“十二五”职业教育国家规划教材
经全国职业教育教材审定委员会审定

应用型本科 计算机系列规划教材

基于Struts、Hibernate、Spring架构的Web应用开发（第2版）

◆ 范新灿 主编
◆ 刘凯洋 聂 哲 黄 新 副主编

SOFTWARE

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

配备
电子课件、源代码

“十二五”职业教育国家规划教材
经全国职业教育教材审定委员会审定

应用型本科计算机系列规划教材

基于 Struts、Hibernate、Spring 架构的 Web 应用开发

（第 2 版）

范新灿　主　编
刘凯洋　聂　哲　黄　新　副主编

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

内 容 简 介

作为当今最为实用的框架组合SSH（Struts+Hibernate+Spring），其实用性、优越性已经得到认可，并在Java Web应用开发中得到广泛应用。本书以Struts 2为重点进行深入剖析，采用技术专题分类、项目牵引的方式撰写，注重实例与应用技术点的结合。Hibernate章节的讲解以实际项目的应用展开，Spring技术讲解抽取核心的IOC、AOP、Spring MVC技术通过实例解析，并实例讲解了Spring与Struts的整合开发。

本书适用于中、高级系统程序员，可作为高职或本科教材使用，也可作为有一定经验的Java Web编程者和学习者的参考书。

图书在版编目（CIP）数据

基于 Struts、Hibernate、Spring 架构的 Web 应用开发 / 范新灿主编. —2 版. —北京：电子工业出版社，2014.9
“十二五”职业教育国家规划教材

ISBN 978-7-121-24133-8

Ⅰ. ①基… Ⅱ. ①范… Ⅲ. ①软件工具—程序设计—高等职业教育—教材②JAVA 语言—程序设计—高等职业教育—教材 Ⅳ. ①TP311.56②TP312

中国版本图书馆 CIP 数据核字（2014）第 191947 号

策划编辑：徐建军（xujj@phei.com.cn）
责任编辑：郝黎明
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：19.75　字数：505.6 千字
版　　次：2011 年 8 月第 1 版
　　2014 年 9 月第 2 版
印　　次：2014 年 9 月第 1 次印刷
印　　数：3 000 册　　定价：39.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前言

在基于 J2EE 应用程序开发过程中，难于控制开发进度、开发效率低下、部署环境复杂、维护困难等问题层出不穷。对于中小企业，使用完整的 J2EE 实现过于庞大，最终常导致开发的失败。J2EE 轻量级框架 Struts+Spring+Hibernate 应运而产生，并逐渐流行。表现层用 Struts，Struts 充当视图层和控制层；业务层用 Spring，Spring 通过控制反转让控制层间接调用业务逻辑层；持久层用 Hibernate，Hibernate 充当数据访问层。每个层在功能上职责明确，不应该与其他层混合，通过通信接口而相互联系。

本书的组织结构。

本书共 10 章，从内容安排上可以分为六个部分。

第一部分是第一章，该章首先对软件架构进行定义，并系统阐述了 Web 应用发展的进程，从 JSP 开发的 Model 1、Model 2 讲解到 MVC 的开发思想。重点对 J2EE 轻量级框架 Struts+Spring+Hibernate 进行介绍，从结构到各层的技术实现进行深入剖析。

第二部分是第二章，该章讲解了 SSH 框架技术的应用开发环境安装和配置，该章首先介绍了 MyEclipse 开发平台的安装和配置，并以用户登录程序的开发过程，实例演练了如何熟练利用 MyEclipse 平台进行开发。

第三部分包括第三、四、五、六、七章，这五章以技术专题的方式讲解了 Struts 2 关键技术，包括框架拦截器、类型转换、国际化、输入校验。通过这些技术的实例学习，读者不仅从理论上认识和理解 Struts 2，并能实际进行 Struts 2 的基本开发。

第四部分是第八章，该章首先通过 ORM 和数据持久化来帮助读者认识 Hibernate，并通过开发关键技术的讲解和留言板程序的开发，掌握 JDBC 主流持久化框架。

