Java+MyEclipse+Tomcat (五)DAO和Java Bean实现数据库和界面分开操作

正如前面一篇文章的介绍，当使用Servlet提交表单和JSP数据库查询时，总是相互交叉着的处理，要么在JSP中通过<%...%>内嵌Java代码操作数据库，要么JSP中通过Post方法提交表单Form，在Java中通过Servlet获取请求/响应，再通过Java中out.println("<HTML>...")输出数据库中值。

此篇文章主要讲述通过DAO和Java Bean操作数据库，把链接数据库、数据库操作、前端界面显示分模块化实现。参考前文：

Java+MyEclipse+Tomcat (一)配置过程及jsp网站开发入门
Java+MyEclipse+Tomcat (二)配置Servlet及简单实现表单提交
Java+MyEclipse+Tomcat (三)配置MySQL及查询数据显示在JSP网页中
Java+MyEclipse+Tomcat (四)Servlet提交表单和数据库操作

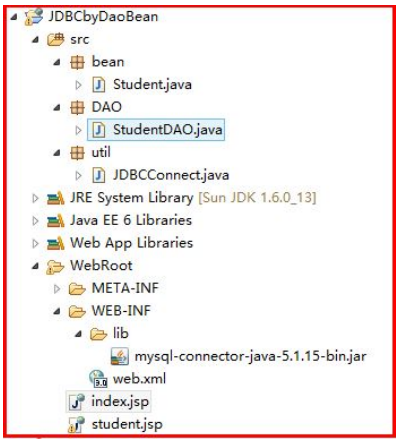
DAO和Java Bean是对JDBC进行分层、模块化的最有效两个方法。DAO(数据库操作对象，Database Access Object)是JDBC下常用模式，DAO出现之前，操作数据库的代码与业务代码都出现在Servlet或者JSP中，不利用业务代码的分离。DAO出现后，所有与数据库相关的操作全被拿到了DAO层实现，Servlet或JSP只操作Java Bean或者DAP层，而DAO层值操作数据库。

下面直接上代码，希望文章对你有所帮助~文章部分参考Java Web王者归来。

下载地址：http://download.csdn.net/detail/eastmount/8714395

一. 项目结构

该项目的结构如下图所示：



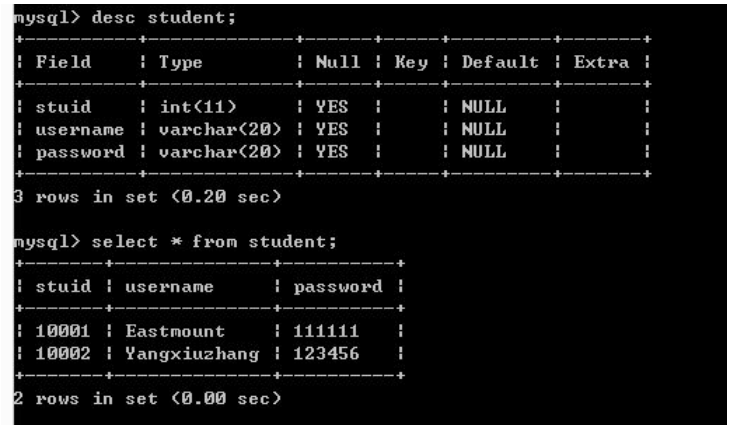
其中bean中Student.java是对应数据库学生表，主要包括setId()、getId()等操作；DAO中StudentDAO.java是对学生表的数据库增删改查操作；util中JDBCConnect.java主要是连接数据库MySQL的操作；student.jsp是显示数据的JSP前端界面。同时需要lib文件夹中加载mysql-connector-java.jar包。

二. 创建数据库

打开MySQL，输入默认超级root用户的密码，然后数据库的操作如下代码：

```
--创建数据库
create database TestDao;
--使用数据库
use TestDao;
--创建学生表
create table student(
        stuid int,
        username varchar(20),
        password varchar(20)
);
--插入数据
insert student(stuid,username,password)
        values ("10001","Eastmount","111111");
insert student(stuid,username,password)
        values ("10002","Yangxiuzhang","123456");
--显示表结构
desc student;
--查询表中数据
select * from student;
```

