.NET开发经典名著

Microsoft

ASP.NET MVC 5 编程实战(第3版)

——构建在桌面和移动设备运行同样精彩的Web应用

Professional

[美] Dino Esposito 著
潘丽臣 译

清华大学出版社

.NET 开发经典名著

ASP.NET MVC 5 编程实战

(第 3 版)

——构建在桌面和移动设备运行同样精彩的 Web 应用

[美] Dino Esposito 著

潘丽臣 译

清华大学出版社

北 京

北京市版权局著作权合同登记号 图字：01-2015-1186

图书在版编目(CIP)数据

ASP.NET MVC 5 编程实战(第 3 版)——构建在桌面和移动设备运行同样精彩的 Web 应用/(美)埃斯波西托(Esposito, D.) 著；潘丽臣 译. —北京：清华大学出版社，2015
(.NET 开发经典名著)
书名原文：Programming Microsoft ASP.NET MVC, Third Edition
ISBN 978-7-302-39480-8

Ⅰ. ①A… Ⅱ. ①埃… ②潘… Ⅲ. ①网页制作工具 ②网页设计 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字(2015)第 036197 号

责任编辑：王　军　于　平
封面设计：孔祥峰
责任校对：成凤进
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×230mm　　印　　张：30　　字　　数：653 千字
版　　次：2015 年 3 月第 1 版　　印　　次：2015 年 3 月第1 次印刷
印　　数：1～4000
定　　价：59.80 元

产品编号：060534-01

译 者 序

凡是用过 ASP.NET MVC 的开发人员必定获益匪浅。ASP.NET MVC 具有耦合性低、重用性高、生命周期成本低、部署快、可维护性高、有利于软件工程化管理方面的优点。该架构模式对于大中型应用程序的架构和开发来说，优势尤为明显。

一直以来，国内大多数ASP.NET开发人员都限于用控件拼装Web表单进行Web应用程序的开发，这使得不少开发人员更关注用控件实现呈现和功能交互，而往往忽视了在应用程序总体架构和软件层次体系方面的考虑，也相应造成了Web应用程序良莠不齐的局面。随着ASP.NET MVC的逐步推广应用，相信这种情况会有所改善。

本书对 ASP.NET MVC 进行了全面阐述，读者通读本书后，会对 ASP.NET MVC 有更为详尽的了解。知易行难，希望有志于从事 Web 应用程序开发的读者能够从本书中获得启发并尝试用于实践，将书本内容转化成自己的知识。在此要特别强调的是，本书第III部分重点介绍了移动端开发方面的内容，也算是迎合了当前的时代潮流。管中窥豹，我们由此可以预见 ASP.NET 整体框架将愈发加大对移动端开发的支持，而 ASP.NET MVC 已经走在了前面，所以 ASP.NET 开发人员在转向移动端开发的同时也必然愈发感受到微软强大的支持后盾。

在此要特别感谢清华大学出版社的编辑们，他们在本书翻译过程中为译者提供的巨大帮助，没有其热情付出，本书将难以顺利付梓。本书全部章节由潘丽臣翻译，参与本次翻译活动的还有蒲成、杨晔、杨达辉、申成龙、杨帆、赵栋、王滨、李鹏、负书谦、林超、陈世佳。在此一并表示感谢。

译者具有多年的 ASP.NET 开发经验，最近几年也一直在实际开发中应用 ASP.NET MVC 的先进理念和技术，深感 ASP.NET MVC 作为一种架构模式能为开发人员带来极大的便捷，其必将成为新的主流开发指导思想。所以，开发人员都应该尽早掌握 ASP.NET MVC，对于 ASP.NET 开发人员来说尤其如此。由于译者水平有限，难免会出现一些错误或翻译不准确的地方，如果有读者能够指出并勘正，译者将不胜感激。

译 者

作者简介

Dino Esposito 是 e-tennis.net 网站的 CTO 和创始人之一，这是一个新创办的为专业网球和运动公司提供软件和 IT 服务的网站。Dino 仍然在进行大量的培训工作和写作，并著有多本 Web 开发和.NET 设计方面的书籍。他最新的两本著作是由微软出版社出版的 *Architecting Mobile Solutions for the Enterprise* 和 *Microsoft .NET: Architecting Applications for the Enterprise*。Dino 经常在行业会议(包括 DevConnections)以及像 Software Architect、DevWeek 和 BASTA 这样的欧洲著名活动中演讲。Dino 不仅是 JetBrains 公司在 Android 和 Kotlin 开发方面的技术专家，还是提供 WURFL 的 ScientiaMobile 数据库开发团队的成员。该数据库存储了大量移动设备性能，被多家大型组织使用，比如 Facebook。

