SpringBoot【短信微服务】

作者：haxLook
原文链接：https://ld246.com/article/1617152922638
来源网站：链滴

SpringBoot搭建短信微服务

SpringBoot快速搭建Spring工程，所以非常适合进行搭建微服务，本片中主要介绍的是使用阿里大进行短信为服务的搭建。

SpringBoot和activeMQ的整合

MQ是消息中间件，是一种在分布式系统中应用程序借以传递消息的媒介，常用的有ActiveMQ，RabbtMQ，kafka。ActiveMQ是Apache下的开源项目，完全支持JMS1.1和J2EE1.4规范的JMS Provider现。

特点：

1. 支持多种语言编写客户端
2. 对spring的支持，可以和Spring进行整合
3. 支持多种传输协议：TCP,SSL,NIO,UDP等
4. 支持AJAX

消息形式：

5. 点对点（queue）
6. 一对多（topic）

- activeMQ下载地址 http://activemq.apache.org(案例使用的是activeMQ-5.15.3的版本)

1. 使用工具构建Springboot工程，可参考链接： https://blog.csdn.net/hax1435501085/article/etails/105146038
2. 配置文件pom.xml

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
<!-- 消息队列连接池-->
<dependency>
    <groupId>org.messaginghub</groupId>
    <artifactId>pooled-jms</artifactId>
</dependency>
```

这里注意说明，如果SpringBoot是2.1.X版本以上使用以上的配置，如果是使用以下配置会出现找不到msMessagingTemplate;

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
<!--消息队列连接池-->
<dependency>
   <groupId>org.apache.activemq</groupId>
   <artifactId>activemq-pool</artifactId>
</dependency>
```

3. application.yml文件配置

```
server:
  port: 8080
spring:
  activemq:
     # 看自己activeMQ安装地方配置 我在这里使用的window安装
    broker-url: tcp://localhost:61616
    #broker-url: tcp://192.168.1.103：61616
    user: admin
    password: admin
    pool:
      enabled: true
      max-connections: 10
  jms:
    pub-sub-domain: false  #false为queues  true 为topics
  #服务名称
  application:
    name: boot-activemq

queueName: test-queue
topicName: test-topic
```

4. ActiveMQConfig.java文件

```
package com.example.demo.config;
import javax.jms.Queue;
import javax.jms.Topic;
import org.apache.activemq.ActiveMQConnectionFactory;
import org.apache.activemq.command.ActiveMQQueue;
import org.apache.activemq.command.ActiveMQTopic;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jms.config.DefaultJmsListenerContainerFactory;
import org.springframework.jms.config.JmsListenerContainerFactory;
/**
 * @author Administrator_hax
 */
@Configuration
public class ActiveMQConfig {
    @Value("${queueName}")
    private String queueName;

    @Value("${topicName}")
    private String topicName;

    @Value("${spring.activemq.user}")
    private String usrName;

    @Value("${spring.activemq.password}")
    private  String password;

    @Value("${spring.activemq.broker-url}")
```

```
    private  String brokerUrl;

    @Bean
    public Queue queue(){
        return new ActiveMQQueue(queueName);
    }

    @Bean
    public Topic topic(){
        return new ActiveMQTopic(topicName);
    }

    @Bean
    public ActiveMQConnectionFactory connectionFactory() {
        return new ActiveMQConnectionFactory(usrName, password, brokerUrl);
    }

    @Bean
    public JmsListenerContainerFactory<?> jmsListenerContainerQueue(ActiveMQConnectionF
ctory connectionFactory){
        DefaultJmsListenerContainerFactory bean = new DefaultJmsListenerContainerFactory();
        bean.setConnectionFactory(connectionFactory);
        return bean;
    }

    @Bean
    public JmsListenerContainerFactory<?> jmsListenerContainerTopic(ActiveMQConnectionFa
tory connectionFactory){
        DefaultJmsListenerContainerFactory bean = new DefaultJmsListenerContainerFactory();
        //设置为发布订阅方式, 默认情况下使用的生产消费者方式
        bean.setPubSubDomain(true);
        bean.setConnectionFactory(connectionFactory);
        return bean;
    }
}
```

