

文章编号:1001-9081(2005)08-1884-03

基于 Struts 与 Hibernate 集成架构的项目管理系统

冯国仕,李志蜀

(四川大学 计算机学院, 四川 成都 610065)

(fengguoshi@ boco. com. cn)

摘 要:Struts 架构与 Hibernate 架构有助于快速构建基于 Web 的信息系统,但是它们都有各自的优缺点。利用 DAO 模式将这两者有机集成起来,使得它们优势互补,形成一个统一的架构。详细论述了集成这两个架构的原理和方法,阐述了一个基于 Struts 架构和 Hibernate 架构实现的项目管理信息系统,以说明利用这个集成的架构来开发一个功能强大而又灵活的信息系统的高效方法。

关键词: Struts 架构; Hiberante 架构; 集成; 项目管理系统

中图分类号: TP311.52 文献标识码: A

Project management system based on the integrated architecture of Struts and Hibernate

FENG Guo-shi, LI Zhi-shu

(College of Computer Science, Sichuan University, Chengdu Sichuan 610065, China)

Abstract: Both Struts architecture and Hibernate architecture contribute to building a Web information system, but both of them have their respective benefits and drawbacks. Integrating the two architecture with the DAO pattern not only avoid their disadvantages while retain their advantages, but also make them more powerfull. The mechanism and the way to integrate the two architecture was discussed, and an efficient way was brought forth to develop a powerful and flexible information system by expatiating the developing of a practical project management information system based on the integrated architecture.

Key words: Struts architecture; Hibernate architecture; integration; project management system

随着计算机信息技术的迅猛发展和 Web 信息系统的广泛应用,一些传统的开发方法很难胜任迅速开发一个灵活而又功能强大的 Web 信息系统。为此人们不断在实际中总结、并在理论上证明了很多成功的模式,包括 MVC 与 O/R 映射等,有很多技术都实现了这些模式,其中包括 Struts 与 Hibernate 两种架构。但是 Struts 架构只解决了视图层、业务层和控制层的分离,并没有对复杂的持续层提供灵活的架构支持,反之,Hibernate 架构则提供了灵活的持续层支持,对应用的整体架构却没有任何帮助,因此通过将这两个架构整合起来,可以得到一个迅速地开发灵活、低耦合及易于维护的信息系统的完整解决方案。

1 MVC 设计模式与 Struts 架构

传统的 Web 应用基本都是将页面显示、商业逻辑和数据处理等大部分功能集中在页面中,此外,页面代码通常还包含用来控制应用程序流程的逻辑,这样导致商业逻辑、业务流程和页面显示的耦合性高、可维护性低。为了降低这种耦合性,改善系统的质量,提出了 MVC 设计模式,它强制性地使应用程序的输入、处理和输出分开。MVC 应用程序被分成三个核心部件:模型、视图、控制器,它们各自处理自己的任务。其中视图是用户看到并与之交互的界面,模型表示企业数据和业务规则,控制器接受用户的输入并调用模型和视图去完成用户的需求。这样 MVC 模式消除了信息展示、业务逻辑与业务流控制之间的耦合性,增强了系统的灵活性,从而也增强了系

统的可维护性。其原理图如图 1 所示。

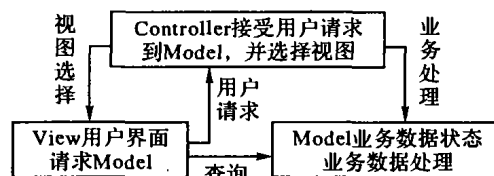


图 1 MVC 设计模式

现在,Java 广泛地用于开发各种信息管理系统,因此怎么用 Java 来实现 MVC 设计模式,便成了广大开发人员的迫切需求,而一个开放源代码的 Apache 项目——Jakarta Struts Framework 正是满足这种需求的,它通过利用 JSP,Servlet 和标签技术实现了这个模式,它的原理图如图 2 所示。

