**SonarQube学习资料**

# SonarQube是什么

[SonarQube](http://www.sonarqube.org/) 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。 通过插件形式，可以支持包括 java, C#, C/C++, PL/SQL, Cobol, JavaScrip, Groovy 等等二十几种编程语言的代码质量管理与检测。Sonar可以从以下七个维度检测代码质量，而作为开发人员至少需要处理前5种代码质量问题。

**1. 不遵循代码标准**

sonar可以通过PMD,CheckStyle,Findbugs等等代码规则检测工具规范代码编写。

**2. 潜在的缺陷**

sonar可以通过PMD,CheckStyle,Findbugs等等代码规则检测工具检 测出潜在的缺陷。

**3. 糟糕的复杂度分布**

文件、类、方法等，如果复杂度过高将难以改变，这会使得开发人员 难以理解它们, 且如果没有自动化的单元测试，对于程序中的任何组件的改变都将可能导致需要全面的回归测试。

**4. 重复**

显然程序中包含大量复制粘贴的代码是质量低下的，sonar可以展示 源码中重复严重的地方。

**5. 注释不足或者过多**

没有注释将使代码可读性变差，特别是当不可避免地出现人员变动 时，程序的可读性将大幅下降 而过多的注释又会使得开发人员将精力过多地花费在阅读注释上，亦违背初衷。

**6. 缺乏单元测试**

sonar可以很方便地统计并展示单元测试覆盖率。

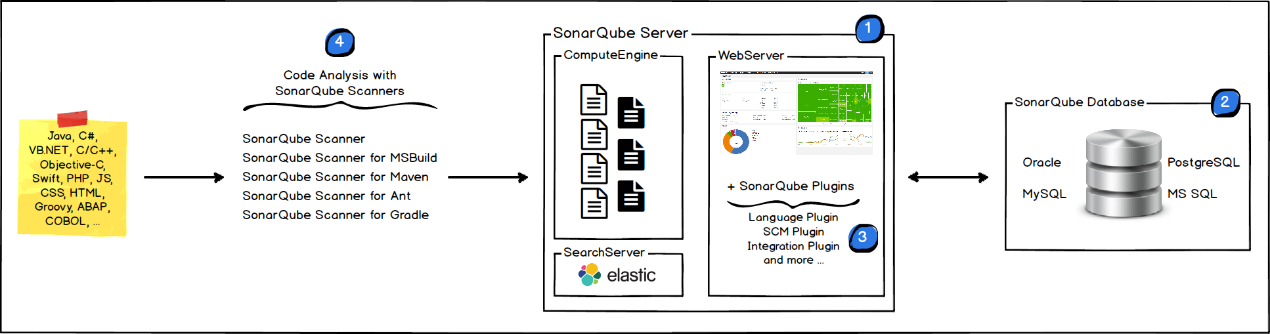
**7. 糟糕的设计**

通过sonar可以找出循环，展示包与包、类与类之间的相互依赖关系，可以检测自定义的架构规则 通过sonar可以管理第三方的jar包，可以利用LCOM4检测单个任务规则的应用情况， 检测耦合。

# SonarQube构架

SonarQube框架包含以下四个部分：

* Project
* SonarQube Scanner
* SonarQube Server
* SonarQube Database



### Project

是需要被分析的源码，如我们的app工程源码，SonarQube支持多种语言和多种工程结构，Andriod是属于一种多模块的java工程。

### SonarQube Scanner

是用于执行代码分析的工具，在Project的根目录下执行，我们还需要在Project下进行[SonarQube配置](https://blog.csdn.net/skyunlin/article/details/50673742#sonarqube%E9%85%8D%E7%BD%AE)，其中指定了工程的相关信息，还指定了SonarQube Server的地址，SonarQube Scanner分析完毕之后，会将结果上报到该Server。

注：官方对Scanner的描述是“The SonarQube Scanner is recommended as the default launcher to analyze a project with SonarQube”，个人理解是可以用任何其它的工具替代，只要能对Source进行分析，并生成Server能构解析的数据格式上报给Server

