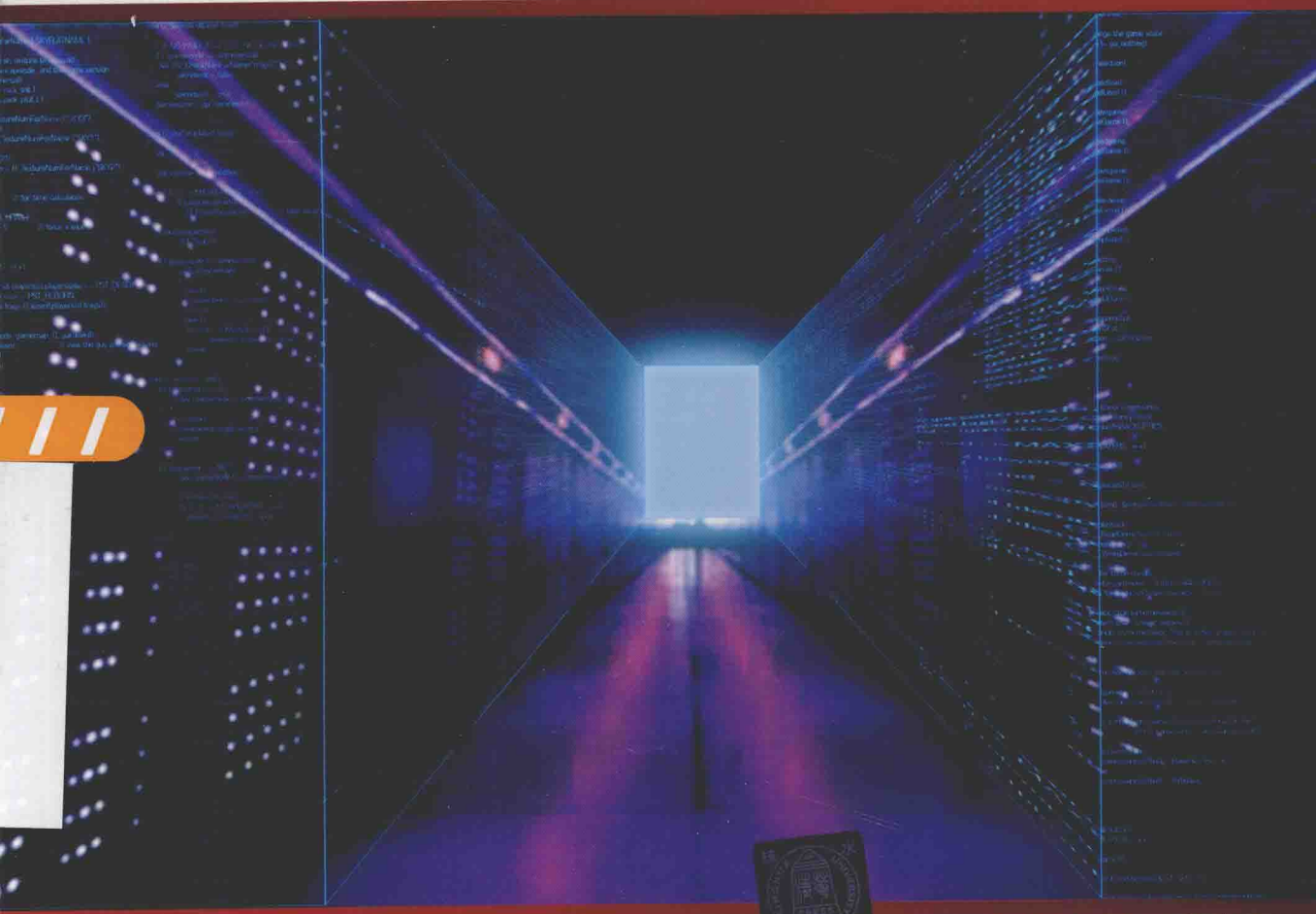
J轻量级 Java Web整合开发入门

——Struts 2+Hibernate 4+Spring 3

段鹏松　李占波　主　编
张　晗　曹仰杰　宋　冰　副主编



清华大学出版社

轻量级 Java Web 整合开发入门——
Struts 2+Hibernate 4+Spring 3

主　编　段鹏松　李占波
副主编　张　晗　曹仰杰　宋　冰

清华大学出版社

内 容 简 介

SSH(Struts、Spring、Hibernate)框架是目前 Java Web 开发中应用非常广泛的开源框架组合，基于 SSH 框架，开发人员可以在短期内搭建结构清晰、可复用性好、维护方便的 Java Web 应用程序。

