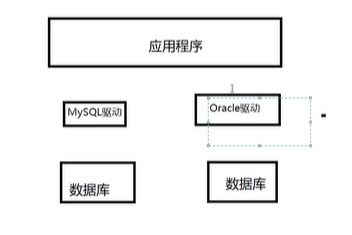
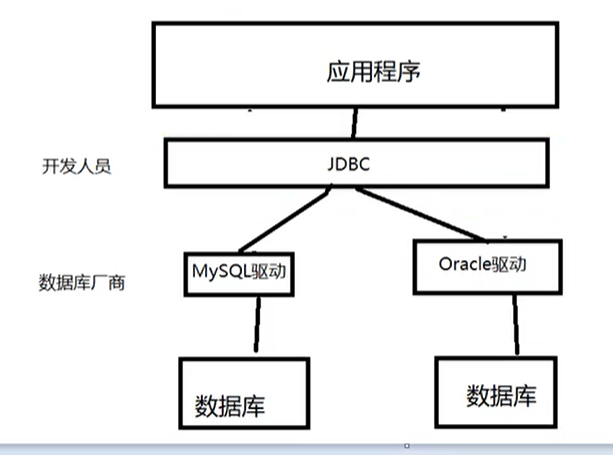
应用程序不能和数据库直接相连，不同的数据库有不同的驱动

****

为了简化开发人员对数据库的统一的操作，提供了一个JAVA操作数据库的规范， JDBC

这些规范的实现由具体厂商去做，我们只需要学习JDBC接口的操作



import java.sql.\*;  
  
public class jdbcStudy {  
 public static void main(String[] args) throws ClassNotFoundException, SQLException {  
 //1.加载JDBC驱动，固定语句，就是查找这个Class存不存在  
 Class.forName("com.mysql.jdbc.Driver");  
 //2.输入用户信息 //jdbc:mysql:就是类似于http的东西，  
 String url="jdbc:mysql://localhost:3306/jdbcstudy?useUnicode=true&characterEncoding=utf8&useSSL=false"; //url: jdbc:mysql://主机地址：端口号/数据库名?参数1&参数2&参数3  
 String username="root";  
 String password="123456";  
 //3.连接成功，创建一个数据库对象//也就是Connection的object，connection代表数据库  
 Connection connection= DriverManager.getConnection(url,username,password);  
 //4.执行sql的语句的对象  
 Statement statement=connection.createStatement();  
  
 //5.利用执行sql的语句的对象 来进行SQL query, 返回的结果是一个结果集，里面封装了我们查询出来的结果表格  
 String sql="SELECT \* FROM users";  
 ResultSet resultSet=statement.executeQuery(sql);  
  
 while(resultSet.next()){  
 System.out.println("id:"+resultSet.getObject("id"));  
 System.out.println("name:"+resultSet.getObject("NAME"));  
 }  
  
 resultSet.close();  
 statement.close();  
 connection.close();  
  
  
  
