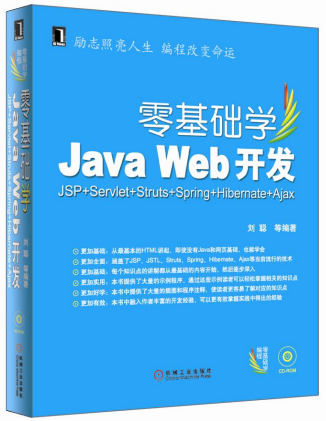
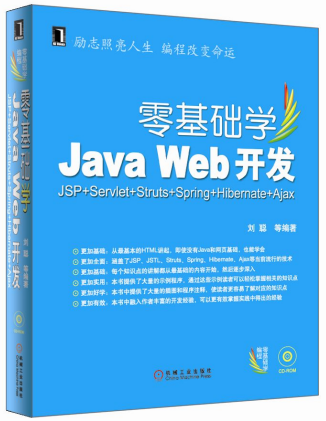
第一章　背景知识简介

- 本书的重点内容是讲解Java Web开发的知识，在本章中，首先简单介绍了Java语言的历史和现状，然后对网站运行的基本知识进行了简单的介绍，在本章的最后，对比了各种动态开发语言之间，介绍了各种动态Web开发语言的优劣，通过本章知识的学习，读者可以掌握Java Web开发所需的基本知识。



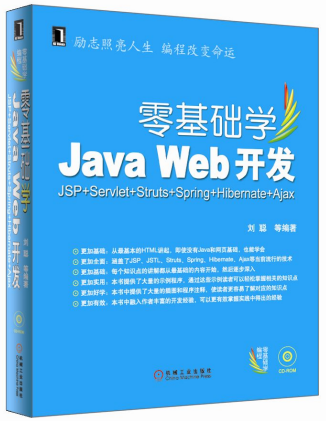
1.1　Java语言简介

- Java是一种跨平台的面向对象语言，Java语言的目标就是为了满足在复杂的网络环境中开发软件，在这种复杂的网络环境中，充满这各种各样的硬件平台和不同的软件环境，而Java语言就是针对这种复杂的平台环境设计，使用Java语言，可以开发出适应这种复杂网络环境的应用系统。



1.1.1　Java语言简介

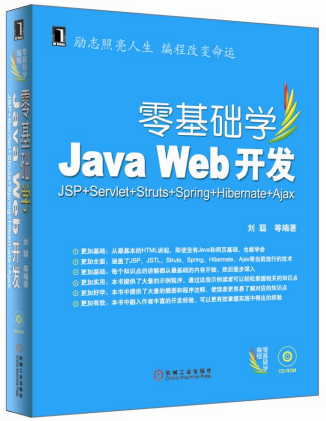
- Java是一种优秀的面向对象语言，在Java语言中，有着健壮的安全设计，它的结构是中立的，可以一直到不同的系统平台，优秀的多线程设计也是Java语言的一大特色，但是Java语言的最大优势还是在于其对多种操作系统平台的支持，这种特性是其他编程语言所无法比拟的。



1.1.2　Java语言的特性和优势

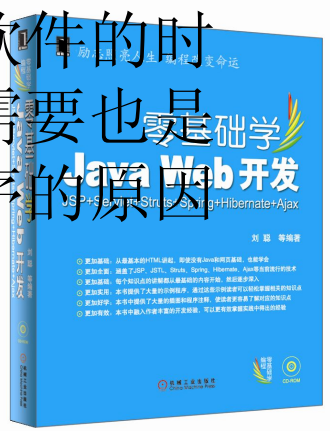
- 在目前的软件开发中，尤其是应用系统的开发中，Java语言成为大部分开发人员的选择，Java语言的特性：

- （1）　平台无关性
- （2）　安全性
- （3）　面向对象
- （4）　异常处理



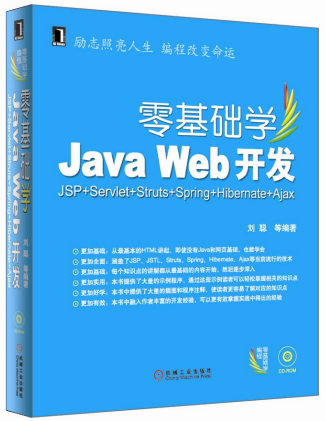
1.1.3　Java语言的发展现状

- Java语言并不是为网络环境设计的，用户可以使用Java语言来编写独立的桌面应用程序，在桌面应用程序这个领域，Java已经被各大厂商接受，例如Oracle数据库、Borland的JBuilder开发环境，Eclipse开发环境等工具都是使用Java语言编写的，这些软件产品的性能都是非常优秀的，可见使用Java同样可以编写出功能强大的应用软件。而且，如果用户需要开发跨平台运行的软件的时候，Java就成了唯一的选择，跨平台的需要也是各大厂商选择使用Java开发桌面应用程序的原因之一。



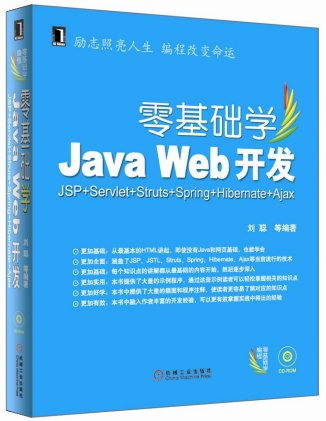
1.1.4　Java语言的发展前景

- 随着网络技术的急速发展，Java语言必然会取得更大的发展，在这个复杂的网络环境中，Java语言有着广阔的前景。例如在如下几种开发需求中，Java语言都有着很大的发展前景：



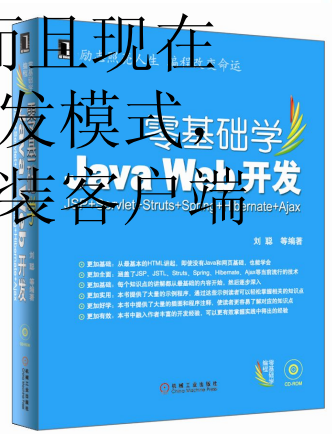
1.2　Web应用程序开发基本知识

- Java Web开发也就是基于B/S结构的Java应用程序开发，在接下来的章节中，将介绍Java Web开发最基本的知识，在这里不涉及具体的技术实现，只对Java Web开发的基本原理进行介绍。



1.2.1　Web应用程序的运行原理

- 在传统的Web应用程序开发中，需要同时开发客户端和服务器端的程序，由服务器端的程序提供基本的服务，客户端是提供给用户的访问接口，用户可以通过客户端的软件访问服务器提供的服务，这种Web应用程序的开发模式就是传统的C/S开发模式，在这种模式中，由服务器端和客户端的共同配合来完成复杂的业务逻辑。例如以前的网络软件中，一般都会采用这种模式，而且现在的网络游戏中，一般还会采用这种Web开发模式。在这些Web应用程序中，都是需要用户安装客户端才可以使用的。



1.2.2 Web服务器汇总

- 在C/S架构的开发模式中，服务器端完全是有开发人员自己提供，开发人员自己制定客户端的访问规则，这时候的服务器就是不仅要提供逻辑功能的服务，还要提供一点的协议支持，通过这样的协议，客户端程序才可以与服务器端进行通信，从而享受服务器端提供的服务。在B/S架构的开发模式中，客户端就是简单的浏览器程序，可以通过HTTP协议访问服务器端的应用，在服务器端，与通信相关的处理都是由服务器软件负责，这些服务器软件都是有第三方的软件厂商提供，开发人员只需要把功能代码部署在Web服务器中，客户端就可以通过浏览器访问到这些功能代码，从而实现向客户提供的服务

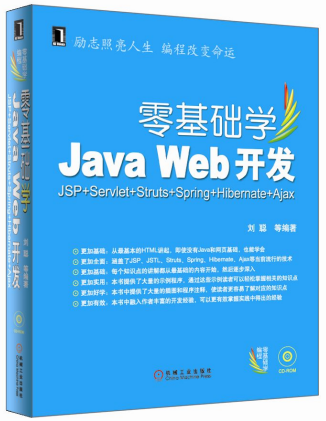
1.2.3　开发一个Web应用程序的简单流程

- 在传统Web应用程序的开发过程中，开发一个应用系统一般情况下需要以下几个步骤：客户端/服务器端软件的开发、服务器端程序的部署、客户端软件的安装，只有完成这几个步骤，用户才可以通过客户端访问服务器提供的服务。

- 而在基于B/S架构的Web程序大开发过程中，只需要开发服务器端的功能代码，然后把服务器端的程序部署在Web服务器软件中即可，在部署结束之后，启动Web服务器，用户就可以通过浏览器访问Web应用程序提供的服务。

1.3 Web 应用程序开发

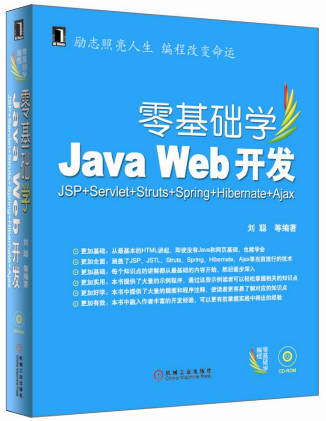
- 由于技术的进步和网络环境的进化，Web应用程序开发的技术也在不断的进步，在Web应用程序开发的过程中，存在着不少争议，当然，这些争议都是开发人员对各种技术的看法不同造成的，在接下来的内容中，简单介绍这方面的内容，是读者对技术进化过程中的一些问题有所了解。



1.3.1　C/S 与B/S之争

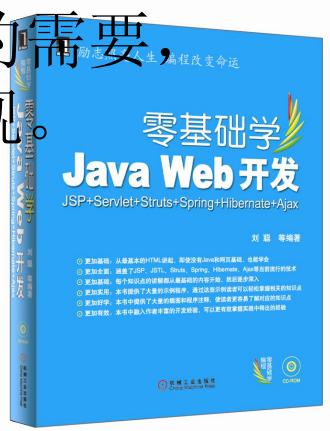
- 在前面的章节中已经介绍过，在Web应用程序的开发中，存在这两种开发模式，一种是传统的C/S架构，另一种是近些兴起的B/S架构。

- 由于硬件成本的降低，再加上应用系统复杂程度的提高，Web应用程序的开发逐渐转向到C/S架构，



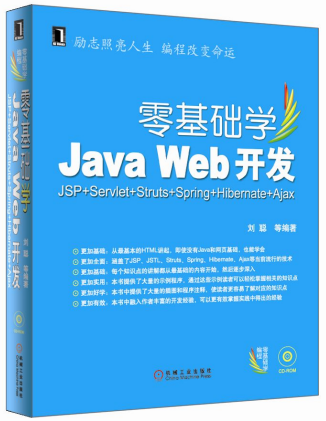
1.3.2　动态页面语言对比

- 在互联网发展的最初阶段，所有的网页内容都是静态的HTML网页，在这种情况下，网站所能实现的任务仅仅是静态的信息展示，而不能与客户产生互动，当然这样的网站是不能满足用户不同的需要。在现实的生活中，用户的需要总是各种各样的，这就需要网站或者是Web应用程序具有收集并处理响应用户需要的功能，而静态的HTML是不能满足这种需要的，为了满足这种特殊的需要，就有了后来一系列的动态页面语言的出现。



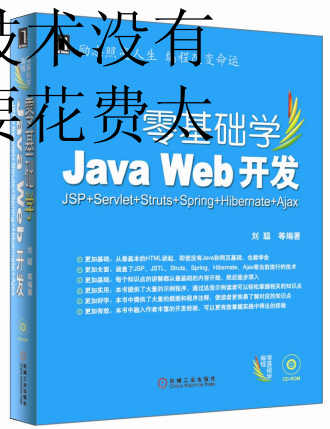
1.3.3 .NET 与 J2EE之争

- 自从.NET和J2EE推出以来，对J2EE和.NET的比较已经不是一天两天的事了，钟情于Windows的用户会选择.NET，而选择Unix\Linux的用户会更钟情于J2EE，其实这两种技术都有各自的优势和不足（具体内容请参照书。）



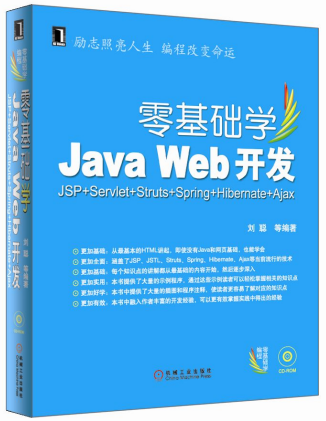
1.4 小结

- 在本章内容中，对Java Web开发中的一些基本知识进行简单的介绍，读者通过本章的学习可以了解开发Java Web应用程序的一些基本的概念，而且对于Java Web开发中的一些存在争议的问题也有所了解，尤其是一些有争议的问题，读者可以稍加注意，在初学者中，很容易犯这些错误，例如会过多关注具体技术的优劣，期望学到一种最有用的技术，这些想法都是不可取的。技术没有高低分，只有应用场合的不同。所以不要花费大多的精力来考虑这种没有意义的问题。



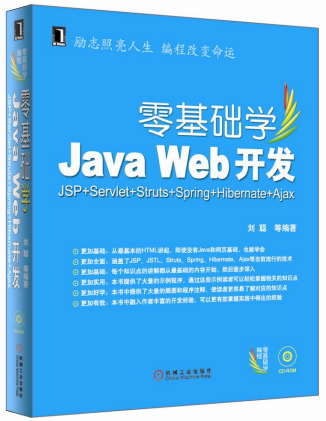
第二章　Java开发环境及开发工具

- 本章主要介绍Java开发环境的搭建，首先介绍JDK的下载安装和环境变量的设置，并通过一个简单的示例程序展示JDK的简单使用方法，对于Java开发工具方面，简单介绍集成开发环境Eclipse的基本使用方法，通过本章的学习，读者可以迅速掌握Java开发环境的搭建，并对Eclipse开发工具的基本用法有所了解。



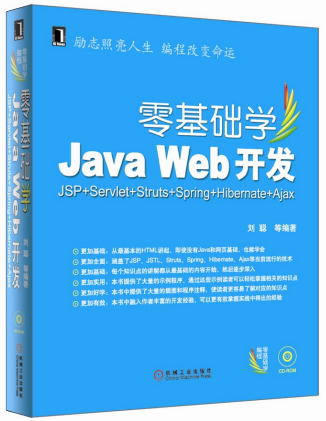
2.1　下载安装JDK

- JDK中包含了Java开发中必需的工具和Java程序的运行环境（即JRE）。（具体内容请参照书。）



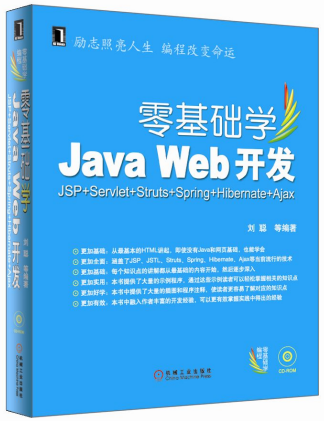
2.2　环境变量设置

- 在上面的章节中，介绍了JDK的安装方法，但是在JDK安装结束之后，必需进行环境变量的设置，然后才可以使用JDK提供的开发工具。下面对环境变量的设置步骤进行详细的介绍。（具体内容请参照书。）



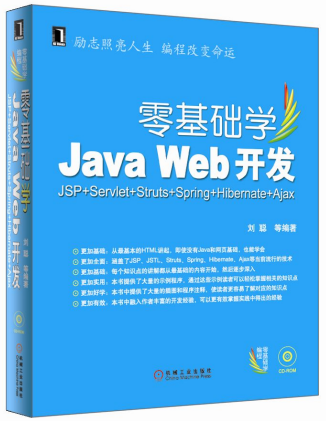
2.3　小试身手——HelloWorld

- 在上面两个小节的介绍中，已经成功安装配置JDK，在JDK中提供了编译执行Java的基本工具，使用这些工具已经可以进行基本的Java程序的编写工作，虽然在使用继承的开发环境进行开发的效率会更高，但是，为了是读者对JDK的基本使用方法有基本的了解，在接下来的内容中，将不使用集成开发环境，而是通过DOS命令行对简单的Java示例程序进行编译和运行。



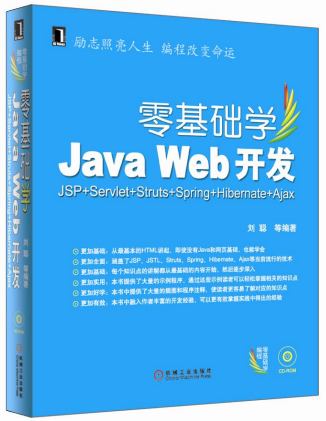
2.3.1　编辑Java源文件

- 在编辑Java源文件的时候，有很多工具可供选择，只要是能够进行简单文本编辑的工具都可以用来编辑Java源文件。在这里我们选择使用Windows中自带的记事本工具。在记事本中输入下面的代码。



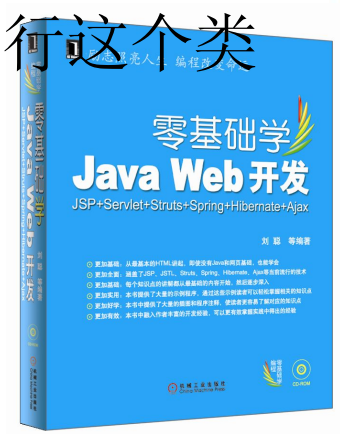
2.3.2　编译Java源文件

- 完成对Java源文件的编辑工作以后，就可以对源代码进行编译，在JDK中提供了编译Java源文件的工具，可以在DOS命令行中调用JDK中的javac命令，这个命令可以对Java源文件进行编译。



2.3.3　执行类文件

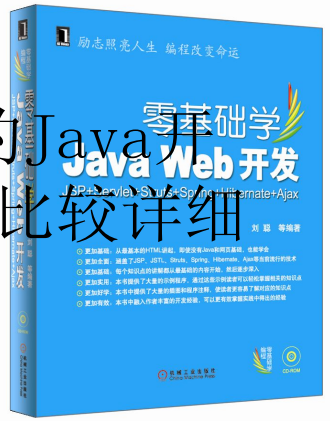
- 在编译工作成功通过以后，可以得到对应的Java类文件，在JDK中同样提供了执行Java类文件的工具，可以在DOS命令行中调用java命令执行Java的类文件。在上面的操作中，成功编译了HelloWorld.java这个Java源文件，并在C盘的根目录下生成了HelloWorld.class文件，在DOS命令行中需要把当前的路径切换到Java类文件的目录，然后调用JDK中的java命令就可以执行这个类文件，



2.4　开发工具Eclipse简介

- 在前面章节的内容中，介绍了直接使用JDK提高的工具开发一个简单的示例程序，在这个示例程序的开发过程中，没有使用任何集成的开发工具，这只是为了使读者对JDK的功能有一个大体的了解，在实际的开发过程中，是不可能脱离集成开发工具的帮助的，使用集成开发工具可以大大提高开发效率，从而保证项目的进度。

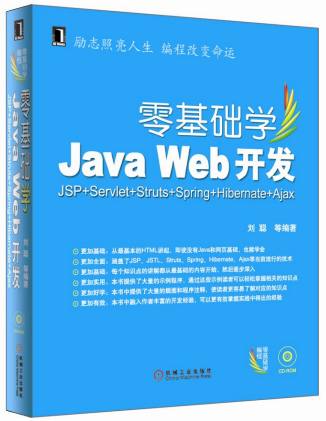
- 在本节的内容中，将简单介绍几种常用的Java开发工具，其中，对Eclipse开发平台会做比较详细的介绍。



2.4.1　Java开发工具简介

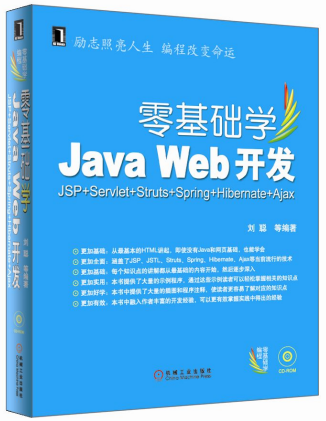
- 目前常用的Java开发工具基本上可以分为两大类。

- 一种是简单小巧的开发工具。例如TextPad、JCreator等，另一种是具有强大功能的集成开发环境，例如Eclipse、JBuilder等，



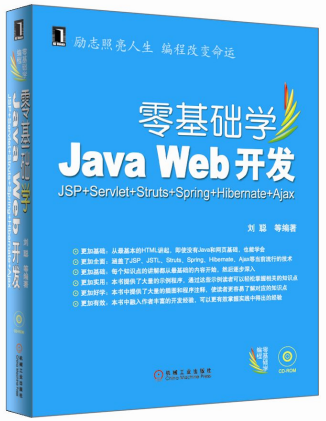
2.4.2　Eclipse安装

- 在Java项目的开发过程中，越来越多的开发人员选择使用Eclipse，在这里就介绍Eclipse开发环境的安装和使用。（具体内容请参照书。）



2.4.3　Eclipse使用简单例程

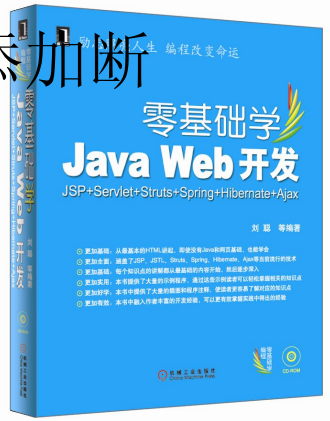
- 在Eclipse中，编译运行Java程序的方法和其他Java开发工具稍微有些不同，在本节的内容中，将对Eclipse的基本使用方法进行简单的介绍。



2.4.4　在Eclipse中调试程序

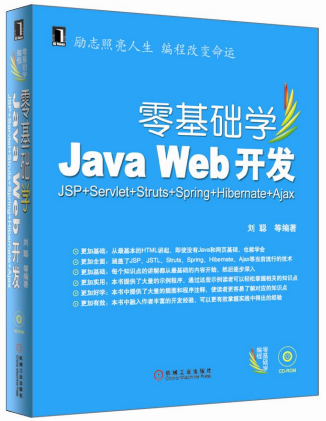
- 在Eclipse中不仅可以方便的编辑执行Java程序，而且还提供了功能强大的调试功能，在调试Java程序的过程中，可以给程序设置断点，程序在运行到断点以后会暂停执行，通过设置断点，可以跟踪程序中的变量，从而对程序中的错误进行定位。

- 要调试程序，首先需要在Java源文件中添加断点，（具体内容请参照书。）



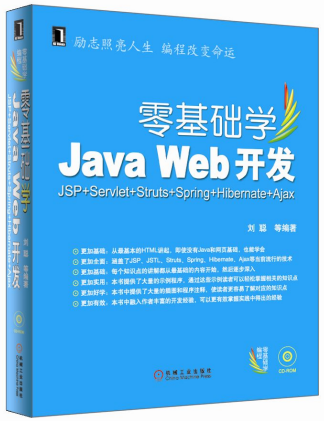
2.4.5　Eclipse常用快捷键

- Eclipse提供了丰富的辅助开发功能，而且很多常用的功能都提供了快捷键，在本节内容中，整理出一些相对比较常用的快捷键。（具体内容请参照书。）



2.4.6 Java Web开发工具MyEclipse简介

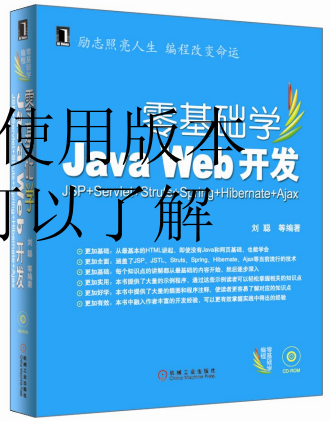
- 在开发Java桌面应用程序的时候，使用Eclipse是非常方便的，但是在进行Web开发的时候，Eclipse的一些功能就不能够满足用户的需求了，在开发Web应用的时候，我们选择使用MyEclipse集成开发工具，MyEclipse是依赖于Eclipse的一个开发工具，对Eclipse的功能进行了扩展，主要是给Eclipse增加了一系列的Web开发工具，从而是Web开发的效率大大提高。



2.5　使用版本控制工具CVS

- 在团队开发中，需要团队各个成员之间进行分工配合，这就给源代码的版本控制带来很大困难，不可能手工来完成源代码版本的迭代，在这里选择使用版本控制工具，通过版本控制工具完成对源代码的控制，各个团队成员只需要把修改过的版本提交给版本控制工具，有版本控制工具来把每个成员提交的版本整合成一个最新的版本。

- 在本节内容中，将介绍如何在Eclipse中使用版本控制工具，通过本节内容的学习，读者可以了解团队合作开发中版本控制的基本知识。



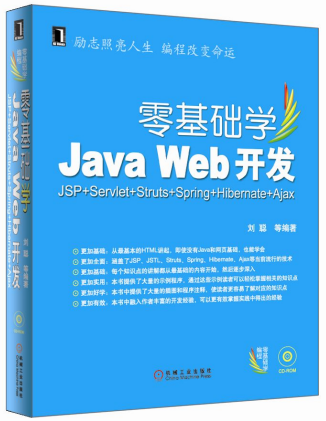
2.5.1　CVSNT的安装与配置

- 在源代码版本控制方面，有很多工具可供选择，在这里选择使用CVS来对版本进行控制，其中CVS需要客户端和服务器端配合使用，在使用CVS的时候，首先需要建立一个CVS服务器，然后团队中的每个成员此可以把自己的版本通过客户端提交给CVS服务器，从而由CVS服务器完成版本的整合更新任务。

- 在这里选择使用CVSNT作为CVS的服务器，CVSNT的安装文件可以从http://www.cvsnt.org/archive/下载，在这里有各种版本的CVSNT安装文件可供下载，用户可以选择下载自己需要的版本，在本书中使用cvsnt-2.5.03.2382，所以选择下载cvsnt-2.5.03.2382.msi这个文件即可。（具体内容请参照书。）

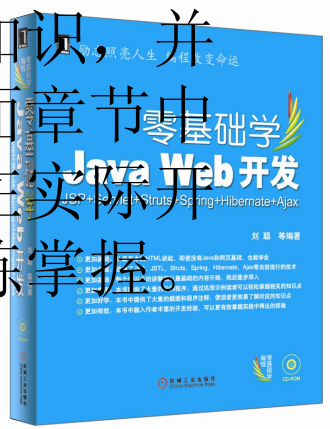
2.5.2　使用Eclipse集成的CVS客户端

- 当CVS服务器安装配置结束以后，就可以通过CVS客户端访问CVS服务器中的资源。其中CVS客户端有很多中选择，例如WinCVS、TortoiseCVS等，使用这些工具都可以非常方便的访问CVS资源，在Eclipse中同样也内置了CVS客户端的功能。在下面的内容中就简单介绍如何使用Eclipse内置的CVS客户端来访问CVS资源。（具体内容请参照书。）



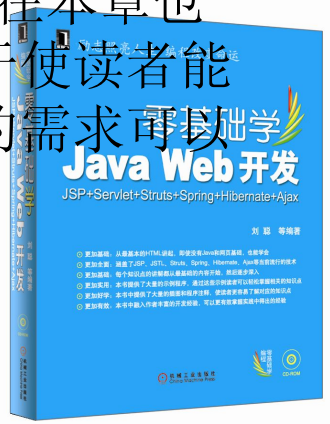
2.6 小结

- 在本章内容中，对Java开发环境的搭建进行了大体的介绍，其中重点讲述了JDK的安装设置和Eclipse的基本使用方法，而且还提供了大量Eclipse中的快捷键，在本章最后的内容中，介绍了团队写作中源代码的版本控制问题，介绍了如何架设CVS服务器，如何使用Eclipse中内置的CVS客户端访问CVS服务器，通过本章内容的学习，读者可以对基本了解Java开发环境的基本知识，并且学会自己搭建设置这样的环境，为后面章节中的开发打下坚实的基础。这些技能都是在实际开发过程中必备的基础技能。读者需要熟练掌握。



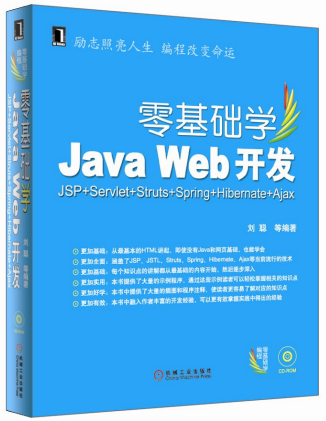
第三章 HTML相关技术基础知识

- 纵观各种动态页面开发技术，无论是JSP、ASP还是PHP都无法摆脱HTML的影子。这些动态的页面开发技术无非是在静态HTML页面的基础上添加了动态的可以交互的内容。HTML是所有动态页面开发技术的基础。在接下来的章节将要详细介绍的就是HTML相关的一系列技术，包括HTML、JavaScript和CSS。其中HTML是一组标签，负责网页的基本表现形式；JavaScript是在客户端浏览器运行的语言，负责在客户端与用户的互动；CSS是一个样式表，起到美化整个页面的功能。本书不是详细介绍HTML的专著，在本章也只是讲解Web开发中最常见的HTML知识，目的在于使读者能尽快进入Web开发的状态。如果读者有更深层次的需求可以参考专门讲解HTML的书籍。



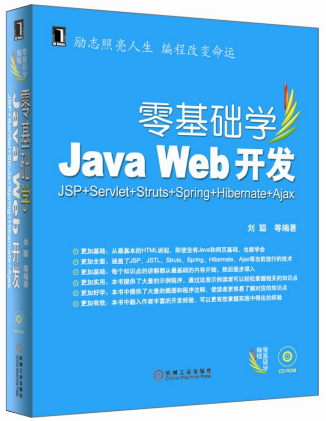
3.1　HTML 基础知识

- 在各种Web开发技术中，HTML无疑是最为基础的。任何动态语言都离不开HTML的支持。所以在开始Web开发的学习之前，读者还是需要静下心来打好这个基础。在这个章节中将会概述HTML的框架知识。



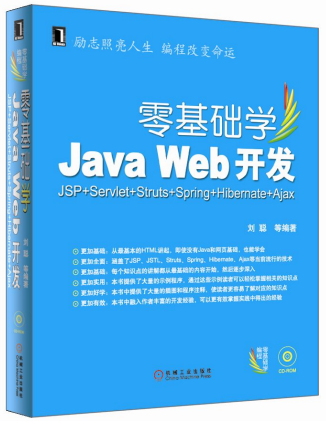
3.1.1　什么是HTML

- HTML（Hyper Text Markup Language）即超文本标记语言，用来描述Web文档数据。用户可以通过URL链接来访问这种Web文档，从而达到信息展示、信息共享的目的。（具体内容请参照书。）



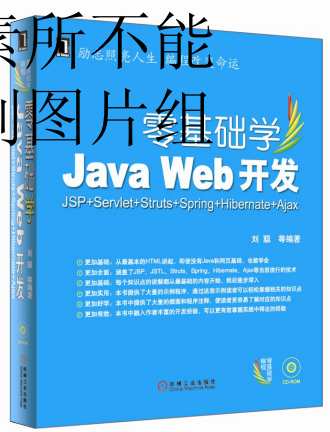
3.1.2　HTML运行原理

- 前面介绍HTML定义的时候就说过，HTML是一种标记语言，每一种HTML标签都是有一定表现含义的。浏览器就是按照HTML标签的语义规则把HTML代码翻译成漂亮的网页。



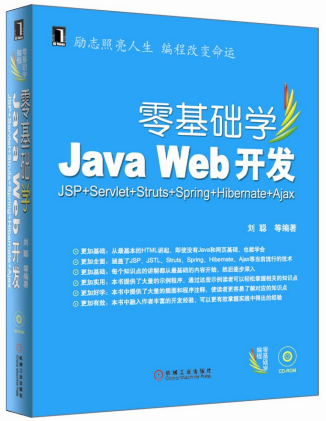
3.1.3　HTML常用标签

- 在本节要介绍的是常用标签的基本用法。
  1．<table>在HTML的布局标签中，<table>标签是使用频率最高的一个。它可以把一组信息用表格的形式表示出来，　2．DIV在以往的Web页面开发中，表格是首选的布局元素，　3．<a>在浏览一个网站的时候，我们经常会遇到一些链接，单击这些链接就会导航的其他的页面。4．<img>在目前的网站开发中，对图片的依赖是其他元素所不能替代的，一个漂亮的网页往往是由一系列图片组合而成。



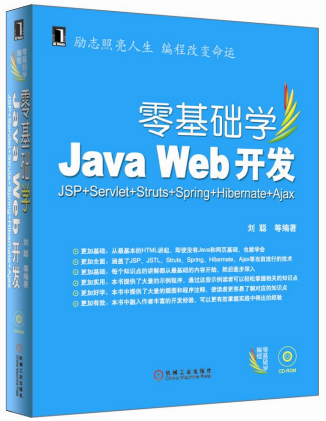
3.1.4　HTML表单标签

- 前面讲述的都是HTML向用户展示信息的标签，在本节要介绍的内容就是HTML用来收集用户输入的标签。<form></form>是表单标签，只有在这个标签中的用户输入才会被提交给服务器。表单标签的使用方法类似下面这种格式。（具体内容请参照书。）



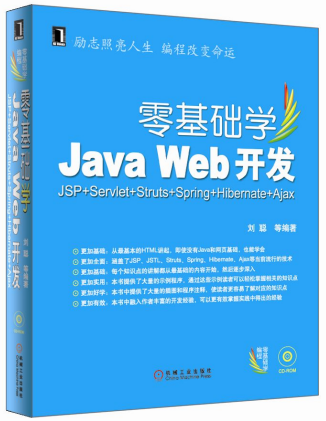
3.1.5　HTML其他标签

- 在本章只是介绍了HTML最基本最常用的几个标签，HTML还有很多其他标签，例如应用程序标签中可以加入不同动态程序代码，多媒体标签中可以加入多媒体文件，Flash标签中可以加入Flash动画，文本标签可以用各种方式组织文本内容的显示。读者如果要深入全面的研究HTML标签，可以参考这方面的参考手册。在本书中不再对这些内容进行详细的介绍。



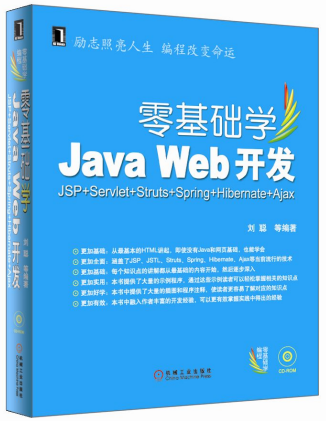
3.2　JavaScript基础知识

- JavaScript的出现给静态的HTML网页带来很大的变化。JavaScript增加了HTML网页的互动性，使以前单调的静态页面变得富有交互性，它可以在浏览器端实现一系列动态的功能，仅仅依靠浏览器就可以完成一些与用户的互动。在下面的章节中将要简单介绍这种技术的基础知识。



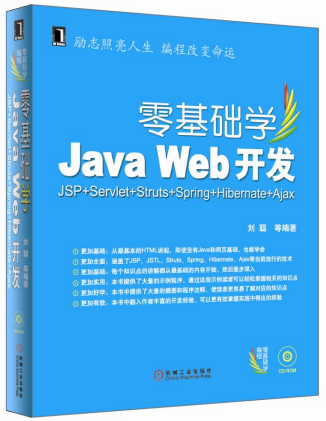
3.2.1　什么是JavaScript

- JavaScript是一种简单的脚本语言，可以在浏览器中直接运行，无需服务器端的支持。这种脚本语言可以直接嵌套在HTML代码中，它响应一系列的事件，当一个JavaScript函数响应的动作发生时，浏览器就会执行对应的JavaScript代码，从而在浏览器端实现与客户的交互。



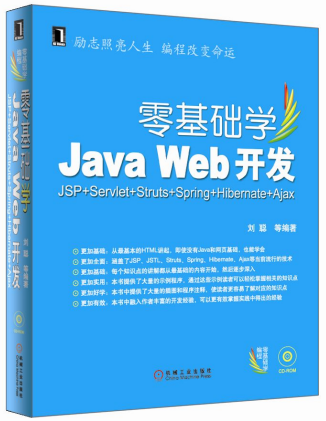
3.2.2　JavaScript中的事件

- 在HTML的标签中，绝大部分都可以触发一些事件，例如鼠标单击、双击、鼠标经过、鼠标离开等一系。JavaScript最主要的功能就是与用户的交互，而用户只能在表单中提交输入内容，所以表单的所有输入标签都可以出发鼠标事件、键盘事件等JavaScript所能响应的常见事件。（具体内容请参照书。）



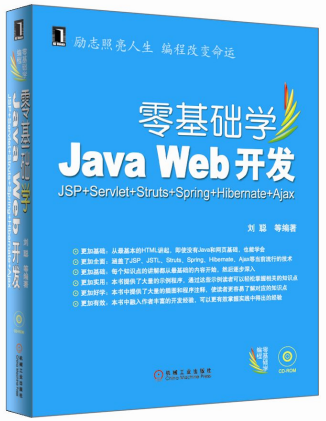
3.2.3　JavaScript中的对象简介

- JavaScript所实现的动态功能，基本上都是对HTML文档或者是HTML文档运行的环境进行的操作。那么要实现这些动态功能就必需找到相应的对象。JavaScript中有已经定义过的对象供开发者调用，



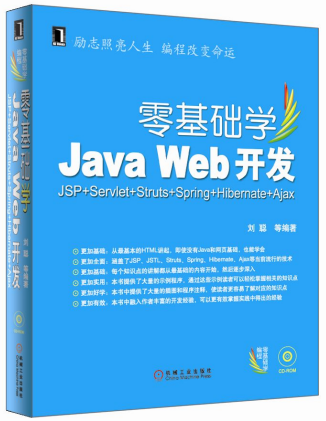
3.2.4　window对象简介

- window对象是所有JavaScript对象中最顶层的对象，整个HTML文档就是在一个浏览器的一个窗口，即window对象中显示。（具体内容请参照书。）



3.2.5　document对象简介

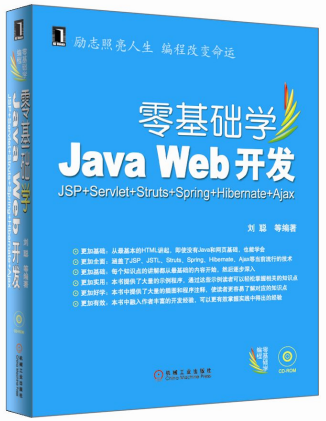
- document对象是在具体的开发过程中用的最频繁的对象，利用document对象可以访问页面上任何的元素。通过控制这些元素可以完成与用户的互动。（具体内容请参照书。）



3.2.6　location对象简介

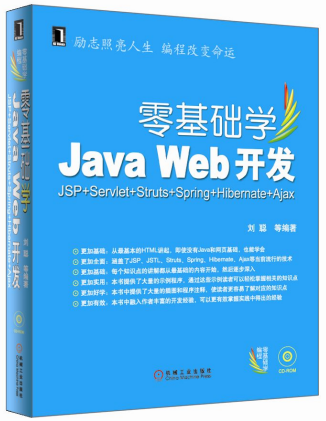
- 在HTML标签中可以用<a>超链接标签来控制网页中的跳转，在JavaScript中如果要实现类似的网页跳转功能只能选择location对象，这个对象的使用方法非常简单，只需要在JavaScript代码中添加下面这行代码即可。

```
window.location.href =
“http://www.sohu.com”;
```



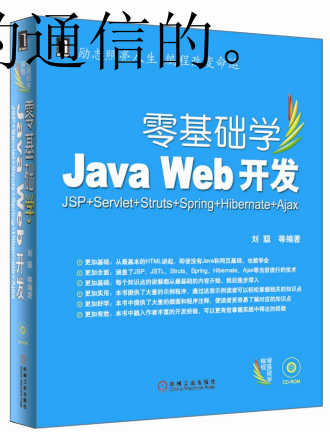
3.2.7　JavaScript输入验证

- 在本章将介绍在浏览器端对用户输入的简单验证，这种验证仅仅局限于输入格式等方面。（具体内容请参照书。）



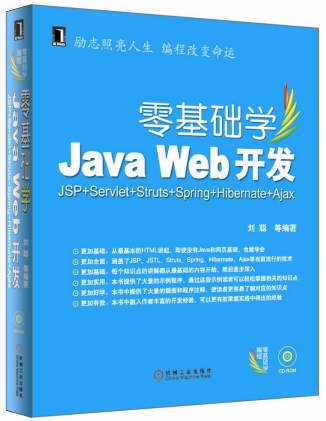
3.2.8 JavaScript高级应用探讨

- 上面介绍的示例中，JavaScript都没有和服务器进行互动，都是在浏览器中独立执行，这样所能实现的与客户互动的功能就比较有限了，例如现在用户注册的时候需要验证这个用户名是否已经被占用，这个功能便需要到服务器中进行查询。然而在我们上面的验证中，只有当表单提交的时候服务器才会相应请求，其他情况下，如果没有刷新整个页面是不能实现与服务器之间的通信的。



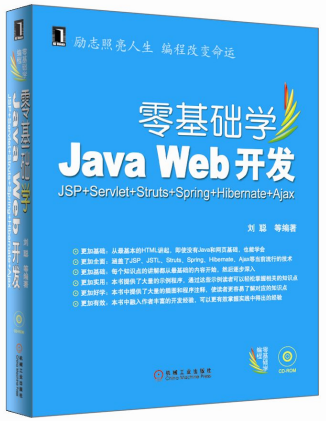
3.3　CSS基础知识

- 在前面的内容中讲解了HTML和JavaScript，现在我们已经基本可以编出具有简单互动的网页，但是仅仅这样还是不够的，一个专业的网页需要在字体、颜色、布局等方面进行各种设置，需要给用户带来视觉的冲击。接下来的内容将要介绍这种美化页面的技术。



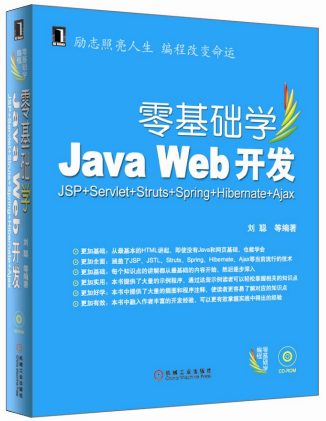
3.3.1　什么是CSS

- CSS（Cascading Style Sheets）即层叠样式表，也就是通常所说样式表。CSS是一种美化网页的技术。通过使用CSS，可以方便、灵活地设置网页中不同元素的外观属性，通过这些设置可以使网页在外观上达到一个更高的级别。



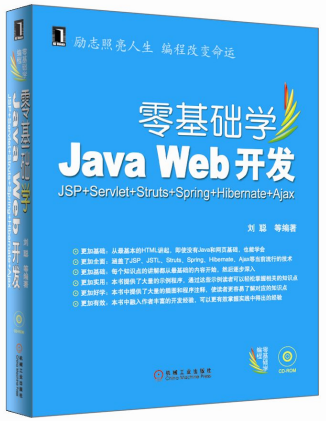
3.3.2　CSS属性设置

- CSS美化网页就是通过设置页面元素的属性来实现的，在下面的内容中将介绍CSS属性设置的基本方法。



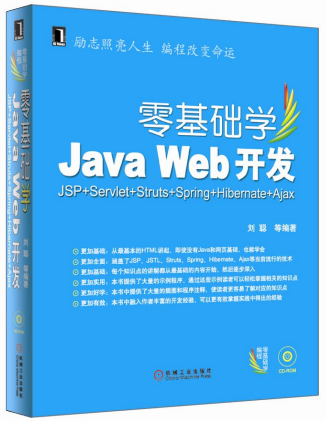
3.3.3　CSS绝对定位示例

- 在HTML中布局一般情况下需要使用表格，如果要定位只有通过表格的嵌套来实现，这种方法的确可以在一定程度上解决问题，但是却不能随意定位页面元素，而且对某个元素位置的改变有可能影响到整个页面的布局。



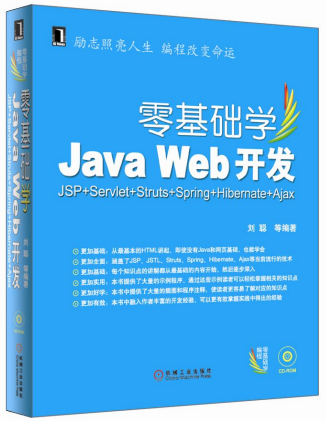
3.3.4 JavaScript+DIV+CSS实现下拉菜单

- 在Web应用中，下拉菜单的可以说是随处可见，在学习了JavaScript和CSS以后实现起来毫无难度。其原理就是在用JavaScript控制不同DIV的显示和隐藏，其中所有的DIV都是用CSS定位方法提前定义好位置和表现形式，下拉的效果只是当鼠标经过的时候触发一个事件，（具体内容请参照书。）



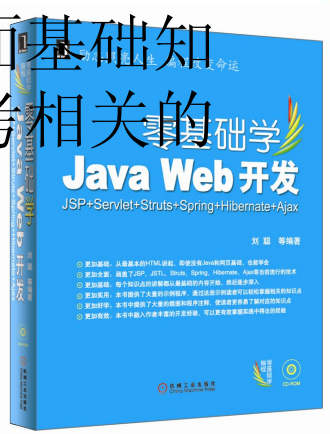
3.3.5 JavaScript+CSS实现表格变色

- 在一些Web应用中间经常会用表格来展示数据，当表格行数比较多的时候，就容易后看错行的情况发生，所以需要一种方法来解决这个问题。在这里我们采取这样一种措施，当鼠标移到某一行的时候，这行的背景颜色发生变化，这样当前行就会比较突出，不容易出错。（具体内容请参照书。）



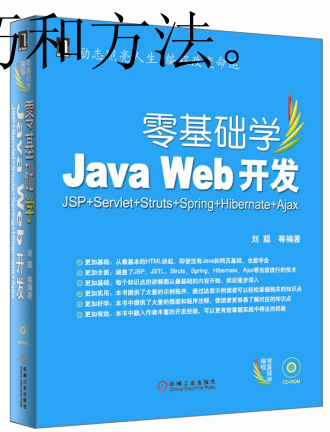
3.4　小结

- HTML是组织展示内容的标记语言，JavaScript是客户端的脚本语言，CSS是美化页面的样式表，这三种技术结合在一起构成了Web开发最基础的知识，所有的Web应用开发都是在这个基础之上进行的。在本章的讲解中，仅仅对这三种技术的大体情况进行了介绍，使读者可以迅速对Web开发的基础知识有一个宏观的清楚的认识，从而可以快速进入后面章节的学习，如果读者对这方面基础知识有更深一步了解的需要，就有必要参考相关的专题书籍。



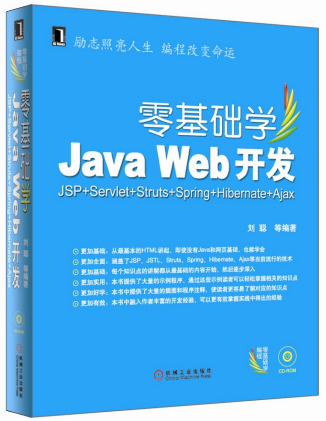
第四章　JSP技术基础知识

- JSP（Java Server Page）是SUN公司开发的一种服务器端的脚本语言，自从1999年推出以来，逐步发展为开发Web应用一项重要技术。JSP可以嵌套在HTML中，而且支持多个操作系统平台，一个用JSP开发的Web应用系统，不用做什么改动就可以在不同的操作系统中运行。在本章接下来的内容中，首先将简单介绍JSP的运行原理和基本语法，然后重点介绍在实际开发过程中技巧和方法。



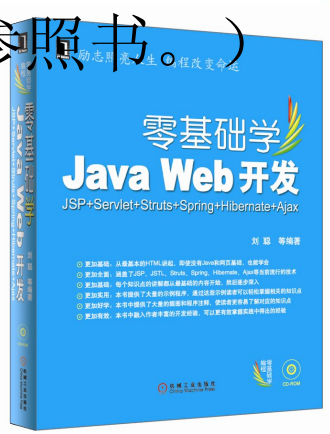
4.1　JSP简介

- JSP本质上就是把Java代码嵌套到HTML中，然后经过JSP容器的编译执行，可以根据这些动态代码的运行结果生成对应的HTML代码，从而可以在客户端的浏览器中正常显示。在这个小节中将介绍JSP的运行原理、JSP的优点和其运行环境的搭建。



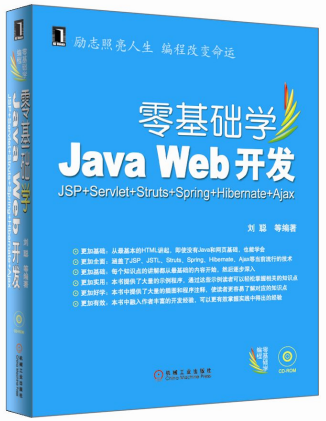
4.1.1　运行原理

- 如果JSP页面是第一次被请求运行，服务器的JSP编译器会生成JSP页面对应的Java代码，并且编译成类文件。当服务器再次收到对这个JSP页面请求的时候，会判断这个JSP页面是否被修改过，如果被修改过就会重新生成Java代码并且重新编译，而且服务器中的垃圾回收方法会把没用的类文件删除。如果没有被修改，服务器就会直接调用以前已经编译过的类文件。（具体内容请参照书。）



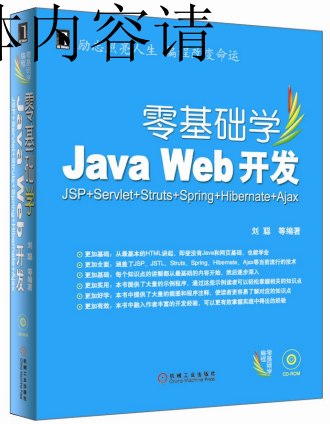
4.1.2　选择JSP的原因

- 在Web应用开发中，可供选择的动态页面语言技术有很多，例如PHP，ASP，JSP等，在这些动态页面语言中，JSP凭借其自身的优点成为开发人员最喜欢的语言之一。下面列出的几条就是开发人员钟爱JSP的重要原因。



