Contents

[Requirements 2](#_Toc481678738)

[Define de Swagger API into a YAML file 3](#_Toc481678739)

[Generate all layers of an OASP4J project from that YAML file 5](#_Toc481678740)

[Include the validations defined at the API into the generation 7](#_Toc481678741)

[Problems 9](#_Toc481678742)

[Deadlocks 10](#_Toc481678743)

# Requirements

* Define Swagger API into a YAML file.
* Generate all layers of an OASP4J project from that YAML file.
* Include the validations defined at the API into the generation.

# Define de Swagger API into a YAML file

Using the Swagger Editor (<http://editor.swagger.io/#/>) is the easiest and faster way to define the API.



It would be a good idea include a swagger API definition YAML file at the root of the core project at the OASP4J archetype. That way, developers only need to copy/paste and replace model names.

[YAML FILE](https://github.com/EastWindShak/swagger4OASP4J/blob/master/devonfw/core/src/main/resources/devonfw.yaml)

The file also includes a definition with all usage examples of the type properties definitions.

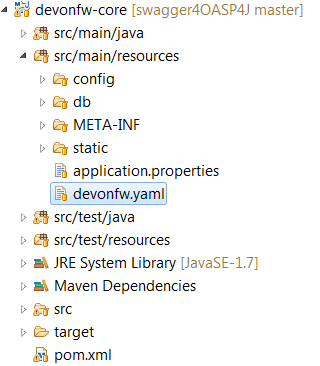


Figure 1: YAML at resources

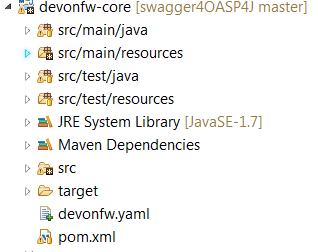
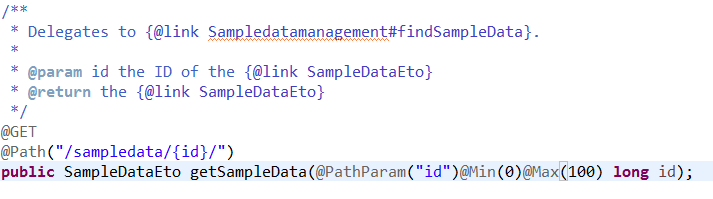


Figure 2: YAML at core root

# Generate all layers of an OASP4J project from that YAML file

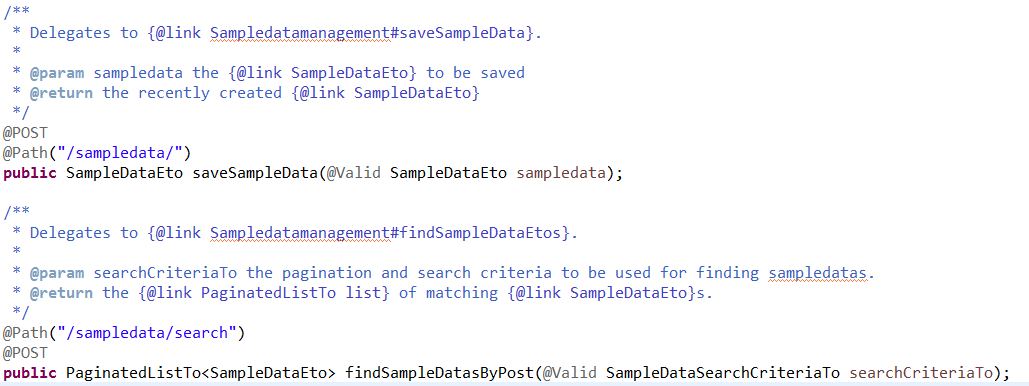
1. Implement new CobiGen plugin with the input reader that reads the YAML file.
   1. YAML Parser:
      1. [SnakeYAML](https://bitbucket.org/asomov/snakeyaml/wiki/Documentation)
      2. [YamlBeans](http://yamlbeans.sourceforge.net/)
      3. [JYaml](http://jyaml.sourceforge.net/)
   2. Freemarker Model
      1. Following this structure as a first approach:
2. Adapt cobigen-core
   1. A recursion is needed. Generate/merge a set of templates for entity entry.
3. Design new wizard
   1. First page can show 3 options as radio buttons
      1. Generate server
      2. Generate client
      3. Generate server and client
   2. Second page differs depending of the previous choose
      1. In case that an option with client generation has been chosen, choose between Angular2, ExtJS and Sencha Architect as checkboxes. Next page would be the first page of the existent wizard for the server generation.
      2. In case of only server selected, show the directly the existent first page of the wizard.
      3. The last page is the optional one that allow to check/uncheck fields to include at the generation.

# Include the validations defined at the API into the generation

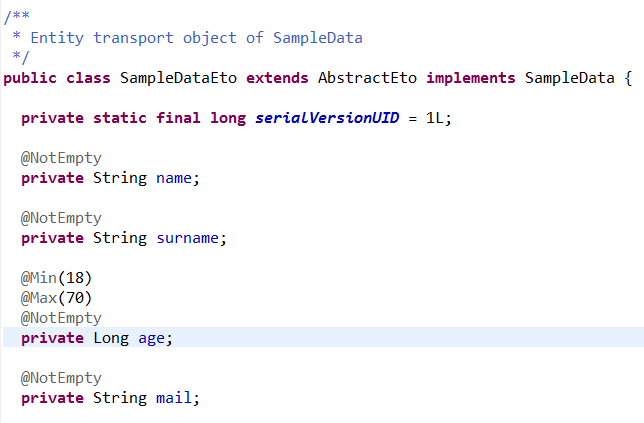


The validation *required=true* is not needed to be added because is obviously needed at the path. OASP4J also handles the exception in case of empty id value.

The save and search services need to have a valid ETO and SearchCriteria objects respectively



To have this objects validated, the validations must be done at the objects themselves:



# Problems

1. I searched over the [Swagger Docs](http://swagger.io/docs/) and I have not been able to find a way to define the validations for the @Valid annotations over the definition swagger object.
2. Re-design the wizard and core can take a lot of time and must be done carefully to avoid to break the existent functionality.
3. Re-structuration of the templates project to fit the new wizard design.
4. I was not able neither to find a list of validations that can be used depending of the object type of the service parameters over the documentation.

# Deadlocks

* InputReader can take at first approach about 3-4 weeks having into account I have to be at 50% at My Thai Star project and sporadic (for the moment) tasks at Daimler FEBE.
* I can’t estimate a deadlock for wizard and core changes because I am not really aware about the needed tasks for them.