Design Basics

This lesson provides a brief introduction to the test automation framework and lists some primary considerations while designing one.

WE'LL COVER THE FOLLOWING

What is a test automation framework?

A test automation framework is a set of libraries/files that helps you to write automated tests in a particular structure/format without worrying about handling reports, logs, failure analysis screenshots/videos, integration to third parties, etc. which will be taken care of internally by the framework.

Design considerations for a test automation framework

We should consider the below points while designing a test automation framework.

Re-usability

The framework should be designed keeping in mind the re-use of the utilities

The framework should be designed keeping in mind the re-use of the utilities and common functionalities. It can be reused by multiple teams for solving

the same purpose (UI/API/Mobile automation, DB connectivity, reading from/writing to files, etc.). The core functionality and capabilities should be decoupled from any specific projects.

Extensibility

It should be easy to integrate UI with any third-party libraries. It should be flexible enough to add new use cases in the future (e.g., UI supporting grid as per need, supporting API or WebSocket automation, etc.). The framework should be designed in a way that, in the future, if needed, the underlying core library (say WebDriver for UI) could be changed to something better without impacting any existing tests. It should be easy to integrate it with any CI (Continuous Integration) tools like Jenkins.

Maintainability

The framework should be easy to understand, debug, and upgrade. Each method should have method comments, and any complex logic should be properly commented with the explanation. Adding more tests in the near future can result in an unmaintainable project. So, it is very important to use the right design pattern.

Reporting

The report should be detailed and granular. Screenshots and failure stack trace should be attached as part of the report. The reports should be available in multiple formats and have enough information for the engineer to analyze the failure and help in debugging. We should be able to integrate with third-party reporting libraries like **Allure** and **Extent**.

Adaptability

The framework should be easy to adapt with minimum onboarding and ramp-up time for a new developer. The complex and repeated code should be encapsulated within the framework which will make life easy for an automation developer.

Supported platforms

The framework should be platform agnostics and work seamlessly for Mac

The framework should be platform agnostics and work seamlessly for Mac, Linux and Windows. The code should work in different IDE's like Eclipse,

Netbeans, etc.

Build management

The framework should be designed to support multiple build managements by test projects, such as Maven and Gradle.

The build management will help to compile, build, run the framework code and tests from the command line. It will also help in creating distributed jars.

Logging

The framework should provide the capability to add multiple levels of logs (INFO, DEBUG, ERROR, etc.) in the test code.

Documentations

Proper documentation will be of great help. For example:

- The java docs which can be generated, hosted and shared with the team.
- Documents for writing tests - will help onboarding new team members easily.

The next chapter will bring detailed information on WebDriver.