# JDBC(Java Database Connectivity)

Java数据库连接，（Java Database Connectivity，简称JDBC）是Java语言中用来规范客户端程序如何来访问数据库的应用程序接口，提供了诸如查询和更新数据库中数据的方法。JDBC也是Sun Microsystems的商标。我们通常说的JDBC是面向关系型数据库的。接口都有调用者和实现者，面向接口调用、面向接口写实现类，这都属于面向接口编程。

## 为什么SUN指定一套JDBC接口

因为每一个数据库的底层实现原理都不一样，每一个数据库产品都有自己独特的实现原理。

## JDBC编程六步

1、注册驱动（告诉java即将连接的是哪个品牌的数据库）

2、获取连接（表示JVM的进程和数据库进程之间的通道打开了，这属于进程间的通信，重量级的，使用完后一定要关闭）

3、获取数据库操作对象（专门执行sql语句的对象）

4、执行sql语句

5、处理查询结果集（只有是select才会有这一步）

6、释放资源（用完后一定要关闭）

```
/*
* JDBC编程六步
* */
public class JDBCTest01 {
    public static void main(String[] args) {
        List<User> list = new ArrayList<>();
        Connection connection = null;
        Statement statement = null;
        ResultSet rs = null;
        //注册驱动
        try {
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
            //获取连接
            //url:同一资源定位符（网络中某个资源的决定路径）
            connection =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/pzhu","root","1234");
            //获取数据库操作对象(statement专门执行sql)
            statement = connection.createStatement();
            String sql = "select * from user";
            //执行sql语句
            rs = statement.executeQuery(sql);
            while (rs.next()){
                //处理结果集
                User user = new User();
                user.setId(rs.getInt(1));
                user.setUsername(rs.getString(2));
                user.setPassword(rs.getString(3));
                user.setGender(rs.getString(4));
```

```
                user.setEmail(rs.getString(5));
                list.add(user);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            //释放资源(从小到大关闭)
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        System.out.println(list);

    }
}
```

或:

```
public class JDBCTest02 {
    public static void main(String[] args) {
        List<User> list = new ArrayList<>();
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet rs = null;
        try {
            //加载驱动
            Class.forName("com.mysql.jdbc.Driver");//反射机制，类加载会执行类中的静态
代码块
            //创建连接
            connection =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/pzhu","root","1234");
            //创建sql语句
            String sql = "select * from user";
            //预处理
            statement = connection.prepareStatement(sql);
            //执行并获取返回值
            rs = statement.executeQuery();
            //处理结果集
            while (rs.next()){
                User user = new User();
                user.setId(rs.getInt(1));
                user.setUsername(rs.getString(2));
                user.setPassword(rs.getString(3));
                user.setGender(rs.getString(4));
                user.setEmail(rs.getString(5));
                list.add(user);
            }
        } catch (ClassNotFoundException e) {
```

```
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            //释放资源
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        System.out.println(list);
    }
}
```

## 从属性资源文件中读取连接数据库信息

创建属性配置文件（需要在src目录下，如果是放在package下，则程序的filename应该package/myresource）

db.properties:

```
Driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/pzhu
user=root
password=1234
```

主程序:

```
/*
 * 将连接数据库的所有信息配置到配置文件中
 * */
public class JDBCPropertiesTest03 {
    public static void main(String[] args) {
        //使用资源绑定器绑定属性配置文件
        ResourceBundle resourceBundle = ResourceBundle.getBundle("db");
        String Driver = resourceBundle.getString("Driver");
        String url = resourceBundle.getString("url");
        String user = resourceBundle.getString("user");
        String password = resourceBundle.getString("password");
        List<User> list = new ArrayList<>();
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet rs = null;
        try {
            //加载驱动
            Class.forName(Driver);//反射机制，类加载会执行类中的静态代码块
```

```
            //创建连接
            connection = DriverManager.getConnection(url,user,password);
            //创建sql语句
            String sql = "select * from user";
            //预处理
            statement = connection.prepareStatement(sql);
            //执行并获取返回值
            rs = statement.executeQuery();
            //处理结果集
            while (rs.next()){
                User users = new User();
                users.setId(rs.getInt(1));
                users.setUsername(rs.getString(2));
                users.setPassword(rs.getString(3));
                users.setGender(rs.getString(4));
                users.setEmail(rs.getString(5));
                list.add(users);
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            //释放资源
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        System.out.println(list);
    }
}
```

输出：

## SQL注入

当输入用户名：aaaa

密码：aaaa' or '1' = '1

就会登陆成功

查询时为where password = 'aaaa' or '1' = '1'

