Mini-Project: Test Automation

Let’s test the DHL website and APIs!

For this mini-project we will use the following resources:

- The DHL developer support website: https://support-developer.dhl.com/support/home
- The DHL API documentation: https://developer.dhl.com/api-catalog

For the purpose of this project, there is no need to create an account, we will use publicly available resources! Feel free to pick a language you feel comfortable using. I chose python for the code samples I included below.

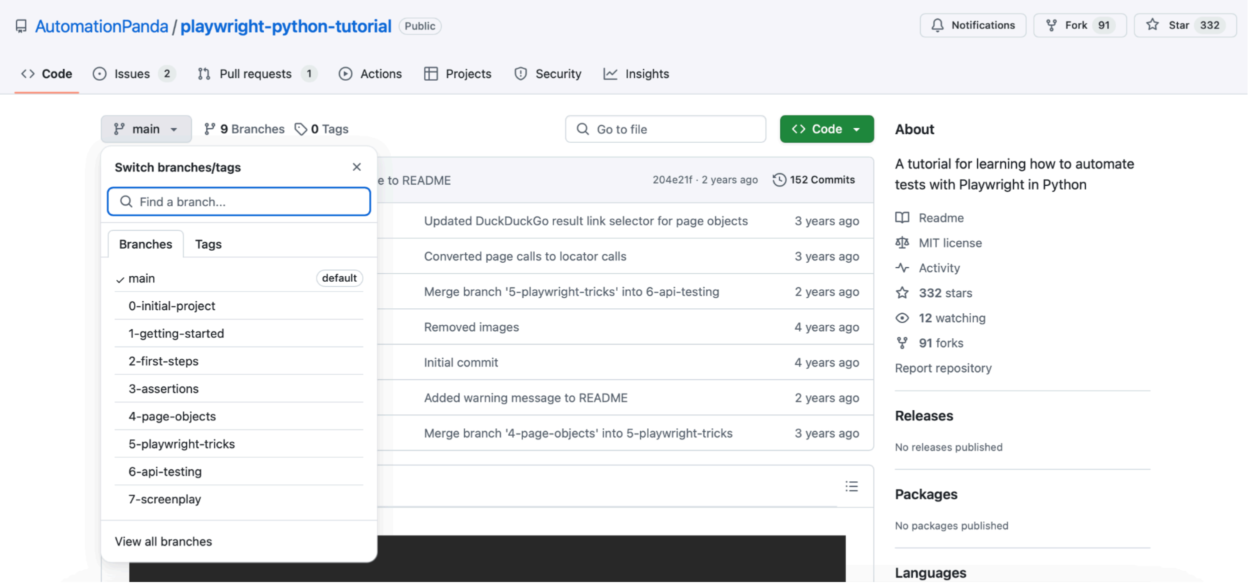
Task 1

Create a function that is able to open the browser, navigate to https://support-developer.dhl.com/support/home, click on “Knowledge base”, then click on “General”. Once the page of General articles opens, the function should take a screenshot of the browser and save it somewhere on disk.

✨ **Bonus points**: after taking the screenshot, scroll down and click on page “2” of the list of articles. Once page two has opened, please click on the first article displayed on the page.

Some tips

- We need to be able to open a browser, which means we should use a framework such as playwright or cypress or selenium… How to get started with playwright in python
- We’ll need to use the concept of “waiting”. (For example, I wait for the page to load, and then I click on an element). You could read about implicit and explicit waits and about auto-waiting https://playwright.dev/docs/actionability.
- Playwright has a really good documentation, and it supports multiple languages. You should check it out: https://playwright.dev/docs/api/class-playwright
- If you would like to dive straight into some code samples, check this repo: https://github.com/AutomationPanda/playwright-python-tutorial. There are different tutorials on different branches, see screenshot below.



- ✓ main
- 0-initial-project
- 1-getting-started
- 2-first-steps
- 3-assertions
- 4-page-objects
- 5-playwright-tricks
- 6-api-testing
- 7-screenplay

e to README — 204e21f · 2 years ago — 152 Commits

Updated DuckDuckGo result link selector for page objects	3 years ago
Converted page calls to locator calls	3 years ago
Merge branch '5-playwright-tricks' into 6-api-testing	2 years ago
Removed images	4 years ago
Initial commit	4 years ago
Added warning message to README	2 years ago
Merge branch '4-page-objects' into 5-playwright-tricks	3 years ago