5. 生产者ProviderController.java文件

```
package com.example.demo.Controller;
import java.util.UUID;
import javax.jms.Queue;
import javax.jms.Topic;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.annotation.JmsListener;
import org.springframework.jms.core.JmsMessagingTemplate;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/provider")
public class ProviderController {
        @Autowired
        private JmsMessagingTemplate jms;
```

```
    @Autowired
    private Queue queue;
    @Autowired
    private Topic topic;
    //点对点的消息发送
    @RequestMapping("/queue")
    public String queue(){
        for (int i = 0; i < 10 ; i++){
            jms.convertAndSend(queue, "    queue"+i);
        }
        return "queue 发送成功";
    }
    //接受消息
    @JmsListener(destination = "out.queue")
    public void consumerMsg(String msg){
        System.out.println(msg);
    }
    //订阅消息发送
    @RequestMapping("/topic")
    public String topic(){
        for (int i = 0; i < 10 ; i++){
            jms.convertAndSend(topic, "topic"+i);
        }
        return "topic 发送成功";
    }

    // 定时推送queues消息
    @Scheduled(fixedDelay = 3000)
    public void timeSendQueueMsg() {
        String str = UUID.randomUUID().toString().replace("-", "").substring(0, 10);
        System.out.println("定时推送Queue消息:" + str);
        jms.convertAndSend(queue, str);
    }

    //定时推送topics消息
    @Scheduled(fixedDelay = 8000)
    public void timeSendTopicMsg() {
        String str = UUID.randomUUID().toString().replace("-", "").substring(0, 10);
        System.out.println("定时推送Topic消息:" + str);
        jms.convertAndSend(topic, str);
    }
}
```

6. 消费者ActiveMQListener.java

```
package com.example.demo.Listener;
import javax.jms.JMSException;
import javax.jms.TextMessage;
import org.springframework.jms.annotation.JmsListener;
import org.springframework.messaging.handler.annotation.SendTo;
import org.springframework.stereotype.Component;
@Component
public class ActiveMQListener {
    //queues消息监听
```

```
    @JmsListener(destination = "test-queue",containerFactory = "jmsListenerContainerQueue")
    //将检测到的消息写入到 queue : out.queue 中去.
    @SendTo("out.queue")
    public String  receiveQueue(TextMessage textMessage) throws JMSException {
        System.out.println("消费者接收到queues消息:"+textMessage.getText());
        return textMessage.getText();
    }
    //topic消息监听
    @JmsListener(destination = "test-topic",containerFactory = "jmsListenerContainerTopic")
    public void receiveTopic(TextMessage textMessage) throws JMSException {
        System.out.println("消费者接收到topics消息:"+textMessage.getText());
    }
}
```

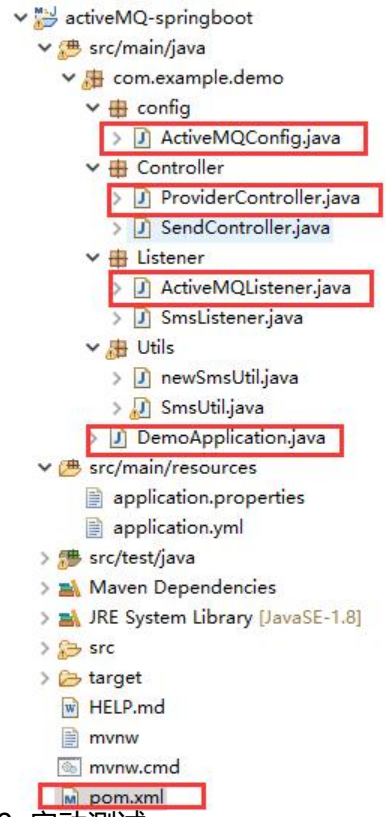
7. DemoApplication.java文件

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
//yml包扫描
//@EnableScheduling //开启定时任务
@ComponentScan
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

