

0.前置 java 和 maven 环境

配置 maven 参照

https://blog.csdn.net/qq_32588349/article/details/51461182

https://blog.csdn.net/qq_35868412/article/details/79499946

错误 1： mvn 命令有错误提示，见图 0-1。

The JAVA_HOME environment variable is not defined correctly
This environment variable is needed to run this program
NB: JAVA_HOME should point to a JDK not a JRE

图 0-1

解决： JAVA_HOME 之前有多个版本的。只保留一个版本

参照： https://blog.csdn.net/qq_36238595/article/details/79451032

错误 2： 命令行中 java 可用，javac 不可用

解决： 下移第一个变量，见图 0-2。

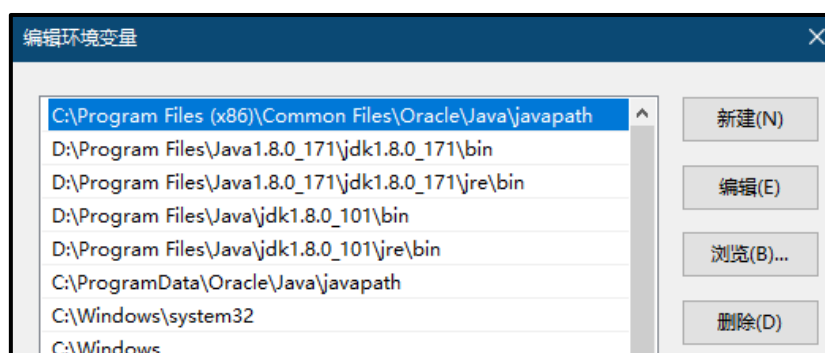


图 0-2

参照： <https://jingyan.baidu.com/article/4ae03de3e736d43eff9e6b94.html>

最终配置成功， 见图 0-3。

```
C:\Users\54251>javac -version
javac 1.8.0_171

C:\Users\54251>mvn -version
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5fffb20918da4f719f3; 2018-10-25T02:41:47+08:00)
Maven home: D:\Program Files\Maven\apache-maven-3.6.0\bin\..
Java version: 1.8.0_171, vendor: Oracle Corporation, runtime: D:\Program Files\Java1.8.0_171\jdk1.8.0_171\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\54251>
```

图 0-3

错误 3： 运行项目时 jar 包解析失败，如图 0-4。

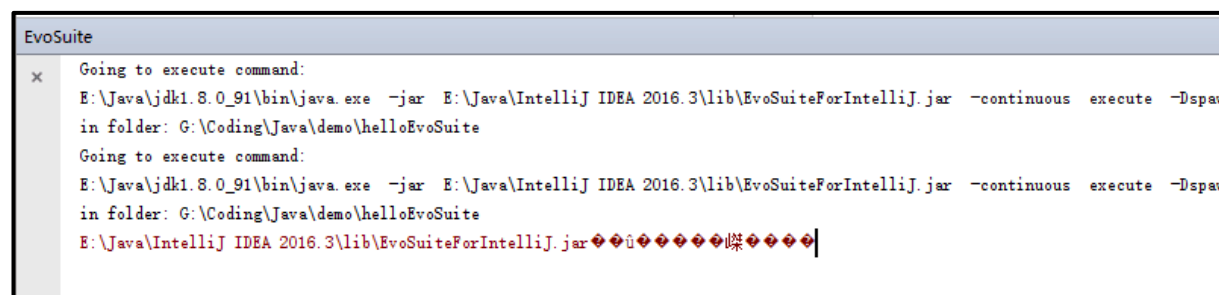


图 0-4

解决： 更换为最新版本的 jar 包。

1. 生成测试

下载示例文档，测试一个简单的 Stack 数据结构实现，见图 1-1。文档包括：**pom.xml**（Maven 构建文件）、**Stack.java**（被测试的类，图 1-2）、以及 **StackTest.java**（图 1-3）。(下表粗体为文件，斜体为文件夹)

```
--src
--main
--java
--tutorial
--Stack.java
--test
--java
--tutorial
--StackTest.java
--pom.xml
并添加
--evosuite-1.0.6.jar
```

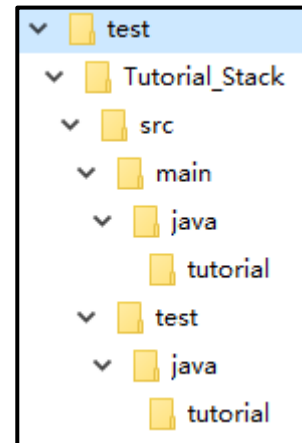


图 1-1

```
Stack.java x
1 package tutorial;
2
3 import java.util.EmptyStackException;
4
5 public class Stack<T> {
6     private int capacity = 10;
7     private int pointer = 0;
8     private T[] objects = (T[]) new Object[capacity];
9
10    public void push(T o) {
11        if(pointer >= capacity)
12            throw new RuntimeException("Stack exceeded capacity!");
13        objects[pointer++] = o;
14    }
15    public T pop() {
16        if(pointer <= 0)
17            throw new EmptyStackException();
18        return objects[--pointer];
19    }
20    public boolean isEmpty() {
21        return pointer <= 0;
22    }
23 }
```

