Living documentation

Table of Contents

Living Documentation

This project show some techniques to extract documentation from your project like:

- Extract code documentation (javadoc)
- Use annotation
- Extract xml information
- Execute code to find values
- Formating documentation
- Generate graph

It’s also present use cases like:

- Create a glossary
- Retrieve good example from code
- Visualize workflow
- Document tests
- Visualize dependencies
- Create a release note

Getting started

Prerequisites

- JDK 11
- Maven
- Docker

This project is auto-documented. To generate full documentation, just call

```
. ./generateDoc.sh
```

When it’s done, you can find documentation in ./target/doc/demo.html.

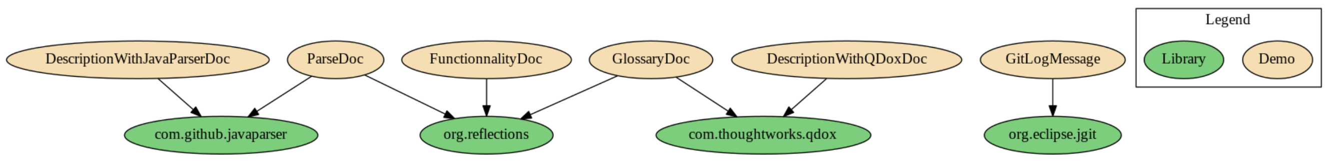
Library dependencies

In these demos, we use libraries:

- com.thoughtworks.qdox
- org.reflections
- com.github.javaparser

- org.eclipse.jgit

The graph below shows which libraries is used in demos.



Available demos

List of demo classes available in this project.

Each demo is a simple program that extract some documentation from the code. It illustrates a use case or a technique. It contains a 'main' to make it a standalone application to be able to see what is generated and to execute it independently. We try to not use utilities classes to keep all generation code into a single class to have all information necessary to reproduce example

These demonstrations are minimalist. They just show what is possible to do but may not worked on more generic cases.

Annotation

Annotated method demo

Example 1. Functionnalities to document.

findAnnotatedMethod (ClassToDocument): Find all method with a specific annotation.

functionnalityToDocument (FunctionnalityDoc): Living Documentation

Glossary demo

Example 2. Glossary generated

City
Description of a city.

Person
A physical person.

Change log

Extract changelog from git messages

Example 3. Git history:

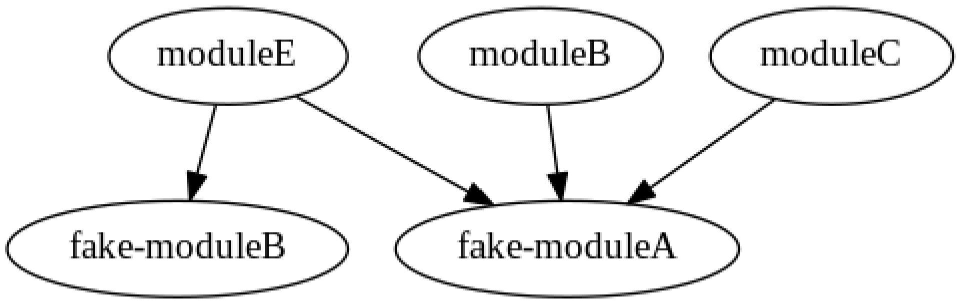
- **21/11/2019** (7dc8ed7417): Reorganized project, add style, generate full and light doc
- **20/11/2019** (634bf44e48): Reorganize demo
- **20/11/2019** (831d425b5c): Add sequence diagram with plantuml
- **19/11/2019** (e64c84b03d): Add example of running a shell command to rerieve informations
- **14/11/2019** (f139fe0d62): Begin to execute Maven command
- **14/11/2019** (fd63952379): Improve workflow documentation
- **14/11/2019** (353117a3f2): Clean some useless examples
- **14/11/2019** (55923159b4): Inprove documentation parsing code
- **13/11/2019** (710cd00ab7): Inprove documentation from execution
- **13/11/2019** (5069669ce9): Inprove documentation from qdox

Include a changelog file

Execute to get information

Execute maven command

Example 4. Dependencies executing maven command



Get information executing code.

