比瓴科技数据脱敏组件使用指南V1.1

2023-04-06

|  |  |  |  |
| --- | --- | --- | --- |
| ■版本变更记录 | | | |
| 时间 | 版本 | 说明 | 修改人 |
| 2023-04-06 | V1.0 | 文档创建 | 申晏键 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| ■适用性声明 |
| 本文档适用于信息系统软件开发，阅读对象为研发相关人员。 |

目录

[一、 文档说明](#_Toc2016142020)

[1.1. 编制说明](#_Toc63612822)

[二、组件介绍](#_Toc1446207238)

[2.1. 数据脱敏](#_Toc123839900)

[2.2. 数据脱敏防护组件调用方式如下：](#_Toc2105716871)

[1、对象域值脱敏](#_Toc1760181207)

[2.3. 更新](#_Toc1908790017)

[2.4. 打包](#_Toc1648755184)

# 文档说明

## 编制说明

本文档主要目的是指导研发人员方便调用相关组件进行开发测试。

在web系统研发过程中，安全控制一直是治理的重要环节，数据脱敏属于安全控制的范畴。对互联网公司、传统行业来说，数据安全一直是极为重视和敏感的话题。数据脱敏是指对某些敏感信息通过脱敏规则进行数据的变形，实现敏感隐私数据的可靠保护。涉及客户安全数据或者一些商业性敏感数据，如身份证号、手机号、卡号、客户号等个人信息按照相关部门规定，都需要进行数据脱敏。

基于Java反射api及注解、简单易用、支持各种复杂数据结构的数据脱敏库，包含但不限于以下类型的数据脱敏手机号码：

* 中英文名称
* 身份证号码
* 银行卡号码
* 手机号码
* 固定号码
* 邮箱
* 默认缺省

# 二、组件介绍

## 数据脱敏

使用方法：通过maven引入已经编译好的lib包使用。

脱敏成功提示码和返回信息：

{

"Code":0,

"Message":ReturnData //ReturnData为脱敏后的数据，例如字符串，数组等。

}

脱敏失败提示码和返回信息：

{

"Code":1,

"Message":"数据格式错误，数据脱敏失败!"

}

### 2.1.1 信息缺省屏蔽

显示需脱敏数据的前1/3内容和后1/3内容（向下取整），其它内容用\*代替。

Java中Test用例如下：

System.*out*.println("演示缺省值脱敏");  
String omission1\_raw = "shenyanjian@gmail.com";  
System.*out*.println("测试:"+omission1\_raw);  
JsonObject omission1 = Sensitive.*desensitize*(omission1\_raw, new TypeToken<@CommonOmissionSensitive String>() {  
});  
System.*out*.println(omission1);  
String omission2\_raw = "sm";  
System.*out*.println("测试:"+omission2\_raw);  
JsonObject omission2 = Sensitive.*desensitize*(omission2\_raw, new TypeToken<@CommonOmissionSensitive String>() {  
});  
System.*out*.println(omission2);

结果如下：

演示缺省值脱敏

测试:shenyanjian@gmail.com

{"Code":0,"Message":"\*\*\*\*\*\*\*jian@gm\*\*\*\*\*\*\*"}

测试:sm

{"Code":1,"Message":"Wrong Data"}

### 2.1.2 银行卡号

展示前4位+\*\*\*\*\*\*+后4位。

先对需脱敏数据（银行卡卡号）的长度进行判断：如果需脱敏数据的长度不在14~19之间，则返回通用错误提示码和提示信息，即：提示码：“1”，提示信息：“数据格式错误，数据脱敏失败！”。

当需脱敏数据（银行卡卡号）的长度在14（含14）~19（含19）之间时，提示码：“0”，提示信息：为脱敏后的数据。

JavaTest用例：

*// bankId*JsonObject bankId = Sensitive.*desensitize*("4926309485126374", new TypeToken<@BankCardNumberSensitive String>() {  
});  
System.*out*.println("演示银行卡脱敏");  
System.*out*.println(bankId);  
*// bankId too long*String bankId\_too\_long\_raw = "4926302222229485126374";  
JsonObject bankId\_too\_long = Sensitive.*desensitize*(bankId\_too\_long\_raw, new TypeToken<@BankCardNumberSensitive String>() {  
});  
System.*out*.println("演示银行卡数据过长脱敏 "+bankId\_too\_long\_raw);  
System.*out*.println(bankId\_too\_long);

