Code License Manager

User Guide

Version: 2.0

Author: Tommy Svensson

Table of Contents

1 Code License Manager 2.0

1.1 Introduction

CodeLicenseManager (henceforth called CLM) was born out of frustration in handling licenses. Different IDE's have come up with different solutions for supplying ready "boilerplate" texts for different licenses, but go no further than that.

CLM solves this in an IDE independent way and also installs license texts of project license and third party licenses including a list of which of the used third party products uses each license. CLM can be used with any programming language and not only Java even though it is written in Java. It currently supports the following languages: Java (+ BeanShell), Java properties files, JSP, Groovy, Python, Ruby, C, C++, C#, CSS, HTML, XHTML, XML, XML schema, XSL Stylesheets, Bourne shell, AWK, Perl, but can easily be extended to other languages. See "Making your own ..." below for more information.

CLM also makes it extremely easy to change license (does happen) and allows you to not decide on a license at the start of a project, a license can be applied at any time.

If your CM tool allows pre checkin scripts to be run it can be plugged in at that time to make sure all checked in code has the license text. If a source file already has the license text nothing will be changed, if it doesn't it will be added.

1.2 Features

Code License Manager is more or less what it sounds like. Following is a short feature list:

- Updating source code with license "boilerplate" text and copyrights.
- Very easy to change license.
- No need for special source templates containing the license boilerplate text. Just create new source files from whatever template or none at all, and then run CodeLicenseManager "apply" before checkin.
- Updating source code with project information (description, current version, etc).
- Installing license texts for project license and third party licenses.
- For third party licenses a file listing the external dependencies using each license is also produced.
- Configurable where license texts should be installed.
- Running own source editing scripts.
- Can be run from:
- Maven (plugin).
  - Can resolve project license, third party licenses, copyrights, project description, project version, etc from maven pom.
  - Can download a third party license full text from "licenses/license/url" in the pom if the license cannot be found in an available license library.
  - Can produce APT documents for maven-site-plugin with a page specifying the project license with a link to a generated license page, and all licenses used by third party products (with links to license) and a list of third party products using each license. This is a better replacement for the maven license report.
- Ant (task).
  - Currently not supported due to a classloading problem with beanshell when run within an Ant task.
- Command line (java -jar).
  - This to support running from makefiles for other languages than Java. This tool is not limited to only Java development, it can be used for any programming language. The platform the development is done on must however have a Java5 or higher Java VM available for running CodeLicenseManager.

    - Requires an XML configuration file that is identical to the <configuration> section of the maven plugin.
- Uses license libraries for license boilerplate text and full license texts.
- A license library can point to the full license text on the web or include it as a text file.
- Includes a license library with the most common open source licenses.
- Easy to make own license libraries.
- Not only for open source. Can have private, closed source license in library.
- Uses source code update libraries (simply called source updaters) based on Bean Shell scripts for different languages and formats. Each script gets a programmable only text file editor instance with the current source file loaded and uses the editor methods to manipulate the text.
- Includes source updaters for the following types of comments:
    - / ... */_ - Recognized language extensions: bsh, c, cpp, css, groovy, java.
    - # - Recognized language extensions: sh, properties, ruby, py, perl, awk.
    - <!-- ... --> - Recognized language extensions: htm, html, xml, xsd, xsl, xhtml, ent.
    - <%-- ... --%> - Recognized language extensions: jsp.
    - Annotations - Recognized language extensions: java, groovy.
- Easy to make your own source updater.
- Has 2 variants of source updaters for Java and Groovy:
    - Project, license, and copyright information is provided in a comment block at the top of the file, just like for any other language.
    - Project, license, and copyright information is provided in annotations on the class. Annotations jars available in 3 variants:
        - Retention source only. (This is used by all Code License Manager code).
        - Some with retention runtime, and some with retention source.
        - All with retention runtime.
- Users can specify own scripts to run on source files in configuration either as separate script files or inline. These scripts can do whatever you want. For example update a constant with the products current version number, or whatever your creative mind can come up with. It is possible to run only user scripts without running source updater libraries.
- Requires Java5 or higher.

1.3 Definitions

CLM - Short for "Code License Manager".

License library - A jar file containing one or more licenses. When CLM is run from Maven it needs to be available in classpath. From the command-line tool a library jar is pointed out via an argument.

Source Updater (library) - A jar file containing source code update scripts for a specific comment type, and a set of source code extensions that are known to use that comment type for automatic resolving of updater to use.

1.4 Using

There are 4 actions that can be preformed:

apply - Apply license on source files.

install - Install full license texts in build.

script - Runs only configuration scripts and nothing else. This action has nothing to do with license handling at all, but is a quite useful function that can be used even if you never run apply or install.

delete – Runs delete scripts if they are provided by the used updater. This is the opposite of "apply".

2 Configuration

No matter how you run CLM it has an identical configuration. Here is an example:

configuration

The "project" section defines project information. It is required by *apply* and *install*.

```
<project>
    <name>My Project</name>
    <description>This is a description of my test project.</description>
    <codeVersion>1.5</codeVersion>
    <license>
        <type>Apache</type>
        <version>2.0</version>
    </license>
    <copyright>
        <holder>Me myself & I</holder>
        <year>2010</year>
    </copyright>
    <copyright>
        <holder>...</holder>
        <year>...</year>
    </copyright>
</project>
```

The "thirdpartyLicenses" section supplies information about third party code used by the project. This is required by *install*.

```
<thirdpartyLicenses>
    <license>
        <type>LGPL</type>
        <version>v3</version>
        <licenseProducts>
            <product>
                <name>BeanShell-1.3.0</name>
                <web>http://www.beanshell.org/</web>
            </product>
            <product>
                <name>Hibernate-3.5.0</name>
                <web>https://www.hibernate.org/</web>
            </product>
        </licenseProducts>
    </license>
    <license>
        <type>ASF</type>
        <version>2.0</version>
        <licenseProducts>
            <product>
                <name>log4j-1.2</name>
                <web>http://logging.apache.org/log4j/1.2/</web>
            </product>
        </licenseProducts>
    </license>
</thirdpartyLicenses>
```

The "scripts" section is a utility that lets projects make code edits in a simple way. This section is entirely optional, but when it is available it is used by *apply* and *script*.

```
<scripts>
    <script>
        <fileFilter>.*CodeLicenseManager\.java</fileFilter>
        <code>
            if (editor.find("Display.msg\\(\"CodeLicenseManager ")) {
                display("Updating version, copyright and maintainer!");
                editor.deleteCurrentLine();
                editor.insertLine("        Display.msg(\"CodeLicenseManager " +
                    "${app.version}\\nCopyright (C) ${copyrightYear} by " +
                    "${copyrightHolder}\\nMaintained by ${developer1Name} " +
```

```
                    "(${developer1Email})\");");
            }
        </code>
    </script>
    <script>
        <fileFilter>.*-source-updater-.*\.properties</fileFilter>
        <scriptFile>scripts/props.bsh</scriptFile>
    </script>
</scripts>
```

The "codeOptions" section provides options for and is required by *apply*.

```
<codeOptions>
    <verbose>true</verbose>
    <codeLanguage>by-extension</codeLanguage>
    <updateLicenseInfo>true</updateLicenseInfo>
    <updateCopyright>true</updateCopyright>
    <updateProject>true</updateProject>
    <addAuthorsBlock>true</addAuthorsBlock>
    <sourceCodeDirs>
        src/main/java/**/.*\.java,
        src/main/resources/**/.*\.xml,
        src/main/resources/**/.*\.properties,
        src/main/webapp/**/.*\.jsp
    </sourceCodeDirs>
</codeOptions>
```

