武汉纺织大学

Java应用开发课程设计

**代码生成器开发**

**学 院： 数学与计算机学院**

**班 级： 软件工程11801**

**姓 名： 何琛**

**学 号： 1804240725**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年6月20日**

目 录

[1 需求分析 1](#_Toc2965)

[1.1实体类源文件 1](#_Toc6465)

[1.2mapper映射文件 1](#_Toc27274)

[1.3DAO接口文件 1](#_Toc17655)

[1.4总分析 ...............................................................................................................................1](#_Toc17655)

[2 系统设计 2](#_Toc15323)

[2.1 使用 2](#_Toc5309)

[2.2类.................................................................................................................................2](#_Toc5309)

[3 系统实现 9](#_Toc27433)

[3.1 项目结构 9](#_Toc1061)

[3.2 配置文件 9](#_Toc2679)

[3.2.1 daoTemp.ftl文件 9](#_Toc19162)

[3.2.2 Data.java文件 9](#_Toc31259)

[3.2.3 FreemakerToXML.Java文件 10](#_Toc7117)

[3.2.4 GenerateUtils.java文件 11](#_Toc6856)

[4 系统测试 14](#_Toc19341)

[5 系统总结 14](#_Toc30615)

# 1 需求分析

## 1.1实体类源文件

自动生成实体类文件，从数据库中读取表，生成实体类

## 1.2mapper映射文件

自动生成mapper映射文件，.xml格式，并且包含简单的jdbc操作

## 1.3DAO接口文件

自动生成DAO接口文件，包含不同对象的不同方法声明.

**1.4 总分析**

借助Freemarker机制可以方便的根据模板生成文件，同时也是组成代码生成器的核心部分。对于Freemarker而 言，其强调 数据模型 + 模板 = 文件 的思想，所以代码生成器重要的一个部分之一就是数据模型。在这里数据 模型共有两种形式组成：

数据库中表、字段等信息 针对这部分内容，可以使用元数据读取并封装到java实体类中 用户自定义的数据 为了代码生成器匹配多样的使用环境，可以让用户自定义的数据，并且以key-value的形式配置到properties 文件中

# **2 系统设计**

## **2.1使用**

**执行程序，自动生成三个文件夹到d:/ftl/目录下**

## 2.2 类

Data类，对获取的表进行转换

DataStruct类，对表的属性进行获取

FreeMakerToXml类，使用freemaker技术 调用相关类进行转化

GenerateUtils.java类，主程序类，连接数据库，获取表，调用类模板执行程序，完成功能，生成相关文件

## 2.3 流程图

### 2.3.1 登陆

登录操作需要验证用户名，密码，只有当用户名和密码同时存在即用户存在的情况下才能成功登录系统，否则会提示错误信息，具体流程如图2-6所示。

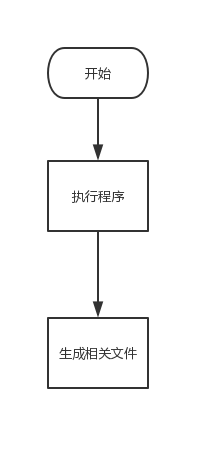


图2-2 流程图

# **3 系统实现**

## 3.1 项目结构

## 

## 3.2 配置文件



### 3.2.1 daoTemp.ftl文件

该文件为dao模板文件，使用freemaker技术

### package ${packetageDAO};

### import ${entityType};

### import org.apache.ibatis.annotations.Param;

### public interface ${entityName}Dao {

### Long save(${entityName} ${entityNameLow});

3.2.2Data.java文件

对数据库中数据进行处理

public class Data {

private String nativeColumn;

private String convertColumn;

public String getNativeColumn() {

return nativeColumn;

}

public void setNativeColumn(String nativeColumn) {

this.nativeColumn = nativeColumn;

}

public String getConvertColumn() {

return convertColumn;

}

public void setConvertColumn(String convertColumn) {

this.convertColumn = convertColumn;

}

}

3.2.3 DataStrute.java文件

对数据的属性进行加工，封装，便于模板进行调用

public class DataStrute {

private String type;

private String column;

private String pColumn;

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getColumn() {

return column;

}

public void setColumn(String column) {

this.column = column;

}

public String getpColumn() {

return pColumn;

}

public void setpColumn(String pColumn) {

this.pColumn = pColumn;

