Cobertura Plugin

Plugin Information
View Cobertura on the plugin site for more information.

The current thinking is to merge this plugin into more generic coverage plugin. Help appreciated.

This plugin allows you to capture code coverage report from Cobertura. Jenkins will generate the trend report of coverage.The Cobertura plugin can be downloaded here.

- Configuring the Cobertura Plugin
- Configuring build tools
- Version History

Configuring the Cobertura Plugin

1. Install the cobertura plugin (via Manage Jenkins -> Manage Plugins)
2. Configure your project's build script to generate cobertura XML reports (See below for examples with Ant and Maven2)
3. Enable the "Publish Cobertura Coverage Report" publisher
4. Specify the directory where the coverage.xml report is generated.
5. (Optional) Configure the coverage metric targets to reflect your goals.

Configuring build tools

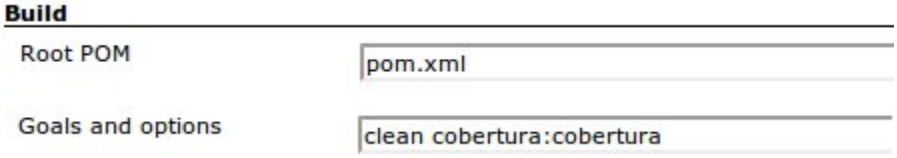
Here are the configuration details for common build tools. Please feel free to update this with corrections or additions.

Maven 2

Quick configuration

You can either, enable "cobertura" analysis in your 'pom.xml' files or just tell Jenkins to run "cobertura" goal.

If you don't want to change your pom files, just add the goal 'cobertura:cobertura' to your Maven project in Jenkins.



Single Project

If you are using a single module configuration, add the following into your pom.xml. This will cause cobertura to be called each time you run "mvn package".

```
<project ...>
    ...
    <build>
        ...
        <plugins>
            ...
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>cobertura-maven-plugin</artifactId>
                <version>2.5.1</version>
                <configuration>
                    <formats>
                        <format>xml</format>
                    </formats>
                </configuration>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>cobertura</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            ...
        </plugins>
        ...
    </build>
    ...
</project>
```

Execute cobertura only from Jenkins using profiles

You can reduce the load of your developer machine by using maven profiles to execute the plugin only if you are running within Jenkins. The configuration below shows how to enable the plugin based on the information if maven was started by Jenkins.

```
<project ...>
    ...
    <profiles>
        <!-- Jenkins by default defines a property BUILD_NUMBER which is used to enable the profile. -->
        <profile>
            <id>jenkins</id>
            <activation>
                <property>
                    <name>env.BUILD_NUMBER</name>
                </property>
            </activation>
            <build>
                <plugins>
                    <plugin>
                        <groupId>org.codehaus.mojo</groupId>
                        <artifactId>cobertura-maven-plugin</artifactId>
                        <version>2.2</version>
                        <configuration>
                            <formats>
                                <format>xml</format>
                            </formats>
                        </configuration>
                        <executions>
                            <execution>
                                <phase>package</phase>
                                <goals>
                                    <goal>cobertura</goal>
                                </goals>
                            </execution>
                        </executions>
                    </plugin>
                </plugins>
            </build>
        </profile>
    </profiles>
    ...
</project>
```

Project hierarchies

If you are using a common parent for all Maven2 modules you can move the plugin configuration to the pluginManagement section of the common *parent*...

```
<project ...>
    ...
    <build>
        ...
        <pluginManagement>
            <plugins>
                ...
                <plugin>
                    <groupId>org.codehaus.mojo</groupId>
                    <artifactId>cobertura-maven-plugin</artifactId>
                    <version>2.2</version>
                    <configuration>
                        <formats>
                            <format>xml</format>
                        </formats>
                    </configuration>
                    <executions>
                        <execution>
                            <phase>package</phase>
                            <goals>
                                <goal>cobertura</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>
                ...
            </plugins>
        </pluginManagement>
        ...
    </build>
    ...
</project>
```

