第 35 卷 第 6 期 计 算 机 工 程 2009 年 3 月
Vol.35 No.6 Computer Engineering March 2009
·软件技术与数据库· 文章编号：1000—3428(2009)06—0041—03 文献标识码：A 中图分类号：TP393.02

基于 JavaEE 的电信 CRM 数据持久层的实现

张 俐
(江苏技术师范学院计算机科学与工程学院，常州 213001)

摘 要：结合中国电信行业信息化建设的实际需求，引入 JavaEE 架构体系，提出一种在 JavaEE 平台上开发大型电信客户关系管理数据持久层的方法。通过门面模式、VO 模式和 DAO 模式在 Spring JDBC 架构中的应用，实现持久层之间的松散耦合，使整个系统具备良好的稳定性、可伸缩性和可扩展性。

关键词：电信客户关系管理系统；Spring 框架；设计模式

Implementation of Telecom CRM Data Persistent Layer Based on JavaEE

ZHANG Li
(College of Computer Science & Engineering, Jiangsu Teachers College of Technology, Changzhou 213001)

【Abstract】 Combined with the practical requirement in the informatization process of the China Telecom Commission, JavaEE architecture is introduced, and a method for developing large scale telecom Custom Relationship Management(CRM) data persistent layer on the platform of JavaEE. This paper also analyzes the application of façade pattern, VO pattern and DAO pattern in the Spring JDBC framework, which implements the loose coupling in persistent layer. The whole system has better performance of stability, flexibility and extension.

【Key words】 telecom CRM system; Spring framework; design mode

1 概述

客户关系管理(Customer Relationship Management, CRM)是选择和管理有价值客户及其关系的一种商业策略，它要求以客户为中心的商业哲学和企业文化共同支持有效的市场营销与服务流程。

本文对电信客户关系管理系统中由数据持久化所引发的问题进行研究，提出一种利用设计模式和 Spring JDBC 技术改进 JavaEE 数据持久层的实现方案[1]，该方案解决以下 3 个方面问题：(1)业务层和数据持久层的逻辑分离，便于数据持久化操作的重用，有利于数据持久化操作的重用和改动，并进一步减少重复代码，同时有利于代码的长期维护；(2)减少网络负载，使数据库底层结构信息不暴露，实现可移植性；(3)节省内存和提高运行效率，同时可以方便的运用 JDBC 中 Batch 语句，调整 PreapredStatement 的 Batch Size 和 Fetch Size 等参数以及在必要的情况下采用结果集 cache 等，且可以自由使用数据库 view，调用 store procedure(存储过程)。

2 电信客户关系管理系统功能概述

2.1 营销管理功能

营销管理功能主要有以下几个方面：

(1)营销自动化：它与 SFA 关联紧密并互为补充，为公司的市场营销活动提供功能；

(2)营销渠道管理：产品和价格配置，通过预先定制的信息支持帮助营销活动的计划编制和执行、计划结果的分析；把营销活动与业务、客户、联系人等建立关联并管理任务完成进度；

(3)提供公告板功能，为市场营销活动中锁订潜在客户，并反馈给 SFA；

2.2 销售管理功能

销售管理功能如下：

(1)客户管理：包括客户信息的搜集，记录与此客户相关的基本活动和活动历史，定单的录入和跟踪，客户的分类与定级；优质客户与黑名单客户的识别和管理，建议书和销售合同的生成等；

(2)联系人管理：包括联系人概况的记录、存储和检索；跟踪与时间、任务、类型、简单的描述等；

(3)潜在客户管理：包括业务线索记录、升级和分配；销售机会的升级和分配；潜在客户的寻找和跟踪；

(4)销售管理：组织和浏览销售信息，如客户、业务描述、联系人、时间、销售阶段、业务规模、业务额；产生各销售业务的阶段报考，并给出业务进展的预测；销售费用管理等；

(5)个性化日程安排：包括日历、约会设计、活动安排，团队的人员安排，预告提示等；

(6)佣金管理：允许销售经理创建和管理销售队伍及奖励和佣金计划。

2.3 客户服务和支持

客户服务和支持涉及以下几个方面：

(1)呼叫中心：为客户提供每周 168 h 不间断服务，并将

基金项目：江苏省高校自然科学基础研究基金资助项目(08KJD 520005)