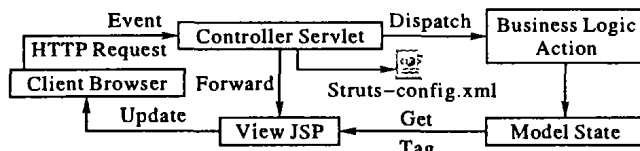


图2 Struts 架构

由图2可以看出,在 Struts 架构中,控制器是以 Servlet 实现的一个命令设计模式,并利用 struts-config.xml 文件配置控制器,而模型部分仅仅用 Action 类做简要的包装,视图部分用 JSP 来实现,不过 Struts 架构提供了大量的标签来帮助开发人员构建视图,这些标签可以很容易地与控制器交互并从模型中获取数据。

收稿日期:2005-03-11;修订日期:2005-05-24

作者简介:冯国仕(1981-),男(苗族),重庆人,硕士研究生,主要研究方向:计算机网络与信息系统; 李志蜀(1946-),男,重庆人,博士生导师,主要研究方向:计算机网络与信息系统、智能控制。

2 O/R 映射与 Hibernate 架构

在信息系统的开发过程中,有很大精力要花费在业务逻辑的开发上面。由于绝大多数业务模型都涉及到关系数据库,在采用 Java 作为信息系统的开发语言时,传统 Web 应用开发方法是直接用 JDBC 与数据库交互。但是这个工作量很大,而且一旦业务逻辑稍微有一些变更,就要大量地更改这些 JDBC 中的 SQL 语句,因此不管是开发还是维护系统都很不方便。考虑到 Java 的面向对象性和关系型数据库的关系型结构相差甚远,因此很有必要引入一种在对象与关系型数据库之间的直接映射机制,这种映射应该是最大量地使用配置文档,以便今后业务逻辑更改后是尽可能地修改映射文件而不是 Java 源代码,因此出现了 O/R 映射模式。有很多开源项目都使用 Java 实现了这个 O/R 映射,而 Hibernate 是其中最优秀的实现架构之一,它的结构如图 3 所示。

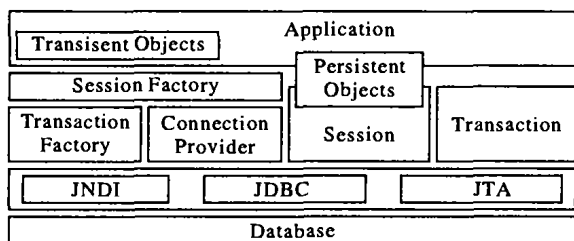


图3 Hibernate 架构

其中 Session 代表应用程序和持续化层之间的一次对话,封装了一个 JDBC 连接,由 SessionFactory 创建,也是 Transaction 的工厂。Transaction 代表一个事务对象,它是 JDBC 事务、JTA 或者 CORBA 事务的抽象。Persistent Object 包含了持续化状态和商业功能,它们可能是普通的 JavaBeans,唯一特别的是它们现在从属于且仅从属于一个 Session。TransientObject 是目前还没有从属于一个 Session 的持续化类的实例。

现在只需要写出持续化类,并给出映射的配置文档,然后持续化类与数据库之间的转换便由 Hibernate 来做了,而且还可以用 Hibernate 的 HQL(一个对象查询语言)来通过对象的关系查询与操纵对象。由于 Hibernate 实现了 O/R 映射,现在模型部分使用持续化对象类而不再是冗长的 JDBC 语句,从而使得模型部分得到极大的简化。

3 Struts 与 Hibernate 的整合

由前面的介绍可以看到,Struts 架构实现了 MVC 设计模式,通过利用 Struts 控制器部分的配置文档,业务流程可以不再硬编码到源程序中去,这使得维护性和灵活性大大加强,因此 Struts 的核心是控制器部分。虽然 Struts 在视图方面没有任何要求,但是它提供了很多的标签类供开发人员使用。但是 Struts 在模型部分,它仅仅提供一个 Action 类,让这个类来“瘦包装”所有的后台业务逻辑,因此 Struts 对模型部分的支持还有所欠缺。

相反就 Hibernate 来说,它完全只提供模型部分支持,如果仅仅采用 Hibernate 架构开发应用系统,系统只会业务模型部分有所改善,但是系统的逻辑处理、流程控制与视图并没有得到有效分离,因此系统的耦合性还是太高,不易于维护。

