Cobertura
Open Source Tool for Code Coverage

Cobertura is Open Source Java technology based tool for Code Coverage. It calculates the percentage of the code accessed by tests. It accomplishes this by inserting instrumentation code into compiled Java byte-code for the application classes.

Saleemuddin Mohammad (mohdsaleem@yahoo.com)
2/14/2011
# Table of Contents

Table of Contents .......................................................................................................................... 2
Table of Figures ............................................................................................................................... 3
Introduction ......................................................................................................................................... 4
Integrate Cobertura into NetBeans IDE ............................................................................................ 5
Step 1. Download and install Cobertura .......................................................................................... 5
Step 2. Define Cobertura as Library using NetBeans IDE’s Library Manager ................................. 5
Step 3. Include Cobertura Library for “Run”, “Compile Tests” and “Run Tests” profiles ................. 7
Step 4. Customize NetBeans IDE’s build process ......................................................................... 10
Step 5. Use Cobertura to instrument application code and generate reports ............................... 13
Conclusion ........................................................................................................................................ 15
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cobertura installation</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Define new library</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Adding Cobertura JAR files to class path</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Adding Cobertura sources</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>Project properties</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Adding a Library</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>Adding Cobertura to Run-time libraries</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>Adding Cobertura to Compile-time Test libraries</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>Adding Cobertura to Run-time Test libraries</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>Customized Build.xml with Cobertura targets for instrumentation and report generation</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>Output from “Clean and Build” target for the project JUnit_4.8</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>Generate Cobertura coverage report</td>
<td>14</td>
</tr>
</tbody>
</table>
Introduction

Cobertura is Open Source Java technology based tool for Code Coverage. It calculates the percentage of the code accessed by test. It accomplishes this by inserting instrumentation of the Java byte-code. This document covers,

1. Download and installation
2. Integration into Software Development Lifecycle for projects using NetBeans IDE
3. Instrumentation of the application code
4. Testing and generating test coverage reports
Integrate Cobertura into NetBeans IDE

Follow the process illustrates below to install and integrate Cobertura into NetBeans IDE. This process includes following steps:

1. Download and install Cobertura.
2. Define Cobertura as Library using NetBeans IDE’s Library Manager.
3. Include Cobertura Library for “Run”, “Compile Tests” and “Run Tests” profiles.
4. Customize NetBeans IDE’s build process.
5. Use Cobertura to instrument application code and generate reports

Step 1. Download and install Cobertura

Cobertura is an Open Source tool and is available for free.

B. Install by unzipping the download to the folder of your choice. For example Figure 1: Cobertura installation illustrates installation to D:\INSTSW\cobertura-1.9.4.1 directory

![Cobertura Installation](Image)

**Figure 1: Cobertura installation**

Step 2. Define Cobertura as Library using NetBeans IDE’s Library Manager

NetBeans IDE provides capability to define external libraries. NetBeans projects can refer to these libraries during development lifecycle.

A. Select Libraries menu item from Tools menu. This will bring up NetBeans library manager
B. Define new library clicking “New Library…” button. Enter “Cobertura” as library name and “Class Libraries” as library type.

![New Library](image1.png)

**Figure 2: Define new library**

C. Add Cobertura jar files from following locations to Classpath by using “Add JAR/Folder…” button.
   a. D:\INSTSW\cobertura-1.9.4.1
   b. D:\INSTSW\cobertura-1.9.4.1\lib

![Library Manager](image2.png)

**Figure 3: Adding Cobertura JAR files to class path**
D. Optionally Add Cobertura library sources if you have downloaded by using “Add JAR/Folder...” button. For example, D:\INSTSW\cobertura-1.9.4.1\src\cobertura-1.9.4.1-src.zip

![Library Manager](image)

**Figure 4: Adding Cobertura sources**

**Step 3. Include Cobertura Library for “Run”, “Compile Tests” and “Run Tests” profiles**

With the above step complete, Cobertura library is ready to be used in NetBeans IDE.

A. Select “Project Properties” menu item from “File” menu (or Right click on project node and select “Project Properties” menu item from context menu) to bring up properties window.
B. Select “Libraries” category to display “Compile”, “Run”, “Compile Tests” and “Run Tests” tabs.
C. Select “Run” tab and add Cobertura to Run-time libraries list by clicking “Add Library...” and selecting Cobertura from available libraries.
D. Select “Compile Tests” tab and add Cobertura to Compile-time Test libraries list by clicking “Add Library...” and selecting Cobertura from available libraries.
E. Select “Run Tests” tab and add Cobertura to Run-time Test libraries list by clicking “Add Library...” and selecting Cobertura from available libraries. Note that Cobertura is the first library in the list of libraries. This ensures that the instrumented classes are used during test execution.

