Code Coverage

1. Introduction
2. Tools
- 2.1 JCov
  - 2.1.1 JCov for Ant
- 2.2 JaCoCo
  - 2.2.1 JaCoCo for Eclipse
- 2.3 Clover
  - 2.3.1 Clover for Eclipse
- 2.4 EMMA
  - 2.4.1 EMMA for Ant
- 2.5 Cobertura
  - 2.5.1 Ant+JUnit+Cobertura
  - 2.5.2 Command

1. Introduction

> 代码覆盖（**code coverage**）：为了全面地覆盖测试，必须测试程序的状态以及程序流程，设法进入和退出每一个模块，执行每一行代码，进入软件每一条逻辑和决策分支。——[Software Testing]
>
> **Code coverage** is An information on what source code is exercised in execution.——*Alexandre Iline*

其要求通过完全访问代码以查看运行测试用例时经过了哪些部分。

- 语句覆盖（statement coverage）

  这是代码覆盖最直接的表现形式，进行语句覆盖目的是保证程序中每一条语句至少执行一次。

- 分支覆盖（branch coverage）

  试图覆盖软件中所有的路径称为路径覆盖，路径覆盖中最简单的形式则是分支覆盖。但是语句100%覆盖不等于分支100%

- 方法覆盖（method coverage）

  软件中方法被测试执行的情况

2. Tools

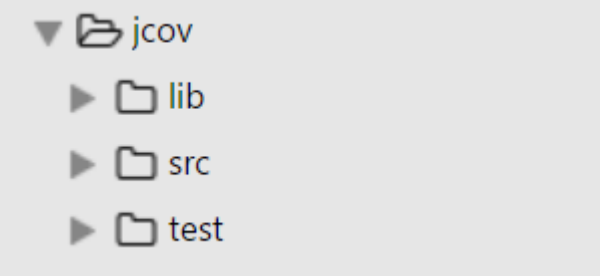
Java代码覆盖工具有两类：第一种添加语句到源码并要求重新编译；第二种是在执行中或执行前修改（instrument）字节码。

2.1 JCov

JCov是Java开始之初由Sun JDk（更早之前是Oracle JDK）开发和使用的。从1.1版本开始，Jcov就可以对Java代码覆盖进行测试和报告。2014年开始作为OpenJDK codetools项目的一部分开始开放源码。其主页https://wiki.openjdk.java.net/display/CodeTools/jcov。

2.1.1 JCov for Ant

项目结构：




```
lib folder：相关的jar
src folder：业务代码
test folder：测试代码
```

主要在Command line上运行ant（当然，Eclipse也是可以的），build.xml配置步骤如下（大多工具步骤也是如此）：

- 编译业务代码以及测试代码

```
 <javac encoding="iso-8859-1" debug="true" target="1.5" source="1.5"
                srcdir="src"
                destdir="classes">
        </javac>
```

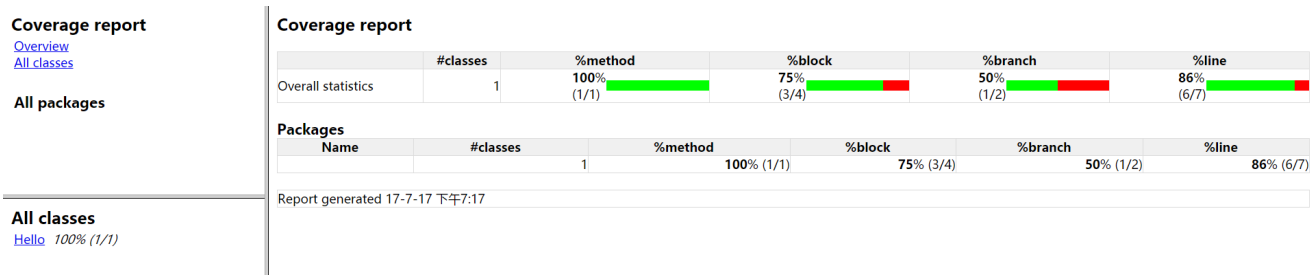
- 在业务代码class文件中插入instrumentation。

```
<instrument productdir="classes" destdir="instr_classes" outtemplate="template.xml">
            <!-- 需要的jar-->
            <implantTo path="." implantRT="lib/jcov.network.saver.jar"/>
        </instrument>
```

- 打开Jcov的grabber，再次运行测试代码的class文件（以此收集覆盖信息），关闭grabber，最后打印报告（指定地址以及各式）

```
    <!--start grabber-->
        <grabber output="result.xml" template="template.xml"/>
        <!--运行测试用例字节码（此时运行，测试用例调用的是被插入instrutation的业务class文件）-->
        <java classname="HelloTest" fork="true" failonerror="true">
            <classpath>
                <pathelement location="lib/jcov.network.saver.jar"/>
                <pathelement location="test_classes"/>
                <pathelement location="instr_classes"/>
            </classpath>
        </java>
        <!--stop grabber-->
        <!-- 生成报告-->
        <grabber-manager command="kill"/>
        <report output="report" jcovfile="result.xml"/>
```

