一、 JDBC 简介

1-1 JDBC 简介

二、 JDBC 使用详解

2-1 JDBC 项目效果展示

2-2 JDBC实战---打通数据库

·加载驱动程序：Class.forName(driveClass)

加载Mysql驱动：Class.forName("com.mysql.jdbc.Driver");

加载Oracle驱动：Class.forName("oracle.jdbc.driver.OracleDriver");

·获得数据库连接：

DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/imooc","root","root");

·创建Statement对象：conn.createStatement();

·Class.forName(xxx.xx.xx)返回的是一个类

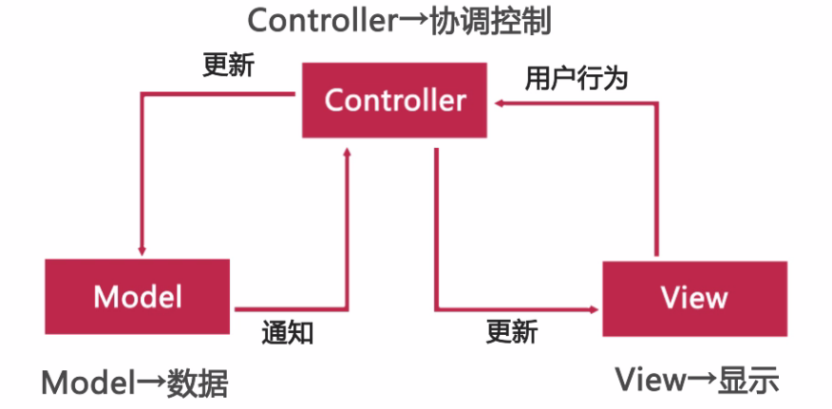
Class.forName(xxx.xx.xx)的作用是要求JVM查找并加载指定的类，也就是说JVM会执行该类的静态代码段

2-3 JDBC实战---搭建模型层 Ⅰ

·详设

·采用MVC三层架构





2-4 JDBC实战---搭建模型层 Ⅱ

2-5 JDBC实战---搭建模型层 Ⅲ

2-6 JDBC实战---搭建模型层 Ⅳ

2-7 JDBC实战---搭建控制层

2-8 JDBC实战---搭建视图层（上）

·流程

·程序启动后，一直保持在运行状态

·循环接收控制台的输入参数

·调用Action（控制层）响应，并将返回结果展示在控制台中

·问题点：

·循环接收参数

·某个功能的保持

2-9 JDBC实战---搭建视图层（下）

三、 整理总结

3-1 课程总结

·JDBC的基本概念

·连接数据库的桥梁

·由java语言编写的类和接口组成

·可以为多种数据库提供统一的访问

·采用MVC三层架构 DB->Model->Control->View

·JDBC各种连接方式的对比

·JDBC+ODBC桥的方式

特点：需要数据库的ODBC驱动，仅适用于微软的系统

·JDBC+厂商API的形式

特点：厂商API一般使用C编写

·JDBC+厂商Database Connection Server+DataBase的形式

特点：在JAVA与DATABASE之间架起了一台专门用于数据库连接的服务器一般由 数据库厂商提供（）

·JDBC+DATABASE的连接方式

特点：这使得Application与数据库分开，开发者只需关心内部逻辑的实现而不注 重数据库连接的具体实现

·PrepareStatement优点

·效率高：在缓存区中可以发现预编译的命令，不必重复编译。

·代码可读性和可维护性更好

·安全性更好：能够预防注入攻击

Connection conn == DBUtil.getConnection();

String sql;

·Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql);

·PreparedStatement ptmt = conn.prepareStatement(sql);

Result rs = ptmt.executeQuery();

·PreparedStatement ptmt = conn.preparecall(sql);

//设置变量值

ptmt.execute();

四、存储过程的调用

4-1 调用无参的存储过程

·存储过程代码：

Create PROCEDURE imooc\_db.sp\_select\_nofilter()

BEGIN

select \* from imooc\_goddess;

END;

·调用代码：

Connection conn = DBUtil.getConnection();

CallableStatement c = conn.prepareCall("call sp\_select\_nofilter");

c.execute();

ResultSet rs = c.getResultSet();

4-2调用带输入参数的存储过程

·存储过程代码：

CREATE PROCEDURE sp\_select\_filter(IN sp\_name VARCHAR(20))

