个“迷你版”的ASP.NET MVC。
第2章 URL路由ASP.NET MVC借助于URL路由系统实现了URL模式与目标Controller和Action的映射，以及内嵌于URL的参数传递。
基于URL路由的编程主要体现在路由映射的注册和基于注册路由的URL生成上面，本章对这两个方面作了非常详细的介绍。
URL路由最终是借助于自定义的HttpModule（UrlRoutingModule）实现的，它利用动态注册HttpHandler映射的方式提供针对URL路由的实现，这是本章着重讲述的重点。
第3章 Controller的激活本章对以ControllerFactory为核心的Controller激活系统，以及通过DefaultControllerFactory提供的Controller默认激活机制进行了详细介绍。
以IoC的方式激活Controller在实际的Web应用开发中具有重要的意义，本章以较多的篇幅讲述了如何将不同的IoC框架（Unity和Ninject）应用到ASP.NET MVC的Controller激活系统中。
具体来说，我们以实例演示的方式讲述了三种不同的实现方式，包括自定义ControllerFactory、ControllerActivator和DependencyResolver。
第4章 Model元数据的解析Model元数据是针对数据类型的一种描述信息，ASP.NET MVC提供了基于数据注解特性的声明式Model元数据定义方式，本章对所有与此相关的数据注解特性，以及它们对Model元数据的影响进行了全面的介绍。
ASP.NET MVC利用Model元数据实现了模板化的HTML生成方式，本章重点讲述了如何为具体的数据类型定义编辑和显示模板，以及定义的模板在调用HtmlHelper/HtmlHelper的模板方法过程中是如何控制最终生成的HTML的。

本章的最后关注于以ModelMetadataProvider为核心的Model元数据提供机制，以及如何通过自定义ModelMetadataProvider实现对Model元数据提供机制的定制。
第5章 Model的绑定ASP.NET MVC的Model绑定旨在为目标Action方法提供参数列表。
ParameterDescriptor为Model绑定提供了相关的元数据信息，本章以介绍ParameterDescriptor以及相关的ControllerDescriptor和ActionDescriptor作为开篇。
Model绑定所需的最终数据通过ValueProvider来提供，本章接下来会对实现在各种不同ValueProvider中的数据值提供机制，以及以ValueProviderFactory为核心的ValueProvider提供机制进行全面而深入的介绍。
本章的最后部分着重介绍以ModelBinder为核心的Model绑定系统，以及实现在DefaultModelBinder中的默认Model绑定机制。
第6章 Model的验证Action方法在执行之前需要通过Model验证机制确保提供参数的有效性。
本章会着重讲述以ModelValidator为核心的Model验证系统，以及通过ModelValidatorProvider实现的ModelValidator提供机制。
Model验证是伴随着Model绑定进行的，具体执行流程的介绍也包含在本章之中。
ASP.NET MVC利用ValidationAttribute特性为Model验证提供了一种声明式编程方式，其背后的实现机制是本章重要讲述的内容。
jQuery验证框架被默认用于客户端验证，jQuery验证的编程方式，以及与ASP.NET MVC验证系统的协作方式会在本章的最后一部分予以介绍。
第7章 Action的执行针对请求的处理最终体现在对目标Action方法的执行上面。
Action方法可以以同步或者异步的方式执行，所以本章以介绍两种不同的异步Action编程模式作为开篇；此外，同步与异步的差异体现在整个请求的处理过程中，MvcHandler、Controller、ActionInvoker、ControllerDescriptor和ActionDescriptor等都具有同步和异步的版本，本章会对它们作一个系统的比较。
Action的执行还伴随着筛选器的执行，在本章的最后对四种筛选器的作用和执行流程进行单独介绍。
第8章 View的呈现ActionResult作为执行Action返回的结果，实现了对请求的最终响应，本章介绍了所有预定义的ActionResult分别是如何完成针对请求的响应的。
作为最重要的ActionResult，ViewResult将整个预定义的View呈现出来，而它背后是一套完整的View引擎，View引擎的运行机制，以及与ViewResult的协作方式是本章介绍的一个重点。
ASP.NET MVC默认提供了ASPX和Razor这两种原生View引擎的支持，针对Razor引擎的深入剖析被放在本章的最后一部分。
第9章 ASP.NET Web APIASP.NET Web API使我们可以很容易地定义REST服务，本章会提供Web API基本编程模式的介绍。
ASP.NET Web API采用了与ASP.NET MVC独立但类似的执行管道，对整个管道从请求接收到响应回复的整个流程的介绍是本章的重点，包括HttpController的激活与执行、Action的选择、Model元数据的解析、Action参数的绑定与执行等。
第10章 案例实践本章提供了一个名为Video Mall（简称VM）的在线电子商务购物网站来模拟ASP.NET MVC在真实项目中的应用。
VM以SQL Server作为数据存储，并采用Entity Framework作为ORM框架进行数据存取。
VM利用了在前面章节中定义的一系列扩展，同时还涉及了一些架构思想和涉及模式，比如模块化设计、IoC、AOP以及Repository等。
关于作者蒋金楠（网名Artech）现就职于某知名软件公司担任高级软件顾问。
连续5届微软MVP（最有价值专家），同时也是少数的双料MVP（Solutions Architecture + Connected System）之一。
国内较早接触WCF的人之一，2007年2月起在个人博客上发表超过两百篇深入介绍WCF的文章，成为了目前国内WCF在线资料的主要来源。
致谢本书得以出版，需要感谢本书的编辑张春雨先生和葛娜小姐，你们的专业水准和责任心是为本书提供的质量保证，期待着与你们第三度合作的机会。