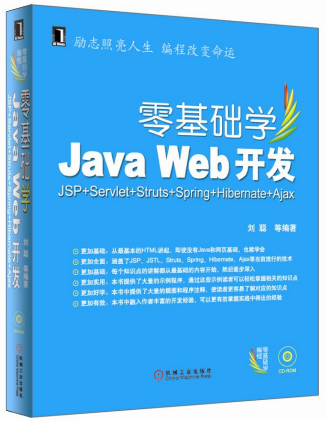
4.1.3　环境搭建

- 要运行JSP程序，必需为其提供一个JSP容器，也就是需要一个Web服务器。支持JSP的服务器非常多，Tomcat、Resin、Weblogic、WebSphere等对JSP的支持都非常好，但是由于Weblogic和WebSphere都是功能非常强大的重量级服务器，而且价格昂贵，对计算机的硬件配置要求也比较高，所以在一般情况下，如果只用到JSP的技术，是没有必要选择这两个服务器的。（具体内容请参照书。）



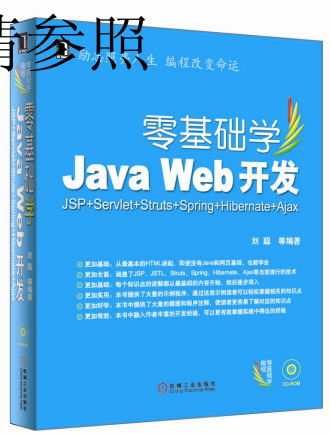
4.2　JSP基本语法

- 本书的重点内容是介绍基于JSP的Web开发技术，对于Java的语法在此不做详细的介绍，这里所涉及JSP语法指的是在JSP中所特有的语法规范，在接下来的章节中将假设读者已经了解Java的基本语法，只介绍JSP的结构、变量声名、表达式、动作、指令等JSP的特有语法。如对Java语法有疑问的读者可以参考相关语法书籍。



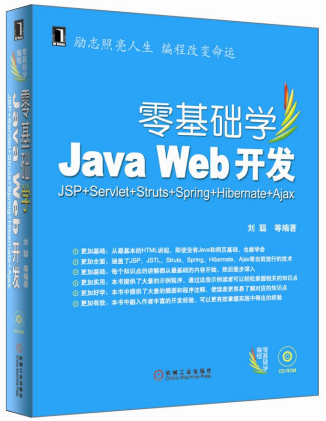
4.3　程序结构

- JSP就是把Java代码嵌套在HTML中，所以JSP程序的结构可以分为两大部分：一部分是静态的HTML代码；另一部分是动态的Java代码和JSP自身的标签和指令；当JSP页面第一次被请求的时候，服务器的JSP编译器会把JSP页面编译成对应的Java代码，根据动态Java代码执行的结果，生成对应的纯HTML的字符串返回给浏览器，这样就可以把动态程序的结果展示给用户。（具体内容请参照书。）



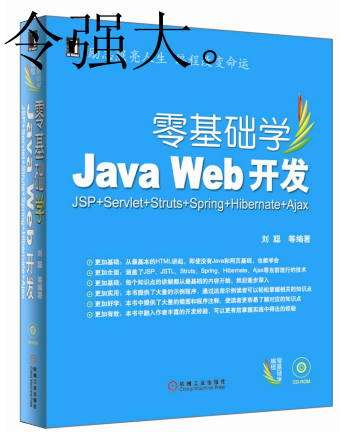
4.4　JSP动作指令

- 在Web程序涉及中经常需要用到JSP的动作指令，例如在使用JavaBean的时候就离不开userBean的指令，JSP的强大功能和它丰富的动作指令标签是分不开的。在接下来的章节中将对这些指令进行详细的介绍，读者可以仔细体会每个动作的示例程序，在示例程序中掌握这些动作指令的基本用法。



4.4.1　include动作指令

- include动作指令可以在JSP页面中动态包含一个文件，这与include指令不同，前者可以动态包含一个文件，文件的内容可以是静态的文件也可以是动态的脚本，而且当包含的动态文件被修改的时候JSP引擎可以动态对其进行编译更新。而include指令仅仅是把一个文件简单的包含在一个JSP页面中，从而组合成一个文件，仅仅是简答的组合的作用。其功能没有include动作指令强大。（具体内容请参照书。）

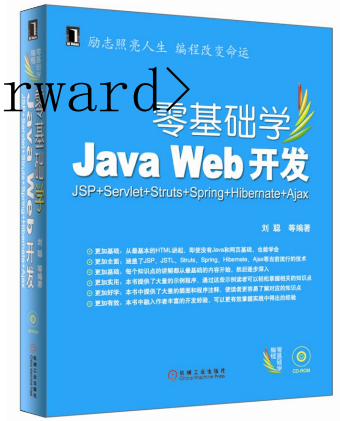


4.4.2　forward 动作指令

- forward动作指令可以用来控制网页的重定向。即从当前页面跳转到另一个页面。

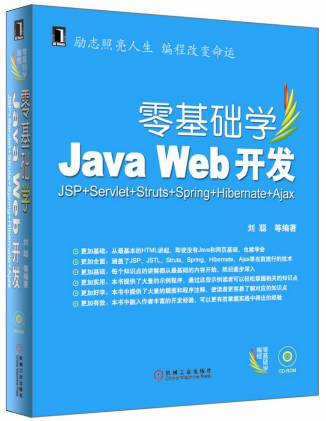
- forward动作的使用方法非常简单，具体使用格式如下。

- <jsp:forward page="http://www.sohu.com"></jsp:forward>



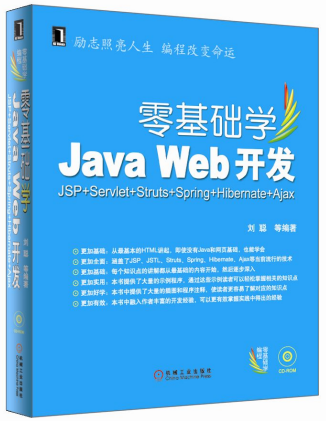
4.4.3　param动作指令

- 在上面forward动作指令中可以用程序控制页面的跳转，如果需要在跳转的时候同时传递参数，这时候就需要用到param动作指令。param动作指令的具体使用方法可以参考下面的示例程序。（具体内容请参照书。）



4.4.4　plugin动作指令

- <jsp:plugin>元素用于在浏览器中播放或显示一个对象（典型的就是applet和bean),而这种显示需要在浏览器的java插件。当jsp文件被编译，送往浏览器时，<jsp:plugin>元素将会根据浏览器的版本替换成<object>或者<embed>元素。

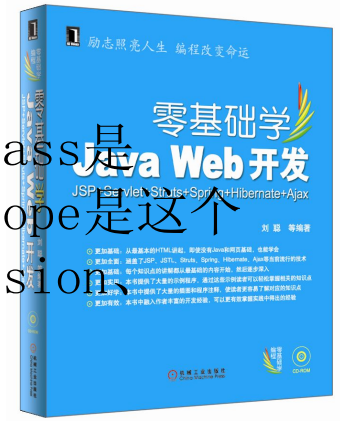


4.4.5　useBean动作指令

- useBean动作指令可以在JSP中引用JavaBean，这个动作指令在实际开发过程中经常会用到。在第六章JavaBean的讲解过程中将对这个动作指令做详细的介绍。在这里我们仅仅知道其基本用法即可，而且在这里不在用示例程序说明。useBean的使用格式如下。

- <jsp:useBean id="" class="" scope="">></jsp:useBean>

- 其中id为所用到的JavaBean的实例对象名称，class是JavaBean对应类的包路径，包括包名和类名。scope是这个JavaBean的有效范围，共有page、request、session、application四个值可以选择。



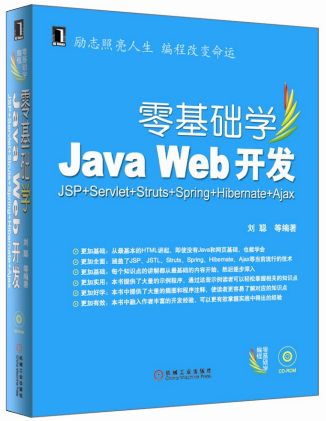
4.4.6　setProperty动作指令

- setProperty一般情况下是和JavaBean配合使用的，用来给JavaBean的实例对象进行赋值操作，setProperty的基本方法有以下两种。<jsp:setProperty name="JavaBean的实例名称" property="属性名" value="属性值"/>

- 上面这种方法是setProperty动作指令最基本的用法，用来给JavaBean实例对象的某一个属性赋值。

- <jsp:setProperty name="JavaBean的实例名称" property="*" />

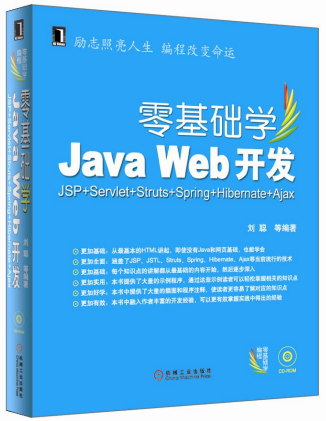
- 上面这种JavaBean的赋值方法也是经常用到的，



4.4.7　getProperty动作指令

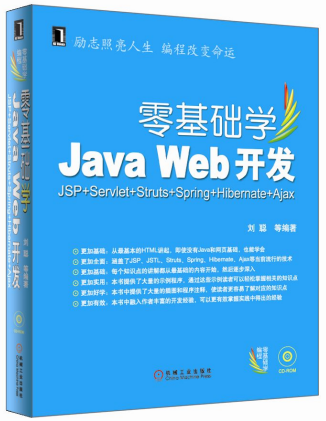
- getProperty一般情况下也是和JavaBean配合使用的，用来取出JavaBean实例对象的属性值。这个动作指令的基本使用方法如下。

- <jsp:getProperty name="JavaBean的实例名称" property="属性名" value="属性值"/>



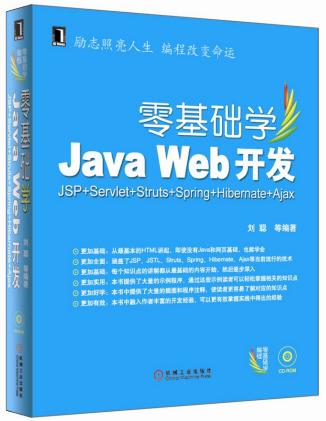
4.5　JSP指令

- JSP的指令虽然没有动作指令那么丰富，但是其作用却是不容忽视的，例如page指令，在设置显示编码、引入类的包路径、设置错误页面等方面都是必不可少的。在接下来的章节中将介绍JSP的两个指令标签。



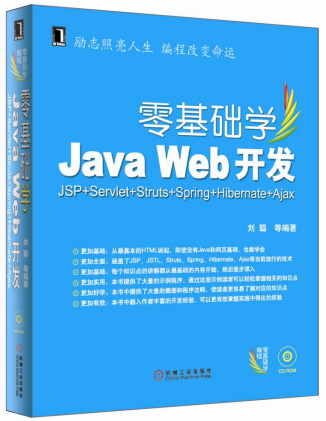
4.5.1　page指令

- page指令可以用来定义JSP也面的全局属性。例如编码、错误页面等。page指令的属性很多，下面来具体介绍它的各个属性。（具体内容请参照书。）



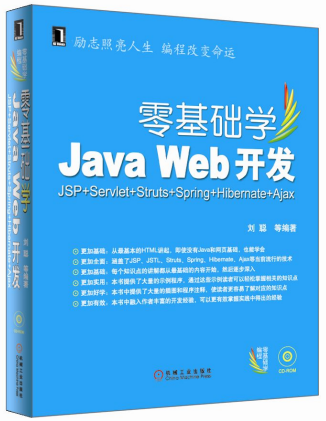
4.5.2　include指令

- include指令可以在当前的JSP页面中包含一个文件，从而和当前页面组成一个整体的文件。这中包含仅仅是静态包含。（具体内容请参照书。）



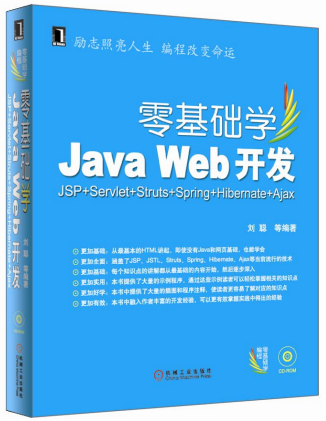
4.6　JSP内置对象简介

- JSP内置对象即无需声名就可以直接使用的对象实例，在实际的开发过程中，比较常用的JSP内置对象有reqest、response、session、out、application等，在接下来的章节中将详细介绍这几个JSP内置对象的使用方法。JSP其他的几个内置对象在实际的开发中并不十分常用，在这里不做具体介绍。



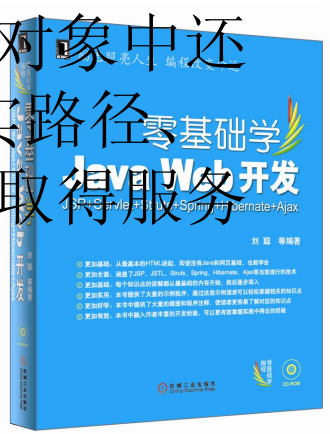
4.7 request对象

- request对象代表这从用户发送过来的请求，从这个对象中间可以取出客户端用户提交的数据或者是参数。这个对象只有接受用户请求的页面才可以访问。



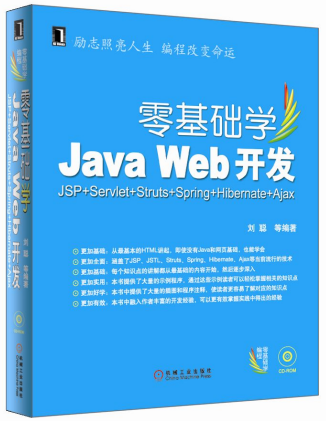
4.7.1　request对象使用场合

- 如果要与用户的互动，必须要知道用户的需求，然后根据这个需求生成用户期望看到的结果。这样才能实现与用户的互动。在Web应用中，用户的需求就抽象成一个request对象，这个对象中间包括用户所有的请求数据，例如通过表单提交的表单数据，或者是通过URL等方式传递的参数，这些就是用户的需求。request正是用来收集类似这些用户的输入数据和参数。同时，request对象中还包括一些服务器的信息，例如端口、真实路径、访问协议等信息，通过request对象可以取得服务器的这些参数。



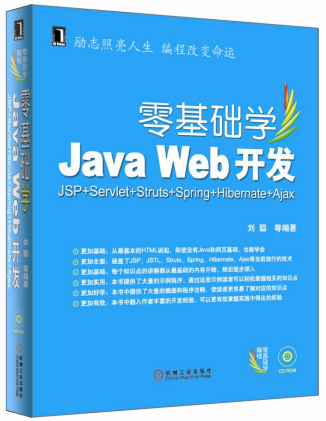
4.7.2 request对象主要方法

- request对象的方法非常多，在这里我们只介绍其中最常用的几种方法，其他方法可以参考相关类库的介绍。（具体内容请参照书。）



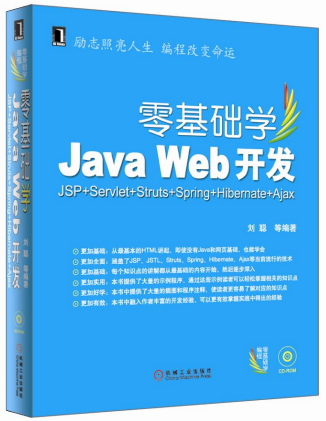
4.7.3　request对象使用示例

- 1．使用request对象取得表单数据
- request获取用户数据的一个主要方式就是获取表单数据，（具体内容请参照书。）



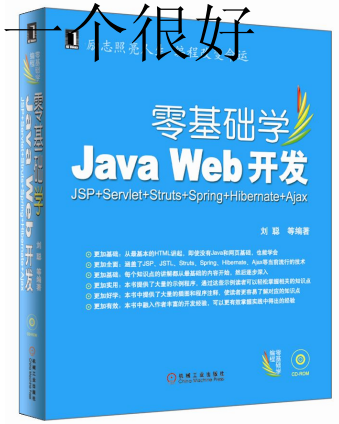
4.8　response对象

- response对象是服务器端向客户端返回的数据，从这个对象中间可以取出一部分与服务器互动的数据和信息。只有接受这个对象的页面才可以访问这个对象。



4.8.1　response对象使用场合

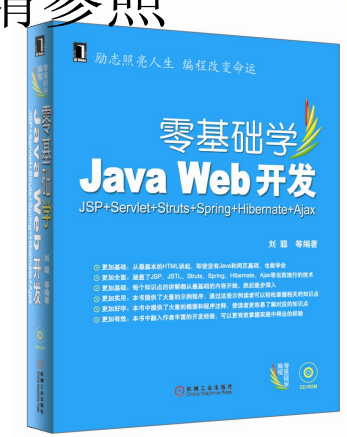
- 既然用户可以对服务器发出请求，服务器就需要对用户的请求做出反应。这里服务器就可以使用response对象向用户发送数据。response是对应request的一个对象。如果需要获取服务器返回的处理信息，就可以对response进行操作，同时当服务器需要再客户端进行某些操作的时候也需要用到response对象，例如服务器要在客户端生成Cookies，那么这时候response对象就是一个很好的选择。



4.8.2　response对象主要方法

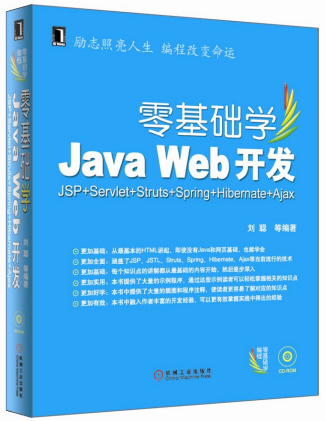
- response的方法也很多，但是常用的也就其中的几个，下面介绍比较常用的几个方法。
  1．addCookie（Cookie cookie）这个方法可以添加一个Cookie对象，用来保存客户端的用户信息。

- 2．containsHeader（String name）这个方法判断指定的头信息是否存在。（具体内容请参照书。）



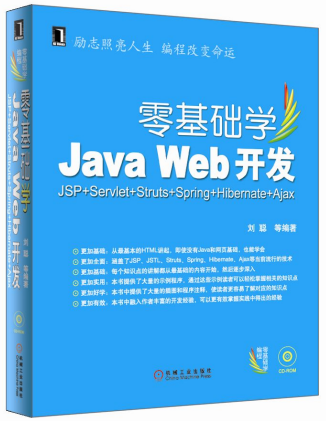
4.8.3　response对象使用示例

- response的用法很多，在这里我们用response来实现一个页面的重定向，



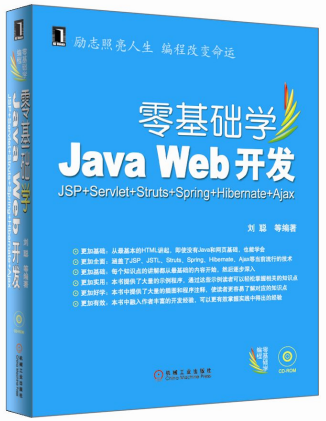
4.9 session对象

- session对象维护着客户端用户和服务器端的状态，从这个对象中间可以取出用户和服务器交互的过程中的数据和信息。这个对象在用户关闭浏览器离开Web应用之前一直有效。



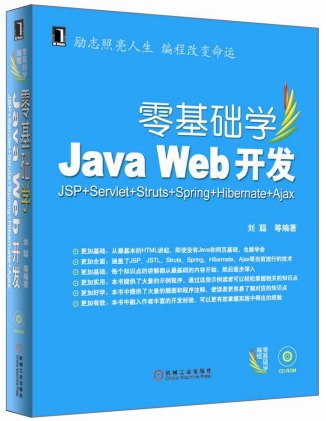
4.9.1　session对象使用场合

- session对象中保存的内容是用户与服务器整个交互过程中的信息，如果是想在整个交互的过程中都可以访问到的信息，就可以选择存放在session对象中。例如在用户登录的过程中，可以在session中记录用户的登录状态，这样用户就不必在每个页面都重新登录，只要用户没有离开当前的Web应用系统，就可以一直保存登录的状态。



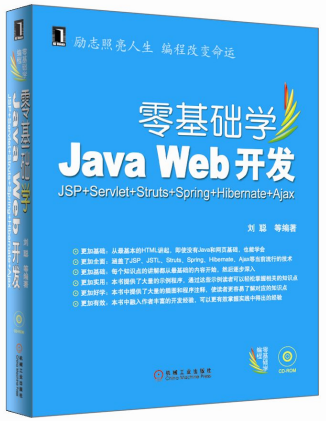
4.9.2　session对象主要方法

- session所提供的方法并没有前面几个内置对象那么多，但是基本都是非常常用的。



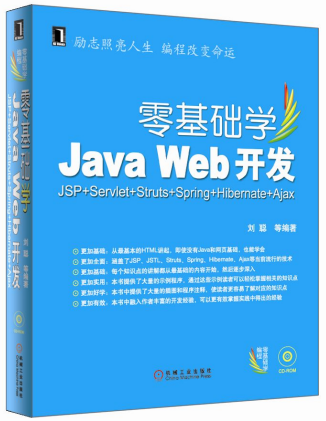
4.9.3　session对象使用示例

- 在这里我们模拟一个简单的用户登录动作，在这个示例程序中，我们不对提交的登录信息做具体的验证，只要用户名和密码都不为空就可以登录系统，这样处理只是为了方便展示session的使用方法，在具体的开发中必须要对登录信息进行验证的。（具体内容请参照书。）



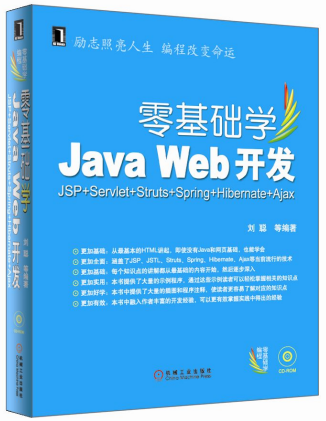
4.10　out对象

- 这个对象是在Web应用开发过程中使用最多的一个对象，其功能就是动态的向JSP页面输出字符流，从而把动态的内容转化成HTML形式来展示。这个对象在任何JSP页面中都可以任意访问。



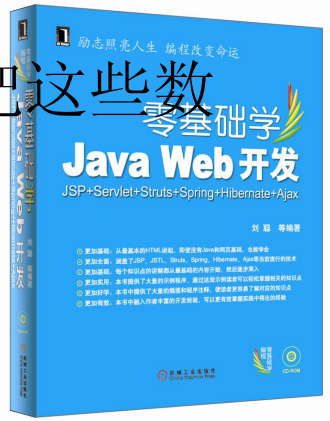
4.10.1　out对象使用场合

- out对象的功能就是向JSP页面输出数据信息。所以当有动态信息要展示给用户的时候就要用到out对象。在前面的很多示例中已经多次用到这个对象，读者从中可以很清楚的看到，out对象就是用来输入动态内容信息的。



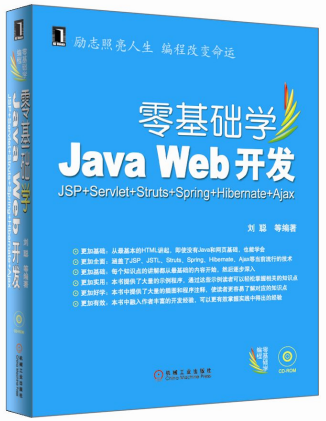
4.10.2　out对象主要方法

- 在这里只介绍out对象最常用的方法。
    - 1. clear（）
- 这个方法可以清除缓冲区的数据，但是仅仅是清除，并不向用户输出。
    - 2. clearBuffer（）
- 这个方法可以清除缓冲区的数据，同时把这些数据向用户输出。（具体内容请参照书。）



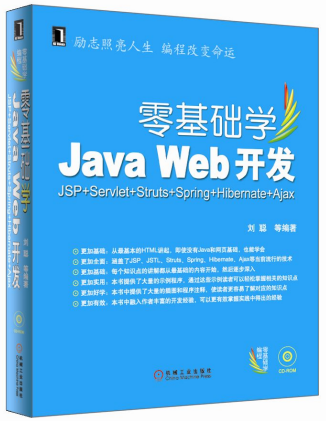
4.10.3　out对象使用示例

- out对象在前面的示例中已经多次使用到，在这里就不再针对这个对象举例说明。



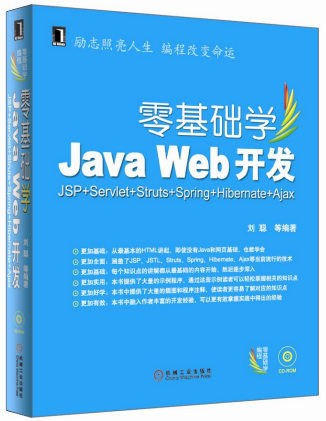
4.11 application对象

- application对象保存着整个Web应用运行期间的全局数据和信息，从Web应用开始运行开始，这个对象就会被创建，在整个Web应用运行期间可以在任何JSP页面中访问这个对象。



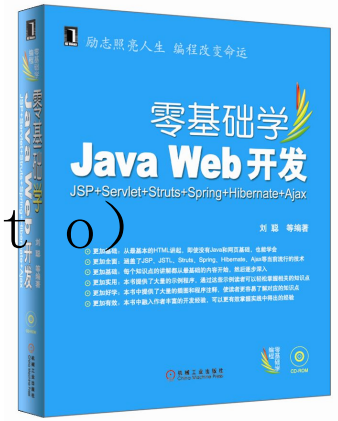
4.11.1 application对象使用场合

- application中保存的信息可以在整个应用的任何地方访问，这个session对象类似，但和session对象还是有所区别的。只要Web应用还在正常运行，application对象就可以访问，而session对象在用户离开系统就被注销。



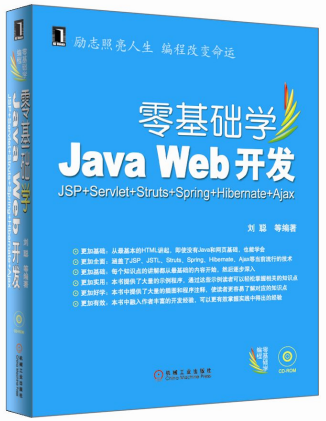
4.11.2 application对象主要方法

- 下面介绍application对象的最常用的主要方法。
- 1. getAttribute（String name）
- 2. getServerInfo（）
- 3. removeAttribute（String name）
- 4. setAttribute（String name，Object o）



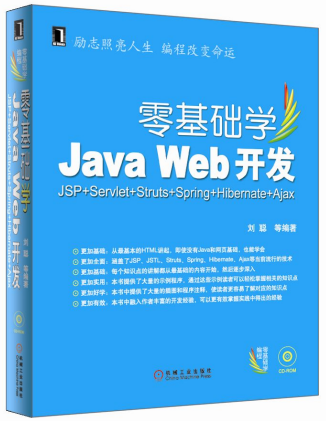
4.11.3 application对象使用示例

- 在这里我们要实现一个简单的计数器，这个计数器就是利用application对象来储存计数器的值，用来统计服务器开始运行以来的访问量。



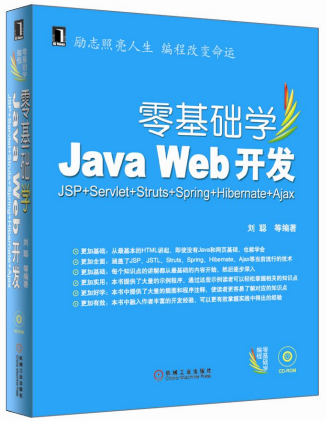
4.12　JSP中文问题完全解决方案

- 对于中文乱码问题’Java中才用的默认编码方式是Unicode，而中文的编码方式一般情况是GB2312,因为编码格式的不同，导致在中文不能正常显示。在不同的JDK版本和不同的应用服务器中的处理方法是不同的。但是其本质上都是一样的，就是把中文字符转化成合适的编码方式，或者是把在显示中文的环境中声名采用GB2312的编码。统一编码方案之后自然可以正常显示。



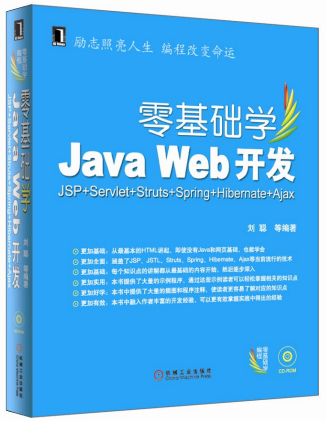
4.12.1　JSP页面中文乱码

- 在JSP页面中，中文显示乱码有两种情况：一种是HTML中的中文乱码，另一中是在JSP中动态输出的中文乱码。（具体内容请参照书。）



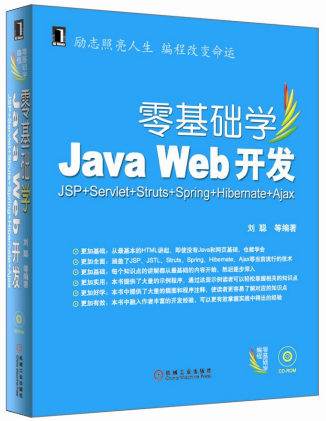
4.12.2　URL传递参数中文乱码

- 在一般情况下，可以用类似http://localhost:8080/chapt4/URLCharset.jsp?param=’中文’这种形式来传递参数，而且HTML在处理表单的时候，当表单的method采用get方法的时候，传递参数的形式与URL传递参数的形式基本一样。（具体内容请参照书。）



4.12.3　表单提交中文乱码

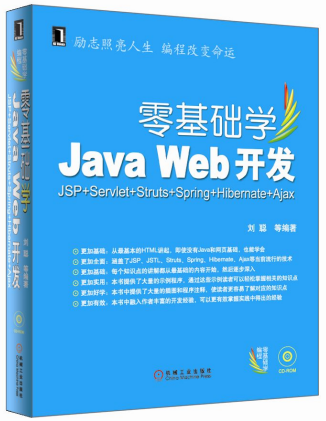
- 对于表单中提交的数据，可以使用request.getParameter("")的方法获取。但是当表单中如果出现中文数据的时候就会出现乱码。（具体内容请参照书。）



4.12.4　数据库操作中文乱码

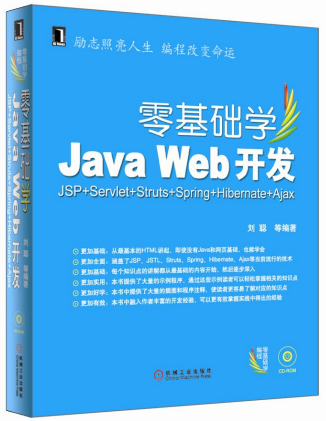
- 在建立数据库的时候，应该选择支持中文的编码格式，最好能和JSP页面的编码格式保持一致，这样就可以尽可能减少数据库操作的中文乱码问题。同时在JDBC连接数据库的时候可以使用类似下面这种形式的URL。

  jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=pubs;useUnicode=true;characterEncoding=gb2312



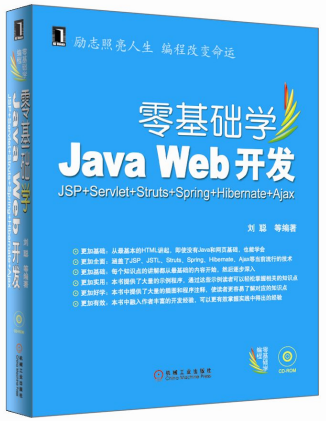
4.12.5　Eclipse开发工具中JSP文件中文不能保存

- 在Eclipse中，JSP文件默认的编码格式为ISO-8859-1，所以在JSP代码中间如果出现中文就不能保存，



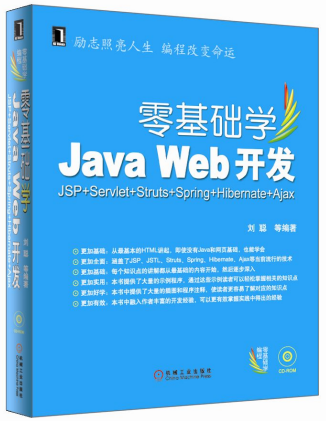
4.12.6　Eclipse开发工具中中文显示乱码

- 在Eclipse中，由于默认的JSP编码格式为ISO-8859-1，所以当打开由其他编辑器编辑的JSP页面就会出现乱码，（具体内容请参照书。）



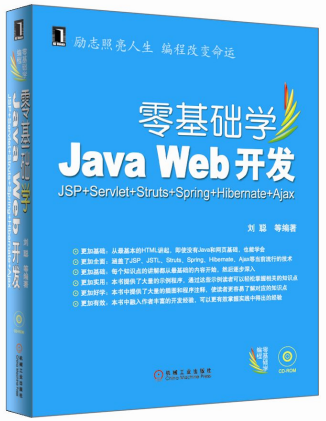
4.12.7　JSP下载中文文件名乱码

- 在实现文件下载功能的时候，如果出现中文文件名，如果不进行特殊的处理，下载下来的中文文件名会变成乱码，在下载前，就需要对这个文件名进行处理，然后才能正常显示中文的文件名，



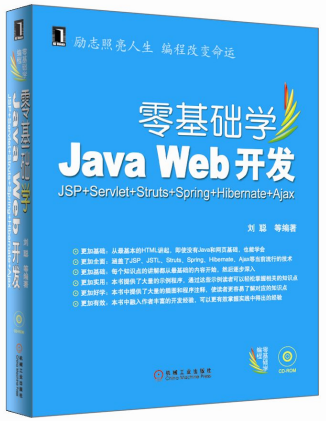
4.13　其他JSP开发技巧

- （具体内容请参照书。）



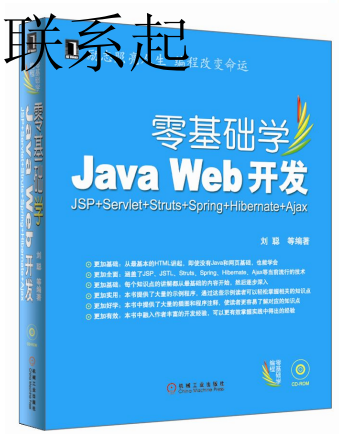
4.13.1　自定义错误页面

- 在JSP中，如果出现代码的错误，就会直接在页面上打印类似



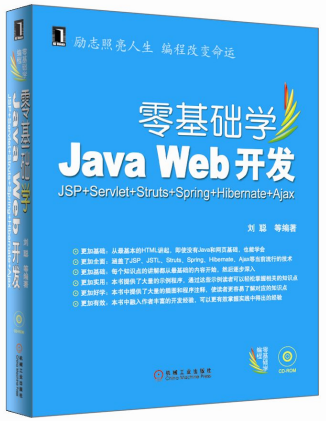
4.13.2　在MyEclipse中快速部署项目

- 在Web应用开发的过程中，部署项目往往十分麻烦，虽然在后续章节中介绍的Ant可以非常方便的完成这个任务，但是Ant复杂的操作不适合初学者，在这里我们使用前面推荐的MyEclipse这个集成开发工具来部署项目。MyEclipse的安装在前面第二章中已经详细介绍，在这里直接开始介绍如何发布Web应用项目。要想发布部署一个项目，首要任务就是把MyEclipse和服务器Tomcat联系起来，



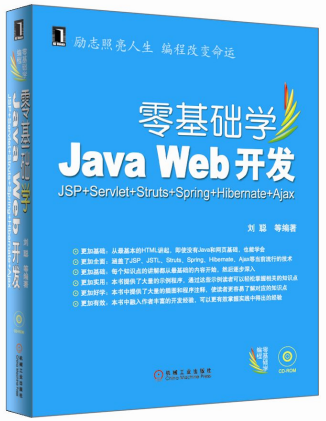
4.13.3　测试配置是否成功

- 下面来测试下配置是否成功（具体内容请参照书。）



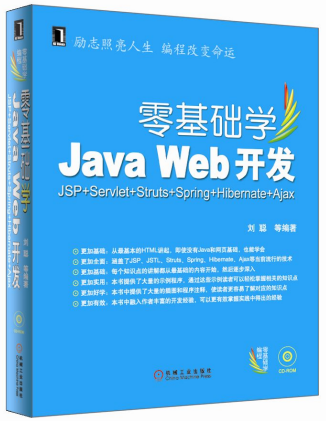
4.13.3　在MyEclipse中调试Web应用程序

- 在MyEclipse中，对JSP页面进行调试也是非常方便的，如果需要调试JSP页面，只需要在JSP页面源代码的左侧双击鼠标左键，（具体内容请参照书。）



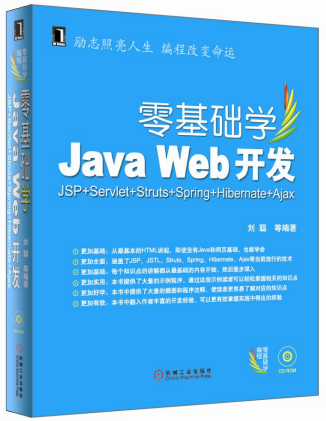
4.13.4　学习使用日志Log4j

- 在JSP Web开发中，有很多方便的日志工具可供选择，利用这些日志工具可以很方便的对系统中的错误信息进行管理，在这里我们选择使用Log4j，Log4j是目前JSP开发中使用最多的日志工具。Log4j按照严重程度给日志风味5个等级：DEBUG（调试）、INFO（提示）、WARN（警告）、ERROR（错误）、FATAL（严重错误）



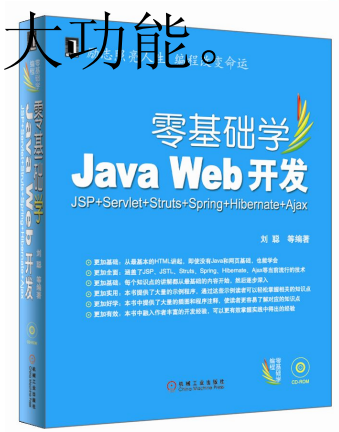
4.14 小结

- 在本章中，对JSP的基本语法和对象等知识进行了系统的介绍，而且对于其中大部分的知识点都给出了具体示例，这些示例在具体的开发过程中都有很大的参考价值，读者可以在这些示例程序的基础上进行尝试，试着修改其中的功能，只有这样才肯定能对其运行原理有更深入的了解和体会，这就是学习程序语言的最基本最有效的方法。



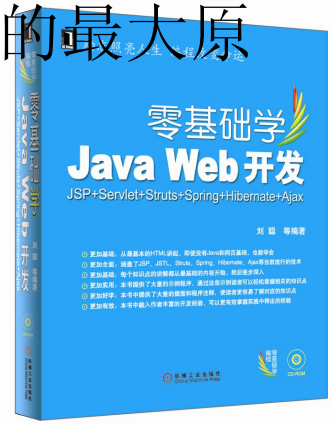
第五章　Servlet技术基础知识

- Servlet是一种服务器端的编程语言，是J2EE中比较关键的组成部分，Servlet技术的推出，扩展了Java语言在服务器端开发的功能，巩固了Java语言在服务器端开发中的地位，而且现在使用非常广泛的JSP技术也是基于Servlet的原理，JSP+JavaBeans+Servlet成为实现MVC模式的一种有效的选择。在本章中将介绍Servlet的基础知识，并通过具体的示例介绍Servlet的强大功能。



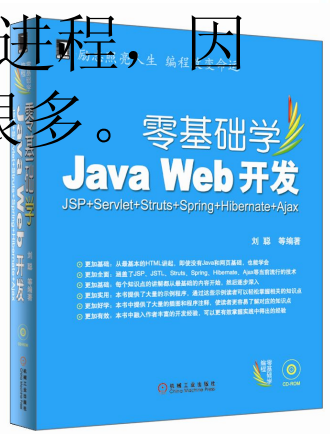
5.1　Servlet简介

- Servlet在本质上就是Java类，编写Servlet需要遵循Java的基本语法，但是与一般Java类所不同的是，Servlet是只能运行在服务器端的Java类，而且必需遵循特殊的规范，在运行的过程中有自己的生命周期，这些特性都是Servlet所独有的。另外Servlet是和HTTP协议是紧密联系的，所以使用Servlet几乎可以处理HTTP协议各个方面的内容，这也正是Servlet收到开发人员青睐的最大原因。



5.1.1　Servlet的工作原理

- Servlet容器环境在HTTP通信和web服务器平台之间实现了一个抽象层。Servlet容器负责把请求传递给Servlet，并把结果返回结客户。在使用Servlet的过程中，并发访问的问题由Servlet容器处理，当多个用户请求同一个Servlet的时候，Servlet容器负责为每个用户启动一个线程，这些线程的运行和销毁由Servlet容器负责，而在传统的CGI程序中，是为每一个用户启动一个进程，因此Servlet的运行效率就要比CGI的高出很多。



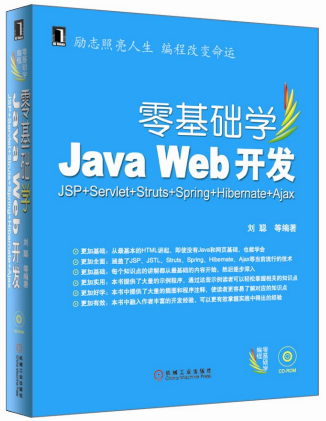
5.1.2　Servlet的生命周期

- Servlet是运行在服务器端的程序，所以Servlet的运行状态完全由Servlet容器维护，一个Servlet的生命周期一般有三个过程。

- 1．初始化

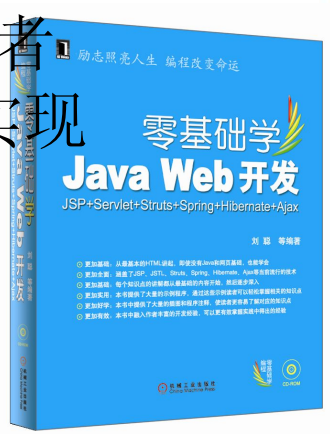
- 2．提供服务

- 3．销毁



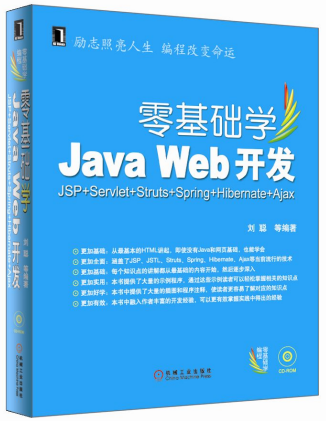
5.1.3　简单Servlet开发配置示例

- Java Servlet API包括两个基本的包，javax.servlet 和javax.servlet.http，其中javax.servlet提供了用来控制Servlet生命周期所需的类和接口，是编写Servlet必需要实现的。javax.servlet.http提供了处理与HTTP相关操作的类和接口，每个Servlet必需实现Servlet接口，但是在实际的开发中，一般情况都是通过继承 javax.servlet.http.HttpServlet或者javax.servlet.GenericServlet来间接实现Servlet接口。



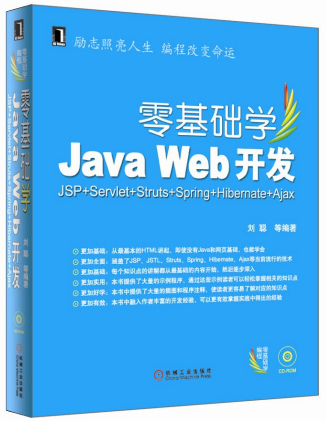
5.1.4　使用Servlet实现MVC开发模式

- Java语言之所以受到开发人员支持，是因为Java语言实现科学方便的开发模式，在这些开发模式中，最出色而且应用最广的就是MVC模式，对于MVC模式的研究由来已久，但是一直没有得到很好的推广和应用，随着J2EE技术的成熟，MVC逐渐成为了一种常用而且重要的设计模式。MVC（Model-View-Controller）把应用程序的开发分为三个层面：视图层、控制层、模型层。



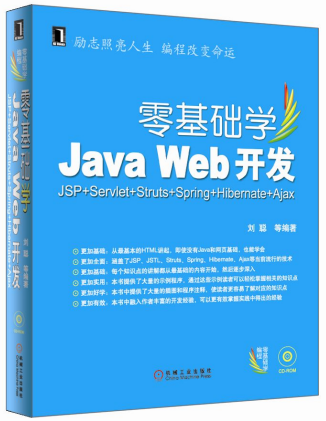
5.2　JSP页面调用Servlet的方法

- 在上面HelloWorld的示例程序中，我们直接在浏览器中输入具体的地址进行访问，在实际的应用中，不可能让用户在浏览器中直接输入Servlet的地址进行访问，一般情况下，可以通过调用Servlet进行访问，在这里介绍通过提交表单和超链接两种方式调用Servlet。



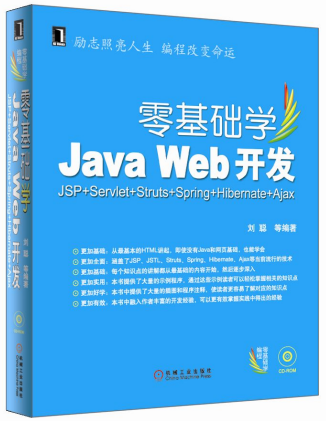
5.2.1　通过表单提交调用Servlet

- 在通过提交表单调用Servlet的时候，只需要把表单的action指向对应的Servlet即可，下面是一个简单的表单，通过这个表单可以调用指定的Servlet。（具体内容请参照书。）



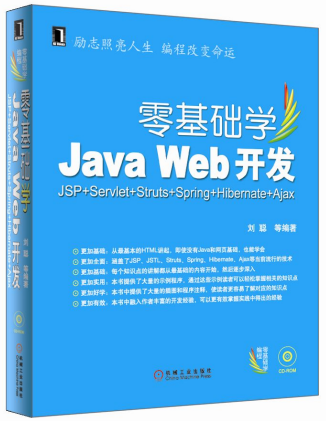
5.2.2　通过超链接调用Servlet

- 在上面这个例子中，用户有输入的内容需要提交给服务器，所以需要用表单来调用Servlet，但是在没有输入的数据内容需要提交的情况下，使用表单就不是很合理了，在这里介绍Servlet的第二种调用方法，直接通过超链接的方式来调用Servlet，在这种情况下还可以给Servlet传递参数。（具体内容请参照书。）



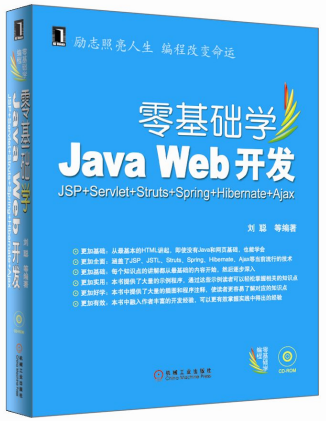
5.3 Servlet中的文件操作

- 在JSP的开发过程中，经常会遇到需要把相关内容存储为文件的情况，在JSP中是用输入输出流进行操作的，在Servlet中也可以使用输入输出流实现对文件的读写，同时，使用Servlet还可以很方便的实现文件的上传下载。接下来的内容将通过具体的示例展示Servlet文件操作的方法。



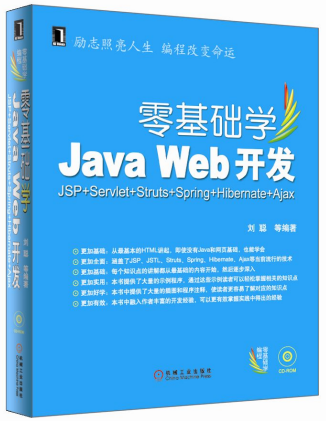
5.3.1　Servlet读取文件

- 在这个例子中将要读取一个文本文件的内容，并且在页面上打印文件的内容。



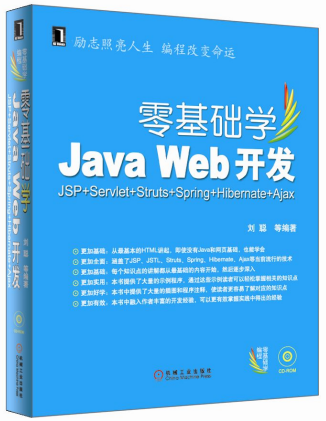
5.3.2 Servlet写文件

- Servlet写文件的处理方法和读取文件的处理方法非常类似，只是把文件输入流换成文件输出流，在下面这个示例程序中，将在指定位置生成文件。（具体内容请参照书。）



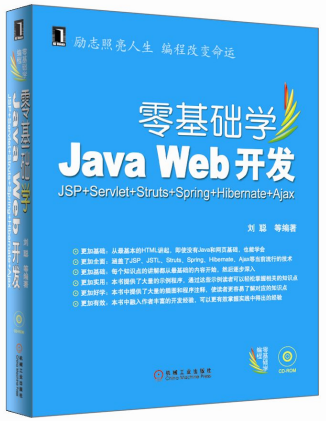
5.3.3 Servlet上传文件

- 文件的上传下载在Web开发中会经常遇到，使用基本的IO输入输出流当然可以完成这项操作，但是出于对开发的效率和程序运行的效率方面的考虑，在实际的开发过程中一般采用第三方的组件来完成这个上传的功能。在实际开发过程中用的比较多的是commons-fileupload组件和jspSmartUpload组件，这两个组件都可以很好地完成文件上传的功能，



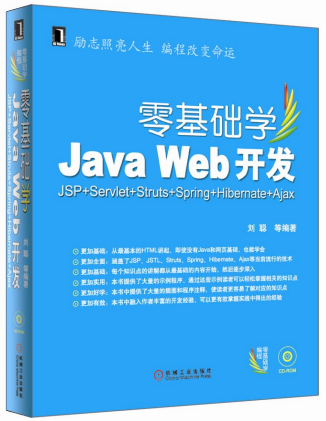
5.3.4　Servlet下载文件

- 用Servlet下载文件的时候，并不需要第三方组件的帮助，只需要对服务器的响应对象response进行简单的设置即可，（具体内容请参照书。）



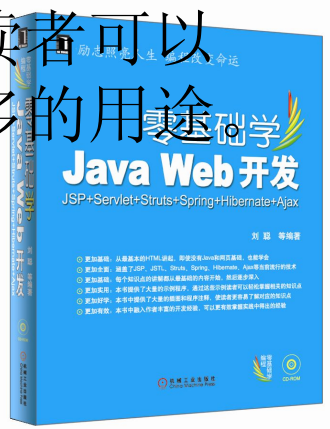
5.4　Servlet过滤器

- 在Web应用中可以使用过滤器对所有的访问和请求进行统一的处理，IP访问限制，用户发送请求的字符编码转换等，在进行具体的业务逻辑处理之前，首先要经过过滤器的统一处理，然后才开始进入真正的逻辑处理阶段。在本节内容中，将介绍过滤器的原理的实际应用。



