MAL

Nikolaos Kakouros

Oct 05, 2020

CONTENTS:

CHAPTER

ONE

INTRODUCTION

CHAPTER

TWO

WELCOME TO THE MAL-DOCUMENTATION WIKI!

CHAPTER

THREE

EXAMPLELANG

exampleLang is a MAL language intended to demonstrate the Maven project structure of a MAL language.

This project has the following structure:

- The file `pom.xml` is the Maven configuration file of the project.
- The directory `src/main/mal` contains the MAL specification `exampleLang.mal`, which is the MAL specification of exampleLang.
- The directory `src/main/resources/icons` contains SVG icons for the assets in exampleLang.
- The directory `src/test/java/org/mal_lang/examplelang/test` contains the unit tests of exampleLang.

3.1 Apache Maven

Apache Maven is a build tool and dependency management tool for Java projects. You can read more about Maven at https://en.wikipedia.org/wiki/Apache_Maven. Follow the instructions at https://maven.apache.org/download.cgi to download Maven, and follow the instructions at https://maven.apache.org/install.html to install Maven.

3.2 Building exampleLang and running the unit tests

The MAL compiler compiles MAL specifications (`.mal` files) into different formats, using different backends. The reference backend generates Java code that is suitable for testing purposes and evaluating your language. The securiCAD backend generates a `.jar` file that can be used with foreseeti's products, including securiCAD, which is a tool that can be used to graphically create models using your language and to simulate attacks on those models.

3.2.1 Building with the reference backend and running the unit tests

To compile exampleLang with the reference backend of the MAL compiler and then run the unit tests, execute the following command from the `exampleLang` directory:

```
mvn test
```

This will invoke the MAL compiler's reference backend to generate `.java` files under `target/generated-test-sources`. These `.java` files and the unit tests in `src/test/java` will then be compiled into `.class` files under `target/test-classes`. The unit tests will then finally be executed.

To only compile exampleLang into `.java` files, execute the following command:

```
mvn generate-test-sources
```

To compile exampleLang into `.java` files and then compile these `.java` files and the unit tests in `src/test/java` into `.class` files, execute the following command:

```
mvn test-compile
```

To run a specific test class, execute the following command:

```
mvn test -Dtest=TestExampleLang
```

Where `TestExampleLang` is the test class.

To run a specific test method in a test class, execute the following command:

```
mvn test -Dtest=TestExampleLang#testNoPassword
```

Where `TestExampleLang` is the test class and `testNoPassword` is the test method.

3.2.2 Building a securiCAD compatible .jar file

To build a securiCAD compatible `.jar` file, you need access to foreseeti's maven repository. To request access, please contact support@foreseeti.com. When you have received your credentials, you can store them in a file `~/.aws/credentials` (`%UserProfile%\.aws\credentials` on windows). For example:

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

To compile exampleLang with the securiCAD backend of the MAL compiler, execute the following command:

```
mvn package -PsecuriCAD
```

The resulting `.jar` file will be located in `target/examplelang-1.0.0.jar`.

If you don't want to run the unit tests when building a securiCAD compatible `.jar` file, execute the following command:

```
mvn clean package -PsecuriCAD -Dmaven.test.skip=true
```

3.3 Using exampleLang as a template MAL language

To create a new language using exampleLang as a template, you need to do the following:

- Create a new MAL language project using exampleLang as the template
  - `cp -r exampleLang/ myLang/`
- Enter the directory of the new MAL language project
  - `cd myLang/`
- Remove build scripts
  - `rm -rf .buildscript`
  - `rm .travis.yml`

- Update `LICENSE` with a license of your choice
  - Update copyright notices to reflect your license in
    * `NOTICE`
    * `pom.xml`
    * `src/main/mal/exampleLang.mal`
    * `src/test/java/org/mal_lang/examplelang/test/*.java`
- Update `README.md` with relevant information about your language. Information about how to use Markdown can be found at https://help.github.com/en/articles/basic-writing-and-formatting-syntax.
- Update `pom.xml` to reflect your project
  - Update `<groupId>` with a reverse domain name that you can use
    * Example: `com.example`
  - Update `<artifactId>` with a suitable name
    * Example: `mylang`
  - Update `<version>` with the version of your language
    * Example: `1.0.0`
  - Update `<name>` with the name of your language
    * Example: `myLang`
  - Update `<mal.file>` with the name of the main MAL specification of your language
    * Example: `myLang.mal`
  - Update `<mal.securicad.package>` with the package name of your language
    * Example: `com.example.mylang`
  - Update `<mal.reference.package>` with the test package name of your language
    * Example: `com.example.mylang.test`
- Rename `src/main/mal/exampleLang.mal` to the name of the main MAL specification of your language
  - `mv src/main/mal/exampleLang.mal src/main/mal/myLang.mal`
- Update your main MAL specification's `#id` and `#version`
  - Example: `#id: "com.example.mylang"`, `#version: "1.0.0"`
- Rename unit tests in `src/test/java` to reflect your language
- Change the package name of the unit tests to the test package name of your language
  - Example: `package com.example.mylang.test;`

3.4 License

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search