The "installOptions" section provides options for and is required by *install*.

```
<installOptions>
    <verbose>true</verbose>
    <licenseDir>license</licenseDir>
    <thirdpartyLicenseDir>license/thirdparty</thirdpartyLicenseDir>
</installOptions>
```

The "userData" section provides name and value pairs that will be available to scripts as String variables with the specified name. There are available to both "scripts" section scripts and scripts in source-updater libraries. This section is optional and when available used by *apply* and *script*.

```
<userData>
    <name>...</name><value>...</value>
    <name>...</name><value>...</value>
</userData>
```

2.1 Configuration explanations

configuration/project (section)

Supplies project information.

configuration/project/name text

The name of the project.

configuration/project/description text

A description of the project.

configuration/project/codeVersion text

The current version of the project code.

configuration/project/subProjectOf text

A project of which this is subproject of.

configuration/project/license (section)

The license of the project.

configuration/project/license/type text

The license type. Example "LGPL" or "Apache".

configuration/project/license/version text

The version of the license. Example "v3" or "2.0".

configuration/project/copyright (section)

Copyright(s) held by the code.

configuration/project/copyright/year text

The copyright year.

configuration/project/copyright/holder text

The copyright holder.

configuration/project/copyright/rights text

Is holds the string "All rights reserved." by default. If you want something else, change it.

---

configuration/thirdpartyLicenses (section)

Third party licenses.

configuration/thirdpartyLicenses/license (section)

Specifies licenses of used third party products.

configuration/thirdpartyLicenses/license/licenseProducts (section)

The used third party products using this license. This is only relevant for third party licenses!

configuration/thirdpartyLicenses/license/licenseProducts/product (section)

A used third party product having this license.

configuration/thirdpartyLicenses/license/licenseProducts/product/name text

The name of the product.

configuration/thirdpartyLicenses/license/licenseProducts/product/version text

The version of the product.

configuration/thirdpartyLicenses/license/licenseProducts/product/web text

The products web site.

configuration/thirdpartyLicenses/license/licenseUrl text

The url to the license text on the web. This is required if the license cannot be found in license library!

configuration/thirdpartyLicenses/license/type text

The license type. Example "LGPL" or "Apache".

configuration/thirdpartyLicenses/license/version text

The version of the license. Example "v3" or "2.0".

---

configuration/installOptions (section)

Provides license file install options.

configuration/installOptions/verbose true/false

If true verbose output is provided.

configuration/installOptions/licenseDir text

The directory to where the license text should be copied.

Defaults to 'license'.

configuration/installOptions/thirdpartyLicenseDir text

The directory to where the third party license texts are copied. Defaults to 'license/thirdparty'.

---

configuration/codeOptions (section)

Provides source code update options.

configuration/codeOptions/verbose true/false

If true verbose output is provided.

configuration/codeOptions/codeLanguage text

The language in which to process source code for.

configuration/codeOptions/updateLicenseInfo true/false

If true the license information in the source will be updated.

configuration/codeOptions/updateCopyright true/false

If true the Copyright information in the source will be updated.

configuration/codeOptions/updateProject true/false

If true the Project information will be updated with the information specified.

configuration/codeOptions/addAuthorsBlock true/false

If true The Authors information will be added if it does not exist.

configuration/codeOptions/sourceCodeDirs text

A comma separated list of source paths of files to update. The directory part of the path can have **wildcard which means all directories below**. **The file part of the path is regular expression**. **For example: ".***.java".*

Here are some examples:

*.../myproj/src/main/java/**/*.java_ - All java files under .../src/main/java and below.

.../myproj/src/main/java/some/package - All files directly in the "package" directory.

.../myproj/src/main/resources/help/.hlp_ - All .hlp files in the help directory.

*.../myproj/src/main/*** - All files in the main directory and below.

---

configuration/userData (section)

Provides user information for use in personal source code updaters. A name/value pair can occur multiple times!

configuration/userData/name text

Specifies name for the data.

configuration/userData/value text

Specifies data value.

---

configuration/scripts (section)

Specifies scripts to run on source files of specified file extension.

configuration/scripts/script (section)

A script to execute.

configuration/scripts/script/fileFilter text (Required)

A regular expression file filter for which files to apply script to.

configuration/scripts/script/code text

Inline script code to run if <scriptFile/> is not specified.

configuration/scripts/script/scriptFile text

This points to script in separate file to execute instead of the <code/> block.

2.2 Maven

When you are using maven the <configuration>...</configuration> section above is part of the maven plugin specification in the pom.xml file. Example:

```
...
<plugins>
    <plugin>
        <groupId>se.biltmore.tools.codelicmgr</groupId>
        <artifactId>CodeLicenseManager-maven-plugin</artifactId>
        <version>2.0</version>
```

```
        <dependencies>
            <dependency>
                <groupId>se.biltmore.tools.codelicmgr</groupId>
                <artifactId>
                    CodeLicenseManager-licenses-common-opensource
                </artifactId>
                <version>2.0</version>
            </dependency>
        </dependencies>

        <configuration>
                ...
        </configuration>

        <executions>
            <execution>
                <id>install-lics</id>
                <goals>
                    <goal>install</goal>
                </goals>
                <phase>install</phase>
                <configuration>
                    <installOptions>
                    ...
                    </installOptions>
                </configuration>
            </execution>
        </executions>
    </plugin>
    ...
</plugins>
...
```

If you are running the *apply* goal you also need one or more source updater libraries among the dependencies.

2.2.1 Maven special feature 1 - Skip project section

With maven you can skip the project section since that information can be resolved elsewhere in the pom:

```
<project ...>
    ...
    <version>1.0</version>
    ...
    <name>CodeLicenseManager</name>
    <description>
        Manages project and license information in project sourcecode
        and provides license text files for inclusion in builds. Supports
        multiple languages and it is relatively easy to add a new
        language and to make your own custom source code updater.
    </description>
    <licenses>
        <license>
            <!--
                Name needs to be in "{type} {version}" or
                "{type}-{version} format to be reused by the plugin.
            -->
            <name>Apache 2.0</name>
            <url>http://www.apache.org/licenses/LICENSE-2.0</url>
        </license>
    </licenses>
    <organization>
        <!-- copyright holder -->
        <name>Natusoft AB</name>
        <url>http://www.natusoft.se/</url>
    </organization>
    <!-- copyright year -->
    <inceptionYear>2009</inceptionYear>
    <developers>
        <developer>
            <name>Tommy Svensson</name>
            <email>tommy@natusoft.se</email>
        </developer>
    </developers>
    ...
</project>
```

2.2.2 Maven special feature 2 - Automatic resolve of third party licenses

Unless you specify:

```
...
<configuration>
    ...
    <autoResolveThirdPartyLicenses>false</autoResolveThirdPartyLicenses>
</configuration>
...
```

the "install" goal will inspect all the dependencies and try to resolve the licenses used by third party code. For this to work it requires that the dependency has specified license information in its pom. Most pom's out there in the central repository are unfortunately quite sloppy. Any dependencies having the same group id as the project being built will be ignored. Any dependency with scope "test" will also be ignored.