 }  
}

connection:数据库对象

statement:执行类，执行SQL语句的对象，**jdbc中的Statement对象用于向数据库发送SQL语句，完成对数据库的增删改查**

statement.execute();//可以执行任何sql

**statement.executeQuery();//查询操作 返回结果集ResultSet**



Select语句用executeQuery

**statement.executeUpdate();//更新 插入 删除都是用这个，返回int=受影响的行数**

****

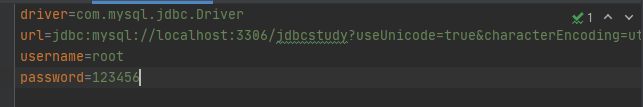
**update delete啥的都是他**

提取工具类：

上面的除了第5步，其他12346每次都是一样的，所以我们写一个工具类来省略他

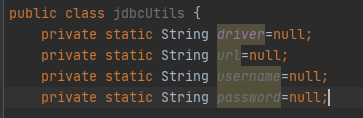
Properties，就是一些数据





怎么读取properties

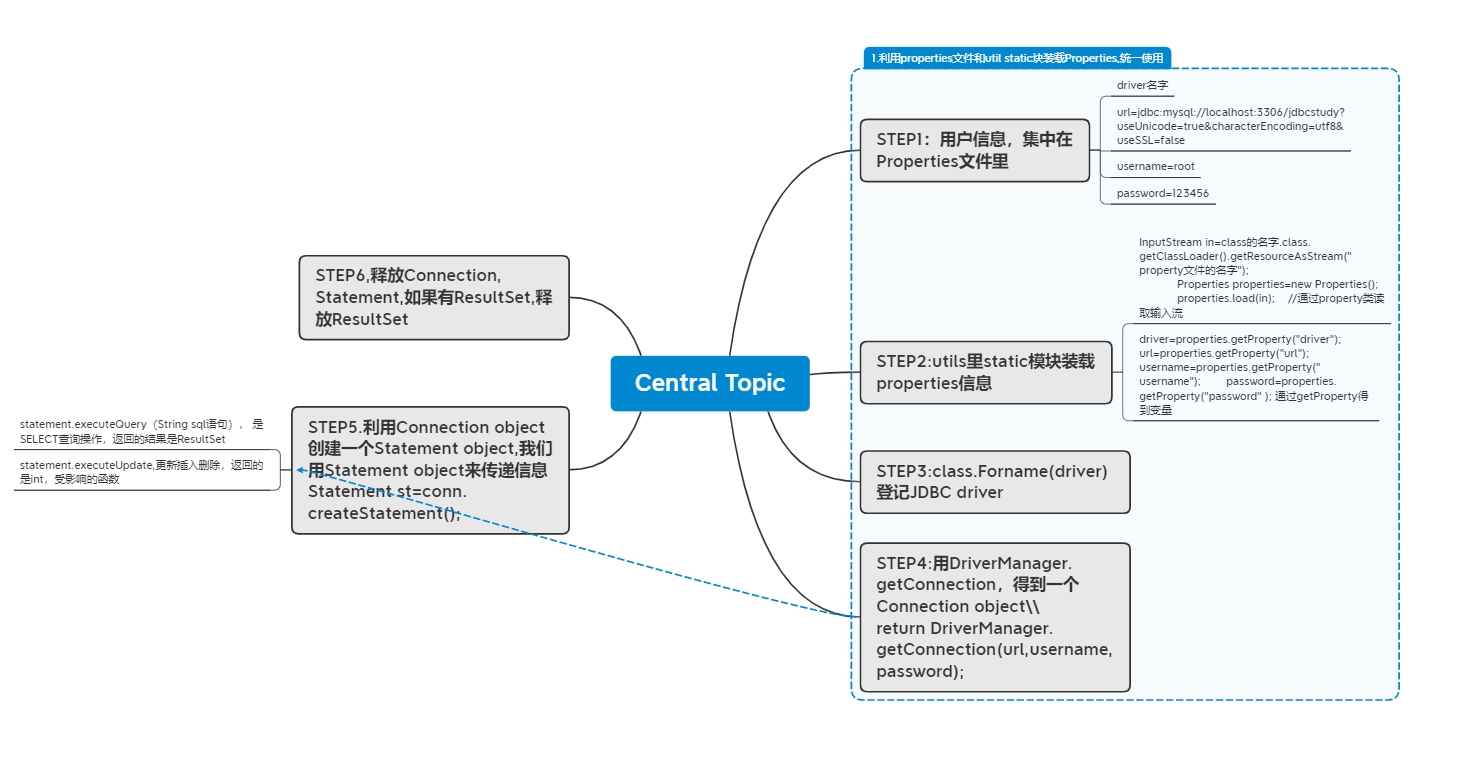
建立一个输入流，

创建我们对应想要的变量



Class.forName：加载指定的类，并初始化他， **类初始化操作就是执行一遍类的静态语句，**包括静态变量的声明还有静态代码块。

例如我们util里的static块，这个class被使用就会加载且仅加载一次



Prepared Statement

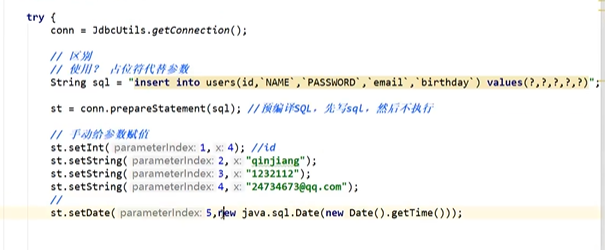
为了防止SQL注入问题，

很简单，原来利用conn.createStatement（）来创建Statement Object

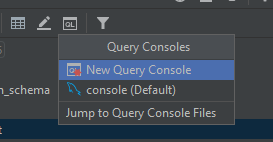
现在改成conn.prepareStatement

而sql部分提前写，原来value的地方改成占位符

然后一个一个设置问号



IDEA添加数据库，右边database,加号，输入用户密码，test connection,然后 schemas选择要用的table



右边的console可以输入命令

JDBC事物： transaction

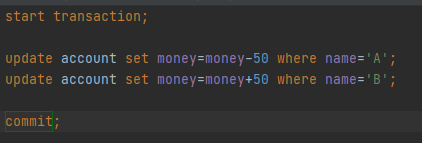
要么都成功，要么失败

流程：

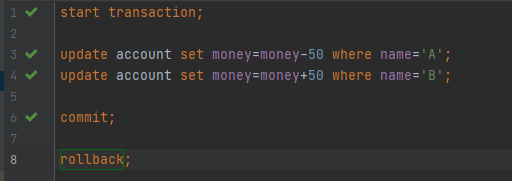
start transaction;

你要的几个操作

commit;



没有commit不会影响结果



commit 以后再rollback是没用的，因为持久性Durability

而先rollback再commit，那么数据库完全不会动，

我们怎么使用rollback呢

public class test {  
  
 @Test //jUnit装了以后，直接加个test,不用main也能测试  
 public void test() {  
