sopra steria

DT

Bibliothèque

Annotation

Documentation Technique

Cette bibliothèque a pour objectif de permettre à un développeur de vérifier si les règles de gestion d'un projet sont bien couvertes par des tests

Table des matières

Description

Cette bibliothèque a pour fonction de permettre la vérification de la couverture des règles de gestion d'un projet par les tests unitaires et d'intégrations.

Elle utilise pour cela des annotations situer au niveau des TU et TI ou bien des classes les contenant.

À la fin de l'exécution des tests, un rapport est généré et indique pour chaque règle de gestion si elle est couverte, et si oui, les fonctions/classes qui la couvrent.

La bibliothèque offre la possibilité de générer un fichier log contenant toutes les règles de gestion présentes dans les annotations, mais pas dans le fichier d'entrée.

Composition du projet

Package : Annotations

Il contient toutes les classes en lien avec les annotations

Classe : annotation

Un objet du type de l'annotation, c'est cette classe qui est appelée par les utilisateurs pour lancer la vérification des règles de gestion

Classe : AnnotationReflection

La classe qui va permettre à notre librairie de chercher les annotations du projet.

Fonctions

- getDecorators

```
public Map<String, ArrayList<String>> getDecorators(Boolean tu)
```

Cette fonction qui permet à la librairie de récupérer toutes les annotations du projet de les mapper avec les classes et les fonctions correspondantes.

La fonction prend un booléen en entrée afin de choisir si elle retourne les informations pour les TU (true) ou pour les TI (false)

En sortie on a une map du nom de la règle de gestion avec une liste des classes et fonctions associées.

Variables de la fonction :

- ✓ **allClasses** (`List<Object>`): liste des classes du projet.
- ✓ **multiValueMap** (`Map<String, ArrayList<String>>`): map des annotations / classes&fonctions correspondantes.
- ✓ **methods** (`Method[]`): liste de toute les méthode d'une classe

- ✓ **words classnames classnames2** et **functionnames** (String[]) : nom de fonctions et de classes.
- ✓ **annotation** (`RGDTIc` ou `RGDTIf`) : un objet du type de l'annotation

- findAllClassesUsingReflectionsLibrary

```
public List<Object> findAllClassesUsingReflectionsLibrary(String packageName)
```

Cette fonction récupère toutes les classes du projet.

Classe : RGDTUc

L'annotation pour les classes qui couvrent des tests TU

Classe : RGDTUf

L'annotation pour les fonctions qui couvrent des tests TU

Classe : RGDTIc

L'annotation pour les classes qui couvrent des tests TI

Classe : RGDTIf

L'annotation pour les fonctions qui couvrent des tests TI

Package : process

Contient la classe ProcessProgram qui contient les fonctions sans lien direct avec les annotations

Fonctions

- processFile

```
public Boolean processFile(String fileCSV, String outputFile) throws IOException
```

Cette fonction représente le cœur de la bibliothèque, elle vérifie la correspondance des TI/TU avec les règles de gestion. Elle gère également tout le processus avec la lecture et l'écriture des dans le CSV et du log. Elle prend en entrée le fichier listant les règles de gestions et le dossier de destinations du rapport. Elle renvoie un booléen indiquant s'il est à true que toutes les règles sont couvertes.

Variables de la fonction :

- ✓ **doc** (`List<String[]>`): Contient le fichier CSV d'entrée.
- ✓ **noKO** (`Boolean`): est à true jusqu'à ce qu'une RG soit KO
- ✓ **multiValueMapTU** (`Map<String, ArrayList<String>>`): map des annotations / classes&fonctions des TU correspondantes.

- ✓ **multiValueMapTI** (`Map<String, ArrayList<String>>`): map des annotations / classes&fonctions des TI correspondantes.
- ✓ **rgName** (`List<String>`) : Contient le nom de toutes les RG du fichier d'entrée.

- readCSVFile

```
public List<String[]> readCSVFile(String fileCSV)
```

Cette fonction permet de lire un CSV. Elle prend en entrée le chemin du fichier CSV et retourne une liste de tableaux de string qui correspond aux différentes colonnes et lignes du CSV.

- writeToCsv

```
public void writeToCsv(List<String[]> doc, String outputFile)
```

Cette fonction permet de d'écrire dans un CSV. Elle prend en entrée une liste de tableaux de string à écrire dans le fichier et le chemin du dossier de destination.

- writeLog

```
public void writeLog(String outputFile) throws IOException
```

Cette fonction permet d'écrire le log.txt. Elle prend en entrée le chemin du dossier de destination.

- printLog

```
public void printLog()
```

Cette fonction affiche le log dans la console