Broadview®
www.broadview.com.cn

ASP.NET MVC 4 框架揭秘

◎蒋金楠 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

ASP.NET MVC 4
框架揭秘

◎蒋金楠 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

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。

图书在版编目（CIP）数据

ASP.NET MVC 4 框架揭秘 / 蒋金楠著. —北京：电子工业出版社，2013.1
ISBN 978-7-121-19049-0

Ⅰ. ①A… Ⅱ. ①蒋… Ⅲ. ①网页制作工具－程序设计 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字（2012）第 281603 号

策划编辑：张春雨
责任编辑：葛 娜
印 刷：涿州市京南印刷厂
装 订：涿州市京南印刷厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1092 1/16 印张：37 字数：855 千字
印 次：2013 年 4 月第 3 次印刷
印 数：3000 册 定价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

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（http://www.asp.net/mvc）来学习 ASP.NET MVC。

本书结构

第 1 章 ASP.NET + MVC

ASP.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<TModel>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 API

ASP.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设计模式，比如模块化设计、IoC、AOP 以及 Repository 等。

关于作者

蒋金楠（网名 Artech），《WCF 全面解析（上、下册）》作者，现就职于一知名软件公司担任高级软件顾问。2007—2012 年被连续 6 次评为微软 MVP（Solution Architecture、Connected System 和 Microsoft Integration）。拥有一个访问量超过 200 万的个人博客（http://www.cnblogs.com/artech），长年排名博客园推荐榜榜首。2012 年度 51CTO IT 博客大赛 10 佳得主。

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 MVC 4 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。

- 作者博客：http://www.cnblogs.com/artech
- 作者微博：http://www.weibo.com/artech
- 电子邮箱：jiangjinnan@gmail.com

本书每一章节都会提供一系列实例演示，读者可以根据编号（比如 S101、S202 等）从下载的源代码压缩包中找到对应的实例。本书的附录给出了所有源代码可供下载的所有的实例演示的列表和相关描述。

- 源代码下载地址：http://files.cnblogs.com/artech/asp.net.mvc.4.samples.rar

目 录

第 1 章　ASP.NET + MVC

ASP.NET MVC 是一个全新的 Web 应用框架。将术语 ASP.NET MVC 拆分开来，即 ASP.NET+MVC，前者代表支撑该应用框架的技术平台，意味着 ASP.NET MVC 和传统的 Web Forms 应用框架一样都是建立在 ASP.NET 平台之上；后者则表示该框架背后的设计思想，意味着 ASP.NET MVC 采用了 MVC 架构模式。

1.1 传统 MVC 模式

对于大部分面向最终用户的应用来说，它们都需要具有一个可视化的 UI 界面与用户进行交互，我们将这个 UI 称为视图（View）。在早期，我们倾向于将所有与 UI 相关的操作糅合在一起，这些操作包括 UI 界面的呈现、用于交互操作的捕捉与响应、业务流程的执行以及对数据的存取，我们将这种设计模式称为自治视图（Autonomous View，AV）。

1.1.1 自治视图

说到自治视图，很多人会感到陌生，但是我们（尤其是.NET 开发人员）可能经常在采用这种模式来设计我们的应用。Windows Forms 和 ASP.NET Web Forms 虽然分别属于 GUI 和 Web 开发框架，但是它们都采用了事件驱动的开发方式，所有与 UI 相关的逻辑都可以定义在针对视图（Windows Form 或者 Web Form）的后台代码（Code Behind）中，并最终注册到视图本身或者视图元素（控件）的相应事件上。

一个典型的人机交互应用具有三个主要的关注点，即数据在可视化界面上的呈现、UI 处理逻辑（用于处理用户交互式操作的逻辑）和业务逻辑。自治视图模式将三者混合在一起，势必会带来如下一些问题：

- 业务逻辑是与 UI 无关的，应该最大限度地被重用，由于业务逻辑定义在自治视图中，相当于完全与视图本身绑定在一起。如果我们能够将 UI 的行为抽象出来，基于抽象化 UI 的处理逻辑也是可以被共享的，但是定义在自治视图中的 UI 处理逻辑完全丧失了重用的可能。
- 业务逻辑具有最强的稳定性，UI 处理逻辑次之，而可视化界面上的呈现最差（比如我们经常会为了更好地呈现效果来调整 HTML）。如果将具有不同稳定性的元素融为一体，那么具有最差稳定性的元素决定了整体的稳定性，这是“短板理论”在软件设计中的体现。
- 任何涉及 UI 的组件都不易测试。UI 是呈现给人看的，并且用于与人进行交互，用机器来模拟活生生的人来对组件实施自动化测试不是一件容易的事，自治视图严重损害了组件的可测试性。

为了解决自治视图导致的这些问题，我们需要采用关注点分离（Seperation of Concerns，SoC）的方针将可视化界面呈现、UI 处理逻辑和业务逻辑三者分离出来，并且采用合理的交互方式将它们之间的依赖降到最低。将三者“分而治之”，自然也使 UI 逻辑和业务逻辑变得更容易测试，测试驱动设计与开发变成了可能。这里用于进行关注点分离的模式就是 MVC。