eBook Gratuit

APPRENEZ cucumber

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#cucumber

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: cucumber

It is an unofficial and free cucumber ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official cucumber.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec le concombre

Remarques

À propos du concombre

Cucumber est un outil qui exécute des spécifications exécutables de logiciels. Les spécifications, appelées "caractéristiques", sont écrites en langage naturel structuré. Cucumber exécute une fonctionnalité en mappant chacune de ses étapes sur une "définition d'étape" écrite dans le langage de programmation pris en charge par cette implémentation de Cucumber. Le concombre est implémenté dans de nombreux langages de programmation, y compris Ruby (l'original), Java et Javascript . Il est également traduit dans de nombreuses langues humaines.

Concombre a été écrit pour soutenir la méthodologie agile appelée Développement comportemental (BDD). Dans BDD, on commence le développement en écrivant des tests d'acceptation qui décrivent la fonctionnalité du logiciel du point de vue de l'utilisateur (plutôt que du point de vue d'un programmeur, comme dans les tests unitaires). Les caractéristiques du concombre servent de ces tests d'acceptation.

En général, les fonctionnalités Cucumber sont une documentation lisible par l'homme, qui est également une suite de tests exécutables, ce qui signifie que la documentation et les tests sont toujours compatibles. Le concombre est utile pour communiquer avec les parties prenantes autres que les programmeurs à propos de la documentation et des tests. Il permet également aux programmeurs d'écrire des tests à un niveau conceptuel sans se soucier des problèmes de langage de programmation.

Le concombre est le plus souvent utilisé pour spécifier et tester des applications Web, en utilisant un pilote de navigateur tel que Selenium ou PhantomJS. Cependant, il peut être utilisé avec n'importe quel logiciel pouvant être exécuté et dont l'état ou les résultats peuvent être déterminés à partir du langage de programmation pris en charge par une implémentation de Cucumber.

Autre documentation

La documentation officielle est à https://cucumber.io/docs . La documentation générée à partir des fonctionnalités de Cucumber décrivant les implémentations de Cucumber est à

- JavaScript: https://relishapp.com/cucumber/cucumber-js/docs
- Ruby: https://relishapp.com/cucumber/cucumber/docs

https://relishapp.com/explore comprend d'autres outils et exemples liés à Cucumber, mais malheureusement pas Cucumber-JVM.

Ce sujet

Ce sujet ne devrait donner que quelques exemples qui présentent au lecteur les concepts de concombre. D'autres sections donneront des exemples complets d'installation, d'utilisation en ligne de commande et IDE, de fonctionnalités, de définitions d'étapes, etc.

Examples

Une fonctionnalité de concombre

Le concombre utilise la syntaxe Gherkin pour décrire les comportements de votre logiciel en langage naturel structuré.

En tant que tel, Cucumber n'est **pas** un framework de test (un malentendu commun), mais un *cadre de documentation système* , pas très différent des autres comme le scénario de cas d'utilisation. L'incompréhension commune est due au fait que la documentation de Concombre *peut être automatisée afin de s'assurer qu'elle reflète le comportement réel du système* .

Une suite de documentation Cucumber est composée de `Features` , chacune décrivant une fonctionnalité de votre logiciel, écrite en Gherkin et hébergée dans son propre fichier. En organisant ces fichiers dans une structure de répertoires, vous pouvez *regrouper* et *organiser les* fonctionnalités:

- bancaire/
    - retrait.feature
    - atm.feature
    - personal-loan.feature
- commerce/
    - portfolio.feature
    - intraday.feature
- hypothèque/
    - évaluation.features
    - caractéristique comptable

Chaque `Feature` est un fichier texte composé d'une section d'introduction optionnelle, non structurée, purement informative et d'un ou plusieurs `Scenarios` , chacun représentant une condition d'utilisation ou un cas d'utilisation.

Exemple:

```
Feature: Documentation
As a StackOverflow user or visitor
I want to access the documentation section

    Scenario: search documentation on Stack Overflow
        Given I am on StackOverflow
        And I go to the Documentation section
        When I search for "cucumber"
        And I follow the link to "cucumber"
        Then I should see documentation for "cucumber"
```

Chaque ligne commençant par *Given, Quand, Et, Mais* ou est *ensuite* appelé `Step` . N'importe quelle étape peut commencer par n'importe lequel de ces mots, peu importe l'ordre, mais il est classique de les utiliser de la manière la plus naturelle possible.

Les fonctionnalités peuvent également être organisées via des `Tags` , des annotations que l'éditeur peut placer sur une `Feature` ou un `Scenario` pour mieux les classer.

Exécutifs une fonction est réalisée par le code *de* la *colle* qui peut être écrit dans de nombreuses langues (Java, Ruby, Scala, C / C ++): chaque `Step` est en correspondance avec le code de la colle afin d'identifier `Step Definitions` (communément abrégé en *StepDef)* par expressions régulières.

Chaque `Step` ne peut avoir qu'une seule `Step Definition` associée.

Lorsqu'une `Feature` est exécutée, chaque `Scenario` composition est exécuté, ce qui signifie que chaque StepDef correspondant aux `Step` s dans chaque `Scenario` est exécuté.

Installation Ruby Pure

Pour installer Cucumber à utiliser avec Ruby, utilisez simplement la commande

```
gem install cucumber
```

Si vous utilisez un bundler, vous pouvez également ajouter la ligne suivante à votre Gemfile.

```
gem 'cucumber'
```

Et puis exécutez bundler

```
bundle install
```

[Je pense que cela fait partie de son propre sujet, Installation. J'ai créé ce sujet et y ai copié cet exemple. Lorsque ce sujet est approuvé, je vais le déplacer et supprimer la copie.]