注意：如果在运行出现了 无法加载文件application.yml，文件的话，可以在pom.xml配置文件中加入

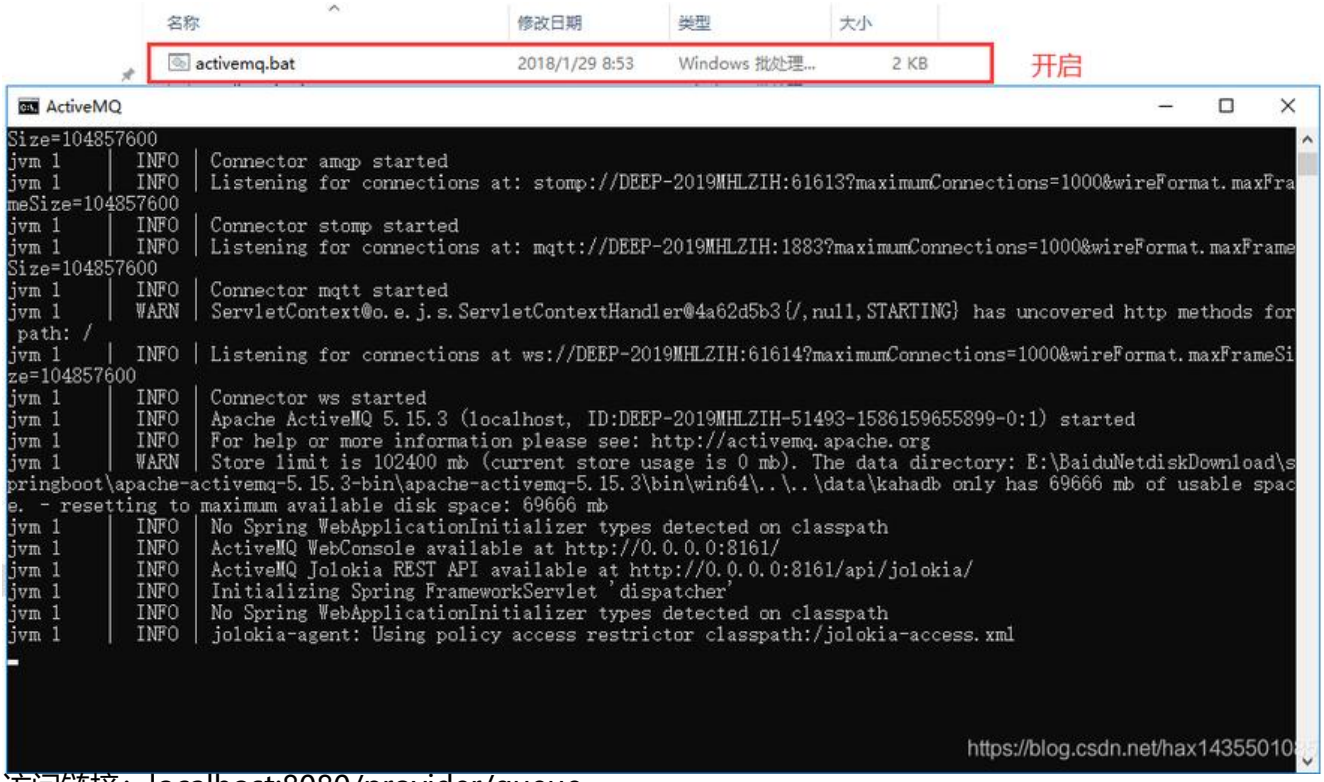
```
<!--yml文件解析  -->
    <dependency>
        <groupId>org.yaml</groupId>
        <artifactId>snakeyaml</artifactId>
    </dependency>
```

8. 目录结构



9. 启动测试

开启activeMQ服务



访问链接：localhost:8080/provider/queue

queue 发送成功

原文链接：SpringBoot【短信微服务】

```
消费者接收到queues消息：        queue0
        queue0
消费者接收到queues消息：        queue1
        queue1
消费者接收到queues消息：        queue2
        queue2
消费者接收到queues消息：        queue3
        queue3
消费者接收到queues消息：        queue4
        queue4
消费者接收到queues消息：        queue5
        queue5
消费者接收到queues消息：        queue6
        queue6
消费者接收到queues消息：        queue7
        queue7
消费者接收到queues消息：        queue8
        queue8
```

访问链接：localhost:8080/provider/topic

在进行topic 访问的时候需要修改application.yml配置文件

```
jms:
  pub-sub-domain: true  #false为queues  true 为topics
```


```
消费者接收到topics消息:topic0
消费者接收到topics消息:topic1
消费者接收到topics消息:topic2
消费者接收到topics消息:topic3
消费者接收到topics消息:topic4
消费者接收到topics消息:topic5
消费者接收到topics消息:topic6
消费者接收到topics消息:topic7
消费者接收到topics消息:topic8
消费者接收到topics消息:topic9
```

测试成功

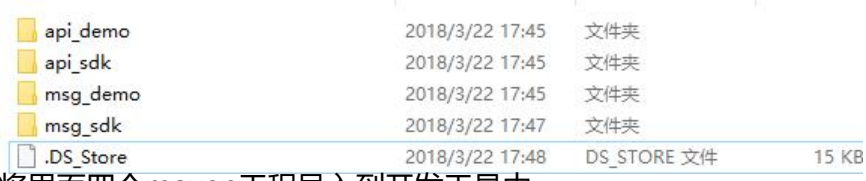
整合好了SpringBoot和ActiveMQ下面开始了解关于阿里大于的相关信息,搭短信微服务

2种方式进行阿里大于短信服务的开发

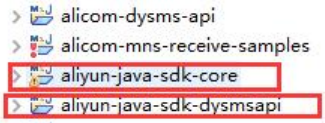
1. 在阿里大于里面下载demo测试文件　www.alidayu.com　dysms_java.zip　2020/4/1 15:53　WinRAR Z

