比瓴科技数据脱敏组件使用指南V1.0

2021-06-04

|  |  |  |  |
| --- | --- | --- | --- |
| ■版本变更记录 | | | |
| 时间 | 版本 | 说明 | 修改人 |
| 2022-07-29 | V1.0 | 文档创建 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| ■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 4](#_Toc2016142020)

[1.1. 编制说明 4](#_Toc63612822)

[二、组件介绍 5](#_Toc1446207238)

[2.1. 数据脱敏 5](#_Toc123839900)

[2.2. 数据脱敏防护组件调用方式如下： 5](#_Toc2105716871)

[1、对象域值脱敏 5](#_Toc1760181207)

[2.3. 更新 7](#_Toc1908790017)

[2.4. 打包 7](#_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、零依赖、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

* 邮箱
* 手机号码
* 中文名称
* 身份证号码
* 银行卡号码
* 密码
* 统一社会信用代码
* 任意CharSequence类型的值
* 级联脱敏

# 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，使用内置的规则包，或者自行配置规则包使用源码重新打包。

说明：文件上传组件通过检测上传的文件名，文件类型，文件大小等来识别恶意文件。

## 数据脱敏防护组件调用方式如下：

### **1、对象域值脱敏**

下面是一个Child类，其中包含了一些敏感域以及一些嵌套的敏感域

|  |
| --- |
| public class Child {  @ChineseNameSensitive  private String name = "李富贵";  @IdCardNumberSensitive  private String idCardNumber = "321181199301096000";  @UsccSensitive  private String unifiedSocialCreditCode = "91310106575855456U";  @CharSequenceSensitive  private String string = "123456";  @EmailSensitive  private String email = "123456@qq.com";  @PasswordSensitive  private String password = "123456";  @CascadeSensitive  private Mother mother = new Mother();  @CascadeSensitive  private Father father = new Father();  private @PasswordSensitive  String[] passwords = {"123456", "1234567", "12345678"};  private List<@CascadeSensitive Parent> parents1 = Stream.of(new Father(), new Mother()).collect(Collectors.toList());  private List<@EmailSensitive String> emails1 = Stream.of("123456@qq.com", "1234567@qq.com", "1234568@qq.com").collect(Collectors.toList());  private Map<@ChineseNameSensitive String, @EmailSensitive String> emails2 = Stream.of("张三", "李四", "小明").collect(Collectors.toMap(s -> s, s -> "123456@qq.com"));  } |

只需要在敏感数据字段上标记相应类型的敏感注解，例如@ChineseNameSensitive、@EmailSensitive等注解，如果你的数据字段是需要级联脱敏的对象，你只需要在该字段上标注@CascadeSensitive 注解，最后调用以下方法即可擦除对象中的所有敏感信息然后返回一个新的Child对象。

|  |
| --- |
| Child child=Sensitive.desensitize(new Child()); |

1. 值脱敏

可能你的敏感信息是一个字符串类型的值或者是一个Collection、Array、Map之类的值，同样擦除它们的敏感信息也很简单

|  |
| --- |
| static void desensitize(){  // String  System.out.printf("字符串脱敏: %s%n",Sensitive.desensitize("123456@qq.com",new TypeToken<@EmailSensitive String>(){  }));  // Collection  System.out.printf("集合脱敏: %s%n",Sensitive.desensitize(Stream.of("123456@qq.com","1234567@qq.com","1234568@qq.com").collect(Collectors.toList()),  new TypeToken<List<@EmailSensitive String>>(){  }));  // Array  System.out.printf("数组脱敏: %s%n",Arrays.toString(Sensitive.desensitize(new String[]{"123456@qq.com","1234567@qq.com","12345678@qq.com"},  new TypeToken<@EmailSensitive String[]>(){  })));  // Map  System.out.printf("Map脱敏: %s%n",Sensitive.desensitize(Stream.of("张三","李四","小明").collect(Collectors.toMap(s->s,s->"123456@qq.com")),  new TypeToken<Map<@ChineseNameSensitive String, @EmailSensitive String>>(){  }));  } |

## 更新

无

## 打包

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

注意：由于项目未使用其他依赖信息，直接调用desensitization-1.0.jar即可。