结果如下：

演示银行卡脱敏

{"Code":0,"Message":"4926\*\*\*\*\*\*\*\*6374"}

演示银行卡数据过长脱敏 4926302222229485126374

{"Code":1,"Message":"wrong!"}

### 2.1.3 中英文姓名

对中文姓名：

1. 单名，屏蔽名字，比如：张三---->张\*。
2. 双名，屏蔽中间，比如：张某三---->张\*三。
3. 其它，阿不都沙拉克---->阿不\*\*\*\*。

对英文姓名：

SenRanja Hunt显示为S\*\*\*\*\*\*\* H\*\*\*

Bunny Polt显示为B\*\*\*\* P\*\*\*

Innovation显示为I\*\*\*\*\*\*\*\*\*

Java Test用例如下：

// Name

JsonObject name0 = Sensitive.desensitize("三", new TypeToken<@NameSensitive String>() {

});

JsonObject name1 = Sensitive.desensitize("张三", new TypeToken<@NameSensitive String>() {

});

JsonObject name2 = Sensitive.desensitize("张某三", new TypeToken<@NameSensitive String>() {

});

JsonObject name3 = Sensitive.desensitize("阿不都沙拉克", new TypeToken<@NameSensitive String>() {

});

System.out.println("演示姓名脱敏");

System.out.println(name0);

System.out.println(name1);

System.out.println(name2);

System.out.println(name3);

JsonObject name4 = Sensitive.desensitize("SenRanja Hunt", new TypeToken<@NameSensitive String>() {

});

JsonObject name5 = Sensitive.desensitize("Bunny Polt", new TypeToken<@NameSensitive String>() {

});

JsonObject name6 = Sensitive.desensitize("Innovation", new TypeToken<@NameSensitive String>() {

});

System.out.println(name4);

System.out.println(name5);

System.out.println(name6);

输出如下：

演示姓名脱敏

{"Code":1,"Message":"Wrong Data!"}

{"Code":0,"Message":"张\*"}

{"Code":0,"Message":"张\*三"}

{"Code":0,"Message":"阿不\*\*\*\*"}

{"Code":0,"Message":"S\*\*\*\*\*\*\* H\*\*\*"}

{"Code":0,"Message":"B\*\*\*\* P\*\*\*"}

{"Code":0,"Message":"I\*\*\*\*\*\*\*\*\*"}

### 2.1.4 身份证号

提供两种脱敏规则，保留头3位，比如330100199903032000---->330\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*；另一种展示头3和尾号4位，屏蔽中间，比如：3301001999003032000---->330\*\*\*\*\*\*\*\*\*\*\*\*\*\*2000。

调用组件时对注解传参，默认只显示头三位，即默认传参为desensitizeType = IdCardNumberSensitive.DesensitizeType.TYPE\_1。如果要显示头三位和后四位，指定传参desensitizeType = IdCardNumberSensitive.DesensitizeType.TYPE\_2。

Java Test 用例如下：

System.out.println("演示身份证号脱敏");

String idcard1\_raw = "141181199904230063";

System.out.println("测试用身份证号 方式一 仅显示前三位:"+idcard1\_raw);

JsonObject idcard1 = Sensitive.desensitize(idcard1\_raw, new TypeToken<@IdCardNumberSensitive String>() {

});

System.out.println(idcard1);

String idcard3\_raw = "141181199904230063";

System.out.println("测试用身份证号 方式二 显示前三位和后四位:"+idcard3\_raw);

JsonObject idcard3 = Sensitive.desensitize(idcard3\_raw, new TypeToken<@IdCardNumberSensitive(desensitizeType = IdCardNumberSensitive.DesensitizeType.TYPE\_2) String>() {

});

System.out.println(idcard3);

String idcard2\_raw = "1411811999042300623";

System.out.println("测试用身份证号:"+idcard2\_raw);

JsonObject idcard2 = Sensitive.desensitize(idcard2\_raw, new TypeToken<@IdCardNumberSensitive String>() {

});

System.out.println(idcard2);

