==========

Tutor: **Kaniel Outis**

Reference: **UDEMY**

Course: **Automated Software Testing with Playwright**

Content: **Summary of the course**

==========

1. Course URL:
   https://www.udemy.com/course/automated-software-testing-with-playwright/

2. Document prepared by: **Rajat Verma**
   a. https://www.linkedin.com/in/rajat-v-3b0685128/
   b. https://github.com/rajatt95
   c. https://rajatt95.github.io/

Softwares:

1. Programming language - Javascript, Typescript
2. IDE - VS Code
   a. https://code.visualstudio.com/download
   b. Plugin
      i. vscode-pdf
3. Engine:
   a. https://nodejs.org/en/download/
4. Playwright:
   a. https://playwright.dev/
   b. npm init playwright

---

1. **Learnings from Course (UDEMY - KO - Playwright)**
   a. **Links:**
      i. Playwright:
         1. https://playwright.dev/
         2. Page Object Model:
            a. https://playwright.dev/docs/test-pom
         3. https://github.com/topics/playwright

      ii. Practice website:
         1. https://example.com/
         2. http://zero.webappsecurity.com/

iii. Percy:
    1. https://docs.percy.io/docs/playwright
iv. API testing:
    1. https://reqres.in/
v. Jenkins:
    1. https://www.jenkins.io/download/
vi. CodeceptJS:
    1. https://codecept.io/basics/
    2. https://codecept.io/playwright/
    3. https://github.com/codeceptjs/configure#setcommonplugins

—----------

b. Playwright:

i. Developed by Microsoft
ii. Multiple Languages Support
    1. Javascript, Typescript, Java, Python, C#
iii. By default,
    1. Playwright runs the tests in
        a. Headless mode
        b. Chromium Browser

—----------

c. Playwright features:

i. Zero Configuration
ii. Very fast and Stable
    1. Better than Selenium and Cypress
iii. Parallel Cross Browser Testing
iv. Execution mode
    1. Headless, Normal
v. No Flaky test
    1. Auto-wait
vi. Annotations:
    1. Skip (To Skip the test cases)
    2. Only (To run only some test cases from the spec file)
    3. Describe (To group multiple test cases together)
vii. Tagging:
    a. Execute only Smoke:
        i. npx playwright test tests/*/10_Tagging.spec.ts --headed --browser=all --grep @Smoke
    b. Execute other than Smoke
        i. **npx playwright test tests/*/10_Tagging.spec.ts --headed --browser=all --grep-invert @Smoke**

—----------
—----------

viii. **Reporters**:
1. Line
    a. **playwright test tests/*/06_Assertions.spec.ts --reporter=line**
2. List (Playwright uses by default)
    a. **playwright test tests/*/06_Assertions.spec.ts --reporter=list**
3. Dot
    a. **playwright test tests/*/06_Assertions.spec.ts --reporter=dot**
4. JUnit
    a. **playwright test tests/*/06_Assertions.spec.ts --reporter=junit**
5. HTML (Best)
    a. **playwright test tests/*/06_Assertions.spec.ts --reporter=html**
    b. To open the HTML reports:
        i. **npx playwright show-report**
    c. Includes:
        i. Test steps
        ii. Screenshot
        iii. Video
        iv. Trace

ix. **Screenshots**:
1. Element, Visible Screen, Full Page

x. **Hooks**:
1. BeforeAll (Executes first in the spec file)
2. BeforeEach (Executes before every test)
3. Test
4. AfterEach (Executes after every test)
5. AfterAll (Executes last in the spec file)

xi. Tests Execution:
1. Example: 3 test cases in 1 spec file
    a. **Default**:
        i. Runs sequentially
        ii. If 2nd test case, it will go and execute 3rd test case
    b. Serial
        i. Runs sequentially
        ii. If 2nd test case, it will not execute 3rd test case (3rd will be skipped)

c. **Parallel**
    i. Runs parallely

—----------

xii. **Visual Testing**

1. **Full Page Snapshot:**
    a. **expect(await page.screenshot()).toMatchSnapshot('homepage.png')**
2. **Single Element Snapshot:**
    a. **expect(await pageElement.screenshot()).toMatchSnapshot('page-title.png')**
3. Page Objects Model with Snapshots
4. Update snapshots:
    a. Update Snapshots:
        i. **npx playwright test tests/*/35_Tests_Visual_POM.spec.ts --config=playwright-custom.config.ts --project=Webkit --update-snapshots**

