

# 性能测试设计与执行

软件质量保障与测试课程 Lab7 课程作业（第 9 组）

Tian, Jiahe<sup>\*</sup>    Hu, Xiaoxiao<sup>†</sup>    Huang, Jiani<sup>‡</sup>    Liu, Jiaxing<sup>§</sup>  
Shi, Ruixin<sup>¶</sup>    Wu, Chenning<sup>||</sup>    Zhang, Cenyuan<sup>\*\*</sup>  
Zhang, Yihan<sup>††</sup>    Wang, Chen<sup>‡‡</sup>

2020 年 5 月 14 日

---

<sup>\*</sup>Equal Contribution, Fudan University, 17307130313 ([tianjh17@fudan.edu.cn](mailto:tianjh17@fudan.edu.cn))

<sup>†</sup>Equal Contribution, Fudan University, 17302010077 ([xxhu17@fudan.edu.cn](mailto:xxhu17@fudan.edu.cn))

<sup>‡</sup>Equal Contribution, Fudan University, 17302010063 ([huangjn17@fudan.edu.cn](mailto:huangjn17@fudan.edu.cn))

<sup>§</sup>Equal Contribution, Fudan University, 17302010049 ([jiaxingliu17@fudan.edu.cn](mailto:jiaxingliu17@fudan.edu.cn))

<sup>¶</sup>Equal Contribution, Fudan University, 17302010065 ([rxshi17@fudan.edu.cn](mailto:rxshi17@fudan.edu.cn))

<sup>||</sup>Equal Contribution, Fudan University, 17302010066 ([cnuwu17@fudan.edu.cn](mailto:cnuwu17@fudan.edu.cn))

<sup>\*\*</sup>Equal Contribution, Fudan University, 17302010068 ([cenyuanzhang17@fudan.edu.cn](mailto:cenyuanzhang17@fudan.edu.cn))

<sup>††</sup>Equal Contribution, Fudan University, 17302010076 ([zhangyihan17@fudan.edu.cn](mailto:zhangyihan17@fudan.edu.cn))

<sup>‡‡</sup>Equal Contribution, Fudan University, 16307110064 ([wangc16@fudan.edu.cn](mailto:wangc16@fudan.edu.cn))

# 性能测试设计与执行

软件质量保障与测试课程 *Lab7* 课程作业

## 摘要

本次作业为软件质量保障与测试课程的 Lab6 课程作业，需要我们以小组为单位完成对出题系统的性能测试。本文档分为两小节。第一小节介绍了本小组进行性能测试采用的策略；第二小节介绍了性能测试的结果及系统性能分析。

## 关键词

系统与软件工程; 系统与软件质量要求和评价; 测试文档

## 目录

<b>摘要</b>	<b>2</b>
<b>关键词</b>	<b>2</b>
<b>1 Sonar 工具静态测试</b>	<b>4</b>
1.1 测试结果完整报告 . . . . .	4
1.2 测试结果核心内容截图 . . . . .	4
1.2.1 前端检测报告 . . . . .	5
1.2.2 后端检测报告 . . . . .	7
1.3 测试结果分析 . . . . .	9
1.3.1 sonar 测试报告特点 . . . . .	9
1.3.2 扫描效果 . . . . .	9
<b>2 p3c 工具静态测试</b>	<b>10</b>
2.1 测试结果完整报告 . . . . .	10
2.2 测试结果核心内容截图 . . . . .	10
2.3 测试结果分析 . . . . .	11
2.3.1 阿里巴巴 JAVA 开发手册 . . . . .	11
2.3.2 扫描效果 . . . . .	11
2.3.3 插件使用 . . . . .	11
<b>3 jshint 工具静态测试</b>	<b>12</b>
3.1 测试结果完整报告 . . . . .	12
3.2 测试结果核心内容截图 . . . . .	12
3.3 测试结果分析 . . . . .	12
3.3.1 配置项 . . . . .	13
3.3.2 扫描效果 . . . . .	14
3.3.3 工具使用 . . . . .	14
<b>4 不同工具之间的对比分析</b>	<b>14</b>
<b>5 测试总结</b>	<b>15</b>
<b>参考文献</b>	<b>16</b>

## 1 Sonar 工具静态测试

### 1.1 测试结果完整报告

“出题系统”前端（Javascript）检测报告见附件 2020-05-24-test-maker-fore-report “出题系统”后端（Java）检测报告见附件 2020-05-24-my\_project-report

