St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2024-2025
Class: TE-ITA/B, Semester: V

Subject: **DevOps Lab**

Experiment – 8: To setup and run Selenium tests in Jenkins using Maven.

1. **Aim:** To setup and run Selenium tests in Jenkins using Maven
2. **Objectives:** Aim of this experiment is that, the students will learn:
   - Selenium and how to automate your test cases for testing web elements
   - Introduction to X-Path, TestNG and integrate Selenium with Jenkins and Maven.
3. **Outcomes:** After study of this experiment, the students will learn following:
   - Introduction to Selenium
   - Installing Selenium
   - Creating Test Cases in Selenium WebDriver
   - Run Selenium Tests in Jenkins Using Maven
4. **Prerequisite:** Knowledge of Software Engineering concept of testing and test cases.
5. **Requirements:** Jenkins, JDK, Eclipse IDE, Firefox browser, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**
   Brief Theory: Refer shared material
7. **Laboratory Exercise**
   A. **Procedure:**
      a. **Answer the following:**
         - Explain Selenium suite?
         - What are the limitations of Selenium IDE?
      b. **Execute following (Refer the shared material) and attach screenshots:**
         - Create and run a test case on Chrome/Firefox browser with selenium IDE addon
         - Create a Maven Project in Jenkins and run selenium tests using selenium Grid

8. **Post-Experiments Exercise**
   A. **Extended Theory:**
      Nil
   B. **Questions:**
      - What are Locators? Explain its types.
      - What is the benefit of using Selenium Grid with Jenkins?
   C. **Conclusion:**
      - Write what was performed in the experiment.
      - Write the significance of the topic studied in the experiment.
   D. **References:**
      https://jenkins.io/doc/
      https://www.slideshare.net/abediaz/introduction-to-jenkins
      https://q-automations.com/2019/09/26/selenium-grid-with-jenkins/

#Answer the following:
Q) Explain Selenium suite?

The **Selenium Suite** is a collection of tools for automating web application testing across different browsers and platforms. It includes:

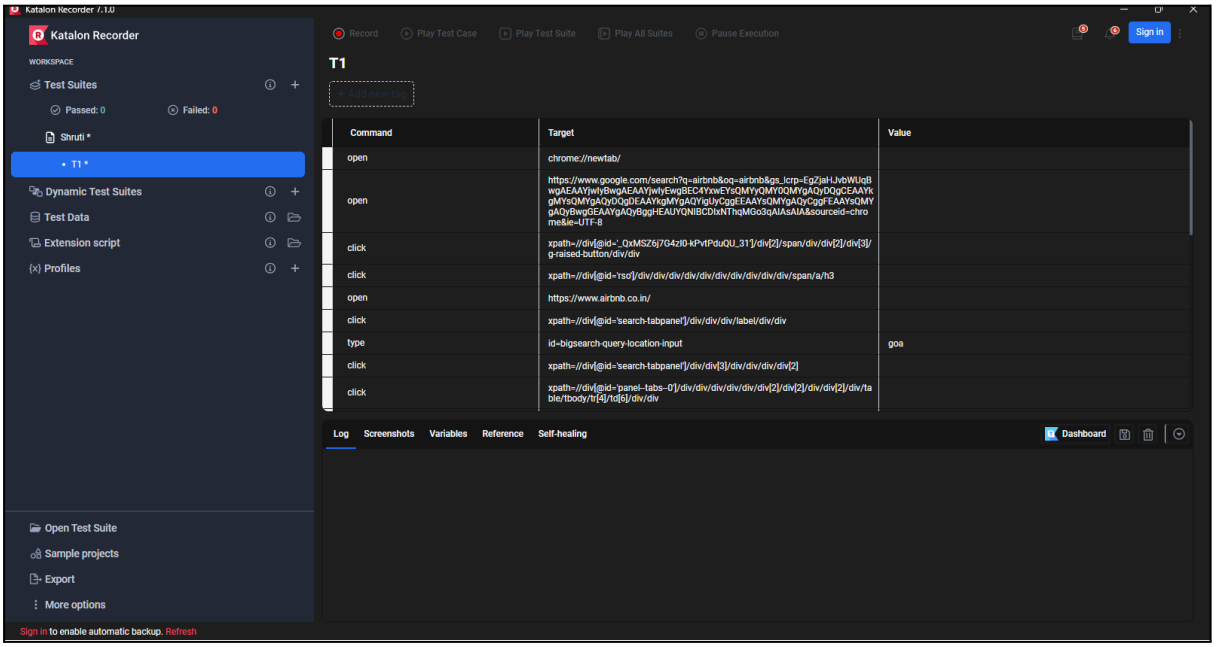
1. **Selenium IDE (Integrated Development Environment)**:
    - A browser extension (for Chrome and Firefox) used for recording and replaying test cases.
    - Best suited for prototyping simple test cases without programming knowledge.
2. **Selenium WebDriver**:
    - A robust API that interacts directly with web browsers, simulating user interactions with HTML elements.
    - Supports multiple programming languages (e.g., Java, Python, C#) and offers cross-browser compatibility (e.g., Chrome, Firefox, Safari).
    - Ideal for more complex, customized, and large-scale automated testing.
3. **Selenium Grid**:
    - Allows parallel test execution on multiple machines and browsers, enhancing test efficiency.
    - Helps distribute test scripts across different environments simultaneously, reducing overall test time.

