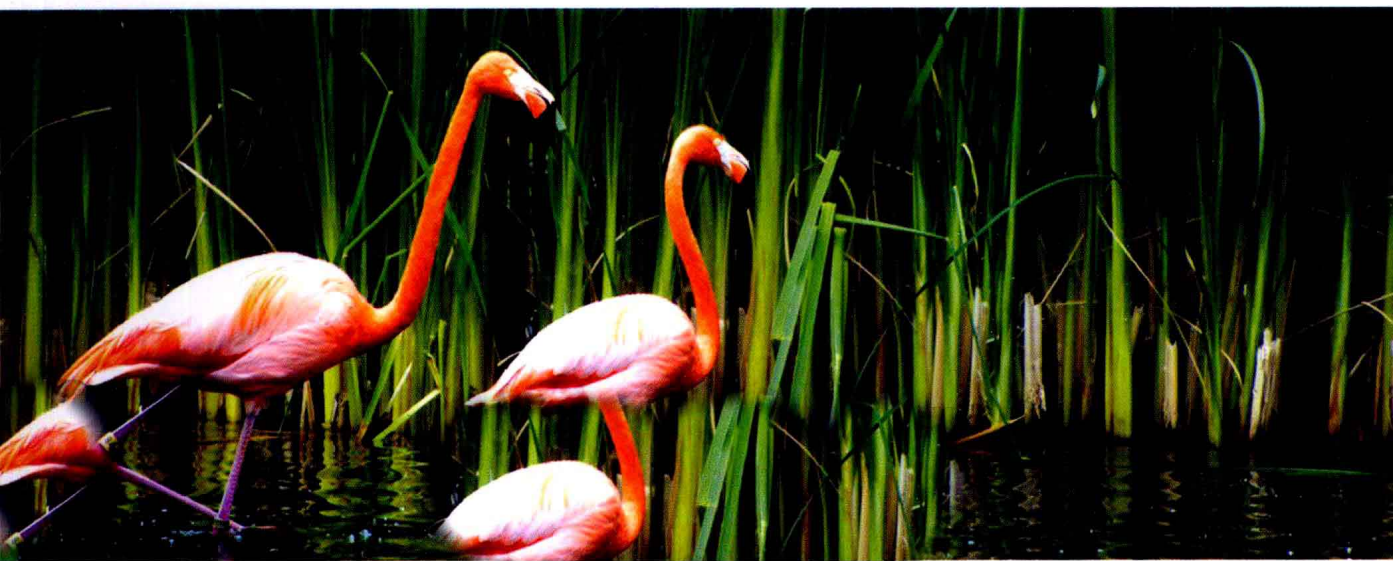


Essential ASP.NET

ASP.NET

本质论



郝冠军◎著



机械工业出版社
China Machine Press

揭秘系列丛书
UNLEASH

Essential ASP.NET

ASP.NET

本质论



郝冠军◎著



机械工业出版社
China Machine Press

如果你只是想系统地学习如何简单地利用 ASP.NET 快速地进行开发,这本书也许不是你想要的;如果你不满足于只是会利用 ASP.NET 强大的控件功能完成一些常规应用的开发,而是想深入探究 ASP.NET 的本质和精髓,实现从一个控件使用人员向系统开发人员的过渡,那么这本书是你不能错过的,也是你目前的唯一选择。

本书以 ASP.NET 应用中的请求处理过程为主线,对每一步处理所涉及的技术和原理进行了深入的剖析,同时对开发过程中在各处理环节可能会遇到的经典疑难问题进行了分析并给出了解决方案。