Une définition d'étape de concombre dans Ruby

Dans features / step_definitions / documentation.rb:

```
When /^I go to the "([^"]+)" documentation$/ do |section|
  path_part =
    case section
      when "Documentation"
        "documentation"
      else
        raise "Unknown documentation section: #{section}"
    end
  visit "/documentation/#{path_part}/topics"
end

Then /^I should see the "([^"]+) documentation"$/ do |section|
  expect(page).to have_css('h2.doctag_title a', text: section)
end
```

Ces étapes exercent une application Web. Ils sont à peu près aussi simples que possible tout en

étant pratiques.

Chaque étape commence par un mot-clé Gherkin, qui dans un fichier de définition d'étape est une méthode qui enregistre une étape avec Concombre. La méthode de définition des étapes prend une expression régulière, qui correspond à une ligne dans un scénario, et un bloc, qui est exécuté lorsque le scénario atteint une ligne correspondante. Les groupes de capture dans l'expression régulière sont transmis au bloc en tant que paramètres de bloc.

L'étape `When` contient un exemple simple, en ligne, de passage d'une référence lisible par l'homme à une page ("Documentation") à une URL. Les suites de concombres réels placent généralement cette logique dans une méthode distincte. La méthode de `visit` est fournie par Capybara. Capybara n'est pas obligé d'utiliser le Concombre, bien qu'il soit très couramment utilisé avec lui. `visit` indique au navigateur contrôlé par Capybara de visiter l'URL donnée.

L'étape `Then` montre comment tester le contenu d'une page. `expect` / `to` est fourni par RSpec (encore une fois, pas requis par le concombre mais très couramment utilisé avec elle). `have_css` est fourni par Capybara. Le sélecteur CSS donné doit correspondre à un élément de la page contenant le texte donné. Notez que cette attente échoue si la requête du navigateur a échoué.

Pour plus d'exemples, voir la rubrique "Définition de l'étape" .

Lire Commencer avec le concombre en ligne:
https://riptutorial.com/fr/cucumber/topic/4875/commencer-avec-le-concombre

Chapitre 2: Caractéristiques

Introduction

Vous pouvez utiliser le concombre comme plug-in dans QTP et Selenium. Les étapes du scénario du concombre sont des variables globales. Vous pouvez implémenter une fois et appeler plusieurs fois. De ce fait, réduit la maintenance du code et peut réutiliser le même code si nécessaire.

Remarques

Les fonctions de concombre sont écrites dans le langage Gherkin et stockées dans des fichiers avec le suffixe `.feature` . Cette rubrique donne des exemples de chaque fonctionnalité de Gherkin.

Examples

Une fonctionnalité minimale de concombre

Dans fonctionnalités / documentation.feature:

```
Feature: Documentation

  Scenario: User views documentation
    When I go to the "Cucumber" documentation
    Then I should see the "Cucumber" documentation
```

Une fonctionnalité minimale possède une ligne de `Feature` et un `Scenario` comportant une ou plusieurs étapes commençant par `When` , `Then` ou un autre mot-clé Gherkin.

Un scénario judicieux aurait probablement plus d'une étape.

Plan du scénario

Modèle comme ci-dessous

```
Scenario Outline: As a homemaker i want to buy and pay for the below product
  Given I purchase <a product>
    And I require a carry bag to take things to home
  When I pay bill using <payment method> to successfully checkout
  Then I should have a receipt

Examples:
| a product     | payment method |
|  Cake         | Visa           |
|  Coke         | Paypal         |
```

Utilisation de la syntaxe

```
Feature: Some terse yet descriptive text of what is desired
     Textual description of the business value of this feature
     Business rules that govern the scope of the feature
     Any additional information that will make the feature easier to understand

Background:
    Given some precondition needed for all scenarios in this file
        And another precondition

Scenario: Some determinable business situation
  Textual description of the business value of this scenario
  Business rules that govern the scope of the scenario
  Any additional information that will make the scenario easier to understand
    Given some precondition
      And some other precondition
    When some action by the actor
      And some other action
      And yet another action
    Then some testable outcome is achieved
      And something else we can check happens too
      But something else we can check does not happen

Scenario Outline: Some determinable business situation
    Given I am <precondition>
        And some other precondition
    When some action by the actor
    Then I have <outcome> rights

Examples:
    | precondition | outcome  |
    | username1    | customer |
    | username2    | admin    |
```

Les mots-clés suivants sont interchangeables, mais selon le contexte, il peut être préférable d'utiliser:

- `Feature:` | `Ability:` | `Business Need:`
- `Scenario Outline:` | `Scenario Template:`
- `Examples:` | `Scenarios:`
- `Given` | `When` | `Then` | `And` | `But` | `*` |

Lire Caractéristiques en ligne: https://riptutorial.com/fr/cucumber/topic/6023/caracteristiques

Chapitre 3: Définitions d'étape

Remarques

Les définitions d'étape se trouvent dans le langage de programmation pris en charge par une implémentation donnée de Concombre. Cette rubrique donne des exemples de définitions d'étape dans chaque langage de programmation pris en charge et des exemples d'utilisation des appels de l'API Cucumber dans les définitions d'étape.

