# 一. BeanUtils组件

使用BeanUtils必须下载下载两个包:

commons-beanutils-1.9.3.jar

commons-logging-1.2.jar

BeanUtils的包org.apache.commons.beanutils.BeanUtils;

基本操作:

1)对象的属性的赋值:

BeanUtils.setProperty(a,"name","haha");

BeanUtils.copyProperty(a,"age","18");//注意:对于基本数据类型,会自动进行数据转换

2)对象的拷贝:

//相同类型的拷贝

Admin newAdmin = new Admin();

BeanUtils.copyProperties(newAdmin,a);

//map的拷贝,将map中的某个key-value拷贝到javabean中

Admin adminMap = new Admin();

Map<String,Object> map = new HashMap<>();

map.put("name","Jerry");

map.put("age",29);

//注意:map的的key要和javabean的属性名一致

BeanUtils.populate(adminMap,map);

System.out.println(adminMap);

规律:上述函数左边是要被赋值的对象

3)日期类型拷贝:

方式1:自定义Convert

|  |
| --- |
| Admin a = new Admin();  String name = "jack";  int age = 20;  String birth = "1999-09-09";  //注册日期类型转换器:  ConvertUtils.register(new Converter() {  //转换的内部实现方法,需要重写convert方法  @Override  //字符串转换为日期  public Object convert(Class type, Object value) {  //判断  if(type != Date.class) return null;  if(value ==null||"".equals(value.toString().trim())){  return null;  }  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  try {  return sdf.parse(value.toString());  } catch (ParseException e) {  throw new RuntimeException(e);  }  }  },Date.class);  BeanUtils.copyProperty(a,"name",name);  BeanUtils.copyProperty(a,"age",age);  BeanUtils.copyProperty(a,"birth",birth);  System.out.println(a); |

注意:当执行这一行的时候BeanUtils.copyProperty(a,"birth",birth);,会去调用ConvertUtils.register(Converter converter, Class clazz)方法,这个方法有两个参数,第二个参数为我们要转换的类型Date的class第一个参数是一个接口,这个接口要被我们重写里面的一个Object convert(Class type, Object value) 方法,这个方法的第一个参数是我们要转换的Date类型的class,第二个参数是copyProperty的第三个参数birth字符串,

方式2:使用组件提供的Convert

ConvertUtils.register(new DateLocaleConverter(),Date.class);

# 2.元数据

在jdbc中获取数据库的定义,例如:数据库,表,列的定义信息,就会用到元数据

在jdbc中可以使用:数据库元数据,参数元数据,结果集元数据

元数据定义相关的api:为XXXMetaData

|  |
| --- |
| Class.forName(driver);  Connection conn = DriverManager.getConnection(url,user,passwd);  // String sql = "select \* from dept where id=? and name=?";  String sql = "select \* from dept";  PreparedStatement pstmt = conn.prepareStatement(sql);  // pstmt.setInt(1,2);  // pstmt.setString(2,"abc");  //获取sql参数元数据  ParameterMetaData parameterMetaData = pstmt.getParameterMetaData();  //获取sql参数的个数  // int count = parameterMetaData.getParameterCount();  //结果集元数据(通过结果集元数据,得到列名称)  ResultSet resultSet = pstmt.executeQuery();  //迭代每一行  ResultSetMetaData r\_metaData = resultSet.getMetaData();  //1.获取列的个数  int c\_count = r\_metaData.getColumnCount();  while(resultSet.next()) {  //2.遍历,获取每一列的名称  for (int i = 1; i <= c\_count; i++) {  String columnName = r\_metaData.getColumnName(i);  //获取每一行的的每一列的值  Object columnValue = resultSet.getObject(columnName);  System.out.println(columnValue);  }  } |

# 3.Dao操作的抽取

Admin.java

|  |
| --- |
| public class Admin {  private int id;  private String name;  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  @Override  public String toString() {  return "Admin{" +  "id=" + id +  ", name='" + name + '\'' +  '}';  }  } |

