# DbUtils组件

DbUtils:关闭资源,加载驱动

QueryRunner:组件的核心工具类,定义了所有与数据库操作的方法(查询,更新)

int update(Connection conn,String sql,Object param); 执行更新,带一个占位符的sql

int update(Connection conn,String sql,Object ... param); 执行更新,带多个占位符的sql

int [] batch(Connection conn,String sql,Object[][] params) 批处理

不带Connection

int update(String sql,Object param); 执行更新,带一个占位符的sql

int update(String sql,Object ... param); 执行更新,带多个占位符的sql

int [] batch(String sql,Object[][] params) 批处理

注意:如果调用DbUtils组件的操作数据库方法,没有传入连接对象,那么在实例化

QueryRunner对象的时候需要传入数据库源对象 QueryRunner qr = new QueryRunner(ds);

构造函数:

1. QueryRunner(DataSource ds) //传入一个连接池
2. QueryRunner();//无参构造函数

## 1.1自定义封装数据

|  |
| --- |
| //1.查询,自定义结果集封装数据  private static Connection conn;  public static void main(String args[]) throws Exception{  String sql = "select \* from admin where id=?";  conn = JdbcUtil.getConnection();  //创建DbUtils核心工具类对象  QueryRunner qr = new QueryRunner();  //查询  Admin admin = qr.query(conn, sql, new ResultSetHandler<Admin>(){  @Override  //如果封装一个Admin对象  public Admin handle(ResultSet resultSet) throws SQLException {  Admin admin = new Admin();  resultSet.next();  admin.setId(resultSet.getInt("id"));  admin.setName(resultSet.getString("name"));  return admin;  }  },2); //这个2表示sql参数对应的值  } |

## 2.使用组件提供的结果集来封装数据

|  |
| --- |
| String sql = "select \* from admin where id=?";  conn = JdbcUtil.getConnection();  //创建DbUtils核心工具类对象  QueryRunner qr = new QueryRunner();  //查询  Admin admin = qr.query(conn,sql,new BeanHandler<Admin>(Admin.class),2);  System.out.println(admin);  conn.close(); |