国际物流云商系统第六天

一．回顾

1.用户---角色

Span 标签的样式： display:inline-block;width:200px

2.角色---模块

zTree 树展示权限列表
实际项目用得多，zTree 的使用步骤：1.导入 js/css　　2.jsp 中引入 js/css
3.组织 json 数据，手动拼接 json 串[{id:"",pId:"",name:"",checked:"true|false"}]
4.将这个 json 串向浏览器输出:response 对象手动输出
5.客户端可以发出 ajax 请求，来得到这个 json 串
$.ajax({url:"",dataType:"text",type:"GET",success:function(){} });

注意多对多操作时，如果要操作中间表的关系，不需要加 **cascade**

3.struts2 异常处理框架

struts.xml 文件中配置：

```
<global-results>
        <result name="error">/WEB-INF/pages/error.jsp</result>
</global-results>
<global-exception-mappings>
        <exception-mapping exception="cn.itcast.jk.exception.SysException" result="error"/>
</ global-exception-mappings>
```

Action 类中

```
throw new SysException("");
```

4.细粒度权限控制

什么是细粒度权限控制，它的作用及实现方式。

二． JavaMail

1.什么是 JavaMail

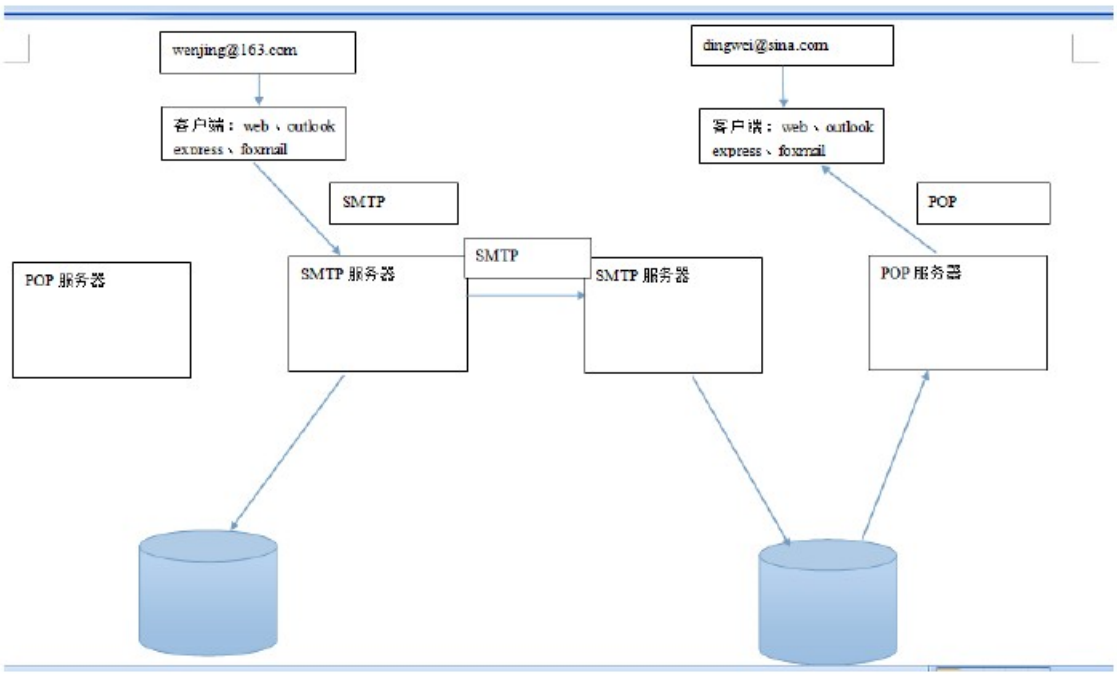
JavaMail 是提供给开发人员在应用程序中实现邮件发送和接收功能而提供的一套标准开发类库，支持常用的邮件协议，如 SMTP、POP3、IMAP，开发人员使用 JavaMail 编写邮件程序时，无需考虑底层的通信细节(Socket)，JavaMail 也提供了能够创建出各种复杂 MIME 格式的邮件内容的 API。使用 JavaMail，我们可以实现类似 OutLook、FoxMail 的软件。

2. JavaMail 的协议

- 邮件开发的相关协议

SMTP：Simple Message Transfer Protocal 发送协议 默认端口：25
POP：Post Office Protocal 邮局协议。POP3 这个版本用的最多，接收协议 默认端口：110

