Welcome to E Commerce Website Testing

Table of Contents

1.Introduction

The purpose of this framework is to provide a testing environment for WordPress developers to test website functionality, Unit testing, Visual testing ,end to end testing security testing etc .. , the framework uses pytest, SQL, Docker , and CI/CD pipeline to facilitate testing.

2. Set Up The Framework:

2.1 Important things:

- The project supports only chrome.
- The project supports only WordPress sites.
- You must install WooCommerce in your WordPress through the plugins.
- Check your browser version for the web driver. (Version 111 included in the project)
- If your using Local to run WordPress locally you will need to fill the port option.

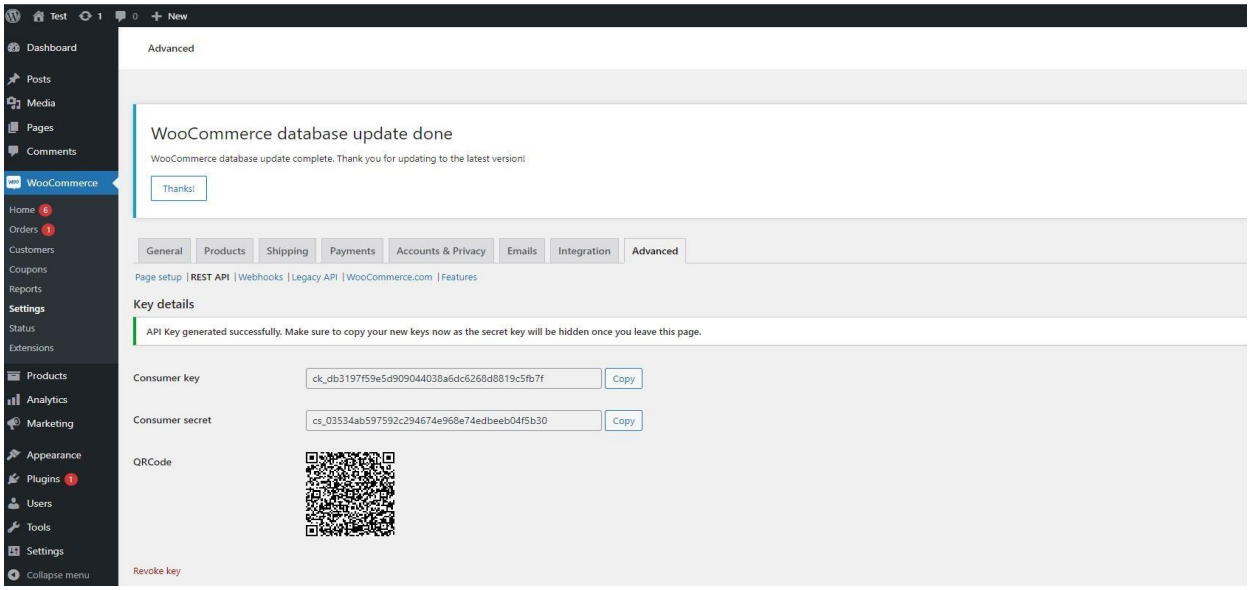
2.2 Must do in WordPress:

1) Download the zip file from here if u run WordPress locally.
2) Pull the sample_products.xml from the zip then Go back to WordPress: Tools -> WordPress -> install -> run Importer -> choose file -> provide the location of the XML file.