其中表结构和表中数据显示如下图：



三. Java代码

1.在src下新建文件夹util，然后添加类JDBCConnect.java。代码如下：

```
package util;

import java.sql.*;
import com.mysql.jdbc.Driver;

public class JDBCConnect {

        //获取默认数据库连接
        public static Connection getConnection() throws SQLException {
                return getConnection("TestDAO", "root", "123456"); //数据库名 默认用户 密码
        }

        //连接数据库    参数:数据库名 root登录名 密码
        public static Connection getConnection(String dbName, String userName,
                        String password) throws SQLException {

                String url = "jdbc:mysql://localhost:3306/" + dbName
```

```
                        + "?characterEncoding=utf-8";
                //连接MySQL"com.mysql.jdbc.Driver"
                                                        DriverManager.registerDriver(new Driver());
                return DriverManager.getConnection(url, userName, password);
        }

        //设置 PreparedStatement 参数
        public static void setParams(PreparedStatement preStmt, Object... params)
                        throws SQLException {

                if (params == null || params.length == 0)
                        return;
                for (int i = 1; i <= params.length; i++) {
                        Object param = params[i - 1];
                        if (param == null) {
                                preStmt.setNull(i, Types.NULL);
                        } else if (param instanceof Integer) {
                                preStmt.setInt(i, (Integer) param);
                        } else if (param instanceof String) {
                                preStmt.setString(i, (String) param);
                        } else if (param instanceof Double) {
                                preStmt.setDouble(i, (Double) param);
                        } else if (param instanceof Long) {
                                preStmt.setDouble(i, (Long) param);
                        } else if (param instanceof Timestamp) {
                                preStmt.setTimestamp(i, (Timestamp) param);
                        } else if (param instanceof Boolean) {
                                preStmt.setBoolean(i, (Boolean) param);
                        } else if (param instanceof Date) {
                                preStmt.setDate(i, (Date) param);
                        }
                }
        }

        //执行 SQL，返回影响的行数 异常处理
        public static int executeUpdate(String sql) throws SQLException {
                return executeUpdate(sql, new Object[] {});
        }

        //带参数执行SQL，返回影响的行数 异常处理
        public static int executeUpdate(String sql, Object... params)
                        throws SQLException {

                Connection conn = null;
                PreparedStatement preStmt = null;
                try {
                        conn = getConnection();
                        preStmt = conn.prepareStatement(sql);
                        setParams(preStmt, params);
                        return preStmt.executeUpdate(); //执行SQL操作
                } finally {
                        if (preStmt != null)
                                preStmt.close();
                        if (conn != null)
                                conn.close();
                }
        }
}
```

其中主要是调用getConnection(url, userName, password); 方法进行连接数据库操作，我数据库的名称为TestDAO，默认的连接对象为root，密码为123456。同时定义两个函数执行无参数的SQL语句操作和有参数的SQL语句

操作。

2.在src下新建文件夹bean，然后添加类Student.java。代码如下：

```
package bean;

public class Student {

        private Integer id;       //学号
        private String name;      //姓名
        private String password;  //密码
        public Integer getId() { return id; }
        public String getName() { return name; }
        public String getPassword() { return password; }
        public void setId(Integer id) { this.id =  id; }
        public void setName(String name) { this.name =  name; }
        public void setPassword(String pwd) { this.password = pwd; }
}
```

该Student中的变量及类型与数据库中一一对应，在通过get和set方法获取和设置其值。同样如果你的数据库中有老师、学校表，你只需要在bean文件夹下添加Teacher.java和School.java即可。

