

Les tests d'acceptances

Introduction

Qui suis je ?

Déroulement du cours

¥ N'hésitez pas à interrompre ou à intervenir

¥ Merci de ne pas faire de bruit :)

Le TP

¥ Le dernier TP de ce cours sera noté.

¥ A rendre quand ? : avant le vendredi 21 février 23h59 :)

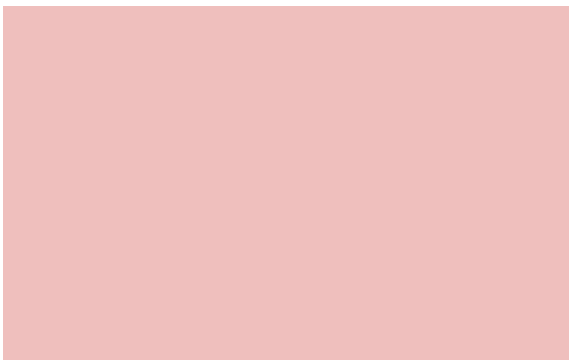
¥ Comment ?

! sur un Repo Git public genre GitHub (recommandé)

! par zip envoyé par mail à csilari@sqli.com

¥ Des questions pendant le temps imparti ?

! par mail à csilari@sqli.com



Vérifier que votre projet compile !!!

Les sources

¥ <https://github.com/csilari/cours-iut-tests-auto>

Rédiger des scénarios avec Gherkin

La syntaxe Gherkin

Le Gherkin est une syntaxe utilisée pour faire du BDD (Behaviour-Driven Development) en écrivant des scénarios de test compréhensibles par des individus non techniques.

Un langage commun

- ¥ C'est un langage naturel, ce qui simplifie sa compréhension et son utilisation
- ¥ On explique le déroulement d'une fonctionnalité (Feature)

Le scénario au coeur du langage Gherkin

- ¥ Un scénario est ensuite décrit par un ensemble d'étapes (Scenario).
- ¥ Chaque étape est introduite par un mot-clé suivi d'un texte libre qui correspond à sa description.

On retrouve ici le paradigme Given-When-Then du BDD

!

Pour alléger la lecture d'un scénario, Gherkin propose deux autres mots clés **And** et **But**.

Un exemple en français

Plus de 60 langues sont supportées (voir [Langages supportés](#))

```
Scenario:  
  Etant donné que offrir un gâteau rend heureux  
  Lorsque on m'offre 1 gâteau  
  Alors je suis heureux
```

Ecrire un scénario paramétré

A l'aide du mot clé **Scenario Outline**, on va pouvoir paramétrer de tels scénarios.

des tableaux d'exemples sont là pour faciliter l'écriture/lecture de ce scénario.

Remarques :

! Un même scénario paramétré peut être composé de plusieurs tableaux d'exemples

Un exemple de **Scenario Outline**

Factoriser des étapes communes ^ différents scénarios

- ¥ Comme le scénario et le scénario paramétré, le background n'est ni plus ni moins qu'une nouvelle description de scénario composée.
- ¥ Le bloc Contexte sera joué systématiquement avant chaque scénario.

Des scénarios, pour décrire une fonctionnalité !

- ¥ Une fonctionnalité identifiée par le mot clé **Feature** est un groupement de plusieurs scénarios et/ou plans de scénario.

Conventions

- ¥ Un document rédigé en Gherkin doit respecter certaines conventions :
 - ! il doit être enregistré dans un fichier portant l'extension **.feature**
 - ! il ne peut contenir la description que d'une seule fonctionnalité (Feature)

!

Fonctionnalité: Servir un café

Ê En tant qu'utilisateur

Ê Je veux consommer un café

Ê dont le prix fixe est de 40 centimes

Ê Contexte:

Ê Etant donné que j'ai inséré 40 centimes d'euros

Ê Scénario: Servir un café court sans sucre

Ê Quand je demande un "café court sans sucre"

Ê Alors la machine me remplit un gobelet de "café court sans sucre"

Ê Scénario: Servir un café court avec sucre quand je donne trop de monnaie

Ê Etant donné que j'ai inséré 10 centimes d'euros

Ê Quand je demande un "café court avec sucre"

Ê Alors la machine me remplit un gobelet de "café court sans sucre"

Des tags pour finir..

Les tags permettront par exemple de :

- ! marquer le domaine fonctionnel que le scénario couvre
- ! filtrer l'exécution de certains scénarios.

Cucumber

Une seule source de vérité

Cucumber fusionne la spécification et la documentation de test en un seul ensemble cohérent.

Documentation vivante

¥ Parce qu'ils sont automatiquement testés par Cucumber, vos spécifications sont toujours à jour.