本书详细讲解了 Struts 2、Hibernate 和 Spring 的基本用法，及其相互之间的整合流程，可以作为初学者学习 Java EE 整合开发的入门教程。全书共 7 章，可分为 3 部分：第 1～2 章是第 1 部分，介绍了 Java EE 开发的基础知识以及一些常见的设计模式；第 3～5 章是第 2 部分，详细介绍了 Struts 2 框架、Hibernate 框架和 Spring 框架的概念及基本使用方法，该部分内容是本书的核心；第 6～7 章是第 3 部分，主要介绍 SSH 框架的整合流程，该部分是作者多年使用 SSH 框架整合过程的经验总结，以及对一些典型整合中可能遇到问题的归纳总结，希望读者在整合的过程中，少走弯路，提高效率。

本书介绍的 Struts 框架的版本为 Struts 2.3.16，Hibernate 框架的版本为 Hibernate 4.2.0，Spring 框架的版本为 Spring 3.0。因为不同版本相互整合时可能会存在一些兼容性问题，所以若以本书作为学习 Java EE 框架的教程，或是运行本教程附带源代码时，最好选择和本书一样的版本。

本书语言简洁，内容丰富，既可作为 SSH 框架初学者的入门教材，也可作为高等院校相关专业的教材和辅导用书。

图书在版编目(CIP)数据

轻量级Java Web整合开发入门——Struts 2+Hibernate 4+Spring 3/段鹏松，李占波 主编.
—北京：清华大学出版社，2015

ISBN 978-7-302-40111-7

Ⅰ. ①轻… Ⅱ. ①段… ②李… Ⅲ. ①JAVA语言—程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2015)第 089491 号

责任编辑：王　定　程　琪
封面设计：杜丽雅
版式设计：思创景点
责任校对：邱晓玉
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn,010-62794504

印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市少明印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：15　　**字　　数**：374 千字
版　　次：2015 年 9 月第 1 版　　**印　　次**：2015 年 9 月第1 次印刷
印　　数：1～3000
定　　价：38.00 元

产品编号：063454-01

前言

PREFACE

由于 Java 语言的跨平台性以及 Web 应用的广泛发展，Java EE 平台已经成为各大行业应用的首选开发平台。Java EE 开发可分为两种模式：一种是以 Spring 为核心的轻量级 Java EE 企业开发；另一种是以 EJB3+JPA 为核心的经典 Java EE 开发。无论使用哪种平台进行开发，应用的性能及稳定性都有很好的保证，开发人群也较多。近年来，随着开源力量的迅速增长，使用轻量级 Java EE 开发的人数和市场占有率基本上已经超过了经典 Java EE 开发，有后来居上之势。

本书的主要内容就是介绍轻量级 Java EE 开发的相关框架，主要包括 Struts 2、Hibernate 和 Sping 框架，以及这 3 个框架的整合流程，也称 SSH 整合开发。这种整合开发模式在保留经典 Java EE 应用架构、高度可扩展性、高度可维护性的基础上，降低了 Java EE 应用的开发和部署成本，对于大部分的中小型企业应用是首选。

目前市面上讲述 SSH 框架的书籍不少，但是有一个缺点就是内容过多，大多是七八百页的大部头书籍，让初学者望而生畏。编者从事 Java EE 开发已经超过 5 年，深感对于框架初学者来说，并不是需要一本把所有的知识点都全部包含，把所有相关内容都讲到的书籍，而是需要一本能引导他们掌握框架的基本知识和基本使用流程的书籍。有过开发经验的人都知道，要想非常透彻和深刻地理解框架，没有几个实际项目的锻炼是不可能的。鉴于此，本书不求大而全，但求基础和实用。本书具有以下特点：

- 以精炼的语言，讲述 SSH 框架的基础知识；
- 完整实例介绍+经验总结+详细操作步骤；
- 所讲内容不仅仅是 SSH 框架，也涉及 Java 领域常用的其他框架，如经典 Java EE 框架等；
- 使学生不仅掌握 SSH 框架，更要明白框架的原理，并能对 Java 开发常见的框架有一定的了解和认识。

从 2010 年开始教授 SSH 框架课程至今，从对框架的肤浅认识，到对框架的熟练掌握，再到能掌握其基本原理。一路过来，走了不少弯路。也深深地体会到，教会学生学习的方法比掌握知识更重要。谨以此书，把个人学习和教授框架的经验，与广大初学者分享，希望能帮助大家在框架学习的道路上少走弯路。

本书共 7 章，可以分为 3 部分。

第 1 部分(第 1 章和第 2 章)，Java EE 开发的基础知识。其中，第 1 章主要介绍 Java EE 开发的基础知识，以及经典 Java EE 开发和轻量级 Java EE 开发的概念以及区别；第 2 章主要介

绍一些常见的设计模式。实际上，框架的实现就是一系列设计模式的应用(如 Struts 2 框架就体现了 MVC 模式的思想，Spring 框架从整体来说实际是工厂模式的思想)，掌握了设计模式的原理，就能对框架的底层实现有更深刻的理解。