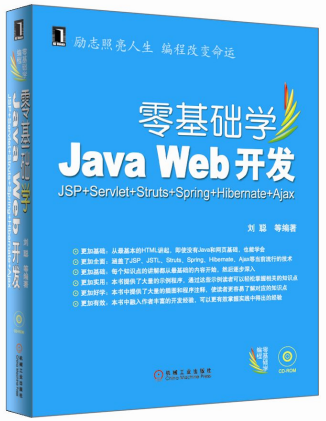
5.4.1　过滤器的基本原理

- 过滤器的功能就是在服务器和客户中间增加了一个中间层，可以对两者之间的交互进行统一的处理，每一个从客户端提交的请求都需要通过过滤器的处理，然后再进行其他的操作。在实际开发中，过滤器器可以用来对用户进行统一的身份判断、IP访问限制，用户发送请求的字符编码转换、对请求和响应进行加密和解密、记录用户登录日志等。当然过滤器的用途不仅仅这些，读者可以根据过滤器的实现原理，思考过滤器更多的用途。



5.4.2　IP访问filter

- 在实际的应用中，可能会遇到这样的情况，需要对某些IP进行访问限制，不让非法的IP访问应用系统，这个时候就需要用到过滤器进行限制，当一个用户发出访问请求的时候，首先通过过滤器进行判断，如果用户的IP地址被限制，就禁止访问，只有合法的IP才可以继续访问。（具体内容请参照书。）

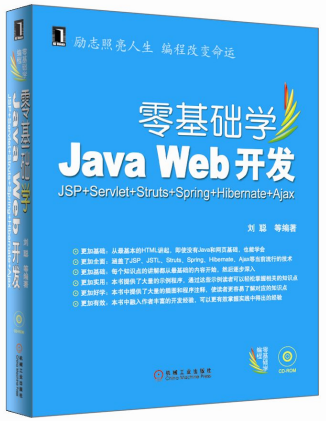


5.4.3　转换字符编码filter

- 在Java语言中，默认的编码方式是ISO-8859-1，这种编码格式不支持中文的显示，我们可以用类似<%@ page contentType="text/html;charset=gb2312"%>这样的方式来规定页面字符编码格式，但是如果要显示的内容是表单提交、或者是经过Servlet处理，这时候字符内容本身的编码格式就是ISO-8859-1，所以尽管页面指定的字符编码方案为gb2312，在这种情况下中文内容仍然不能正常显示。在第四章中已经对中文处理的问题做了详细的介绍，所以在本章仅仅对其中使用过滤器解决中文乱码问题进行详细的分析。

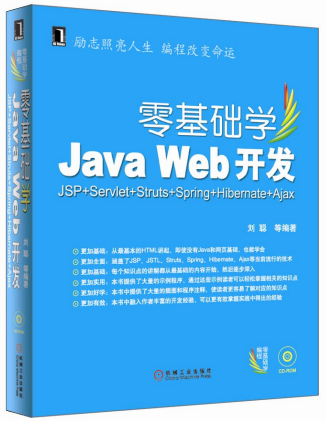
5.5 Servlet应用示例

- 在上面的内容中已经提到，Servlet是与HTTP协议紧密结合的，使用Servlet几乎可以处理HTTP协议各个方面的内容，在本节的几个示例程序中，将集中展示Servlet在HTTP方面的具体应用。



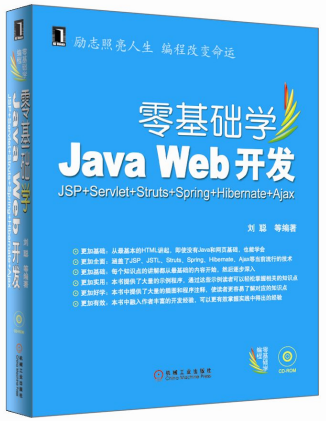
5.5.1　获取请求信息头部内容

- 当客户访问一个页面的时候，会提交一个HTTP请求给服务器的Servlet引擎，在这个请求中有HTTP的文件头信息，其中包含这个请求的详细属性信息，在下面这个示例Servlet中将取出HTTP头部内容，并在页面打印，这个Servlet的具体代码如下。（具体内容请参照书。）



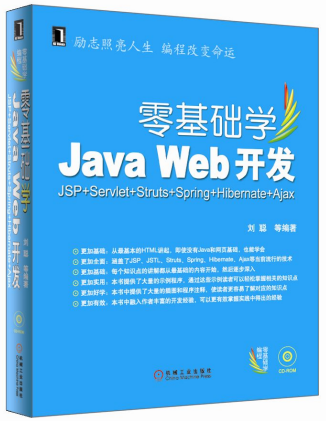
5.5.2　获取请求信息

- 在上面这个Servlet示例中，我们取出所有的HTTP文件头信息，在Servlet中还可以很方便取出客户发出请求对象自身的信息。这些信息是和客户的请求密切相关的，例如客户提交请求所使用的协议，客户提交表单的方法是POST还是GET等，在下面这个示例程序中将介绍集中常见属性的取值方法。这个示例程序的具体代码如下。（具体内容请参照书。）



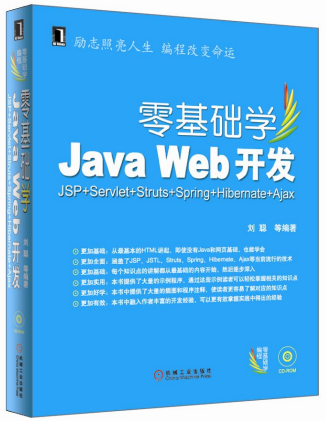
5.5.3　获取参数信息

- 在Servlet中，同样可以很方便的取出用户请求中的参数信息，这种参数包括以POST方法或者是GET方法提交的表单，也包括直接使用超链接传递的参数，Servlet都可以取出这些信息并且加以处理，在下面的例子中将具体展示Servlet获取各种参数的方法。



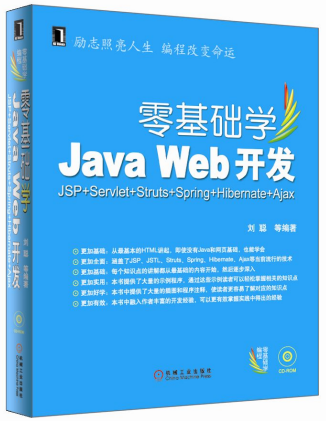
5.5.4　Cookies操作

- Cookies是指在Web应用中，为了辨别用户身份而存储在用户本地计算机上的数据。Servlet API提供了Cookie操作类，封装了操作Cookie常用的方法，在（具体内容请参照书。）



5.5.5 Session操作

- 在JSP中有内置的Session对象，可以用来保持服务器与用户之间的会话状态，在Servlet中间，同样可以对Session进行方便的操作，在现面的例子中，将详细介绍Servlet中处理Session的具体方法。

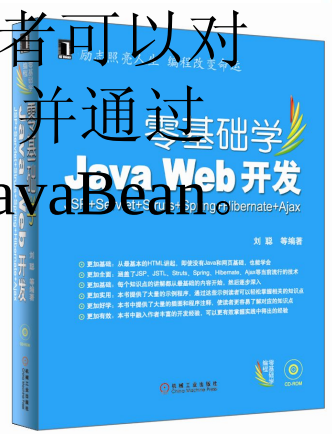


5.6 小结

- 在本章的内容中，详细讲解了Servlet的工作原理，并且通过实际的示例程序详细介绍了Servlet的调用方法，对Servlet常见的文件操作也做了比较详细的介绍，Servlet是和HTTP协议密切联系的，所以在本章最后的部分对Servlet的HTTP操作方法做了细致的讲解。通过本章内容的讲解，读者已经可以对Servlet有一个总体上的把握，Servlet在本质上就是Java类，在了解这Servlet的基本原理和基本使用方法以后，如果要想在Servlet领域有更大的提高，还是需要回头巩固Java的基础，这才是学习Servlet的最根本的途径。

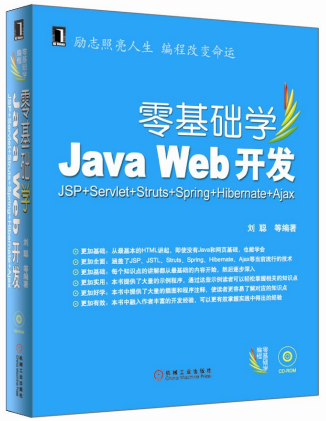
第六章　JavaBean技术基础知识

- JavaBean是Java中的一个组件技术，类似于微软的COM组件，其本质上是一个封装了一系列属性和方法的类。该类遵循一定的标准，提供公共的方法，只要遵循同样的标准，用户就可以调用封装在JavaBean里面已经设计好的方法，从而达到代码重复利用的目的。本章从JavaBean的基本概念开始介绍，通过具体的例子重点讲解JavaBean在JSP中的使用方法，通过本章的学习读者可以对JavaBean的概念体系有一个整体的把握，并通过示例程序的学习学会在实际开发中使用JavaBean。



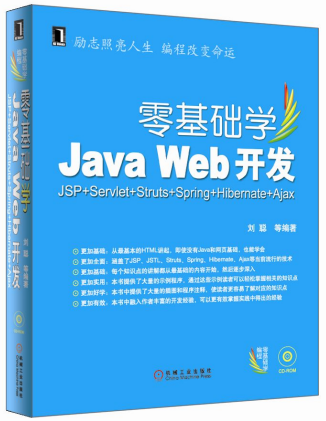
6.1　JavaBean简介

- Sun公司对JavaBean的定义为：可以重复利用的软件组件，它在遵循JavaBean技术规范的基础上提供特定的功能，这些功能模块可以组合成更大规模的应用系统。



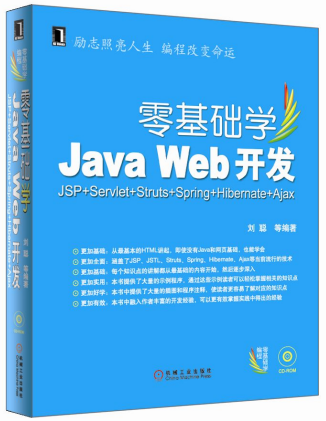
6.1.1　什么是JavaBean

- 在开始认识JavaBean之前先浏览以下的程序代码，这段程序的功能就是求出a,b的和并且在控制台输出。



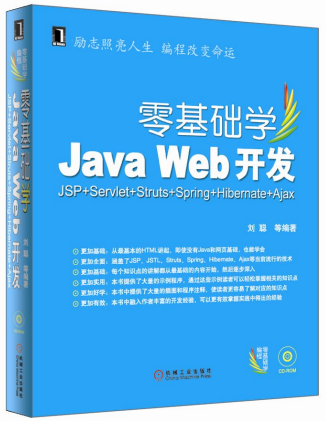
6.1.2　JavaBean的种类

- 在章节6.1.1中通过一个简单的示例认识了什么是JavaBean，下面进一步了解JavaBean的分类。JavaBean大体可以分为两类，第一类是可视化JavaBean，第二类是非可视化JavaBean。



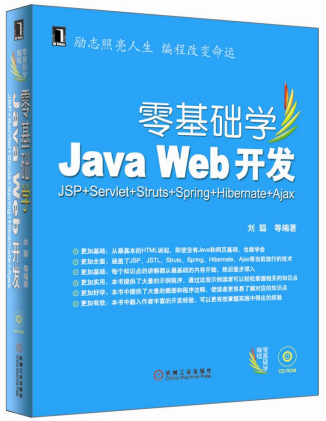
6.1.3　JavaBean的编码规则

- 在前面的章节中一再强调JavaBean要遵循特定的规范标准，这个规范标准是JavaBean区别于普通的Java类的一个标志，而这个标准在JavaBean的体现就是特定的编码规则，下面还是通过简单的示例代码来了解这个规范标准。



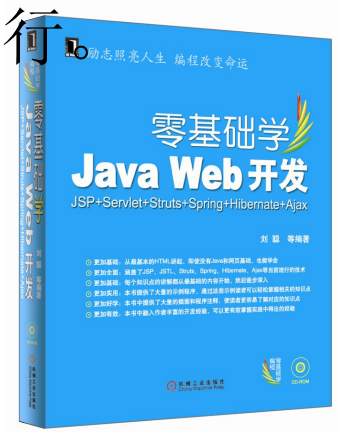
6.1.4　JavaBean典型示例

- 在上面的章节中讲述了JavaBean的概念、分类和编码规则，下面来将给出一个比较综合的JavaBean的示例程序，用来全面展示JavaBean特性和编码规则。请浏览下面的程序代码。



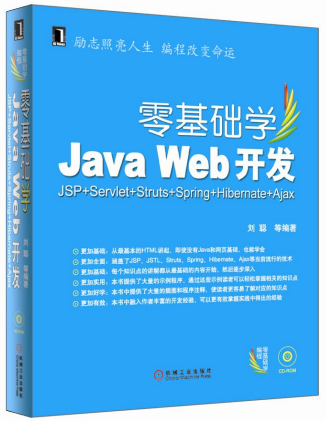
6.2　在JSP中使用JavaBean

- JSP＋JavaBean的组合已经成为开发小型B/S应用的最佳选择，使用JavaBean可以把业务处理功能从JSP页面分离，从而减少JSP页面中间的Java代码量，使JSP页面专注处理数据的显示，从而使页面的逻辑变得十分清晰，自从采用JSP＋JavaBean的组合，编写和维护JSP的程序已经不在是一件令人头疼的事情，JavaBean的出现给JSP的开发带来了质的变化，JSP的开发从此变得简单可行。



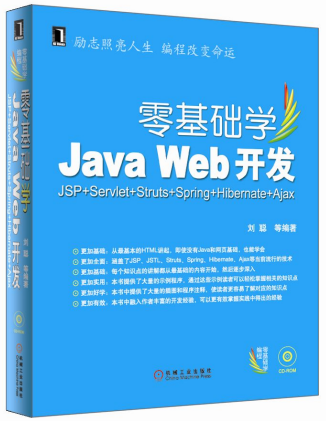
6.2.1　为什么要在JSP中使用JavaBean

- JSP本质就是把Java代码嵌套在静态的HTML页面中间，从而是静态的HTML页面有了动态的功能，从原理上说，仅仅用JSP就可以实现所有动态功能，既然这样采用JavaBean的原因何在，（具体内容请参照书。）



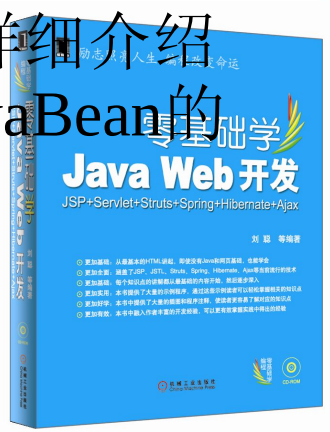
6.2.2　JSP中使用JavaBean的具体方法

- 在本章节将实现一个最简单的HelloWorld的JavaBean，并且在JSP页面中调用这个JavaBean，展示的重点在如何从JSP中调用JavaBean。



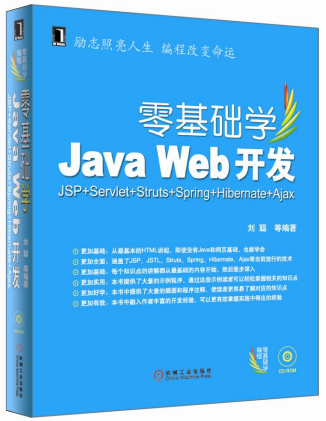
6.2.3　JavaBean的编译部署

- 编译JavaBean就是编译Java类文件，这和编译一般的Java文件没有什么区别，在本书的第二章有详细的介绍，此处不再赘述。所谓的部署就是把编译好的JavaBean的类文件放在合适的位置，以便在JSP中可以访问这些类文件。而JavaBean类文件的组织方法有两种，一种是单独的class文件，另一种是把多个class文件打包成一个jar文件。而这两种方法的部署方法是不同的，下面详细介绍JavaBean的部署方法，重点是在JSP中JavaBean的部署。



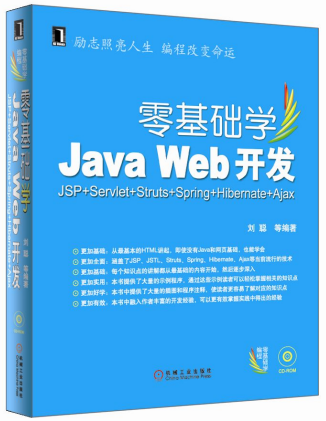
6.3　计数器JavaBean

- 对于一个Web应用来说，计数器的功能几乎是必不可少的。在接下来的章节将要介绍的就是实现一个简单的计数器。这个计数器使用JavaBean来实现。



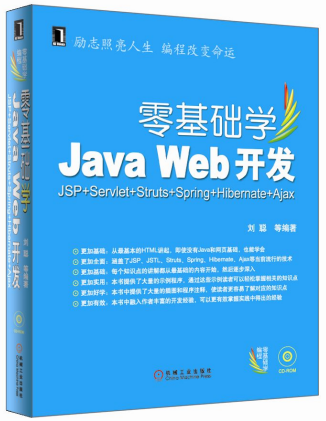
6.3.1　计数器JavaBean背景知识

- 为了统计访问顾客的数量，很多网站都会提供计数器的功能。计数器的实现思路有很多，可以把访问的数量记录在数据库库中，也可以把访问的数量记录放在一个文本文件中，当然也可一把访问的数量记录设置成为一个有效范围为application的变量，这样在整个应用运行期间这个变量都是有效的。



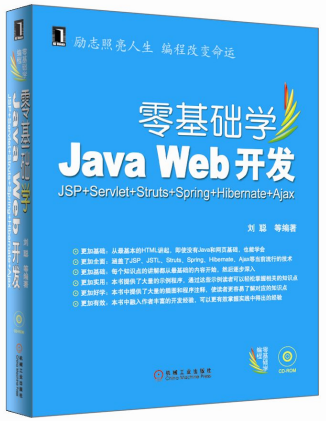
6.3.2　计数器JavaBean的具体实现

- 本章的重点讲解内容是如何在实际的应用场景中使用JavaBean，为了不让数据库操作或者是文本操作干扰读者的视线，在这里选用在application中存储计数器的值。（具体内容请参照书。）



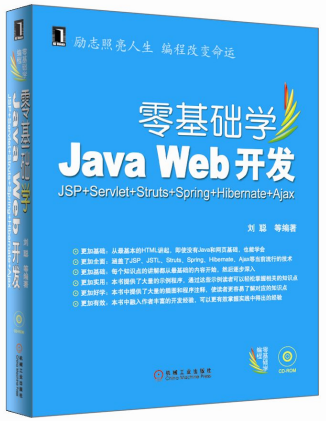
6.3.3　计数器JavaBean的调用方法示例

- 上面这个JavaBean的功能是定义一个计数器变量，并且给出这个变量的取值和赋值的方法，按照上面JavaBean的部署方法把这个文件放在适当的位置就可以在JSP页面中调用，（具体内容请参照书。）



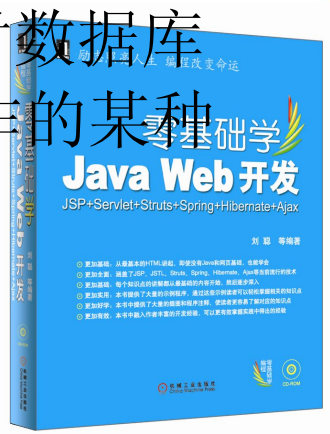
6.4　数据库操作封装的JavaBean

- JavaBean同样可以使用到数据库开发中，从而简化开发过程，提高代码的可重用性。接下来的将要介绍的内容就是利用JavaBean封装数据库操作。



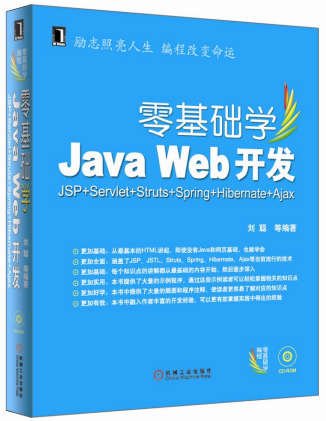
6.4.1　数据库操作封装的JavaBean的背景知识

- 在大量的Web应用中，数据库操作占据着相当大的比重。初学者经常性的习惯在每一个需要数据库操作的地方重复进行下列动作——“加载驱动类，提供连接字符串、用户名和密码，利用DriverManager取得连接，在此连接基础上执行数据库操……”，针对上面提出的问题依然可以采用JavaBean来解决，我们可以把数据库操作的方法都封装在一个JavaBean中，这样如果要对数据库的用户名密码做改动或者是对数据库操作的某种方法进行修改的话都会变的非常容易。



6.4.2　数据库操作封装JavaBean具体实现过程

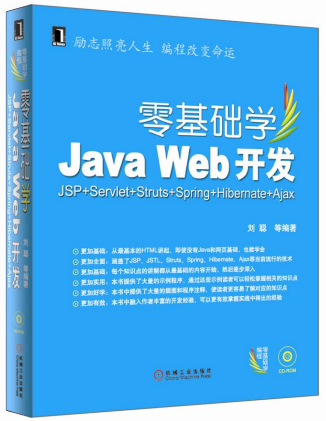
- （具体内容请参照书。）



6.4.3　数据库操作封装JavaBean关键代码解析

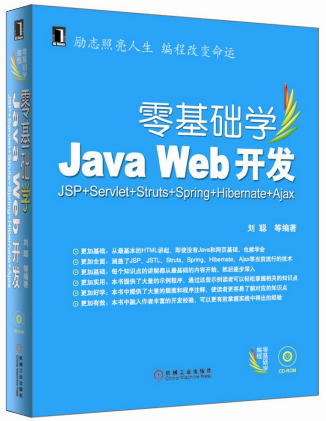
- 在使用上面这个JavaBean之前现了解下中间关键代码的实现过程，至于数据库连接的详细内容在接下来的第七章将有详细的讲解，在这里仅仅做简单的介绍，读者只需知道怎么使用这个JavaBean即可，详细的数据库操作理论可以在接下来的章节详细学习。请看下面的代码片段。

```
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver").newInstance();
```



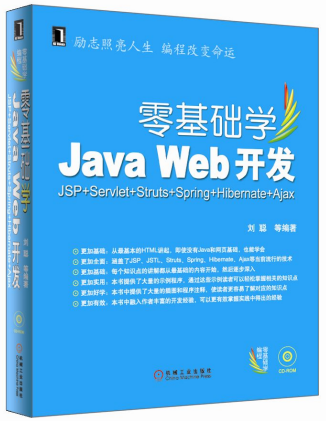
6.4.4　数据库操作封装JavaBean调用方法示例

- 上面这个JavaBean中的使用方法都是类似的，此处仅仅以数据库查询方法为例进行讲解。（具体内容请参照书。）



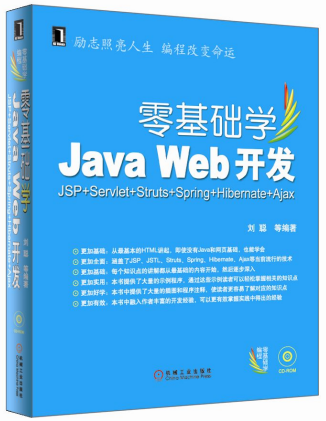
6.5　对应数据库表结构的实体JavaBean

- 把数据库表抽象成对应的Java类，这样就可以在数据库操作中引入面向对象的操作理念。接下来的章节将要讲述的就是这个理论的一种实现方法。



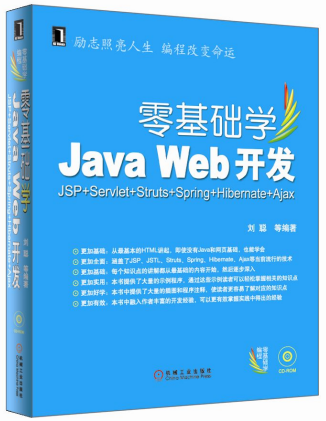
6.5.1　实体JavaBean的背景知识

- 如何在数据库操作中引入面向对象的操作方法，现在这个问题已经有了比较好的解决方案，那就是ORM（Object / Relational Mapping），在本书第十九章将要讲述的Hibernate就是基于这种理论，在本章我们将采用JavaBean来简单的实现ORM理论。



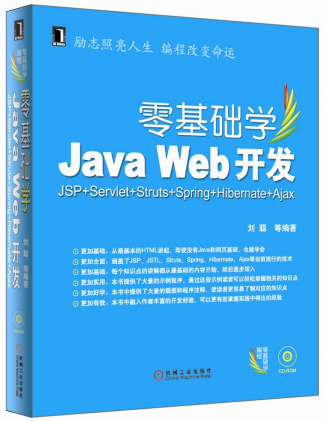
6.5.2　对应数据库表结构实体JavaBean的实现过程

- 下面将讲述怎样利用JavaBean实现ORM，在这里我们仍然使用SqlServer 2000自带的数据库pubs，数据库表选用pubs中的jobs表，（具体内容请参照书。）



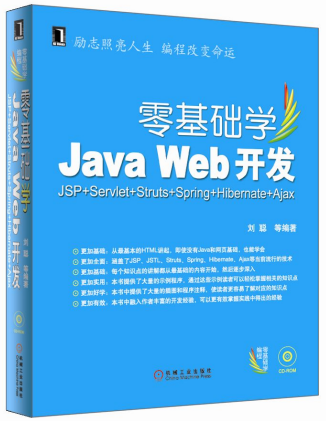
6.5.3　采用实体JavaBean以后对数据库封装方法的改造

- 上面这个JavaBean就可以清楚的描述jobs表的结构，接下要解决的问题是怎么在把数据库表和这个JavaBean类映射起来，使对JavaBean类对象的操作能够被持久化到数据库中，在Hibernate中采用的是单独的XML映射文件来实现。在这里仅仅是简单的实现ORM，所以就省略了这个文件，把数据库的操作直接放在数据库操作JavaBean中，（具体内容请参照书。）



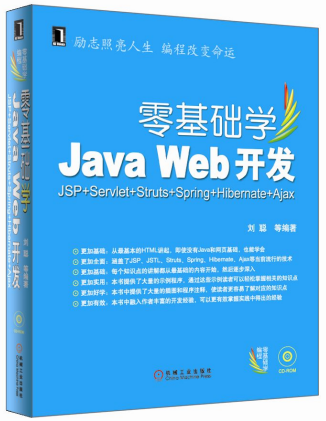
6.5.4　实体JavaBean及对应数据库操作方法的调用示例

- 经过这样的处理以后对jobs表的处理就可以直接对Jobs的对象执行操作，此处仅仅展示使用这个JavaBean进行数据库查询的处理方法，其他各种数据库操作的方法可以参考这个示例，这个改进后的数据库操作JavaBean的具体调用方法请参考下面的代码，（具体内容请参照书。）



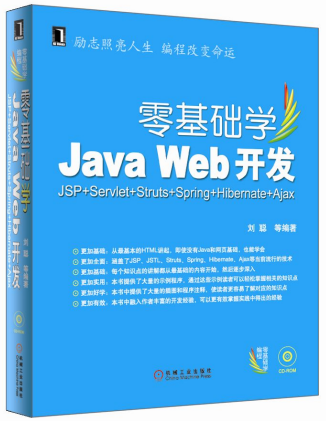
6.6　分页操作JavaBean

- 分页操作在Web应用中也是必不可少的，记下来要介绍的就是如何对在上面例子的基础上添加分页的效果。



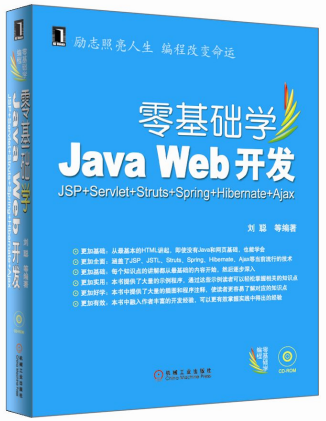
6.6.1　分页操作的背景知识

- 在上一个章节的数据库查询示例中可以看到所有的记录都显示在一个页面中，在数据量比较小的情况下这样处理并没有什么不妥之处，但是当数据量比较大的时候，例如上千甚至上万条记录，这时候不仅页面浏览不方便，而且每次打开也页面都要一次性的查处所有记录并显示，这样页面访问的速度就可想而知了。



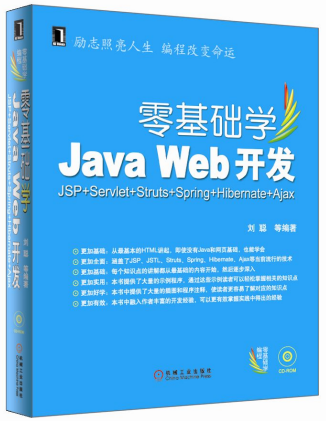
6.6.2　分页操作JavaBean具体实现

- 下面就是分页显示JavaBean的代码。（具体内容请参照书。）



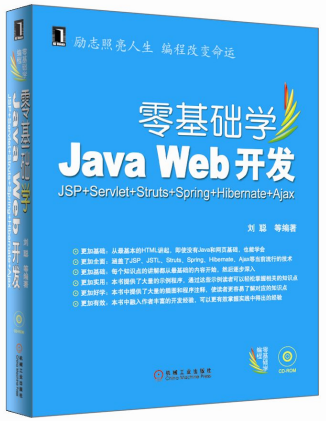
6.6.3　分页操作JavaBean调用方法解析

- 分页显示的原理其实非常简单，即根据当前记录位置和每页要显示的条数，在结果集中取出对应的子集，同时在计算出页码等相关参数即可。具体在JSP中分页显示的处理可以参考下面的代码，这里是在前面章节中数据库查询页面的基础上添加上分页显示的功能。（具体内容请参照书。）



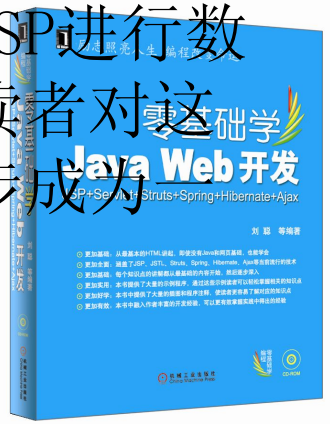
6.7　小结

- JavaBean就是把一些属性和方法封装起来的Java类，它遵循一定的编码规则，把可以重复利用的功能代码封装起来在其他地方或者是提供给其他开发人员调用，同时利用JavaBean还可以简化JSP的页面逻辑，降低了JSP程序维护修改的难度。



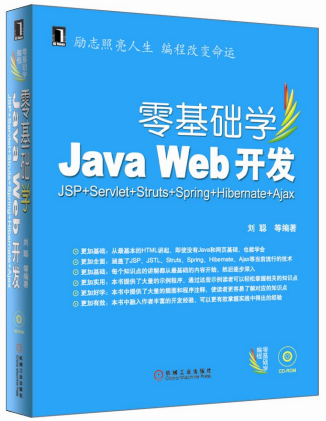
第七章 JSP数据库开发基础知识

- 在Java语言中提供了方便的数据库操作API，通过这些API可以非常方便的操作各种数据库，而且对于不同的数据库来说，只有取得数据库连接部分的操作稍有不同，其他部分基本都是相同的，同样的数据库操作代码可以非常方便的使用到另一个数据库中。在本章的内容中，首先搭建起JSP开发数据库的环境，然后对数据库操作的SQL语言进行简单的介绍，本章的核心部分是使用JSP进行数据库开发的介绍，通过具体的示例，使读者对这项技术有一个整体的认识，从初学者逐步成为一个JSP数据库开发的高手。



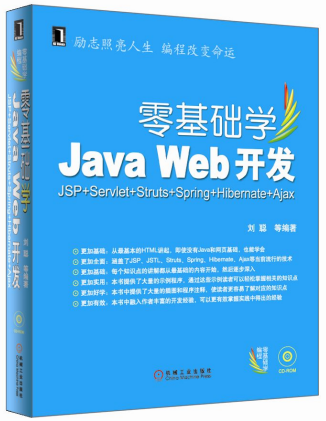
7.1　JSP数据库开发环境搭建

- 工欲善其事，必先利其器。要进行数据库的开发，第一步要做的工作就是安装数据库软件，目前常用的数据库有MySQL、SQL，Oracle、DB2等，在本章我们以MySQL和SQLServer2000的安装过程为例，简单介绍JSP数据库开发环境的搭建过程。



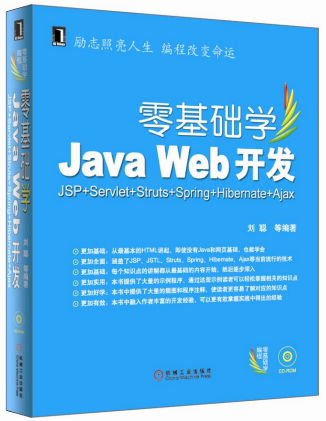
7.1.1　MySQL数据库的安装

- MySQL的安装程序可以从http://www.mysql.com网站上下载，目前的版本是5.0.41，下载下来的安装文件为mysql-essential-5.0.41-win32.msi。直接运行这个文件就可以进行安装，MySQL提供了图形界面的安装向导，根据安装向导的提示，可以非常方便的安装MySQL。下面简单介绍MySQL的安装步骤。（具体内容请参照书。）



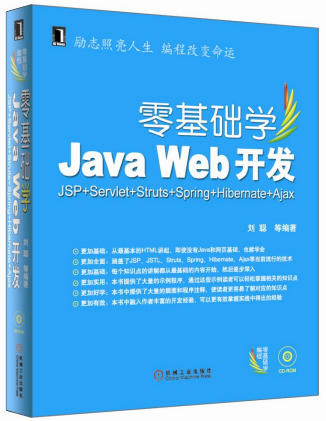
7.1.2　SQL Server2000数据库安装

- SQL Server2000的安装过程更加简便，在安装向导的提示可以非常方便进行安装，在这里不在对SQL Server的安装过程进行详细的介绍。



7.2　SQL基础

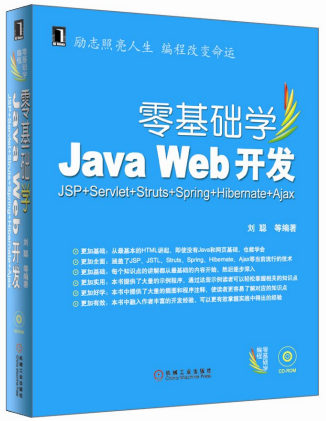
- 在与数据库交互的过程中，可以使用各种语言，例如Java、C#甚至是C语言，但是所有这些语言并不是直接操作数据库，而是通过向数据库传递SQL语句来实践数据的操作，SQL语言才是直接处理数据库的语言，在目前常用的关系数据库中，都支持SQL语言的操作。



7.2.1　创建和删除数据库

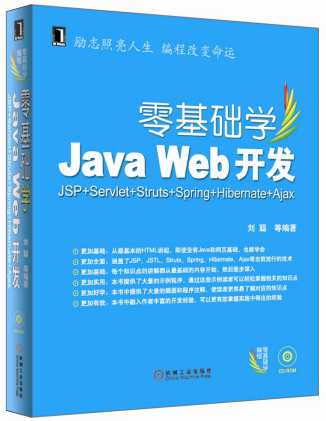
- 创建数据库的SQL语句如下。

- CREATE DATABASE　数据库名称（具体内容请参照书。）



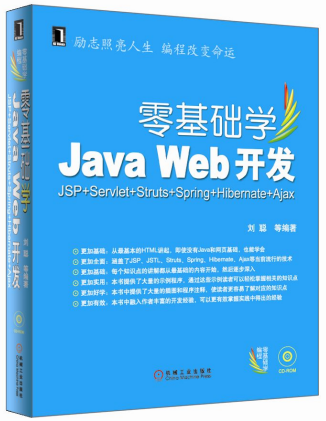
7.2.2　创建、修改、删除数据库表

- 创建数据库表可以是用CREATE TABLE指令，这个指令的使用语法如下。CREATE TABLE 表名（　列名1　数据类型　[DEFAULT（默认值）] [NOT NULL] [UNIQUE],　列名2　数据类型 [DEFAULT（默认值）] [NOT NULL] [UNIQUE]　… … … … …　[PRIMARY KEY（列名）]　[FOREIGN KEY （列名） REFERENCES （表名）])



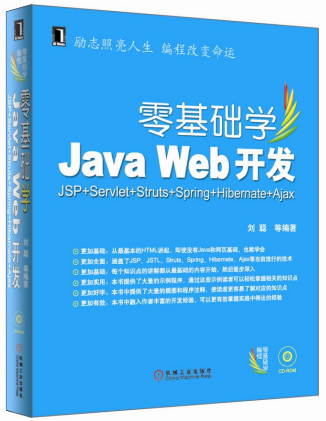
7.2.3　查询语句

- 在数据库操作中，查询数据的方式有很多，包括基本的查询、选择性查询、限制条件查询、统计查询、多表联合查询等查询操作，在接下来的内容中将逐步介绍这些基本的查询操作。（具体内容请参照书。）



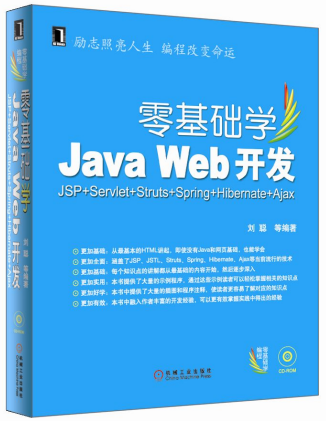
7.2.4　插入、更新、删除语句

- 在本部分的内容中，将介绍表格中数据的基本操作，包括数据的插入、更新和删除操作。（具体内容请参照书。）



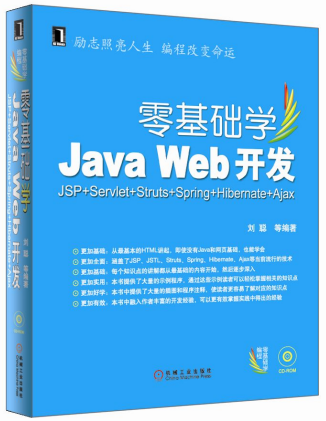
7.2.5　简单视图操作

- 视图是虚拟的表，其功能和基本的数据库表类似，但是在视图中并不存储数据，视图仅仅是把实际表格中的数据按照条件提取出来，以表格的形式展示出来，从而方便用户的查询，其实在这个虚拟的表格中并没有任何物理上存储的数据，其中的数据都是从其他表格中提出出来的。（具体内容请参照书。）



7.2.6　简单存储过程操作

- 存储过程（Stored Procedure）是一段可以完成特定功能的SQL 语句，它经过编译后存储在数据库中，用户通过名称调用指定的存储过程，如果这个存储过程中带有参数，在调用的时候需要提供对应的参数输入。（具体内容请参照书。）

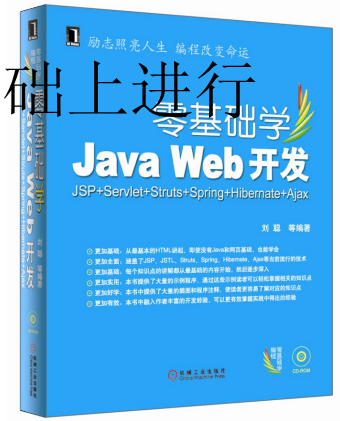


7.3　JSP与数据库建立连接

- 一般来说，使用JSP操作数据库都需要下面三个步骤。（1）根据提供的驱动程序名加载对应的数据库驱动程序。

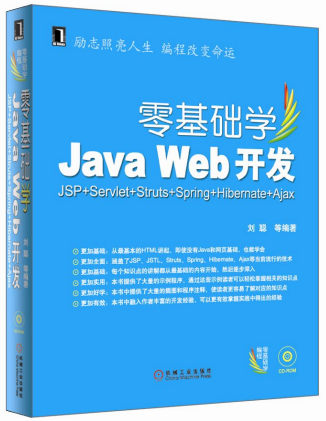
- （2）根据连接字符串，从DriverManager中取的与数据库的连接。

- （3）在取得的Connection数据库对象基础上进行各种数据操作。



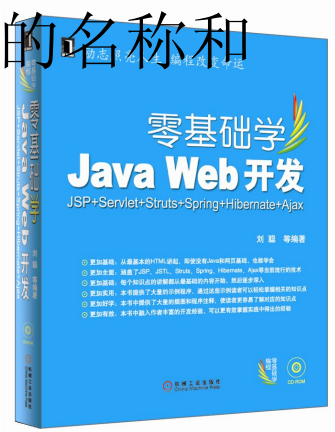
7.3.1　JSP连接SQL Server2000数据库

- JSP连接SQL Server的时候，需要提供SQL Server的数据库驱动，在这里可以有两种选择，一种是微软官方提供的数据库驱动，另一中是开源的JTDS驱动，在这里我们选择微软广方提供的JDBC数据库驱动程序。（具体内容请参照书。）



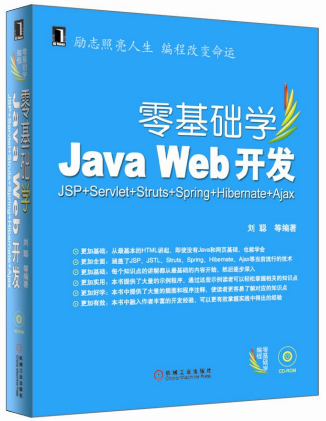
7.3.2　JSP连接MySQL数据库

- JSP连接MySQL数据库的时候，同样也需要数据库驱动，　MySQL的驱动程序名为MySQL Connector/J，读者可以在http://dev.mysql.com/downloads/上找到各个版本的驱动程序。这个驱动无需安装，直接解压下载下来的文件，把　　拷贝到应用项目的/WEB-INF/lib目录中，就可以在JSP中同通过这个JDBC驱动连接MySQL数据库，这个数据库驱动的名称和连接字符串如表7.2所示。



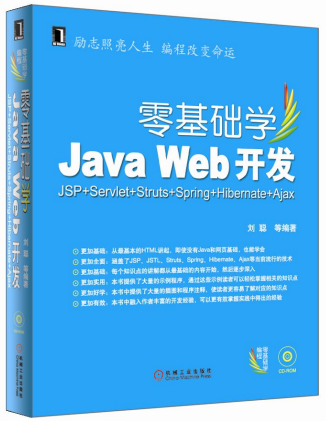
7.3.3　JSP连接Oracle数据库

- Oracle为不同版本的数据库提供了不同的JDBC驱动程序，可以在Oracle的官方网站上下面这个链接下载对应版本的驱动程序。（具体内容请参照书。）



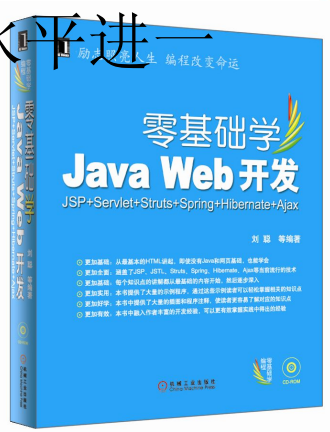
7.3.4　JSP连接数据库简单示例

- 上面介绍了JSP连接常见的几种数据库开发环境的基本配置，（具体内容请参照书。）



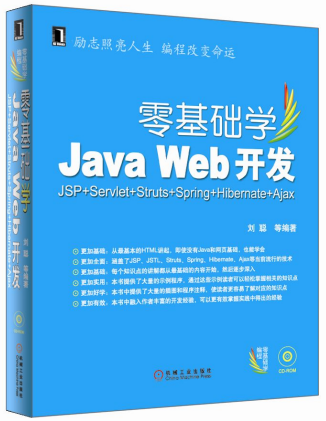
7.4　JSP操作数据库

- 在以上的内容中，介绍了JSP数据库开发的准备知识，在接下来的内容中，将使用这些基础知识进行数据库的操作，操作的数据库选择在前面章节中已经创建的STUDENTS数据库。在本节的内容中，首先介绍数据库最简单最基本的增删改查操作，在了解这些基本知识的基础上引入数据库事务操作的概念，最后介绍两中具有实用价值的技术。视图和存储过程，这是数据库操作水平进一步提高所必须掌握的。



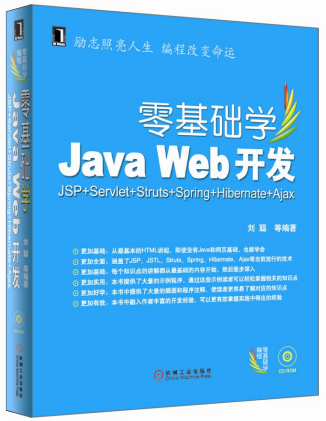
7.4.1　JSP插入数据操作

- 在执行数据的插入操作时，首先要建立与数据库的连接，然后把SQL语句传递到数据库中，通过SQL语句来操作数据库中的数据。在下面的示例程序中，将向student表中插入一条记录，具体代码如下。（具体内容请参照书。）



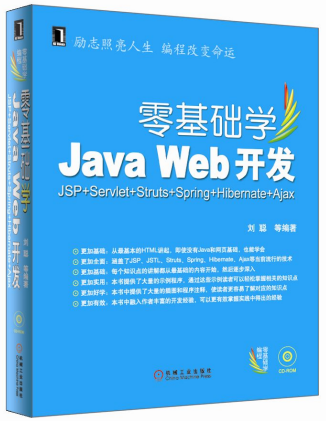
7.4.2　JSP删除数据操作

- 在JSP中，删除数据的操作和插入数据的操作很相似，不同之处只是执行的SQL语句不同，在下面的示例程序中，将从student表中删除满足条件的记录，（具体内容请参照书。）



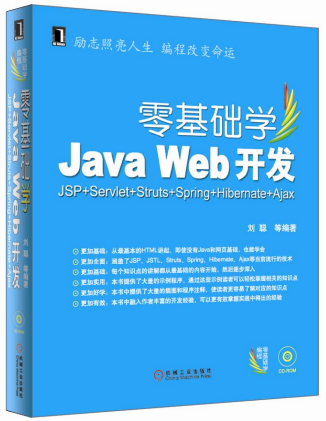
7.4.3　JSP更新数据库操作

- 在数据库的更新操作中，并没有新的知识点，数据库的连接和操作的方法和前面数据库插入删除操作的方法是一样的，不同之处就是在这里执行的是更新数据库的SQL语句，在下面的示例程序中，将更新数据库中符合条件的数据记录，（具体内容请参照书。）



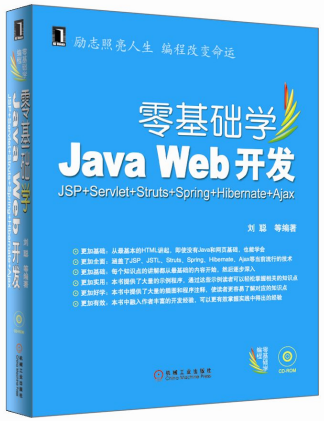
7.4.4　JSP查询数据库操作

- 在数据库的查询操作中，可以通过数据库连接执行SQL查询语句，返回的结果是ResultSet结果集对象，可以在程序中取出并展示结果集中的数据，在下面的示例程序中，将从student表中取出满足条件的数据，并以表格的形式展示取出的结果，（具体内容请参照书。）



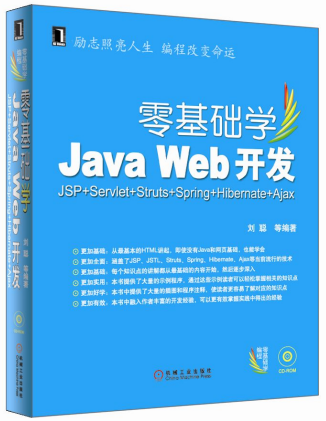
7.4.5　JSP中的数据库操作事务处理

- 在数据库操作中，事务处理是经常用到的，例如在银行的业务中，甲方给乙方账户转账10万人民币，首先要从甲方的账户减去10万，然后再给乙方的账户增加10万，整个操作过程是一个整体，这就是一个简单的事务，在这个事务中必须保证操作的完整性，两步操作要么全执行，如果其中一步出错全都不执行，从而保证这个业务的正确性和完整性。（具体内容请参照书。）



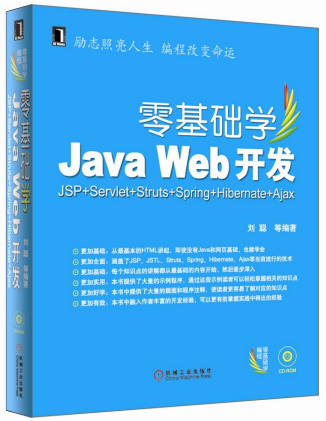
7.4.6　JSP查询视图

- 视图是一种虚拟表，所以视图的查询和表格的查询方法一样，无非是在这里把表格名称换成视图的名称，　在SQL语法介绍中，我们创建了一个名为V_STUDENTS的视图，在下面这个示例程序中将与对这个视图进行简单的查询，（具体内容请参照书。）



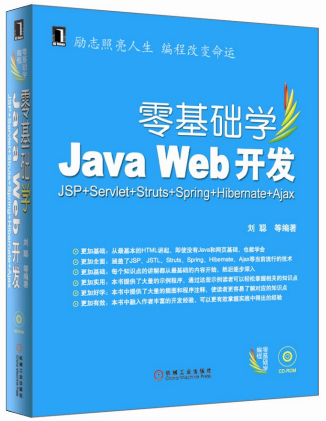
7.4.7　JSP调用存储过程

- 在本书的7.2.6小节中，分两种情况介绍了存储过程的创建方法，所以在这里也分为两种情况介绍存储过程的调用方法，包括调用无参数的存储过程和调用有参数的存储过程。（具体内容请参照书。）



7.5　数据库连接池

- 在大量数据库访问的应用系统中，数据库处理的速度往往会成为系统性能的瓶颈，数据库资源的处理不当往往会给系统的性能带来很大的影响，严重的情况下甚至可以导致整个系统的瘫痪，在本节的内容中将会详细介绍数据库资源的利用问题。

