

Single Page WebTable Handling in Playwright

What is a WebTable?

A web table looks like a table with rows and columns that displays data on a web page. In a single-page web table, all the data is displayed without requiring navigation between pages.

Example DOM Structure

```
<table id='paginationTable'>
  <thead>
    <tr>
      <th>sno</th>
      <th>name</th>
      <th>company</th>
      <th>employeeOfTheMonth</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Yogesh</td>
      <td>Comcast</td>
      <td>
        <input type='checkbox'>
      </td>
    </tr>
    <tr>
      <td>2</td>
      <td>Siva</td>
      <td>Infosys</td>
      <td>
        <input type='checkbox'>
      </td>
    </tr>
    <tr>
      <td>3</td>
      <td>Indu</td>
      <td>HindiTution</td>
      <td>
        <input type='checkbox'>
      </td>
    </tr>
    <tr>
      <td>4</td>
      <td>Chandru</td>
      <td>RenugaMill</td>
      <td>
        <input type='checkbox'>
      </td>
    </tr>
  </tbody>
</table>
```

Table:

sno	name	company	employeeOfTheMonth
1	Yogesh	Comcast	[]
2	Siva	Infosys	[]
3	Indu	HindiTution	[]
4	Chandru	RenugaMill	[]

The checkboxes are represented as [] to indicate they are currently unchecked.

Step 1: Capture the Table

First, capture the table element using Playwright:

```
const paginationTable = await page.locator('#paginationTable');
```

Step 2: Extract Rows and Columns

Extract the total number of rows and columns:

```
const columns = await paginationTable.locator('thead tr th');
console.log('Total number of columns:', await columns.count());

const rows = await paginationTable.locator('tbody tr');
console.log('Total number of rows:', await rows.count());
```

Step 3: Add Assertions for Rows and Columns

Consider 4 rows and 4 columns for validation:

```
await expect(await columns.count()).toBe(4);
await expect(await rows.count()).toBe(4);
```

Step 4: Select a Specific Checkbox

Filter the rows to select the checkbox for a specific name (e.g., Siva):

```
const matchedRow = rows.filter({
  has: page.locator('td'),
  hasText: 'Siva'
});
await matchedRow.locator('input').check();
```

Step 5: Create a Reusable Function

Make a reusable function to select an employee by name:

```
async function selectEmployee(rows, page, name) {  
    const matchedRow = rows.filter({  
        has: page.locator('td'),  
        hasText: name  
    });  
    await matchedRowlocator('input').check();  
}
```

Step 6: Test Script

```
import { test, expect } from '@playwright/test';  
  
test('pagination', async ({ page }) => {  
    await page.goto('https://..');  
    const paginationTable = await page.locator('#paginationTable');  
    const columns = await paginationTable.locator('thead tr th');  
    const rows = await paginationTable.locator('tbody tr');  
  
    await expect(await columns.count()).toBe(4);  
    await expect(await rows.count()).toBe(4);  
  
    //await selectEmployee(rows, page, 'Yogesh');  
    await selectEmployee(rows, page, 'Siva');  
    //await selectEmployee(rows, page, 'Indu');  
    //await selectEmployee(rows, page, 'Chandru');  
});
```

Result:

sno	name	company	employeeOfTheMonth
1	Yogesh	Comcast	[]
2	Siva	Infosys	[/]
3	Indu	HindiTution	[]
4	Chandru	RenugaMill	[]

Summary of Functions Used to Handle a Single-Page Web Table in Playwright:

Locator Functions:

- `locator()`: Captures the table element and specific rows/columns.
- `filter()`: Finds rows containing specific text.
- `has()`: Filters elements within the specified locator.
- `hasText()`: Finds rows or elements that contain specific text.

Counting Functions:

- `count()`: Gets the number of rows and columns.

Assertion Functions:

- `expect().toBe()`: Validates the number of rows and columns.

Interaction Functions:

- `check()`: Selects checkboxes within the table.