**JDBC小结**

JDBC，Java中用来和数据库进行关联操作。

一般而言，JDBC都是屏蔽了各个数据库之间的差异，实现Java本身“一次编译，多处运行”的目标。使用这技术访问数据库时，操作顺序是：

1. 获取数据库驱动
2. 获取数据库链接Connection对象
3. 通过connection对象，创建statement或者preparestatement对象
4. 执行第statement等对象中的execute等等语句
5. 返回结果集

现在，以链接并操作MySQL为例，开始逐一细讲。

1. 获取数据库驱动

使用Class.forName(driverClass)语句进行加载，其中，driverClass对应的是各个数据库驱动对应的字符串，对于MySQL来说，它的值是：com.mysql.jdbc.Driver

1. 获取数据库链接Connection

一般通过DriverManager对象的getConnection()方法获取，语句为DriverManager.getConnection(String url,String user,String password) 括号内的参数分别是数据库的地址，用户名和密码，其中，MySQL数据库地址按照以下的格式jdbc:mysql://hostname:port/databasename

1. 创建statement等对象

语句为Connection对象.createStatement()方法。

1. 执行sql语句

Statement对象.execute()方法

1. 返回结果集

使用循环等方法返回

说的简单，实际上，在Java中使用这一技术时，都是按照固定的繁琐套路。以下为例子，代码如下

**package** jdbcDemo;

**import** java.io.FileInputStream;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.ResultSetMetaData;

**import** java.sql.Statement;

**import** java.util.Properties;

**public** **class** **Demo1** {

/\*\*

\* 使用String保存数据库驱动，数据库url，user，password等信息

\*/

**private** **String** url;

**private** **String** user;

**private** **String** pass;

**private** **String** driver;

/\*\*

\* initParam方法为链接数据库作初始化，确定url等信息。

\* **@param** initFile 指定文件作为url等等数据的来源，易于后面修改

\* **@throws** Exception

\*/

**public** **void** **initParam**(**String** initFile) **throws** **Exception**

{

**Properties** **props** = **new** Properties();

props.load(**new** FileInputStream(initFile));

url = props.getProperty("url");

user = props.getProperty("user");

pass = props.getProperty("pass");

driver = props.getProperty("driver");

}

/\*\*

\* 链接数据库，进行查询操作并返回输出结果

\* **@param** sql 是需要操作的sql语句

\* **@throws** Exception

\*/

**public** **void** **executeSql**(**String** sql) **throws** **Exception**

{

**Class**.*forName*(driver);//加载数据库驱动

**try**

(

Connection **conn** = **DriverManager**.*getConnection*(url,user,pass);//获得数据库链接

Statement **stmt** = conn.createStatement();//创建Statement对象，为操作数据库作准备

)