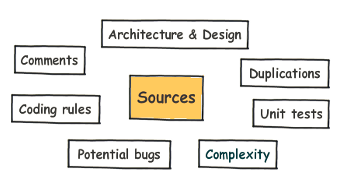
### SonarQube Server

显示分析结果的Web Server，在SonarQube Scanner第一次将一个工程的分析结果上报给SonarQube Server后，Server上会自动创建一个工程显示分析的结果，可以在Server上设置代码质量管理相关的各种配置，如设置代码检查规则（Rule）和质量门限（Quality Gate）等。

# SonarQube 的工作原理介绍

SonarQube 并不是简单地将各种质量检测工具的结果（例如 FindBugs，PMD 等）直接展现给客户，而是通过不同的插件算法来对这些结果进行再加工，最终以量化的方式来衡量代码质量，从而方便地对不同规模和种类的工程进行相应的代码质量管理。

SonarQube 在进行代码质量管理时，会从图 1 所示的七个纬度来分析项目的质量

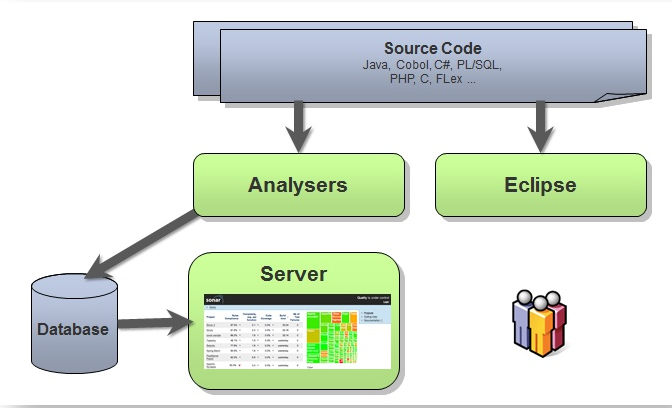


SonarQube 可以支持 25+ 种编程语言，针对不同的编程语言其所提供的分析方式也有所不同：

1. 对于所有支持的编程语言，SonarQube 都提供源了代码的静态分析功能；
2. 对于某些特定的编程语言，SonarQube 提供了对编译后代码的静态分析功能，比如 java 中的 class file 和 jar 和 C# 中的 dll file 等；
3. 对于某些特定的编程语言，SonarQube 还可以提供对于代码的动态分析功能，比如 java 和 C# 中的单元测试的执行等。

SonarQube 平台的整体架构如图 2 所示：

图 2. SonarQube 的服务架构



远程客户机可以通过各种不同的分析机制，从而将被分析的项目代码上传到 SonarQube server 并进行代码质量的管理和分析，SonarQube 还会通过 Web API 将分析的结果以可视化、可度量的方式展示给用户。

# SonarQube配置

SonarQube支持多种工程构建方式的配置，也对应需要用不同的Scanner来执行分析过程：

* Ant
* Maven
* MSBuild（Microsoft Build Engine）
* Gradle
* Sonar Runner

## Scanner配置说明

* scanner：[sonar-runner](http://docs.sonarqube.org/display/SONAR/Installing+and+Configuring+SonarQube+Scanner" \t "_blank)
* config方式：sonar-project.properties (.properties文件是java工程的一种配置文件，格式为”key=value”)
* 运行方式：在工程根目录下执行sonar-runner

sonar-project.properties范例

sonar.host.url=http://sonarqube.example.com

sonar.projectKey=AndroidDemo

sonar.projectName=AndroidDemo

sonar.projectVersion=2.3.2

sonar.sources=src/main/java

sonar.sourceEncoding=UTF-8

sonar.language=java

sonar.modules=app

presentation.sonar.projectBaseDir=app

# SonarQube3.6 代码质量分析

当使用 SonarQube3.6 对某一个项目进行代码质量管理时，通常可以采用三种不同的方法来发起代码分析，分别为 Analyzing with SonarQube Runner、，Analyzing with SonarQube Ant Task 和 Analyzing with Maven 方法。这三种方法的主要区别是 , 它们分别适用于不同架构和组织形式的项目，并且其相应的配置文件的写法也不尽相同。

