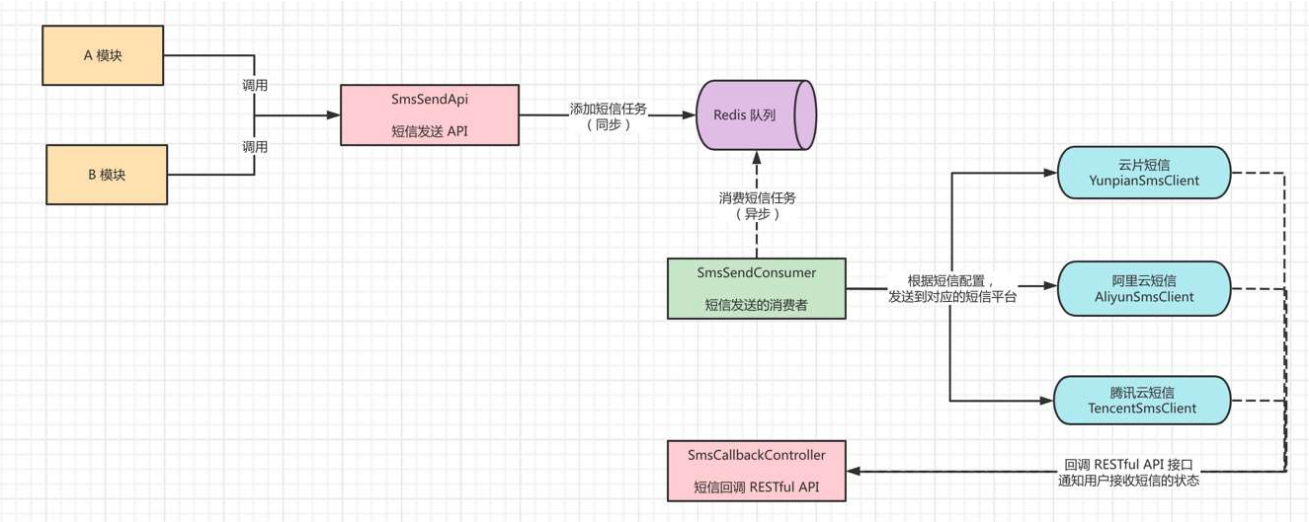


🔗 短信配置

本章节，介绍项目的短信功能。该功能提供统一的短信 API 给其它模块，使它们可以快速接入短信功能，无需关心不同短信平台的具体对接。
短信采用异步发送，基于 [消息队列](#)，如下图所示：



该功能由 `yudao-module-system` 模块实现，其中：

- `service/sms` 📁：短信【业务】，提供短信渠道、模板的配置，短信日志的查看，短信的发送等功能
- `frameowrk/sms` 📁：短信【组件】，封装阿里云、腾讯云等短信平台的客户端。

1. 表结构

system_sms_channel	system_sms_template	system_sms_log
- id: bigint(0) - signature: varchar(10) - code: varchar(63) - status: tinyint(0) - remark: varchar(255) - api_key: varchar(128) - api_secret: varchar(128) - callback_url: varchar(255) - creator: varchar(64) - create_time: datetime(0) - updater: varchar(64) - update_time: datetime(0) - deleted: bit(1)	- id: bigint(0) - type: tinyint(0) - status: tinyint(0) - code: varchar(63) - name: varchar(63) - content: varchar(255) - params: varchar(255) - remark: varchar(255) - api_template_id: varchar(63) - channel_id: bigint(0) - channel_code: varchar(63) - creator: varchar(64) - create_time: datetime(0) - updater: varchar(64) - update_time: datetime(0) - deleted: bit(1)	- id: bigint(0) - channel_id: bigint(0) - channel_code: varchar(63) - template_id: bigint(0) - template_code: varchar(63) - template_type: tinyint(0) - template_content: varchar(255) - template_params: varchar(255) - api_template_id: varchar(63) - mobile: varchar(11) - user_id: bigint(0) - user_type: tinyint(0) - send_status: tinyint(0) - send_time: datetime(0) - send_code: int(0) - send_msg: varchar(255) - api_send_code: varchar(63) - api_send_msg: varchar(255) - api_request_id: varchar(255) - api_serial_no: varchar(255) - receive_status: tinyint(0) - receive_time: datetime(0) - api_receive_code: varchar(63) - api_receive_msg: varchar(255)