图 1-2

```
Stack.java StackTest.java x
1 package tutorial;
2
3 import org.junit.Test;
4 import org.junit.Assert;
5
6 public class StackTest {
7
8     @Test
9     public void test() {
10         Stack<Object> stack = new Stack<Object>();
11         stack.push(new Object());
12         Assert.assertFalse(stack.isEmpty());
13     }
14 }
15
```

图 1-3

首先调用 maven 来编译项目，并构建成功，见图 1-4。

```
e:\test\Tutorial_Stack>mvn compile
[INFO] Scanning for projects...
[INFO] -----< org.evosuite.tutorial:Tutorial_Stack >-----
[INFO] Building Tutorial_Stack 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Tutorial_Stack ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory e:\test\Tutorial_Stack\src\main\resources
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Tutorial_Stack ---
[INFO] BUILD SUCCESS
[INFO] Total time: 1.734 s
```

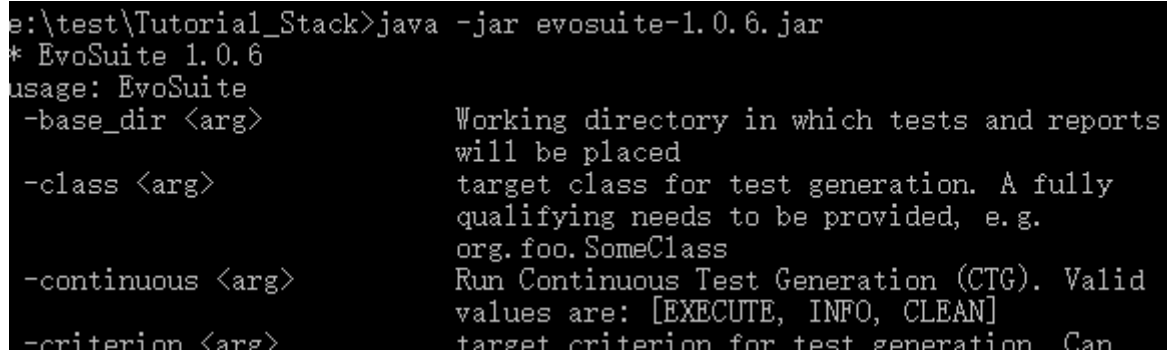
图 1-4

此时文件增加了以下黑色部分。

Tutorial_Stack

```
--src
--main
--java
--tutorial
--Stack.java
--test
--java
--tutorial
--StackTest.java
--pom.xml
--evosuite-1.0.6.jar
--target
--classes
--tutorial
--Stack.class
--maven-status
--maven-compiler-plugin
--compile
--default-compile
--createdFiles.lst
--inputFiles.lst
```

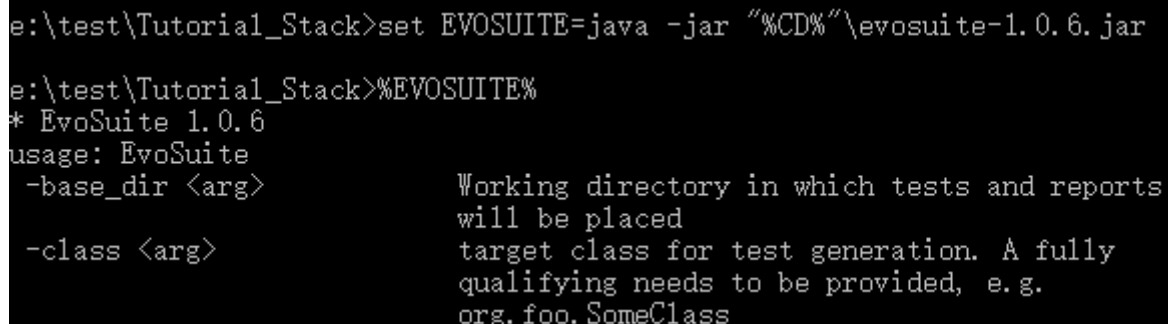
尝试执行 jar 文件，可以看到所有的命令行选项，如图 1-5。



```
e:\test\Tutorial_Stack>java -jar evosuite-1.0.6.jar
* EvoSuite 1.0.6
usage: EvoSuite
  -base_dir <arg>           Working directory in which tests and reports
                             will be placed
  -class <arg>              target class for test generation. A fully
                             qualifying needs to be provided, e.g.
                             org.foo.SomeClass
  -continuous <arg>        Run Continuous Test Generation (CTG). Valid
                             values are: [EXECUTE, INFO, CLEAN]
  -criterion <arg>         target criterion for test generation. Can
```

图 1-5

创建一个指向 EvoSuite 的环境变量，之后输入%EVOSUITE%即可以直接调用，如图 1-6。



```
e:\test\Tutorial_Stack>set EVOSUITE=java -jar "%CD%"\evosuite-1.0.6.jar

e:\test\Tutorial_Stack>%EVOSUITE%
* EvoSuite 1.0.6
usage: EvoSuite
  -base_dir <arg>           Working directory in which tests and reports
                             will be placed
  -class <arg>              target class for test generation. A fully
                             qualifying needs to be provided, e.g.
                             org.foo.SomeClass
```

