**什么是JDBC**

JDBC是java database connectivity的缩写，是SUN公司提供的一套操作数据库的标准规范。JDBC提供一些操作数据的API，开发者可以在java中使用这些API操作数据库，实现对表中数据的增删改查操作，**JDBC相当于java和数据库之间的一座桥梁**。

SUN公司制定了JDBC标准，各大数据库厂商会提供数据库驱动现这个标准，这样java才可以通过JDBC来操作实现了这个标准的数据库。如果将JDBC看做是接口的话，数据库厂商提供的数据库驱动就是这个接口的实现类。各个数据库厂商会将各自的数据库驱动打成jar包对外发布，开发者在使用时需要下载与当前数据库匹配的数据库驱动jar包。

**JDBC规范让Java程序和数据库驱动实现了松耦合，使切换不同的数据库变得更加简单。**

**JDBC连接是和数据库服务器建立的一个会话**

**JDBC的四个核心接口**

**DriverManager**：用于注册驱动并创建符合该驱动的数据库的连接。

**Connection**：表示与数据库创建的连接对象，即一个connection对应着一个会话，相当于在mysql workbench中打开了一个连接。

**Statement**: 操作数据库sql语句的对象，有个两个实现类：**Statement**和**PreparedStatement**（常用）。

**ResultSet**: 从数据库中查询的结果集。

**使用JDBC编程的步骤**

**注册驱动**

**建立连接**

**创建执行SQL的语句的对象**

**执行语句，返回结果**

**处理结果**

**释放资源**

**DriverManager**

**该类的主要作用就是创建连接，使用了反射的机制注册了驱动**，不同的数据库，在forName中的参数写法不同，MySQL的写法：com.mysql.jdbc.Driver。

**获取连接Connection时，调用了DriverManager类中的getConnection方法**

**Statement**

**该接口的作用是操作sql语句，并返回相应结果的对象。**

**常用的方法**

ResultSet executeQuery(String sql)：根据查询语句返回结果集。只能执行select语句。

int executeUpdate(String sql)：根据执行的DML（insert update delete）语句，返回受影响的行数。

boolean execute(String sql)：此方法可以执行任意sql语句。返回boolean值，表示是否返回ResultSet结果集。仅当执行select语句，且有返回结果时返回true, 其它语句都返回false;

