Networking

Jetzt wird's interaktiv

Einführung

Interfaces:

- Response
- Request

Functions:

- status()
- allHeaders()
- text()

Einführung

Hier eine Einführung zu den Netzwerk-Aktivitäten

```
1  // 12_networking/tests/01_intro/intro.test.ts
2  import { test, expect } from '@playwright/test';
3
4  test("Response / Request", async ({ page }) => {
5
6      const response = await page.goto('');
7
8      if (response === null) return;
9
10     console.log(response.url());
11     console.log(response.status());
12     console.log(response.ok());
13
14     expect(response.ok()).toBeTruthy();
15
16     console.log(await response.allHeaders());
17     console.log(await response.headersArray());
18
19     console.log(await response.body());
20
21 });
```

Einführung

Es ist relativ einfach API-Tests durchzuführen oder Stati abzufragen:

```
1  // 12_networking/tests/01_intro/intro.test.ts
2  import { test, expect } from '@playwright/test';
3
4  test("Response / Request", async ({ page }) => {
5
6      const response = await page.goto('');
7
8      if (response === null) return;
9
10     console.log(response.url());
11     console.log(response.status());
12     console.log(response.ok());
13
14     expect(response.ok()).toBeTruthy();
15
16 });
```

Einführung

Natürlich ist das nicht alles, wir können mit jeglichen offenen und autorisierten APIs kommunizieren.

Hier an einem etwas Aufwändigerem Github Beispiel.

API-Tests mit Github

Dieses Beispiel ist etwas länger, daher mehrere Teile.

```
1  // 11_test_runner/tests/02_api/api.test.ts
2  import {expect, test} from '@playwright/test';
3
4  const REPO = 'mein-repo-name';
5  const USER = 'dpinezich';
6
7  test.use({
8      extraHTTPHeaders: {
9          'Accept': 'application/vnd.github.v3+json',
10         'Authorization': `token xxxx`,
11     },
12     baseURL: 'https://api.github.com',
13
14 });
15
16 ....
```

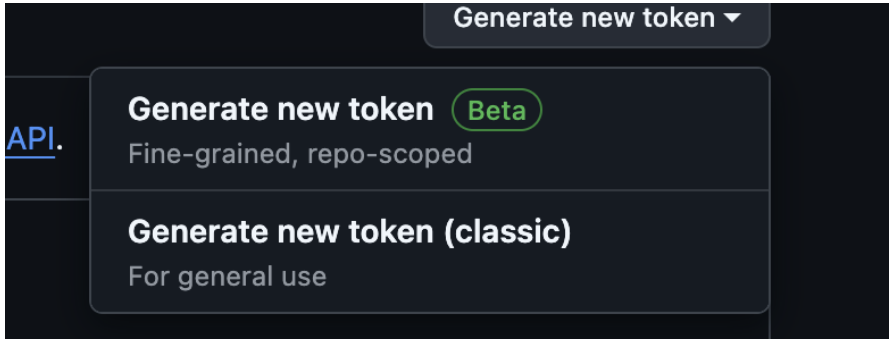
Token im Chat / via David

Hier erstellen wir Konstanten und einen default-use für unsere Tests.