第 2 部分(第 3～5 章)，SSH 框架介绍。该部分内容是本书的核心。第 3 章主要介绍 Struts 2 框架的概念、下载方法、标签库以及一些高级应用；第 4 章主要介绍 Hibernate 框架相关的概念、基本用法以及高级应用；第 5 章主要介绍 Spring 框架的概念、基本用法以及高级应用。学习完这 3 个章节的内容后，可以分别掌握 Struts 2 框架、Hibernate 框架和 Spring 框架的的基本使用流程。但是该部分介绍的框架是相互独立的，若要掌握框架整合的知识，还需学习第 3 部分。

第 3 部分(第 6 章和第 7 章)，SSH 框架的整合流程。该部分是作者多年使用 SSH 框架整合过程的经验总结，以及对一些典型整合中可能遇到问题的归纳总结，希望读者在整合的过程中，提高效率，少走弯路。其中，第 6 章主要介绍 SSH 框架相互整合的流程，以及轻量级整合和经典整合的区别；第 7 章主要总结了一些 Java Web 开发中常见的问题，以及相应的解决方案。学习就是不断遇到问题，然后在解决问题的过程中不断提高的过程。

学习框架，要先学会使用，在此基础之上再深入了解其原理，理解其思想。编程时使用框架和盖房子使用框架是一个道理。修一个小房子不需要框架，甚至可以边修边设计，但是要盖高楼大厦，则必须要使用框架。对于写程序也是一样的道理，小程序使用框架有点“杀鸡用牛刀”的感觉，也没有必要。程序规模到一定程度后，为了程序的协同开发及后期的软件扩展和维护，则必须使用框架。或者可以这么说，使用框架就相当于站在了巨人的肩膀上，用得好，可以达到事半功倍的效果。

本书由段鹏松、李占波，以及负责制定编写大纲、规划各章节内容，并完成全书统稿以及代码调试工作。其中，段鹏松主编第 1、5、6、7 章，张晗主编第 2 章，曹仰杰主编第 3 章，宋冰主编第 4 章。此外，参与本书资料搜集和整理的还有史晓东、徐鹏飞、袁振风等人，在此，编者对他们表示衷心感谢。

由于时间仓促，加之编者水平有限，书中难免存在疏漏和不足之处，恳请读者批评、指正。

编　者

2015 年 5 月

目录

CONTENTS

第 1 章

轻量级Java Web开发概述

1.1 Java 概述

Java 语言是 Sun Microsystems 公司于 1995 年 5 月推出的一种完全面向对象的程序设计语言，由 James Gosling 和同事们共同研发。Java 平台一经推出，就受到开发者的广泛好评及大量使用，到目前为止，仍非常流行。以至于微软公司推出了与之竞争的 .NET 平台以及模仿 Java 的 C# 语言。2009 年，Sun 公司被 Oracle 公司收购，目前 Java 平台的维护和扩展都是由 Oracle 公司负责的。

运行 Java 程序必须先安装 JDK(Java Development Kit)。JDK 是整个 Java 的核心，包括 Java 运行环境、Java 工具和 Java 基础类库。从 JDK 5.0 开始，提供了泛型等非常实用的功能，其版本也不断更新，运行效率得到了非常大的提高。目前最新的 JDK 版本为 JDK 8.0。

Java 语言的风格十分接近 C、C++语言。Java 是一个纯粹面向对象的程序设计语言，它继承了 C++语言面向对象技术的核心内容；舍弃了 C 语言中容易引起错误的指针(以引用取代)、运算符重载(Operator Overloading)、多重继承(以接口取代)等特性，增加了垃圾回收器功能，用于回收不再被引用对象所占据的内存空间，使得程序员不用再为内存管理而担忧。在 Java 1.5 版本后，Java 又引入了泛型编程(Generic Programming)、类型安全的枚举、不定长参数和自动装/拆箱等语言特性。

Java 不同于一般的编译执行计算机语言和解释执行计算机语言。它首先将源代码编译成二进制字节码(Bytecode)，然后依赖各种不同平台上的虚拟机来解释执行字节码。从而实现了“一次编译、到处执行”的跨平台特性。不过，每次执行编译后的字节码需要消耗一定的时间，这同时也在一定程度上降低了 Java 程序的运行效率。

根据开发应用程序的不同类型，Java 分为 3 个开发体系。

- J2SE：Java 2 Platform，Standard Edition，主要开发桌面 Application 应用程序。

- J2EE：Java 2 Platform，Enterprise Edition，主要开发企业级的 Web 应用程序。
- J2ME：Java 2 Platform，Micro Edition，主要开发嵌入式设备的应用程序。

Java 语言的跨平台功能使其得到了广泛应用，这一点是 C#语言所不及的。虽然 Java 的执行效率相比其他语言有一定的下降，但是随着硬件性能的提升，这点效率损耗几乎可以忽略不计。