这种现象被称为SQL注入（安全隐患）

**导致SQL注入的根本原因**

关键原因因为用户输入的信息正好完成了sql语句的拼接，其中的内容被当做sql语句中的关键字处理，导致原来的sql语句含义被扭曲。

**实现一个简单的用户登陆**:

```
public class JDBCLoginTest {
    public static void main(String[] args) throws SQLException {
        Map<String,String> loginUserInfo = loginWindow();
        Boolean login = Login(loginUserInfo);
        System.out.println(login?"登陆成功！":"登陆失败！");
    }

    private static Boolean Login(Map<String, String> loginUserInfo) throws
SQLException {
        Connection conn = null;
        Statement statement = null;
        ResultSet rs = null;
        Boolean loginStatus = null;
        try {
            //加载驱动
            Class.forName("com.mysql.jdbc.Driver");
            //创建连接
            conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/pzhu","root","1234");
            //创建数据库操作对象
            statement = conn.createStatement();
            //定义sql语句
            String sql = "select * from user where username = '"+
loginUserInfo.get("username") + "' and '" + loginUserInfo.get("password")+"'";
            //执行sql语句
            rs = statement.executeQuery(sql);

            if (rs.next()){
                loginStatus = true;
            }else
                loginStatus = false;
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                rs.close();
                statement.close();
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return loginStatus;
    }

    private static Map<String, String> loginWindow() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入用户名：");
        String username = scanner.nextLine();
```

```
        System.out.println("请输入密码：");
        String password = scanner.nextLine();
        //将输入的信息作为map值传入userInfo
        Map<String,String> userInfo = new HashMap<>();
        userInfo.put("username",username);
        userInfo.put("password",password);
        return userInfo;
    }
}
```

结果：

可以实现登陆

但是如果：

此时产生sql注入现象

## 解决SQL注入

只要用户提供的信息不参与sql语句的编译过程，问题就解决了

及时用户提供的信息中含有sql关键字，但是没有参与编译，不起作用

想要用户信息不参与sql语句的编译，就必须使用java.sal.PreparedStatement

PreparedStatement接口继承了java.sql.Statement

PreparedStatement是属于预编译的数据库操作对象

PreparedStatement的原理是，预先对SQL语句的框架进行编译，然后再给SQL语句传“值”。

**关键**

解决sql注入的关键是用户提供的信息即使有sql语句的关键字，但是这些关键字并没有参与编译。

```
public class JDBCLoginPreparedTest {
    public static void main(String[] args) throws SQLException {
        Map<String,String> loginUserInfo = loginWindow();
        Boolean login = Login(loginUserInfo);
        System.out.println(login?"登陆成功！":"登陆失败！");
    }

    private static Boolean Login(Map<String, String> loginUserInfo) throws
SQLException {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Boolean loginStatus = null;
        try {
            //加载驱动
            Class.forName("com.mysql.jdbc.Driver");
            //创建连接
            conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/pzhu","root","1234");
            //定义sql语句
            //?为占位符，占位符下标从1开始
            String sql = "select * from user where username = ? and password =
?";
```

```
            //预编译
            ps = conn.prepareStatement(sql);
            ps.setString(1,loginUserInfo.get("username"));
            ps.setString(2,loginUserInfo.get("password"));
            //执行sql语句
            rs = ps.executeQuery();

            if (rs.next()){
                loginStatus = true;
            }else
                loginStatus = false;
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                rs.close();
                ps.close();
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return loginStatus;
    }

    private static Map<String, String> loginWindow() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入用户名：");
        String username = scanner.nextLine();
        System.out.println("请输入密码：");
        String password = scanner.nextLine();
        //将输入的信息作为map值传入userInfo
        Map<String,String> userInfo = new HashMap<>();
        userInfo.put("username",username);
        userInfo.put("password",password);
        return userInfo;
    }
}
```

结果：

使用预编译时遇到条件查询或插入需要用`?`作为占位符，并从下标1开始赋值。

安装MYSQL:

第一步：去官网下载安装
第二步：先解压，然后在mysql下创建一个my.ini文件，更改my.ini文件里面的前两行安装目录，第二行加上\data，my.ini文件不能多一个符号或者少一个符号，在path（环境变量里面）加上mysql路径（;E:\mysql\mysql-8.0.25-winx64\bin）
(填写自己的mysql安装路径)
第三步：进入命令指示符（在bin目录下运行cmd），
输入mysqld --initialize-insecure --user=mysql,初始化数据库，并设置默认root为空，初始化完成后，在mysql根目录中会自动生成data文件
再输入mysqld -install,为windows安装mysql服务，默认服务名为mysql

出现service successfully installed.表示配置完成
启动数据库net start mysql,
输入mysql -u root -p ,不用输入密码直接回车
出现mysql>配置完成
输入(alter user user() identified by "密码";)
mysql退出 mysql>quit;
输入net stop mysql关闭数据库