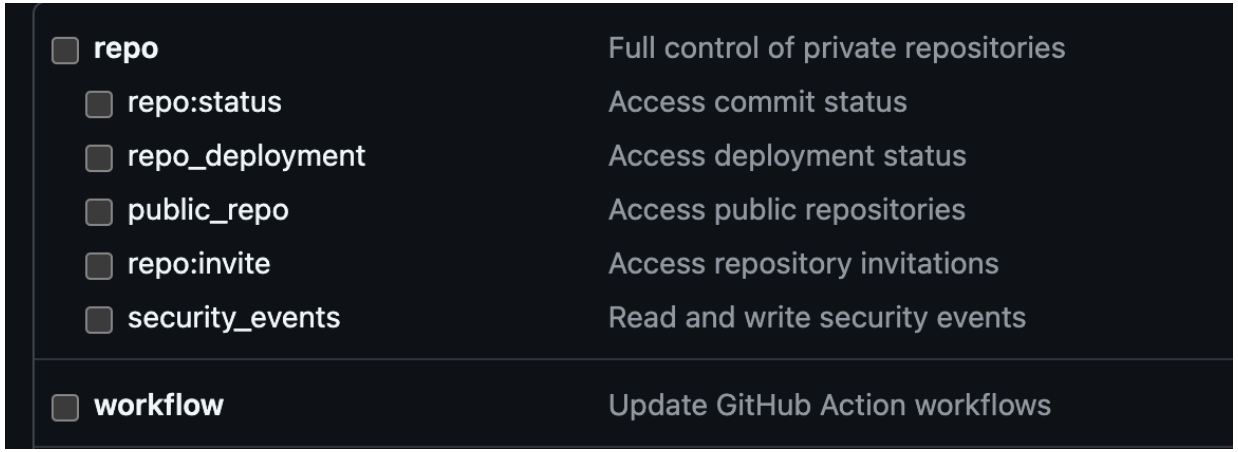
Exkurs: Github Token

https://github.com/settings/tokens

Generate new token (classic)



Wir brauchen repo, workflow und delete repo (insgesamt 3 Kreuze)



API-Tests mit Github

Als Nächstes möchten wir ein Repo erstellen

```
1  // 11_test_runner/tests/02_api/api.test.ts
2  ...
3  test.beforeAll('Create repo', async ({request}) => {
4      const response = await request.post('user/repos', {
5          data: {
6              name: REPO,
7          }
8      });
9      expect(response.ok()).toBeTruthy();
10 });
```

Wir verwenden hierfür den Playwright-hook beforeAll (einmalig, vor allen Tests dieser Datei).

Playwright hooks

Neben beforeAll können wir auch noch folgende hooks verwenden:

- beforeEach (vor jedem Test in dieser Datei)
- afterEach (nach jedem Test in dieser Datei)
- afterAll (am Ende aller Tests in dieser Datei)
- beforeAll (am Anfang aller Tests in dieser Datei)

API-Tests mit Github

Nun wollen wir testen, ob es wirklich funktioniert hat:

```
1 // 11_test_runner/tests/02_api/api.test.ts
2 ...
3 test('Work with newly created repo', async ({page}) => {
4     await page.goto('https://github.com/dpinezich?tab=repositories');
5     await expect(page.getByRole('link', { name: REPO })).toHaveCount(1);
6 })
```

1. auf die Seite navigieren
2. Testen ob exakt 1 Link unserem Namen entspricht

API-Tests mit Github

Nun wollen wir etwas mit der API erstellen:

```
// 11_test_runner/tests/02_api/api.test.ts
...
test('should create a bug report', async ({request}) => {
    const newIssue = await request.post(`/repos/${USER}/${REPO}/issues`, {
        data: {
            title: '[Bug] report 1',
            body: 'Bug description',
        }
    });
    expect(newIssue.ok()).toBeTruthy();

    const issues = await request.get(`/repos/${USER}/${REPO}/issues`);
    expect(issues.ok()).toBeTruthy();
    expect(await issues.json()).toContainEqual(expect.objectContaining({
        title: '[Bug] report 1',
        body: 'Bug description'
    }));
});
```

Issue erstellen

Issue via GET testen

API-Tests mit Github

Am Schluss löschen wir das Repo wieder :-)

```
1 // 11_test_runner/tests/02_api/api.test.ts
2 ...
3 test.afterAll('Delete repo', async ({request}) => {
4     const response = await request.delete(`/repos/${USER}/${REPO}`, {});
5
6     expect(response.ok()).toBeTruthy();
7     expect(response.status() === 204);
8
9 });
```

Standardmässig nicht aktiviert zu Testzwecken.

Ende

Das war alles für dieses Kapitel