I recommend running the *install* goal in the "install" phase of the standard maven build lifecycle. I also recommend running the *apply* goal in an "apply-lics" profile in the "generate-sources" phase of the standard maven build lifecycle and do "mvn -P apply-lics generate-sources" whenever an update is needed, for example when new source files have been created.

2.2.3 Maven special feature 3 - Better license documents for use with maven-site-plugin.

2.2.3.1 APT generation

If you specify:

```
...
<configuration>
    ...
    <createLicencesAPT>true</createLicencesAPT>
</configuration>
...
```

then an apt document for each license will be written to *src/site/apt/licenses* plus a licenses.apt document. This generated page shows the project license with a link to the generated apt document for that license and all third party licenses where license name is both a section title and a link to the generated apt document for that license. The section then lists the products using the license and a link to the products website.

The generated licenses.apt document should be specified in the site.xml instead of license.html (you should actually not include the standard maven site license report when using this!):

```
...
<menu name="Code License Manager">
    ...
    <item name="Project License" href="licenses/licenses.html"/>

    <item name="Issue Tracking" href="issue-tracking.html"/>
    <item name="Project Team" href="team-list.html"/>
    <item name="Source Repository" href="source-repository.html"/>
</menu>
...
```

Note that you only need to include the licenses/licenses.html page since it provides links to each license page.

If you are also using the maven-pdf-plugin you might have noticed that the pdf plugin does not handle pre-formatted texts larger than a page, it will only include one page and the rest is cut off. To go around this problem you can modify

the above config by adding one more parameter:

```
...
<configuration>
    ...
    <createLicencesAPT>true</createLicencesAPT>
    <createMavenPDFPluginLicenseAPTVersions>
        true
    </createMavenPDFPluginLicenseAPTVersions>
</configuration>
...
```

As this very long parameter suggests this will also generate a PDF plugin version of the license apt documents that contains appropriate page breaks. These variants will be prefixed with "pdf-". Use the licenses.apt and these pdf variants in pdf.xml:

```
...
<toc name="Table of Contents">
  <item name="User Guide" ref="UserGuide.apt"/>

  <item name="Licenses" ref="licenses/licenses.apt"/>
  <item name="Apache Software License 2.0" ref="licenses/pdf-Apache-2.0.apt"/>
  <item name="Lesser Gnu Public License v3" ref="licenses/pdf-LGPL-v3.apt"/>
</toc>
...
```

2.2.3.2 Markdown generation

If you use services like GitHub and Bitbucket and want to document in markdown without having to generate a whole site with the maven-site-plugin then you can specify the following options:

```
...
<configuration>
    ...
    <createLicensesMarkdown>true</createLicensesMarkdown>  (1)
    <markdownTargetSubdir>docs</markdownTargetSubdir>       (2)
    <appendUpdateLicensesMarkdownToMarkdownDocument>        (3)
        MyMarkdownDoc.md
    </appendUpdateLicensesMarkdownToMarkdownDocument>

</configuration>
...
```

1. This will generate the same document as for APT listing project license and thirdparty licenses with links, but in markdown format.
2. By default the licenses.md file and the *licence*.md files for each license will be generated in the project root, but specifying the second option in the example above specifies another relative path to put it in.
3. Specifying the third option in the example above will append/update another markdown document with the project license and third party licenses instead of generating it as a separate document. If it doesn't exist in the document already then it will be added to the bottom of the document. If it already exists the existing will be updated with a new version. When first added the license information is always put at the bottom, but if you manually move it somewhere else in the document (including the start and end html comment) then it will be updated in its new place.

Putting the following into your document will place the license info there from the start:

```
<!-- Created by CodeLicenseManager -->
<!-- CLM -->
```

2.3 Command-line

The command line variant can be run with:

```
cd .../CodeLicenseManager-dist-2.0
java -jar bin/CodeLicenseManager-command-line-2.0-exec.jar arg ... arg
```

or

```
bin/clm.sh arg ... arg
```

The following arguments are available:

--config path (Required)

This specifies config file to use for this run. This is the one described in the "Configuration" section above.

--action (constrained text) (Required)

Specifies the action. Valid values are: "apply", "install", "script", "delete".

The "apply" action updates source files according to the supplied configuration file.

The "install" action installs license files in paths specified in the supplied configuration file.

The "script" action only runs the scripts specified in the supplied configuration file.

The "delete" action runs delete scripts in an source updater. This is basically the opposite of "apply".

The <codeOptions> section is only used by the "apply" and "script" action, and the <installOptions> section is only used by the "install" action.

--licenselibrary text (Required)

Specifies the license library jar to use. Please note that this is an url! (Example: file:lib/license/CodeLicenseManager-licenses-common-opensource-2.0.jar).

--sourceupdaters text (Required)

A comma separated list of source updater jars to use. At least one have to be specified. Please note that each specified source updater must be an url! (Example: file:lib/updaters/CodeLicenseManager-source-updater-slashstar-comment-2.0.jar).

--help

Provides help information.

2.4 Ant

Classloading for beanshell fails when it is invoked from an Ant task! Resolving this not being a #1 priority, I will currently not support Ant.

2.5 User Specific Configuration

When source code is updated there is an option to add author information if it is not already available. For the "slashstar" updater it will look like this for example:

- AUTHORS
- Tommy Svensson (tommy@natusoft.se)
- Changes:
- 2009-11-09: Created!

When an author block is added, information about the user is read from *.clm-user.properties* in the users home directory (/home/*user* on most unix systems, /Users/*user* on Mac OS X, and C:\Documents and Settings*user* on windows). This property file contains the following properties:

```
name=
email=
```

If this file is not found the standard "user" system property is used to determine the system username, and email will not be provided.

3 Updating Source Code

This requires the "project" and "codeOptions" configuration sections. The "scripts" and "userData" sections are optional.

3.1 Maven

Please remember that the "project" section is optional if the information is specified elsewhere in the pom. See the "Configuration/Maven" section above.

Use the CodeLicenseManager-maven-plugin with the "apply" goal. Here is an example:

```
    <plugin>

        <groupId>se.natusoft.tools.codelicmgr</groupId>
        <artifactId>CodeLicenseManager-maven-plugin</artifactId>
        <version>2.0</version>

        <dependencies>
            <dependency>
                <groupId>se.natusoft.tools.codelicmgr</groupId>
                <artifactId>CodeLicenseManager-licenses-common-opensource</artifactId>
                <version>2.0</version>
            </dependency>
            <dependency>
                <!-- This must come first since it overrides the comment
                     version for java in the next dependency. -->
                <groupId>se.natusoft.tools.codelicmgr</groupId>
                <artifactId>CodeLicenseManager-source-updater-java-
annotation</artifactId>
                <version>2.0</version>
            </dependency>
            <dependency>
                <!-- For the bsh scripts. -->
                <groupId>se.natusoft.tools.codelicmgr</groupId>
                <artifactId>CodeLicenseManager-source-updater-slashstar-
comment</artifactId>
                <version>2.0</version>
            </dependency>
            <dependency>
                <!-- For properties. -->
                <groupId>se.natusoft.tools.codelicmgr</groupId>
                <artifactId>CodeLicenseManager-source-updater-hash-comment</artifactId>
                <version>2.0</version>
            </dependency>
        </dependencies>

        <configuration>

            <project>
                <name>My Project</name>
                <description>This is a description of my project.</description>
                <codeVersion>${project.version}</codeVersion>
                <license>
                    <type>Apache</type>
                    <version>2.0</version>
                </license>
                <copyright>
                    <holder>Me myself & I</holder>
                    <year>2010</year>
                </copyright>
            </project>

        </configuration>
        <executions>
            <execution>
                <id>apply-licence-info</id>
                <goals>
                    <goal>apply</goal>
                </goals>
```

```
            <phase>generate-sources</phase>
            <configuration>

                <codeOptions>
                    <verbose>true</verbose>
                    <codeLanguage>by-extension</codeLanguage>
                    <updateLicenseInfo>true</updateLicenseInfo>
                    <updateCopyright>true</updateCopyright>
                    <updateProject>true</updateProject>
                    <addAuthorsBlock>true</addAuthorsBlock>
                    <sourceCodeDirs>
                        src/main/java/**/.*.java,
                        src/main/resources/**/.*.bsh,
                        src/main/resources/**/.*.properties
                    </sourceCodeDirs>
                </codeOptions>

            </configuration>
        </execution>
    </executions>
</plugin>
```

The "apply" goal should preferably be run within a profile since it is unnecessary to run it on every build.