3.在src下新建文件夹DAO，然后添加类StudentDAO.java。代码如下：

```
package DAO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import bean.Student;
import util.JDBCConnect;

public class StudentDAO {

        //插入学生
        public static int insert(Student stu) throws Exception {

                String sql = "INSERT INTO student (stuid,username,password) VALUES (?,?,?) ";
                return JDBCConnect.executeUpdate(sql, stu.getId(),stu.getName(),stu.getPassword());
        }

        //更新学生姓名
        public static int update(Student stu) throws Exception {

                String sql = "UPDATE student SET stuid = ? WHERE username = ? ";
                return JDBCConnect.executeUpdate(sql,stu.getId(),stu.getName());
        }


        //删除操作
        public static int delete(Integer id) throws Exception {

                String sql = "DELETE FROM student WHERE stuid = ? ";
                return JDBCConnect.executeUpdate(sql, id);
        }

        //查找记录 某学号
```

```
public static Student find(Integer id) throws Exception {

        String sql = "SELECT * FROM student WHERE stuid = ? ";
        Connection conn = null;
        PreparedStatement preStmt = null;
        ResultSet rs = null;

        try {
                //链接数据库执行SQL语句
                conn = JDBCConnect.getConnection(); //连接默认数据库
                preStmt = conn.prepareStatement(sql);
                preStmt.setInt(1, id);
                rs = preStmt.executeQuery();
                //获取查询结果
                if (rs.next()) {
                        Student student = new Student();
                        student.setId(rs.getInt("stuid"));
                        student.setName(rs.getString("username"));
                        return student;
                } else {
                        return null;
                }

        } finally { //依次关闭 记录集 声明 连接对象
                if (rs != null)
                        rs.close();
                if (preStmt != null)
                        preStmt.close();
                if (conn != null)
                        conn.close();
        }
}

//查询所有学生信息
public static List<Student> listStudents() throws Exception {

        List<Student> list = new ArrayList<Student>();
        String sql = "SELECT * FROM student";
        Connection conn = null;
        PreparedStatement preStmt = null;
        ResultSet rs = null;

        try {
                conn = JDBCConnect.getConnection();
                preStmt = conn.prepareStatement(sql);
                rs = preStmt.executeQuery();
                while (rs.next()) {
                        //设置数据库中表参数 否则报错java.sql.SQLException: Column 'id' not found.
                        Student student = new Student();
                        student.setId(rs.getInt("stuid"));
                        student.setName(rs.getString("username"));
                        student.setPassword(rs.getString("password"));
                        list.add(student);
                }

        } finally {
                if (rs != null)
                        rs.close();
                if (preStmt != null)
                        preStmt.close();
                if (conn != null)
                        conn.close();
```

```
            }                   return list;
        }

}
```

通常DAO（Data Access Object）数据访问对象是负责与数据库连接，主要功能执行对数据表的CUDR操作。

C(Create)操作：创建记录，执行insert into语句；

U(Update)操作：业务表中对应的属性进行更新操作，执行update语句；

D(Delete)操作：将DTO对象对应的记录删除，执行delete语句；

R(Read)操作：读取表中数据，可以返回多个记录列表对应DTO对象多个List容器。

最初同学建议弄这个，不敢接触担心很复杂，但用完后才知道它并不需要导入如何jar包、配置web.xml或安装什么软件，只需通过DAO接口实现DAO对象的CUDR操作。

每个数据表都定义一个DAO接口或类实现，实现对此表的读写操作。换句话说，就是在域名.项目.模块.dao文件夹下创建个DAO类即可。

例如“package com.neusoft.dao;”

四. Jsp代码

然后是WebRoot文件夹下的jsp代码。其中index.jsp如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title>
        <meta http-equiv="pragma" content="no-cache">
        <meta http-equiv="cache-control" content="no-cache">
        <meta http-equiv="expires" content="0">
        <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
        <meta http-equiv="description" content="This is my page">
        <!--
        <link rel="stylesheet" type="text/css" href="styles.css">
        -->
  </head>

  <body>
    This is my JSP page. <br>
    <A href="student.jsp">JDBC操作</A>
  </body>
</html>
```

然后点击JDBC操作跳转到student.jsp操作，代码如下：涉及EL和JSTL。

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<jsp:directive.page import="DAO.StudentDAO"/>
```