图 1-6

接下来进行生成测试，Evosuite 需要知道被测试的类，类的路径。输入

%EVOSUITE% -class tutorial.Stack -projectCP target/classes

运行过程见图 1-7。

```

e:\test\Tutorial_Stack>%EVOSUITE% -class tutorial.Stack -projectCP target/classes
S
* EvoSuite 1.0.6
* Going to generate test cases for class: tutorial.Stack
* Starting client
* Connecting to master process on port 12632
* Analyzing classpath:
  - target/classes
* Finished analyzing classpath
* Generating tests for class tutorial.Stack
* Test criteria:
  - Line Coverage
  - Branch Coverage
  - Exception
  - Mutation testing (weak)

[Progress:>          0%] [Cov:==>
[Progress:>          0%] [Cov:=====>
[Progress:>          0%] [Cov:=====>
[Progress:>          0%] [Cov:=====>
[Progress:>          0%] [Cov:=====>
[Progress:>          0%] [Cov:=====>
[Progress:>          0%] [Cov:=====>
[Progress:>          0%] [Cov:=====>
[Progress:>          0%] [Cov:=====>
[Progress:>          1%] [Cov:==>
[Progress:>          1%] [Cov:=====>

* Generated 6 tests with total length 35
* Resulting test suite's coverage: 99% (average coverage for all fitness functions)
* Generating assertions
* Resulting test suite's mutation score: 67%
* Compiling and checking tests
* Writing JUnit test case 'Stack_ESTest' to evosuite-tests
* Done!

* Computation finished

```

图 1-7

结束后的文件结构如下表。

```

Tutorial_Stack
--src
  --main
    --java
      --tutorial
        --Stack.java
  --test
    --java
      --tutorial
        --StackTest.java
--pom.xml
--evosuite-1.0.6.jar
--target
  --classes
    --tutorial
      --Stack.class
  --maven-status
    --maven-compiler-plugin
      --compile
      --default-compile
      --createdFiles.lst
      --inputFiles.lst

```

--evosuite-report
-- statistics.csv
--evosuite-tests
--tutorial
--Stack_ESTest.java
--Stack_ESTest_scaffolding.java

可以看到，在 Tutorial_Stack\evosuite-tests\tutorial 生成了 Stack_ESTest.java 和 Stack_ESTest_scaffolding.java 两个文件。另外，statistics.csv 文件内容如图 1-8。

A	B	C	D	E
TARGET_CLASS	criterion	Coverage	Total_Goals	Covered_Goals
tutorial.Stack	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.990625	80	77

图 1-8

另一种方法：使用 IntelliJ IDEA 插件。
参考：<http://www.evosuite.org/documentation/intellij-idea-plugin/>
安装完毕插件，见图 1-9。