综合分析了 Struts 与 Hibernate 各自的优点与不足,笔者认为可以将这两个架构有效整合在一起,让 Struts 负责降低系统总架构的耦合性,而让 Hibernate 负责降低业务模型部分

的开发难度。结合之后得到的好处是采用这个集成架构开发出来的信息系统无论是在整体架构上还是在局部的复杂业务模型中都有了更低的耦合性,它们的灵活性与可维护性也得到了提高,从而消除了单独使用这些架构开发系统的不足。

集成 Struts 架构与 Hibernate 架构的方法是利用 Struts 架构作为信息系统的整体基础架构,它负责了 MVC 的分离,而在 Struts 架构的模型部分,利用 Hibernate 架构来提供持续层支持。具体做法是首先分析系统的需求,并利用面向对象的分析方法提出一个完整的领域模型(最好用 UML 表示出来),将这些领域模型采用 Java 语言实现出基本的 Java 对象(POJO)。然后写出基本的 DAO 接口,并给出 Hibernate 的 DAO 实现(同时给出 POJO 与数据库之间的映射文档),在业务类 BO 中,调用采用 Hibernate 架构实现的 DAO 类来实现 Java 类与数据库之间的转换和访问,最后在 Struts 架构的控制器部分的 Action 中调用 BO 来完成业务逻辑。这样,就利用 DAO 模式来实现了 Struts 架构与 Hibernate 架构的无缝集成。集成后的架构如图 4 所示。

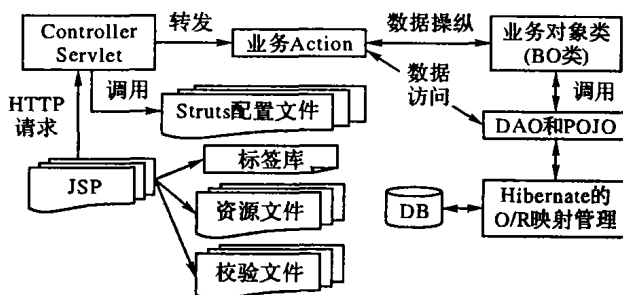


图4 集成 Struts 与 Hibernate 的系统架构

在这个集成架构中,所有的业务流控制都交给 Struts 的配置文档来完成,并由控制器 Servlet 来分开模型与视图部分,从而降低了系统的耦合性,同时增强了系统的可维护性,也正因为通过配置来控制业务流程,所以系统具有很好的灵活性。模型部分的开发借助于 Hibernate 来做 O/R 映射,而不再是硬编码 JDBC 语句,因此维护也比传统应用开发要简单一些。而且由于采用 DAO 模式来集成这两个架构,使得今后即使不采用 Hibernate 架构而是别的技术来实现持续化层也不会给系统带来大的影响,因为只需要替换 DAO 实现即可,这样进一步提高了系统的灵活性和可维护性。当然,正因为采用了 Hibernate 架构作为持续层支持,DAO 的实现已经非常简单,因此这对于系统的可维护性提供了进一步的支持。集成后的系统架构身兼 Struts 架构和 Hibernate 架构的所有长处,并让它们相互补充,克服了各自的不足。

4 基于 Struts 与 Hibernate 实现项目管理系统

4.1 系统简介

随着人们对移动通信的通讯服务要求越来越高,全国各个移动运营商都在迅速建设新的或者改造原有的移动通信服务基础设施,以更快更好地提供高要求高质量的通信服务。这些项目所涉及的人力、物力与财力都十分巨大,因此需要一个好的项目管理系统,这个系统可以帮助管理人员、工作人员以及建设中所涉及的所有外部单位进行及时沟通,以便能够协作解决这些大型项目中涉及到的各种难题。

4.2 系统的总体架构

由于这个项目管理系统涉及到项目经费预算、计划立项、项目实施与项目投资统计与跟踪,因此系统从整体上被分为四个相对对立的模块:投资计划编制、投资计划下达、项目管

理功能和投资统计与跟踪。这四个模块相对独立,但都很复杂,因此分别交给四个不同的工作小组协同开发,并利用 Struts1.1 来对模块进行划分和配置。从业务上来说,这个系统涉及到大量的数据处理与复杂的业务流程,这需要有好的业务模型来简化开发,因此集成的 Struts 与 Hibernate 架构刚适合于开发这个项目管理信息系统,系统总体架构如图 4 所示。