你可以在 Twitter 上通过@despos 及其博客(http://software2cents.wordpress.com)来关注 Dino。

前　言

首先要掌握事实，然后你可以随意歪曲它们。

——*Mark Twain*

ASP.NET 诞生于 20 世纪 90 年代末期各行各业正迅速探索互联网的时代。ASP.NET 的主要目的是为了让开发人员能够快速有效地构建应用程序，而无须处理如 HTTP、 HTML 和 JavaScript 等错综复杂的底层细节。这正是当时的社会环境所强烈要求的。ASP.NET 是微软推出来满足这项需求的，且大大超过了预期的程度。

十多年后的今天，ASP.NET 的发展显得有些滞后，很多人甚至开始质疑 Web 框架存在的必要性。这是一个了不起的时代，为我们提供了若干选项。其中就有 Web Forms 和 ASP.NET MVC 应用程序，还有更多 JavaScript 密集型客户端应用程序(单页面应用程序)，它们使用一个服务器端后台来为实际公开的一些页面提供基本布局和特设服务，比如捆绑。

奇妙的是，使用 Web Forms 模式，你仍可以编写功能性应用程序，尽管 ASP.NET MVC 能够更密切地服务于开发人员的当前需求。Web Forms 的最常见应用场景是，你要开发专注于呈现数据并使用优质第三方控件套装的应用程序。ASP.NET MVC 可用于处理其他所有方面，包括客户端单页面应用程序的框架搭建。

Web 应用程序的改变方式证明了，ASP.NET MVC 可能未能替代 ASP.NET Web Forms 在众多开发人员心目中的地位，但这却是正确的选择，ASP.NET MVC 足以成为任何一个需要实体后台的应用程序的理想 Web 平台，对于那些以多设备实用功能为目标的 Web 应用程序来说尤其如此。是的，这很可能意味着不到两年时间内的所有 Web 应用程序。

转换到 ASP.NET MVC，对于 ASP.NET 开发人员来说是相当自然的过程。

本书读者对象

这几年来，不少人读过我的一些书籍和文章。这些读者已经察觉到了，我并不擅长写作步骤详解类的参考型书籍，同样，我也不能对同一门课程在前后两次的教学中以相同的顺序介绍主题，或提供前后相同的例子。

此书并不适合绝对的初学者；但除此之外的其他人我觉得都可以阅读，包括那些对 ASP.NET MVC 还不甚了解的人。能力和专业水平越高的人，越难在本书中找到相关专业领域的附加值。然而，这本书得益于几年的现实实践，我相信其中一定有很多可能还会吸引专家的解决方案，尤其是涉及移动设备方面的。

如果你使用 ASP.NET MVC，我相信你一定会在此书中找到一些有价值的东西。

假定

这本书假定你对 ASP.NET 开发有基本的了解。

不适合阅读本书的人群

如果你需要的是 ASP.NET MVC 分步指南，那么本书算不上是一本理想的书籍。

本书结构

该书分为三个部分。第 I 部分："ASP.NET MVC 基础"，提供了对 ASP.NET 基础和其核心组件的简短概述。第 II 部分："ASP.NET MVC 软件设计"，着重介绍 Web 应用程序、特定设计模式和最佳实践的常见问题。最后，第III部分："移动客户端"，是有关 JavaScript 和移动界面的。

系统要求

需要安装以下软件以运行本书中所提供的示例：

- 以下的操作系统之一：Windows 8/8.1、Windows 7、Windows Vista with Service Pack 2 (除了简化版)、Windows XP with Service Pack 3(除了简化版)、Windows Server 2008 with Service Pack 2、Windows Server 2003 with Service Pack 2 以及 Windows Server 2003 R2。
- Microsoft Visual Studio 2013 的任意版本(如果你使用 Express Edition 产品，则可能需要多个下载)。

- Microsoft SQL Server 2012 Express Edition 或更高版本，以及 SQL Server Management Studio 2012 Express 或更高版本(与 Visual Studio 一起分发；Express Edition 需要单独下载)。

