**汇购工具类1.0**

**功能手册**

|  |  |
| --- | --- |
| 版本编号 | 0.1.0 |
| 文档编号 |  |
| 发布者 | 系统组 |
| 发布日期 | 2016-4 |

# 概述

## 编制目的

该接口文档手册编写在于描述汇购工具类1.0的功能说明。

## 内容概述

。

## 名称解释

## 参考文献

# 类说明



## 手机号码识别

**类**：hg.common.util MobileUtil

**方法**： public static Result checkMobile(String mobile);

**参数说明**：

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **参数名称** | **类型** | **说明** |
| 输入 | mobile | string | 要检查的手机号 |

**返回**: hg.common.util.MobileUtil.Result内部静态类的结构为

|  |  |
| --- | --- |
| **字段** | **错误描述** |
| code | 其中-1:不是合法手机号，其他值为有效手机号 |
| text | 详细文本信息 |

**示例:**

main(String[] args){

MobileUtil.checkMobile(“15305153869”);

}

**前端**：

hg-util.js

function checkMobile(mobile);

返回结构同后台。

**示例:**

<#inclide ../hg-util.js>

<script>

alert(checkMobile(“25305153869”); // 返回｛code：“-1”，“无效手机号”｝

</script>

## 加解密DES，RSA，AES

* DESUtil：

**加密：**public static byte[] encrpt（byte[] 原文，byte[] key）

**参数说明**：

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **参数名称** | **类型** | **说明** |
| 输入 | text | byte[] | 原文 |
| 输入 | key | byte[] | 密码 |

**返回值：**加密后的密文

**解密：**public static byte[] decrypt（byte[] 密文，byte[] key）

**参数说明**：

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **参数名称** | **类型** | **说明** |
| 输入 | text | byte[] | 密文文 |
| 输入 | key | byte[] | 密码 |

**返回值：**解密密后的原文

**示例：**

main(Sring[] args){

byte[] mm=DesUtil.encrypt（“你们”,”abc”.getBytes()）;

DesUtil.decrpty(mm, ”abc”.getBytes());

}

* **AESUtil：**

public static byte[]加密（byte[] 原文，byte[] key）

public static byte[[解密（byte[] 密文，byte[] key）

**用法以及示例同DESUtil。**

* RSAUtil：

**方法**

**产生密码对**：public static KeyPair genKey();

KeyPari:{pulicKey,privateKey}

**加密**:public static byte[] encrpt(byte[] 原文, byte[] key) return 密文

**解密**：public static byte[] decrpt(byte[] 密文,byte[] key) return 原文

**示例**

public static void main(String [] args){

//使用方要自行保存，密码对。以方便后续加解密

KeyPari kp = RSAUtil.genKey();

byte[] mm = RSAUtil.encript(“你们”,kp.getPrivateKey());

byte[] yy = RSAUtil.decrpt(mm, kp.getPublicKey());

}

## JSON数据转

**对象转json**

com.alibaba.fastjson.JSONObjectJSONObject.toJSONString

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **参数名称** | **类型** | **说明** |
| 输入 | object | Object | 要转换的对象 |

**返回值：**

|  |  |
| --- | --- |
| **字段** | **描述** |
| json字符串 |  |

**json转对象**

com.alibaba.fastjson.JSONObject.JSONObject.parse

**示例：**

Object o = “abc”;

JSONObject.toJSONString(o);

JSONObject.parse(“{age:1,year:1999}”)

## xml操作 读写

**读/写/删某节点**

* **功能描述：**获取某个标签下的内容(不区分大小写)

**调用格式**： XmlUtil.getNodeVal(String, String)

**参数说明**：

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **参数名称** | **类型** | **说明** |
| 输入 | str | string | 整个xml |
|  | tag | string | 标签 |

**返回值：**

|  |  |
| --- | --- |
| **字段** | **描述** |
| 节点值 |  |

* **功能描述：**删除某个标签 (不区分大小写)