作者简介：张 俐(1977－)，男，讲师、硕士，主研方向：软件工程，基于 JavaEE 的系统开发

收稿日期：2008-09-10 **E-mail：**zhangli_3913@yahoo.com.cn

客户的各种信息存入共享的数据库以及时满足客户需求；

(2)技术人员对客户的使用情况进行跟踪，为客户提供个性化服务，并且对服务合同进行管理。

2.4 电子商务功能

包括：电子商店，电子营销，电子货币与支付以及电子支持。

2.5 商业智能功能

运用数据挖掘技术从数据仓库中分析和提取相关规律、趋势等，让客户信息和知识在整个企业内得到有效的流转和共享，并进一步转化为企业的战略规划、科学决策和各业务流程的辅助支持。

电信 CRM 系统的总体功能结构如图 1 所示。

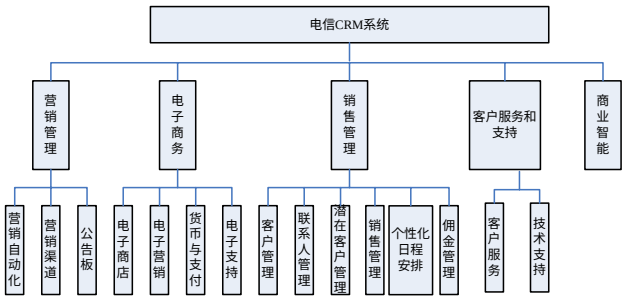

图 1 系统功能结构

3 数据持久化层模型的设计原理与实现

3.1 设计模式

模式提供一个通用的设计，它从一个更高的层次来描述方案的核心结构，以便对许多特定的场景都能够通用。本文主要使用 Facade 模式、VO 模式和 DAO 模式。

3.2 Spring 框架的优点

Spring 框架中的 AOP 将持久化、事务等分解成各个切面(关注点)，从而能够模块化管理这种横切多个对象的关注点，且 Spring 中的 IoC/DI 能极大改善代码的可重用性，在相当程度上降低组件之间的耦合，实现组件真正意义上的即插即用。

3.3 数据持久化层架构模型的设计[2]