第 1~4 章是 ASP.NET 的核心部分,细致地剖析了 ASP.NET 中的请求处理机制、ASP.NET 中的对象与 HTTP 之间的映射关系、应用程序处理管道的处理过程、处理程序的处理机制,以及多线程技术在 ASP.NET 中的应用。第 5~8 章是经典的 WebForm 部分,重点讲解了控件的原理与页面的生成机制,包括流与控件的关系、控件与页面的关系、数据绑定控件与模板的关系,以及 ASP.NET 中的各种状态管理技术。第 9 章分析了 ASP.NET MVC 的处理过程,以及 ASP.NET MVC 应用与经典的 ASP.NET 应用之间的关系。第 10 章讨论了 ASP.NET 与 IIS 服务器之间的关系,并分别针对不同版本的 IIS 分析了其处理过程。第 11 章对 ASP.NET 应用中的用户问题进行了分析,并就各种常见问题给出了解决方案。

封底无防伪标均为盗版

版权所有,侵权必究

本书法律顾问 北京市展达律师事务所

图书在版编目(CIP)数据

ASP.NET 本质论 / 郝冠军著. —北京:机械工业出版社, 2011.3

ISBN 978-7-111-33285-5

I. A… II. 郝… III. 主页制作—程序设计 IV. TP393.092

中国版本图书馆 CIP 数据核字 (2011) 第 017337 号

机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑:陈佳媛

北京市荣盛彩色印刷有限公司印刷

2011 年 3 月第 1 版第 1 次印刷

186mm×240mm·29.5 印张

标准书号:ISBN 978-7-111-33285-5

定价:69.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

客服热线:(010) 88378991; 88361066

购书热线:(010) 68326294; 88379649; 68995259

投稿热线:(010) 88379604

读者信箱:hzjsj@hzbook.com



本书的起源

经常有人问起：应该如何学习 ASP.NET 开发？为什么开始的时候感觉很容易，但是，遇到问题的时候却感到无从下手？太多的人开始学习的时候，对 ASP.NET 有着深深的误解，包括我自己。

很多人选择 ASP.NET 的理由是因为它简单：中文开发环境、简体中文的文档、简单的拖放式开发、类似于 WinForm 的开发体验等。Visual Studio 和 .NET Framework 为我们提供了一个极其方便的开发环境，很多人因此进入了 ASP.NET 开发之门，甚至有相当多的 ASP.NET 程序员都没有了解过 HTTP 协议的内容，或者 HTML 的语法，也同样在完成着开发任务。

这究竟是 ASP.NET 技术的成功，还是失败？

在 ASP.NET 中，组件技术被用到极致，整个 ASP.NET 就是一个高级组件，内部又可以分为控件组件、状态管理组件、用户管理组件等。组件技术的使用将 ASP.NET 程序员分为两大类：开发组件的程序员和使用组件的程序员。

开发组件的程序员需要掌握 ASP.NET 的运行机制，熟悉 ASP.NET 处理 HTTP 请求的整个过程，对通过 .NET 技术来解决 Web 请求处理的方法有着深刻的理解，这一切对程序员有着很高的要求。而使用组件的程序员只需要使用开发完成的组件，将组件通过工具箱拖放到窗体上，然后，通过属性窗口设置组件的属性，再加上一些机械的处理，就可以快速开发出应用程序。甚至完全不需要知道组件的背后在发生着什么。在许多 ASP.NET 技术演示中，好像一次魔术表演，就神奇地完成了开发任务。开发中的这种分工提高了开发的效率。

那么，我们希望成为哪一种程序员呢？

成为使用组件的程序员比较容易，成为开发组件的程序员很难。高内聚、低耦合的组件也造成了很陡峭的技术壁垒，需要辛苦地攀登。那么，在这个组件开发的时代，我们还需要学习组件的知识吗？答案是：需要！一定需要！即使我们不开发组件，也不能不理解组件！

组件的使用千变万化，但是万变不离其宗。如果你希望成为自由驰骋在 ASP.NET 领域的骑手，那就跟随我进行一次穿越 ASP.NET 的开心之旅吧！

面向的读者

本书面向的读者是准备深入学习 ASP.NET 的学生和有 1 ~ 2 年 ASP.NET 工作经验，但是希望进一步提高开发技能，深入掌握 ASP.NET 高级编程的程序员。通过系统地学习 ASP.NET 的处理机制，为成为一个高级的 ASP.NET 软件开发人员打下坚实的基础。