3.邮件发送接收的过程分析



三．邮件开发的准备工作

1. 申请邮箱

我这里申请的是新浪、网易邮箱，进入后开通 POP3/SMTP 服务

2. 引入 JavaMail

在 **jk2601_parent** 工程的 **pom.xml** 中添加如下依赖

```
<!-- Javamail -->
<dependency>
  <groupId>javax.mail</groupId>
  <artifactId>mail</artifactId>
  <version>1.4.4</version>
</dependency>
```

为了让 **Spring** 与 **JavaMail** 集成，还需要在 **ilcbs_parent** 工程的 **pom.xml** 引入如下依赖：

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context-support</artifactId>
  <version>${spring.version}</version>
</dependency>
```

如果是 **web** 项目，引入如下 **jar** 包

四．传统的邮件开发

1.使用 JavaMail 常用 API 实现邮件发送

```
Properties props=new Properties();
props.put("mail.smtp.host","smtp.163.com");
props.put("mail.smtp.auth","true");
Session session=Session.getInstance(props);
session.setDebug(true);
//构造信息体
MimeMessage message =new MimeMessage(session);
 //发件地址
Address address = new InternetAddress("itheima14@163.com");
message.setFrom(address);
//收件地址
Address toAddress = new InternetAddress("3462420264@qq.com");
message.setRecipient(MimeMessage.RecipientType.TO, toAddress);
//主题
message.setSubject("Hello world");
//正文
message.setText("Hello world");
message.saveChanges();
Transport transport = session.getTransport("smtp");
transport.connect("smtp.163.com", "itheima14@163.com", "iamsorry"); //发送
transport.sendMessage(message, message.getAllRecipients());
transport.close();
```

2.邮件发送的工具类的提取

```
c class MailUtil {

   /**
    * 发送电子邮件
    * @param addr 收件人地址
    * @param subject 主题
    * @param text 内容
    * @throws MessagingException
    */
   public static void sendMail(String addr,String subject,String text) throws MessagingException{
           Properties props=new Properties();
           props.put("mail.smtp.host","smtp.sina.com");
           props.put("mail.smtp.auth","true");
           Session session=Session.getInstance(props);
           //构造信息体
           MimeMessage message =new MimeMessage(session);
            //发件地址
           Address address = new InternetAddress("wwwitcastcn@sina.com");
           message.setFrom(address);
           //收件地址
           Address toAddress = new InternetAddress(addr);
           message.setRecipient(MimeMessage.RecipientType.TO, toAddress);
           //主题
           message.setSubject(subject);
           //正文
           message.setText(text);
           message.saveChanges();
           Transport transport = session.getTransport("smtp");
           transport.connect("smtp.sina.com", "wwwitcastcn@sina.com", "itcast"); //发送
           transport.sendMessage(message, message.getAllRecipients());
           transport.close();

   }
```

五.使用 JavaMail 实现员工登录信息的发送

业务需求：在员工信息添加时，同时需要向员工发送一封通知的邮件，以进行提示！

数据库字段的修改：USER_INFO_P 表

1.修改 Userinfo.java 类,添加 email 属性

```
@Column(name="EMAIL")
private String email;
```

2.修改 jUserCreate.jsp 页面

```
<td class="columnTitle">邮箱：</td>
<td class="tableContent"><input type="text" name="userinfo.email" value=""/></td>
```

3.进入业务逻辑层 UserServiceImpl 中

```
//发送邮件，可能需要一些时间，开线程实现邮件发送
Thread th = new Thread(new Runnable() {

    public void run() {
        //发送邮件
        try {
            MailUtil.sendMail(entity.getUserinfo().getEmail(), "新员工入职账户信息提醒", entity.getUserinfo().getName()+"您好，欢迎您加入本集团，您在
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