### 1.2 测试结果核心内容截图

sonar 检测报告网页版视图中，分别以总览、问题、安全热点、指标、代码来记录对代码的分析结果，下面将分别截图这些部分。sonar 检测报告的文档版的内容与网页版一致，具体报告在附件中可以查看。

### 1.2.1 前端检测报告

The screenshot shows the SonarQube dashboard for the project 'test-maker-for' on the 'master' branch. The dashboard includes the following sections:

- 质量门状态:** 正常 (All conditions passed)
- 指标:**
  - 新代码: 自从 2020年5月24日 起始于 3小时前
  - 全部代码: 769 Bugs (可靠性 D)
  - 漏洞: 26 (安全性 E)
  - 安全热点: 7 (0.0% 复审, 安全复审 E)
  - 持续时间: 3小时 28分钟 (100 异味, 可维护性 A)
  - 覆盖率: 0.0% (覆盖率为 161 行, 单元测试数 280, 重复率为 89K 行, 重复块数 3.0%)
- 我的问题:** 显示了 895 个问题的列表，过滤器显示为 '所有'。列表中包含以下几条示例：
  - WEB-INF/pages/admin/syllabus.jsp: Replace this <div> tag by <sem>. 为何是问题? (Bug, 次要, 打开, 未分配, 2min 工作, 评论) - 2个月前, L6 %, 无标签
  - WEB-INF/pages/admin/users.jsp: Replace this <div> tag by <sem>. 为何是问题? (Bug, 次要, 打开, 未分配, 2min 工作, 评论) - 2个月前, L6 %, 无标签
  - Replace this <div> tag by <sem>. 为何是问题? (Bug, 次要, 打开, 未分配, 2min 工作, 评论) - 2个月前, L18 %, 无标签
  - Replace this <div> tag by <sem>. 为何是问题? (Bug, 次要, 打开, 未分配, 2min 工作, 评论) - 2个月前, L22 %, 无标签
  - Add a description to this table. 为何是问题? (Bug, 次要, 打开, 未分配, 5min 工作, 评论) - 2个月前, L43 %, 无标签
  - Add either an 'id' or a 'scope' attribute to this <th> tag. 为何是问题? (Bug, 主要, 打开, 未分配, 评论) - 2个月前, L46 %, 无标签
  - Add either an 'id' or a 'scope' attribute to this <th> tag. 为何是问题? (Bug, 主要, 打开, 未分配, 评论) - 2个月前, L47 %, 无标签
  - Add either an 'id' or a 'scope' attribute to this <th> tag. 为何是问题? (Bug, 主要, 打开, 未分配, 评论) - 2个月前, L48 %, 无标签
  - Add either an 'id' or a 'scope' attribute to this <th> tag. 为何是问题? (Bug, 主要, 打开, 未分配, 评论) - 2个月前, L49 %, 无标签

# 1 SONAR 工具静态测试

6

The screenshots illustrate the SonarQube interface for a Java project named 'test-maker-for'. The first screenshot shows a detailed view of a security hotfix, specifically regarding weak cryptography. It displays a code snippet from 'lib/bootstrap/test-infra/s3\_cache.py' with annotations and a list of related CVEs. The second screenshot shows the 'Reliability' dashboard, which includes a tree view of the project structure and a summary of reliability metrics like bugs and coverage. The third screenshot shows the 'Metrics' dashboard, providing a detailed breakdown of various performance and quality metrics across different project components.

**Screenshot 1: Security Hotfix Detail**

显示了对弱加密的复审，具体是关于命令行参数的安全性。代码片段展示了使用 hashlib.sha256() 对文件进行哈希计算的逻辑。

```
def _sha256(file):
    hasher = sha256()
    with open(filename, 'rb') as input_file:
        hasher.update(input_file.read())
    file_hash = hasher.hexdigest()
    print('sha256({}) = {}'.format(filename, file_hash))
    return file_hash
```

**Screenshot 2: Reliability Dashboard**

展示了项目的可靠性概览，包括总bug数（769）、修复工作量（4天）以及各模块的可靠性评分（如资产、lib、web等）。

**Screenshot 3: Metrics Dashboard**

展示了项目的各种度量指标，如代码行数、缺陷数、漏洞数、异味数、安全热点数、覆盖率和重复率，按模块（assets, lib, web, WEB-INF）进行细分。

### 1.2.2 后端检测报告

The screenshot displays the SonarQube interface for the 'test-maker' project. At the top, there's a navigation bar with links for '项目', '问题', '代码规则', '质量配置', '质量圈', '配置', and '更多...'. Below the navigation is a search bar and a button labeled 'A'.