**调用格式**： XmlUtil.removeNode(String, String)

**参数说明**：

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **参数名称** | **类型** | **说明** |
| 输入 | str | string | 整个xml |
|  | tag | string | 标签 |

**返回值：无。**

* **功能描述：替换**某个标签内容 (不区分大小写)

**调用格式**： XmlUtil.removeNode(String, String)

**参数说明**：

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **参数名称** | **类型** | **说明** |
| 输入 | str | string | 整个xml |
|  | tag | string | 标签 |
|  | newValue | string | 标签包含的新的内容 |

**返回值：无。**

**对象与xml转换**

注意：对象类一定要有 注解：@XmlRootElement

* **功能描述：对象转xml**

String hg.common.util.XmlUtil.beanToXML(Object)

* **功能描述：**xml转对象

Object hg.common.util.XmlUtil.XMLStringToBean(Class, String)

## 常用日期格式转化 日期运算

类路径：

hg.common.util.DateUtil.java

* **功能描述：**日期时间parse和格式化

parseDate(日期时间串)

formatDate(日期对象)

此类型方法有多个，在此不一一列举

* **功能描述：**日期时间加：

add(日期，单位，数量) //单位为本类中的常量，天，小时，分等

* **功能描述：**日期时间差:

diff(日期1，日期2，单位) //返回 日期1到 日期2的间隔（单位为本类中的静态常量，天，小时，分，秒，计算结果向下取整）

* **功能描述：**判断是否为闰年

isLeapYear(日期或整形年份)，是闰年返回true，否则返回false

* **功能描述：**日期时间先后比较

dateCompare(日期1，日期2)

日期1靠后返回true，日期参数可为日期对象和日期字符串

* **功能描述：**转为一天开始或结束时间

dateStr2BeginDate(日期)

dateStr2EndDate(日期)

**示例**：

main(String[] arg){

// yyyyMMdd格式的 格式化和解析，其他类同

DateUtil. yyyyMMdd.format(new Date());

DateUtil. yyyyMMdd.parse(“20190101”);

//日期运算

Date date1,date2;

date2= DateUtil.add(date1, DateUtil.field\_hour, 1);//加一小时

DateUtil.diff(date1,date2, DateUtil.field\_hour); //返回相差多少小时

DateUtil.dateCompare(date3, date2);//若date3在date2后返回true

//日期字符串校验

DateUtil.checkDate("2016-04-22 15:09:00", "yyyy-MM-dd HH:ss")

}

## http client封装 https 封装

static String callWebServiceGet(String url, Map para);

static String callWebServicePost(String url, Map para);

**示例：**

public static void main(String[] fsd){

HttpUtil. callWebServiceGet(<https://baidu.com,new> Map());

}

## 二维码生成和识别

**类**：QRCodeUtil

* 生成二维码：

public static void encode(String content, String imgPath,

OutputStream output, boolean needCompress) throws Exception

* 识别二维码：

public static String decode(InputStream input) throws Exception

**示例：**

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

//二维码内容

String text = "生成二维码";

//LOGO路径

String imagesPath = "test/pic/logo.jpg";

//生成二维码路径

String destPath = " test/pic/logo\_qrcode.jpg";

OutputStream out = new FileOutputStream(destPath);

InputStream input = new FileInputStream(destPath);

//生成带logo 的二维码

QRCodeUtil.encode(text, imagesPath, out, true);

//生成不带logo 的二维码

QRCodeUtil.encode(text,"",out,true);

//识别二维码

System.out.println(QRCodeUtil.decode(input));

}

## 图片操作类 如图片的裁 水印 合成

类PicUtil

* //上传

public static String uploadImage(InputStream 图片,String 图片上传地址)

* //下载

public static InputStream downloadImage(String 图片地址)

* //加图片水印 单个水印

public static InputStream waterMark(InputStream图片, InputStream 水印图片,int 水印位置)

* //加文字水印 单个水印

public static InputStream waterMark(InputStream图片, String 水印文字,int 水印位置)

* //加图片水印 水印平铺

public static InputStream waterMarkTile(InputStream图片, InputStream 水印图片,int 水印疏密度)