3) Settings -> reading -> click on A static page ->change homepage to shop
4) Settings -> general -> check Anyone can register.
5) WooCommerce -> settings -> Accounts & Privacy -> “Guest checkout” check all, “Account Creation” check all but the last one.
6) Must create coupon code: marketing -> coupon (coupon type percentage discount)

2.3 set up the Credentials:

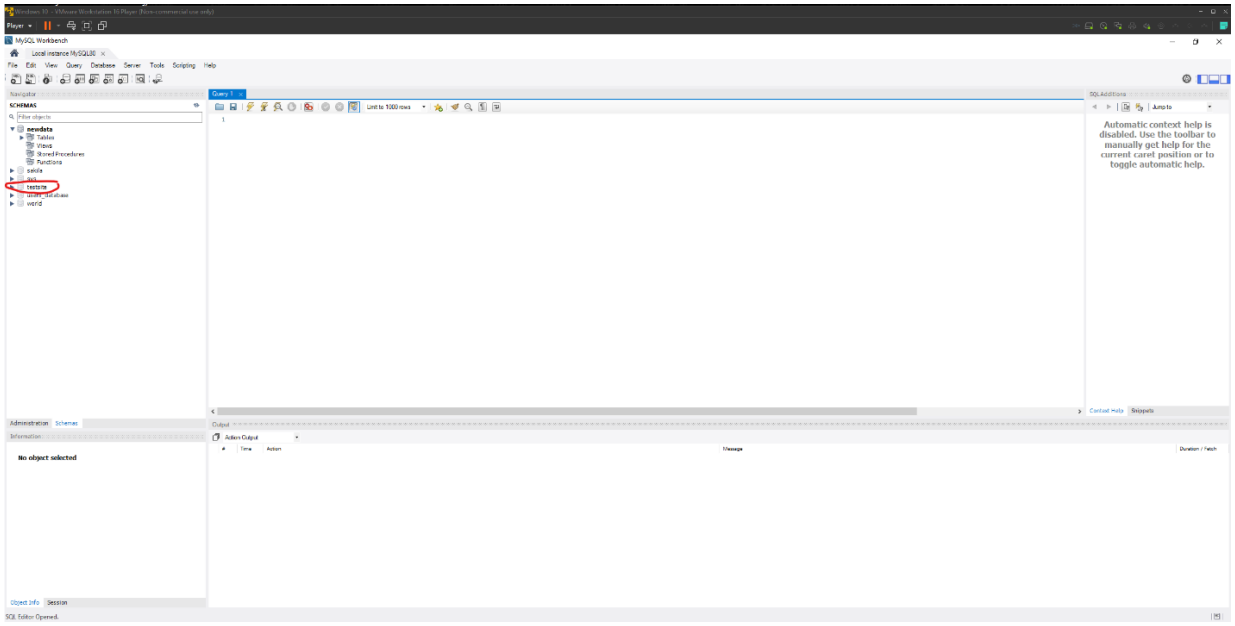
- WC_KEY – WordPress public key
- WC_SECRET – WordPress secret key



*NOTE: im using MYSQL.

- DB_USER – Database Login Name (default root)
- DB_PASSWORD - Database Password (default root)
- DB_SERVER – Database Host(default localhost, if you’re using VMware for DB you will need to provide the IP of the iso.)

- DB_NAME – Database Table Name.(Schema)



- SELPATH – Selenium Driver Path. (click here or you can set up manually)
- ENVSEL – The URL of the site.
- COUPON – the name of your coupon.
- PORT = the port you're using to run WordPress Database. (optional)

2.4 Set up The Testing Environment:

1) Create python virtual environment:

1.1) pip install virtualenv

1.2) py -m venv {Name for your virtual environment.}

1.3) open IDE make the virtual environment as main virtual environment.

*)PyCharm – file -> settings -> Project -> python interpreter ->  -> add -> new environment -> click on Location -> paste you're python.exe path

1.4) activate your virtual environment.

1.5) install/develop setup.py, command: python setup.py install.

2.5 Set Up The Selenium Path:

1) copy the chrome driver.exe to the “front” folder.

1.1) execute the getpath.py.