第五部分是第九章，该章阐述了控制反转（IOC）和面向切面编程（AOP）思想，并通过实例讲解了如何进行开发。对于 Spring 的关键组成 Bean 和容器的实例化和生命周期实例解析。该章重点实例演练了 Spring 的 MVC 框架开发和 Spring 与 Struts 2 的整合开发。

第六部分是第十章，该章采用 SSH 开发框架组合，开发了怀听音乐网站，网站功能完善，设计合理，性能稳定，读者可以在实例实现中进一步锤炼 SSH 开发能力。

本书的特色。

1．丰富的实例引导知识点，将繁杂枯燥的概念融入到实例中，以项目驱动教材的延伸。

2．抽取典型应用，进而以点带面，以面贯穿知识体系。

3．注重启发性、实用性、渐进性

4．适合高职学校教材，将高职教育的理念融入教材的编写中，各章节注重内容的取舍与教学学时、能力点培养的对应

致谢

本书的编撰花费了一年多的时间，在这期间感谢家人的支持，感恩女儿出生带给我的快

乐，感谢同事无私的帮助。聂哲教授提出许多宝贵的建议，徐人凤院长给予的启示与帮助，袁梅冷老师帮我一起研究教材的编撰思路，帮助规划目录层次，赵明与我一起奋斗，编写了第一章和第十章，曾建华、陈健、刘凯洋、肖正兴等老师给予许多无私的帮助。

内容编排

代码导读。对于代码比较重要而不容易理解的内容，在代码前使用标注文字，然后在代码导读中进行解释。

代码导读

①

②

注意。用于强调当前问题的附加信息和注意事项。

注意

技巧。提供编程捷径、技巧和经验。

技巧

链接。对于实例或知识点涉及的内容，为了避免重复，又能让读者方便找到相关的技术解答。通过链接提供对重复内容的快速索引。

链接

技术细节。重点介绍开发过程中用到的关键技术或方法。

技术细节

为了方便教师教学，本书配有电子教学课件，请有此需要的教师登录华信教育资源网（www.hxedu.com.cn）注册后免费进行下载，有问题时可在网站留言板留言或与电子工业出版社联系（E-mail:hxedu@phei.com.cn）。

由于对项目式教学法正处于经验积累和改进过程中，同时，由于编者水平有限和时间仓促，书中难免存在疏漏和不足。希望同行专家和读者能给予批评和指正。

编　者

目 录

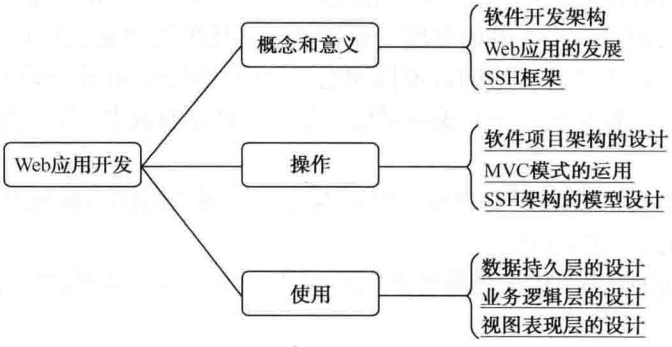
第 1 章 Web 应用开发

学习目标

软件架构（Software Architecture）是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计。Web 应用的发展也经历了不同的阶段，Web 开发框架技术的出现，促进了项目的开发效率。读者通过本章的学习可以掌握以下内容：

- 了解软件架构的定义。
- 了解 Web 应用的发展。
- 掌握 SSH 框架技术的模型及设计。

内容框架



1.1 软件开发架构

1. 软件架构的历史

早在 20 世纪 60 年代，诸如 E·W·戴克斯特拉就已经涉及软件架构这个概念了。自 20 世纪 90 年代以来，部分由于在 Rational Software Corporation 和 MiCROSoft 内部的相关活动，软件架构这个概念开始越来越流行起来。

