

Spring Boot和MyBatis框架实现返回数据的脱敏处理

作者：微信公众号：【架构师老卢】

11-2 13:7

221



概述：在 Spring Boot 和 MyBatis 中实现返回数据的脱敏处理是一种常见的需求，特别是在处理敏感数据时。数据脱敏是一种对数据进行部分隐藏或修改以保护隐私的方法。本文将详细介绍如何在 Spring Boot 和 MyBatis 中实现数据脱敏，并使用 Hutool 工具库来简化脱敏处理，并提供示例代码。### 为什么需要数据脱敏？数据脱敏是一种保护隐私数据的重要手段。当应用程序需要返回敏感信息，如用户手机号码、邮箱地址、银行卡号等时，通常需要对这些信息进行脱敏，以避免泄露用户的敏感信息。脱敏可以帮助保护用户的隐私，降低数据泄露的风险。### 数据脱敏策略 在数据脱敏处理中

在 Spring Boot 和 MyBatis 中实现返回数据的脱敏处理是一种常见的需求，特别是在处理敏感数据时。数据脱敏是一种对数据进行部分隐藏或修改以保护隐私的方法。本文将详细介绍如何在 Spring Boot 和 MyBatis 中实现数据脱敏，并使用 Hutool 工具库来简化脱敏处理，并提供示例代码。

为什么需要数据脱敏？

数据脱敏是一种保护隐私数据的重要手段。当应用程序需要返回敏感信息，如用户手机号码、邮箱地址、银行卡号等时，通常需要对这些信息进行脱敏，以避免泄露用户的敏感信息。脱敏可以帮助保护用户的隐私，降低数据泄露的风险。

数据脱敏策略

在数据脱敏处理中，通常会采用一些策略来修改或隐藏敏感数据，而不影响数据的可用性。以下是一些常见的数据脱敏策略：

部分隐藏： 只显示数据的部分字符，通常是隐藏中间的字符，例如将手机号码 "12345678901" 脱敏为 "123****8901"。

字符替换： 将数据中的字符替换为特定字符，如将邮箱地址中的 "@" 替换为 "[at]"。

数据截断： 截断数据，只显示部分内容，例如只显示姓名的前几个字符，如 "John Doe" 脱敏为 "Jo..."。

数据加密： 对敏感数据进行加密，只有经过解密才能查看原始数据。

模糊处理： 对数据进行模糊处理，如将地址中的详细信息模糊化，只显示城市和国家信息。

Spring Boot 和 MyBatis 数据脱敏示例

以下是一个示例项目，使用 Spring Boot 和 MyBatis 实现数据返回的脱敏处理，并使用 Hutool 工具库来简化脱敏处理。

步骤1：创建 Spring Boot 项目

首先，创建一个 Spring Boot 项目并添加必要的依赖。在 pom.xml 文件中添加以下依赖：

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-web</artifactId>
4 </dependency>
5 <dependency>
6   <groupId>org.mybatis.spring.boot</groupId>
7   <artifactId>mybatis-spring-boot-starter</artifactId>
8 </dependency>
9 <dependency>
10  <groupId>mysql</groupId>
11  <artifactId>mysql-connector-java</artifactId>
12 </dependency>
13 <dependency>
14  <groupId>cn.hutool</groupId>
15  <artifactId>hutool-all</artifactId>
16  <version>5.8.4</version>
17 </dependency>
```

步骤2：创建实体类和数据库表

创建一个用户实体类 User 并在数据库中创建相应的表。表中包含用户的姓名、手机号和邮箱字段。

```
1 @Entity
2 public class User {
3   @Id
4   @GeneratedValue(strategy = GenerationType.IDENTITY)
5   private Long id;
6   private String name;
7   private String phoneNumber;
8   private String email;
9
10  // Getters and setters
11 }
```

步骤3：编写 MyBatis Mapper 接口和 XML 文件

创建 MyBatis Mapper 接口和 XML 文件来定义数据库查询。在 XML 文件中，我们将使用 SQL 的 SUBSTRING 函数来实现部分隐藏和字符替换，并使用 Hutool 工具库来简化脱敏处理。

```
1 <!-- UserMapper.xml -->
2 <mapper namespace="com.example.mapper.UserMapper">
3   <select id="findAllUsers" resultType="com.example.entity.User">
4     SELECT id, name, phoneNumber, email
5     FROM user;
6   </select>
7 </mapper>
```

```
1 public interface UserMapper {
2   List<User> findAllUsers();
3 }
```

步骤4：编写控制器

创建一个 Spring Boot 控制器来处理请求，并使用 MyBatis Mapper 来查询用户数据。在控制器中，使用 Hutool 工具库的脱敏方法来处理返回的数据。

```
1 @RestController
2 @RequestMapping("/users")
3 public class UserController {
4   @Autowired
5   private UserMapper userMapper;
6
7   @GetMapping("/all")
8   public List<User> getAllUsers() {
9     List<User> users = userMapper.findAllUsers();
10    for (User user : users) {
11      user.setPhoneNumber(StrUtil.hide(user.getPhoneNumber(), 3, 7, '*'));
12      user.setEmail(StrUtil.replace(user.getEmail(), "@", "[at]"));
13    }
14    return users;
15  }
16 }
```

步骤5：运行和测试

启动 Spring Boot 应用程序，然后访问 /users/all 路径，您将看到返回的用户数据已经经过脱敏处理。

数据脱敏的注意事项

在实际项目中，数据脱敏需要根据具体情况和法律法规进行审慎处理。以下是一些需要注意的事项：

敏感数据的类型： 确保只对敏感数据进行脱敏，而不是所有数据。

数据访问权限： 控制谁可以访问脱敏数据，确保只有授权的用户可以查看原始数据。

数据持久性： 脱敏后的数据应该是持久的，而不是在每次请求时进行脱敏处理。

合规性： 遵循适用的法律法规，如 GDPR（通用数据保护条例），以确保用户隐私得到保护。

性能： 数据脱敏可能会影响性能，因此需要进行性能测试和优化。

加密： 对于极其敏感的数据，考虑使用加密来保护数据

数据脱敏是保护用户隐私的一种重要手段，但必须谨慎处理以确保数据的安全和合规性。在 Spring Boot 和 MyBatis 中，可以轻松地实现数据脱敏，并使用 Hutool 工具库来简化脱敏处理，根据具体需求选择适当的脱敏策略。