3.2 Command line

Create a configuration file. For example: apply.xml:

```
<configuration>

    <project>
        <name>My Project</name>
        <description>This is a description of my project.</description>
        <codeVersion>1.0</codeVersion>
        <license>
            <type>Apache</type>
            <version>2.0</version>
        </license>
        <copyright>
            <holder>Me myself & I</holder>
            <year>2010</year>
        </copyright>
    </project>

    <codeOptions>
        <verbose>true</verbose>
        <codeLanguage>by-extension</codeLanguage>
        <updateLicenseInfo>true</updateLicenseInfo>
        <updateCopyright>true</updateCopyright>
        <updateProject>true</updateProject>
        <addAuthorsBlock>true</addAuthorsBlock>
        <sourceCodeDirs>
            src/main/java/**/.*.java,
            src/main/resources/**/.*.bsh,
            src/main/resources/**/.*.properties
        </sourceCodeDirs>

    </codeOptions>

</configuration>
```

Then run:

java -jar bin/CodeLicenseManager-command-line-2.0-exec.jar --config apply.xml --action apply --licenselibrary lib/license/CodeLicenseManager-licenses-common-opensource-2.0.jar --sourceupdaters lib/updaters/CodeLicenseManager-source-updater-java-annotation-2.0.jar, lib/updaters/CodeLicenseManager-source-updater-slashstar-comment-2.0.jar,

```
    lib/updaters/CodeLicenseManager-source-updater-hash-comment-2.0.jar
```

This is preferably put in a script, which in turn can be called from a makefile. The above example assumed you were standing in the distribution root directory.

4 Installing License Files

Installing license files requires the "project" (only "license" part), "thirdpartyLicenses", and "installOptions" configuration sections.

4.1 maven

Please remember that the "project" section is optional if the information is specified elsewhere in the pom. See the "Configuration/Maven" section above.

When run with maven CLM will try to automatically resolve third party licenses from their poms for any license not configured in the plugin (se below). When a license is found that is not available in a license library the license text will be downloaded from the web and installed as an html file if a license url specification is available.

If APT generation of licenses have been configured and the license is a downloaded such it will be generated only if the license url does not end in .html or .htm! If the url does not have one of these extensions it is assumed to be a text file and can still be used to generate APT. HTML can not.

if Markdown generation of licenses have been configured and the license is a downloaded such it will still be generated since Markdown supports HTML mixed with Markdown. The HTML will however be slightly filtered to improve the result. It will also obviously not be generated as a code block as is done for text licenses.

Use the CodeLicenseManager-maven-plugin with the "install" goal. Here is an example:

```
    <plugin>
        <groupId>se.natusoft.tools.codelicmgr</groupId>
        <artifactId>CodeLicenseManager-maven-plugin</artifactId>
        <version>2.0</version>

        <dependencies>
            <dependency>
                <groupId>se.natusoft.tools.codelicmgr</groupId>
                <artifactId>CodeLicenseManager-licenses-common-opensource</artifactId>
                <version>2.0</version>
            </dependency>
        </dependencies>

        <configuration>

            <project>
                <license>
                    <type>Apache</type>
                    <version>2.0</version>
                </license>
            </project>
            <thirdpartyLicenses>
                <license>
                    <type>LGPL</type>
                    <version>v3</version>
                    <licenseProducts>
                        <product>
                            <name>bsh</name>
                            <version>1.3.0</version>
                            <web>http://www.beanshell.org/</web>
                        </product>
                    </licenseProducts>
                </license>
                <license>
                    <type>Apache</type>
                    <version>2.0</version>
                    <licenseProducts>
                        <product>
                            <name>FileEditor</name>
                            <version>2.0</version>
                            <web>http://fileeditor.sf.net/</web>
```

```
                        </product>
                        <product>
                            <name>OptionsManager</name>
                            <version>2.0</version>
                            <web>http://optionsmanager.sf.net/</web>
                        </product>
                    </licenseProducts>
                </license>
            </thirdpartyLicenses>

        </configuration>

        <executions>
            <execution>
                <id>install-lics</id>
                <goals>
                    <goal>install</goal>
                </goals>
                <phase>install</phase>
                <configuration>

                    <installOptions>
                        <verbose>true</verbose>
                        <licenseDir>
                            target/license
                        </licenseDir>
                        <thirdpartyLicenseDir>
                            target/license/thirdparty
                        </thirdpartyLicenseDir>
                    </installOptions>

                    <!-- APT generation (se section above about special maven features)
-->
                    <createLicencesAPT>
                        true
                    </createLicencesAPT>
                    <createMavenPDFPluginLicenseAPTVersions>
                        true
                    </createMavenPDFPluginLicenseAPTVersions>

                    <!-- Markdown generation (se section above about special maven
features) -->
                    <createLicensesMarkdown>true</createLicensesMarkdown>
                    <markdownTargetSubdir>docs</markdownTargetSubdir>
                    <appendUpdateLicensesMarkdownToMarkdownDocument>
                        MyMarkdownDoc.md
                    </appendUpdateLicensesMarkdownToMarkdownDocument>
                </configuration>
            </execution>
        </executions>

    </plugin>
```

4.2 Command line

Create a configuration file. For example: install.xml:

```
    <configuration>

        <project>
            <license>
                <type>Apache</type>
                <version>2.0</version>
            </license>
        </project>

        <thirdpartyLicenses>
            <license>
                <type>LGPL</type>
                <version>v3</version>
                <licenseProducts>
                    <product>
                        <name>bsh</name>
                        <version>1.3.0</version>
```

```
                    <web>http://www.beanshell.org/</web>
                </product>
            </licenseProducts>
        </license>
        <license>
            <type>Apache</type>
            <version>2.0</version>
            <licenseProducts>
                <product>
                    <name>FileEditor</name>
                    <version>2.0</version>
                    <web>http://fileeditor.sf.net/</web>
                </product>
                <product>
                    <name>OptionsManager</name>
                    <version>2.0</version>
                    <web>http://optionsmanager.sf.net/</web>
                </product>
            </licenseProducts>
        </license>
    </thirdpartyLicenses>

    <installOptions>
        <verbose>true</verbose>
        <licenseDir>
            .../CodeLicenseManager/license
        </licenseDir>
        <thirdpartyLicenseDir>
            .../CodeLicenseManager/license/thirdparty
        </thirdpartyLicenseDir>
    </installOptions>

</configuration>
```

Then run:

> *java -jar bin/CodeLicenseManager-command-line-2.0-exec.jar --config install.xml --action install --licenselibrary lib/license/CodeLicenseManager-licenses-common-opensource-2.0.jar*

This is preferably put in a script, which in turn can be called from a makefile. The above example assumed you were standing in the distribution root directory.

