DevOps Infrastructure

Introduction to Automation Testing with Selenium

Automation testing plays a critical role in DevOps by enabling rapid and reliable validation of software functionality. As part of the **DevOps Infrastructure Level 2** course, this class introduces automation testing using **Selenium**, a widely adopted tool for web UI testing.

In a DevOps workflow, continuous integration and continuous delivery (CI/CD) pipelines rely heavily on automated tests to ensure that software changes do not introduce regressions. Selenium helps automate these tests, particularly for web applications, by simulating user interactions and validating outcomes programmatically.

What is Automation Testing?

Automation testing is the process of using software tools to run tests on applications automatically, reducing the need for manual intervention. It improves accuracy, efficiency, and test coverage while saving time and resources.

Why is Automation Testing Important?

- **Consistency**: Automated tests execute the same way every time.
- **Efficiency**: They are faster than manual testing.
- **Coverage**: They allow more scenarios to be tested quickly.
- **CI/CD Integration**: Easily integrated into pipelines for continuous feedback.

Types of Software Testing

1. **Unit Testing**: Tests individual components or functions.
2. **Integration Testing**: Tests interactions between components.
3. **System Testing**: Tests the entire application as a whole.
4. **Acceptance Testing**: Validates the app against business requirements.
5. **Regression Testing**: Ensures changes haven't broken existing functionality.
6. **Performance Testing**: Measures speed, responsiveness, and stability.
7. **Security Testing**: Ensures data protection and app resilience against threats.
8. **UI Testing**: Verifies that the user interface behaves as expected.
9. **Black Box Testing**: Focuses on the application's output and functionality without knowledge of internal code or logic.
10. **White Box Testing**: Involves knowledge of the internal workings of the application, including code paths, conditions, and logic.

Where Selenium Fits

Selenium primarily falls under **Black Box Testing**, particularly in the areas of **UI Testing** and **Regression Testing**. Since it tests the behavior of the web application through user interactions, it does not require access to or knowledge of the internal code. However, Selenium tests can be paired with white box tests (like unit tests) for comprehensive coverage.

Understanding Selenium with Node.js

What is Selenium WebDriver?

Selenium WebDriver is a tool that allows you to automate interactions with web browsers. It controls the browser by simulating user actions such as clicks, typing, navigation, etc.

Headless Browser Mode

Headless Chrome means the browser runs without a graphical user interface (GUI). This is useful for running automated tests in environments where displaying the browser isn't necessary (e.g., CI/CD pipelines or servers). It runs faster and consumes fewer resources.

Can Selenium Work with Any Browser?

Yes. Selenium supports:

- Google Chrome (via ChromeDriver)
- Mozilla Firefox (via GeckoDriver)
- Safari
- Microsoft Edge

Drivers are required for each browser so Selenium can control them.

Test #1: Basic Selenium Test in Node.js

Step-by-Step Breakdown

```
const { By, Key, Builder, until } = require("selenium-webdriver");
const chrome = require("chromedriver");

// === TEST FUNCTION ===
async function testTeamRegistration() {
    let driver = await new Builder()
        .forBrowser("chrome")
        .build();

    try {
        // Step 1: Open the application URL
        await driver.get("http://localhost:5500/index.html");

        // Step 2: Fill in First Name
```

```
        await driver.findElement(By.id("firstname")).sendKeys("John");
        await driver.sleep(1000); // Wait 1 second after typing the first name

        // Step 3: Fill in Last Name
        await driver.findElement(By.id("lastname")).sendKeys("Doe");
        await driver.sleep(1000); // Wait 1 second after typing the first name

        // Step 4: Enter Group Size
        await driver.findElement(By.id("GroupSize")).sendKeys("3");
        await driver.sleep(1000); // Wait 1 second after typing the first name

        // Step 5: Click 'Add Member' Button
        await driver.findElement(By.id("addMemberBtn")).click();
        await driver.sleep(1000); // Wait 1 second after typing the first name

        // Step 6: Wait for the list to be populated with members
        await driver.wait(until.elementsLocated(By.css("#members option")), 1000);
        await driver.sleep(1000); // Wait 1 second after typing the first name

        // Step 7: Count the members added
        const optionsList = await driver.findElements(By.css("#members option"));
        await driver.sleep(1000); // Wait 1 second after typing the first name

        if (optionsList.length === 3) {
            console.log("✅ Test passed: 3 members were added.");
        } else {
            console.log(`❌ Test failed: Expected 3 members, but got
${optionsList.length}.`);
        }
    } catch (err) {
        console.error("⚠️ Test Error:", err);
    } finally {
        // Close the browser after 2 seconds to observe the result
        setTimeout(async function () {
            await driver.quit();
        }, 2000);  // Wait for 2 seconds before closing the browser
    }
}

// Run the test
testTeamRegistration();
```

Test #2: Validation Test (Missing Field)

```
const { By, Key, Builder, until } = require("selenium-webdriver");
const chrome = require("chromedriver");

// === TEST FUNCTION ===
async function testMissingFirstName() {
    let driver = await new Builder()
        .forBrowser("chrome")
        .build();

    try {
        await driver.get("http://localhost:5500/index.html");

        // Leave First Name blank
        await driver.findElement(By.id("lastname")).sendKeys("Doe");
```

```
        await driver.sleep(1000); // Wait 1 second after typing the last name

        await driver.findElement(By.id("GroupSize")).sendKeys("2");
        await driver.sleep(1000); // Wait 1 second after typing the group size

        // Click the Add Member button
        await driver.findElement(By.id("addMemberBtn")).click();

        // Wait briefly to check for the alert before continuing
        await driver.sleep(1000);

        // Handle the alert if it appears
        try {
            await driver.wait(until.alertIsPresent(), 1000); // Wait for the alert
            const alert = await driver.switchTo().alert();
            await alert.accept(); // Dismiss the alert
            console.log("⚠️ Alert dismissed");
        } catch (alertErr) {
            console.log("⚠️ No alert appeared");
        }

        // Wait for the form processing after clicking the button
        await driver.sleep(1000); // Wait for any UI updates

        // Check if the member was added or not by looking at the members list
        const optionsList = await driver.findElements(By.css("#members option"));

        // If no members are added, the test passed
        if (optionsList.length === 0) {
            console.log("✅ Test passed: No members added without first name.");
        } else {
            console.log("❌ Test failed: Members were added despite missing field.");
        }
    } catch (err) {
        console.error("⚠️ Test Error:", err);
    } finally {
        await driver.quit();
    }
}

// Run the test
testMissingFirstName();
```

Final Notes

- Selenium is flexible and powerful for web UI testing.
- Tests can validate DOM changes, form submissions, alerts, and more.
- These tests can be integrated into Jenkins pipelines to automate testing after deployments.