基于 J2EE 多层架构的 Web 开发框架研究*

李小平，肖岳峰，宿　元，宋瀚涛，姚永标
(北京理工大学 计算机学院，北京 100081)

摘　要：在经典的 J2EE 四层体系结构的基础上增加数据持久层，提出了基于 J2EE 五层体系结构的 Web 开发框架；分析了基于 Struts 框架的 J2EE 架构中实现对象持久性的局限性，从中分离出对象持久层，并将 Hibernate 这个面向对象的轻量级对象持久性技术集成到该架构中；应用 DAO 设计模式在业务逻辑层与持久层之间设计了多源数据访问组件，抽象和封装了对不同数据源的数据访问操作，实现对不同类型、结构、环境、用法的异构数据库的统一访问。

关键词：J2EE；模型—视图—控制器；Struts；Hibernate；多源数据
中图分类号：TP302.7　　**文献标志码**：A　　**文章编号**：1001-3695(2008)05-1429-03

Study of Web-based framework based on J2EE multi-tier architecture

LI Xiao-ping, XIAO Yue-feng, SU Yuan, SONG Han-tao, YAO Yong-biao
(School of Computer Science, Beijing Institute of Technology, Beijing 100081, China)

Abstract: This paper put forward J2EE 5-tier architecture by increasing data persistence tier based on classical J2EE 4-tier architecture. Analyzed the localization in implementing object persistence of the J2EE architecture based on Struts and divided the object persistence tier from it, then integrated Hibernate, a lightweight object oriented persistence technology, into the architecture, so that the advantages of each could be shared by all. A multi-source data accessing module was designed by applying the DAO design pattern between business tier and data persistence tier in this architecture to implement identically access various isomerous database with different types, structure, environment and usage.

Key words: J2EE; MVC; Struts; Hibernate; multi-source data

1　多层架构的 Web 开发框架模型

1.1　J2EE 五层体系结构设计

随着 Web 应用需要更复杂的表现和逻辑处理，J2EE 提出了一种四层体系结构，如图 1 所示。该结构分为客户层、Web 层、业务逻辑层、数据库层。应用的逻辑处理和表现相分离，使得系统具有更为逻辑流程清晰、功能代码复用性强、分布式部署的特点。但是在大多数的实际项目应用中，数据层是关系型数据库，不能明确映射要实现的面向对象的机制，业务逻辑层与数据库交互的过程需要对数据库调用接口作进一步的封装，这在项目的开发和维护上会增加一定的复杂性和管理的难度。一种有效的解决方案是把业务信息按照功能模块拆分开：业务逻辑与数据库访问分开，用户界面与业务逻辑分开，彼此相对独立，任何一方的改变均不会影响对方，或者影响不大。因此本文提出在业务逻辑层与数据库层之间增加一个数据持久层，将四层体系扩展为如图 2 所示的五层体系，即客户层、Web 层、业务逻辑层、数据持久层和数据库层。数据持久层位于数据库之上，隐藏数据读取和操纵中的所有数据访问代码细节，将客户应用程序与底层存储机制隔离开，完全抽象出开发应用程序时使用的数据物理细节。

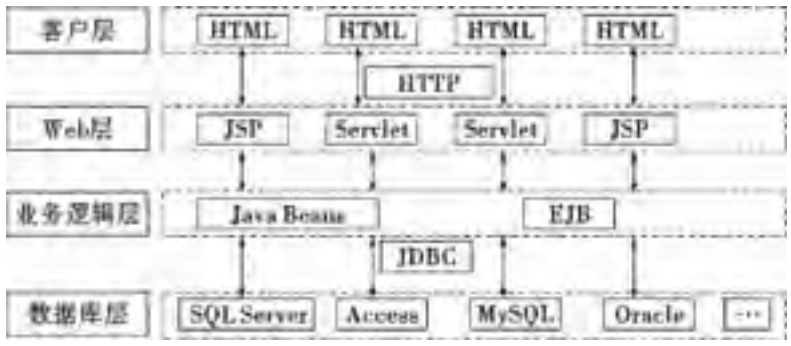

图 1　J2EE 四层体系结构

图 2　基于 J2EE 的五层体系结构设计

业务逻辑层与数据层之间增加一个数据持久层，实现对象与关系数据库之间的映射。利用这个映射框架的机制，对象与关系数据库之间的转换就可以透明地进行，而不用去关心数据库连接、并发性、事务等问题。业务逻辑层直接获取或存储的是清晰的对象，中间的转换过程交给映射框架来处理。