{

stmt.executeQuery(sql);//操作特定的数据库语句

ResultSet **rs** = stmt.getResultSet();//获取结果集

ResultSetMetaData **rsmd** = rs.getMetaData();//获取结果集的元数据，例如行数等等

**int** **columnCount** = rsmd.getColumnCount();

/\*\*

\* 循环输出结果

\*/

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

{

**for**(**int** **i**=0;i<columnCount;i++)

{

**System**.***out***.print(rs.getString(i+1)+"\t");

}

**System**.***out***.println("");

}

}

}

/\*\*

\* 测试，验证

\* **@param** args

\* **@throws** Exception

\*/

**public** **static** **void** **main**(**String**[] args) **throws** **Exception**

{

**Demo1** **d1** = **new** Demo1();

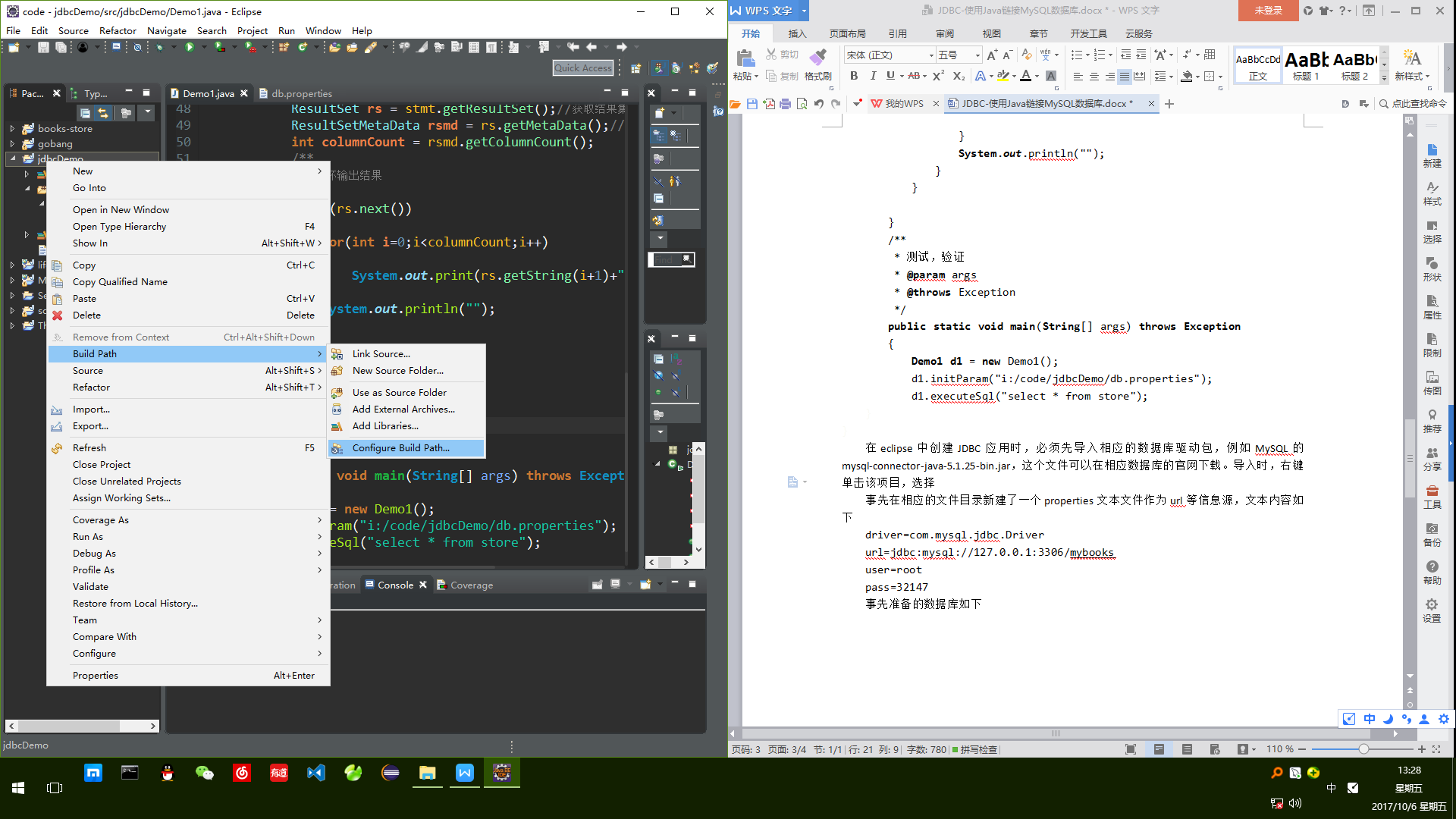
d1.initParam("i:/code/jdbcDemo/db.properties");

