

JDBC: 是JAVA数据库连接的通用接口, 可以连接MySQL, Oracle不同的数据库。

在使用时只需要找到相应的驱动jar包

附件: mysql-connector-java驱动jar包

开发流程:

1、创建Java项目, 或Web项目

★ 2、导入相应的jar包, 并引用jar包

★ 注意: 在Java项目中直接在项目里创建一个libs目录 将jar包放入 Build Path——>Add 一下即可

但是在Web项目中 jar包放入 不能放入项目里的libs目录 只能是放入

项目下WebRoot——> WEB-INF目录下的lib目录里 然后 Build Path——>Add 一下即可

3、代码编写实现

3.1、加载注册驱动

```
// 通过反射注册驱动
```

```
Class.forName("com.mysql.jdbc.Driver");
```

3.2、获得与数据库的连接 获取连接对象connection

```
// 获得连接 jdbc:mysql://localhost: 3306/数据库库名, 数据库用户名, 数据库密码
```

```
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + table_name, SqlNar
```

3.3、通过连接对象, 创建可以发送SQL语句的Statement对象

```
Statement statement = connection.createStatement();
```

3.4、通过执行SQL语句 (用于插入、删除、更新)

```
statement.executeUpdate(sql);
```

3.5、解析结果集 获取结果集对象ResultSet (用于查询)

```
ResultSet rs = statement.executeQuery(sql);;
```

条件查询解析结果

```
if(rs.next()){
    int id = rs.getInt("id");
    String username = rs.getString(2);
    String password = rs.getString(3);
    String nickname = rs.getString(4);
    System.out.println("id : "+id +", username:"+username+",password:"+password+", nickname : " + nickname);
}
```

查询所有解析结果集

```
while(rs.next()){
    int id = rs.getInt("id");
    String username = rs.getString(2);
    String password = rs.getString(3);
    String nickname = rs.getString(4);
    System.out.println("id : "+id +", username:"+username+",password:"+password+", nickname : " + nickname);
}
```

3.6、释放资源

```
/**
 * 关闭并释放资源
 */
private void closes(ResultSet RS){
    if(RS == null){
        if(statement != null && connection != null){
            try {
                connection.close();
                statement.close();
                connection = null;
                statement = null;
            } catch (Exception e) {
            }
        }
    }
    else{
        try {
            connection.close();
        }
    }
}
```

```

        statement.close();
        RS.close();
        connection = null;
        statement = null;
        RS = null;
    } catch (Exception e) {
    }
}
}

```

3、多条件查询 关键字 and（同时） or（或者）

select * from 表名 where 字段=值 and 字段>值;

如: select * from user22 where name="xm" and age>20;

查询name是xm并且age大于20的数据

注意在: java代码中 如果SQL数据类型是字符串（varchar类型）那么JAVA中一定要用 ‘’ 单引号 阔起来

如: String sqls ="select * from user where id=2018001 and name='"+liurong'";

11、SQL注入: 当用户登录时需要输入用户名和密码, SQL语句是接收了用户输入的用户名和密码, 使用了and多条件查询, 如:

```
String name = "liurong";
```

```
String spwd = "123456";
```

```
String sqls ="select * from user where name='"+name+"' and "+"pwd='"+spwd+'";
```

这个语句的意思是 查询表里 用户输入的 用户名和密码 都同时存在数据库中的一条数据

注入是这样的: 用户输入用户名通过拼装SQL语句传入SQL关键字参数, String name = "liurong or '1'='1'";

如此形成的新SQL就是:

```
select * from user where name=liurong or 1=1 and pwd=123456;
```

SQL语句执行这条新SQL时 是先判断 and 再判断 or and时必须左右同时符合条件才成立, 但是 or 是只要一边为符合就可以 条件就成立。

所以此时 实际就是 一个单条件的查询语句, 绕开了密码的比对。

解决SQL注入问题:

使用PreparedStatement是Statement的子接口它继承了extends Statement

通过连接对象获取一个对SQL语句的预处理对象

操作步骤:

1、确定要使用的SQL语句, 并定义好, 条件参数用 ? 代提

```
String sql ="select * from users where username=? and password=?";
```

2、使用与数据库的连接对象获取一个对SQL语句的预处理对象, 并传入SQL语句,

```
PreparedStatement stmt = connection.prepareStatement(sql);
```

3、获取要使用的参数

```
String username = "xiaoli001";
```

```
String password = "1234";
```