th.start();//启动发送邮件的线程
```

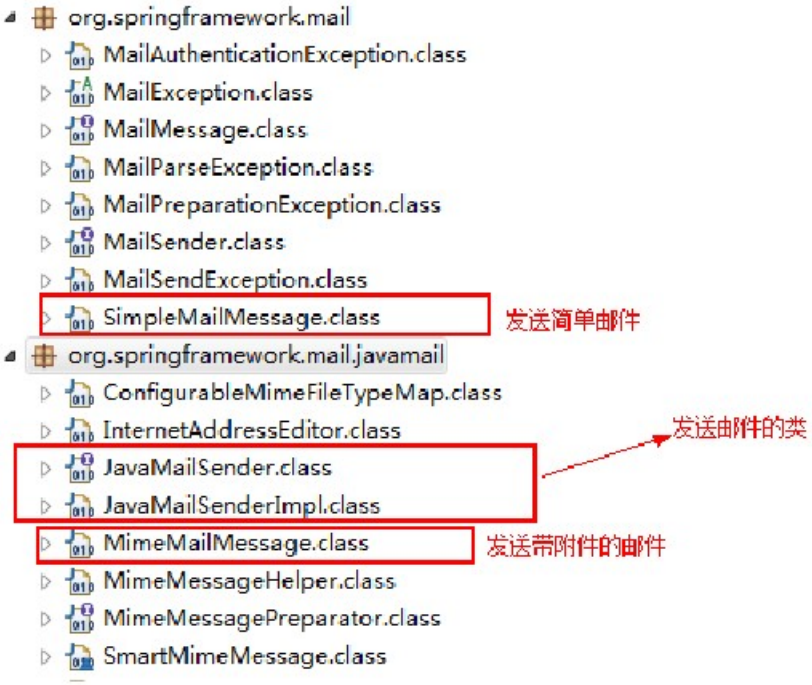
六． JavaMail 与 Spring 集成开发

Spring 邮件抽象层的主要包为 org.springframework.mail。它包括了发送电子邮件的主要接口 MailSender，和其实现类 SimpleMailMessage，它封装了简单邮件的属性如 *from*, *to*,*cc*, *subject*,*text*。包里还包含一棵以MailException为根的 checked Exception 继承树，它们提供了对底层邮件系统异常的高级别抽象。要获得关于邮件异常层次的更丰富的信息。

为了使用 *JavaMail* 中的一些特色，比如 MIME 类型的信件，spring 提供了 MailSender 的一个子接口，即 org.springframework.mail.javamail.JavaMailSender。Spring 还提供了一个回调接口 org.springframework.mail.javamail.MimeMessagePreparator，用于准备

JavaMail 的 MIME 信件。

1.Spring 对 JavaMail 支持的 API



2.创建邮件的配置文件

1、在 src 目录下建立 mail.properties 文件里边包含一下内容

mail.host=smtp.qq.com

mail.username=你的邮箱名

mail.password=你的邮箱密码

mail.from=发送方的邮箱

name	value
mail.smtp.host	smtp.163.com
mail.username	itheima14
mail.password	iamsorry
mail.smtp.auth	true
mail.from	itheima14@163.com

3.Spring 与 JavaMail 整合的配置文件

```
<context:property-placeholder location="classpath:mail.properties"/>

<bean id="mailMessage" class="org.springframework.mail.SimpleMailMessage">
    <property name="from" value="${mail.from}"></property>
</bean>

<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
     <property name="host" value="${mail.host}"></property>
     <property name="username" value="${mail.username}"></property>
     <property name="password" value="${mail.password}"></property>
     <property name="javaMailProperties">
         <props>
             <prop key="mail.smtp.auth">true</prop>
             <prop key="mail.smtp.timeout">0</prop>
             <prop key="mail.debug">true</prop>
         </props>
     </property>
</bean>
```

4.Spring 与 javaMail 整合测试

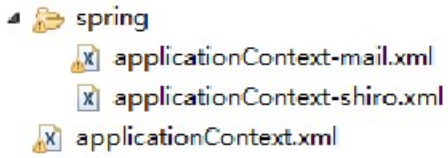
```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:spring/applicationContext-mail.xml")
public class MailTest {
    @Autowired
    private SimpleMailMessage mailMessage;

    @Autowired
    private MailSender mailSender;
    @Test
    public void testMail(){
        mailMessage.setTo("1955723102@qq.com");
        mailMessage.setSubject("邮件啊，成功吧");
        mailMessage.setText("你好，测试spring与javaMail整合");

        mailSender.send(mailMessage);
    }
}
```

5.整合测试成果

1.要在 UserServiceImpl 中注入相关的对象

```
@Autowired
private SimpleMailMessage mailMessage;
@Autowired
private JavaMailSender mailSender;
```

2.添加 spring 与 javaMail 集成的配置文件



3 使用线程实现邮件发送