4.3 采用 Struts 与 Hibernate 架构实现项目管理系统

采用 Struts1.1,系统中的每个模块都有一个自己的配置文档,以控制该模块的流程。每个模块除了业务逻辑不一样之外,采用 Struts 和 Hibernate 架构来实现模块的技术与过程完全一样,因此这部分主要阐述这个集成的架构在投资统计部分的实现。

4.3.1 投资统计模块功能简介

投资统计模块用于从整体上统计和跟踪一个省在移动通信项目工程实施中的项目能力和投资金额,并实时地随着项目进展,让工作人员将项目进展信息反馈给系统。项目能力部分包括获取每个项目的能力,就地区的所有项目或者全省的项目按照不同的能力类别进行统计或者对所有的能力进行统计;投资管理是指获取各个项目的实际投资进度信息,并统计各个地区和全省的投资信息,还包括统计不同类别方面的投资进度信息。

4.3.2 利用 Struts 控制投资统计业务流

Struts 的主要功能是实现 MVC,它的核心是控制器部分的 ActionServlet,而 ActionServlet 通过一个 xml 配置文件来配置业务流程,因此这个模块用了一个 struts-config-statistics.xml 来配置业务流程,同时在 web.xml 中申明这个配置项。

在 struts-config-statistics.xml 中,声明每个业务所需要的客户端请求、处理该请求的 Action 类,收集该次请求数据信息的 FormBean 类,并说明业务执行完毕后各种结果应该给用户返回的视图。例如,对于填报项目投资进度分摊表来说,有如下一个映射:

```
<action path = "statistics/addInvestment"
  type = "com. boco. action. statistics. AddInvestmentAction"
  name = "investmentForm"
  scope = "request"
  validate = "true"
  input = "statistics/addInvestmentReq. jsp" >
  <forward name = "success" path = "statistics/getInvestment. do"/>
  <forward name = "failure" path = "/errors. jsp"/>
</action>
```

ActionServlet 将会按照这个配置,在收到客户端填报投资进度分摊表的请求 (statistics/addInvestment) 之后,会将用户提交过来的数据填充到模型类 InvestmentForm 中,然后将这个模型类信息传递给 AddInvestmentAction,由后者来做业务处理,即将一个新的投资分摊表信息添加到系统中去。

其中 InvestmentForm 类只负责收集数据信息,它在 struts-config-statistics.xml 中声明如下:

```
<form-beans>
<form-bean type = " com. boco. form. statistics. InvestmentForm"
  name = "investmentForm"/>
</form-beans>
```

InvestmentForm 继承自 ValidatorForm 类,实现了它的 validate() 方法,检查用户的输入是否满足项目进度要求,如果不符合,将会将错误添加到 ActionErrors 中,然后错误会返回到输入页面。否则这个对象将被作为参数传递给

AddInvestmentAction 的 execute 方法。

AddInvestmentAction 类是 Action 的子类,负责所有的业务逻辑,它的实现如下:

```
public class AddInvestmentAction extends Action
{
  public ActionForward execute( ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
  {
    ActionForward myforward = null;
    try
    {
      InvestmentMag investMag = new InvestmentMag();
      Investment investment =
        PjBeanUtil. convert( ( InvestmentForm) form);
      investMag. addInvestment( investment);
      return mapping. findForward( "success");
    }
    catch ( Exception ex)
    {
      ActionErrors. add( new ActionError
        ( ex. getMessage()));
      return mapping. findForward( "failure");
    }
  }
}
```