4、使用预处理对象的相应方法替换占位符

```
stmt.setString(1, username);
```

参数说明: 参数一 代表SQL语句第几个占位符,

```
stmt.setString(2, password);
```

参数二 要提换的参数

5、执行查询

```
ResultSet rs = stmt.executeQuery();
```

6、对结果解析

```
if(rs.next()){  
while(rs.next()){}
```

进行预编译处理，预编译的时候就确定了sql语句中关键字，那么后续在再传入关键字的时候就不会当作关键字去处理了，从而就解决了sql注入的问题。

```
String sql ="select * from users where username=? and password=?";
```

```
stmt = conn.prepareStatement(sql);
```

 获取PreparedStatement stmt对象

```
String username = "xiaoli' or '1'='1";  
String password = "1234";
```

```
//替换占位符  
stmt.setString(1, username);  
stmt.setString(2, password);
```

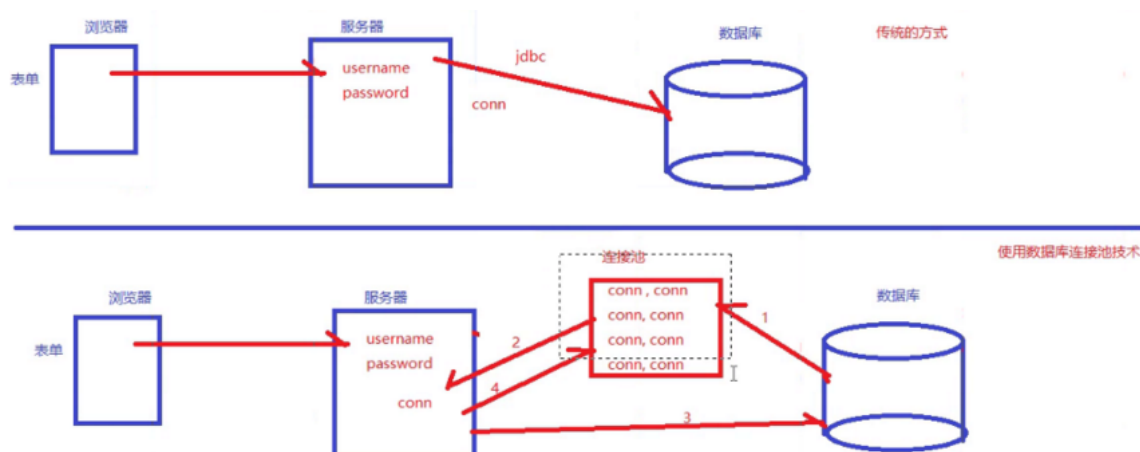
```
//执行查询  
rs = stmt.executeQuery();
```

 获取ResultSet rs对象

```
if(rs.next()){
```

数据库连接池——重要的数据库优化技术

图解



连接池原理要了解的 自定义的连接池：

一次性的批量的弄多个连接,保存起来,需要使用的时候从池子中去取,使用完了之后,在把连接放回到池子中. 而不是真正的去释放连接。

1、自定义类实现 implements DataSource接口 包 import javax.sql.DataSource;

2、实现相关方法:

2.1、通过集合生成容器,用来放连接 LinkedList<Connection> list = new LinkedList<Connection>();

用LinkedList集合的原因: LinkedList使用remove()方法时不需要给移除数据索引

2.2、自定义类构造中创建多个Connection对象,放到 池子

```
public MyDataSource() {  
    for (int i = 0; i < 10; i++) {  
        Connection conn = JdbcUtils.getConn();  
        list.add(conn);  
    }  
}
```

2.3、实现getConnection()方法: 从池子中获得连接

判断是否还有连接可用

```
if(list.size()==0){  
  
    for (int i = 0; i < 3; i++) {  
  
        Connection conn = JdbcUtils.getConn();  
  
        list.add(conn);  
  
    }  
  
}
```

将连接取出来, 给 一个出去.

```
//被装饰的 connection,  
  
Connection conn = list.remove();
```

返回一个返回一个 装饰 connection

```
MyConnection myconn = new MyConnection(conn);  
  
return myconn;
```

2.4、自定义实现Connection的装饰类

MyConnection自定义类实现 implements Connection

```
private Connection conn;  
  
public MyConnection(Connection conn) {  
  
    this.conn = conn;  
  
}
```

