

SonarQube Server

Release 1.0

Written by CI

持续集成项目组

版本号	修订内容	修订时间	修订人
1.0	SonarQube 服务搭建	2017/08/16	陈杰、许明
2.0	SonarScanner 使用	2017/09/04	陈杰、许明

目录

[1.SonarQube 服务的原理与流程](#)

[1.1 原理](#)

[1.2 工作流程](#)

[2.用户指南](#)

[2.1 项目页面](#)

[2.1.1 仪表盘](#)

[2.1.1.1 项目仪表盘展示](#)

[2.1.1.1 配置仪表盘模块空间](#)

[2.1.2 质量阈](#)

[2.1.3 问题](#)

[2.1.3.1 项目问题页面展示](#)

[2.1.3.2 问题修复](#)

[2.1.4 代码](#)

[2.1.5 配置](#)

[2.1.5.1 项目权限](#)

[2.1.5.2 设置](#)

[2.1.5.3 后台任务](#)

[2.2 全局页面](#)

[2.2.1 代码规则](#)

[2.2.2 质量配置](#)

[2.2.3 质量阈](#)

[2.2.4 配置](#)

[2.2.5 更多](#)

[3.SonarLint 插件](#)

[3.1 环境准备](#)

[3.1.1 环境依赖](#)

[3.1.2 当前服务环境](#)

[3.2 Eclipse 集成 SonarLint 插件](#)

[3.3 SonarLint 使用](#)

[3.3.1 SonarLint 绑定项目](#)

[3.3.2 SonarLint 分析项目](#)

[4.SonarQube Scanner](#)

[5.附录](#)

[5.1 参数](#)

[5.1.1 强制参数](#)

[5.1.1.1 服务](#)

[5.1.1.2 项目配置](#)

[5.1.2 可选参数](#)

[5.1.2.1 项目标识](#)

[5.1.2.2 身份验证](#)

[5.1.2.3Web 服务](#)

[5.1.2.4 项目配置](#)

[5.1.2.5 排除/新增](#)

[5.1.2.6 重复](#)

[5.2 指标](#)

[5.2.1 复杂度](#)

[5.2.2 软件文档](#)

[5.2.3 重复](#)

[5.2.4 问题](#)

[5.2.5 严重性](#)

[5.2.6 可维护性](#)

[5.2.7 质量阈](#)

[5.2.8 安全性](#)

[5.2.9 指标](#)

[5.2.10 测试](#)

[6.参考资料](#)

特设集成公司

本篇用户指南是一篇关于 SonarQube 服务的用户指南文章，详细阐述了 SonarQube 服务的使用方法，注意事项和工作原理。

提示： 本篇用户指南仍然处于不断完善的过程中，如果有相关建议和问题请联系

谢谢！

1.SonarQube 服务的原理与流程

1.1 原理

– 如图 1.1 SonarQube 平台是由 4 部分组成:

1. **SonarQube 服务器** (3 大主要内容):

- a. **Web 服务器**
- b. 基于 Elasticsearch 的**搜索服务器**
- c. **计算引擎服务器**负责处理代码分析报告并保存在 SonarQube 数据库

2. **SonarQube 数据库**用于存储数据:

- SonarQube 实例的配置(安全、插件设置等)
- 项目的质量快照、视图等。

3. 多个 **SonarQube 插件**安装在服务器上,可能包括语言、单片机、集成、验证、和治理插件

4. 一个或多个 **SonarQube 扫描仪**用于分析在服务器上的项目

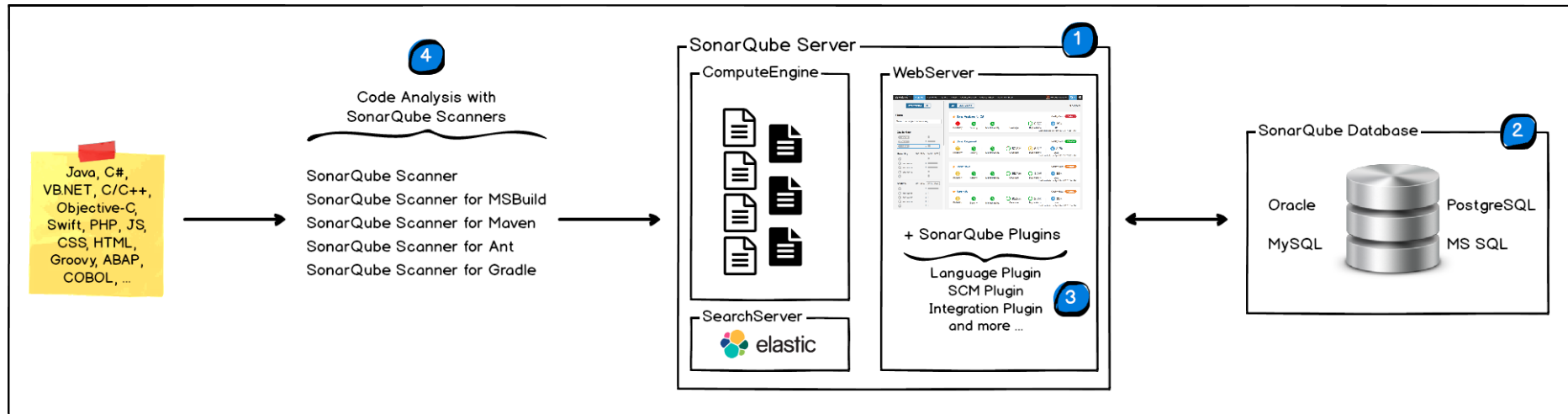


图 1.1

1.2 工作流程

如图 1.2

1. 开发人员在本地使用 SonarLint 运行分析 IDEs 上的代码
 2. 开发者将他们的代码上传到他们最喜爱的持续集成服务器上，例如:git,SVN,TFVC,.....
 3. 持续集成服务器触发自动构建，SonarQube 扫描仪执行代码分析
 4. 分析报告发送到 SonarQube 服务器进行处理
 5. SonarQube 服务器分析和存储分析报告结果到 SonarQube 数据库中,并显示结果在 UI 界面中
 6. 开发人员通过修改、评审代码来管理和减少他们 SonarQube UI 页面上的技术债务
 7. 项目经理收到分析报告
- 运维使用 api 来从 SonarQube 自动化配置和提取数据
- 运维使用 JMX 监控 SonarQube 服务器

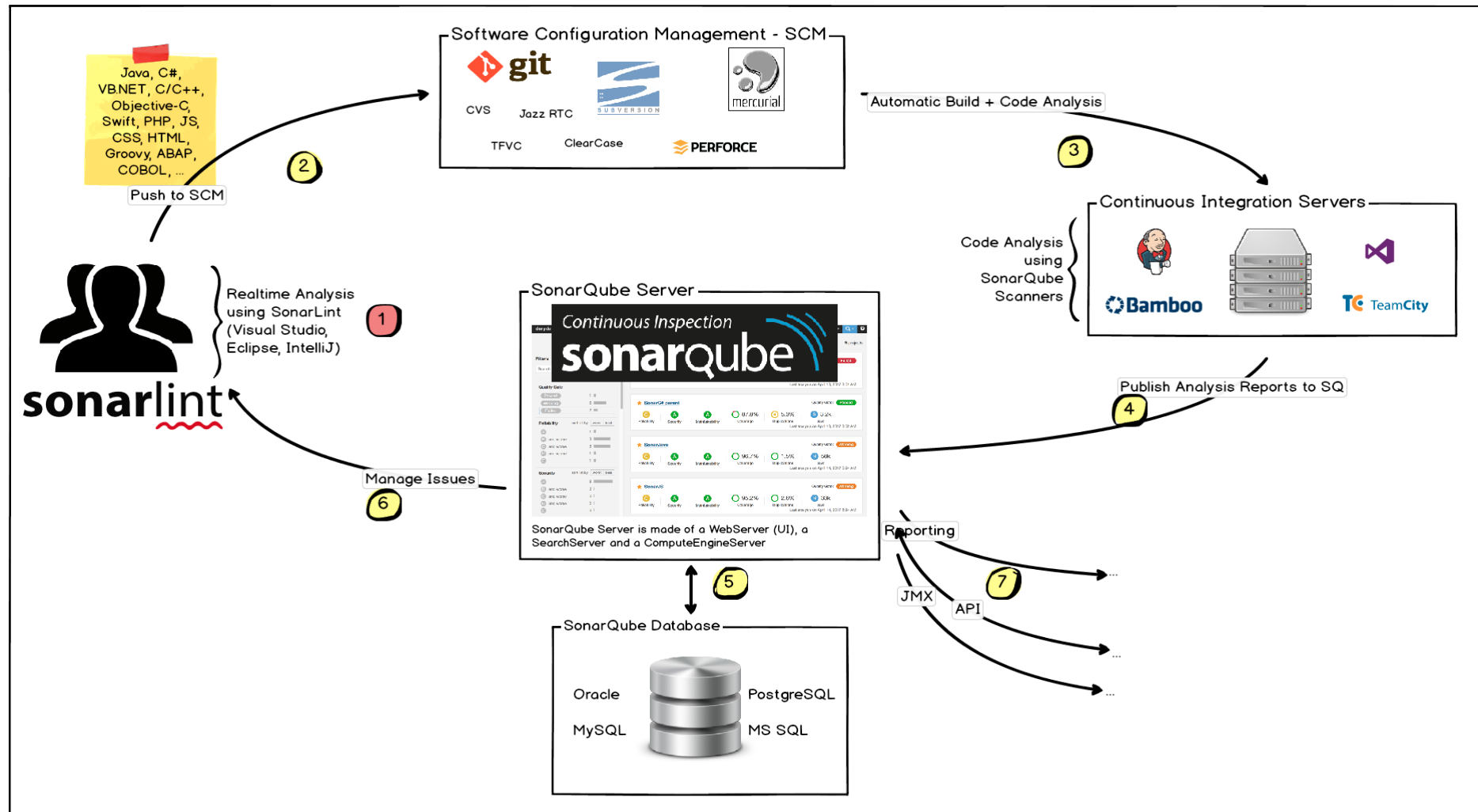


图 1.2

2.用户指南

2.1 项目页面

2.1.1 仪表盘

- 页面右上角，用户可以登录自己的账号，登录成功后可以看到自己在 SonarQube 上进行过代码检测的项目，也可以点击登录按钮旁的搜索按钮查询特定项目 [图 2.1](#)

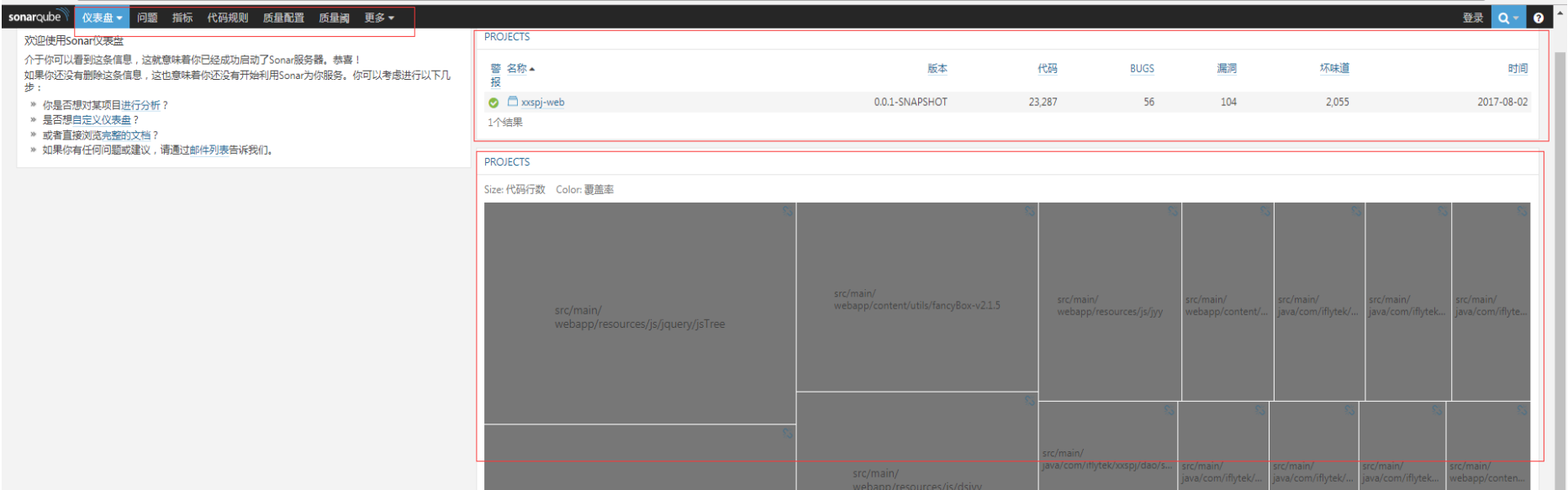


图 2.1

2.1.1.1 项目仪表盘展示

- 单击项目进入**项目主页**，单击**仪表盘**
- 页面左上角的**五角星标记**，可以收藏项目，方便查询
- **事件模块**：可以筛选项目的代码分析报告；可以显示项目的代码行数，文件数，方法等信息；显示代码的重复信息
- **问题模块**：显示了 SonarQube 扫描项目后发现的问题，问题由严重程度划分为五类（参考[附录-指标-严重性](#)）：
 - ◆ **阻断**：影响应用程序正常运行的、错误概率高的问题：例如内存泄漏、未关闭的 JDBC 连接。代码必须立即修复
 - ◆ **严重**：影响应用程序正常生产行为的、错误出现概率低的问题或者代表一个安全漏洞：例如空 catch 块，SQL 注入等。代码必须立即修复
 - ◆ **主要**：很大程度影响开发人员开发效率的问题：例如重复的代码块，未使用的参数等
 - ◆ **次要**：略微影响开发人员开发效率的质量缺陷：例如代码行不行太长，开关语句至少有 3 例等
 - ◆ **信息**：仅展示发现代码错误和质量缺陷，不做处理
- **历史模块**：可以查看项目什么时间，代码扫描的行数

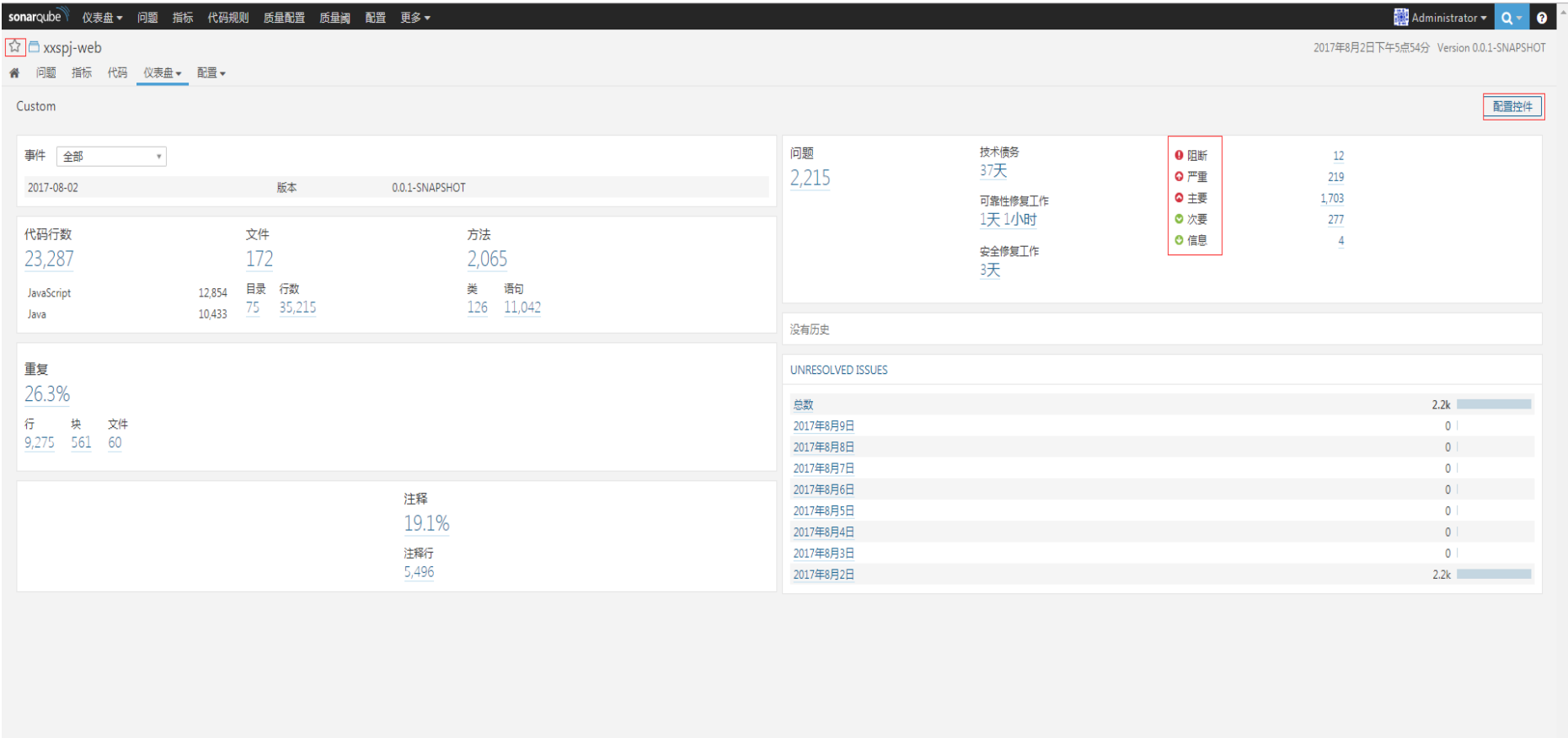


图 2.2

2.1.1.2 配置仪表盘模块空间

- 如图2.3 单击项目仪表盘页面右上角**配置控件**，自由设置搭配仪表盘的模块
- 用户可以自由添加编辑删除**控件**，并调整仪表盘的布局；添加/删除/编辑的组件都会实时在页面响应
- 如图2.3 用户也可以单击**仪表盘**，选择**管理仪表盘**，创建多个仪表盘，并且编辑仪表盘或者配置控件