Example 5. Default values of getter methods

Default values of Configuration class

Field	Default value
isVerbose	false
getVersion	5.2

Show methods called

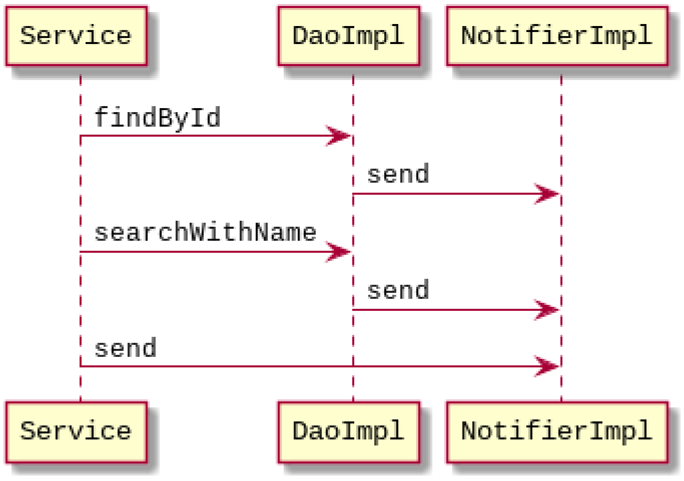


Figure 1. Calls from Service.findHomonyms method

Show workflow from code

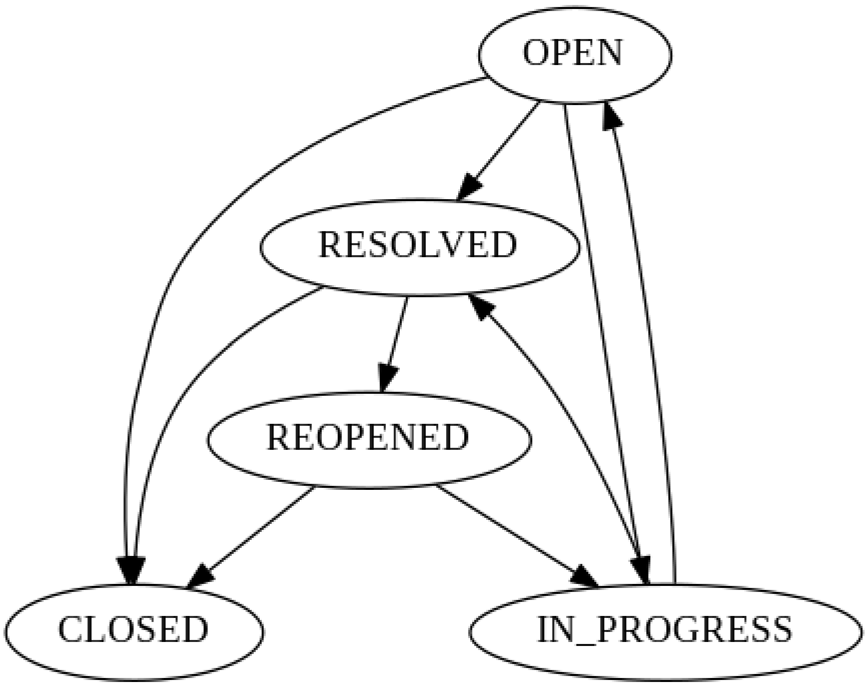


Figure 2. Workflow graph generated

Extract javadoc

JavaDoc with JavaParser

Example 6. Javadoc extracted from class with a parser

org.dojo.livingdoc.application.ClassToDocument: Class to show a javadoc extraction.

- main: Starting point of the application.
- simpleMethod: Simple method documented.
- findAnnotatedMethod: No description.

JavaDoc with QDox

Example 7. Javadoc extracted from class with QDox

ClassToDocument: Class to show a javadoc extraction.

- main: Starting point of the application.
- simpleMethod: Simple method documented.
- findAnnotatedMethod: null

Static analysis

Extract a code fragment

Example 8. Include a fragment of code

Best practice to follow

```
public void doNothing() {
    // Really interesting code.
}
```

Extract imports parsing code

Example 9. Parse code to extract information

- ParseDoc
  - com.github.javaparser.ast
  - com.github.javaparser.ast.visitor
  - com.github.javaparser.utils
  - org.dojo.livingdoc.annotation
  - org.reflections
- DescriptionWithJavaParserDoc
  - com.github.javaparser.ast
  - com.github.javaparser.ast.body
  - com.github.javaparser.ast.comments
  - com.github.javaparser.ast.visitor
  - com.github.javaparser.javadoc.description
  - com.github.javaparser.utils
  - org.dojo.livingdoc.application
  - org.dojo.livingdoc.annotation
- ExecuteDoc
  - org.dojo.livingdoc.application
  - org.dojo.livingdoc.annotation
- …

Extract information from pom.xml

Example 10. Content of tag 'description' from pom.xml

Demo of living documentation