此外，最需要感谢的是我的老婆徐妍妍，只有我知道你在本书提交给出版社之前所作的校对工作有多么重要。
本书支持本书针对最新版本的ASP.NET MVC，同时涉及太多底层实现的内容，所以大部分内容是找不到任何现成参考资料的，这些内容大都来自于作者对源码的分析和试验的证明。
本书的最初版本是根据ASP.NET MVC 4 Beta撰写的，差不多快写完的时候微软发布了ASP.NET MVC 4 RC，然后我根据RC对原来的内容作了不小的改动。
在ASP.NET MVC4正式推出之后，我第一时间联系到了Scott Guthrie，从他们团队得到了一份正式版与RC之间变化的列表，并据此又作了一些修改。
这些因素加上我本人能力的限制，都可能造成一些无法避免的错误或者偏差，如果读者在阅读过程中发现了任何问题，希望能够反馈给我。
如果读者遇到任何ASP.NET MVC或者是WCF的问题，也欢迎与我通过以下的方式进行交流。

内容概要

针对最新版本的ASP.NET MVC 4，深入剖析底层框架从请求接收到响应回复的整个处理流程（包括URL路由、Controller的激活、Model元数据的解析、Model的绑定、Model的验证、Action的执行、View的呈现和ASP.NET Web API等），并在此基础上指导读者如何通过对ASP.NET MVC框架本身的扩展解决应用开发中的实际问题。

作者简介

蒋金楠，网名Artech，高级软件顾问。
微软6任MVP（Solutions Architecture、Connected System与Microsoft Integration）。
著《WCF全面解析》（上、下册）、《ASP.NET MVC 4框架揭秘》等。
对.NET Framework、C#、ASP.NET、SQL Server、设计模式、软件架构，以及主流的开源框架有着深入的研究。
尤其是在WCF技术方面，属国内较早接触WCF的人之一，同时对.NET Remoting、MSMQ通信技术有深入的理解。
博客园推荐博客（目前排名第一），2012年度51CTO IT博客大赛10佳。

书籍目录

第1章 ASP.NET +
MVC
1.1 传统MVC模式
1.1.1 自治视图
1.1.2 什么是MVC模式
1.2 MVC的变体
1.2.1 MVP
1.2.2 Model
2
1.2.3 ASP.NETMVC与Model 2
1.3 IIS/ASP.NET管道
1.3.1 IIS
5.x与ASP.NET
1.3.2 IIS
6.0与ASP.NET
1.3.3 IIS
7.0与ASP.NET
1.3.4 ASP.NET管道
1.4 ASP.NET
MVC是如何运行的
1.4.1 建立在“迷你版”ASP.NET
MVC上的Web应用
1.4.2 URL路由
1.4.3
Controller的激活
1.4.4 Action的执行
本章小结
第2章 URL路由
2.1 ASP.NET路由系统
2.1.1 请求URL与物理文件的分离
2.1.2
实例演示：通过URL路由实现请求地址与.aspx页面的映射（S201）
2.1.3 Route与RouteTable
2.1.4 路由映射
2.1.5 根据路由规则生成URL
2.2 ASP.NET
MVC扩展
2.2.1 路由映射
2.2.2
实例演示：注册路由映射与查看路由信息（S208）
2.2.3 缺省URL参数
2.2.4 基于Area的路由映射
2.2.5 链接和URL的生成
2.3 动态HttpHandler映射
2.3.1