打开文件里面有

原文链接：SpringBoot【短信微服务】

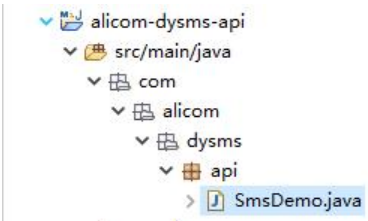


将里面四个maven工程导入到开发工具中



将红色的线框进行maven安装本地

2. alicom-dysms-api进行测试



SmsDemo.java

```
package com.alicom.dysms.api;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.dysmsapi.model.v20170525.QuerySendDetailsRequest;
import com.aliyuncs.dysmsapi.model.v20170525.QuerySendDetailsResponse;
import com.aliyuncs.dysmsapi.model.v20170525.SendSmsRequest;
import com.aliyuncs.dysmsapi.model.v20170525.SendSmsResponse;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.profile.IClientProfile;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Created on 17/6/7.
 * 短信API产品的DEMO程序,工程中包含了一个SmsDemo类，直接通过
 * 执行main函数即可体验短信产品API功能(只需要将AK替换成开通了云通信-短信产品功能的AK即可)
 * 工程依赖了2个jar包(存放在工程的libs目录下)
 * 1:aliyun-java-sdk-core.jar
 * 2:aliyun-java-sdk-dysmsapi.jar
 *
 * 备注:Demo工程编码采用UTF-8
 * 国际短信发送请勿参照此DEMO
 */
public class SmsDemo {

    //产品名称:云通信短信API产品,开发者无需替换
    static final String product = "Dysmsapi";
    //产品域名,开发者无需替换
    static final String domain = "dysmsapi.aliyuncs.com";

    // TODO 此处需要替换成开发者自己的AK(在阿里云访问控制台寻找)
    static final String accessKeyId = "youraccessKeyId";
```

```
    static final String accessKeySecret = "youraccessKeySecret";

    public static SendSmsResponse sendSms() throws ClientException {

        //可自助调整超时时间
        System.setProperty("sun.net.client.defaultConnectTimeout", "10000");
        System.setProperty("sun.net.client.defaultReadTimeout", "10000");

        //初始化acsClient,暂不支持region化
        IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou", accessKeyId, accessKeyS
cret);
        DefaultProfile.addEndpoint("cn-hangzhou", "cn-hangzhou", product, domain);
        IAcsClient acsClient = new DefaultAcsClient(profile);

        //组装请求对象-具体描述见控制台-文档部分内容
        SendSmsRequest request = new SendSmsRequest();
        //必填:待发送手机号
        request.setPhoneNumbers("15671381019");
        //必填:短信签名-可在短信控制台中找到
        request.setSignName("CRM客户管理系统");
        //必填:短信模板-可在短信控制台中找到
        request.setTemplateCode("SMS_186968097");
        //可选:模板中的变量替换JSON串,如模板内容为"亲爱的${name},您的验证码为${code}"时,此处
值为
        request.setTemplateParam("{\"code\":\"520111\"}");

        //选填-上行短信扩展码(无特殊需求用户请忽略此字段)
        //request.setSmsUpExtendCode("90997");

        //可选:outId为提供给业务方扩展字段,最终在短信回执消息中将此值带回给调用者
        request.setOutId("yourOutId");

        //hint 此处可能会抛出异常，注意catch
        SendSmsResponse sendSmsResponse=null;
        try {
            sendSmsResponse = acsClient.getAcsResponse(request);

        } catch (Exception e) {
            // TODO: handle exception
        }
        return sendSmsResponse;
    }


    public static QuerySendDetailsResponse querySendDetails(String bizId) throws ClientExcept
on {

        //可自助调整超时时间
        System.setProperty("sun.net.client.defaultConnectTimeout", "10000");
        System.setProperty("sun.net.client.defaultReadTimeout", "10000");

        //初始化acsClient,暂不支持region化
        IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou", accessKeyId, accessKeyS
cret);
```

```
        DefaultProfile.addEndpoint("cn-hangzhou", "cn-hangzhou", product, domain);
        IAcsClient acsClient = new DefaultAcsClient(profile);

        //组装请求对象
        QuerySendDetailsRequest request = new QuerySendDetailsRequest();
        //必填-号码
        request.setPhoneNumber("15671381019");
        //可选-流水号
        request.setBizId(bizId);
        //必填-发送日期 支持30天内记录查询，格式yyyyMMdd
        SimpleDateFormat ft = new SimpleDateFormat("yyyyMMdd");
        request.setSendDate(ft.format(new Date()));
        //必填-页大小
        request.setPageSize(10L);
        //必填-当前页码从1开始计数
        request.setCurrentPage(1L);

        //hint 此处可能会抛出异常，注意catch
        QuerySendDetailsResponse querySendDetailsResponse = acsClient.getAcsResponse(req
est);

        return querySendDetailsResponse;
    }

    public static void main(String[] args) throws ClientException, InterruptedException {

        //发短信
        SendSmsResponse response = sendSms();
        System.out.println("短信接口返回的数据----------------");
        System.out.println("Code=" + response.getCode());
        System.out.println("Message=" + response.getMessage());
        System.out.println("RequestId=" + response.getRequestId());
        System.out.println("BizId=" + response.getBizId());

        Thread.sleep(3000L);

        //查明细
        if(response.getCode() != null && response.getCode().equals("OK")) {
            QuerySendDetailsResponse querySendDetailsResponse = querySendDetails(response.
etBizId());
            System.out.println("短信明细查询接口返回数据----------------");
            System.out.println("Code=" + querySendDetailsResponse.getCode());
            System.out.println("Message=" + querySendDetailsResponse.getMessage());
            int i = 0;
            for(QuerySendDetailsResponse.SmsSendDetailDTO smsSendDetailDTO : querySendDe
ailsResponse.getSmsSendDetailDTOs())
            {
                System.out.println("SmsSendDetailDTO["+i+"]:");
                System.out.println("Content=" + smsSendDetailDTO.getContent());
                System.out.println("ErrCode=" + smsSendDetailDTO.getErrCode());
                System.out.println("OutId=" + smsSendDetailDTO.getOutId());
                System.out.println("PhoneNum=" + smsSendDetailDTO.getPhoneNum());
                System.out.println("ReceiveDate=" + smsSendDetailDTO.getReceiveDate());
                System.out.println("SendDate=" + smsSendDetailDTO.getSendDate());
```

```
                System.out.println("SendStatus=" + smsSendDetailDTO.getSendStatus());
                System.out.println("Template=" + smsSendDetailDTO.getTemplateCode());
            }
            System.out.println("TotalCount=" + querySendDetailsResponse.getTotalCount());
            System.out.println("RequestId=" + querySendDetailsResponse.getRequestId());
        }

    }
}
```

