# **Auto-generation of API Test Cases using Playwright**

## **Table of Contents**

1. Introduction
2. Project Overview
3. Detailed Explanation
   * convertCollections.js
   * collections Directory
   * tests Directory
   * package.json
   * playwright.config.js
4. Workflow
5. Learning and Benefits
6. Conclusion

## **1. Introduction**

The "Auto-generation of API Test Cases using Playwright" project aims to streamline the API testing process by automating the generation of test scripts from existing Postman collections. This approach leverages Playwright, an advanced end-to-end testing framework, to execute the tests, ensuring comprehensive and efficient validation of APIs.

## **2. Project Overview**

This project provides a solution for automatically converting Postman collections into executable Playwright test scripts. By doing so, it significantly reduces the manual effort involved in writing test cases and ensures consistency in API testing. The generated test scripts can be executed using Playwright, taking advantage of its robust testing capabilities to perform detailed validations.

## **3. Detailed Explanation**

### **convertCollections.mjs**

This script is the core of the project, responsible for converting Postman collections into Playwright test scripts.

**Key Functions:**

* **Read Postman Collections:** Reads JSON files from the collections directory.
* **Parse Collection Files:** Extracts API request details, including name, method, URL, headers, and body.
* **Generate Test Scripts:** Creates Playwright test scripts based on the extracted details.
* **Save Test Scripts:** Writes the generated scripts to the tests directory with a “.spec.js” extension.

**Code Breakdown:**

import fs from 'fs/promises';

import path from 'path';

import { fileURLToPath } from 'url';

const \_\_filename = fileURLToPath(import.meta.url);

const \_\_dirname = path.dirname(\_\_filename);

const collectionsDir = path.join(\_\_dirname, 'collections');

async function convertCollections() {

try {

const files = await fs.readdir(collectionsDir);

for (const file of files) {

const collectionFile = path.join(collectionsDir, file);

const collectionData = await fs.readFile(collectionFile, 'utf8');

const collection = JSON.parse(collectionData);

const requests = collection.item.map(item => {

return {

name: item.name,

method: item.request.method,

url: item.request.url.raw,

headers: item.request.header ? item.request.header.reduce((acc, header) => {

acc[header.key] = header.value;

return acc;

}, {}) : {},

body: item.request.body ? JSON.stringify(item.request.body[Object.keys(item.request.body)[0]]) : null

};

});

const testScript = `

import { test, expect } from '@playwright/test';

test.describe('API Tests for ${path.basename(collectionFile)}', () => {

${requests.map(request => `

test('${request.name}', async ({ request }) => {

const response = await request.${request.method.toLowerCase()}('${request.url}', {

headers: ${JSON.stringify(request.headers)},

data: ${request.body || 'null'}

});

expect(response.status()).toBe(${request.method === 'POST' ? 201 : 200});

const responseBody = await response.json();

});

`).join('')}

});

`;

const testFileName = `test\_${path.basename(collectionFile, '.json')}.spec.js`;

const testFilePath = path.join(\_\_dirname, 'tests', testFileName);

await fs.writeFile(testFilePath, testScript);

}

} catch (err) {

console.error('Error converting collections:', err);

}

}

convertCollections();

### **collections Directory**

This directory stores Postman collection files in JSON format. Each file represents a collection of API requests that need to be tested. The convertCollections.js script reads these files, extracts the necessary details, and generates corresponding Playwright test scripts.

### **tests Directory**

The tests directory is where the generated Playwright test scripts are saved. Each test script corresponds to a Postman collection file and includes test cases for the API requests defined in the collection.

### **package.json**

This file manages the project's dependencies and scripts. It includes Playwright as a dependency and defines a test script to run the generated test cases.

**Content:**

{

"name": "api-test-framework",

"version": "1.0.0",

"description": "",

"main": "index.js",

"type": "module",

"scripts": {

"convert": "node convertCollections.js",

"test": "npx playwright test"

},

"dependencies": {

"@playwright/test": "^1.20.0"

},

"devDependencies": {},

"author": "",

"license": "ISC"

}

## **4. Workflow**

* **Setup:**
  + Install the required dependencies (@playwright/test).
  + Ensure the project is configured to use ES modules by setting "type": "module" in package.json.
* **Convert Collections:**
  + Execute convertCollections.js to read and parse Postman collections.
  + Generate Playwright test scripts and save them in the tests directory.
* **Run Tests:**
  + Use the command npx playwright test to run the generated test scripts.
  + Playwright executes the test cases, validating the APIs based on the Postman collections.

## **5. Learning and Benefits**

* **Automation Benefits:** Automating the conversion of Postman collections into test scripts saves time and reduces manual effort.
* **Playwright Proficiency:** Understanding how to leverage Playwright for API testing, including making HTTP requests and performing assertions.
* **Project Structuring:** Learning to structure a project to support automated testing and ES module syntax.
* **Error Handling:** Handling errors effectively when reading files and parsing JSON data.

## **6. Conclusion**

This project demonstrates the power of automation in API testing by converting Postman collections into Playwright test scripts. It highlights the efficiency gains and error reduction that can be achieved through automation. By integrating Playwright, this project provides a robust framework for validating APIs, making it a valuable tool for developers and testers.