xiii. **Rest API Testing**

1. **HTTP Methods:**
    a. GET
        i.
```
const response = await request.get(`${baseUrl}/users/3`)
```
    b. POST
        i.
```
test('POST Request - Create New User', async ({ request }) => {
  const response = await request.post(`${baseUrl}/user`, {
    //Request Body
    data: {
      id: 1000,
    },
  })
```
    c. PUT
        i.
```
test('PUT Request - Update User', async ({ request }) => {
  //This is a PUT Request
  const response = await request.put(`${baseUrl}/users/2`, {
    //Request Body
    data: {
      name: 'new name',
      job: 'new job',
    },
  })
```
    d. DELETE

i.

```
test('DELETE Request - Delete User', async ({ request }) => {
  //This is a DELETE Request
  const response = await request.delete(`${baseUrl}/users/2`)
```

—----------

—----------

2. **Extract the Response:**
   a.
```
//Extract the Response
const responseBody = JSON.parse(await response.text())
```
3. **Assertions:**
   a. Response Status Code
      i.
```
//Assertion for Response Status Code
expect(response.status()).toBe(200)
```
   b. Response JSON data:
      i.
```
data: {
  id: 1,
  email: 'george.bluth@reqres.in',
  first_name: 'George',
  last_name: 'Bluth',
  avatar: 'https://reqres.in/img/faces/1-image.jpg'
},
support: {
  url: 'https://reqres.in/#support-heading',
  text: 'To keep ReqRes free, contributions towards server costs are appreciated!'
}
}
```
      ii.
```
//Assertions
expect(response.status()).toBe(200)
expect(responseBody.data.id).toBe(1)
expect(responseBody.data.first_name).toBe('George')
expect(responseBody.data.last_name).toBe('Bluth')
expect(responseBody.data.email).toBeTruthy()
```

—----------

d. Framework practices:

i. Custom Functions

ii. Node scripts

1.

iii. Page Objects Model:

1. Design Pattern
2. Benefits:
   a. Readability
   b. Re-usability
   c. Maintenance
3. Components

—----------

—----------

e. **Playwright Configuration properties:**
   i. Timeout
   ii. Retries
   iii. testDir
   iv. Use
      1. Headless
      2. Viewport
      3. ActionTimeout
      4. IgnoreHTTPSErrors
      5. Video
      6. Screenshot
   v. Projects:
      1. Name
      2. Use
         a. BrowserName

—----------

f. **Playwright methods:**
   i. Go to URL
      1. **await page.goto("https://example.com/")**
   ii. Find element:
      1. **const heading_PageTitle = await page.locator('h1')**
   iii. Click on Element:
      1. Text:
         a. **await page.click('text=Sign in')**
      2. CSS Selector:
         a. **await page.click('button')** // Tag
         b. **await page.click('#id')** // ID
         c. **await page.click('.class')** // Class
      3. Only visible:
         a. **await page.click('.submit-button:visible')** //Class
      4. Combinations:
         a. **await page.click('#username .first')** // **ID** and **Class**
      5. Xpath:
         a. **await page.click('//button')** // Tag

iv. Fill value in text box:

1. **await page.type('#user_login','some username')**

—----------

—----------

v. Screenshots:

1. **await page.screenshot({path:'./screenshots/Screenshot_FullPage.png', fullPage:true})**
2. **await page.screenshot({path:'./screenshots/Screenshot_VisibleScreen.png'})**
3. **await profile_editable_area.screenshot({path:'./screenshots/Screenshot_Element.png'})**

vi. Playwright Inspector:

1. **await page.pause()**

vii. Wait for Element:

1. **await page.waitForSelector('#feedback-title')**

viii. Wait for sometime

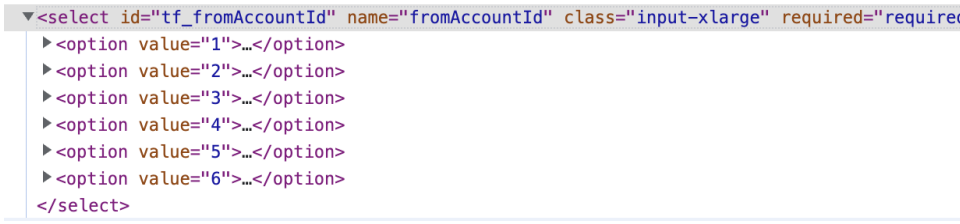
1. **await page.waitForTimeout(3000)**

ix. Keyboard simulations

1. **await page.keyboard.press('Enter')**

x. Dropdown:

1. Select-Option Tag:
   a. **await page.selectOption('#tf_fromAccountId', '2')**
   b.



—----------

—----------

g. Assertions with Playwright:

i. Page level:

1. **await expect(page).toHaveURL('https://example.com/')**
2. **await expect(page).toHaveTitle('Example Domain')**

ii. Element level:

1. **await expect(element_heading).toBeVisible()**
    a. Wait for Element:
        i. **await page.waitForSelector('#feedback-title')**
2. **await expect(element_heading).toHaveAttribute('class','alert alert-error')**
3. **await expect(element_heading).toHaveCSS()**
4. **await expect(element_heading).toContainText(Domain')**
5. **await expect(element_heading).toHaveText('Example Domain')**
6. **await expect(element_heading).toHaveCount(1)**
7. **await expect(element_noExistence).not.toBeVisible()**
8. **await expect(nameInput).toBeEmpty()**
9.

—----------

h. Tips and Trickes:

i. TestInfo object:

1.
```
test('TestInfo Object',async ({page},TestInfo)=>{
    console.log(TestInfo.title)
    console.log(TestInfo.file)
    console.log(TestInfo.outputDir)
    console.log(TestInfo.timeout)
})
```

ii. Skip Browser:

1.
```
test('Test Skip Browser',async ({page, browserName})=>{

    //Skip for Chromium browser
    test.skip(browserName === 'chromium','Feature is not yet implemented for Chromium browser')

    await page.goto('https://example.com/')

})
```

iii. Fixme Annotation:

1.
```
test('Test Fixme Annotation',async ({page, browserName})=>{

    //Skip for Chromium browser
    test.fixme(browserName === 'chromium','Test is not stable, needs Revision')

    await page.goto('https://example.com/')
```

iv. Retries:

1. **playwright test tests/*/06_Assertions.spec.ts --retries=1**

—----------

—----------

v. Parameterization (Multiple Set of Data):

1.
```
//Rahul Shetty
test.describe.configure( { mode : 'parallel' } );

//test.describe.parallel('Test suite - Execution: PARALLEL',()=>{})//describe

    for(const automationTool of automationTools){
        test("Running Test for "+automationTool, async({page})=>{
            await page.goto('https://www.google.com/')
            await page.type("[name='q']",automationTool)
```

vi. Mouse Movement Simulation:

1.
```
test('Mouse Movement Simulation',async ({page})=>{

    await page.goto('https://example.com/')

    await page.mouse.move(0,0)
    await page.waitForTimeout(2000)
    await page.mouse.down()
    await page.waitForTimeout(2000)
    await page.mouse.move(0,100)
    await page.waitForTimeout(2000)
    await page.mouse.up()
    await page.waitForTimeout(2000)

})//test
```

vii. Multiple Browser Pages

1.
```
test('Multiple Browser_Tabs',async ({browser})=>{

    const context = await browser.newContext()
    const page1 = await context.newPage()
    const page2 = await context.newPage()
    const page3 = await context.newPage()

    await page1.goto('https://rajatt95.github.io/')
    await page2.goto('https://www.linkedin.com/in/rajat-v-3b0685128/')
    await page3.goto('https://github.com/rajatt95')

    await page1.waitForTimeout(5000)

})//test
```

viii. Device Emulation

1. **npx playwright open --device="iPhone 11" https://www.google.com/**

ix. Generate PDF files

1. **npx playwright pdf https://github.com/rajatt95 ./PDFs/Github_Profile-Rajatt95.pdf**
2. **npx playwright pdf https://rajatt95.github.io/ ./PDFs/Github_Page-Rajatt95.pdf**

x. Generate Customized Screenshots

1. **npx playwright screenshot --device="iPhone 11" --color-scheme=dark --wait-for-timeout=3000 https://github.com/rajatt95 ./screenshots/iPhone11-dark-github_profile-rajatt95.png**
2. **npx playwright screenshot --device="iPhone 11" --color-scheme=light --wait-for-timeout=3000 https://github.com/rajatt95 ./screenshots/iPhone11-light-github_profile-rajatt95.png**

xi. Emulate Browser language & Timezone

1. **npx playwright open --timezone="Europe/Rome" --lang="it-IT" google.com**

xii. Data Helper:

1. Random Number
   a. **Math.floor(Math.random() * 10000 + 1)**
2. Random String
   a. **Math.random().toString(36).substring(2,10)**
3. Random Email
   a. **Math.round(Math.random()*100000)+"@email.com"**

