# JDBC-Java Database Connectivity

jdbc是指使用java操作数据库.

Java程序提供JDBC连接的接口,各个数据库厂商对接口进行实现.

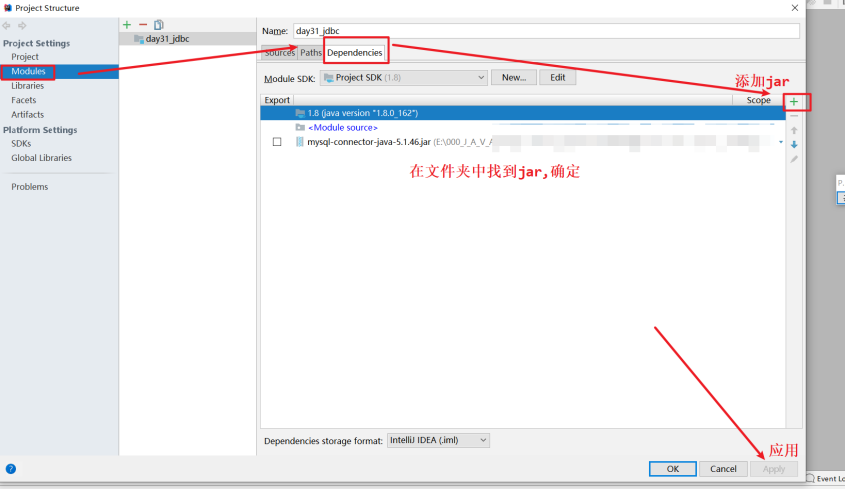
 这就是mysql对java数据库操作接口的实现.

# JDBC入门程序-插入

1 创建java项目

2 导入jar包

file-project structure



3 编写jdbc程序

jdbc执行步骤:

1 加载驱动

2 通过驱动管理器获得连接

3 通过连接对象获得执行语句对象

4 通过连接对象执行sql

5 关流

|  |
| --- |
| **package** com.zhiyou100.jdbc;  **import** java.sql.\*;  */\*\*  \** ***@author*** *QiuShiju  \** ***@date*** *2021/1/12  \** ***@desc*** *jdbc 入门演示,插入数据  \*/* **public class** Demo1 {   **public static void** main(String[] args) **throws** ClassNotFoundException, SQLException {  *// 1 加载驱动* Class.*forName*(**"com.mysql.jdbc.Driver"**);   *// 2 通过驱动管理器,获得连接  // https://www.baidu.com:80?tn=49  // jdbc协议:mysql协议://ip:端口/数据库名?数据库参数* String url = **"jdbc:mysql://localhost:3306/java3201"**;  String username = **"root"**;  String password = **"123456"**;  Connection conn = DriverManager.*getConnection*(url,username,password);   *// 3 通过连接对象获得执行语句对象* Statement statement = conn.createStatement();   *// 4 通过执行语句对象,执行sql  // 建议执行增删改的方法使用executeUpdate()* String sql = **"insert into student values (22,'孙悟空',500,'公',100,1,'2021-01-12')"**; *// 数据不对. String sql = "insert into student values (22,'唐僧',100,'男',100,1,"+new Date(new java.util.Date().getTime())+")";  // 执行增删改,返回受影响的行数* **int** num = statement.executeUpdate(sql);  **if**(num > 0) {  System.***out***.println(**"插入成功"** );  }   *// 5 关联* conn.close();  statement.close();  }  } |

# jdbc - 删

步骤与插入基本一致,只有sql不同

|  |
| --- |
| **package** com.zhiyou100.jdbc;  **import** java.sql.Connection; **import** java.sql.DriverManager; **import** java.sql.SQLException; **import** java.sql.Statement;  */\*\*  \** ***@author*** *QiuShiju  \** ***@date*** *2021/1/12  \** ***@desc*** *删除  \*/* **public class** Demo2 {   **public static void** main(String[] args) **throws** ClassNotFoundException, SQLException {  *// 1 加载驱动* Class.*forName*(**"com.mysql.jdbc.Driver"**);   *// 2 获得连接对象* Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://localhost:3306/java3201"**, **"root"**, **"123456"**);   *// 3 获得执行语句对象* Statement statement = conn.createStatement( );   *// 4 执行sql* String sql = **"delete from student where sid = 22"**;  **int** i = statement.executeUpdate(sql);  **if** (i > 0) {  System.***out***.println(**"删除成功"** );  }**else** {  System.***out***.println(**"删除失败,没有该条数据!"** );  }   *// 5 关流* conn.close();  statement.close();    }  } |