1.2) copy the path you got in the terminal to **Credentials** (SELPATH=” whatever path you have copied”). (sh,ps1,bat)

3. Testing Section:

3.1 BackEnd Section:

Commands:

- tcid stands for – test case id.
- tcidc stands for – test case id for customers.
- tcidp stands for – test case id for products.
- tcido stands for – test case id for orders.
- customers – pytest -m customers. (Run all the costumer tests)
- products – pytest -m products. (Run all the product tests.)
- orders – pytest -m orders. (Run all the orders tests)

Section	ID	Type	COMMAND	FunctionName
BACKEND	tcidc1	customers	Pytest -m tcidc1	`test_create_customer_only_email_password():`
BACKEND	tcidc2	customers	Pytest -m tcidc2	`test_create_customer_fail_for_existing_email():`
BACKEND	tcidc3	customers	Pytest -m tcidc3	`test_get_all_customers():`
BACKEND	tcidco1	coupons	Pytest -m tcidco1	`test_create_coupon()`
BACKEND	tcidco2	coupons	Pytest -m tcidco2	`test_delete_single_coupon_with_id()`
BACKEND	tcidco3	coupons	Pytest -m tcidco3	`test_update_coupon_percentage()`
BACKEND	tcidp1	products	Pytest -m tcidp1	`test_list_all_products():`
BACKEND	tcidp2	products	Pytest -m tcidp2	`test_list_product_with_specific_id():`
BACKEND	tcidp3	products	Pytest -m tcidp3	`test_create_single_product():`
BACKEND	tcido1	Orders	Pytest -m tcido1	`test_create_paid_order_guest_user(random_product_setup):`

BACKEND	**tcido2**	**Orders**	**Pytest -m tcido2**	`test_create_paid_order_new_customer(random_product_setup)::`

3.2 Frontend Section:

- tcids stands for – test case id for selenium.

FRONTEND	**tcids1**	**Smoke**	**Pytest -m tcids1**	`test_login_none_existin_user(self):`
FRONTEND	**tcids2**	**Smoke**	**Pytest -m tcids2**	`test_register_new_user(self):`
FRONTEND	**tcids3**	**Smoke**	**Pytest -m tcids3**	`test_register_new_user_failed(self):`
FRONTEND	**tcids4**	**End to end**	**Pytest -m tcids4**	`test_end_to_end_checkout_guest_user(self)`

*) if you would like to add more tests here the docs.
*) https://woocommerce.github.io/woocommerce-rest-api-docs/#introduction

4. Docker Section:

Must update the Credentials:

1. DB_SERVER = “host.docker.internal”
2. ENVSEL = internal IP Address/portnumber(which port your'e using to run WordPress locally , mamp default =8888, local default=10005)
3. Go to WordPress -> settings -> general -> replace the URL with ipv4 instead of localhost so docker can interact with it.

- Make sure you’re on the root folder (the folder where the docker file exists)
  or provide the path of your Docker file instead of the dot in the first step.

- I'm copying the Credentials manually you can skip step 3 and 4 just by adding to your Docker file "COPY ./ Credentials.sh /automation" **without the quotes**.
- You can do a lot of more things with docker like mount, run test outside etc ..
  navigate to the URL below to learn about docker.
- [Docker Official Site.]

4.1 Linux Section (Container) - inside the container:

1. docker build -t automation **.** (pay attention to the dot)
2. docker run -it automation /bin/bash
3. open new terminal (don't close the first one)-> docker ps -> copy the name of the container
4. run this command:
   docker cp Credentials.sh {the name you've copied}:/automation
5. go back to the first terminal.
6. Source Credentials.sh
7. Pytest (without report)
8. Pytest –html=report.html --self-contained-html (with report inside the container if you need to report outside the container you will need to mount the folder).

Now you can run the tests the way you want.