```
 //第二种方式，使用spring与javaMail整合
Thread th = new Thread(new Runnable() {
    public void run() {
       try {
        mailMessage.setTo(entity.getUserinfo().getEmail());
        mailMessage.setSubject("新员工入职须知");
        mailMessage.setText("欢迎您加入传智播客集团，" + entity.getUserinfo().getName() +
                "先生，您在公司的账号：" + entity.getUserName()
                                + ",初始密码：" + SysConstant.DEFAULT_PASS);
        mailSender.send(mailMessage);
    } catch (Exception e) {
        e.printStackTrace();
       }
    }
});
th.start();
```

4.测试结果

5.扩展：带图片和附件的邮件发送

1. 发送带有图片的邮件，以嵌入 HTML 的方式

```
import java.io.File;

import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.core.io.FileSystemResource;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.mail.javamail.MimeMessageHelper;
```

```
public class SpringAttachedImageMail {

	public static void main(String[] args) throws MessagingException {

		ApplicationContext ctx = new ClassPathXmlApplicationContext(
				"applicationContext.xml");
		JavaMailSenderImpl sender = (JavaMailSenderImpl) ctx
				.getBean("mailSender");
		MimeMessage mailMessage = sender.createMimeMessage();
		MimeMessageHelper messageHelper = new
MimeMessageHelper(mailMessage,true);
		messageHelper.setFrom("XXXXX@qq.com");
		messageHelper.setTo("XXXXXXXX@qq.com");

		messageHelper.setSubject("测试邮件中嵌套图片!! ");
		// true 表示启动 HTML 格式的邮件
		messageHelper.setText(
				"<html><head></head><body><h1>hello!!spring
image html mail</h1>"
					+"<a
href=http://www.baidu.com>baidu</a>" + "<img src=cid:image/></body></html>",
true);

		FileSystemResource img = new FileSystemResource(new
File("itcast.png"));

		messageHelper.addInline("image", img);//跟 cid 一致

		sender.send(mailMessage);
		System.out.println("邮件发送成功...");

	}
}
```

2. 发送带附件的邮件

```
import java.io.File;

import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;
```

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.core.io.FileSystemResource;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.mail.javamail.MimeMessageHelper;

public class SpringAttachedImageMail {

      public static void main(String[] args) throws MessagingException {

            ApplicationContext ctx = new ClassPathXmlApplicationContext(
                        "applicationContext.xml");
            JavaMailSenderImpl sender = (JavaMailSenderImpl) ctx
                        .getBean("mailSender");
            MimeMessage mailMessage = sender.createMimeMessage();
            MimeMessageHelper messageHelper = new
MimeMessageHelper(mailMessage,true);
            messageHelper.setFrom("9197****1@qq.com");
            messageHelper.setTo("9197****1@qq.com");

            messageHelper.setSubject("测试邮件中嵌套图片!! ");
            // true 表示启动 HTML 格式的邮件
            messageHelper.setText(
                        "<html><head></head><body><h1>hello!!spring
image html mail</h1>"
                                    +"<a
href=http://www.baidu.com>baidu</a>" + "<img src=cid:image/></body></html>",
true);

            FileSystemResource img = new FileSystemResource(new
File("itcast.png"));

            messageHelper.addAttachment("itcast.png", file);//添加到附件

            sender.send(mailMessage);
            System.out.println("邮件发送成功...");
      }
}
```

七. HttpClient 应用

1.HttpClient 概述

HTTP 协议可能是现在 Internet 上使用得最多、最重要的协议了，越来越多的 Java 应用程序需要直接通过 HTTP 协议来访问网络资源。虽然在 JDK 的 java net 包中已经提供了访问 HTTP 协议的基本功能，但是对于大部分应用程序来说，JDK 库本身提供的功能还不够丰富和灵活。HttpClient 是 Apache Jakarta Common 下的子项目，用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。

2.HttpClient 开发准备

1.引入 HttpClient 开发的坐标

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.4</version>
</dependency>
```

3.HttpClient 开发步骤

使用 HttpClient 发送请求、接收响应很简单，一般需要如下几步即可。

1. 创建 HttpClient 对象。
   HttpClient client = new HttpClient();
2. 创建请求方法的实例，并指定请求 URL。
   如果需要发送 GET 请求，创建 HttpGet 对象；
   如果需要发送 POST 请求，创建 HttpPost 对象。

3.如果需要发送请求参数，可调用 HttpGet、HttpPost 共同的
   String params = EntityUtils.toString(new UrlEncodedFormEntity(list, Consts.UTF_8));再通过?实

现 url 拼接
对于 HttpPost 对象而言，也可调用 setEntity(HttpEntity entity)方法来设置请求参数。

4.调用 HttpClient 对象的 execute(HttpUriRequest request)发送请求，
该方法返回一个 HttpResponse。
5. 调用 HttpResponse 的 getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头；调用 HttpResponse 的 getEntity()方法可获取 HttpEntity 对象，该对象包装了服务器的响应内容。程序可通过该对象获取服务器的响应内容。

6. 释放连接。无论执行方法是否成功，都必须释放连接

4.HttpClient 发送 GET 请求

