可重用 Java 数据库操作组件的设计和实现*

岳旭平，王道顺
(清华大学 计算机系，北京 100084)

摘 要：基于 Data Access Object(DAO) 设计模式,利用抽象、接口、类反射技术设计和实现了可重用 Java 数据库操作组件。它封装了注册和载入 JDBC 驱动程序、建立数据库连接、运行 Structured Query Language(SQL) 语句及处理查询结果集、查询结果集的分页显示操作,简化了 Java DataBase Connectivity(JDBC) 代码的编写,通过编写 SQL 语句,完成与数据库的交互。它设计的目标是简化中小型 Java 数据库应用的开发,提高开发的速度和测试的速度,降低数据库应用与数据来源的相关性,实现集中管理数据存取逻辑的功能。它与 Struts 框架有机结合,可进一步提高开发 Web 应用的效率。为方便使用它,通过一个例子给予了详细描述。

关键词：DAO 设计模式；数据库；JDBC；连接；抽象；Struts；类反射；JavaBeans
中图法分类号：TP311.13 **文献标识码**：A **文章编号**：1001-3695(2006) 02-0160-03

Design and Implementation of Module for Database in Java

YUE Xu-ping, WANG Dao-shun
(Dept. of Computer Science & Technology, Tsinghua University, Beijing 100084, *China)*

Abstract: The module for the database in Java through abstract is designed and Implemented, interface and reflection technique. It encloses the common operations such as registering and loading JDBC driver, building database connection, running SQL and dealing with result sets, paging the record. It simplifies the coding of JDBC and completes the operations of accessing database through writing SQL. It will simplify the development of the database application in Java, make the procedure of the development more easy, decrease the relativity between the database and the application and centralize the management of the data access. It can integrate with Struts model perfectly. This paper will demonstrate how to use this module through a simple example.

Key words: DAO Design Pattern; Database; JDBC; Connection; Abstract; Struts; Reflection Tag; JavaBeans

在数据库应用开发中,设计好的组件,可使开发过程易于控制、降低编码复杂性、提高代码的重用性和可维护性。Struts 是 Apache 组织的开放源码项目,基于 Model-View-Controller (MVC) 的设计模式, Web 应用程序的“展示逻辑”、“商业逻辑”与“工作流程”三者分别写成不同的应用程序组件,利用控制器来分离模型和视图。该框架成为使用 Java 创建 Web 应用的最流行的框架工具[1]。Model 部分的业务组件封装了具体的业务逻辑,可以使用 Enterprise JavaBeans (EJB) 和 JavaBeans 来实现。EJB 适用于大型应用,在中小型应用中使用 JavaBeans 将降低系统的开销。在文献[2] 中描述了应用 Struts 框架开发 Web 应用的基本步骤。在文献[3] 中介绍了如何使用 Java DataBase Connectivity (JDBC) 技术和类反射技术构造可重用的 Query 方法组件,该方法类似于 EJB 中的 Finder 方法,屏蔽了 Structured Query Language(SQL) 语句的编写,本文未对其他数据库操作进行抽象封装。在文献[4] 中讨论了 Java 数据库分页技术,对每个查询编写相应 PageBean,在 JavaServer Pages(JSP) 页面中嵌入处理页面的脚本。在文献[5] 中讨论了使用 Java API 和 Java2 组件进行数据库开发的优缺点,使用 Java API 灵活,但编程复杂,使用 Java2 组件又缺乏灵活性。在文献[6] 中介绍了符合 CORBA 规范的数据库访问中间件的设计,该技术解决了在分布式网络环境中透明访问异种数据库资源的问题。在文献[7] 中介绍了使用 JDBC-ODBC 桥建立数据库连接,并把数据库连接保存到 Session 中的 JavaBeans 的设计。

收稿日期：2005-02-25；修返日期：2005-04-16
基金项目：国家自然科学基金资助项目(90304014)