卡内基梅隆大学和加州大学埃尔文分校在这个领域作了很多研究。卡内基·梅隆大学的 Mary Shaw 和 David Garlan 于 1996 年写了一本叫做 Software Architecture perspective on an emerging DIscipline 的书，提出了软件架构中的很多概念，例如软件组件、连接器、风格等等。加州大学埃尔文分校的软件研究院所做的工作则主要集中于架构风格、架构描述语言以及动态

架构。

计算机软件的历史开始于20世纪50年代，历史非常短暂，而相比之下建筑工程则从石器时代就开始了，人类在几千年的建筑设计实践中积累了大量的经验和教训。建筑设计基本上包含两点，一是建筑风格，二是建筑模式。独特的建筑风格和恰当选择的建筑模式，可以使得一个建筑独一无二。几乎所有的软件设计理念都可以在浩如烟海的建筑学历史中找到更为遥远的历史回响。

图1-1所示为中美洲古代玛雅建筑——Chichen-Itza大金字塔，9个巨大的石级堆垒而上，91级台阶（象征着四季的天数）夺路而出，塔顶的神殿耸入云天。所有的数字都如日历般严谨，风格雄浑。与此类似地，自从有了建筑以来，建筑与人类的关系就一直是建筑设计师必须面对的核心问题。

图1-1　位于墨西哥的古玛雅建筑

软件与人类的关系是架构师必须面对的核心问题，也是自从软件进入历史舞台之后就出现的问题。与此类似地，自从有了建筑以来，建筑与人类的关系就一直是建筑设计师必须面对的核心问题。在软件设计界曾经有很多人认为功能是最为重要的，形式必须服从功能。与此类似地，在建筑学界，现代主义建筑流派的开创人之一Louis Sullivan也认为形式应当服从于功能（FORMs follows function）。

2. 软件架构的定义

软件架构（software architecture）是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计。软件架构是一个系统的草图。软件架构描述的对象是直接构成系统的抽象组件。各个组件之间的连接则明确和相对细致地描述组件之间的通讯。在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象。在面向对象领域中，组件之间的连接通常用接口来实现。

正如同软件本身有其要达到的目标一样，架构设计要达到的目标是什么呢？一般而言，软件架构设计要达到如下的目标：

（1）可靠性（Reliable）。软件系统对于用户的商业经营和管理来说极为重要，因此软件系统必须非常可靠。

（2）安全性（Secure）。软件系统所承担的交易的商业价值极高，系统的安全性非常重要。

（3）可扩展性（Scalable）。软件必须能够在用户的使用率和用户数目增加很快的情况下，保持合理的性能。只有这样，才能适应用户的市场扩展。

（4）可定制化（Customizable）。同样的一套软件，可以根据客户群的不同和市场需求的变化来进行调整。

（5）可扩展性（Extensible）。在新技术出现的时候，一个软件系统应当允许导入新技术，从而对现有系统进行功能和性能的扩展。

（6）可维护性（Maintainable）。软件系统的维护包括两个方面，一是排除现有的错误，二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的花费。

（7）客户体验（Customer Experience）。软件系统必须易于使用。

（8）市场时机（Time to Market）。软件用户要面临同业竞争，软件提供商也要面临同业竞争。以最快的速度争夺市场先机则非常重要。

基于 Java 技术的软件开发架构，宏观上的层次如图 1-2 所示。

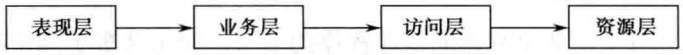


图 1-2 软件开发架构

在具体的实现中，表现层可为 Struts/JSF 等，业务层、访问层可为 JavaBean 或 EJB 等，资源层一般为数据库。

3. 种类

根据我们关注的角度不同，可以将架构分成三种：

■ 逻辑架构

软件系统中元件之间的关系，比如用户界面，数据库，外部系统接口，商业逻辑元件，等等。

系统被划分成三个逻辑层次，即表象层次，商业层次和数据持久层次。每一个层次都含有多个逻辑元件。比如 Web 服务器层次中有 HTML 服务元件、Session 服务元件、安全服务元件、系统管理元件等。