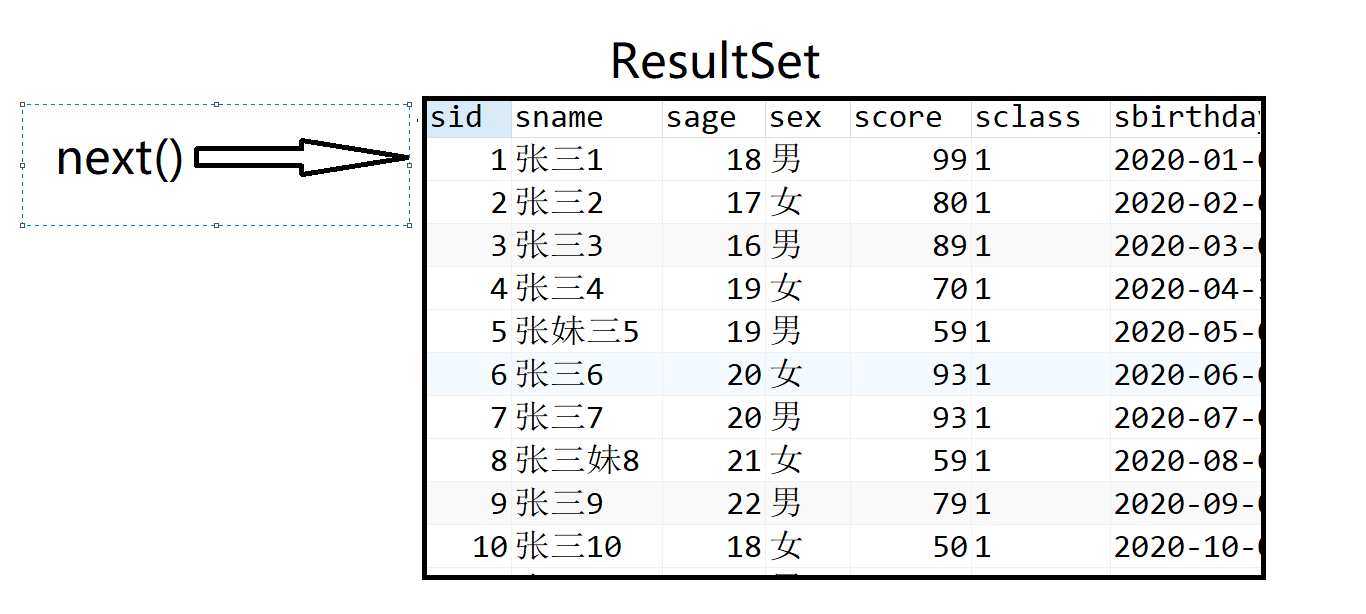
# jdbc-更新

步骤与插入一致,只有sql不同.