输出如下：

演示身份证号脱敏

测试用身份证号 方式一 仅显示前三位:141181199904230063

{"Code":0,"Message":"141\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"}

测试用身份证号 方式二 显示前三位和后四位:141181199904230063

{"Code":0,"Message":"141\*\*\*\*\*\*\*\*\*\*\*0063"}

测试用身份证号:1411811999042300623

{"Code":1,"Message":"Wrong Data!"}

### 2.1.5 手机号

提供四种脱敏规则，针对不同地区的手机号进行脱敏。分别包括：大陆手机号、香港/澳门手机号、台湾手机号、其他海外地区手机号。需要调用方根据实际使用场景选择性调用。

仅保留手机尾号后4位；或者除区号外，至少隐藏中间4位。

1. 大陆：显示前3位+\*\*\*\*+后4位，比如：137\*\*\*\*9050;
2. 香港、澳门：显示前2位+\*\*\*\*+后2位，比如：90\*\*\*\*85;
3. 台湾：显示前2位+\*\*\*\*+后3位，比如：90\*\*\*\*856;
4. 其它海外地区：使用缺省屏蔽规则。

Java Test用例如下：

System.out.println("演示手机号脱敏");

String tel1\_raw = "18536864913";

System.out.println("测试: 大陆手机"+tel1\_raw);

JsonObject tel1 = Sensitive.desensitize(tel1\_raw, new TypeToken<@PhoneNumberSensitive String>() {

});

System.out.println(tel1);

String tel2\_raw = "90888812";

System.out.println("测试: 港澳手机"+tel2\_raw);

JsonObject tel2 = Sensitive.desensitize(tel2\_raw, new TypeToken<@PhoneNumberSensitive String>() {

});

System.out.println(tel2);

String tel3\_raw = "908888123";

System.out.println("测试: 台湾手机"+tel3\_raw);

JsonObject tel3 = Sensitive.desensitize(tel3\_raw, new TypeToken<@PhoneNumberSensitive String>() {

});

System.out.println(tel3);

String tel4\_raw = "908888123222222";

System.out.println("测试: 其他地区手机"+tel4\_raw);

JsonObject tel4 = Sensitive.desensitize(tel4\_raw, new TypeToken<@PhoneNumberSensitive String>() {

});

System.out.println(tel4);

输出如下：

演示手机号脱敏

测试: 大陆手机18536864913

{"Code":0,"Message":"185\*\*\*\*4913"}

测试: 港澳手机90888812

{"Code":0,"Message":"90\*\*\*\*12"}

测试: 台湾手机908888123

{"Code":0,"Message":"90\*\*\*\*123"}

测试: 其他地区手机908888123222222

{"Code":0,"Message":"\*\*\*\*\*81232\*\*\*\*\*"}

### 2.1.6 固定电话

对于包括含区号的电话号码串，仅保留前4位和后2位。如：“86-10-66778899”，脱敏后为：“86-10-\*\*\*\*\*\*99”。

*// LandLineNumber*System.*out*.println("演示固定电话脱敏");  
String LandLineNumber1\_raw = "(555) 123-4567";  
JsonObject landLineNumber1 = Sensitive.*desensitize*(LandLineNumber1\_raw, new TypeToken<@LandLinePhoneSensitive String>() {  
});  
System.*out*.println("测试固定电话: "+LandLineNumber1\_raw);  
System.*out*.println(landLineNumber1);  
String LandLineNumber2\_raw = "86-10-66778899";  
JsonObject landLineNumber2 = Sensitive.*desensitize*(LandLineNumber2\_raw, new TypeToken<@LandLinePhoneSensitive String>() {  
});  
System.*out*.println("测试固定电话: "+LandLineNumber2\_raw);  
System.*out*.println(landLineNumber2);

输出如下：

演示固定电话脱敏

测试固定电话: (555) 123-4567

{"Code":1,"Message":"Wrong Data!"}

测试固定电话: 86-10-66778899

{"Code":0,"Message":"86-10\*\*\*\*\*\*\*99"}

### 2.1.7 电子邮箱

1. @前面的字符长度大于3位，展示前3位，3位后展示3个\*，@后面完整显示，比如：[zhangsan@cib.com.cn则展示为zha\*\*\*@cib.com.cn；](mailto:zhangsan@cib.com.cn则展示为zha***@cib.com.cn；)
2. @前面的字符长度若小于等于3位，则全部展示，@前面\*\*\*，比如：li@cib.com.cn则展示为li\*\*\*@cib.com.cn。

