ATG-Automation

Automation Solution Document

Charu Sachdeva

Purpose

This document provides the high-level design overview of automation testing assignment for V4 Horse Race and Pet Store APIs.

This document is also intended to help reviewer to run the testing framework for selected flows.

Design Considerations

The architecture and design of the solution is catering following key considerations:

- I have implemented the assignment with the structured framework
- I have structured the code from a maintainability and reusability perspective

Technology Stack

All the frameworks and technologies used are well established and have extensive industry wide acceptance. Automation suite for web applications is built upon below technology stack.

Area	**Technology Used**
IDE	Intellij Idea
Build Tool	Maven
Language	JAVA 17
Automation Tool	Selenium (WebDriver 4)
API testing	Rest Assured
Framework for Tests	Junit 5
Logger	Log4j2

Advantages of using Selenium

Selenium is an open-source automation testing tool which is used for automating tests carried out on different web-browsers. It has lot of features, listing few below here:

- Language and Framework Support
- Open-Source Availability
- Multi-Browser Support
- Support Across Various Operating Systems
- Reusability and Integrations

Advantages of using Rest Assured

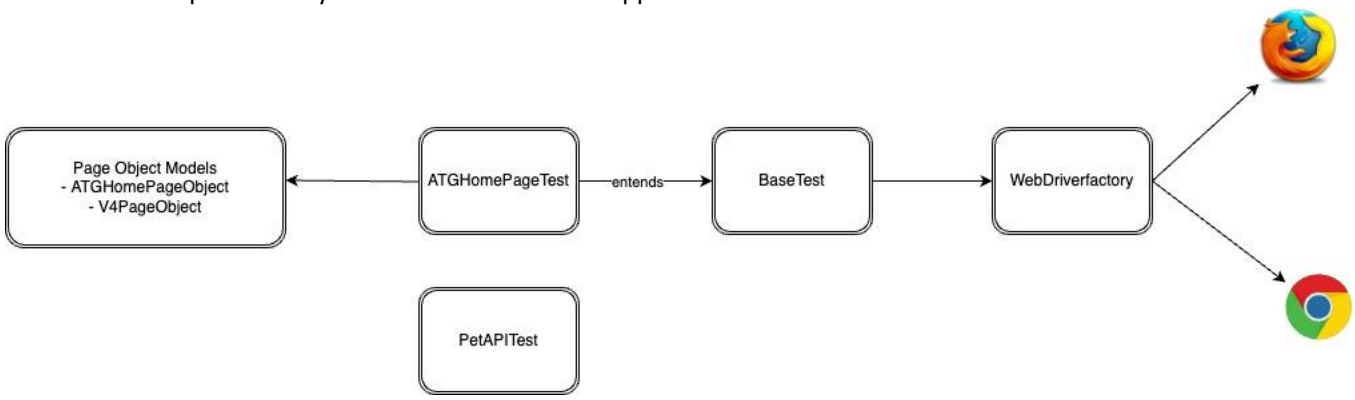
Rest Assured is an open-source Java-based Domain-Specific Language (DSL) that allows you to write powerful, readable, and maintainable automated tests for your RESTful APIs.

With the help of this library:

- It removes the need for writing a lot of boilerplate code required to set up an HTTP connection, send a request and receive and parse a response
- It supports a Given/When/Then test notation, which instantly makes your tests human readable
- It helps in writing clean code and maintaining a clear separation of concerns within a test, thus leading to very readable test.

Solution Architecture

This section explains the system architecture of the application.



Page Object Models

Page Object model is an object design pattern in Selenium, where web pages are represented as classes, and the various elements on the page are defined as variables on the class. All possible user interactions can then be implemented as methods on the class.

WebDriverFactory

WebDriverFactory is used to create WebDriver for various browsers.

Driver Managers

I have used two different driver managers, FirefoxDriverManager and ChromeDriverManager, which return instance of WebDriver for respective browser type.

Code Description

Automation testing application has following packages.

Component	Description
logs	This folder contains generated log files
drivermanager	This folder will contain classes of WebDriverManager for various browsers
dto	This folder will contain data transfer object used for the PetStore APIs automation testing
enums	This folder contains enum for various browsers (used while creating the WebDriver)

factory	This folder contains factory for creating WebDriver
pageobjects	This folder contains page object models having all web element locators
util	This folder contains utility classes for giving waits and loading properties from property file
job	This folder contains MainApplication class and CronJob for scheduling the test suit
tests	This folder contains all tests written in jUnit5 for Horse racing and PetStore APIs
test.resources	This folder contains properties files for application and log4j2

Challenges faced

While I enjoyed working on the assignment but below is a small list of few challenges that I faced

- Asynchronous behavior of few elements
    - Solution: Used WedDriverWait having condition for visibility with timeout
- Understanding the V4 format of horse racing
- On translation the html elements in the browser developer tool were getting changed sometimes
- How to run the tests from java
- Ordering of the tests was required in API testing
- PetStore APIs are not consistent sometimes
    - In update API call, sometimes petId is not right in the response
    - In findById API, sometimes name is not correct in the response

Testing the application

Prerequisite

You should have jdk17 and maven installed for running this automation application

You should have any of following browsers (support for other browsers can be added in future):

- Chrome
- Mozilla

All configurable properties have been defined in the application.properties file for ease of updating.

Running tests

The solution can be tested by running:

1. “mvn test” command in the project main folder where pom.xml is present
2. “MainApplication” class which will schedule the cron job for triggering the tests
3. “mvn surefire-report:report” command in the project main folder where pom.xml is present
    a. Path to the unit test report is “/target/site/surefire-report.html”

Possible Improvements for future

There is scope of many improvements in this assignment, few things which could be improved:

- Cover all testing scenarios
- Add support for other browsers
- Cover all API’s and all flows in API testing
- Better reporting using allure reports
- Continuous Integration and Continuous Deployment for the application