**实训3 JDBC**

**第一部分：**

**实现目的：了解JDBC 连接程序的编写**

**作业要求：**

1. **安装Mysql数据库，新建数据库表Users，字段为**

**id（关键字，自增，int）**

**name（String）**

**age（int）**

**birthday（date）默认1994.5.6**

**money（int）**

1. **新建DBConfig.Properties文件，参数为mysql数据库的驱动、url、用户名、密码。**
2. **新建数据库连接类DBUtil.java**

**实现函数1，Connection getConnection（）；直接连接数据库。**

**实现函数2，Connection getConnection（String，String，String）；通过参数连接数据库。**

**实现函数3，Connection openConnection（），通过读取配置文件连接数据库。**

1. **新建类DBUtilTest.java,完成最基本的对DBUtil函数的测试例子。**
2. **在DBUtil.java中**

**实现函数1， void List （）；显示数据库表信息到控制台。**

**实现函数2，void add（UserBean usr）；把user插入数据库表User中，其中sql语句采用预设置参数的形式（PrepareStatement）。**

**实现函数3，void change（int uid1，int uid2，int money）；把用户1的钱，转账到用户2的手中，其中需要使用事务处理，保证两个动作完整性提交。**

参考例子：

1 新建文件DBConfig.Properties，放在src根目录下

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://127.0.0.1:3306/test

username=root

pwd=root

2 新建类DBUtil.java

**public** Connection getConnection(){

**try**{

Class.*forName*("com.mysql.jdbc.Driver");

**return** DriverManager.*getConnection*("jdbc:mysql://localhost:3306/test","root","root");

}**catch**(ClassNotFoundException e){

e.printStackTrace();

}**catch**(SQLException e){

e.printStackTrace();

}

**return** **null**;

}

**public** Connection getConnection(String url,String username,String pwd){

**try**{

Class.*forName*("com.mysql.jdc.Driver");

**return** DriverManager.*getConnection*(url,username,pwd);

} **catch**(ClassNotFoundException e){

e.printStackTrace();

}**catch**(SQLException e){

e.printStackTrace();

}

**return** **null**;

}

**public** Connection openConnection(){

Properties prop=**new** Properties();

String driver=**null**;

String username=**null**;

String url=**null**;

String pwd=**null**;

**try**{

prop.load(**this**.getClass().getClassLoader().getResourceAsStream("DBConfig.properties"));

driver=prop.getProperty("driver");

url=prop.getProperty("url");

username=prop.getProperty("username");

pwd=prop.getProperty("pwd");

Class.*forName*(driver);

**return** DriverManager.*getConnection*(url,username,pwd);

}**catch**(ClassNotFoundException e){

e.printStackTrace();

}**catch**(SQLException e){

e.printStackTrace();

}**catch**(Exception e){

e.printStackTrace();

}

**return** **null**;

}

**public** **void** closeConn(Connection conn){

**try** {

conn.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*out*.println("数据库关闭异常");

e.printStackTrace();

}

}

}

3 新建junit类

**public** **class** DBUtilTest {

**private** DBUtil dbu;

@Before

**public** **void** setUp() **throws** Exception {

dbu=**new** DBUtil();

}

@After

**public** **void** tearDown() **throws** Exception {

}

@Test

**public** **void** testAddpp() {

int result=dbu.addpp(4, 5);

Assert.assertEquals(8, result);

Assert.fail("Not yet implemented");

//fail("Not yet implemented");

}

}

4 新建 list方法

**public** **void** list(){

DBUtil util=**new** DBUtil();

Connection conn=util.getConnection();

String sql = "select id ,username,password from user";

**try**{

Statement stmt=conn.createStatement();

ResultSet rs= (ResultSet) stmt.executeQuery(sql);

**while**(rs.next()){

**int** id=rs.getInt("id");

String username=rs.getString(2);

String pwd=rs.getString("password");

System.*out*.println("id="+id+",username="+username);

}

}**catch**(SQLException e){

e.printStackTrace();

}

}

5 add方法

**public** **void** add(UserBean usr)

{

**try**{

DBUtil util = **new** DBUtil();

Connection conn=util.openConnection();

String sql="insert into users（name，age，money） values（？，？，？）";

Stmt.setString(1,usr.getName());

Stmt.setInt(1,usr.getAge());

Stmt.setInt(1,usr.getMoney());

stmt.executeUpdate(sql);

}**catch**(SQLException e){

e.printStackTrace();

**try**{

conn.rollback();

}**catch**(SQLException el){

el.printStackTrace();

}

}**finally**{

util.closeConn(conn);

}

}

6 change方法

**public** **void** change(int user1，int user2，int money)

{

DBUtil util = **new** DBUtil();

Connection conn=util.openConnection();

String sql1="update user set money=money- "+money;

String sql2="update user set money=money+ "+money;

**try**{

conn.setAutoCommit(**false**);//true

Statement stmt=conn.createStatement();

stmt.executeUpdate(sql1);

stmt.executeUpdate(sql2);

conn.commit();

}**catch**(SQLException e){

e.printStackTrace();

**try**{

conn.rollback();

}**catch**(SQLException el){

el.printStackTrace();

}

}**finally**{

util.closeConn(conn);

}

}

第二部分

实验目的：了解jdbc代码的基本编写方法，掌握Web开发项目中jdbc的实际应用。

作业要求：

在Web中通过Jdbc操作数据库中的表格数据。

* + 1. 在mysql数据库软件中创建一个数据库mydatebase
    2. 在数据库中创建一个表students（可以通过create命令，或者图形界面创建）

CREATE TABLE IF NOT EXISTS `students` (

`id` int(11) NOT NULL auto\_increment,

`link` varchar(255) default NULL,

`pwd` varchar(255) default NULL,

`roles` varchar(255) default NULL,

`uid` varchar(255) default NULL,

`username` varchar(255) default NULL,

PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO\_INCREMENT=3 ;

* + 1. 创建jsp页面list.jsp ,功能:从数据库中读取students表的内容，并显示在网页上的表格中；另外在表格之前使用一个表单，表单中包含若干个text控件（text数跟students的字段数相同），以及一个submit按钮，可以模糊查询表中的内容，并且更新页面；在list.jsp页面的最下面，创建一个链接（链接到new.jsp），链接文字是“增加学生”。
    2. 创建jsp页面new.jsp，内容是一个表单，表单项是students的字段的内容,以及submit和reset按钮。当点击submit按钮以后，请用js做简单的表单验证（包括字段非空验证），然后提交到页面newConfirm.jsp处理。
    3. 创建页面newConfirm.jsp，功能是把前面表单的内容存储到数据库的表students中（不能违反数据库表的主键约束），然后显示提示信息“新建学生成功”，并且5秒以后跳转返回到页面list.jsp。

注意：

* 1. 使用到的jsp对象主要包括out、request、response
  2. 代码编写主要包括jdbc数据库表的增加（executeUpdate）与查询(executeQuery)。
  3. mysql数据库的安装可以上外网下载5.0以上的版本。
  4. 代码架构符合DAO的命名规则：

目录结构包括dao（StudentDAO负责表的基本增删改查）,bean(StudentBean, 学生表的javaBean)，utility（DBUtility数据的连接、关闭等基本功能类，也可以包含其他工具类），test（StudentDaoTest，基本的junit测试类）

* 1. 完成上述任务后，有时间的继续完成在jsp页面上对数据库表student的其他编辑功能（增删改查）：