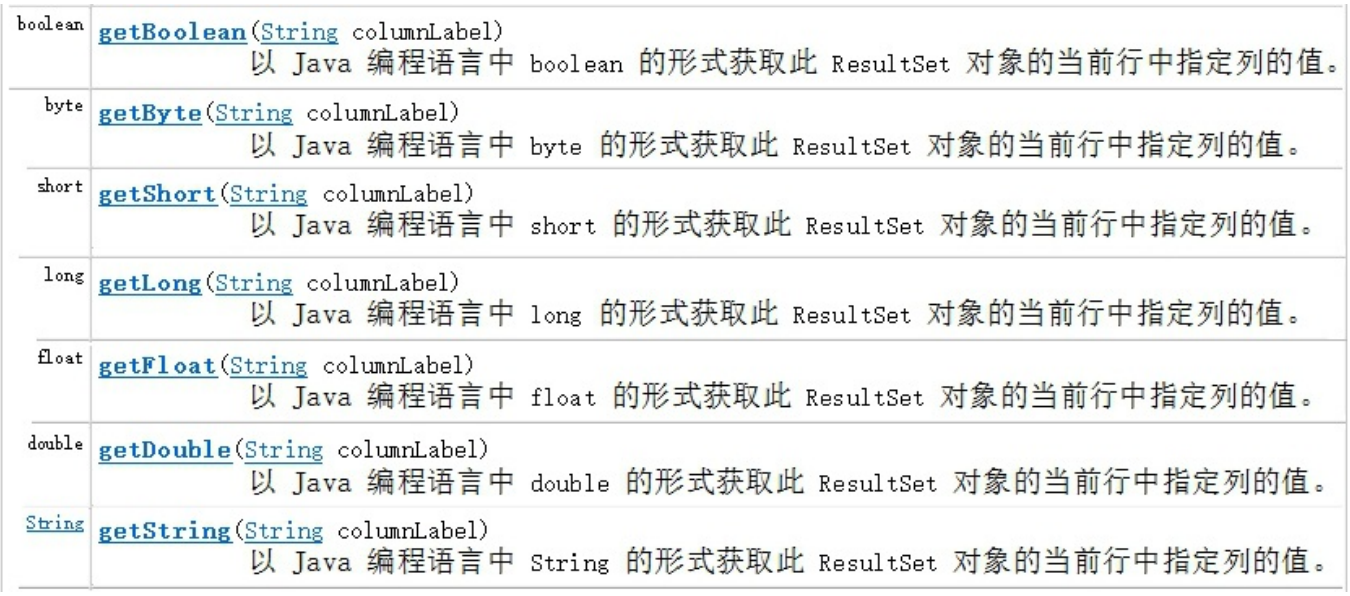
02 JDBC执行DQL 操作

JDBC

一：ResultSet接口

> 1、封装数据库查询的结果集，对结果集进行遍历，取出每一条记录

方法名	描述
boolean next()	1) 游标向下移动 1 行 2) 返回 boolean 类型，如果还有下一条记录，返回 true，否则返回 false
数据类型 getXxx()	1) 通过字段名，参数是 String 类型。返回不同的类型 2) 通过列号，参数是整数，从 1 开始。返回不同的类型



boolean	**getBoolean**(String columnLabel) 以 Java 编程语言中 boolean 的形式获取此 ResultSet 对象的当前行中指定列的值。
byte	**getByte**(String columnLabel) 以 Java 编程语言中 byte 的形式获取此 ResultSet 对象的当前行中指定列的值。
short	**getShort**(String columnLabel) 以 Java 编程语言中 short 的形式获取此 ResultSet 对象的当前行中指定列的值。
long	**getLong**(String columnLabel) 以 Java 编程语言中 long 的形式获取此 ResultSet 对象的当前行中指定列的值。
float	**getFloat**(String columnLabel) 以 Java 编程语言中 float 的形式获取此 ResultSet 对象的当前行中指定列的值。
double	**getDouble**(String columnLabel) 以 Java 编程语言中 double 的形式获取此 ResultSet 对象的当前行中指定列的值。
String	**getString**(String columnLabel) 以 Java 编程语言中 String 的形式获取此 ResultSet 对象的当前行中指定列的值。

> 2、常用数据类型转换表
> java.sql.Date、Time、Timestamp(时间戳)，三个共同父类是：java.util.Date

SQL 类型	Jdbc 对应方法	返回类型
BIT(1) bit(n)	getBoolean()	boolean
TINYINT	getByte()	byte
SMALLINT	getShort()	short
INT	getInt()	int
BIGINT	getLong()	long
CHAR,VARCHAR	getString()	String
Text(Clob) Blob	getClob getBlob()	Clob Blob
DATE	getDate()	java.sql.Date 只代表日期
TIME	getTime()	java.sql.Time 只表示时间
TIMESTAMP	getTimestamp()	java.sql.Timestamp 同时有日期和时间

