**【数字钱包平台】**

**开发指导**

**文档修订记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修订人 | 修订说明 | 审批人 | 审批日期 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目 录**

[1 引言 4](#_Toc72086313)

[1.1 编写目的 4](#_Toc72086314)

[1.2 目标用户 4](#_Toc72086315)

[1.3 术语定义 4](#_Toc72086316)

[2 技术栈 4](#_Toc72086317)

[3 文件结构 5](#_Toc72086318)

[3.1 工程结构 5](#_Toc72086319)

[3.1.1 支付联机应用包结构 6](#_Toc72086320)

[3.1.2 支付批量应用包结构 6](#_Toc72086321)

[3.1.3 前端页面目录结构 7](#_Toc72086322)

[4 业务流程开发 7](#_Toc72086323)

[4.1 联机接入业务 7](#_Toc72086324)

[4.2 联机接出业务-行内系统发起调用 8](#_Toc72086325)

[4.3 联机接出业务-管理端页面发起调用 9](#_Toc72086326)

[4.4 批量对账业务 9](#_Toc72086327)

[4.4.1 创建任务组 9](#_Toc72086328)

[4.4.2 定义任务组 10](#_Toc72086329)

[4.4.3 任务实现类 10](#_Toc72086330)

[5 管理页面开发 12](#_Toc72086331)

[5.1 代码生成 12](#_Toc72086332)

[5.1.1 选择数据表 12](#_Toc72086333)

[5.1.2 修改生成配置 13](#_Toc72086334)

[5.1.3 生成代码 15](#_Toc72086335)

# 引言

## 编写目的

本文档旨在指导开发人员基于本平台开发业务功能。

## 目标用户

本系统设计、开发人员。

## 术语定义

|  |  |  |
| --- | --- | --- |
| 序号 | 术语 | 术语说明 |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

# 技术栈

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 内容 | 选型 | 版本 | 备注 |
| 微服务 | 服务网关 | Spring Cloud Gateway | 2.2.6.RELEASE |  |
| 认证中心 | - |  | 基于token的用户校验 |
| 注册中心 | nacos-discovery | 2.2.5.RELEASE |  |
| 配置中心 | nacos-config | 2.2.5.RELEASE |  |
| 服务调用 | openfeign | 10.10.1 |  |
| 负载均衡 | ribbon | 2.3.0 |  |
| 熔断和降级 | Sentinel | 1.5.18 |  |
| 服务监控 | actuator | 2.3.7.RELEASE |  |
| 服务监控UI | Spring Boot Admin | 2.3.1 |  |
| 链路追踪 | -- | -- | 本系统不提供，支持集成skywalking等 |
| 日志收集 | -- | -- | 本系统不提供，支持集成ELK |
| 缓存 | 缓存服务 | Redis  redisson | 6.2.0  3.15.4 |  |
| 数据库 | 书库服务 | Mysql | >5.7 |  |
| 动态数据源 | dynamic-datasource | 3.2.1 |  |
| 连接池 | druid | 1.2.4 |  |
| 持久化 | mybatis | 3.5.5 |  |
| 异步队列 | 队列 | rocketmq | 4.6.0 |  |
| 前端 | 部署容器 | Nginx | - |  |
| 语言 | Vue | 2.6.x | 2.6.x |
| 组件 | Element | 2.14.x |  |