特色

本书不从 C# 讲起，也不讲解控件的属性及使用，没有设计模式的分析，甚至没有任何数据库的内容。所以，不要希望通过本书来学习一个 ASP.NET 的购物网站如何完成。

在本书中，有 ASP.NET 与 HTTP 关系的详细分析，有事件处理机制在 ASP.NET 中的应用，有多线程程序在 ASP.NET 中的应用与分析，有控件与流的关系，有控件与 HTML 之间关系的详细分析，有各种状态管理机制实现的内幕。总起来说，这里只有 ASP.NET 的内在运行机制的分析。本书对 ASP.NET 的组件机制进行了详细的分析，希望能够帮助你理解 ASP.NET 应用程序为什么这样写的问题。从前，你可能会使用各种控件，可能还掌握各种开发技巧。通过本书你可以创建自己的组件，发现未知的技巧！看了此书之后，希望你说：原来如此！

如何阅读本书

本书从 HTTP 请求开始，将会带领你穿越整个 ASP.NET 的处理过程，以请求的处理过程为主线，对每一步处理所涉及的技术进行深入的剖析，结合开发中常见的问题，分析问题产生的原因并给出解决方案。包括最新的 MVC 技术。书中的每一章也独立成篇，你可以根据自己的需要来选择阅读。

从第 1 章~第 4 章是 ASP.NET 的核心部分，重点讨论了 ASP.NET 中对于请求的处理机制，ASP.NET 中对象与 HTTP 之间的映射关系，应用程序处理管道的处理过程，处理程序的处理机制，以及多线程技术在 ASP.NET 中的应用。

从第 5 章~第 8 章是经典的 WebForm 部分，重点讨论了控件的原理与页面的生成机制。详细讨论了流与控件的关系，控件与 HTML 的关系，数据绑定控件与模板的关系，以及 ASP.NET 中各种状态管理技术。

第 9 章分析了开发 ASP.NET 程序的另外一种选择 ASP.NET MVC 的处理过程，分析了 ASP.NET MVC 应用程序与经典的 ASP.NET 程序之间的关系，以及 MVC 内部的处理过程。并对关键的处理过程进行了详细的讨论。

ASP.NET 应用程序不是一个独立的应用程序，需要寄宿于 Web 服务器之上。第 10 章讨

论了 ASP.NET 与 IIS 服务器之间的关系，并针对不同的 IIS 版本分别分析了其处理过程。

最后一章针对 ASP.NET 应用程序中的用户问题进行了分析，剖析了在 ASP.NET 中处理用户问题的各种方案，对处理不同环节中的用户及其转换关系进行了详细的分析。

致谢

首先要感谢的是华章公司的杨福川编辑，没有他的鼓励和帮助，就不会有这本书的诞生。

特别要感谢博客园的帮助和支持，作为专业的 .NET 技术网站，博客园这个灿若繁星的技术宝库为我提供了成长的土壤，本书中的许多问题和案例来源于博客园。

感谢众多的朋友们在我完成本书的过程中给予我的支持和鼓励，你们的帮助是我完成本书的强大动力。

本书的支持

很荣幸出版社给了我这样一个机会与大家分享这些知识。但是由于作者本人水平所限，虽然进行了多次修改，仍然很难尽如人意，已近完稿，心中更加忐忑不安，只是希望能够给大家在学习 ASP.NET 的过程中带来一些帮助。如果大家在读完某些章节之后，对以前希望了解又无从查询的问题感觉豁然开朗，我就聊以自慰了。

