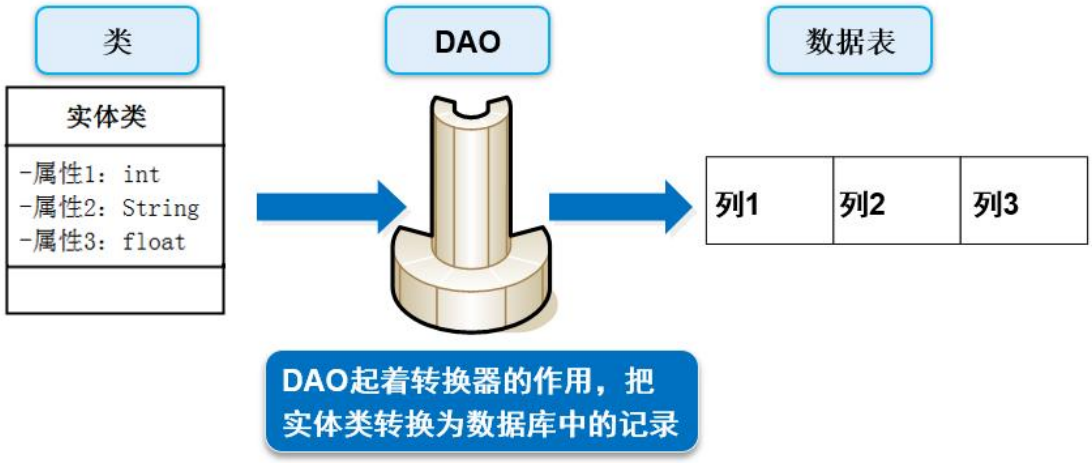
第五讲 DAO 模式

一、什么是 DAO 模式

- Data Access Object(数据存取对象)
- 位于业务逻辑和持久化数据之间
- 实现对持久化数据的访问



二、DAO 模式的组成部分

- 实体类
- 数据库连接和关闭工具类
- 通用 DAO 接口及其实现类
- 自定义 DAO 接口及其实现类

三、数据库连接和关闭工具类

1. 配置文件

Java 中的配置文件常为 properties 文件
后缀为.properties
格式是“键 = 值”格式
使用“ # ”来注释

database.properties

```
url=jdbc:mysql://localhost:3306/shoppingstreet?useUnicode=true&amp&characterEncoding=utf-8
driver=com.mysql.jdbc.Driver
username=root
password=root
```

2. Properties 类

Java 中提供了 Properties 类来读取配置文件

方法名	说　　明
String getProperty(String key)	用指定的键在此属性列表中搜索属性。通过参数 key 得到其所对应的值
Object setProperty(String key, String value)	调用 Hashtable 的方法 put。通过调用基类的 put()方法来设置键-值对
void load(InputStream inStream)	从输入流中读取属性列表 （键和元素对）。通过对指定文件进行装载获取该文件中所有键-值对
void clear()	清除所装载的键-值对，该方法由基类 Hashtable 提供

3. DataSourceUtil.java

```
package com.shoppingstreet.utils;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
import java.util.Properties;

public class DataSourceUtil {
	private static DataSource dataSource;

	private static String driver;
	private static String url;
	private static String user;
	private static String password;

	static {
		init();
	}

	public static void init(){
		Properties params=new Properties();
		String configFile = "database.properties";
		InputStream
is=DataSourceUtil.class.getClassLoader().getResourceAsStream(configFile);
		try {
			params.load(is);
		} catch (IOException e) {
			e.printStackTrace();
		}
		driver=params.getProperty("driver");
		url=params.getProperty("url");
```

```
            user=params.getProperty("username");
            password=params.getProperty("password");
        }

        //获取连接
        public static Connection openConnection() throws SQLException {
            Connection connection = null;
            try {
                Class.forName(driver);
                connection = DriverManager.getConnection(url, user, password);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            return connection;
        }
        //关闭连接
        public static void closeConnection(Connection connection) {
            try {
                if (connection != null)
                    connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
}
```

四、通用 DAO 接口及其实现类

数据库工具类 BaseDao：增、删、改的通用方法

1. IBaseDao 接口

```
package com.shoppingstreet.dao;

import java.sql.ResultSet;

public interface IBaseDao {
	public ResultSet executeQuery(String sql,Object[] params);
	public int executeUpdate(String sql,Object[] params);
	public int executeInsert(String sql,Object[] params);
	public boolean closeResource();
	public boolean closeResource(ResultSet reSet);
	public Object tableToClass(ResultSet rs) throws Exception;
}
```