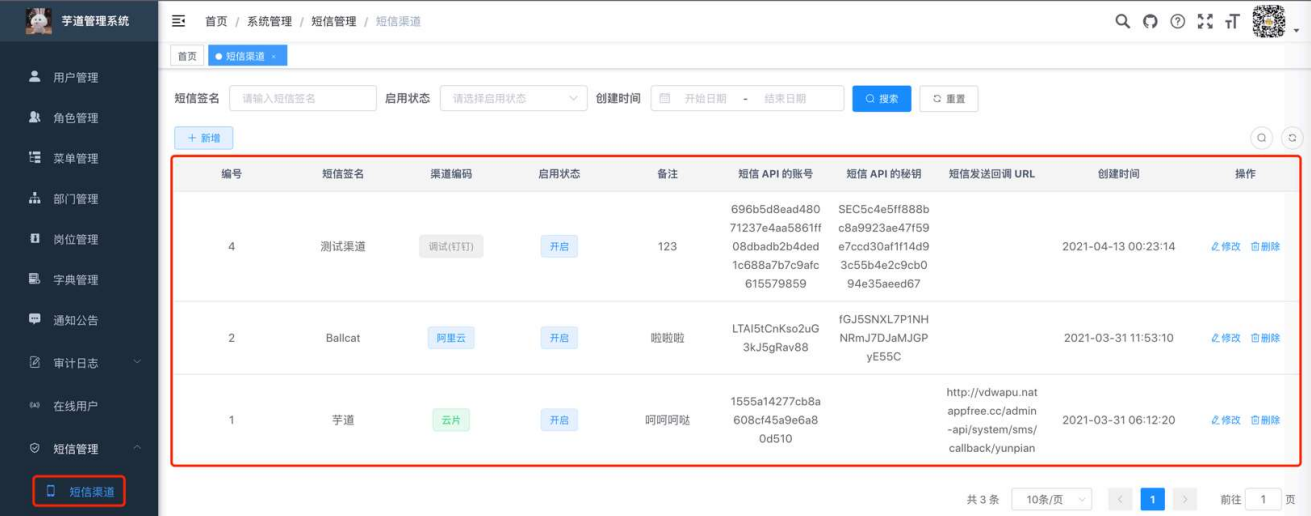
2. 短信配置

本小节，讲解如何配置短信功能，整个过程如下：

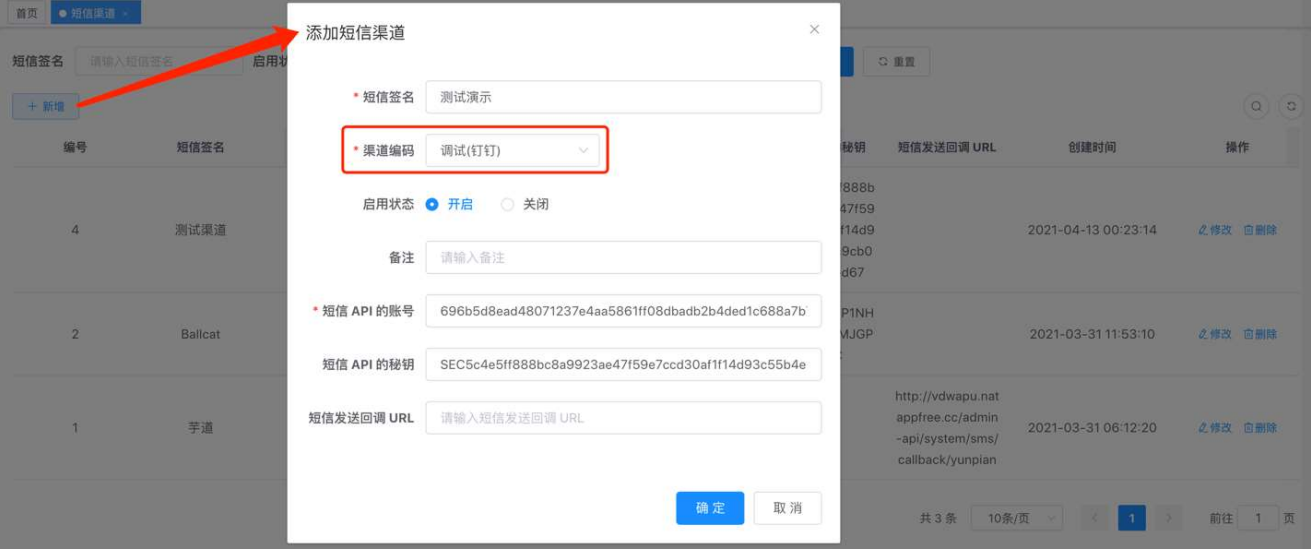
- 新建一个短信【渠道】，配置对应短信平台的账号
- 新建一个短信【模版】，配置对应短信平台的模板
- 测试该短信模板，查看对应的短信【日志】，确认是否发送成功

2.1 新建短信渠道

① 点击 [系统管理 -> 短信管理 -> 短信渠道] 菜单，查看短信渠道的列表。如下图所示：



② 点击 [新增] 按钮，选择渠道编码为【调试（钉钉）】，并填写信息如下图：



短信 API 的账号：696b5d8ead48071237e4aa5861ff08dbadb2b4ded1c688a7b7c9afc615579859
短信 API 的密钥：SEC5c4e5ff888bc8a9923ae47f59e7ccd30af1f14d93c55b4e2c9cb094e35aee