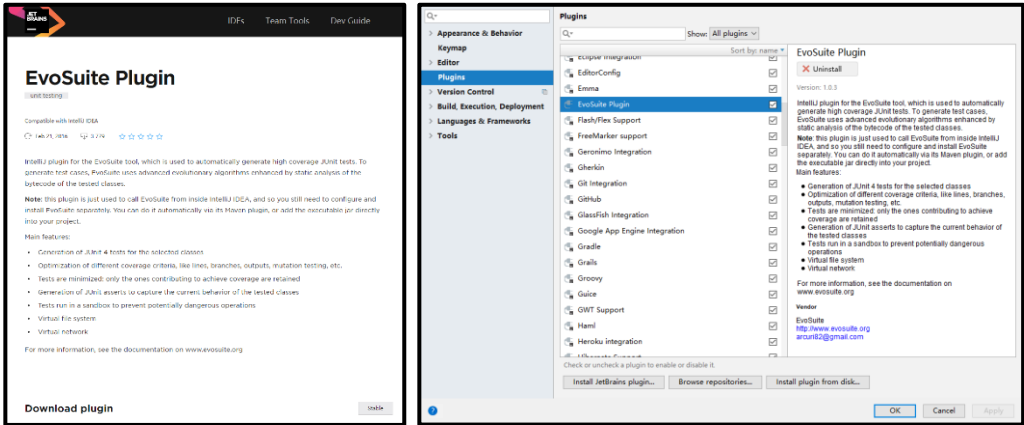


图 1-9

打开程序，程序结构与内容见图 1-10。

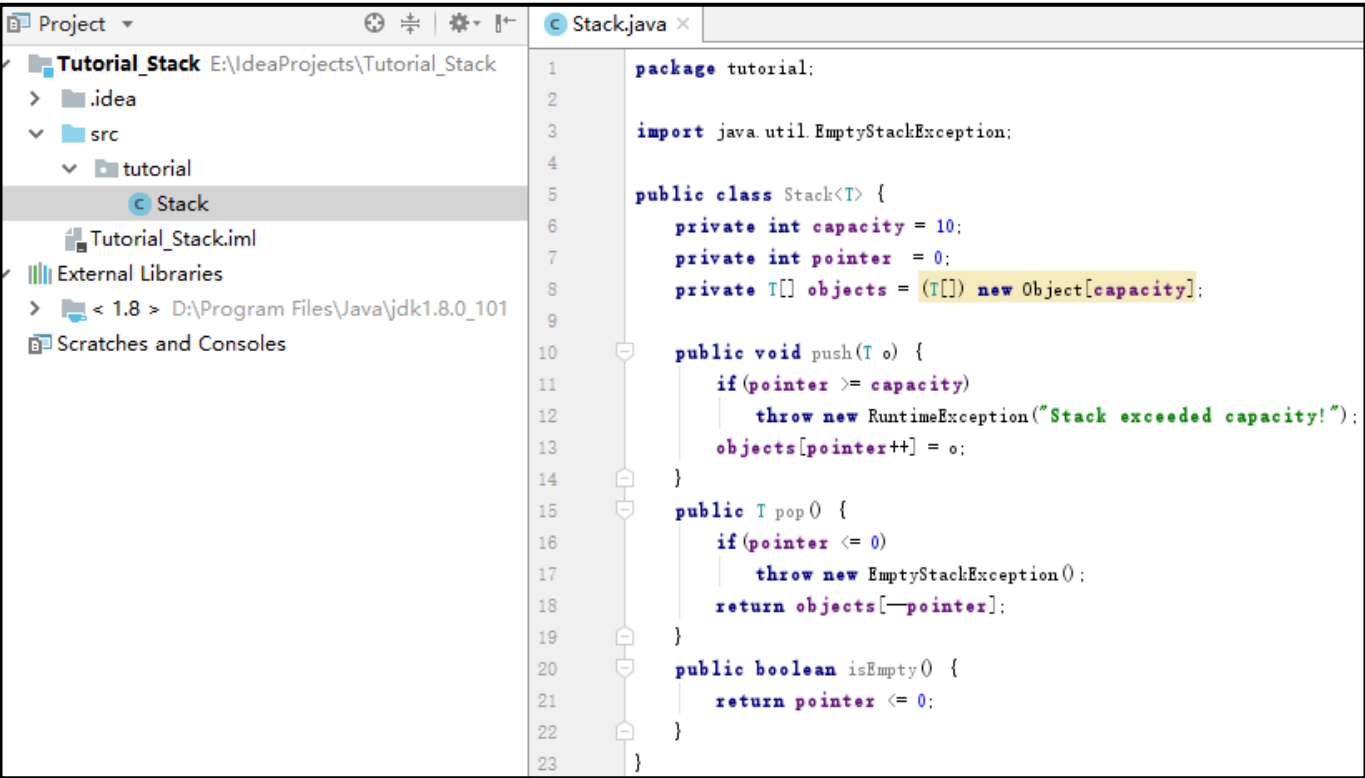


图 1-10

右键点击并设置好文件路径并按下回车，等待生成结束。同样可以看到生成的 Stack_ESTest.java 和 Stack_ESTest_scaffolding.java 两个文件。具体过程见图 1-11。