* //加文字水印 水印平铺

public static InputStream waterMarkTile(InputStream图片, String 水印文字,int 水印疏密度)

* // 压缩

public static InputStream compact (InputStream 图片, int 压缩百分比)

* // 剪裁

public static InputStream cut (InputStream 图片,CutArea 裁剪区域)。//

CutArea为｛x, y, width, height｝ 分别为剪裁区域的顶点坐标（相对于图片来说），和宽，高。

**示例：**

//上传

public void upload(){

File pic = new File("src/test/pic/pic.jpg");

try {

System.out.println(PicUtils.uploadImage(new FileInputStream(pic),"192.168.2.214"));

} catch (Exception e) {

e.printStackTrace();

}

}

//下载

public void down(){

try {

File outFile = new File("src/test/pic/downPic.jpg");

FileInputStream inStream = (FileInputStream) PicUtils.downImage("http://192.168.2.214/group1/M00/00/00/wKgC1lcgbDSIVY-aABFeO-k4YUEAAAATQJeZJgAEV5T717.jpg");

FileOutputStream outStream = new FileOutputStream(outFile);

byte[] inOutb = new byte[inStream.available()];

inStream.read(inOutb);

outStream.write(inOutb);

inStream.close();

outStream.close();

} catch (Exception e) {

e.printStackTrace();

}

}

//压缩

public void compactTest() {

File pic = new File("src/test/pic/pic.jpg");

try {

// 压缩

InputStream result = PicUtils.compact(new FileInputStream(pic), 99);

byte[] data = readInputStream(result);

File imageFile = new File("src/test/pic/waterMarkZip.jpg");

FileOutputStream outStream = new FileOutputStream(imageFile);

outStream.write(data);

outStream.close();

} catch (Exception e) {

e.printStackTrace();

}

}

//剪裁

public void cutTest() {

File pic = new File("src/test/pic/pic.jpg");

try {

// 剪裁

CutArea cutArea = new CutArea();

cutArea.setX(100);

cutArea.setY(100);

cutArea.setHeight(100);

cutArea.setWidth(100);

InputStream result = PicUtils.cut(new FileInputStream(pic), cutArea);

byte[] data = readInputStream(result);

File imageFile = new File("src/test/pic/cutPic.jpg");

FileOutputStream outStream = new FileOutputStream(imageFile);

outStream.write(data);

outStream.close();

} catch (Exception e) {

e.printStackTrace();

}

}

//添加水印

public void watermarkTest() {

File pic = new File("src/test/pic/pic.jpg");

try {

// 图片水印

File watermark = new File("src/test/pic/watermark.png");

InputStream result = PicUtils.waterMark(new FileInputStream(pic),new FileInputStream(watermark));

byte[] data = readInputStream(result);

File imageFile = new File("src/test/pic/waterMarkPic.jpg");

FileOutputStream outStream = new FileOutputStream(imageFile);

outStream.write(data);

outStream.close();

// 文字水印

InputStream result1 = PicUtils.waterMark(new FileInputStream(pic),

"String");

byte[] data1 = readInputStream(result1);

File imageFile1 = new File("src/test/pic/waterMarkString.jpg");

FileOutputStream outStream1 = new FileOutputStream(imageFile1);

outStream1.write(data1);

outStream1.close();

} catch (Exception e) {

e.printStackTrace();

}

}

前端js：

picUtil.js

//裁剪.

function cutPic(图片选择器,回调函数); //回调函数可以得到CutArea

**示例：**

<script type="text/javascript" src=" jquery.js"></script>

<script src="../PicCut.js" type="text/javascript"></script>

<div id=img> <img src=””/></div>

<script>

picCut($("#pic0"),0,function(s){

//选择结束后显示区域坐标

alert("select:"+s.x+","+s.y+","+s.x2+","+s.y2+","+s.w+","+s.h);

});

</script>

完整实例见hg-common-0.8 src/main/js目录中，实例文件位于其demo文件夹下