疑问 1：为什么选择渠道编码为【调试（钉钉）】？

该类型使用钉钉机器人来模拟短信发送，用于日常调试。

- 短信 API 的账号，对应机器人的 Webhook 的 `access_token` 参数
- 短信 API 的密钥，对应机器人的安全设置的加签

上图使用的配置，是芳芳自己的钉钉机器人。正式使用时，必须参考《钉钉开放平台——自定义机器人接入》[文档](#)，申请自己的专属机器人。

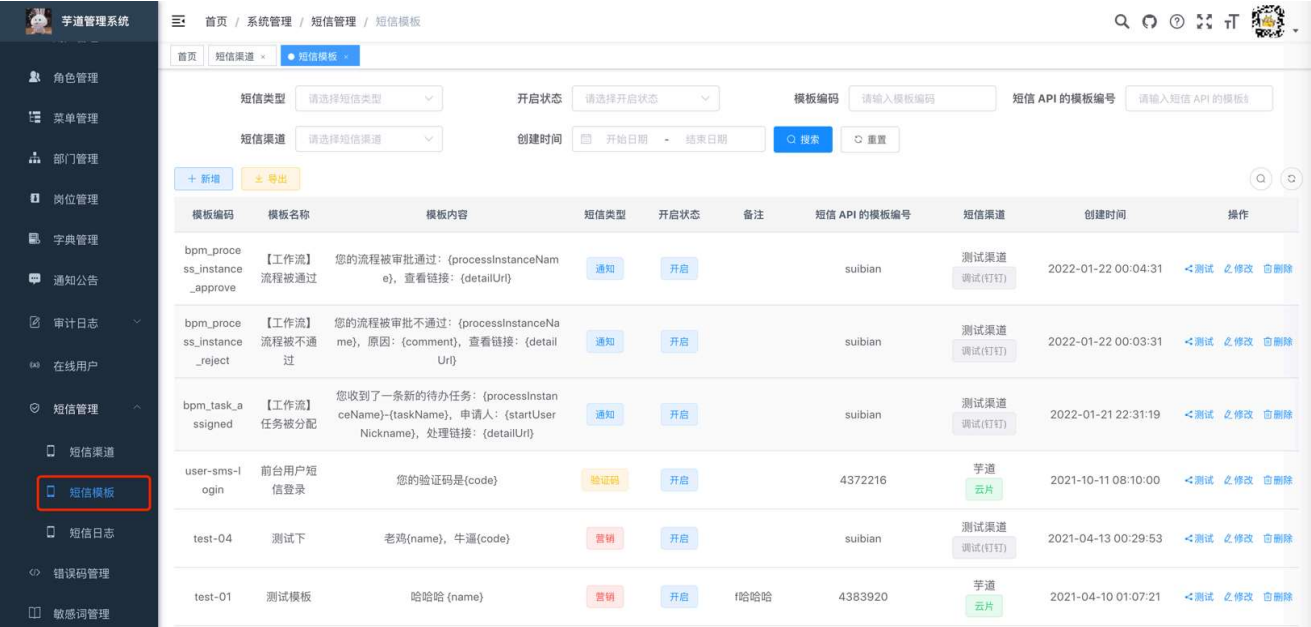
疑问 2：可以选择其它渠道编码吗？

当然可以，这里主要考虑部分同学暂时没有申请短信平台，所以使用【调试（钉钉）】渠道编码。

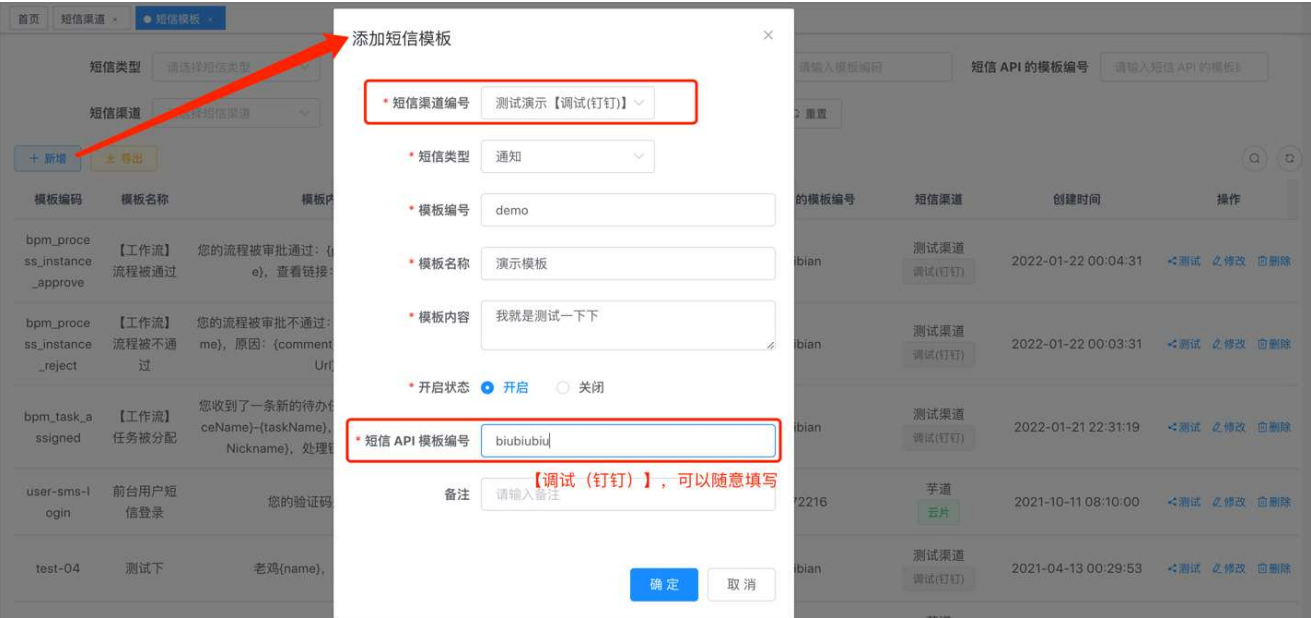
不同短信平台的配置，可见「6. 短信平台附录」小节。

2.2 新建短信模板

① 点击 [系统管理 -> 短信管理 -> 短信模板] 菜单，查看短信模板的列表。如下图所示：



② 点击 [新增] 按钮，选择刚创建的短信渠道，并填写信息如下图：



- 短信渠道编号：发送该短信模板时，使用的短信渠道，即使用哪个短信平台进行发送
- 模板编号：短信模板的唯一标识，使用短信 API 时，通过它标识使用的短信模板
- 模板内容：短信模板的内容，使用 `{var}` 作为占位符，例如说 `{name}`、`{code}` 等
- 短信 API 模板编号：短信平台的短信模板的编号，需要保证该模板在短信平台已经审核通过
- 开启状态：短信模板被禁用时，该短信模板将不发送短信，只记录短信日志