Mise en oeuvre

¥ Indiquer à JUnit que vous voulez que votre classe de test soit lancée avec Cucumber :

```
@RunWith(Cucumber.class)
public class MyServiceTest {

}
```

!

¥ Créer le fichier feature qui contiendra les scenarios de test.

¥ Cucumber vous demande alors d'implémenter les phrases présentes dans le scenario.

¥ Il faut les mettre dans une nouvelle classe :

```
public class MyServiceSteps {
```

TP noté à rendre

¥ Mr Pignon et Mr Leblanc vont au bar le Juste qui est un bar à cocktail.

¥ Il y a dans le bar seulement 10 places assises.

Première histoire :

¥ *Ils arrivent mais il y a déjà 9 personnes dans le bar donc ils se font refuser le droit d'entrée.*

Deuxième histoire :

¥ *Il arrivent, il y a déjà 8 personnes dans le bar.*

¥ *La personne derrière eux ne peut pas rentrer, le bar affiche complet.*

¥ *Ils commandent chacun un cocktail du mois à 10!.*

¥ *C'est Mr Leblanc qui paie l'ensemble.*

¥ *A la fin de leur verre, on vérifie la note, et Mr Leblanc paie.*

¥ *Mr Pignon est heureux car ils n'ont consommé qu'un verre (problème de foie de Mr Pignon).*

TP noté à rendre (suite)

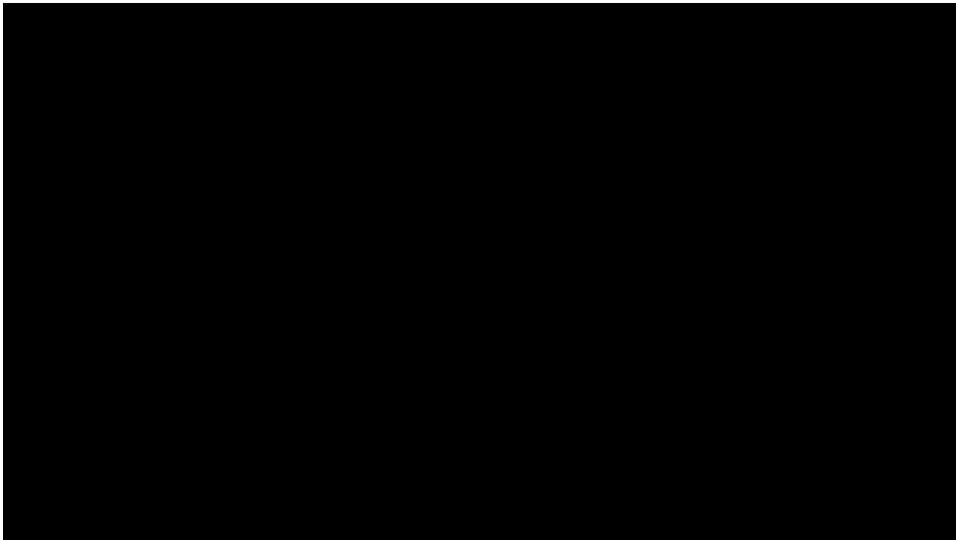
Troisième histoire :

¥ *Ils arrivent, il y a 3 personnes dans le bar.*

- ¥ Ils commandent chacun un cocktail du mois ^ 10!.
- ¥ Personne ne paie le verre de l'autre.
- ¥ A la fin de leur verre, ils vérifient chacun leur note, Mr Pignon paie mais Mr Leblanc insiste pour payer un autre cocktail du mois.
- ¥ Il commande donc 2 autres cocktails du mois pour sa note.
- ¥ A la fin de leur verre, Mr Leblanc vérifie la note et paie.
- ¥ Mr Pignon est triste car il sait qu'au dessus d'un cocktail ce dernier va passer une très mauvaise nuit (problème de foie de Mr Pignon).

TP - Travail Demandé

- ¥ Ecrire en langage Gherkin l'ensemble de l'histoire ainsi que les trois histoires.
- ¥ Générer dans un projet Java les tests d'acceptance avec Cucumber.
- ¥ Concevoir et développer la solution pour que tous les scénarios Cucumber passent et en vert.



C'est à vous ;)

Quelques conseils

- ¥ Privilégiez l'Anglais pour écrire les scénarios Gherkin.
- ¥ Adoptez les principes du BDD (le test d'abord !).
- ¥ Appuyez-vous sur le Tutoriel de prise en main de Cucumber (voir le PDF [acceptance-tests/exercices/cucumber-tutorial.pdf](#)).
- ¥ Pour commencer, vous pouvez vous baser sur le projet [junit5-cucumber-maven-starter](#).
 - ! Lancer les tests avec votre IDE,
 - ! ou bien en ligne de commandes `mvn test` ou `./mvnw test` (si Maven non installé).



Vérifier que votre projet compile !!!

Q&A