利用 Facade 模式、VO 模式、DAO 模式以及 Spring JDBC 框架技术，形成新的 JavaEE 的数据持久层应用架构，如图 2 所示。

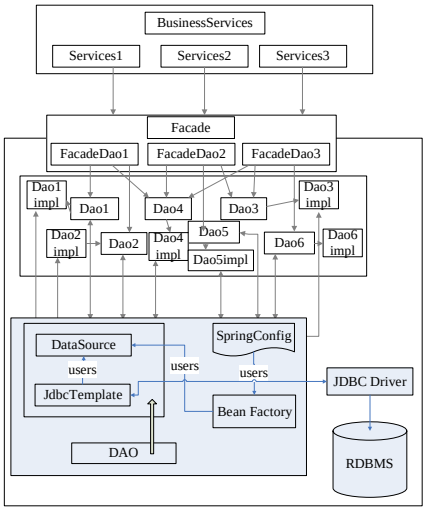

图 2 JavaEE 的数据持久层架构模型

从图 2 可以看出：(1)Facade 模式实现了数据持久层与业务逻辑层之间的接口的统一。现在在 BusinessService 中的 Service 仅需访问 Façade，由其访问其他相应的 FacadeDao 即可。因为，该 Facade 与 Dao 在同一层中，在 Facade 执行操作过程中不用与 Service 通信，仅在执行结束后才会向其返回结果，所以，Service 只需一次远程调用的开销就可。这不仅减少了网络传输的次数，提高系统的性能，而且较好地分隔了 BusinessService 层与数据持久层。(2)解耦各个子系统：在系统分析过程中，一个复杂的系统往往可以分解为若干个逻辑上比较独立的抽象概念子系统。这种抽象概念子系统可以作为数据持久化外观层对象设计的起点，即可以把逻辑上比较相关的数据持久化操作组合成为一个 Facade 对象。这些 Facade 对象封装了纷繁复杂的数据持久化对象调用接口，提供给 BusinessService 层一个清晰的接口集。(3)进行安全和事务处理：Facade 对象不包含数据持久逻辑，但是在这个层次却非常适合进行应用程序安全验证和事务处理。在此处进行用户访问权限控制能有效地隔离非法用户对数据持久规则对象的访问；作为一个系统领域逻辑的高层视图，在此进行事务处理能够以高效的方式协调各个操作之间的事务关系。

4 电信 CRM 系统数据持久层的实现

4.1 数据持久化层实现的步骤

数据持久化层实现的步骤为：(1)从 Facade 接受请求；(2)根据数据持久化规则处理请求；(3)使用 DAO 模式完成与数据库的交互任务；(4)将处理结果传递回 Façade。

4.2 Spring JDBC 在电信 CRM 系统开发中的应用[3-4]

现以电信 CRM 系统中的客户管理子系统为实例，描述基于图 2 框架的系统实现，而客户管理子系统的用例如图 3 所示。

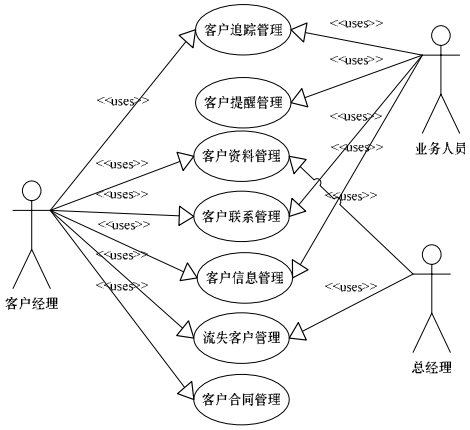

图 3 客户管理用例

在客户管理子系统中所用到的主要数据表有：t_user compact(客户合同管理表)，t_usertrace(客户追踪表)，t_user remind(客户提醒表)，t_userrelation(客户联系表)，t_userinfo (客户信息表)，t_userloss(客户流失表)以及 t_userdata(客户资料表)。

4.2.1 使用 Spring JDBC 框架进行数据持久化的实现

数据持久化基于 Spring JDBC 框架，需要创建 ValueObjects(值对象 VO)、Data Access Objects(数据访问对象 DAO)。其中，每个 VO 对应于数据库的一张表，DAO 用于持久化 VO。

客户管理子系统主要用到以下 7 个 VO：客户资料管理，客户联系管理，客户跟进管理，客户提醒管理，流失客户管理，客户合同管理，客户信息管理。这些 VO 封装了持久层对象的数据，在业务层和持久层之间进行流动，以传递所需

要的数据。每个 VO 就是个普通的 JavaBean，由一些属性及对应的 getter/setter 方法组成。DAO 用于持久化 VO，与 VO 相对应的 DAO 有 UserRelationDAO, UserInfoDAO, User RemindDAO, UserDataDAO 等 7 个类，每个 DAO 类均实现了接口 DataAccess0bject(接口 DataAccess0biect 定义了数据库操作的方法，如 insert(UserRelation vo), update(UserRelation vo), delete(UserRelation vo)等)。为了让 Spring JDBC 知道如何把一个对象存储到数据库中，必须创建 Spring JDBC 数据源配置，即在 ApplicationContext.xml 中配置数据源的 bean，其关键代码如下：

```
<bean id="dataSource"
class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
<!-- MySQL 数据库的驱动-->
<property name="driverClassName"
value="com.mysql.jdbc.Driver"/>
<!-- 数据库的 URL-->
<property name="url"
value="jdbc:mysql:///localhost:3306/CRM"/>
<!-- 指定数据库的用户名-->
<property name="username" value="root"/>
<!-- 指定数据库的密码-->
<property name="password" value=" root "/>
</bean>
```

