Interblog后端

测试评估报告

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <27/04/2021> | <1.0> | 完成评估报告初稿 | 杨宇晗 |
| <27/04/2021> | <1.0> | 完成代码测试覆盖部分 | 谭世炜 |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 测试结果摘要 4

3. 基于需求的测试覆盖 4

4. 基于代码的测试覆盖 4

5. 建议措施 4,5

6. 图表 5

测试评估报告

# 简介

## 目的

编写软件测试评估报告有以下几个目的：

1. 若测试缺陷由需求设计产生，设计人员将根据测试评估报告的数据修改设计；
2. 若测试缺陷由现实操作产生，设计人员将根据测试评估报告的数据修改代码；
3. 软件测试人员、开发人员以及运维人员将根据软件测试评估报告，来最终确定项目子版本的发布时间。

## 范围

此评估报告的范围仅适用于Interblog后端项目。

## 定义、首字母缩写词和缩略语

Interblog: 为我们自己开发的类知乎博客项目

后端: 后端是在后台工作的，控制着前端的内容，主要负责程序设计架构思想，管理数据库等。后端更多的是与数据库进行交互以处理相应的业务逻辑，需要考虑的是如何实现功能、数据的存取、平台的稳定性与性能等，涉及动态语言如PHP、ASP、JSP等。

Junit : JUnit是一个Java语言的单元测试框架。它由Kent Beck和Erich Gamma建立.

## 概述

本次测试以代码覆盖率达到为目的，是基于白盒测试，使用Junit单元测试框架，分别进行了路径分析及数据流分析，若基于两种分析进行白盒测试得到的代码覆盖率达到要求，则表示测试结果为通过，否则测试为没有通过，并且基于测试结果对于我们测试的软件提出我们的建议。

# 测试结果摘要

本次测试使用基于路径分析的白盒测试和基于数据流分析的白盒测试两种方法, 测试结果如下

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试类型** | **Class覆盖率** | **Method覆盖率** | **Line覆盖率** | **总测试率** | **是否通过** |
| 基于路径分析的白盒测试 | 100% (1/1) | 100% (37/37) | 99.3% (267/269) | 99.3% (305/307) | 通过 |
| 基于数据流分析的白盒测试 | 100% (1/1) | 100% (37/37) | 99.3% (267/269) | 99.3% (305/307) | 通过 |

# 基于需求的测试覆盖

由于本次测试是基于白盒测试的测试, 而基于需求的测试为黑盒测试, 故不作涉及.

# 基于代码的测试覆盖

* 1. 基于路径的测试代码覆盖率及缺陷分布

**表1 基于路径的测试代码覆盖率及缺陷分布**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试内容** | | **测试**  **用例数** | **代码**  **覆盖率** | **缺陷数** | **缺陷率** | **备注** |
| **方法接口** | UserServiceImpl | 19 | 100% | 0 | 0% |  |
| byte2Hex | 3 | 100% | 0 | 0% |  |
| hash | 3 | 75% | 0 | 0% | 未覆盖的代码对应算法不存在的情况 |
| register | 4 | 100% | 0 | 0% |  |
| banUser | 6 | 100% | 0 | 0% |  |
| getMyInfo | 1 | 100% | 0 | 0% |  |
| getBanned | 4 | 100% | 0 | 0% |  |
| getUserIdentityInfos | 1 | 100% | 0 | 0% |  |
| setMyInfo | 5 | 100% | 0 | 0% |  |
| getUserPublicInfo | 1 | 100% | 0 | 0% |  |
| getUsers | 3 | 100% | 0 | 0% |  |
| authorize | 3 | 100% | 0 | 0% |  |
| existsEmail | 2 | 100% | 0 | 0% |  |
| existsNickname | 2 | 100% | 0 | 0% |  |
| getBannedReason | 2 | 100% | 0 | 0% |  |
| setUserRole | 2 | 100% | 0 | 0% |  |
| getBanRecords | 2 | 100% | 0 | 0% |  |
| getPosts | 4 | 100% | 0 | 0% |  |
| getAllPosts | 2 | 100% | 0 | 0% |  |
| getDeletedPosts | 2 | 100% | 0 | 0% |  |
| getHomePage | 2 | 100% | 0 | 0% |  |
| searchPostsByContent | 2 | 100% | 0 | 0% |  |
| follow | 2 | 100% | 0 | 0% |  |
| getFollows | 2 | 100% | 0 | 0% |  |
| getFollowers | 2 | 100% | 0 | 0% |  |
| getFollowed | 2 | 100% | 0 | 0% |  |
| visit | 3 | 100% | 0 | 0% |  |
| getVisitors | 1 | 100% | 0 | 0% |  |
| matchInterests | 4 | 100% | 0 | 0% |  |
| likePost | 4 | 100% | 0 | 0% |  |
| likeRemark | 3 | 100% | 0 | 0% |  |
| searchInterest | 1 | 100% | 0 | 0% |  |
| searchUserByInterest | 2 | 100% | 0 | 0% |  |
| searchUser | 4 | 100% | 0 | 0% |  |
| popularInterests | 1 | 100% | 0 | 0% |  |
| getPopular | 1 | 100% | 0 | 0% |  |
| flush | 1 | 100% | 0 | 0% |  |
|  | **总计** | 108 | 99% | 0 | 0% |  |