在分层设计中，各层提供的接口是进行层间通信的基础，遵循的原则是实现严格的层间独立、分离，各层的实现细节不

收稿日期：2007-04-28；修回日期：2007-06-29　　基金项目：中国成人教育协会“十一五”成人教育科研规划资助项目(07B045Y)

作者简介：李小平(1963-)，男，北京人，教授，主要研究方向为计算机网络、多媒体技术(lxpmx@x263.net)；肖岳峰(1978-)，男，北京人，硕士研究生，主要研究方向为计算机网络、Web 开发；宿元(1982-)，女，河北张家口人，硕士研究生，主要研究方向为计算机网络、人工智能、负载均衡；宋瀚涛(1944-)，男，北京人，教授，博导，主要研究方向为人工智能；姚永标(1976-)，男，北京人，硕士研究生，主要研究方向为多媒体、计算机网络。

对外公开。分层设计图层间通信说明如下:

a) 客户端与表示逻辑层通过 HTTP 通信,即通过 HttpRequest和 HttpResponse 分别接收用户输入和返回执行结果给用户。

b) 表示逻辑层使用 Struts 框架技术实现,提供了接收/响应客户端请求、控制整个系统工作流程、通过调用 action 与业务逻辑层交互,以及格式化业务数据并动态生成 Web 页面等功能。

c) 业务逻辑层与持久层的交互则是 JavaBean 对 Hibernate 的调用,通过数据访问对象 DAO 进行调用。

d) 持久层与数据库的通信是完全由 Hibernate 负责的,它将实体映射到数据库,对持久对象操作,并将缓存中的结果同步到数据库。

1.2 集成 Struts 框架与 Hibernate 框架实现 MVC 设计模式

Struts 作为基于模型—视图—控制器模式的应用架构,具有组件的模块化、灵活性和重用性的优点。但是 Struts 框架主要是针对表示层设计的,对于后端的业务逻辑层支持不是很强,在进行项目开发中存在以下局限: a) 只能横向分工,按模块来划分工作,软件开发成本相应较高; b) 需要花很多时间在数据层的包装以及不同模块之间进行协调和沟通,导致开发时间的增加; c) 项目移植性相对较差,可能需要为不同数据库编写不同的 SQL 语句; d) 项目扩展性相对较差,适应新的需求或变更时要修改数据库表结构、重新编写 SQL 语句、备份数据库等,对人员要求也相应较高; e) 由于开发人员数据库操作水平参差不齐,开发经验也不尽相同,导致系统性能可能会相对较差。

Hibernate 是一个开放源代码的 O/R mapping(对象/关系映射)框架,它对 JDBC 进行了轻量级的对象封装,以面向对象机制来处理数据库操作。Hibernate 不仅管理 Java 类到数据库表的映射,还提供数据查询和获取数据的方法,大幅度减少开发时人工使用 SQL 和 JDBC 处理数据的时间。

笔者认为应该将 Hibernate 集成到 J2EE 架构中,从 Struts 框架中分离出对象持久性相关部分,交给 Hibernate 来实现。根据 Struts 的体系结构,数据库操作工作是由模型层负责处理的,因此可以将 Struts 中的模型层分成两部分:一部分负责业务逻辑;另一部分使用 Hibernate 实现对象持久性处理。同时分离出具体业务逻辑,新建一个业务逻辑层,专门负责用 Hibernate 来实现业务逻辑和持久性对象的交互。

