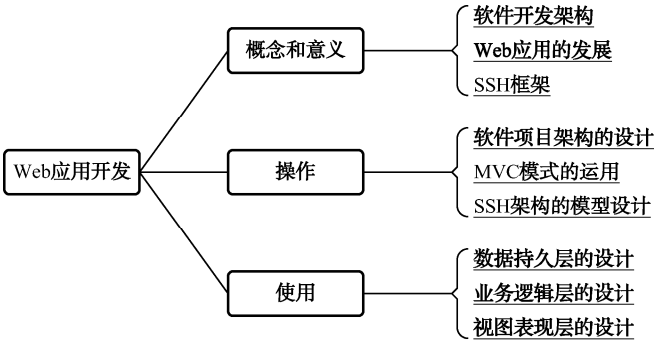
第1章 Web应用开发

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提高了项目的开发效率。通过本章的学习，读者可以掌握以下内容：

- 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 90 年

代以来，由于在 Rational 软件有限公司（Rational Software Corporation）和微软（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做了很多研究。卡内基 • 梅隆大学的 Mary Shaw 和 David Garlan 于 1996 年写了一本名为 *Software Architecture Perspective on an Emerging Discipline* 的书，提出了软件架构中的很多概念，如软件组件、连接器、风格等。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、架构描述语言及动态架构上。

图 1-1 所示为位于墨西哥的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。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　位于墨西哥的中美洲古代玛雅建筑

同样，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。在软件设计界曾经有很多人认为功能是最为重要的，形式必须服从于功能。与此类似，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，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信。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。

一般而言，软件架构设计要达到如下目标：

（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）安全行（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。

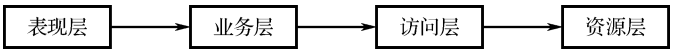


图 1-2　软件开发架构

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，可以将架构分为 3 种：

■ 逻辑架构

逻辑架构指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。

系统被划分成 3 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

物理架构指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、存储服务器和主机等。

■ 系统架构

系统架构指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）是软件行业中的一种新兴职业，是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。其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。

（1）能力要求。软件架构师要求技术全面、成熟练达、洞察力强、经验丰富，在缺乏完整信息、众多问题交织一团、模糊和矛盾的情况下，应具备迅速抓住问题要害，并做出合理的关键决定的能力，并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。主要能力要求如下：

① 对项目开发涉及的所有问题领域都有经验，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，以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 以目标导向和主动的方式不带任何感情色彩地关注项目结果，软件架构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 精通架构设计的理论、实践和工具，并掌握多种参考架构、主要的可重用架构机制和模式（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；确定用例或需求的优先级、进行架构分析、创建架构的概念验证原型、评估架构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、描述运行时刻架构、确定设计机制、确定设计元素、合并已有设计元素、架构文档、参考架构、分析模型、设计模型、实施模型、部署模型、架构概念验证原型、接口、事件、信号与协议等。

（2）主要任务。软件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架构工作。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，一个软件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，并最终表达为软件架构;
- 确定和文档化系统的相对架构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地传达和贯彻;
- 理解、评价并接收系统需求;
- 评价和确认软件架构的实现。

1.2 J2EE 轻量级框架 Struts+Spring+Hibernate

1.2.1 轻量级 J2EE 架构技术

轻量级是指一种开发方法，指简化的编程模型和更具响应能力的容器等。轻量级开发旨在消除传统 API 的复杂性与限制，同时，采用轻量级的方式进行开发也缩短了应用程序的开发周期与部署上的复杂度。

轻量级的软件开发不强迫业务对象遵循平台接口，可以使用 POJO 来实现业务。IOC 模式在轻量级的领域中起着巨大的作用，它的引入解决了对象依赖性的问题，有助于简化代码，将业务逻辑与基础架构分离，使应用程序高内聚、低耦合，从而使应用程序更易于维护，提高了开发效率，也使框架响应能力提高，达到了简化的目的。

为解决经典架构中的一系列问题，J2EE 逐渐流行起非 EJB 架构的“轻量级容器”，它与 EJB 架构一样，由容器管理业务对象，然后再组织整个架构，但是业务对象运行在“轻量级容器”中。轻量级容器不和 J2EE 绑定，既可运行在 Web 容器中，也可运行在一个标准应用程序中。