2. IBaseDao 接口的实现类--BaseDao

```
package com.shoppingstreet.dao.impl;

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

import com.shoppingstreet.dao.IBaseDao;

public abstract class BaseDaoImpl implements IBaseDao {
	protected Connection connection;

	protected PreparedStatement pstm;

	public BaseDaoImpl(Connection connection) {
```

```
        this.connection = connection;
    }

    public ResultSet executeQuery(String sql,Object[] params){
        ResultSet rs=null;
        try {
            pstm = connection.prepareStatement(sql);
            for(int i = 0; i < params.length; i++){
                pstm.setObject(i+1, params[i]);
            }
            rs = pstm.executeQuery();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return rs;
    }

    //增删改操作
    public int executeUpdate(String sql,Object[] params){
        int updateRows = 0;
        try {
            pstm = connection.prepareStatement(sql);
            for(int i = 0; i < params.length; i++){
                pstm.setObject(i+1, params[i]);
            }
            updateRows = pstm.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
            updateRows = -1;
        }

        return updateRows;
    }

    public int executeInsert(String sql,Object[] params){
```

```
        Long id = 0L;
        try {
            pstm                                                                    =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS);
            for(int i = 0; i < params.length; i++){
                pstm.setObject(i+1, params[i]);
            }
            pstm.executeUpdate();
            ResultSet rs = pstm.getGeneratedKeys();
            if (rs.next()) {
                id = rs.getLong(1);
                System.out.println("数据主键：" + id);
            }

        } catch (Exception e) {
            e.printStackTrace();
            id =null;
        }

        return id.intValue();
    }


    //释放资源
    public boolean closeResource(){
        if(pstm != null){
            try {
                pstm.close();
            } catch (SQLException e) {
                e.printStackTrace();
                return false;
            }
        }
        return true;
    }
```

```
    public boolean closeResource(ResultSet reSet){
        if(reSet != null){
            try {
                reSet.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                return false;
            }
        }
        return true;
    }

     // 需要重写的方法，将结果集转换为对象
    public abstract Object tableToClass(ResultSet rs) throws Exception;

}
```

五、自定义 DAO 接口及其实现类

以用户登录注册为例

1. UserDao 接口

```
package com.shoppingstreet.dao;

import com.shoppingstreet.entity.User;

public interface UserDao extends IBaseDao{
    int add(User user) throws Exception;//新增用户信息
    User getLoginUser(String loginName,String password);
}
```

2. UserDao 实现类

```
package com.shoppingstreet.dao.impl;

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

import com.shoppingstreet.dao.UserDao;
import com.shoppingstreet.entity.User;

/**
 * 用户 dao
 */
public class UserDaoImpl extends BaseDaoImpl implements UserDao {

    public UserDaoImpl(Connection connection) {
        super(connection);
    }

    @Override
    public User tableToClass(ResultSet rs) throws Exception {
        User user = new User();
        user.setLoginName(rs.getString("loginName"));
        user.setPassword(rs.getString("password"));
        user.setEmail(rs.getString("email"));
        user.setMobile(rs.getString("mobile"));
        user.setId(rs.getInt("id"));
        return user;
    }
    /**
     * 保存用户
     *
     * @param user
```

```
     * @throws java.sql.SQLException
     */
    public int add(User user){//新增用户信息
        Integer id=0;
        try {
            String sql=" insert into user(loginName,password,email,mobile) values(?,?,?,?) ";
            try {
                Object param[]=new Object[]{user.getLoginName(),user.getPassword(),user.getEmail(),user.getMobile()};
                id=this.executeInsert(sql,param);
                user.setId(id);
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }finally{
            this.closeResource();
        }
        return id;
    }

    @Override
    public User getLoginUser(String loginName, String password) {
        String sql = "select id,loginName,password,email,mobile from user where loginName=? and password=?";

        ResultSet resultSet = this.executeQuery(sql.toString(),new String[]{loginName,password});
        User user=null;
        try {
            if(resultSet.next()){
                user = this.tableToClass(resultSet);
            }
```

```
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } finally{
                this.closeResource();
                this.closeResource(resultSet);
            }
            return user;
        }
}
```

六、实例

1. 使用 DAO 模式完成商品查询操作

① ProductDao

```
package com.shoppingstreet.dao;

import java.util.List;

import com.shoppingstreet.entity.Product;

public interface ProductDao extends IBaseDao {
    public List<Product> getProductByName(String proName)throws Exception;
    public Product getProductById(Integer id)throws Exception;
}
```