图 3 显示了集成 Struts 和 Hibernate 的 MVC 模型。Struts 和 Hibernate 框架的整合实现了控制流、业务调用、表示这三者的分离,使系统在开发效率、可维护性、可扩展性方面均有良好的改进。在本模型中,客户层可以看做是 V,它是由浏览器显示给用户的 DHTML 界面;业务逻辑层和持久层则为 M,它是系统的核心部分,由 JavaBeans 和 Hibernate 组成;Web 层对应 C,它控制用户对业务逻辑的操作,同时控制将操作结果显示给用户,由 Servlet 和 JSP 程序组成。其中 PO 是调用模型层产生的持久对象,通过 JSP 页面显示给用户。

1.3 集成 Struts 和 Hibernate 实现 J2EE 分层架构

根据前面的分析,将 Struts 与 Hibernate 框架进行集成,构成了一个新的 Web 应用的开发框架,实现了 J2EE 应用系统的多层架构。该框架一方面继承 Struts 框架在表示层的优点,提供完善的标记库,负责页面请求的接收和转发,实现了表示逻辑和业务逻辑的分离;另一方面在数据持久层等方面发挥 Hibernate 框架的特点,由 Hibernate 框架实现持久层和事务的封装,使业务逻辑与数据库访问分开。这样形成一个层次清楚、更加简洁、功能更加完善的 Web 框架,降低各个层次之间的耦合,提高组件的可重用性,减少在代码编辑方面的复杂性,有利于开发人员将注意力集中在业务逻辑的实现上,有利于系统的可维护性等。

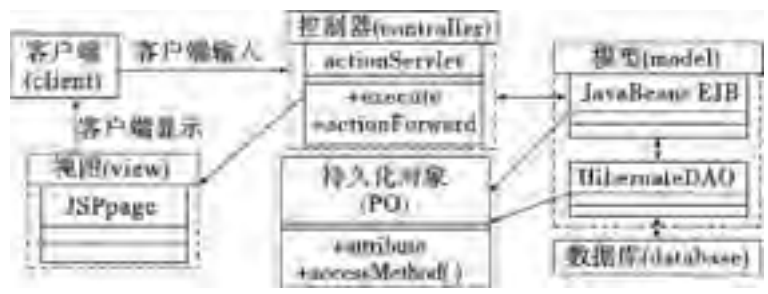

图 3 集成 Struts 和 Hibernate 的 MVC 模型图

集成 Struts 和 Hibernate 实现 J2EE 分层架构如图 4 所示。系统采用的五层结构设计由客户层、Web 层、业务逻辑层、数据持久层、数据库层组成。客户层运行在用户计算机的 Web 浏览器中; Web 层运行在 Web 服务器中,它使用 Struts 框架技术实现,提供了接收/响应客户端请求,控制整个系统工作流程,与业务逻辑层交互以及格式化业务数据并动态生成 Web 页面等功能;业务逻辑层负责实现整个系统的核心业务逻辑,由 JavaBeans 或 EJB 来实现;数据持久层使用 Hibernate 框架技术实现,完成对象和关系的映射,负责对数据库进行操作。

图 4 集成 Struts 和 Hibernate 实现 J2EE 分层架构

下面根据图 4 对系统的各分层功能进行简单分析:

a) 客户层。它是用户用浏览器看到并直接与系统交互的层,主要是由 HTML 语言形成的网页界面。

b) Web 层。它从客户层获得客户的输入,传递给业务逻辑层的组件,再将从业务逻辑层获得的处理结果以 HTML 文件的形式输出到客户端,形成网页界面。Web 层由 Servlet 和 JSP 程序组成,封装在 Web 容器中。业务流程控制一般均由控制器 Servlet 来开发,响应用户的查询等请求并调用业务逻辑层的 JavaBean 来实现复杂的商务逻辑。

c) 业务逻辑层。处理表示逻辑层传递过来的用户响应,并将结果返回给客户层。业务逻辑层封装了系统提供给用户的接口,是直接处理用户请求的中心。这一层主要由 JavaBean 来实现。JavaBean 的主要任务是处理商务逻辑,与客户端交互,返回给它相应的操作结果等。