}

}

3.2.4 FreemakerToXML.Java文件

package ftl;

import freemarker.cache.FileTemplateLoader;

import freemarker.cache.TemplateLoader;

import freemarker.template.Configuration;

import freemarker.template.ObjectWrapper;

import freemarker.template.Template;

import java.io.\*;

import java.util.Map;

public class FreeMakerToXML {

private static Configuration configuration = null;

private static Template t = null;

public static final String xmlTempPath = "xmlTemp.ftl";

public static final String daoTempPath = "daoTemp.ftl";

public static final String pojoTempPath = "pojoTemp.ftl";

static {

try {

configuration = new Configuration();

configuration.setDefaultEncoding("UTF-8");

TemplateLoader templateLoader = new FileTemplateLoader(new File("src/main/java/com/ftl/template"));

configuration.setTemplateLoader(templateLoader);

} catch (IOException e) {

e.printStackTrace();

}

}

public static void createWord(Map<String,Object> dataMap, String fileName, String templte){

File outFile = new File("d:/ftl/"+fileName); //生成文件的路径

try(Writer out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(outFile)));) {

t = configuration.getTemplate(templte); //文件名

t.process(dataMap, out, ObjectWrapper.BEANS\_WRAPPER);

} catch (Exception e1) {

e1.printStackTrace();

}

}

}

3.2.5 GenerateUtils.java文件

主文件 ，数据库的连接，各种模板的使用，最终生成文件

package com.ftl;

import java.sql.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class GenerateUtils {

private static String driverClass = "com.mysql.jdbc.Driver";

private static String connectionURL = "jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&zeroDateTimeBehavior=convertToNull&autoReconnect=true&characterEncoding=utf-8&useSSL=false";

private static String userName="root";

private static String passwd = "123456";

private static String packetageDAO = "com.ftl.dao";

private static String packetagePojo = "com.ftl.vo";

public static void main(String[] args) {

// 第二个参数穿入null代表将库下所有的表都生成相应文件，传入具体表明可生成具体表对应的文件

getData(getConnection(),"student");

}

public static Connection getConnection(){

Connection conn = null;

try {

Class.forName(driverClass); //classLoader,加载对应驱动

conn = (Connection) DriverManager.getConnection(connectionURL, userName, passwd);

} catch (Exception e) {

e.printStackTrace();

}

return conn;

}

public static String underlineToCamel(String param) {

if (param == null || "".equals(param.trim())) {

return "";

}

int len = param.length();

StringBuilder sb = new StringBuilder(len);

for (int i = 0; i < len; i++) {

char c = param.charAt(i);

if (c == '\_') {

if (++i < len) {

sb.append(Character.toUpperCase(param.charAt(i)));

}

} else {

sb.append(c);

}

}

return sb.toString();

}

public static void getData(Connection conn, String tableName){

try {

DatabaseMetaData databaseMetaData = conn.getMetaData();

tableName = tableName != null ? tableName : "%";

ResultSet colRet = databaseMetaData.getTables(null, "%", tableName, new String[] {"TABLE"});

while(colRet.next()) {

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

String name = colRet.getString("TABLE\_NAME");

String camelCase = underlineToCamel(name);

String className =convert(camelCase);

dataMap = getColumnsData(databaseMetaData, dataMap, name,camelCase);

dataMap.put("tableName",name);

dataMap.put("entityName", className);

dataMap.put("entityNameLow", camelCase);

dataMap.put("entityType",packetagePojo+"."+className);

dataMap.put("packetageDAO",packetageDAO);

dataMap.put("packetagePojo",packetagePojo);

FreeMakerToXML.createWord(dataMap,"mapper/"+className+"Mapper.xml", FreeMakerToXML.xmlTempPath);

FreeMakerToXML.createWord(dataMap,"dao/"+className+"Dao.java", FreeMakerToXML.daoTempPath);

FreeMakerToXML.createWord(dataMap,"vo/"+className+".java", FreeMakerToXML.pojoTempPath);

}

}catch (SQLException e){

e.getErrorCode();

}

}

private static Map<String, Object> getColumnsData(DatabaseMetaData databaseMetaData, Map<String, Object> dataMap, String name,String camelCase) throws SQLException {

ResultSet columns = databaseMetaData.getColumns(null,"%", name,"%");

List<Data> dataList = new ArrayList<>();

List<DataStrute> dataStruteList = new ArrayList<>();

while(columns.next()) {

String column = columns.getString("COLUMN\_NAME");

Data data = new Data();

data.setNativeColumn(column);

String camelClonmn = underlineToCamel(column);

data.setConvertColumn(camelClonmn);

dataList.add(data);

String columnType = columns.getString("TYPE\_NAME");

String convertType = getJavaType(columnType);

DataStrute dataStrute = new DataStrute();

dataStrute.setColumn(underlineToCamel(column));

dataStrute.setpColumn(convert(camelClonmn));

dataStrute.setType(convertType);

dataStruteList.add(dataStrute);

dataMap.put("columnList",dataList);

dataMap.put("dataStrutes",dataStruteList);