书中的示例都有相应的源码，可以在我的博客下载，也可以在华章公司的官方网站（www.hzbook.com）上下载。

联系作者

电子邮件：haoguanjun@gmail.com

博客：<http://haogj.cnblogs.com>

对本书的任何问题请通过电子邮件联系，我会尽快处理和答复，并在博客上发布与更新。

作者

2011 年 1 月

目 录

前 言

第 1 章 网站应用程序 /1

- 1.1 Web 应用程序的简单回顾 /2
 - 1.1.1 资源的地址——通用资源标识符 /2
 - 1.1.2 找到主机 /3
 - 1.1.3 HTTP 协议 /4
- 1.2 最简单的 Web 服务器 /6
 - 1.2.1 网络插座 Socket /6
 - 1.2.2 基于 TcpListener 的 Web 服务器 /10
 - 1.2.3 基于 HttpListener 的 Web 服务器 /11
- 1.3 进入 ASP.NET /13
 - 1.3.1 Web 应用程序域 /13
 - 1.3.2 不使用 GAC 和 bin 加载 Web 应用程序域 /14
 - 1.3.3 默默无闻的工作者对象 /16
 - 1.3.4 Web 应用程序的运行时 /18
- 1.4 对象化的 HTTP /19
 - 1.4.1 请求参数的对象类型 HttpRequest /19
 - 1.4.2 处理回应的对象类型 HttpResponse /21
 - 1.4.3 辅助的常用工具类 HttpServerUtility /23
 - 1.4.4 编码与解码 /24
 - 1.4.5 浏览器类型 /26

- 1.5 创建自定义的 ASP.NET 服务器 /28
 - 1.5.1 ASP.NET Web 服务器 /28
 - 1.5.2 监听程序 /28
 - 1.5.3 部署程序集 /29
 - 1.5.4 各种各样的 Cassini /30
- 1.6 本章小结 /30

第 2 章 应用程序对象 /31

- 2.1 请求的处理参数——上下文对象 HttpContext /32
 - 2.1.1 常用成员 /32
 - 2.1.2 底层方法 /33
- 2.2 应用程序对象 HttpApplication /33
 - 2.2.1 处理管道 /34
 - 2.2.2 HttpApplication 的处理管道 /38
 - 2.2.3 处理过程的简单介绍 /38
 - 2.2.4 HttpContext 状态管理 /40
- 2.3 处理 HttpApplication 的事件 /40
 - 2.3.1 通过 IHttpModule 创建 HttpApplication 的事件处理程序 /40
 - 2.3.2 注册 HttpModule /41
 - 2.3.3 不使用配置文件注册 HttpModule /42
 - 2.3.4 常见的 HttpModule /44
 - 2.3.5 HttpModule 的事件 /46
 - 2.3.6 通过 global.asax 创建 HttpApplication 的事件处理程序 /46
 - 2.3.7 global.asax 中 HttpApplication 事件的自动注册 /48
 - 2.3.8 特殊的 HttpApplication 事件处理 /48
- 2.4 两个特殊的事件 /51
- 2.5 大文件上传问题 /51
 - 2.5.1 文件上传的规范 /51
 - 2.5.2 ASP.NET 中的文件上传 /53
 - 2.5.3 文件上传的解决方案 /54
 - 2.5.4 通过 HttpModule 接管请求参数 /54
 - 2.5.5 自定义的请求参数对象 /58
 - 2.5.6 读取上传数据的接口和实现 /62
 - 2.5.7 读取上传数据流 /65
 - 2.5.8 注册自定义的上传管理 /76
 - 2.5.9 使用自定义的上传管理 /77

- 2.6 各种各样的文件上传 /78
 - 2.6.1 无刷新的上传：jQuery form /78
 - 2.6.2 基于客户端技术的上传进度：SWFUpload /79
- 2.7 本章小结 /79

