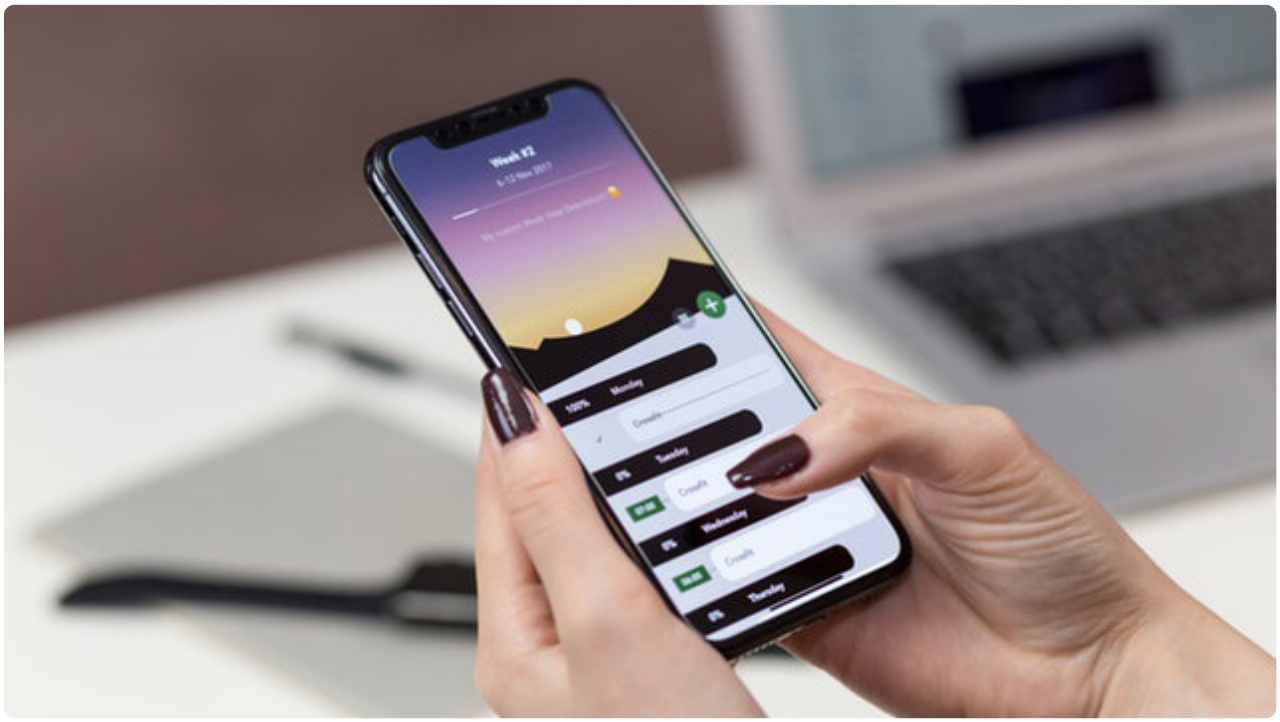
09 阿里云短信服务

更新时间：2019-08-05 10:24:55



> 宝剑锋从磨砺出，梅花香自苦寒来。
>
> ——佚名

说起目前较为流行的用户登录方式，自然是利用手机号的唯一性，在登录前收取一条短信验证码进行验证，进而完成登录操作。这种短信验证服务大多数借助于第三方服务来实现，本章节将要讲解：如何使用阿里云的短信服务，来完成我们项目的登录验证。

短信服务

短信服务（Short Message Service）是指通过调用短信发送API，将指定短信内容发送给指定手机用户。用户收到的短信来自106开头的号码，短信的内容多用于企业向用户传递验证码、系统通知、会员服务等信息，很多云平台都有提供，这里我们采用阿里云的短信服务。想要使用短信服务必须满足下面前提：

1. 需要阿里云注册，入口：点击注册页面。
2. 实名认证：点击进行实名认证。
3. 绑定支付宝：点击绑定支付宝。

开通短信服务

进入短信服务入口

获取**AccessKey**相关**key**

首先，调用阿里云提供的相关API是必须需要授权的，这时我们需要获取一些标识，表明你是一个合法的用户：

1. 创建AccessKey: 点击创建AccessKey
2. 获取AccessKey ID和AccessKey Secret：点击查看AccessKey ID和AccessKey Secret