■ 物理架构

软件元件是怎样放到硬件上的。一般包括网络分流器、代理服务器、Web 服务器、应用服务器、报表服务器、整合服务器、存储服务器、主机等等。

■ 系统架构

系统的非功能性特征，如可扩展性、可靠性、强壮性、灵活性、性能等。

4. 软件架构师

软件架构师（Software Architect）是软件行业中一种新兴职业，工作职责是在一个软件项目开发过程中，将客户的需求转换为规范的开发计划及文本，并制定这个项目的总体架构，指导整个开发团队完成这个计划。主导系统全局分析设计和实施、负责软件构架和关键技术决策的人员。

（1）能力要求

软件架构师能应技术全面、成熟练达、洞察力强、经验丰富，具备在缺乏完整信息、众多问题交织一团、模糊和矛盾的情况下，迅速抓住问题要害，并做出合理的关键决定的能力，具备战略性和前瞻性思维能力，善于把握全局，能够在更高抽象级别上进行思考。主要包括如下：

① 对项目开发涉及的所有领域都有经验，包括彻底地理解项目需求，开展分析设计之类软件工程活动等；

② 具备领导素质，能在各小组之间推进技术工作，并在项目压力下做出牢靠的关键决策；

③ 拥有优秀的沟通能力，用以进行说服、鼓励和指导等活动，并赢得项目成员的信任；

④不带任何感情色彩地以目标导向和主动的方式来关注项目结果，构架师应当是项目背后的技术推动力，而非构想者或梦想家（追求完美）；

⑤ 精通构架设计的理论、实践和工具，并掌握多种参考构架、主要的可重用构架机制和模式（例如 J2EE 架构等）；

⑥ 具备系统设计员的所有技能，但涉及面更广、抽象级别更高；活动包括确定用例或需求的优先级、进行构架分析、创建构架的概念验证原型、评估构架的概念验证原型的可行性、

组织系统实施模型、描述系统分布结构、描述运行时刻构架、确定设计机制、确定设计元素、合并已有设计元素、构架文档、参考构架、分析模型、设计模型、实施模型、部署模型、构架概念验证原型、接口、事件、信号与协议等。

（2）主要任务

架构师的主要任务不是从事具体的软件程序的编写，而是从事更高层次的开发构架工作。他必须对开发技术非常了解，并且需要有良好的组织管理能力。可以这样说，一个架构师工作的好坏决定了整个软件开发项目的成败。

- 领导与协调整个项目中的技术活动（分析、设计和实施等）
- 推动主要的技术决策，并最终表达为软件构架
- 确定和文档化系统的相对构架而言意义重大的方面，包括系统的需求、设计、实施和部署等“视图”
- 确定设计元素的分组以及这些主要分组之间的接口
- 为技术决策提供规则，平衡各类涉众的不同关注点，化解技术风险，并保证相关决定被有效传达和贯彻
- 理解、评价并接收系统需求
- 评价和确认软件架构的实现、专业技能

1.2 Web 应用的发展

1.2.1 Web 技术的发展

随着 Internet 技术的广泛使用，Web 技术已经广泛应用于 Internet，但早期的 Web 应用全部是静态的 HTML 页面，用于将一些文本信息呈现给浏览者，但这些信息是固定写在 HTML 页面里的，该页面不具备与用户交互的能力，没有动态显示的功能。

于是，人们希望 Web 应用里包含一些能动态执行的页面，最早的 CGI（通用网关接口）技术满足了该要求，CGI 技术使得 Web 应用可以与客户端浏览器交互，不再需要使用静态的 HTML 页面。CGI 技术可以从数据库中读取信息，将这些信息呈现给用户；还可以获取用户的请求参数，并将这些参数保存到数据库里。

CGI 技术开启了动态 Web 应用的时代，给予了这种技术无限的可能性。但 CGI 技术存在很多缺点，其中最大的缺点就是开发动态 Web 应用难度非常大，而且在性能等各方面也存在限制。到 1997 年，随着 Java 语言的广泛使用，Servlet 技术迅速成为动态 Web 应用的主要开发技术。与传统的 CGI 应用相比，Servlet 具有大量的优势：

