Xolti Documentation

Release stable

Jun 05, 2017

Contents

Contents:

CHAPTER 1

Introduction

Xolti is a tool assisting you to manage license headers in source files, powered by Ruby.

Xolti is licensed under the terms of the GNU-GPL3, meaning it's free software.

You can find the source code in the Github repository

Xolti is still in developer preview.

Purpose

If you ever had to manage the license of a project, you probably know how much a burden this task can become, especially if you try to keep it accurate among numerous files!

Xolti is a piece of software aiming to assist the management of license-related information in a project. Its main functionality is to add and maintain license headers at the top of source files. Moreover, Xolti can also generate `LICENSE` files.

CHAPTER 2

Installation

Xolti is available on *RubyGems.org*, the Ruby community's gem hosting service. Assuming you have already installed Ruby, this makes Xolti fairly easy to install :

```
gem install xolti
```

Once the installation completes, `xolti` is added to your `$PATH`, so you can access it from everywhere.

Requirements

In order to properly work, Xolti requires Ruby to be installed. It has been tested with Ruby >= 2.1, but probably works with slightly older versions too. In addition, and as stated in its GemFile, Xolti requires Thor, a ruby gem used to easily create command line interfaces.

Building from sources

You can also create the gem from the source files, using the following commands (assuming you are in the project root):

```
gem build xolti.gemspec
```

You can then install it with :

```
gem install xolti-[VERSION].gem
```

where [VERSION] must be replaced by the current version of Xolti.

Running the tests

Running the tests requires the gem `test-unit`, and can be achieved with the command :

```
ruby test/ts_suite.rb
```

Alternatively, if you have installed `rake` in addition to `test-unit`, you can use the command :

```
rake test
```

CHAPTER 3

Getting started with a simple example

Let's suppose you have a project, for example in NodeJS, with files organized like this :

```
|- myAwesomeProject
|- app.js
|- package.json
|- node_modules
        |- ...
```

Now, how do you use Xolti to manage your license ?

Installation

If not done already, this is the time to install Xolti. You can find detailed explanations in the installation section of this documentation, but running this command is probably sufficient:

```
gem install xolti
```

Initiating the project

Then it's time to create the xolti configuration file, named `xolti.yml`. This file will contain information for Xolti such as the name of your project and the license you have chosen.

You can do this by running this command:

```
xolti init
```

This will trigger a command line utility asking you some information, and will create a `xolti.yml` for your project based on what you answered.

```
remi ~/myAwesomeProject]$ xolti init
Initiating xolti project
name (myAwesomeProject):
author: Rémi Even
license (GPL3.0):
```

Values between parenthesis will be selected if you do not type anything.

Creating a LICENSE file

Before adding license headers to your source file, you probably want to generate a `LICENSE` file, which will be in the root of your project, and will contain the complete text of the license you have chosen. To do so, use the following command :

```
remi ~/myAwesomeProject]$ xolti generate-license
Created the LICENSE file (GPL3.0)
```

Telling Xolti which files to modify (or not)

Similarly to git and the `.gitignore` file, you can create a `.xoltignore` file to indicate xolti files to ignore. The syntax is the same; for this project, the content of the xoltignore would be :

```
node_modules
package.json
```

Tip: More details on `.xoltignore` files can be found *in the dedicated documentation page*

Checking which files are missing headers

Now that xolti knows which files to handle, we can ask it which ones are missing headers. We can either use the dedicated command:

```
[22:14 remi ~/myAwesomeProject]$ xolti list-missing
Files missing (proper) header:
app.js
```

... or use `xolti status`, which will report the files without correct headers.

```
xolti status
-- ./app.js
No header found.
```

Adding the header to your files

Looks like `app.js` is missing a header... Xolti can create and insert one for you, with the `add` command:

```
xolti add app.js
```

Tip: We could have also used `.` instead of specifying `app.js`; xolti would have add a header in each file (recursively) from the current folder.

Note: Xolti detects, based on its extension, that the `app.js` file contains Javascript. This allows Xolti to know how to create a comment in this file (in this case, with `/*`, `*` and `*/`).

Verifying the result

Of course, you can verify that Xolti have actually added the header by simply opening the file, but you can also use once again the `status` command:

```
remi ~/myAwesomeProject]$ xolti status app.js
Correct header.
```

That's it ! Your project is correctly licensed :).

Detecting incorrect headers

Now that we think of it, `myAwesomeProject` is not such a good name. `myFantasticProject` is way better ! To let xolti know of our change of mind, we can edit the `xolti.yml` file, and replace the value of the key `project_name` by `myFantasticProject`.

When we ask xolti once again about the status of the `app.js` file, it warns us about the now incorrect header:

```
remi ~/myAwesomeProject]$ xolti status app.js
Line 5: expected "myFantasticProject" but got "myAwesomeProject".
Line 7: expected "myFantasticProject" but got "myAwesomeProject".
Line 12: expected "myFantasticProject" but got "myAwesomeProject".
Line 18: expected "myFantasticProject" but got "myAwesomeProject".
```

You can then manually correct this outdated header.

Deleting the header in a file

What if you decide that you no longer needs a header in your `app.js` ? Simply use the `delete` command:

```
xolti delete app.js
```

CHAPTER 4

Xoltignore reference

In the same fashion git uses `.gitignore` files to know which files to track or not, xolti uses `.xoltignore` files to detect which files needs a header or not.

You can create one `.xoltignore` in each directory of your project.

Syntax

`.xoltignore` files are plain text files, using a sub-set of the syntax of `.gitignore` :

- Each line specifies a pattern used to ignore or not a path
- A line is ignored if it is blank or starts with `#`
- Globing (use of `*` and/or `**` wildcards) is supported
- A pattern can be inverted by prefixing it with `!`
- Pattern are read line by line, from top to bottom; lower rules override higher ones, and rules from a deeper folder override rules from higher folders
- A pattern ending with `/` matches only directories

Example : js project

Folders/files structure :

```
|- Javascript_Project
|- app.js
|- package.json
|- node_modules
        |- ...
```

Possible `.xoltignore` :

```
# Ignore files installed by npm
node_modules

# Ignore package.json
package.json
```

Example : java maven project

Folders/files structure :

```
|- Java_Maven_Project
|- pom.xml
|- readme
|- src
        |- main
        |- java
                    |- App.java
```

Possible `.xoltignore` :

```
# Ignore all files but pom.xml and java sources
*
!pom.xml
!src/**/*.java
```