- 关于测试文件里面的（解释代码里面有）

1.accessKeyId

2.accessKeySecret

3.setSignName

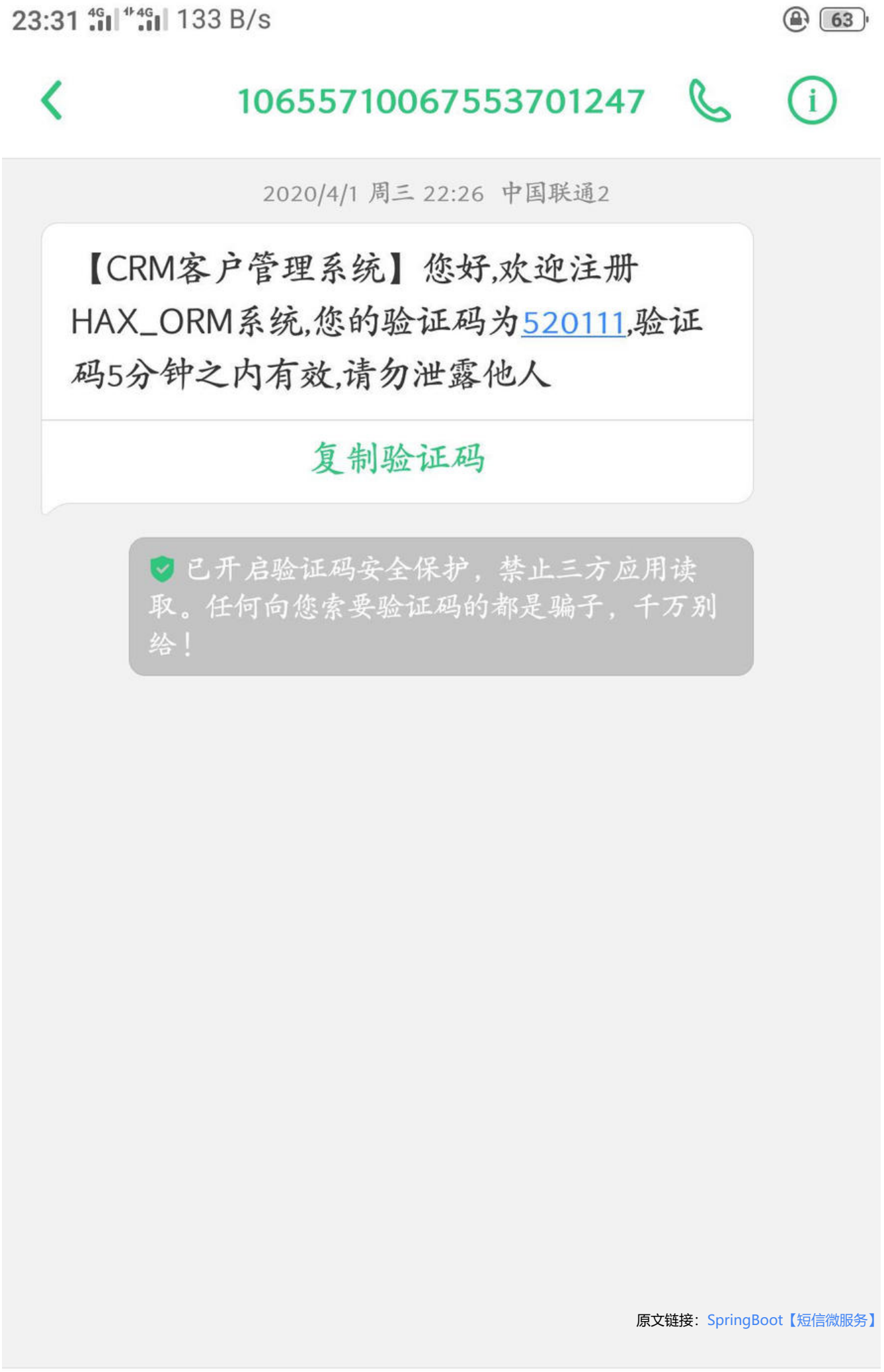
4.setTemplateCode

5.setTemplateParam

都需要自己去阿里大于的官网注册申请

可以参考链接：https://blog.csdn.net/u014079773/article/details/65939797

- 测试截图



原文链接：SpringBoot【短信微服务】

3. 然后将安装好依赖添加到pom.xml工程中即可

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-dysmsapi</artifactId>
    <version>1.0.0-SNAPSHOT</version>
</dependency>
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.1.0</version>
</dependency>
```

- 第二种是直接添加依赖，镜像链接下载

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.1.0</version>
</dependency>
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-dysmsapi</artifactId>
    <version>1.0.0</version>
</dependency>
```

4. 封装SmsUtil.java工具类

```
package com.example.demo.Utils;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Component;

import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.dysmsapi.model.v20170525.QuerySendDetailsRequest;
import com.aliyuncs.dysmsapi.model.v20170525.QuerySendDetailsResponse;
import com.aliyuncs.dysmsapi.model.v20170525.SendSmsRequest;
import com.aliyuncs.dysmsapi.model.v20170525.SendSmsResponse;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.profile.IClientProfile;
/**
 *
 * @author Administrator_hax
 *     短信发送工具类
 */
@PropertySource("classpath:application.properties")//在加载钥密
@Component
public class SmsUtil {
```

```
    //产品名称:云通信短信API产品,开发者无需替换
    static final String product = "Dysmsapi";
    //产品域名,开发者无需替换
    static final String domain = "dysmsapi.aliyuncs.com";

    @Autowired
    private Environment env;

    /**
     * 发送短信
     * @param mobile 手机号
     * @param template_code 模板号
     * @param sign_name 签名
     * @param param 参数
     * @return
     * @throws ClientException
     */
    public SendSmsResponse sendSms(String mobile,String template_code,String sign_name,St
ing param) throws ClientException {
        // TODO 此处需要替换成开发者自己的AK(在阿里云访问控制台寻找)
        String accessKeyId =env.getProperty("accessKeyId");
        String accessKeySecret = env.getProperty("accessKeySecret");

        //可自助调整超时时间
        System.setProperty("sun.net.client.defaultConnectTimeout", "10000");
        System.setProperty("sun.net.client.defaultReadTimeout", "10000");

        //初始化acsClient,暂不支持region化
        IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou", accessKeyId, accessKeyS
cret);