1 可重用 Java 数据库组件设计与实现

1.1 分析

使用 JDBC 进行数据存取操作时,主要完成同一个数据库建立连接、向数据库发送 SQL 语句、处理数据库返回的结果集三个任务[8~10]。在实际开发时,首先必须编写建立数据库连接的代码,而且开发人员必须熟悉 JDBC。如果数据库的表非常多,维护数据存取逻辑的代码非常困难,在编写代码时,也存在许多冗余。在开发 Web 应用时,使用 Struts 模型能大大提高开发效率,中小型应用中 Model 部分的业务逻辑通过设计 JavaBeans 组件实现。在查询数据时,分页显示查询结果集将方便用户的浏览。在文献[4] 中需要为每个查询编写相应的 PageBean 实现分页显示。在文献[3] 中组件使用类似于 EJB 中的 Finder 方法,屏蔽了 SQL 语句的编写。本文从简化 JDBC 操作编写、集中管理数据存取逻辑、集中管理分页显示查询结果集的角度出发,基于 DAO 模式,设计了能与 Struts 模型有机结合并完成数据存取逻辑的组件,设计的核心思想是通过抽象、类反射技术,简化 JDBC 的编程,封装建立数据库连接的步

骤, 封装分页显示操作的编写, 通过编写 SQL 语句、修改属性文件, 完成数据存取逻辑。

1.2　设计和实现

设计主要使用了抽象、接口和类反射技术, 基于 DAO 设计模式, 定义的组件主要由抽象类 BaseDao、接口 DaoProxy、类 PageBean、PageTag 组成。

BaseDao 类图如图 1 所示。

图 1　BaseDao 类图

该类定义了抽象方法 insertString, deleteString, updateString, queryString, queryListString, queryCountString, 通过实现这些抽象方法返回相应的 SQL 语句字符串与数据源进行交互。

该类定义了抽象方法

fillInsertStatement, fillUpdateStatement,

fillDeleteStatement,

fillFindListStatement,

fillFindStatement, fillFindStatement 为 SQL 语句中的变量赋值。

该类定义了抽象方法 extractFromFindResultSet, extractFromFindResultSet 从记录集中解析表中记录。

上述参数中, Flag 指对表更新、删除、查询的不同类型。而 Object 为用户定义的类型。

定义了接口 DaoProxy, 该接口完成对数据存取的逻辑。类图如图 2 所示。

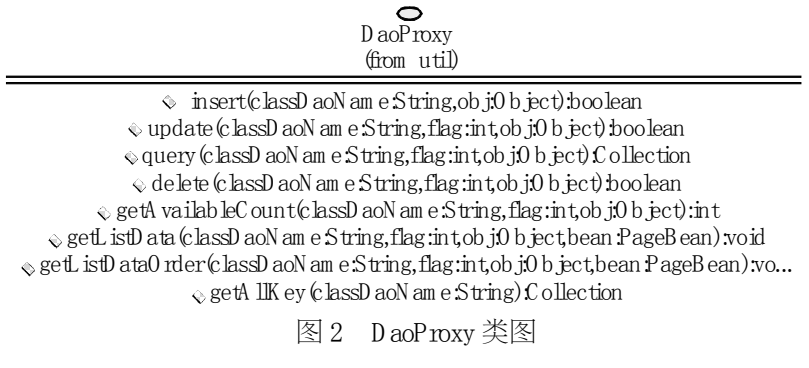

图 2　DaoProxy 类图

参数 classDaoName 指实现了 BaseDao 中抽象方法的类的类名。该方法通过类反射机制, 使不同的实现具有不同的表现行为。

本文提供了两个 DaoProxy 接口的实现 AppDaoProxy, JspDaoProxy, AppDaoProxy 应用在应用程序环境中, JspDaoProxy 应用在 Web 环境中。类关系图如图 3 所示。