# 文件结构

## 工程结构

dcwlt

├── dcwlt-ui // 前端

├── dcwlt-gateway // 应用网关

├── dcwlt-gateway // 城银清-互联互通网关

├── dcwlt-auth // 认证中心

├── dcwlt-api // 接口模块

│ └── dcwlt-api-system // 系统接口

├── dcwlt-common // 通用模块

│ └── dcwlt-common-core // 核心模块

│ └── dcwlt-common-datascope // 权限范围

│ └── dcwlt-common-datasource // 多数据源

│ └── dcwlt-common-log // 日志记录

│ └── dcwlt-common-redis // 缓存服务

│ └── dcwlt-common-security // 安全模块

│ └── dcwlt-common-mq // 消息队列

│ └── dcwlt-common-pay // 支付业务基础

│ └── dcwlt-common-swagger // 系统接口

├── dcwlt-modules // 应用

│ └── dcwlt-system // 系统管理后台

│ └── dcwlt-gen // 代码生成

│ └── dcwlt-job // 定时任务

│ └── dcwlt-monitor // 微服务监控

│ └── dcwlt-pay-online // 支付业务联机应用

│ └── dcwlt-pay-batch // 支付业务批量应用

├── dcwlt-mock // 挡板

│ └── core-baffle // 模拟核心应用挡板

│ └── dcps-client // 模拟城银清挡板

├──pom.xml // 公共依赖

├──sql // 数据库脚本 // 公共依赖

### 支付联机应用包结构

com.dcits.dcwlt.pay.online

├── DcwltPayOnlineApplication // 启动类

├── config // 配置

├── controller // 访问入口

│ └── front // 管理端页面请求入口

│ └── DcepController // 城银清-互联互通请求入口

│ └── DcwltController // 行内应用请求入口

├── event // 异常事件

├── flow // 业务流程

│ └── builder // 流程构建

│ └── receive // 接收流程

│ └── send // 发送流程

│ └── DcepTransInTradeFlow // 城银清-互联互通接入处理流程

│ └── DcwltTransInTradeFlow // 行内请求接出处理流程

├── mapper // 数据库Mapper

├── service // 服务实现

### 支付批量应用包结构

com.dcits.dcwlt.pay.batch

├── DcwltPayBatchApplication // 启动类

├── controller // 访问入口

├── event // 异常事件

├── mapper // 数据库Mapper

├── service // 服务实现

├── task // 批处理任务

### 前端页面目录结构

├── build // 构建相关

├── bin // 执行脚本

├── public // 公共文件

│ ├── favicon.ico // favicon图标

│ └── index.html // html模板

├── src // 源代码

│ ├── api // 所有请求

│ ├── assets // 主题 字体等静态资源

│ ├── components // 全局公用组件

│ ├── directive // 全局指令

│ ├── layout // 布局

│ ├── router // 路由

│ ├── store // 全局 store管理

│ ├── utils // 全局公用方法

│ ├── views // view

│ ├── App.vue // 入口页面

│ ├── main.js // 入口 加载组件 初始化等

│ ├── permission.js // 权限管理

│ └── settings.js // 系统配置

├── .editorconfig // 编码格式

├── .env.development // 开发环境配置

├── .env.production // 生产环境配置

├── .env.staging // 测试环境配置

├── .eslintignore // 忽略语法检查

├── .eslintrc.js // eslint 配置项

├── .gitignore // git 忽略项

├── babel.config.js // babel.config.js

├── package.json // package.json

└── vue.config.js // vue.config.js

# 业务流程开发

## 包路径规划

1. 数据库实体存放于 **dcwlt-common-pay** 模块的**com.dcits.dcwlt.pay.api.model** 包下。
2. 从页面调用联机业务的 Controller 放**dcwlt-pay-online** 模块的**com.dcits.dcwlt.pay.online.controller.front** 包下
3. 从页面发起的纯管理和查询的 Controller 放 **dcwlt-pay-batch** 模块的 **com.dcits.dcwlt.pay.batch.controller** 包下

## 联机接入业务

1. 创建业务处理流程

在com.dcits.dcwlt.pay.online.flow.receive 包下创建 处理流程实现类 FreeFrmt401RTradeFlow

装配处理流程

@Bean(name = ***FREEFRMT\_TRADE\_FLOW***)  
**public** TradeFlow freeFrmtTradeFlow() {  
 **return** EcnyTradeFlowBuilder.*get*()  
 .process(**this**::initFreeFrmt) *//封装保存自由格式报文* .response(**this**::packComConf) *//响应902* .errHandler(**this**::freeFrmtTradeErrHandler) *//异常处理* .build();  
}

1. 配置流程

在配置文件 applicationContext.xml 中注册流程，指定流程的交易码和对应的报问题，已经十分根据流水号防重。

<value>dcep.401.001.01,FreeFrmt401RTradeFlow,com.dcits.dcwlt.pay.api.domain.dcep.freefrmt.FreeFrmtDTO,Y</value>

## 联机接出业务-行内系统发起调用

1. 在DcwltController 创建接入口

@PostMapping(value = ApiConstant.***PAYCONVERT\_SERVICE\_NAME***)  
**public** ECNYRspDTO<PayConvertChnlRspDTO> payConvert(@RequestBody ECNYReqDTO<PayConvertChnlReqDTO> payConvertChnlReqDTO) {  
 **return ecnyTransInTradeFlow**.execute(payConvertChnlReqDTO, PayConvert227STradeFlow.***PAY\_CONVERT\_TRADE\_FLOW***);  
}