下面将分别介绍这三种方式。

***Analyzing with SonarQube Runner***

通常我们会推荐使用 SonarQube Runner 这种方式，因为该方法适用于所有不同架构的项目，包括没有使用任何源代码管理工具的项目形式，以及使用各种不同代码管理工具（SVN、Git、ClearCase 等）和编译工具（ant，maven）的项目形式，它都能够适用。

# SonarQube3.6 代码质量分析结果解析

当项目代码被成功分析完成后，SonarQube Server 会将分析的结果存储在相应的数据库中，并以报告的形式将分析结果在 webserver 中展示给用户。

下面将以一个真实项目代码的分析结果为例，对其各项分析指标进行简要介绍。

##### 图 13.分析结果示例-01

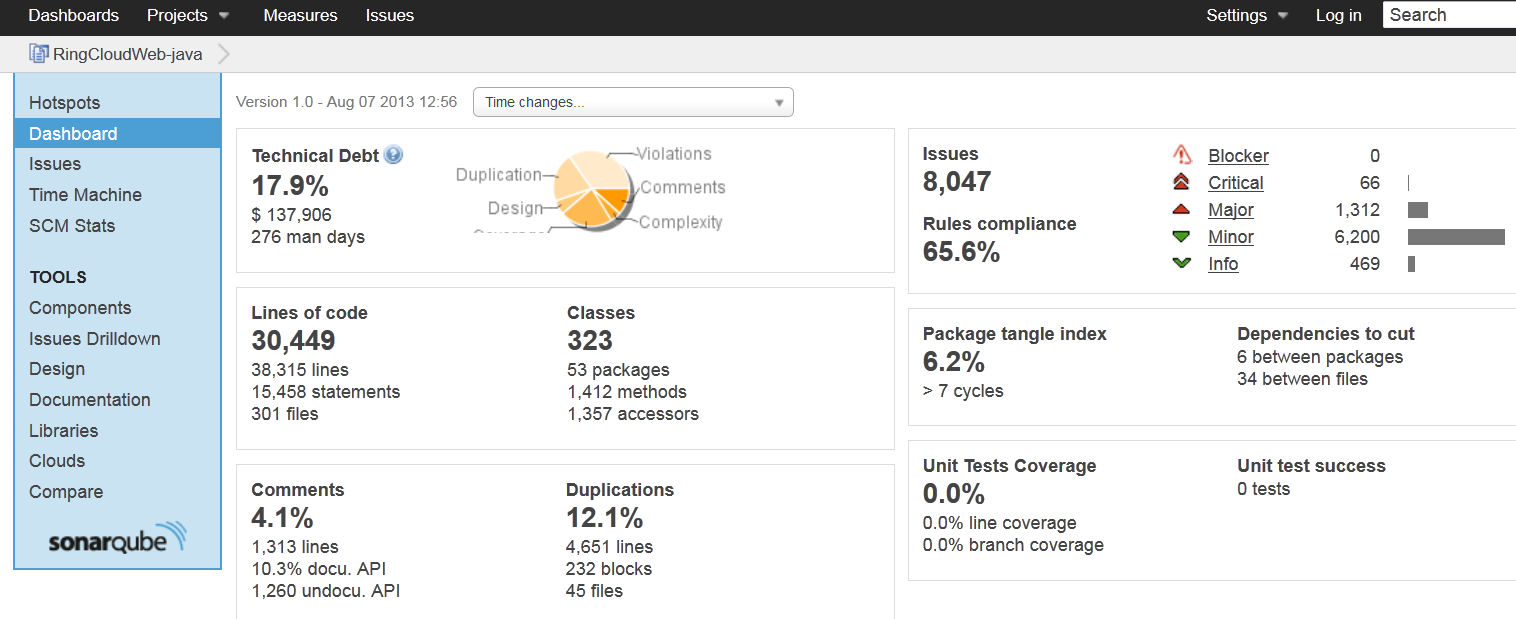


图 13 为 SonarQube3.6 的 Project Dashboard，它是查看任意项目分析结果的入口，展示了该项目各项指标总的分析数据结果，其中包括项目的重复率，注释的比例，单元测试覆盖率以及 Issue 的比例等。通过 Dashboard 就可以对项目总体的质量情况有一个全局性的认识。