**ResultSet**

**该接口的作用主要用来封装结果集。**

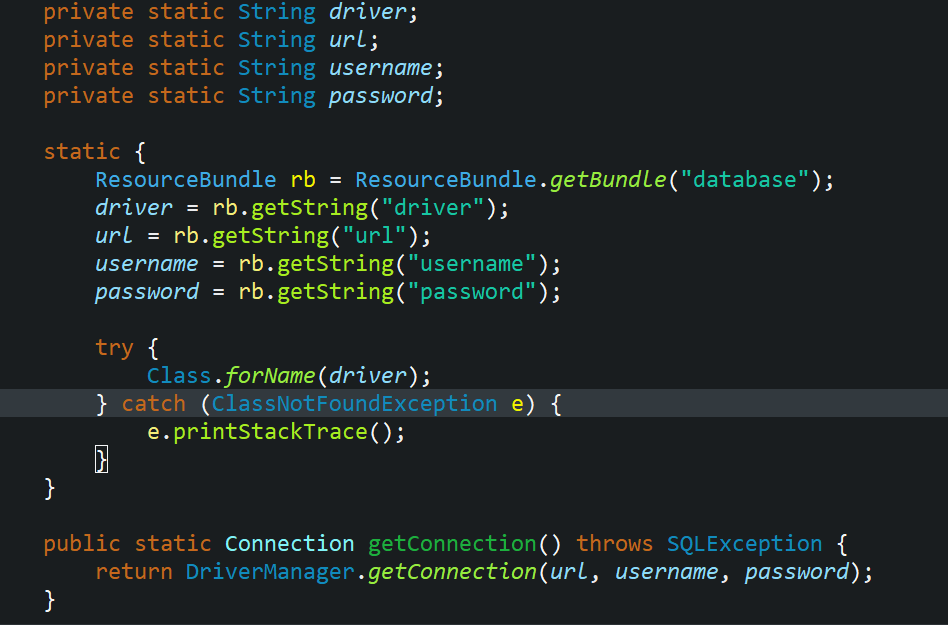
**常用的封装数据的方法：**

getObject(String ColomnName); 根据列名取值。

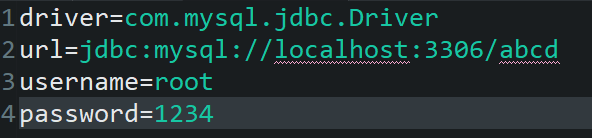
getObject(int columnIndex); 根据序号取值，索引从1开始（不建议使用，可读性差）

**在做查询操作时，可能会返回多条数据结果，此时可以定义一个JavaBean类，将数据封装到该JavaBean中，多条数据的话将此JavaBean放到集合中。**

**DBUtil.java**

****

**database.properties**

****

**查询**



**PreparedStatement**

PreparedStatement是一个接口，它继承了Statement，代表的是一个预编译的SQL语句

性能比Statement高，会把sql预编译

可以解决sql注入问题

**JDBC是如何实现Java程序和JDBC驱动的松耦合的？**

JDBC API使用Java的反射机制来实现Java程序和JDBC驱动的松耦合。所有操作都是通过JDBC接口完成的，而驱动只有在通过Class.forName反射机制来加载的时候才会出现。

**execute，executeQuery，executeUpdate的区别**

**execute执行增删改查操作**

**execute返回的结果是个boolean型**，当返回的是true的时候，表明有ResultSet结果集，通常是执行了select操作，当返回的是false时，通常是执行了insert、update、delete等操作。execute通常用于执行不明确的sql语句。尽量避免使用。

**executeQuery执行查询操作**

**executeQuery返回的是ResultSet结果集**，通常是执行了select操作。

**executeUpdate执行增删改操作**

**executeUpdate返回的是int型**，表明受影响的行数，通常是执行了insert、update、delete等操作。

**PreparedStatement中如何注入NULL值？**

可以使用它的setNull方法来把null值绑定到指定的变量上。setNull方法需要传入参数的索引以及SQL字段的类型

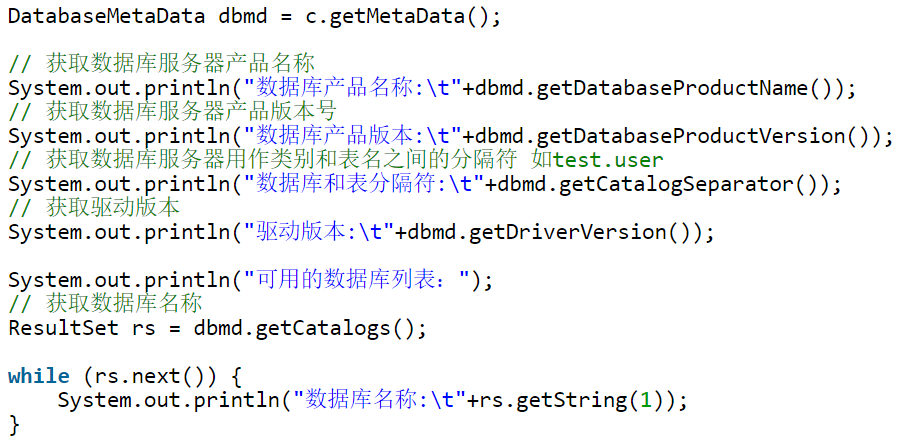
**getGeneratedKeys方法有什么用？**

**获取自动生成的主键的值**

ResultSet rs = ps.getGeneratedKeys();

**在Java程序中，如何获取数据库服务器的相关信息？**

使用DatabaseMetaData可以获取到服务器的信息。当和数据库的连接成功建立了之后，可以通过调用getMetaData()方法来获取数据库的元信息。DatabaseMetaData里面有很多方法，通过它们可以获取到数据库的产品名称，版本号，配置信息等。