1.2 Java Web 开发概述

简言之，Java Web 开发是使用 Java 语言开发 Web 项目。一般来说，Web 项目包含服务器端和客户端。Java 的 Web 开发主要集中在服务器端，所使用的技术主要有 JSP、Servlet 以及一些集成的快速开发框架等。

Java 的 Web 框架虽然各不相同，但基本都遵循相同的思想：使用 Servlet 或者 Filter 拦截请求，使用 MVC 的思想设计架构，使用约定、XML 或 Annotation 实现配置，运用 Java 面向对象的特点，面向抽象实现请求和响应的流程，支持 JSP、Freemarker、Velocity 等视图。

Java Web 项目需要在容器内运行，这个容器就是支持 Java 的一些 Web 服务器，比如 Tomcat、Jetty、GlassFish、WebLogic、JBoss 等。不同类型的容器，其基本功能类似，但是对于 Java Web 项目的管理和扩展力度不同。

1.2.1 Java Web 项目基本结构

目前支持 Java 语言的集成开发环境(Integrated Development Environment，IDE)非常多，对于 Web 项目支持较好的 IDE 工具有 Eclipse、MyEclipse(实际上是对 Eclipse 插件的集成)、NetBeans 等。开发者使用 IDE 工具，可以快速高效地开发出适用于各种需求的 Web 应用程序。但是开发者不能太依赖于 IDE 工具，因为软件开发的主体从来都是人，而不是工具。IDE 工具可以把一些重复性的工作快速完成，提高开发者的工作效率，但是决不能替代人的作用。对于有经验的开发者，IDE 工具可以提高开发效率，但是对于初学者，IDE 工具掩盖了软件开发的基本步骤，使开发者愈加迷茫。建议初学者先不要使用 IDE，等掌握了 Java 程序运行的基本原理后再使用 IDE。

开发者应该对 Java Web 项目的基本结构非常了解，并且能理解 Java Web 项目的运行原理，这样才能从本质上掌握 Java Web 项目的开发基础。以下演示一个手动建立 Java Web 项目，并完成该项目部署的操作流程。