报告：



2.2 JaCoCo

JaCoco是开放源码的工具包，作为EMMA的替代品被开发出来（同一个开发团队）。项目主页http://www.eclemma.org/jacoco/。它可以集成到ANT、Maven中，也可以使用Java Agent技术监控Java程序，并提供了Eclipse插件EclEmama。以下工具可以使用或者包含了Jacoco：

- EclEmma Eclipse Code Coverage Plugin
- Jenkins JaCoCo Plugin
- SonarQube JaCoCo plugin
- Netbeans JaCoCo support
- IntelliJ IDEA since v11
- Gradle JaCoCo Plugin
- Visual Studio Team Services
- TeamCity

2.2.1 JaCoCo for Eclipse

使用EcliEmama插件测试，步骤指导：http://www.eclemma.org/installation.html

- 下载EclEmma，在Eclipse软件市场搜索EclEmma，点击安装
- 出现  表示安装完成
- 点击上述按钮进行测试

  该插件支持的启动类型：

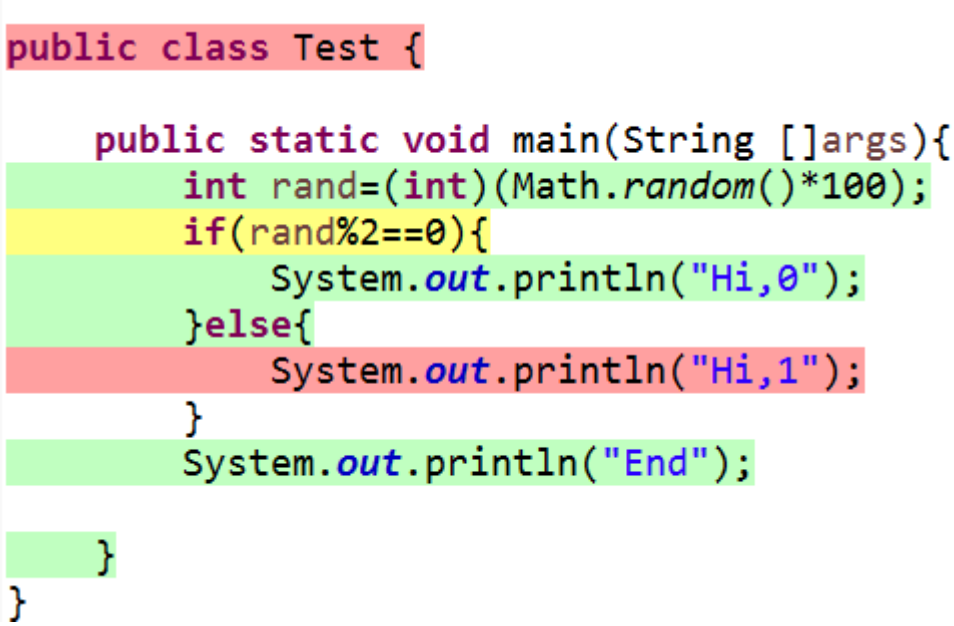
  - Local Java application
  - Eclipse/RCP application
  - Equinox OSGi framework
  - JUnit test
  - TestNG test
  - JUnit plug-in test
  - JUnit RAP test
  - SWTBot test
  - Scala application

1. **Local Java Application**

测试demo：

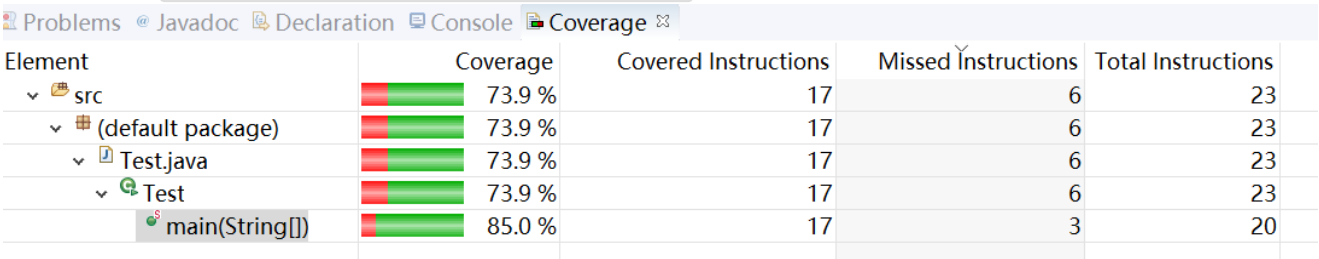
```java
public class Test {
    public static void main(String []args){
        int rand=(int)(Math.random()*100);
        if(rand%2==0){
            System.out.println("Hi,0");
        }else{
            System.out.println("Hi,1");
        }
        System.out.println("End");
    }
}
```