![Figure 9: Adding Cobertura to Run-time Test libraries](image)

**Step 4. Customize NetBeans IDE’s build process**

Now that we have defined and configured the Cobertura library for use, we can customize NetBeans IDE’s build process.

A. Add following to “project.properties” file to define the folders for the instrumented code and generated reports.

```properties
build.instrumented.dir=${build.dir}/instrumented
build.report.dir=${build.dir}/report
build.report.cobertura.dir=${build.report.dir}/Cobertura
```
B. Define Cobertura tasks by including following ant task in Build.xml file. This will define cobertura-instrument, cobertura-merge, cobertura-report, and cobertura-check Ant tasks for use in the build script.

```xml
<taskdef classpath="cobertura.jar" resource="tasks.properties"/>
```

C. Add "-init-check-cobertura" task by updating build.xml file to include following target. This task validates that the environment is initialized properly.

```xml
<target depends="-init-check,init" name="-init-check-cobertura">
  <fail unless="build.classes.dir">Must set build.classes.dir</fail>
</target>
```

D. Add "-post-compile" task by updating build.xml file to include following target. This task is called after the compilation of sources is complete. Note that this ensures that the code is instrumented immediately after compilation. I used this approach to keep the instrumented application to be in synch with the compiled application. If you desire to have a build process that is different than this, you need to identify appropriate targets to override/substitute or create new targets that you can call direct from the NetBeans IDE.

```xml
<target name="-post-compile" depends="-init-check-cobertura,init">
  <delete dir="${build.instrumented.dir}" />
  <delete dir="${build.report.cobertura.dir}" />
  <mkdir dir="${build.instrumented.dir}" />
  <mkdir dir="${build.report.cobertura.dir}" />
  <cobertura-instrument todir="${build.instrumented.dir}"
    fileset dir="${build.classes.dir}"
    include name="**/*.class"/>
</target>
```

E. Add “coberturareport” task by updating build.xml file to include following target. Based on the naming convention this target ("coberturareport") is available and you can call it direct from the NetBeans IDE. This target creates the reports from the “Cobertura.ser” file that collects the coverage information. Note that “Cobertura.ser” is created and updated incrementally by the instrumented byte code when executed. If desired you can choose to
delete the “Cobertura.ser” file after generating reports by uncommenting the “<delete file="cobertura.ser" />” task.

```
<target name="coberturareport" depends="init,test">
  <!-- You can disable the html portion if you are using the hudson plugin and rely on the xml -->
  <cobertura-report format="html" srcdir="${src.dir}"
destdir="${build.report.cobertura.dir}"/>
  <!-- Used by the hudson plug-in -->
  <cobertura-report format="xml" srcdir="${src.dir}"
destdir="${build.report.cobertura.dir}"/>
  <!-- <delete file="cobertura.ser" /> -->
</target>
```

Final build.xml when all steps in this section are executed will look as below.

![Customized Build.xml with Cobertura targets for instrumentation and report generation](image-url)

Figure 10: Customized Build.xml with Cobertura targets for instrumentation and report generation
Step 5. Use Cobertura to instrument application code and generate reports

A. Application code is instrumented each time the project is compile. The output will look as below If you perform “Clean and Build” of the project. Note that the instrumented code is in “D:\INSTSW\Sun\NetBeansProjects\SCJP\JUnit_4.8\build\instrumented” folder.

![Cobertura Output](image)

**Figure 11:** Output from “Clean and Build” target for the project JUnit_4.8

B. You can generate coverage report by selecting “coberturareport” target from the context menu by right clicking on build.xml and selecting Run Target→Other Targets→coberturareport. Note that
Figure 12: Generate Cobertura coverage report

C. You can view Cobertura coverage report by browsing index.html from “D:\INSTSW\Sun\NetBeansProjects\SCJP\JUnit_4.8\build\report\cobertura” folder. Sample output from the target “cobertura-report” is provided below. Note that the tests are executed prior to generating the reports based on the dependencies for this target.
Conclusion

Incorporating Cobertura into Software Development Life Cycle (SDLC) enables project leadership to measure the test coverage and improve the quality of the product delivered by making required changes to the test cases and testing.