The main dashboard area shows the following metrics:

- 质量圈状态:** 正常 (Green box: 所有条件都通过了.)
- 指标:**
  - 新代码:** 自从 2020年5月24日 起始于 5小时前
  - 全部代码:**
  - Bugs:** 29 (可靠性: D)
  - 漏洞:** 11 (安全性: B)
  - 安全热点:** 11 (0.0% 复审, 安全复审: E)
  - 债:** 7天 (520 异味, 可维护性: A)
  - 覆盖率:** 21.6% (覆盖行数: 6.1K 行)
  - 重复率:** 1.7% (重复行数: 13K 行)

Below the dashboard, there's a section titled '我的问题' (My Issues) with a '所有' (All) tab selected. It includes a '过滤器' (Filter) sidebar with options like '类型' (Type), '严重程度' (Severity), '处理方式' (Handling), etc. The main list shows issues from various files:

- src/.../java/cn/cstqb/exam/testmaker/IndexHelper.java: Remove this unused "request" private field. 为问题是? (次要, 296)
- src/.../java/cn/cstqb/exam/testmaker/Release.java: Remove this unused import 'java.text.SimpleDateFormat'. 为问题是? (次要, 13)
- src/.../java/cn/cstqb/exam/testmaker/actions/AbstractDeleteFileAction.java: Remove this unused import 'cn.cstqb.exam.testmaker.configuration.ApplicationConfigContext'. 为问题是? (次要, 2min 工作)

The screenshot displays two main sections of the SonarQube web application.

**Top Section (Security Hotspots):**

- Project:** test-maker (master)
- Module:** Authentication
- Severity:** High
- Issue Type:** 'password' detected in this expression, review this potentially hard-coded credential.
- Description:** 'password' detected in this expression, review this potentially hard-coded credential.
- Code Snippet:**

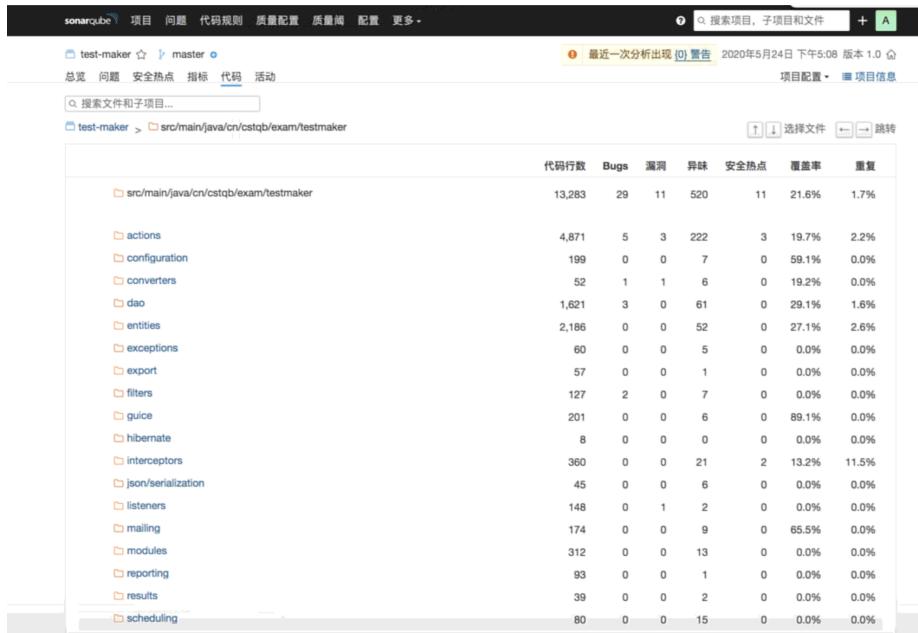
```

243     User firstUser = userDao.first();
244     if (firstUser == null) {
245       Config builtInUser=configContext.getConfig().getConfig("application.built-in");
246       logger.debug("UserServiceImpl.findFirstUser: built-in: {}", builtInUser);
247       firstUser = new User(builtInUser.getString("username"), builtInUser.getString("email"));
248       logger.info("There is no user yet. Creating first user as admin. username={}", firstUser.getEmail());
249       firstUser.setPhone(builtInUser.getString("phone"));
250       firstUser.setAdmin(true);
251       firstUser.setFullName(builtInUser.getString("fullName"));
252       userDao.create(firstUser);
253