Examples

Quelques définitions simples de l'étape Ruby

Dans features / step_definitions / documentation.rb:

```
When /^I go to the "([^"]+)" documentation$/ do |section|
  path_part =
    case section
      when "Documentation"
        "documentation"
      else
        raise "Unknown documentation section: #{section}"
    end
  visit "/documentation/#{path_part}/topics"
end

Then /^I should see the "([^"]+) documentation"$/ do |section|
  expect(page).to have_css('h2.doctag_title a', text: section)
end
```

Ces étapes exercent une application Web. Ils sont à peu près aussi simples que possible tout en étant pratiques.

Chaque étape commence par un mot-clé Gherkin, qui dans un fichier de définition d'étape est une méthode qui enregistre une étape avec Concombre. La méthode de définition des étapes prend une expression régulière, qui correspond à une ligne dans un scénario, et un bloc, qui est exécuté lorsque le scénario atteint une ligne correspondante. Les groupes de capture dans l'expression régulière sont transmis au bloc en tant que paramètres de bloc.

L'étape `When` contient un exemple simple, en ligne, de passage d'une référence lisible par l'homme à une page ("Documentation") à une URL. Les suites de concombres réels placent généralement cette logique dans une méthode distincte. La méthode de `visit` est fournie par Capybara. Capybara n'est pas obligé d'utiliser le Concombre, bien qu'il soit très couramment utilisé avec lui. `visit` indique au navigateur contrôlé par Capybara de visiter l'URL donnée.

L'étape `Then` montre comment tester le contenu d'une page. `expect` / `to` est fourni par RSpec (encore une fois, pas requis par le concombre mais très couramment utilisé avec elle). `have_css` est fourni par Capybara. Le sélecteur CSS donné doit correspondre à un élément de la page

contenant le texte donné. Notez que cette attente échoue si la requête du navigateur a échoué.

Lire Définitions d'étape en ligne: https://riptutorial.com/fr/cucumber/topic/5681/definitions-d-etape

Chapitre 4: Installer le plugin de concombre dans Intellij

Introduction

Les plugins Cucumber pour IntelliJ IDEA offrent des fonctionnalités IDE pratiques pour travailler avec les fichiers de fonctionnalités Gherkin dans un projet IntelliJ utilisant le framework Cucumber. Les plugins sont disponibles pour les langages Java, Scala ou Groovy.

Remarques

Le plug-in Cucumber pour Java IntelliJ offre des fonctionnalités IDE pour un développement pratique avec Concombre, y compris

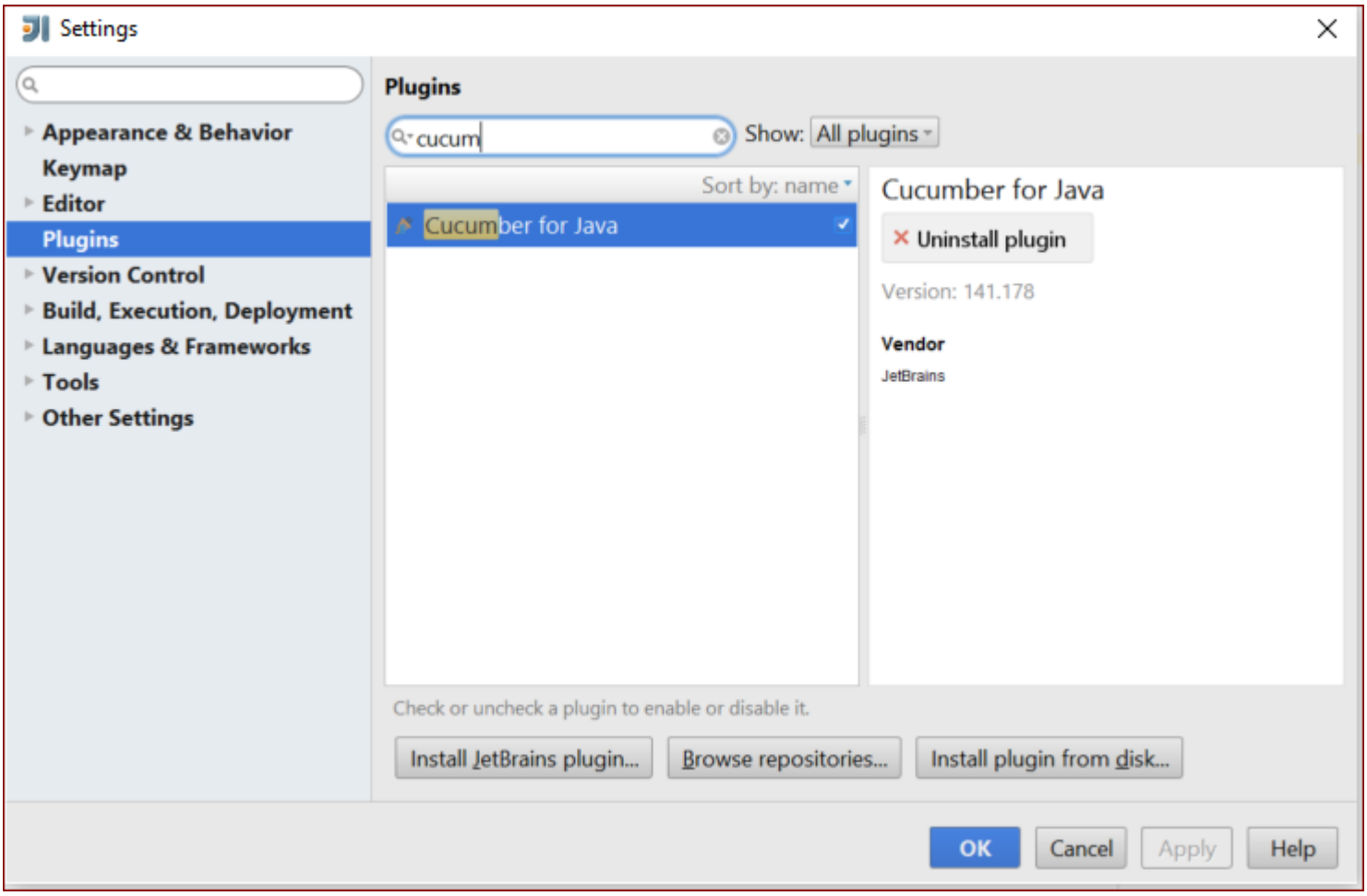
- Gherkin step génération de colle pour les étapes non implémentées
- Achèvement du code pas à pas du cornichon
- Saut à la méthode de collage
- Mise en évidence de la syntaxe Gherkin dans les fichiers ".feature" correspondant à l'étape regex