        DefaultProfile.addEndpoint("cn-hangzhou", "cn-hangzhou", product, domain);
        IAcsClient acsClient = new DefaultAcsClient(profile);

        //组装请求对象-具体描述见控制台-文档部分内容
        SendSmsRequest request = new SendSmsRequest();
        //必填:待发送手机号
        request.setPhoneNumbers(mobile);
        //必填:短信签名-可在短信控制台中找到
        request.setSignName(sign_name);
        //必填:短信模板-可在短信控制台中找到
        request.setTemplateCode(template_code);
        //可选:模板中的变量替换JSON串,如模板内容为"亲爱的${name},您的验证码为${code}"时,此处
值为
        request.setTemplateParam(param);

        //选填-上行短信扩展码(无特殊需求用户请忽略此字段)
        //request.setSmsUpExtendCode("90997");

        //可选:outId为提供给业务方扩展字段,最终在短信回执消息中将此值带回给调用者
        request.setOutId("yourOutId");

        //hint 此处可能会抛出异常，注意catch
        SendSmsResponse  sendSmsResponse=null;
        try {
```

```
            sendSmsResponse = acsClient.getAcsResponse(request);
        } catch (Exception e) {
            e.printStackTrace();
        }


        return sendSmsResponse;
    }

    /**
     *     消息查询响应
     * @param mobile 电话号码
     * @param bizId
     * @return
     * @throws ClientException
     */
    public  QuerySendDetailsResponse querySendDetails(String mobile,String bizId) throws Cli
ntException {
        String accessKeyId =env.getProperty("accessKeyId");
        String accessKeySecret = env.getProperty("accessKeySecret");
        //可自助调整超时时间
        System.setProperty("sun.net.client.defaultConnectTimeout", "10000");
        System.setProperty("sun.net.client.defaultReadTimeout", "10000");
        //初始化acsClient,暂不支持region化
        IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou", accessKeyId, accessKeyS
cret);
        DefaultProfile.addEndpoint("cn-hangzhou", "cn-hangzhou", product, domain);
        IAcsClient acsClient = new DefaultAcsClient(profile);
        //组装请求对象
        QuerySendDetailsRequest request = new QuerySendDetailsRequest();
        //必填-号码
        request.setPhoneNumber(mobile);
        //可选-流水号
        request.setBizId(bizId);
        //必填-发送日期 支持30天内记录查询，格式yyyyMMdd
        SimpleDateFormat ft = new SimpleDateFormat("yyyyMMdd");
        request.setSendDate(ft.format(new Date()));
        //必填-页大小
        request.setPageSize(10L);
        //必填-当前页码从1开始计数
        request.setCurrentPage(1L);
        //hint 此处可能会抛出异常，注意catch
        QuerySendDetailsResponse querySendDetailsResponse = acsClient.getAcsResponse(req
est);
        return querySendDetailsResponse;
    }
}
```