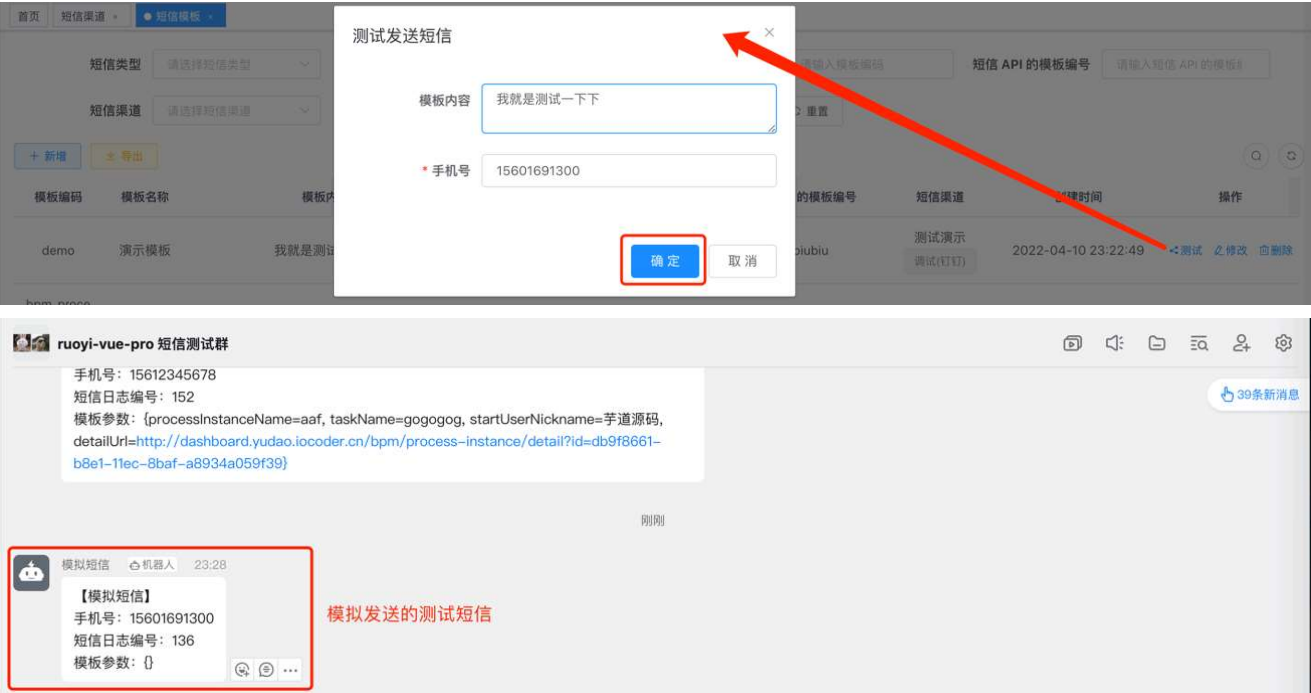
疑问：为什么设计短信模板的功能？

在一些场景下，需要修改短信模板所使用的短信平台。例如说：短信平台出现故障，或者切换短信平台等等。

此时，只需要修改短信模板的两个属性：短信渠道编号、短信 API 模板编号，无需重启应用。

2.3 查看短信日志

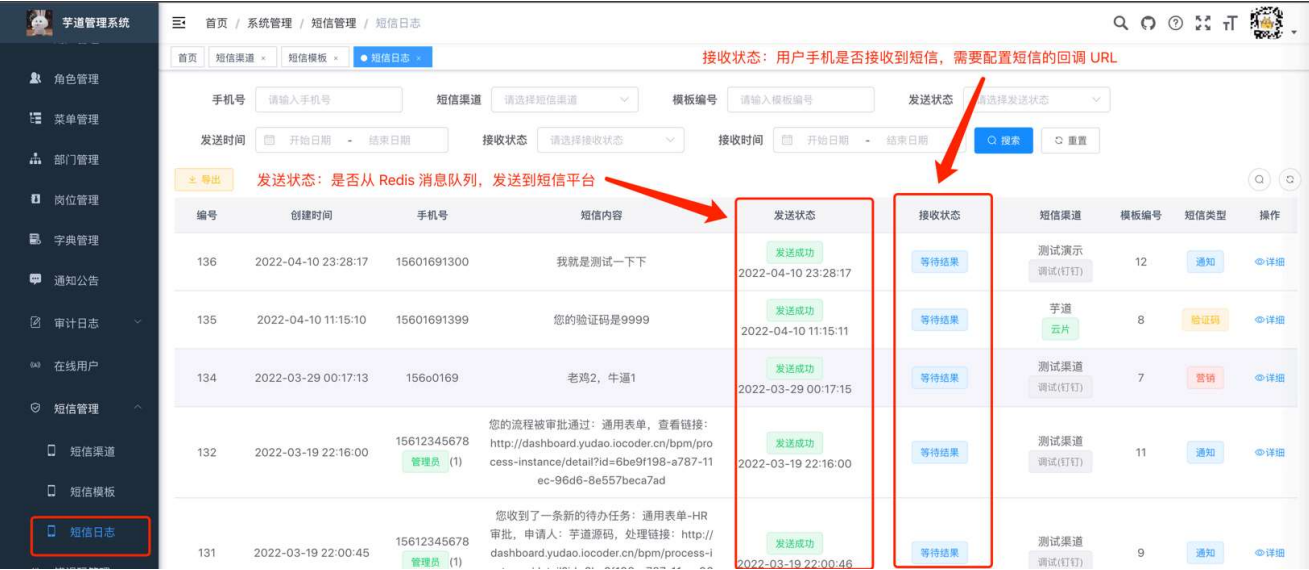
- ① 使用钉钉，扫码 [图片](#) 加入机器人所在的【ruoyi-vue-pro 短信测试群】，查看测试短信的模拟发送。
- ② 点击 [测试] 按钮，输入任一手机号，进行该短信模板的模拟发送。如下图所示：



友情提示：如果使用的短信渠道是阿里云、腾讯云等正式的短信平台，则会发送到填写的手机号中。例如说：



③ 点击 [系统管理 -> 短信管理 -> 短信日志] 菜单，可以查看到每条短信的发送状态、接收状态。如下图所示：



3. 短信发送

3.1 SmsSendApi

使用 [SmsSendApi](#) 进行短信的发送，支持多种用户类型。它的方法如下：

SmsSendApi.java

```
1 package cn.iocoder.yudao.module.system.api.sms;
2
3 import ...
4
5 /** 短信发送 API 接口 ... */
6
7 public interface SmsSendApi {
8
9     /**
10      * 发送单条短信给 Admin 用户
11      * 在 mobile 为空时, 使用 userId 加载对应 Admin 的手机号
12      *
13      * @param reqDTO 发送请求
14      * @return 发送日志编号
15      */
16     Long sendSingleSmsToAdmin(@Valid SmsSendSingleToUserReqDTO reqDTO);
17
18     /**
19      * 发送单条短信给 Member 用户
20      * 在 mobile 为空时, 使用 userId 加载对应 Member 的手机号
21      *
22      * @param reqDTO 发送请求
23      * @return 发送日志编号
24      */
25     Long sendSingleSmsToMember(@Valid SmsSendSingleToUserReqDTO reqDTO);
26
27 }
```

SmsSendSingleToUserReqDTO.java

```
1 package cn.iocoder.yudao.module.system.api.sms.dto.send;
2
3 import ...
4
5 /** 短信发送给 Admin 或者 Member 用户 ... */
6
7 @Data
8 public class SmsSendSingleToUserReqDTO {
9
10     /**
11      * 用户编号
12      */
13     private Long userId;
14
15     /**
16      * 手机号
17      */
18     private String mobile;
19
20     /**
21      * 短信模板编号
22      */
23     @NotEmpty(message = "短信模板编号不能为空")
24     private String templateCode;
25
26     /**
27      * 短信模板参数
28      */
29     private Map<String, Object> templateParams;
30
31 }
```