BaseDao.java

|  |
| --- |
| public class BaseDao{  private Connection con;  private PreparedStatement pstmt;  private ResultSet rs;  /\*\*  \* 更新的通用方法  \* @param sql 更新的sql语句(update/insert/delete)  \* @param paramsValues sql语句中占位符对应的值(如果没有占位符,传入null)  \*/  public void update(String sql,Object[] paramsValues){  try {  this.con = JdbcUtil.getConnection();  this.pstmt = con.prepareStatement(sql);  //参数元数据,得到占位符参数的个数  int count = pstmt.getParameterMetaData().getParameterCount();  //设置占位符参数的值  if(paramsValues!=null&&paramsValues.length>0){  //循环给参数赋值  for(int i=1;i<=count;i++){  pstmt.setObject(i,paramsValues[i-1]);  }  //执行更新  pstmt.executeUpdate();  }  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JdbcUtil.closeAll(con,pstmt,null);  }  }  /\*\*  \*  \* 查询的通用方法  \* @param sql  \* @param paramsValue  \*/  public <T> List<T> query(String sql,Object[] paramsValue,Class<T> clazz){  try {  //返回的集合  List<T> list = new ArrayList<T>();  //对象  T t;  //1.获取连接  this.con = JdbcUtil.getConnection();  //2.创建stmt对象  this.pstmt = this.con.prepareStatement(sql);  //3.获取占位符参数个数,并设置每个参数的值  int count = this.pstmt.getParameterMetaData().getParameterCount();  if(paramsValue !=null && paramsValue.length > 0){  for(int i=1;i<=count;i++){  this.pstmt.setObject(i,paramsValue[i-1]);  }  }  //4.执行查询  this.rs = this.pstmt.executeQuery();  //5.获取结果集元数据  ResultSetMetaData rsmd = rs.getMetaData();  //获取列的个数  int columnCount = rsmd.getColumnCount();  //6.遍历rs  while (rs.next()){  //要封装的对象  t = clazz.newInstance();  //7.遍历每一行的每一列,封装数据  for(int i=1;i<=columnCount;i++){  //获取列名称  String columnName = rsmd.getColumnName(i);  //获取列名称对象的值  Object value = rs.getObject(columnName);  //封装  BeanUtils.copyProperty(t,columnName,value);  }  //把封装完毕的对象添加到集合中  list.add(t);  }  return list;  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JdbcUtil.closeAll(con,pstmt,null);  }  }  } |

JdbcUtil.java

|  |
| --- |
| public class JdbcUtil{  private static String url = "jdbc:mysql://47.96.27.61:3306/test";  private static String user = "lw";  private static String passwd = "linwei";  private static String driver = "com.mysql.jdbc.Driver";  public static Connection getConnection(){  try {  Class.forName(driver);  return DriverManager.getConnection(url,user,passwd);  } catch (Exception e) {  throw new RuntimeException(e);  }  }  public static void closeAll(Connection con, Statement stmt, ResultSet rs){  try{  if(rs!=null){  rs.close();  rs = null;  }  if(stmt!=null){  stmt.close();  stmt = null;  }  if(con!=null&&!con.isClosed()){  con.close();  con = null;  }  }catch (SQLException e){  throw new RuntimeException(e);  }  }  } |

AdminDao.java

|  |
| --- |
| public class AdminDao extends BaseDao{  //删除  public void delete(int id){  String sql = "delete from admin where id=?";  Object[] paramsValue = {id};  super.update(sql,paramsValue);  }  //插入  public void insert(Admin admin){  String sql = "insert into admin(id,name)values(?,?)";  Object[] paramsValue = {admin.getId(),admin.getName()};  super.update(sql,paramsValue);  }  //查询所有  public List<Admin> findAll(){  String sql = "select \* from admin";  return super.query(sql,null,Admin.class);  }  //按照条件查询(主键)  public Admin findById(int id){  String sql = "select \* from admin where id=?";  List<Admin> list = super.query(sql,new Object[]{id},Admin.class);  return (list!=null&&list.size()>0)?list.get(0):null;  }  } |