**事务**

Connection类中提供了4个事务处理方法

setAutoCommit(Boolean autoCommit):设置是否自动提交事务,默认为自动提交,即为true,通过设置false禁止自动提交事务;

commit():提交事务;

rollback():回滚事务.

savepoint:保存点，savepoint不会结束当前事务，普通提交和回滚都会结束当前事务的

**PreparedStatement的缺点**

不能直接用它来执行in条件语句

**JDBC的ResultSet是什么?**

**在查询数据库后会返回一个ResultSet，它就像是查询结果集的一张数据表。**

**ResultSet对象维护了一个游标，指向当前的数据行。**开始的时候这个游标指向的是第一行。如果调用了ResultSet的next()方法游标会下移一行，如果没有更多的数据了，next()方法会返回false。可以在for循环中用它来遍历数据集。

**默认的ResultSet是不能更新的，游标也只能往下移。**也就是说你只能从第一行到最后一行遍历一遍。不过也可以创建可以回滚或者可更新的ResultSet

Statement stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

**当生成ResultSet的Statement对象要关闭或者重新执行或是获取下一个ResultSet的时候，ResultSet对象也会自动关闭。**

**可以通过ResultSet的getter方法，传入列名或者从1开始的序号来获取列数据。**

**有哪些不同的ResultSet？**

根据创建Statement时输入参数的不同，会对应不同类型的ResultSet。如果你看下Connection的方法，你会发现createStatement和prepareStatement方法重载了，以支持不同的ResultSet和并发类型。

**一共有三种ResultSet对象。**

ResultSet.TYPE\_FORWARD\_ONLY：**这是默认的类型，它的游标只能往下移。**

ResultSet.TYPE\_SCROLL\_INSENSITIVE：**游标可以上下移动，一旦它创建后，数据库里的数据再发生修改，对它来说是透明的。**

ResultSet.TYPE\_SCROLL\_SENSITIVE：**游标可以上下移动，如果生成后数据库还发生了修改操作，它是能够感知到的。**

**ResultSet有两种并发类型。**

ResultSet.CONCUR\_READ\_ONLY:**ResultSet是只读的，这是默认类型。**

ResultSet.CONCUR\_UPDATABLE:**我们可以使用ResultSet的更新方法来更新里面的数据。**

**java.util.Date和java.sql.Date有什么区别？**

java.util.Date包含日期和时间，而java.sql.Date只包含日期信息，而没有具体的时间信息。如果你想把时间信息存储在数据库里，可以考虑使用Timestamp或者DateTime字段

**JDBC的DataSource是什么，有什么好处**

DataSource即数据源，它是定义在javax.sql中的一个接口，跟DriverManager相比，它的功能要更强大。我们可以用它来创建数据库连接，当然驱动的实现类会实际去完成这个工作。除了能创建连接外，它还提供了如下的特性：

缓存PreparedStatement以便更快的执行

可以设置连接超时时间

提供日志记录的功能

ResultSet大小的最大阈值设置

通过JNDI的支持，可以为servlet容器提供连接池的功能

**SQLWarning是什么，在程序中如何获取SQLWarning？**

SQLWarning是SQLException的子类，通过Connection, Statement, Result的getWarnings方法都可以获取到它。**SQLWarning不会中断查询语句的执行，只是用来提示用户存在相关的警告信息。**

**Statement中的setFetchSize和setMaxRows方法有什么用处？**

**setMaxRows可以用来限制返回的数据集的行数。**当然通过SQL语句也可以实现这个功能。比如在MySQL中我们可以用LIMIT条件来设置返回结果的最大行数。

setFetchSize理解起来就有点费劲了，因为你得知道Statement和ResultSet是怎么工作的。当数据库在执行一条查询语句时，查询到的数据是在数据库的缓存中维护的。**ResultSet其实引用的是数据库中缓存的结果。**

假设我们有一条查询返回了100行数据，我们把fetchSize设置成了10，那么数据库驱动每次只会取10条数据，也就是说得取10次。当每条数据需要处理的时间比较长的时候并且返回数据又非常多的时候，这个可选的参数就变得非常有用了。