The screenshot displays the SonarQube dashboard configuration interface for the 'xxspj-web' project. The interface is divided into several sections:

- Dashboard Configuration:** A grid of widgets that can be added, edited, or deleted. The widgets include:
 - 欢迎 (Welcome): 欢迎信息可以提供最有价值的资源，比如文档和支持。
 - 气泡图 (Bubble Chart): 使用气泡图展示组件的源代码。坐标轴和气泡大小都可以配置。
 - 组件树形 (Component Tree): 显示选择资源的所有直接组件的树形。
 - 自定义描述 (Custom Description): 显示选择的描述列表。
 - 警报 (Alerts): 显示项目当前警报。
 - 过滤器列表 (Filter List): 显示已配置好的过滤器列表。
 - 过滤器树图 (Filter Tree): 显示已配置好的过滤器树图。
 - 遵守规则 (Rules): 对编码标准的违规和遵守的报告。
 - 重复 (Duplication): 复制粘贴和代码重复。
 - 问题过滤器 (Issue Filters): 显示配置的问题过滤器搜索结果。
 - 集成测试覆盖率 (Integration Test Coverage): 集成测试的代码覆盖率报告。
 - 项目文件标签云 (Project File Tags): 把组件的源文件显示为标签云。两个坐标轴都可以配置。
 - 项目问题标签云 (Project Issue Tags): 显示未解决问题对应的标签云。
 - 项目问题过滤器 (Project Issue Filters): 为项目显示预先配置好的问题搜索结果。
- 事件 (Events):** A section showing events for the project. It includes a table with columns for event name, version, and details.
- 大小指标 (Size Metrics):** A section showing size metrics for the project. It includes a table with columns for metric name, value, and details.
- 代码覆盖率 (Code Coverage):** A section showing code coverage for the project. It includes a table with columns for metric name, value, and details.
- 重复 (Duplication):** A section showing duplication for the project. It includes a table with columns for metric name, value, and details.
- 文档或注释 (Documentation or Comments):** A section showing documentation or comments for the project. It includes a table with columns for metric name, value, and details.

图 2.3

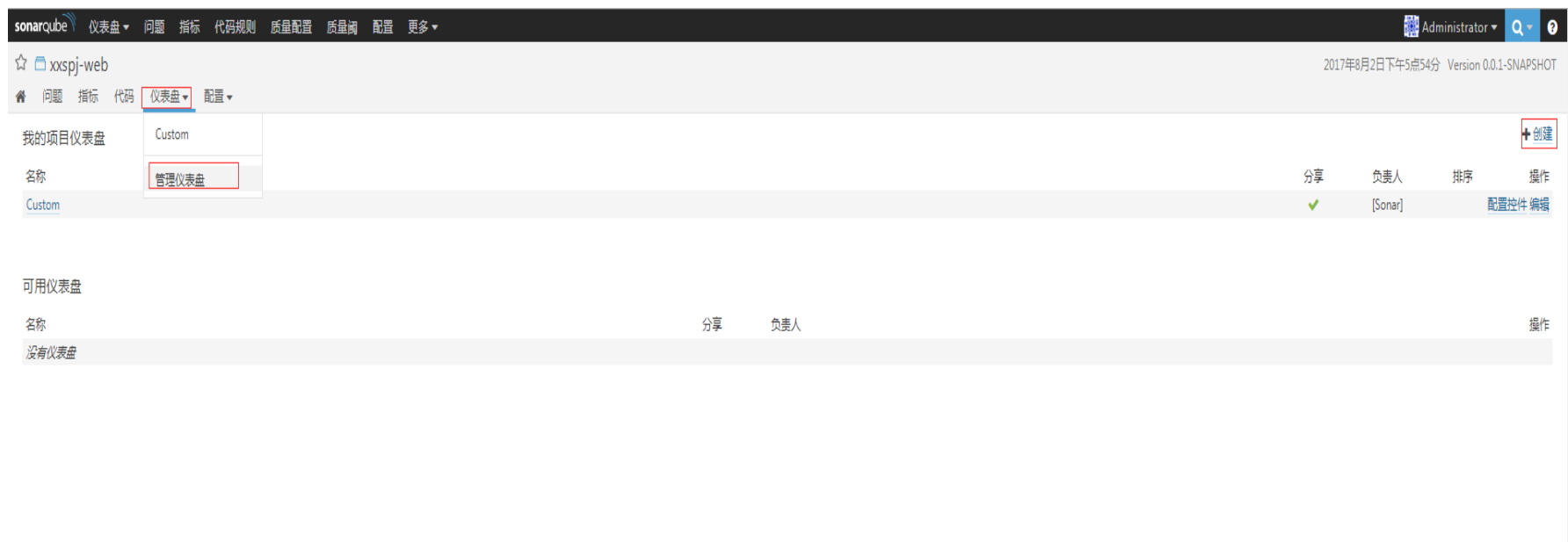


图 2.4

2.1.2 质量阈

- 质量阈是为了确保项目质量不断提升，SonarQube 有一套默认的质量阈，但是用户可以创建符合自己项目的质量阈；用户可以创建多个质量阈，并对它们进行规则的编辑（图 2.5）



图 2.5

2.1.3 问题

2.1.3.1 项目问题页面展示

- 质量阈未通过，变红并标识错误原因；修复质量阈的问题，质量阈会变绿色，用户就可以发布自己项目
- Bugs&漏洞（[参考附录-参数-可维护性/安全性](#)）：用户每次提交代码并使用 SonarQube 进行代码检测的时候，至少要保证 Bugs 和漏洞不会比上一次代码扫描的问题更多，这样才能保证代码的质量；[如图 2.6](#) 代码检测完毕时，应该第一时间修复最新的 Bugs 和漏洞

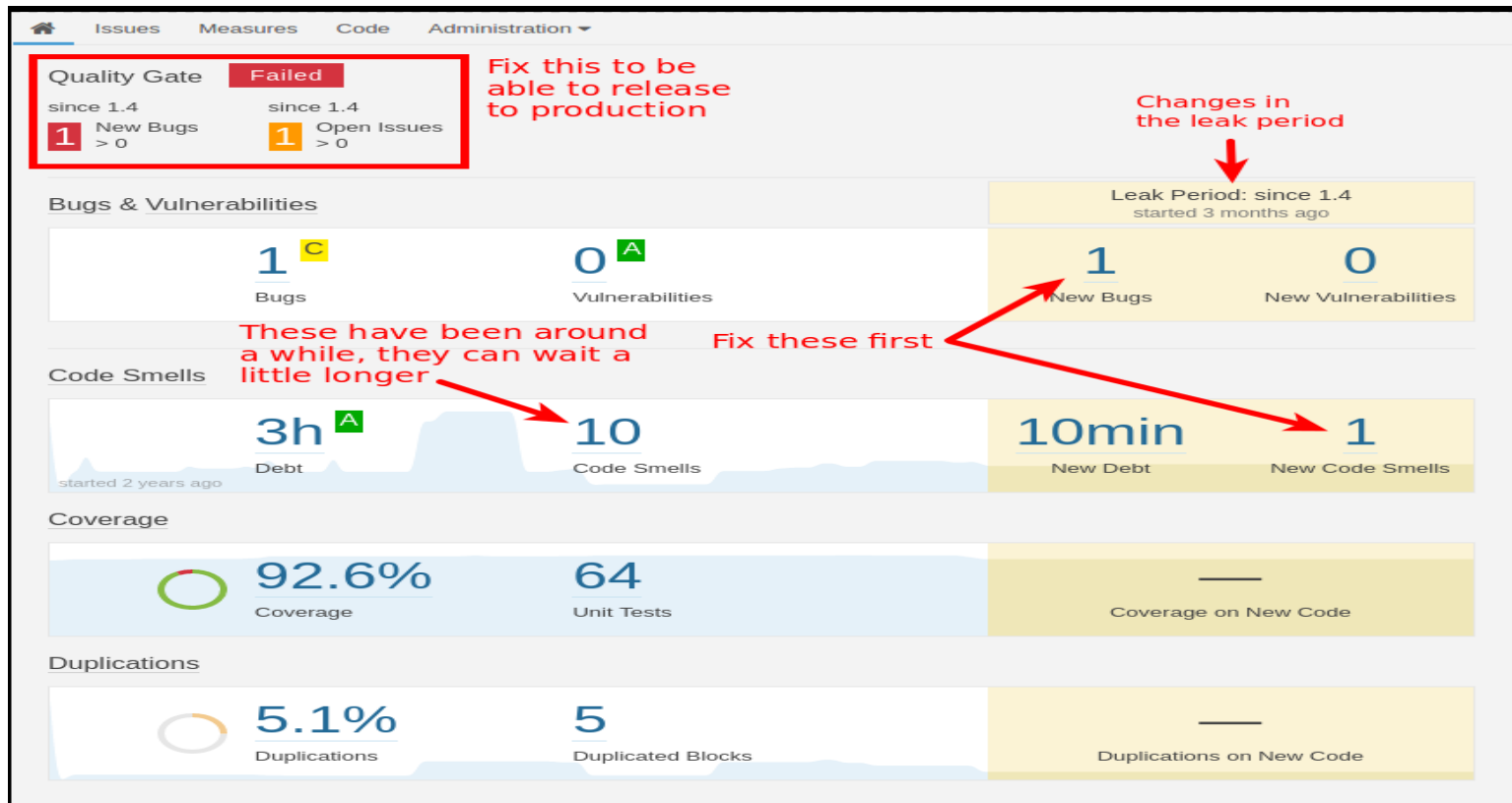


图 2.6

2.1.3.2 问题修复

- 按照安全性程度（参考附录-指标-），从 Bugs 开始修复，单击 **Bugs 数值** 进入问题详情页面（图 2.7）

- 页面左边栏可以通过**类型**、**处理方式**筛选展示问题信息，应优先处理新问题
- 每个问题条代表一个问题，都可以通过超链接到代码进行修复；用户可以设置问题的严重程度、修复进度、分配修复人员、评论问题

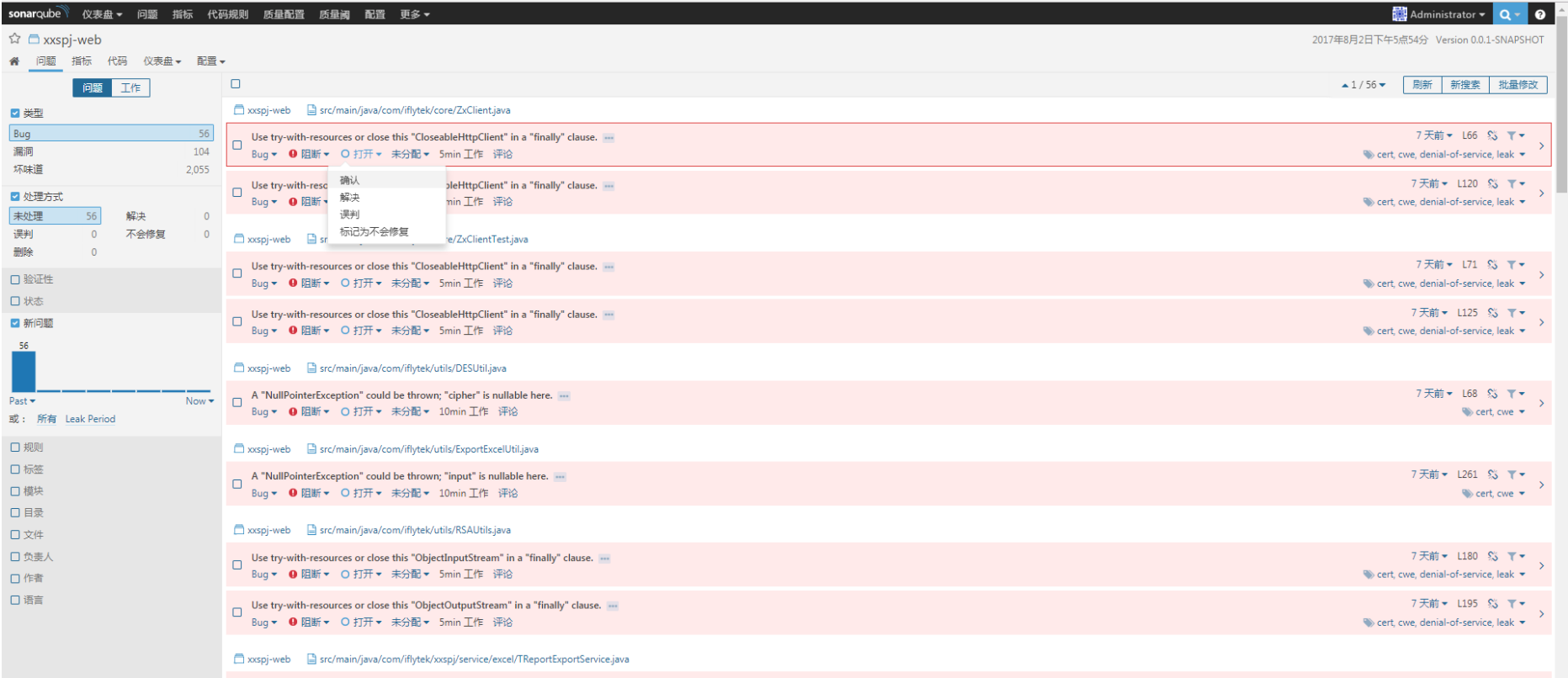
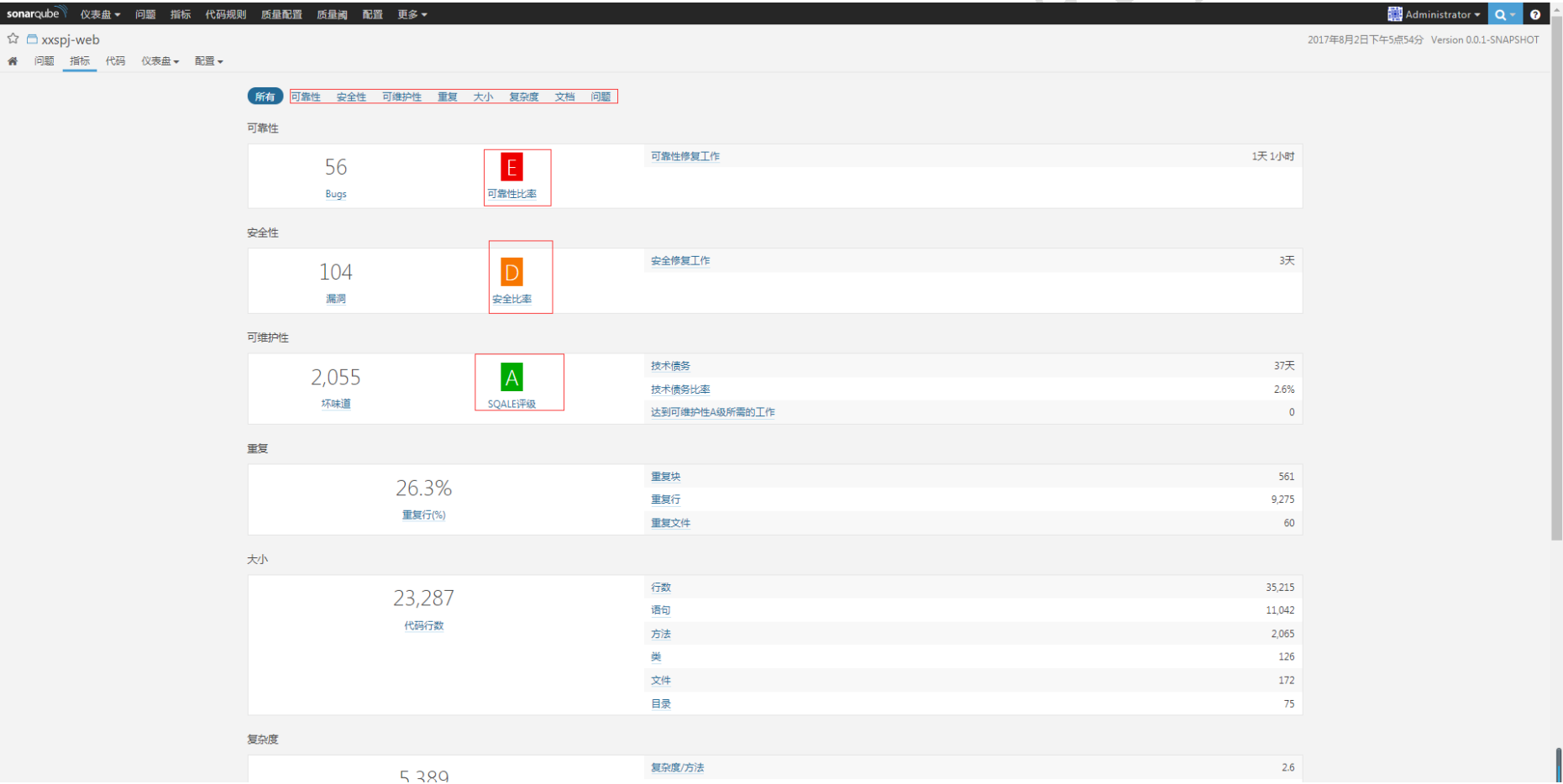