更新student表中id为21的人姓名为嫦娥

# jdbc-查询

|  |
| --- |
| **package** com.zhiyou100.jdbc;  **import** java.sql.\*;  */\*\*  \** ***@author*** *QiuShiju  \** ***@date*** *2021/1/12  \** ***@desc*** *\*/* **public class** Demo4 {   **public static void** main(String[] args) **throws** ClassNotFoundException, SQLException {  *// 1 加载驱动* Class.*forName*(**"com.mysql.jdbc.Driver"**);   *// 2 获得连接* Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://localhost:3306/java3201"**, **"root"**, **"123456"**);   *// 3 获得执行语句对象* Statement statement = conn.createStatement( );   *// 4 执行sql* String sql = **"select score,sbirthday,sid id,sname,sage,sex,sclass from student"**;  *// 1) 执行查询的方法是executeQuery()  // 2) 执行后返回的是结果集(ResultSet),其中存储查询返回的虚拟表的所有数据* ResultSet rs = statement.executeQuery(sql);  */\*\*  \* 从ResultSet中取出值,类似于集合的迭代  \* 0 结果集可能有多条数据,需要重复取值,所以需要循环操作  \* 1 先判断有无下一个数据  \* 2 如果有数据,开始取出该行数据  \* ps:取值是按照查询返回的虚拟表进行操作  \* 2.1 可以通过列顺序取值  \* 列的顺序从1开始  \* getObject(int i) -- 不确定取出的数据类型时使用  \* getInt(int i) -- getXxx 取出值即为对应的数据类型  \* 2.2 可以通过列名取值  \* getObject(String columnname) -- 不确定取出的数据类型时使用  \* getInt(String columnname) -- getXxx 取出值即为对应的数据类型  \*/* **while**(rs.next()) { *// int id = rs.getInt(1); // String name = rs.getString(2); // int age = rs.getInt(3); // String sex = rs.getString(4); // double score = rs.getDouble(5); // Date birthday = rs.getDate(7);* **int** id = rs.getInt(**"id"**);  String name = rs.getString(**"sname"**);  **int** age = rs.getInt(**"sage"**);  String sex = rs.getString(**"sex"**);  **double** score = rs.getDouble(**"score"**);  **int** sclass = rs.getInt(**"sclass"**);  Date birthday = rs.getDate(**"sbirthday"**);   System.***out***.println(id+**"-"**+name+**"-"**+age+**"-"**+sex+**"-"**+score+**"-"**+birthday );  }   *// 5关流* rs.close();  statement.close();  conn.close();    }  } |



