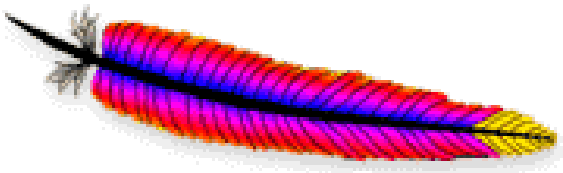
[2008]

commons-dbutils 使用



Apache Commons

http://commons.apache.org/

commons

dbutils

邱加永

[2008-8-20]

声明：

此文档为免费资料，仅供个人研究和学习之用，不得用于任何商业目的。在免费、且无任何附加条件的前提下，可在网络媒体中自由传播。转载时请保持文档的完整性，作者不能保证文档的完全正确，希望大家对其中的错误进行更正并与我联系。

在写作过程中，我参考了网上大量的资料，并摘取了其中的一部分内容，在这里向这些资料的作者表示深深的感谢，如果您认为我侵犯了您的著作版权，请告之我，我会将相关内容删除并将结果通知您。

本文档仅代表作者本人的观点。

联系 QQ：20978405

联系邮箱：qjyong@gmail.com

技术博客： http://blog.csdn.net/qjyong

技术博客 2：http://qjyong.javaeye.com/

目录

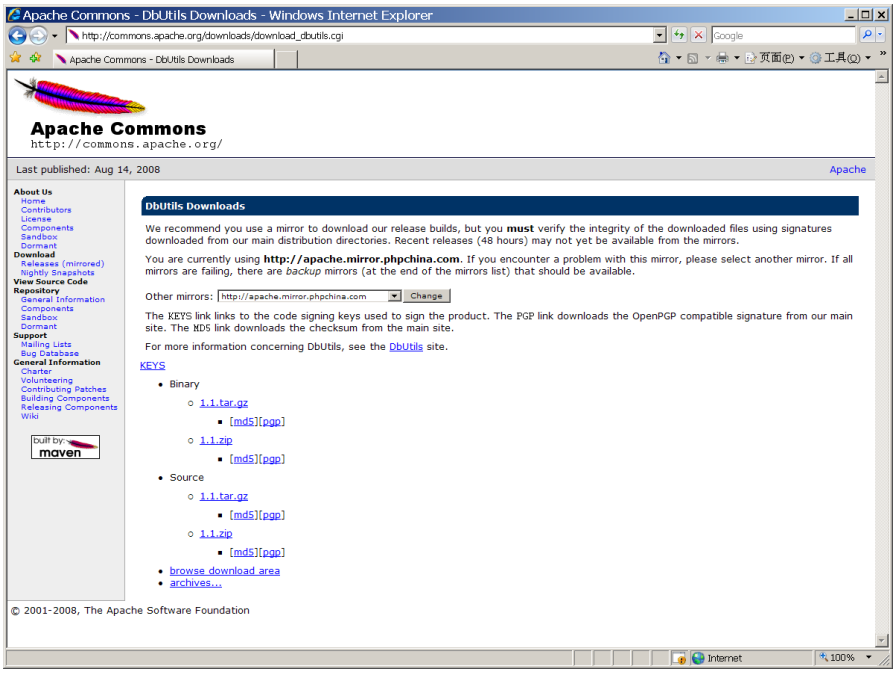
1. 介绍

commons-dbutils 是 Apache 组织提供的一个开源 JDBC 工具类库，能让我们更简单的使用 JDBC。它是一个非常小的类包，花几分钟的时间就能掌握它的使用。

