Automating build and test using Selenium and Maven is a common approach for web application testing in the Java ecosystem. Maven is a build automation tool used primarily for Java projects, and Selenium is a popular framework for automating web browsers. By combining these two technologies, you can easily set up and execute automated tests for your web applications. Below is a step-by-step guide to get you started:

1. **Prerequisites**:
   * Install Java Development Kit (JDK): Ensure you have Java installed on your system.
   * Install Maven: Download and set up Maven on your machine.
   * Set up your Selenium WebDriver: Choose the appropriate browser driver (e.g., ChromeDriver, GeckoDriver, etc.) and make sure it's available in your system's PATH.
2. **Project Setup**:
   * Create a new Maven project: Start by creating a new Maven project using your preferred IDE (Eclipse, IntelliJ, etc.), or you can use Maven's command-line archetype to create a new project structure.
   * Add Selenium and TestNG dependencies: Update your project's **pom.xml** file to include the Selenium WebDriver and TestNG dependencies. TestNG is a testing framework that can be used with Selenium to manage and execute test cases efficiently.
3. **Write Test Cases**:
   * Create your Selenium test classes: Write Java classes that contain your test cases. Each test class should extend **TestNG** and use **@Test** annotations to mark individual test methods.
4. **Configure WebDriver**:
   * In your test classes, set up the WebDriver instance before running the tests, and quit the WebDriver after the test execution is complete. Use appropriate annotations like **@BeforeTest** and **@AfterTest**.
5. **Build Automation**:
   * Maven's build lifecycle will take care of the compilation, dependency resolution, and test execution. By default, the **mvn test** command will trigger the execution of your test classes.
6. **Running Tests**:
   * Execute your tests using Maven: Open a terminal/command prompt, navigate to your project's root directory, and run **mvn test**.
7. **Reporting**:
   * TestNG generates test reports automatically. You can find the reports in the **target/surefire-reports** directory. Additionally, you can integrate other reporting libraries like ExtentReports to get more detailed and interactive reports.

Here's a basic example of a Selenium test class using TestNG:

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.\*;

public class MySeleniumTest {

WebDriver driver;

@BeforeTest

public void setUp() {

// Set up the WebDriver instance (e.g., ChromeDriver)

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

driver = new ChromeDriver();

}

@Test

public void testGoogleSearch() {

// Test logic

driver.get("https://www.google.com");

// Perform actions and assertions

}

@AfterTest

public void tearDown() {

// Quit the WebDriver instance

driver.quit();

}

}

Remember to replace **"path/to/chromedriver"** with the actual path to your ChromeDriver executable.

That's the basic outline for automating build and test using Selenium and Maven. You can expand upon this foundation by adding more test cases, organizing your code, and incorporating reporting tools to get detailed test results.