第3章 HTTP 请求处理程序 /80

- 3.1 处理程序 /81
 - 3.1.1 处理程序与 `HttpApplication` 的关系 /81
 - 3.1.2 处理程序接口 `IHandler` 和 `IAsyncHandler` /81
 - 3.1.3 在处理程序中使用会话 /82
 - 3.1.4 处理程序工厂 /83
 - 3.1.5 注册处理程序 /83
 - 3.1.6 使用处理程序生成验证码 /84
- 3.2 一般处理程序 /86
 - 3.2.1 一般处理程序工厂 /87
 - 3.2.2 使用一般处理程序的场合 /87
 - 3.2.3 使用一般处理程序生成验证码图片 /87
 - 3.2.4 使用一般处理程序生成 JSON /87
- 3.3 页面处理程序 /91
 - 3.3.1 页面处理程序工厂 /91
 - 3.3.2 创建页面处理程序 /92
 - 3.3.3 生成的代码 /93
 - 3.3.4 使用页面处理程序 /93
- 3.4 Web 服务处理程序 /94
 - 3.4.1 Web 服务处理程序工厂 /94
 - 3.4.2 使用 Web 服务处理程序 /95
 - 3.4.3 Web 服务的常用标签 /96
 - 3.4.4 派生自 `System.Web.Services.WebService` 类的意义 /98
- 3.5 MVC 处理程序 /98
 - 3.5.1 MVC 的路由接口 `IRouteHandler` /99
 - 3.5.2 自定义的 `IRouteHandler` /100
 - 3.5.3 注册路由处理程序 /101
 - 3.5.4 获取控制器的工厂接口 `ControllerFactory` /102
 - 3.5.5 MVC 请求的处理过程 /102
- 3.6 资源处理程序 /103
 - 3.6.1 资源的处理程序配置 /103

- 3.6.2 定义嵌入的资源 /103
- 3.6.3 获取资源的地址 /104
- 3.6.4 使用嵌入的资源 /104
- 3.7 禁止的处理程序 /105
 - 3.7.1 配置禁止访问的资源 /105
 - 3.7.2 禁止访问 Excel /106
- 3.8 虚拟路径提供器 /106
 - 3.8.1 定义虚拟路径提供器 /106
 - 3.8.2 注册虚拟路径提供器 /107
 - 3.8.3 压缩文件中的网站 /109
 - 3.8.4 SharpZipLib /114
- 3.9 本章小结 /115

第4章 ASP.NET 中的线程与异步 /116

- 4.1 线程基础 /117
 - 4.1.1 线程 /117
 - 4.1.2 自定义线程 /118
 - 4.1.3 前台线程和后台线程 /119
 - 4.1.4 工作者线程和 I/O 线程 /119
 - 4.1.5 线程池 /120
- 4.2 .NET 中线程处理 /121
 - 4.2.1 线程的创建与启动 /121
 - 4.2.2 线程的状态 /123
 - 4.2.3 线程的执行上下文 /124
 - 4.2.4 异步编程模式 APM /125
 - 4.2.5 基于事件的异步编程模式 EPM /128
 - 4.2.6 异步线程的状态与同步问题 /129
 - 4.2.7 处理管道中的异步问题 /133
- 4.3 线程池 /137
 - 4.3.1 线程池的工作原理 /137
 - 4.3.2 将工作者线程加入线程池 /138
 - 4.3.3 将 I/O 线程加入线程池 /138
- 4.4 HttpApplication 中的异步线程 /139
 - 4.4.1 ASP.NET 中的线程池设置 /139
 - 4.4.2 异步步骤中的异步点 /141
 - 4.4.3 启动和完成异步步骤 /142

- 4.5 异步处理程序 /142
 - 4.5.1 异步处理程序接口 /143
 - 4.5.2 在处理程序中异步调用 Web 服务 /143
- 4.6 异步页面 /144
 - 4.6.1 页面异步任务的启动和完成 /144
 - 4.6.2 异步页面任务 /145
 - 4.6.3 异步页面中访问 Web 服务三种方式 /146
 - 4.6.4 实例——查询 QQ 在线状态 /148
- 4.7 本章小结 /151