1. 手动建立 Java Web 项目

(1) 在任意路径下建立一个文件夹，名字可以是符合 Java 项目命名规则的任意名字，此处命名为 FirstWeb。

(2) 在 FirstWeb 文件夹内，分别建立一个 index.jsp 文件和 WEB-INF 文件夹，所建立的目录结构如图 1-1 所示。

(3) 在 WEB-INF 文件夹内，建立一个 web.xml 文件。

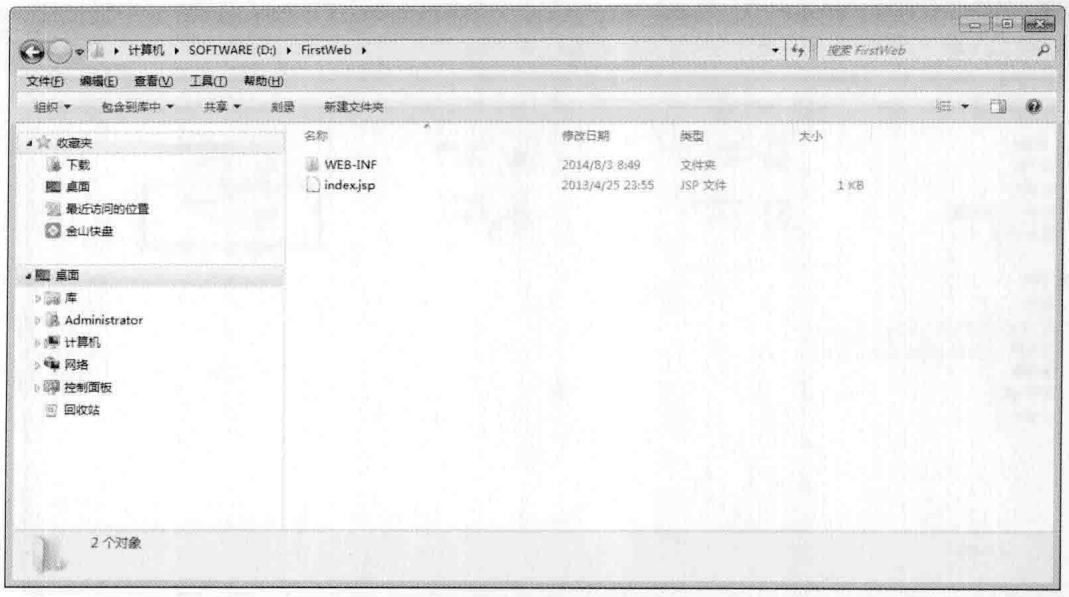


图 1-1 Web 项目目录结构

该项目中，index.jsp 文件的内容如下：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head><title>Simple jsp page</title></head>
<body>
    第一个 Java Web 项目
</body>
</html>
```

web.xml 文件的内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_9" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

至此，该 Java Web 项目的基本构建完成。该项目虽然简单，却是一个最基本的 Web 项目，可以直接运行。

2. 手动部署 Java Web 项目

把前面步骤中创建的 FirstWeb 文件夹复制到 Tomcat 的 webapps 目录下，即完成该项目的部署，如图 1-2 所示。

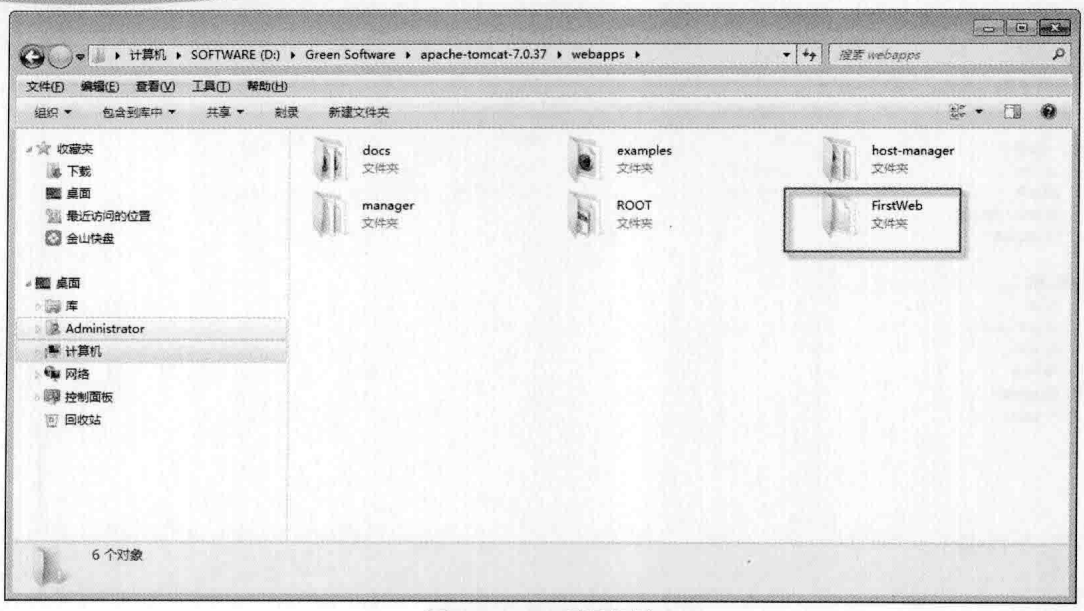