```
//测试GET请求
public class HttpClientTest {
    @Test
    public void testGet() throws Exception{
        //1.创建客户端
        CloseableHttpClient client = HttpClients.createDefault();
        //2.创建HttpGet对象
        HttpGet httpget = new HttpGet("http://www.itcast.cn");
        //3.执行get请求
        CloseableHttpResponse response = client.execute(httpget);
        //4.获取响应实体
        HttpEntity entity = response.getEntity();
        //5.获取响应状态
        System.out.println(response.getStatusLine());
        if(entity!=null){
            //获取响应内容的长度
            System.out.println(entity.getContentLength());
            //获取响应内容
            System.out.println(EntityUtils.toString(entity));
        }
        //6.关闭
        response.close();
    }

}
```

如果发送 GET 请求的同时还要带上参数，那么可以使用 BasicNameValuePair 来添加参数

```
//2.封装请求参数
List<BasicNameValuePair> list = new ArrayList<BasicNameValuePair>();
list.add(new BasicNameValuePair("username", "cgx"));
list.add(new BasicNameValuePair("password","123456"));

//3.转化参数
String params = EntityUtils.toString(new UrlEncodedFormEntity(list, Consts.UTF_8));
System.out.println(params);
//4.产生一个HttpGet请求
HttpGet httpGet = new HttpGet("http://localhost:8080/bj44/loginAction_login"+"?"+params);
```

```
//Get请求同时带参数
public class ClientTest02 {
    @Test
    public void testMai102() throws Exception{
        //1.得到HttpClient
        CloseableHttpClient client = HttpClients.createDefault();
        //2.封装请求参数
        List<BasicNameValuePair> list = new ArrayList<BasicNameValuePair>();
        list.add(new BasicNameValuePair("username", "cgx"));
        list.add(new BasicNameValuePair("password","123456"));

        //3.转化参数
        String params = EntityUtils.toString(new UrlEncodedFormEntity(list, Consts.UTF_8));
        System.out.println(params);
        //4.产生一个HttpGet请求
        HttpGet httpGet = new HttpGet("http://localhost:8080/bj44/loginAction_login"+"?"+params);
        //5.发送请求
        CloseableHttpResponse response = client.execute(httpGet);

        //5.得到实体
        HttpEntity entity = response.getEntity();

        System.out.println(EntityUtils.toString(entity));

    }
}
```

测试结果：

```
<html>
<head>
<title>国际物流云商系统</title>
</head>
<frameset rows="125,*" name="topFrameset" border="0">
    <frame name="top_frame" scrolling="no"  target="middleFrameSet" src="homeAction_title">
    <frameset cols="202,*" height="100%" name="middle" frameborder="no" border="0" framespacing="0">
        <frame name="leftFrame" class="leftFrame" target="main" scrolling="no" src="homeAction_toleft.action?moduleName=home" />
        <frame name="main" class="rightFrame" src="homeAction_tomain.action?moduleName=home" />
    </frameset>
</frameset>

<noframes>
<body>
    <p>此网页使用了框架，但您的浏览器不支持框架。</p>
</body>
</noframes>

</html>
```

5.HttpClient 发送 POST 请求

第一种方式

```
//发送POST请求
public class ClientTest03 {
    @Test
    public void testPost03() throws Exception{
        //1.创建ClosableHttpClient
        CloseableHttpClient client = HttpClients.createDefault();
        //2.设置参数
        List<BasicNameValuePair> params = new ArrayList<>();
        params.add(new BasicNameValuePair("username","cgx"));
        params.add(new BasicNameValuePair("password", "123456"));
        //转化参数
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(params,Consts.UTF_8);
        //3.产生HttpPost对象
        HttpPost post = new HttpPost("http://localhost:8080/bj44/loginAction_login");
        //设置参数
        post.setEntity(entity);
        //4.发送请求
        CloseableHttpResponse response = client.execute(post);
        //5.处理结果
        System.out.println(EntityUtils.toString(response.getEntity()));
        //6.关闭
        response.close();
    }
}
```

第二种方式：