et d'autres fonctionnalités pratiques.

Examples

Installer le plugin Concombre

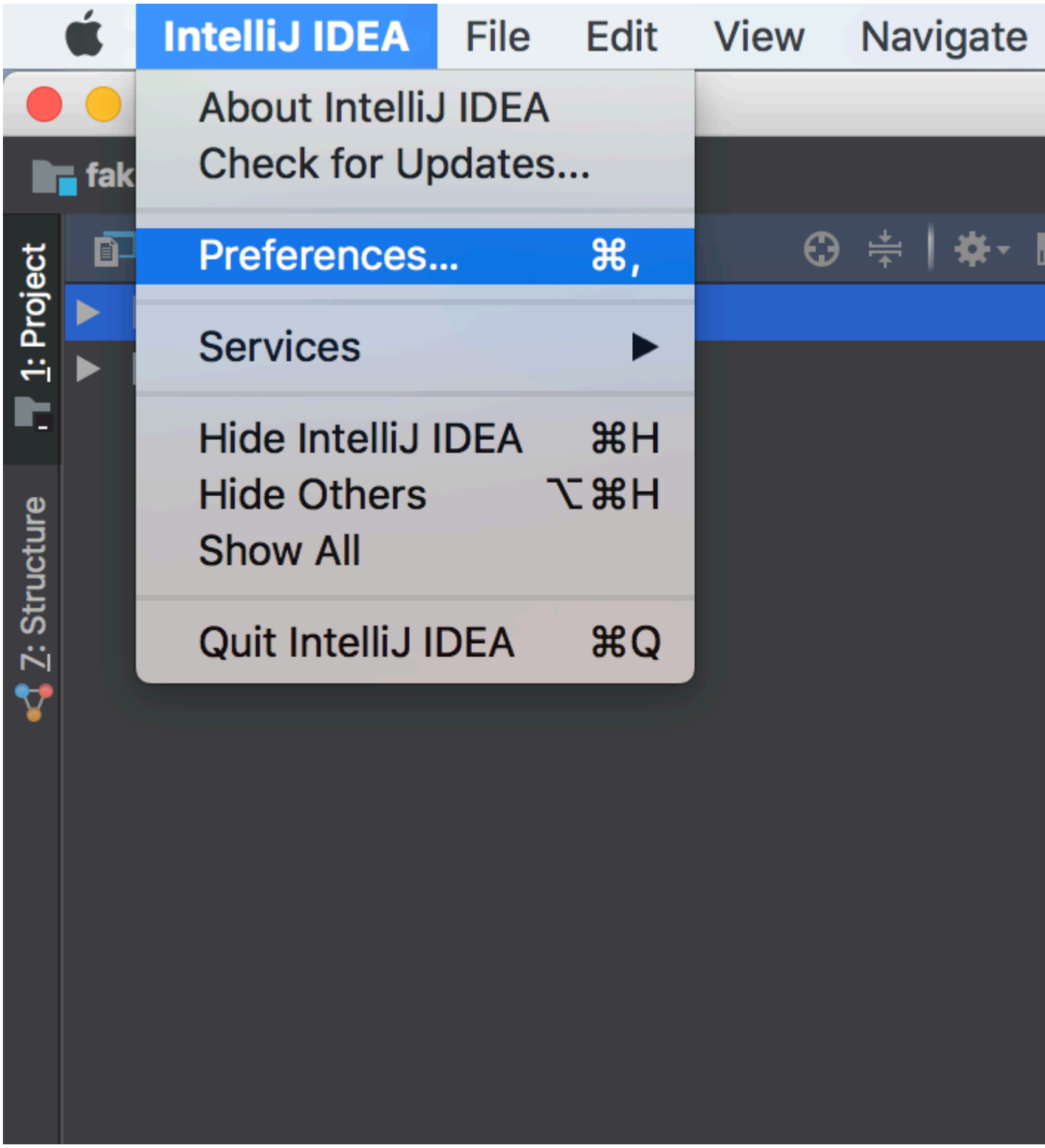
Allez dans Fichier -> Paramètres -> cliquez sur les plug-ins dans le volet gauche -> Rechercher le concombre -> Installer le plug-in



