Sonu Madheshiya

Playwright: Mastering the Essentials - Part 4

Mastering Playwright Locators Strategies : A Comprehensive Guide

Prepared By:

Sonu Madheshiya
Automation Tester

https://www.linkedin.com/in/sonumadheshiya/

https://www.linkedin.com/in/sonumadheshiya/

Playwright : Mastering the Essentials - Part 4

Mastering Playwright Locators Strategies : A Comprehensive Guide

Locators are crucial in Playwright for interacting with elements on a page. They provide a way to find and interact with elements using various strategies, ensuring actions are performed on the correct elements.

Quick Guide to Recommended Built-in Locators :

- page.getByRole(): Finds elements by explicit and implicit accessibility attributes.
- page.getByText(): Locates elements based on their text content.
- page.getByLabel(): Identifies form controls by the associated label's text.
- page.getByPlaceholder(): Finds input elements by their placeholder text.
- page.getByAltText(): Targets elements, typically images, by their text alternative.
- page.getByTitle(): Locates elements using their title attribute.
- page.getByTestId(): Identifies elements based on their data-testid attribute

```
// Finds an input field by its associated label and fills it with the name
await page.getByLabel('Name').fill('Sonu Madheshiya');

// Locates an input field by its associated label and fills it with the password
await page.getByLabel('Password').fill('sonu1234');

// Finds an input element by its placeholder text and fills it with the value
await page.getByPlaceholder('Search...').fill('Playwright');

// Finds a button by its role and name, then clicks it
await page.getByRole('button', { name: 'Log In' }).click();

// Checks if the text 'Hello, Sonu!' is visible on the page
await expect(page.getByText('Hello, Sonu!')).toBeVisible();

// Targets an image by its alternative text and verifies it's visible
await expect(page.getByAltText('Playwright Logo')).toBeVisible();

// Locates an element by its title attribute and clicks it
await page.getByTitle('Settings').click();

// Identifies an element based on its data-testid attribute and verifies its visibility
await expect(page.getByTestId('submit-button')).toBeVisible();
```

Locating Elements

Playwright provides various built-in locators to enhance test reliability. It's recommended to use user-facing attributes and explicit contracts, such as

1. Role Locators:

- **Method:** page.getByRole(role, { name: 'name' })
- **Description:** Finds elements based on their role and accessible name.
- **Example:**

```
// Clicks a button with the role 'button' and name 'Submit'
await page.getByRole('button', { name: 'Submit' }).click();
```

2. Label Locators:

- **Method:** page.getByLabel('label text')
- **Description:** Locates form controls by their associated label text.
- **Example:**

```
// Fills a text input field labeled 'Username'
await page.getByLabel('Username').fill('myUserName');
```

3. Placeholder Locators:

- **Method:** page.getByPlaceholder('placeholder text')
- **Description:** Finds input elements with a specific placeholder attribute.
- **Example:**

```
// Fills a phone number input with the placeholder 'Enter phone number'
await page.getByPlaceholder('Enter phone number').fill('123-456-7890');
```

4. Text Locators:

- **Method:** page.getByText(text, { exact: true })
- **Description:** Locates elements based on the text they contain, supporting exact match or regular expressions.
- **Example:**

```
// Asserts visibility of an element containing 'Hello, World!'
await expect(page.getByText('Hello, World!')).toBeVisible();

// Asserts visibility of an element with exact text 'Welcome to the site'
await expect(page.getByText('Welcome to the site', { exact: true })).toBeVisible()

// Asserts visibility of an element matching the regex 'user [0-9]+'
await expect(page.getByText(/user [0-9]+/i)).toBeVisible();
```

5. Alt Text Locators:

- **Method:** page.getByAltText('alt text')
- **Description:** Finds image elements based on their alt attribute.
- **Example:**

```
// Clicks an image with the alt text 'logo image'
await page.getByAltText('logo image').click();
```

6. Title Locators:

- **Method:** page.getByTitle('title text')
- **Description:** Locates elements by their title attribute.
- **Example:**

```
// Asserts that an element with the title 'Profile Picture' has the text 'User Pro
await expect(page.getByTitle('Profile Picture')).toHaveText('User Profile');
```