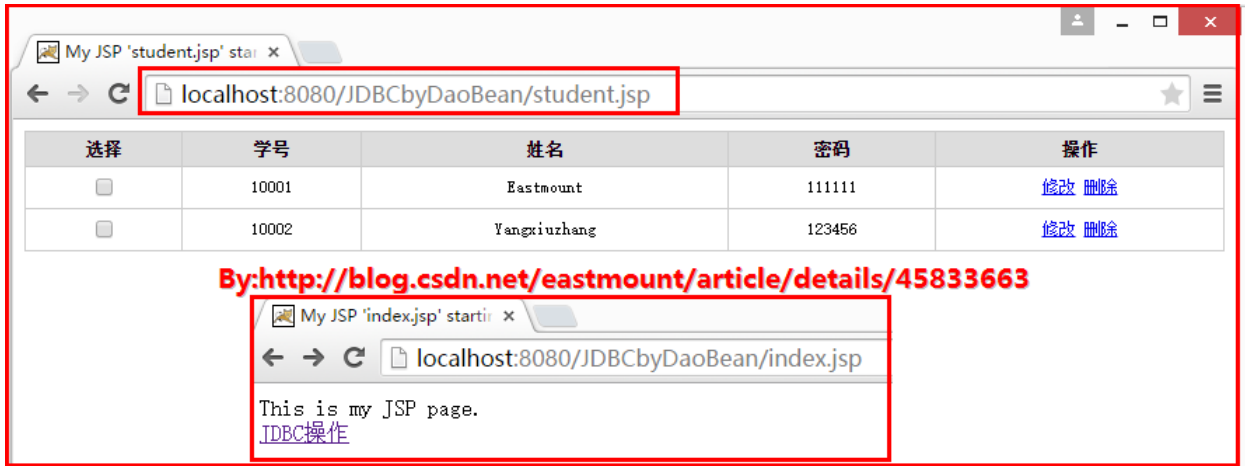
```
<jsp:directive.page import="java.util.List"/>
                                        <%
        List studentList = StudentDAO.listStudents();
        request.setAttribute("studentList", studentList);
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'student.jsp' starting page</title>
        <meta http-equiv="pragma" content="no-cache">
        <meta http-equiv="cache-control" content="no-cache">
        <meta http-equiv="expires" content="0">
        <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
        <meta http-equiv="description" content="This is my page">
        <style type="text/css">
                body, td, th, input {font-size:12px; text-align:center; }
        </style>
  </head>

  <body>
    <form action="operateStudent.jsp" method=get>
                        <table bgcolor="#CCCCCC" cellspacing=1 cellpadding=5 width=100%>
                                <tr bgcolor=#DDDDDD>
                                        <th>选择</th>
                                        <th>学号</th>
                                        <th>姓名</th>
                                        <th>密码</th>
                                        <th>操作</th>
                                </tr>

                                <c:forEach items="${studentList}" var="stu">
                                        <tr bgcolor="#FFFFFF">
                                                <td><input type="checkbox" name="id" value="${stu.id}"
/></td>
                                                <td>${stu.id}</td>
                                                <td>${stu.name}</td>
                                                <td>${stu.password}</td>
                                                <td>
                                                        <a href="addEmployee.jsp?action=edit&id=${stu.id}">修
改</a>
                                                        <a href="addEmployee.jsp?action=del&id=${stu.id}"
                                                                onclick="return confirm('确定删除?')">删除</a>
                                                </td>                                                </tr>
                                </c:forEach>                            </table>
                </form>
  </body>
</html>
```

文章运行结果如下图所示:



最后总结下，文章主要讲述了如何通过DAO和Java Bean实现Java数据库操作、界面显示分离的操作；同样的道理，实现修改、删除、插入方法类似，后面可能会讲述。该方法主要是通过上一篇自己的体会，找到的解决办法。最后希望文章对你有所帮助，如果有错误或不足之处，还请海涵~

最重要的一个问题，在这过程中你可能会遇到以下两个错误：(困扰我4小时)

Servlet.service() for servlet [jsp] in context with path

javax.el.PropertyNotFoundException: Property 'id' not found on type bean.Student

其解决方案参考：http://blog.csdn.net/eastmount/article/details/45835481

（By:Eastmount 2015-5-19 凌晨2点 http://blog.csdn.net/eastmount/）