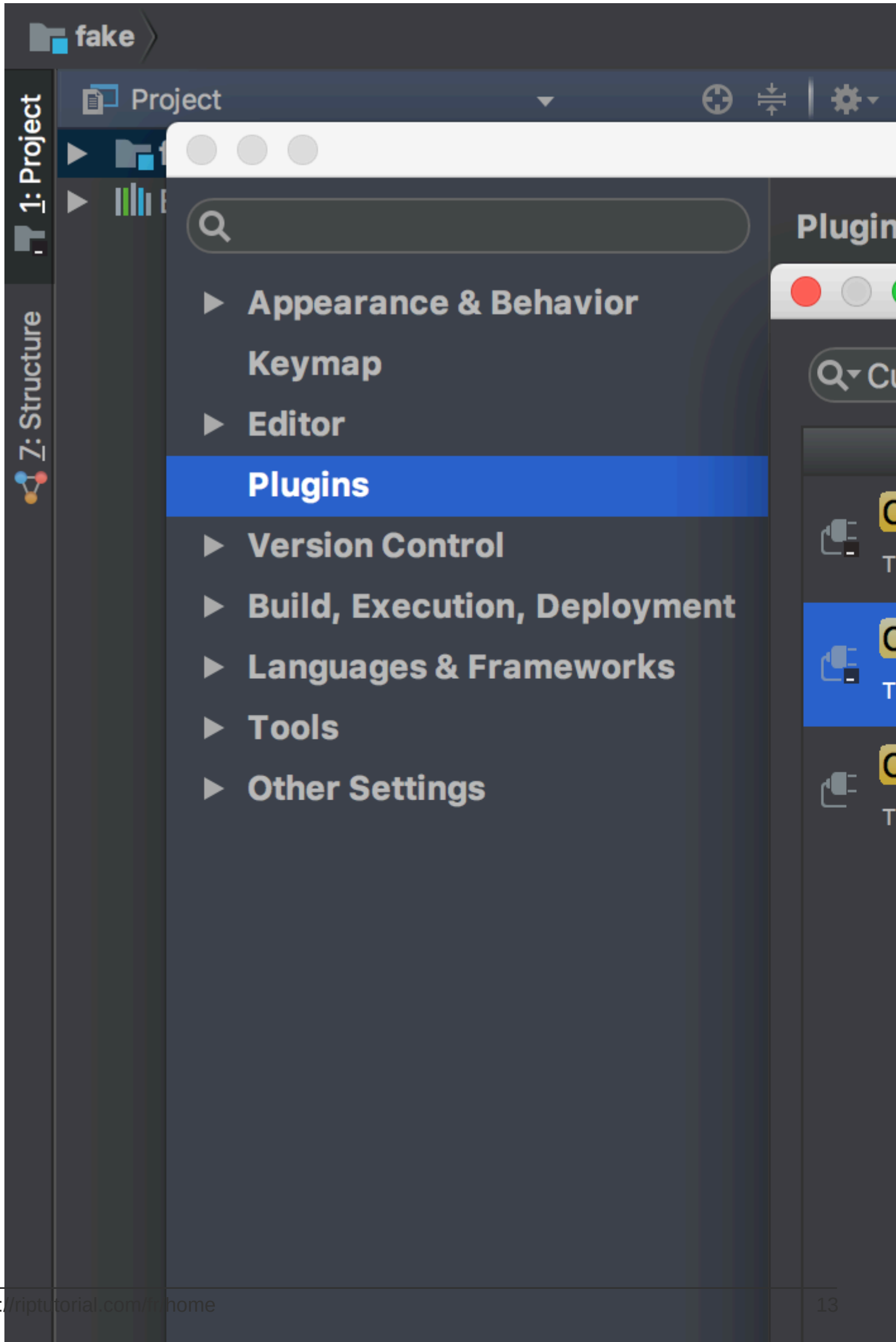
Installez IntelliJ Cucumber for Java Plugin (Mac)

Pour installer le plugin Cucumber for Java pour IntelliJ sur un Mac,

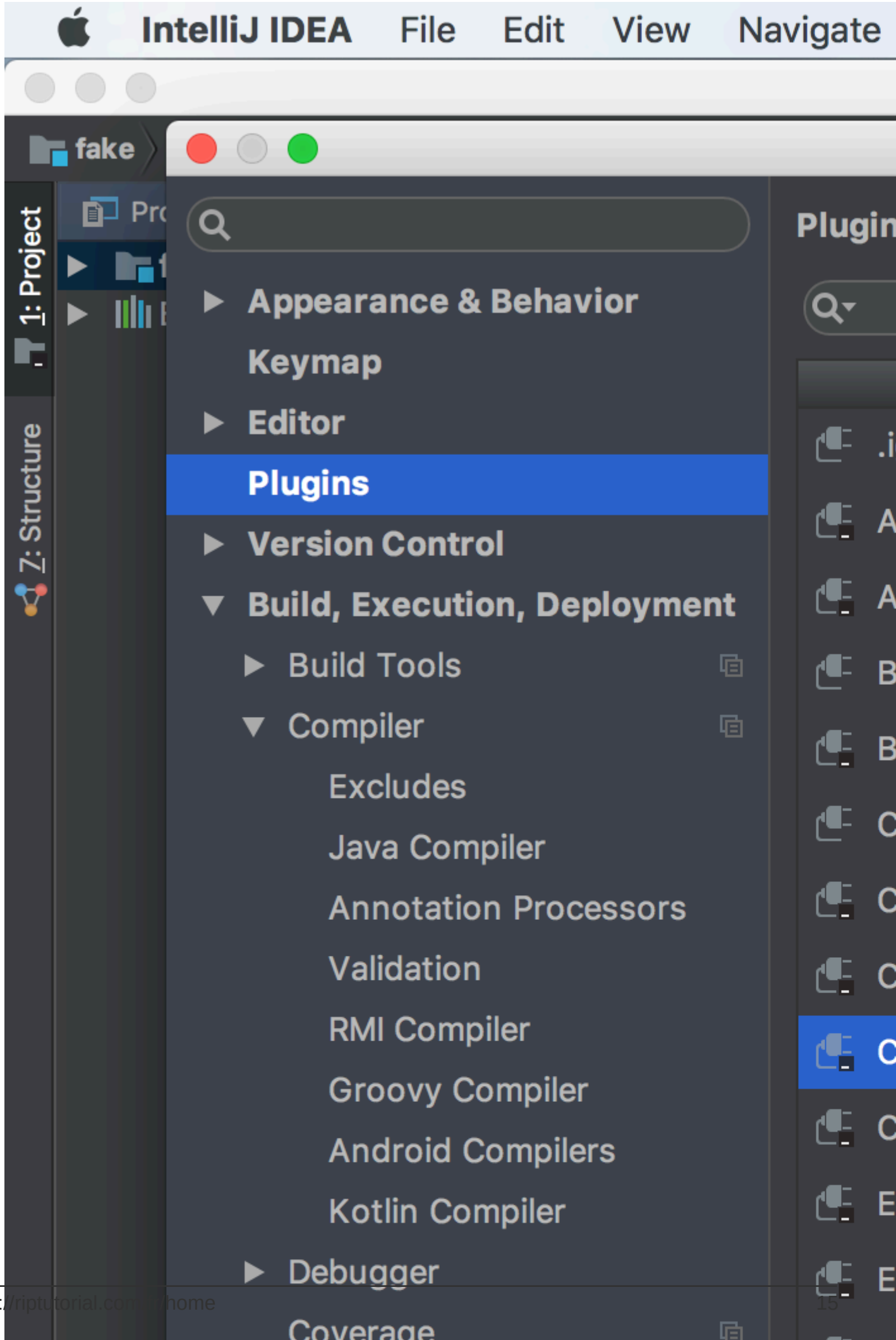
1. Démarrer IntelliJ IDEA.
2. Cliquez sur l'onglet "IntelliJ IDEA" dans la barre du haut.