AccessKeyId 用于标识用户，AccessKeySecret 是用来验证用户的密钥。AccessKeySecret 必须保密。

创建签名和模版

签名：

短信签名是短信服务提供的一种快捷、方便的个性化签名方式。当发送短信时，短信平台会根据设置，在短信内容里附加个性签名，再发送给指定手机号码。入口

创建需要审核，审核成功之后，我们需要保存签名名称：WECIRCLE，在后面调用API时会用到。

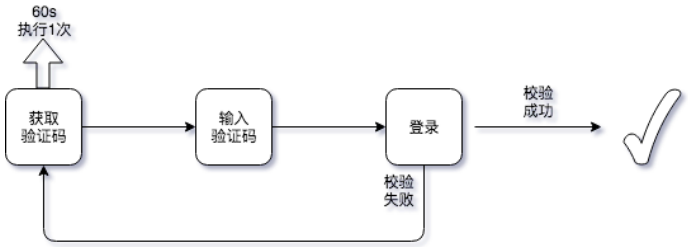
模版：

短信模版，即具体发送的短信内容。入口

创建需要审核，审核成功之后，我们需要保存模板code：sms_xxxxx，在后面调用API时会用到。

代码里调用短信API接口

经过上述步骤，我们在阿里云平台已经注册的短信服务，接下来我们需要在程序里使用这个服务。



阿里云给我们提供了基于 Node.js 的 SDK，当然也会提供基于其他语言的，这里我们使用Node.js的SDK。

如果想要查看更多的文档，可以参考这里。

安装@alicloud/pop-core：

```
npm install @alicloud/pop-core --save
```

新建sms.js:

我们在后端项目的 utils 文件夹下新建一个 sms.js ,用来封装我们的短信服务相关的逻辑，下面这段代码主要创建一个 client 实例，这个是阿里短信 SDK 要求的，同时将我们的之前得到的 accessKeyId 和 accessKeySecret 传进去。

```
var Core = require('@alicloud/pop-core');

//创建请求client实例
var client = new Core({
  accessKeyId: 'LTAIEGH3OV5cRwBW',//accessKeyId
  accessKeySecret: '',//accessKeySecret，请各位使用自己的accessKeySecret
  endpoint: 'https://dysmsapi.aliyuncs.com',//固定写死即可
  apiVersion: '2017-05-25'//固定写死即可
});
```


创建发送验证码的接口调用：

完成了相关实例的新建之后，就需要开始写具体的方法了，下面这段代码封装一个 sendSms 方法，方法里面主要调用 SDK 里的方法，同时会用到我们之前得到的短信模板和短信签名。

```
/*
* 发送手机验证码
*/
sendSms(data,succ,fail){
  //随机获取6位数字
  var s2msCode = Math.random().toString().slice(-6)

  var params = {
    PhoneNumbers:data.phoneNum,//需要发送的手机号
    SignName: SignName,//短信签名名称
    TemplateCode: TemplateCode,//模板code字符串
    TemplateParam:JSON.stringify({"code":s2msCode})//短信模板变量对应的实际值，JSON格式，将生成的随机数传进去
  }

  var requestOption = {
    method: 'POST'
  };

  //apiKey是SendSms
  client.request('SendSms', params, requestOption).then((result) => {
    console.log(result);
    succ && succ(result)
  }, function(ex) {

    console.log(ex);
    fail && fail(ex)
  })
},
```