—----------

i. **Commands:**

i. **npm init**
   1. To create project

ii. **npm init –yes**
   1. No need to give all information
   2. Package.json file will be created

iii. **npm install prettier**
   1. To install node package ‘prettier’

iv. **npm install @playwright/test**
   1. To install playwright test

v. **npx playwright install**
   1. To install Playwright

vi. **npx playwright test**
   1. To execute Tests

vii. To execute the scripts in
    1. Normal/headed mode:
        a. **npx playwright test --headed**
            i. This runs in Chromium browser (Default)
        b. Firefox Browser:
            i. **npx playwright test --headed --browser=firefox**
        c. Webkit Browser:
            i. **npx playwright test --headed --browser=webkit**
        d. All Browser:
            i. **npx playwright test --headed --browser=all**

viii. To execute the specific spec file:
    1. **npx playwright test tests/example.spec.ts --headed --browser=all**

ix. Tagging:
    c. Execute only Smoke:
        i. npx playwright test tests/*/10_Tagging.spec.ts --headed --browser=all --grep @Smoke
    d. Execute other than Smoke
        i. **npx playwright test tests/*/10_Tagging.spec.ts --headed --browser=all --grep-invert @Smoke**

x. Commands to execute the test cases using Playwright config file:
    1. npx playwright test tests/*/06_Assertions.spec.ts --config=playwright.config.ts
    2. Execution on specific project:
        a. playwright test tests/*/06_Assertions.spec.ts --config=playwright.config.ts --project=Chromium
        b. playwright test tests/*/06_Assertions.spec.ts --config=playwright.config.ts --project=Webkit
        c. playwright test tests/*/06_Assertions.spec.ts --config=playwright.config.ts --project=Firefox

xi. **Reporters**:
    1. Line
        a. **playwright test tests/*/06_Assertions.spec.ts --reporter=line**
    2. List (Playwright uses by default)
        a. **playwright test tests/*/06_Assertions.spec.ts --reporter=list**
    3. Dot
        a. **playwright test tests/*/06_Assertions.spec.ts --reporter=dot**
    4. JUnit

a. **playwright test tests/*/06_Assertions.spec.ts --reporter=junit**

5. HTML (Best)
   a. **playwright test tests/*/06_Assertions.spec.ts --reporter=html**
   b. To open the HTML reports:
      i. **npx playwright show-report**

xii. Retries:

1. **playwright test tests/*/06_Assertions.spec.ts --retries=1**

—----------

—----------

xiii. Device Emulation

1. **npx playwright open --device="iPhone 11" https://www.google.com/**

xiv. Generate PDF files

1. **npx playwright pdf https://github.com/rajatt95 ./PDFs/Github_Profile-Rajatt95.pdf**
2. **npx playwright pdf https://rajatt95.github.io/ ./PDFs/Github_Page-Rajatt95.pdf**

xv. Generate Customized Screenshots

1. **npx playwright screenshot --device="iPhone 11" --color-scheme=dark --wait-for-timeout=3000 https://github.com/rajatt95 ./screenshots/iPhone11-dark-github_profile-rajatt95.png**
2. **npx playwright screenshot --device="iPhone 11" --color-scheme=light --wait-for-timeout=3000 https://github.com/rajatt95 ./screenshots/iPhone11-light-github_profile-rajatt95.png**

xvi. Emulate Browser language & Timezone

1. **npx playwright open --timezone="Europe/Rome" --lang="it-IT" google.com**

xvii. Playwright_JS_BDD_Cucumber:

1. **npm init**
2. **npm install playwright chai prettier @cucumber/cucumber cucumber-html-reporter**
3. **Commands:**
   a. **npm run test**
   b. **npm run report**
4.

xviii. Playwright_JS_BDD_CodeceptJS:

1. **npm init –yes**
   a. No need to give all information

b. Package.json file will be created

2. **npm install playwright codeceptjs prettier**
3. **npx codeceptjs init**
4. **npx codeceptjs run**
    a. Run all tests
5. **npx codeceptjs run e2e_test.js**
    a. Run specific test
6. **npx codeceptjs gpo**
    a. Generate Page Object

—----------

j. Integration with Playwright:

i. BDD Style with
    1. Cucumber
    2. CodeceptJS

ii. CI/CD integration
    1. Jenkins

iii. 3rd party integrations
    1. Playwright with Mocha
    2. Playwright with Jest
    3. Playwright with Ava

—----------

==========

1. **To connect:**
    a. https://www.linkedin.com/in/rajat-v-3b0685128/
    b. https://github.com/rajatt95
    c. https://rajatt95.github.io/

==========