Automatisation des tests avec Playwright, TypeScript, Cucumber, et Jenkins

Plan du cours

---

1. Introduction à Playwright et TypeScript

Playwright est une bibliothèque open-source pour l'automatisation des navigateurs (Chromium, Firefox, WebKit). Elle permet d'effectuer des tests de bout en bout en simulant les interactions utilisateur. **TypeScript**, avec son typage statique, facilite la maintenance et améliore la robustesse du code.

2. Prérequis

- **Node.js** (version 14 ou supérieure)
- **NPM** ou **Yarn** pour la gestion des dépendances
- **Jenkins** installé et configuré pour l'intégration continue

3. Installation de Playwright et TypeScript

Créer un nouveau projet Node.js

```
bash
Copier le code
mkdir playwright-ts-cucumber
cd playwright-ts-cucumber
npm init -y
```

Installer les dépendances nécessaires

```
bash
Copier le code
npm install --save-dev playwright typescript ts-node @types/node
npm install --save-dev @cucumber/cucumber @cucumber/pretty-formatter
```

Configuration de TypeScript

Ajoutez un fichier `tsconfig.json` :

```
json
Copier le code
{
  "compilerOptions": {
    "target": "ES6",
```

```
    "module": "commonjs",
    "strict": true,
    "esModuleInterop": true,
    "outDir": "./dist",
    "moduleResolution": "node"
  },
  "include": ["./src/**/*.ts"]
}
```

4. Configuration de Playwright

Installer les navigateurs

bash
Copier le code
npx playwright install

Créer un fichier `playwright.config.ts`

Ce fichier contient les paramètres de configuration pour Playwright.

typescript
Copier le code

```
import { PlaywrightTestConfig } from '@playwright/test';

const config: PlaywrightTestConfig = {
  use: {
    browserName: 'chromium',
    headless: false,
    screenshot: 'on',
    video: 'retain-on-failure'
  },
  testDir: './tests',
  retries: 2
};

export default config;
```

5. Intégration de Cucumber

Configurer Cucumber

Créez un fichier `cucumber.ts` pour l'initialisation :

typescript
Copier le code

```
import { setDefaultTimeout } from '@cucumber/cucumber';
import { Before, After, Given, When, Then } from '@cucumber/cucumber';

setDefaultTimeout(60 * 1000);

Before(() => {
  console.log('Démarrage d\'un nouveau scénario');
});

After(() => {
  console.log('Scénario terminé');
});
```

```
Given('I navigate to the application', async () => {
  // Navigation
});

When('I perform some action', async () => {
  // Actions
});

Then('I expect some result', async () => {
  // Résultat attendu
});
```

Configurer les fichiers Gherkin

Créez un répertoire `features` et ajoutez un fichier `example.feature` :

```
gherkin
Copier le code
Feature: Example Feature

  Scenario: Navigating to a website
    Given I navigate to the application
    When I perform some action
    Then I expect some result
```

6. Automatisation avec Jenkins

Créer un Job Jenkins pour exécuter les tests

1. **Installer les plugins nécessaires :** Installez les plugins Jenkins pour **NodeJS**, **Cucumber Reports**, et **Git**.
2. **Configurer le Job :** Créez un nouveau Job Jenkins en tant que projet de pipeline.

Exemple de `Jenkinsfile`

```
groovy
Copier le code
pipeline {
    agent any

    triggers {
        cron('H H * * *') // Planification quotidienne à minuit
    }

    stages {
        stage('Install Dependencies') {
            steps {
                sh 'npm install'
            }
        }
        stage('Run Tests') {
            steps {
                sh 'npx cucumber-js --require-module ts-node/register --
require src/**/*.ts'
            }
        }
    }

    post {
        always {
```

```
            archiveArtifacts artifacts: '**/reports/*.json',
allowEmptyArchive: true
            cucumber 'reports/*.json'
        }
    }
}
```

7. Écriture de tests avec Playwright et Cucumber

Exemple de test avec Playwright

Créez un fichier `step-definitions.ts` :

```
typescript
Copier le code
import { Given, When, Then } from '@cucumber/cucumber';
import { chromium, Browser, Page } from 'playwright';

let browser: Browser;
let page: Page;

Given('I open the browser', async () => {
  browser = await chromium.launch({ headless: false });
  page = await browser.newPage();
});

When('I navigate to {string}', async (url: string) => {
  await page.goto(url);
});

Then('I should see the title {string}', async (expectedTitle: string) => {
  const title = await page.title();
  if (title !== expectedTitle) {
    throw new Error(`Expected title to be ${expectedTitle} but got
${title}`);
  }
  await browser.close();
});
```

8. Exécution des tests

Pour exécuter les tests, utilisez la commande suivante :

```
bash
Copier le code
npx cucumber-js --require-module ts-node/register --require src/**/*.ts
```

9. Planification des tests dans Jenkins

- **Déclenchement automatique :** Configurez le déclencheur "Poll SCM" pour exécuter les tests après chaque commit.
- **Déclenchement manuel :** Exécutez les tests manuellement en accédant au Job Jenkins et en cliquant sur "Build Now".

10. Bonnes pratiques et optimisation

Utilisation des hooks

Utilisez les hooks `BeforeAll`, `AfterAll`, `Before`, et `After` pour gérer les étapes globales comme le lancement et la fermeture du navigateur.

```
typescript
Copier le code
import { BeforeAll, AfterAll } from '@cucumber/cucumber';
import { chromium } from 'playwright';

BeforeAll(async () => {
  global.browser = await chromium.launch({ headless: false });
});

AfterAll(async () => {
  await global.browser.close();
});
```

Modularisation

- **Structurez votre code** : Créez des classes pour les pages (`Page Object Model`) pour faciliter la réutilisation du code.
- **Divisez les tests** : Séparez les tests longs en scénarios plus courts pour une meilleure traçabilité.

Génération de rapports

Configurez les rapports dans Jenkins pour inclure des captures d'écran et des vidéos des tests échoués.