```
- Review Status:** 需要复审 (Needs Review)
- Review Priority:** 高 (High)
- Other Issues:**
  - Denial of Service (DoS) (5)
  - Weak Cryptography (3)
  - Others (2)

**Bottom Section (Dashboard):**

- Project:** test-maker
- Module:** test-maker
- Status:** 294 / 294 文件
- Metrics:**
  - 可靠性: A
  - 安全性: A
  - 安全复审: A
  - 可维护性: A
  - 覆盖: A
  - 重复: A
  - 大小: A
  - 复杂度: A
  - 问题: A
- Risk Heatmap:** A bubble chart showing risk levels across code complexity and technical debt. A callout for the 'DaoUtil.java' file indicates:
  - Technical Debt: 2 小时 52 分钟
  - Coverage: 10.0%
  - Lines of Code: 226
  - Reliability Rating: C
  - Security Rating: A



### 1.3 测试结果分析

Sonar 是一个用于代码质量管理的开源平台，用于管理源代码的质量。

#### 1.3.1 sonar 测试报告特点

sonar 的 code viewer (代码) 部分向开发者展示代码源文件和高层次的数据，包含了行数、问题数、单元覆盖率、重复度、代码近期提交信息等。其中，coverage 用三种色彩可视化标记，红色表示没有覆盖，橙色表示部分覆盖，绿色表示完全覆盖。duplications 计算重复代码的行数并定位。

sonar 的 issues (问题) 部分分析的粒度为类型、严重程度、处理方式、状态、标准、新问题、语言、规则、标签、目录、文件、负责人、作者。严重程度划分为 blocker, critical, major, minor, info，对应致命（阻断），关键（严重），主要，微小（次要），未知（提示）。在 issues 中，sonar 还会列出 maintainability, documentation, complexity, bulk change, dispositioning 等项。

#### 1.3.2 扫描效果

选择的是 sonar 默认的扫描规则，效果非常细致，相对来说扫描过程用时也比较久。sonar 将问题分为 bug, code\_smell 和 vulnerability，严重程度划分为 minor,major,blocked,critical,info。bug 类型的问题多涉及到比如比较判断符错误使用，空指针重定向，条件分支不可达等，code\_smell 类型

涉及到方法返回值不能一直不变，方法体不能为空等，vulnerability 类型涉及引用了已知有漏洞的函数的代码。sonar 扫描的问题比较全面，也可以做到持续的代码检查跟进，具有高可用性和较短的反馈循环。提供了多种语言检测支持。

## 2 p3c 工具静态测试

### 2.1 测试结果完整报告

请通过网址[第 9 组 P3C 报告网站](#)来查看本小组的 P3C 完整报告。

### 2.2 测试结果核心内容截图

IntelliJ IDEA inspection report:

Inspection tree:	Problem description:
<input checked="" type="checkbox"/> 'InspectionViewTree' project 172 blockers 99 criticals 628 majors	Location public method <code>QuestionReportingDaoImpl</code> in class <code>QuestionReportingDaoImpl</code>
<input checked="" type="checkbox"/> Blocker 172 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> All-Check group 172 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> long或者Long初始赋值时，必须使用大写的L，不能是小写的l，小写容易跟数字1混淆，造成误解。 inspection 10 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> class QuestionReportingDaoImpl 10 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> BLOCKER 'long' 型常量 'or' 应该以大写L结尾	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> 在if#elif for while do语句中必须使用大括号，即使只有一行代码，避免使用下面的形式： if (condition) statements; inspection 107 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> 在使用正则表达式时，利用好其预编译功能，可以有效加快正则匹配速度。 inspection 4 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> 所有的包装类对象之间值的比较，全部使用equals方法比较。 inspection 2 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> 所有的覆盖方法，必须加@Override注解。 inspection 48 blockers	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> 线程池不允许使用Executors去创建，而是通过ThreadPoolExecutor的方式，这样的处理方式让写的同学更加明确线程池的运行规则，规避资源耗尽的风险。 inspection 1 blocker	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> Critical 99 criticals	Problem synopsis <code>'long' 型常量 'or'</code>
<input checked="" type="checkbox"/> Major 628 majors	Problem synopsis <code>'long' 型常量 'or'</code>