Java Test用例如下：

*// email*String email\_raw = "123456@qq.com";  
JsonObject email = Sensitive.*desensitize*(email\_raw, new TypeToken<@EmailSensitive String>() {  
});  
System.*out*.println("演示邮箱脱敏 "+email\_raw);  
System.*out*.println(email);

输出如下：

演示邮箱脱敏 123456@qq.com

{"Code":0,"Message":"123\*\*\*@qq.com"}

## 数据脱敏防护组件调用方式如下：

### 2.2.1依赖

调用组件生成的json对象为com.google.gson.JsonObject。

本组件的pom.xml如下：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>cn.blingsec</groupId>  <artifactId>ttest</artifactId>  <version>1.0.0</version>  <properties>  <maven.compiler.source>8</maven.compiler.source>  <maven.compiler.target>8</maven.compiler.target>  </properties>  <dependencies>  <dependency>  <groupId>cn.blingsec</groupId>  <artifactId>SSLSocket</artifactId>  <version>1.0.0</version>  </dependency>  <dependency>  <groupId>cn.blingsec</groupId>  <artifactId>desensitization</artifactId>  <version>1.1</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <artifactId>maven-assembly-plugin</artifactId>  <configuration>  <descriptorRefs>  <descriptorRef>jar-with-dependencies</descriptorRef>  </descriptorRefs>  </configuration>  </plugin>  </plugins>  </build>  </project> |