UrlRoutingModule
2.3.2 PageRouteHandler与MvcRouteHandler
2.3.3
ASP.NET路由系统扩展
本章小结
第3章 Controller的激活
3.1 总体设计
3.1.1 Controller
3.1.2
ControllerFactory
3.1.3
ControllerBuilder
3.1.4 Controller的激活与URL路由
3.2 默认实现
3.2.1
Controller类型的解析
3.2.2
Controller类型的缓存
3.2.3
Controller的释放和会话状态行为的控制
3.3 IoC的应用
3.3.1 从Unity来认识IoC
3.3.2
Controller与Model的分离
3.3.3 基于IoC的ControllerFactory
3.3.4 基于IoC的ControllerActivator
3.3.5
基于IoC的DependencyResolver
本章小结
第4章 Model元数据的解析
4.1
Model元数据及其定制
4.1.1 Model元数据层次化结构
4.1.2 基本Model元数据信息
4.1.3 Model元数据的定制
4.1.4
IMetadataAware接口
4.2
Model元数据与Model模板
4.2.1
实例演示：通过模板将布尔值显示为RadioButton（S409）
4.2.2
预定义模板
4.2.3
DataTypeName与模板名称
4.2.4 模板的获取与执行
4.2.5 实例演示：通过定制Model元数据和自定义模板

实现预定义列表的呈现（S412）
4.3 Model元数据的提供机制
4.3.1
再谈ModelMetadata
4.3.2 ModelMetadataProvider
4.3.3 Model元数据提供系统的扩展
本章小结
第5章 Model的绑定
5.1
ControllerDescriptor、ActionDescriptor与ParameterDescriptor
5.1.1 ControllerDescriptor
5.1.2
ActionDescriptor
5.1.3 ParameterDescriptor
5.2 ValueProvider
5.2.1 NameValueCollectionValueProvider
5.2.2
DictionaryValueProvider
5.2.3 ValueProviderFactory
5.2.4
ValueProviderFactories
5.3 ModelBinder
5.3.1
ModelBinder与ModelBinderProvider
5.3.2
ModelState与Model绑定
5.3.3 ModelBindingContext的创建
5.4
Model绑定的默认实现
5.4.1 简单类型
5.4.2 复杂类型
5.4.3 数组
5.4.4 集合
5.4.5 字典
本章小结
第6章 Model的验证
6.1 ModelValidator与ModelValidatorProvider
6.1.1 ModelValidator
6.1.2
ModelValidatorProvider
6.1.3 ModelValidatorProviders
6.2
Model绑定与验证
6.2.1 ModelState
6.2.2
验证消息的呈现
6.2.3 Model绑定中的验证

6.3
基于数据注解特性的Model验证
6.3.1 ValidationAttribute特性
6.3.2
DataAnnotationsModelValidator
6.3.3 DataAnnotationsModelValidatorProvider
6.3.4
将ValidationAttribute应用到参数上
6.3.5 一种Model类型，多种验证规则
6.4 客户端验证
6.4.1
jQuery验证
6.4.2
基于jQuery的Model验证
6.4.3
自定义验证
本章小结
第7章 Action的执行
7.1
异步Action的定义
7.1.1 基于线程池的请求处理机制
7.1.2 两种异步Action方法的定义
7.1.3 AsyncManager
7.2
Action方法的执行
7.2.1
MvcHandler对请求的处理
7.2.2
Controller的执行
7.2.3 ActionInvoker的执行
7.2.4 ControllerDescriptor的同步与异步
7.2.5 ActionDescriptor的执行
7.3 筛选器的执行
7.3.1 Filter及其提供机制
7.3.2
AuthorizationFilter
7.3.3 ActionFilter
7.3.4 ExceptionFilter
7.3.5 实例演示：集成EntLib实现自动化异常处理（S713, S714,
S715）
7.3.6 ResultFilter
本章小结
第8章 View的呈现
8.1 ActionResult
8.1.1
EmptyResult
8.1.2 ContentResult