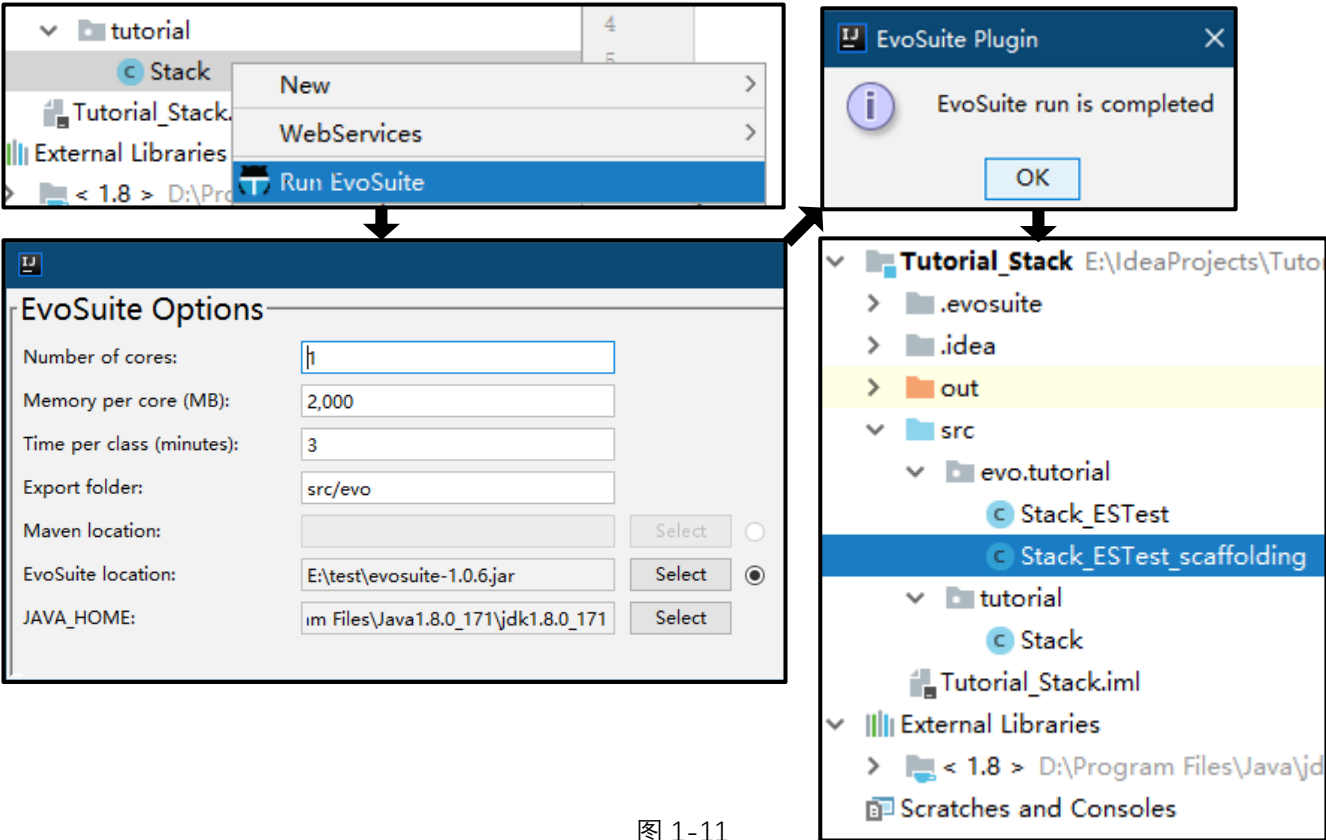


图 1-11

2. 执行测试

由于需要 Junit 来执行测试，需要获得 hamcrest-core-1.3.jar 和 junit-4.12.jar 包。使用 Maven，输入命令 mvn dependency:copy-dependencies，显示下载成功的提示，见图 2-1。

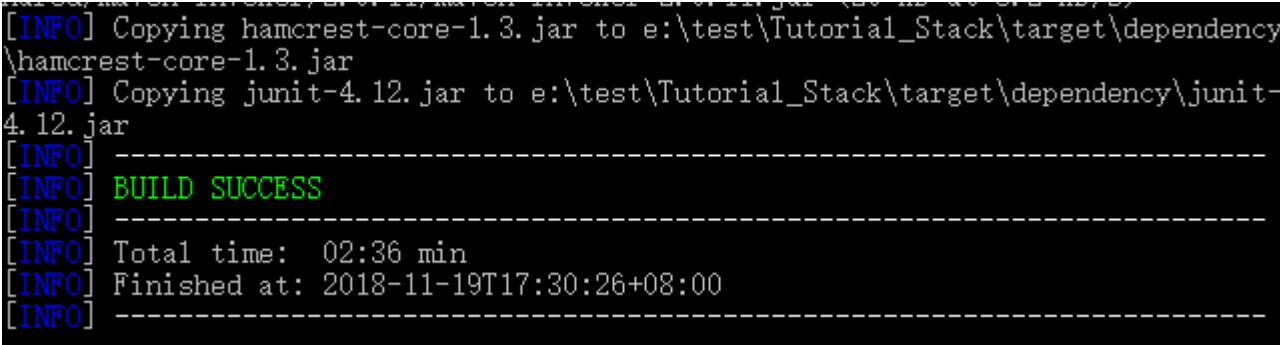


图 2-1

得到两个 jar 文件，在表中以下目录。

Tutorial_Stack

```
--target
--classes      (tutorial.Stack 类所需的根目录)
...
--dependency
--hamcrest-core-1.3.jar
--junit-4.12.jar
添加 CLASSPATH 环境变量，输入
set CLASSPATH=target/classes;evosuite-standalone-runtime-1.0.6.jar;evosuite-tests;target/dependency/junit-4.12.jar;target/dependency/hamcrest-core-1.3.jar
注意 evosuite-standalone-runtime-1.0.6.jar 文件需要放在对应的位置（根目录）。
错误 4：环境变量设置失败。
```

解决：Windows 上的路径分隔符是分号，而在类 Unix 系统上是冒号。

简单进行编译测试，如图 2-2 并生成两个 class 文件，在表中一下目录。

```
E:\test\Tutorial_Stack>javac evosuite-tests/tutorial/*.java
```

图 2-2

Tutorial_Stack

--src

.....

--evosuite-tests

--tutorial

--Stack_ESTest.java

--Stack_ESTest_scaffolding.java

--Stack_ESTest.class

--Stack_ESTest_scaffolding.class

运行测试，并得到相应输出，见图 2-3。

```
E:\test\Tutorial_Stack>java org.junit.runner.JUnitCore tutorial.Stack_ESTest
JUnit version 4.12
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
.....
Time: 0.609
OK (6 tests)
```

图 2-3

至此，生成并执行了 EvoSuite 的测试文件。

回看 Stack.java 程序和生成的 Stack_ESTest.java 程序，如图 2-4 和图 2-5。

```
Stack.java x
1  package tutorial;
2
3  import java.util.EmptyStackException;
4
5  public class Stack<T> {
6      private int capacity = 10;
7      private int pointer = 0;
8      private T[] objects = (T[]) new Object[capacity];
9
10     public void push(T o) {
11         if(pointer >= capacity)
12             throw new RuntimeException("Stack exceeded capacity!");
13         objects[pointer++] = o;
14     }
15     public T pop() {
16         if(pointer <= 0)
17             throw new EmptyStackException();
18         return objects[--pointer];
19     }
20     public boolean isEmpty() {
21         return pointer <= 0;
22     }
23 }
```

图 2-4

EvoSuite 居然可以自动生成这些测试函数。

```

32  @Test(timeout = 4000)
33  public void test1() throws Throwable {
34      Stack<Integer> stack0 = new Stack<Integer>();
35      stack0.push((Integer) null);
36      stack0.pop();
37      stack0.push((Integer) null);
38      assertFalse(stack0.isEmpty());
39
40      stack0.pop();
41      boolean boolean0 = stack0.isEmpty();
42      assertTrue(boolean0);
43  }
44
45  @Test(timeout = 4000)
46  public void test2() throws Throwable {
47      Stack<Integer> stack0 = new Stack<Integer>();
48      assertTrue(stack0.isEmpty());
49
50      stack0.push((Integer) null);
51      boolean boolean0 = stack0.isEmpty();
52      assertFalse(boolean0);
53  }
54