5 Running Own Configurable Code Update Scripts

This requires the "codeOptions" (only "sourceCodeDirs") and the "scripts" configuration sections.

You may notice in the examples below that there are 2 different file specifications in 2 places (codeOptions/sourceCodeDirs and scripts/script/fileFilter). Both use regular expressions to match files and both must be specified. The first (codeOptions/sourceCodeDirs) specifies all the files to process. The second (scripts/script/fileFilter) limits the first result even more. The reason for this is that the scripts section can also be used for the "apply" goal/action and in that case you might not want to run the script for all files that have a license boilerplate applied. The "script" goal/action are basically the same as the "apply" goal/action. The only difference is that the "script" goal/action never runs the "apply" parts, but only the scripts. Thereby the "script" goal/action also has a subset of the "apply" configuration. So everything that can be specified for the "script" goal/action can also be specified for the "apply" goal/action.

5.1 maven

Use the CodeLicenseManager-maven-plugin with the "script" goal. Here is an example:

```
    <plugin>

        <groupId>se.natusoft.tools.codelicmgr</groupId>
        <artifactId>CodeLicenseManager-maven-plugin</artifactId>
        <version>2.0</version>

        <dependencies>
        </dependencies>

        <configuration>

            <scripts>
                <script>
                    <fileFilter>.*CodeLicenseManager.java</fileFilter>
                    <code>
                        editor.moveToTopOfFile();
                        if (editor.find("Display.msg\\(\"CodeLicenseManager ")) {
                            display("Updating version, copyright and maintainer!");
                            editor.deleteCurrentLine();
                            editor.insertLine("        Display.msg(\"CodeLicenseManager
" +
                                "${app.version}\\nMaintained by " +
                                "${developer1Name} (${developer1Email})\");");
                        }
                    </code>
                    <!-- alternative:
                    <scriptFile>scripts/updVersion.bsh</scriptFile>
                    -->
                </script>
            </scripts>

        </configuration>
        <executions>
            <execution>
                <id>run-scripts</id>
                <goals>
                    <goal>script</goal>
                </goals>
                <phase>generate-sources</phase>
                <configuration>

                    <codeOptions>
                        <verbose>true</verbose>
                        <sourceCodeDirs>
                            src/main/java/**/.*.java
                        </sourceCodeDirs>
                    </codeOptions>

                </configuration>
```

```
        </execution>
    </executions>
</plugin>
```

5.2 Command line

Create a configuration file. For example scripts.xml:

```
<configuration>

   <codeOptions>
        <verbose>true</verbose>
        <sourceCodeDirs>
            src/main/java/**/.*.java
        </sourceCodeDirs>
    </codeOptions>

    <scripts>
        <script>
            <fileFilter>.*CodeLicenseManager.java</fileFilter>
            <code>
                editor.moveToTopOfFile();
                if (editor.find("Display.msg\\(\"CodeLicenseManager ")) {
                    display("Updating version, copyright and maintainer!");
                    editor.deleteCurrentLine();
                    editor.insertLine("        Display.msg(\"CodeLicenseManager " +
                        "${app.version}\\nMaintained by " +
                        "${developer1Name} (${developer1Email})\");");
                }
           </code>
           <!-- alternative:
           <scriptFile>scripts/updVersion.bsh</scriptFile>
           -->
        </script>
    </scripts>

</configuration>
```

Then run:

java -jar bin/CodeLicenseManager-command-line-2.0-exec.jar --config script.xml --action script

6 Deleting license / project information from source

This requires the "codeOptions" (only "sourceCodeDirs") configuration section.

This will do the opposite of "apply". Why is this available ? Each source updater works only with itself. That is it recognizes only information added by itself. You cannot use one source updater to apply information to your code and then change to another source updater and expect it to update the old information with the new. In that case you will get both. So if you for some reason want to switch to another source updater (for example, you have decided to make your own updater) then you must run the "delete" goal/action using the same updater that created the information in the first place. Then you can switch to another updater and apply again.

Also note that it is entirely optional for updaters to support delete! Currently only "java-annotation" and "groovy-annotation" does!

6.1 maven

Use the CodeLicenseManager-maven-plugin with the "delete" goal. Here is an example:

```
    <plugin>

        <groupId>se.biltmore.tools.codelicmgr</groupId>
        <artifactId>CodeLicenseManager-maven-plugin</artifactId>
        <version>2.0</version>

        <dependencies>
            <dependency>
                <groupId>se.biltmore.tools.codelicmgr</groupId>
                <artifactId>CodeLicenseManager-source-updater-java-
annotation</artifactId>
                <version>2.0</version>
            </dependency>
        </dependencies>

        <configuration>

        </configuration>
        <executions>
            <execution>
                <id>delete</id>
                <goals>
                    <goal>delete</goal>
                </goals>
                <phase>generate-sources</phase>
                <configuration>

                    <codeOptions>
                        <verbose>true</verbose>
                        <sourceCodeDirs>
                            src/main/java/**/.*.java
                        </sourceCodeDirs>
                    </codeOptions>

                </configuration>
            </execution>
        </executions>
    </plugin>
```

6.2 Command line

Create a configuration file. For example delete.xml:

```
    <configuration>

        <codeOptions>
```

```
            <verbose>true</verbose>
            <sourceCodeDirs>
                src/main/java/**/.*.java
            </sourceCodeDirs>
        </codeOptions>

    </configuration>
```

Then run:

java -jar bin/CodeLicenseManager-command-line-2.0-exec.jar --config delete.xml --action delete

7 Available Libraries

7.1 License

A license library contains both a full license text and a boilerplate text for source files required by the license. All opensource licenses are available on the web and it is possible for a license library to point to a license text on the web instead of a local copy, but the license boilerplate text for source files are difficult to extract from the license info on the web so that must be available as a local in library text file. All the licenses in the supplied license library also have the full license text locally in the library.

CodeLicenseManager-licenses-common-opensource

This contains the most common open source licenses:

- Apache Public License (APL) 2.0
- Common Development and Distribution License (CDDL) 1.0
- Eclipse Public License (EPL) 1.0
- GNU General Public License (GPL) v2
- GNU General Public License (GPL) v3
- GNU Affero General Public License v3
- GNU Lesser General Public License (LGPL) v2.1
- GNU Lesser General Public License (LGPL) v3
- MIT License (MIT) 1.0
- Mozilla Public License (MPL) 1.1

The information for these are taken from www.opensource.org.

7.2 Source Updaters

A source updater is a set of beanshell www.beanshell.org scripts that uses preloaded FileEditor instance to update source files with license information and boilerplate text, and optionally also project information and author information. There are different updaters for different kinds of comment types.

CodeLicenseManager-source-updater-java-annotation

This updater uses annotations for all information instead of putting it within a comment. This also requires a dependency on one of the *CodeLicenseManager-annotations* variants (*-retention-source* / *-retention-runtime* / *-retention-runtime-all*). If the *-retention-source* variant is used there is only a compile time dependency. For *-retention-runtime* you get copyright and license information as runtime retention, and with *-retention-runtime-all* all annotations are runtime.