d) 数据持久层。它完成持久对象到关系数据库的映射,并对持久对象进行操作。业务逻辑模块由 Hibernate 通过 O/R 映射文件实现对具体数据源的操作(即穿过持久层映射到具体的某个数据表),完成对数据库的操作。对于小型的项目来说,这种实现确实是很高效且低成本的。因为采用这种方式,不需要添加含有 EJB 容器的应用服务器,只需要一台 Web 服务器就可以让系统正常运行。

e) 数据库层。对象持久性的具体实现,可以是关系数据库管理系统、文件存储和对象数据库存储管理系统。

2 多源数据整合模型设计

目前的企业级应用系统普遍存在着在整体上缺乏统一性、分散的信息资源导致实现数据共享非常困难等问题。这些针对于不同业务独立开发的数据管理系统虽然能够满足业务数

据存储和管理要求。但在许多情况下为作出一个决策, 可能需要访问分布在网络不同位置上的多个业务数据管理系统中的数据。因此, 整合分散的信息资源就显得非常重要, 通过对原有的多个相对独立的应用系统进行整合, 消除原来存在的 "信息孤岛"现象, 实现数据共享并且消除数据冗余, 实现企业内部信息资源一体化, 为企业综合应用系统提供集成的、统一的、安全的、快捷的信息查询、数据挖掘和决策支持服务。

2.1　多源数据整合策略

Hibernate 使用数据库配置文件来为应用程序提供持久化服务(持久化的对象) 。Hibernate 在 Java 程序与数据库之间进行转换, 通过 Hibernate 访问数据库时, 底层数据库使用的数据类型对 Java 应用程序是透明的。JDBC 驱动程序对底层数据库使用的 SQL 类型进行了封装, 向上提供标准 SQL 类型接口, 使得 Hibernate 可以使用标准 SQL 类型来生成 DML(data manipulation language) 。系统发布时根据不同的数据源定义不同的 Hibernate 配置文件, 就能访问不同的数据库, 实现多源数据的整合。

2.2　多源数据对象映射

1) 定义　根据需要整合的应用系统业务领域及范围确定数据整合的数据源和数据对象。

定义 1　数据源 A 为一个相对完备的数据集合, 由若干数据对象 B 组成, 即 $A=\{B_1, B_2, \cdots, B_m\}$。一个数据源在实践中可以是一个独立的数据库。

定义 2　设 B 为数据源 A 中的一个数据对象, B 由若干数据元素组成, 即 $B=\{b_1, b_2, \cdots, b_n\}$, 数据对象在数据源中可以是数据库中的表和视图。

2) 数据映射规则　从需要整合的数据源到 Hibernate 的持久化数据类的集合之间建立以下映射规则:

规则 1　对于一个数据源 A, 在 Hibernate 中创建一个与之对应的会话配置文件, 在会话配置文件中指明该数据源对应的数据库, 如 Hibernate. Acfg. xml。

规则 2　对于数据源 A 中每个数据对象 B, 创建一个与之相对应的数据持久化类 C, 并且为每一个数据持久化类 C 与数据对象 B 配置一个相应的映射文件 D。其中 B 的每一个数据元素对应 C 的一个类属性。

2.3　多源数据访问接口设计

在数据库操作中, 有相当一部分是数据的新增、修改、删除和查询, 这些功能分别在业务逻辑层和 Web 两层中实现。在业务逻辑层, 通过 JavaBean 调用 Hibernate 实现数据库数据的增、删、改、查这些数据库的基本操作; 在 Web 层, 需要通过相应界面来启动后台 JavaBean 的数据库操作功能。数据的新增、修改、删除和查询是一种机械固定的工作, 大部分是琐碎的细节, 因而在代码稳定性方面存在极大的隐患。因此在持久层与业务逻辑层之间设计多源数据访问组件, 在多源数据访问组件中抽象和封装所有对数据存储的数据访问实现。多源数据访问组件实现了使用特有数据存储所要求的 API 和必需的逻辑。其中实现的细节与组件的代码相隔离, 但却可以通过简单的外部接口来访问。业务组件能够使用这些简单的外部接口访问底层数据存储中的持久数据, 实现数据库数据的增、删、改、查这些数据库的基本操作。开发人员只是根据具体不同的数据对象作一些简单的参数设置, 大大降低编程工作量, 提高开发速度和质量, 开发出来的代码也将具有较强的稳定性和可维护性。