2.5、实现两个方法:

```
prepareStatement(String sql)  
  
return conn.prepareStatement(sql);
```

需要增强的close方法, 本来是要销毁的现在是放回到池子里

@Override

```
public void close() throws SQLException {  
  
    list.add(conn);  
  
}
```

使用: 创建MyDataSource()对象, 调用getConnection()方法返回一个Connection的子对象MyConnection

使用结束调用Connection的close方法将Connection连接放回到池子中去。

java中对方法加强的方式:

使用匿名内部类的形式

使用 装饰者模式 包装

使用 动态代理

★配置文件的使用方法:

1、在项目里创建配置文件

文件扩展名: *.properties

文件内容如下图:



```
1 driverClassName=com.mysql.jdbc.Driver
2 url=jdbc:mysql://localhost:3306/test001
3 username=root
4 password=123456
5 #bds.setDriverClassName("com.mysql.jdbc.Driver");
6 #bds.setUrl("jdbc:mysql:///day09_jdbc_dbutils");
7 #bds.setUsername("root");
8 #bds.setPassword("abc");
```

2、在需要的类中读取及调用

2.1需要读取 properties 配置文件

```
try {
```

```
    InputStream in = new FileInputStream("src/jdbc.properties"); 参数properties 配置文件所在位置
```

```
    //数据进入到 props 对象 中了
```

```
    Properties props = new Properties();
```

```
    props.load(in);
```

2.2获得每个 常量的值了 (如果需要将每个数据读出可使用下面方法)

```
    driverClass = props.getProperty("driverClassName");
```

```
    url = props.getProperty("url");
```

```
    username = props.getProperty("username");
```

```
    password = props.getProperty("password");
```

```
    in.close();
```

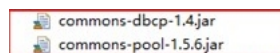
实际使用 开源 的连接池技术:

Dbcp

DBCP (DataBase Connection Pool) 数据库连接池, 是java数据库连接池的一种, 由Apache开发, 通过数据库连接池, 可以让程序自动管理数据库连接的释放和断开。

使用:

1、导入 dbcp的jar包



2、具体代码实现 (手动设置连接参数)

2.1、通过设置必要的连接的参数信息, 获取Connection连接对象

```
BasicDataSource bds = new BasicDataSource(); 获取数据库连接池
```

```
// 设置 必要的 连接的参数信息
```

```
bds.setDriverClassName("com.mysql.jdbc.Driver"); // 加载驱动
```

```
bds.setUrl("jdbc:mysql://localhost:3306/test001"); // 数据库连接地址
```

```
bds.setUsername("root"); // 数据库用户名
```

```
bds.setPassword("123456"); // 数据库密码
```

```
// 获取连接
```

```
Connection connection = bds.getConnection();
```

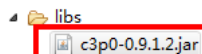
2.2使用配置文件设置连接参数 通过工厂方式获取连接

```
//把配置文件的信息读进来 到 props对象中
InputStream in = new FileInputStream("src/dbcp.properties");
Properties props = new Properties();
props.load(in);
//通过工厂创建连接池
DataSource bds = BasicDataSourceFactory.createDataSource(props);
//获取Connection连接对象
Connection connection = bds.getConnection();
```

c3p0

C3P0是一个开源的JDBC连接池，它实现了数据源和JNDI绑定，支持JDBC3规范和JDBC2的标准扩展。目前使用它的[开源项目](#)有Hibernate, Spring等。

1、导入jar包



2、具体代码实现（手动配置连接参数）

```
public Connection getConnection() {
    try {
        // 获取数据库连接池
        ComboPooledDataSource cpds = new ComboPooledDataSource();
        // 设置必要的 连接的参数信息
        cpds.setDriverClass("com.mysql.jdbc.Driver"); // 加载驱动
        cpds.setJdbcUrl("jdbc:mysql://localhost:3306/test001"); // 数据库连接地址
        cpds.setUser("root"); // 数据库用户名
        cpds.setPassword("123456"); // 数据库密码
        // 获得连接
        return cpds.getConnection();
    } catch (Exception e) {
    }
    return null;
}
```

通过配置xml文件实现，注意：**c3p0**的配置文件 要求文件名必须是 **c3p0-config.xml** 当配置了这个文件调用程序项目会自动去找这个文件并解析数据到项目中去