3. Cliquez sur "Préférences".
4. Dans Préférences / Paramètres, cliquez sur "Plug-ins" dans le volet de gauche.
5. Cliquez sur le bouton "Parcourir les dépôts", ce qui ouvre une nouvelle fenêtre.
6. Recherchez "Concombre" dans la barre de recherche.


fake
Project
1: Project
Z: Structure
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Other Settings

7. Installez le plugin "Concombre pour Java".
8. Redémarrez l'IDE pour que le plugin prenne effet. Le concombre pour Java est maintenant installé.


IntelliJ IDEA
File
Edit
View
Navigate
fake
1: Project
7: Structure
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Build Tools
Compiler
Excludes
Java Compiler
Annotation Processors
Validation
RMI Compiler
Groovy Compiler
Android Compilers
Kotlin Compiler
Debugger
Coverage

https://riptutorial.com/fr/cucumber/topic/8356/installer-le-plugin-de-concombre-dans-intellij

Chapitre 5: pom.xml pour le projet Maven_ concombre.

Introduction

Le modèle d'objet de projet ci-dessous est le modèle pom.xml. Si vous souhaitez créer un projet Maven avec concombre, vous pouvez utiliser l'exemple ci-dessous comme modèle

Examples

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

4.0.0

```
<groupId>Project name</groupId>
<artifactId>MulitClients</artifactId>
<version>1.0-SNAPSHOT</version>


<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-core</artifactId>
        <version>1.2.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-testng</artifactId>
        <version>1.2.0</version>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>1.2.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>1.2.0</version>
        <scope>test</scope>
    </dependency>
```

```
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>gherkin</artifactId>
        <version>2.12.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>2.53.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-firefox-driver</artifactId>
        <version>2.53.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-htmlunit-driver</artifactId>
        <version>2.53.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.yaml</groupId>
        <artifactId>snakeyaml</artifactId>
        <version>1.13</version>
    </dependency>
    <dependency>
        <groupId>com.esotericsoftware.yamlbeans</groupId>
        <artifactId>yamlbeans</artifactId>
        <version>1.06</version>
    </dependency>


</dependencies>
```

Lire pom.xml pour le projet Maven_ concombre. en ligne:
https://riptutorial.com/fr/cucumber/topic/8331/pom-xml-pour-le-projet-maven--concombre-

Chapitre 6: Syntaxe du cornichon

Introduction

Gherkin est un langage lisible par l'entreprise pour l'automatisation des tests et la documentation de test. Cucumber le comprend et, ensemble, il constitue un outil de développement axé sur le comportement.

Syntaxe

- Fonctionnalité: ce mot-clé signifie que ce qui suit est une description de base ou un nom de la fonctionnalité testée ou documentée.
- Background: ce mot-clé indique les étapes à exécuter avant chaque scénario de la fonctionnalité.
- Scénario: ce mot clé représente le nom ou la description de base d'un scénario particulier testant la fonctionnalité.
- Aperçu du scénario: Ce mot-clé signifie que le scénario exécutera N fois pour chaque argument répertorié dans les exemples explicitement transmis par le nom de la colonne, entre crochets.
- Exemples: ce mot clé indique la liste des arguments statiques qui seront transmis dans une structure de scénario.
- Étant donné que ce mot clé représente une étape donnée ou une condition préalable supposée avant de continuer. Dans l'Arrange, Act, le paradigme Assert, donné représente "Arranger".
- When: ce mot-clé représente un step ou le comportement contre lequel il faut agir. Dans l'Arrange, Act, Assert paradigm, donné représente "Act".
- Ensuite: ce mot-clé représente une étape puis, autrement dit, l'étape de validation du résultat d'un comportement. Dans l'Arrange, Act, le paradigme Assert, donné représente "Assert".
- Et: Ce mot-clé est utilisé avec l'un des mots-clés ci-dessus. Si vous avez deux déclarations données, au lieu d'appeler explicitement deux fois donné, vous pouvez dire "donné A et B".

