DevOps Certification Training

Study Material – Selenium

What is Continuous Testing :

Continuous testing (CT) is a software development process in which applications are tested continuously throughout the entire software development life cycle (SDLC). The goal of CT is to evaluate software quality across the SDLC, providing critical feedback earlier and enabling higher-quality and faster deliveries.

Why is Continuous Testing needed ?

Continuous testing helps track testing for application, microservice, and API security vulnerabilities or logic flaws by working with existing CI tools to detect issues early, mitigating costly time and effort downstream.

With many organizations adopting **DevOps** and **DevSecOps**, embracing automation is a large part of enabling efficiency and speed. In modern AppSec, continuous testing is one of these key practices.

How Does Continuous Testing works with DevOps ?

In the increasingly fast development environment, software release cycles are shortening, pushing organizations to adjust their practices in order to keep up. DevOps practices and tools are essential to this success, and continuous testing plays an important role.

CT helps boost the DevOps pipeline because it fosters testing at all stages of the SDLC, from development to deployment. At the center of DevOps and DevSecOps is the idea of performing activities (like security testing) as soon as possible, speeding up all development activities. Incorporating continuous testing into this framework helps guarantee that development moves forward unhindered, and software of the highest quality is released.

Types of Testing:-

1. Unit Testing :
2. Integration Testing
3. Regression Testing
4. Performance Testing
5. Acceptance Testing

1. Unit Testing :

It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs. We use frameworks like Junit to perform unit testing.

2. **Integration Testing :**

The objective is to take unit-tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.

3. **Regression Testing :**

Every time a new module is added leads to changes in the program. This type of testing makes sure that the whole component works properly even after adding components to the complete program. We use frameworks like Selenium to test the web application end to end.

4. **Performance Testing :**

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test the speed and effectiveness of the program. It is also called load testing. In it we check, what is the performance of the system in the given load.

5. **Acceptance Testing :**

Acceptance testing is done by the customers to check whether the delivered products perform the desired tasks or not, as stated in requirements. We use this OOT, for discussing test plans and for executing the projects.

What is Selenium ?

Selenium is not just one tool or API but it composes many tools.

WebDriver
If you are beginning with desktop website or mobile website test automation, then you are going to be using WebDriver APIs. WebDriver uses browser automation APIs provided by browser vendors to control the browser and run tests. This is as if a real user is operating the browser. Since WebDriver does not require its API to be compiled

with application code; It is not intrusive. Hence, you are testing the same application which you push live.

IDE

IDE (Integrated Development Environment) is the tool you use to develop your Selenium test cases. It's an easy-to-use Chrome and Firefox extension and is generally the most efficient way to develop test cases. It records the users' actions in the browser for you, using existing Selenium commands, with parameters defined by the context of that element. This is not only a time-saver but also an excellent way of learning Selenium script syntax.

Grid

Selenium Grid allows you to run test cases in different machines across different platforms. The control of triggering the test cases is on the local end, and when the test cases are triggered, they are automatically executed by the remote end.

After the development of the WebDriver tests, you may face the need to run your tests on multiple browsers and operating system combinations. This is where Grid comes into the picture.

Selenium is many things but at its core, it is a toolset for web browser automation that uses the best techniques available to remotely control browser instances and emulate a user's interaction with the browser.

Selenium allows users to simulate common activities performed by end-users; entering text into fields, selecting drop-down values and checking boxes, and clicking links in documents. It also provides many other controls such as mouse movement, arbitrary JavaScript execution, and much more.

Although used primarily for front-end testing of websites, Selenium is, at its core, a browser user agent *library*. The interfaces are ubiquitous to their application, encouraging composition with other libraries to suit your purpose.