The annotations are all "Type" annotations. That is they are annotations on the class/interface/enum/etc. For JavaDoc to be able to see the class javadoc comment the annotations must be below the class comment, so this updater will insert them below the class comment if such is found.

CodeLicenseManager-source-updater-groovy-annotation

This does exactly the same as CodeLicenseManager-source-updater-java-annotation, but for groovy source. The same annotations are used in groovy, but groovy have a different way of specifying an array of information than java.

Java uses '{' and '}' while groovy uses '' and ''. Some of the annotations uses arrays and thereby a different updater is needed for groovy source.

CodeLicenseManager-source-updater-slashstar-comment

This updates source code with all license and other information within a

```
/*
 * ...
 */
```

comment block. If the source file does not start with such a comment one is added otherwise the information is inserted within the existing comment, but first in the comment block.

This updater recognizes the following source code extensions:

- bsh
- c
- cpp
- cs
- css
- groovy
- java

If you know of other valid source code extensions using this comment format please inform me about it and I will update this. See contact information at the bottom of this document. I'm quite sure there are things I've missed.

CodeLicenseManager-source-updater-hash-comment

This updates source code with all license and other information within a

```
#
# ...
#
```

comment block. If the source file does not start with such a comment one is added otherwise the information is inserted within the existing comment, but first in the comment block.

This updater recognizes the following source code extensions:

- awk
- perl
- properties
- py
- ruby
- sh

If you know of other valid source code extensions using this comment format please inform me about it and I will update this. See contact information at the bottom of this document.

CodeLicenseManager-source-updater-html-xml

This updates source code with all license and other information within a

```
<!--
     ...
-->
```

comment block. If the source file does not start with such a comment one is added otherwise the information is inserted within the existing comment, but first in the comment block.

This updater recognizes the following source code extensions:

- ent
- htm
- html
- xhtml
- xml
- xsd
- xsl

If you know of other valid source code extensions using this comment format please inform me about it and I will update this. See contact information at the bottom of this document.

CodeLicenseManager-source-updater-jsp

This updates jsp pages with all license and other information within a

```
<%--
     ...
--%>
```

comment block. If the jsp file does not start with such a comment one is added otherwise the information is inserted within the existing comment, but first in the comment block.

8 Making Your Own ...

8.1 License Library

The "common-opensource" license library only contains reusable open source licenses. If you think it is missing some license that belongs there, please inform me. CLM is however not only for open source. You can make a license library with closed source company license(s). Some open source licenses are not reusable by others and can only be used by a certain organization. For such licenses a organization private license library needs to be created to use with CLM. Creating a license library is however quite easy.

Create the following structure:

```
codelicmgr/
    licenses/
```

In the "licenses" directory create a java property file having the name *licensename-licenseversion*.properties.

Here is an example of Apache-2.0.properties:

```
type=Apache
version=2.0
description=Apache Software License
source=open
sourceblock=${type}-${version}-SourceBlock.txt
fulltext=${type}-${version}-License.txt
```

It is more or less self explanatory. The "source" property can only have "open" or "closed" as value. The "sourceblock" and "fulltext" properties points to files containing the source block (usually called license boilerplate text) to put in source files, and the full license text. The "fulltext" property can also be an http url in which case no local fulltext needs to be available. Also note the "${type}" and "${version}" variables. They can only be used for the "sourceblock" and "fulltext" properties. The specified path to the text files are relative to the properties file. It is possible to put the texts in a subdirectory.

Do not indent the text in the file pointed to by "sourceblock"! CLM will handle any indents when the text is applied to source code.

Some licenses might be referenced with different names. For example the Apache license again is known as "Apache", "ASL", "Apache Software License". When maven is used and thirdparty licenses are automatically resolved the license reference in a pom might use any of the known names for a license. To support different names for a license, copy/duplicate the property file to cover the different names. For multi word names the spaces can be removed in the property file name. CLM will try to remove spaces from multi word license specifications and lookup the license again if it fails the first time.

When the property/properties file(s) and the referenced text files are in place, jar the codelicmgr directory. To use the license library the jar should be made available on the classpath or specified with --licenselibrary for the command line version.

8.2 Source Updater

Why would you want to make an own source code updater ? Well, you might work with some source files having a comment variant not supported by default by CLM. Or you might think that the output of the supplied CLM updaters really suck. You might want to follow an organizational standard layout. Making an own source updater gives you

more personalization than is possible with the execution options. Please note however that a source updater is not just configuration, it contains script java code (through beanshell) and if you implement errors/bugs CLM might fail when calling it. That is why an updater supplies maintainer and support information that is displayed on execution to make a clear line between CLM and the updater. I don't want to get bug reports on private updaters :-).

The simplest way to get started on making an own source updater is to copy an existing and modify it. But to create one from scratch do the following:

Create the following structure:

```
codelicmgr/
 sourcecodeupdaters/
```

In the "sourcecodeupdaters" directory you place a property file having the name of a source code extension (for example "java") and extension "properties". Example java.properties:

```
# The full name of the langugae.
lang.fullname=Java

# The property file relative directory of the scripts for updating
# source code of this language.
lang.scriptdir=scripts

# the properties file to read for general updater information.
updater.properties=updater
```

The last property "updater.properties" points to another property file containing general information relevant for all <lang extension>.properties files. It can have any name, but I have used updater.properties:

```
# A description of the updater.
updater.description=CodeLicenseManagers /* ... */ comment block multilanguage
source code updater.

# A copyright message (optional).
updater.copyright=

# The name and email of the maintainer.
updater.maintainer=Maintained by Tommy Svensson (opensource@biltmore.se)

# Where to turn for support.
updater.support=For support goto http://sourceforge.net/projects/codelicmgr/support
or contact maintainer.
```

If the "updater.copyright" property is specified the copyright message will be displayed on updater execution.

As said above this information is to make it clear who is responsible for the specific updater.

Copy/duplicate the *langextension***.properties file to any valid extensions for the updater.**

When CLM sees a "whatever.java" file and want to update it, it looks for a codelicmgr/sourcecodeupdaters/java.properties file in the classpath and as per normal classpath resource lookup will use the first found. So if you have several source updater libraries in the classpath supporting the same language the order they are provided are important. CLM itself for example uses the "annotation" updater for java but the "slashstar" updater for bsh scripts. The "slashstar" updater also supports java so the "annotation" updater is placed before the "slashstar" updater in the classpath.

