

|  |
| --- |
|  |

**Clover, Cobertura, Jacoco, Istanbul Guidelines**

# Table of Contents

[Table of Contents 2](#_Toc423524654)

[1. Introduction 3](#_Toc423524655)

[2. GUIDELINES 4](#_Toc423524656)

[3. DIFFERENCES 4](#_Toc423524656)

# 

# Introduction

This document tells about the conventions in application development coverage as best practices.

Code coverage helps in understanding the findings from test case writings to encourage test driven development and to avoid rework. Since the tests are written first before implementation of a functionality, less number of bugs are found during testing.

Some of the tools to do code coverage are Emma, Cobertura, JaCoCo and Clover for Java and PHP projects while Istanbul is for java scripts. Basic introduction about each tool is as given below.

**1) Clover:**

Clover is a Java Code Coverage Analysis application bought and further developed by Atlassian. It is a commercial product freely available to open source projects and non-profit institutions.

Clover is using a source code instrumentation which has its advantages (like an ability to collect code metrics) and disadvantages (re-compilation of sources is necessary).

2) Cobertura:

Cobertura is free Java tool that calculate percentage of code accessed by tests. It is based on jcoverage.

3) Jacoco:

Jacoco is free code coverage library for java, which has been created by Java EclEmma team. Jacoco is mainly designed for Integration. Since it is based on bytecodes ,therefore can work also without source files. Jacoco has support for different JVM languages.

4) Istanbul:

Istanbul is code coverage tool written in JS to perform code coverage task for java scripts. It can be run through command line tool and grunt(build tool).It can be used as a middleware which can be tested on the browser.

# GUIDELINES

1. Always ensure the code quality by writing unit tests to cover at least 80 to 90 percent of the code.
2. Integrate coverage through plugins from build tools to generate HTML and XML reports.
3. Integrate coverage through CI plugins, like Jenkins Cobertura plugin, for showing up the reports through its dashboard.
4. Fail the build if the prescribed coverage percentage is not achieved.
5. Integrate it with static analysis tool. For example a Cobertura coverage.xml file can be input to a Sonarqube Dashboard.

# Differences

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Points of Difference** | **Clover** | **Cobertura** | **Jacoco** | **Istanbul** |
| **1. Language Support** | Java, Groovy, PHP, etc. | Java, Groovy | Java, Groovy | Java script |
| **2. Coverage Metrics** | Methods, Statements, Per-test coverage | Line, Branch, Per-test coverage with sonar integration | Method, Line, Branch, Instruction, Per-test coverage for JMX, Sonar | Statement, branch, function |
| **3. Available metrics** | 20 + , also custom one | Cyclomatic complexity | Cyclomatic Complexity | Cyclomatic Complexity |
| **4. Report Types support** | HTML,PDF,JSON ,text and XML | HTML and text | HTML | HTML,PDF,JSON ,text and XML |
| **5. Supported test frameworks** | Junit, TestNG, Spock | Mocha,Jasmine | Junit, TestNG | Mocha,Jasmine,Karma |
| **6. Build Tool Integration** | Command line, Maven, ANT, Graddle | Command line, Maven, ANT, Graddle | Command line, Maven, ANT. | Grunt,Gulp |
| **7. CI Server Integration** | Jenkins, Bamboo | Jenkins | Jenkins | Jenkins |
| **8. Sonar Integration** | Yes | Yes | Yes | Yes |