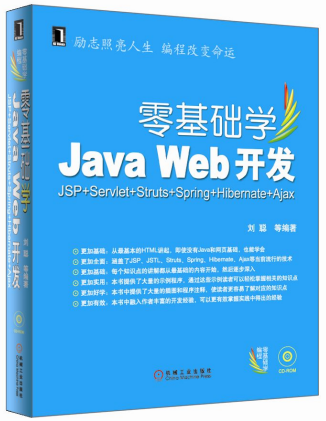


7.5.1　数据库开发中面临的数据库资源问题

- 在数据库应用程序的开发中，经常会遇到数据库资源的问题，一般情况下有一下几种情况。

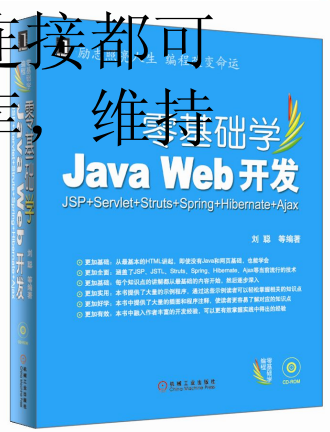
- 1. 普通的JDBC连接带来的效率问题

- 2. 数据库资源使用不当带来的性能问题



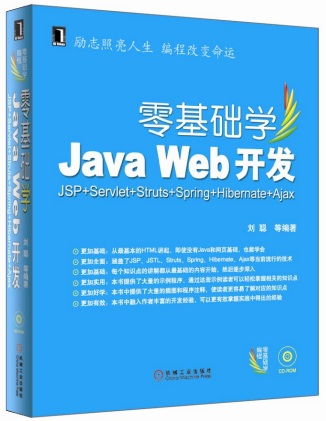
7.5.2　数据库连接池的工作原理

- 数据库连接池就是在系统初始化的时候，建立起一定数量的数据库连接，然后通过一套数据库连接的使用、分配、管理策略，使数据库连接可以得到高效、安全的复用，避免了频繁建立、关闭数据库连接带来的系统开销，另外，使用数据库连接池可以把系统的逻辑实现和数据库分离，在传统的实现过程种，是在应用程序中直接访问数据库，使用连接池以后，所有的数据库连接都可以从连接池中取出，由连接池访问数据库，维持连接池中可用连接的数量。



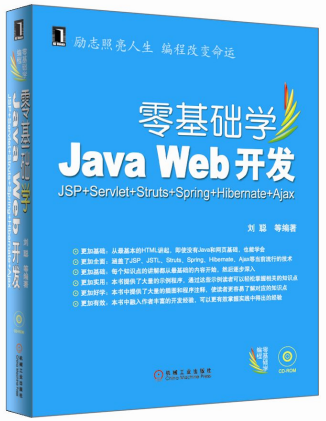
7.5.3　常用的数据库连接池简介

- 在实际的开发中，可以选择自己实现一个连接池，但是一般情况下我们会选择第三方提供的程序的连接池产品，目前有很多成熟的连接池可以选择。



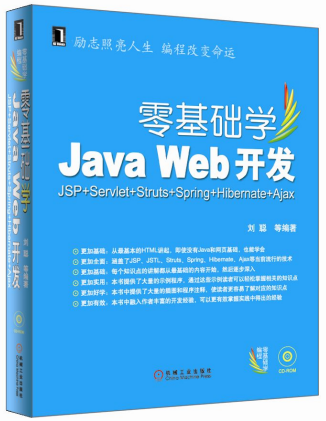
7.5.4　在Tomcat中配置DBCP数据库连接池

- 在Tomcat中配置DBCP数据库连接池的方法有很多，但是配置都相对比较繁琐，在这里提供一种比较简单的DBCP连接池的配置方法。（具体内容请参照书。）



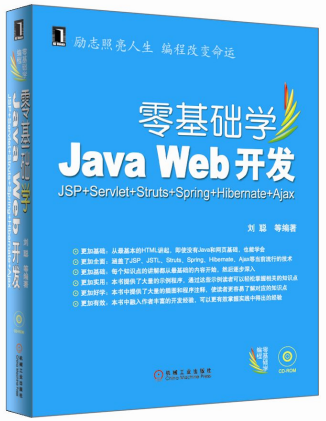
7.5.5　从连接池中取得连接示例

- 通过上节的配置，已经可以在程序中使用DBCP连接池的功能，在下面的实例程序中，将在一个JSP页面中调用DBCP的功能，从连接池中取出一个数据库连接，（具体内容请参照书。）



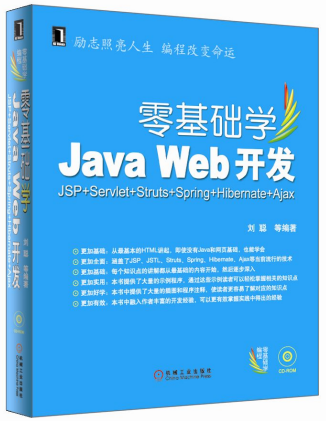
7.5.6　测试连接池设置是否生效

- 在上面的实例程序中，仅仅展示了从连接池中取出连接的情况，但是还没有测试连接池是否可以正常释放资源，这一点是很多开发人员容易忽视的问题，一般情况下，很多开发人员只要可以从连接池中取出连接就认为连接池已经配置成功，然后就开始在程序中使用连接池的功能，这种做法往往在埋下错误的隐患。



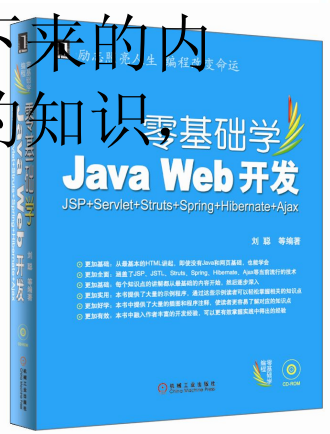
7.5.7　使用DBCP自动回收数据库连接资源

- 在进行数据库操作的时候，数据库的连接是必须释放的，连接池的最大功能是维护一定数量的连接，从而节省建立、关闭数据库连接的时间，在程序中调用的数据库连接是必须关闭的，只有关闭以后，这个连接才会被释放到连接池中，从而可以供其他程序使用。（具体内容请参照书。）



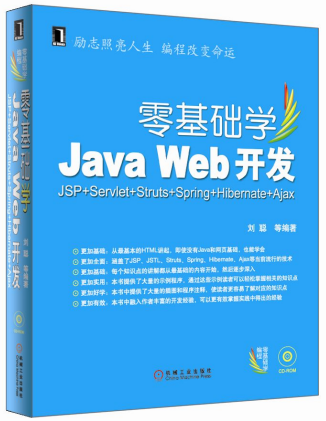
7.6　数据库访问的代码优化

- 通过前面章节中的介绍，读者可以对数据库的基本操作有了一个整体的认识，利用这些基本知识，读者可以进行基本的数据库操作，在本章前面的所有示例程序中，展示的都是最基本的数据库访问操作，目的是让读者能够不受其他因素的干扰，专注于数据库操作功能的实现。但是，从可复用、从软件架构的方面、从软件的升级维护等方面来看，这些是远远不够的，在接下来的内容中，将深入剖析数据库操作更深层次的知识，试读者可以在技术层面有一个质的飞越。



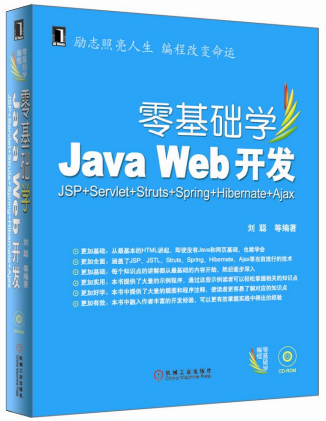
7.6.1　最原始的数据库访问代码示例

- 在上面的所以例子中，都需要使用下面这段代码来建立与数据库的连接。



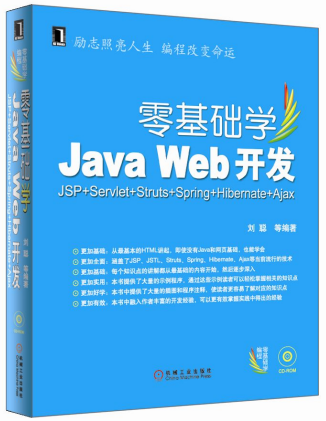
7.6.2　初步优化——数据库配置信息基础类

- 在这种原始的数据库访问操作中，更改数据库的配置信息是非常麻烦的。现在可以考虑这样的处理方法，把数据库的配置信息用常量表示，所有的数据库配置信息都可以放在一个类中，具体实现方式如下。（具体内容请参照书。）



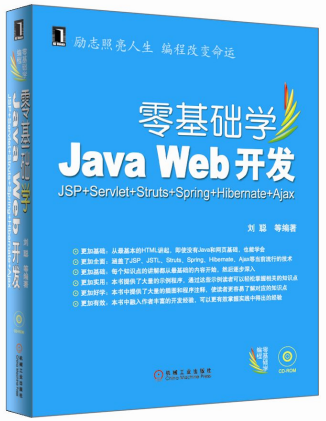
7.6.3　进一步优化——数据库配置文件

- 为了解决用常量表示数据库配置信息带来的问题，避免重新编译部署应用项目，就需要把数据库配置的信息从Java类中提出出来，在这里我们属性文件来代替描述常量的类，经过这样的处理就可以避免重新编译、部署应用系统的问题。



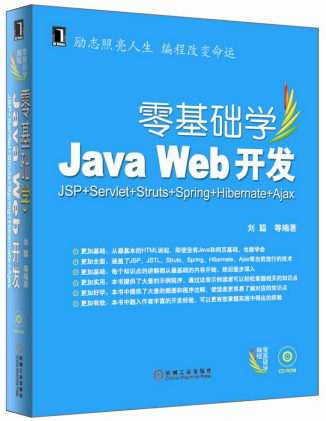
7.6.4　继续优化——数据库访问操作类的封装

- 在前面的几节内容中，对如何合理组织数据库配置信息进行了简单的结束，但是在这些示例程序中我们可以看出，取得数据库连接的依然要重复操作，同样需要加载数据库驱动类、取得连接和进行数据库操作这些步骤，在每个需要数据库操作的地方都要重复这样的代码，那么有没有方法避免这些代码的重复？答案就是对这些代码进行封装。（具体内容请参照书。）



7.6.5　更进一步优化——DAO类封装数据库操作

- 在进行本节内容的介绍之前，先来看下面这段代码。（具体内容请参照书。）



7.6.6　追求完美的优化——添加视图、存储过程

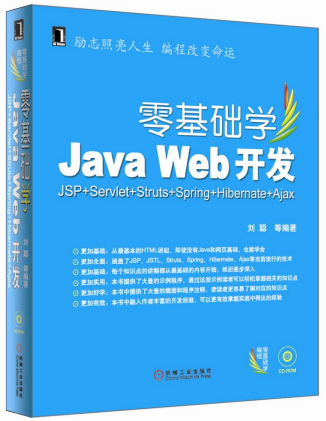
- 在上面介绍的各项优化措施中，都是从软件工程方面考虑，对程序的代码进行科学的优化，单是这些措施并不涉及数据库操作的性能问题，在接下来的内容中，将对数据库操作的性能问题进行探讨。数据库操作性能的要求，一般情况下在需要大量数据库操作的项目种会显得特别迫切。在提升数据库性能方面可以通过合理涉及数据库结构和SQL优化等措施，但这些都不是本节中要讨论的主题，本章的侧重点是在数据库结构已经定型的其功能下，通过合理的访问策略来优化数据库访问的性能。

7.7 小结

- 在本章的内容中，介绍了SQL语言的基础知识，虽然介绍的这些操作都可以在数据库管理工具中实现，但是理解这些基础的知识有利于读者在更深的层次了解数据库的原理和操作。本章还介绍了在JSP中操作数据库的基本方法，并且通过一系列的优化措施，把复杂的数据库操作优化成简单的、优美的代码，在整个优化过程中，读者不仅可以了解到数据库操作优化的方法，更重要的是要体会这些优化中所体现的思想，了解这些思想以后，读者自己就可以根据需要来优化自己的数据库操作，或者优化其他非数据库操作的代码。

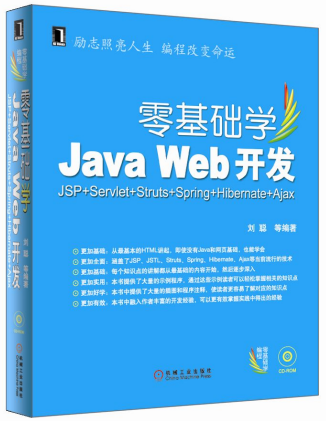
第八章 XML技术基础知识

- 本章将对XML基本知识进行详细的分析，然后介绍几种常见的XML文档处理技术，最后用Java对XML文档的解析来结束本章内容的讲解，通过本章的学习，读者将对XML的基础知识有一个比较清楚的认识，并学会使用常见的XML文档操作技术，并且学会使用JAVA语言处理常见的XML文档。



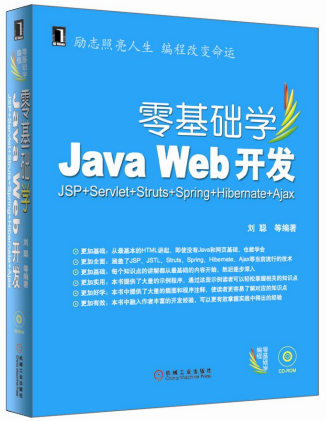
8.1　XML 入门基础

- XML是元置标语言，只要针对具体的业务需要制定对应的词表，互联网上的用户就可以通过词表读出XML文档的具体语义，这样就可以满足互联网上分布式业务数据处理的需要。这就是XML文档能得到广泛应用的最大原因。



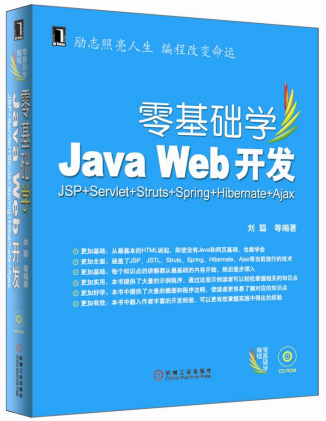
8.1.1　XML简介

- XML是一种元置标语言，它可以使用标签来描述数据，这些标签不是XML中预定义的，而是由用户自己来定义这些标签，用户可以根据实际中的需要创建各种标签，



8.1.2　XML和HTML的区别

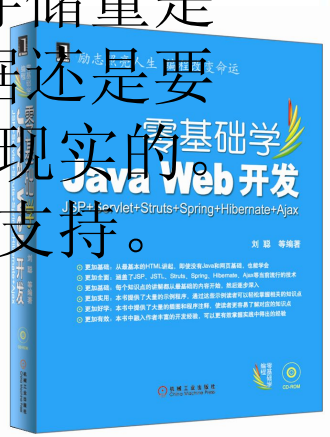
- 在介绍XML和HTML的区别之前，（具体内容请参照书。）



8.1.3　XML和数据库的区别

- XML虽然也可以存储数据，但是并不是说可以替代数据库用来存储数据，XML的主要目的是为了共享数据，这里所说的存储数据是指可以用XML描述对象的结构和信息，从而实现在用户之间共享数据。

- XML并不是要替代数据库，仅仅是在共享数据的时候用来存储对象的结构和信息，这里的存储量是有限的。如果需要存储并管理大量的数据还是要选择数据库。在XML中存储大量数据是不现实的。而且在XML中并不提供数据的管理查询的支持。



8.1.4　XML简单语法

- XML是一种自描述的语言，虽然语法比较简单，但是还是有比较严格的要求。

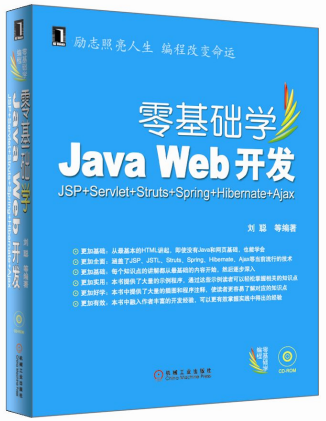
- 1．在XML文档中，必须有XML声明

- 2．XML文档必须有一个而且只能有一个根结点

- 3．XML文档的标签必须成对出现

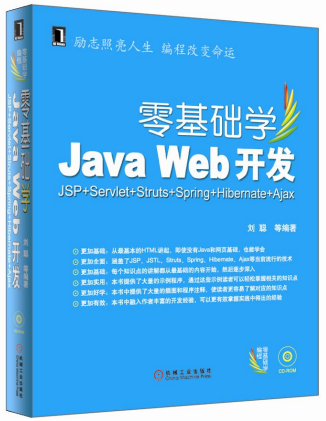
- 4．XML文档中的标签不可以嵌套使用

- 5．XML文档对大小写敏感



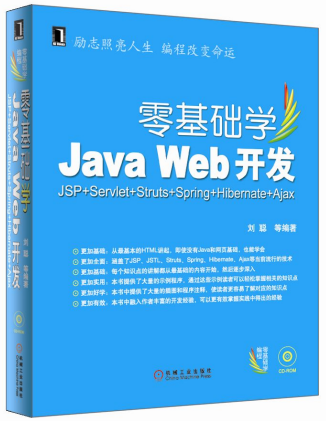
8.1.5　XML简单示例

- 在上面简单介绍做了XML的简单语法，（具体内容请参照书。）



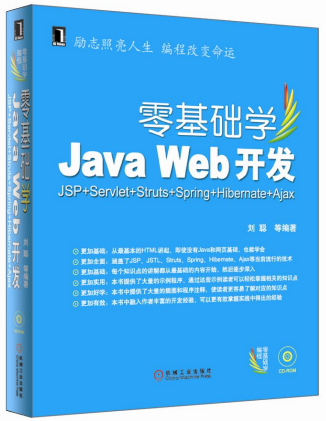
8.2　文档数据类型定义 DTD

- XML文档中的标签都是用户自己定义的，如果这种标签可以任意书写而没有任何约束，那这样的文档就没有任何的用处，因为没有人知道这种文档的含义，更别提什么数据共享了，这样的文档只能成为垃圾信息。



8.2.1　DTD概述

- DTD（Document Type Definition）即文档结构类型定义，DTD用来定义XML文档的模式，通俗的将就是制定XML文档中标签的使用规则。在XML文档中可以包括DTD数据，也可以没有DTD 数据，但是出于共享信息的需求，一般情况下还是需要提供DTD数据。DTD数据使用DTD语言描述的，其中DTD语言是专门描述文档模式的指标语言，DTD本身并不遵循XML的语法。

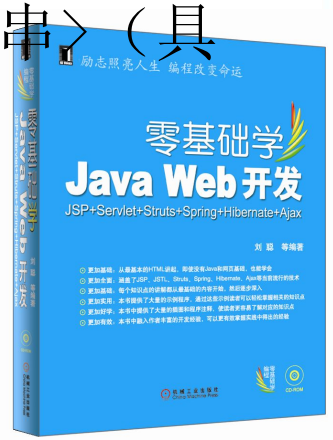


8.2.2　XML文档元素内容定义

- 使用DTD可以对文档的内容进行约束，XML中的DTD数据必须用<!DOCTYPE>标记说明，其语法格式如下。<!DOCTYPE 根元素名称[DTD定义数据]>

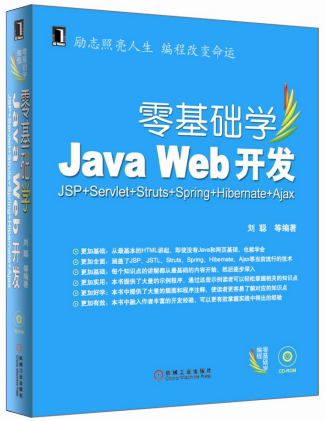
- XML的元素内容定义格式如下。

- <!ELEMENT 元素名称 元素内容描述字符串>（具体内容请参照书。）



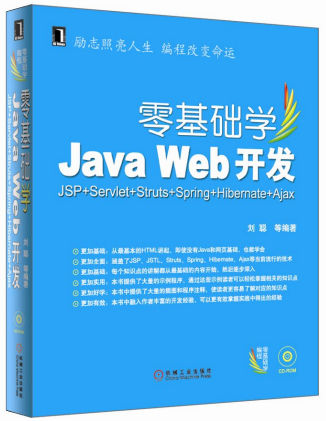
8.2.3　XML文档元素属性定义

- XML中的节点元素可以有自己的属性，同样可以使用DTD对XML文档节点元素的属性进行定义。XML文档节点元素的定义语法如下。<!ATTLIST 节点元素名称 属性名称 属性类型 取值方式>（具体内容请参照书。）



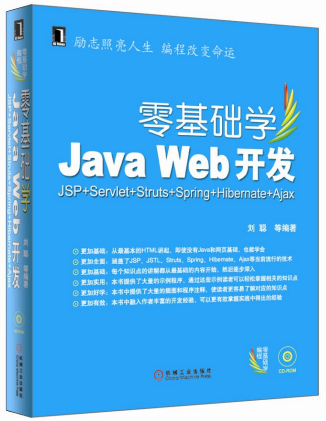
8.2.4　XML外部引用DTD示例

- 在上面的两个示例程序中，DTD的内容都是包含在XML文档之中，如果一个DTD定义对于多个XML文档都是有效的，把这个DTD包括在每个XML文档中当然是可行的，但是在这里可以又更好的方法，就像在XML中引用外部的CSS文件一样，在XML中也可以引用外部的DTD，这样就可以是多个XML文档共用一个DTD文件，而不是在每个XML文档中都重新定义。（具体内容请参照书。）



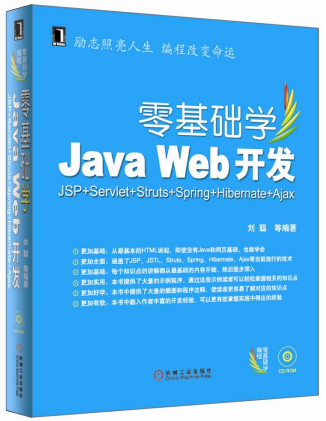
8.3　XML文档的显示技术

- XML是用来描述数据，存储数据的，在前面的例子中我们可以看到，XML文档在浏览器中运行的效果仅仅是节点的树状结构。其实借助与CSS、XSL、DSO等技术，XML文档数据同样可以在浏览器中有丰富的现实效果。这些XML文档的显示技术各有优劣，在本节的内容中将会详细介绍这几种技术的内容。



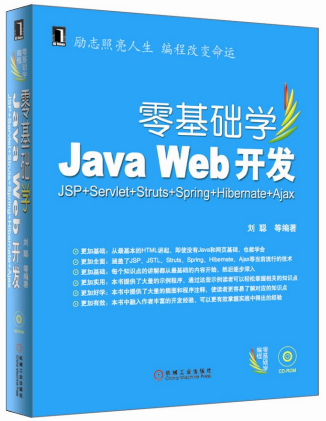
8.3.1　XML文档显示技术概述

- XSL是可以显示XML文档数据内容的另一种技术，XSL可以从XML文档中提取部分内容，然后和HTML模版相结合，从而可以用丰富的形式显示XML文档的内容。DSO即数据源对象，也可以用来显示XML文档的内容，DSO可以取出XML文档的数据内容，从而把这些内容嵌套到HTML标签中，借助于HTML标签强大的显示功能来显示XML文档的内容。



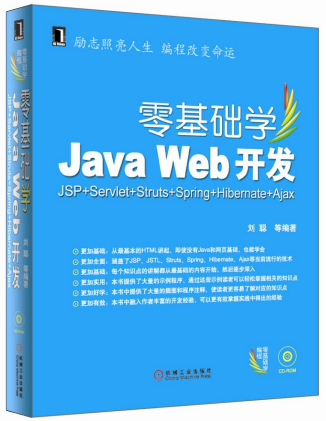
8.3.2　CSS样式表显示

- 在本书的3.3节中简单介绍了CSS样式表的基本知识，在本章中着重介绍如何使用CSS来显示XML文档，（具体内容请参照书。）



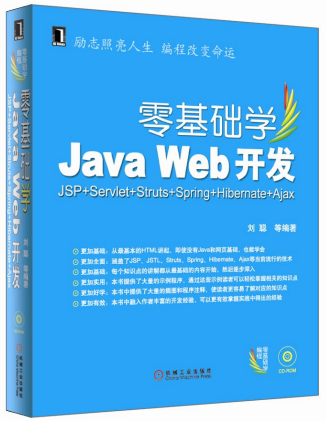
8.3.3　XSL样式表显示

- XSL样式表定义了重组输出XML文档内容的规则，它仅仅是一个规则，定义如何输出XML文档中的内容，输出XML文档的哪部分的内容。（具体内容请参照书。）



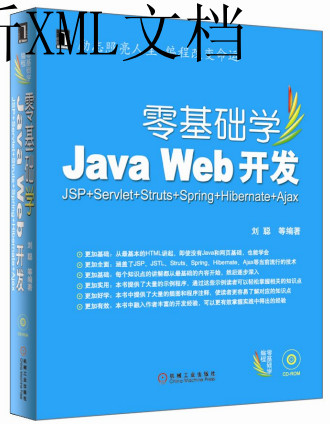
8.3.4　数据岛对象（DSO）显示

- 使用IE5.0或者更高的版本，XML数据可以以数据岛的形式嵌入HTML页面。数据岛对象（DSO）技术是在HTML文档中加入结构化数据进行处理的技术，结构化数据指的是满足一定结构规范的数据集合，而对称结构的XML文档就是结构化数据，所以使用DSO技术可以把XML的内容嵌入HTML页面中，从而达到显示XML文档的目的。



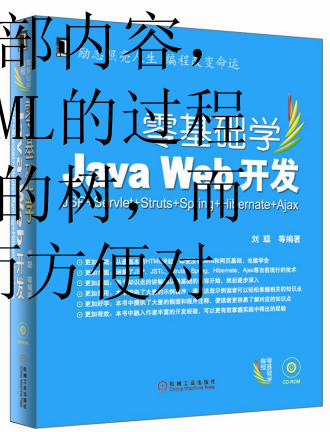
8.4　XML文档DOM解析技术

- 在上面的章节中介绍了XML文档的显示技术，但是在更多的时候，我们需要读取、创建、操作一个XML文档，这就需要对XML文档进行解析，在接下来的章节中将详细介绍XML的解析技术，在IE浏览器中自带有MSXML解析器，只要你装有IE浏览器就已经有了解析XML文档的环境，在本节中，只介绍使用JavaScript语言操作XML文档的技术，在本章的最后一节将详细介绍使用Java语言解析XML文档的技术。



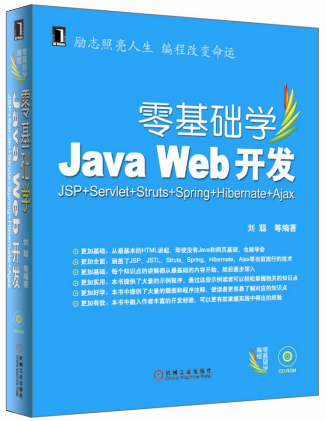
8.4.1　XML文档DOM解析技术简介

- XML DOM（XML Document Object Model）即XML文档对象模型，它是XML分析器提供的处理XML文档的API接口，这种接口与具体的语言无关，可以采用任何一种程序设计语言调用这个接口，通过XML DOM编程接口来操作XML文档，包括操作XML文档的结构数据和内容数据。XML DOM对象模型把XML文档理解为有文档节点构成的一个节点树，树和节点都是抽象的概念，树代表XML文档的全部内容，节点代表文档数据的结构单元，在学习XML的过程中清楚认识到XML文档就是一颗节点构成的树，而DOM就是在内存中构建XML的节点树，从而方便对XML文档的各种操作。



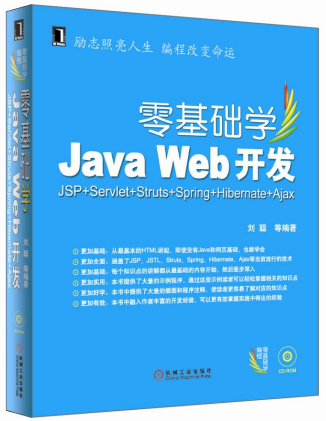
8.4.2　DOM解析示例之验证文档的有效性示例

- 在本书的8.2小节中曾提及XML文档的有效型验证的问题，XML有效性的验证有很多方法，在MSXML分析器中提供了验证XML文档有效性的方法，只是在浏览器中默认是不验证XML文档的有效性的，在下面这个示例程序中，将调用MSXML的解析器对XML文档的有效性进行验证。（具体内容请参照书。）



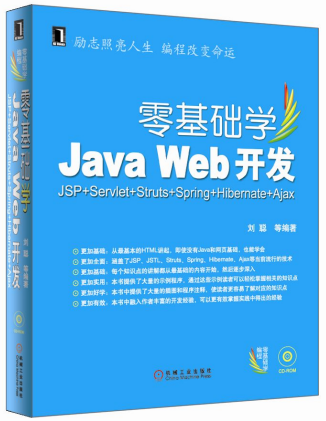
8.4.3　DOM解析示例之动态创建节点示例

- 在上面的实例程序中，只是对XML文档的有效性进行了验证，并没有设计DOM解析的实质内容，DOM最具标志性的操作就是对文档节点的控制，在下面这个示例程序中将介绍XML文档节点操作的方法，



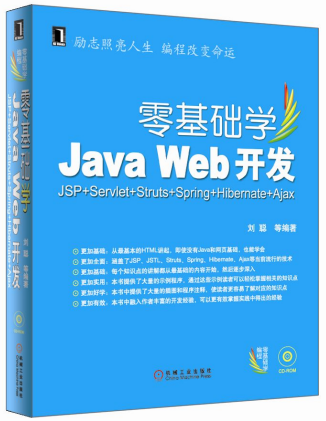
8.4.4　DOM解析示例之操作XML文档节点属性示例

- 在上面的示例程序中，分别展示了XML文档有效性的验证和动态创建XML文档节点，但是对于节点的操作仅仅限于节点的内容，在接下来的内容中将介绍XML文档属性的操作方法。（具体内容请参照书。）



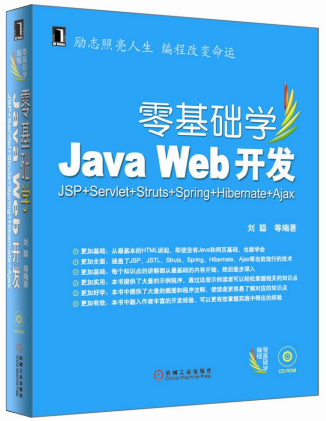
8.5　XPATH节点表达式基础知识

- （具体内容请参照书。）



8.5.1　XPATH简介

- 表达XML文档中节点集合可以用XPath语言来描述，XPath也是由W3C定义、用于在XML文档中描述部分节点集合的语言，这里的节点集合可以是单个节点、以某个节点为根的子树或者是节点与子树的集合，即XPath表达式就是节点和子树的集合。

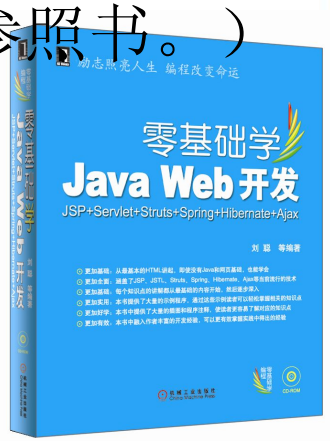


8.5.2　XPATH语言中常用的简写符号

- 在XPath中，有一些常用的简写符号，具体如下。

- （1）“/”代表根节点，就是XML文档节点树的起始节点。

- （2）“A/B”代表层次，即在XML文档节点树中，A节点是B节点的父节点。（具体内容请参照书。）



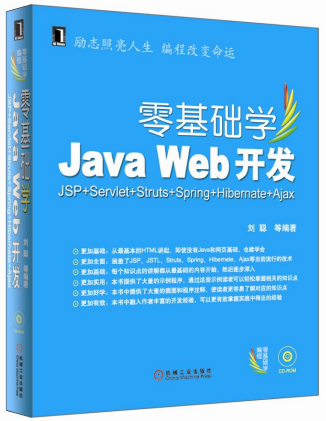
8.5.3　常用文档节点及其表达式

- 在提取XML源文件内容的时候常用的节点是：文档根节点、元素节点、属性节点、和文本节点。为了方便定位这些元素，下面提供常用的文档节点以及器表达式。

- （1）文档根节点

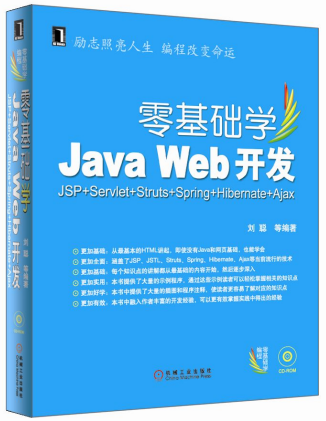
- （2）文档元素节点

- （3）元素属性节点



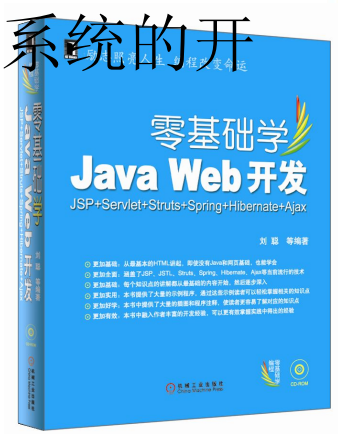
8.6 Java 解析XML

- XML在Java中的地位是相当重要的，尤其在J2EE中，XML的身影更是随处可见，例如在每个J2EE应用项目中都需要有一个web.xml的配置文件，在Tomcat服务器中，几乎所有的配置文件都是XML格式的文档，而且在Web Services中间，XML的地位更显得重要。在接下来的内容中将介绍使用Java解析XML文档的技术。



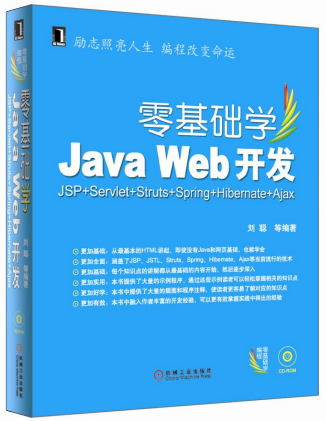
8.6.1　Java处理XML概述

- 在Java解析XML的时候，无论是使用JAXP还是使用第三方的解析器，都有两种解析方式：DOM方式解析和SAX方式解析。其中DOM方式解析就是在内存中构建整个XML文档的节点树，从而使对XML文档的操作变成对内存中结点树的操作。这种方式的优点是可以方便定位节点，缺点是需要把整个文档读入内存，然后才能构建起这个文档的结点树，如果XML文档比较大的时候就会增加系统的开销。



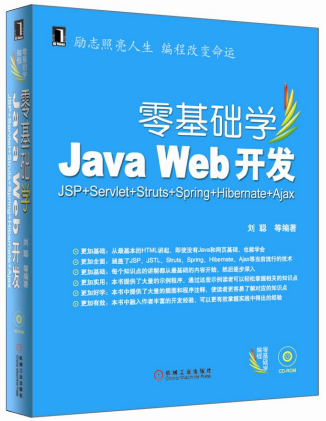
8.6.2　在JSP中生成XML文档

- 在J2EE项目开发的过程中，经常会遇到生成XML文档的需要，使用Java生成XML文档有多种方法可供选择，可以使用JAXP或者是第三方的XML解析工具进行创建，或者使用Java输出字符串，按照XML的标准输出的字符串就是XML文档，在这里我们展示后一种XML文档的生成方法。（具体内容请参照书。）



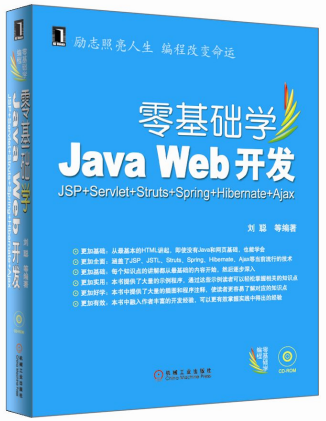
8.6.3　使用JAXP按SAX方式解析XML文档

- JAXP也支持使用SAX的方式解析XML文档，在SAX方式的解析过程中，可以对节点的开始和结束事件进行监听和处理，在这个示例程序中，我们使用一个JavaBean来监听和处理XML的节点解析事件。（具体内容请参照书。）



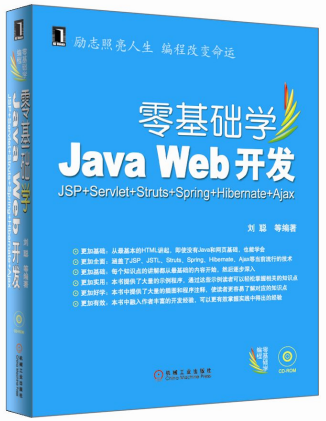
8.6.4　使用JDom按DOM方式解析XML文档

- 在上面的章节中介绍了使用Java语言自带的JAXP解析XML文档的具体方法，但是JAXP只是解析XML文档的一种简单的实现，在实际的解析过程中，某些场景下使用JAXP并不方便，这样就需要使用第三方提供的XML解析API，在这方面做的比较出色的有JDom、Dom4J、Xerces等，这些API的使用方法都很相似，在本章的以JDom为例进行介绍。（具体内容请参照书。）



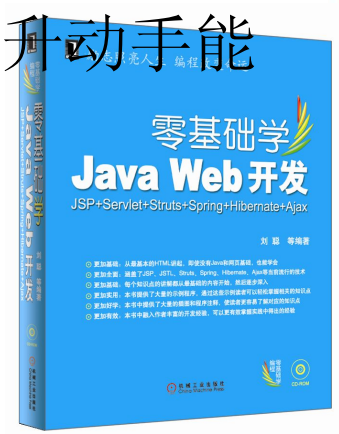
8.6.5　使用JDom按SAX方式解析XML文档

- 在JDom中对于SAX方式解析XML文档的功能支持相当优秀，大家都知道SAX方式解析XML文档的效率很高，但是节点操作不是很方便，在JDom中，使用SAX方式解析XML文档和使用DOM方式没有很大的区别，在SAX方式中，同样可以对节点进行操作，所以在实际的开发中，开发人员一般情况下会选择JDom的SAX解析方式。



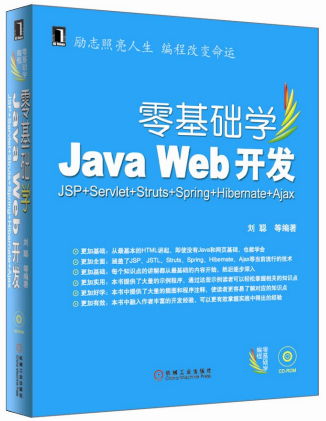
8.7 小结

- 在本章的内容中，简单介绍了XML的基本语法，对XML的显示技术进行了比较详细的介绍，介绍了三种比较常用的XML文档显示技术，XML文档的解析是XML学习中的重点，在本章中占了比较大的篇幅，其中详细介绍了使用Java解释XML的几种常用的技术，在学习完本章的知识以后，读者应该对XML的基本知识有一个清楚的认识，并尝试自己写程序来解析XML文档。在实践的基础上提升动手能力，同时加深对XML知识的理解。



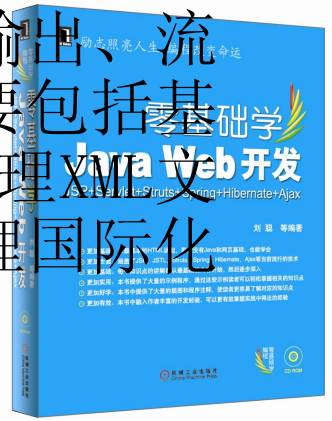
第九章 JSTL基础知识

- JSTL的全名为JavaServerPages Standard Tag Library，即JSP标准标签库，它是由Apache基金组织的jakarta小组开发维护的，其主要功能是为JSP Web开发人员提供一个标准通用的标签库。开发人员可以利用这些标签取代JSP页面上的Java代码，从而提高程序的可读性、降低程序的维护难度。



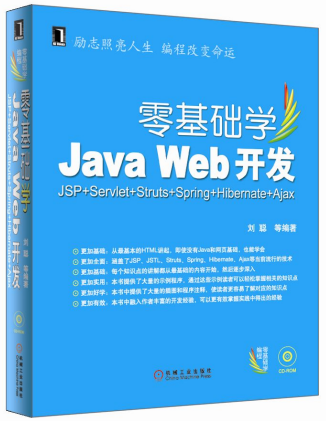
9.1　JSTL简介

- JSTL标签是基于JSP页面的，这些标签可以插入在JSP代码中，在本质上JSTL也就是提前定义好了的一组标签，这些标签封装了不同的功能，当在页面上调用这个标签的时候，等于就是调用了封装起来的功能，这些标签可以在页面上输出内容、查询数据库、处理XML文档等，JSTL的标签库基本上可以分为五类，包括JSTL核心库、数据库标签库、XML操作标签库、国际化和格式标签库和函数标签库，其中在核心库包括基本的输入输出、流程控制、循环等功能；数据库标签库主要包括基本的数据库操作功能；XML标签库用来处理XML文档；国际化和格式标签库主要功能是处理国际化和文字格式的标准化。



9.2　JSTL开发环境简单配置

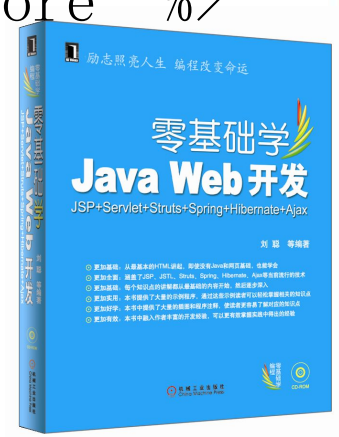
- 如果要在项目中使用JSTL，就需要在配置JSTL的环境，JSTL环境配置非常简单，首先要下载JSTL，（具体内容请参照书。）



9.3　核心标签库

- JSTL核心库主要有输入输出、流程控制、迭代操作、URL操作等功能，如要要在JSP页面中使用核心库的标签，需要用taglib指令指明这个标签库的路径为。

- <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

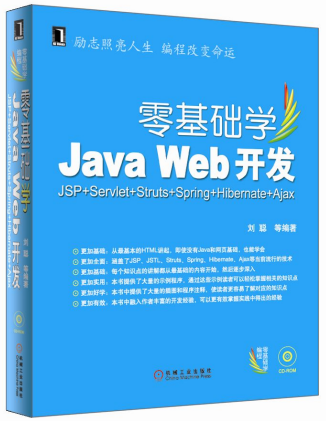


9.3.1 <c:out>标签

- <c:out>标签的功能就相当于JSP中的out对象，可以在JSP页面上打印字符串，也可以打印一个标达式的值。使用语法如下。<c:out value="value" [escapeXml="{true|false}"] [default="defaultValue"] />或者使用下面这种格式。<c:out value="value" [escapeXml="{true|false}"]>

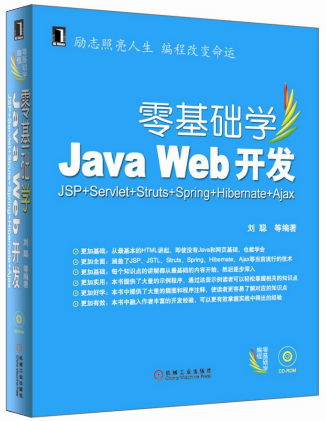
- default value

- </c:out>



9.3.2 <c:set>标签

- <c:set>的主要功能是把变量的值设置到JSP内置对象中，或者是设置到JavaBean的属性中。<c:set>的功能和JSP动作指令中的<jsp:setProperty>类似。（具体内容请参照书。）

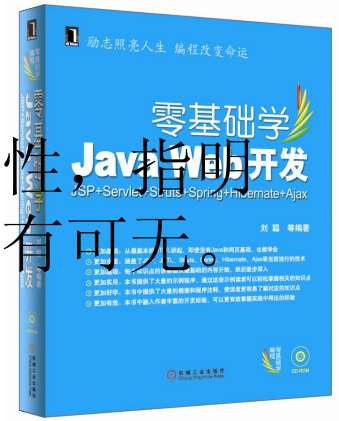


9.3.3 <c:remove>标签

- <c:remove>标签执行的功能和<c:set>标签的功能正好相反，<c:remove>标签可以移除在<c:set>标签中设置的变量。具体使用语法如下。

- <c:remove var="varName"
  [scope="{ page|request|session|application }"] />

- 在上面这行代码中，必需提供varName属性，指明需要移除哪个变量，而scope的属性则可有可无。



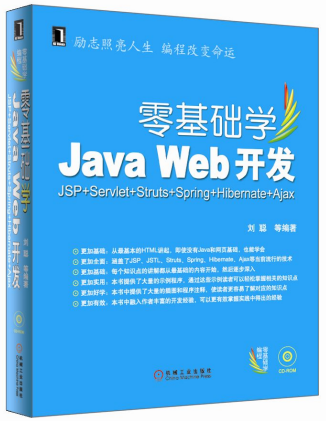
9.3.4 <c:if>标签

- 和JSP中的if功能一样，<c:if>的功能也是用来做条件判断，使用语法如下。

- <c:if test="testCondition" [var="varName"] [scope="{page|request|session|application}"]]>

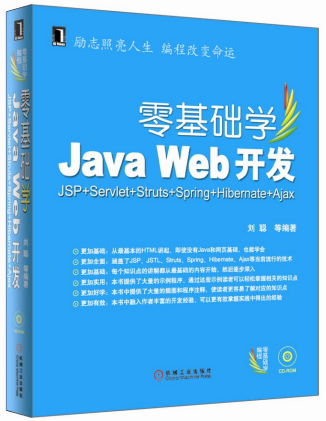
- 结果为真时执行的操作代码

- </c:if>



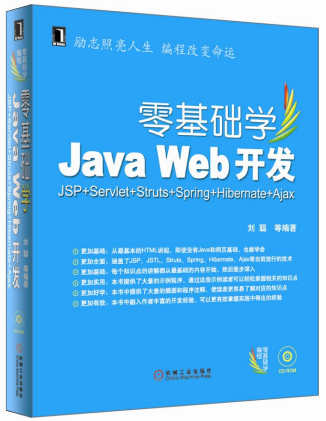
9.3.5 <c:choose>、<c:when>、<c:otherwise>标签

- <c:choose>标签本身没有具体的功能，它仅仅是做为<c:when>和<c:otherwise>的父标签。为了方便理解，在这里把<c:choose>、<c:when>、<c:otherwise>合并为一节介绍。（具体内容请参照书。）



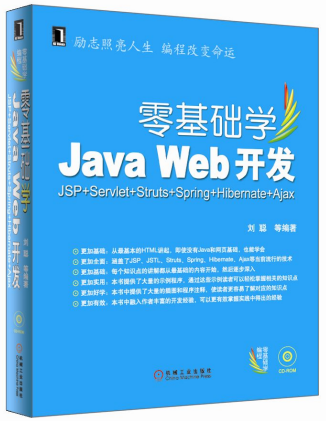
9.3.6 <c:forEach>标签

- <c:forEach>标签是一个迭代标签，它可以用来做循环的控制，可以循环遍历一个集合中的内容，这里的集合何以是数组、List、Array、ListLinkedList、set、Vector、Map等常用集合对象。（具体内容请参照书。）



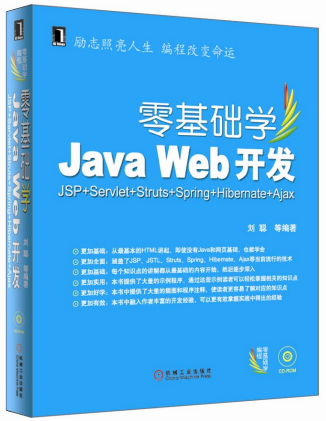
9.3.7 <c:forTokens>标签

- <c:forTokens>标签是JSTL种的另一个迭代循环标签，它可以用来对一个字符串进行迭代循环，这个字符串是用符号分开的。（具体内容请参照书。）



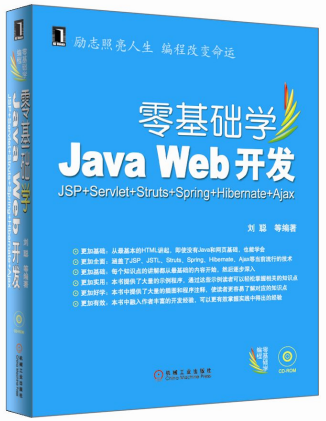
9.3.8 <c:import>标签

- <c:import>标签可以把其他动态或者静态的文件包含到当前的JSP页面，功能和JSP动作指令标签<jsp:include>类似，但是和<jsp:include>不同的是，<jsp:include>只可以包含当前Web应用的内容，而<c:import>标签不仅可以包含当前Web应用的内容，而且还可以包含其他Web应用的内容，包括其他网站的内容。（具体内容请参照书。）



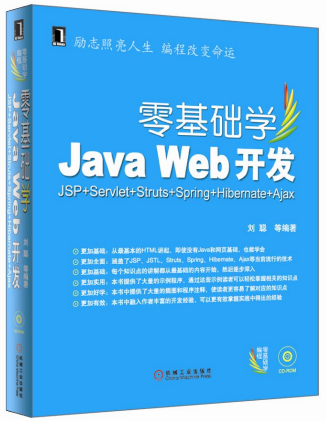
9.3.9 <c:redirect>标签

- <c:redirect>标签可以把用户的请求从一个页面转向另一个页面，同JSP中response内置对象的跳转功能类似，这个标签也用两种使用语法，（具体内容请参照书。）



9.3.10 <c:url>标签

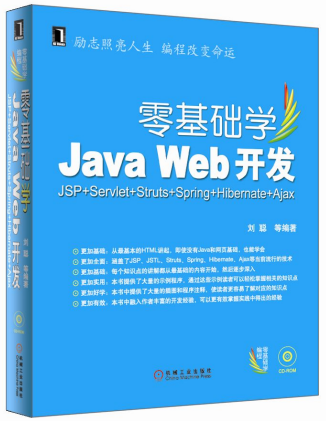
- <c:url>标签主要用来产生一个URL，这个标签也有两种使用语法，第一种语法如下。（具体内容请参照书。）



9.3.11 <c:param>标签

- <c:param>标签的作用就是向一个页面传递一个参数，其使用语法如下。

- <c:param name="paramName" value="paramValue"/>



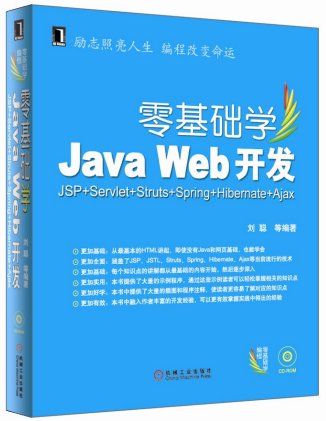
9.3.12 <c:catch>标签

- <c:catch>标签用来捕捉JSP页面产生的错误，和Java语言中的catch功能是类似的。其使用语法如下。

- <c:catch [var="varName"] >

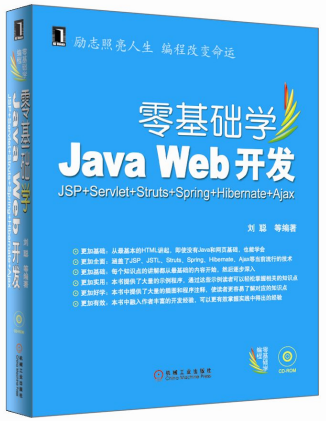
- 要捕捉异常的部分

- </c:catch>



9.4　数据库标签库

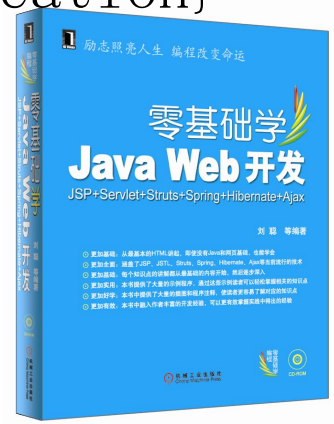
- 数据库开发在JSP中占有非常重要的地位，JSTL也提供了对数据库操作的支持，通过是使用JSTL，数据库操作可以简化为简单的几个标签，大大提高了数据库开发的效率和程序的可维护性。在接下来的章节中将详细介绍JSTL数据库操作标签库。



