Authentication

Introduction

Playwright executes tests in isolated environments called browser contexts. This isolation model improves reproducibility and prevents cascading test failures. Tests can load existing authenticated state. This eliminates the need to authenticate in every test and speeds up test execution.

Core concepts

Regardless of the authentication strategy you choose, you are likely to store authenticated browser state on the file system.

We recommend to create `playwright/.auth` directory and add it to your `.gitignore`. Your authentication routine will produce authenticated browser state and save it to a file in this `playwright/.auth` directory. Later on, tests will reuse this state and start already authenticated.

Bash PowerShell Batch

```
mkdir -p playwright/.auth
echo "\nplaywright/.auth" >> .gitignore
```

Signing in before each test

The Playwright API can automate interaction with a login form.

The following example logs into GitHub. Once these steps are executed, the browser context will be authenticated.

Sync Async

```
page = context.new_page()
page.goto('https://github.com/login')

# Interact with login form
page.get_by_label("Username or email address").fill("username")
page.get_by_label("Password").fill("password")
page.get_by_role("button", name="Sign in").click()
# Continue with the test
```

Redoing login for every test can slow down test execution. To mitigate that, reuse existing authentication state instead.

Reusing signed in state

Playwright provides a way to reuse the signed-in state in the tests. That way you can log in only once and then skip the log in step for all of the tests.

Web apps use cookie-based or token-based authentication, where authenticated state is stored as cookies or in local storage. Playwright provides browserContext.storageState([options]) method that can be used to retrieve storage state from authenticated contexts and then create new contexts with pre-populated state.

Cookies and local storage state can be used across different browsers. They depend on your application's authentication model: some apps might require both cookies and local storage.

The following code snippet retrieves state from an authenticated context and creates a new context with that state.

Sync **Async**

```
# Save storage state into the file.
storage = context.storage_state(path="state.json")

# Create a new context with the saved storage state.
context = browser.new_context(storage_state="state.json")
```

Advanced scenarios

Session storage

Reusing authenticated state covers cookies and local storage based authentication. Rarely, session storage is used for storing information associated with the signed-in state. Session storage is specific to a particular domain and is not persisted across page loads. Playwright does not provide API to persist session storage, but the following snippet can be used to save/load session storage.

Sync **Async**

```
import os
# Get session storage and store as env variable
session_storage = page.evaluate("() => JSON.stringify(sessionStorage)")
os.environ["SESSION_STORAGE"] = session_storage

# Set session storage in a new context
session_storage = os.environ["SESSION_STORAGE"]
context.add_init_script("""(storage => {
  if (window.location.hostname === 'example.com') {
    const entries = JSON.parse(storage)
    for (const [key, value] of Object.entries(entries)) {
      window.sessionStorage.setItem(key, value)
    }
  }
})('""" + session_storage + "')")
```