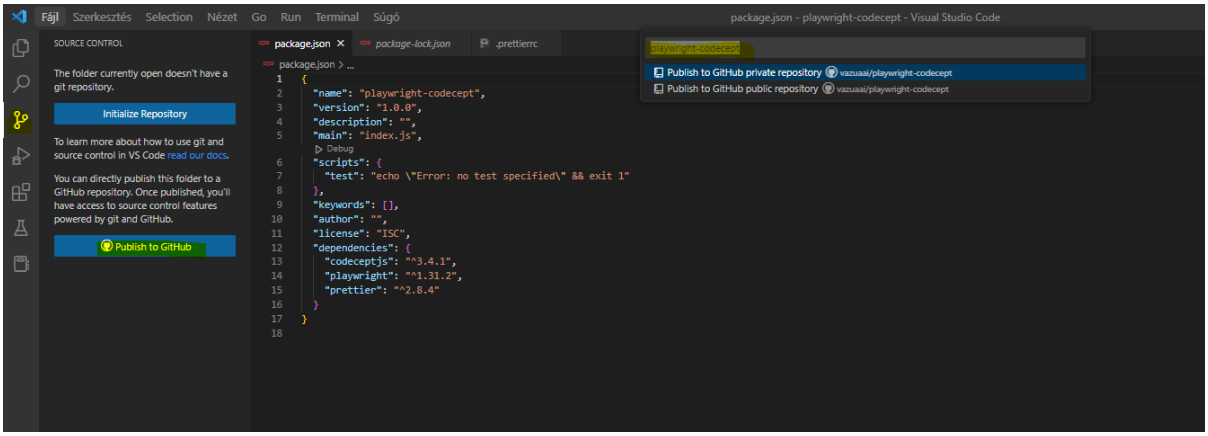
# Playwright - Cucumber

## Project létrehozása VS Code-ban + GitHub

1. Új project létrehozása VS Code-ban
2. A GitHub verziókövető rendszer összekapcsolása a filerendszerrel




✅Lérejött a projekt, a verziókövetés megvalósult.

---

## Inicializálás és dependenciák telepítése

1. Project inicializálása, létrejön a package.json file

```
npm init --yes
```

2. Dependenciák telepítése, ezek bekerülnek a package.json-ba

```
npm install playwright chai prettier @cucumber/cucumber cucumber-html-reporter
```

```
"dependencies": {
     "@cucumber/cucumber": "^8.11.1",
     "chai": "^4.3.7",
     "cucumber-html-reporter": "^5.5.0",
     "playwright": "^1.30.0",
     "prettier": "^2.8.4"
 }
```

3. A `.prettierrc` file létrehozása a project könyvtárban az alábbi tartalommal

```
{
   "semi": false,
   "singleQuote": false
 }
```

✅Inicializálódott a projekt, telepítésre kerültek a függőségek.

---

## Projekt struktúra elkészítése

1. Az alábbi mappák létrehozása a project mappában (`udemy-playwright-cucumber\`)
    - `features`
    - `step-definitions`
    - `page-objects`
    - `setup`
2. Az alábbi file-ok létrehozása (`udemy-playwright-cucumber\`)
    - `cucumber.js`
    - `reporter.js`

✅Elkészült a projekt struktúrája

---

## Global Assertions

1. A `setup` mappában `assertion.js` létrehozása
    a. Mivel sem a Cucumber sem a Playwright nem tartalmaz assertion library-t, ezért telepítettük a chai-t
2. Globális chai assertion-ök létrehozása, hogy szabadon használhatóak legyenek a kódban bárhol

```
const chai = require("chai")

 global.expect = chai.expect
 global.assert = chai.assert
 global.should = chai.should
```

✅Elkészültek a globális assertion-ök.

---

## Global Hooks

1. A `setup` mappában `hooks.js` létrehozása
    a. Defiánálásra kerülnek a hookok

```
const playwright = require("playwright")
 const { Before, After, BeforeAll, AfterAll } = require("@cucumber/cucumber")

 BeforeAll(async () => {
   console.log("Launch Browser")
   global.browser = await playwright["chromium"].launch({ headless: false })
 })

 AfterAll(async () => {
   console.log("Close Browser")
   await global.browser.close()
 })
```

```
Before(async () => {
  console.log("Create new context page")
  global.context = await global.browser.newContext()
  global.page = await global.context.newPage()
})

After(async () => {
  console.log("Close context and page")
  await global.context.close()
  await global.page.close()
})
```

2. Az assertion-ök és a hook-ok összkötése a `cucumber.js`-ben
    a. Ezzel a pattern-nel összeragasztottuk az assertion-öket, a hook-okat, step definition-öket és feature-ket. Így a Cucumber már elér mindent ami szükséges.

```
const common = `
     --require setup/assertions.js
     --require setup/hooks.js
     --require step-definitions/**/*.step.js
 `
 module.exports = {
   default: `${common} features/**/*.feature`,
 }
```

✅Elkészültek a globális hook-ok, illetva a Cucumber már minden szükségesről tud.