### 2.2.2调用方法

|  |
| --- |
| package cn.blingsec.tttteeessttttt;  *// 脱敏组件* import cn.blingsec.desensitization.Sensitive; import cn.blingsec.desensitization.annotation.\*; import cn.blingsec.desensitization.support.TypeToken; import com.google.gson.JsonObject;  public class Main {  public static void main(String[] args) throws Exception {  *// 【脱敏组件】   // email* String email\_raw = "123456@qq.com";  JsonObject email = Sensitive.*desensitize*(email\_raw, new TypeToken<@EmailSensitive String>() {  });  System.*out*.println("演示邮箱脱敏 "+email\_raw);  System.*out*.println(email.get("Code"));    *// LandLineNumber* System.*out*.println("演示固定电话脱敏");  String LandLineNumber1\_raw = "(555) 123-4567";  JsonObject landLineNumber1 = Sensitive.*desensitize*(LandLineNumber1\_raw, new TypeToken<@LandLinePhoneSensitive String>() {  });  System.*out*.println("测试固定电话: "+LandLineNumber1\_raw);  System.*out*.println(landLineNumber1);  String LandLineNumber2\_raw = "86-10-66778899";  JsonObject landLineNumber2 = Sensitive.*desensitize*(LandLineNumber2\_raw, new TypeToken<@LandLinePhoneSensitive String>() {  });  System.*out*.println("测试固定电话: "+LandLineNumber2\_raw);  System.*out*.println(landLineNumber2);   *// bankId* JsonObject bankId = Sensitive.*desensitize*("4926309485126374", new TypeToken<@BankCardNumberSensitive String>() {  });  System.*out*.println("演示银行卡脱敏");  System.*out*.println(bankId);  *// bankId too long* String bankId\_too\_long\_raw = "4926302222229485126374";  JsonObject bankId\_too\_long = Sensitive.*desensitize*(bankId\_too\_long\_raw, new TypeToken<@BankCardNumberSensitive String>() {  });  System.*out*.println("演示银行卡数据过长脱敏 "+bankId\_too\_long\_raw);  System.*out*.println(bankId\_too\_long);  *// Name* JsonObject name0 = Sensitive.*desensitize*("三", new TypeToken<@ChineseNameSensitive String>() {  });  JsonObject name1 = Sensitive.*desensitize*("张三", new TypeToken<@ChineseNameSensitive String>() {  });  JsonObject name2 = Sensitive.*desensitize*("张某三", new TypeToken<@ChineseNameSensitive String>() {  });  JsonObject name3 = Sensitive.*desensitize*("阿不都沙拉克", new TypeToken<@ChineseNameSensitive String>() {  });  System.*out*.println("演示姓名脱敏");  System.*out*.println(name0);  System.*out*.println(name1);  System.*out*.println(name2);  System.*out*.println(name3);   System.*out*.println("演示身份证号脱敏");  String idcard1\_raw = "141181199904230063";  System.*out*.println("测试用身份证号 方式一 仅显示前三位:"+idcard1\_raw);  JsonObject idcard1 = Sensitive.*desensitize*(idcard1\_raw, new TypeToken<@IdCardNumberSensitive String>() {  });  System.*out*.println(idcard1);  String idcard3\_raw = "141181199904230063";  System.*out*.println("测试用身份证号 方式二 显示前三位和后四位:"+idcard3\_raw);  JsonObject idcard3 = Sensitive.*desensitize*(idcard3\_raw, new TypeToken<@IdCardNumberSensitive(desensitizeType = IdCardNumberSensitive.DesensitizeType.*TYPE\_2*) String>() {  });  System.*out*.println(idcard3);  String idcard2\_raw = "1411811999042300623";  System.*out*.println("测试用身份证号:"+idcard2\_raw);  JsonObject idcard2 = Sensitive.*desensitize*(idcard2\_raw, new TypeToken<@IdCardNumberSensitive String>() {  });  System.*out*.println(idcard2);   System.*out*.println("演示邮箱脱敏");  String email1\_raw = "shenyanjian@gmail.com";  System.*out*.println("测试用邮箱:"+email1\_raw);  JsonObject email1 = Sensitive.*desensitize*(email1\_raw, new TypeToken<@EmailSensitive String>() {  });  System.*out*.println(email1);  String email2\_raw = "l@com";  System.*out*.println("测试用邮箱:"+email2\_raw);  JsonObject email2 = Sensitive.*desensitize*(email2\_raw, new TypeToken<@EmailSensitive String>() {  });  System.*out*.println(email2);  String email3\_raw = "l.com";  System.*out*.println("测试用邮箱:"+email3\_raw);  JsonObject email3 = Sensitive.*desensitize*(email3\_raw, new TypeToken<@EmailSensitive String>() {  });  System.*out*.println(email3);   System.*out*.println("演示缺省值脱敏");  String omission1\_raw = "shenyanjian@gmail.com";  System.*out*.println("测试:"+omission1\_raw);  JsonObject omission1 = Sensitive.*desensitize*(omission1\_raw, new TypeToken<@CommonOmissionSensitive String>() {  });  System.*out*.println(omission1);  String omission2\_raw = "sm";  System.*out*.println("测试:"+omission2\_raw);  JsonObject omission2 = Sensitive.*desensitize*(omission2\_raw, new TypeToken<@CommonOmissionSensitive String>() {  });  System.*out*.println(omission2);   System.*out*.println("演示手机号脱敏");  String tel1\_raw = "18536864913";  System.*out*.println("测试: 大陆手机"+tel1\_raw);  JsonObject tel1 = Sensitive.*desensitize*(tel1\_raw, new TypeToken<@PhoneNumberSensitive String>() {  });  System.*out*.println(tel1);  String tel2\_raw = "90888812";  System.*out*.println("测试: 港澳手机"+tel2\_raw);  JsonObject tel2 = Sensitive.*desensitize*(tel2\_raw, new TypeToken<@PhoneNumberSensitive String>() {  });  System.*out*.println(tel2);  String tel3\_raw = "908888123";  System.*out*.println("测试: 台湾手机"+tel3\_raw);  JsonObject tel3 = Sensitive.*desensitize*(tel3\_raw, new TypeToken<@PhoneNumberSensitive String>() {  });  System.*out*.println(tel3);  String tel4\_raw = "908888123222222";  System.*out*.println("测试: 其他地区手机"+tel4\_raw);  JsonObject tel4 = Sensitive.*desensitize*(tel4\_raw, new TypeToken<@PhoneNumberSensitive String>() {  });  System.*out*.println(tel4);   } } |

只需要在敏感数据字段上标记相应类型的敏感注解，例如@NameSensitive、@EmailSensitive等注解。

## 打包

使用maven进行打包，执行以下指令进行打包：mvn clean package assembly:single。