XML的配置：

<c3p0-config>

<default-config>

<property name="driverClass">com.mysql.jdbc.Driver</property>

<property name="jdbcUrl">jdbc:mysql://localhost:3306/test001</property>

<property name="user">root</property>

<property name="password">123456</property>

</default-config>

</c3p0-config>

```

/**
 * 使用配置文件设置连接参数 通过工厂方式获取连接
 *
 * @return Connection连接对象
 */
public Connection getConnection2() {
    try {
        // 当你在new 了这个对象，会自动到 classpath的路径下去寻找叫做 c3p0-config.xml配置文件
        // 然后把 配置文件中的信息 加载进来,去获得连接
        // 把配置文件的信息读进来 到 props对象中
        ComboPooledDataSource cpds = new ComboPooledDataSource();
        // 获得连接
        return cpds.getConnection();
    } catch (Exception e) {
    }
    return null;
}

```

Dbutils

用于解决对数据库的增删改查，使用这个开源框架对数据库的增删改查只需要一个JavaBean用来保存数据，然后调用相关方法只需要几行代码即可完成。

原理：

1、底层有获取使用了获取数据库元数据技术（主要是获取SQL语句参数的个数）

```
PreparedStatement stmt = conn.prepareStatement(sql);
```

//通过发送SQL语句的对象，获得占位符个数 获取传入SQL语句元数个数

```
ParameterMetaData pmd = stmt.getParameterMetaData();
```

```
int count = pmd.getParameterCount();
```

2、主要使用反射来实现数据的封装（主要是对数据进行封装）

3、自定义接口技术（主要用于回调实现特有的方法）

4、JavaBean对数据的封装技术（javabean主要封装对应表里的字段）

5、Object数组参数

Object[] objs为参数数组

使用方法：

1、首先要先集成JDBC连接池 优选C3P0框架

2、导入DbutilsJAR包

3、集成的JDBC连接池工具类主要有三个方法

3.1、返回连接池

//这里 new 就会去 c3p0-config.xml文件，连接就会放到这个 c3p0的连接池对象中

```
private static DataSource cpds = new ComboPooledDataSource();
```

//1、返回连接池

```
public static DataSource getDataSource(){
```

```
    return cpds;
```

```
}
```

3.2、获得Connection连接

```
public static Connection getConn(){
```

```

        try {
            return cpds.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
}

```

3.3、释放资源

```

public static void release(ResultSet rs,PreparedStatement stmt, Connection conn ){
    if(rs!=null){
        try {
            rs.close();
        } catch (SQLException e) {
        }
        rs = null;
    }
    if(stmt!=null){
        try {
            stmt.close();
        } catch (SQLException e) {
        }
        stmt = null;
    }
    if(conn!=null){
        try {
            conn.close();
        } catch (SQLException e) {
        }
        conn = null;
    }
}
}

```

4、创建数据库表信息的JavaBean类

5、完成 增删改 查

Apache的dbutils也是提供一个核心的类叫做 [QueryRunner](#)，用于完成 增删改查，其中增删改是一个方法，查询是另外的一个方法

update("参数1",参数2); 该方法用于增删改，具体使用如下代码：

QueryRunner runner = new QueryRunner(参数); 参数为：连接池对象

Object[] params ={"lala","123","拉拉"};

runner.update("insert into users values(null,?,?,?)",params);

runner.query(参数1, 参数2.....); 该方法用于查询具体使用见如下代码：

`QueryRunner runner = new QueryRunner(参数);` 参数为: 连接池对象

`String sql ="select * from users";`

`List<User> list = runner.query(sql, new BeanListHandler<User>(User.class));`

参数说明:

`sql`: SQL语句

`new BeanListHandler<User>(User.class)` : 内部定义接口的实现类

```
for (User user : list) {  
  
}
```

实现类说明:

`BeanListHandler<javaBean类名>(javaBean类名.class)` 查询的结果有多个的用于查询所有数据

`BeanHandler<javaBean类名>(javaBean类名.class)` 查询的结果有一个的 条件查询

ScalarHandler: 将单个值封装、例如select count (*), 求内容的条数

针对"`SELECT COUNT(*) FROM ajaxs where username=?`"这种条件查询

可以返回查询的结果条数

`//0`就是没有查询到 大于`0`就是有查询到

`long count = (long)qr.query(sql, new ScalarHandler(),username);`

关于JDBC的操作项目中的用法是

1、集成连接池

2、集成Dbutils

3、创建javabean

4、创建业务类调用方法返回结果