测试结果：



- 红色：测试未覆盖
- 绿色：测试已覆盖
- 黄色：测试部分覆盖（if、switch）

查看测试率：`Window->Show View->Other->Java->Coverage`



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
src	73.9 %	17	6	23
(default package)	73.9 %	17	6	23
Test.java	73.9 %	17	6	23
Test	73.9 %	17	6	23
main(String[])	85.0 %	17	3	20

2. **JUnit**

下面使用**JUnit Test**覆盖测试

测试类：

```
import static org.junit.Assert.*;

import org.junit.Test;

public class TestTest {

    @Test
    public void test() {
        JUnitTest t=new JUnitTest();
    }

}
import static org.junit.Assert.*;

import org.junit.Test;

public class TestTest {

    @Test
    public void test() {
        JUnitTest t=new JUnitTest();
    }

}
```

- 测试结果查看（Window显示，步骤如上）：

TestTest (2017-7-13 15:56:47)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Jacoco	49.1 %	26	27	53
src	49.1 %	26	27	53
(default package)	49.1 %	26	27	53
Test.java	0.0 %	0	23	23
JUnitTest.java	81.8 %	18	4	22
TestTest.java	100.0 %	8	0	8

- 测试报告的导出:结果处右键，选择`Export Session`，选择报告类型以及目标地址

TestTest (2017-7-13 15:56:47)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Jacoco		49%		25%	4	7	10	19	2	5	1	3
Total	27 of 53	49%	3 of 4	25%	4	7	10	19	2	5	1	3

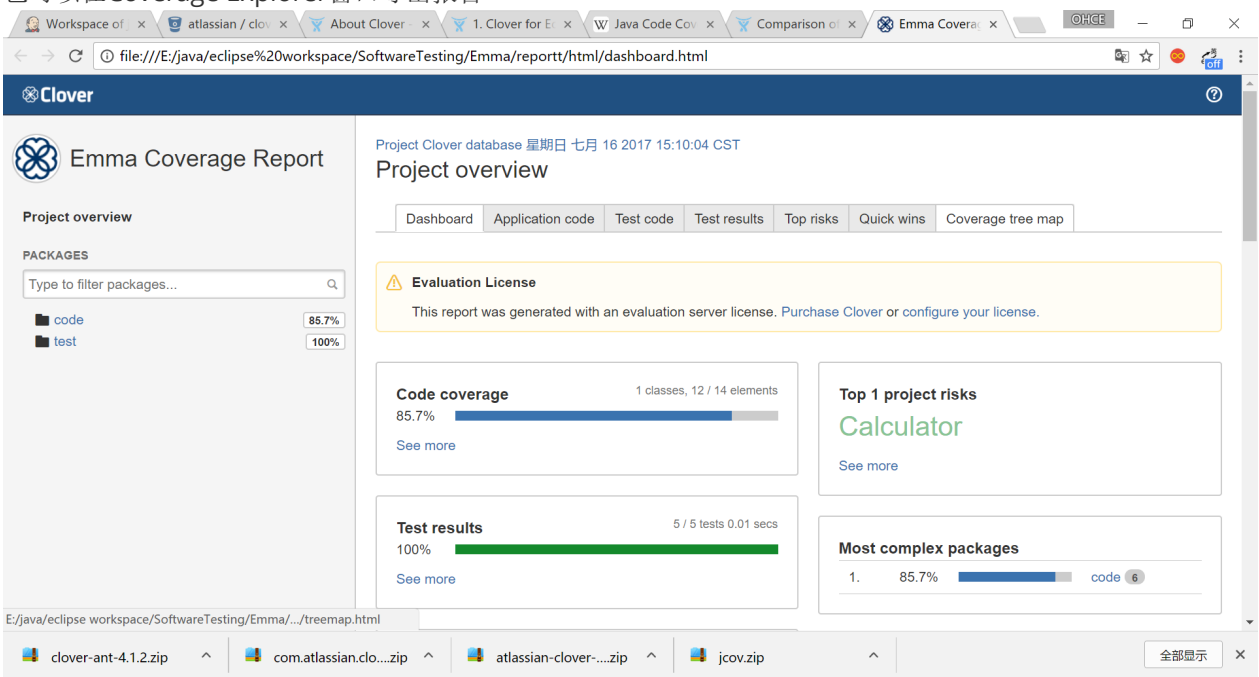
2.3 Clover

Clover最早由Atlassian公司开发，是商业产品，但在2017年开源。下载可试用30天，支持环境：

- Ant
- Eclipse
- Maven
- IDEA
- Grails
- Bamboo

2.3.1 Clover for Eclipse

- 安装插件`"http://update.atlassian.com/eclipse/clover"`
- 项目右击，选择`"Clover > Enable on this Project"`
- 然后运行，出现以下窗口，可查看代码覆盖情况
    - Coverage Explorer
    - Test Run Explorer
    - Clover Dashboard
    - Test Contributions
- 也可以在Coverage Explorer窗口导出报告