（1）Servlet 是基于 Java 语言创建的，而 Java 语言则内建了多线程支持，这一点大大提高了动态 Web 应用的性能。

（2）Servlet 应用可以充分利用 Java 语言的优势，如 JDBC（Java DataBase Connection）等。同时，Java 语言提供了丰富的类库，这些都简化了 Servlet 的开发。

（3）除此之外，Servlet 运行在 Web 服务器中，由 Web 服务器去负责管理 Servlet 的实例化，并对客户端提供多线程、网络通信等功能，这都保证 Servlet 有更好的稳定性和性能。

Servlet 在 Web 应用中被映射成一个 URL（统一资源定位），该 URL 可以被客户端浏览器

请求，当用户向指定 URL 对应的 Servlet 发送请求时，该请求被 Web 服务器接收到，该 Web 服务器负责处理多线程、网络通信等功能，而 Servlet 的内容则决定了服务器对客户端的响应内容。

图 1-3 所示为 Servlet 的响应流程，浏览器向 Web 服务器内指定的 Servlet 发送请求，Web 服务器根据 Servlet 生成对客户端的响应。

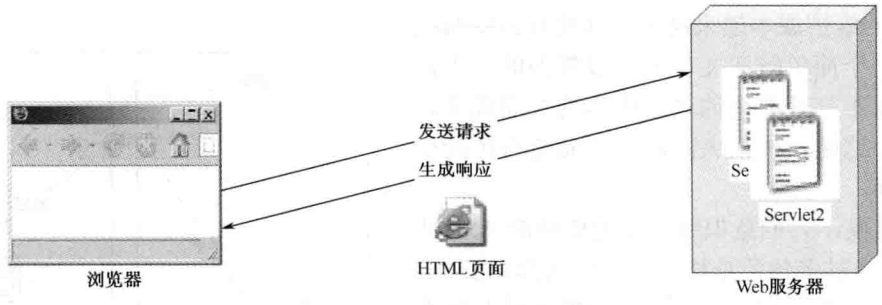


图 1-3 Servlet 的响应流程

实际上，这是后来所有的动态 Web 编程技术所使用的模型，这种模型都需要一个动态的程序或一个动态页面，当客户端向该动态程序或动态页面发送请求时，Web 服务器根据该动态程序来生成对客户端的响应。

到了 1998 年，微软发布了 ASP 2.0，它是 Windows NT 4 Option Pack 的一部分，作为 IIS 4.0 的外接式附件。它与 ASP 1.0 的主要区别在于它的外部组件是可以初始化的，这样，在 ASP 程序内部的所有组件都有了独立的内存空间，并可以进行事务处理。这标志着 ASP 技术开始真正作为动态 Web 编程技术。

当 ASP 技术在世界上广泛流行时，人们很快感受到这种简单技术的魅力：ASP 使用 VBScript 作为脚本语言，它的语法简单、开发效率非常高。而且，世界上已经有了非常多的 VB 程序员，这些 VB 程序员可以很轻易地过渡成 ASP 程序员。因此，ASP 技术马上成为应用最广泛的动态 Web 开发技术。

随后，由 Sun 带领的 Java 阵营，立即发布了 JSP 标准，从某种程度上来看，JSP 是 Java 阵营为了对抗 ASP 推出的一种动态 Web 编程技术。

ASP 和 JSP 从名称上如此相似，但它们的运行机制存在一些差别，这主要是因为 VBScript 是一种脚本语言，无需编译，而 JSP 使用 Java 作为脚本语句，但 Java 从来就不是解释型的脚本语言，因此 JSP 页面并不能立即执行。JSP 必须编译成 Servlet，也就是说：JSP 的实质还是 Servlet。不过，书写 JSP 比书写 Servlet 简单得多。