第5章 页面即对象 /152

- 5.1 流动的网页 /153
 - 5.1.1 字节流 /154
 - 5.1.2 字符编码 /155
 - 5.1.3 字符流 /156
 - 5.1.4 回应对象中的流 /158
 - 5.1.5 专门输出 HTML 的字符流 /158
- 5.2 控件——页面对象的基石 /160
 - 5.2.1 控件类 /160
 - 5.2.2 Render 和 RenderControl /161
 - 5.2.3 控件基类 /161
 - 5.2.4 组合模式 Composite /163
 - 5.2.5 ID 是一个问题 /165
- 5.3 形形色色的控件 /171
 - 5.3.1 HTML 控件 /172
 - 5.3.2 Web 控件 /172
 - 5.3.3 WebPart 控件 /174
- 5.4 控件实现的常用接口 /175
 - 5.4.1 生成和回发 /175
 - 5.4.2 控件的任意属性 IAttributeAccessor /177
 - 5.4.3 数据的回发 IPostBackDataHandler /177
 - 5.4.4 回发服务器端事件 IPostBackEventHandler /178
- 5.5 页面 /178
 - 5.5.1 页面与模板 /179
 - 5.5.2 母版页 /184
 - 5.5.3 页面就是一个处理程序 /185

- 5.5.4 页面的事件处理管道 /185
- 5.5.5 处理页面的事件 /187
- 5.6 生成的过程 /188
 - 5.6.1 从模板到对象模型——BuildProvider /189
 - 5.6.2 从标记到控件——ControlBuilder /191
 - 5.6.3 进入生成阶段——ControlAdapter /191
 - 5.6.4 控件适配器——ControlAdapter /192
 - 5.6.5 Web 控件适配器——WebControlAdapter /193
 - 5.6.6 页面适配器——PageAdapter /193
 - 5.6.7 使用 Adapter 定制表单的 action /194
- 5.7 自定义的 URL 重写 /195
 - 5.7.1 URL 重写的原理 /196
 - 5.7.2 使用 HttpModule 实现 URL 重写 /196
 - 5.7.3 在配置文件中处理重写映射 /199
 - 5.7.4 无扩展名请求的处理问题 /204
- 5.8 本章小结 /205

第 6 章 状态 /206

- 6.1 基本状态管理 /207
 - 6.1.1 隐藏域 /207
 - 6.1.2 Cookie /209
 - 6.1.3 URL /213
- 6.2 视图状态 ViewState /215
 - 6.2.1 序列化和反序列化 /215
 - 6.2.2 控制序列化 /217
 - 6.2.3 Base64 /218
 - 6.2.4 视图状态属性与 IStateManager 接口 /219
 - 6.2.5 保存和恢复的时间点 /221
 - 6.2.6 视图状态的序列化器 /223
 - 6.2.7 使用视图状态实现路径导航 /224
- 6.3 控件状态 ControlState /228
- 6.4 应用程序状态 Application /229
- 6.5 会话状态 Session /229
 - 6.5.1 服务器端的 Session /230
 - 6.5.2 客户端的 SessionID /231
 - 6.5.3 Session 保存的位置 /233

- 6.5.4 Session 的过期问题 /237
- 6.5.5 压缩 Session 数据 /238
- 6.6 HttpContext 状态 /238
- 6.7 Cache /238
 - 6.7.1 缓存的原理 /238
 - 6.7.2 .NET 中的缓存管理实现 /239
 - 6.7.3 基于文件的缓存依赖 /241
 - 6.7.4 基于 SQL 的缓存依赖 /242
 - 6.7.5 组合的缓存依赖 /244
 - 6.7.6 删除所有的缓存项目 /246
 - 6.7.7 Web 服务器端的页面缓存 /247
 - 6.7.8 页面局部缓存 /249
 - 6.7.9 自定义的输出缓存提供器 /253
- 6.8 Memcached /254
 - 6.8.1 下载和安装 Memcached /255
 - 6.8.2 在 ASP.NET 中访问 Memcached /256
- 6.9 统计当前在线用户 /258
 - 6.9.1 Module 的处理 /258
 - 6.9.2 注册 Module /261
 - 6.9.3 Module 的配置参数 /262
 - 6.9.4 Module 的事件处理 /263
- 6.10 本章小结 /263