1.1 userId 和 mobile 二选一, 其中 mobile 的优先级更高

1.2 未传递 mobile 时, 会通过 userId 获取对应的 mobile

如果最终无法获取到 mobile 时, 会抛出异常

2. 短信模板中的 {var} 占位参数 需要保证每个参数都传递

3.2 实战案例

以工作流申请通过时, 发送短信为例子, 讲解 SmsSendApi 的使用。

① 引入 yudao-module-system-api 依赖, 如下图所示:

Project

ruoyi-vue-pro-2022b [yudao]

yudao-module-bpm-base

pom.xml

<dependencies>

<dependency>

<groupId>cn.iocoder.boot</groupId>

<artifactId>yudao-module-bpm-api</artifactId>

<version>\${revision}</version>

</dependency>

<dependency>

<groupId>cn.iocoder.boot</groupId>

<artifactId>yudao-module-system-api</artifactId>

<version>\${revision}</version>

</dependency>

② 新建对应的短信模板, 如下图所示:

短信渠道

短信模板

短信日志

短信类型

开启状态

模板编码

短信 API 的模板编号

短信渠道

创建时间

开始日期

结束日期

搜索

重置

模板编码

模板名称

模板内容

短信类型

开启状态

备注

短信 API 的模板编号

短信渠道

创建时间

操作

demo

演示模板

我就是测试一下下

通知

开启

biubiubiu

测试演示

2022-04-10 23:22:49

测试(订)

测试

修改

删除

bpm_proce

【工作流】

您的流程被审批通过: {processInstanceNam

通知

开启

suibian

测试渠道

2022-01-22 00:04:31

测试(订)

测试

修改

删除

③ 使用 Spring 注入 SmsSendApi Bean, 调用对应的短信发送方法。如下图所示:

BpmMessageServiceImpl.java

```
1 /** BPM 消息 Service 实现类 ... */
2
3 @Service
4 @Validated
5 @Slf4j
6 public class BpmMessageServiceImpl implements BpmMessageService {
7
8     @Resource
9     private SmsSendApi smsSendApi;
10
11     @Resource
12     private WebProperties webProperties;
13
14     @Override
15     public void sendMessageWhenProcessInstanceApprove(BpmMessageSendWhenProcessInstanceApproveReqDTO reqDTO) {
16         Map<String, Object> templateParams = new HashMap<>();
17         templateParams.put("processInstanceName", reqDTO.getProcessInstanceName());
18         templateParams.put("detailUrl", getProcessInstanceDetailUrl(reqDTO.getProcessInstanceId()));
19         smsSendApi.sendSingleSmsToAdmin(BpmMessageConvert.INSTANCE.convert(reqDTO.getStartUserId(), BpmMessageEnum.PROCESS_INSTANCE_APPROVE.getSmsTemplateCode(), templateParams));
20     }
21 }
```

BpmMessageEnum.java

```
1 package cn.iocoder.yudao.module.bpm.enums.message;
2
3 import ...
4
5 /** Bpm 消息的枚举 ... */
6
7 @AllArgsConstructor
8 @Getter
9 public enum BpmMessageEnum {
10
11     PROCESS_INSTANCE_APPROVE2(
12         smsTemplateCode: "bpm_process_instance_approve", // 流程任务被审批通过;
13         // 发送给申请人
14     ),
15     PROCESS_INSTANCE_REJECT(
16         smsTemplateCode: "bpm_process_instance_reject", // 流程任务被审批不通过时, 发送给申请人
17     ),
18     TASK_ASSIGNED(
19         smsTemplateCode: "bpm_task_assigned", // 任务被分配时, 发送给审批人
20     );
21
22     /**
23      * 短信模板的标识
24      * 关联 SmsTemplateDO 的 code 属性
25      */
26 }
```

① 注入 SmsSendApi Bean

② 构建短信模板的参数

发送短信

4. 验证码发送

4.1 SmsCodeApi

使用 `SmsCodeApi` 进行【验证码】短信的发送，例如说：用户手机验证码登录、用户忘记密码等等。它的方法如下：

SmsSendApi.java

package cn.iocoder.yudao.module.system.api.sms;

import ...

/** 短信发送 API 接口 ... */
public interface SmsSendApi {

 /**
 * 发送单条短信给 Admin 用户
 * 在 mobile 为空时，使用 userId 加载对应 Admin 的手机号
 * @param reqDTO 发送请求
 * @return 发送日志编号
 */
 Long sendSingleSmsToAdmin(@Valid SmsSendSingleToUserReqDTO reqDTO);

 /**
 * 发送单条短信给 Member 用户
 * 在 mobile 为空时，使用 userId 加载对应 Member 的手机号
 * @param reqDTO 发送请求
 * @return 发送日志编号
 */
 Long sendSingleSmsToMember(@Valid SmsSendSingleToUserReqDTO reqDTO);
}

SmsSendSingleToUserReqDTO.java

package cn.iocoder.yudao.module.system.api.sms.dto.send;

import ...