BEGIN

IF sp\_name IS NULL OR sp\_name = '' then

SELECT \* FROM imooc\_goddess;

ELSE

IF length(sp\_name)=11 AND substring(sp\_name,1,1)=1 THEN

SELECT \* FROM imooc\_goddess WHERE mobile=sp\_name;

ELSE

SELECT \* FROM imooc\_goddess WHERE user\_name LIKE concat('%',sp\_name,'%');

END IF;

END IF

END

·调用代码：

Connection conn = DBUtil.getConnection();

CallableStatement cs = conn.prepareCall("call sp\_select\_nofilter**(?)**");

**cs.setString(1, sp\_name);**

cs.execute();

ResultSet rs = cs.getResultSet();

4-3调用带输出参数的存储过程

·存储过程代码：

CREATE PROCEDURE sp\_select\_count(OUT count INT(10))

BEGIN

SELECT count(\*) INTO count FROM imooc\_goddess;

END

·调用代码：

**Integer count = 0;**

Connection conn = DBUtil.getConnection();

CallableStatement cs = conn.prepareCall("call sp\_select\_nofilter**(?)**");

**cs.registerOutParameter(1, Types.INTEGER);**

cs.execute();

**count = cs.getInt(1);**

1. 事务管理

5-1 JDBC事务的概念和特点

·概念

·事务（TRANSACTION）是作为单个逻辑工作单元执行的一系列操作。

·这些操作作为一个整体一起向系统提交，要么都执行，要么都不执行。

·特点

·原子性（Atomicity）：事务是一个完整的操作。

·一致性（Consistency）：当事务完成时，数据必须处于一致状态。

·隔离性（Isolation）：对数据进行修改的所有并发事务是彼此隔离的。

·永久性（Durability）：事务完成后，他对数据库的修改被永久保持。

5-2 JDBC实现事务管理

·我们通过提交commit()或是回退rollback()来管理事务的操作。

·事务默认是自动提交。

·可通过调用setAutoCommit(false)来禁止自动提交。

5-2 通过代码实现事务管理

·conn.setAutoCommit(false);//禁止自动提交

try{

...

conn.commit();//事务执行完毕手动提交

} catch(Exception e) {

conn.rollback();//事务中间环节出错，回滚事务

}

1. 连接池

6-1 JDBC数据库连接池简介

·背景

·数据库连接是一种重要的资源。

·常用的开源数据库连接池

·dbcp

·c3p0

6-2 配置并测试dbcp连接

·导入相关jar包

·commons-dbcp2-2.1.1.jar

·commons-pool2-2.4.2.jar

·commons-logging-1.2.jar

·在项目根目录增加配置文件或直接在代码中设置



·dbcp.properties

driverClassName=com.mysql.jdbc.Driver

url=jdbc\:mysql\:///imooc?useSSL\=false

username=root

password=54pengcheng

maxActive=30

maxIdle=10

maxWait=1000

initialSize=5

minIdle=1

removeAbandoned=true

removeAbandonedTimeout=180

·代码中配置

private static void initDS(String driverClassName, String username, String password, String url) {

BasicDataSource ds = new BasicDataSource();

ds.setDriverClassName(driverClassName);

ds.setUrl(url);

ds.setUsername(username);

ds.setPassword(password);

//.......

DS = ds;

}

·读取配置文件来配置

private static void initDbcp() {

Properties pops = new Properties();

try {

pops.load(Object.class.getResourceAsStream(configFile));

DS = BasicDataSourceFactory.createDataSource(pops);

System.out.println(DS == null);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

6-3 配置并测试c3p0连接

·c3p0是一个开源的JDBC连接池，它实现了数据源和JNDI绑定，支持JDBC3和JDBC2的标准扩展。

·c3p0.properties

c3p0.driverClass=com.mysql.jdbc.Driver

c3p0.jdbcUrl=jdbc\:mysql\://localhost\:3306/imooc?characterEncoding\=utf8&useSSL\=false

c3p0.user=root

c3p0.password=54pengcheng

·C3P0Util

public class C3P0Util {

private static ComboPooledDataSource ds = new ComboPooledDataSource();

public static Connection getConnection() {

try{

return ds.getConnection();

} catch(SQLException e) {

throw new RuntimeException(e);

}

}

}

6-4 小结