4.3.3 利用 Hibernate 实现投资统计的模型部分

在 AddInvestmentAction 的 execute 方法中,它只是“瘦包装”业务逻辑,真正的业务是在 InvestmentMag 中完成的,通过调用 InvestmentMag 的 addInvestment 方法将 Investment 对象加入到系统中。这儿出现的 Investment 就是 Hibernate 里面的 POJO 类,它的属性通过 Hibernate 的配置映射文档 investment.xml 映射到关系数据库 Informix 中。Hibernate 的映射文件说明了一个持续化类与关系数据库表的映射关系,包括属性映射到哪个字段,以及集合属性映射到哪些相关字段。还要通过映射文件来说明持续化类之间的关系,例如父子关系,一对多和多对一等,而这些关系最终体现着关系型数据库表之间的关联关系。

例如 Investment 的部分属性映射如下:

```
<hibernate-mapping>
<class name = "com. boco. dao. statistics. Investement" table = "
  INVESTEMENT" >
  <id name = "id" type = "integer" unsaved-value = "null" >
    <column name = " ID" not-null = "true"/>
    <generator class = "increment"/>
  </id>
  <property name = "power"/>
  ...
</class>
</hibernate-mapping>
```

类 InvestmentMag 是系统中真正负责所有的操纵投资分摊表的业务逻辑类,它完成相应的业务请求操作。这个类调用利用 Hibernate 实现的 DAO 来实现相应的业务处理,例如增加一个投资分摊表的方法如下所示:

```
public void addInvestment( Investment investment) throws Exception
{
  ...
  InvestmentDAO investmentDao = new InvestmentDaoImpl();
  investmentDao. newInvestment( investment);
  ...
}
```

这里面的 InvestmentDAO 接口给出新增、修改、删除和访问投资分摊表的访问接口,而 InvestmentDaoImpl 利用

(3)注册信息处理页 `regist_do.jsp` 创建 `UserBean` 实例。

```
UserBean user = new UserBean();
```

调用 `UserBean` 的 `getPhone` 方法获取用户的手机号。

```
String phone = user.getPhone();
```

`getPhone` 方法中手机号的取得:

```
String phone = request.getHeader("x-up-calling-line-id");
```

若无法取到公开的手机号码,可以取加密的手机号码 `deviceid` (`deviceid` 为 256 位的加密手机号):

```
String phone = request.getHeader("deviceid");
```

调用 `UserBean` 的 `isRegist` 方法判断该用户是否已注册。

```
boolean isReg = user.isRegist(phone);
```

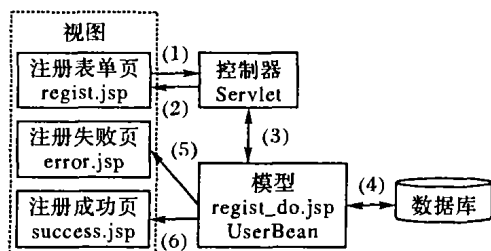


图4 注册系统的MVC模型

(4)在获取用户信息和新建用户信息的过程中通过连接池与数据库相连,并读取和插入信息。

(5)如果 `isReg` 为 `true` 则表示该用户之前已注册,跳转到注册失败页面 `error.jsp` 并提示其进入登录页面凭密码进主页面。

(6)如果 `isReg` 为 `false` 则表示该用户还未注册,调用 `UserBean` 的 `NewUser` 方法注册该用户,在数据库内新建一条

用户数据,存入该用户的姓名、密码、手机号,并返回其 `userid`。

```
int userid = user.NewUser(name, pwd, phone);
```

跳转到注册成功页面 `success.jsp`,并传递 `userid` 参数。提示用户进入主页面。

5 结语

从上述简单应用实例可以看到,基于MVC模式的WAP应用程序结构清晰、便于管理、便于模块重用,也为系统的维护与改进提供了方便。近年来WAP技术的发展我们有目共睹,而MVC模式的应用为WAP开发注入了新的生命力。

参考文献:

- [1] (美) Wireless Application Protocol Forum Ltd. WAP 无线应用协议 [M]. 侯春萍, 宋梅, 蔡涛, 等译. 北京: 机械工业出版社, 2000.
- [2] 王宏锦, 杨明极, 纪合宝. WAP 增值业务网站建设研究[J]. 哈尔滨理工大学学报, 2004, (9) 6: 128.
- [3] 张为, 李绍滋. 基于 MVC 模式的 Web 应用研究[J]. 长沙电力学院学报, 2004, (5): 33.
- [4] 刘学英. 基于 J2EE 的 WAP 增值应用软件的设计与实现[J]. 计算机工程与应用, 2002, 23: 144.
- [5] 郭梅, 江红. Struts 在实现 MVC 架构中的应用[J]. 计算机与现代化, 2004, (1): 106.
- [6] BATNI RP, LEE CC, VARNEY DW. Enhanced services in WAP-enabled networks[J]. Bell Labs Technical Journal, 2000, 5(3): 145 - 152.