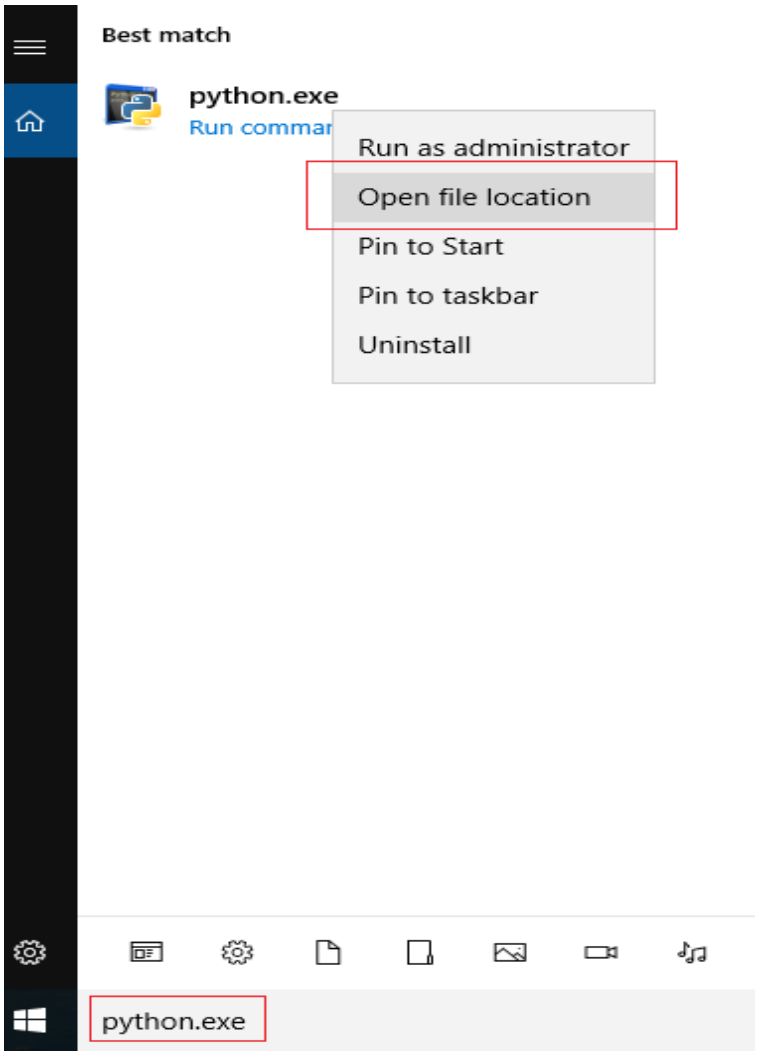
4.2 Windows Section (Container) - inside the container:

- All the steps are the same but the second step.(without the add command if you're using windows) **/bin/bash** replace it with /cmd.exe or /powershell.exe .
- Step5: execute the ps1 or bat file.(powershell = . ./Credantials.ps1, bat= type the name of the file into the command line, or use "call Credantials.bat")
- Step 7 & 8 are the same

5. Jenkins Section:

1. open the Jenkins.txt.
2. got to the second stage “create virtual environment” replace the {path to your python.exe} with your actual path.

2.1 in the windows search bar, type python.exe but don’t click on it in the menu. Instead, right-click on it, and select Open file location.



3. right click on the file “python.exe” copy as path.
4. paste it on notepad, txt file doesn’t matter then delete the quotes replace the backslash with double backslash (\\) then copy it to the Jenkins.txt.
5. go to the third stage “Test” replace the {path to your credentials file} with your actual path.
6. make sure you update the path to your chrome driver in the credentials file.

Run in Jenkins:

1. Go to your Jenkins URL.
2. Click on new item, name your project then select pipeline and click ok.
3. Open Jenkins.txt copy the code then go to the pipeline section and paste the code click save.
4. Click on build.
5. Open the folder where you ran the Jenkins file there will be results folder you will have test reports there.

6. Conclusion:

In summary this framework provides a complete testing solution for WordPress developers it includes testing for both front-end and back-end , as well api testing.

This framework can set up quickly and easily it includes support for Docker and CI/CD pipelines.

By following the provided instructions, developers can ensure their WordPress site is thoroughly tested before deployment.