```

图 2-5

为验证是否真的是自动生成的，再写了一个简单的计算程序 Add.java，如图 2-6。

```

Add.java x
1  package Calcu;
2  public class Add {
3      public int add2(int a, int b) {
4          return a + b;
5      }
6      public int add3(int a, int b, int c) {
7          return a + b + c;
8      }
9  }
10

```

图 2-6

在 IntelliJ IDEA 中编写完成后按照上面介绍的另外一种方法生成了 EvoSuite 的两个文件，Add_ESTest.java 和 Add_ESTest_scaffolding.java。

（它的 package 前面要加一个 evo. 不然会报错。）

发现果然也生成了许多测试函数，见图 2-7，很厉害。

此外，EvoSuite 可以进一步设定不同参数并保存相应的设置。如果已经编写好一部分测试函数时，仍然可以使用 EvoSuite 来生成一些没有被覆盖到的测试函数。同时，也支持在多个类上运行 EvoSuite，但如果项目很大的情况下可能会不方便，此时可以将 EvoSuite 集成到 Maven 项目中。

```

18  @Test(timeout = 4000)
19  public void test0() throws Throwable {
20      Add add0 = new Add();
21      int int0 = add0.add3(a: 0, b: 348, (-506));
22      assertEquals((-158), int0);
23  }
24
25  @Test(timeout = 4000)
26  public void test1() throws Throwable {
27      Add add0 = new Add();
28      int int0 = add0.add3(a: 1502, b: 0, c: 0);
29      assertEquals(expected: 1502, int0);
30  }
31
32  @Test(timeout = 4000)
33  public void test2() throws Throwable {
34      Add add0 = new Add();
35      int int0 = add0.add2(a: 0, b: 0);
36      assertEquals(expected: 0, int0);
37  }
38
39  @Test(timeout = 4000)
40  public void test3() throws Throwable {
41      Add add0 = new Add();
42      int int0 = add0.add2((-2080), b: 0);
43      assertEquals((-2080), int0);
44  }

```

图 2-7

3. 集成到 Maven

下载 Maven 示例文档，见图 3-1，目录结构如下表。

Tutorial_Maven

```
--src
--main
--java
--tutorial
--Stack.java
--Node.java
--LinkedList.java
--LinkedListIterator.java
--test
--java
--tutorial
--StackTest.java
--pom.xml
```

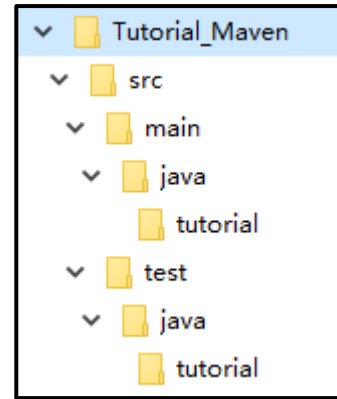


图 3-1

输入命令 `mvn compile` 生成四个 class 文件在根目录 `target/classes` 文件夹内，如图 3-2。

```
E:\test\Tutorial_Maven>mvn compile
[INFO] Scanning for projects...
[INFO] -----< org.evosuite.tutorial:Tutorial_Maven >-----
[INFO] Building Tutorial_Maven 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Tutorial_Maven ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory E:\test\Tutorial_Maven\src\main\resources
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Tutorial_Maven ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 4 source files to E:\test\Tutorial_Maven\target\classes
[WARNING] /E:/test/Tutorial_Maven/src/main/java/tutorial/Node.java: 某些输入文件使用了未经检查或不安全的操作。
[WARNING] /E:/test/Tutorial_Maven/src/main/java/tutorial/Node.java: 有关详细信息，请使用 -Xlint:unchecked 重新编译。
[INFO] BUILD SUCCESS
[INFO] Total time: 1.692 s
```

图 3-2

运行 `mvn test` 进行一些测试，中途过程与最终结果见图 3-3 和图 3-4。

```
[INFO] Scanning for projects...
[INFO] -----< org.evosuite.tutorial:Tutorial_Maven >-----
[INFO] Building Tutorial_Maven 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Tutorial_Maven ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory E:\test\Tutorial_Maven\src\main\resources
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Tutorial_Maven ---
[INFO] Nothing to compile - all classes are up to date
```

图 3-3

```

-----
T E S T S
-----
Running tutorial.StackTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.07 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.084 s
[INFO] Finished at: 2018-11-19T20:50:04+08:00
[INFO] -----

```

图 3-4

现在开始集成 EvoSuite。首先了解 pom.xml 文件，文件内容见图 3-5。

```

pom.xml x
1  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3     <modelVersion>4.0.0</modelVersion>
4
5     <groupId>org.evosuite.tutorial</groupId>
6     <artifactId>Tutorial_Maven</artifactId>
7     <version>1.0-SNAPSHOT</version>
8     <packaging>jar</packaging>
9
10    <name>Tutorial_Maven</name>
11    <url>http://maven.apache.org</url>
12
13    <properties>
14        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15    </properties>
16
17    <dependencies>
18        <dependency>
19            <groupId>junit</groupId>
20            <artifactId>junit</artifactId>
21            <version>4.12</version>
22            <scope>test</scope>
23        </dependency>
24    </dependencies>
25 </project>