图 2.7

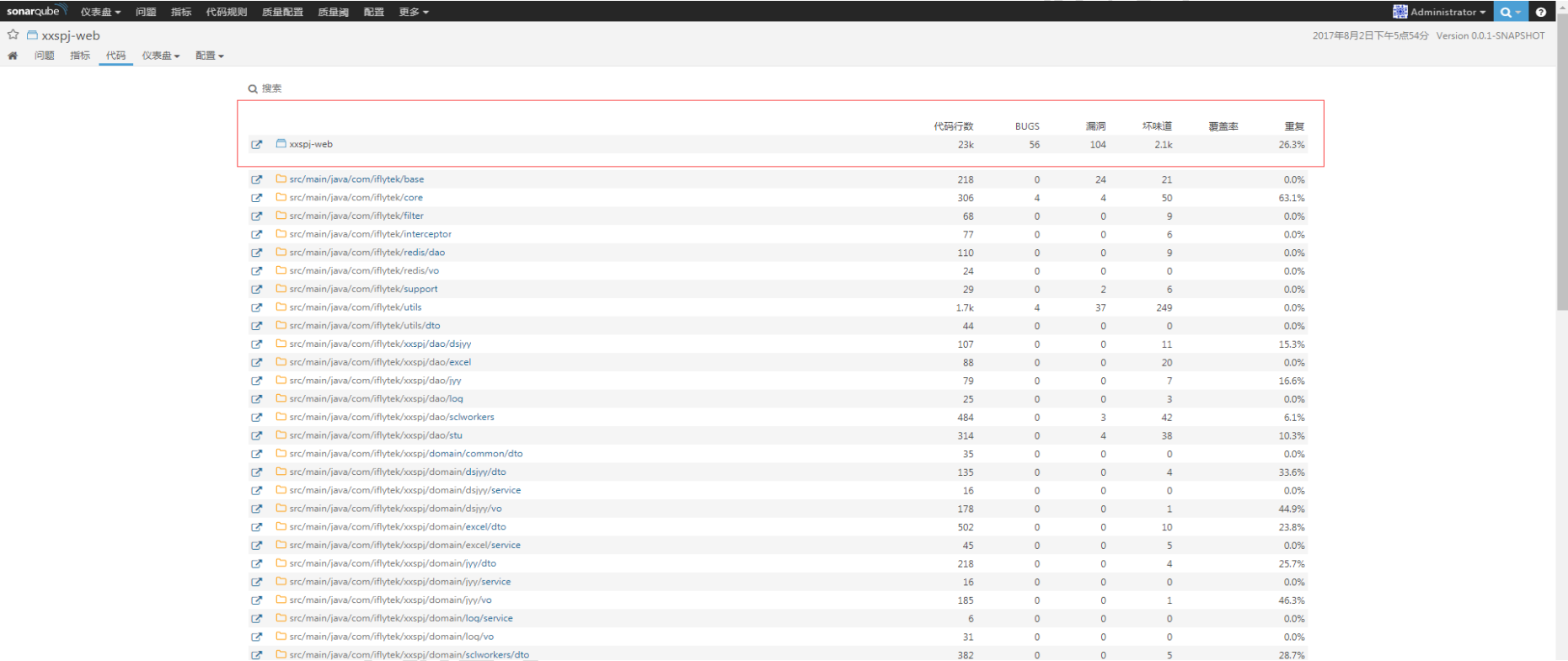
2.1.4 指标

- SonarQube 通过 8 种指标衡量项目质量的优劣（参考[附录-指标](#)），如图，用户应由上而下的指标优先级修复代码。



2.1.4 代码

- 用户可以在代码中，搜索项目的代码检测情况；从项目包路径的维度，展示代码行数、Bugs、漏洞...等检测信息；用户可直接点击进入进行修复



	代码行数	BUGS	漏洞	坏味道	覆盖率	重复
xxspj-web	23k	56	104	2.1k		26.3%
src/main/java/com/flytek/base	218	0	24	21		0.0%
src/main/java/com/flytek/core	306	4	4	50		63.1%
src/main/java/com/flytek/filter	68	0	0	9		0.0%
src/main/java/com/flytek/interceptor	77	0	0	6		0.0%
src/main/java/com/flytek/redis/dao	110	0	0	9		0.0%
src/main/java/com/flytek/redis/vo	24	0	0	0		0.0%
src/main/java/com/flytek/support	29	0	2	6		0.0%
src/main/java/com/flytek/utlis	1.7k	4	37	249		0.0%
src/main/java/com/flytek/utlis/dto	44	0	0	0		0.0%
src/main/java/com/flytek/xxspj/dao/dsivy	107	0	0	11		15.3%
src/main/java/com/flytek/xxspj/dao/excel	88	0	0	20		0.0%
src/main/java/com/flytek/xxspj/dao/fyy	79	0	0	7		16.6%
src/main/java/com/flytek/xxspj/dao/loq	25	0	0	3		0.0%
src/main/java/com/flytek/xxspj/dao/sclworkers	484	0	3	42		6.1%
src/main/java/com/flytek/xxspj/dao/stu	314	0	4	38		10.3%
src/main/java/com/flytek/xxspj/domain/common/dto	35	0	0	0		0.0%
src/main/java/com/flytek/xxspj/domain/dsivy/dto	135	0	0	4		33.6%
src/main/java/com/flytek/xxspj/domain/dsivy/service	16	0	0	0		0.0%
src/main/java/com/flytek/xxspj/domain/dsivy/vo	178	0	0	1		44.9%
src/main/java/com/flytek/xxspj/domain/excel/dto	502	0	0	10		23.8%
src/main/java/com/flytek/xxspj/domain/excel/service	45	0	0	5		0.0%
src/main/java/com/flytek/xxspj/domain/fyy/dto	218	0	0	4		25.7%
src/main/java/com/flytek/xxspj/domain/fyy/service	16	0	0	0		0.0%
src/main/java/com/flytek/xxspj/domain/fyy/vo	185	0	0	1		46.3%
src/main/java/com/flytek/xxspj/domain/loq/service	6	0	0	0		0.0%
src/main/java/com/flytek/xxspj/domain/loq/vo	31	0	0	0		0.0%
src/main/java/com/flytek/xxspj/domain/sclworkers/dto	382	0	0	5		28.7%

图 2.7

2.1.5 配置

2.1.5.1 项目权限

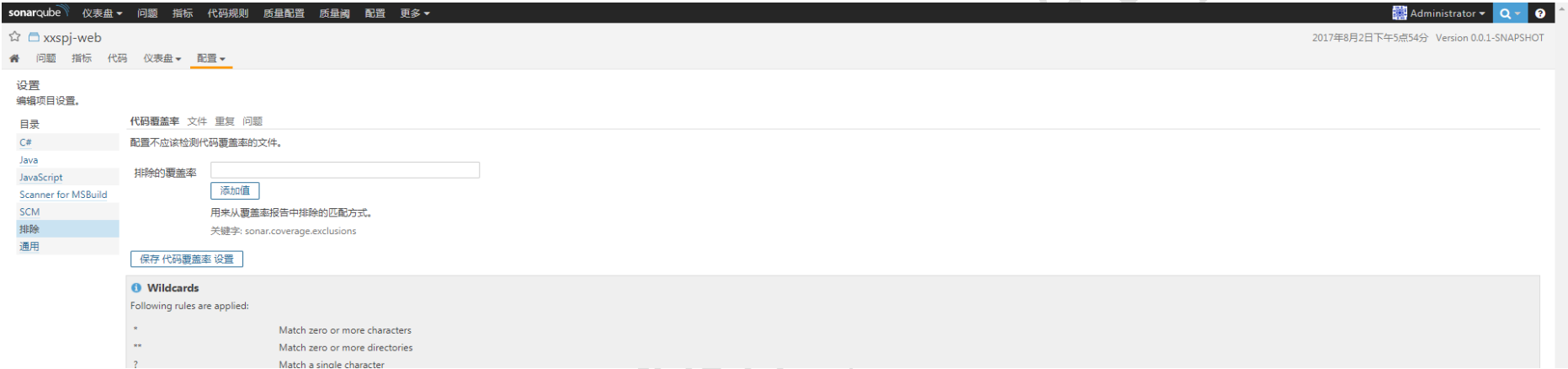
- [图 2.8](#) 是 SonarQube 默认提供的角色和权限，用户可以根据自己的项目设定不同的**角色/角色组**和**权限**
 - 用户：能浏览项目的数据并创建和修改事务。
 - 代码复审：能浏览项目源码。
 - 问题管理员：能编辑问题的权限。
 - 管理员：能通过访问它的配置对项目执行管理任务。
 - 执行分析：允许执行分析，可以获取执行分析的所有配置，甚至包含 SCM 账号密码，JIRA 密码等安全配置。



图 2.8

2.1.5.2 设置

- [图 2.9](#) 可以设置项目的参数（参考[附录-参数](#)）



2.1.5.3 后台任务

- [图 2.10](#) 监控等待执行的报告队列，同时可以访问历史分析信息，用户可以打开项目的日志，查看代码检测失败的原因



图 2.10

2.2 全局页面

2.2.1 代码规则

- SonarQube 有一套默认的全语言的代码规则；用户可以根据自身情况设置自定义的代码质量配置文件
 - 页面左边栏可以通过语言、类型筛选代码规则（[如图 2.11](#)）

- 用户可以查看每一条代码规范的详细信息

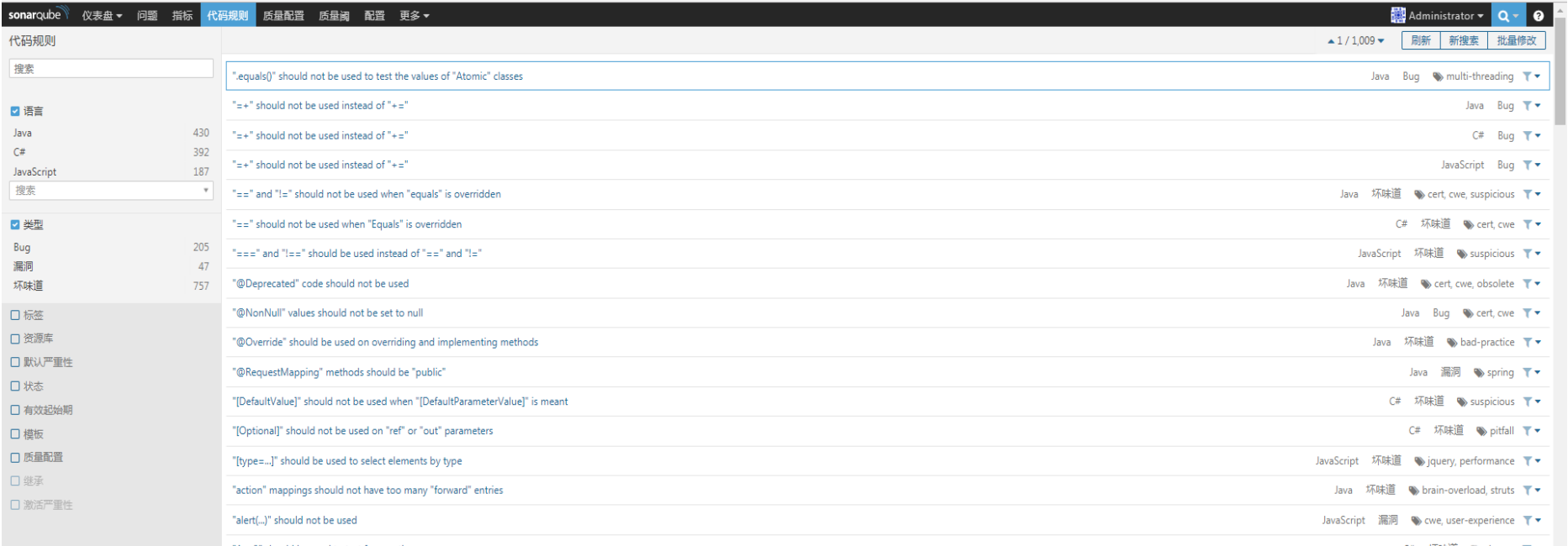


图 2.11

2.2.2 质量配置

- 质量配置：指的是代码检测时，所需要遵守的规范集合；每个语言都有默认配置文件，没有指定特定配置的项目会使用默认配置

- 用户根据自身需要创建自定义的配置文件（图 2.12）



图 2.12

2.2.3 质量阈

- 质量阈是一系列对项目指标进行度量的条件。
 - 项目必须达到所有条件才能算整体上通过了质量阈。
 - 用户可以自定义质量阈，没有特别指定质量阈的项目会设置一个默认的质量阈。

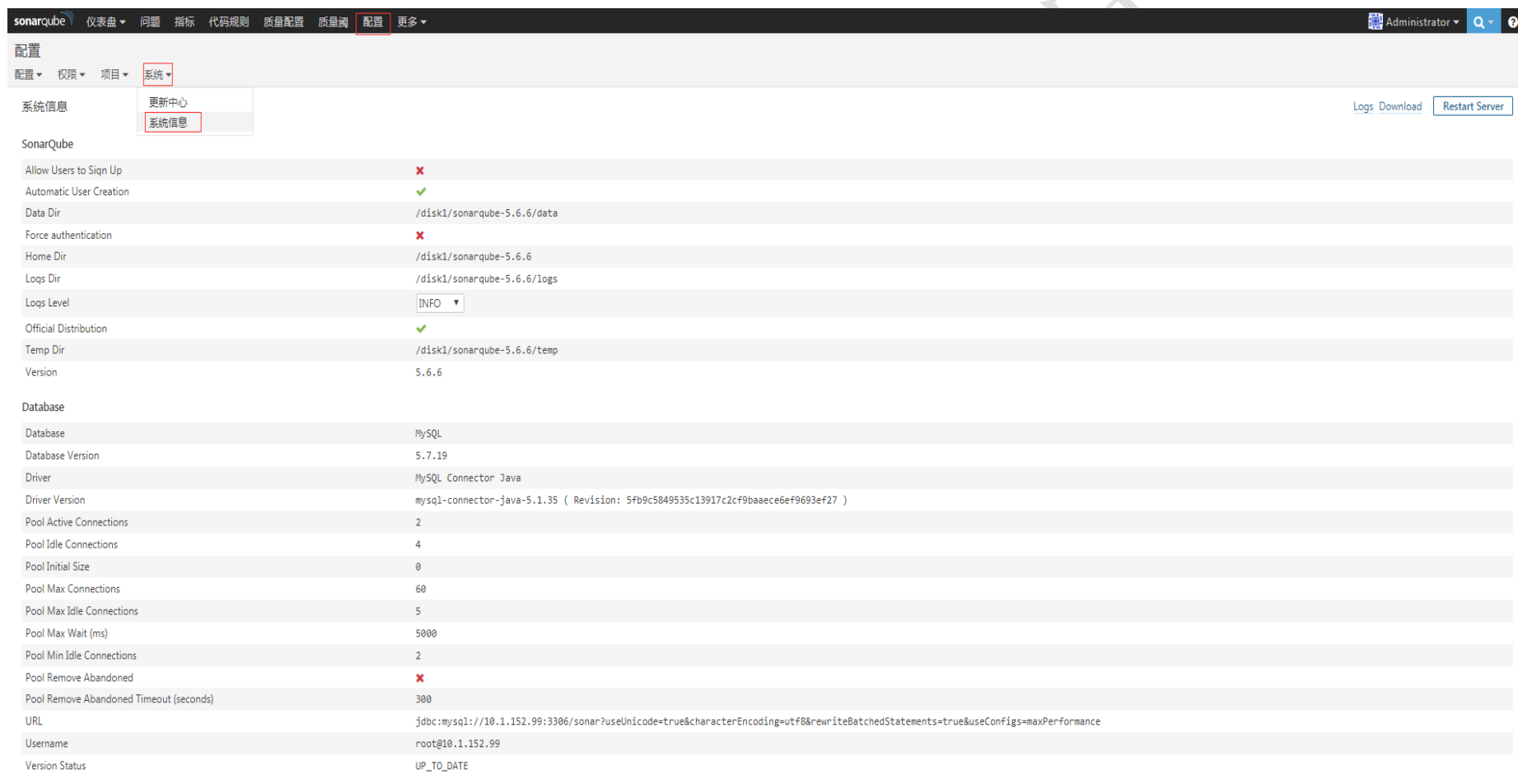
2.2.4 配置

- 通用设置：如图 2.13 编辑 SonarQube 实例的全局设置（参考附录-参数）



图 2.13

- 系统信息：如图 2.14 用户可查看 SonarQube 配置的系统信息



The screenshot shows the SonarQube web interface. The top navigation bar includes '仪表盘', '问题', '指标', '代码规则', '质量配置', '质量阈', '配置', and '更多'. The '配置' (Configuration) section is active, with a sub-menu showing '系统' (System) and '权限' (Permissions). The '系统' sub-menu is expanded, showing '更新中心' (Update Center) and '系统信息' (System Information). The '系统信息' page displays two sections: 'SonarQube' and 'Database'.