9.4.1 <sql:setDataSource>标签

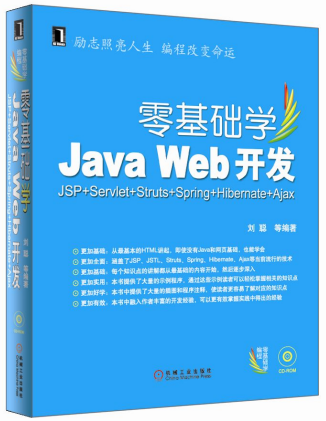
- <sql:setDataSource>标签可以用来设置数据源，其使用语法有两种，第一中语法如下。

- <sql:setDataSource driver="driverName" url="URL" [user="userName"] [password="password"] [var=" varName"] [scope="{page|request|session|application}" />



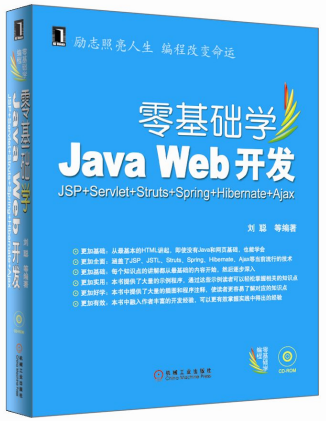
9.4.2 <sql:query>标签

- 在上面我们使用<sql:setDataSource>标签已经取得了与数据库的连接，在这个数据连接的基础上就可以对数据库进行各种操作，其中对数据库的查询操作就要用到<sql:query>标签，<sql:query>标签的作用就是从数据库中取出指定的结果集，其中对数据库的操作还是依靠传递SQL语句作为参数。（具体内容请参照书。）



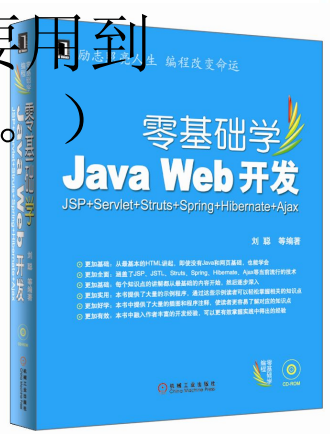
9.4.3 <sql:update>标签

- <sql:update>标签的功能是对数据库进行更新操作，和<sql:query>标签的用法类似，<sql:update>标签的使用方法也有两种形式，



9.4.4 <sql:param>标签

- 在第七章讲解数据库开发的时候，已经详细介绍过PreparedStatement的使用方法，在使用PreparedStatement进行数据库操作的时候，可以使用类似“select * from stores where stor_id=?”这样的SQL语句，其中问号的位置取代的是一个参数，可以在后面动态设置，这样操作的好处在第七章已经详细介绍过。如果在JSTL中要实现这样的传递参数的功能，就需要用到<sql:param>标签。（具体内容请参照书。）



9.4.5 <sql:dateParam>标签

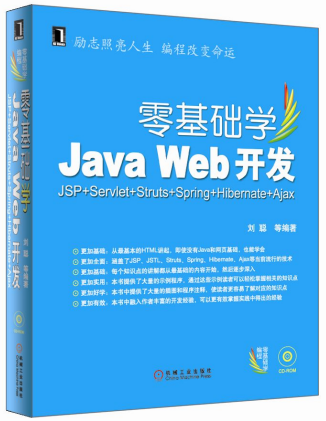
- <sql:dateParam>标签和<sql:param>标签的功能和用法完全相同，不同指出是<sql:dateParam>标签是用来设置日期格式的参数，其使用语法如下。

- <sql:dateParam value=”value” [type=”{timestamp|time|date}”]/>

- 其中，type指明了参数的类型，time是时间格式，date是日期格式，timestamp是日期加时间的完整格式。<sql:dateParam>标签的使用方法和<sql:param>标签是一样的，在这里不再赘述。

9.4.6 <sql:transaction>标签

- 在JSTL中，同样支持数据库操作的事务处理，在JSTL中是采用<sql:transaction>标签来实现这个功能的。（具体内容请参照书。）

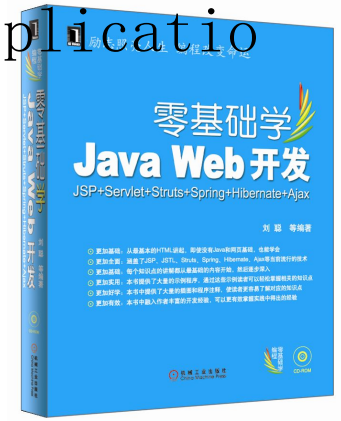


9.5　XML操作标签库

- 在Java中可以使用SAX或者DOM等API接口来操作XML文档，尽管这种操作功能非常强大而且灵活，但是要很快熟练掌握是有相当大难度的，在JSTL中提供了一组专门处理XML文档的标签，这些标签所提供的功能尽管非常有限，但是已经可以满足基本的XML文档处理需要，而且这些标签学习起来明显比掌握复杂的API接口要容易。接下来的章节中将介绍JSTL中用来处理XML文档的标签。

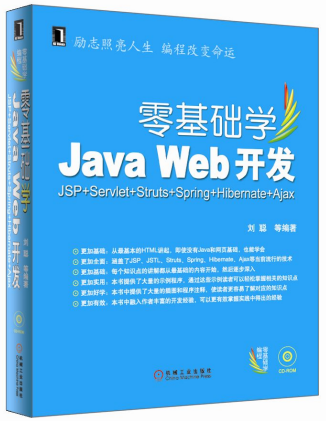
9.5.1 <x:parse>标签

- <x:parse>标签可以用来解析一个XML文档，这个标签也有两种基本用法，第一种用法的语法格式如下。<x:parse
  {doc=" xmlDoc" |xml=" xmlDoc" }
  {var=" varName"
  [scope="{page|request|session|application}"]
  |varDom=" varName"
  [scopeDom="{page|request|session|application}"] } [systemId=" systemId"]
  [filter=" filter"]/>



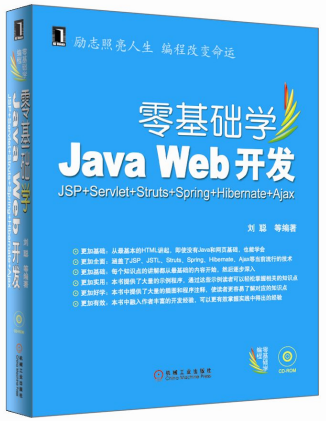
9.5.2 <x:out>标签

- 使用<x:parse>把一个XML文档解析以后，就可以使用<x:out>标签输出XML节点的值，<x:out>在这里的功能和<c:out>有点类似。<x:out>标签的使用语法如下。<x:out select=”node” [escapeXML=”{true|false}”]/>



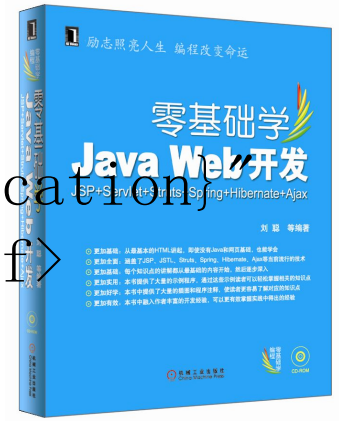
9.5.3 <x:set>标签

- <x:set>标签的功能和<c:set>标签的功能类似，这两个标签的功能都是把一个值设置到一个变量中，不同指出在于<x:set>标签是把XML中某个节点的内容赋值到一个变量。这个标签的使用语法如下。<x:set select=”node” var=”varName” [scope=″{page|request|session|application}″]/ >



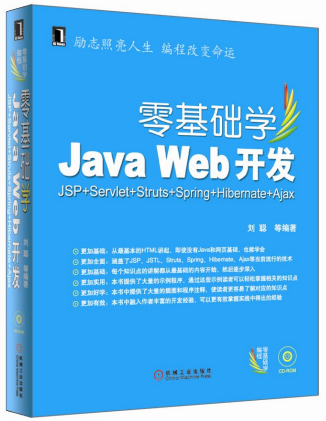
9.5.4 <x:if>标签

- <x:if>标签的功能和<c:if>的功能类似，只不错在这里判断的条件变成了XPath表达式，即判断在XML文档中是否有满足这个XPath表达式的节点。<x:if>标签也有两种使用格式，第一种用法的语法如下。<x:if select=”node” var=”varName” [scope=″{page|request|session|application}″]/><x:if>的另一种用法如下。<x:if select=”node” var=”varName” [scope=″{page|request|session|application}″]> 判断条件成立时执行的操作</x:if>



9.5.5 <x:choose>标签

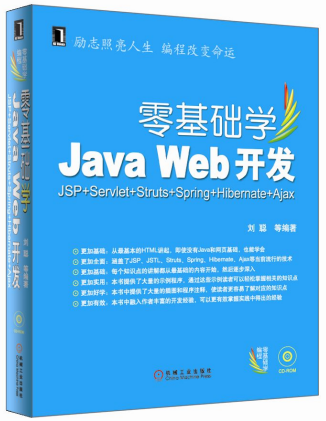
- <x:choose>标签是<x:when>和<x:otherwise>的父标签，其使用方法和核心库中的对应标签<c:choose>、<c:when>、<c:otherwise>的使用方法类似，只不过在这里的判断条件变成了XPath的表达式，其他使用方法完全相同，在这里可以参考<c:choose>、<c:when>、<c:otherwise>的使用方法。



9.5.6 <x:forEach>标签

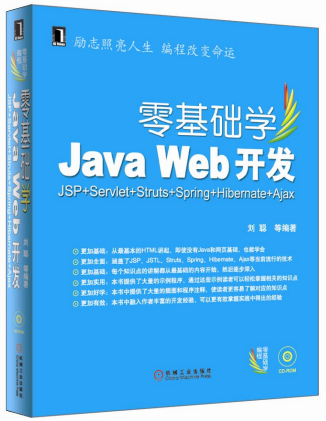
- 使用<x:forEach>标签可以对XML文档进行遍历，这个标签的用法和<c:forEach>标签的用法非常相似，这个标签的使用语法如下。<x:forEach [var="varName"] select=" node" [varStatus="varStatusName"] [begin="begin"][end="end"] [step="step"]>
  循环体中要执行的内容

- </x:forEach>



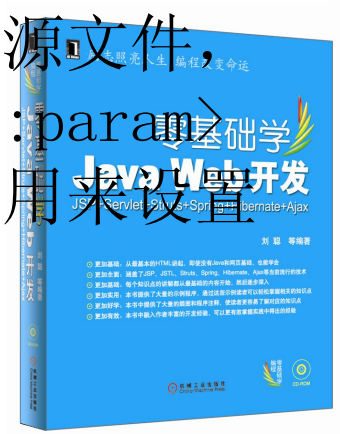
9.6　其他标签

- 除了上面介绍的核心库、数据库操作库和XML文档处理库，JSTL还有其他比较常用的标签库，包括处理国际化和格式的标签库、函数标签库，接下来的章节将简单介绍这两个标签库的基本知识。



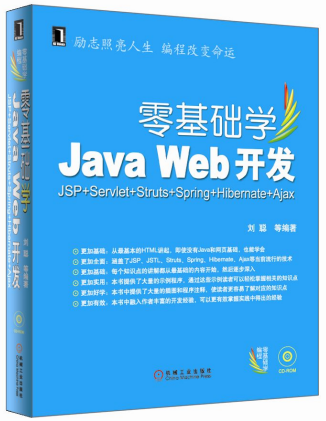
9.6.1　国际化标签

- 在JSTL可以使用<fmt:setLocale>标签设置国家地区参数，从而可以使用本地化的表示方式。另外使用JSTL还可以自动根据Local的设置搜索资源文件，从而实现其他国际化的问题，这样就需要另外几个国际化标签，包括<fmt:bundle>、<fmt:setBundle>、<fmt:message>、<fmt:param>、<fmt:requestEncoding> 等，其中 <fmt:bundle>、<fmt:setBundle>这两个标签用来绑定资源文件，<fmt:message>用于显示资源文件，<fmt:param>用于传递参数，<fmt:requestEncoding>用来设置请求的字符编码。



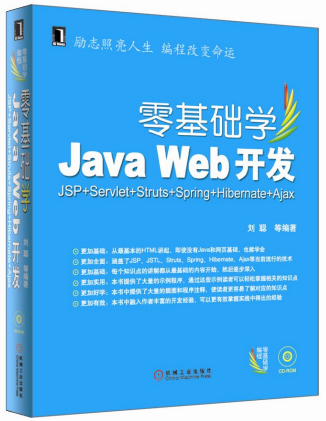
9.6.2　数字格式标签

- JSTL也可以使用不同的格式来表示数字，可以用百分数或者是货币等，下面是一个综合国家地区设置和数字货币格式的示例程序。（具体内容请参照书。）



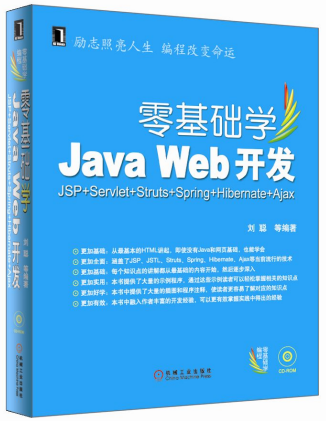
9.6.3　日期格式标签

- 使用JSTL的标签，可以很方便的处理日期的格式，其中<fmt:formatDate>可以把日期格式的对象按照格式输出，而<fmt:parseDate>则可以把一个字符串解析成一个日期格式的对象。



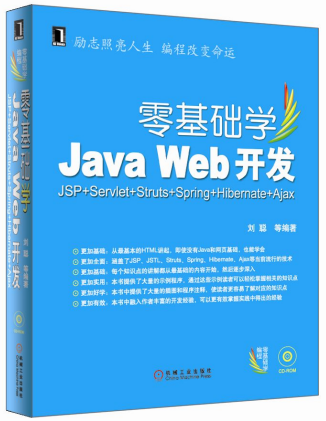
9.6.4　函数标签库

- JSTL函数标签库就是一些常用的函数，在JSTL中把这些常用的函数封装成标签的形式，然后可以在JSP页面上进行方便的调用。（具体内容请参照书。）



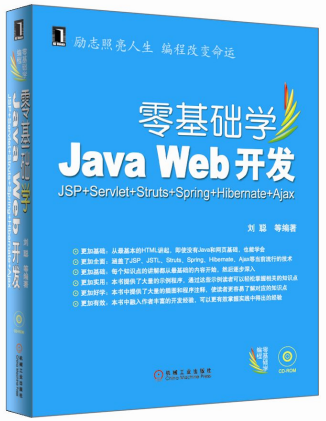
9.7　自定义标签库

- JSTL自带的标签功能非常强大，但是仅仅这些标签并不能完全满足实际开发中的需要，在这个章节将介绍如何定制开发自己的标签库。开发自定义标签，首先需要开发标签所对应的功能类，其次要编写标签的描述文件tld，并把这个文件放在项目的WEB-INFO/目录下，然后才可以在JSP页面上调用自定义的标签。（具体内容请参照书。）



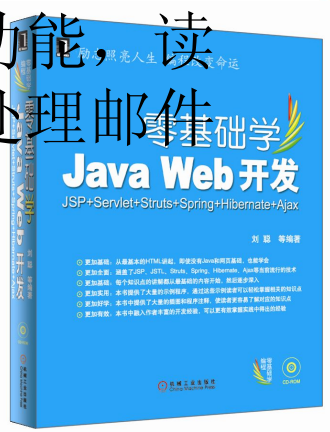
9.8 小结

- 本章介绍了JSTL的基本知识，包括JSTL核心标签库、数据库操作标签库、XML文档处理标签库等，而且还介绍了自定义标签的开发过程，读者在实际的开发过程中可以参考本章的示例程序，在学习本章知识的基础上熟练运用JSTL提高开发速度和质量，并在自己的开发中尝试自定义标签库来简化开发的代码量。



第十章　JSP中邮件功能开发

- 在Web应用程序的开发中，经常会有处理邮件的需求，在J2EE中提供了邮件处理的API，使用这些API，可以非常方便的接收和发送邮件，在本章的内容中，将简单介绍邮件处理的基础知识，然后通过具体的示例展示使用Java提供的API处理邮件的基本方法。通过本章内容的学习，读者可以掌握使用Java完成基本的邮件操作功能，本章的示例程序基本涵盖大部分常用的邮件处理功能，读者可以通过这些示例程序体会使用Java处理邮件的基本方法和技巧。



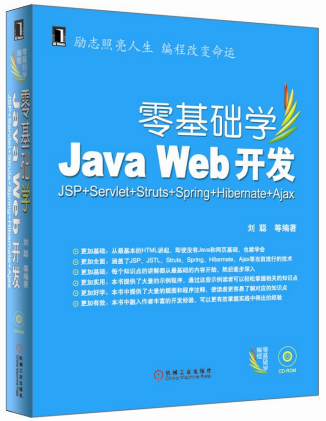
10.1　邮件协议简介

- 当电子邮件在不通的服务器之间进行传递的时候，需要遵循特定的规则，也就是说在发送和接受邮件的时候，都需要通过特定的邮件协议，只有在协议的保证下，电子邮件才可以在不通的服务器之间正常传递，下面介绍几种常用的电子邮件协议。

- 1. SMTP

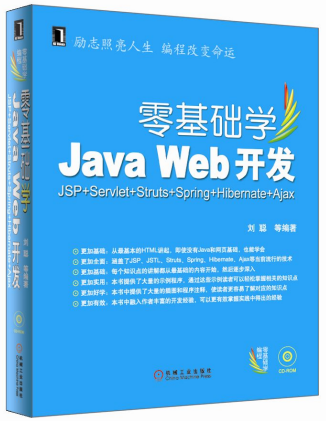
- 2. POP3

- 3. IMAP（具体内容请参照书。）



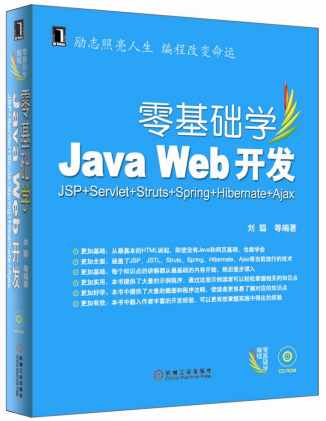
10.2　JavaMail简介及开发环境配置

- JavaMail API是读取、撰写、发送电子邮件的可选包，它支持常见的邮件传递协议，例如SMTP、POP3、IMAP等，开发人员在使用JavaMail进行开发的时候，无需考虑邮件协议的底层实现细节，在JavaMail API中，提供了统一的邮件操作接口，无论选择使用何种邮件协议，都可以调用同样的方法进行邮件的操作。（具体内容请参照书。）



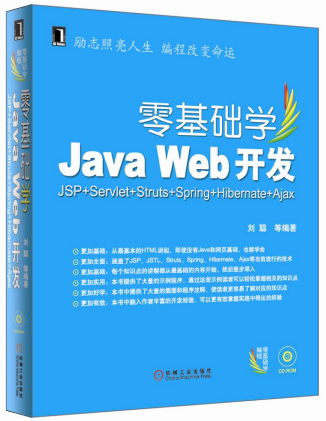
10.3　发送邮件示例详细解析

- 在上面的内容中，简单介绍了JavaMail的基本配置，在接下来的内容中，将集中介绍使用JavaMail发送各种邮件的操作，在具体的示例程序中展示JavaMail发送邮件的基本操作方法。在JavaMail中，提供了发送文本邮件、HTML邮件、附件等功能，这些功能在下面的示例程序中将依次介绍。在这些示例程序中，发送邮件时都选择使用SMTP协议。



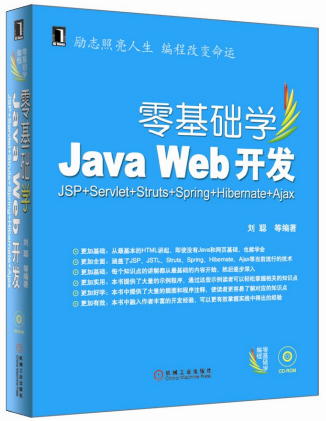
10.3.1　发送文本邮件

- 发送文本邮件是JavaMail中最基本的操作，下面的示例程序就展示了JavaMail中发送简单文本邮件的基本方法。在这个示例程序中，用户可以在JSP页面上输入要发送邮件的内容和地址，通过表单把邮件的内容提交给一个Servlet进行处理，这个Servlet调用一个JavaBean进行邮件发送。（具体内容请参照书。）



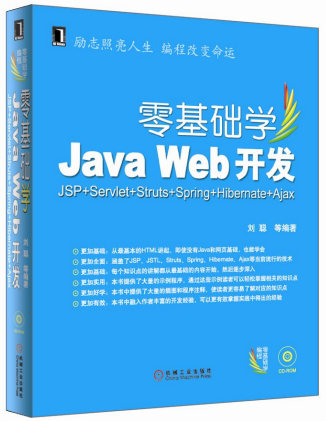
10.3.2　发送HTML邮件

- 在JavaMail中，不仅可以发送简单的文本邮件，而且还可以发送HTML格式的邮件，下面就是发送HTML格式邮件的简单示例程序。（具体内容请参照书。）



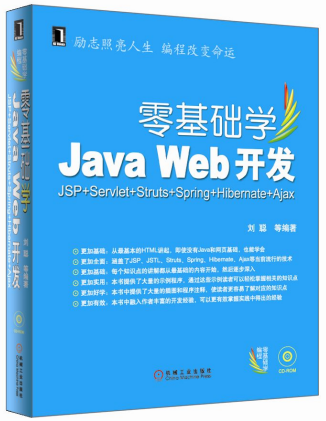
10.3.3　发送带有附件的邮件

- 在JavaMail中不仅可以发送文本邮件和HTML格式的邮件，而且还可以发送附件内容，在下面的示例程序中，就展示了使用JavaMail发送附件的基本操作方法。（具体内容请参照书。）



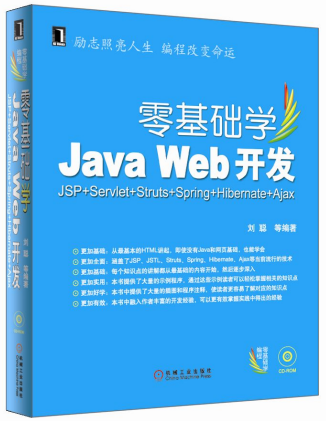
10.4　接收邮件示例详细解析

- 在上面的内容中，介绍了使用JavaMail发送各种邮件的基本方法，在接下来的内容中，将简单介绍使用JavaMail API接收邮件的基本方法。



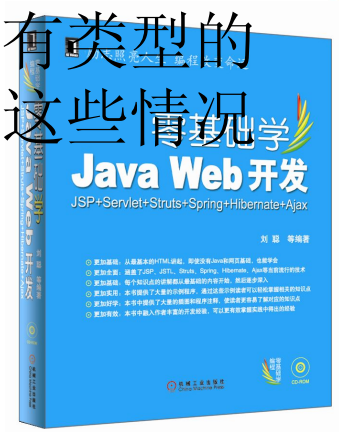
10.4.1　接收简单邮件

- 在JavaMail中，接收文本邮件和HTML格式邮件的处理方法基本上是一样的，在下面这个示例程序中，将把文本邮件和HTML格式的邮件放在一起进行处理，通过一个简单的示例程序，展示使用JavaMail接收简单邮件的基本方法。在这个示例程序中，使用Servlet接收用户提交的表单，在Servlet中调用JavaBean实现邮件的接收。（具体内容请参照书。）



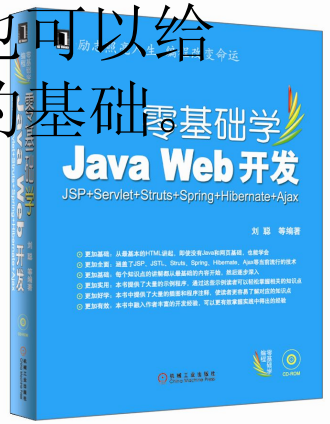
10.4.2　接收附件

- 在读取邮件的过程中，带有附件的邮件读取过程比较复杂。带有附件的邮件一般情况下是由多个部分组成的，在读取的时候，需要读取每一个部分，从而取得邮件的内容和附件的内容。在读取附件的时候可以使用Part类中的getDisposition（）方法，这个方法可以获得邮件体部分的类型，当这个部分是附件的时候，这个方法的返回值会是Part.ATTACHMENT，附件也可以没有类型的方式存在或者类型为Part.INLINE，所有这些情况都可以通过JavaMail读取出附件的内容。



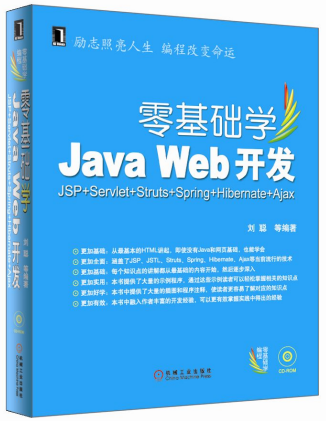
10.5 小结

- 在本章的内容中，简单介绍了JavaMail的基本知识和开发环境的配置，然后通过具体的示例程序展示了使用JavaMail API发送和接收邮件的基本方法，对常见类型的邮件处理方法做了简单的介绍。通过本章内容的学习，读者可以了解JavaMail的基本知识，从而可以在开发过程中学习使用JavaMail API提供的工具进行邮件功能的开发，而且通过本章内容的简单学习，也可以给读者进一步深入学习JavaMail打下坚实的基础。



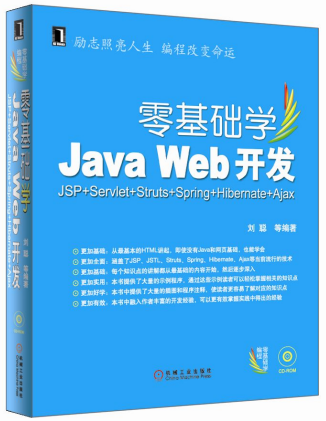
第十一章 Web报表基础知识

- Web报表的开发是一个比较常见的功能，然而在B/S架构上实现这些功能并没有在C/S架构上那么简单，针对这样的问题，本章提供相应的解决方案，在下面的内容中将介绍JSP与Excel的交互、图形报表的制作和基本的Web打印功能，这些功能都不是很难，可以在掌握这些功能的基础上举一反三，思考其他新的应用。



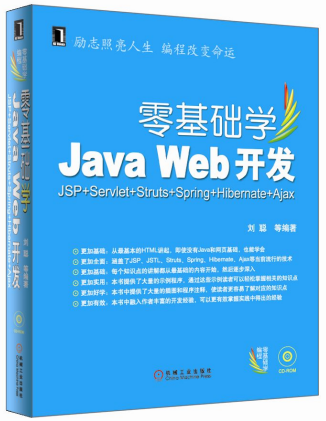
11.1　JSP对Excel报表的处理

- 在应用系统开发的过程中，很多客户会提出把数据表格导出为Excel文件的需求，这样就可以在利用Excel的强大功能做一些统计计算。Java自带的API中并没有直接操作Excel文档的方法，如果要在Java中处理Excel文档只有借助于第三方的解决方案。在接下来的章节中将要介绍的就是利用这些第三方的类库处理Excel文档的具体方法。



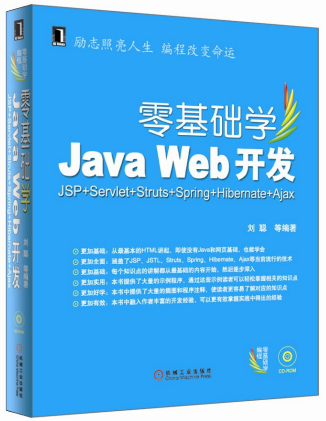
11.1.1　JSP操作Excel工具汇总

- Apache POI是Apache基金组织Jakarta项目的子项目。POI包括一系列的API，可以操作多种格式的MicroSoft Office文件，通过这些API可以在Java中很方便地读写Excel、Word等文件。POI是比较完整的Java Excel和Java Word解决方案。其子项目包括：POIFS、HSSF、HDF、HPSF。其中HSSF是Java到Microsoft Excel 97(-2002)文件的接口，支持读写功能。



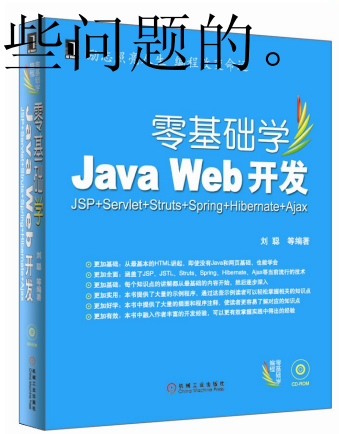
11.1.2　JExcelApi开发环境简单配置

- JExcelApi是一个开源的项目，基于GPL发布，可以在官方网站下载其最新版本。（具体内容请参照书。）



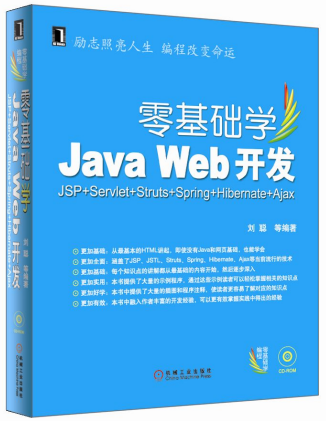
11.1.3　JSP生成Excel报表

- 在接下来的章节中，将讲解在JSP中使用JExcelApi生成不同格式的Excel文档。在Web应用开发的过程中，可能会遇到各种各样的报表需求，这些报表不仅布局格式复杂，而且数据类型也是多中多样，甚至有的报表需要在指定的位置显示图片。当这些报表需要导出为Excel的时候，相应的问题就会出现，而在本章接下来的内容要阐述的就是怎样使用JExcelApi来解决这些问题的。（具体内容请参照书。）



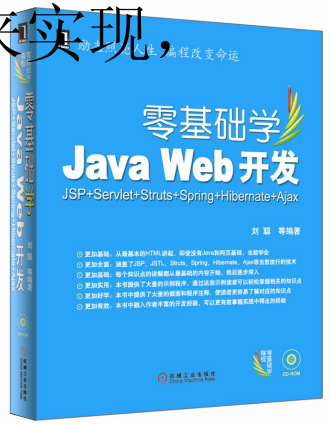
11.1.4　JSP读取Excel报表

- 相对于生成Excel文件，读取Excel文件的需求就小了很多。在这里我们只对这个功能做简单的介绍。（具体内容请参照书。）



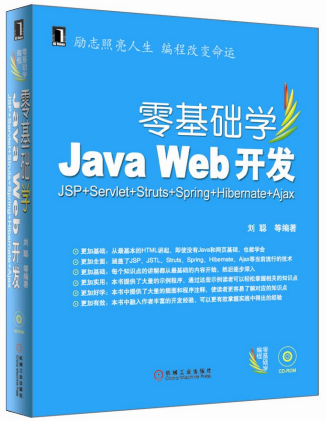
11.2　Java对图表的处理

- 在Web程序开发中，图形报表的功能几乎是比不可少的，很多场合下需要把表格结构的统计数据用图形形象的表示出来。Java自带的API中提供了图形图像操作的方法，对于一些简单的图表可以使用Java图形图像API来绘制所需的图形，但是如果要生成比较复杂、界面效果要求比较高的图形报表的时候采用Java API手动绘图的方法就不可取了，这是就要借助于第三方的解决方案来实现，接下来要介绍的内容就是



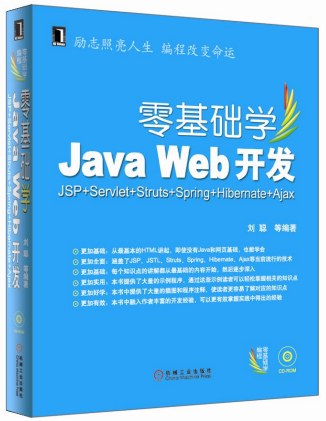
11.2.1　JSP图表工具汇总

- Java操作图表的可选解决方案有很多，例如JFreeChart，JavaReport等，这些工具的功能都非常强大。无论采用哪种都可以很方便的实现普通图形报表操作的功能。在本书中选用JFreeChart作为图形报表处理的工具。在以下的章节将详细讲述JFreeChart的使用方法。



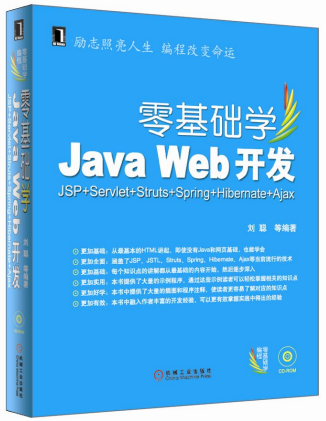
11.2.2　JFreeChart开发环境简单配置

- JFreeChart也是一个开源项目，（具体内容请参照书。）



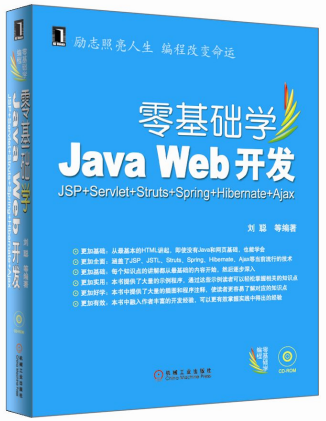
11.2.3　JSP生成简单二维柱状图

- 在柱状图的应用中，JFreeChart可以生成简单的二维图形，同时也可以生成三维的立体图形。在表现形式上不仅可以生成垂直的柱体，而且可以生成水平的柱体。下面将通过几个具体的例子详细展示这些功能的实现过程。（具体内容请参照书。）



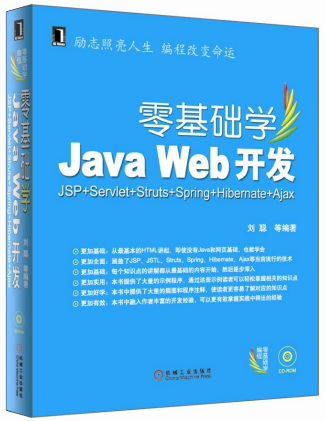
11.2.4　JSP生成简单三维柱状图

- 在很多情况下客户可能会要求实现三维的立体柱状图。这个时候需要做的仅仅是把二维柱状图做一点小小的改动即可实现由二维转换成三维的功能。在这里我们继续采用上面的数据情形，即要展示的数据还是学校的生源统计，无非是把平面的柱状图改变成三维立体的柱状图。现面是具体的改造过程。（具体内容请参照书。）



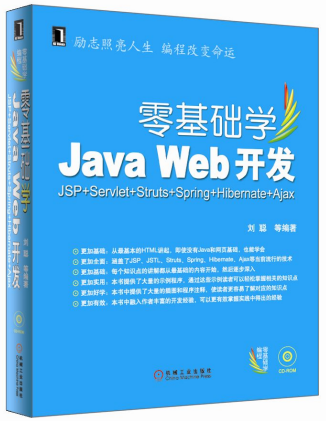
11.2.5　JSP生成水平方向的柱状图

- 在一些特定的应用领域，可能客户会要求实现水平方向的柱状图。这样的图形实现起来也是非常容易的。在这里我们依然采用学校生源分布的数据。不同的是要生成水平方向的柱状图。（具体内容请参照书。）



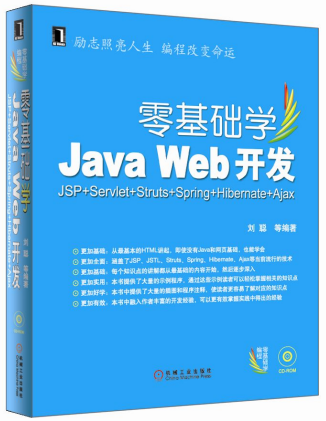
11.2.6　JSP生成多组柱体的柱状图

- 在前面介绍的柱状图中，所展示的数据只有一组柱体，然而在具体的应用中往往会遇到多组数据的情况，下面继续采用学校生源统计的例子，只不过在这里进行了细化，对于生源的统计添加了性别的指标。（具体内容请参照书。）



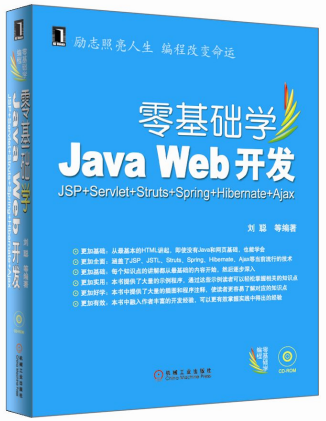
11.2.7　对柱状图柱体显示效果的具体设置

- 下面还采用学校生源统计的例子，数据依然采用表11.4中不分性别的情况。JFreeChart默认是可以自动调整柱体的宽度的。在这种情况下当柱体的数量比较少的时候每个柱体的会变得非常宽，这样就影响了这个报表的美观。而且默认的颜色和边框等设置并非最佳，这就需要我们自己来重新设置。



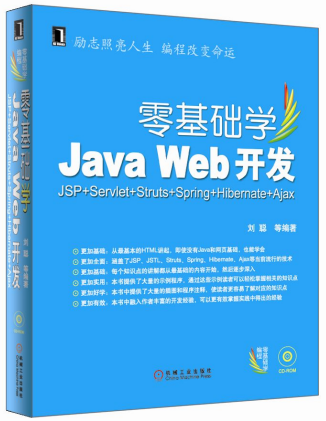
11.2.8　JSP生成一般的折线图

- 使用JFreeChart可以生成各种各样的折线图，例如一般的没有特殊要求的折线图、横坐标为时间序列、XY坐标轴都有刻度的折线图等。在这里近介绍几种常用的折线图的实现方法，其他更多折线图的实现方法可以参考这些例子，基本思路都是相同的。（具体内容请参照书。）



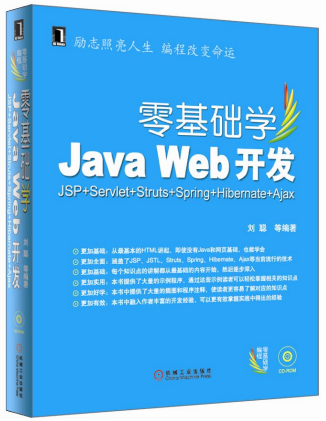
11.2.9　JSP生成横坐标为时间序列的折线图

- 在上面的折线图中，横坐标都是字符串，在JFreeChart中同样支持横坐标为时间序列的折线图。（具体内容请参照书。）



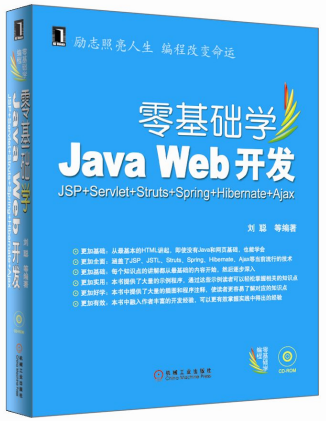
11.2.10　JSP在同一个报表中生成多条曲线

- 在有些时候需要在一个图表中同时展示多条曲线，（具体内容请参照书。）



11.2.11　JSP生成二维饼状图

- 饼状图在实际的开发过程中也是非常常见的，在JFreeChart中对饼图的支持也是非常好的。不仅可以生成二维的平面饼图，也可以生产那个三维立体的饼图。现面将对这两种饼图进行详细的介绍。（具体内容请参照书。）

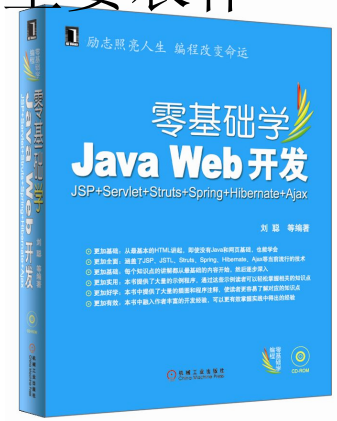


11.2.12　JSP生成三维饼状图

- 和柱状图一样，饼图同样也有三维展示的需要，和前面介绍二维柱状图转换成三维柱状图类似，在这里也仅仅需要对上面这个程序稍作调整即可。修改过程如下所示。

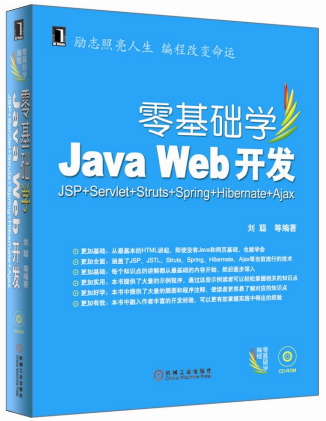
- JFreeChart jfreechart = ChartFactory.createPieChart("某地区主要农作物种植比例饼状图",

- piedataset, true, true, false);



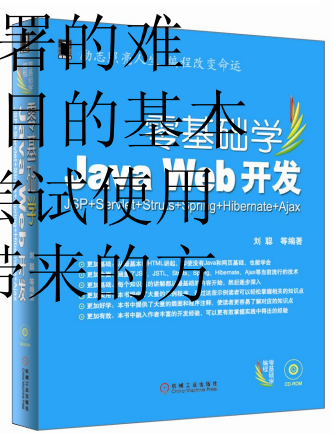
11.3 小结

- 在本章内容中，介绍了Java Web开发中报表开发的基本知识，在操作Excel文件的时候，本书选择使用JExcelApi来处理，对于图形报表的处理选择了JFreeChart。在上面的内容中主要介绍的就是这两种工具的具体使用方法，通过本章内容的学习，读者就可以掌握基本的Web报表开发知识，可以运用本章介绍的知识进行实际的报表处理工作。



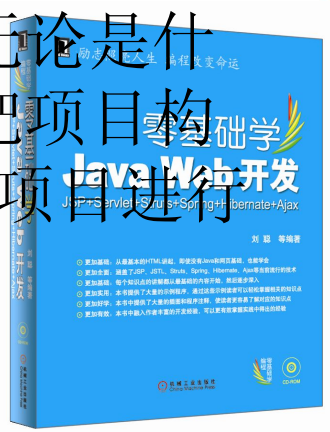
第十二章 学习使用Ant

- Apache Ant是一个基于Java的构建工具，它可以会构建多种项目，但是目前主要被用于Java项目的构建， Ant是用Java语言编写，可以在多个操作系统中运行，目前在绝大部分的Java开源项目中，都选择使用Ant作为构建项目的工具，Ant已经成为Java开源项目构建的事实标准，而且越来越多的开发人员选择了Ant来构建自己的项目，合理的使用Ant可以大大降低项目构建、部署的难度，在本章的内容中，将介绍Ant构建项目的基本用法，利用这里基本的知识，读者可以尝试使用Ant来构建自己的项目，体验Ant给我们带来的方便和高效。



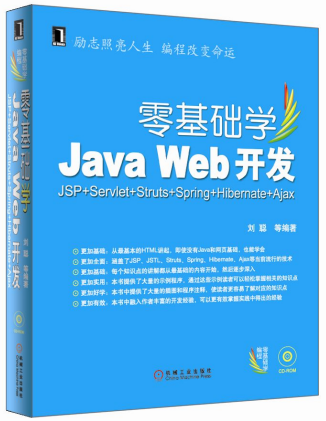
12.1　Ant简介

- 对于一般的Java项目来说，构建也就是对这个项目进行编译，　Ant就是针对这一系列问题而推出的，Ant不仅可以对Java源文件进行编译，而且还可以执行其他各种项目构建任务。使用Ant，仅仅需要编写一个构建配置文件，Ant会根据这个配置文件执行对应的操作，在这个配置文件中，可以指定构建、部署项目中的各种动作，而且还可以配置各种动作之间的依赖关系，因此，无论是什么类型的Java项目，只要在配置文件中把项目构建的要求说明清楚，就可以使用Ant来对项目进行构建、部署。



12.2　Ant的安装配置

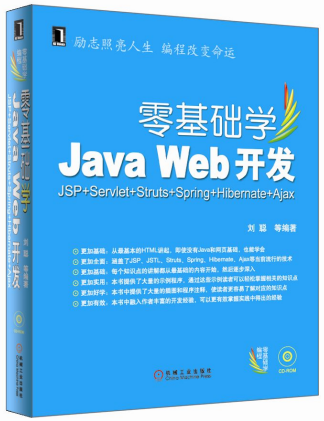
- Ant目前最新的版本是1.7，可以在Ant的官方网站下载，下载下来的文件为apache-ant-1.7.0-bin.zip，　Ant无需安装，直接解压然后设置环境变量即可。下面来设置Ant的环境变量。Ant环境变量的设置需要两个步骤，添加ANT_HOME和修改系统的Path变量，（具体内容请参照书。）



12.3 Ant简单示例——构建HelloWorld工程

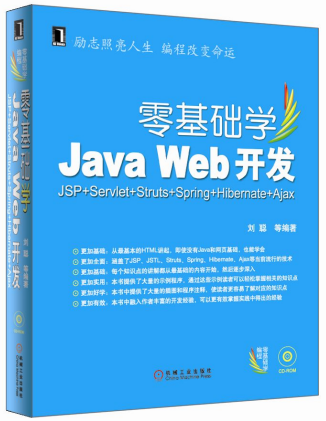
- 在本节中，将对一个HelloWorld工程进行构建，在这个工程中，有一个src文件夹，用来放置工程中的源代码，其中只有一个HelloWorld类的源代码。HelloWorld的代码如下。

```
//--------文件名：HelloWorld.java--------------------
--                                  public class HelloWorld{


    public static void main(String[] args){
                                System.out.println("Hello
world!");


    }
                }
```



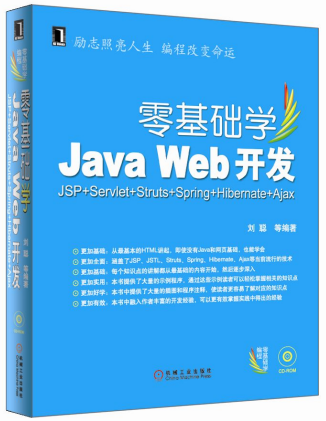
12.3.1　手工构建HelloWorld工程

- 在没有使用Ant之前，我们先采用手工的方式来构建这个HelloWorld工程，构建工程简单的说就是编译、打包、运行、部署等操作，而对于这个简单的HelloWorld工程来说，就没有部署这个步骤，下面将介绍手工在命令行中编译、运行、打包HelloWorld这个工程。（具体内容请参照书。）



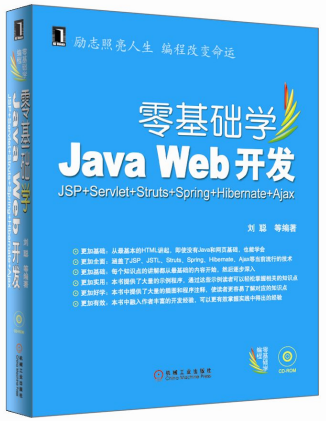
12.3.2　使用Ant分步构建HelloWorld工程

- 手工构建工程是相当麻烦的，这就促使我们要寻找一种构建工具，来辅助完成工程构建的任务，在这里我们选择使用Ant，Ant在构建工程的时候，只需要一个简单的脚本，在这个脚本中描述构建任务，Ant就可以根据这个构建描述文件完成构建工程的任务。



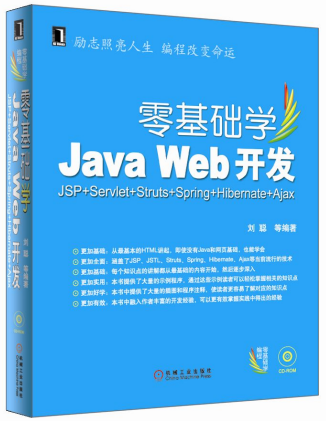
12.3.3　自动构建HelloWorld工程

- 在上节的示例中，我们使用Ant分步对HelloWrold工程进行构建，在本节的内容中，将使用Ant对HelloWorld这个工程那个进行自动构建。（具体内容请参照书。）



12.4　Ant构建文件解析

- 在以上章节的内容中，展示了Ant的使用方法，但是对于Ant脚本的具体内容并没有做详细的解释，在本节的内容中，将详细解释各个配置标签的具体含义和用法，通过本节内容的学习，读者可以掌握最基本的Ant脚本的简单语法。

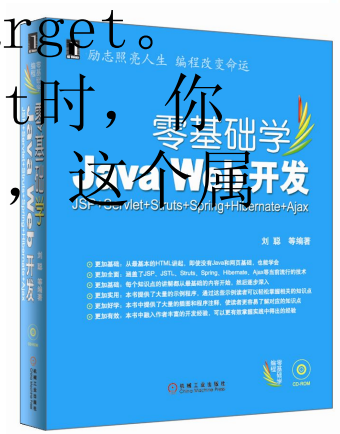


12.4.1 Project

- 在Ant脚本中，Project是这个XML文档的根节点，project节点有以下几个属性。

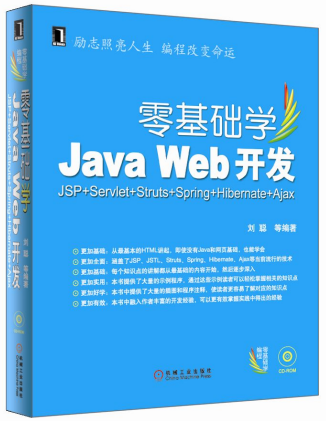
- name属性，这个属性指明构建任务的名称，一般情况下选择要构建的项目名称即可。

- default属性，一个项目可以定义多个target。target就是需要Ant执行的动作，执行Ant时，你可以选择执行哪个target。basedir属性，这个属性这个Ant脚本工作的根路径，