```

图 3-5

Maven 通过 groupId, artifactId 和 version 来识别项目，该项目依赖 junit 来执行测试。

添加片段从而使用 EvoSuite Maven 插件，见图 3-6，同时需要添加<pluginRepositories>来指定 URL。

```

25  <build>
26    <plugins>
27      <plugin>
28        <groupId>org.evosuite.plugins</groupId>
29        <artifactId>evosuite-maven-plugin</artifactId>
30        <version>1.0.6</version>
31      </plugin>
32    </plugins>
33  </build>

```

图 3-6

先输入 `mvn evosuite:help` 来调用 EvoSuite 插件，等待较长时间后下载完毕并提示构建成功。

输入 `mvn evosuite:generate` 来生成 Junit 测试。下载一部分文件后会提示 EvoSuite 会为 4 个类生成测试，大概需要 8 分钟，具体过程见图 3-7。

```
[INFO] * EvoSuite 1.0.6
[INFO] Registered remote process from /127.0.0.1:3407
[INFO] Going to execute 4 jobs
[INFO] Estimated completion time: 8 minutes, by 2018-11-19T21:39:53.515
[INFO] Going to start job for: tutorial.LinkedList. Expected to end in 190
[INFO] Registered remote process from /127.0.0.1:3408
[INFO] Registered remote process from /127.0.0.1:3415
[INFO] Completed job. Left: 3
[INFO] Going to start job for: tutorial.Stack. Expected to end in 125 seconds
[INFO] Registered remote process from /127.0.0.1:3449
[INFO] Registered remote process from /127.0.0.1:3518
[INFO] Completed job. Left: 2
[INFO] Going to start job for: tutorial.LinkedListIterator. Expected to end in 125 seconds
[INFO] Registered remote process from /127.0.0.1:3777
[INFO] Registered remote process from /127.0.0.1:3784
[INFO] Completed job. Left: 1
[INFO] Going to start job for: tutorial.Node. Expected to end in 60 seconds
[INFO] Registered remote process from /127.0.0.1:3922
[INFO] Registered remote process from /127.0.0.1:3929
[INFO] Completed job. Left: 0
[INFO] * Updating database to tutorial.LinkedListIterator
[INFO] * Updating database to tutorial.Node
[INFO] * Updating database to tutorial.Stack
[INFO] * Updating database to tutorial.LinkedList
[INFO] === CTG run results ===
[INFO] Removed test suites: 0
[INFO] New test suites: 4
[INFO] Stopping spawn process manager
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 06:18 min
[INFO] Finished at: 2018-11-19T21:36:36+08:00
[INFO] -----
```

图 3-7

输入 `mvn evosuite:info` 来了解具体信息，会输出有关信息，见图 3-8。

```
[INFO] * EvoSuite 1.0.6
[INFO] Total number of classes in the project: 4
[INFO] Number of classes in the project that are testable: 4
[INFO] Number of generated test suites: 4
[INFO] Overall coverage: 0.98625
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.181 s
```

图 3-8

可以看到 EvoSuite 为每个类都生成了测试文件，如下表。

Tutorial_Maven

--src

--main

--java

--tutorial

--Stack.java

--Node.java

--LinkedList.java

--LinkedListIterator.java

```

--test
--java
--tutorial
--StackTest.java
--evosuite
.....
--best-tests
--tutorial
--LinkedList_ESTest.java
--LinkedList_ESTest_scaffolding.java
--LinkedListIterator_ESTest.java
--LinkedListIterator_ESTest_scaffolding.java
--Node_ESTest.java
--Node_ESTest_scaffolding.java
--Stack_ESTest.java
--Stack_ESTest_scaffolding.java
--pom.xml

```

将生成的文件复制到 src\test\java\tutorial。接着开始使用 Maven 来执行 EvoSuite 测试，在 pom.xml 中添加新的依赖项目，见图 3-9。

```

24      <dependency>
25          <groupId>org.evosuite</groupId>
26          <artifactId>evosuite-standalone-runtime</artifactId>
27          <version>1.0.6</version>
28          <scope>test</scope>
29      </dependency>

```

图 3-9

接着尝试输入 mvn test 来执行测试，等待文件下载完毕。可以看到每个类的具体测试结果，如图 3-10。

```

-----
T E S T S
-----
Running tutorial.LinkedListIterator_ESTest
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further detail
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.832 sec
Running tutorial.LinkedList_ESTest
Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.034 sec
Running tutorial.Node_ESTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec
Running tutorial.StackTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running tutorial.Stack_ESTest
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 sec

Results :

Tests run: 28, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----

```

图 3-10

4. 进行实验

下载实验示例文档，Tutorial_Experiments，其 src\main\java\tutorial 内有许多个 java 文件。

运行 `mvn compile` 进行编译。输入 `mvn dependency:copy-dependencies -DincludeScope=runtime` 来下载相关的依赖文件并保存到 `target/dependency` 目录中。

依次输入 `set EVOSUITE=java -jar "%CD%\evosuite-1.0.6.jar`、`set CLASSPATH=target/classes;evosuite-standalone-runtime-1.0.6.jar;evosuite-tests;target/dependency/junit-4.12.jar;target/dependency/hamcrest-core-1.3.jar;target/dependency/commons-collections-3.2.2.jar` 和 `$EVOSUITE -setup target/classes target/dependency/commons-collections-3.2.2.jar`