1. 在 send 包下创建业务处理流程实现类 PayConvert227STradeFlow

并按业务逻辑装配实现处理流程

@Bean(name = ***PAY\_CONVERT\_TRADE\_FLOW***)  
**public** TradeFlow payConvertTradeFlow() {  
 **return** EcnyTradeFlowBuilder.*get*()  
 .initTxn(**this**::initTxn) *//交易流水初始化* .process(**this**::businessCheck) *//业务检查* .process(**this**::coreProcess)*// 上核心* .process(**this**::sendToDcep)*// 发送汇款兑出请求报文* .response(**this**::response) *//响应报文* .errHandler(**this**::convertTradeErrHandler)*// 异常响应报文* .build();  
}

## 联机接出业务-管理端页面发起调用

1. 在front下创建接入口 FreefrmtController

*/\*\*  
 \* 发送自由格式  
 \** ***@param req*** *\** ***@return*** *\*/*@PreAuthorize(hasPermi = **"pay-online:freefrmt:pymtFrdmFmtMsgSnd"**)  
@PostMapping(**"/pymtFrdmFmtMsgSnd"**)  
**public** AjaxResult pymtFrdmFmtMsgSnd(@RequestBody Map<String, String> req) {  
 ECNYReqDTO<EcnyFreeFrmtReqDTO> ecnyReqDTO = ECNYReqDTO.*getInstanceWithDefaultHead*();  
  
 EcnyFreeFrmtReqDTO ecnyFreeFrmtReqDTO = **new** EcnyFreeFrmtReqDTO();  
 ecnyFreeFrmtReqDTO.setTlrNO(req.get(**"tlrNO"**));  
 ecnyFreeFrmtReqDTO.setInstdDrctPty(req.get(**"instdDrctPty"**));  
 ecnyFreeFrmtReqDTO.setMsgContext(req.get(**"msgContext"**));  
 ecnyReqDTO.setBody(ecnyFreeFrmtReqDTO);  
  
 ECNYRspDTO<FreeFrmtRspDTO> rspDTO = **ecnyTransInTradeFlow**.execute(ecnyReqDTO, FreeFrmt401STradeFlow.***FREEFRMT\_TRADE\_FLOW***);  
  
 **logger**.info(**"body的返回信息{}"**,rspDTO.getHead());  
 **if** (**"000000"**.equals(rspDTO.getHead().getRetCode())) {  
 **return** toAjax(1);  
 } **else** {  
  
 **logger**.error(**"响应失败原因：{}"**,rspDTO.getHead().getRetInfo());  
 **return** AjaxResult.*error*(**"自由格式发送失败"**);  
 }  
}

之后流程同上。

注意：联机接出业务兼容本系统前端发起请求和行内其他系统发起请求，如果是本系统的前端页面发起的请求，Controller收到请求后会组装报文，之后按其他系统调用处理方式统一处理。