如图所示，可分为 Blocker、Critical、Major 三部分。

IntelliJ IDEA inspection report:

Inspection tree:	Problem description:
<input checked="" type="checkbox"/> 'InspectionViewTree' project 172 blockers 99 criticals 628 majors	Select a problem element in tree
<input checked="" type="checkbox"/> Blocker 172 blockers	
<input checked="" type="checkbox"/> Critical 99 criticals	
<input checked="" type="checkbox"/> Major 628 majors	

### 2.3 测试结果分析

P3C 是阿里巴巴推出的《阿里巴巴 Java 开发规约》扫描插件，目前在 IDEA 和 Eclipse 都有较好的支持。P3C 扫描结果文档给出了项目的 bug，并根据 bug 的严重程度分为三个级别展示。三个级别分别是：Blocker, Critical, Major，严重程度由高到低。

#### 2.3.1 阿里巴巴 JAVA 开发手册

根据《阿里巴巴 JAVA 开发手册》版本 1.3.0，我们可以对扫描结果进行分析。该版本对 JAVA 开发的规约含有编程规约、异常日志、单元测试、安全规约等六大类，而本项目的扫描结果重点体现的是编程规约。编程规约在开发手册共分为 9 类：

- 命名规范
- 常量定义
- 代码格式
- OOP 规约
- 集和处理
- 并发处理
- 控制语句
- 注释规约
- 其他

#### 2.3.2 扫描效果

根据扫描结果可以发现：同一类型的编程规约由于其严重性不同可以划分为不同级别的 bug。比如注释规约中，抽象方法的 javadoc 注释属于 Major，而枚举字段的注释属于 Critical；命名规范中也有类似的例子。另外，bug 的严重程度分类比较合理。注释规约、命名规范的约定内容严重程度较低，并发处理、控制语句的约定内容严重程度较高。整体上，P3C 扫描插件发现的问题比较基础，它侧重 JAVA 编程细节可能导致的系统失效。

#### 2.3.3 插件使用

P3C 插件有以下的优点：

- 基本满足代码规范检测的需求。
- 能够检测出细致和易忽略的问题，可以提高开发过程中对代码细枝末节的注意。

- Quick fix, 检测出问题后可以快速查看 bug 位置、解决方案并一键替换。
- 中文提示, 解释内容与开发手册一致。

此外 P3C 插件也存在平台限制、功能不成熟、扫描能力有限等问题。

### 3 jshint 工具静态测试

#### 3.1 测试结果完整报告

请通过网址[第 9 组 JSHint 报告网站](#)来查看本小组的 JSHint 完整报告。

#### 3.2 测试结果核心内容截图

如图所示, 共发现 236 个 failures, 1 个 error 和 235 个 warnings。其中每

Code	Line	Column	Evidence	Reason
<code>./webapp/assets/js/ajax-utils.js</code>				
W097	1	1	'use strict';	Use the function form of "use strict".
W033	60	11	)	Missing semicolon.
W033	61	2	}	Missing semicolon.
W117	12	16	<code>var data = _._isNull(param)    _._isUndefined(param) ? null : param;</code>	'._' is not defined.
W117	12	35	<code>var data = _._isNull(param)    _._isUndefined(param) ? null : param;</code>	'._' is not defined.