The "lang.scriptdir" property points to a subdirectory containing the standard scripts for an updater. Always use a subdirectory for the scripts! Never specify "lang.scriptdir=.". The "lang.scriptdir" directory contains the following scripts:

init.bsh

This script is only run once when an updater is loaded. It should setup things needed by the other scripts, like imports and support functions. Here is an example:

```
    import java.util.*;
    import java.text.*;


    boolean findInsertPosition(searchStrings) {
        String endOfSection = "findEndOfBlock();";
        String endOfCommentSearch = "^[^ ^\\*].*|^ \\*/.*|^[a-zA-Z &#\\*][a-zA-Z &#][a-
zA-Z &#].*";
        return findInsertPosition(endOfSection, endOfCommentSearch, "/* ", " * ", " */",
searchStrings);
    }

    /**
     * Support function for finding the end of a specific comment block.
     */
    void findEndOfBlock() {
        boolean done = false;
        while (!done) {
            editor.moveDown(1);
            String line = editor.getLine();
            if (line.trim().equals("*/") ||
                (line.startsWith(" * ") && line.length() >= 4 && line.charAt(3) != ' ')
||
                editor.isOnLastLine() ||
                line.matches("^[a-zA-Z &#][a-zA-Z &#][a-zA-Z &#].*")) {
            done = true;
            }
        }
        editor.moveUp(1);
        editor.moveToEndOfLine();
    }
```

updateAuthorsBlock.bsh

This will only be run if "updateAuthorsBlock" is set to true in the "codeOptions". It will add author information if and only if it is not already available. Once added this is never modified. So this only provides initial information after it is added it need to be manually maintained. This sections also list changes done by each author. When this block is added information about the user is read from .clm-user.properties in the users home directory. See the "User specific configuration" section above for more information. The user and email is provided to the script by CLM. Here is an example:

```
    if (!editor.find(" \\* AUTHORS")) {

        findInsertPosition(
        "find=first;insert=after;section=true;search='^ \\* LICENSE *'," +
        "find=first;insert=after;section=true;search='^ \\* COPYRIGHTS *'," +
        "find=first;insert=after;section=true;search='^ \\* PROJECT *'," +
        "find=first;insert=before;section=false;search='^package.*'," +
        "find=first;insert=before;section=false;search='^/\\*.*'"
        );

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        display("Adding AUTHORS");
        editor.insertLine(" * AUTHORS");
        editor.insertLine(" *     " + userName + " (" + userEmail + ")");
        editor.insertLine(" *         Changes:");
        editor.insertLine(" *         " + sdf.format(new Date()) + ": Created!");
        editor.insertLine(" *         ");
    }
```

updateCopyrightInfo.bsh

This will only run if "updateCopyrights" is set to true in the "codeOptions". It will add copyright information to a source file if it is not already available or update the existing copyright information (easiest accomplished by removing old and inserting new).

This differs from other scripts since there can be more than one copyright and looping through the copyrights are handled by CLM, so the "updatecopyrightInfo.bsh" script must define a set of functions explained by the following example:

```
    /**
     *   setup
     *       Does whatever setup is required, like finding and removing an
     *       old copyright block.
     */
    void setup(TextFileEditor editor) {
    if (editor.find(" \\* COPYRIGHTS")) {
        display("Updating COPYRIGHTS");
        editor.startSelection();
        findEndOfBlock(); // Defined in init.bsh!
        editor.endSelection();
        editor.deleteSelection();
        editor.moveUp(1);
    }
    else {
        display("Adding COPYRIGHTS");

        findInsertPosition(
            "find=first;insert=after;section=true;search='^ \\* PROJECT'," +
            "find=first;insert=before;section=true;search='^ \\* LICENSE'," +
            "find=first;insert=before;section=true;search='^ \\* AUTHOR'," +
            "find=first;insert=before;section=false;search='^package.*'," +
            "find=first;insert=before;section=false;search='^/\\*.*'"
        );

    }
    editor.insertLine(" * COPYRIGHTS");
    }


    /**
     *   startCopyrights
     *       This will only be called if there are more than one copyright.
     */
    void startCopyrights(TextFileEditor editor) {
    }

    /**
     *   forEachCopyright
     *       This will be called once for each copyright specification. last
     *       will be true for the last entry.
     */
    void forEachCopyright(TextFileEditor editor, String year, String holder,
        String rights, boolean last) {
        editor.insertLine(" *     Copyright (C) " + year + " by " + holder + " " +
rights);
    }

    /**
     *   endCopyrights
     *       This will only be called if there are more than one copyright.
     */
    void endCopyrights(TextFileEditor editor) {
    }

    /**
     *   finish
     *       Adds anything that needs to be added efter each copyright is processed.
     */
    void finish(TextFileEditor editor) {
        editor.insertLine(" *     ");
    }
```

updateImports.bsh

This is always run, but for most updaters it will be empty. It should add any required imports. The 2 updaters using annotations use this to add imports for the annotations.

updateLicenseInfo.bsh

This is only run if "updateLicenseInfo" is set to true in "codeOptions". It adds or updates license information in the source file. Here is an example:

```
    TextBuffer getLicenseTextAsTextBuffer() {
        TextBuffer buffer = new TextFileBuffer();
        buffer.addLine(" * LICENSE");
        buffer.addLine(" *     " + licenseType + " " + licenseVersion + " (" + source +
" Source)");
        buffer.addLine(" *     ");

        formatMultiLineTextToCode(buffer, sourceBlock, " * ", "", "", "", 4, false);

        buffer.addLine(" *     ");

        return buffer;
    }

    if (editor.find(" \\* LICENSE")) {
        display("Updating LICENSE!");
        editor.startSelection();
        findEndOfBlock(); // Defined in init.bsh!
        editor.endSelection();
        editor.replaceSelectionWithTextBuffer((TextBuffer)getLicenseTextAsTextBuffer());
    }
    else {
        display("Adding LICENSE!");
        editor.moveToTopOfFile();

        if (!findInsertPosition(
        "find=first;insert=after;section=true;search='^ \\* COPYRIGHTS *'," +
        "find=first;insert=after;section=true;search='^ \\* PROJECT *'," +
        "find=first;insert=before;section=true;search='^ \\* AUTHOR *'," +
        "find=first;insert=before;section=false;search='^package.*'," +
        "find=first;insert=before;section=false;search='^/\\*.*'"
        )) {
            editor.insertLineAbove(" * ");
        }

        editor.insertBuffer(getLicenseTextAsTextBuffer());
    }
```

updateProjectInfo.bsh

This is only run if "updateProjectInfo" is set to true in "codeOptions". It adds or updates project information in the source file. Here is an example:

```
    if (editor.find("^ \\* PROJECT")) {
        display("Updating PROJECT");
        editor.startSelection();
        findEndOfBlock();
        editor.endSelection();
        editor.deleteSelection();
        editor.moveUp(1);
    }
    else {
        display("Adding PROJECT");

        findInsertPosition(
        "find=first;insert=before;search='^ \\* COPYRIGHTS *'," +
        "find=first;insert=before;search='^ \\* LICENSE *'," +
        "find=first;insert=before;search='^ \\* AUTHOR *'," +
        "find=first;insert=before;section=false;search='^package.*'," +
        "find=first;insert=before;section=false;search='^/\\*.*'"
        );
```

```
    }

    editor.insertLine(" * PROJECT");

    editor.insertLine(" *     Name");
    editor.insertLine(" *         " + projectName);
    editor.insertLine(" *     ");

    if (hasProjectSubProjectOf) {
        editor.insertLine(" *     Subproject Of");
        editor.insertLine(" *         " + projectSubProjectOf);
        editor.insertLine(" *     ");
    }

    if (hasProjectCodeVersion) {
        editor.insertLine(" *     Code Version");
        editor.insertLine(" *         " + projectCodeVersion);
        editor.insertLine(" *     ");
    }

    if (hasProjectDescription) {
        editor.insertLine(" *     Description");
        TextBuffer buffer = new TextFileBuffer();
        formatMultiLineTextToCode(buffer, projectDescription, " * ","", "", "", 8,
true);
        editor.insertBuffer(buffer);
        editor.insertLine(" *         ");
    }
```

userBefore.bsh / userAfter.bsh

These are optional and are only executed if they are available. userBefore.bsh will execute after init.bsh but before any other scripts. userAfter.bsh will execute after all other scripts.