Property	Value	Status
SonarQube		
Allow Users to Sign Up		✗
Automatic User Creation		✓
Data Dir	/disk1/sonarqube-5.6.6/data	
Force authentication		✗
Home Dir	/disk1/sonarqube-5.6.6	
Logs Dir	/disk1/sonarqube-5.6.6/logs	
Logs Level	INFO	
Official Distribution		✓
Temp Dir	/disk1/sonarqube-5.6.6/temp	
Version	5.6.6	
Database		
Database	MySQL	
Database Version	5.7.19	
Driver	MySQL Connector Java	
Driver Version	mysql-connector-java-5.1.35 (Revision: 5fb9c5849535c13917c2cf9baace6ef9693ef27)	
Pool Active Connections	2	
Pool Idle Connections	4	
Pool Initial Size	0	
Pool Max Connections	60	
Pool Max Idle Connections	5	
Pool Max Wait (ms)	5000	
Pool Min Idle Connections	2	
Pool Remove Abandoned		✗
Pool Remove Abandoned Timeout (seconds)	300	
URL	jdbc:mysql://10.1.152.99:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance	
Username	root@10.1.152.99	
Version Status	UP_TO_DATE	

图 2.14

2.2.5 更多

- 如图 2.15 用户可以设置代码检测后项目展示的**指标**，也可以对多个项目进行指标比较



图 2.15

3.SonarLint 插件

3.1 环境准备

3.1.1 环境依赖

- SonarLint (<http://www.sonarlint.org/eclipse/index.html>) 是一个 Eclipse 插件，它可以向开发人员及时反馈 Java、JavaScript、PHP、Python...代码中编码时产生的新的缺陷和质量问题
- SonarLint 是开源、免费的，使用时需要注意，它需要运行在 Java 8 环境下，SonarQube 5.6 + 版本（如[图3.1](#)）

➤ SonarQube 5.6+版本（<https://www.sonarqube.org/downloads/>）需要 Mysql 版本 5.6/5.7，Maven 版本 3.x

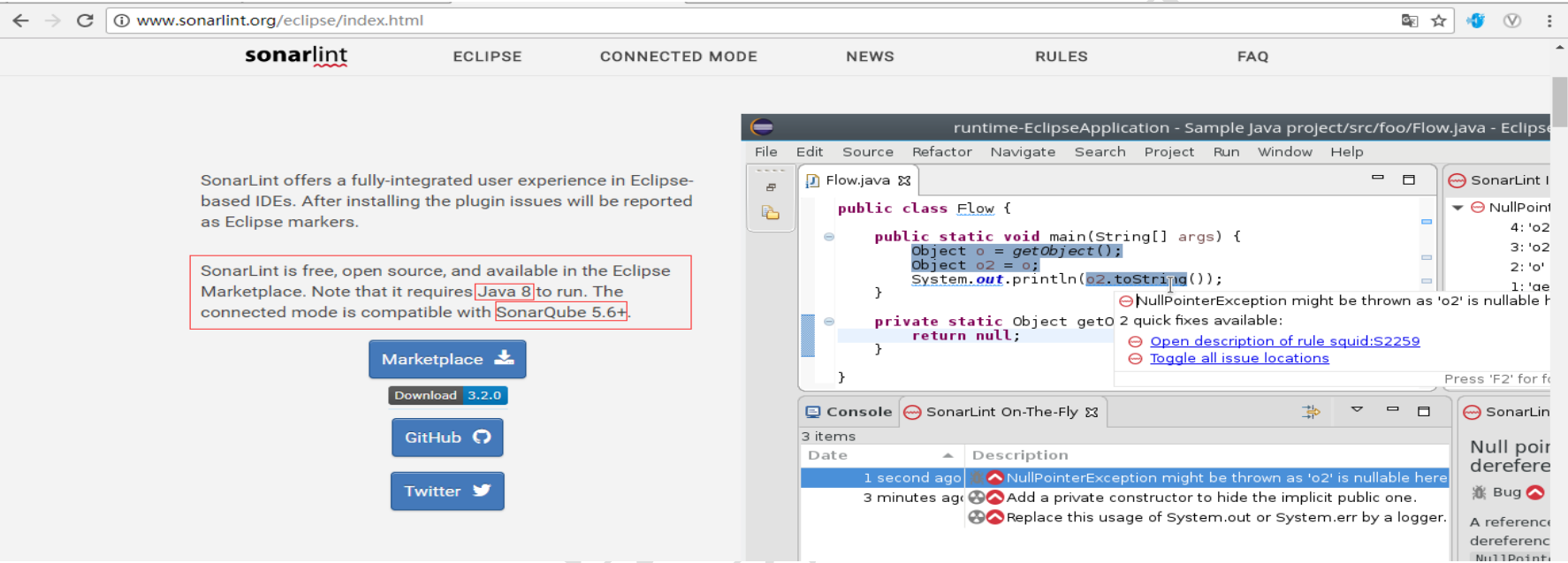


图 3.1

3.1.2 当前服务环境

➤ 如表 1，是当前搭建的 SonarQube 服务器的环境

SonarQube 版本	JDK 版本	Mysql 版本	SonarLint Eclipse 插件
5.6.6	1.8	5.7	3.2.0

表 1

3.2. Eclipse 集成 SonarLint 插件

- 如图 3.2 Eclipse 的 MarkPlace 中查询并安装 SonarLint 插件

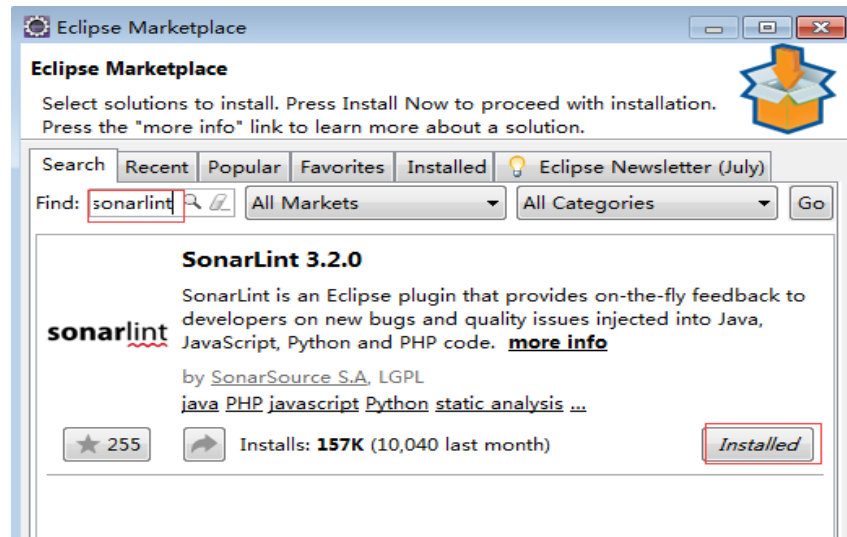


图 3.2

- 安装成功后新增三个视图窗口，如图 3.3：SonarQube Server（连接 SonarQube 服务）、SonarLint Report(显示 SonarQube 默认质量配置下代码扫描报告)、SonarLint On-The-Fly(实时显示 SonarQube 在不同质量配置下代码扫描报告)

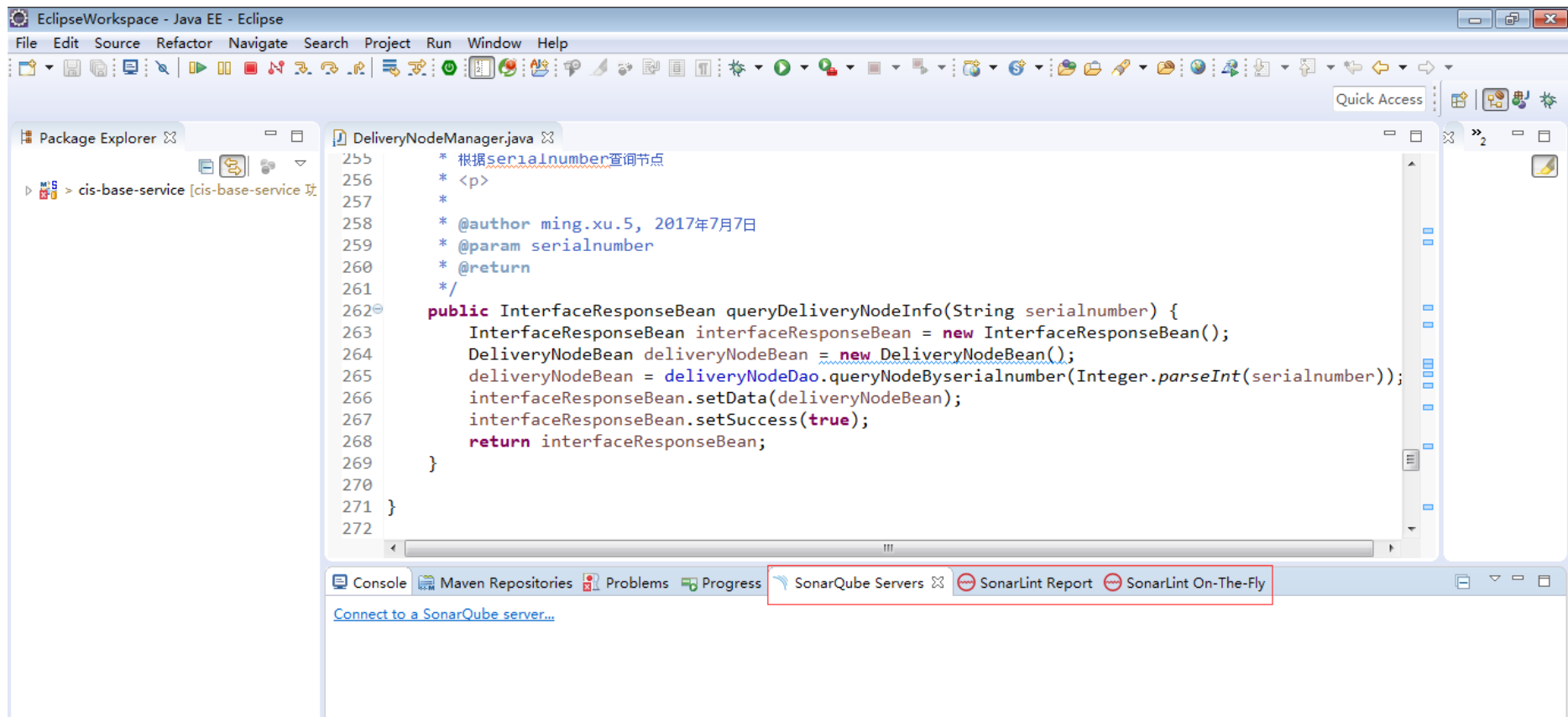


图 3.3

- 绑定 SonarQube 服务器：你可以将 Eclipse 中的项目绑定到一个 SonarQube 项目上（支持 SonarQube 服务器版本 5.6+），下面是项目绑定的步骤
- 第一步配置与 SonarQube 服务器的连接：
 - Eclipse 中 SonarQube 服务器视图中，右键单击选择“New → Server connection”
 - 选择目标服务器: SonarQube 服务器，如下图 3.4
 - 根据向导设置连接的服务器的 url 地址以及账户认证信息
 - 点击“Finish”，服务器后台开始更新配置（可能需要一段时间），如图 3.5 服务器连接成功

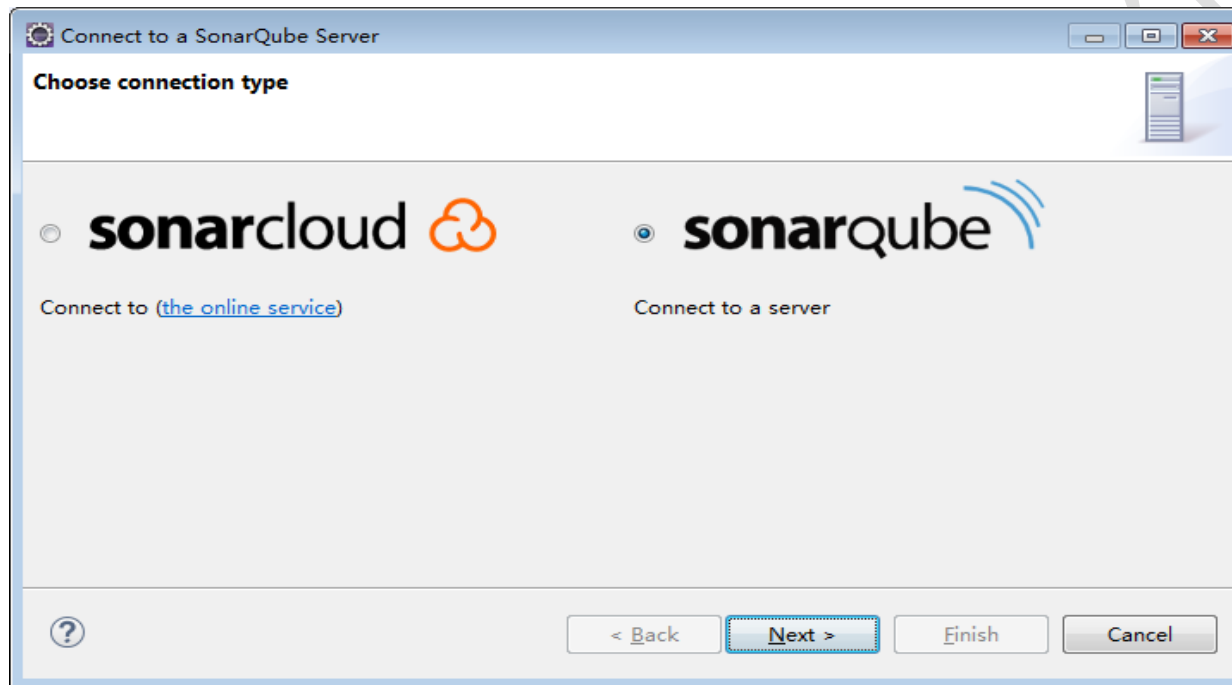


图 3.4

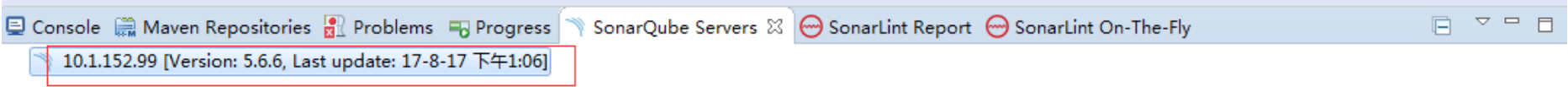


图 3.5

3.3 SonarLint 使用

3.3.1 SonarLint 绑定项目

- 连接上 SonarQube 服务器后，需要将 Eclipse 中项目绑定到 SonarQube 服务器中的项目。下面是项目绑定的步骤
 - 登录 SonarQube 服务器，新建一个项目 cis-base-service，如图 3.6 依次点击操作 **配置**→**项目**→**Management**→**Create Project**