监听SmsListener.java

```
package com.example.demo.Listener;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.annotation.JmsListener;
```

```
import org.springframework.stereotype.Component;

import com.aliyuncs.dysmsapi.model.v20170525.SendSmsResponse;
import com.aliyuncs.exceptions.ClientException;
import com.example.demo.Utils.SmsUtil;
@Component
public class SmsListener {
        @Autowired
        private SmsUtil smsUtil;
        //使用监听器进行消息的监听操作 ,将消息发送过来,进行短信发送
        @JmsListener(destination="sms")
        public void sendSms(Map<String,String> map){
            try {
                SendSmsResponse response = smsUtil.sendSms(
                        map.get("mobile"),
                        map.get("template_code"),
                        map.get("sign_name"),
                        map.get("param")  );
                    System.out.println("Code=" + response.getCode());
                    System.out.println("Message=" + response.getMessage());
                    System.out.println("RequestId=" + response.getRequestId());
                    System.out.println("BizId=" + response.getBizId());
            } catch (ClientException e) {
                e.printStackTrace();
            }
        }
}
```

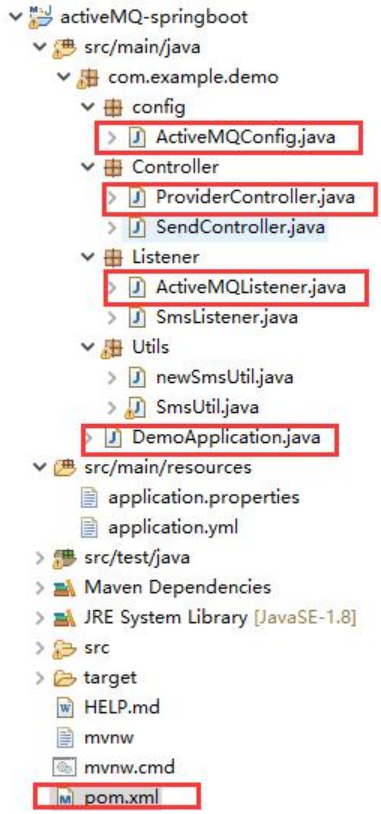
- 拦截SendController.java

```
package com.example.demo.Controller;
import java.util.HashMap;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.core.JmsMessagingTemplate;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class SendController {
    @Autowired
    private JmsMessagingTemplate jmsMessagingTemplate;
    @RequestMapping("/sendsms")
    public void sendSms(){
        Map<String,String > map=new HashMap<String,String>();
        map.put("mobile", "15671381019");
        map.put("template_code", "SMS_186968097");
        map.put("sign_name", "CRM客户管理系统");
        map.put("param", "{\"code\":\"520520\"}");
        jmsMessagingTemplate.convertAndSend("sms",map);
}
}
```