图 1-2　Web 项目部署

3. 测试 Java Web 项目

启动 Tomcat 服务器，测试该项目的运行，运行结果如图 1-3 所示。可以看到，第一个 Java Web 项目已经正常运行。

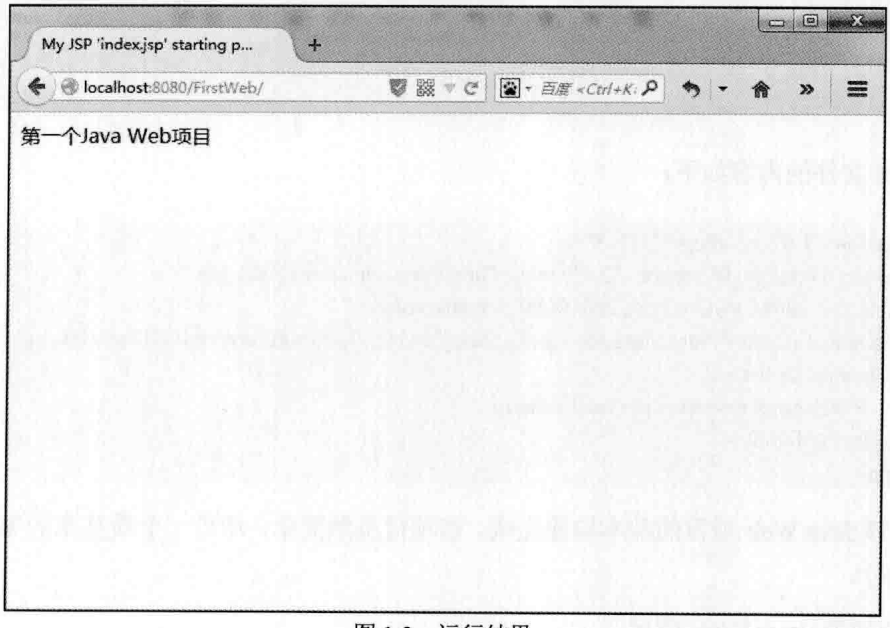


图 1-3　运行结果

4. Java Web 项目运行原理

从以上过程，可以看出一个 Java Web 项目的基本结构。一般来说，WEB-INF 是 Java Web 应用的安全目录，客户端无法访问，只有服务端可以访问。web.xml 文件为项目部署描述 XML

文件。如果想直接访问项目中的文件，必须通过 web.xml 文件对要访问的文件进行相应映射才能访问。WEB-INF 文件夹下除 web.xml 外，一般还有一个 classes 文件夹，用以放置 *.class 文件，有时还有 lib 文件夹，用于存放需要的 JAR 包。本演示项目因为功能非常简单，没有后台代码，也没有添加额外的 JAR 包，所以在 WEB-INF 下面没有 classes 文件夹和 lib 文件夹。

那么，一个 Web 项目究竟是如何运行的呢？打开图 1-4 所示的路径，可以看到，对于 index.jsp 文件，tomcat 容器相应地生成了一个 index_jsp.java 文件，并且把 Java 文件编译后又生成了一个 class 文件，即 Java 的字节码文件。

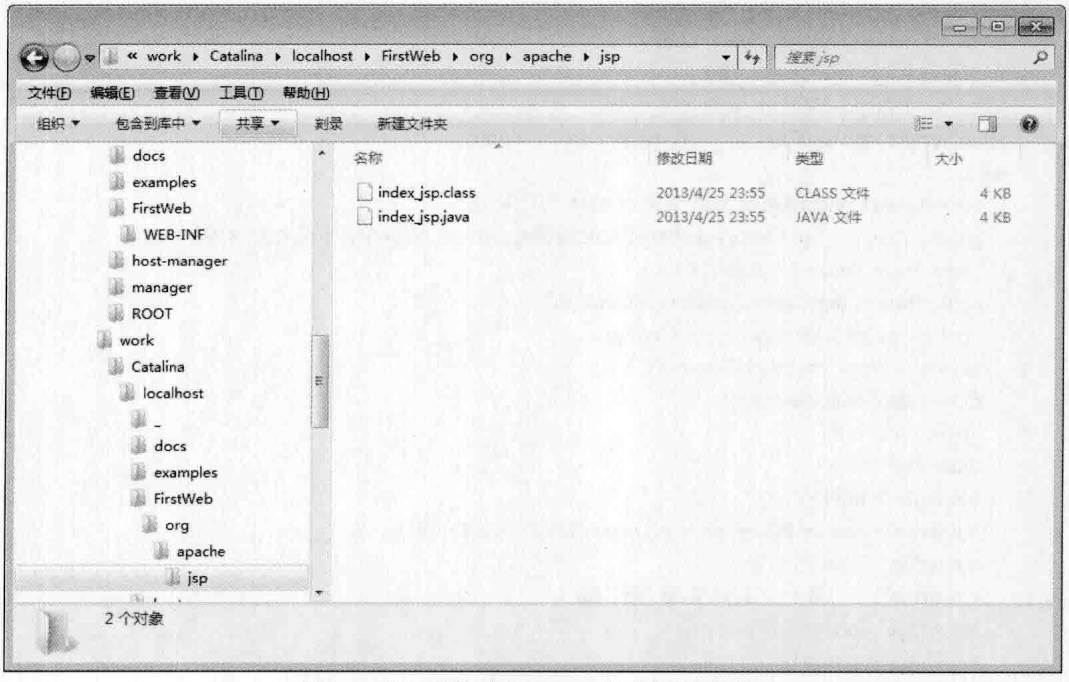


图 1-4　JSP 文件对应的 Java 类及字节码文件

index_jsp.java 文件的内容如下：