And add the plugin group and artifact to the children

```
<project ...>
    ...
    <build>
        ...
        <plugins>
            ...
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>cobertura-maven-plugin</artifactId>
            </plugin>
            ...
        </plugins>
        ...
    </build>
    ...
</project>
```

Execute cobertura only from Jenkins using profiles

It is highly recommend to reduce the workload of the developers machines by disabling the cobertura plugin and only using it from within Jenkins. The following excerpt from the *parent* shows how to do so:

```
<project ...>
    ...
    <profiles>
        <!-- Jenkins by default defines a property BUILD_NUMBER which is used to enable the profile. -->
        <profile>
            <id>jenkins</id>
            <activation>
                <property>
                    <name>env.BUILD_NUMBER</name>
                </property>
            </activation>
            <build>
                <pluginManagement>
                    <plugins>
                        <plugin>
                            <groupId>org.codehaus.mojo</groupId>
                            <artifactId>cobertura-maven-plugin</artifactId>
                            <version>2.2</version>
                            <configuration>
                                <formats>
                                    <format>xml</format>
                                </formats>
                            </configuration>
                            <executions>
                                <execution>
                                    <phase>package</phase>
                                    <goals>
                                        <goal>cobertura</goal>
                                    </goals>
                                </execution>
                            </executions>
                        </plugin>
                    </plugins>
                </pluginManagement>
            </build>
        </profile>
    </profiles>
    ...
</project>
```

Now that your parent is only using the plugin management section if it is running from within Jenkins, you need the childern poms adapted as well:

```
<project ...>
    ...
    <!-- If we are running in Jenkins use cobertura. -->
    <profiles>
        <profile>
            <id>jenkins</id>
            <activation>
                <property>
                    <name>env.BUILD_NUMBER</name>
                </property>
            </activation>
            <build>
                <plugins>
                    <plugin>
                        <groupId>org.codehaus.mojo</groupId>
                        <artifactId>cobertura-maven-plugin</artifactId>
                    </plugin>
                </plugins>
            </build>
        </profile>
    </profiles>
    ...
</project>
```

Ant

You must first tell Ant about the Cobertura Ant tasks using a taskdef statement. The best place to do this is near the top of your build.xml script, before any target statements.

```
<property name="cobertura.dir" value="C:/javastuff/cobertura" />

<path id="cobertura.classpath">
    <fileset dir="${cobertura.dir}">
        <include name="cobertura.jar" />
        <include name="lib/**/*.jar" />
    </fileset>
</path>

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

You'll need to instrument the classes that JUnit will be testing (not the test classes themselves) as such:

```
<cobertura-instrument todir="${instrumented.dir}">
    <ignore regex="org.apache.log4j.*" />
    <fileset dir="${classes.dir}">
        <include name="**/*.class" />
        <exclude name="**/*Test.class" />
    </fileset>
    <fileset dir="${guiclasses.dir}">
        <include name="**/*.class" />
        <exclude name="**/*Test.class" />
    </fileset>
    <fileset dir="${jars.dir}">
        <include name="my-simple-plugin.jar" />
    </fileset>
</cobertura-instrument>
```

Here's an example call to the JUnit ant task that has been modified to work with Cobertura.

```
<junit fork="yes" dir="${basedir}" failureProperty="test.failed">
        <!--
                Specify the name of the coverage data file to use.
                The value specified below is the default.
        -->
        <sysproperty key="net.sourceforge.cobertura.datafile"
                file="${basedir}/cobertura.ser" />

        <!--
                Note the classpath order: instrumented classes are before the
                original (uninstrumented) classes.  This is important.
        -->
        <classpath location="${instrumented.dir}" />
        <classpath location="${classes.dir}" />

        <!--
                The instrumented classes reference classes used by the
                Cobertura runtime, so Cobertura and its dependencies
                must be on your classpath.
        -->
        <classpath refid="cobertura.classpath" />