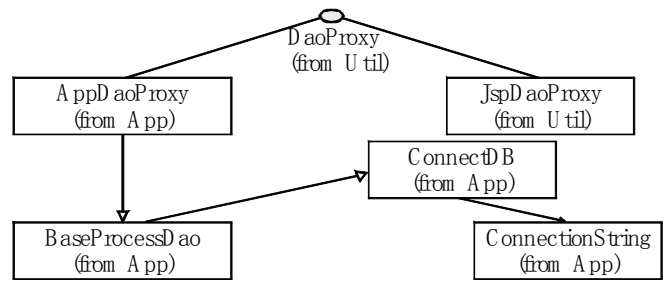

图 3　类关系图

AppDaoProxy 继承自 BaseProcessDao, BaseProcessDao 类完成数据库连接的处理。

JspDaoProxy 定义了类变量 DataSource pool, 该对象从 ServletContext 对象中获取。连接的建立过程将在下一部分详细介绍。

PageBean 类封装了分页显示的数据, 包括当前页码、总页数、当前页的显示数据、每页显示的记录数、分页标记等, 完成对查询结果的分页显示功能。

PageTag 类继承自 TagSupport, 实现了在页面显示总记录数、根据用户点击跳转页面等功能。

PageTag 实现: 根据 Tag 的属性 pageBeanName 从 PageContext 对象中获取 PageBean 的实例化对象, 向响应流中输出 JScript 脚本和显示页码、记录总数的 HTML 数据。

PageTag 的使用:

```
< app: pageTag pageBeanName = " pagebean"  action = " accountBalanceSearchAction" / >
```

其中 action 属性指明页面跳转时, 处理页面请求的 Servlet; PageBeanName 属性指明 PageBean 实例的名字。

PageTag 在浏览器中的显示结果如图 4 所示。

图 4　PageTag 显示结果

通过 PageTag 标签, 进一步简化了 Web 应用中分页操作的编码。

2　数据库连接的处理

在不同的应用环境中, 数据库连接的管理方式不同, 在 Web 应用中, 因为存在并发访问数据库资源, 一般使用数据库连接池来提高程序的运行效率, 而在应用程序中, 只需直接获得与数据库资源的物理连接。本文对数据库的连接进行了封装。

2.1　应用程序

修改 Properties/Connection. properties 文件中的下列值:

```
SqlDriver = 数据库 JDBC 驱动
SqlUrl = 访问数据库 url 地址
UserName  = 数据库用户名
Password  = 数据库密码
示例: sqlserver 2000
SqlDriver = com. microsoft. jdbc. sqlserver. SQLServerDriver
SqlUrl = jdbc: microsoft: sqlserver: //192. 168. 1. 20: 1434; DatabaseName = 数据库名
```

该功能由 networkj2ee. base. app. ConnectDB 实现, 该类完成从文件读取属性字符串并建立连接的功能。在 BaseProcessDao 定义了 static ConnectDB db:

```
static {
db = new ConnectDB( ) ;
}
```

AppDaoProxy 继承 BaseProcessDao, 数据库连接对象在构造 AppDaoProxy 实例时创建。

2.2 Web 应用

在 Web. xml 中添加以下 Servlet 配置:

```
< Servlet >
< Servlet-name > connectdb < / Servlet-name >
< Servlet-class > netj2ee. jsp. dao. util. ConnectDB < / Servlet-class >
< init-param >
< param-name > recourceName < / param-name >
< param-value > java: comp/env/jdbc/interpbx < / param-value >
< / init-param >
< load-on-startup > 1 < / load-on-startup >
< / Servlet >
```

使用相应的数据资源名代替 java: comp/env/jdbc/interpbx。

该功能由组件中的 netj2ee. jsp. dao. util. ConnectDB 类完成, 该类的核心代码:

```
Context context = new InitialContext( ) ;
String resourceName = this. getInitParameter( " recourceName" ) ;
DataSource ds = ( DataSource) context. lookup( resourceName) ;
this. getServletContext( ) . setAttribute( Constants. POOL, ds) ;
```