```
package org.apache.jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {
  private static final javax.servlet.jsp.JspFactory _jspxFactory=javax.servlet.jsp.JspFactory.getDefaultFactory();
  private static java.util.Map<java.lang.String,java.lang.Long> _jspx_dependants;
  private javax.el.ExpressionFactory _el_expressionfactory;
  private org.apache.tomcat.InstanceManager _jsp_instancemanager;
  public java.util.Map<java.lang.String,java.lang.Long> getDependants() {
    return _jspx_dependants;
  }
  public void _jspInit() {
    _el_expressionfactory=
        _jspxFactory.getJspApplicationContext(getServletConfig().getServletContext()).getExpressionFactory();
    _jsp_instancemanager=
```

```
            org.apache.jasper.runtime.InstanceManagerFactory.getInstanceManager(getServletConfig());
    }
    public void _jspDestroy() {  }
    public void _jspService(final javax.servlet.http.HttpServletRequest request,
            final javax.servlet.http.HttpServletResponse response)
            throws java.io.IOException, javax.servlet.ServletException {
      final javax.servlet.jsp.PageContext pageContext;
      javax.servlet.http.HttpSession session = null;
      final javax.servlet.ServletContext application;
      final javax.servlet.ServletConfig config;
      javax.servlet.jsp.JspWriter out = null;
      final java.lang.Object page = this;
      javax.servlet.jsp.JspWriter _jspx_out = null;
      javax.servlet.jsp.PageContext _jspx_page_context = null;
      try {
        response.setContentType("text/html;charset=UTF-8");
        pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
        _jspx_page_context = pageContext;
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();
        _jspx_out = out;
        out.write("\r\n");
        out.write("<html>\r\n");
        out.write("<head><title>Simple jsp page</title></head>\r\n");
        out.write("<body>\r\n");
        out.write("      第一个 Java Web 项目\r\n");
        out.write("</body>\r\n");
        out.write("</html>");
      } catch (java.lang.Throwable t) {
        if (!(t instanceof javax.servlet.jsp.SkipPageException)){
          out = _jspx_out;
          if (out != null && out.getBufferSize() != 0)
            try { out.clearBuffer(); } catch (java.io.IOException e) {}
          if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
          else throw new ServletException(t);
        }
      } finally {
        _jspxFactory.releasePageContext(_jspx_page_context);
      }
    }
}
```

可以看出，JSP 是一种编译执行的前台页面技术。对于每个 JSP 页面，Web 服务器都会生成一个相应的 Java 文件，然后再编译该 Java 文件，生成相应的 Class 类型文件。在客户端访问到的 JSP 页面，就是相应 Class 文件执行的结果。

至此，应该了解 JSP 页面运行的基本原理了。所有的 Web 项目，不管有多复杂，都是基于这个原理的。

1.2.2 轻量级 Java Web 开发概述

所谓轻量级，是指该组件或框架启动时依赖的资源较少，系统消耗较小，是一种相对的说法。轻量级框架侧重于减小开发的复杂度，相应地，它的处理能力便有所减弱(如事务功能弱，不具备分布式处理能力)，比较适用于开发中小型企业应用。采用轻量级框架，一方面可以尽可能地采用基于 POJO(Plain Old Java Object，简单 Java 对象)的方法进行开发，使应用不依赖于任何容器，这可以提高开发调试效率；另一方面轻量级框架多数是开源项目，开源社区提供了良好的设计和许多快速构建工具以及大量现成可供参考的开源代码，这有利于项目的快速开发。