Examples

Les bases

Cet exemple passera en revue la structure de base d’un fichier de caractéristiques de concombre dans Gherkin. Les fichiers de fonctionnalités utilisent plusieurs mots-clés dans la syntaxe de base.

Regardons les mots-clés de base:

- **Fonctionnalité:** ce mot-clé signifie que ce qui suit est une description de base ou un nom de la fonctionnalité testée ou documentée.
- **Scénario:** ce mot clé représente le nom ou la description de base d'un scénario particulier

testant la fonctionnalité.

- **Étant donné que** ce mot clé représente une étape donnée ou une condition préalable qui est supposée avant de continuer. Dans l'Arrange, Act, le paradigme Assert, donné représente "Arranger".
- **Lorsque** ce mot-clé représente une étape ou le comportement contre lequel Dans l'Arrange, Act, Assert paradigm, donné représente "Act".
- **Ensuite** , ce mot - clé représente une étape puis, autrement dit, l'étape dans laquelle est validé le résultat d'un comportement. Dans l'Arrange, Act, le paradigme Assert, donné représente "Assert".
- **Et** Ce mot-clé est utilisé conjointement avec l'un des mots-clés ci-dessus. Si vous avez deux déclarations données, au lieu d'appeler explicitement deux fois donné, vous pouvez dire "donné A et B".
- **Mais** ce mot-clé est utilisé en conjonction avec **Given** , **When** et **Then** pour signifier que quelque chose ne doit pas arriver. Alors A mais pas B.

Tous les mots-clés doivent figurer sur une nouvelle ligne et doivent être le premier mot d'une nouvelle ligne pour être reconnus par l'analyseur de Gherkin. Les mots-clés Feature et Scenario doivent comporter un signe deux-points immédiatement après, comme indiqué dans l'exemple ci-dessous. Étant donné, Quand, Alors et Et ne nécessitent pas de deux-points.

En plus des mots-clés, vous pouvez écrire des descriptions et des commentaires. Les descriptions se produisent après le mot-clé mais sur la même ligne, où des commentaires apparaissent sur les lignes sous les mots-clés. Lors de la rédaction de commentaires sur les fonctionnalités, il est d'usage de fournir des règles explicites décrivant les limites et les conditions qui conduisent à des résultats différents des comportements.

```
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Scenario: The user successfully logs in with valid credentials
        This scenario tests that a user is able to successfully login
        provided they enter a valid username, valid password, and
        currently have an active subscription on their account.

        Given the user is on the login page
        When the user signs in with valid credentials
        Then the user should be logged in
```

Étapes paramétrées

Lors de l'écriture de Gherkin, il peut arriver que vous souhaitiez paramétrer les étapes de réutilisation par l'ingénieur qui implémente les plans de test. Les étapes reçoivent des paramètres

via des groupes de capture d'expression régulière. (**Note d'ingénierie:** Si vous ne disposez pas de paramètres correspondants pour chaque groupe de capture dans votre expression régulière, vous pouvez vous attendre à une incompatibilité "CucumberException: Arity") Dans l'exemple ci-dessous, nous avons décidé comme accepter les entiers comme arguments.

```
Feature: Product Login
   As a user, I would like to be able to use my credentials to successfully
   login.

   Rules:
   - The user must have a valid username
   - The user must have a valid password
   - The user must have an active subscription
   - User is locked out after 3 invalid attempts
   - User will get a generic error message following
     login attempt with invalid credentials

   Scenario: The user successfully logs in with valid credentials
       This scenario tests that a user is able to successfully login
       provided they enter a valid username, valid password, and
       currently have an active subscription on their account.

       Given the user is on the login page
       When the user signs in with "valid" credentials
       Then the user should be logged in

   Scenario: The user attempts to log in with invalid credentials
       This scenario tests that a user is not able to log in when
       they enter invalid credentials

       Given the user is on the login page
       When the user signs in with "invalid" credentials
       Then the user should be logged in

   Scenario: The user is locked out after too many failed attempts
       This scenario validates that the user is locked out
       of their account after failing three consecutive
       attempts to log in

       Given the user is on the login page
       When the user fails to log in 3 times
       Then the user should be locked out of their account
```

Arrière-plan de la fonctionnalité

Comme vous l'avez peut-être remarqué dans l'exemple ci-dessus, nous réécrivons plusieurs fois la même étape:

```
Given the user is on the login page
```

Cela peut être exceptionnellement fastidieux, surtout si plusieurs étapes sont réutilisées. Gherkin fournit une solution pour cela en nous donnant un autre mot-clé pour travailler avec: **Background:.**

Le mot-clé background sert à exécuter les étapes déclarées sous chaque mot-clé de la

fonctionnalité. Veillez à ne pas ajouter d'étape d'arrière-plan, sauf si vous êtes certain que cela est nécessaire pour chaque scénario. Comme les autres mots-clés, Background est suivi d'une description ou d'un nom et peut contenir des commentaires. Tout comme la fonction et le scénario, l'arrière-plan doit être suivi d'un deux-points.

```
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Background: The user starts out on the login page
        Given the user is on the login page

    Scenario: The user successfully logs in with valid credentials
        This scenario tests that a user is able to successfully login
        provided they enter a valid username, valid password, and
        currently have an active subscription on their account.

        When the user signs in with "valid" credentials
        Then the user should be logged in

    Scenario: The user attempts to log in with invalid credentials
        This scenario tests that a user is not able to log in when
        they enter invalid credentials

        When the user signs in with "invalid" credentials
        Then the user should be logged in

    Scenario: The user is locked out after too many failed attempts
        This scenario validates that the user is locked out
        of their account after failing three consecutive
        attempts to log in

        When the user fails to log in 3 times
        Then the user should be locked out of their account
```