第 7 章 模板和数据绑定 /264

- 7.1 页面与绑定 /265
 - 7.1.1 嵌入式代码块和表达式 /265
 - 7.1.2 绑定表达式 /267
 - 7.1.3 目标 Target /269
 - 7.1.4 容器 Container /269
 - 7.1.5 触发绑定事件的方法 DataBind /270
- 7.2 控件内的模板 /271
 - 7.2.1 基于模板的控件 /271
 - 7.2.2 控件模板中的 Container /272
 - 7.2.3 DataBinder /272
 - 7.2.4 Page 中的 Eval /273
 - 7.2.5 在属性中使用绑定表达式 /273

- 7.3 Repeater 控件 /273
 - 7.3.1 数据的来源 DataSource /273
 - 7.3.2 Repeater 的基石——RepeaterItem /274
 - 7.3.3 绑定的过程 /275
 - 7.3.4 绑定中的事件 /276
 - 7.3.5 绑定的结果：Controls 集合和 Items 集合 /276
 - 7.3.6 回发中的 ItemCommand 事件 /277
- 7.4 高级数据控件 /279
 - 7.4.1 唯一支持分栏的控件-DataList /279
 - 7.4.2 GridView /283
 - 7.4.3 ListView 和 DataPager /289
- 7.5 数据源控件 /291
 - 7.5.1 反射 /292
 - 7.5.2 两种数据源 /293
 - 7.5.3 对象数据源 /297
 - 7.5.4 业务对象的标签 /298
 - 7.5.5 页面控件与数据源控件之间的关系 /300
 - 7.5.6 数据源控件相关的事件点 /303
 - 7.5.7 基于数据源控件的分页 /304
- 7.6 本章小结 /305

第 8 章 自定义控件 /306

- 8.1 自定义控件的继承体系 /307
- 8.2 自定义控件涉及的相关类型 /307
- 8.3 自定义的带有上传进度的按钮 /308
 - 8.3.1 控件的工作原理 /308
 - 8.3.2 选择控件的基类 /309
 - 8.3.3 自定义的数据类型 /309
 - 8.3.4 状态的持久化 /310
 - 8.3.5 控件的属性 /312
 - 8.3.6 属性转换问题——TypeConverter /313
 - 8.3.7 编辑属性数据 UITypeEditor /320
 - 8.3.8 保存在 ASPX 中 /326
 - 8.3.9 设计器中的显示效果 /329
 - 8.3.10 工具栏中控件的图标 /330
 - 8.3.11 脚本嵌入和使用 /330

- 8.3.12 控件的呈现 /332
- 8.3.13 使用自定义控件 /334
- 8.4 PetShop 中的自定义控件 /335
 - 8.4.1 基类 /335
 - 8.4.2 表格 /335
 - 8.4.3 处理当前页码参数 /336
 - 8.4.4 DataSource 属性 /336
 - 8.4.5 事件 /337
 - 8.4.6 生成 /338
- 8.5 本章小结 /342

第9章 MVC /343

- 9.1 ASP.NET MVC 是表现层的 MVC /344
- 9.2 在 HttpApplication 中的 ASP.NET MVC /344
 - 9.2.1 创建 RouteTable /345
 - 9.2.2 UrlRoutingModule 事件处理 /347
- 9.3 从 URL 进入 MVC 之门 /348
 - 9.3.1 有意义的 URL /349
 - 9.3.2 在 IIS 6.0 和 IIS 7 中的配置 /349
 - 9.3.3 从 URL 到 Route /350
 - 9.3.4 约束 /354
 - 9.3.5 Routing /356
 - 9.3.6 RequestContext 的前世今生 /357
 - 9.3.7 在 ASP.NET MVC 中防盗链 /358
- 9.4 控制器 /361
 - 9.4.1 控制器工厂 /361
 - 9.4.2 使用自定义的控制器工厂 /362
 - 9.4.3 为 Controller 类传递构造函数的参数 /362
 - 9.4.4 Controller 的继承关系 /364
 - 9.4.5 Controller 中的状态管理 /365
 - 9.4.6 基于过滤器的扩展 /368
 - 9.4.7 选择 Action /372
- 9.5 模型 /373
 - 9.5.1 绑定 Model /374
 - 9.5.2 简单参数和复杂参数 /374
 - 9.5.3 模型对象的元数据 /375