轻量级容器的启动开销很小，而且无须 EJB 部署，提供了一种管理、定位业务对象的方法，不必使用 JNDI 寻址、定制服务器之类的额外辅助，并为应用对象提供注册服务。轻量级容器较 EJB 功能强大，避免了容器强制业务对象采用特定的接口，降低了侵入性。

1.2.2　认识 SSH

SSH 在 J2EE 项目中表示 3 种框架，即 Spring + Struts +Hibernate。

在基于 J2EE 的应用程序开发过程中，难以控制开发进度，并且开发效率低下、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 应运而生，并逐渐流行，轻量级是和以 EJB 为核心的复杂框架对比而言的。轻量级框架致力于提供最简单的组件来构筑 Web 应用系统，Spring 是典型的一种轻量级架构，越来越多的开发人员开始关注并使用这种架构。通过 Spring 组合其他专一的开源产品，如表示层的 Struts、持久对象层的 Hibernate，来构建应用系统，实现 J2EE 简单化编程。Struts+Spring+Hibernate 框架体系结构图如图 1-3 所示。

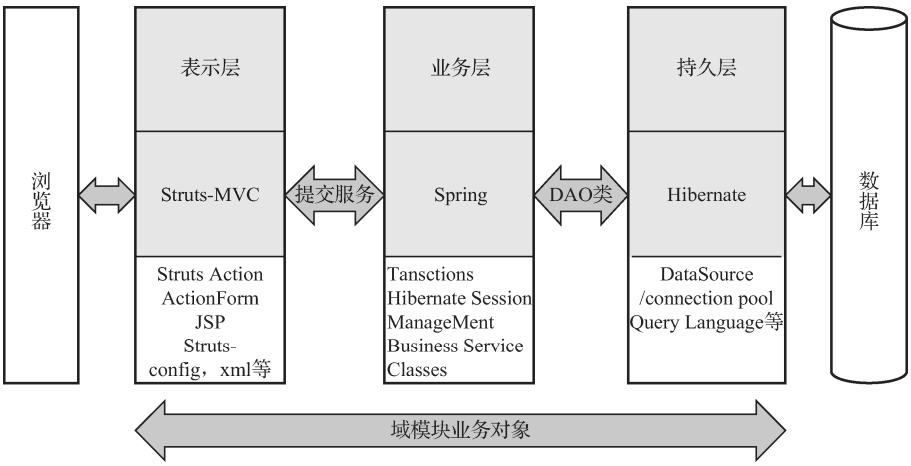


图 1-3　Struts+Spring+Hibernate 框架体系结构图

整体框架和业务层用 Spring，表示层用 Struts，而持久层用 Hibernate。Spring 是一个开放的框架，不要求一定要用 Spring 自己的解决方案。Struts 作为表示层的成熟技术已经在市场上广泛应用，可以很好地和 Spring 技术中间层紧密结合，Struts 可以使用 Spring 提供的事务处理等特性，所以选择 Struts 作为框架的表示层技术。Spring 按照资源管理的方法提供和 Hibernate 的集成及 DAO（Data Access Object）实现和事务策略支持，Spring 通过 IOC（控制反转）机制和 Hibernate 集成，Spring 能够很好地支持开发人员选择 O/R 映射技术。

1. Struts

如图 1-4 所示，Struts 对 Model、View 和 Controller 都提供了对应的组件。ActionServlet 类是 Struts 的核心控制器，负责拦截来自用户的请求。Action 类通常由用户提供，该控制器负责接收来自 ActionServlet 的请求，并根据该请求调用模型的业务逻辑方法处理请求，然后将处理结果返回 JSP 页面显示。

（1）Model 部分。Model 部分由 ActionForm 和 JavaBean 组成，其中 ActionForm 用于封装用户的请求参数，将其封装成 ActionForm 对象，该对象被 ActionServlet 转发给 Action，Action 根据 ActionForm 里面的请求参数处理用户的请求。JavaBean 则封装了底层的业务逻辑，包括数据库访问等。