选择图 13 中的各项分析指标，就可以具体的去查看该项目在这一指标上的详细情况。

下面以重复率和 Issue 为例，对其分析结果进行解析。

1. 选择图 13 中的重复率指标，将会得到图 14 所示的关于重复代码的详细分析结果：

##### 图 14. 分析结果示例-02

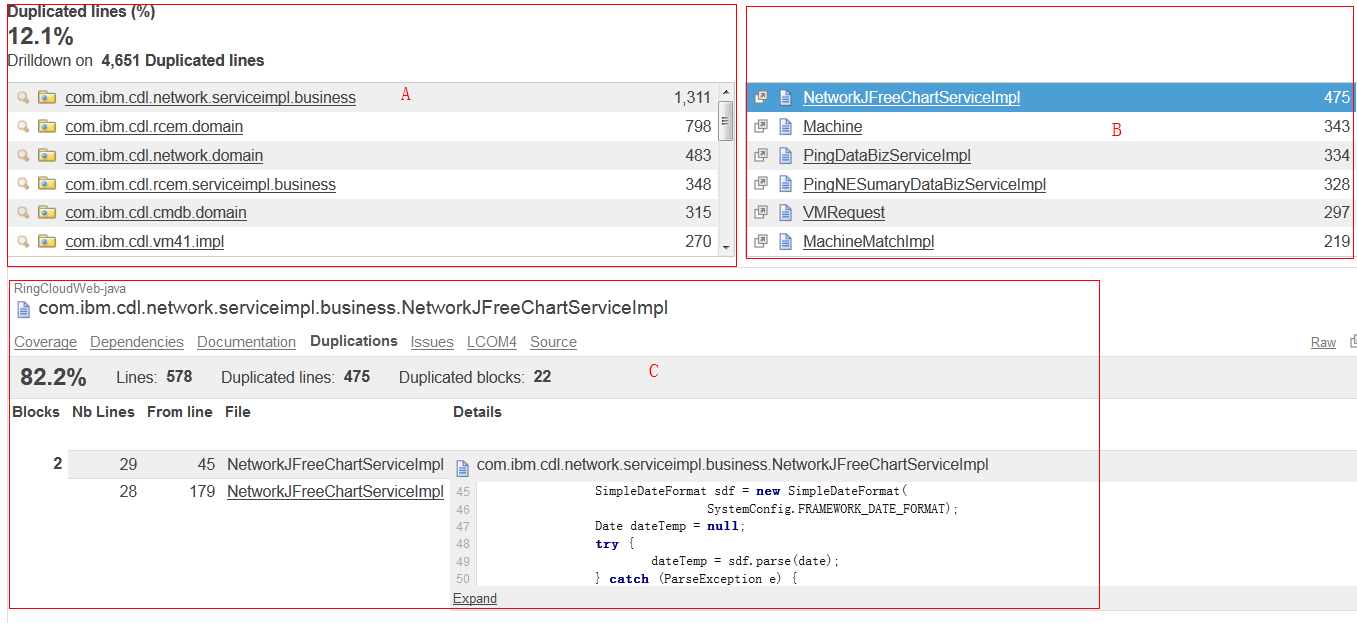


图 14 中的区域 A 展示了整个项目中产生代码重复的所有模块及其重复的代码行数。选择区域 A 中的任一某块，在区域 B 中则将显示该模块下产生代码重复的所有文件；选择区域 B 中的某一文件，则在下方区域 C 中会显示项目中与该文件产生代码重复的所有文件，同时还会显示该文件相应的重复代码的内容。这样的分析结果，可以有效地指导项目组成员进行项目代码优化并降低重复率。

