

🏠 / 开发指南 / 系统手册

👤 芋道源码 📅 2023-01-21

数据脱敏

接口在返回一些**敏感**或**隐私**数据时，是需要进行脱敏处理，通常的手段是使用 * 隐藏一部分数据。例如说：

类型	原始数据	脱敏数据
手机	13248765917	132****5917
身份证	530321199204074611	530321*****11
银行卡	9988002866797031	998800*****31

1. 脱敏组件

`yudao-spring-boot-starter-desensitize`  基于 Jackson 拓展，只需要在字段上添加脱敏注解，即可实现对该字段进行脱敏。

使用步骤如下：

① 在 `pom.xml` 引入该依赖，如下所示：

```
<dependency>
  <groupId>cn.iocoder.boot</groupId>
  <artifactId>yudao-spring-boot-starter-desensitize</artifactId>
</dependency>
```

② 在字段上添加脱敏注解。如下所示：

```
@Data
public static class DesensitizeDemo {

    @MobileDesensitize // 手机号的脱敏注解
    private String phoneNumber;

}
```

2. 内置脱敏注解

根据不同的脱敏处理方式，项目内置了两类脱敏注解：正则脱敏、滑块脱敏。

2.1 regex 正则脱敏

2.1.1 @RegexDesensitize 注解

正则脱敏注解 `@RegexDesensitize` [🔗](#)：根据**正则表达式**，将原始数据进行**替换**处理。

```
public @interface RegexDesensitize {  
  
    /**  
     * 匹配的正则表达式（默认匹配所有）  
     */  
    String regex() default "[\\s\\S]*$";  
  
    /**  
     * 替换规则，会将匹配到的字符串全部替换成 replacer  
     */  
    String replacer() default "*****";  
  
}
```

例如说：`regex=123; replacer=*****` 表示将 `123` 替换为 `*****`

- 原始字符串 123456789
- 脱敏后字符串 *****456789

2.1.2 其它正则脱敏注解

项目内置了其它基于正则脱敏的常用注解，无需手动填写 `regex`、`replacer` 属性，更加方便。例如说：


```
@Data  
public static class DesensitizeDemo {  
  
    @EmailDesensitize  
    private String email;  
  
}
```

所有注解如下：

注解	原始数据	脱敏数据
@EmailDesensitize 	example@gmail.com	e*****@gmail.com

2.2 slider 滑块脱敏

2.2.1 @SliderDesensitize 注解

滑块脱敏注解 [@SliderDesensitize](#) ：根据设置的左右明文字符长度，中间部分全部替换为 *。

例如说：`prefixKeep=3; suffixKeep=4; replacer=*` 表示前 3 后 4 保持明文，中间都替换成 *

- 原始字符串 13248765917
- 脱敏后字符串 132*****5917

2.2.2 其它滑块脱敏注解

项目内置了其它基于滑块脱敏的常用注解，无需手动填写 `prefixKeep`、`suffixKeep`、`replacer` 属性，更加方便。例如说：

```
@Data
public static class DesensitizeDemo {

    @MobileDesensitize
    private String mobile;

}
```

所有注解如下：

注解	原始数据	脱敏数据
@MobileDesensitize 	13248765917	132*****5917
@FixedPhoneDesensitize 	01086551122	0108*****22
@BankCardDesensitize 	9988002866797031	998800*****31
@PasswordDesensitize 	123456	*****
@CarLicenseDesensitize 	粤A66666	粤A6***6
@ChineseNameDesensitize 	刘子豪	刘**
@IdCardDesensitize 	530321199204074611	530321*****11

3. 自定义脱敏注解

如果内置的注解无法满足你的需求，只需要自定义一个脱敏注解，并实现它的脱敏处理器即可。

例如说，我们要实现一个新的脱敏处理方法，将编号使用 MD5 或 SHA256 计算后返回。步骤如下：

① 创建 `@DigestDesensitize` 注解，使用 `@DesensitizeBy` 标记它使用的处理器。代码如下：

```
import cn.iocoder.yudao.framework.desensitize.core.base.annotation.DesensitizeBy;
import cn.iocoder.yudao.framework.desensitize.core.handler.DigestHandler;
import com.fasterxml.jackson.annotation.JacksonAnnotationsInside;

import java.lang.annotation.*;

@Documented
@Target({ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@JacksonAnnotationsInside
@DesensitizeBy(handler = DigestHandler.class) // 使用 @DesensitizeBy 设置它的处理器
public @interface DigestDesensitize {

    /**
     * 摘要算法，例如说：MD5、SHA256
     */
    String algorithm() default "md5";

}
```

② 创建 `DigestHandler` 类，实现 `DigestHandler` 接口，将编号使用 MD5 或 SHA256 处理。代码如下：

```
import cn.hutool.crypto.digest.DigestUtil;
import cn.iocoder.yudao.framework.desensitize.core.annotation.DigestDesensitize;
import cn.iocoder.yudao.framework.desensitize.core.base.handler.DesensitizationHandler;

public class DigestHandler implements DesensitizationHandler<DigestDesensitize> {

    @Override
```

```

    public String desensitize(String origin, DigestDesensitize annotation) {
        String algorithm = annotation.algorithm();
        return DigestUtil.digester(algorithm).digestHex(origin);
    }
}

```

友情提示:

① 如果自定义的是基于正则脱敏的注解，可选择继承 [AbstractRegexDesensitizationHandler](#) 处理器。

① 如果自定义的是基于滑块脱敏的注解，可选择继承 [AbstractSliderDesensitizationHandler](#) 处理器。

③ 在需要使用的字段上，添加 `@DigestDesensitize` 注解。示例代码如下：

```

@Data
public static class DesensitizeDemo {

    @DigestDesensitize
    private String email;

}

```

完事~

4. 脱敏工具类

Hutool 提供了 [DesensitizedUtil](#) 脱敏工具类，支持用户 ID、中文名、身份证、座机号、手机号、地址、电子邮件、密码、车牌、银行卡号的脱敏处理。

使用方式，代码如下：

```

DesensitizedUtil.desensitized("100", DesensitizedUtils.DesensitizedType.USER_ID)
DesensitizedUtil.desensitized("段正淳", DesensitizedUtils.DesensitizedType.CHINESE_NAME)
DesensitizedUtil.desensitized("51343620000320711X", DesensitizedUtils.DesensitizedType.ID_CARD)
DesensitizedUtil.desensitized("09157518479", DesensitizedUtils.DesensitizedType.PHONE)
DesensitizedUtil.desensitized("18049531999", DesensitizedUtils.DesensitizedType.PASS)
DesensitizedUtil.desensitized("北京市海淀区马连洼街道289号", DesensitizedUtils.DesensitizedType.ADDRESS)
DesensitizedUtil.desensitized("duandazhi-jack@gmail.com.cn", DesensitizedUtils.DesensitizedType.EMAIL)
DesensitizedUtil.desensitized("1234567890", DesensitizedUtils.DesensitizedType.PASSWORD)
DesensitizedUtil.desensitized("苏D40000", DesensitizedUtils.DesensitizedType.CAR)

```

适合场景，逻辑里需要直接对某个变量进行脱敏处理，然后打印 logger 日志，或者存储到数据库中。

← 站内信配置

敏感词→