 String url="jdbc:mysql://localhost:3306/jdbcstudy?useUnicode=true&character-Encoding=utf-8&useSSL=false";  
 String username="root";  
 String password="123456";  
 Connection connection=null;  
 try{  
 Class.forName("com.mysql.jdbc.Driver");  
  
 connection= DriverManager.getConnection(url,username,password);  
 connection.setAutoCommit(false); //setAutoCommit(false)代表Start transaction  
  
 String sql1="update account set money=money-50 where name='A'";  
 connection.prepareStatement(sql1).executeUpdate();  
  
 String sql2="update account set money=money+50 where name='B'";  
 connection.prepareStatement(sql2).executeUpdate();  
  
 connection.commit();  
 System.out.println("success");  
 } catch (Exception e){  
 try{ //如果中间出现了错误，我们直接进行一个rollback  
 connection.rollback();  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 e.printStackTrace();  
 }finally{  
 try {  
 connection.close();  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
  
  
 }  
}

ACID原则：保证数据的安全

原子性（Atomicity）

表示组成一个事务的多个数据库操作是一个不可分割的原子单元，只有所有的操作执行成功，整个事务才提交。事务中的任何一个数据库操作失败，已经执行的任何操作都必须被撤销，让数据库返回初始状态。

一致性（Consistency）

一致性是指在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏，即数据库事务不能破坏关系数据的完整性及业务逻辑上的一致性。

 隔离性（Isolation）

主要用于实现并发控制，隔离能够确保并发执行的事务按顺序一个接一个地执行。通过隔离，一个未完成事务不会影响另外一个未完成事务。

持久性（Durability）

一旦一个事务被提交，它应该持久保存，即使在事务提交后，数据库马上崩溃，在数据库重启时，也必须保证能够通过某种机制恢复数据。