2.4 EMMA

EMMA是开源工具，但很久之前便停更了，上一个稳定版本在2005年，Jacoco则是它的进化版。它通过对编译后的Java字节码进行插装，之后在执行过程中收集覆盖率信息，并通过多种报表格式对覆盖率结果进行展示。项目主页http://emma.sourceforge.net/。EMMA的工具具体有：

- Intellij Idea Plugin
- SonarQube EMMA Plugin
- Google CodePro AnalytiX
- Jenkins Emma Plugin

2.4.1 EMMA for Ant

它的步骤与JCov类似，区别在于它没有将测试代码与业务代码分开装入instrumentati，而是整体操作，在最后的运行字节码时指定它的启始类（即测试类）即可。

- 编译（同JCov）
- 插入instrumentation

```
<instr instrpathref="run.classpath"
           destdir="${out.instr.dir}"
           metadatafile="${coverage.dir}/metadata.emma"
           merge="true"
      >
```

- 运行字节码以及导出报告

这里插入instrumentation以及导出报告时，包含在标签下，所以在命令行运行ant时不再是以往的`ant`命令或者`ant -f name.xml`，而是运行`ant emma run`。
报告：

EMMA Coverage Report (generated Mon Jul 17 21:23:00 CST 2017)

[all classes]

OVERALL COVERAGE SUMMARY

name	class, %	method, %	block, %	line, %
all classes	60% (3/5)	70% (7/10)	77% (120/156)	68% (27/40)

OVERALL STATS SUMMARY

total packages:	2
total executable files:	5
total classes:	5
total methods:	10
total executable lines:	40

COVERAGE BREAKDOWN BY PACKAGE

name	class, %	method, %	block, %	line, %
default package	33% (1/3)	50% (3/6)	65% (56/86)	52% (11/21)
search	100% (2/2)	100% (4/4)	91% (64/70)	84% (16/19)

[all classes]

EMMA 2.0.5312 (C) Vladimir Roubtsov

on

2.5 Cobertura

Cobertura是开源的工具，可以与Junit集成。项目主页http://cobertura.github.io/cobertura/。
可以通过以下方式执行：

- Ant
- Command line(Shell,CMD)
- Gradle
- Maven

2.5.1 Ant+JUnit+Cobertura

弄了两天有个error一直没解决，先写下步骤
Ant是一个基于Java的自动化脚本引擎，Eclipse中提供了对它的支持，因此不用再次安装。
关于使用ant进行cobertura测试，主要是脚本语言的编写，分为以下几个部分：

- 向已经编译好的class文件中添加instrumentation
- 执行测试用例，此时cobertura会统计代码的执行情况，也就是正常的Junit测试，不同的就是使用刚刚被注入instrumentation的class文件。
- 生成测试报告

2.5.2 Command

使用命令行进行测试需要安装ant，
进入官网 http://ant.apache.org/下载后解压，之后配置环境变量：

```
ANT_HOME：`E:/ apache-ant`
path：`E:/apache-ant/bin`
classpath：`E:/apache-ant/lib`
```

进入cmd输入`ant`验证是否配置成功
下面进行测试报告的生成：

1. 将`src`与`lib`文件夹放入`bin`中，复制出来（我放在了C根目录），同时将下载的`cobertura`文件解压也放到`bin`文件夹中
2. 执行`c:\bin\cobertura-2.1.1\cobertura-instrument.bat --destination instrumented  code`
   此时`bin`文件夹中会出现`instrumented`文件夹和`cobertura.ser`文件
   这里主要是在编译好的class文件中添加日志文件，并放入instrumented文件夹中
3. 进行代码测试的工作，

```
java -cp lib/cobertura.jar;lib/hamcrest-core-1.3.jar;lib/junit-4.12.jar;lib/slf4j-api-
1.7.25.jar;instrumented;.;-Dnet.sourceforge.cobertura.datafile=cobertura.ser
org.junit.runner.JUnitCore code.CalculatorTest
```

4. 生成测试报告

```
c:\bin\cobertura-2.1.1\cobertura-report.bat --format html -datafile=cobertura.ser --destination
report src
```

下面会出现一个report文件夹，打开查看该报告