1、查询一张表所有的数据

```
@Test
    public void test08() throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn =
DriverManager.getConnection("jdbc:mysql://192.168.0.115:3306/cost?chara
cterEncoding=utf8",
                "root", "root");
        // 3.定义sql
        String sql = "select * from t_cost";
        // 4.获取执行sql对象
        Statement stmt = conn.createStatement();
        // 5.执行sql
        ResultSet rs = stmt.executeQuery(sql);
        // 处理结果
        // 让游标向下移动一行
        rs.next();
        // 获取数据
        int id = rs.getInt("1");
        String costName = rs.getString("2");
        String costDesc = rs.getString("costDesc");
        String costMark = rs.getString("costMark");
```

```
            System.out.println(id + "-" + costName + "-" + costDesc+"-"+cost
Mark);
            //  让游标向下移动一行
            rs.next();
            // 获取数据
            // 获取数据
            int id1 = rs.getInt("costId");
            String costName1 = rs.getString("costName");
            String costDesc1 = rs.getString("costDesc");
            String costMark1 = rs.getString("costMark");
            System.out.println(id1 + "-" + costName1 + "-" + costDesc1+"-"+c
ostMark1);
            //  让游标向下移动一行
            rs.next();
            // 获取数据
            int id2 = rs.getInt("costId");
            String costName2 = rs.getString("costName");
            String costDesc2 = rs.getString("costDesc");
            String costMark2 = rs.getString("costMark");
            System.out.println(id2 + "-" + costName2 + "-" + costDesc2+"-"+c
ostMark2);
            //  让游标向下移动一行
            rs.next();
            // 获取数据

            stmt.close();
            conn.close();
        }
```

2、优化查询一张表所有的数据

```
    @Test
    public void test09() throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn =
DriverManager.getConnection("jdbc:mysql://192.168.0.115:3306/cost?chara
cterEncoding=utf8",
                "root", "root");
        // 3.定义sql
        String sql = "select * from t_cost";
        // 4.获取执行sql对象
        Statement stmt = conn.createStatement();
        // 5.执行sql
        ResultSet rs = stmt.executeQuery(sql);
```

```
            ArrayList arr=new ArrayList<Cost>();
            // 让游标向下移动一行
            while(rs.next()){
            // 获取数据
            Cost cost=  new Cost();
            cost.setCostId(rs.getInt("costId"));
            cost.setCostDesc(rs.getString("costDesc"));
            cost.setCostMark(rs.getString("costMark"));
            cost.setCostName(rs.getString("costName"));
            arr.add(cost);
            }
            System.out.println(arr);
            stmt.close();
            conn.close();
        }
```

二：数据库工具类 JdbcUtils

> 1、什么时候自己创建工具类
>
> 2、如果一个功能经常要用到，我们建议把这个功能做成一个工具类，可以在不同的地方重用

1、创建连接数据源信息

> 1、在src目录下获取jdbc.properties数据的资源信息

```
jdbc_url=jdbc:mysql://192.168.0.115:3306/cost?characterEncoding=utf8
jdbc_user=root
jdbc_password=root
jdbc_driver=com.mysql.jdbc.Driver
```

2、创建JDBC工具类

```
import java.io.FileReader;
import java.io.IOException;
import java.net.URL;
```

```java
import java.sql.*;
import java.util.Properties;

/**
 * JDBC工具类
 */
public class JDBCUtils {
    private static String jdbc_url;
    private static String jdbc_user;
    private static String jdbc_password;
    private static String jdbc_driver;
    /**
     * 文件的读取，只需要读取一次即可拿到这些值。使用静态代码块
     */
    static{
        //读取资源文件，获取值。

        try {
            //1. 创建Properties集合类。
            Properties pro = new Properties();

            //获取src路径下的文件的方式--->ClassLoader 类加载器
            ClassLoader classLoader = JDBCUtils.class.getClassLoader();
            URL res  = classLoader.getResource("jdbc.properties");
            String path = res.getPath();
            //2. 加载文件
           // pro.load(new FileReader("D:\\IdeaProjects\\itcast\\day04_jdbc\\src\\jdbc.properties"));
            pro.load(new FileReader(path));

            //3. 获取数据，赋值
            jdbc_url = pro.getProperty("jdbc_url");
            jdbc_user = pro.getProperty("jdbc_user");
            jdbc_password = pro.getProperty("jdbc_password");
            jdbc_driver = pro.getProperty("jdbc_driver");
            //4. 注册驱动
            Class.forName(jdbc_driver);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }


```

```
48.     /**
49.      * 获取连接
50.      * @return 连接对象
51.      */
52.     public static Connection getConnection() throws SQLException {
53.         return DriverManager.getConnection(jdbc_url, jdbc_user,
    jdbc_password);
54.     }
55.
56.     /**
57.      * 释放资源
58.      * @param stmt
59.      * @param conn
60.      * @throws SQLException
61.      */
62.     public static void close(Statement stmt,Connection conn) throws SQL
    Exception{
63.         if( stmt != null){
64.                 stmt.close();
65.         }
66.
67.         if( conn != null){
68.                 conn.close();
69.         }
70.     }
71.
72.
73.     /**
74.      * 释放资源
75.      * @param stmt
76.      * @param conn
77.      * @throws SQLException
78.      */
79.     public static void close(ResultSet rs,Statement stmt, Connection co
    nn) throws SQLException{
80.         if( rs != null){
81.                 rs.close();
82.         }
83.         if( stmt != null){
84.                 stmt.close();
85.         }
86.         if( conn != null){
87.                 conn.close();
88.         }
89.     }
```

```
90.   }
```