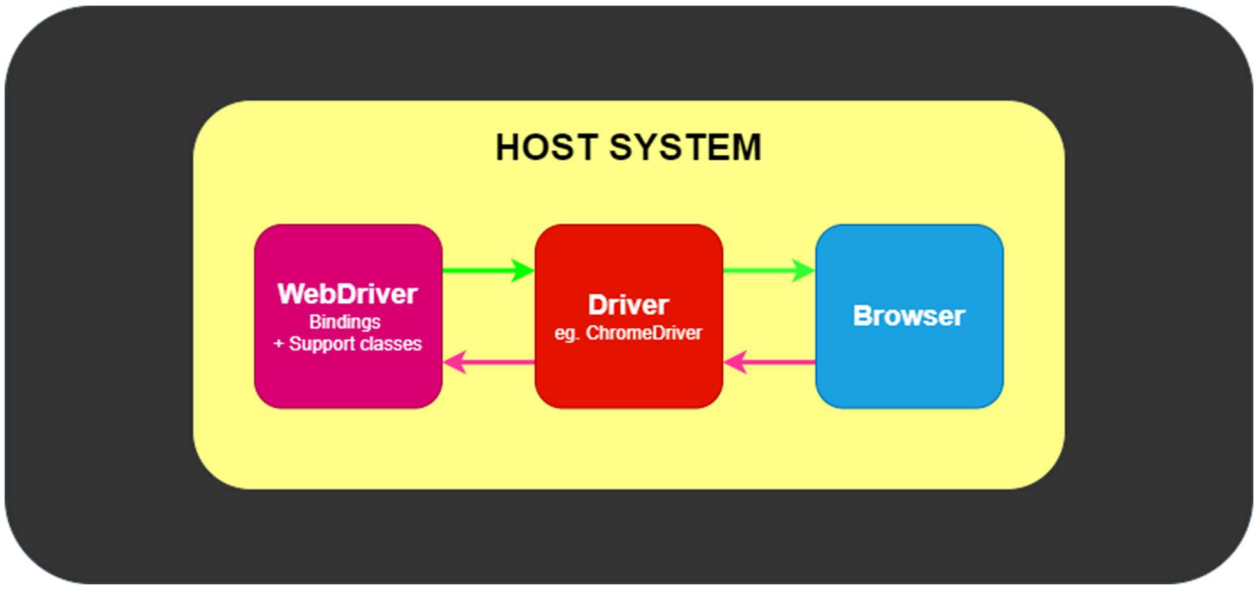
Selenium Components :

Building a test suite using WebDriver will require you to understand and effectively use several components. As with everything in software, different people use different terms for the same idea. Below is a breakdown of how terms are used in this description.

Terminology

- **API:** Application Programming Interface. This is the set of "commands" you use to manipulate WebDriver.

- **Library:** A code module that contains the APIs and the code necessary to implement them. Libraries are specific to each language binding, eg .jar files for Java, .dll files for .NET, etc.
- **Driver:** Responsible for controlling the actual browser. Most drivers are created by the browser vendors themselves. Drivers are generally executable modules that run on the system with the browser itself, not the system executing the test suite. (Although those may be the same system.) NOTE: *Some people refer to the drivers as proxies.*
- **Framework:** An additional library that is used as a support for WebDriver suites. These frameworks may be test frameworks such as JUnit or NUnit. They may also be frameworks supporting natural language features such as Cucumber or Robotium. Frameworks may also be written and used for tasks such as manipulating or configuring the system under test, data creation, test oracles, etc.



The driver is specific to the browser, such as ChromeDriver for Google's Chrome/Chromium, GeckoDriver for Mozilla's Firefox, etc. The driver runs on the same system as the browser. This may or may not be the same system where the tests themselves are executed.

WebDriver

WebDriver drives a browser natively, learn more about it.

WebDriver drives a browser natively, as a user would, either locally or on a remote machine using the Selenium server, marks a leap forward in terms of browser automation.

Selenium WebDriver refers to both the language bindings and the implementations of the individual browser controlling code. This is commonly referred to as just *WebDriver*.

Selenium WebDriver is a W3C Recommendation

- WebDriver is designed as a simple and more concise programming interface.
- WebDriver is a compact object-oriented API.
- It drives the browser effectively.

Waits

WebDriver can generally be said to have a blocking API. Because it is an out-of-process library that *instructs* the browser what to do, and because the web platform has an intrinsically asynchronous nature, WebDriver does not track the active, real-time state of the DOM. This comes with some challenges that we will discuss here.

From experience, most intermittent issues that arise from use of Selenium and WebDriver are connected to *race conditions* that occur between the browser and the user's instructions. An example could be that the user instructs the browser to navigate to a page, then gets a **no such element** error when trying to find an element.