# 封装结果集

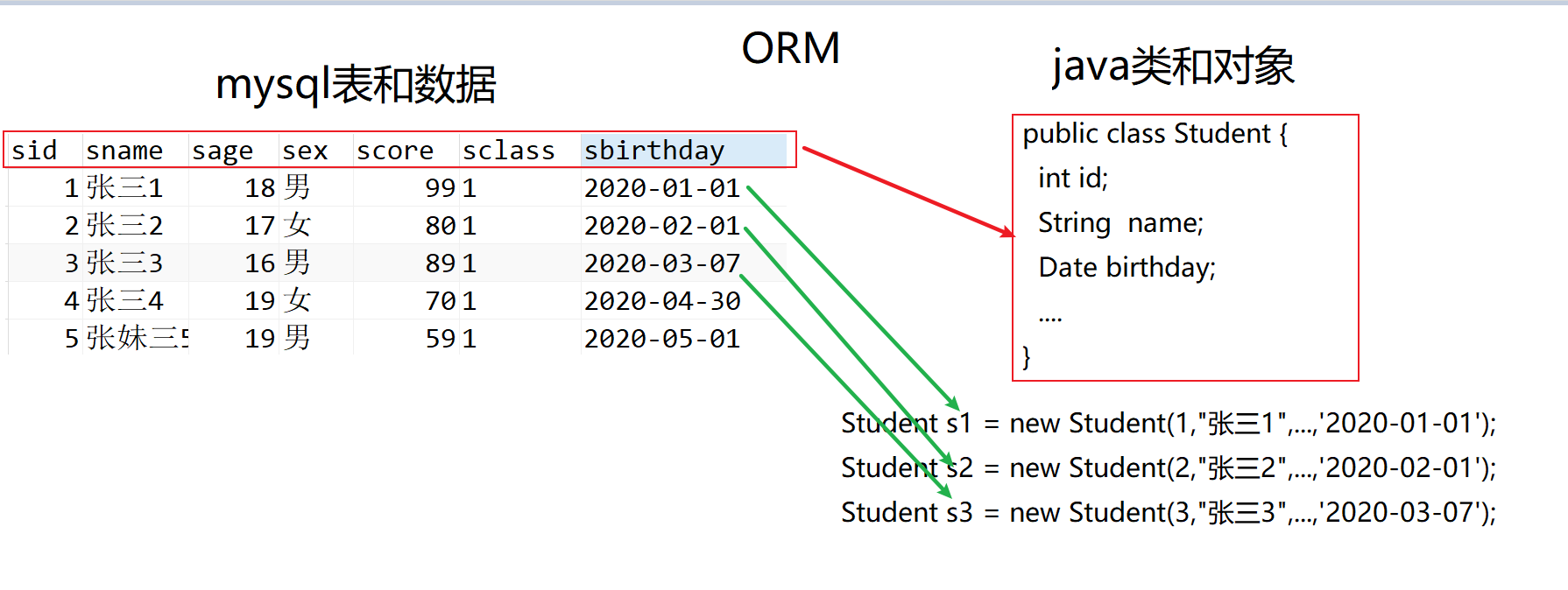
java类 - 表

属性名 - 字段名

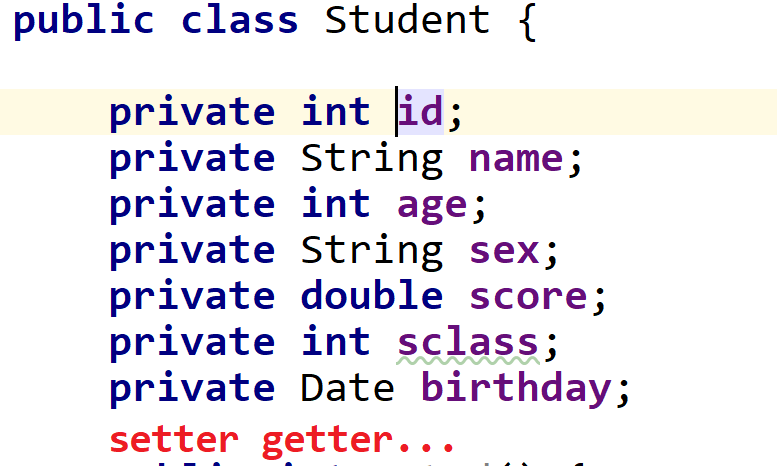
属性值 - 列值

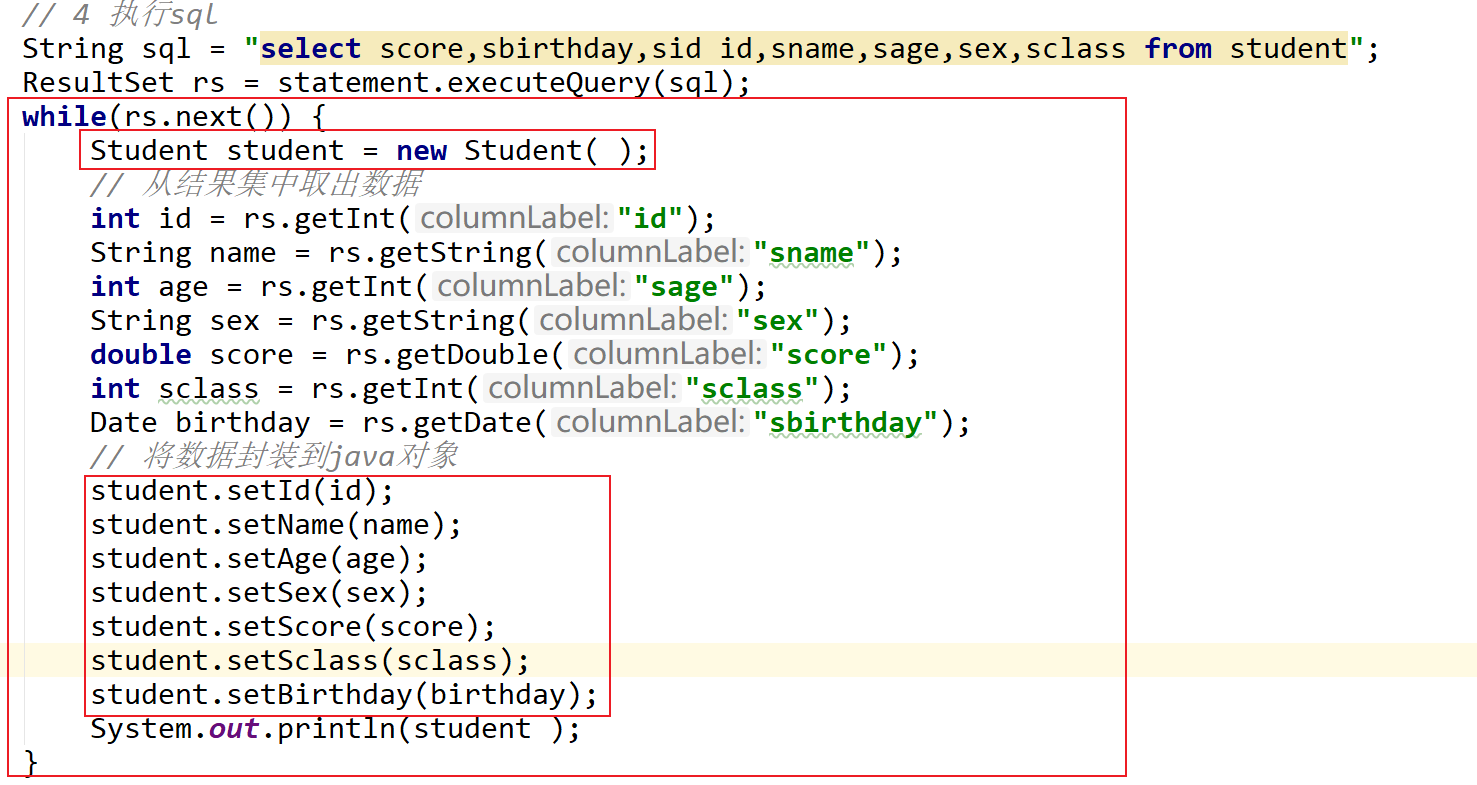
对象 - 行

对象关系模型 – ORM



* 根据数据表的字段及数据类型创建对应的java类
* 将查询出的数据,封装到对应的java类对象的属性中





# 练习+作业

## 1根据id从数据库查询数据

public static Student getOne(int id)

|  |
| --- |
| **package** com.zhiyou100.jdbc;  **import** java.sql.\*;  */\*\*  \** ***@author*** *QiuShiju  \** ***@date*** *2021/1/12  \** ***@desc*** *\*/* **public class** Demo6 {   **public static void** main(String[] args) {  Student one = *getOne*(220);  System.***out***.println(one );  }   **public static** Student getOne(**int** id){  Connection conn = **null**;  Statement statement = **null**;  ResultSet rs = **null**;  Student student = **null**;  **try** {  Class.*forName*(**"com.mysql.jdbc.Driver"**);  conn = DriverManager.