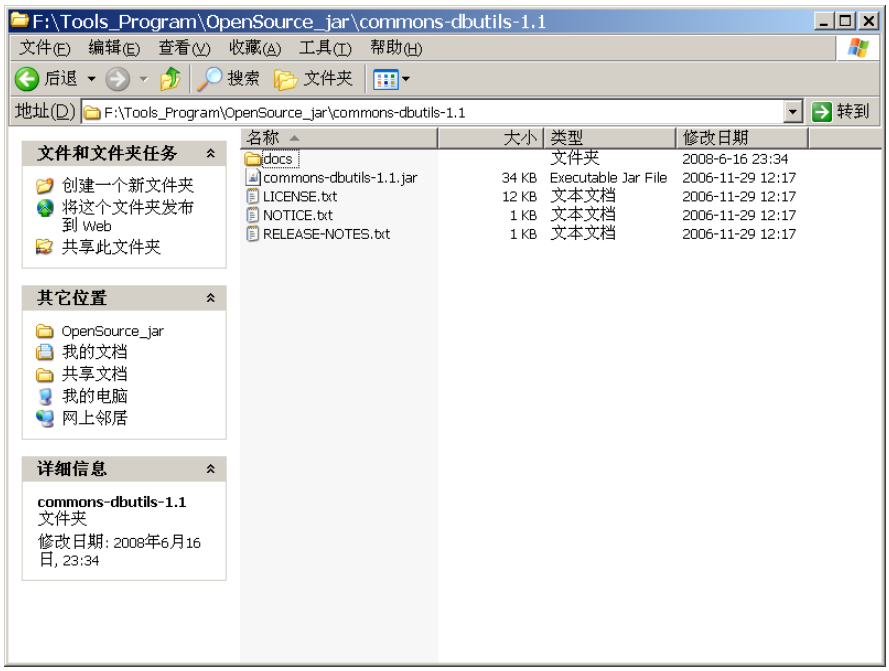
2. 下载

进入 apache 官方网站的 commons-dbutils 网页：http://commons.apache.org/dbutils/

选择左边导航栏的 Download 进入下载页面：



Binary：是编译好的 jar 包；source 是源代码包。下载一份 Binary 到本地。解开后如下图所示：



其中 commons-dbutils-1.1.jar 就是供使用的 jar 包；docs 是这个组件的帮助文档。下面就通过示例介绍它的使用。

3. API 介绍

commons-dbutils 是一个非常小的类包， 无需花费太多时间去阅读它的 doc，核心只是两个类 org.apache.commons.dbutils.DbUtils、org.apache.commons.dbutils.QueryRunner 和一个接口 org.apache.commons.dbutils.ResultSetHandler。

3.1. DbUtils 类

DbUtils 是一个用来做一些如关闭连接、装载 JDBC 驱动程序之类的常规工作提供有用方法的类，它里面所有的方法都是静态的。

这个类里的重要方法有：

- public static void close(…) throws java.sql.SQLException： DbUtils 类提供了三个重载的关闭方法。这些方法检查所提供的参数是不是 NULL，如果不是的话，它们就关闭 Connection、Statement 和 ResultSet。
- public static void closeQuietly(…): 这一类方法不仅能在 Connection、Statement 和 ResultSet 为 NULL 情况下避免关闭，还能隐藏一些在程序中抛出的 SQLEception。如果你不想捕捉这些异常的话，这对你是非常有用的。在重载 CloseQuietly 方法时，特别有用的一个方法是 closeQuietly(Connection conn,Statement stmt,ResultSet rs)，这是因为在大多数情况下，连接、声明和结果集（ResultSet）是你要用的三样东西，而且在最后的块你必须关闭它们。使用这一方法，你最后的块就可以只需要调用这一方法即可。

- public static void CommitAndCloseQuietly(Connection conn)： 这一方法用来提交连接，然后关闭连接，并且在关闭连接时不向上抛出在关闭时发生的一些 SQL 异常。
- public static boolean loadDriver(java.lang.String driverClassName)：这一方法装载并注册 JDBC 驱动程序，如果成功就返回 true。使用这种方法，你不需要去捕捉这个异常 ClassNotFoundException。

3.2. *QueryRunner 类*

这个类简单化了执行 SQL 查询，它与 ResultSetHandler 组合在一起使用就可能完成大部分的数据库操作，它能够大大减少你所的编码量。

QueryRunner 类提供了两个构造方法：一个是默认的构造方法；另一个需要一个 javax.sql.DataSource 来作为参数。因此，在你不用为一个方法提供一个数据库连接的情况下，提供给构造器的 DataSource)被用来获得一个新的连接并将继续进行下去。

这个类中的重要方法包括以下这些：

- public Object query(Connection conn, String sql, Object[] params, ResultSetHandler rsh) throws SQLException：执行一个查询操作，在这个查询中，对象数组中的每个元素值被用来作为查询语句的置换参数。该方法它会内在地处理 PreparedStatement 和 ResultSet 的创建和关闭。最重要的是参数 ResultSetHandler 会把从 ResultSet 中获得的数据转换成一个更容易的或是应用程序特定的格式供我们使用。
- public Object query(String sql, Object[] params, ResultSetHandler rsh) throws SQLException: 这几乎与第一种方法一样；唯一的不同在于它不将数据库连接提供给方法，并且它是从提供给构造方法的数据源(DataSource) 或使用的 setDataSource 方法中重新获得的。
- public Object query(Connection conn, String sql, ResultSetHandler rsh) throws SQLException： 执行一个不需要置换参数的查询操作。
- public int update(Connection conn, String sql, Object[] params) throws SQLException: 用来执行一个更新（插入、更新或删除）操作。
- public int update(Connection conn, String sql) throws SQLException：用来执行一个更新操作，不需要置换参数。