About

A tutorial for learning how to automate tests with Playwright in Python

- Readme
- MIT license
- Activity
- 332 stars
- 12 watching
- 91 forks

Report repository

Releases

No releases published

Packages

No packages published

Languages

Task 2

Create a function that is verifying the status of a shipment.
For this, use the tracking API and provide the shipment number 123.
https://developer.dhl.com/api-reference/shipment-tracking#reference-docs-section/

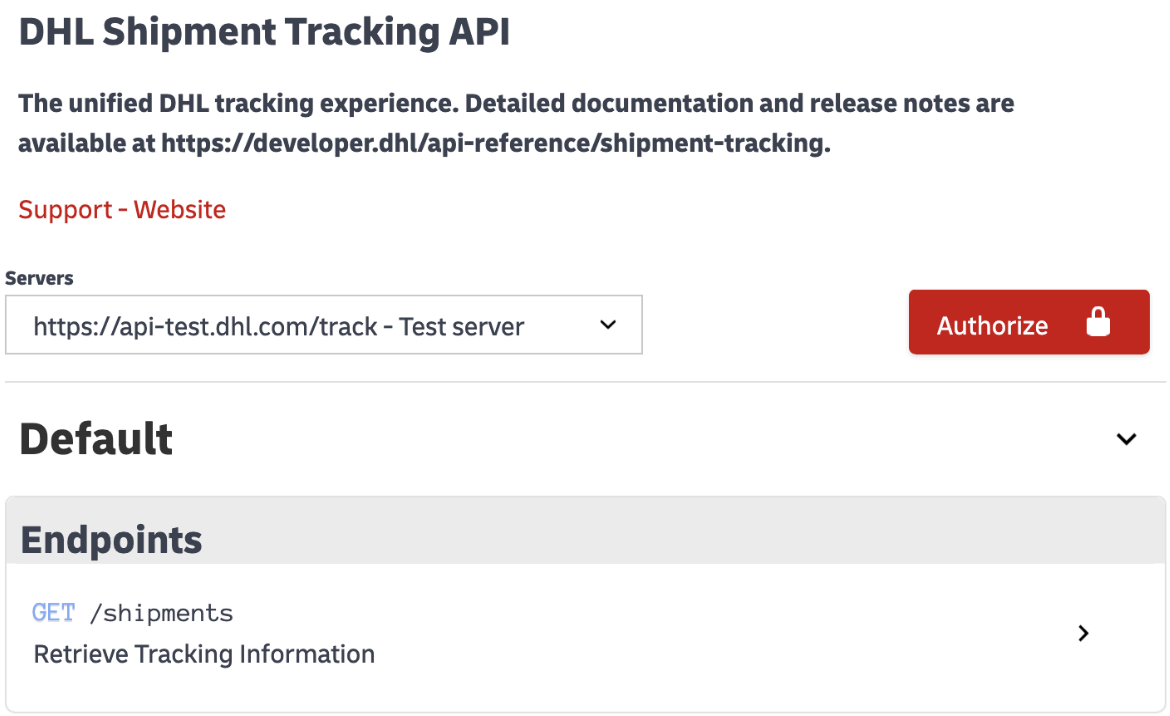
The function should check that the return code of the GET request is 200.

✨ **Bonus points**: after verifying that the status code is 200, verify that the origin address has the country code "US". Here is a sample of how the response would look like:

```
… something
"origin": {
        "address": {
          "countryCode": "US",
          "postalCode": "41048",
          "addressLocality": "HEBRON"
        }
… something else…
```

Some tips

- For this task there is no need to open a browser, we are simply verifying the response to an API call.
- The API from DHL offers a Swagger documentation, which is super useful for us, because we can test some requests directly on their page. (You can read shortly about **Swagger** and why it's so cool to have it in a software project.)



