sopra steria

DU

Bibliothèque

Annotation

Documentation Utilisateur

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.

Importation

Dans le projet

Sur Eclipse :

```
Project >> Build Path >> Add External Archives
```

Sur Intellij :

```
File >> Project Structure >> Libraries
```

Dans le code

- Il faut dans un premier temps importer la bibliothèque.

```
import Annotations.Annotations;
```

- Il faut ensuite appeler la fonction qui va lancer l'analyse des annotations (par exemple dans les tests unitaires).

Voici la fonction à appeler :

```
Annotations.annotation(fileCSV, outputFolder, falseIfKo, activateLog);
```

Paramètres :

✓ **fileCSV** (String): chemin du fichier contenant la liste des règles de gestion. Celui-ci doit être sous la forme suivante :

Exigence		Couverture			
RDG	Description synthétique	TU	TI	Tests croisés	Bilan couverture
RG1	Commentaires				
RG2	Commentaires				
RG3	Commentaires				
RG4	Commentaires				
RG5	Commentaires				
...					

Les cases bleues ne sont pas traitées par la bibliothèque et peuvent contenir n'importe quoi.

Les cases orange sont lues ou écrites par la bibliothèque, les noms des RGD doivent être indiqués dans la première colonne. Les autres doivent de préférence rester vide (la bibliothèque est censée fonctionner même si elles sont remplies, mais mieux vaut ne pas tenter le diable ;)

✓ **outputFolder** (String) : correspond au chemin du dossier de destination du rapport (et des logs) une fois généré.

✓ **falseIfKo** (Boolean) : permet de faire échouer des TU si les RG ne sont pas toutes couvertes.

- Si `false`, la fonction retournera `true` quoi qu'il arrive.
- Si `true`, la fonction retournera `true` si toutes les RG sont vérifiés par des TU/TI et false si au moins une règle de gestion ne l'est pas.

✓ **activateLog** (Boolean) : les fichiers logs indique les règles de gestions présentes dans les annotations, mais pas dans le fichier d'entrée.

- Si `false`, aucun fichier log n'est généré (les logs apparaissent tout de même dans la console).
- Si `true`, la fonction génère des fichiers log dans le dossier **outputFolder**.

Exemple d'utilisation

```
// ce code se trouve dans les tests unitaires (avant ou après sans importance)
@Test
public void testCouverture() throws CsvValidationException,
FileNotFoundException, IOException
{ assert(Annotations.annotation(inputFile, outputFolder, false, true));}
```

Utilisation des annotations

Pour définir les règles de gestion qui sont couvertes nous allons utiliser des annotations.

- Pour le TU, utiliser les **@RDGTUc** ou **@RGDTUf** pour respectivement les classes et les fonctions. Exemple d'utilisation avec plusieurs règles de gestions :

```
@RGDTUf(rules = { "RG1", "RG2" })
@Test
public void testAdition()
{ assertEquals(MainDemo.addition(2, 2), 4); }
```

Les annotations pour les classes permettent de diminuer la précision de la vérification. Il n'est pas obligatoire d'utiliser les deux types d'annotation.

On peut n'utiliser que les annotations de classe s'il n'est pas nécessaire de savoir quelles fonctions valide les règles de gestion.

- Pour le TI il faudra utiliser les **@RDGTIc** ou **@RGDTIf** pour respectivement les classes et les fonctions. Voici un exemple ci-dessous.

```
@Test
@RGDTIf(rules = { "RG1", "RG2" })
public void testSoustraction()
{ assertEquals(MainDemo.soustraction (3, 2), 1); }
```