生成的文件 `evosuite-files\evosuite.properties` 第一行见图 4-1。

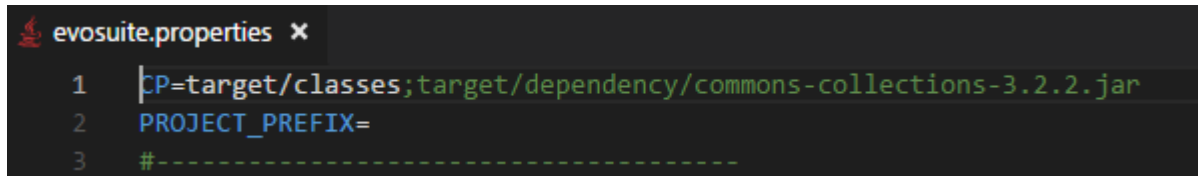


图 4-1

使用 EvoSuite 来生成测试数据，输入 `%EVOSUITE% -class tutorial.Person`，运行结果见图 4-2。

```
* Coverage of criterion CBRANCH: 100%
* Total number of goals: 3
* Number of covered goals: 3
* Generated 6 tests with total length 12
* Resulting test suite's coverage: 100% (average coverage for
* Generating assertions
* Resulting test suite's mutation score: 0%
* Compiling and checking tests
* Writing JUnit test case 'Person_ESTest' to evosuite-tests
* Done!

* Computation finished
```

图 4-2

生成了 `evosuite-repor\statistics.csv`，在前面也提到过。csv 文件的内容为，

`TARGET_CLASS,criterion,Coverage,Total_Goals,Covered_Goals`

`tutorial.Person,LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH,1.0,24,24`

第一列包含测试的类的名称 (`tutorial.Person`)。第二列是使用的覆盖标准 (目前是 EvoSuite 默认使用的标准的完整列表，用分号分隔)。第三列是实现的覆盖率 (1.0 为 100% 的覆盖率)，是根据覆盖目标覆盖率与总目标 (最后两列) 的比率计算得出。

再次使用 EvoSuite 来生成测试数据，输入 `%EVOSUITE% -class tutorial.Person -criterion branch`，只使用分支覆盖。可以看到 csv 文件出现了新的一行。

`TARGET_CLASS,criterion,Coverage,Total_Goals,Covered_Goals`

`tutorial.Person,LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH,1.0,24,24`

`tutorial.Person,BRANCH,1.0,3,3`

接着再次使用 EvoSuite 来生成测试数据，输入 `%EVOSUITE% -class tutorial.Company -criterion line`，这次同时更换了类和覆盖标准。可以看到 csv 文件又出现了新的一行。

`TARGET_CLASS,criterion,Coverage,Total_Goals,Covered_Goals`

`tutorial.Person,LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH,1.0,24,24`

`tutorial.Person,BRANCH,1.0,3,3`

`tutorial.Company,LINE,1.0,4,4`

此外，可以通过设置来更改 csv 中列的属性，如 `%EVOSUITE% -criterion branch -prefix tutorial -Doutput_variables=TARGET_CLASS,criterion,Size,Length,MutationScore`。不过要提前删除 csv 文件。运行完毕后 csv 文件的具体内容见图 4-3。

	A	B	C	D	E
1	TARGET_CLASS	criterion	Size	Length	MutationScore
2	tutorial.ATM	BRANCH	9	67	0.361111111
3	tutorial.ATMCard	BRANCH	8	32	0.666666667
4	tutorial.Bank	BRANCH	4	15	0.8
5	tutorial.BankAccount	BRANCH	2	6	0.8
6	tutorial.Owner	BRANCH	1	1	1
7	tutorial.CurrentAccount	BRANCH	2	7	0.695652174
8	tutorial.SavingsAccount	BRANCH	3	10	0.852941176
9	tutorial.Company	BRANCH	1	2	1
10	tutorial.Person	BRANCH	2	4	0

图 4-3

可以使用统计方法来对这些结果进行分析处理，例如使用 python 来收集数据并创建不同的图表，图 4-4 是给出的一个例子。

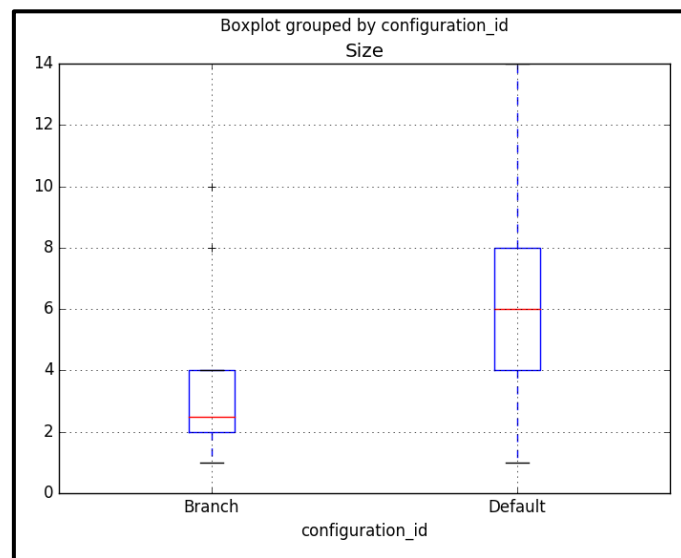


图 4-4