PreparedStatment 是 Statement 的子接口，用来执行 SQL 语句。

- 不需要拼接SQL字符串，因为可以使用占位符来替代参数，然后通过方法将参数赋给占位符。

```
String sql = "insert into user(name,age) values(?,?)";
preparedStatement.setString(1,"张三");
preparedStatement.setInt(2,22);
```

- 防止SQL注入。
- 执行效率更高。

将二进制数据存入数据库，图片、视频、音频。

- 在数据表中创建存储二进制数据的字段。

  blob 65KB

  tinyblob 255KB

  mediumblob 16MB

  longblob 4GB

- 通过Java代码使用JDBC向数据库添加、查询记录。
  - 将图片转成流，IO流相关方法。
  - 将流存入数据库，JDBC相关方法。
- 读取二进制
  - 从数据库中将二进制数据读取到Java程序中，JDBC相关方法。
  - 将二进制数据通过输出流保存到本地，IO流相关方法。

JDBC如何操作事务。事务就是将完成同一个需求的多条SQL语句看成一个整体，要么全部执行成功，要么一条都不执行，事务的四大特性：

- 原子性
- 隔离性
- 一致性
- 持久性

```
public static void main(String[] args) {
    Connection connection = JDBCTools.getConnection();
    //张三借给李四500
    String sql = "update user set money = ? where id = ?";
    PreparedStatement preparedStatement = null;
    PreparedStatement preparedStatement1 = null;
    try {
        connection.setAutoCommit(false);
```

```
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1,500);
            preparedStatement.setInt(2,11);

            preparedStatement1 = connection.prepareStatement(sql);
            preparedStatement1.setInt(1,1500);
            preparedStatement1.setInt(2,12);

            preparedStatement.executeUpdate();
            int num = 10/10;
            preparedStatement1.executeUpdate();
            connection.commit();
        } catch (SQLException e) {
            e.printStackTrace();
            try {
                connection.rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }finally {
            JDBCTools.release(connection,preparedStatement,null);
            JDBCTools.release(null,preparedStatement1,null);
        }
    }
```

数据库连接池

传统JDBC开发方式

- 建立数据库连接
- 执行SQL语句
- 断开数据库连接

普通的JDBC数据库连接使用DriverManager来获取，每次创建连接都需要向数据库申请获取连接，验证用户名和密码，比较消耗资源，连接对象的获取需要耗费一定的资源。

执行完SQL语句中华直接断开连接，这样的方式会消耗大量的资源和时间，数据库连接资源并没有得到良好的重复利用。

数据库连接池可以解决这个问题，基本思想是为数据库建立一个缓冲池，预先向缓冲池中放入一定数量的数据库连接对象，当需要获取数据库连接时，只需要从缓冲池中取出一个来使用，用完之后不要断开，而是还回到缓冲池中，供下一次请求使用，做到资源的重复利用。允许请求重复使用一个现有的数据库连接对象，而不是重新创建一个。

当数据库连接池中没有空闲的连接对象时，请求会进入等待队列，等待其他线程释放连接。

JDBC的数据库连接池如何使用，JDBC中提供了javax.sql.DataSource来完成数据库连接池的使用，DataSource是一个接口，只定义了功能，但是没有实现，实际开发中使用的实现是C3P0。

- 导入jar
- 创建C3P0对象

```
ComboPooledDataSource dataSource = new ComboPooledDataSource();
try {
    dataSource.setDriverClass("com.mysql.cj.jdbc.Driver");
    dataSource.setJdbcUrl("jdbc:mysql://localhost:3306/test");
    dataSource.setUser("root");
    dataSource.setPassword("root");
    dataSource.setInitialPoolSize(20);
    dataSource.setAcquireIncrement(5);
    dataSource.setMaxPoolSize(40);
    dataSource.setMinPoolSize(2);
    Connection connection = dataSource.getConnection();
} catch (PropertyVetoException e) {
    e.printStackTrace();
} catch (SQLException e) {
    e.printStackTrace();
} finally {

}
```

c3p0-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<c3p0-config>

    <named-config name="testc3p0">

        <!-- 指定连接数据源的基本属性 -->
        <property name="user">root</property>
        <property name="password">root</property>
        <property name="driverClass">com.mysql.cj.jdbc.Driver</property>
        <property name="jdbcUrl">jdbc:mysql://localhost:3306/test</property>

        <!-- 数据库连接池相关属性 -->
        <!-- 连接池中连接数不足时，一次向数据库申请的连接个数 -->
        <property name="acquireIncrement">5</property>
        <!-- 初始化连接数 -->
        <property name="initialPoolSize">20</property>
        <!-- 最大连接数 -->
        <property name="maxPoolSize">40</property>
        <!-- 最小连接数 -->
        <property name="minPoolSize">2</property>

    </named-config>
```

```
</c3p0-config>
```

```
try {
        ComboPooledDataSource dataSource = new
ComboPooledDataSource("testc3p0");
        Connection connection = dataSource.getConnection();
        System.out.println(connection);
    } catch (SQLException e) {
        e.printStackTrace();
    }
```

```
package com.southwind.util;

import com.mchange.v2.c3p0.ComboPooledDataSource;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class C3P0Tools {
    private static ComboPooledDataSource dataSource;

    static {
        dataSource = new ComboPooledDataSource("testc3p0");
    }

    public static Connection getConnection(){
        Connection connection = null;
        try {
            connection = dataSource.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }

    public static void release(Connection connection, Statement statement,
ResultSet resultSet){
        try {
            if(connection != null){
                connection.close();
            }
```

```
            if(statement != null){
                statement.close();
            }
            if(resultSet != null){
                resultSet.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

DBUtils：

直接使用原生的JDBC查询数据，需要手动进行封装，步骤繁琐，非常麻烦，使用DBUtils工具可以帮助开发者完成数据的封装，直接查询数据表返回的对象不再是ResultSet，而是一个对象或者一个集合，DBUtils工具内部会完成ResultSet的解析，封装成对象。

使用步骤：

- 导入jar
- 创建QueryRunner对象

  ResultSetHandler接口的实现类

  - BeanHandler 将ResultSet转换为JavaBean
  - BeanListHandler 将ResultSet转换为List
  - BeanMapHandler 将ResultSet转换为Map集合
  - MapListHandler 将ResultSet转换为List