These are not very useful as part of an updater. These are intended for extending an existing updater. Place the script(s) in the same path as the updater you want to extend, and place it before the extended updater in the classpath. When the command line version is used the order of updaters specified with --sourceupdaters is the same order they are added to the classpath.

When all properties and scripts have been provided, jar from the "codelicmgr" directory.

9 Standard functions and objects available to scripts

The following objects are available to this script:

editor : TextFileEdtior

A TextFileEditor instance with the current source code file loaded. You are not allowed to do load(file) or save() on it! If you feel the need to do that then you are probably doing something strange and should reconsider what you are doing :-). All other methods on this object can be called.

The TextFileEditor is part of the FileEditor tool available at http://github.com/tombensve/FileEditor.

projectName : String

The name of the project.

hasProjectDescripion : boolean

True if a project description has been provided.

projectDescription : String

The projectDescription if 'hasProjectDescription' is true.

hasProjectCodeVersion : boolean

True if a project code version has been provided.

projectCodeVersion : String

The version of the project code if 'hasProjectCodeVersion' is true.

hasProjectSubProjectOf : boolean

True if the 'subProjectOf' config has been provided.

projectSubProjectOf : String

The name of the parent project the project is a subproject of if 'hasSubProjectOf' is true.

licenseType : String

The type of the license. Example: "Apache" or "GPL".

licenseVersion : String

The version of the license. Example "2.0" or "v3".

licenseDescription : String

A short description of the license, usually just the full name of the license. Provided by license library.

source : String

"Open" or "Closed" depending on license type.

sourceBlock : String

The license notice to put at the top of the source file. Sometimes called the license boilerplate. Use the formatMultiLineTextToCode(...) function with this.

--

The following functions are available:

display(String str)

str - The string to display. This will only be displayed if verbose is turned on!

formatMultiLineTextToCode(TextBuffer buffer, **String text**, **String lineEnd**, **final String stringSpec**, **final String escapedStringSpec**, **int indent**, **boolean trim)**

buffer - The buffer to put formatted result in.

text - The text to format.

lineEnd - The string that terminates each line. For example " +" or ",".

stringSpec - A character or set of characters that indicates start and end of a String. For example """ or "'".

escapedStringSpec - A character or set of characters that indicates an escaped start and end of a string. For example "\"" or "\'"

indent - The number of spaces to indent each line.

trim - If true each line is also trimmed.

formatMultiLineTextToCode(TextBuffer buffer, **String text**, **String lineBeg**, **String lineEnd**, **final String stringSpec**, **final String escapedStringSpec**, **int indent**, **boolean trim)**

buffer - The buffer to put formatted result in.

text - The text to format.

lineBeg - The beginning of each line. The indent will be added after this.

lineEnd - The string that terminates each line. For example " +" or ",".

stringSpec - A character or set of characters that indicates start and end of a String. For eample """ or "'".

escapedStringSpec - A character or set of characters that indicates an escaped start and end of a string. For example "\"" or "\'"

indent - The number of spaces to indent each line.

trim - If true each line is also trimmed.

The following functions makes use of the current editor instance, but they receive it automatically, it does not have to be specified!!

moveToEndOfSection(char starting, **char ending)**

Use this when you can specify the end of a section (project, copyright, license, author) with a starting character that end with a matching ending character at the same level. For example '(' and ')'.

moveToEndOfSection(String startsWith, noSpaceAt)

Use this when each section starts with "comment-char + space + section-heading" and all text in the section are indented with at least one more space than the heading. This actually finds the next section or end of comment to determine the end of a section.

startsWith - The line indicating a new section must start with this.

noSpaceAt - The line indicating a new section must not have a space at this position.

findInsertPosition(String endOfSection, String endOfCommentSearch, String commentStart, String commentMiddle, String commentEnd, String searchStrings)

This should be used if a specific section is not already available in the file to find an optimal position to insert the section in.

Please note that this function actually returns a boolean and will be false if none of the specified matches came true. This information is mostly useful in updateLicenseInfo.bsh which is the first section update script run and might require an editor.insertLineAbove("# "); (or whatever comment character is relevant) if this returns false.

endOfSection - This is a string with a call to any function that will move the end of a section. eval() will be done on this string when it is needed. Specify one of the above 2 functions or provide your own in init.bsh.

endOfCommentSearch - This is only needed if you specify the "belowComment" attribute in searchStrings (see below). If that attribute is specified and this value is not null then 'commentStart' and 'commentEnd' will be used to identify a top of file comment, usually a class javadoc comment for java. This value specifies a regular expression that gets passed to a special version of editor.find(...) and that will stop the search if the current line matches this regular expression. This is to avoid finding the wrong comment if no comment is available at the top of the file (i.e class comment). This used by the java and groovy source updaters that uses annotations instead of comments. It is also used when all searchString:s fail to identify a comment to position in. If that also fails a new comment will be created, but only if all 3 comment sections are non null and non blank.

commentStart - A string that indicates the beginning of a comment block. If it is a comment type that has no start or end just specify the same value for all 3 comment specifications. The "belowComment" attribute of searchStrings will also make us of this, and in this case it can be a regular expression since it will never be used to create a comment in that case. In all other cases it cannot be a regular expression!

commentMiddle - A string that indicates the middle of a comment.

commentEnd - A string that indicates the end of a comment. The "belowComment" attribute of searchStrings will also make us of this, and in this case it can be a regular expression (see start comment).

searchStrings - This specifies a set of comma separated search expressions with semicolon separated attributes used to find a position. The first that matches will stop the search and the function will return with the editor at that position, ready for insert. It has the following format: attribute=value;...;attribute=value, ... ,attribute=value;...;attribute=value

The following attributes are available:

find - first (default) or last. If first the first match found will be used. If last the last match found will be used.

insert - before or after (default). If before the insert position will be before the matched line. If after the insert position will be after the matched line.

section - true or false (default). If true the specified match is the start of a section and the endOfSection value will be used if insert==after.

belowComment - true or false (default). If true then "commentStart","commentEnd", and "endOfCommentSearch" will be used to move the insert position down below a possible comment.

newLine - true or false (default). If true a new empty line will be inserted at the insert position leaving an empty line between the match and the new insert position.

search - This is a regular expression search string surrounded by ''. Example: '^# LICENSE'. This attribute must always be specified.

10 Special note to users of version < 2.0.

A previous version of this utility exists on http://codelicmgr.sf.net/. As of version 2.0 this has moved to github and also changed package from *se.biltmore...* to *se.natusoft...* The package change was required due to a bad decision by me to use the package name of a company my company was part owner of, but in the end didn't work out as intended and I left. Thereby I have changed package to my own company name.

If you used the previous version then a search and replace of

```
<groupId>se.biltmore.tools.codelicmgr</groupId>
```

to

```
<groupId>se.natusoft.tools.codelicmgr</groupId>
```

in your pom(s) should be enough. Some new options have been added and nothing have been removed.

11 Licenses

11.1 Project License

Apache Software License version 2.0

11.2 Third Party Licenses

Lesser GNU Public License version v3

The following third party products are using this license:

- bsh-1.3.0

Apache Software License version 2.0

The following third party products are using this license:

- FileEditor-2.0
- OptionsManager-2.0
- MarkdownDoc-1.0

12 License Texts

12.1 Apache Software License version 2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of,

publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or

implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

12.2 Lesser GNU Public License version v3

GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU

General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.