根据你的 Windows 配置，可能需要本地管理员权限才能安装或配置 Visual Studio 2013 和 SQL Server 2012 产品。

示例代码下载

该书的大多数章节都包含一些练习，你可以用交互的方式尝试正文中所学到的新材料。可以从以下网页下载所有的示例项目，包括它们实践前和实践后的格式：

```
http://aka.ms/programASP-NET_MVC/files
http://www.tupwk.com.cn/downpage
```

请按照说明下载 asp-net-mvc-examples.zip 文件。

安装代码示例

通过执行下列步骤，在你的计算机中安装代码示例，以便在你做本书中的练习时可以使用。

(1) 将对本书的网站上下载的 asp-net-mvc-examples.zip 文件进行解压(如有必要，指定一个特定的目录和路径来创建它)。

(2) 如果弹出提示，请查看所显示的最终用户许可协议。如果你接受这些条款，请选择 Accept 选项，然后单击 Next。

注意：

如果没有显示许可协议，你可以从下载 asp-net-mvc-examples.zip 文件的网页访问它。

使用示例代码

Setup.exe 程序所创建的文件夹包含每一章的一个子文件夹。反之，每一章可能包含额外的子文件夹。所有示例都被组织在一个单独的 Visual Studio 2013 解决方案中。你要打开 Visual Studio 2013 中的解决方案文件并导航到这些示例。

勘误及相关支持

我们尽一切努力确保此书及其同步内容的准确性。本书自出版以来所报告的一切错误都列在微软出版社网站上：

```
http://aka.ms/programASP-NET_MVC/errata
```

如果你发现了未列出的错误，可以通过相同的页面发送报告给我们。

如果你需要额外的支持，请发送电子邮件到 mspinput@microsoft.com 给微软出版社的支持部。

请注意，上述地址不提供微软软件的产品支持。

我们期待你的反馈

在微软出版社，你的满意才是我们的首要任务，你的反馈是我们最宝贵的财富。请告诉我们你对此书的看法：

```
http://aka.ms/tellpress
```

这项调查是短暂的，但我们认真阅读你的每一条意见和想法。提前感谢你的输入！

保持联系

让我们将交流继续下去！这里是我们的 Twitter 网址：http://twitter.com/MicrosoftPress。

目 录

第 I 部分

ASP.NET MVC 基础

第 I 部分

ASP.NET MVC 基础

第 1 章
ASP.NET MVC 控制器

人们总说时间会改变一切，但实际上你必须自己动手去改变一切。

——*Andy Warhol*

我认为从 Ajax 征服大众的那一天开始，ASP.NET Web Forms 就开始变得不堪胜任了。正如有些人所说，Ajax 已经成为一只射中 ASP.NET 之踵的另一支 Achilles 毒箭了。Ajax 使得对 HTML 和客户端代码拥有越来越多的控制权这一情形成为了事实存在。随着时间的推移，这引发了不同架构的形成，使 ASP.NET Web Forms 逐渐丧失了对任务的胜任能力。

由于可以适用于现有的 ASP.NET 运行时，MVC 模式产生了一个新的框架——ASP.NET MVC——它能让 Web 开发与开发人员的当前需求相匹配。

在 ASP.NET MVC 中，每个请求的结果最终都会执行某个操作——根本上来说也就是特定类上的方法。操作执行的结果会与一个视图模板一起传递给视图子系统。结果和模板随后会用于生成浏览器的最终响应。用户不需要将浏览器指向某个页面，他们只需要放置一个请求即可。难道这还不是很大的变化吗？

与 Web Forms 不同，ASP.NET MVC 是由连接在一起的各种代码层所组成，而不是交织在一起形成一个单一的整体块。鉴于此，可以很容易地以自定义组件替换任何层，用以提高解决方案的可维护性和可测试性。使用 ASP.NET MVC，可以获得对标记的完全控制，并能随意用你最喜欢的 JavaScript 框架来套用样式和注入脚本代码。

基于与 Web Forms 相同的运行时环境，ASP.NET MVC 推出了一个适合于 Web 应用程序的经典模型-视图-控制器(Model-View-Controller)模式，使 Web 应用程序的开发有了明显不同的体验。在这一章中，你将会了解控制器的结构与作用——ASP.NET MVC 应用程序的基础——以及请求被路由到控制器的方式。