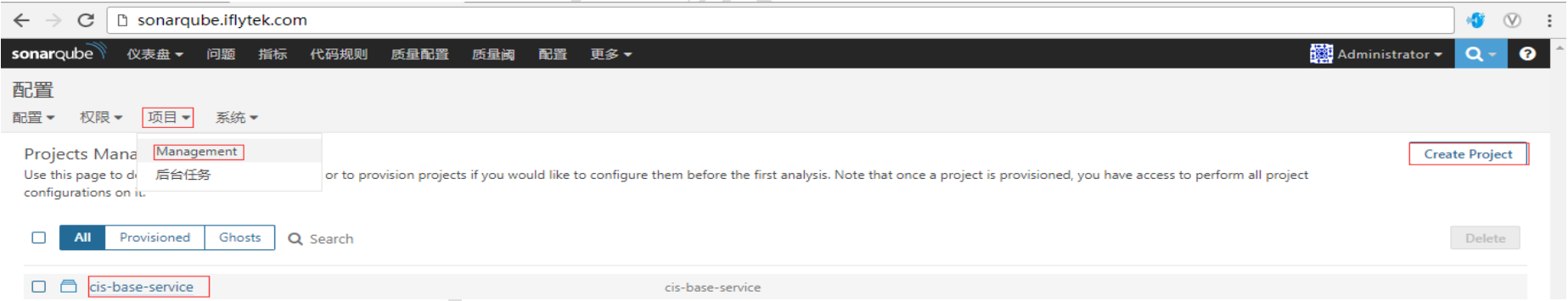


图 3.6

► 如图3.7，将 Eclipse 中的项目 cis-base-service，绑定到 SonarQube 服务器中创建的 cis-base-service 项目

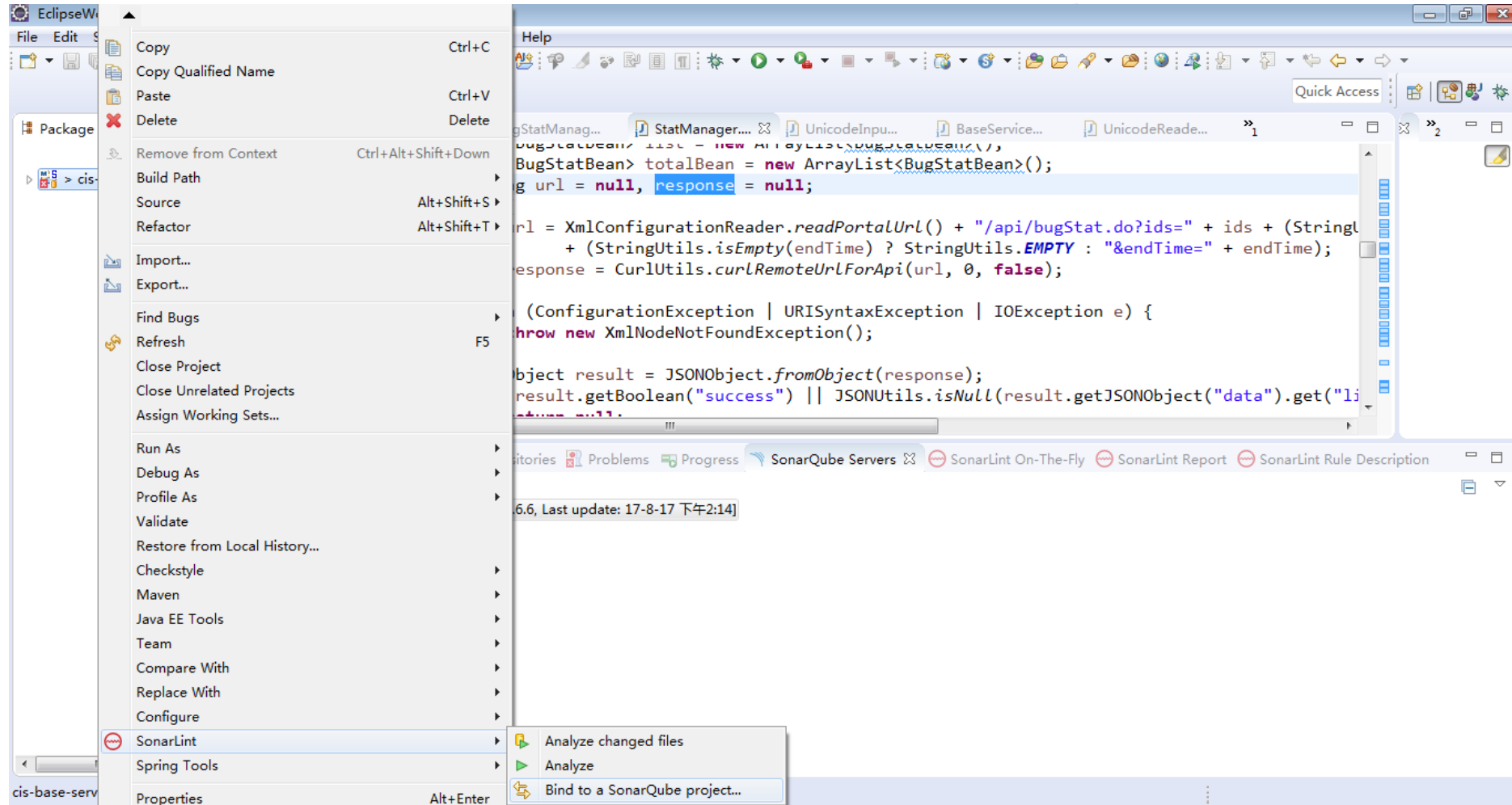


图 3.7

- 如 [图 3.8](#)，先刷新项目列表，然后选择自动绑定选择的项目；如果找不到没有刷新出来你创建的 SonarQube Project，可以手动填入项目名；如 [图 3.9](#)，eclipse project 与 SonarQube Project 绑定成功

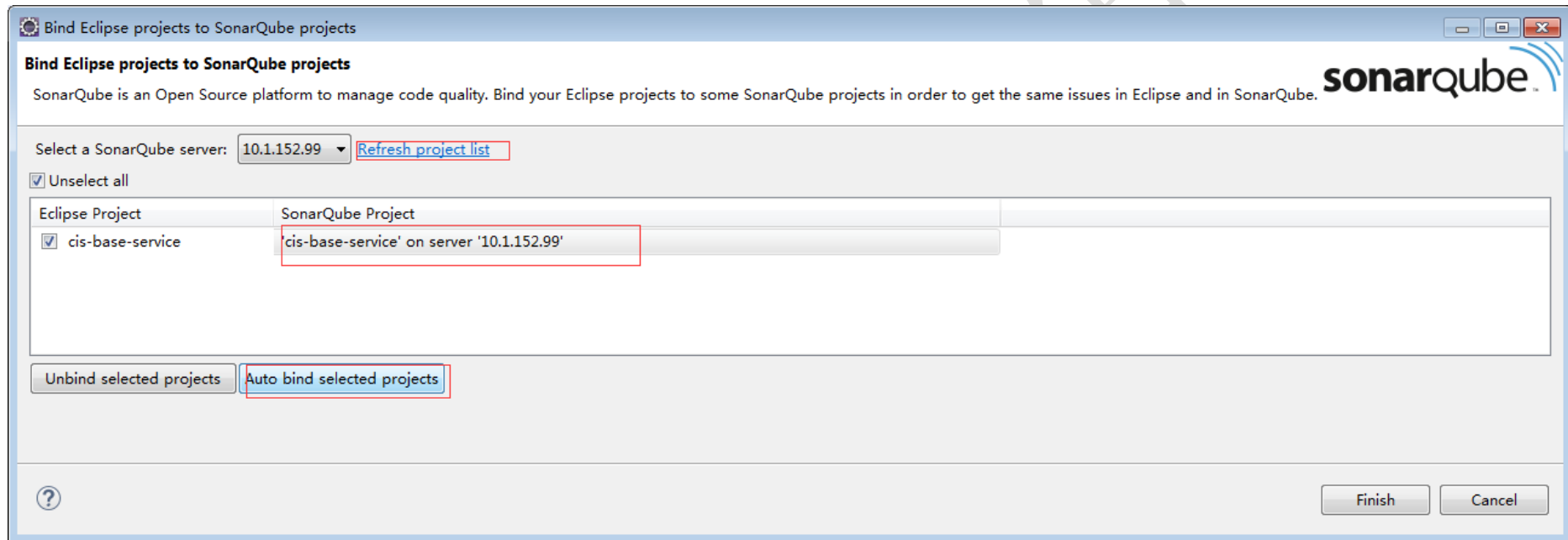


图 3.8

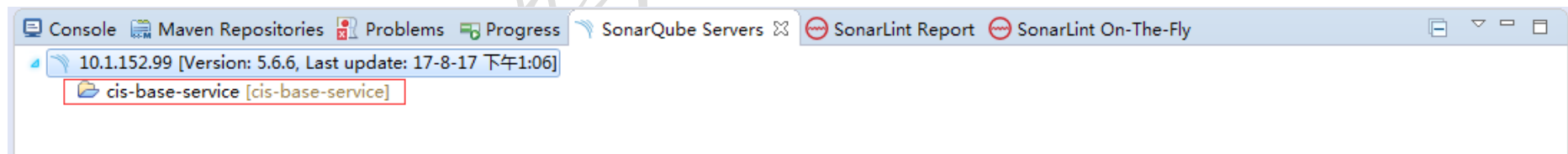


图 3.9

3.3.2 SonarLint 分析项目

- 项目绑定完成后，如 [图 3.10](#)，使用 sonarlint 分析代码；
- [图 3.11](#)，控制台坐下角的倒三角符号可以切换到 SonarLint Console 窗口，查看代码检测进度；
- [图 3.12](#)，代码检测完毕后，可以在 SonarLint Report 窗口查看项目的检测报告。点击每个 description 可以定位到问题位置，并给出问题的表述和修改的建议，如 [图 3.13](#)