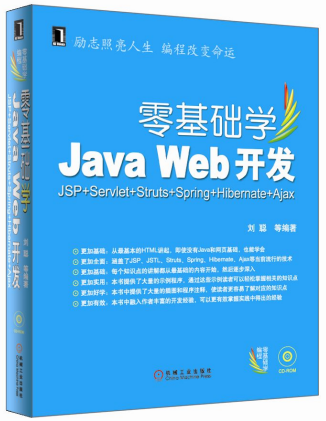
12.4.2 Target

- 在Ant脚本中，可以把想做的事情用target描述，每个target描述一件事情，target有以下几种常用的属性。



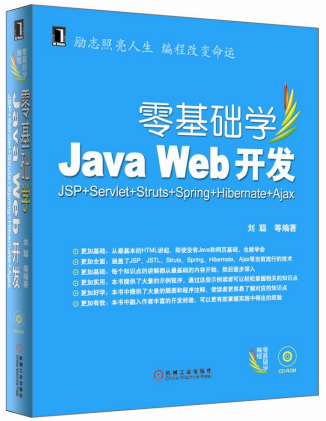
12.4.3 Properties

- 在一个Ant脚本中可以有多个属性，这些属性可以用property标签指定，也可以在Ant脚本之外使用properties属性文件描述。在本节中着重介绍使用property标签指定的情况，使用properties属性文件的情况在本书的12.6.1小节中将会详细介绍。（具体内容请参照书。）



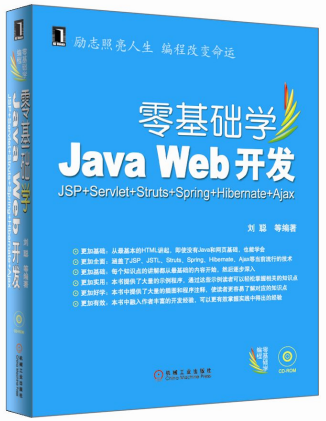
12.4.4 Classpath

- 在Java程序的开发和部署过程中，经常遇到的问题就是classpath的设置问题，在使用Ant构建工程的时候，不可避免的要涉及到classpath的问题，例如在程序中使用到第三方的类库，那么在使用Ant构建这个工程的时候，必须指明这个第三方类库的classpath，否则工程的编译工作就不能通过。（具体内容请参照书。）



12.5　Ant中的文件操作

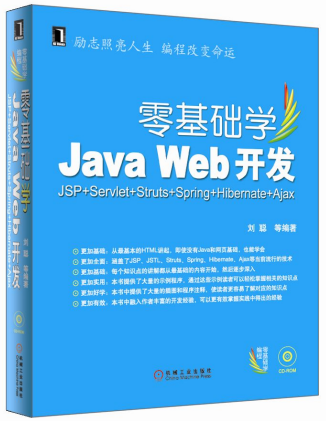
- 在构建过程中，通常会遇到很多的文件操作，像创建目录、拷贝文件或者目录这种工作是比不可少的，在Ant中，提供了一系列命令类支持文件的操作，从而可以方便的实现工程构建过程中的文件操作需要。在接下来的内容中，将介绍Ant文件操作的基本使用方法。



12.5.1　创建目录

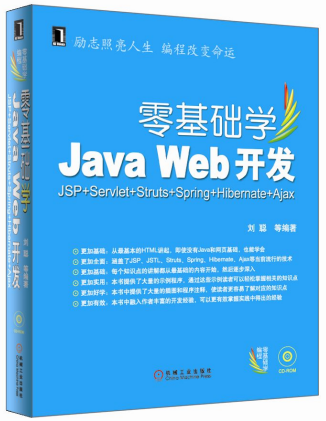
- 在Ant中，创建目录可以使用mkdir指令，这个指令可以创建指定的目录，如果这个目录的父目录不存在，会被同时创建。这个指令的使用方法如下。

- <mkdir dir="build/classes"/>



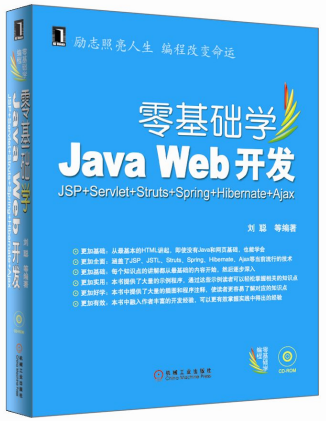
12.5.2　拷贝文件或者目录

- 在构建部署项目的时候，尤其是部署Web项目的时候，需要把构建得到的文件拷贝到目标位置，例如在Web项目中，需要把构建得到的文件部署在Web服务器下，也就是把构建得到的文件拷贝到Web服务器的指定目录下。在Ant中，可以使用copy命令拷贝文件或者整个目录，copy命令的具体基



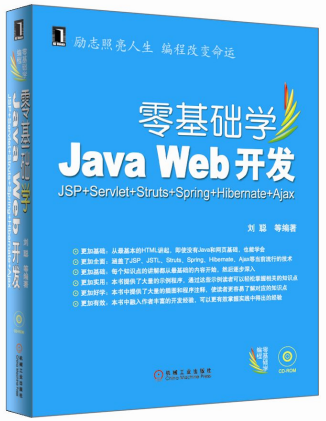
12.5.3　移动文件或者目录

- 在Ant中，同样可以移动文件或者目录，实现这个功能的命令是move，这个命令的基本用法有以下几种。



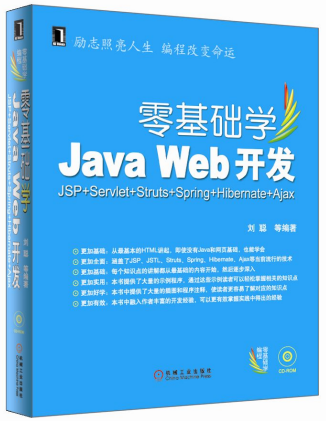
12.5.4　删除文件或者目录

- 在移除工程或者是重新部署工程的时候，都需要删除以前构建生成的文件，在这种情况下，可以使用Ant提供的删除命令delete来完成这个任务。



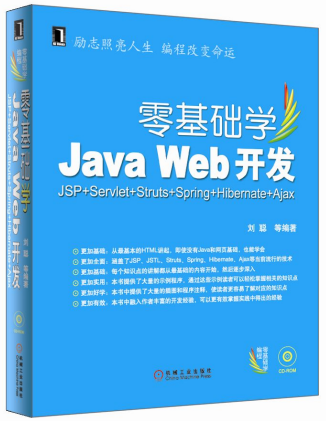
12.6　在Ant中使用属性配置文件

- 在Ant脚本中，同样可以使用配置文件类，例如连接数据库的配置信息，或者是构建工程过程中用到的一些属性，使用配置文件统一管理这些信息以后，在这些配置信息发生改变的时候，只需修改配置文件即可，而不用修改Ant脚本，是Ant脚本可以重复使用，这样就方便了项目工程的构建。（具体内容请参照书。）



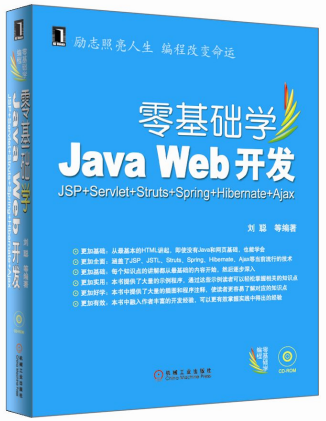
12.7　在Ant中执行数据库脚本

- 目前应用的开发过程中，大部分都离不开数据库的支持，在部署这种需要数据库的应用项目的时候，都需要初始话数据库，在一般情况下需要执行一个数据库脚本，用来完成那个建表、初始化数据等工作，在Ant中，同样执行这样的功能，我们可是使用Ant的sql直接执行数据库脚本，从而完成数据库初始化的工作。



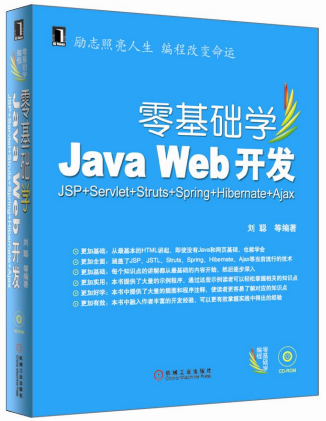
12.7.1　直接执行数据库脚本

- 在Ant中执行数据库脚本需要使用sql指令，（具体内容请参照书。）



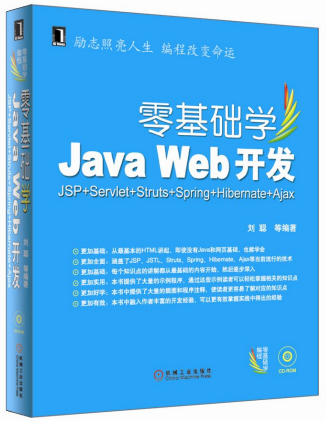
12.7.2　使用配置文件执行数据库脚本

- 在上面的示例脚本中，我们直接把数据库的连接信息放在Ant的sql指令中，这样处理以后，如果要更换数据库用户名或者是密码，就需要修改Ant脚本，为了是Ant脚本有更好的通用性，在这里我们使用配置文件提供数据库的连接信息。



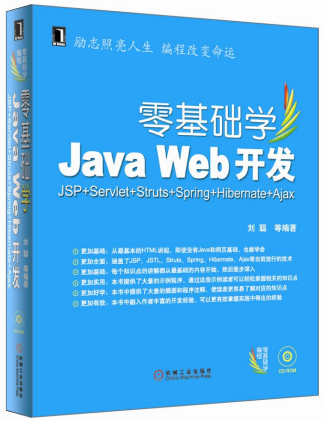
12.8　Ant构建部署Web应用综合示例

- 在前面的章节中，介绍了Ant构建工程的基本知识，在本节接下来的内容中，将通过一个具体的Web应用（假设这个Web应用的项目名称就是HelloWorld），展示使用Ant构建类似工程的具体操作方法。具体内容包括属性文件的配置、执行数据库初始化脚本、配置数据库连接池、配置classpath、编译部署工程等内容。



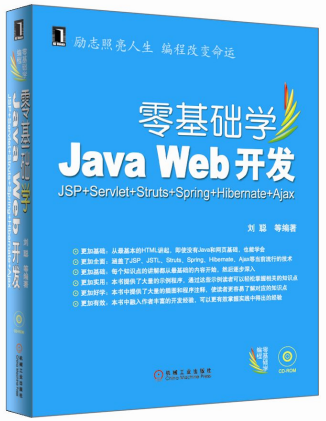
12.8.1　属性配置

- 在我们要部署的这个Web应用项目中，执行数据库脚本需要数据库连接的信息，还有在生成数据库连接池配置文件的时候，同样需要使用数据库的配置信息，在这种情况下，如果更改数据库配置信息势必要修改Ant脚本，为了提高这个Ant脚本的适应性，在这里选择使用属性配置文件来提供数据库的配置信息，这个配置文件和前面使用的内容是一样的，具体配置信息如下。



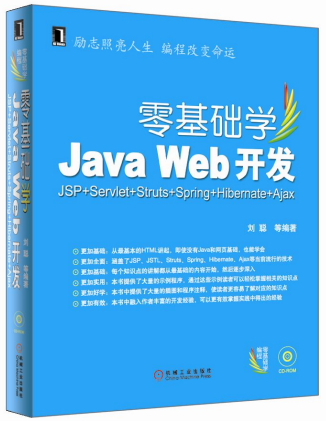
12.8.2　执行数据库初始化脚本

- 在部署项目之前，初始化数据库是必需要做的一项工作，例如在我们要部署的这个Web项目中，提供了一个数据库脚本，这个数据库脚本文件的名称为initate.sql，这个数据库脚本的内容是创建相关的数据库表格，并且初始化系统的基础数据。



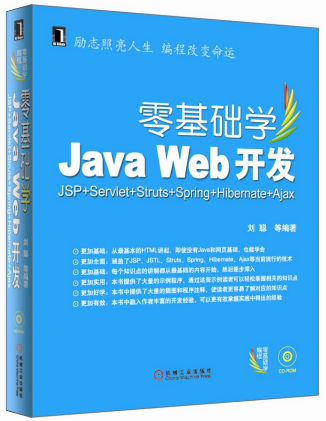
12.8.3　配置数据库连接池

- 在我们要部署的这个项目中，需要使用的数据库连接池的功能，而且我们选择在tomcat的conf\Catalina\localhost目录下单独配置数据库连接池的配置信息，



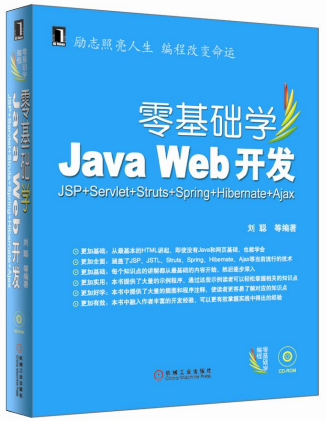
12.8.4　配置path

- 如果在项目中使用到了第三方的类，在编译部署工程的时候都需要指明路径，尤其是在部署的时候，需要把指定的第三方类放在固定的文件夹中，这样就需要使用到这些类库的路径信息，下面这段Ant脚本中，就定义了一个id为project.class.path的路径，在后面的脚本中可以根据id来引用这个path的信息。



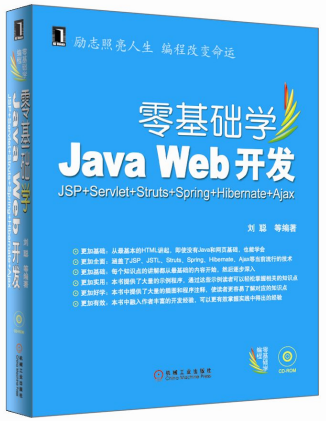
12.8.5　编译工程

- 在前面的准备工作完成以后，就可以对工程进行编译，编译工程的任务就是编译src中的所有java源代码，并把编译得到的所有class文件放在WebRoot\WEB-INF\classes目录下。



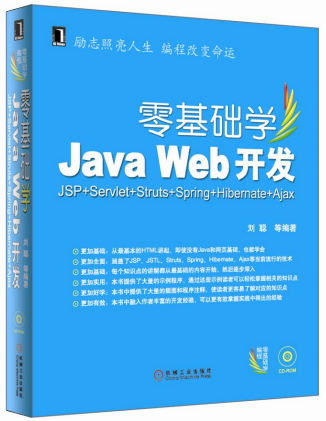
12.8.6　部署工程

- 在完成了编译工作以后和数据库的初始化工作以后，就可以部署编译过的项目，其中部署项目的target的具体内容如下。



12.8.7　完整构建脚本文件

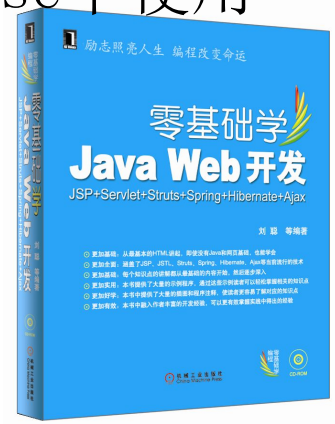
- 这个项目构建Ant脚本的完整内容如下所示。（具体内容请参照书。）



12.9　在Eclipse中使用Ant

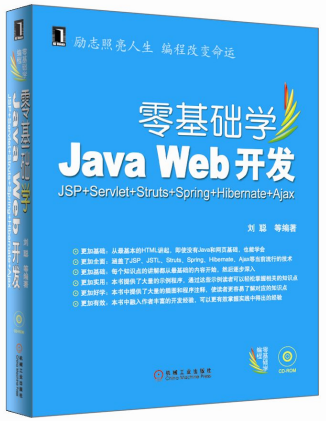
- Ant不仅可以在DOS命令行中运行，也可以在一些集成的开发工具中运行，其中在Eclipse中，即内置了Ant的功能，在Eclipse中，提供了Ant脚本的语法高亮显示，而且还可以非常方便的执行Ant脚本，并在控制台显示Ant脚本的运行信息。

- 在接下来的内容中，将简单介绍在Eclipse中使用Ant的基本方法。



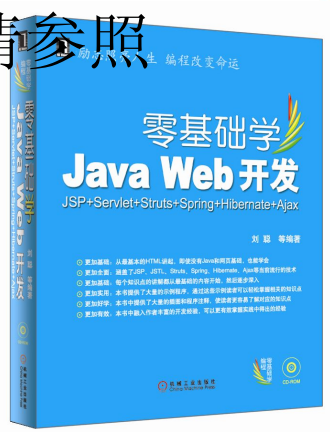
12.9.1　在Eclipse中编写Ant脚本

- 如果要在Eclipse中使用Ant，需要打开“Ant视图”，在Eclipse的菜单中选择“Window”|“Show View”|“Other”可以得到（具体内容请参照书。）



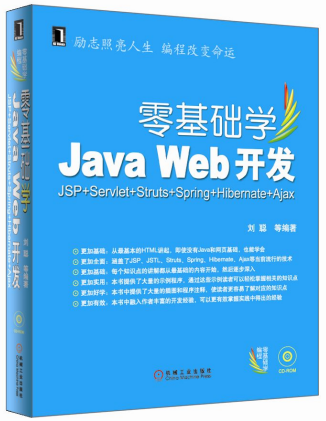
12.9.2　在Eclipse中运行Ant任务

- 在Eclipse中运行Ant任务也是非常方便的。在这里仍然以HelloWorld这个工程为例，使用的Ant脚本是12.3.3中自动构建工程的脚本。在Eclipse的文件目录中，右键选择build.xml，在弹出的菜单中选择“Open With”|Ant Editor，就可以在Eclipse集成的Ant编辑器中打开Ant脚本，同时在右侧的Ant视图中列出了各个target，而且默认执行的target使用蓝色表示，（具体内容请参照书。）



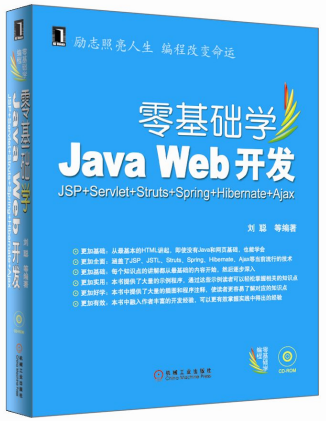
12.10 小结

- 在本章的内容中，介绍了Ant的基础知识和在实际项目构建中的用法，通过本章的学习，读者可以使用Ant自己构建应用项目，逐步适应脱离集成开发环境来构建部署自己的项目，虽然在这些开发环境中都会支持项目的构建和部署，但是为了适应不同的运行环境，提高项目部署的效率和速度，熟练使用Ant的功能是必不可少的。



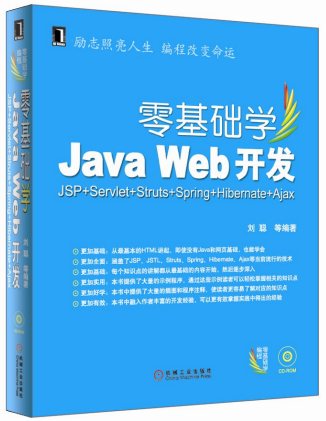
第十三章　Struts应用详解

- 在本章的内容中，将主要介绍Struts的基本知识，对Struts处理用户请求过程的各个环节进行了详细的介绍，并且还介绍在Struts中使用Validator、Tiles等其他框架的基本方法，通过本章内容的学习，读者可以掌握Struts的基本知识，具备Struts开发的基本技能。



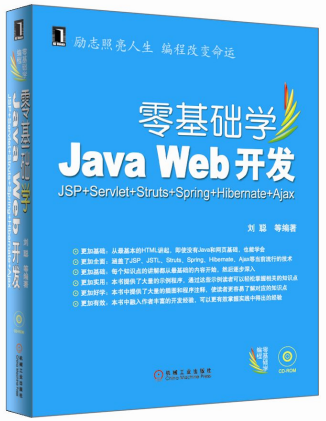
13.1 Struts基础知识

- 随着Web开发技术的日益成熟，在Web开发领域中出现了许多软件开发框架，Struts就是一种基于MVC经典设计模式的框架，也是目前Web开发中比较成熟的一种开发框架，要使用Struts框架，首先需要了解Struts的工作原理，在本节的内容中，将简单介绍Strut框架的工作原理，并且对Struts开发环境的配置方法做简单的说明



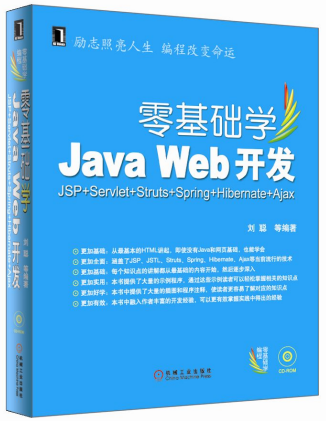
13.1.1 Struts 的工作原理

- Struts是对JSP Model2设计标准的一种实现，下面分别场区那个模型（Model）、视图（View）和控制器（Controller）三个部分介绍Struts的体系结构和工作原理，在一般情况下，Struts框架中的模型是由JavaBean或者EJB构成，视图是有JSP页面组成，控制器是由ActionServlet和Action实现。（具体内容请参照书。）



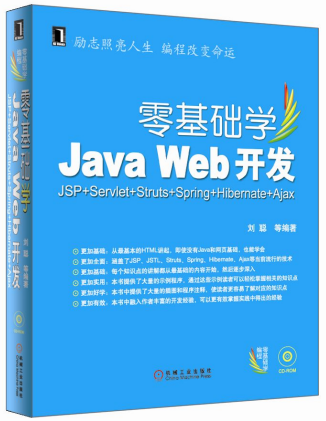
13.1.2　Struts的工作流程

- ActionServlet是Struts中核心的控制器，所有的用户请求都必须通过ActionServlet的处理，而struts-config.xml是Struts中核心的配置文件，在这个文件中配置了用户请求URL和控制器Action的映射关系，ActionServlet就是通过这个配置文件把用户请求发送到对应的控制器中。



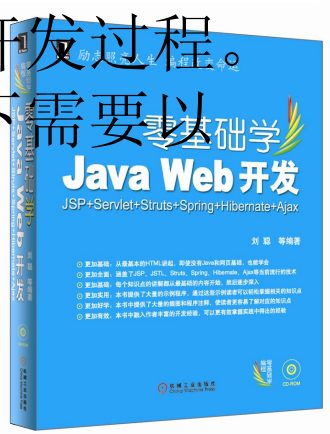
13.1.3　Struts的基本配置

- 在开发Struts应用程序的时候，需要对开发环境进行简单的配置，其中需要把Struts的类库放到项目WEB-INF/lib文件夹下，而且要把Strut的TLD标签库表述文件放到WEB-INF目录下，然后还需要在web.xml中配置ActionServlet这个控制器，从而保证所有的用户请求都能被Struts框架接收并处理。



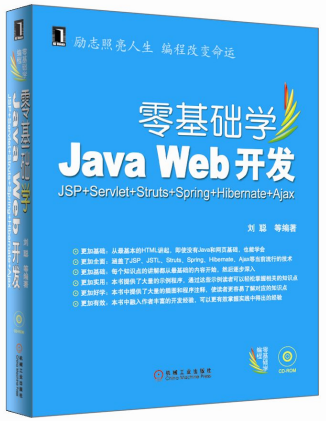
13.2　简单Struts应用示例

- 在本节内容中，将展示一个简单的Struts应用程序，在这个示例程序中，需要向服务器提交一个简单的表单，服务器接收这个表单以后，把处理的结果返回给用户。整个业务逻辑的处理非常简单，这个示例程序的目的也就是向读者展示Struts处理用户请求的具体流程，通过这个示例程序需要读者了解的是如何使用Struts处理用户请求，接下来介绍这个示例程序的具体开发过程。在开发这个示例程序的时候，一般情况下需要以下几个步骤。（具体内容请参照书。）



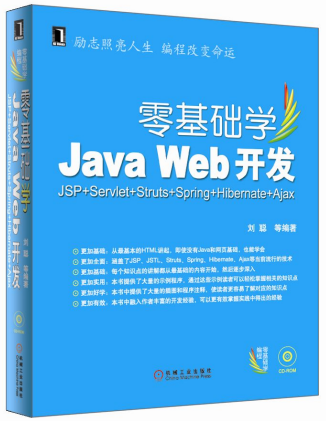
13.2.1　创建JSP页面

- 下面创建向用户提供表单输入的JSP页面，在这个JSP页面中，仅仅包含了一个简单的文本输入域，用来接收用户的输入，（具体内容请参照书。）



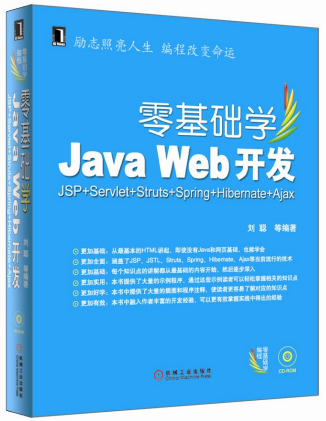
13.2.2　创建ActionForm

- 在上节内容中，创建了一个简单的JSP页面，在这个JSP页面中提供了一个简答的用户输入表单，在Struts中，需要为每一个用户输入表单提供一个ActionForm对象，当用户提交表单的时候，Struts会自动把用户提交的表单信息保存在对应的AcionForm中，在这个示例程序中，用户提交的表单仅有一条输入信息，（具体内容请参照书。）



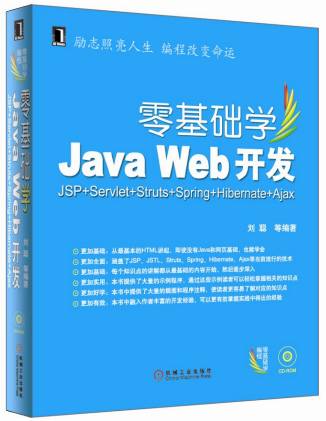
13.2.3　创建Action

- Struts中使用Action处理用户的请求，在这个示例程序中，可以使用下面的Action来处理用户提交的表单信息。



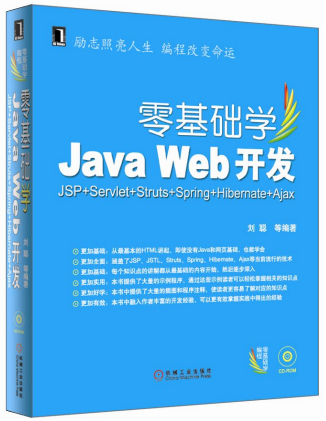
13.2.4　创建JavaBean模型组件

- 在上面的Action中，调用了一个JavaBean，在这里仅仅是为了模拟在Action调用JavaBean模型组件完成业务逻辑处理的方法，在这个JavaBean中并没有提供具体的业务逻辑，仅仅是展示使用，这个JavaBean的具体代码如下。（具体内容请参照书。）



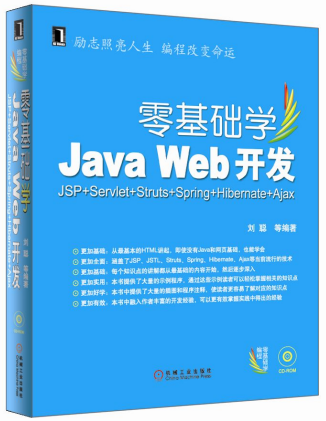
13.2.5　资源文件

- 在这个示例程序的JSP页面中，多次用到Struts中的资源文件，在本质上这个资源文件就是简单的属性文件，通过名值对应提供具体的配置内容，下面就是在这个示例程序中用到的资源文件。（具体内容请参照书。）



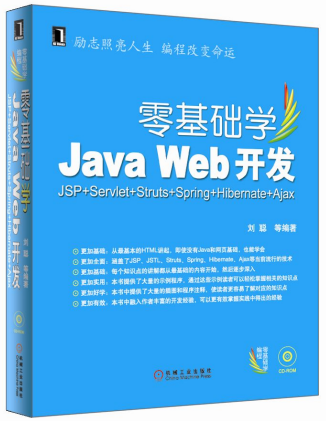
13.2.6　基本配置信息

- struts-config.xml是Struts配置文件中的核心配置文件，在这个配置文件中可以配置用户请求URL和控制器Action之间的映射信息，ActionServlet就是通过这个配置文件中完成用户请求的转发工作，下面就是这个示例程序中struts-config.xml配置文件的具体内容。（具体内容请参照书。）



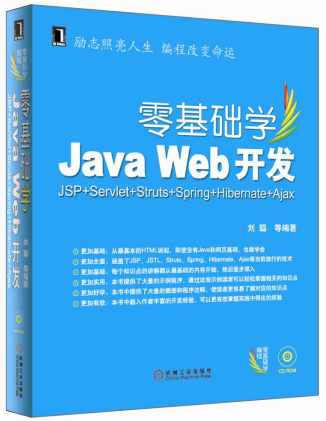
13.2.7　运行结果

- （具体内容请参照书。）



13.3 Struts中的表单处理器ActionForm

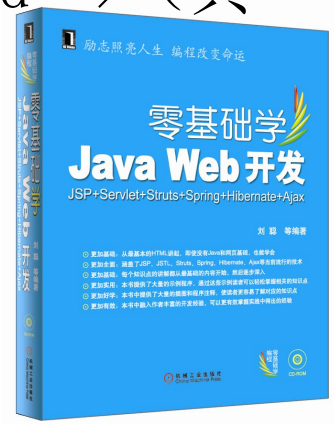
- 在Web应用程序的开发过程中，开发人员需要花费大量的时间和精力处理表单问题，有的时候是通过表单提交一些新的数据，有的是通过表单修改数据，所有这些表单的处理在传统的Web开发中都是非常复杂的。在本节中将介绍Struts中的表单处理组件ActionForm。



13.3.1 ActionForm简介

- 在传统的Web应用程序开发中，繁杂的表单处理工作给开发人员带来了巨大的困难，在传统的Web开发语言中，没有组件可以自动收集用户输入的表单内容，开发人员不得不在程序中手工提取表单的值。例如在表单中有这样一个文本输入域。

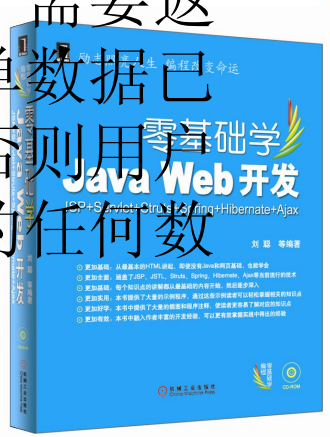
- <input type=”text” name=”password”>（具体内容请参照书。）



13.3.2 ActionForm基本功能

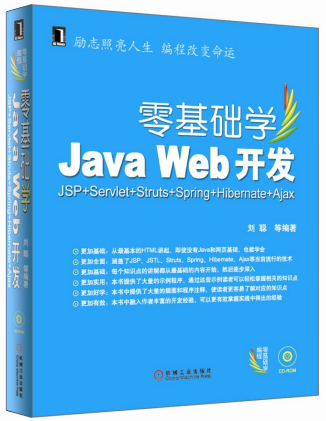
- ActionForm不仅实现了表单字段保存的功能，而且还提供了数据缓冲、数据验证的功能。

- 在传统的Web开发中，验证用户提交的表单数据可以采取两种方法，一种是在表单提交到服务器后，在服务器端使用JavaBean来进行验证，在这种验证方法中，如果数据验证失败以后，需要返回原来的页面，但是用户提交的所有表单数据已经被刷新，除非采用特殊的程序处理，否则用户得到的会是一个全新的表单，前面输入的任何数据都没有了。



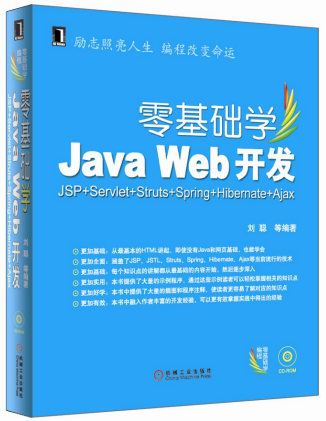
13.4 Struts中的控制器

- Action是Struts中的主要控制器，在本节的内容中，将介绍Action的工作原理和基本使用方法，同时对DispatchAction和LookupDispatchAction这两个控制器的使用方法进行简单的介绍，并通过两个示例程序展示这两个控制器的基本使用方法。



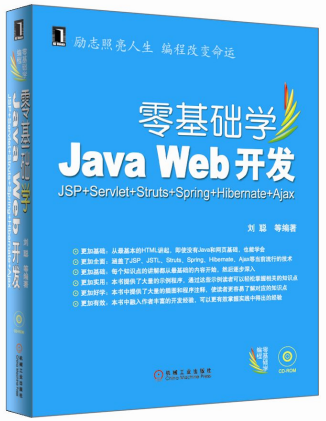
13.4.1　Action简介

- 在Struts中，所有的用户请求都会经过ActionServlet的处理，而实际的工作是交给Action对象来处理的，ActionServlet可以从配置文件中创建ActionMapping对象，并从ActionMapping对象中可以找到要使用的Action，然后将用户的请求转交给Action。



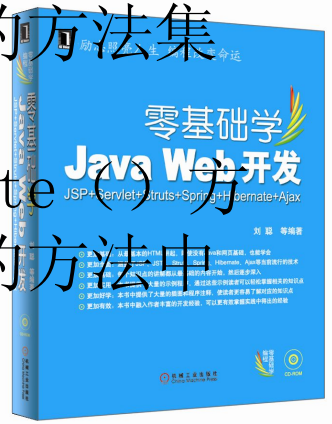
13.4.2　Action的基本使用方法

- 在开发Action的时候，需要继承org.apache.struts.action.Action这个类，在子类中加入所需的业务逻辑处理，这些子类会返回ActionForward对象，ActionServlet接受这个对象，把页面转发到指定页面，从而把用户请求的结果发送到对应页面。（具体内容请参照书。）



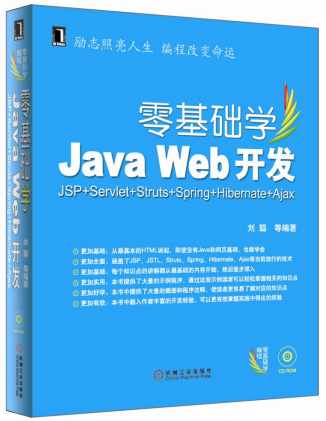
13.4.3 DispatchAction的使用方法

- 在前面的介绍中，为每一个动作提供一个Action类，但是在很多时候，一些相似的动作是可以在统一个模块中进行处理的，但是在Struts的Action类中，只提供一个execute（）方法，一个用户请求URL只能对应一个Servlet，在Struts中提供了另一个控制器类org.apache.struts.actions.DispatchAction，这个类可以将完成相关业务逻辑所需要的方法集中在一个DispatchAction类中，在继承DispatchAction类之后，不是重写execute（）方法，而是编写自己需要的方法，在不同的方法中处理不同的动作



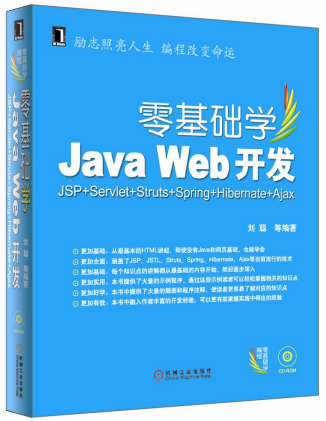
13.4.4 LookupDispatchAction的使用方法

- 在DispatchAction中可以在同一个控制器中处理多个动作，这个时候只能是通过URL调用控制器，控制器根据用户提交的参数决定调用哪个方法来处理用户的请求。这种情况下不能通过表单提交用户的请求信息。（具体内容请参照书。）



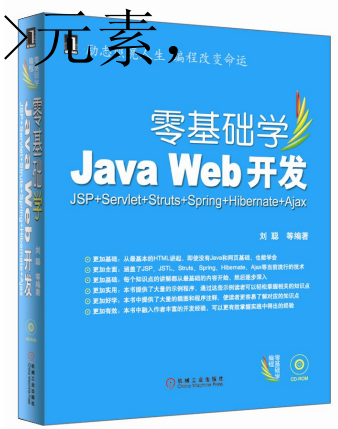
13.5 Struts中的页面转发控制

- Struts提供了ActionForward和ActionMapping这两个类用来控制页面转发，在本节内容中将简单介绍这两个类的基本使用方法。



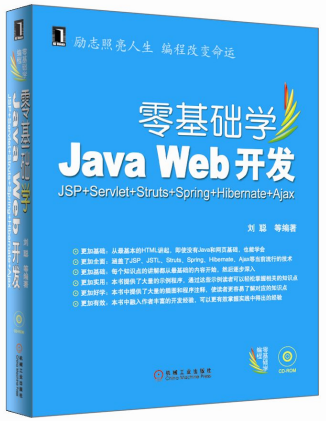
13.5.1 ActionForward简介

- 在使用Struts开发的Web应用程序中，Action在完成用户逻辑处理以后，需要把处理结果展示给用户，这个时候就需要程序控制页面的转发，在Struts中使用ActionForward对象控制程序的转向，ActionForward对象是一种配置对象，代表了一般的Web资源，可以是JSP页面、Servlet以及其他Action，ActionForward对象映射的是Struts配置文件struts-config.xml中的<forward>元素，在这个元素中封装了目标页面的URI。



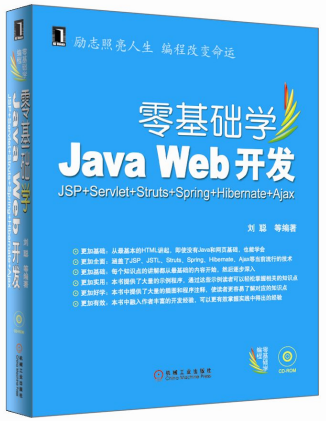
13.5.2　使用ActionForward传递参数

- ActionForward不仅承担着控制程序转发的任务，而且还可以在转发页面的时候同时传递参数，在ActionForward中，可以在struts-config.xml定义<forward>元素的时候指定参数以及内容。也可以在程序中动态添加参数。下面将介绍ActionForward传递参数的更多用法。



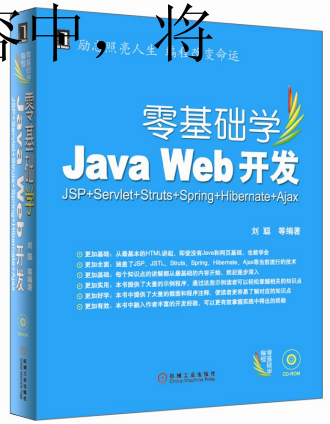
13.5.3 ActionMapping简介

- 在struts-config.xml配置文件中，每一个<action>元素都对应一个ActionMapping对象，当用户请求被ActionServlet接收以后，ActionServlet会根据用户请求URL以及<action>元素设定的path属性确定对应的ActionMapping对象，ActionMapping对象会告诉ActionServlet使用哪个Action对象处理用户请求。



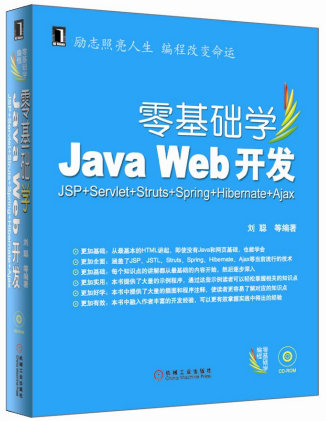
13.6 Struts标签库

- 为了方便开发人员使用Struts开发Web应用系统，在Struts框架中提供了内置的标签库，使用这些标签库可以方便构造表示层的JSP页面。在前面章节中介绍的示例程序中，已经使用到一部分Struts标签，虽然目前很大一部分的Web程序仍然采用传统的HTML标记来实现表示层的页面，但是在Struts应用的开发过程中，Struts标签的使用还是不可避免的，在本章的接下来的内容中，将简单介绍Struts中常用的基本标签库。



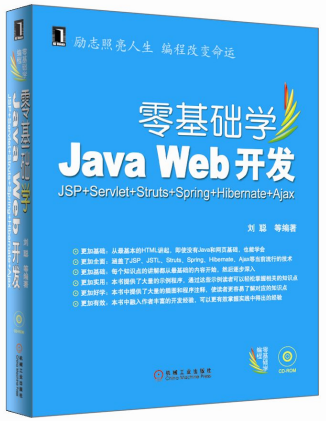
13.6.1　html标签

- HTML标签是Struts标签中基本的标签。Struts框架的HTML标签库中提供了对应普通HTML页面中的标签元素。接下来介绍Struts标签中的常用的标签。（具体内容请参照书。）



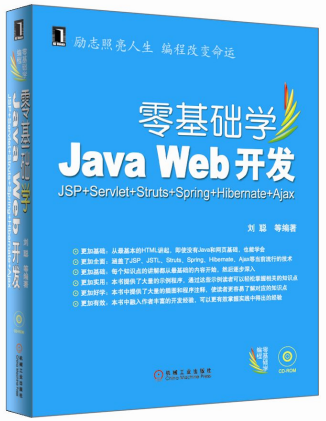
13.6.2　bean标签

- Struts的bean标签库用来在JSP页面中处理JavaBean，不仅可以访问已经存在的JavaBean，而且还可以定义新的JavaBean。接下来介绍bean标签库中常用的标签。



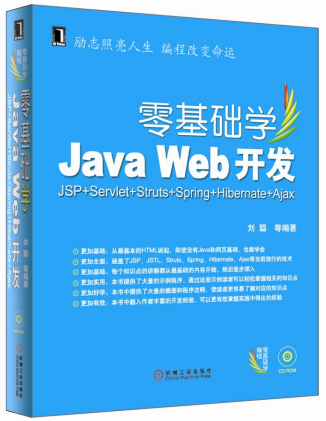
13.6.3　logic标签

- 在传统的处理方式中，是用脚本控制页面中的逻辑，在Struts中提供logic标签库控制页面的基本逻辑处理。接下来介绍logic标签库中常用的标签（具体内容请参照书。）



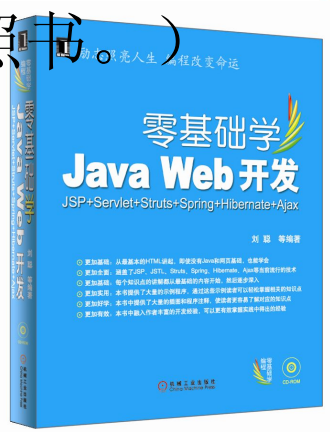
13.7　在Struts中使用Validator验证框架

- 在使用ActionForm时，可以把验证用户表单输入的工作放在ActionForm的vailidate（）方法中，在Struts中，可以整合Validator验证框架进行表单的输入验证工作。在接下来的内容中，将介绍Struts中使用Validator验证框架的基本方法。



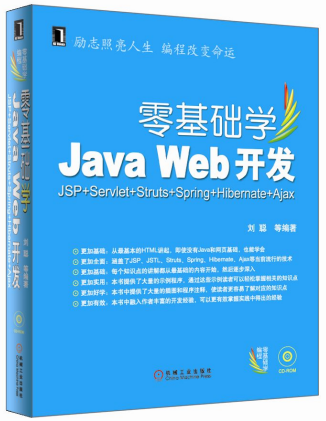
13.7.1　Validator验证框架的配置

- 在前面章节中使用MyEclipse配置Struts开发环境的时候，已经默认把Validator框架的类库文件拷贝到项目的WEB-INF/lib中，而且还把Validator框架需要的validator-rules.xml文件拷贝到WEB-INF目录中。同时如果要使用Validator，还需要在WEB-INF目录下添加Validator验证规则文件，在这里我们需要在WEB-INF目录下添加一个validation.xml文件。（具体内容请参照书。）



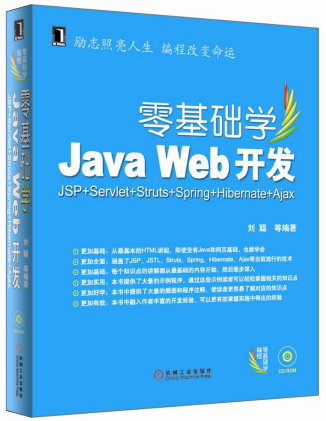
13.7.2 Validator的内置验证器

- 在Validator框架中内置了一些验证器，例如非空验证、Email验证、整型数据验证、最大长度验证、最小长度验证等，这些验证方法都是Validator框架中内置的，可以调用这些验证器对用户提交的表单进行验证。下面将简单介绍这几种内置验证器的基本使用方法。



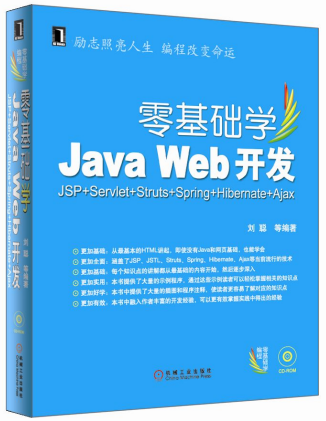
13.7.3 Validator验证框架的简单示例

- 在下面的示例程序中，展示了Validator中内置的几个验证器的基本使用方法，下面就是这个示例程序中的JSP页面，在这个页面中提供了一个简单的用户注册表单，在这个示例程序中将对用户输入的表单信息进行验证，（具体内容请参照书。）



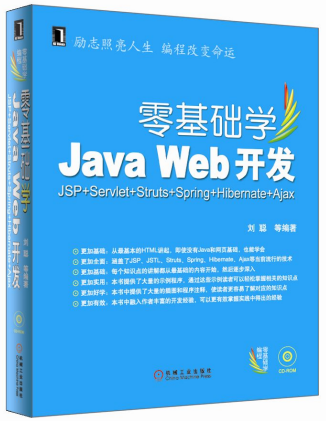
13.8　在Struts中使用Tiles

- Tiles提供一个类似桌面应用程序版面管理的机制，通过配置文件可以定义版面配置，以及其中需要插入的内容网页，在Tiles中内容网页可以和配置文件相分离，通过使用Tiles可以随时更换内容网页，从而可以重用版面配置文件。在下面的内容中，将介绍在Struts中使用Tiles框架的基本方法。



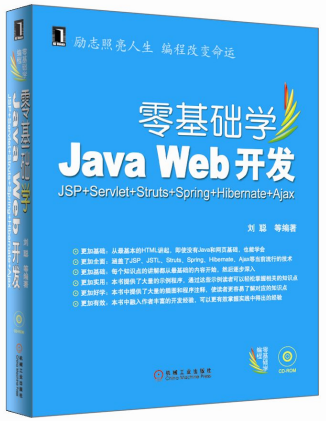
13.8.1　在Struts中配置Tiles框架

- 在Struts中使用Tiles框架的方法是非常方便的。在前面使用MyEclipse配置Struts开发环境的时候，已经自动吧Tiles需要用到的标签库描述文件struts-tiles.tld拷贝在项目的WEB-INF目录下，现在还需要在WEB-INF目录下添加一个tiles-defns.xml文件，在这个文件中描述了Tiles框架的配置信息。



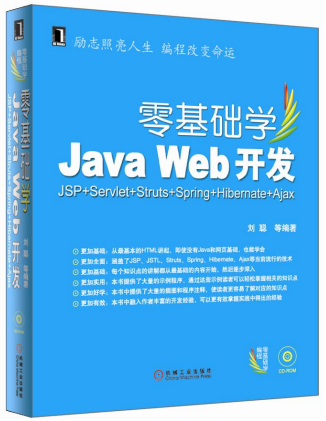
13.8.2　创建模版页面

- 在下面的示例程序中，将展示Tiles模版的基本使用方法，在使用Tiles框架的时候，首先需要创建Tiles模版，然后在需要使用这个模版的时候，直接调用即可。（具体内容请参照书。）



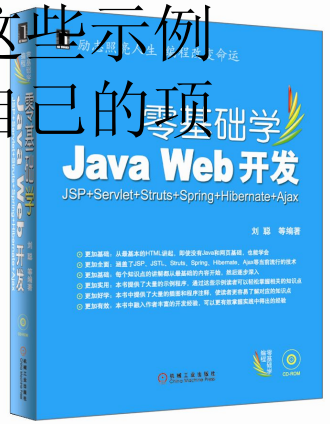
13.8.3　使用模版页面

- 在上一章节中，创建了个名为mainlayout的页面布局，定义好这个布局以后，就可以在后就可以在程序中使用这个布局，在这个布局中，已经指定了header和footer的内容，在使用这个布局的时候，只需要添加网页中间部分的主体内容即可。（具体内容请参照书。）



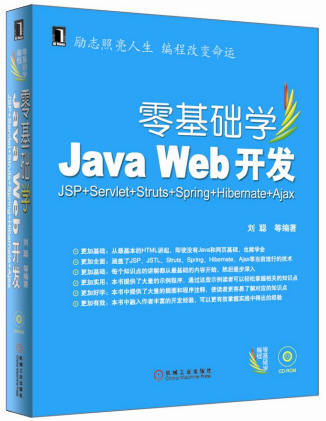
13.9 小结

- 在本章内容中，介绍了Struts的基本知识，在目前的Web开发中，Struts已经成了MVC事实上的标准大量的Web程序使用Struts进行开发，所以读者有必要掌握这项基本的技术，本章中不仅介绍了Struts的基本工作原理、基本使用方法，而且还介绍了在Struts中如何整合第三方的框架，另外，在本章的内容中，对于比较复杂的知识点都提供了具体的示例展示，读者可以通过这些示例程序深入了解对应的知识点，并逐步在自己的项目中学习使用。



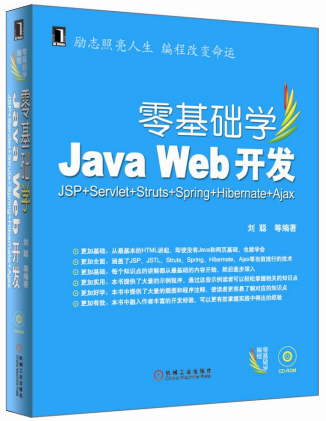
第十四章　Spring应用详解

- 在本章的内容中，首先介绍Spring核心技术控制反转和面向切面编程的基本知识，在Web方面，详细介绍了如何使用Spring实现MVC，然后对Spring中的数据库操作进行简单的介绍，在本章的最后，展示了如何在Spring中进行事务处理，通过本章内容的学习，读者可以从整体上了解Spring的基本知识，通过具体示例程序的学习，读者可以逐步学习编写自己的Spring程序。