        <formatter type="xml" />
        <test name="${testcase}" todir="${reports.xml.dir}" if="testcase" />
        <batchtest todir="${reports.xml.dir}" unless="testcase">
                <fileset dir="${src.dir}">
                        <include name="**/*Test.java" />
                </fileset>
        </batchtest>
</junit>
```

Finally, you need a task to generate the xml report, where 'destdir' is where you want the report (coverage.xml) generated.

Your cobertura.ser is generated to your module root.

srcdir is where your *.java files are located. If you use multiple modules in one build process you need to include the module name, if you use the simple srcdir parameter. It is not required to include module name if you use fileset.

```
<cobertura-report format="xml" destdir="${coveragereport.dir}" srcdir="${src.dir}" />
You can use multiple source directories this way:
<cobertura-report format="xml" destdir="${coveragereport.dir}" >

        <fileset dir="${src.dir.java}">

                <include name="**/*.java" />

        </fileset>

        <fileset dir="${src.dir.main}">

                <include name="**/*.java" />

        </fileset>

</cobertura-report>
```

Gradle

Running Cobertura in gradle, copied from Piotr Gabryanczyk's post at http://piotrga.wordpress.com/2010/04/17/gradle-cobertura-integration-revisited/ and tweaked to work for gradle 1.5:

Create cobertura.gradle in the root of your project:

```
logger.info "Configuring Cobertura Plugin"

configurations{
  coberturaRuntime {extendsFrom testRuntime}
}

dependencies {
  coberturaRuntime 'net.sourceforge.cobertura:cobertura:1.9.4'
}

def serFile="${project.buildDir}/cobertura.ser"
def classes="${project.buildDir}/classes/main"
def classesCopy="${classes}-copy"


task cobertura(type: Test){
  dependencies {
    testRuntime 'net.sourceforge.cobertura:cobertura:1.9.4'
  }

  systemProperty "net.sourceforge.cobertura.datafile", serFile
}

cobertura.doFirst  {
  logger.quiet "Instrumenting classes for Cobertura"
  ant {
    delete(file:serFile, failonerror:false)
    delete(dir: classesCopy, failonerror:false)
    copy(todir: classesCopy) { fileset(dir: classes) }

    taskdef(resource:'tasks.properties', classpath: configurations.coberturaRuntime.asPath)
    'cobertura-instrument'(datafile: serFile) {
      fileset(dir: classes,
              includes:"**/*.class",
              excludes:"**/*Test.class")
    }
  }
}

cobertura.doLast{
  if (new File(classesCopy).exists()) {
    //create html cobertura report
    ant.'cobertura-report'(destdir:"${project.reportsDir}/cobertura",
            format:'html', srcdir:"src/main/java", datafile: serFile)
    //create xml cobertura report
     ant.'cobertura-report'(destdir:"${project.reportsDir}/cobertura",
            format:'xml', srcdir:"src/main/java", datafile: serFile)
    ant.delete(file: classes)
    ant.move(file: classesCopy, tofile: classes)
  }
}
```

Apply Cobertura.gradle in your build.gradle

Either (if single project build)

```
apply plugin: 'java'
apply from: 'cobertura.gradle'
```

Or (if multi project build)