7. Test ID Locators:

- **Method:** page.getByTestId('test-id')
- **Description:** Finds elements using a custom test id attribute.
- **Example:**

```
// Clicks an element with the test id 'submit-form'
await page.getByTestId('submit-form').click();
```

8. CSS or XPath Locators:

- **Method:** page.locator('selector') or page.locator('xpath=//selector')
- **Description:** Finds elements using CSS or XPath selectors.
- **Example:**

```
// Clicks a button using a CSS selector
await page.locator('css=.submit-button').click();

// Clicks a button using an XPath selector
await page.locator('xpath=//button[@id="submitBtn"]').click();
```

Locating Elements in Shadow DOM:

Playwright supports locating elements within Shadow DOM by default, except for a few cases:

- **XPath Locators:** Do not work with elements inside Shadow DOM.
- **Closed-Mode Shadow Roots:** Are not supported.

Here's how you can interact with elements in Shadow DOM:

```
<x-details role="button" aria-expanded="true" aria-controls="inner-details">
  <div>Title</div>
  #shadow-root
    <div id="inner-details">Details</div>
</x-details>
```

```
// Click on the text 'Details'
await page.getByText('Details').click();

// Click on the <x-details> component
await page.locator('x-details', { hasText: 'Details' }).click();

// Ensure that <x-details> contains the text 'Details'
await expect(page.locator('x-details')).toContainText('Details');
```

Filtering Locators:

1. Filter by Text:

- Use locator.filter({ hasText: 'text' }) to find elements containing specific text.
- Example:

```
await page
    .getByRole('listitem')
    .filter({ hasText: 'Item B' })
    .getByRole('button', { name: 'Add to Bag' })
    .click();
```

2. Filter by Not Having Text:

- Use locator.filter({ hasNotText: 'text' }) to exclude elements containing specific text.
- Example:

```
await expect(page.getByRole('listitem').filter({ hasNotText: 'Sold' })).toHaveCount(5);
```

3. Filter by Child/Descendant:

- Filter elements based on the presence of a descendant element.
- Example:

```
await page
    .getByRole('listitem')
    .filter({ has: page.getByRole('heading', { name: 'Item B' }) })
    .getByRole('button', { name: 'Add to Bag' })
    .click();
```

4. Filter by Not Having Child/Descendant:

- Exclude elements based on the absence of a descendant element.
- Example:

```
await expect(page
    .getByRole('listitem')
    .filter({ hasNot: page.getByText('Item B') }))
    .toHaveCount(1);
```

Locator Operators :

1. Matching Inside a Locator:

- Chain methods to narrow down searches within a locator.
- Example:

```
const item = page.getByRole('listitem').filter({ hasText: 'Item B' });
await item.getByRole('button', { name: 'Add to Bag' }).click();
await expect(item).toHaveCount(1);
```

2.Matching Two Locators Simultaneously:

- Use locator.and() to match elements that meet both locators' criteria.
- Example:

```
const button = page.getByRole('button').and(page.getByTitle('Subscribe Now'));
```

3. Matching One of the Two Alternative Locators:

- Use locator.or() to match any of the alternative locators.
- Example:

```
const newEmail = page.getByRole('button', { name: 'Compose' });
const dialog = page.getByText('Confirm Security Settings');
await expect(newEmail.or(dialog).first()).toBeVisible();
if (await dialog.isVisible())
  await page.getByRole('button', { name: 'Dismiss' }).click();
await newEmail.click();
```

4. Matching Only Visible Elements:

- Filter elements to interact only with visible ones.
- Example:

```
await page.locator('button').locator('visible=true').click();
```

Other Use Cases:

1.Do Something with Each Element in the List:

- Iterate over list items to perform actions.
- Example:

```
for (const row of await page.getByRole('listitem').all())
  console.log(await row.textContent());
```

- **2.** Iterate Using Regular For Loop:
- Example:

```
const rows = page.getByRole('listitem');
const count = await rows.count();
for (let i = 0; i < count; ++i)
  console.log(await rows.nth(i).textContent());
```

Stay tuned with Sonu Madheshiya on LinkedIn for more interesting content on automation testing.