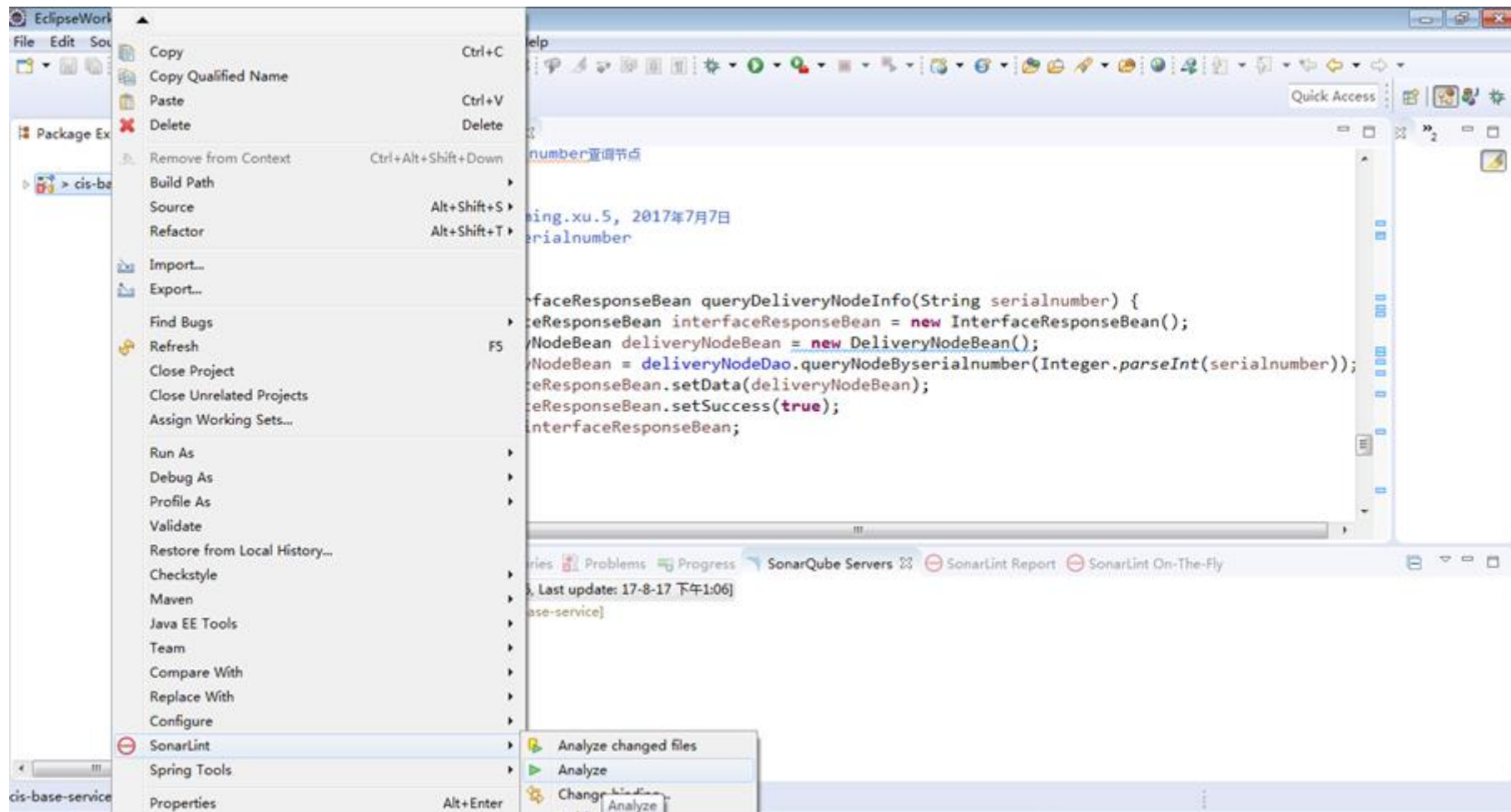


图 3.10

SonarQube User Guide, Release 1.0

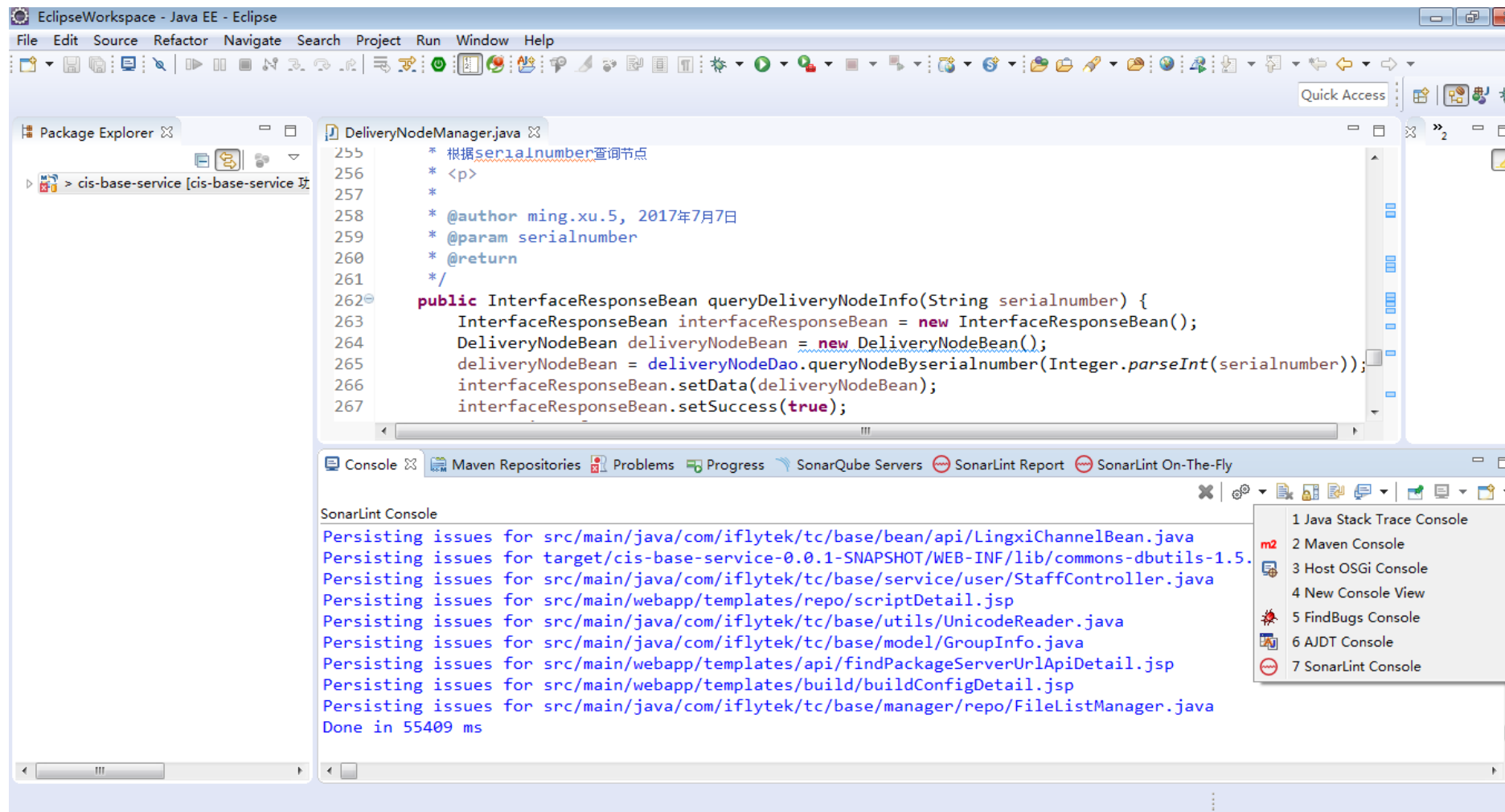


图 3.11

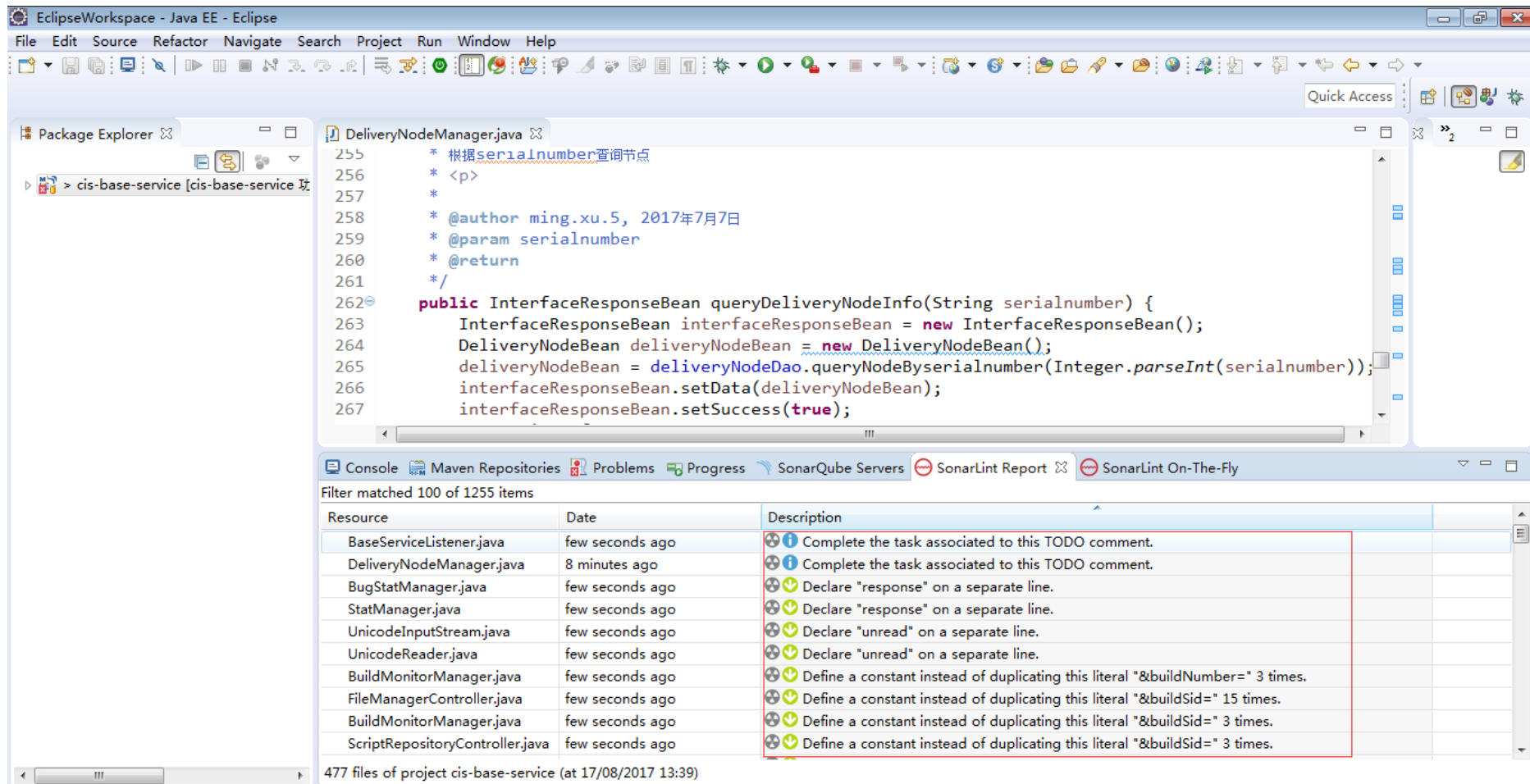


图 3.12

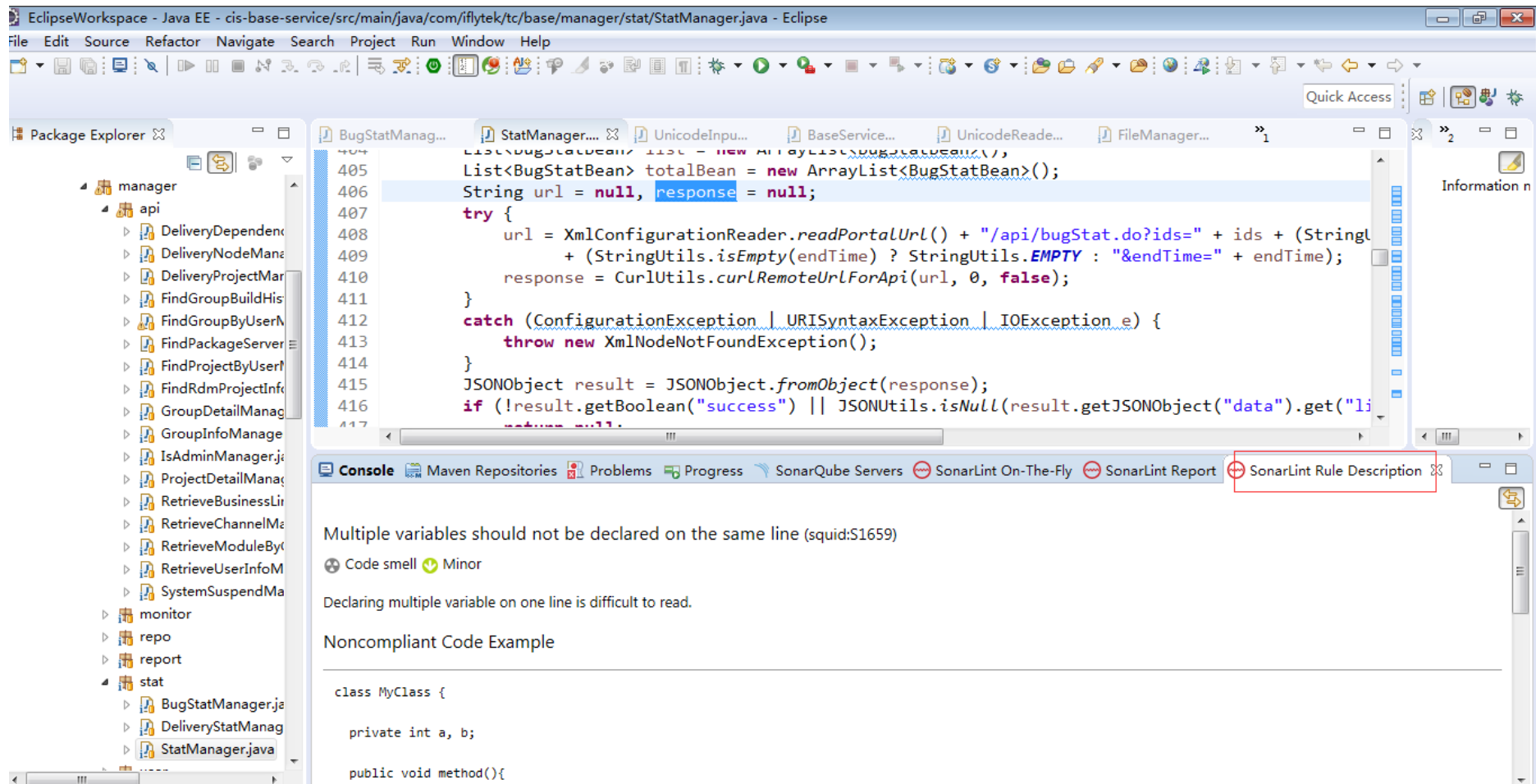


图 3.13

- 如 [图 3.14](#) 使用 **SonarLint** 的时候，基于语言的配置使用的是 **SonarQube** 默认配置；以 java 为例，改变配置后，eclipse 中执行 **SonarLint Analyze** 时，分析代码时仍旧使用的是原先的配置，代码检测标准出现不一致



图 3.14

- 为保持 SonarQube 服务器和 Eclipse 保持配置信息的一致，每次 SonarQube 服务器更新配置后，如 [图 3.15](#) Eclipse 重启都会提醒更新绑定的服务器，用户应及时更新

SonarQube User Guide, Release 1.0

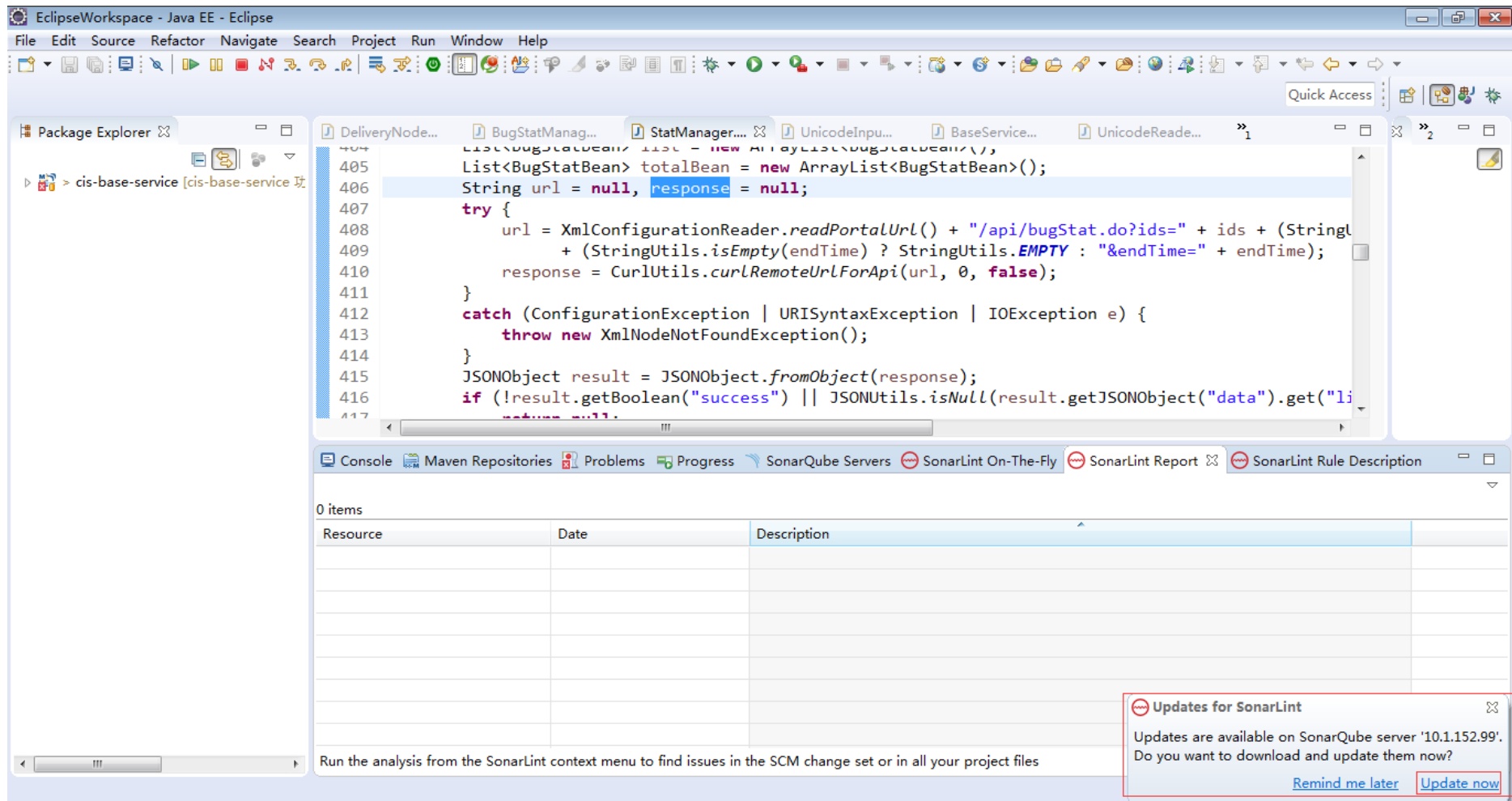


图 3.15

- 假设 SonarQube 服务器静测配置成自己定义的 Server-project（如图 3.16 不包含任何代码规则，Eclipse 更新后，代码分析后就不会出现错误（如图 3.17 有重新更新绑定的 SonarQube 服务器，Eclipse 沿用的将是之前 SonarQube 默认配置，检测就会根据默认配置包含的代码规则检测出问题。