JSP 的运行机理如图 1-4 所示。

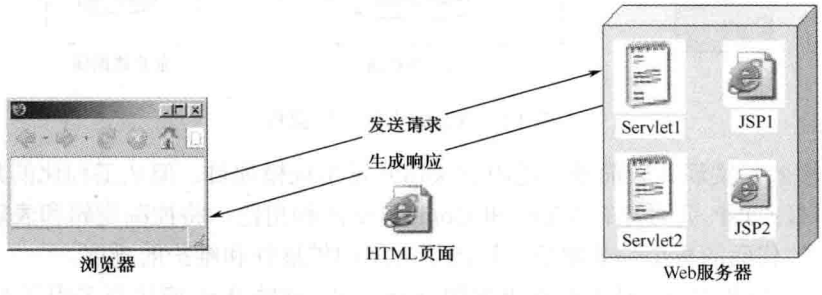


图 1-4 JSP 的运行机理

对比图 1-3 和图 1-4，发现无论是 Servlet 动态 Web 技术，还是 JSP 动态 Web 技术，它们的实质完全一样。可以这样理解：JSP 是一种更简单的 Servlet 技术，这也是 JSP 技术出现的意义——作为一个和 ASP 对抗的技术，简单就是 JSP 的最大优势。

随着实际 Web 应用的使用越来越广泛，Web 应用的规模也越来越大，开发人员发现动态 Web 应用的维护成本越来越大，即使只需要修改该页面的一个简单按钮文本或一段静态的文本内容，也不得不打开混杂的动态脚本的页面源文件进行修改，这是一种很大的风险，完全有可能引入新的错误。

这个时候，人们意识到，使用单纯的 ASP 或 JSP 页面充当过多角色是相当失败的选择，这对于后期的维护相当不利。慢慢地，开发人员开始在 Web 开发中使用 MVC 模式。

随后 Java 阵营发布了一套完整的企业开发规范：J2EE（现在已经更名为 Java EE），紧接着，微软也发布了 ASP.NET 技术，它们都采用一种优秀的分层思想，力图解决 Web 应用维护困难的问题。动态 Web 编程技术的发展历史如图 1-5 所示。

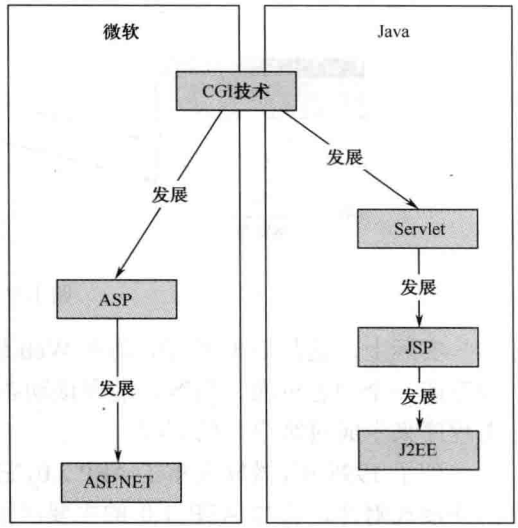


图 1-5　动态 Web 编程技术的发展历史

1.2.2 Model 1 和 Model 2

Java 阵营的动态 Web 编程技术经历了所谓的 Model 1 和 Model 2 时代。

Model 1 就是 JSP 大行其道的时代，在 Model 1 模式下，整个 Web 应用几乎全部由 JSP 页面组成，JSP 页面接收处理客户端请求，对请求处理后直接作出响应。用少量的 JavaBean 来处理数据库连接、数据库访问等操作。