3.3. *ResultSetHandler 接口*

正如它的名字所提示的，这一接口执行处理一个 jaca.sql.ResultSet，将数据按要求转换为另一种形式。

ResultSetHandler 接口提供了一个单独的方法：Object handle (java.sql.ResultSet .rs)。因此任何 ResultSetHandler 的执行需要一个结果集（ResultSet）作为参数传入，然后才能处理这个结果集，再返回一个对象。因为返回类型是 java.lang.Object，所以除了不能返回一个原始的 Java 类型之外，其它的返回类型并没有什么限制。。

common-dbutils 组件包针对这个接口提供了九个实现类：

- ArrayHandler：把结果集中的第一行数据转成对象数组。
- ArrayListHandler：把结果集中的每一行数据都转成一个对象数组，再存放到 List 中。
- BeanHandler：将结果集中的第一行数据封装到一个对应的 JavaBean 实例中。

- BeanListHandler：将结果集中的每一行数据都封装到一个对应的 JavaBean 实例中，存放到 List 里。
- ColumnListHandler：将结果集中某一列的数据存放到 List 中。
- KeyedHandler：将结果集中的每一行数据都封装到一个 Map 里，然后再根据指定的 key 把每个 Map 再存放到一个 Map 里。
- MapHandler：将结果集中的第一行数据封装到一个 Map 里，key 是列名，value 就是对应的值。
- MapListHandler：将结果集中的每一行数据都封装到一个 Map 里，然后再存放到 List
- ScalarHandler：将结果集中某一条记录的其中某一列的数据存成 Object

如果这九个实现类中没有任何一个提供了你想要的服务，你可以自己写一个实现类。

3.4. 其它类和接口

org.apache.commons.dbutils.QueryLoader 类：属性文件加载器，主要用于加载属性文件中的 SQL 到内存中。

org.apache.commons.dbutils.wrappers.SqlNullCheckedResultSet 类：这个类是用来对 sql 语句执行完成之后的的数值进行 null 的替换。

org.apache.commons.dbutils.wrappers.StringTrimmedResultSet 类: 去除 ResultSet 中字段的左右空格。

org.apache.commons.dbutils.RowProcessor 接口： 它提供了把结果集的行数据转换成其它格式的功能。它的实现类是 org.apache.commons.dbutils.BasicRowProcessor 类。

4. 使用

4.1. 准备数据库表

这边使用 MySQL5 数据库服务器，数据库名是 test，建表语句：

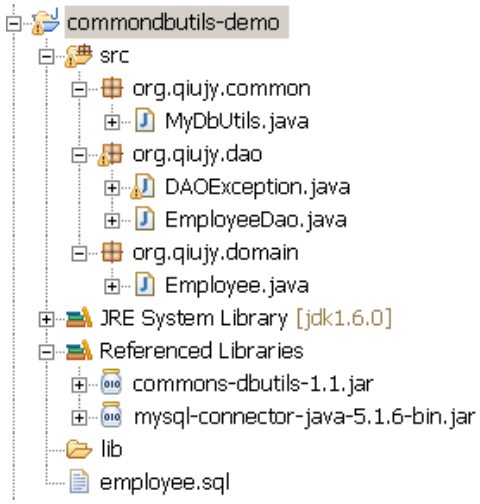
```
CREATE TABLE `employee` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) default NULL,
  `age` int(11) default NULL,
  `position` varchar(50) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

数据如下：

id	name	age	position
1	赵六	18	JavaEE工程师
2	李四	26	JavaEE高级工程师
3	张三	25	JavaEE工程师
4	王五	28	JavaEE分析师
5	abc	23	测试工程师
6	admin	32	项目经理

4.2. 创建 java 工程

整个工程结构如下图所示



主要的三个类，一个是 MyDbUtils 数据库连接工具类，Employee 实体类，另一个是 EmployeeDao 类。

4.2.1. MyDbUtils.java

```
/**
 * ClassName: MyDbUtils.java
 * Author: qiujy
 * CreateTime: 2008-8-20
 * EMAIL: qjyong@gmail.com
 * Copyright 2008 ++YONG All rights reserved.
 */
package org.qiujy.common;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;

/**
 * 数据库连接工具
 * @author qiujy
 *
 */
public class MyDbUtils {
    private MyDbUtils() {
    }
```