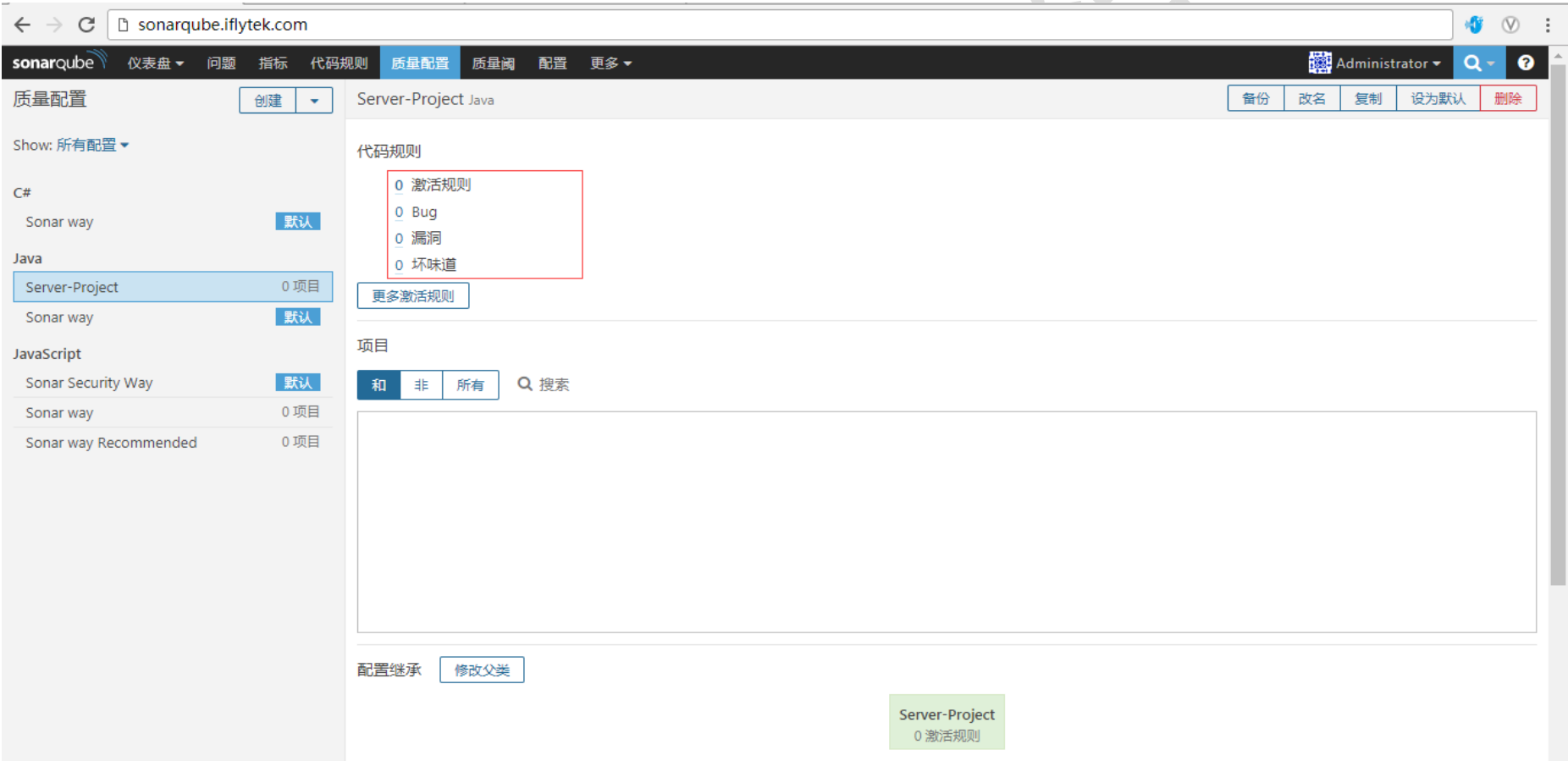


图 3.16

SonarQube User Guide, Release 1.0

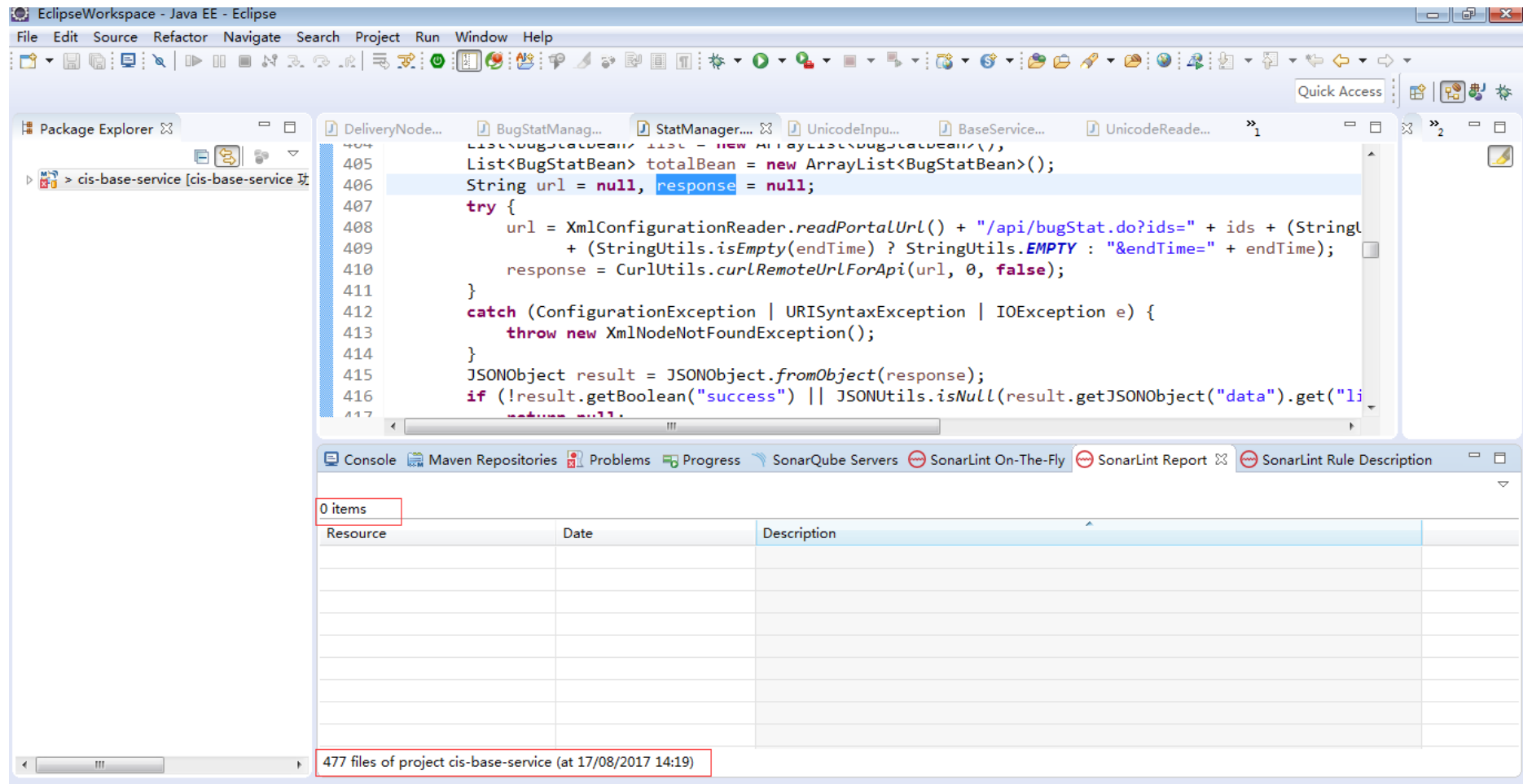


图 3.17

4.SonarQube Scanner

步骤 1.在解压后的包的 conf 文件下，找到 sonar-scanner.properties，配置 sonarqube 服务器地址（<http://sonarqube.iflytek.com/sonar/>），如 图 4.1

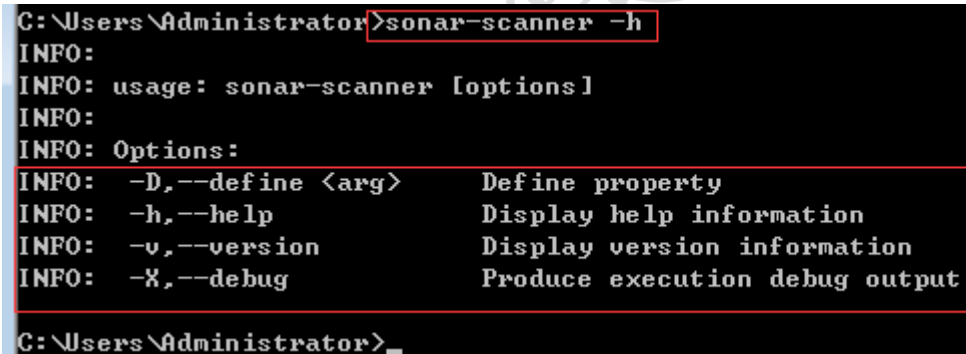
A screenshot of a text editor showing the sonar-scanner.properties file. The file contains configuration for the SonarQube scanner. Line 9, which sets sonar.host.url to http://sonarqube.iflytek.com/sonar/, is highlighted with a red box. Other lines show default settings for the SonarQube server, JDBC connection, and source code encoding.

```
1 #Configure here general information about the environment, such as SonarQube DB details for example
2 #No information about specific project should appear here
3
4 #----- Default SonarQube server
5 #sonar.host.url=http://localhost:9000
6 #sonar.jdbc.username=sonar
7 #sonar.jdbc.password=sonar
8 #sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance
9 sonar.host.url=http://sonarqube.iflytek.com/sonar/
10 #sonar.sourceEncoding=UTF-8
11 #sonar.login=admin
12 #sonar.password=admin
13 #----- Default source code encoding
14 #sonar.sourceEncoding=UTF-8
15
16
```

图 4.1

步骤 2.将 sonar-scanner 的 <install_directory>/bin 目录添加到环境变量

步骤 3.运行命令行，输入 sonar-scanner -h ,如图 4.2 显示表明前面的步骤设置正确

A screenshot of a Windows command prompt window. The user has entered the command 'sonar-scanner -h' at the prompt 'C:\Users\Administrator>'. The output shows the usage and options for the sonar-scanner tool. The command and its output are enclosed in a red box.

```
C:\Users\Administrator>sonar-scanner -h
INFO:
INFO: usage: sonar-scanner [options]
INFO:
INFO: Options:
INFO: -D,--define <arg>      Define property
INFO: -h,--help              Display help information
INFO: -v,--version            Display version information
INFO: -X,--debug              Produce execution debug output

C:\Users\Administrator>
```

图 4.2

步骤 4. [图 4.3](#) 在项目 pom.xml 文件的目录下添加配置文件 [图 4.4](#): **sonar-project.properties** ([附录-参数](#))

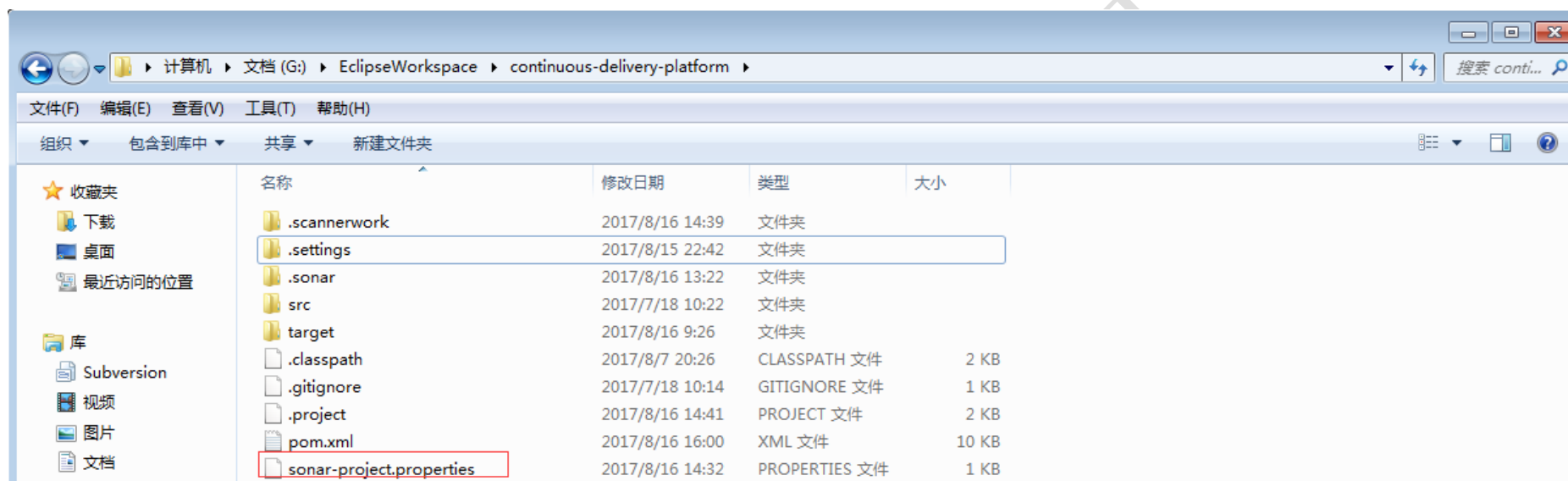
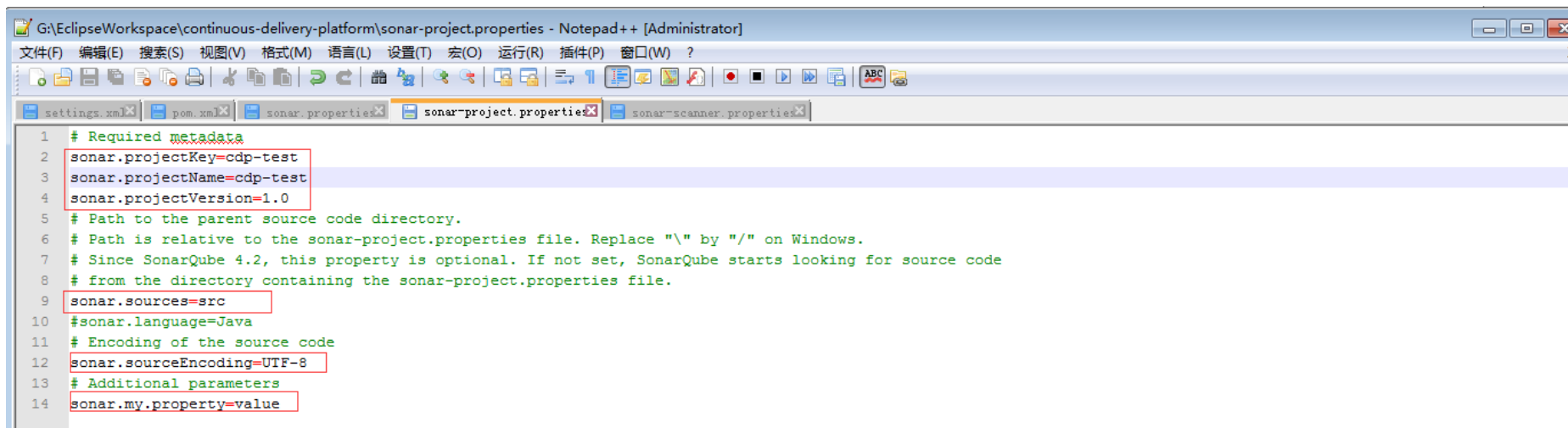


图 4.3



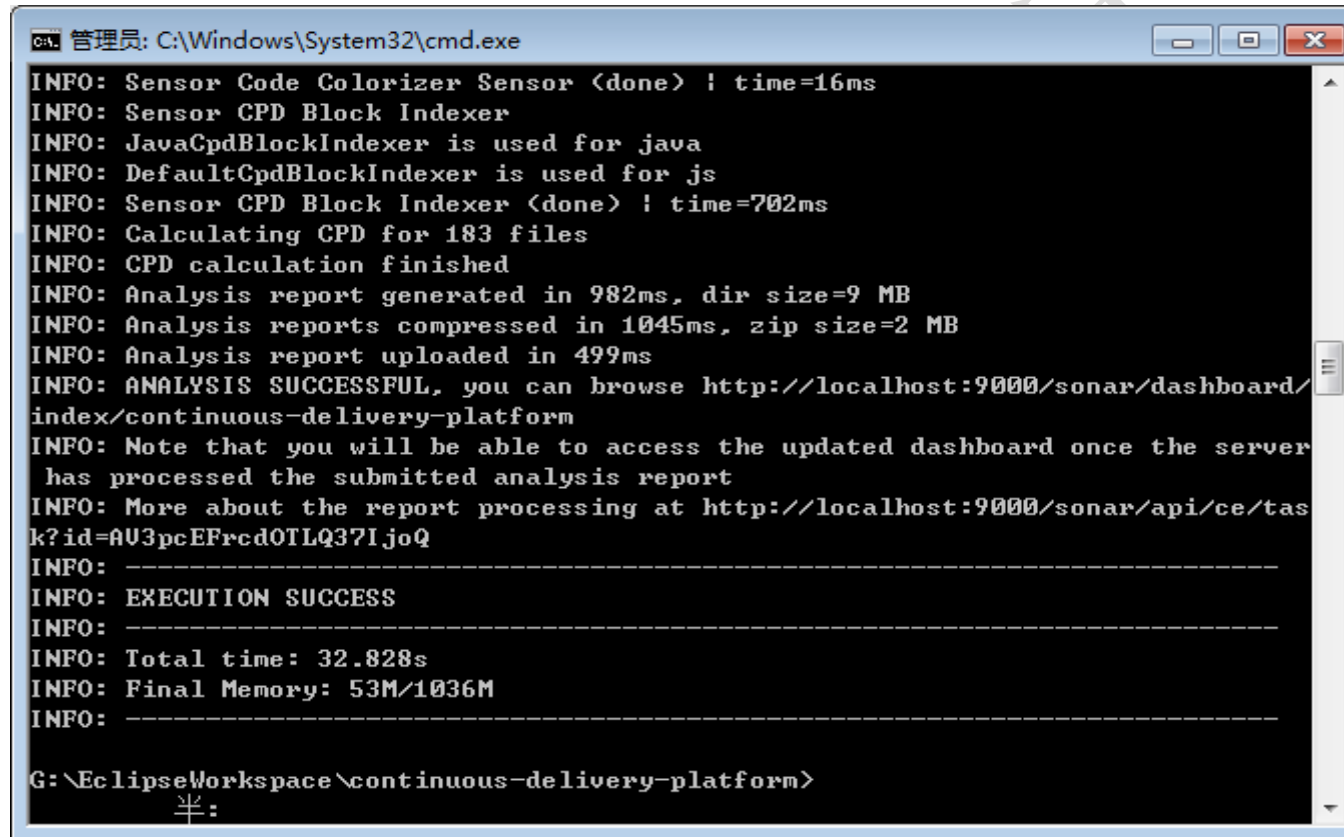
```
G:\EclipseWorkspace\continuous-delivery-platform\sonar-project.properties - Notepad++ [Administrator]
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(M) 语言(L) 设置(T) 宏(O) 运行(R) 插件(P) 窗口(W) ?
settings.xml pom.xml sonar.properties sonar-project.properties sonar-scanner.properties
1 # Required metadata
2 sonar.projectKey=cdp-test
3 sonar.projectName=cdp-test
4 sonar.projectVersion=1.0
5 # Path to the parent source code directory.
6 # Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
7 # Since SonarQube 4.2, this property is optional. If not set, SonarQube starts looking for source code
8 # from the directory containing the sonar-project.properties file.
9 sonar.sources=src
10 #sonar.language=Java
11 # Encoding of the source code
12 sonar.sourceEncoding=UTF-8
13 # Additional parameters
14 sonar.my.property=value
```

图 4.4

注意: sonar.source 指示的是会被 sonarqube 服务器扫描的项目目录，如上配置仅供参考，详细信息请参考附录，用户可视自己项目情况，自行配置参数

步骤 5. 如图 4.5 项目目录路径下运行 cmd，输入命令：**sonar-scanner**

SonarScanner 扫描代码并将项目分析报告上传到远程 SonarQube Server



```
管理员: C:\Windows\System32\cmd.exe
INFO: Sensor Code Colorizer Sensor <done> ! time=16ms
INFO: Sensor CPD Block Indexer
INFO: JavaCpdBlockIndexer is used for java
INFO: DefaultCpdBlockIndexer is used for js
INFO: Sensor CPD Block Indexer <done> ! time=702ms
INFO: Calculating CPD for 183 files
INFO: CPD calculation finished
INFO: Analysis report generated in 982ms, dir size=9 MB
INFO: Analysis reports compressed in 1045ms, zip size=2 MB
INFO: Analysis report uploaded in 499ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/sonar/dashboard/
index/continuous-delivery-platform
INFO: Note that you will be able to access the updated dashboard once the server
has processed the submitted analysis report
INFO: More about the report processing at http://localhost:9000/sonar/api/ce/tas
k?id=AU3pcEFrcd0TLQ37IjoQ
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 32.828s
INFO: Final Memory: 53M/1036M
INFO: -----

G:\EclipseWorkspace\continuous-delivery-platform>
```