*getConnection*(**"jdbc:mysql://localhost:3306/java3201"**, **"root"**, **"123456"**);  statement = conn.createStatement( );  String sql = **"select** *\** **from student where sid = "**+id;  rs = statement.executeQuery(sql);  **while** (rs.next()) {  student= **new** Student( );  student.setId(rs.getInt(**"sid"**));  student.setName(rs.getString(**"sname"**));  student.setScore(rs.getDouble(**"score"**));  student.setBirthday(rs.getDate(**"sbirthday"**));  }  } **catch** (Exception e) {  e.printStackTrace( );  } **finally** {  **try** {  rs.close();  statement.close();  conn.close();  } **catch** (SQLException e) {  e.printStackTrace( );  }  }  **return** student;  }  } |

## 2调用方法向数据库插入数据

public static boolean addOne(Student stu)

## 3 设计方法,通过id删除用户数据

public static int deleteById(int id)

返回值是受影响的行数

## 4 设计方法,通过id更改用户数据

public static int updateById(Student stu)

注意:根据id更新,即参数Student对象中一定有id属性值

更新Student表的数据,根据Student对象的属性更新,如果属性值是null则不跟更新,不为null就更新对应的字段

返回值是受影响的行数

## 5 设计方法,查询全部数据

public static List<Student> findAll()

返回值是List集合,集合中是全部学生数据