(上接第 1886 页)

Hibernate 架构实现了 `InvestmentDAO` 给出的方法,例如 `InvestmentDaoImpl` 的 `newInvestment` 方法实现如下:

```
public void newInvestment(Investment investment) throws Exception
{
    Session session = HibernateUtil.currentSession();
    Transaction tx = session.beginTransaction();
    session.save(investment);
    tx.commit();
    HibernateUtil.closeSession();
}
```

可见在采用了 Hibernate 之后,访问数据库操作的 SQL 语句会大量减少,像这种插入数据的操作甚至可以不用 SQL 语句。这样使系统不仅开发起来更快,也使系统更容易更新和维护,例如假设 `Investment` 类多了几个属性或者少了几个属性我们都可以不用修改业务逻辑,只需要修改映射文件即可。

4.4 采用多架构实现的系统与传统开发系统的对比

按照以前的开发模式(例如 Model1)来开发这个系统,所有的视图和业务逻辑都会混杂在一起,很难维护,例如可能用大量的 JSP 来写页面,里面嵌套很多的 Java 业务代码,也可能用很多的 `JavaBean` 来做业务,但是这些业务和页面是紧紧连接在一起的,往往修改视图或者业务逻辑都会造成另外一个部分的更改。在采用了 Struts 之后,视图、模型和控制器三个部分得到了彻底的分离,特别是控制器部分,通过采用了配置文件来配置业务流程,使得修改和维护业务流程的工作减轻。

在采用 O/R 技术以前,如果直接用 JDBC 来写更新和管理信息系统的代码,会有很大的工作量,而且可移植性也不是太好。在采用 Hibernate 之后,需要的 SQL 语句就很少了,所以无论是开发还是维护都轻松了很多。

最后,由于利用 DAO 模式来把 Struts 和 Hibernate 架构无缝集成起来,让它们优势互补,使得系统的业务流程控制与业务逻辑开发都得到了最好的解决方案,使得系统的开发工作量减小,系统的可维护性、灵活性和健壮性都得到了提高。

5 结语

本文详细解释了 MVC 模式,论述了 Struts 架构的运行原理,并详细探讨了 O/R 映射工作机制,阐述了 Hibernate 的实现方式,归纳了它们各自的优缺点,最后将这两个架构有机集成起来。然后将它们实际应用于开发移动项目管理信息系统,说明了用这个集成起来的架构开发应用系统的方法,以简化应用系统的开发,并提高应用系统的健壮性、可维护性和灵活性。

参考文献:

- [1] 阎宏. Java 与模式[M]. 北京: 电子工业出版社, 2002.
- [2] (美) GAMMA E, HELM R, VUSSIDES RJ. 设计模式: 可复用面向对象软件的基础[M]. 李英军, 等译. 北京: 机械工业出版社, 2000.
- [3] (美) GRAND M. Java 企业设计模式[M]. 张威, 卢庆龄, 等译. 北京: 电子工业出版社, 2003.
- [4] 孙卫琴. 精通 Struts: 基于 MVC 的 Java Web 设计与开发[M]. 北京: 电子工业出版社, 2004.
- [5] 赵晨希. 用 Struts 建立 MVC 应用的介绍[EB/OL]. <http://www-900.ibm.com/developerWorks/cn/java/1-struts-mvc/index.shtml>, 2003 - 12.
- [6] Apache Software. Struts Project[EB/OL]. <http://jakata.apache.org/struts/>, 2005.
- [7] Hibernate org. Hibernate Project[EB/OL]. <http://www.hibernate.org>, 2005.