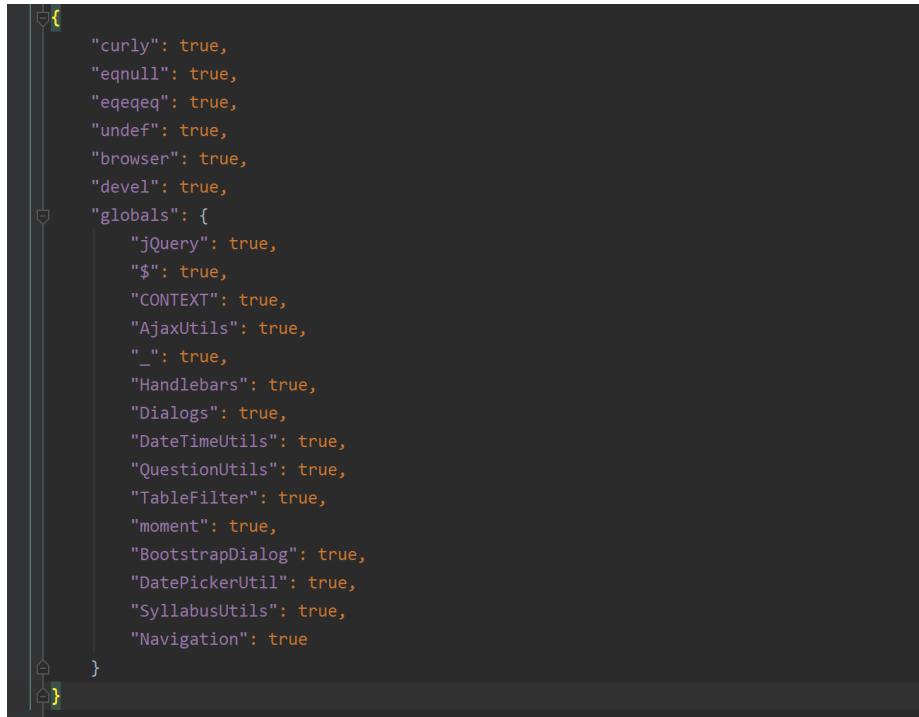
一项都标明了具体的行数、问题代码和问题原因。

Code	Line	Column	Evidence	Reason
<code>./webapp/assets/js/ajax-utils.js</code>				
W097	1	1	'use strict';	Use the function form of "use strict".
W033	60	11	)	Missing semicolon.
W033	61	2	}	Missing semicolon.
W117	12	16	<code>var data = _._isNull(param)    _._isUndefined(param) ? null : param;</code>	'._' is not defined.
W117	12	35	<code>var data = _._isNull(param)    _._isUndefined(param) ? null : param;</code>	'._' is not defined.

#### 3.3 测试结果分析

JSHint 是由 Anton Kovalyov 基于 JSLint 的代码实现的开源项目, JSHint 与 JSLint 相比较之下, 更友好, 也更容易配置, 所以发展很快并得到了众多 IDE 和编辑器的支持。JSHint 是一个 JavaScript 语法和风格的检查工具, 但检查不出逻辑问题。它可以根据配置参数扫描 JavaScript 代码, 分析其中的语法与风格从而给出代码质量报告。

### 3.3.1 配置项

A screenshot of a code editor showing a JSHint configuration file. The file contains several configuration options, including 'curly', 'eqnull', 'eqeqeq', 'undef', 'browser', 'devel', and a 'globals' object. The 'globals' object lists various global variables and objects like jQuery, \$, CONTEXT, AjaxUtils, \_, Handlebars, Dialogs, DateTimeUtils, QuestionUtils, TableFilter, moment, BootstrapDialog, DatePickerUtil, SyllabusUtils, and Navigation, all set to true.

```
    "curly": true,
    "eqnull": true,
    "eqeqeq": true,
    "undef": true,
    "browser": true,
    "devel": true,
    "globals": {
        "jQuery": true,
        "$": true,
        "CONTEXT": true,
        "AjaxUtils": true,
        "_": true,
        "Handlebars": true,
        "Dialogs": true,
        "DateTimeUtils": true,
        "QuestionUtils": true,
        "TableFilter": true,
        "moment": true,
        "BootstrapDialog": true,
        "DatePickerUtil": true,
        "SyllabusUtils": true,
        "Navigation": true
    }
}
```

JSHint 工具使用的关键是配置项。如果不设置配置项，那么可能会有很多“假”错误或警告。比如自定义全局变量，不同脚本上下文的符号引用。这些内容既不算错误的语法，也不算差劲的风格，可是 JSHint 依旧把他们认错了。JSHint 有一些常见的配置项：

- “strict”: true 严格模式
- “asi”: true 允许省略分号
- “bitwise”: true 禁止使用位运算符，比如经常把 && 写错 & 规避此错误
- “eqeqeq”: true 禁止使用 == 和 != 强制使用 === 和 != ==
- “undef”: true 禁止使用不在全局变量列表中的未定义变量
- “curly”: true 循环或者条件语句必须使用花括号包围
- “devel”: true 定义用于调试的全局变量：console,alert
- “jquery”: true 定义全局暴露的 jQuery 库（可以去掉）
- “browser”: true 暴露浏览器属性的全局变量如 window,document
- “globals”: {"\$":true,"require":true,"\_":true}

JSHint 的配置项一般是放入项目根目录下的.jshintrc 文件中，JSHint 工具在扫描的时候就会运用这些配置项。配置项的作用一方面是明确项目的编程规范，约束开发人员的行为。另一方面是避免某些规范，减轻开发人