## 批量对账业务

### 创建任务组



### 定义任务组

装配任务步骤和执行顺序



### 任务实现类

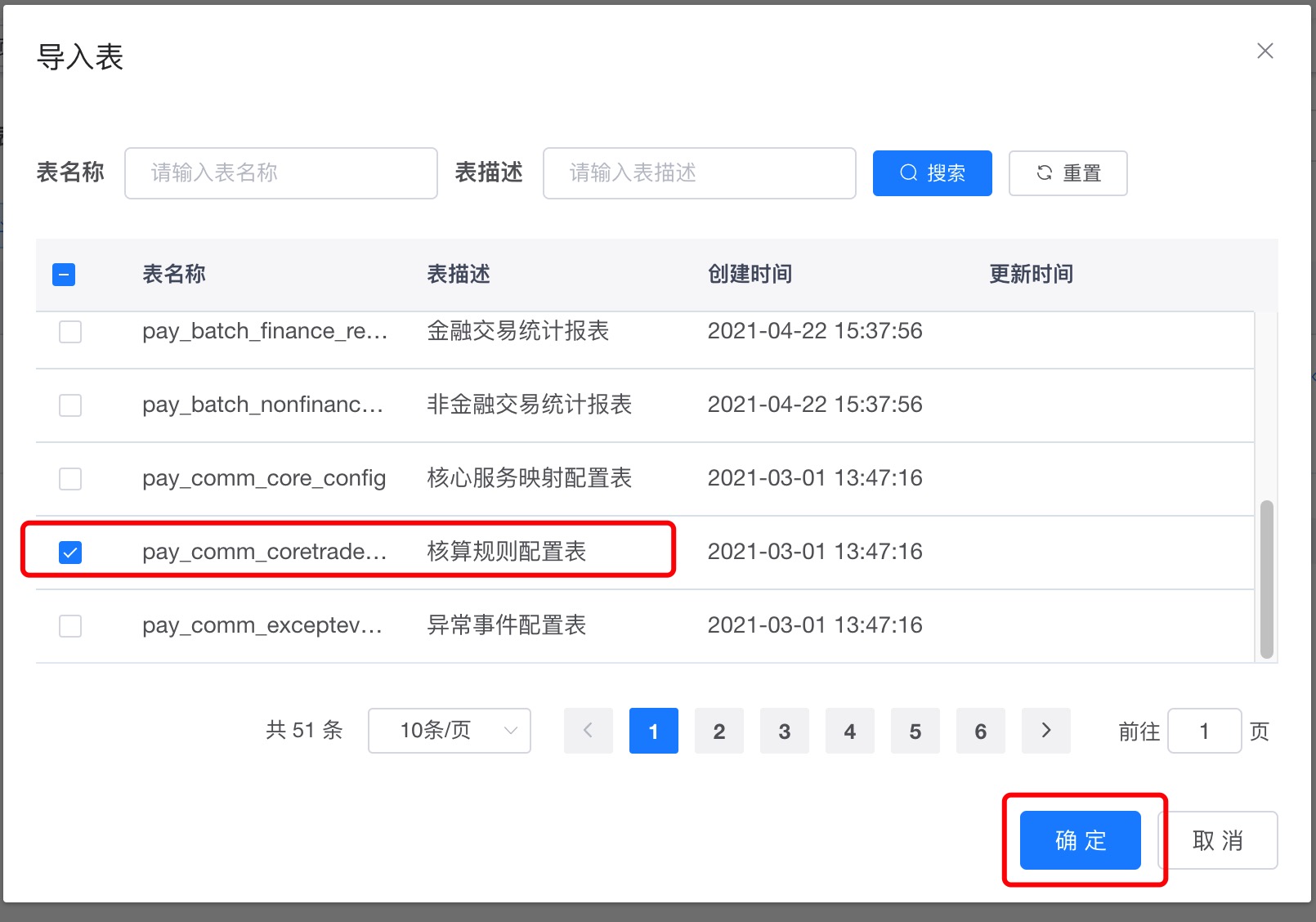
**package** com.dcits.dcwlt.pay.batch.task;  
  
  
**import** com.dcits.dcwlt.common.pay.enums.SettleTaskErrorEnum;  
**import** com.dcits.dcwlt.common.pay.enums.TaskCodeEnum;  
**import** com.dcits.dcwlt.common.pay.enums.TaskExecStatusEnum;  
**import** com.dcits.dcwlt.common.pay.enums.TaskGroupEnum;  
**import** com.dcits.dcwlt.common.pay.exception.SettleTaskException;  
**import** com.dcits.dcwlt.pay.api.model.SettleTaskExecDO;  
**import** com.dcits.dcwlt.pay.batch.service.ISettleTaskExecService;  
  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Service;  
  
*/\*\*  
 \* 抽数前置任务检查  
 \*/*@Service  
**public class** ImportTradeDataCheckTask **implements** ISettleTask {  
 **private static final** Logger ***logger*** = LoggerFactory.*getLogger*(ImportTradeDataCheckTask.**class**);  
   
 @Autowired  
 **private** ISettleTaskExecService **settleTaskExecService**;  
   
 */\*\*  
 \* 任务执行前初始化  
 \*/* @Override  
 **public void** initTask(SettleTaskExecDO taskExec) **throws** SettleTaskException {  
 *//检查抽取交易数据是否完成* }  
   
 */\*\*  
 \* 执行任务  
 \*/* @Override  
 **public void** runTask(SettleTaskExecDO taskExec) **throws** SettleTaskException {  
 ***logger***.info(**"开始执行任务，对账日期:{},对账批次:{}"**, taskExec.getSettleDate(), taskExec.getBatchId());  
 SettleTaskExecDO execTask = **settleTaskExecService**.queryTaskExecByCode(taskExec.getSettleDate(), TaskGroupEnum.***CHK\_DATA***.getCode(),  
 TaskCodeEnum.***CD001***.getCode(), taskExec.getBatchId());  
 **if** (execTask == **null**) {  
 **throw new** SettleTaskException(SettleTaskErrorEnum.***BC0016***.getCode(), SettleTaskErrorEnum.***BC0016***.getDesc());  
 }  
 **if** (!execTask.getExecState().equals(TaskExecStatusEnum.***SUCC***.getCode())) {  
 **throw new** SettleTaskException(SettleTaskErrorEnum.***BC0017***.getCode(), SettleTaskErrorEnum.***BC0017***.getDesc());  
 }  
 }  
   