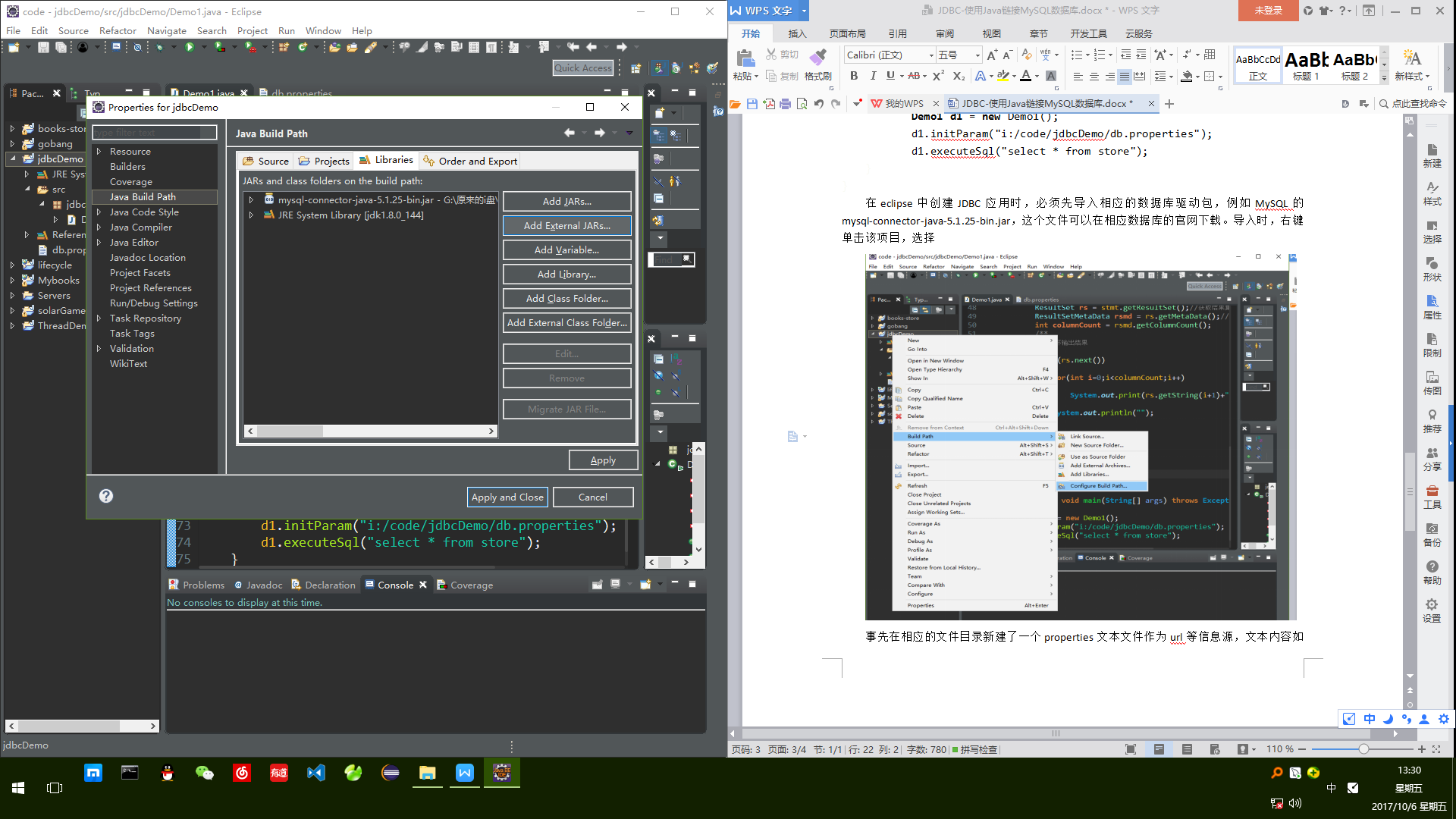
d1.executeSql("select \* from store");

}

}

在eclipse中创建JDBC应用时，必须先导入相应的数据库驱动包，例如MySQL的mysql-connector-java-5.1.25-bin.jar，这个文件可以在相应数据库的官网下载。导入时，右键单击该项目，选择





