BDD Automation Framework

The Behavioral Driven Development Automation Framework (BDD) allows automating testing on frontend E-Commerce Website and backend test of pet store using RestAssured.

Table of Contents

Description

BDD FW combines allows users to automate tests on ECommerce website and API calls on pet store with various scenarios mentioned in the feature file.

Installation

The project folders and dependencies for BDD FW have been managed using maven pom and uploaded on Github. Use the latest version by using the git pull command below.

```
docker pull delphix/automation-framework
```

The delphix.yaml file

The `delphix.yaml` file is the configuration file that defines the data management as code strategy for the project. Create a `delphix.yaml` file based on this guide: Configure Delphix YAML

Usage

automation.properties file has already been created and pre-filled with below values:Press Ctrl+Click to View * WebURL=http://automationpractice.com/index.php? * browser=chrome * userName=nikagrawal774@gmail.com * password=feX.Nin3YtG2ehE * implicitTimeout=10 * baseUrl=https://petstore.swagger.io/v2 * appId=123456

Optionally, the `userName` and 'password' can be set by creating an account on the Ecommerce website: `http://automationpractice.com/index.php?`. Run the docker container with your project mounted as a volume and environment file instantiated.

```
docker run -v ${PWD}:/daf/app --env-file ${PWD}/.env delphix/automation-framework
```

Project Structure

Press Ctrl+Click to View

Cucumber.options package has TestRunner features package has all feature files pages package has all classes related to ecommerce UI test commonActions has one class with the same name that contains all common actions of UI setpDefinations has all step definitions and Hooks utils have all helper classes automation.properties have all configurable items.

There are two Feature Files created, PetStore.feature PlaceOrder.feature

PetStore.feature contains REST API testing feature file for PETSTORE API

PlaceOrder.feature contains UI automation testing feature file for E-Commerce website

Test Scenarios Covered

For E-Commerce Testing, a total of three test scenarios are covered namely: newUserPlaceOrderBySearchingCategoryProduct

This includes the creation of a new account on the UI, and then searches for products and adds to cart, and then checkout testing giving address and payment

information

existingUserPlaceOrderBySearchingPopularProducts

Similar to the first test scenario except an existing credential is used, details of which are fetched from the properties file.

existingUserChecksTotalOfProducts

The existing user adds a total of three products, during checkout, checks and verifies the total of the products matches with the total value shown in checkout, then deletes one product and checks that the total should reflect the change.

For PetStore RestAssured is used where adding of a new pet, fetching pet information, updating and then deleting the pet testing is done. On all mentioned steps, verification is done on status codes, pet name and id.

Usage

Running E-Commerce Web Test

To run only UI automation, in the directory of project location, run using the below command

mvn test verify -Dcucumber.filter.tags="@UI"

Alternatively, you can also run using the runUI.bat file found in the project directory

Running PetStore API Test

To run only PetStore API automation testing, in the directory of project location, run using the below command

mvn test verify -Dcucumber.filter.tags="@Backend"

Alternatively, you can also run using the runBackend.bat file found in the project directory

Running Regression Test

To run both UI and API automation testing, in the directory of project location, run using the below command

mvn test verify -Dcucumber.filter.tags="@Backend or @UI"

Alternatively, you can also run using the runAll.bat file found in the project directory

Contribute

1. Fork the project.
2. Make your bug fix or new feature.
3. Add tests for your code.
4. Send a pull request.

Contributions must be signed as `User Name <user@email.com>`. Make sure to set up Git with user name and email address. Bug fixes should branch from the current stable branch. New features should be based on the `master` branch.

Reports, Logs and Screenshots

Reports are generated in the following folders in the project directory in following paths

/target/cucumber-html-reports/overview-features.html

/target/cucumber/report.html

/test output/PdfReport/ExtentPdf.pdfCtrl+Click to View

overview-features.html contains detailed reports showing different tabs wise reports for features, tags and tests

report.html contains reports with screenshot attached in it

ExtentPdf.pdf contains a detailed pdf report is presented with total execution time along with speedometer, pie and bar chart representation of Features, Scenarios and steps representation. A detailed chapter is presented showing tags wise summary and for each test step details: hooks associated, total time taken, and error logs if any.

Logs are generated in the below directory Ctrl+Click to View /log/application.html /log/application.log

Screenshots are attached in the report.htmlCtrl+Click to View

Known Issues on Ecommerce Site

While running UI automation test execution, the E-Commerce website during random times throws a "Resource Limit Reached" error due to which test execution fails. On random testing, it is found the error is more likely to be encountered during 11 pm to 9 am IST time. It is advisable to run before or after the mentioned time

Code of Conduct

This project owner is Nikhil Agrawal. By participating in this project you agree to abide by its terms.

Contributor Agreement

All contributors are required to get pre-authorization in writing by sending an email to `nik_agrawal2000@yahoo.com`. Procedure for the same will be send as an email response thereafter.

Reporting Issues

Issues should be reported in the GitHub repo's issue tab. Include a link to it.

Statement of Support

Thank you for taking the time in reading this document. I appreciate it

License