Plan du scénario

Dans certains cas, vous souhaiterez peut-être réexécuter le même scénario encore et encore, en remplaçant les arguments. Dans ce cas, Gherkin fournit plusieurs nouveaux mots-clés pour prendre en compte cette situation, **Aperçu du scénario:** et **Exemple:.** Le mot clé Contour de scénario indique à Cucumber que le scénario va s'exécuter plusieurs fois en substituant les arguments d'une liste. Le mot-clé Examples est appelé avant que la liste ne soit explicitement notée. Les arguments pour les schémas de scénario doivent être entourés de crochets. Dans l'exemple ci-dessous, notez que les noms des arguments placés entre les crochets correspondent aux noms de colonne répertoriés sous Exemples. Chaque colonne est séparée par des barres verticales, avec des noms de colonne sur la première ligne.

```
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Background: The user starts out on the login page
        Given the user is on the login page

    Scenario Outline: The user successfully logs in with their account
         This scenario outlines tests in which various users attempt
         to sign in successfully

         When the user enters their <username>
         And the user enters their <password>
         Then the user should be successfully logged on

         Examples:
         | username | password |
         | frank    | 1234     |
         | jack     | 4321     |
```

Mots clés

Pour les besoins de la documentation, vous pouvez filtrer les plans de test ou les scénarios par catégories. Les développeurs peuvent vouloir exécuter des tests basés sur ces mêmes catégories. Gherkin vous permet de classer les fonctionnalités ainsi que les scénarios individuels via l'utilisateur des tags. Dans l'exemple ci-dessous, notez que le mot clé ci-dessus est la balise "@Automation". Gherkin reconnaît cela comme une étiquette par l'utilisateur du symbole "@". Dans cet exemple, l'ingénieur souhaite préciser que ces tests sont utilisés pour l'automatisation, lorsque tous les tests ne sont pas automatisables, certains tests doivent être effectués manuellement.

Notez également que la balise @Production a été ajoutée au verrouillage de l'utilisateur du test de scénario. Dans cet exemple, cela est dû au fait que ce scénario est uniquement actif dans l'environnement de production de l'application. Les développeurs ne veulent pas que leurs comptes sandbox soient bloqués pendant le développement. Cette balise leur permet de faire en sorte que ce test ne soit exécuté que sur l'environnement de production.

Enfin, l'aperçu du scénario porte la balise @Staging. Pour les besoins de cet exemple, cela est dû au fait que les comptes utilisés sont des comptes intermédiaires et ne fonctionnent pas dans les autres environnements. Comme pour le tag @Production, cela garantit que ces tests ne seront exécutés que dans l'environnement de stockage intermédiaire.

Ce ne sont que quelques exemples de où, comment et pourquoi vous pourriez utiliser des tags. En fin de compte, ces balises auront un sens pour vous et les développeurs et peuvent être n'importe quoi et utilisées pour classer comme bon vous semble.

```
@Automation
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Background: The user starts out on the login page
        Given the user is on the login page

    Scenario: The user successfully logs in with valid credentials
        This scenario tests that a user is able to successfully login
        provided they enter a valid username, valid password, and
        currently have an active subscription on their account.

        When the user signs in with "valid" credentials
        Then the user should be logged in

    Scenario: The user attempts to log in with invalid credentials
        This scenario tests that a user is not able to log in when
        they enter invalid credentials

        When the user signs in with "invalid" credentials
        Then the user should be logged in

    @Production
    Scenario: The user is locked out after too many failed attempts
        This scenario validates that the user is locked out
        of their account after failing three consecutive
        attempts to log in

        When the fails to log in 3 times
        Then the user should be locked out of their account

    @Staging
    Scenario Outline: The user successfully logs in with their account
         This scenario outlines tests in which various users attempt
         to sign in successfully

         When the user enters their <username>
         And the user enters their <password>
         Then the user should be successfully logged on

         Examples:
         | username | password |
         | frank    | 1234     |
         | jack     | 4321     |
```

Bouts de cornichons

- Chaque scénario teste un comportement
- Les scénarios sont écrits de manière déclarative
- Évitez les détails accessoires à l'intérieur du scénario

- Omettre l'évidence
- Eviter les étapes conjonctives
- Gardez vos scénarios courts
- Ne pas avoir beaucoup de scénarios dans la même fonctionnalité
- Utiliser des noms de scénarios descriptifs
- N'en avoir qu'un quand pas
- Utilisez le “devrait” dans Etapes

Lire Syntaxe du cornichon en ligne: https://riptutorial.com/fr/cucumber/topic/9296/syntaxe-du-cornichon

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec le concombre	Community, Dave Schweisguth, Mo H., Roberto Lo Giacco, SirLenz0rlot, user3554664
2	Caractéristiques	Dave Schweisguth, Kyle Fairns, Priya
3	Définitions d'étape	Dave Schweisguth
4	Installer le plugin de concombre dans Intellij	George Pantazes, Priya
5	pom.xml pour le projet Maven_ concombre.	user
6	Syntaxe du cornichon	jordiPons, tramstheman, user3554664