事先在相应的文件目录新建了一个db.properties文本文件作为url等信息源，文本文件内容如下

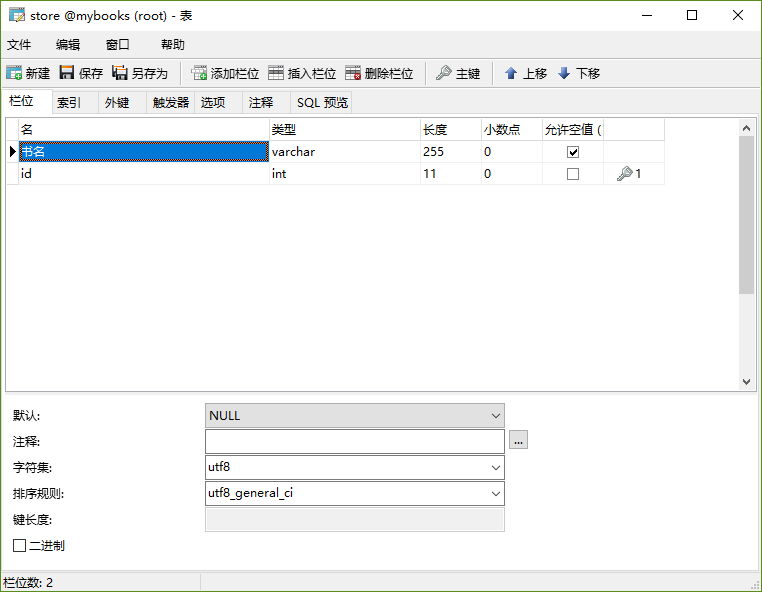
driver=com.mysql.jdbc.Driver

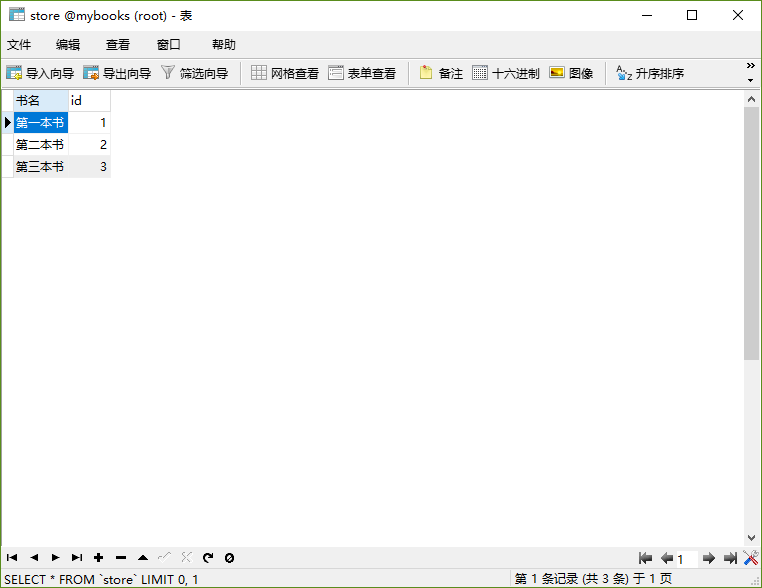
url=jdbc:mysql://127.0.0.1:3306/mybooks

user=root

pass=32147

事先准备的数据库如下





上面算是简单使用了一次JDBC，下面讲述一些细节问题。

1. 获取数据库链接等等操作，最好使用一个文本文档作为操作源，杜绝人为输错等错误，而且，这样也使得程序没有那么硬编码。
2. Statement等等对象使执行sql语句的接口。他有一些不同的执行方法，例如 ResultSet executeQuery返回的是对应的ResultSet对象，只能用于查询

Int executeUpdate ，可以执行DML语句， 即增删改查数据，并且返回被影响的数据行数。

Boolean execute方法，可以执行任何sql语句。

1. ResultSet对象通常使用其next()方法作为输出循环的判断条件。而ResultSet对象一 般通过statement对象的getResultSet()方法获取。

通过ResultSet对象的getMetaData()方法可以取出结果集的元数据，进一步操作结果集。

Java7之后，RowSet接口下，有一些并不需要一直连接数据库的结果集，例如CachedRowSet FilteredRowSet JoinRowSet WebRowSet等等，他们对应的实现对象，例如CachedRowSetImpl，默认是可滚动、可更新、可序列化的，很多时候作为JavaBean使用。

1. Statement与PreparedStatement

一般来说，都是使用PreparedStatement,因为statement可能会发生sql注入问题。

而且，在处理二进制长文件Blob类型数据，例如图片和音乐的时候，必须使用PreparedStatement。

在使用批量更新功能时，一般使用statement对象。

PreparedStatement可以使用“？”配位符。

在执行多条类似的语句时，PreparedStatement执行效率更高。

1. Java5之后，可以通过一系列方法操作结果集，例如直接到达结果集最后的last()方 法等等。
2. JDBC的事务支持

在JDBC中使用事务，必须先将自动提交功能设为false，conn.setAutoCommit(false)；后面如果确定要提交更改，则使用conn.commit()提交。

在批量更新时，整个更新被视为一个事务。

1. 使用conn.getMetaData()，可以获取数据库的元数据，分析数据库的各种信息。
2. 在实际的应用中，由于链接数据库时，是一种socket链接，非常耗费资源和时间， 所以，一般都使用连接池管理连接。