* 1. 基于数据流的测试代码覆盖率及缺陷分布

**表2 基于数据流的测试代码覆盖率及缺陷分布**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试内容** | | **测试**  **用例数** | **代码**  **覆盖率** | **缺陷数** | **缺陷率** | **备注** |
| **方法接口** | UserServiceImpl | 19 | 100% | 0 | 0% |  |
| byte2Hex | 3 | 100% | 0 | 0% |  |
| hash | 3 | 75% | 0 | 0% | 未覆盖的代码对应算法不存在的情况 |
| register | 4 | 100% | 0 | 0% |  |
| banUser | 6 | 100% | 0 | 0% |  |
| getMyInfo | 1 | 100% | 0 | 0% |  |
| getBanned | 4 | 100% | 0 | 0% |  |
| getUserIdentityInfos | 1 | 100% | 0 | 0% |  |
| setMyInfo | 5 | 100% | 0 | 0% |  |
| getUserPublicInfo | 1 | 100% | 0 | 0% |  |
| getUsers | 3 | 100% | 0 | 0% |  |
| authorize | 3 | 100% | 0 | 0% |  |
| existsEmail | 2 | 100% | 0 | 0% |  |
| existsNickname | 2 | 100% | 0 | 0% |  |
| getBannedReason | 2 | 100% | 0 | 0% |  |
| setUserRole | 2 | 100% | 0 | 0% |  |
| getBanRecords | 2 | 100% | 0 | 0% |  |
| getPosts | 4 | 100% | 0 | 0% |  |
| getAllPosts | 2 | 100% | 0 | 0% |  |
| getDeletedPosts | 2 | 100% | 0 | 0% |  |
| getHomePage | 2 | 100% | 0 | 0% |  |
| searchPostsByContent | 2 | 100% | 0 | 0% |  |
| follow | 2 | 100% | 0 | 0% |  |
| getFollows | 2 | 100% | 0 | 0% |  |
| getFollowers | 2 | 100% | 0 | 0% |  |
| getFollowed | 2 | 100% | 0 | 0% |  |
| visit | 3 | 100% | 0 | 0% |  |
| getVisitors | 1 | 100% | 0 | 0% |  |
| matchInterests | 4 | 100% | 0 | 0% |  |
| likePost | 4 | 100% | 0 | 0% |  |
| likeRemark | 3 | 100% | 0 | 0% |  |
| searchInterest | 1 | 100% | 0 | 0% |  |
| searchUserByInterest | 2 | 100% | 0 | 0% |  |
| searchUser | 4 | 100% | 0 | 0% |  |
| popularInterests | 1 | 100% | 0 | 0% |  |
| getPopular | 1 | 100% | 0 | 0% |  |
| flush | 1 | 100% | 0 | 0% |  |
|  | **总计** | 108 | 99% | 0 | 0% |  |

* 1. 缺陷严重程度

目前测试未发现缺陷.

# 建议措施

查阅Junit生成的测试结果报告知, 在两种测试方法中代码覆盖率均仅于两行尚未覆盖, 如下图所示.



图中可以看出, 在两种测试方法中，对于抛出NoSuchAlgorithmException类型异常的处理未被覆盖，但以目前的能力无法在测试代码中触发这一异常，应在今后出现相关手段后进行改进，以达到100%的测试覆盖率.除此之外，两种测试方法均覆盖了剩余的所有代码.

# 图表