员的负担（比如允许省略分号等）。

### 3.3.2 扫描效果

不同的配置项扫描结果是不尽相同的。项目最初未定义配置项中的全局变量，这导致了非常多的未定义错误或警告。这显然不是我们需要的扫描结果。完成配置项后，扫描结果报告逐渐明晰，其中报告了出现了缺少分号、使用 `==` 和 `!=` 错误、缺少 `{` 或 `}` 错误和未定义变量等错误。正如上面所言，JSHint 可以分析出 JavaScript 代码的语法和风格，但是它无法识别出项目的逻辑错误，比如冗余代码、死循环、无效代码等问题。它的分析功能是非常基础和有限的。

### 3.3.3 工具使用

JSHint 工具不可否认具有一定的优点：

- 有了很多参数可以配置
- 支持配置文件，方便使用
- 支持了一些常用类库（如 jquery 等）
- 支持了基本的 ES6

但与其他功能强大的 Web 项目静态扫描工具比较而言，它具有不支持自定义规则、无法根据错误定位到对应的规则和不提供快捷的修正方式等缺点。不支持自定义规则自然让它只能检测出很基础的预定义规则，无法根据错误定位到对应的规则使得扫描结果不易阅览，不提供快捷的修正方式（不能跳转到指定代码位置和不能提供修正方案）自然无法让开发人员方便的处理扫描结果。

## 4 不同工具之间的对比分析

- sonar：定位是代码质量平台，本身不进行代码分析，但可以集成各个静态分析工具以及其他软件开发测试工具，并基于集成工具的结果数据按照一定的质量模型，对软件的质量进行评估。甚至可以选择接入 p3c 规范进行代码扫描。sonar 基于扫描规则进行扫描，因此扫描的问题可以比较全面，本身是代码质量平台，可以做到持续的代码检查跟进，具有高可用性和较短的反馈循环。提供了多种语言检测支持。生成的检测报告也比较详细，数据可视化好。是几个工具中专业性最高的。

- p3c 是一套自动化的 IDE 检测插件（主要是 IDEA、Eclipse）该插件在扫描代码后，将不符合《手册》的代码按 Blocker/Critical/Major 三个等级显示在下方，根据错误定位到对应的规则，在 IDE 中提供快捷修正方式，但 p3c 扫描检测的问题较为基础，侧重 JAVA 编程细节可能导致的系统失效，之后仍需要类似 FindBugs 的插件再次扫描检测 bug。
- JSHint 是由 Anton Kovalyov 基于 JSLint 的代码实现的开源项目，是一个 JavaScript 语法和风格检查的命令行工具，不能检查出逻辑问题。它可以根据配置参数扫描 JavaScript 代码，分析其中的语法风格从而给出代码质量报告。相比于 sonar 它不支持自定义规则、相比于 p3c 它无法根据错误定位到对应的规则，也不提供快捷的修正方式，分析功能也非常基础和有限，是并不算强大的 web 静态扫描工具。

## 5 测试总结

在我们选取的三个测试工具中，sonar 定位于代码质量平台，集成各种静态分析工具和其它测试工具，提供持续代码检测跟进，是最强大和专业的静态测试工具。而 p3c 和 JSHint 分别是针对 Java 和 JS 开发的基于编码及设计实践的静态检测工具，p3c 做成了 IDE 检测插件，JSHint 则是命令行工具，由于二者都偏重于代码编写格式，及是否符合编码规范的检验，内置编程规范比较基础，对代码 bug 发现功能较弱。但是他们都能快速定位代码隐藏错误和缺陷，显著减少在代码逐行检查上花费的时间，提高软件可靠性并节省软件开发和测试成本。

## 参考文献

International Organization for Standardization. 2014. *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*. International Organization for Standardization. Vol. 2014. <https://www.iso.org/standard/64764.html>.

中国国家标准化管理委员会. 2016. GB/T 25000.51-2016《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 51 部分：就绪可用软件产品 (RUSP) 的质量要求和测试细则》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 51. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2017a. GB/T 25000.12-2017《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 12 部分：数据质量模型》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 12. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2017b. GB/T 25000.24-2017《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 24 部分：数据质量测量》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 24. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2018. GB/T 25000.40-201《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 40 部分：评价过程》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 40. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2019. GB/T 25000.23-2019《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 23 部分：系统与软件产品质量测量》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 23. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.