② ProductDaoImpl

```
package com.shoppingstreet.dao.impl;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.shoppingstreet.dao.ProductDao;
import com.shoppingstreet.entity.Product;

public class ProductDaoImpl extends BaseDaoImpl implements ProductDao {

    public ProductDaoImpl(Connection connection) {
        super(connection);
    }

    @Override
    public Product tableToClass(ResultSet rs) throws Exception {
        Product product = new Product();
        product.setId(rs.getInt("id"));
        product.setName(rs.getString("name"));
        product.setDescription(rs.getString("description"));
        product.setPrice(rs.getFloat("price"));
        product.setStock(rs.getInt("stock"));
        product.setCategoryLevel1Id(rs.getInt("categoryLevel1Id"));
        product.setCategoryLevel2Id(rs.getInt("categoryLevel2Id"));
        product.setCategoryLevel3Id(rs.getInt("categoryLevel3Id"));
        product.setFileName(rs.getString("fileName"));
        return product;
    }

    @Override
```

```
    public List<Product> getProductByName(String proName) throws Exception {
        List<Object> paramsList=new ArrayList<Object>();
        List<Product> productList=new ArrayList<Product>();
        StringBuffer sql=new StringBuffer(" select id,name,description,price,stock,categoryLevel1Id,categoryLevel2Id,categoryLevel3Id,fileName,isDelete from product  where name like ? ");
        ResultSet resultSet = null;
        try{
            paramsList.add("%"+proName+"%");
            resultSet=this.executeQuery(sql.toString(),paramsList.toArray());
            while (resultSet.next()) {
                Product product = this.tableToClass(resultSet);
                productList.add(product);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }finally{
            this.closeResource(resultSet);
            this.closeResource();
        }
        return productList;
    }

    @Override
    public Product getProductById(Integer id) throws Exception {
        String sql = " select id,name,description,price,stock,categoryLevel1Id,categoryLevel2Id,categoryLevel3Id,fileName,isDelete from product where id = ? ";
        ResultSet resultSet = null;
        Product product = null;
        try {
            Object params[] = new Object[] { id };
            resultSet = this.executeQuery(sql, params);
            while (resultSet.next()) {
```

```
                product = tableToClass(resultSet);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            this.closeResource(resultSet);
            this.closeResource();
            return product;
        }
    }
}
```

2. 使用 DAO 模式完成生成订单操作

涉及到订单与订单详情两个接口

① 订单与订单详情接口

- OrderDao

```
package com.shoppingstreet.dao;

import com.shoppingstreet.entity.Order;

public interface OrderDao extends IBaseDao {
    public void add(Order order) ;
}
```

- OrderDetailDao

```
package com.shoppingstreet.dao;

import com.shoppingstreet.entity.OrderDetail;

public interface OrderDetailDao extends IBaseDao {
    public void add(OrderDetail detail) throws Exception;
```

```
}
```

订单与订单详情接口实现

OrderDaoImpl

```
package com.shoppingstreet.dao.impl;

import java.sql.Connection;
import java.sql.ResultSet;
import java.util.Date;

import com.shoppingstreet.dao.OrderDao;
import com.shoppingstreet.entity.Order;

public class OrderDaoImpl extends BaseDaoImpl implements OrderDao {

    public OrderDaoImpl(Connection connection) {
        super(connection);
    }

    @Override
    public Order tableToClass(ResultSet rs) throws Exception {
        Order order = new Order();
        order.setId(rs.getInt("id"));
        order.setUserId(rs.getInt("userId"));
        order.setCreateTime(rs.getDate("createTime"));
        order.setCost(rs.getDouble("cost"));
        order.setUserAddress(rs.getString("userAddress"));
        order.setSerialNumber(rs.getString("serialNumber"));
        order.setLoginName(rs.getString("loginName"));
        return order;
    }
    /**
     * 保存订单
     * @param order
     * @throws java.sql.SQLException
     */
```

```
    public void add(Order order) {//保存订单
        Integer id=0;
        String sql="insert into shoppingStreet.order(userId,loginName,userAddress,createTime,cost,serialNumber) values(?,?,?,?,?,?) ";
        Object[] param=new Object[]{order.getUserId(),order.getLoginName(),order.getUserAddress(),new Date(),order.getCost(),order.getSerialNumber()};
        try {
            id=this.executeInsert(sql, param);
            order.setId(new Integer(id).intValue());
        }catch (Exception e) {
            e.printStackTrace();
        }finally{
            this.closeResource();
        }
    }
}
```

OrderDetailDaoImpl

```
package com.shoppingstreet.dao.impl;

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

import com.shoppingstreet.dao.OrderDetailDao;
import com.shoppingstreet.dao.ProductDao;
import com.shoppingstreet.entity.OrderDetail;
import com.shoppingstreet.entity.Product;

public class OrderDetailDaoImpl extends BaseDaoImpl implements OrderDetailDao{
```

```
    public OrderDetailDaoImpl(Connection connection) {
        super(connection);
    }



    @Override
    public OrderDetail tableToClass(ResultSet rs) throws Exception {
        OrderDetail orderDetail = new OrderDetail();
//          orderDetail.setId(rs.getInt("id"));
//          orderDetail.setOrderId(rs.getInt("orderId"));
//                                                  orderDetail.setProduct((Product)
productDao.getProductById(rs.getInt("productId")));
//          orderDetail.setProductId(rs.getInt("productId"));
//          orderDetail.setQuantity(rs.getInt("quantity"));
//          orderDetail.setCost(rs.getFloat("cost"));
        return orderDetail;
    }

    public void add(OrderDetail detail) throws SQLException {//保存订单详情
        Integer id=0;
        String sql=" insert into order_detail(orderId,productId,quantity,cost)
values(?,?,?,?) ";
        try {
            Object                                                  param[]=new
Object[]{detail.getOrderId(),detail.getProduct().getId(),detail.getQuantity(),detail.get
Cost()};
            id=this.executeInsert(sql,param);
            detail.setId(id);
        } catch (Exception e) {
            this.closeResource();
            e.printStackTrace();
        }
    }
}
```