}

return dataMap;

}

private static String getJavaType(String columnType) {

String convertType = null;

switch (columnType){

case "BIGINT" :

convertType = "Long";

break;

case "VARCHAR":

convertType = "String";

break;

case "TIMESTAMP":

convertType = "Date";

break;

case "TINYINT":

case "INT":

convertType = "Integer";

break;

case "DECIMAL":

convertType = "Double";

break;

default:

convertType = "String";

}

return convertType;

}

private static String convert(String nativeString){

String camelCase = underlineToCamel(nativeString);

char[] cs=camelCase.toCharArray();

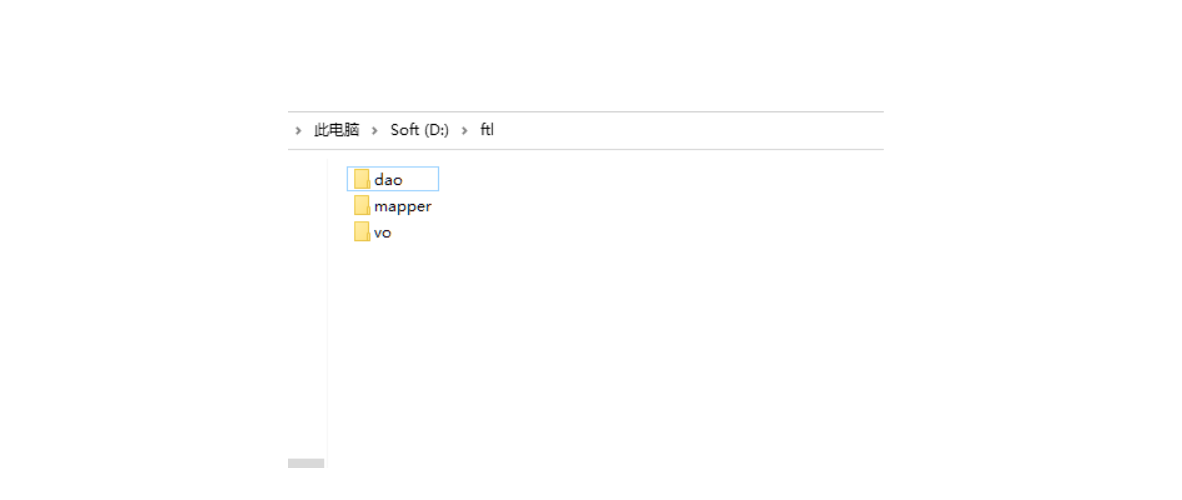
cs[0]-=32;

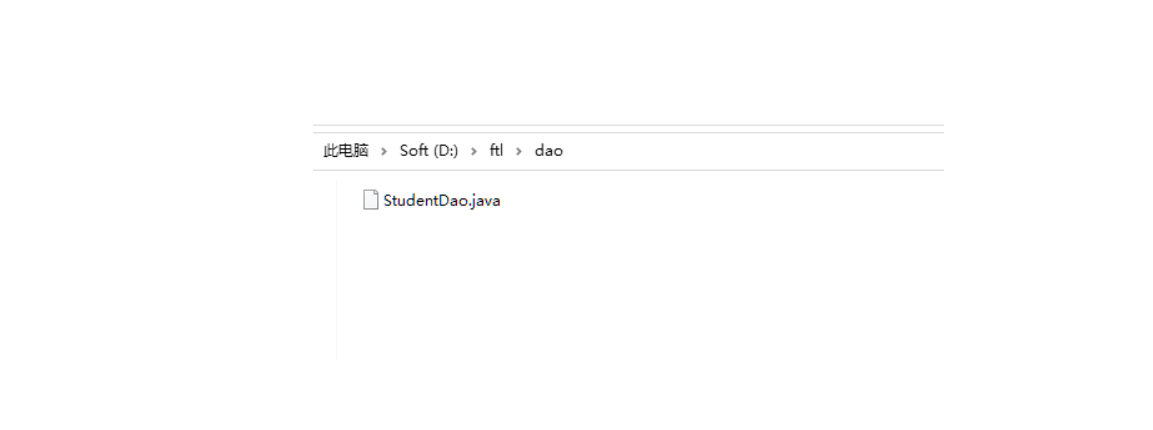
return String.valueOf(cs);

}

}

# **4 系统测试**





# **5 系统总结**

做的功能比较少，没有达到完整的MyBtis结构的完全操作，可视化的界面没有，不够人性化