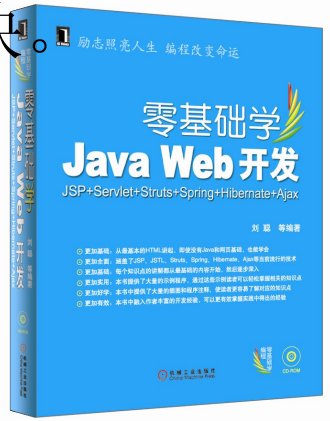
14.1　Spring简介

- Spring是一个开源框架，是为简化企业级应用系统开发而推出的，通过使用Spring，用户可以用简单的Java Bean实现以前只能用EJB才能完成的任务，虽然Spring是为企业级应用推出的，但是所有的Java系统开发都可以使用Spring，包括桌面应用程序和企业级的Web应用，在本节中，将对Spring做一个大体的介绍，关于Spring的核心技术和使用方法在后面的章节中进行介绍。



14.1.1　Spring简介

- Spring是一个轻量级的框架，Spring所耗费的系统资源开支比较少，而且Spring是非侵入失的，在一般情况下，引入Spring的系统中，具体的对象并不依赖于Spring的API。在Spring中，提供了对反转控制（IoC）和面向切面编程（AOP）的良好支持，Spring是由以下几个模块组成的，这些模块提供了开发企业级应用所需要的基本功能，可以在自己的程序中选择使用需要的模块。

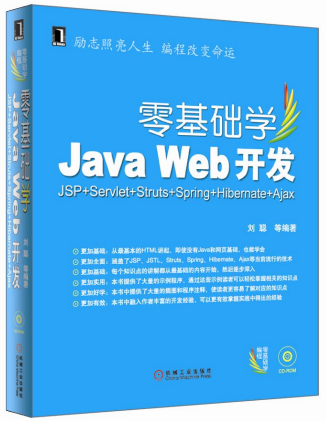


14.1.2　Spring开发环境的简单配置

- Spring开发环境的配置方法比较简单，可以下载Spring的开发包，把下载下来的压缩文件解压到硬盘中，然后把Spring.jar和其他相关类库加入项目即可。另外，还可以在Eclipse（已经安装MyEclipse插件）中配置Spring的开发环境，在Eclipse中新建一个项目，Java Project或者Web Project均可，这个可以根据需要而定，新建工程完成以后，在Eclipse的菜单栏中选择“MyEclipse”|“Capabilities”|“Add Spring Capabilities”就可以进入添加Spring模块的界面，在这个界面中选择所需的Spring模块的类库即可。

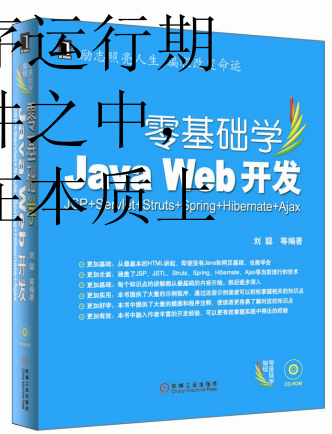
14.2　Spring核心理论控制反转介绍

- Spring框架本身提供了很多的功能，这些功能之所以能组合成为一个整体，就是因为使用了控制反转技术，控制反转是Spring的核心技术之一，在本小节中，将介绍控制反转的基本知识，同时详细展示了如何在Spring中实现控制反转。



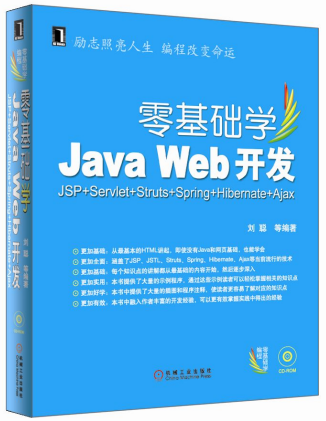
14.2.1　控制反转基础知识

- 在学习Spring的时候，往往会遇到控制反转（Inversion of Control）、依赖注入（Dependency Injection）这些新名词，IoC也就是由容器控制程序之间的关系，而不是在程序中直接使用代码控制，控制权由程序代码转移到外部容器，控制权的转移就是所谓的反转，这就是控制反转（IoC）的本质含义。 由于程序组件之间的依赖关系是有容器控制的，在程序运行期间，是由容器动态将依赖关系注入到组件之中，这就是依赖注入的本质含义。依赖注入在本质上也就是控制反转的另一种解释。



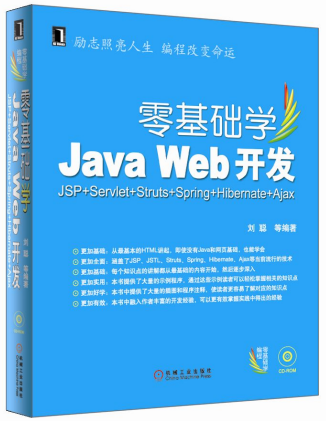
14.2.2　控制反转简单示例

- 在介绍Spring控制反转的具体内容之前，先展示一个具体的示例程序，通过这个示例程序，读者可以提前体验使用控制反转带来的便捷之处。在下面这个示例程序中，通过控制反转实现了问候用户的基本功能。（具体内容请参照书。）



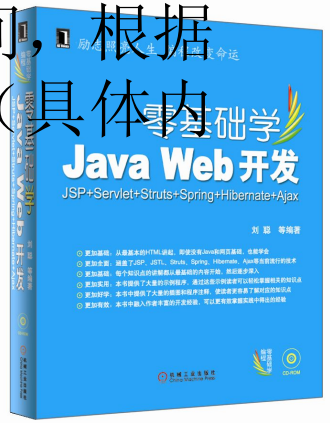
14.2.3　Spring中注入依赖的方法

- Spring中对象之间的依赖是由容器控制的，在程序运行期间，容器会根据配置文件的内容把对象之间的依赖关系注入到组件中，从而实现对象之间的协同工作。在Spring中，注入对象之间依赖关系的方式有以下几种。（具体内容请参照书。）



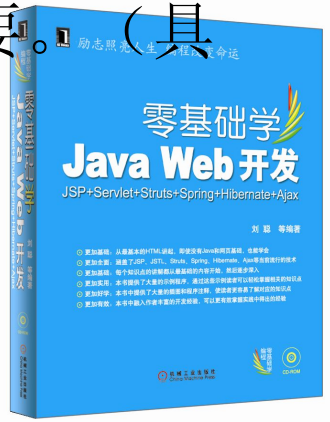
14.2.4　Spring中通过赋值方法注入依赖

- 在接下来的内容中，将简单介绍Spring中注入依赖的几种方法，在上面的简单的Spring依赖注入的示例中，并没有展示复杂的注入特性，仅仅是在程序的运行期间，把Spring IoC这个字符串赋值给Welcome和Bye这两个Bean中的name属性，同时在Welcome和Bye这个两个Bean中还提供了name属性的getter和setter方法，Spring可以调用Bean中的属性设置方法，在程序运行期间，根据配置文件中的内容给Bean的属性赋值。（具体内容请参照书。）



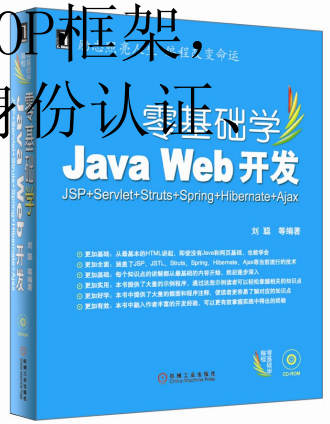
14.2.5　Spring中通过构造器注入依赖

- 赋值注入的方法虽然非常方便，但是也有自身的缺点，在赋值注入的方式中，无法确定哪些属性是必需的，哪些属性是可选的，这样就容易造成初始化bean的时候，有的属性可能并没有被正确设置，而在Spring中，提供了构造器注入依赖的方式，这种方式在构造器中强制需要初始化的方法，而且还可以指定初始化参数的顺序，这种方式可以满足一些赋值注入无法实现的需要。（具体内容请参照书。）



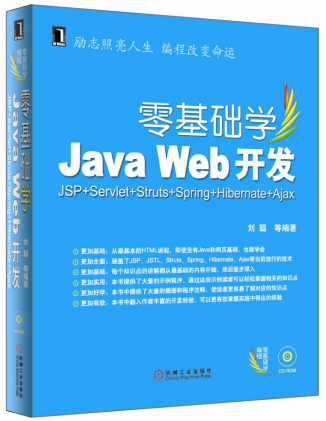
14.3　Spring核心理论面向切面编程介绍

- 面向切面编程（Aspect Oriented Programming即AOP）是Spring中的另一中核心技术， AOP提供另一种角度来思考程序结构，通过使用AOP可以给面向对象编程提供强大的辅助功能，在Spring框架中，提供了对AOP的支持，Spring的AOP框架允许将分散在系统中的模块集中起来，通过AOP中的切面实现，并通过Spring中强大的切入点机制在程序中随时引入切面，通过使用Spring的AOP框架，就可以给系统中添加强大的服务，例如身份认证、声明式事务管理等服务。



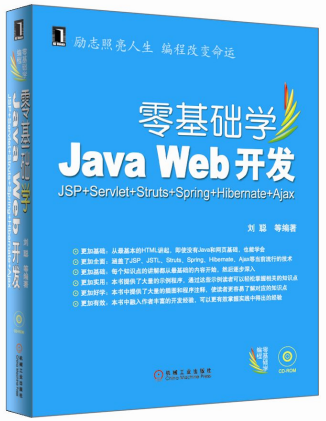
14.3.1　面向切面编程基础知识

- 在一般的应用系统中，会通过多个模块实现系统的总体功能，每个模块的主要功能是提供不同的业务逻辑，但是这些模块会需要一些相似的底层功能的支持，例如安全、用户身份认证、事务处理等。如果在各个模块中都通过代码来调用这些底层功能，就会是调用代码分布在系统的各个角落，从而增加了系统的耦合性，给后继的维护升级都带来很大的潜在困难。



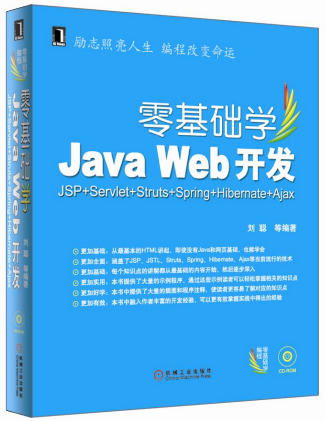
14.3.2　在Spring中创建前置通知

- 在下面的实例程序中，将展示Spring中创建前置通知的基本方法，下面Customer这个类中，实现了最基本的buy（）方法，在这个方法中相控制台打印购买商品的信息。（具体内容请参照书。）



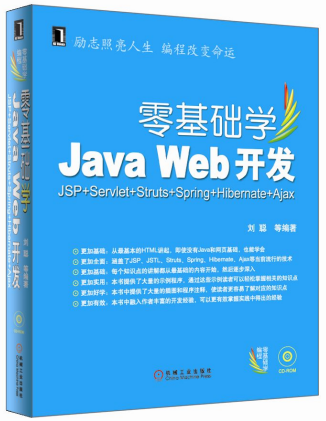
14.3.3　在Spring中创建后置通知

- 在上面的示例程序中，展示了Spring中前置通知的使用方法，在下面的实例程序中，将展示在Spring中创建后置通知的基本方法，在这里仍然使用Customer这个bean作为目标对象，在顾客执行购买商品的动作之后，在控制台打印告别信息。（具体内容请参照书。）



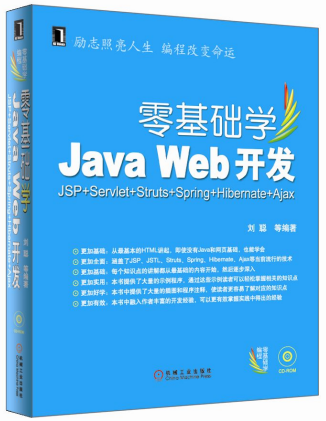
14.3.4　在Spring中创建拦截通知

- 在上面两个实例程序中，分别展示了Spring中前置通知和后置通知的具体实现方法，在实际的开发过程中，也会有创建拦截通知的需要，也就是在方法执行之前和方法返回之后都能够进行特殊的处理，这在Spring中就是拦截通知。（具体内容请参照书。）



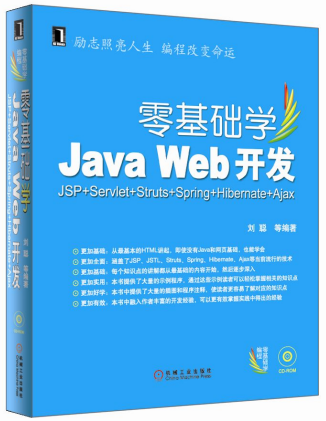
14.3.5　在Spring中创建异常通知

- 在上面的示例程序中，展示的都是程序正常运行的时候附加的通知，在Spring中，也提供了异常通知，这种通知会在目标对象抛出异常的时候被调用，下面是在Spring中创建异常通知的具体方法。（具体内容请参照书。）



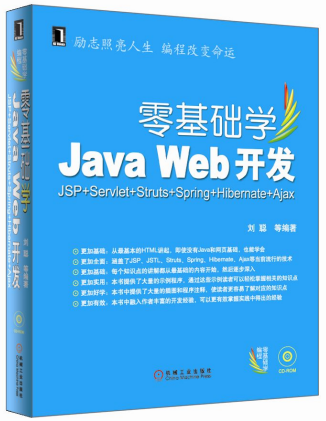
14.3.6　使用Spring静态切入点

- 在上面展示的各种通知中，使用的就是静态切入点把通知应用在目标对象上，但是在上面的示例程序中，在目标对象中，只有一个方法，所以就不用指定把通知应用在哪个方法中，Spring会把通知应用在这个唯一的方法上。



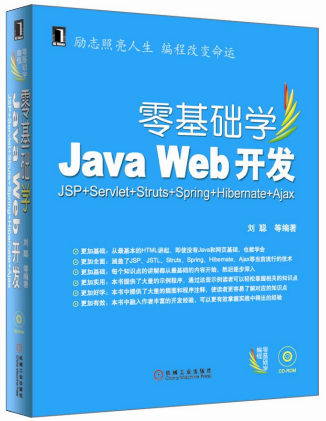
14.4　Spring实现MVC

- 虽然在Spring中可以非常方便的与其他MVC框架集成，例如Struts、WebWork等都可以集成在Spring中，而且在Spring中也实现了自身的MVC框架，在Spring的MVC框架中，可以透明地将Web参数绑定到业务对象中，　同时在Spring中还可以使用现存的多种视图技术，Spring解决了传统MVC框架中的不足。



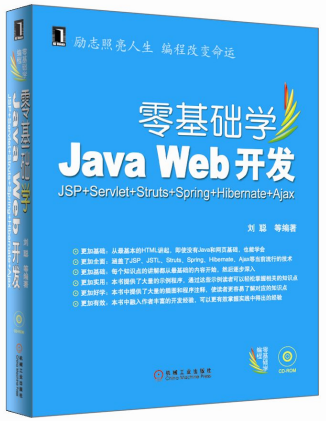
14.4.1　Spring中MVC的实现原理

- Spring MVC的核心组件是DispatcherServlet，这个Sevlet是Spring的前端控制器，DispatcherServlet和其他普通的Servlet一样，需要在web.xml中进行配置，下面就是一个DispatcherServlet的基本配置，具体代码如下。（具体内容请参照书。）



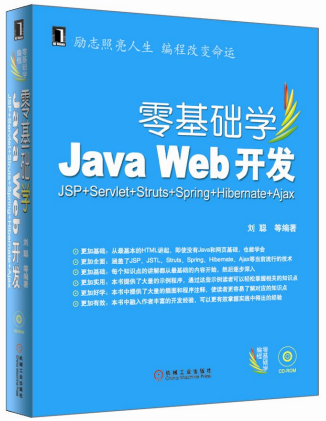
14.4.2　Spring中简单MVC示例

- 在本节中，将通过一个简单的示例程序，用来展示Spring MVC的基本处理流程，在上面的章节中，介绍了Spring MVC的基本流程，在这里我们先辨析控制器的逻辑，在这个示例程序中，控制器的具体代码如下。（具体内容请参照书。）



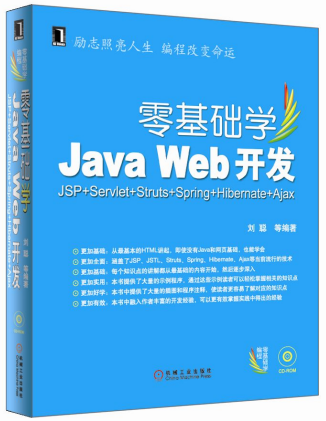
14.4.3　把用户请求映射到控制器

- 在Spring中还提供了另一种映射处理器，即SimpleUrlHandlerMapping，这个映射处理器的使用也非常简单，例如在下面的配置示例中，有两个控制器bean，分别对应着两个不同的用户请求URL，这就可以使用SimpleUrlHandlerMapping来进行处理，具体的配置过程如下。（具体内容请参照书。）



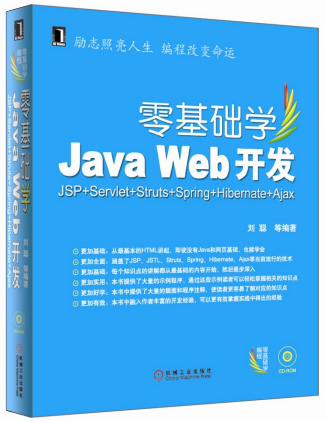
14.4.4　在控制器中处理带参数的用户请求

- 通过上面章节中配置的映射处理器，可以把用户的请求URL映射到控制器中，然后控制器就可以根据用户请求的信息进行相应的处理，在本节中将介绍Spring MVC中处理用户请求参数的具体方法。（具体内容请参照书。）



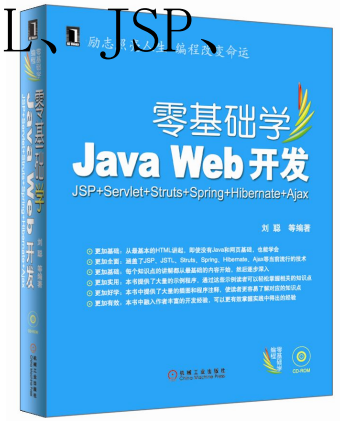
14.4.5　在控制器中处理简单的表单

- 在Spring MVC中，提供了处理表单的处理器，不仅可以处理简单的表单，而且可以提供复杂表单的向导，在下面会展示Spring中处理简单表单的基本方法。（具体内容请参照书。）



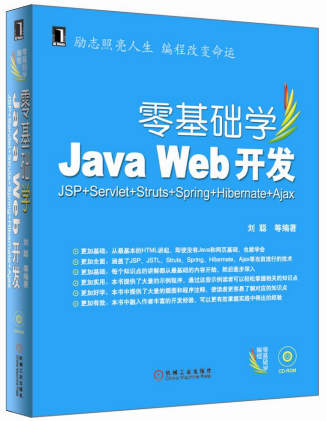
14.4.6　解析视图

- 在上面的示例程序中可以看出，Spring MVC的控制器在处理结束之后，会把处理的结果用ModelAndView对象的形式返回，这个对象是不能直接展示给用户的，需要通过视图解析器的解析，把这个对象中的信息提取出来以后在展示给用户。上面的示例程序中，仅仅使用jsp模版来展示ModelAndView对象的信息，其实在Spring MVC中提供了很多中视图展示技术，例如JSTL、JSP、Velocity等，这些内容可以参考Spring reference，在这里限于篇幅不再赘述。



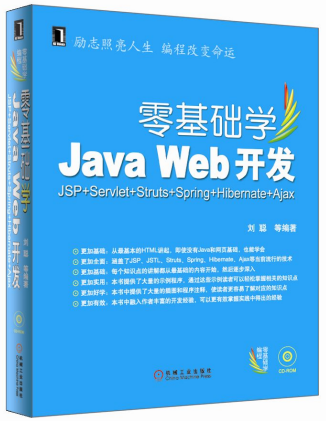
14.5　Spring中的数据库操作

- 数据库操作是Spring独具特色的地方，再Spring的数据库操作中，不用再担心数据库资源释放的问题，Spring中提供了常用的数据库操作模版，用户可以专注书写自己的数据库操作业务代码，而不再过多考虑数据库连接的取得与释放。而且Spring还可以非常方便与其他ORM工具整合，Spring提供了对这些ORM工具良好的支持。



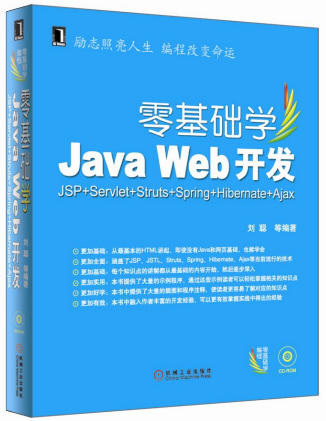
14.5.1　在Spring中配置数据源

- 在对数据库进行操作的时候，首先需要取得一个Connection对象，即需要首先取得与数据库的连接，在Spring中，是从DataSource中获取Connection对象的，通过下面的配置文件就完成了对数据源DataSource的配置。



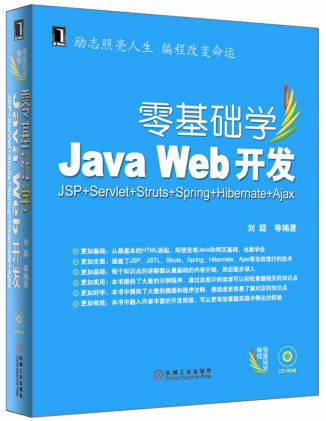
14.5.2　使用JdbcTemplate进行简单的数据库操作

- 虽然Spring可以很好的和其他ORM工具整合，但是普通的JDBC操作还是大部分开发人员的首选，所以在Spring中提供很好的JDBC支持， Spring中的JDBC框架承担了资源管理和错误处理的任务，从而使数据库操作的代码非常简洁。在Spring这些功能可以用JdbcTemplate类来完成，我们可以在程序用下面的代码创建一个JdbcTemplate对象。



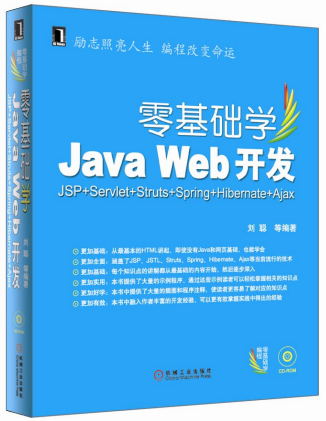
14.5.3　使用JdbcTemplate进行Java对象查询操作

- 在上面的实例程序中，返回的结果都是Java中基本的数据库类型，在Spring中，同样可以返回对象格式的结果，而且在Spring中返回对象比在简单的JDBC重返回对象的操作要简洁很多。在下面的实例程序中，将展示Spring返回基本Java对象的具体方法。（具体内容请参照书。）



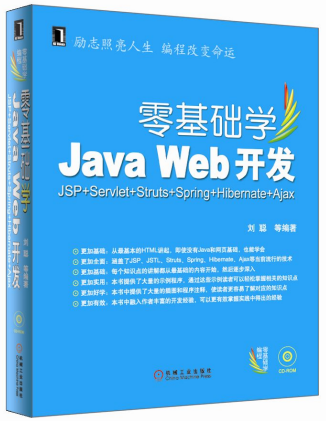
14.5.4　Spring中另一种Java对象查询的实现方法

- 在Spring中，提供了另外一种数据库操作方法，同样可以把查询的结果模拟成Java对象，即使用org.springframework.jdbc.object.MappingSqlQuery，在下面的示例程序中，展示的就是使用MappingSqlQuery把查询的结果模拟成对象的操作过程，详细代码如下。（具体内容请参照书。）



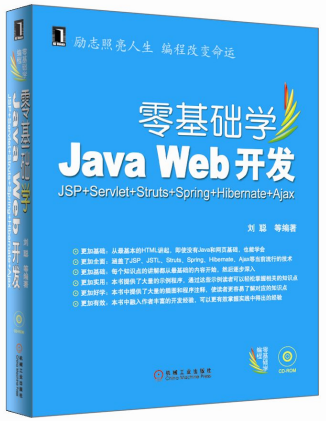
14.5.5　Spring中更新Java对象的方法

- 在Spring中提供了一种真正把数据库操作模拟成对象的方法，下面的这个示例程序就展示了这样一种方法，在这个实例程序中，把更新数据库的操作模拟成面向对象的方法。在这个实例程序中，继承了SqlUpdate类，从而实现了用对象的形式更新数据库的操作。（具体内容请参照书。）



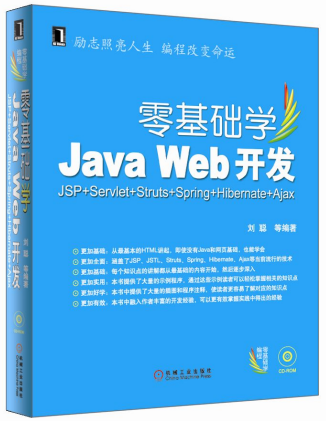
14.5.6　Spring和第三方ORM工具整合

- 在Spring中，本身并没有提供ORM的实现，但是提供了对现在流行的各种ORM工具的支持，例如Hibernate、OJB、iBatis等，Spring可以和这些ORM框架很好的整合，用来提供ORM的服务，而且在Spring中，还提供了一些基于ORM的附加服务，例如强大的事务管理功能、异常处理、模板类等。这就使Spring的ORM使用和操作变得非常方便。



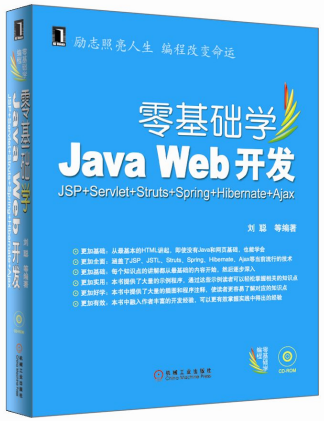
14.6　Spring中的事务处理

- 在数据库操作中，事务处理是非常重要的，在Spring中提供了强大的事务处理功能，不仅可以用编程的方式是实现，而且可以使用声明方式实现，通过使用Spring，可以在简单的JavaBean中使用类似EJB中的声明式事务管理。



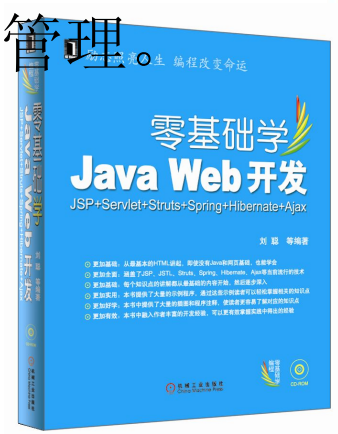
14.6.1　事务处理的基本知识

- 在数据库操作中，事务处理是经常用到的，例如在银行的业务中，甲方给乙方账户转账10万人民币，首先要从甲方的账户减去10万，然后再给乙方的账户增加10万，整个操作过程是一个整体，这就是一个简单的事务，在这个事务中必须保证操作的完整性，两步操作要么全执行，如果其中一步出错全都不执行，从而保证这个业务的正确性和完整性。



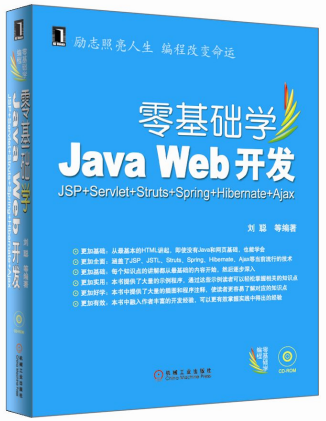
14.6.2　Spring中的事务管策略

- 在Spring中，并没有直接管理事务，而是提供了多种职务管理器，将事务管理的任务委托给这些事务管理器进行处理，每种事务管理器都是针对某种特定的平台，例如使用JDBC DataSource进行数据库操作的时候可以使用DataSourceTransactionManager进行事务处理，使用Hibernate的时候可以使用HibernateTransactionManager进行事务管理。



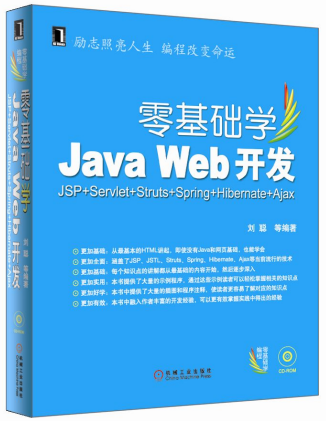
14.6.3　使用Spring编程式事务管理

- 在14.6.1中，可以看到使用JDBC进行事务处理的代码，在Spring中，处理事务的代码会简单很多， Spring中处理事务可以通过编程的方式实现，即通过代码调用Spring中的事务管理API，也可以通过Spring的上下文配置，用声明的方式进行事务管理。



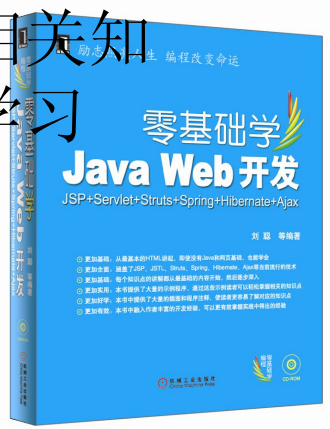
14.6.4　使用Spring声明式事务管理

- 在Spring中，提供了声明式的事务管理，使简单的JavaBean中也拥有了类似EJB的声明式事务管理功能。Spring中的声明式事务管理是使用AOP框架实现的，可以在程序中需要事务管理的地方用AOP的方式引入事务管理。



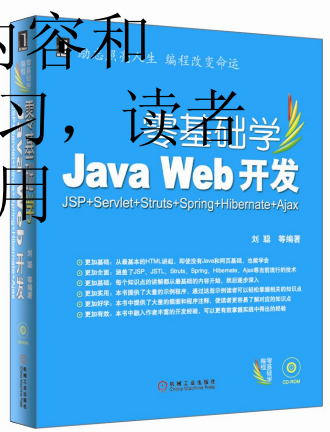
14.7 小结

- 在本章的内容中，介绍了Spring的基本知识，对Spring中各个部分的主要内容做了基本的介绍，在Spring中，核心的技术就是IoC和AOP，这两种技术贯穿了Spring的所有主题。Spring数据库方面、Web方面、事务处理方面都有着强大的功能，在本章中仅仅是对Spring的核心内容做了简单的介绍，从而是读者对这项技术有一个清醒的认识，如果读者有意进一步研究Spring的相关知识，可以参考Spring Reference，这是学习Spring的一个经典的资料。



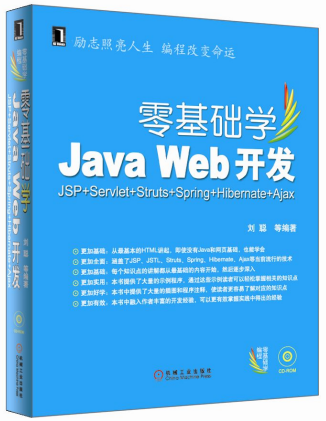
第十五章　Hibernate应用详解

- Hibernate是一个基于Java的对象/关系数据库映射工具，它将对象模型表示的数据映射到用SQL表示的关系模型上去。Hibernate管理Java到数据库的映射，还提供了数据查询和存取的方法，大幅度减少开发者的数据持久化相关的编程任务。在本章内容中，首先介绍对象关系映射（ORM）及Hibernate的基本概念，再详细介绍关于Hibernate的持久化对象以及ORM的详细内容和Hibernate查询相关知识。通过本章的学习，读者可以从整体上了解Hibernate，并学习使用Hibernate做数据持久化工作。



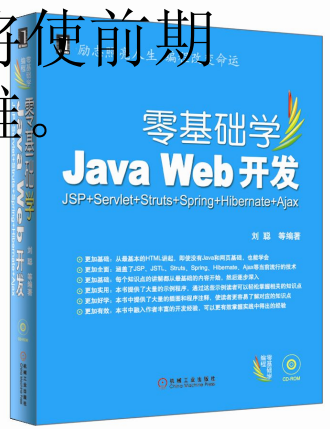
15.1　Hibernate简介

- 在当今的软件开发过程中，同时使用面向对象和关系数据库是一件耗时和头痛的工作。Hibernate是一个开源的对象/关系数据库映射工具，它的目标是较少应用开发过程中数据持久层的编程任务。将开发人员从繁杂的持久化层开发过程中解放出来。在本节中，将对对象关系映射以及Hibernate做基本的介绍。



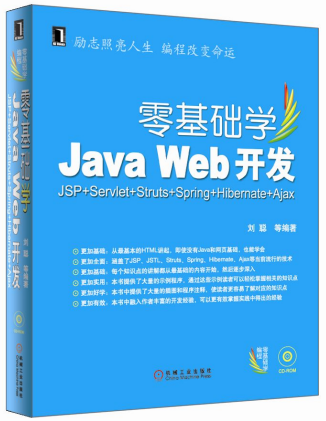
15.1.1　对象持久化与ORM

- 什么是持久化？简单地说，持久化就是把数据同步保存到数据库或者某些存储设备中。在软件的分成体系结构中，持久化是和数据库打交道的层次。在数据库中对数据的增加、删除、查找和修改都是通过持久化来完成的。在常见的JSP相关的Web开发中，经常有很多相关的数据库连接、查询等操作语句，这是把数据库相关的持久化工作和展现以及一些业务处理耦合在一起，这将使前期的代码编写和后期的工程维护都相当困难。



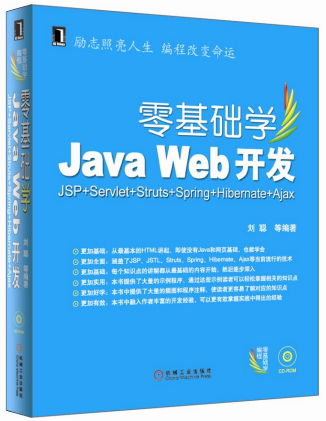
15.1.2　Hibernate架构概述

- Hibernate作为ORM映射工具，了解其整体架构对Hibernate的工作原理和以后介绍到的如何使用将有指导性的作用。Hibernate的高层架构图如图15.1所示。这个图显示了Hibernate利用数据库和配置数据向应用程序提供持久化服务和持久化对象。（具体内容请参照书。）



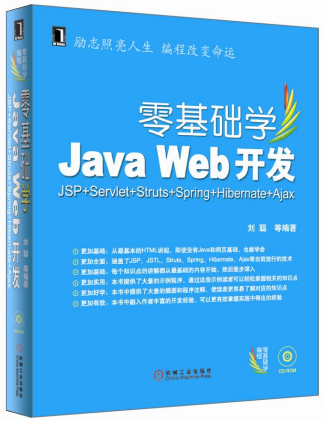
15.1.3　开发环境的简单配置

- 为了开发基于Hibernate应用，都需要一些数据库服务器，在此采用Mysql数据库，可以在下载/安装。Hibernate开发环境的配置方法比较简单，可以在http://www.hibernate.org下载Hibernate的发布包，把下载下来的压缩文件解压到硬盘中，然后把hibernate.jar和其他相关类库加入项目即可。



15.2　Hibernate配置和相关类

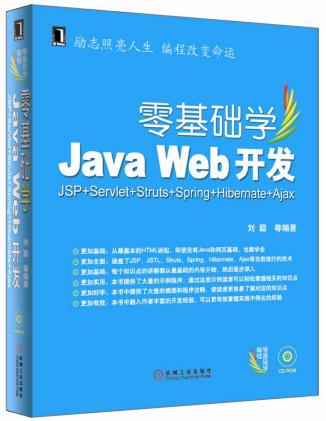
- 在使用Hibernate过程中，会发现Hibernate提供很多类，但常用的不会很多。其中最核心的就是关于整体数据库的配置文件和与之相关的类，在此对这些进行说明。Hibernate被设计为可以在不同的环境下工作，所以有很多配置参数，不过很多参数已经有默认值了，所以配置较少的参数就可以运行了。



15.2.1 Configuration类

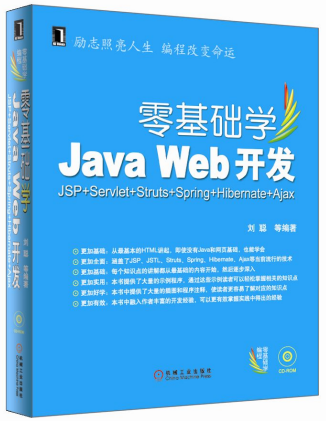
- Configuration类负责管理Hibernate的配置信息，一个Configuration类的实例代表了应用程序中Java类到数据库的映射的集合。应用程序通常只是创建一个Configuration实例，并通过它创建SessionFactory实例。例如下面的代码：

```
SessionFactory sessionFactory = new
Configuration().configure().buildSessionFac
tory();
```



15.2.2　Hibernate配置文件

- 这里我们来重点分析一下Hibernate配置文件，以现在最常用的hibernate.cfg.xml格式的配置文件做说明。这个配置文件默认会期望在类路径的根目录中找到。

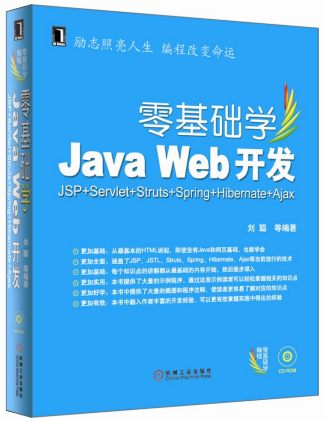


15.2.3 SessionFactory类

- SessionFactory负责Session实例的创建。为了创建一个SessionFactory对象，必须在Hibernate初始化时创建一个Configuration类的实例，并将已写好的映射文件交由它处理。这样，Configuration对象就可以创建一个SessionFactory对象，当SessionFactory对象传教成功后，Configuration对象就没有用了，可以简单地抛弃它。例如下面的实例代码：

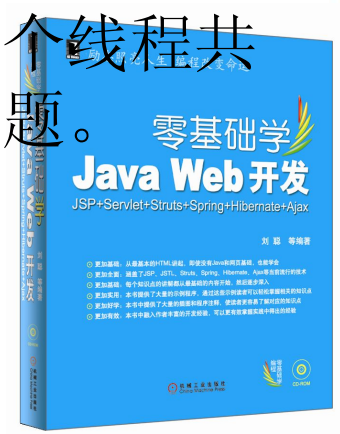
```
Configuration config = new
Configuration().configure();
```

- SessionFactory sessionFactory = config.buildSessionFactory();



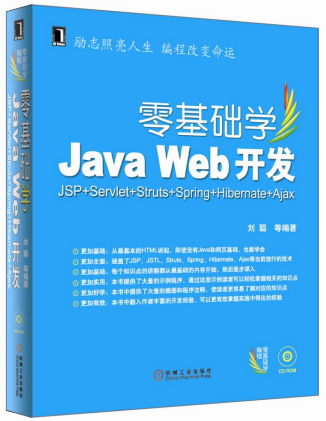
15.2.4　Session管理

- Session是Hibernate运作的核心，对象的声明周期、事务的管理、数据库的存取都与它密切相关。所以有效地管理Session成为使用Hibernate的重点。从上面的描述可以知晓，Session是由SessionFactory所创建。SessionFactory是线程安全的，可以让多个线程同时存取SessionFactory对象而不会引起数据共享的问题。可是Session不是线程安全的，所以让多个线程共享一个Session，会引起冲突和混乱的问题。



15.3 Hibernate中的对象

- Hibernate中对象有三种状态，临时对象（Transient Objects）、持久化对象（Persistent Objects）和脱管对象（Detached Objects）。理解这集中对象，对Hibernate中关于持久化处理是很有帮助的，这一节将对这几种对象做说明。



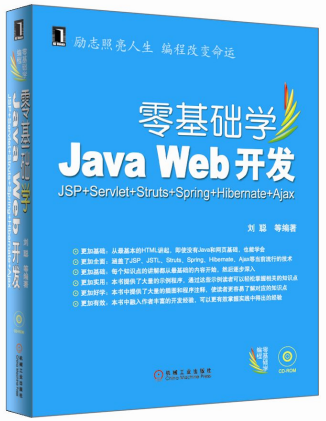
15.3.1对象在Hibernate的状态

- 如图15.3所示，显示了Hibernate中临时对象、持久化对象和托管对象之间的关系，它们之间的转换。下面就对这三种状态进行解释。

- （1）临时状态

- （2）持久化状态

- （3）脱管状态（具体内容请参照书。）

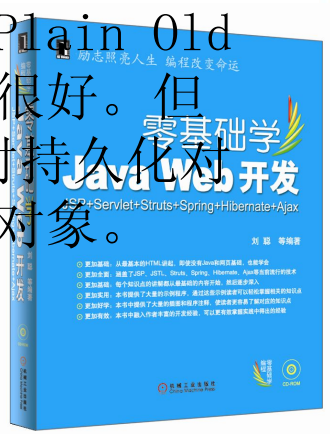


15.3.2　持久化类

- 上面所述的三种对象状态，在编程过程中就体现为持久化类的实例

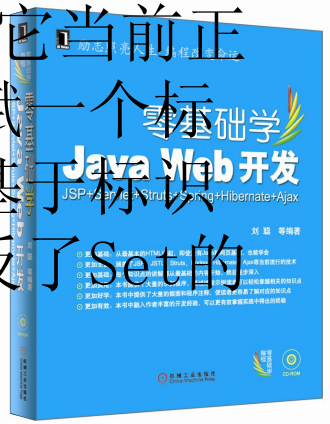
- 持久化类是应用程序用来解决商业问题的类（如我们将在下面提到的Customer类）。持久化类的实例通过Hibernate持久化管理层，将保存到数据库中的。

- 持久化类只需要符合简单的规则，也就是POJO（Plain Old Java Object）编程模型，Hibernate就会工作的很好。但是这些规则不是硬性要求的，最新的Hibernate对持久化对象的要求很少，你可以用自己的方法表示持久化对象。



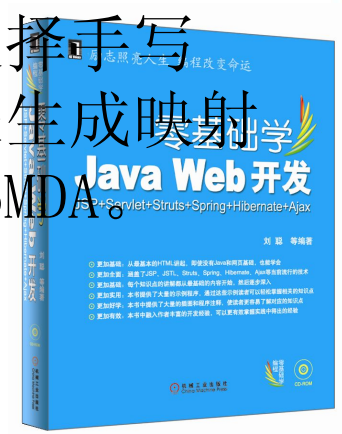
15.3.3　对象识别

- 实现equals()和hashCode()最显而易见的方法是比较两个对象标识符的值。如果值相同，则两个对象对应于数据库的同一行，因此它们是相等的（如果都被添加到 Set，则在Set中只有一个元素），不幸的是，对生成的标识不能使用这种方法。Hibernate仅对那些持久化对象赋标识值，一个新创建的实例将不会有任何标识值。此外， 如果一个实例没有被保存(unsaved)，并且它当前正在一个Set中，保存它将会给这个对象 赋一个标识值。如果equals() 和 hashCode()是基于标识值实现的，则其哈希码将会改变，这违反了Set的契约。



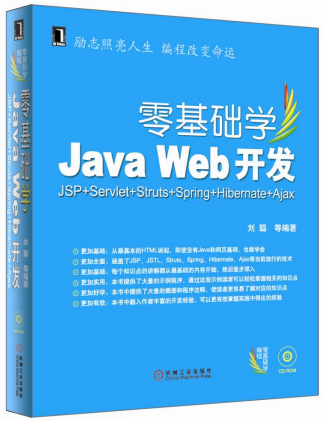
15.4　Hibernate中对象关系映射

- Hibernate的本质就是对象关系映射。映射文件就是将对象和关系数据库关联的纽带，在Hibernate中，映射文件通常以.hbm.xml作为后缀的。对象和关系数据库之间的映射通常是用一个XML文档来定义的。这个映射文档被设计为易读的，　并且可以手工修改。映射语言是以Java为中心，这意味着映射文档是按照持久化类的定义来创建的，　而非表的定义。虽然很多Hibernate用户选择手写XML映射文档，但也有一些工具可以用来生成映射文档，　包括XDoclet,Middlegen和AndroMDA。



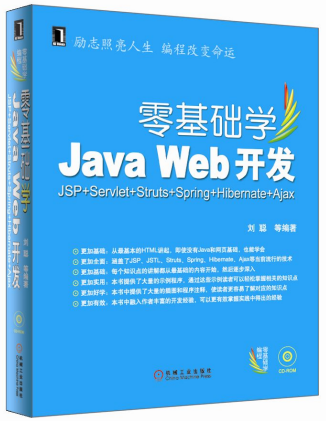
15.4.1　基本类映射

- （具体内容请参照书。）



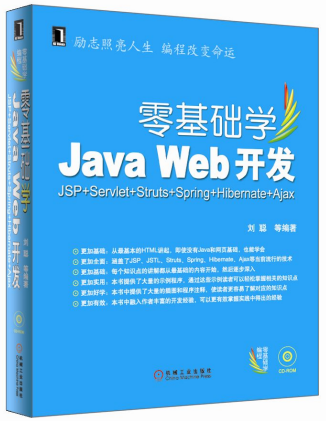
15.4.2　主键策略

- 在关系数据库中，以主键来区分不同的记录。Hibernate的主键策略分为3大类：
- Hibernate对主键id赋值；
- 应用程序对id赋值；
- 由数据库对id赋值



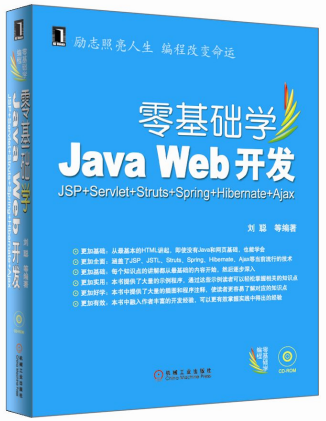
15.4.3　多表对象关系映射

- 多表映射主要是一对一、一对多（多对一）和多对多映射，下面分别对这些映射做分析。（具体内容请参照书。）



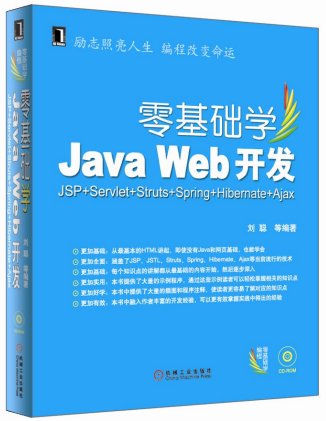
15.4.4　组件映射

- 组件(Component)是一个被包含的对象，在持久化的过程中，它被当作值类型，而并非一个实体的引用。在这篇文档中，组件这一术语指的是面向对象的合成概念（而并不是系统构架层次上的组件的概念）。



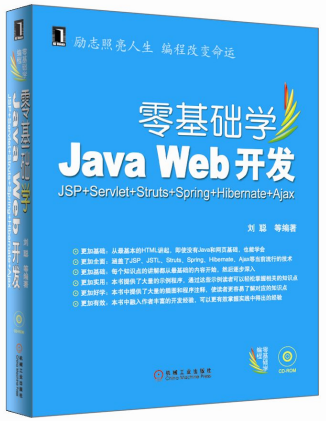
15.5　操作持久化数据

- 有了持久化对象，则可以利用Hibernate API对其进行操作，间接地把持久化类中的属性信息通过底层的JDBC API同步到数据库中。这一节来说明怎么操作持久化数据。



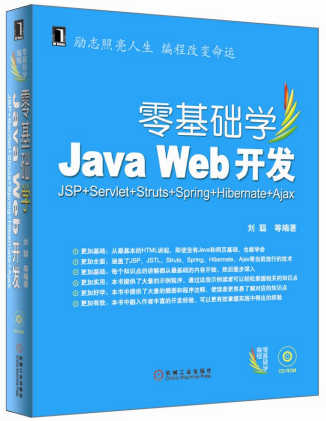
15.5.1　Session缓存与持久化操作

- 在15.3节中，我们提到Hibernate中对象的状态，提到持久化对象不会立即同步到数据库中。在此有必要对Session的flush做些说明。每间隔一段时间，Session会执行一些必需的SQL语句来把内存中的对象的状态同步到JDBC连接中。这个过程被称为刷出(flush)，默认会在下面的时间点执行：



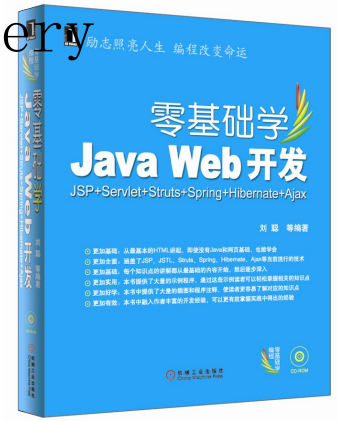
15.5.2　利用DAO来操作数据

- 在操纵持久化数据时，直接利用Session提供的方法，可能对数据操作的封装粒度太小，在实际开发过程中一般都不好使用。在此引入DAO（Data Access Object）的概念，他是持久化对象的客户端，负责所有与数据库操作相关的逻辑。例如数据查询、增加、删除、更新等，在这里提供一个关于Customer的DAO的例子，开如何开发DAO操作数据，提供给高层应用更抽象的API。



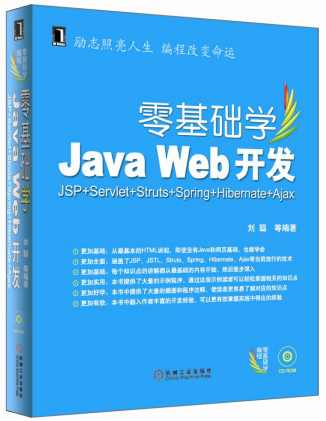
15.6 Hibernate数据查询

- Hibernate支持强大且易于使用的面向对象查询语言(HQL)。如果希望通过编程的方式创建查询，Hibernate提供了完善的按条件(Query By Criteria, QBC)以及按样例(Query By Example, QBE)进行查询的功能。你也可以用原生SQL(native SQL)描述查询，Hibernate额外提供了将结果集(result set)转化为对象的支持。其中最重要的一种就是HQL（Hibernate Query Language)。



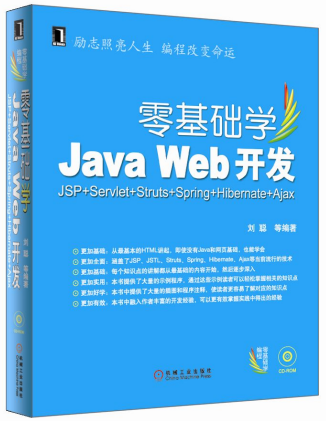
15.6.1　HQL检索方式

- Hibernate配备了一种非常强大的查询语言，这种语言看上去很像SQL。但是不要被语法结构 上的相似所迷惑，HQL是非常有意识的被设计为完全面向对象的查询，它可以理解如继承、多态 和关联之类的概念。（具体内容请参照书。）



