Running and debugging tests

Introduction

You can run a single test, a set of tests or all tests. Tests can be run on one browser or multiple browsers by using the `--browser` flag. By default tests are run in a headless manner meaning no browser window will be opened while running the tests and results will be seen in the terminal. If you prefer you can run your tests in headed mode by using the `--headed` CLI argument.

You will learn

- How to run tests from the command line
- How to debug tests

Running tests

Command Line

To run your tests use the `pytest` command. This will run your tests on the Chromium browser by default. Tests run in headless mode by default meaning no browser window will be opened while running the tests and results will be seen in the terminal.

```
pytest
```

Run tests in headed mode

To run your tests in headed mode use the `--headed` flag. This will open up a browser window while running your tests and once finished the browser window will close.

```
pytest --headed
```

Run tests on different browsers

To specify which browser you would like to run your tests on use the `--browser` flag followed by the name of the browser.

```
pytest --browser webkit
```

To specify multiple browsers to run your tests on use the `--browser` flag multiple times followed by the name of each browser.

```
pytest --browser webkit --browser firefox
```

Run specific tests

To run a single test file pass in the name of the test file that you want to run.

```
pytest test_login.py
```

To run a set of test files pass in the names of the test files that you want to run.

```
pytest tests/test_todo_page.py tests/test_landing_page.py
```

To run a specific test pass in the function name of the test you want to run.

```
pytest -k test_add_a_todo_item
```

Run tests in parallel

To run your tests in parallel use the `--numprocesses` flag followed by the number of processes you would like to run your tests on. We recommend half of logical CPU cores.

```
pytest --numprocesses 2
```

(This assumes `pytest-xdist` is installed. For more information see here.)

For more information see Playwright Pytest usage or the Pytest documentation for general CLI usage.

Debugging tests

Since Playwright runs in Python, you can debug it with your debugger of choice with e.g. the Python extension in Visual Studio Code. Playwright comes with the Playwright Inspector which allows you to step through Playwright API calls, see their debug logs and explore locators.

To debug all tests run the following command.

Bash **PowerShell** **Batch**

```
PWDEBUG=1 pytest -s
```

To debug one test file run the command followed by the name of the test file that you want to debug.

Bash **PowerShell** **Batch**

```
PWDEBUG=1 pytest -s test_example.py
```

To debug a specific test add `-k` followed by the name of the test that you want to debug.

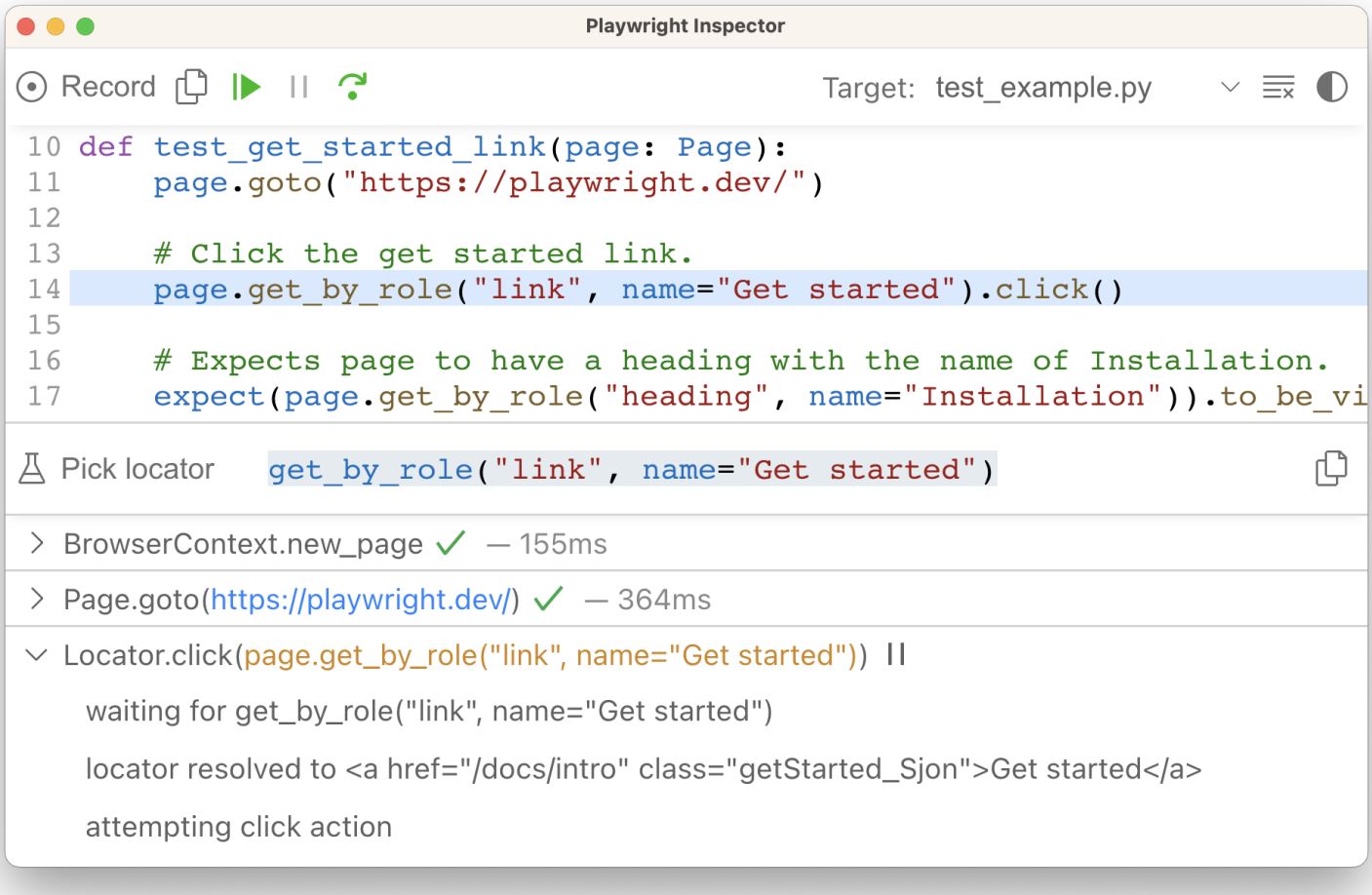
Bash **PowerShell** **Batch**

```
PWDEBUG=1 pytest -s -k test_get_started_link
```

This command will open up a Browser window as well as the Playwright Inspector. You can use the step over button at the top of the inspector to step through your test. Or press the play button to run your test from start to finish. Once the test has finished the browser window will close.

While debugging you can use the Pick Locator button to select an element on the page and see the locator that Playwright would use to find that element. You an also edit the locator and see it

highlighting live on the Browser window. Use the Copy Locator button to copy the locator to your clipboard and then paste it into you test.



Check out our debugging guide to learn more about the Playwright Inspector as well as debugging with Browser Developer tools.

What's next

- Generate tests with Codegen
- See a trace of your tests
- Run your tests on CI with GitHub Actions