}

# 管理页面开发

## 代码生成

### 选择数据表





### 修改生成配置



基本信息配置



配置字段信息，包括插入字段、可修改编辑的字段，查询列表展示的字段，查询条件字段，如果字段是字典类型的选择字典。显示类型可以修改为日期时间、单选多选等控件类型。



修改生成配置

**生成模板：单表还是主子表**

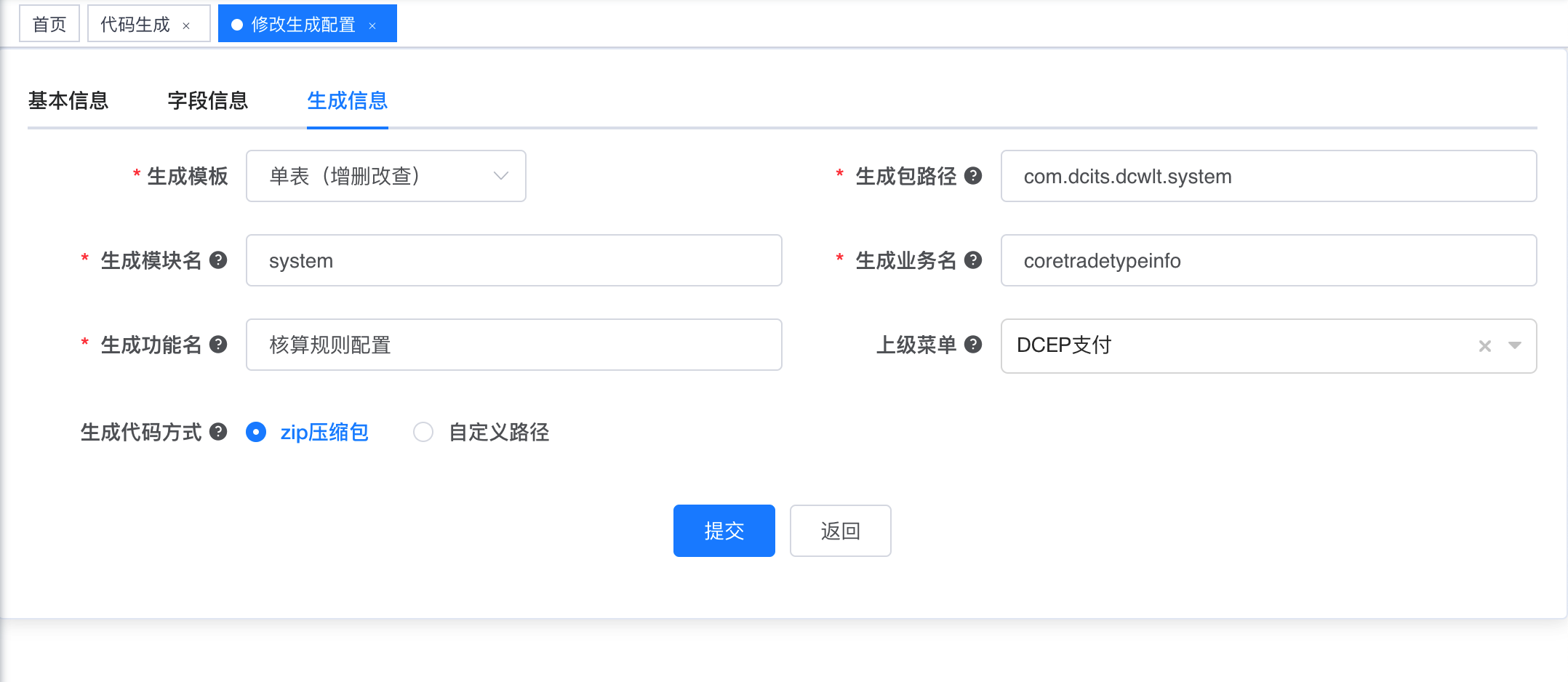
**生成包路径：后端代码包路径**

**生成模块名：模块名**

**生成业务名：功能id**

**生成功能名：功能描述**

**上级菜单：父级菜单**



### 生成代码



生成代码结构

main 后端代码

vue 前端代码

coretradetypeinfoMenu.sql 菜单数据库脚本