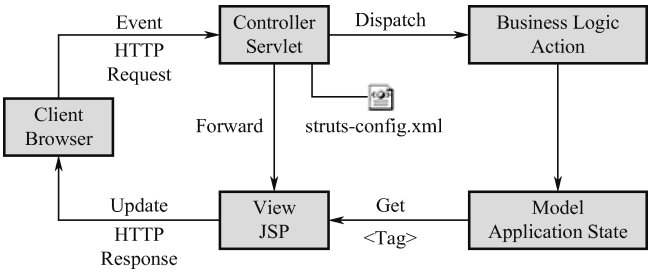


图 1-4　Struts 框架结构

（2）View 部分。该部分采用 JSP 实现。Struts 提供了丰富的标签库，通过标签库可以减少脚本的使用，自定义的标签库可以实现与 Model 的有效交互，并增加了现实功能。对应图中的 JSP 部分。

（3）Controller 组件。Controller 组件由两部分组成——系统核心控制器和业务逻辑控制器。

Struts 的优点在于实现了 MVC 模式，将 Web 系统各组件进行了良好的分工合作；Spring 的特性在于 IoC 机制；Hibernate 的长处在于数据持久化。要结合这 3 种技术的优点，采取的策略有许多种，如可以将 Hibernate 的 sessionFactory、数据操作组件交给 Spring 容器来管理，必要时进行注入处理；可以将 Struts 的 Action 交给 Spring 容器来管理，而不必再在 Action 中声明业务逻辑操作的组件。

2. Spring

Spring 是一个轻量级的控制反转（IOC）和面向切面（AOP）的容器框架。Spring 的目的是解决企业应用开发的复杂性，它使用基本的 JavaBean 代替 EJB，并提供更多的企业应用功能，可在任何 Java 中应用。

Spring 框架是一个分层架构，Spring 模块构建在核心容器之上，核心容器定义了创建、配置和管理 Bean 的方式，其体系结构如图 1-5 所示。

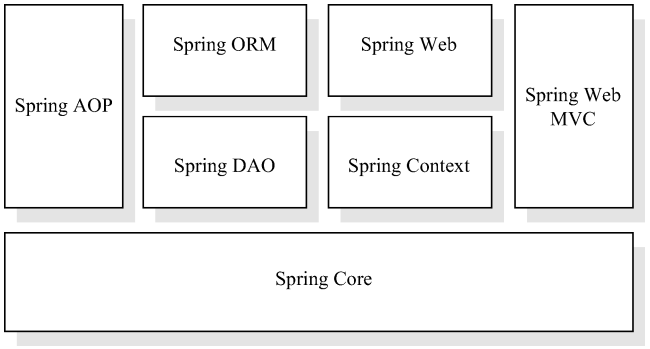


图 1-5　Spring 框架体系结构

从大小与开销两方面而言，Spring 都是轻量的。完整的 Spring 框架可以在一个大小只有 1MB 多的 JAR 文件里发布，并且 Spring 所需的处理开销也是微不足道的。此外，Spring 是非侵入式的，Spring 应用中的对象不依赖于 Spring 的特定类。

（1）控制反转。Spring 通过一种称作控制反转（IOC）的技术促进了松耦合。当应用了 IOC 时，一个对象依赖的其他对象会通过被动的方式传递进来，而不是这个对象自己创建或查找依赖对象。

（2）面向切面。Spring 提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务［如审计（Auditing）和事务（Transaction）管理］进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑，仅此而已。

（3）容器。Spring 包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，用户可以配置每个 Bean 如何被创建，基于一个可配置原型（Prototype），用户的 Bean 可以创建一个单独的实例或每次需要时都生成一个新的实例，以及它们是如何相互关联的。

（4）框架。Spring 可以将简单的组件配置、组合成为复杂的应用。在 Spring 中，应用对象被声明式地组合，典型的是在一个 XML 文件里。

所有 Spring 的这些特征使用户能够编写更干净、更可管理、更易于测试的代码。它们也为 Spring 中的各种模块提供了基础支持。

3. Hibernate

Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲地使用对象编程思维来操纵数据库。Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序中使用，也可以在 Servlet/JSP 的 Web 应用中使用。最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。Hibernate 的工作原理如图 1-6 所示。