图 1-6 所示为 Model 1 的程序流程。

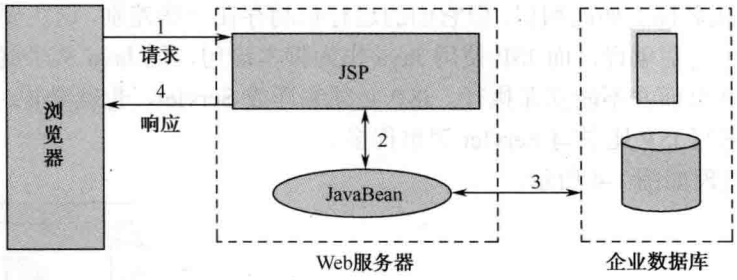


图 1-6　Model 1 的程序流程

Model 1 模式的实现比较简单，适用于快速开发小规模项目。但从工程化的角度看，它的局限性非常明显：JSP 页面身兼 View 和 Controller 两种角色，将控制逻辑和表现逻辑混杂在一起，从而导致代码的重用性非常低，增加了应用的扩展性和维护的难度。

早期有大量 ASP 和 JSP 技术开发出来的 Web 应用，这些 Web 应用都采用了 Model 1 架构。

Model 2 已经是基于 MVC 架构的设计模式。在 Model 2 架构中，Servlet 作为前端控制器，

负责接收客户端发送的请求，在 Servlet 中只包含控制逻辑和简单的前端处理；然后，调用后端 JavaBean 来完成实际的逻辑处理；最后，转发到相应的 JSP 页面处理显示逻辑。其具体的实现方式如图 1-7 所示。

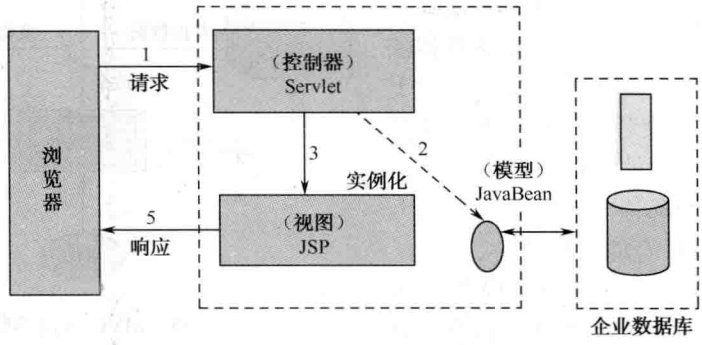


图 1-7 Model 2 的程序流程

从图 1-7 中可以看到，Model 2 下 JSP 不再承担控制器的责任，它仅仅是表现层角色，仅仅用于将结果呈现给用户，JSP 页面的请求与 Servlet（控制器）交互，而 Servlet 负责与后台的 JavaBean 通信。在 Model 2 模式下，模型（Model）由 JavaBean 充当，视图（View）由 JSP 页面充当，而控制器（Controller）则由 Servlet 充当。

由于引入了 MVC 模式，使 Model 2 具有组件化的特点，更适用于大规模应用的开发，但也增加了应用开发的复杂程度。原本需要一个简单的 JSP 页面就能实现的应用，在 Model 2 中被分解成多个协同工作的部分，需花费更多的时间才能真正掌握其设计和实现过程。

注意

对于非常小型的 Web 站点，如果后期的更新、维护工作不是特别多，可以使用 Model 1 的模式来开发应用，而不是使用 Model 2 的模式。虽然 Model 2 提供了更好的可扩展性及可维护性，但增加了前期开发成本。从某种程度上讲，Model 2 降低了系统后期维护的复杂度，却导致了前期开发的更高复杂度。

1.2.3 MVC

1. MVC 思想

MVC 并不是 Java 语言所特有的设计思想，也并不是 Web 应用所特有的思想，它是所有面向对象程序设计语言都应该遵守的规范。

MVC 思想将一个应用分成 3 个基本部分：Model（模型）、View（视图）和 Controller（控制器），这 3 个部分以最少的耦合协同工作，从而提高应用的可扩展性及可维护性。

最初，MVC 模式是针对相同的数据需要不同显示的应用而设计的，其整体效果如图 1-8 所示。

在 MVC 模式中，事件由控制器处理，控制器根据事件的类型改变模型或视图，反之亦然。具体来说，每个模型对应一系列的视图列表，这种对应关系通常采用注册来完成，即把多个视图注册到同一个模型，当模型发生改变时，模型向所有注册过的视图发送通知，接下来，视图