尽管你有可能决定继续使用 Web Forms 来进行当前的 Web 开发，但 ASP.NET MVC 的确是更好的选择。虽然不需要投入大量的时间，但你需要确切地知道发生了什么以及 MVC 背后的原理。如果照此做，那么你的任何投入都将会比预期更早地得到回报。

注意：

本书基于 ASP.NET MVC 5。此版本的 ASP.NET MVC 兼容之前的版本。这意味着可以在同一台计算机上同时安装两个版本，在使用最新版本时不会影响你已经拥有的现成的 MVC 代码。

1.1 对输入请求进行路由

最初，整个 ASP.NET 平台都是围绕着服务物理页面请求的理念来开发的。事实证明大部分在 ASP.NET 应用程序中所使用的 URL 都由两部分组成：分别是包含了逻辑的物理网页路径，和填充在查询字符串中用于提供参数的一些数据。这种方法已经使用数年，现今仍然有效。然而，ASP.NET 运行时环境并不会限制你仅仅调用由特定位置和文件所确定的资源。通过编写一个专门的 HTTP 处理程序并将其绑定到一个 URL，就可以使用 ASP.NET 来执行代码以响应请求，而无须依赖物理文件。这只是 ASP.NET MVC 与 ASP.NET Web Forms 之间的一个主要区别。让我们大致了解一下如何用 HTTP 处理程序来模拟 ASP.NET MVC 行为。

注意：

在软件中，术语 URI(统一资源标识符)是指通过一个位置或一个名称来引用资源。当 URI 通过位置识别资源时，它被称作 URL 或统一资源定位符。当 URI 通过名称识别资源时，它就成为一个 URN 或统一资源名称。在这方面，ASP.NET MVC 旨在处理更通用的 URI，而 ASP.NET Web Forms 主要处理位置感知的物理资源。

1.1.1 模拟 ASP.NET MVC 运行时

让我们构建一个简单的 ASP.NET Web Forms 应用程序，并使用 HTTP 处理程序来理解 ASP.NET MVC 应用程序的内部机制。可以先从微软 Visual Studio 项目管理器处获取基本的 ASP.NET Web Forms 应用程序。

1. 定义可识别的 URL 的语法

由于请求的 URL 不一定与 Web 服务器上的物理文件相匹配，因此第一步即是列出哪些 URL 对应用程序有意义。为了避免过于特殊，我们假设你只支持几个固定的 URL，每一个都会映射到一个 HTTP 处理程序组件。下面的代码片段显示了需要对默认 web.config 文件所做的更改：

```
<httpHandlers>
<add verb="*"
   path="home/test/*"
```

```
    type="MvcEmule.Components.MvcEmuleHandler" />
</httpHandlers>
```

每当应用程序收到与指定 URL 匹配的请求时，它就会将其传递到指定的处理程序。

2. 定义 HTTP 处理程序的行为

在 ASP.NET 中，HTTP 处理程序是一个实现了 IHttpHandler 接口的组件。该接口比较简单，包括以下两个部分：

```
public class MvcEmuleHandler : IHttpHandler
{
   public void ProcessRequest(HttpContext context)
   {
      // Logic goes here
      ...
   }

   public Boolean IsReusable
   {
      get { return false; }
   }
}
```

大部分情况下，HTTP 处理程序都有一个只受某些通过查询字符串传递的输入数据所影响的硬编码行为。不过，我们也可以毫不费力地将处理程序用作抽象工厂来再增加一级间接层。事实上，处理程序可以使用来自请求的信息以确定要调用的外部组件以切实地处理请求。这样一来，单个 HTTP 处理程序就可以服务各种请求，并且只需要在一些专门的组件之间分配调用。

HTTP 处理程序可以解析出令牌中的 URL，并使用该信息来标识要调用的类和方法。这里是一个显示其工作原理的示例：

```
public void ProcessRequest(HttpContext context)
{
   // Parse out the URL and extract controller, action, and parameter
   var segments = context.Request.Url.Segments;
   var controller = segments[1].TrimEnd('/');
   var action = segments[2].TrimEnd('/');
   var param1 = segments[3].TrimEnd('/');

   // Complete controller class name with suffix and (default) namespace
   var fullName = String.Format("{0}.{1}Controller",
                  this.GetType().Namespace, controller);
```