/** 短信发送给 Admin 或者 Member 用户 ... */
@Data
public class SmsSendSingleToUserReqDTO {

 /**
 * 用户编号
 */
 private Long userId;

 /**
 * 手机号
 */
 @Mobile
 private String mobile;

 /**
 * 短信模板编号
 */
 @NotEmpty(message = "短信模板编号不能为空")
 private String templateCode;

 /**
 * 短信模板参数
 */
 private Map<String, Object> templateParams;
}

1.1 userId 和 mobile 二选一，其中 mobile 的优先级更高
1.2 未传递 mobile 时，会通过 userId 获取对应的 mobile
如果最终无法获取到 mobile 时，会抛出异常

2. 短信模板中的 (var) 占位参数
需要保证每个参数都传递

验证码使用 `system_sms_code` 表进行存储，默认每天最多发送 10 条，每分钟发送 1 条，有效期为 10 分钟，可通过 `yudao.sms-code` 配置项进行自定义：

yudao-server

src
main
java
resources
admin-ui
static
application.yaml
application-dev.yaml
application-local.yaml
banner.txt
logback-spring.xml
test
pom.xml
yudao-ui-admin
yudao-ui-admin-vue3
yudao-ui-app-tmp

63

64 yudao:

65 info: <2 keys>

66 web: <3 keys>

67 swagger: <4 keys>

68 captcha: <3 keys>

69 codegen: <2 keys>

70 error-code: <1 key>

71 tenant: <3 keys>

126 sms-code: # 短信验证码相关的配置项

127 expire-times: 10m

128 send-frequency: 1m

129 send-maximum-quantity-per-day: 10

130 begin-code: 9999 # 这里配置 9999 的原因是，测试方便。

131 end-code: 9999 # 这里配置 9999 的原因是，测试方便。

4.2 实战案例

以会员用户手机验证码登录为例子，讲解 `SmsCodeApi` 的使用。

① 引入 `yudao-module-system-api` 依赖，如下图所示：

yudao-module-member

yudao-module-member-api
yudao-module-member-impl
src
pom.xml
pom.xml
yudao-module-pay

20

21 <dependencies>

22 <dependency>

23 <groupId>cn.iocoder.boot</groupId>

24 <artifactId>yudao-module-member-api</artifactId>

25 <version>\${revision}</version>

26 </dependency>

② 新建对应的短信模板，如下图所示：

短信模板

短信日志

user-sms-l

前台用户短信登录

您的验证码是(code)

验证码

开始

4372216

手道

2021-10-11 08:10:00

测试 修改 删除

③ 在 `SmsSceneEnum` 中，枚举会员用户的手机号登录的场景，如下图所示：

```
/** 用户短信验证码发送场景的枚举 ...*/  
@Getter  
@AllArgsConstructor  
public enum SmsSceneEnum implements IntArrayValuable {  
  
    MEMBER_LOGIN(scene: 1, templateCode: "user-sms-login", description: "会员用户 - 手机号登陆"),  
    MEMBER_UPDATE_MOBILE(scene: 2, templateCode: "user-sms-reset-password", description: "会员用户 - 修改手机"),  
    MEMBER_FORGET_PASSWORD(scene: 3, templateCode: "user-sms-update-mobile", description: "会员用户 - 忘记密码");  
}
```

④ 使用 Spring 注入 `SmsCodeApi` Bean，调用对应的短信验证码的发送与使用方法。如下图所示：

```
MemberAuthServiceImpl.java  
90  
91 @Resource  
92 private SmsCodeApi smsCodeApi; ① 使用 Spring 注入 SmsCodeApi Bean  
93  
94 @Override  
95 public void sendSmsCode(Long userId, AppAuthSendSmsReqV0 reqV0) {  
96     smsCodeApi.sendSmsCode(AuthConvert.INSTANCE.convert(reqV0).setCreateIp(getClientIP()));  
97 }  
98  
99 ② 登录前，使用 SmsCodeApi 发送验证码  
100  
101 @Override  
102 @Transactional  
103 public String smsLogin(AppAuthSmsLoginReqV0 reqV0, String userIp, String userAgent) {  
104     // 校验验证码  
105     smsCodeApi.useSmsCode(AuthConvert.INSTANCE.convert(reqV0), SmsSceneEnum.MEMBER_LOGIN.getScene(), userIp);  
106     ③ 登录时，使用 SmsCodeApi 验证并使用验证码  
107  
108     // 获得获得注册用户  
109     MemberUserDO user = userService.createUserIfAbsent(reqV0.getMobile(), userIp);  
110     Assert.notNull(user, errorMsgTemplate: "获取用户失败，结果为空");  
111  
112     // 执行登陆  
113     LoginUser loginUser = AuthConvert.INSTANCE.convert(user);  
114  
115     // 缓存登录用户到 Redis 中，返回 sessionId 编号  
116     return createUserSessionAfterLoginSuccess(loginUser, LoginLogTypeEnum.LOGIN_SMS, userIp, userAgent);  
117 }  
118
```