---

## Features

1. A `features` mappán belül `login.feature` létrehozása
2. A Feature és egy Scenario elkészítése

```
Feature: Login action

     As a user
     I want to login to the application

     Scenario: Login with valid credentials
         Given I visit a login page
         When I fill the login form with valid credentials
         Then I should see the home page
```

✅Elkészült az első feature file.

---

## Step Definitions

1. A `step-definitions\` mappában `login-step.js` létrehozása

```
const { Given, When, Then, defineStep } = require("@cucumber/cucumber")

 Given("I visit a login page", async function () {
   await page.goto("https://www.saucedemo.com/")
   })

 When("I fill the login form with valid credentials", async function () {
   await page.fill("#user-name", "standard_user")
   await page.fill("#password", "secret_sauce")
   await page.click("#login-button")
 })

 Then("I should see the home page", async function () {
   await page.waitForSelector(".inventory_list")
 })
```

✅Elkészültek az első step definition-ök.

---

## Test Run

1. A `package.json` file-ban a `“test”` script megírása

```
"scripts": {
     "test": "C:/Users/varjuz/repos/udemy-playwright-cucumber/node_modules/.bin/cucumber-js --require cucumber.js --require step-defini
   },
```

   - `C:/Users/varjuz/repos/udemy-playwright-cucumber/node_modules/.bin/cucumber-js`
   - `--require cucumber.js`
   - `--require step-definitions/**/*.js`
   - `-f json:cucumber_report.json` → cucumber_report.json létrehozása a riport készítéshez
   - `--publish-quiet` → A terminálon esetlegesen megjelenő kiegeszítő információk elrejtése.

2. A `“test”` script futtatása

```
npm run test
```

✅Lefutotott az első teszt.

---

## Page Object Model

1. A `page-objects` könyvtárban `login-page.js` létrehozása

```
class LoginPage {
   async navigateToLoginScreen() {
     await page.goto("https://www.saucedemo.com/")
   }

   async submitLoginForm() {
     await page.fill("#user-name", "standard_user")
     await page.fill("#password", "secret_sauce")
     await page.click("#login-button")
   }

   async assertUserIsLoggedIn() {
     await page.waitForSelector(".inventory_list")
   }
 }

 module.exports = { LoginPage }
```

2. A meglévő `step-definitions\login-step.js` refaktorálása a POM alapján

```
const { Given, When, Then } = require("@cucumber/cucumber")
 const { LoginPage } = require("../page-objects/login-page")

 const loginPage = new LoginPage()

 Given("I visit a login page", async function () {
   await loginPage.navigateToLoginScreen()
 })

 When("I fill the login form with valid credentials", async function () {
   await loginPage.submitLoginForm()
 })

 Then("I should see the home page", async function () {
   await loginPage.assertUserIsLoggedIn()
 })
```

✅Elkészült a POM és refaktorálásra kerültek step definition-ök.

## Cucumber HTML Reporter

A `reporter.js` a `cucumber_report.json`-t dolgozza fel, ebből készít egy fancy HTML reportot.

1. A `reporter.js` felparaméterezése az alábbi módon:

```
const reporter = require("cucumber-html-reporter")

 const options = {
   theme: "bootstrap",
   jsonFile: "cucumber_report.json",
   output: "reports/cucumber_report.html",
   reportSuiteAsScenario: true,
   scenarioTimestamp: true,
   launchReport: false,
   metadata: {
     "App Version": "1.0.0",
     "Test Environment": "STAGING",
     Browser: "Chrome 54.0",
     Platform: "Windows 10",
   },
 }

 reporter.generate(options)
```

2. `“report”` script elkészítése a `package.json` -ben

```
"scripts": {
     "test": "C:/Users/varjuz/repos/udemy-playwright-cucumber/node_modules/.bin/cucumber-js --require cucumber.js --require step-defini
     "report": "node reporter.js"
   },
```

3. Teszteset futtatása

```
npm run test
```

4. Report script futtatása

```
npm run report
```

5. HTML report file megtekintése

✅Elkészült a riport.

## Scenario Outlines

1. Scenarion Outline létrehozása a `features\login.feature` file-ban

```
Scenario Outline: Try to login with invalid credentials
        Given I visit a login page
        When I fill the login form with "<username>" and "<password>"
        Then I wait for 3 seconds

        Examples:
            | username | password |
            | fail-1   | fail-1   |
            | fail-2   | fail-2   |
            | fail-3   | fail-3   |
```

2. A `step-definitions\login-step.js` -ben a szükséges step definition-ök létrehozása

```
Then("I wait for 3 seconds", async function () {
   await loginPage.pause()
 })

 When(
   /^I fill the login form with "([^"]*)" and "([^"]*)"$/,
   async function (username, password) {
     await loginPage.submitLoginWithParameters(username, password)
   }
 )
```

3. A `page-objects\login-page.js` -ben a szükséges függvények létrehozása

```
async submitLoginWithParameters(username, password) {
     await page.fill("#user-name", username)
     await page.fill("#password", password)
     await page.click("#login-button")
   }

   async pause() {
     await page.waitForTimeout(3000)
   }
```

✅Elkészült egy Scenario Outline.