图 4.5

4.如图 4.6 访问 SonarQube Server，项目的代码检测报告已经上传到页面

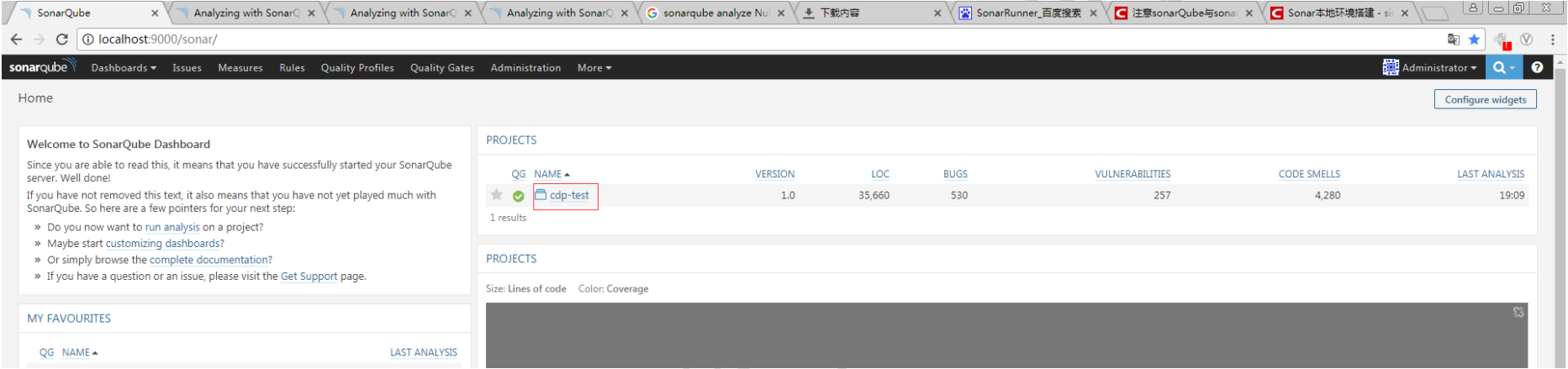


图 4.6

5.附录

5.1 参数

5.1.1 强制参数

5.1.1.1 服务

关键词	描述	默认值
Sonar 访问地址(sonar.host.url)	Sonar 服务的 url 地址	http://localhost:9000

5.1.1.2 项目配置

关键词	描述	默认值
项目关键字(sonar.projectKey)	项目的关键词,对于每一个项目都是独特的。 允许的字符:字母、数字、“-”、“_”、“。”和“:”,至少一个 non-digit。	

	当使用 Maven , 它会自动设置为< groupId >:< artifactId >。	
项目源代码地址(sonar.sources)	用逗号分隔包含源文件目录的路径, 与 maven 兼容。如果没有设置, 默认检索 maven 中设置的源代码地址。	

5.1.2 可选参数

5.1.2.1 项目标识

关键词	描述	默认值
项目名(sonar.projectName)	显示在 web 界面的项目名称, 若使用 maven , 可通过<name>标签设置项目名	<ul style="list-style-type: none"> ■ 如果项目名未定义, 使用项目的关键字 ■ 如果项目名已定义, 不重写
项目版本号(sonar.projectVersion)	项目的版本号, 若使用 maven , 可通过<version>标签进行修改	

5.1.2.2 身份验证

如果用户没有权限执行代码分析, 你需要为用户开通权限

关键词	描述	默认值
账号(sonar.login)	用户使用 SonarQube 登录或身份验证的账号	
密码(sonar.password)	与用户登录账号对应的密码	

5.1.2.3 Web 服务

关键词	描述	默认值
超时(sonar.ws.timeout)	等待一个 Web 服务调用的响应最长时间(以秒为单位)	60

5.1.2.4 项目配置

关键词	描述	默认值
描述 (sonar.projectDescription)	项目的描述, maven 不兼容, 使用<description>标签设置	
主页 (sonar.links.homepage)	项目的主页, maven 不兼容, 使用<url>标签设置	
Ci(sonar.links.ci)	持续集成, maven 不兼容, 使用<ciManagement><url>标签设置	
问题(sonar.links.issue)	问题跟踪, maven 不兼容, 使用<issueManagement><url>标签设置	
Scm(sonar.links.scm)	项目源码地址, maven 不兼容, 使用<scm><url>标签设置	
scm 开发者 (sonar.links.scm_dev)	开发人员链接, maven 不兼容, 使用<scm><developerConnection>标签设置	
测试(sonar.tests)	包含测试的路径目录使用逗号分隔, maven 不兼容, 从默认位置检索测试 java maven 项目	
语言(sonar.language)	设置源代码分析的语言, 浏览插件库页面列表中可以查看所有可用的语言。如果没有设置, 多语言分析将被触发	
源文件编码 (sonar.sourceEncoding)	设置源文件编码, 例如 UTF-8,MacRoman, maven 项目可以通过 project.build.sourceEncoding 属性设置	系统编码
日期(sonar.projectDate)	设定源码分析的日期	当前日期

分支(sonar.branch)	管理 scm 的分支，相同项目的两个分支在 sonarqube 中被认为是不同的项目	
配置文件(sonar.profile)	从 4.5LTS 版本之后，这个属性已经弃用，不应使用	
项目源文件地址 (sonar.projectBaseDir)	可能是相对或绝对路径	
工作目录 (sonar.working.directory)	SonarQube 扫描和 SonarQube Ant 任务会触发创建工作目录；每个项目是相对独立的，注意：源代码分析前会把指定的文件夹删除	.sonar
提供者 (sonar.scm.provider)	这个属性可以用来明确告知 SonarQube 的那个 SCM 插件应该用于抓取 SCM 项目数据；这个属性的值一定是小写	
强制重新加载 (sonar.scm.forceReloadAll)	这个属性用于在你觉得一些 scm 的数据过时，重新抓取最新数据	false

5.1.2.5 排除/附增

缩小问题排查范围的方式

- 分析排除文件
- 防止一些文件被重复检查
- 防止一些文件的单元测试和集成测试被算入代码覆盖率
- 忽略特定组件、特定编码规则产生的问题

关键词	描述	默认值
包含(sonar.inclusions)	以逗号分隔文件路径列表模式，只有匹配的文件路径列表模式才会被分析	
排除(sonar.exclusions)	以逗号分隔的文件路径列表模式被排除在分析之外。	
覆盖范围(sonar.coverage.exclusions)	以逗号分隔的文件路径列表模式被计算在覆盖范围内	
测试排除(sonar.test.exclusions)	以逗号分隔的测试路径列表模式被排除在分析之外	

测试包含(sonar.test.inclusions)	以逗号分隔测试路径列表模式，只有匹配的文件路径列表模式才会被分析	
错误忽略(sonar.issue.ignore.allfile)	满足正则表达式的文件会在分析时被忽略	
导入未知文件(sonar.import_unknown_files)	如果设置为真，没有安装匹配编译语言插件的文件都会被 SonarQube 分析包含或排除	False
重复检测排除(sonar.cpd.exclusions)	以逗号分隔文件路径列表模式被重复检测排除	

5.1.2.6 重复

关键词	描述	默认值
重复检测排除(sonar.cpd.exclusions)	(查看 排除/附增部分 的表格)	
sonar.cpd.\${language}.minimumtokens	每百行出现 10 行重复代码被认为是重复；java 项目，一部分代码若出现 10 行重复的代码就认为是重复。	100
sonar.cpd.\${language}.minimumLines		10

5.2 指标

5.2.1 复杂度

指标名	关键词	描述
复杂度	Complexity	代码能否分割成细粒度的函数、是否能够被清晰的理解、是否清晰的表达，是衡量代码复杂度的标准
认知复杂	Cognitive_Complexity	理解代码控制流的难易程度
类复杂度	class_complexity	类的平均复杂度
文件复杂度	file_complexity	文件的平均复杂度
方法复杂度	function_complexity	方法的平均复杂度

5.2.2 软件文档

指标名	关键词	描述
注释行	comment_lines	包含注释和注释行；没有实际意义的行数不算入；下面是例子

		<pre>/** +0 => empty comment line * +0 => empty comment line * This is my documentation +1 => significant comment * although I don't +1 => significant comment * have much +1 => significant comment * to say +1 => significant comment * +0 => empty comment line ***** +0 => non-significant comment * +0 => empty comment line * blabla... +1 => significant comment */ +0 => empty comment line /** +0 => empty comment line * public String foo() { +1 => commented-out code * System.out.println(message); +1 => commented-out code * return message; +1 => commented-out code * } +1 => commented-out code */ +0 => empty comment line</pre>
注释率	comment_lines_density	<p>注释行密度=注释行/(代码行+注释行)* 100</p> <p>有这样一个公式:</p> <p>50%表示代码行数等于注释行数</p> <p>100%意味着该文件仅包含注释行</p>

公共记录 API 率	public_documented_api_density	公共记录的 API $= (\text{公共 API} - \text{公共非法 API}) / \text{公共 API} * 100$
公共非法 API	public_documented_api_density	没有注释头的公共 API
注释掉的代码行	commented_out_code_lines	注释代码行数

5.2.3 重复

指标名	关键词	描述
重复的代码块	duplicated_blocks	<p>重复的行块数，被认为是重复的代码块：</p> <ul style="list-style-type: none"> 非 java 项目： <p>应该至少有 100 个连续和重复的令牌。</p> <p>这些令牌应该至少在：</p> <p>COBOL 语言的 30 行代码</p> <p>20 行 ABAP 代码</p> <p>其他语言的 10 行代码</p> Java 项目 <p>无论符号和行数，都应该至少有 10 个连续和重复的语句。</p> <p>在检测重复时，忽略了缩进和字符串文字的差异。</p>
重复文件数	duplicated_files	重复的文件的数量
重复行数	duplicated_lines	重复的行数
重复行率	duplicated_lines_density	重复行率=重复行数/总行数*100

5.2.4 问题

指标名	关键词	描述
新的问题	new_violations	新的问题数
新的 xxx 问题	new_xxxxx_violations	新问题的严重性 xxx,xxx 是阻断，关键，主要，次要或信息。
问题	violations	问题数
xxx 问题	xxxxx_violations	严重程度 xxxxx,xxxxx 是阻断，关键，主要，次要或信息。
假阳性问题	false_positive_issues	假阳性问题的数量
已知问题	open_issues	开放的问题数量
确认问题	confirmed_issues	确认的问题数量
重新开放的问题	reopened_issues	重新开放的问题的数量

5.2.5 严重性

指标名	关键词	描述
阻断	Blocker	操作/安全风险：在生产中这个问题可能会使整个应用程序不稳定。例如：调用垃圾收集器，而不是关闭套接字等等
严重	Critical	操作/安全风险：在生产过程中这个问题可能导致出现意外行为，而不会影响整个应用程序的完整性。例如:NullPointerException，严重捕获异常，缺少单元测试等
主要	Major	这个问题可能会对生产效率产生影响。例如：太复杂的方法，包装循环等
次要	Minor	这个问题可能会对生产效率产生影响。例如：命名约定，终结器只调用超类终结器等等
提示	Info	可能会对生产效率产生影响的未知或尚未明确定义的安全风险

5.2.6 可维护性

指标名	关键词	描述
代码异味	code_smells	如果一段代码是不稳定或者有一些潜在问题的，那么代码往往会包含一些明显的痕迹。正如食物要腐坏之前，经常会发出一些异味一样。我们管这些痕迹叫做“代码异味”
新的代码坏异味	new_code_smells	新产生的代码异味
可维护性等级 (以前的 SQALE 等级)	sqale_rating	与你项目的技术债务比率有关的评级。默认的可维护性评价等级量表是： A=0-0.05, B=0.06-0.1, C=0.11-0.20, D=0.21-0.5, E=0.51-1 可维护性等级量表可以交替地说明，如果未解决的补救费用是： 小于等于已进入应用程序的时间的 5%，评级为 A 在 6 到 10%之间，评级为 B 11 到 20%之间的等级是 C 在 21 到 50%之间，评级为 D 任何超过 50%的都是 E
技术债	sqale_index	修复所有可维护性问题所付出的努力。度量标准是对数据库操作的时间
新的代码技术债	new_technical_debt	新代码的技术债务
技术债务比率	sqale_debt_ratio	开发软件的成本与修复软件的成本之间的比率。技术债务比率公式为： 补救成本/开发成本 可重述为： 补救成本/(开发 1 行代码的成本*代码行数) 开发一行代码的成本是 0.06 天
新代码的技术债务比率	new_sqale_debt_ratio	在漏洞出现期间，开发代码的成本与与之相关的问题成本之间的比率

5.2.7 质量阈

指标名	关键词	描述
质量阈状态	alert_status	项目质量阈的状态。可能的值是:错误, 警告, 通过
质量阈详情	quality_gate_details	所有条件下的质量阈, 你知道哪个条件会导致质量阈失败, 哪个不会

5.2.8 安全性

指标名	关键词	描述
漏洞	vulnerabilities	漏洞的数量
新产生的漏洞	new_vulnerabilities	新产生的漏洞数量
安全等级	security_rating	A = 0 漏洞 B = 至少 1 个次要漏洞 C = 至少 1 个主要漏洞 D = 至少 1 个严重漏洞 E = 至少 1 个阻断漏洞
安全补救工作	security_remediation_effort	修复所有漏洞问题所做的努力。度量标准是修复期间操作数据库的时间
新漏洞的安全补救工作	new_security_remediation_effort	与安全补救措施一样, 在缺陷发生时更改了代码。

5.2.9 指标

指标	关键词	描述
类	classes	类的数量(包括嵌套类、接口、枚举和注释)。
目录	directories	目录的数量
文件	files	文件的数量
行	lines	物理行数(回车数)。
代码行	Ncloc	包含至少一个字符的物理行数，既不是空白，也不是表格或注释的一部分
任意编译语言的代码行	ncloc_language_distribution	没有注释的代码行
方法	functions	方法的数量。取决于语言，要么是函数，要么是方法，要么是段落。
项目	projects	视图中的项目数量
公共的接口	public_api	公共类的数量+公共方法的数量+公共属性的数量
声明	statements	声明的数量

5.2.10 测试

指标	关键词	描述
条件覆盖	branch_coverage	每一行代码中包含一些布尔表达式的，条件覆盖仅仅回答以下问题：“是否对每个布尔表达式都进行了正确和错误的评估？”这是在单元测试执行过程，流控制结构中可能出现的情况的密度。
新代码的条件覆盖	new_branch_coverage	与条件覆盖相同，但仅限于新的/更新的源代码
条件覆盖采样率	branch_coverage_hits_data	覆盖条件的列表
条件行	conditions_by_line	条件的行数
覆盖条件的行	covered_conditions_by_line	覆盖了条件的行数

覆盖	coverage	它是行覆盖与条件覆盖的结合，目的是显示：单元测试覆盖了多少源代码？
新代码的覆盖	new_coverage	与覆盖相同，但限于新的/更新的源代码。
行覆盖	line_coverage	在给定的代码行中，行覆盖仅仅回答以下问题：在单元测试执行期间执行了这一行代码了吗？
要覆盖新代码的行	new_line_coverage	与行覆盖相同，但仅限于新的/更新的源代码。
行覆盖率	coverage_line_hits_data	行覆盖列表
可覆盖的行	lines_to_cover	可以通过单元测试覆盖的代码行数(例如，空行或完整的注释行不被认为是用来覆盖的行)。
新代码要覆盖的行	new_lines_to_cover	与可覆盖的行相同，但限于新的/更新的源代码。
跳过单元测试	skipped_tests	跳过单元测试的数量。
未覆盖的条件	uncovered_conditions	单元测试不包括的条件数。
未覆盖条件的新代码	new_uncovered_conditions	与未覆盖的条件相同，但仅限于新的/更新的源代码。
未覆盖的行	uncovered_lines	单元测试没有覆盖的代码行数。
在新代码中未覆盖的行	new_uncovered_lines	与未覆盖的行相同，但仅限于新的/更新的源代码。
单元测试	tests	单元测试的数量。
单元测试持续时间	test_execution_time	执行所有单元测试所需的时间。
单元测试的错误	test_errors	失败的单元测试数。
单元测试失败数	test_failures	在异常中失败的单元测试数。

6. 参考资料

SonarQube 下载官网: <https://www.sonarqube.org/downloads/>

SonarQube 官方使用文档: <https://docs.sonarqube.org/display/SONAR/Documentation>

SonarLint 官网: <http://www.sonarlint.org/eclipse/index.html>