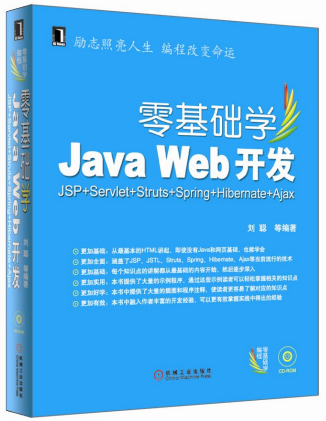
15.6.2　条件查询

- （1）创建一个Criteria实例　（2）限制结果集内容（3）结果集排序　（4）关联（5）动态关联抓取　（6）查询示例



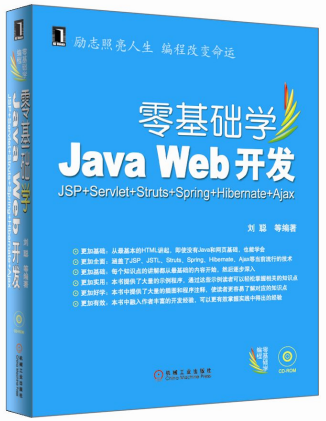
15.6.3　本地SQL查询

- 你也可以使用你的数据库的Native SQL语言来查询数据。这对你在要使用数据库的某些特性的时候(比如说在查询提示或者Oracle中的 CONNECT关键字)，这是非常有用的。这就能够扫清你把原来直接使用SQL/JDBC 的程序迁移到基于 Hibernate 应用的道路上的障碍。



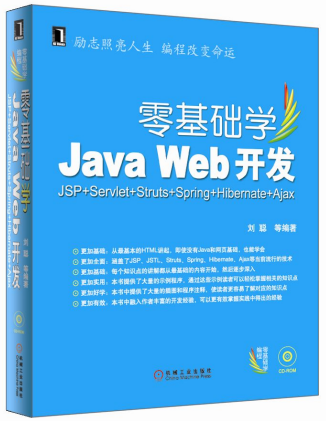
15.7 小结

- 在这一章中，我们介绍了Hibernate的整体架构，Hibernate配置相关内容，并对Hibernate中对象做了较为详尽的描述。重点介绍了对象关系映射的配置和Hibernate数据的查询。通过本章学习，能够对Hibernate技术有着详细的了解，为实际应用中的数据层持久化工作开发打下坚实的基础。



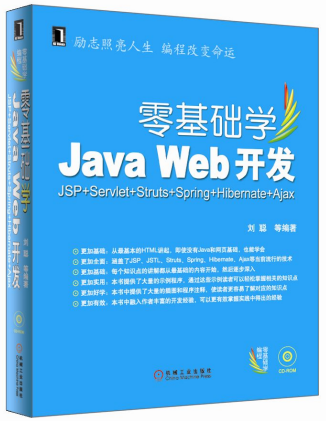
第十六章　Ajax应用详解

- 在本章的内容中，将要介绍Ajax技术，使用这种技术，可以构建出类似传统桌面应用程序的交互界面，可以丰富客户与服务器的交互方式。通过本章内容的学习，读者可以了解Ajax的基本知识，并且在本章中对Ajax处理客户请求的各个环节都进行了详细的分析，在具体的示例中展示了Ajax在各种情景下的具体应用，通过这些示例程序读者可以体会到Ajax的基本用法。



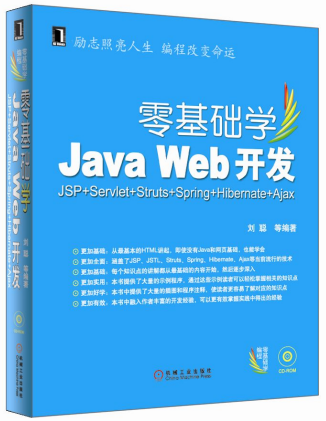
16.1　Ajax技术简介

- 从本质上讲，Ajax并不是一种全新的技术，Ajax只是综合利用已经存在的各种技术，从而诞生了一种全新的应用，在本节内容中，将简单介绍这种技术的诞生过程和其他相关的基本知识。



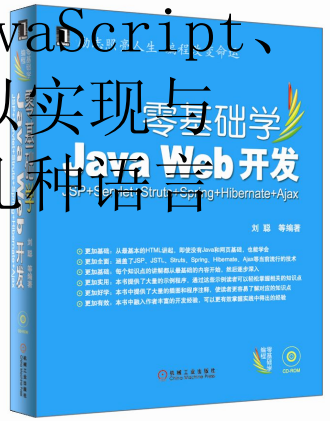
16.1.1　Web开发技术演变过程

- Web应用程序发展至今大体经历了三个阶段，第一个阶段使用的是简单的静态页面，第二个阶段使用的ASP、JSP、PHP等动态脚本语言，第三个阶段是Web2.0阶段，而Ajax就是Web2.0中的核心技术。（具体内容请参照书。）



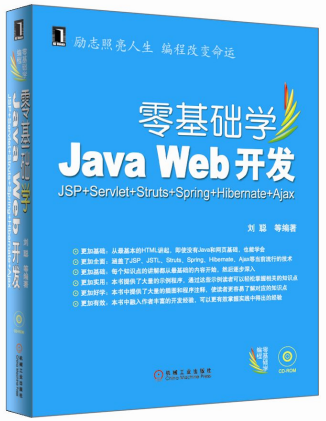
16.1.2　什么是Ajax

- 对于Ajax并没有确切的定义，而且随着Ajax被广泛应用，现在Ajax所包含的范围也更加广泛，所有的实现浏览器与服务器异步交互的技术都可以归入Ajax的范围，即无需刷新当前页面就可以实现与服务器的交互的技术，这种技术就是Ajax，而且Ajax也不像Java、JSP等是一种单独的技术，Ajax是一系列技术的集合，例如在在实现与服务器的异步通信的时候，需要用到XML、JavaScript、XMLHttpRequest等，使用这几种技术可以实现与服务器的异步通信，所以，Ajax就是这几种语言的综合体。



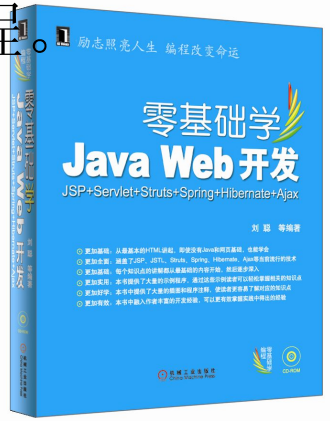
16.1.3　Ajax的相关技术

- 在上面的内容中，介绍了Ajax是一系列技术的集合体，通过这些技术的综合运用从而实现Ajax的目标，实现客户端与服务器端的异步通信。（具体内容请参照书。）



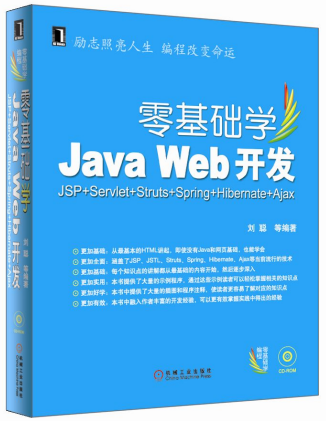
16.2　Ajax工作原理简单示例

- 在本节的内容中，将通过一个具体的例子来展示Ajax的工作原理，通过与传统请求响应方式的对比，展现Ajax中处理请求响应的不同方式。在这个示例程序中，所要完成的任务非常简单，仅仅需要输入一个姓名，然后提交这个表单，在服务器端处理这个请求，然后在页面显示处理的结果，在下面的示例程序中，将展示通过传统方式和Ajax方式处理这个请求响应的具体过程。



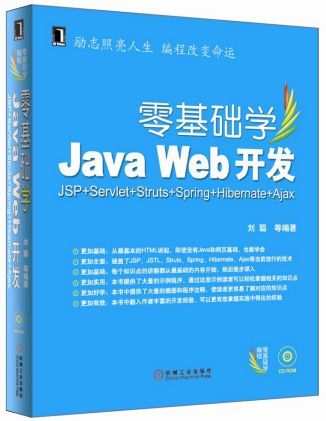
16.2.1　传统的请求响应方式

- 在传统的请求响应方式中，是通过表单向服务器提交用户信息，服务器端处理接收到的信息，并把处理结果返回给用户，在这个过程中需要刷新整个页面才能得到服务器返回的结果。（具体内容请参照书。）



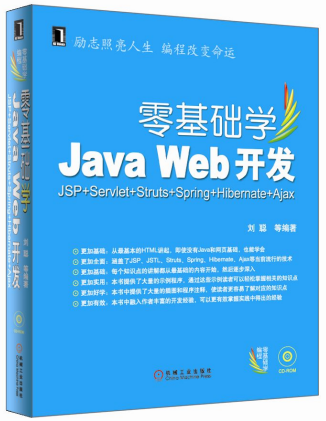
16.2.2　使用Ajax的请求响应方式

- 在上面的示例程序中，展示了传统的请求响应处理方式，在本节中，将展示Ajax中处理请求响应的方式。在Ajax处理请求响应的方式中，不会整个页面进行刷新，对于用户的输入信息，并不依靠表单来提交，而是通过XMLHttpRequest对象传递给服务器。（具体内容请参照书。）



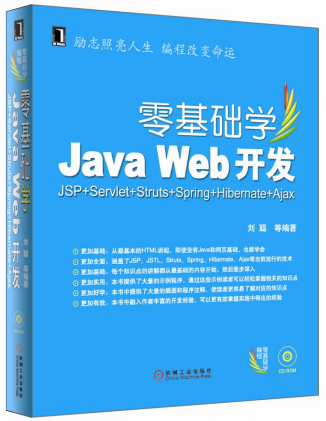
16.3　XMLHttpRequest对象

- XMLHttpRequest对象在Ajax中占据着十分重要的地位，Ajax中的客户端就是通过XMLHttpRequest对象实现与服务器的通信，在本节内容中将详细介绍这个对象的基本知识。



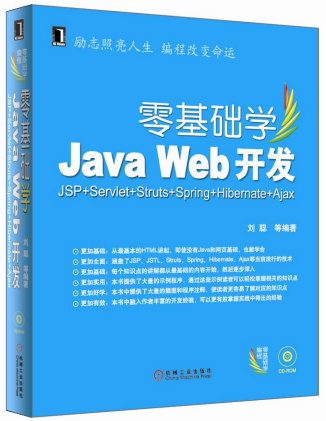
16.3.1　XMLHttpRequest对象简介

- XMLHttpRequest对象也不是一个新的技术，这个对象最早出现在微软的IE浏览器中，是以ActiveX组件的形式出现的，在当时并没有引起开发人员太大的注意，但是在Google推出Google Map和Google Suggest产品以后，Ajax技术以飞快的速度发展起来，而XMLHttpRequest对象又是Ajax的重要组成部分，所以XMLHttpRequest对象也开始重新受到重视。



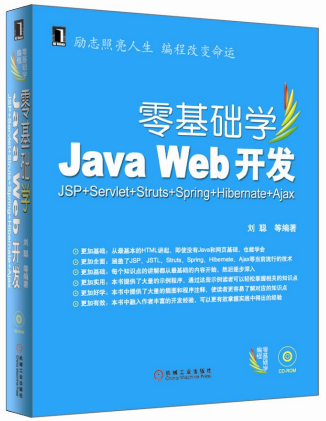
16.3.2　创建XMLHttpRequest对象

- 在使用XMLHttpRequest对象发送请求、接收响应之前，需要创建这个对象，其中，在IE浏览器中，XMLHttpRequest对象是以ActiveX组件的形式提供的，其他浏览器中使用JavaScript本地方法来创建，所以在创建XMLHttpRequest对象的时候，需要对这个差别做对应的判断和处理，下面的代码就是创建XMLHttpRequest对象的通用代码。



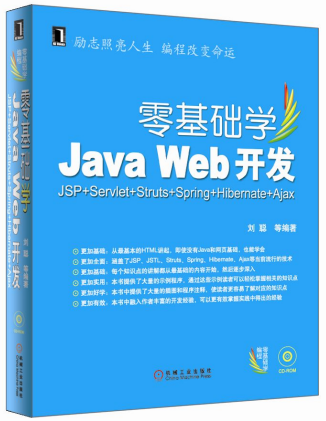
16.3.3　XMLHttpRequest常用方法和属性

- 在XMLHttpRequest对象创建以后，就可以在对这个对象进行各种不同的操作，从而完成和服务器的通信，接下来将介绍XMLHttpRequest对象常用的方法和属性。open（string method,string url,boolean asynch,string name,string password）和send（content）是XMLHttpRequest对象中最常用的方法。



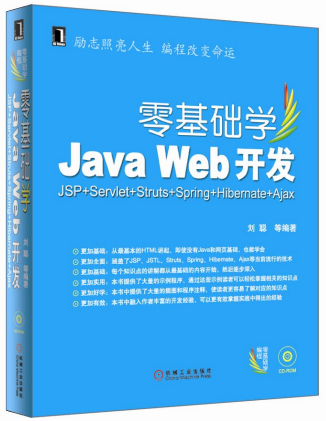
16.4　客户端向服务器发送请求

- 在Ajax中，向服务器端发送请求使用的是XMLHttpRequest对象，在XMLHttpRequest对象成功创建以后，就可以通过这个对象与服务器进行通信，在本节接下来的内容中，将介绍客户端使用XMLHttpRequest对象向服务器发送请求的基本方法。



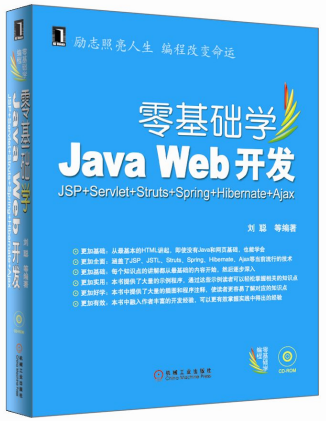
16.4.1　使用XMLHttpRequest对象发送请求

- 在向服务器发送请求之前，首先使用XMLHttpRequest对象的open（string method,string url,boolean asynch,string name,string password）方法建立对服务器的调用，然后才能向服务器发送请求信息。而且在这里还需要指明在请求状态发生改变的时候需要调用的时间处理方法。



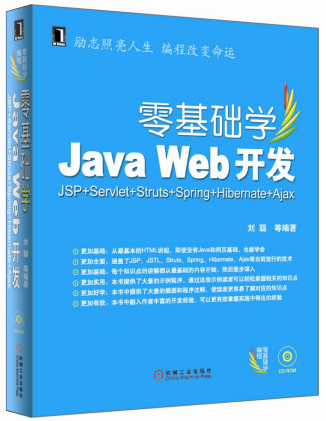
16.4.2　常用发送请求内容的方法

- 在传统的Web应用中，是通过表单向服务器提交用户的输入信息，但是这种提交方式会刷新整个页面，在Ajax中要实现的就是与服务器端的异步通信，所以就不能使用表单向服务器发送请求信息，在Ajax中，可以使用下面几种方法来向服务器发送请求的内容。



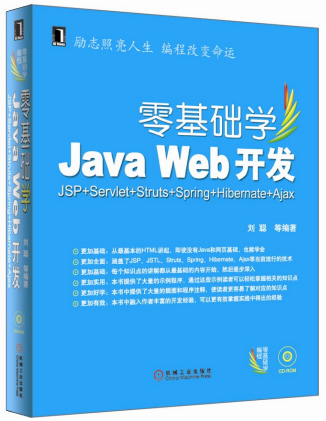
16.5　服务器端处理用户请求

- 在Ajax中，服务器接收到用户的请求以后，可以根据请求的内容进行相应的操作，然后把操作以合适的格式返回给客户端，在本节内容中，将重点介绍在服务器端对客户请求的处理。在服务器端处理方式可以有很多种选择，可以选择JSP、ASP、CGI、Servlet中的任意一种作为相应客户端请求的服务程序，在本章内容中，将选择Servlet实现服务器端逻辑处理的功能。



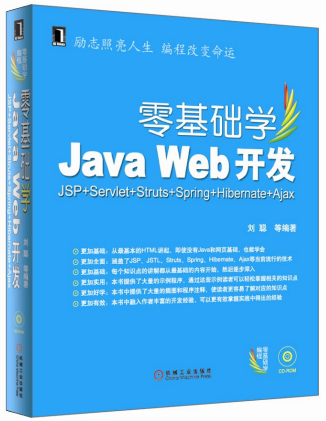
16.5.1　在服务器端处理用户请求

- 由于客户端向服务器发送信息的时候，可以选择多种方式进行发送，所以在服务器端就需要根据客户端发送信息的方式，对接收到的信息进行分析，从而取出进一步操作所需要的信息。



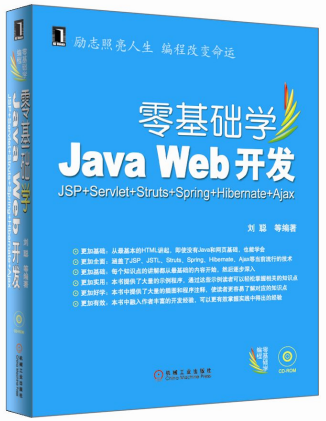
16.5.2　返回XML格式的响应文档

- 在服务器端完成用户需要的逻辑处理以后，需要把处理的结果返回给用户，在这种情况下，一般是把处理的结果组织成XML的格式，然后把这个XML文档返回给客户端。



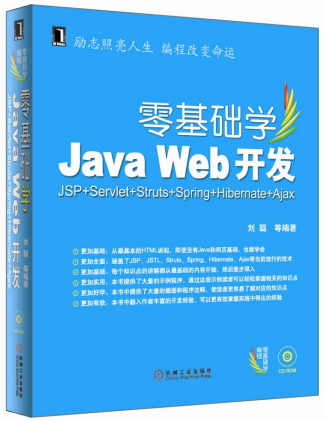
16.6　客户端处理服务器响应

- 在服务器端结束对用户请求的处理以后，会把处理的结果返回给用户，在客户端的需要对返回的内容进行处理，然后根据这些处理结果对页面的内容进行调整，到这一步为止，客户端与服务器端的异步通信就完成了。在本章接下来的内容中，将介绍客户端处理返回的响应内容的基本方法。



16.6.1　分析XML格式的文档

- 在一般情况下，服务器会用XML文档的格式返回逻辑处理的结果，在客户端可以通过XMLHttpRequest对象取得这个响应文档的内容。在JavaScript中，可以以DOM的方式分析这个XML格式的文档，这种分析方法和一般的XML文档的分析方法是完全相同的，在本书第八章XML基础知识中已经介绍过使用JavaScript解析XML文档的相关知识，在这里不再赘述。

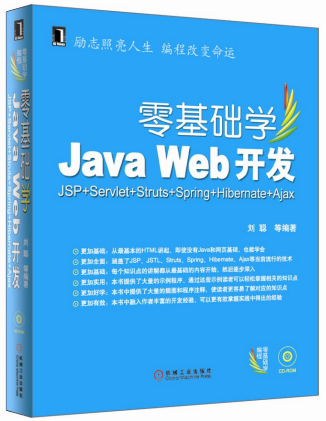


16.6.2　使用JavaScript调整页面内容

- 在对XML文档解析结束以后，就可以根据解析的结果来调整页面的内容，从而把服务器的处理结果表现在页面上，通常情况下会使用JavaScript来完成这个任务，通过使用innerText或者innerHTML可以设置HTML页面元素内的显示内容，通过DOM操作，可以动态创建HTML元素，通过CSS可以控制页面HTML元素的显示风格，通过这些操作，可以把服务器返回的处理结果充分展现在页面上，从而最终完成客户端和服务器的异步通信，而且这种处理方式是不会对整个页面进行刷新的。

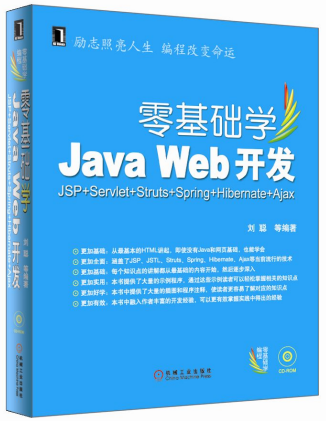
16.6.3　客户端处理服务器相应的示例代码

- 下面的示例代码片段中，展示了如何在客户端处理服务器的响应信息，具体代码片段如下所示。（具体内容请参照书。）



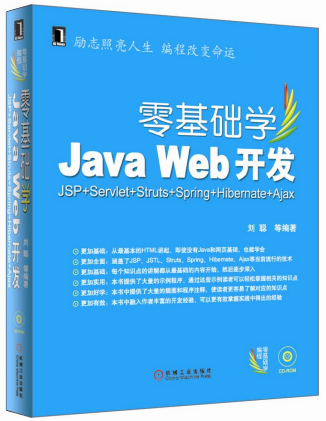
16.7　Ajax典型示例

- 在上面的内容中，对Ajax的基本知识进行了简单的介绍，在本章接下来的内容中，将通过具体的示例程序展示Ajax的魅力，而且通过这些具体的示例，读者可以体会到Ajax的具体使用方法，从而可以在自己的Web应用中添加Ajax的处理方式。



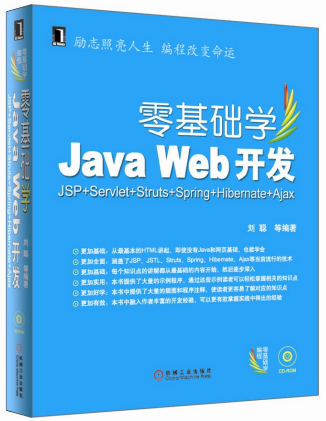
16.7.1　异步身份验证

- 在传统的Web应用中，用户的身份验证是通过向服务器提供表单，服务器对表单中用户信息进行验证，然后再返回验证的结果，在这样的处理方式中，用户端必需等到服务器返回处理结果才能尽进行别的操作，而且在这个过程中，会刷新整个页面。这种处理方式不仅浪费了用户的时间，每次刷新页面也浪费了巨大的带宽。（具体内容请参照书。）



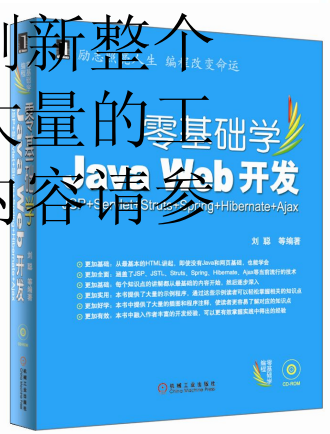
16.7.2　输入提示和自动完成

- 在使用Google搜索或者是Baidu搜索的时候，在输入搜索关键字的同时，会自动弹出匹配的其他关键字的提示，这种输入提示和自动完成的功能是在Google中首先推出的，然后就在各种Web应用中被广泛采用，在下面的示例程序中，就展示了这样一个功能的实现。下面就是输入提示和自动完成功能JSP页面的具体代码。（具体内容请参照书。）



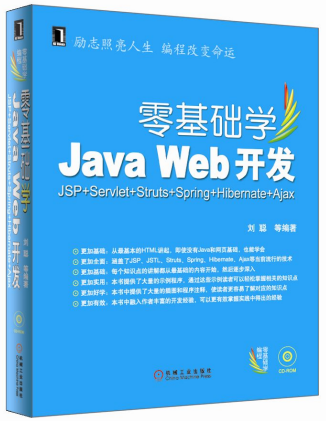
16.7.3　联动动态列表

- 在We应用的开发中，经常会遇到联动动态列表的需求，有其是在查询条件的选择中，所有的下拉列表中的选项都是从数据库中动态取出的，当选择第一个下拉列表的时候，后面的下拉列表要以这个选择为条件从数据库中取出满足条件的内容，从而调整显示选项的内容。然而在传统的Web开发模式中，要实现这样的功能是相当的麻烦，每次调整下拉列表的时候，都需要重新刷新整个页面，而且在刷新页面的时候，需要做大量的工作来保存已经选择列表的状态。（具体内容请参照书。）



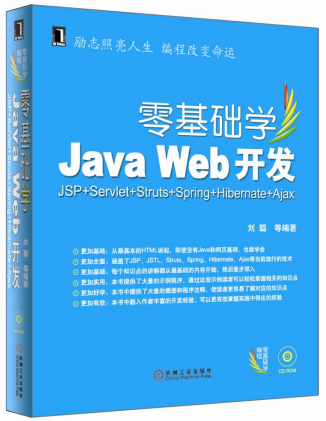
16.7.4　异步输入验证

- 在Ajax中，对用户的输入验证可以使用异步的形式，在用户每项输入完成之后，都对用户输入的合法行进行验证，即时向用户返回验证结果，使用户及时发现输入的错误，及时更改输入，而且这种验证不会刷新整个页面，用户的输入信息都会别保留，从而可以节省用户的时间，方便用户的使用。（具体内容请参照书。）



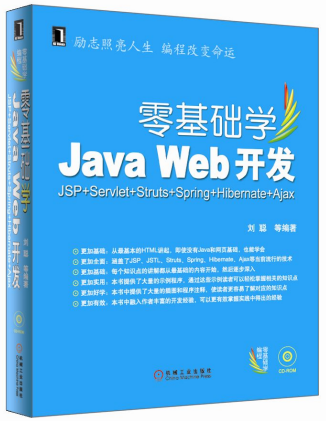
16.7.5　工具条提示

- 在桌面应用程序中，可以给程序添加工具条提示的功能，当鼠标移到指定位置的时候，会弹出一个提示框，提示这个区域的简单信息。（具体内容请参照书。）



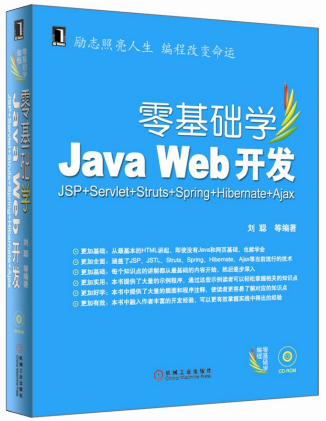
16.7.6　自动刷新

- 使用Ajax也可以实现自动刷新的功能，而且是页面局部的刷新，这种刷新方式不会对页面的其他部分造成影响。（具体内容请参照书。）



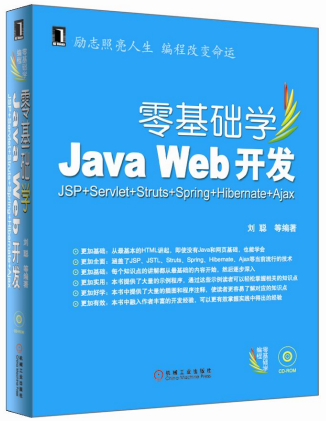
16.7.7　使用DOM动态生成HTML文档

- 在很多时候，在浏览器段，需要根据服务器的返回结果，动态生成HTML文档，这种功能一般情况下都是采用DOM操作的方法来完成，在接下来的示例程序中，将展示使用DOM动态生成HTML文档的基本方法。（具体内容请参照书。）



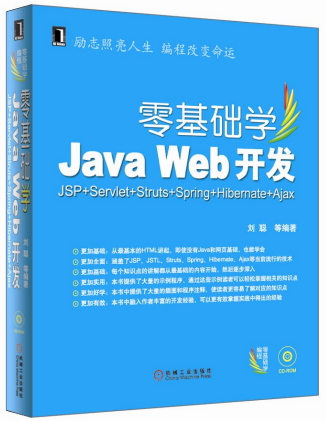
16.8 小结

- 在本章的示例程序中，都是通过手工代码完成Ajax处理的各个过程，目的是为了让读者清楚了解Ajax的操作原理和流程，在实际的开发过程中，读者可以选择使用开源的Ajax框架来进行Ajax程序的开发，目前有很多成熟的Ajax框架可供选择，例如DWR、Bindows、Rico等，这些框架都是比较成熟的Ajax框架，读者可以在这些框架的官方网站浏览相关的效果展示。



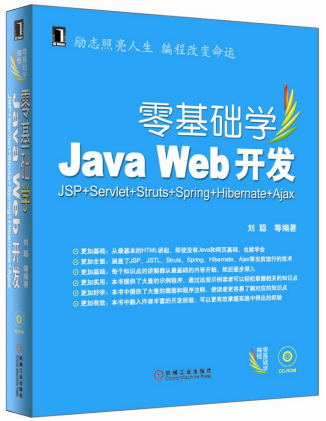
第十七章　Struts+Spring+Hibernate构建电子商务系统

- 在本章内容中，将以Hibernate Jpetstore为基础，详细介绍在实际的开发过程中，如何综合使用Struts、Spring、Hibernate这三种技术，对于开发过程中的主要步骤，在接下来的内容中将会详细介绍。



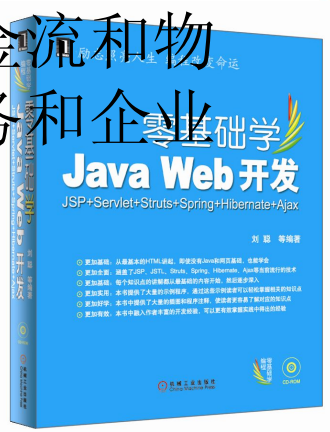
17.1　电子商务简介

- 在互联网日益发达的今天，电子商务的应用也得到了日益广泛的应用，例如目前比较知名的电子商务网站卓越、易趣、阿里巴巴等，这些电子商务网站都取得了巨大的成功，消费者也逐渐接收了网上购物的方式，这都促使电子商务应用的快速发展。在本节的内容中，将简单电子商务应用系统的基本知识。



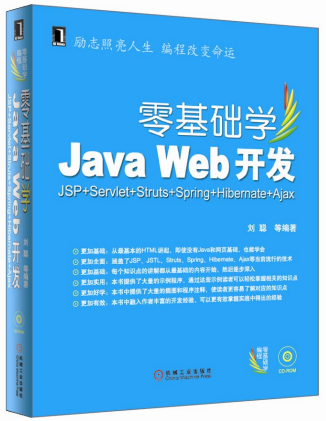
17.1.1　电子商务简介

- 电子商务（Electronic Business）就是基于互联网的商业活动，它不仅包括基于互联网的商品服务交易，而且还包括将企业内外部关系转化为创造价值和开发市场的机会，包括通过互联网实现原料采购、产品生产、产品展示、物流、销售等各个生产的环节。电子商务的物理基础包括计算机互联网、金融电子系统、物流配送系统、安全保障系统等，电子商务通过信息流、资金流和物流来实现。可以划分为企业间的电子商务和企业对消费者的电子商务等。



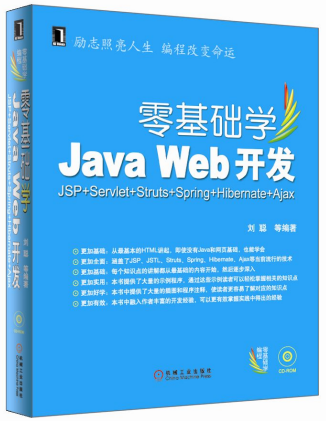
17.1.2　电子商务应用范围

- 目前电子商务主要应用于金融、证券、保险、制造业、服务行业以及IT行业等。在这些行业中已经能够很好的通过互联网辅助业务的处理，尤其是在金融证券等行业，电子商务已经涉及的日常工作的每个方面。



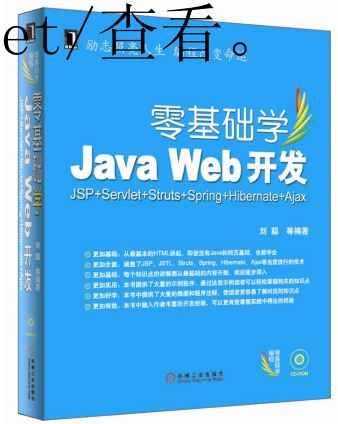
17.2 Hibernate Jpetstore简介

- 在本章的内容中，将以Hibernate Jpetstore为基础介绍使用Struts、Spring、Hibernate开发电子商务网站的基本流程，接下来将简单介绍Hibernate Jpetstore的基本情况。



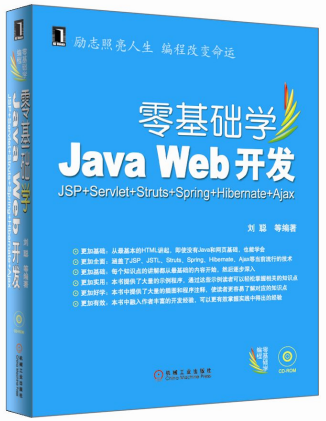
17.2.1 Hibernate Jpetstore简介

- Jpetstore是Spring开发包中的一个示例程序，在这个示例程序中采用了Struts、Spring、iBatis实现了一个简单的电子商务系统，在Java开发者社区中，使用Hibernate替换了Jpetstore中的持久层iBatis，这个改造以后的项目就是Hibernate Jpetstore系统，Hibernate Jpetstore成为Java开发者社区中的一个开源项目，项目的具体信息可以在https://hjpetstore.dev.java.net/查看。



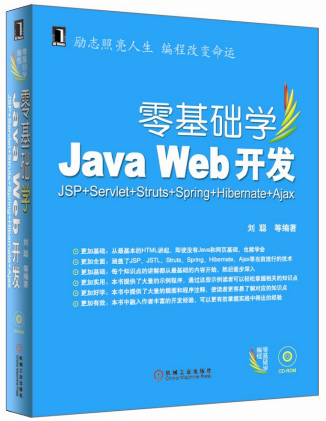
17.2.2　获取Hibernate Jpetstore示例代码

- Hibernate Jpetstore是一个开源项目，系统的源代码可以在项目的CVS服务器中获取。这个项目使用的开发工具是NetBeans，在本书介绍的内容中，使用的开发工具是Eclipse，这两种工具的工程信息文件是不同的，项目文件组织方式也有所不同，其中Eclipse中的WebRoot目录对应着NetBeans中的Web目录，其他的部分是没有区别的。



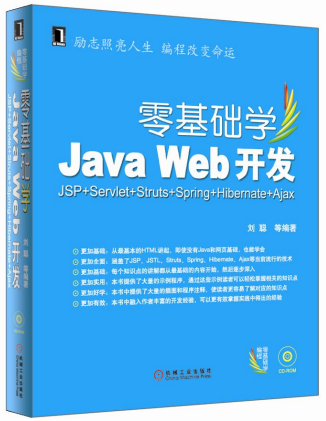
17.3 Hibernate Jpetstore系统总体设计

- 在本节内容中，将介绍Hibernate Jpetstore系统的总体设计，包括系统功能描述和系统架构的设计。



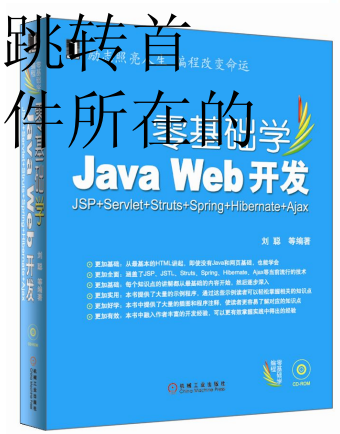
17.3.1　功能分析

- Hibernate Jpetstore系统是一个宠物商店的示例程序，在这个示例项目中，展示各种类型的宠物，用户可以在查询、浏览系统中提供的宠物，挑选并定购喜欢的宠物。（具体内容请参照书。）



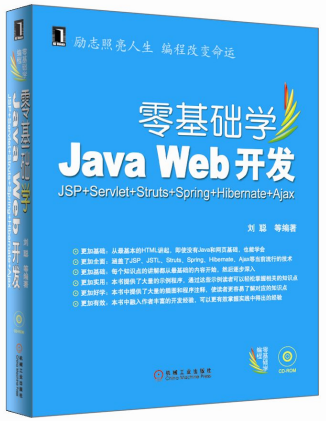
17.3.2　系统整体架构——表示层

- 表示层也叫视图层，在Hibernate Jpetstore这个系统中，表示层是用JSP、JSTL标签库和Struts实现的。在这个系统中，表示层的文件包括hibernateJpetstore/web/WEB-INF/jsp/struts/目录下的所有文件，这个目录下是系统中用到的所有JSP页面，同时hibernateJpetstore/web/index.jsp这个页面是Hibernate Jpetstore这个系统中的前端跳转首页，用户可以通过这个页面跳转到JSP文件所在的目录。（具体内容请参照书。）



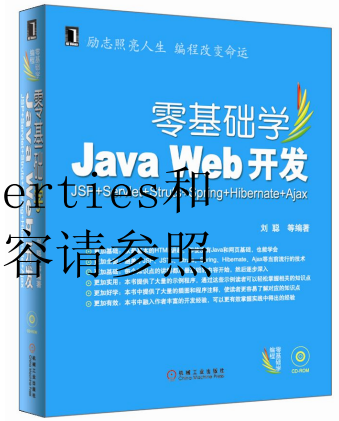
17.3.3　系统整体架构——控制层

- 控制层出于表示层和数据层之间，它的功能是把表示层和数据层联系起来，降低表示层和数据层的联系。这也正是MVC开发模式中一直强调的一点。在Hibernate Jpetstore中，控制层主要由以下几个部分组成。（具体内容请参照书。）



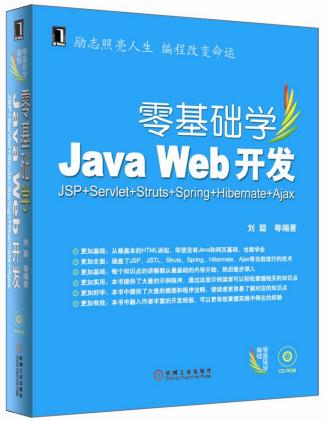
17.3.4　系统整体架构——数据层

- 在这个示例系统中，数据层由以下几部分构成。
- hibernateJpetstore/src/org.springframework/samples/jpetstore/dao/下的所有文件。
- hibernateJpetstore/src/org.springframework/samples/jpetstore/domain/下的所有文件
- hibernateJpetstore/web/WEB-INF/目录下的dataAccessContext-hibernate.xml、jdbc.properties和applicationContext.xml这三个文件。（具体内容请参照书。）



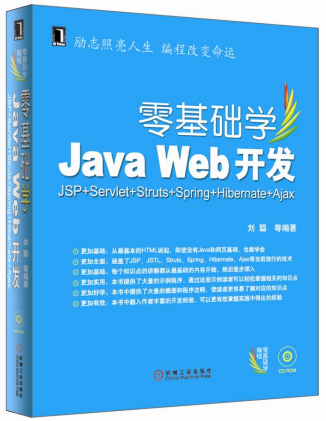
17.4 Hibernate Jpetstore系统数据层设计

- 一般来说，数据层设计的好坏直接关系到整个系统的成败。在关系数据库占工业主导地位的今天，不论采用何种数据层技术，必然牵涉到数据库表，以及表与表之间的关系。在本节接下来的内容中，将介绍Hibernate Jpetstore中的数据库设计。



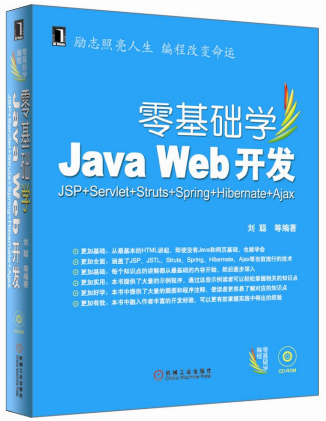
17.4.1　数据库ER图

- 如图17.4所示就是这个示例系统中的数据库ER图。其中，account描述了用户账号的信息，orders描述了用户订单的信息，lineitem描述了订单中每一项的信息，item描述了宠物的详细信息，supplier描述了供应商的信息，inventory描述了库存信息。



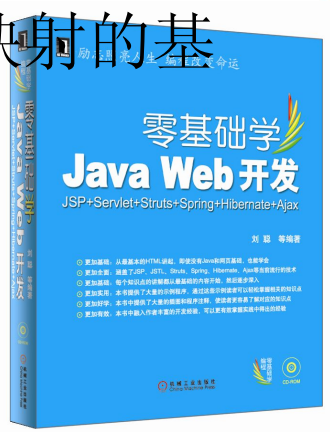
17.4.2　数据库SQL脚本

- 完成了数据库关系的设计以后，就需要提供对应的数据库SQL脚本，下面就是这个示例系统中数据库所对应的SQL脚本，通过这些脚本可以创建系统所需要的基本的数据库表结构。（具体内容请参照书。）



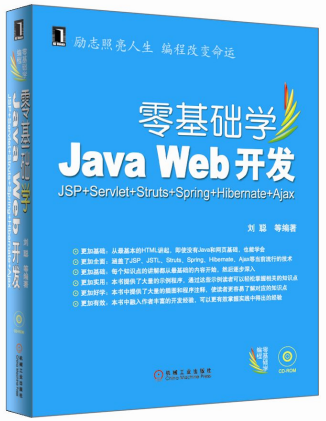
17.4.3　Hibernate数据库映射

- 在Hibernate中，可以把对数据库的操作模拟成对对象的操作，所有的数据库操作都可以通过对象操作实现，在Hibernate中，之所以能实现这样的功能，最关键的部分就是Hibernate可以把数据库的表结构和Java对象映射起来，用Java对象的属性描述数据库表中的字段，用Java对象之间的关系描述数据库表之间的关系，下面以用户表account为例，展示Hibernate中数据库映射的基本方法。（具体内容请参照书。）



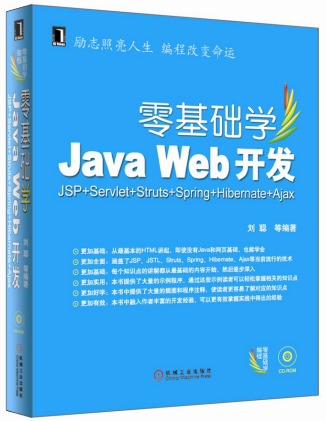
17.4.4　DAO调用序列

- 在Hibernate Jpetstore这个示例项目中，对数据库的操作全都集中在各个DAO中，在程序中可以调用这些DAO进行数据库的操作，（具体内容请参照书。）



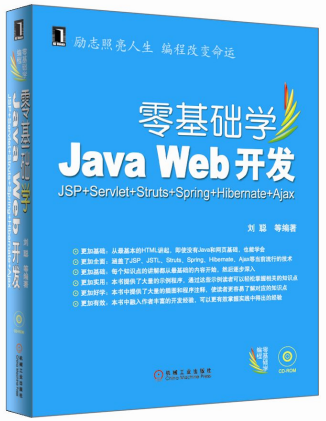
17.4.5 DAO接口设计及Hibernate DAO实现

- 对于DAO的实际，一般情况来说，每个DAO都会包含所有的CRUD（即增删改查操作），利用这些基本功能就可以组成各种复杂的数据库操作，但是在实际的开发过程中，除了基本的CRUD以外，为了使用上层的要求，需要根据需求提供更多其他的方法，在这些方法中可以执行比基本CRUD更多的操作。（具体内容请参照书。）



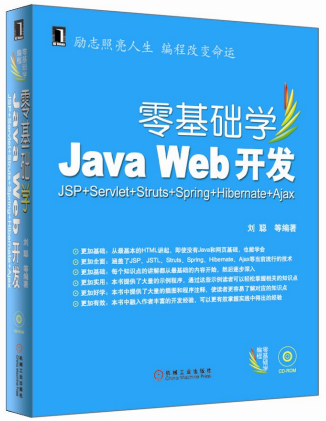
17.5 Hibernate Jpetstore系统控制层设计

- 在传统Struts应用中，控制层组件一般情况下指的是Action，而ActionForm只是用来收集表示层的数据，并把这些数据传递给Action。在这个示例程序中，引入了Spring，所以就可以使用Spring的依赖注入、AOP以及声明式的事务管理机制。在本节接下来的内容中，将介绍Spring的这些特性和常规的Struts的功能。



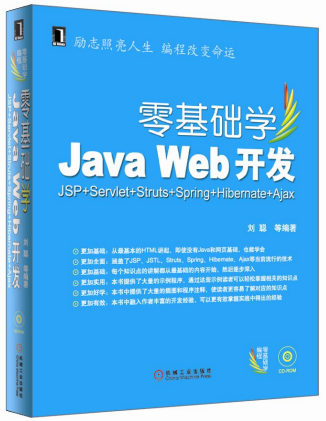
17.5.1　通过Spring AOP实现邮件发送

- Spring AOP的内容在本书的14.3小节中已经做了详细的介绍，在Hibernate Jpetstore这个示例系统中，有两个地方用到了AOP的知识，一个是邮件发送操作中用到，另一个用到AOP的地方是声明式的事务处理，事务处理在接下来的小节中将详细介绍，在这里介绍Hibernate Jpetstore中使用AOP进行邮件发送功能的具体实现方法。（具体内容请参照书。）



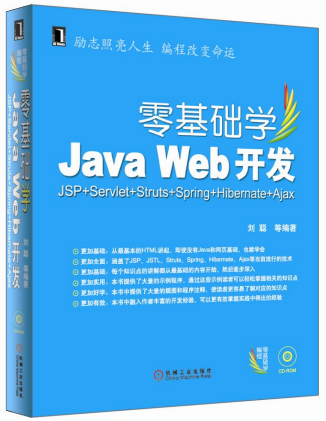
17.5.2　Spring的声明式事务管理

- 在Hibernate Jpetstore这个示例项目中，还有一个地方使用到了AOP的知识，这就是在Spring中的声明式事务管理，在这个示例项目中就使用了Spring中的声明式事务管理机制。在Spring中使用声明式事务管理的方法非常简单，仅仅通过Spring上下文的配置即可。下面就是这个示例项目中的声明式事务管理的具体配置信息。



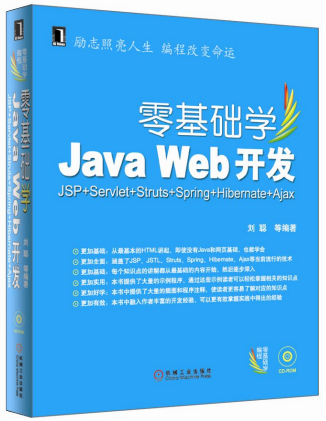
17.5.3　Struts在控制层的功能

- Aciton是Struts的核心组件，它负责这Struts控制层的基本操作，在Action中可以调用业务层或者是DAO完成具体的逻辑操作。在页面上的每一个链接或者是表单都会对应这一个Action，在Struts中每个Action都是org.apache.struts.action.Action的子类，并且需要重写execute（）方法，在这个方法中调用业务层或者是DAO来处理用户的请求。



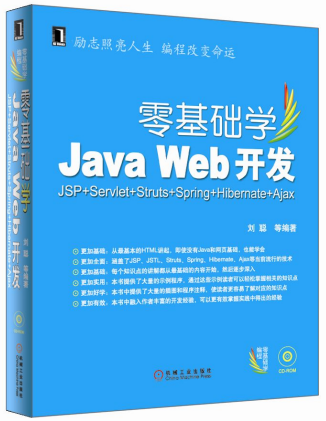
17.6 Hibernate Jpetstore系统表示层设计

- 在Hibernate Jpetstore这个示例程序中，表示层采用了Struts标签、JSTL和JSP页面相结合的处理方法，采用JSTL的原因就是减少JSP页面上的Java代码，而Struts的标签是执行某些Struts操作中必不可少的，在本节内容中，将介绍这个示例程序中表示层的实现方法。



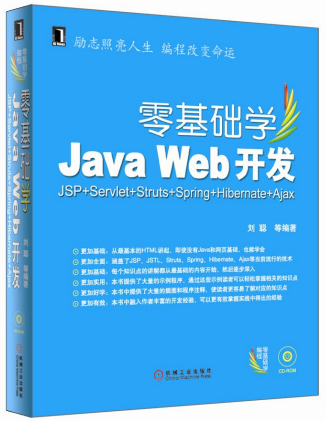
17.6.1 Struts 表示层组件 FormBean

- 在Struts中，FormBean充当了表示层的组件，它可以接收用户的表单输入，并且把表单的内容传递给Action处理，FormBean和我们熟悉的JavaBean非常类似，不同之处在于这些FormBean都需要继承
  org.apache.struts.action.ActionForm这个类，而且在需要的时候还得重写其中的validate（）和reset（）方法。



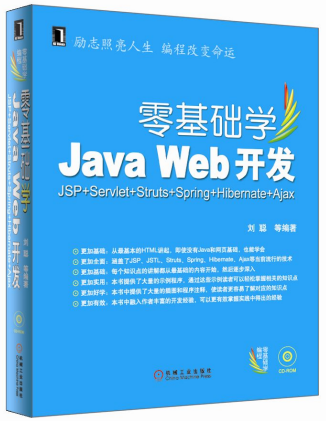
17.6.2 FormBean 类层次

- 于Aciton中的BaseAction类似，FormBean也提供了一个基类，所有的FormBean只需要实现BaseActionForm这个基类的指定约束，既可以复用这个基类中的功能。这个基类ActionForm的具体代码如下。



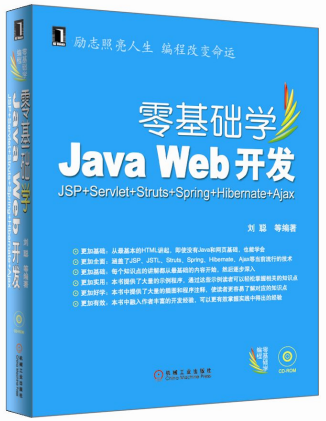
17.6.3　JSP+JSTL+Struts标签库实现信息展示

- 在Hibernate Jpetstore中，使用传统的JSP表态包含指令，来包含进公共部分，如页眉，页脚及导航区域等。所有以 Include 前缀命名的JSP都用来被其他JSP页面包含的页面块。在这个示例程序中，所有的页面结构如下所示。



17.6.4　在Struts中防止重复提交

- 在Struts中可以采用事务Token来放置用户表单的重复提交，在Hibernate Jpetstore这个示例系统中，用户提交订单的例子中就使用了事务Token来放置用户的重复提交。在这个用户提交表单的时候，要把用户输入的表单展示在确认页面中，如果点’Continue’按钮，才会将订单插入到数据库中，在这个处理过程中采用事务Token来避免用户的重复提交。



17.7 小结

- 在本章中，以Hibernate Jpetstore为基础介绍了综合使用Struts、Spring、hibernate进行电子商务网站开发的基本方法。在这个示例项目中，不仅综合使用了这三种技术，而且还使用了其他的技术，在这个章节中仅仅是通过这个示例程序说明开发电子商务应用系统的基本方法，读者可以通过这个项目仔细揣摩，也可以到Hibernate Jpetstore项目的官方网站https://hjpetstore.dev.java.net/查看关于这个项目更多的信息。