数据库的连接池在 Web 程序启动时建立, 并保存在 ServletContex 对象中。使用数据库连接池时, 从 ServletContex 对象中获取。

3 使用该组件

3.1 应用该组件的步骤

编写 Info 类: 该类封装了表中各字段的值。

编写 Column 类: 该类封装了表中的列名。

编写 queryInfo 类: 该类封装了查询条件。

编写 updateInfo 类: 该类封装了更新字段的值。

编写 Dao 类: 该类封装了对表进行的各种操作, 继承自组件中的抽象类 BaseDao, 实现了 BaseDao 中的抽象方法。

3.2 实现 BaseDao 中相应的的抽象方法

insert 操作实现 insertString fillInsertStatement, Update 操作实现 updateString, fillUpdateStatemen, 删除操作实现 deleteString 和 fillDeleteStatement, 查询实现 queryString, fillFindStatement, 分页查询实现 queryListString, 处理结果集实现 extractFromFindResultSet, 获取表的所有主键值实现 getAllKey 和 extractFromFindResultSet 。

3.3 例子

```
public class CharacterDao extends BaseDao {
  public CharacterDao( ) {
  }
/* 插入操作的实现* /
protected void fillInsertStatement( PreparedStatement ps, Object obj)
throws java. sql. SQLException, networkj2ee. base. util. DataTypeException
{
  CharacterInfo info;
  if ( obj instanceof CharacterInfo) {
    info = ( CharacterInfo) obj;
    ps. setString( 1, info. getPostNO( ) ) ;
    ps. setString( 2, info. getPostDes( ) ) ;
    ps. setInt( 3, info. getChaType( ) ) ;
  }
  else {
    DataTypeException e = new DataTypeException(
      " the type of the object error! " ) ;
    throw e;
  }
}
public String insertString( ) {
  String sql;
  sql = " insert into " + TableName. Character + " values ( " + "
?, ?, ? ) " ;
  return sql;
}
...
  }
```

CharacterInfo 采用了 Value Object 设计模式, 该类简单封装了 Character 表中的字段, 包含了该表中一条记录的值。在 insertString 方法中编写带参数的 SQL 语句, 在 fillInsertStatement 方法中为 SQL 语句中的变量赋值。

3.4 在 Struts 框架中使用该组件

该组件与 Struts 模型结合, 能够充分利用 Struts 框架的优点, 达到显示与业务逻辑分开。

在 ActionForm 类中定义 Info 类:

```
private CharacterInfo info = new CharacterInfo( ) ;
public void setInfo( CharacterInfo info) {
    this. info = info;
  }
  public CharacterInfo getInfo( ) {
    return info; }
```

在 JSP 页面中, 通过 ActionForm Bean 的实例来传递表单的值:

```
< html: text property = " info. postNO" size = " 30" styleClass = " text-
boxstyle" / >
```

在 Action 中的调用:

```
CharacterActionForm form = ( CharacterActionForm) actionForm;
CharacterInfo info = form. getInfo( ) ; /* 获得表单中的信息* /
JspDaoProxy proxy = new JspDaoProxy( ) ;
proxy. setPoolConnection( ( DataSource) this. getServlet( ) . getServlet-
Context( ) . getAttribute( Constants. POOL) ) ; /* 获得连接* /
proxy. insert( ClassName. characterDao, info) ;
```

3.5 在应用程序中的调用该组件, 完成数据库操作, 以插入例

```
CharacterInfo info = new CharacterInfo( ) ;
AppDaoProxy proxy = new AppDaoProxy( ) ;
info. setPostNO( " 010" ) ;
proxy. insert( ClassName. characterDao, info) ;
```

4 结束语

本文设计了基于 DAO 设计模式的可重用 Java 数据库操作组件, 并通过实例对组件的使用进行了说明。应用该组件开发 Java 数据库应用程序, 只需根据需求实现 BaseDao 类中的抽象方法, 修改配置文件, 调用 DaoProxy, 通过 DaoProxy 集中管理数据存取逻辑。该组件能够简化开发步骤、降低编程复杂性、提高开发、测试的效率, 便于代码的维护和重用。同时, 使软件开发过程更加容易, 会写 SQL 语句, 即可编写。通过文件配置, 简化了连接的建立和管理。通过 PageTag 的编写, 简化了 Web 应用中的分页显示功能的开发。该组件能与 Struts 模型有机结合, 进一步提高开发中小型 Web 应用的效率。该组件已在 IP 网分布式电话程控交换机管理系统中应用。 (下转第 166 页)