2. 选择图 13 中的 Issues 指标，会得到图 15 所示的关于项目 Issue 的详细分析结果：

##### 图 15. 分析结果示例-03

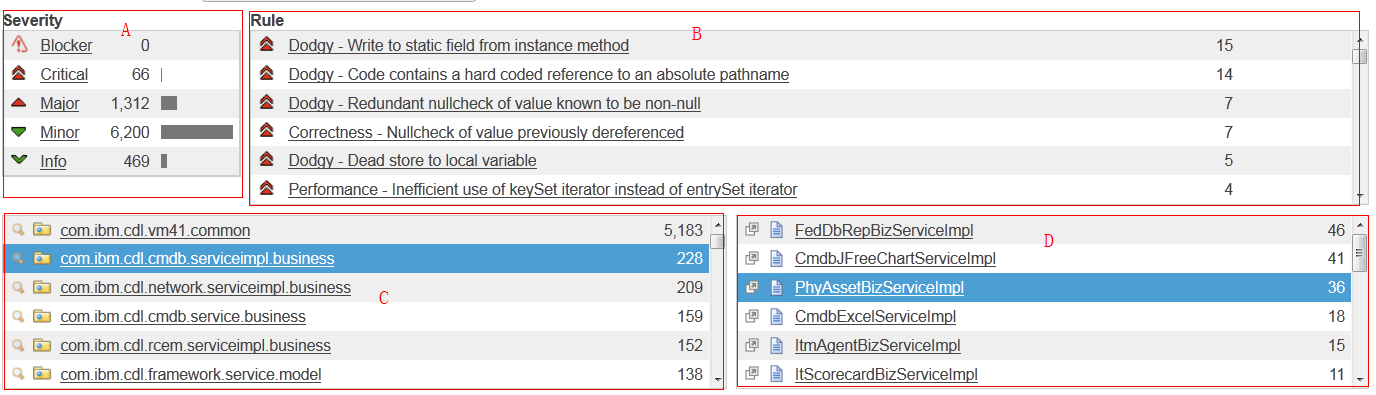


图 15 中的区域 A 展示了整个项目中不同严重程度的 Issues 数量；区域 B 则显示了项目代码具体所违反的各种编码规则以及违反规则的次数；点击区域 B 中相应的规则，在区域 C 中就会显示该项目中违反相应规则的所有模块及其违反规则的次数；选择区域 C 中的某一模块，在区域 D 中就会显示该模块内违反规则的所有文件；如果要查看代码文件到底如何违反了某一规则，只需在区域 D 中点击相应的文件，则会显示如图 16 所示的详细情况。

在图 16 中，可以看到违反相应规则的具体代码，用红色标出；点击所违反的具体规则，SonarQube 则会解释代码违反规则的具体原因，并给出相应的解决方案以供参考，这对于项目组成员定位问题和解决问题都有很大的帮助。

##### 图 16. 分析结果示例-04



其实 SonarQube 还提供了很多其他方面的质量维度分析，如复杂度，测试覆盖率等，这里就不再讨论其具体的参数细节了。本部分旨在从整体上对代码的质量分析结果进行概括，并介绍如何查看相应的质量分析结果，而关于具体如何利用各个分析参数的值来维护、改善代码质量，会在后续系列中与大家分享。

# SonarQube3.6 实际使用中遇到的问题及解决方案

在使用 SonarQube3.6 分析代码质量时，通常会遇到以下三个问题，在实际使用中要注意：

1. find-bug.jar 版本问题

由于在 sonarQube3.3.1 之前， 其所默认使用的 find-bug 的版本为 2.0.0，该版本会导致代码分析时出现时间字符串的格式解析错误，因此在安装 sonarQube3.6 后，务必要确保其所安装的插件 find-bug.jar 的版本为 2.0.1。

2. 认证授权错误

由于在 sonarqube3.4 之前的版本中，连接 sonar server 是不需要 authorized，而在 3.6 版本中则需要 authorized，授权问题可以通过两种方法解决：

* 在 sonar.properties 和 sonar-runner.properties 中同时指定相同的用户名和密码。（sonar.login= sonar.password=）
* 在 sonar.properties 和 sonar-runner.Properties 分别强制设定不需要进行访问授权，即 sonar.forceAuthentication=false

3. 源代码的编码格式错误

由于 find-bug.jar 2.0.1 和 sonarQube3.6 server 的默认编码格式为 GBK，因此必须认真地处理源代码的编码格式（即 sonar.sourceEncoding=UTF-8/GBK/......），避免因编码格式不兼容而导致的问题。