- 9.5.4 Model 的验证 /377
- 9.5.5 自定义 Model 的验证 /379
- 9.6 执行 Action /380
 - 9.6.1 各种 ActionResult /380
 - 9.6.2 向视图传递数据 /381
- 9.7 视图 /382
 - 9.7.1 视图引擎 /382
 - 9.7.2 经典视图——ViewPage /383
 - 9.7.3 视图引擎——Razor /384
- 9.8 本章小结 /384

第 10 章 IIS 与 ASP.NET /385

- 10.1 网站 /386
 - 10.1.1 绑定 /386
 - 10.1.2 网站应用程序 /388
 - 10.1.3 虚拟目录 /389
- 10.2 通过 ISAPI 扩展 IIS /390
 - 10.2.1 ISAPI 扩展 /391
 - 10.2.2 ISAPI 过滤器 /392
 - 10.2.3 CLR 是一个 COM 组件 /394
 - 10.2.4 ASP.NET 中的 ISAPI 扩展和过滤器 /396
 - 10.2.5 ISAPI Rewrite /396
- 10.3 IIS 与 ASP.NET /397
 - 10.3.1 IIS5 与 ASP.NET /397
 - 10.3.2 IIS6 与 ASP.NET /398
 - 10.3.3 IIS7 与 ASP.NET /399
- 10.4 创建网站的两种方法 /403
- 10.5 ASP.NET 中的加密与解密 /405
 - 10.5.1 machineKey /405
 - 10.5.2 加密服务 /407
 - 10.5.3 配置节的加密和解密 /408
- 10.6 本章小结 /410

第 11 章 ASP.NET 中的用户 /411

- 11.1 从 IIS 开始 /412

- 11.1.1 匿名用户方式 /413
 - 11.1.2 基本身份验证 /413
 - 11.1.3 摘要式身份验证 /414
 - 11.1.4 集成 Windows 身份验证 /415
 - 11.1.5 .NET Passport 身份验证 /416
- 11.2 ASP.NET 中的用户信息 /416
 - 11.2.1 基于 Windows 验证的用户 /416
 - 11.2.2 基于 Forms 验证的用户 /417
 - 11.2.3 基于 Passport 验证的用户 /420
 - 11.2.4 在 IIS7 中使用表单验证 /420
- 11.3 .NET 中的用户 /421
 - 11.3.1 用户的标识 IIdentity /421
 - 11.3.2 用户 IPrincipal /422
- 11.4 网站中的用户 /423
- 11.5 成员管理 /423
 - 11.5.1 用户的基本信息 /424
 - 11.5.2 成员管理的约定 /425
 - 11.5.3 基于 SQLServer 的成员管理实现 /426
 - 11.5.4 自定义的成员管理实现 /427
 - 11.5.5 辅助工具类 Membership /432
- 11.6 用户的扩展信息——个性化数据 /433
 - 11.6.1 个性化数据的约定 /433
 - 11.6.2 个性化数据的属性 /434
 - 11.6.3 实现自定义的个性化数据管理 /436
 - 11.6.4 匿名的个性化数据 /439
 - 11.6.5 合并匿名用户的个性化数据 /441
 - 11.6.6 基于 SQL Server 的个性化数据管理 /442
 - 11.6.7 管理个性化数据 /444
- 11.7 执行程序的用户 /444
 - 11.7.1 执行网站程序的 Windows 用户 /445
 - 11.7.2 用户模拟的作用 /446
 - 11.7.3 数据库连接串中的用户 /447
- 11.8 本章小结 /449

附录 自定义配置参数 /450