本模型构建了一个为多源数据提供数据访问操作的 DbOperate类。它封装了各种数据库操作的方法, 当业务类需要访问数据库时, 直接调用 DbOperate 类。DbOperate 实现的方法有: 开始结束事务、开始结束会话以及各种创建、读取、更新、删除操作。其中包括单个操作和批量操作。实现数据访问操作的各种方法的参数包括两种, 即对象 Object 和 HQL 语句。HQL 是 Hibernate 的查询语言, 它与 SQL 非常相似, 但是 HQL 是面向对象的。在 HQL 语句中, 要查询的数据均是以对象的形式存在的。DbOperate 类实现的主要功能如下:

```
public List Query( String condition)  //获取指定查询条件的数据表
public List getDetail( String table, int id)
//获取指定表、指定 id 的记录信息
public List getOnePageLists( String table, String con, String byorder, int dispnum, int startrec)
//获取指定表、查询条件、排序、记录数、起始记录的数据表
public int getRecordCount( String table, String con)
//获取指定表、指定查询条件的记录数
public void save( Object obj)  //插入实体对象所对应的记录
public void update( Object obj)  //修改实体对象所对应的记录
public void delete( Object obj)  //删除实体对象所对应的记录
public void saveOrUpdate( Object obj)
// 插入或修改实体对象所对应的记录
public List Query( String condition)  throws HibernateException {
Session session = HibernateUtil. currentSession( ) ; //获取 Session;
Transaction tx = session. beginTransaction( ) ; //事务开始
List list = null;
try {
   Query query = session. createQuery( condition) ;
   list = query. list( ) ;  //数据查询
      tx. commit( ) ; //事务提交
        } catch ( HibernateException e) {
          if ( tx ! = null)
            tx. rollback( ) ;
          throw e; }
HibernateUtil. closeSession( ) ; //关闭 Session
return list;
} ( 其他操作实现代码略)
```

3　结束语

本文在传统的 J2EE 框架中引入面向对象机制的数据持久层, 并在此基础上提出多源数据整合模型, 使业务逻辑处理层有效地分离和隐藏了数据读取和操纵中的所有数据访问代码细节, 将客户应用程序和底层存储机制隔离开, 完全抽象出开发应用程序时使用的数据物理细节, 构建了更为有效的松耦合多层次 Web 开发模型。

参考文献:

[1] 方巍, 孙涌, 张书奎. 整合 Struts 和 Hibernate 的 Web 系统应用[J]. 计算机与现代化, 2005(12) : 39-41.

[2] KING G. Hibernate reference3[EB/OL] . (2005-01-01) . http: //www. hibernate. org/hib_docs/v3/reference/en. html/architecture. html#architecture overview.

[3] HEUDECKER N. Read introduction to Hibernate[EB/OL] . (2003-07-15) . http: //www. theserverside. com/resources/article. jsp? 1 = Hibernate.

[4] KLEIN N. Data persistence frameworks: an introduction to Hibernate[EB/OL] . (2003-09) . http: //www. baychi. org/calendar/files/nibernate-talk/hibernate-talk. pdf.

[5] 宋秀琴, 侯殿昆, 方中纯. 基于 Struts 和 Hibernate 的 Web 应用的构建[J]. 网络与通信, 2005, 21(3) : 125-127.

[6] 朱庆红, 吴宇红. 一种对象/关系映射框架的分析和应用[J]. 电子科技, 2004(1) : 54-60.

[7] 高昂, 卫文学. 基于 Hibernate 与 Struts 框架的数据持久化应用研究[J]. 计算机应用, 2005, 25(12) : 31-35.

[8] 谢艳平, 胡家宝, 谢承旺. 基于 Struts 和 Hibernate 的 MVC 设计模式[J]. 交通与计算机, 2005, 23(4) : 62-64.