Playwright Test Runner: Codegen (Automatic Test Recording)

In Playwright, we can record our tests using the Test Runner and the **Codegen Plugin**. This feature automatically records your actions on the web page and generates the corresponding test script.

How to Create Tests in Playwright

There are **two main ways** to create tests in Playwright:

1. **By Writing Your Own Script**
2. **Using Test Runner with Codegen Plugin (Automatic Recording)**

Using the Playwright Test Runner with Codegen Plugin

The **Codegen Plugin** allows you to automatically record your test steps by performing actions in the browser. It generates the test script based on your interactions.

Steps to Use Playwright Codegen (Test Runner)

1. **Open Terminal**
   Execute the following command to start the Codegen tool:

   ```
   npx playwright codegen
   ```

2. **Browser and Playwright Inspector**
   After running the above command, the following will happen:
   - A browser window will open (by default, Chromium).
   - The **Playwright Inspector** will launch alongside the browser.

   In the **Playwright Inspector**, you will find options such as:

   - **Record** (Start/Stop recording)
   - **Copy Code** (Copy the generated test code)
   - **Play/Stop** (Control test playback)
   - **Pause** (Pause recording)
   - **Pick Locator** (Helps identify locators easily by clicking on elements in the browser)

Performing Actions to Record Test Steps

Now, you can interact with the browser to record your test actions. For example:

1. **Launch the application**
2. **Click the login link**
3. **Enter username and password**
4. **Click the submit button**
5. **Verify the logged-in user's name**
6. **Verify the presence of the logout button**

As you perform these actions, Playwright will automatically record them.

Stopping the Recording

Once you've performed all the desired actions in the browser:

- Click the **Record** button in Playwright Inspector to stop recording.
- Playwright will then generate the corresponding code for the actions you performed.

Example of the Default Code Setup

After stopping the recording, Playwright generates the test code. Here's an example of what the code will look like:

```
import { test, expect } from '@playwright/test';

test('user login and logout', async ({ page }) => {
  // Go to the website
  await page.goto("https://...");

  // Click on the login link
  await page.getByRole('link', { name: 'login' }).click();

  // Fill in the login credentials
  await page.locator('#username').fill('Yogesh');
  await page.locator('#password').fill('testLife@123');

  // Click the login button
  await page.getByRole('button', { name: 'login' }).click();

  // Wait for user link to appear after login
  const userLink = await page.getByRole('link', { name: 'user' });

  // Verify if the logged-in user's name is correct
  await expect(userLink).toHaveText('Yogesh'); // Verifying text content of
user link

  // Verify if the logout link is displayed after login
  const logoutLink = await page.getByRole('link', { name: 'logout' });
  await expect(logoutLink).toHaveText('Logout'); // Verifying the logout
link text
});
```

Example DOM Structure for above code

The DOM (HTML) structure of the page being tested might look something like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Example</title>
</head>
<body>
  <!-- Login link -->
  <a href="/login" role="link" name="login">Login</a>

  <div id="loginForm">
    <input type="text" id="username" placeholder="Username">
    <input type="password" id="password" placeholder="Password">
    <button role="button" name="login">Login</button>
  </div>

  <!-- User section visible after login -->
  <div id="userSection">
    <a href="/user" role="link" name="user">Yogesh</a> <!-- The logged-in
user's name -->
    <a href="/logout" role="link" name="logout">Logout</a> <!-- Logout link
-->
  </div>
</body>
</html>
```

Pick Locator: Identifying Locators Easily

Playwright's **Pick Locator** feature helps identify locators easily by allowing you to click on an element in the browser and automatically retrieve its locator.

Example: Locating an Image

If you want to locate an image, follow these steps:

1. Click the **Pick Locator** button in Playwright Inspector.
2. In the browser, click on the image element.

For example, an image element may look like this:

```
<img id="sampleImage" src="image.jpg" alt="Sample Image">
```

To generate the locator for this image, Playwright will suggest the following code:

```
await page.getByAltText("Sample Image");
```

Target Option: Choosing the Language

By default, Playwright generates the test code in **JavaScript** (Node.js). However, you can change the language of the generated code.

To do this, follow these steps:

1. Click on the **Target** dropdown in the Playwright Inspector.
2. Choose your preferred language, such as:
   - **JavaScript**
   - **Python**
   - **Java**
   - **C#**

This will generate the test code in the selected language.

Handling Code Export Automatically

One limitation of Playwright's Codegen tool is that after you stop recording, you need to manually copy the generated code and paste it into a `.spec.js` file. However, there's an option to automate this process.

Solution: Automatic File Creation and Code Export

Instead of manually copying the code, you can execute the following command to automatically create a file and export the generated test code:

```
npx playwright codegen --output <file-path>
```

This will create the file at the specified location and save the generated code in it.

Additional Options Available with `npx playwright codegen`

Playwright provides several options to customize your test recording and code generation process. To see the available options, run:

```
npx playwright codegen --help
```

Here are some commonly used options:

- `-o, --output <file>`: Saves the generated script to a file.
- `--target <language>`: Specifies the language for the generated script (default is `javascript`). Options include:
  - `javascript`
  - `python`
  - `java`
  - `csharp`

- `--load-storage <file>`: Loads context storage state from the specified file.
- `--save-storage <file>`: Saves the storage state at the end of the session.
- `--viewport-size <size>`: Sets the viewport size (e.g., `"1280x720"`).
- `--device <device>`: Emulates a mobile device.
- `-h, --help`: Displays help for the command.

Key Points to Remember:

- **Test Creation**: You can create tests either manually by writing your own script or automatically using Playwright's Codegen tool.
- **Playwright Codegen**: Automatically generates test scripts by recording your actions in the browser.
- **Pick Locator**: Easily find element locators by clicking on elements in the browser.
- **File Export**: Automatically export generated code into a specified file using the `--output` flag.
- **Language Target**: Choose your preferred language for the generated code (JavaScript, Python, Java, or C#).

Playwright Test Runner: Using `npx playwright codegen` Commands for practice

The `npx playwright codegen` command is a powerful tool in Playwright that allows you to record user interactions in a browser, automatically generate test code, and save the results in a specific language or file format.

1.Basic Command: Opens Playwright Inspector & Records Actions

Command:

```
npx playwright codegen
```

What Happens?

- **Opens a Browser Window**: Launches a browser (default: Chromium).
- **Captures User Interactions**: Records all interactions like clicks, text input, navigation, etc.
- **Generates Playwright Code**: Automatically generates Playwright test code in JavaScript (default).

2.Open a Specific URL and Start Recording

Command:

```
npx playwright codegen example.com
```

What Happens?

- **Opens example.com**: Opens the specified URL (in this case, `example.com`) in a new browser window.
- **Starts Recording User Actions**: Records the actions you perform on the page.
- **Generates Playwright Test Code**: Generates test code based on your interactions.

3️⃣Generate Code in a Specific Language

You can generate code in different programming languages with the `--target` option.

(a) JavaScript (Default)

Command:

```
npx playwright codegen --target=javascript
```

What Happens?

- **Generates JavaScript Code**: The generated code will be in JavaScript (default behavior).

(b) Python

Command:

```
npx playwright codegen --target=python
```

Example Output (Python):

```
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    browser = p.chromium.launch()
    page = browser.new_page()
    page.goto("https://example.com")
    browser.close()
```

What Happens?

- **Generates Python Code**: The test code will be generated in Python using Playwright's Python API.

(c) Java

Command:

```
npx playwright codegen --target=java
```

Example Output (Java):

```
import com.microsoft.playwright.*;

public class Example {
  public static void main(String[] args) {
    try (Playwright playwright = Playwright.create()) {
      Browser browser = playwright.chromium().launch();
      Page page = browser.newPage();
      page.navigate("https://example.com");
      browser.close();
    }
  }
}
```

What Happens?

- **Generates Java Code**: The generated test code will be in Java, using Playwright's Java API.

4.Save Generated Code to a File (Automatically Create & Save)

Command:

```
npx playwright codegen --output ./tests/myTest.spec.js
```

or

```
npx playwright codegen -o ./tests/myTest.spec.js
```

What Happens?

- **Saves the Recorded Actions**: The generated Playwright test script is automatically saved in the specified file (`./tests/myTest.spec.js` in this example).

5.Save the Storage State at the End

Command:

```
npx playwright codegen --save-storage=storage.json
```

What Happens?

- **Saves Session Storage & Cookies**: After the recording, Playwright will save the session storage and cookies to a file named `storage.json`.
- **Useful for Logged-In Sessions**: This allows you to maintain a logged-in session across multiple runs by loading the storage state later.

6.Load a Previously Saved Storage State

Command:

```
npx playwright codegen --load-storage=storage.json
```

What Happens?

- **Loads Session Storage & Cookies**: Loads the session data (cookies and local storage) from `storage.json`.
- **Useful for Logged-In Sessions**: Helps you skip the login steps by loading a previously saved session, maintaining the user's authenticated state.

7.Set a Custom Viewport Size

Command:

```
npx playwright codegen --viewport-size=1280x720
```

What Happens?

- **Opens the Browser with a Custom Resolution**: The browser window will open with a custom viewport size (in this case, `1280x720` resolution).

8.Simulate a Mobile Device

Command:

```
npx playwright codegen --device="iPhone 13"
```

What Happens?

- **Simulates an iPhone 13 Environment**: The browser will emulate an iPhone 13, simulating its screen size, user agent, and touch events.

You can also check the available mobile device emulators in Playwright using the following command:

Command:

```
npx playwright devices
```

Example Output:

```
Available devices:
==================
iPhone 13 Pro (viewport: 390x844, DPR: 3)
iPhone 12 (viewport: 390x844, DPR: 3)
Pixel 5 (viewport: 393x851, DPR: 2.625)
Samsung Galaxy S21 (viewport: 412x915, DPR: 3)
iPad (viewport: 768x1024, DPR: 2)
iPad Mini (viewport: 768x1024, DPR: 2)
Desktop Chrome (viewport: 1280x720, DPR: 1)
Desktop Firefox (viewport: 1280x720, DPR: 1)
Desktop Safari (viewport: 1200x800, DPR: 1)
...
```

What Happens?

- **Device Information**: Playwright lists the available devices, including:
  - **Device Name** (e.g., iPhone 13 Pro, iPhone 12)
  - **Viewport Size** (width x height)
  - **Device Pixel Ratio (DPR)**

Each device emulation simulates screen size, touch events, and other mobile-specific behaviors.

Summary of Key Commands

- **Start Playwright Codegen**: `npx playwright codegen`
- **Open a Specific URL**: `npx playwright codegen <URL>`
- **Generate Code in a Specific Language**:
  - JavaScript (default): `npx playwright codegen --target=javascript`
  - Python: `npx playwright codegen --target=python`
  - Java: `npx playwright codegen --target=java`

File Management:

- **Save to File**: `npx playwright codegen --output <file-path>`
- **Save Storage State**: `npx playwright codegen --save-storage=<file-path>`
- **Load Storage State**: `npx playwright codegen --load-storage=<file-path>`

Browser and Device Configuration:

- **Set Custom Viewport Size**: `npx playwright codegen --viewport-size=<width>x<height>`
- **Simulate Mobile Device**: `npx playwright codegen --device="<device-name>"`
- **View Available Devices**: `npx playwright devices`