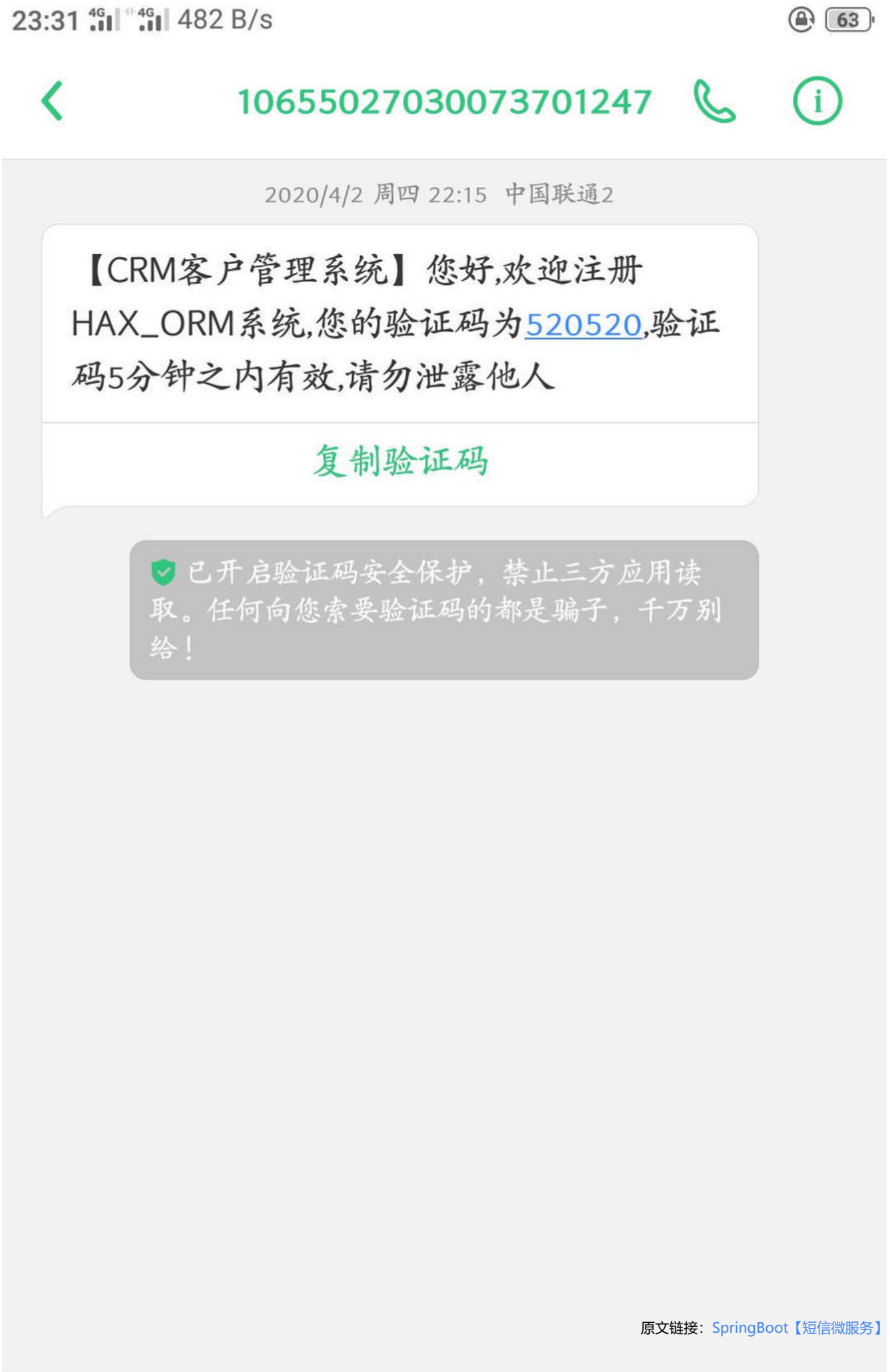
- application.properties文件

accessKeyId:不告诉你
accessKeySecret:不告诉你

- 目录结构

![在这里插入图片描述](

- 测试链接：localhost:8080/sendsms



原文链接：SpringBoot【短信微服务】

分析思路：

● 在实际的开发项目中，只需要依赖该maven工程即可了，然后在进行登陆注册的操作的时候，发送个请求地址，地址拦截进行短信发送，短信验证码使用随机数产生,然后在将用户手机号码可以以key形式 value为验证码存储到redis中 ，可以设置过期时间达到几分钟验证的效果,将用户输入的验证码和edis中的验证码进行比较,已达到注册登录的效果.