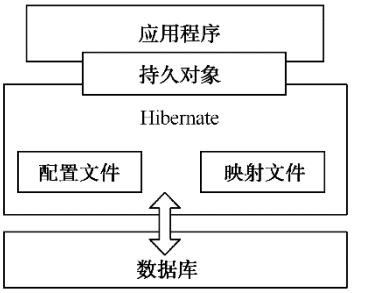


图 1-6　Hibernate 的工作原理

Hibernate 的核心接口一共有 5 个，分别为：Session、SessionFactory、Transaction、Query 和 Configuration。这 5 个核心接口在任何开发中都会用到。通过这些接口，不仅可以对持久化对象进行存取，还能够进行事务控制。

1.2.3　SSH 架构轻量级 Web 应用

SSH 架构是当前非常流行的架构，很多金融、电信项目和大型门户网站均选择该架构作为业务支撑的架构，开发流程已经非常成熟。如图 1-7 所示，SSH 由 3 个开源的框架组合而成，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，各层通过通信接口相互联系。

（1）Struts 负责 Web 层。

ActionFormBean 接收网页中表单提交的数据，然后通过 Action 进行处理，再发送到对应的网页，在 struts-config.xml 中定义<action-mapping>，ActionServlet 会加载。

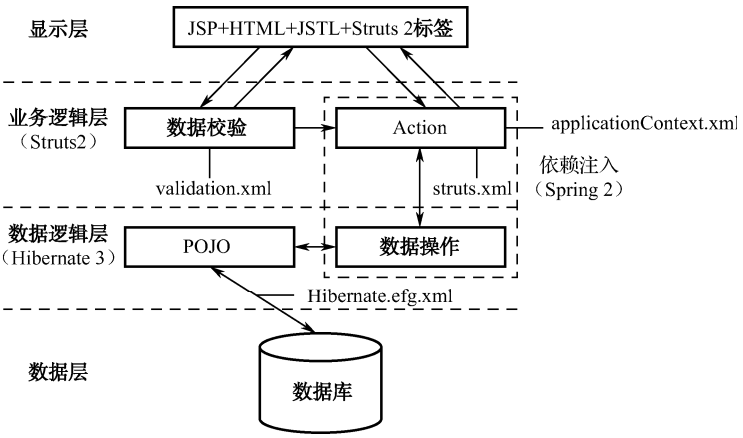


图 1-7　基于 Struts、Spring 和 Hibernate 的整合框架

（2）Spring 负责业务层管理，即 Service（或 Manager）。

Service 为 Action 提供统计的调用接口，封装持久层的 DAO，并集成了 Hibernate，Spring 可对 JavaBean 和事物进行统一管理。

（3）Hibernate 负责持久层，完成数据库的 CRUD 操作。

Hibernate 提供 OR/Mapping，它有一组 hbm.xml 文件和 POJO，是与数据库中的表相对应的，然后定义 DAO，这些是与数据库打交道的类，它们会使用 PO。

在 Struts+Spring+Hibernate 的系统中，对象的调用流程是：JSP→Action→Service→DAO→Hibernate，数据的流向是 ActionFormBean 接收用户的数据，Action 将数据从 ActionFormBean 中取出，封装成 VO 或 PO，再调用业务层的 Bean 类，完成多种业务处理后再发送。业务层 Bean 收到这个 PO 对象之后，会调用 DAO 接口方法，进行持久化操作。

1.3　总结与提高

本章介绍了软件开发架构，讲述了软件开发的历史、定义及种类。本章重点介绍了 J2EE 轻量级框架 Struts+Spring+Hibernate。

在传统的 J2EE 应用中，EJB 一直占据着主导地位，但运行它需要一个庞大的容器，通常称之为“重量级容器”。由于 EJB 暴露出的缺陷和复杂性，以“轻量级容器”为核心的架构 Struts+Spring+Hibernate 组合的开发解决了这个问题。SSH 架构表示层用 Struts，业务层用 Spring，持久层用 Hibernate，使开发更加简单、灵活，系统的维护也更加方便，使开发者更关注程序高层业务逻辑的实现，降低底层框架的设计考虑，提高了开发效率。