**我们可以通过Statement来设置fetchSize参数，不过它会被ResultSet对象设置进来的值所覆盖掉。**

**JDBC的RowSet是什么，有哪些不同的RowSet？**

RowSet用于存储查询的数据结果，和ResultSet相比，它更具灵活性。RowSet继承自ResultSet，因此ResultSet能干的，它们也能，而ResultSet做不到的，它们还是可以。RowSet接口定义在javax.sql包里。

提供了Java Bean的功能，可以通过settter和getter方法来设置和获取属性。RowSet使用了JavaBean的事件驱动模型，它可以给注册的组件发送事件通知，比如游标的移动，行的增删改，以及RowSet内容的修改等。

RowSet对象默认是可滚动，可更新的，因此如果数据库系统不支持ResultSet实现类似的功能，可以使用RowSet来实现。

**RowSet分为两大类：**

**连接型RowSet**——这类对象与数据库进行连接，和ResultSet很类似。JDBC接口只提供了一种连接型RowSet，javax.sql.rowset.JdbcRowSet，它的标准实现是com.sun.rowset.JdbcRowSetImpl。

**离线型RowSet**——这类对象不需要和数据库进行连接，因此它们更轻量级，更容易序列化。它们适用于在网络间传递数据。

**有四种不同的离线型RowSet的实现。**

**CachedRowSet**——可以通过他们获取连接，执行查询并读取ResultSet的数据到RowSet里。我们可以在离线时对数据进行维护和更新，然后重新连接到数据库里，并回写改动的数据。

**WebRowSet继承自CachedRowSet**——他可以读写XML文档。

**JoinRowSet继承自WebRowSet**——它不用连接数据库就可以执行SQL的join操作。

**FilteredRowSet继承自WebRowSet**——我们可以用它来设置过滤规则，这样只有选中的数据才可见。

**RowSet和ResultSet的区别是什么？**

RowSet继承自ResultSet，因此它有ResultSet的全部功能，同时它自己添加了些额外的特性。RowSet一个最大的好处是它可以是离线的，这样使得它更轻量级，同时便于在网络间进行传输。

**JDBC里的CLOB和BLOB数据类型分别代表什么？**

**CLOB是Character Large OBjects，字符大对象**，它是由单字节字符组成的字符串数据，有自己专门的代码页。这种数据类型适用于存储超长的文本信息，那些可能会超出标准的VARCHAR数据类型长度限制（上限是32KB）的文本。

**BLOB是Binary Larget OBject，二进制大对象**，由二进制数据组成，没有专门的代码页。它能用于存储超过VARBINARY限制（32KB）的二进制数据。这种数据类型适合存储图片，声音，图形，或者其它业务程序特定的数据。

**什么是JDBC的最佳实践？**

**数据库资源是非常昂贵的，用完了应该尽快关闭它。**Connection, Statement, ResultSet等JDBC对象都有close方法，调用它就好了。

养成在代码中显式关闭掉ResultSet，Statement，Connection的习惯，如果你用的是连接池的话，连接用完后会放回池里，但是没有关闭的ResultSet和Statement就会造成资源泄漏了。

**在finally块中关闭资源，保证即便出了异常也能正常关闭。**

**大量类似的查询应当使用批处理完成。**

**尽量使用PreparedStatement而不是Statement，以避免SQL注入，同时还能通过预编译和缓存机制提升执行的效率。**

**如果你要将大量数据读入到ResultSet中，应该合理的设置fetchSize以便提升性能。**

你用的数据库可能没有支持所有的隔离级别，用之前先仔细确认下。

数据库隔离级别越高性能越差，确保你的数据库连接设置的隔离级别是最优的。

如果在WEB程序中创建数据库连接，最好通过JNDI使用JDBC的数据源，这样可以对连接进行重用。

**如果你需要长时间对ResultSet进行操作的话，尽量使用离线的RowSet,** **这样可以释放数据库连接。**