Un générateur de code pour *OpenPS*

1. **Introduction** :

OpenPS est une spécification open source pour les APIs web qui utilisent le modèle publication/abonnement. Pour générer les codes clients pour ces APIs nous avons besoin de mettre en place une approche afin de s'assurer que ces API répondent aux exigences définies dans leurs spécifications. Cette approche consiste en un générateur de code automatique pour une plate-forme cible (RabbitMQ) avec un langage de programmation cible (Java). Pour cela nous nous sommes inspirés de celui de OpenAPI. Et avions deux choix, soit utilisé le générateur OpenAPI, soit la personnalisé. Cette personnalisation consiste au minimum en quatre (4) ensemble de fichiers : Codegen file, SPI registration, template et config file. Ce générateur utilise un document de spécification OpenPS en YAML. Il permet la génération automatique de code permettant à deux entités (Agent et Controller) de communiquer à travers un broker (RabbitMQ). Cette communication utilise des fonctions pub/sub générer (publish et subscribe). Il implémentant un modèle de base de communication adopté par Agent et Controller décrit par les templates.

Le but de ce chapitre, est de générer un code d’implémentation d’une API décrit avec les dependencies pour générer et concevoir une API locale basées sur le format de documentation OpenPS.

1. **Codegen file** : Swagger codegen

Swagger Codegen est un projet open source qui permet de générer automatiquement des bibliothèques clientes d'API, génération de SDK, des stubs de serveur et de la documentation à partir d'une spécification OpenAPI. C’est outil apporté parmi d’autre (Swagger Editor, UI) pour la conception et une documentation d'API intégrées, conçues pour les équipes d'API travaillant avec la spécification Swagger (OpenAPI)

Codegen en tant qu’outil de générateur de code limité, son intention est de générer et tester les codes dans un ensemble de modèle et de langage. C’est un outil pour faciliter l’unification et la génération des frameworks. Avant la génération du code, codegen interprète les codes et les instructions dans les fichiers templates. Les informations que contiennent ces templates sont appelées des règles [1].

1. **Templates files** : *Mustache*

Les fichiers templates est un ensemble de fichier contenant des textes qui définissent une part de code source généré. Ils contiennent les codes sources pour le langage cible combiné avec des tokens (variables ou nom d’objet qui peut être utilisé dans les templates) appelés par le codegen. Ces fichiers templates permettent de décrire un modèle et ont des extensions .mustache.

Mustache est une spécification pour les langages de modélisation [2]. Il décrit les manières d’affichage. Il est implémenté dans plusieurs autre langage de programmation [3]. Mustache se sert des accolades pour indiquer les champs. Ces accolades sont à l’origine son nom car en leurs rotant à 90° ils ressemblent à des moustaches. Ces champs contiennent des variables appelés des entrées qui seront remplacés par leurs valeurs (si elles existent) lors de la génération de code.

Fichier.mustache Résultat

Bonjour UMMTO

Bonjour {{universite}}

Entrée

{

" universite ": "UMMTO"

}

**Figure 4.1** : fonctionnement de *Mustache*

C’est un langage moins logique car il n’effectue pas les opérations, les conditions ni les boucles [2]. Il est cependant possible d'afficher ou de masquer sous condition un bloc de texte à travers une section. Une section commence par {{#nomSection}} et se termine par {{/nomSection}} et la section inverse est faite par le même nom que la section : {{^nomSection}} et se termine par {{/nomSection}}

Fichier.mustache

Bonjour

{{#maCondition}}

département informatique

{{/maCondition}}

{{^maCondition}}

Tizi-Ouzou

{{/maCondition}}

Résultat

Bonjour Tizi-Ouzou

Entrée

{

"maCondition": false

}

**Figure 4.2** : condition (if\_else) en *Mustache*

Une boucle en mustache est faite avec les tableaux c’est-à-dire mustache se comporte comme une boucle quand l’entrée est un tableau. Le nombre d’itération de la boucle dépends du nombre d’entrée (indice) du tableau.

Fichier.mustache Résultat

**U**

**M**

**M**

**T**

**O**

{{#universite}}

{{.}}

{{/ universite}}

Entrée

{

" universite " : ["U", "M", "M", "T", "O"]

}

**Figure 4.3** : boucle en *Mustache*

En mustache un fichier peut charger d’autres templates en {{> nom\_template2}}

1. **POM files** :
2. SPI registration
3. Config file
4. Architecture du générateur OpenPS
5. Un cas d’utilisation
6. conclusion

[1] <https://documentation.help/CodeGen/IDH_Topic10.htm>

[2] <https://www.tsmean.com/articles/mustache/the-ultimate-mustache-tutorial/>

[3] <https://mustache.github.io/>