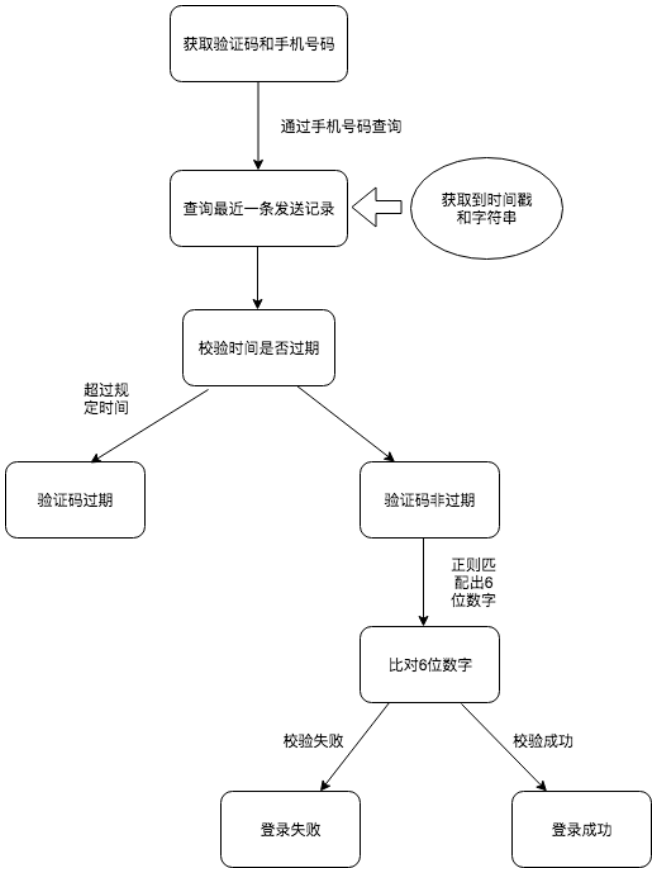
需要注意的是，手机短信收到的随机值是由我们的代码控制，所以你可以自定义英文字母或者数字，这里我们采用6位的随机数字。

创建验证验证码的接口调用：

下面这段代码主要逻辑是用户得到验证码后，需要验证是否有效。

```
  /*
 * 查看已经发送的验证码
 */
checkCode(data,succ,fail){
  var params = {
    PhoneNumber:data.phoneNum,//需要验证的手机号
    SendDate: getFormatedDate(),//短信发送日期，支持查询最近30天的记录。格式为yyyyMMdd，例如20181225
    PageSize: 40,//指定每页显示的短信记录数量
    CurrentPage:1//指定发送记录的的当前页码
  }

  var requestOption = {
    method: 'POST'
  };
  //apiKey是QuerySendDetails
  client.request('QuerySendDetails', params, requestOption).then((result) => {

    if (result.Code === 'OK') {
      var detail = result.SmsSendDetailDTOs.SmsSendDetailDTO[0] || {};

      //只筛选1分钟以内的数据
      if ((new Date() - new Date(detail.ReceiveDate)) < diffRange) {
        //校验查询到的第一条最新的数据，使用正则表达式match到验证码，和用户输入传进来的验证码进行比对
        if (detail.Content.match(pattern)[0] && (detail.Content.match(pattern)[0] === data.code)) {

          succ && succ({
            code:0,
            msg:'验证成功'
          })
        } else {//否则就校验失败
          fail && fail({
            code:1,
            msg:'验证失败'
          });
        }
      } else {//校验失败,验证码过期
        fail && fail({
          code:1,
          msg:'验证失败'
        });
      }
    } else {//否则就校验失败
      fail && fail({
        code:1,
        msg:'验证失败'
      });
    }
  }, (ex) => {//api接口调用失败兼容
    console.log(ex);
    fail && fail(result)
  })
}
```


我们这里采用的验证流程逻辑是：



1. 接收到用户提交的验证码和电话。
2. 通过阿里云接口，查询出当前电话的最近一条短信发送记录字符串同时获取发送时间。
3. 通过正则匹配的方式获取到这个字符串中的6位随机数字。
4. 通过发送时间对比当前收到验证码的时间，判断是否过期。
5. 没过期就验证随机数字是否和传入的验证码相同。
6. 验证成功即表示登录成功。

   当然，这里的另外一种实现方案是不采用从阿里云的api获取数据，而是可以直接将上一次发送的code和当前的电话号码利用一个变量记录在内存中，大家可以自己动手实现一些这里的逻辑。

小结

本章节主要讲解了如果使用阿里云的短信验证服务，并介绍了一般短信验证的整个流程。

相关技术点：

1. 阿里云AccessKey ID和AccessKey Secret以及短信签名，短信模板的获取方法。
2. 安装@alicloud/pop-core来使用短信的Node.js相关API。
3. 发送验证码逻辑和验证验证码逻辑梳理。

本章节完整源代码地址：Github

}