8.1.3 FileResult
8.1.4 JavaScriptResult
8.1.5 JsonResult
8.1.6 HttpStatusCodeResult
8.1.7 RedirectResult/RedirectToRouteResult
8.2 ViewResult与ViewEngine
8.2.1 View引擎中的View
8.2.2 ViewEngine
8.2.3 ViewResult的执行
8.3 Razor引擎
8.3.1 View的编译原理
8.3.2 WebViewPage与WebViewPage
8.3.3 RazorView
8.3.4 RazorViewEngine
本章小结
第9章 ASP.NET Web API
9.1 Web、REST与Web API
9.1.1 Web如此简单
9.1.2 REST是什么
9.1.3 ASP.NET Web API
9.2 服务端管道
9.2.1 ASP.NET Web API管道式设计
9.2.2 HttpMessageHandler
9.2.3 HttpServer
9.2.4 实例演示：自定义HttpMessageHandler实现HTTP方法重写（S903）
9.3 HttpControllerDispatcher
9.3.1 HttpController的激活
9.3.2 HttpController的执行
9.3.3 Action的选择
9.3.4 Model元数据的解析
9.3.5 Action参数绑定
9.3.6 Model验证
9.3.7 Action的执行与结果的响应
9.4 Web API的调用和自我寄宿
9.4.1 HttpClient
9.4.2 HttpSelfHostServer

第一图书网, tushu007.com

章节摘录

版权页： 插图： ASP.NET集成 从上面对IIS 5.x和IIS 6.0的介绍中，我们不难发现IIS与ASENET是两个相互独立的管道（Pipeline）。
在各自管辖范围内，它们各自具有自己的一套机制对HTTP请求进行处理。
两个管道通过ISAPI实现“连通”，IIS是第一道屏障，当对HTTP请求进行必要的前期处理（比如身份验证等）后，通过ISAPI将请求分发给ASENET管道。
当ASENET在自身管道范围内完成对HTTP请求的处理时，处理后的结果再返回到IIS，IIS对其进行后期处理（比如日志记录、压缩等），最终生成HTTP响应。
图1—9反映了IIS 6.0与ASENET之间的桥接关系。
从另一个角度讲，IIS运行在非托管的环境中，而ASENET管道则是托管的，ISAPI还是连接非托管环境和托管环境的纽带。
IIS 5.x和IIS 6.0把两个管道进行隔离至少带来了下面的一些局限与不足。
相同操作的重复执行：IIS与ASP.NET之间具有一些重复的操作，比如身份验证。
动态文件与静态文件处理的不一致：因为只有基于ASP.NET动态文件（比如.aspx、.asmx、.svc等）的HTTP请求才能通过ASENET ISAPI进入ASP.NET管道，而对于一些静态文件（比如.html、.xml、.img等）的请求则由IIS直接响应，那么ASP.ENET管道中的一些功能将不能用于这些基于静态文件的请求，比如我们希望通过Foms认证应用于基于图片文件的请求就做不到。
IIS难以扩展：对于IIS的扩展基本上就体现在自定义ISAPI，但是对于大部分人来说，这不是一件容易的事情。
因为ISAPI是基于Win32的非托管的API，并非一种面向应用的编程接口。
通常我们希望的是诸如定义ASP.NET的HttpModule和HttpHandler一样，通过托管代码的方式来扩展IIS。

编辑推荐

《ASP.NET MVC4框架揭秘》是让处于ASP.NETMVC第一层次的读者快速进入第二和第三层次的书。ASP.NET MVC功能强大，它提供了一种全新的编程方式，使我们可以将MVC模式很自然地融入动态网站的开发之中。
由它带来的对关注点清晰分离和对HTML的细粒度控制使我们真正体会到了敏捷开发的乐趣。
借助于ASP.NET MVc提供的众多特性，不但可以使我们在复杂应用中灵活地采用TDD（测试驱动开发）的梦想变成现实，还能让我们的应用尽可能地拥抱最新的Web标准。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com