5. 短信客户端

`framework/sms` 短信【组件】，对接阿里云、腾讯云等短信平台，提供统一的短信客户端，提供给 `service/sms` 短信【业务】模块来调用。

5.1 SmsClient

`SmsClient` 接口，定义短信客户端的方法。代码如下：

```
/** 短信客户端，用于对接各短信平台的 SDK，实现短信发送等功能 ...*/
public interface SmsClient {

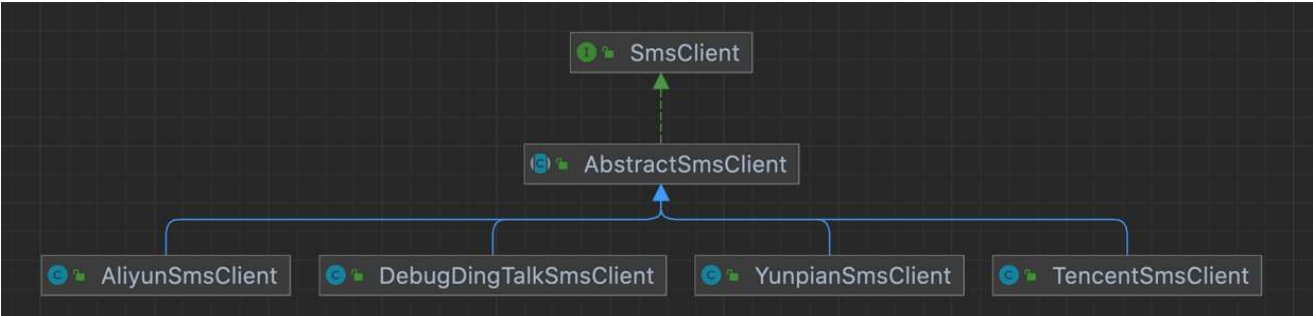
    /** 获得渠道编号 ...*/
    Long getId();

    /** 发送消息 ...*/
    SmsCommonResult<SmsSendRespDTO> sendSms(Long logId, String mobile, String apiTemplateId,
        List<KeyValue<String, Object>> templateParams); ① 发送短信

    /** 解析接收短信的接收结果 ...*/
    List<SmsReceiveRespDTO> parseSmsReceiveStatus(String text) throws Throwable; ② 解析短信回调结果

    /** 查询指定的短信模板 ...*/
    SmsCommonResult<SmsTemplateRespDTO> getSmsTemplate(String apiTemplateId); ③ 校验短信平台的模板是否已经审核通过
}
```

每个短信平台，都对应一个 SmsClient 实现类。



5.2 对接其它短信平台

如果你想要对接其它短信平台，自定义一个 SmsClient 实现类，并使用 `SmsClientFactoryImpl` 进行创建。代码如下：

```
private AbstractSmsClient createSmsClient(SmsChannelProperties properties) {
    SmsChannelEnum channelEnum = SmsChannelEnum.getByCode(properties.getCode());
    Assert.notNull(channelEnum, String.format("渠道类型(%s) 为空", channelEnum));
    // 创建客户端
    switch (channelEnum) {
        case ALIYUN: return new AliyunSmsClient(properties);
        case YUN_PIAN: return new YunpianSmsClient(properties);
        case DEBUG_DING_TALK: return new DebugDingTalkSmsClient(properties);
        case TENCENT: return new TencentSmsClient(properties);
    }
    // 创建失败，错误日志 + 抛出异常
    log.error("[createSmsClient][配置({}) 找不到合适的客户端实现]", properties);
    throw new IllegalArgumentException(String.format("配置(%s) 找不到合适的客户端实现", properties));
}
```

6. 短信平台附录

一般情况下，建议接入 2-3 个短信平台，避免某个短信平台故障时，影响业务的正常运行。例如说，手机验证码的短信平台 A 故障时，赶紧将短信验证码切换到短信平台 B 上，否则用户将无法登录或是注册。

6.1 阿里云

- ① 短信 API 的账号、密钥，可通过 [阿里云 —— AccessKey](#) 获取。
- ② 短信发送回调 URL，可通过 [阿里云 —— 短信服务 —— 通用设置](#) 配置。

6.2 腾讯云

① 短信 API 的账号、密钥，可通过 [腾讯云 —— API 密钥管理](#) 获取。

注意!!!

腾讯云需要额外使用 [SDKAppID](#) 参数，它的账号需要采用 `secretId SDKAppID` 格式。

例如说：在“API 密钥管理”获得了 `SecretId` 为 `A`，`SecretKey` 为 `B`，
在“SDKAppID”获得了 `SDKAppID` 为 `18`，则配置短信 API 的账号为 `A 18`，短信 API 的密钥为 `B`。

② 短信发送回调 URL，可通过 [腾讯云 —— 短信 —— 基础配置](#) 配置。

← [公众号统计](#)

[邮件配置](#) →



Theme by [Vdoing](#) | Copyright © 2019-2024 芋道源码 | MIT License