```
@Test   //实现
public void testPost() throws Exception{
    //1.创建HttpClient对象
    HttpClient client = new HttpClient();
    //2.创建GET或POST请求方法
    PostMethod method = new PostMethod("http://localhost:8080/bj44/loginAction_login");
    //3.设置编码
    client.getParams().setContentCharset("UTF-8");
    //4.设置请求消息头，为表单提交方式
    method.setRequestHeader("Content-Type", "application/x-www-form-urlencoded;charset=UTF-8");

    //5.设置参数:
    method.setParameter("username", "cgx");
    method.setParameter("password", "123456");

    client.executeMethod(method);//执行post请求
    //6.得到响应结果信息
    System.out.println(method.getStatusLine());
    System.out.println(method.getResponseBodyAsString());
}
```

Content-Type: application/x-www-form-urlencoded;charset=UTF-8

主要是指定为表单的 POST 提交方式

设置参数的第二种方式：

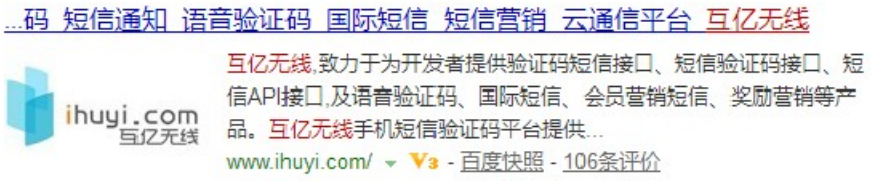
```
//5.设置参数:
NameValuePair[] params = {new NameValuePair("username","cgx"),
                          new NameValuePair("password","123456")
};
method.setRequestBody(params);
```

八．手机短信验证码

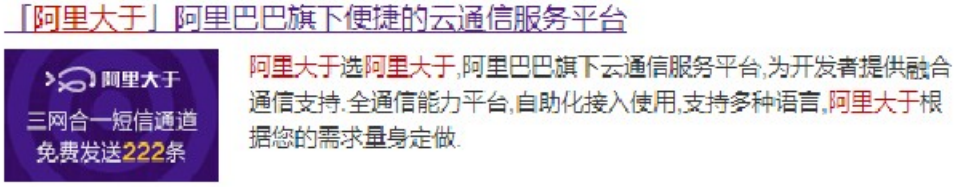
1.手机短信验证概述

近些年来，手机短信验证变得越来越流行，并且这种方式还在不断发展之中，将来的认证方式会变得越来越广，目前的手机短信发送平台也有很多。我们选择使用：吉信通，阿里大于，互亿无线。

互亿无线短信平台的地址：http://sms.ihuyi.com



“阿里大于”：www.alidayu.com

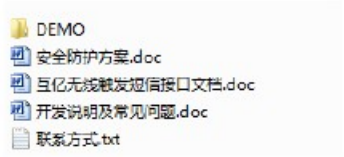


2.使用步骤

1.注册账号

2.登录成功后

3.下载开发接口



4.将 Java 代码导入 Eclipse 中

3.使用 HttpClient 测试短信发送代码

```
private static String Url = "http://106.ihuyi.cn/webservice/sms.php?method=Submit";
public static void main(String [] args) {
    HttpClient client = new HttpClient();
    PostMethod method = new PostMethod(Url);

    client.getParams().setContentCharset("GBK");
    method.setRequestHeader("ContentType","application/x-www-form-urlencoded;charset=GBK");
    int mobile_code = (int)((Math.random()*9+1)*100000);
    String content = new String("您的验证码是：" + mobile_code + "。请不要把验证码泄露给其他人。");
    NameValuePair[] data = {//提交短信
            new NameValuePair("account", "C65809272"), //查看用户名请登录用户中心->验证码、通知短信->
            new NameValuePair("password", "0fbac1987042de1f7272c4cd98db2fdf"),//查看密码请登录
               //new NameValuePair("password", util.StringUtil.MD5Encode("密码")),
            new NameValuePair("mobile", "13466575605"),
           new NameValuePair("content", content),
    };
    method.setRequestBody(data);
    try {
        client.executeMethod(method);

        String SubmitResult =method.getResponseBodyAsString();
```

此时指定的手机上会收到一条短信，这就说明短信发送成功。