```
subprojects {
  apply plugin: 'java'
  apply from: "${parent.projectDir.canonicalPath}/cobertura.gradle"
}
```

Version History

Version 1.12.1 (10-May-2018)

- Failed to scout hudson.plugins.cobertura.MavenCoberturaPublisher (JENKINS-44200)
- Fix highlight for partially covered branches (JENKINS-13489)
- Don't round up 99.x% coverage to 100% (JENKINS-43866)

Version 1.12 (12-Nov-2017)

- Show why build failed when it missed coverage targets (JENKINS-47639)
- Prefix all logs with `[Cobertura]` (JENKINS-25781)
- Fix Phabricator compatibility regression (Issue #73)
- Add support for pipeline snippet generator

Version 1.11 (09-Aug-2017)

- Added pipline support for coverage targets (Issue #67)
- Publish jobs when onlyStable is false even if job fails (Issue #59)

Version 1.10 (25-Apr-2017)

- Support Jenkins pipeline (JENKINS-30700)
- Avoid error when CoverageMetric EnumSet is empty (JENKINS-6425)
- Remove deprecated use of ChartUtil.generateGraph (JENKINS-17800)
- Fix typo in Spanish properties

Version 1.9.8 (8-May-2016)

- Allow later concurrent builds to finish first (JENKINS-26823)
- Find code from Python coverage (JENKINS-13889)

Version 1.9.7 (4-Mar-2015)

- Fixes broken dashboard links when inside folder (JENKINS-26410)

Version 1.9.6 (25-Oct-2014)

- Fixed URL to coverage results in views and folders (JENKINS-24436)

Version 1.9.5 (24-Apr-2014)

- Added coverage column that shows line/branch coverage in views

Version 1.9.4 (17-Apr-2014)

- Fix display when data for one of the columns is missing (JENKINS-22412)

Version 1.9.3 (16-Oct-2013)

- More fixes of file descriptor leaks

Version 1.9.2 (9-Aug-2013)

- Cobertura Unable to delete coverage.xml on windows (JENKINS-18858).

Version 1.9.1 (14-Jun-2013)

- Added "most recent N builds" limiting option for coverage graph.
- Fixed columns order on dashboard(JENKINS-18218).

Version 1.9 (28-Apr-2013)

- SourceCodePainter overwrites original files (JENKINS-16252).
- table.source font-family should not specify courier (JENKINS-3567).
- show greenbar collectly in IE Quirks Mode (JENKINS-8568).
- There should be a cobertura summary item on the build status page (JENKINS-8441).
- show legend under the graph.
- sort order of metrics. package, file, class, method, line, condition.
- Cobertura plugin does not provide data to the REST API (JENKINS-13877).
- Cobertura ClassCastException (JENKINS-15703).

Version 1.8 (15-Dec-2012)

- Crop unusaged whitespace in coverage graph(JENKINS-16038).
- testing if workspace permissions
- fixed layout: added align="right" to be displayed collectly
- Cannot plublish cobertura reports if org.codehaus.mojo:cobertura-maven-plugin is not invoked(JENKINS-14552).
- Cobertura - add option to make build as unstable (or not at all) instead of failed when no coverage xml files are found (JENKINS-12857).

Version 1.7.1 (17-Oct-2012)

- fix regression JENKINS-15518

Version 1.7 (11-Oct-2012)

- Memory footprint reduction.
- JENKINS-15035

Version 1.6 (17-Aug-2012)

- Inconsistent delete button(JENKINS-14589).
- Allow the build to fail on low coverage (JENKINS-11025).
- Support for ratcheting (JENKINS-8326).
- include support for 'cobertura-it-maven-plugin'.

Version 1.5 (20-May-2012)

- Code Coverages dashboard portlet missing column("METHODS") (JENKINS-7366).
- cobertura coverage dashboard portlet not using numeric sort for percent columns (JENKINS-13250).
- updated Japanese localization.
- some fixes.

Version 1.4 (5-May-2012)

- cobertura conditionals not available with a French server + regexps optimizations (JENKINS-7540).
- Cobertura gives LinkageError in new Jenkins version (JENKINS-11398).
- Cobertura plugin should not fail maven build for maven release (JENKINS-12640) (pull-6).

Version 1.3 (13-Aug-2011)

- Change so output format will be in alphabetical order by default
- Put <pre>...</pre> tags around source code content in case cobertura directory is linked to source code
- Added description of the Source Encoding

Version 1.2 (25-Feb-2011)

- Update for Jenkins

Version 1.1 (11-Jan-2011)