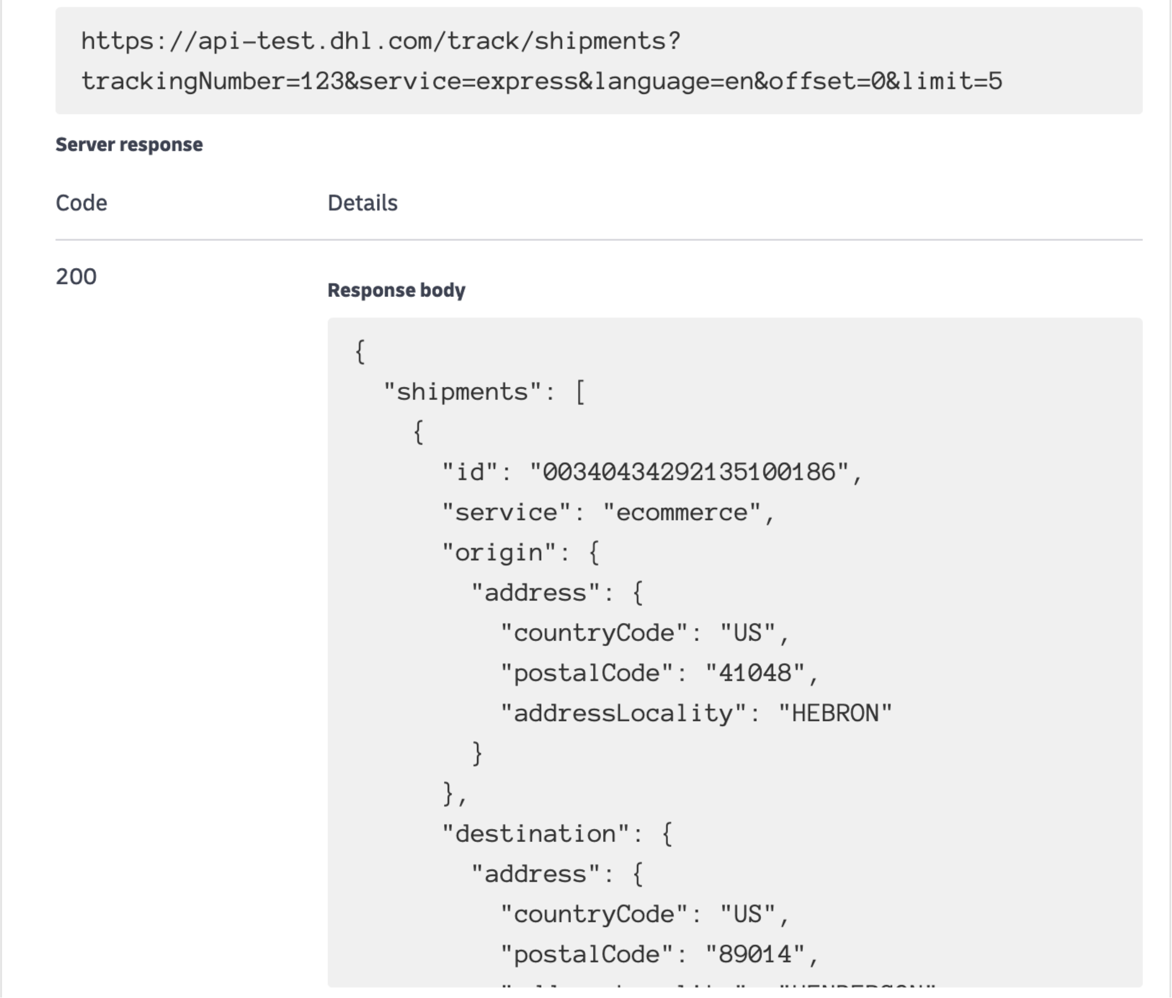
DHL Shipment Tracking API

The unified DHL tracking experience. Detailed documentation and release notes are available at https://developer.dhl/api-reference/shipment-tracking.

Servers: https://api-test.dhl.com/track - Test server

GET /shipments
Retrieve Tracking Information

- If you click on the GET /shipments, a playground will open for you where you can provide parameters to your request. Like this:
  - Click on GET /shipments
  - Scroll down a bit and click on "Try it out"
  - In the field of tracking number, type 123
  - Scroll down until you see the button "Execute"
  - Click on "Execute"
  - You should see something like this:



```
https://api-test.dhl.com/track/shipments?
trackingNumber=123&service=express&language=en&offset=0&limit=5
```

Server response

Code	Details
200	**Response body**

```
{
  "shipments": [
    {
      "id": "00340434292135100186",
      "service": "ecommerce",
      "origin": {
        "address": {
          "countryCode": "US",
          "postalCode": "41048",
          "addressLocality": "HEBRON"
        }
      },
      "destination": {
        "address": {
          "countryCode": "US",
          "postalCode": "89014",
          "addressLocality": "HENDERSON"
```

- GET requests can also be triggered by simply pasting the request link directly into the browser. So let’s try:
- https://api-test.dhl.com/track/shipments?trackingNumber=123&service=express&language=en&offset=0&limit=5

What do you see?
And why?

….

Task 3

Please combine Task 1 and Task 2 into a cucumber scenario.

1. Define what you want to test as a cucumber scenario (e.g. Given… When…. Then….)
2. Change the functions you have already created so that they can be triggered by the Given… When … Then …. logic
3. Run the changed tests and see if you obtain the same results like you did before

Some tips

- For this task you will need to figure out how to refactor your own code. The functionality should stay exactly the same, but the tests should follow the Given… When… Then… structure