```
    public static final String MYSQL_URL =
"jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=utf-8";
    public static final String MYSQL_DRIVER = "com.mysql.jdbc.Driver";
    public static final String MYSQL_USERNAME = "root";
    public static final String MYSQL_PASSWORD = "root";

    public static Connection getConnection(String url, String driver,
            String username, String pwd) throws SQLException {
        DbUtils.loadDriver(driver);
        return DriverManager.getConnection(url, username, pwd);
    }

    public static Connection getConnection() throws SQLException {
        return getConnection(MYSQL_URL, MYSQL_DRIVER, MYSQL_USERNAME,
                MYSQL_PASSWORD);
    }
}
```

4.2.2. Employee 类

```
/**
 * ClassName: Employee.java
 * Author: qiujy
 * CreateTime: 2008-8-20
 * EMAIL: qjyong@gmail.com
 * Copyright 2008 ++YONG All rights reserved.
 */
package org.qiujy.domain;

/** 员工实体类 */
public class Employee {

    private int id;
    private String name;
    private int age;
    private String position;

    public Employee() {
    }

    public Employee(int age, String name, String position) {
```

```
            super();
            this.age = age;
            this.name = name;
            this.position = position;
      }

      //以下省略所有属性的 getter 和 setter
      ……
      @Override
      public String toString() {
            return "[id=" + this.id + ",name=" + this.name + ",age=" + age
                        + ",position=" + this.position + "]";
      }
}
```

4.2.3. EmployeeDao 类

```
/**
 * ClassName: EmployeeDao.java
 * Author: qiujy
 * CreateTime: 2008-8-20
 * EMAIL: qjyong@gmail.com
 * Copyright 2008 ++YONG All rights reserved.
 */
package org.qiujy.dao;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.BeanHandler;
import org.apache.commons.dbutils.handlers.BeanListHandler;
import org.qiujy.common.MyDbUtils;
import org.qiujy.domain.Employee;

/** 员工 dao */
public class EmployeeDao {
      /**
       * 新增一个员工实例
```

```
     * @param empl
     * @param conn
     * @throws DAOException
     */
    public void insertEmployee(Connection conn, Employee empl) throws DAOException {
        String sql = "INSERT INTO employee(name,age,position) VALUES(?,?,?)";
        try {
            QueryRunner qr = new QueryRunner();
            Object[] params = {empl.getName(), empl.getAge(),empl.getPosition()};
            qr.update(conn, sql, params);
        } catch (SQLException e) {
            throw new DAOException(e);
        }
    }

    /**
     *  根据 ID 删除一个员工实例
     * @param conn
     * @param id
     * @throws DAOException
     */
    public void deleteEmloyeeById(Connection conn, int id) throws DAOException {
        String sql = "DELETE FROM employee WHERE id=?";
        try {
            QueryRunner qr = new QueryRunner();
            qr.update(conn, sql, id);
        } catch (SQLException e) {
            throw new DAOException(e);
        }
    }

    /**
     *  更新员工的信息
     * @param conn
     * @param empl
     * @throws DAOException
     */
    public void updateEmployee(Connection conn, Employee empl) throws DAOException {
        String sql = "UPDATE employee SET name=?,age=?,position=? WHERE id=?";
        try {
            QueryRunner qr = new QueryRunner();
            Object[] params = { empl.getName(), empl.getAge(),
```

```
                    empl.getPosition(), empl.getId() };
            qr.update(conn, sql, params);
        } catch (SQLException e) {
            throw new DAOException(e);
        }
}

/**
 * 根据 ID 获取该员工实例
 * @param conn
 * @param id
 * @return
 * @throws DAOException
 */
public Employee getEmployeeById(Connection conn, int id) throws DAOException {
        Employee empl = null;
        String sql = "SELECT id,name,age,position FROM employee WHERE id=?";
        try {
            QueryRunner qr = new QueryRunner();
            empl = (Employee) qr.query(conn, sql, id,
                                new BeanHandler(Employee.class));
        } catch (SQLException e) {
            throw new DAOException(e);
        }
        return empl;
}

/**
 * 获取所有的员工列表
 * @param conn
 * @return  员工实例列表
 * @throws DAOException
 */
@SuppressWarnings("unchecked")
public List<Employee> getEmployeeList(Connection conn) throws DAOException {
        List<Employee> list = null;
        String sql = "SELECT id,name,age,position FROM employee";
        try {
            QueryRunner qr = new QueryRunner();
            list = (List<Employee>) qr.query(conn, sql,
                                new BeanListHandler(Employee.class));
        } catch (SQLException e) {
```

```
                throw new DAOException(e);
            }
            return list;
        }

        public static void main(String[] args) throws SQLException {
            Connection conn = MyDbUtils.getConnection();
            List<Employee> list = new EmployeeDao().getEmployeeList(conn);
            for(Employee empl : list){
                System.out.println(empl);
            }
            DbUtils.close(conn);
        }
}
```

5. 附源码

commondbutils-demo.rar