- Fix http://issues.jenkins-ci.org/browse/JENKINS-8362 : cobertura plugin and maven3.

Version 1.0 (30-Jul-2010)

- Fix so 0/0 is counted as 100% instead of 0% coverage (ie, a method with no conditionals). (JENKINS-6790)
- Fix in source viewer so "\n" and "\r" (backslash+n/r, not actual newlines) are not omitted. (JENKINS-3566)
- Add support for dashboard plugin

Version 0.8.11 (22-Mar-2010)

- Fixed: source code unavailable when unstable (JENKINS-4803)
- Fixed an issue in internationalization on static enum clases which made some texts be shown in English.
- Fixed a bug in the way the tables were sorted (same problem than emma JENKINS-4173). Now they are sorted numerically instead of alphabetically.
- Added Spanish internationalization.

Version 0.8.10 (15-Jan-2010)

- Reorganize data structures to allow processing larger result files
- Use EnumMap and EnumSet for more compact in-memory representation of data
- Update code for more recent Hudson
- Change report colors as described here
- Internationalize messages (JENKINS-4920)

Version 0.8.9 (8-Jul-2009)

- Added green/red results bars to statistic blocks (JENKINS-3869)
- Improved support for multi-module SCMs other than Subversion (such as CVS) (JENKINS-1323)
- Fixed an issue that broke source highlighting for module build result pages (JENKINS-3938)

Version 0.8.8 (11-Jun-2009)

- Revert the memory usage fixes in 0.8.7, since they were breaking source highlighting (JENKINS-3597)

Version 0.8.7 (4-Jun-2009)

- Improved help and error messages to attempt to avoid "Can not find coverage-results" (JENKINS-1423)
- Fixed "Consider only stable builds" setting (JENKINS-3475)
- Improved memory usage when drawing trend graphs (JENKINS-3597)

Version 0.8.6 (7-May-2009)

- The plugin runs before notifications are sent out, to avoid inconsistency in build status reporting (JENKINS-1285)
- The cobertura statistics graphic on a project window isn't rendered (JENKINS-2851)

Version 0.8.4 (21-Oct-2007)

- ???

Version 0.8.3 (12-Oct-2007)

- Fixes JENKINS-915 for SubversionSCM only

Version 0.8.2 (4-Oct-2007)

- Hopefully fixed JENKINS-846

Version 0.8.1 (28-Sep-2007)

- Fixes issues running under JDK 1.5
- Fixes some issues with finding source code

Version 0.8 (20-Sep-2007)

- Works with JDK 5 as well as JDK 6 (removing JDK dependency introduced during regression fixing)

Version 0.7 (20-Sep-2007)

- Better fix of regressions introduced in 0.5

Version 0.6 (20-Sep-2007)

- Fix of regressions introduced in 0.5

Version 0.5 (20-Sep-2007)

- Now with built in source code painting! (Source code is available at the file level for the latest stable build only).

Note that the conditional coverage is the highest coverage from all the cobertura reports aggregated in each build. Thus if you have two reports and one covers only 50% of a conditional and the other covers a *different* 25%, conditional coverage will be reported as 50% and not the 75% that you could argue it should be!

- The trend graph now works when there are broken builds in the build history.

Version 0.4 (29-Aug-2007)

- Initial support for multi-report aggregation (may get totals incorrect if reports overlap for individual classes - I'll need to get source file painting support implemented to remove that issue. However, this is just how the files are parsed. This version will archive the files correctly so when it is fixed your history should report correctly)

Version 0.3 (28-Aug-2007)

- Fixed NPE parsing some cobertura reports generated by Cobertura version 1.8.
- Project level report should now work (except possibly when a build is in progress)

Version 0.2 (28-Aug-2007)

- Fixed problem with configuration (was not persisting configuration details)

- Changed health reporting configuration (now handles the more generic code)

- Tidy-up of reports
- Known issues:
  - Project level report does not work in all cases
  - Class and Method level reports should display source code with coverage if source code is available (in workspace)

Version 0.1 (27-Aug-2007)

- Initial release. Only parses xml report. Some rough edges in the UI.