4.2.2 Spring 中的 ApplicationContext.xml 配置事务

由于事务管理有个很明确的横切概念，进行声明式事务管理，从而避免在众多方法中重复编写大量的事务处理代码。系统把事务管理定义在 transactionManager 方法上，这样，核心代码就只须关注业务逻辑，而将事务管理完全交给配置文件。事务处理的 AOP 配置如下：

```
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.
DataSourceTransactionManager">
<property name="dataSource">
<ref local="dataSource"/>
</property></bean>
```

4.2.3 使用 Spring 中 IoC 管理 DAO 组件

声明由容器管理组件之间的依赖关系，把对组件之间依赖关系的控制进行了倒置。这样，组件间就不存在硬编码的关联，任何组件都可以最大程度地得到重用。Spring 通过 setter 方法注入依赖关系，并在配置上下文文件 application Context.xml 中便配置可轻松实现组件间的引用，指定之间的依赖关系。以 UserRemindDAO 为例，具体配置如下：

```
<bean id=" UserRemindDAO "
class="com.crm.dao.UserRemindDAO ">
<property name="dataSource">
<ref local="dataSource"/>
</property></bean>
```

4.2.4 创建数据操作类

现在以查找客户提醒的 isExist()方法为例，先为该方法创建一个 JdbcTemplate 对象，然后调用该类的 queryForInt() 方法统计 SQL 语句的单一数值，如果该数值大于 0，则表示该有客户需要被提醒，则返回 true，否则返回 false。

```
public boolean isExist(String assetName){
 JdbcTemplate jdbcTemplate = new JdbcTemplate(dateSource);
 String sql = "SELECT count(*) FROM userremind WHERE
username="+ username "";
 Int count = jdbcTemplate.queryForInt(sql);
 if(count>0){return true;}
else{return false;}}
```

以上过程即可实现数据持久化操作，由此可见，采用 Spring JDBC 框架在很大程度上简化系统编程代码，提高开发效率。

4.2.5 在 Spring 中配置 Facade 类

```
<bean id=" UserFacade "
class="com.crm.facade.
UserFacade ">
<!-把 DAO 做为属性注入到 Facade 中去 ->
<property name=" UserRemindDAO"
ref ="UserRemindDAO" /> …
</property> </bean>
```

接下来在 Service 层就可以使用采用这种面向接口而非面向类的编程，从而可以减少组件间的耦合度并极大提升开发过程。具体操作如下：

```
public class userFacade {
  private UserRemindDAO
userRemindDao; …
 public void setUserRemindDao
(UserRemindDao dao){
    this. UserRemindDao =dao;}
…}
```

5 结束语

本文给出一种能够有效满足目前电信行业客户关系管理数据持久层的构建方法，即利用 Facade 模式、VO 模式、DAO 模式以及 Spring 框架构建一个全新的数据持久层框架模型，并证明结合使用这个全新的数据持久层框架模型确实可以：(1)降低网络开销；(2)DAO 的可重用性；(3)简化系统编程代码，提高软件开发效率，同时提高系统的性能和维护性，使得系统在质量上有质的飞跃。

参考文献

[1] 苗晓辉. 基于 J2EE 的数据持久化的研究与实现[J]. 计算机工程, 2007, 33(5): 272-274.

[2] 李光俊, 华庆一, 吴海松. 基于 AOP 技术的 Composite 模式的改进[J]. 计算机工程, 2008, 34(10): 73-74, 77.

[3] 孙小锥, 上官右黎, 文福安. 基于轻量级 J2EE 框架的网络教学系统[J]. 计算机工程, 2008, 34(6): 266-267.

[4] 李守振, 张南平, 常国锋. Web 应用分层与开发框架设计研究[J]. 计算机工程, 2006, 32(22): 274-276.

编辑 陈 文