TP3 – Gherkin, Cucumber et Hooks

Objectifs

- Comprendre les principes du BDD (Behavior Driven Development)
- Découvrir la syntaxe Gherkin
- Configurer Cucumber avec Playwright
- Écrire des scénarios Gherkin et leurs step definitions
- Utiliser les hooks `Before` et `After` pour initialiser et nettoyer le contexte de test

1. Introduction au BDD et à Gherkin

1.1 Le BDD (Behavior Driven Development)

Le BDD est une approche qui décrit le comportement attendu d'une application à travers des scénarios exprimés dans un langage naturel.
Les tests deviennent lisibles par les développeurs, les testeurs et les responsables produit.

1.2 Le langage Gherkin

Gherkin est une syntaxe textuelle structurée permettant d'écrire des scénarios de test.

Exemple :

```
Feature: Gestion des tâches
  Scenario: Ajouter une tâche
    Given je suis sur la page TodoMVC
    When j’ajoute la tâche "Acheter du café"
    Then la tâche "Acheter du café" est visible dans la liste
```

2. Mise en place du projet Cucumber + Playwright

2.1 Initialisation

Créer un nouveau dossier pour ce TP :

```
mkdir tp-gherkin
cd tp-gherkin
npm init playwright@latest
```

Installer les dépendances nécessaires :

```
npm install --save-dev @cucumber/cucumber ts-node typescript
```

2.2 Configuration TypeScript

Créer un fichier `tsconfig.json` à la racine du projet :

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "commonjs",
    "lib": ["ES2020", "DOM"],
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "resolveJsonModule": true,
    "types": ["node", "@cucumber/cucumber", "@playwright/test"]
  }
}
```

3. Création du premier scénario Gherkin

Créer un dossier `features` à la racine et y ajouter un fichier `todo.feature` :

```
Feature: Gestion des tâches
  Scenario: Ajouter une tâche
    Given je suis sur la page TodoMVC
    When j'ajoute la tâche "Acheter du café"
    Then la tâche "Acheter du café" est visible dans la liste
```

4. Création des step definitions

Créer le dossier `features/step_definitions` et ajouter le fichier `todo.steps.ts` :

```ts
import { Given, When, Then, Before, After } from '@cucumber/cucumber';
import { chromium, Browser, Page, expect } from '@playwright/test';

let browser: Browser;
let page: Page;

// Hook exécuté avant chaque scénario
Before(async () => {
  browser = await chromium.launch({
    headless: false, // ← Mode headed
    slowMo: 200 // Optionnel : ralentit les actions pour mieux voir
  });
  const context = await browser.newContext();
  page = await context.newPage();
});

// Hook exécuté après chaque scénario
After(async () => {
  await browser.close();
});

Given('je suis sur la page TodoMVC', async () => {
  await page.goto('https://demo.playwright.dev/todomvc');
});

When('j’ajoute la tâche {string}', async (task: string) => {
  await page.getByPlaceholder('What needs to be done?').fill(task);
  await page.keyboard.press('Enter');
});

Then('la tâche {string} est visible dans la liste', async (task: string) => {
  await expect(page.getByText(task)).toBeVisible();
});
```

5. Exécution du scénario

Lancer le test avec la commande suivante :

```
npx cucumber-js --require-module ts-node/register --require features/step_definitions/**/*.ts
```

6. Ajout d’un deuxième scénario

Modifier le fichier `todo.feature` pour inclure un deuxième scénario :

```
Feature: Gestion des tâches
  Scenario: Ajouter une tâche
    Given je suis sur la page TodoMVC
    When j’ajoute la tâche "Acheter du café"
    Then la tâche "Acheter du café" est visible dans la liste

  Scenario: Supprimer une tâche
    Given je suis sur la page TodoMVC
    When j’ajoute la tâche "Faire le ménage"
    And je supprime la tâche "Faire le ménage"
    Then la tâche "Faire le ménage" n’est plus visible dans la liste
```

Implémenter la nouvelle step definition correspondante :

```
When('je supprime la tâche {string}', async (task: string) => {
  const todoItem = page.locator(`xpath=//label[text()="${task}"]/..`);
  await todoItem.hover();
  await todoItem.locator('.destroy').click();
});
```

Et compléter le `Then` :

```
Then('la tâche {string} n'est plus visible dans la liste', async (task: string) => {
  await expect(page.getByText(task)).toHaveCount(0);
});
```

7. Exercice pratique

1. Ajouter un scénario “Marquer une tâche comme terminée”.

   ```
   Scenario: Marquer une tâche comme terminée
     Given je suis sur la page TodoMVC
     When j'ajoute la tâche "Lire un livre"
     And je coche la tâche "Lire un livre"
     Then la tâche "Lire un livre" apparaît comme terminée
   ```

2. Implémenter les step definitions nécessaires :
   - Une step pour cocher une tâche
   - Une assertion pour vérifier que la classe CSS `"completed"` est appliquée
3. Exécuter les trois scénarios et vérifier leur réussite.

8. Conclusion

À la fin de ce TP, vous devez être capables de :

- Écrire des scénarios Gherkin lisibles
- Mapper des steps à du code Playwright
- Exécuter des tests BDD avec Cucumber

- Utiliser les hooks `Before` / `After` pour gérer les contextes de test
- Organiser les fichiers `features` et `step_definitions`