一般说的轻量级 Java Web 开发，主要是指使用 Struts 2、Hibernate 和 Spring 这 3 个框架整合开发的 Web 项目开发模式，也就是本书所讲的 SSH 框架开发。目前，轻量级 Java Web 开发是使用较多的 Web 项目开发模式。

1.2.3 经典 Java Web 开发概述

所谓重量级，是指该组件或框架启动时依赖的资源较多，系统消耗较大，也是一种相对的说法。重量级框架复杂度较高，运行速度较慢，但是其提供的功能一般来说相比轻量级提供的功能要强大得多。EJB 框架就是一个重量级的框架，其强调高度伸缩性，适合于开发大型企业应用。在 EJB 体系结构中，一切与基础结构服务相关的问题和底层分配问题都由应用程序容器或服务器来处理，且 EJB 容器通过减少数据库访问次数以及分布式处理等方式提供了专门的系统性能解决方案，能够充分解决系统性能问题。

通常说的经典 Java Web 开发，是指使用 JSF+JPA+EJB 这 3 个框架进行的开发。经典 Java Web 模式在一般项目中使用较少，只有在大型的企业级应用项目中才会使用，而且由于其复杂性，入门较为困难，但是其中的一些设计理念和架构思想还是非常值得学习和借鉴的。

1.3 常用的 Java Web 服务器

Java Web 项目必须在容器里面运行，这个容器一般称为 Java Web 服务器。有了 Java Web 服务器，Java Web 项目才能通过网络被不同的用户所访问。以下对常用的 Java Web 服务器进行简单介绍。

1. Tomcat 服务器

官方网站：http://tomcat.apache.org；官方 Logo：  。

Tomcat 是 Apache 软件基金会(Apache Software Foundation)的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有了 Sun 的参与和支持，最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现。因为 Tomcat 技术先进、性能稳定，而且免费，因而深受 Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的 Web 应用服务器。

2. GlassFish 服务器

官方网站：http://glassfish.java.net；官方 Logo：  。

GlassFish 是一款强健的商业兼容应用服务器，达到产品级质量，可免费用于开发、部署和重新分发。它基于 Sun Microsystems 提供的 Sun Java System Application Server PE 9 的源代码以及 Oracle 贡献的 TopLink 持久性代码。因为 GlassFish 由 Sun 公司(已被 Oracle 公司收购)直接负责开发维护，所以其对 Java 企业级开发的最新规范总是最先支持的。但是 GlassFish 对 Java Web 项目的配置有限，所以使用较少。

3. WebLogic

官方网站：http://www.bea.com。

WebLogic 是美国 BEA 公司出品的一个 application server，确切地说，是一个基于 Java EE 架构的中间件，BEA WebLogic 是用于开发、集成、部署和管理大型分布式 Web 应用、网络应用和数据库应用的 Java 应用服务器，将 Java 的动态功能和 Java Enterprise 标准的安全性引入大型网络应用的开发、集成、部署和管理之中。2008 年 1 月 16 日，BEA 公司被 Oracle 公司收购。

4. JBoss

官方网站：http://www.jboss.org。

JBoss 是全世界开发者共同努力的成果，一个基于 J2EE 的开放源代码的应用服务器。因为 JBoss 代码遵循 LGPL 许可，可以在任何商业应用中免费使用，而不需支付费用。2006 年，JBoss 公司被 Red Hat 公司收购。JBoss 是一个管理 EJB 的容器和服务器，支持 EJB 1.1、EJB 2.0 和 EJB 3.0 的规范。但 JBoss 核心服务不包括支持 Servlet/JSP 的 Web 容器，一般与 Tomcat 或 Jetty 绑定使用。

5. WebSphere

官方网站：www.ibm.com/software/websphere。

WebSphere 是 IBM 的服务器平台，它包含编写、运行和监视全天候的工业强度的 Web 应用程序和跨平台、跨产品解决方案所需要的整个中间件基础设施，如服务器、服务和工具。WebSphere 提供了可靠、灵活和健壮的软件。

6. Jetty

官方网站：http://www.eclipse.org/jetty；官方 Logo：  。

Jetty 是一个开源的 Servlet 容器，它为基于 Java 的 Web 内容(例如 JSP 和 Servlet)提供运行环境。Jetty 是使用 Java 语言编写的，它的 API 以一组 JAR 包的形式发布。开发人员可以将 Jetty 容器实例化成一个对象，可以迅速为一些独立运行(stand-alone)的 Java 应用提供网络和 Web 连接。

1.4 轻量级 Java Web 开发环境

1.4.1 环境变量的配置

环境变量一般是指在操作系统中用来指定系统运行环境的一些参数，比如临时文件夹位置和系统文件夹位置等。使用 Java 语言进行程序开发时，需要进行一些环境变量的配置，具体操作步骤如下：

(1) 新建 JAVA_HOME 环境变量，其值设置为 JDK 的安装路径，如图 1-5 所示。

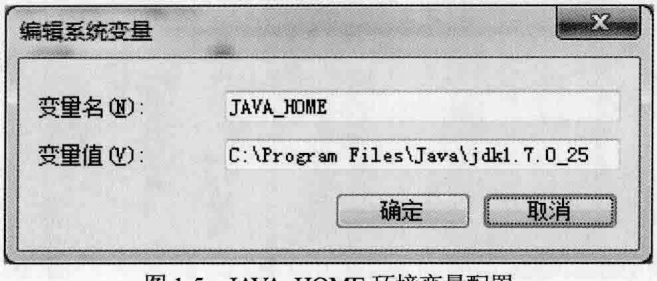


图 1-5 JAVA_HOME 环境变量配置

(2) 新建 classpath 环境变量，其值设置为".;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar"，如图 1-6 所示。

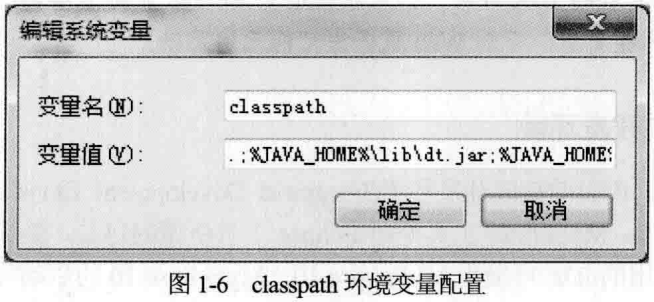


图 1-6 classpath 环境变量配置

(3) 编辑 Path 环境变量，在现有值的基础上添加"%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin"，如图 1-7 所示。

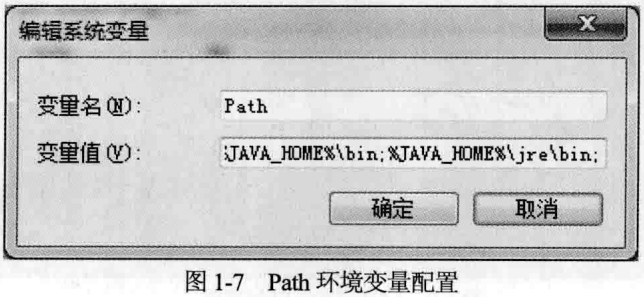


图 1-7 Path 环境变量配置

至此，Java 开发基本的环境变量配置完毕，可以通过在 cmd 中使用 javac 或 java 命令来测