Q)What are the limitations of Selenium IDE?

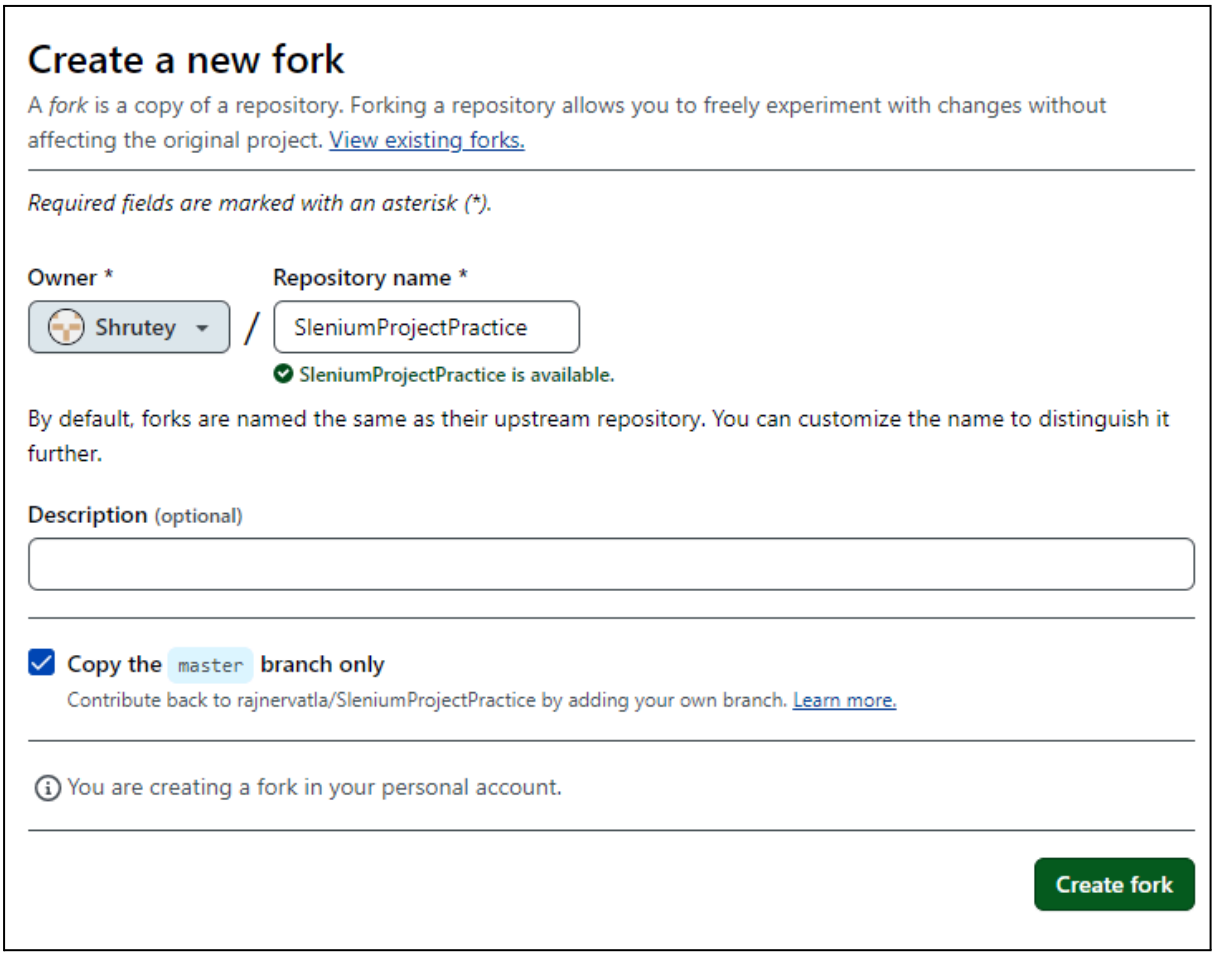
Despite its simplicity and ease of use, Selenium IDE has certain limitations:

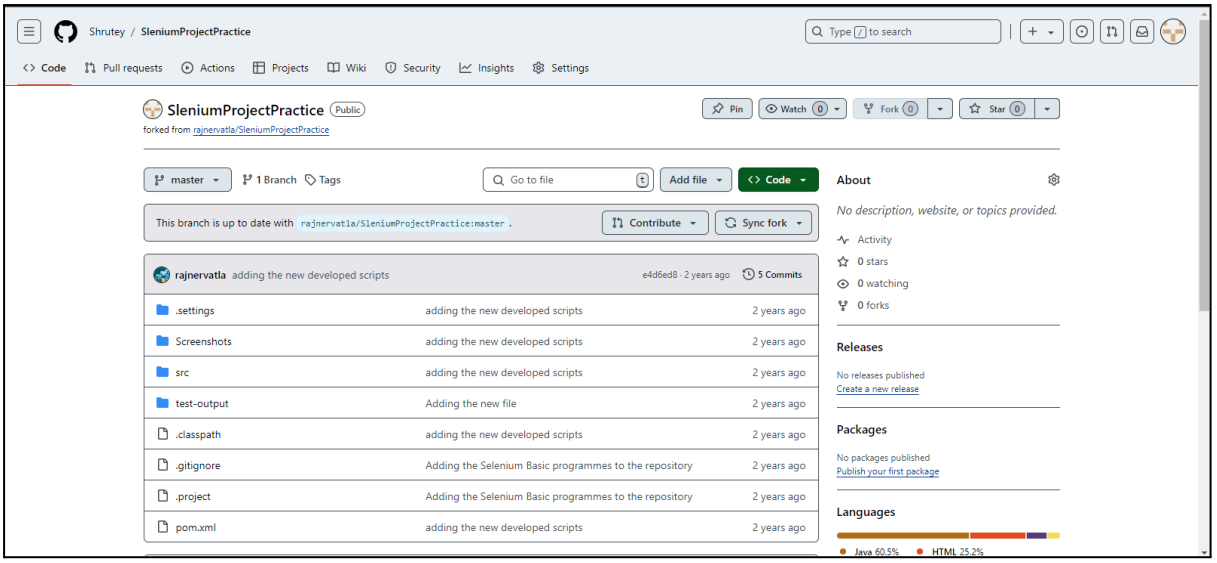
- **Limited Browser Support**: Primarily supports only Firefox and Chrome; it does not support cross-browser testing for all major browsers.
- **No Support for Complex Scenarios**: Lacks the functionality for testing complex, dynamic, or customized test flows, which require programming logic that the IDE cannot handle.
- **Limited Reporting Capabilities**: The IDE provides only basic reporting, which can make it challenging to interpret detailed test results or integrate with external reporting tools.
- **Cannot Handle Non-Web Applications**: Works only for web applications, meaning it cannot automate tasks for desktop, mobile, or other non-web-based applications.
- **Not Suitable for Large-Scale Testing**: Lacks integration with advanced CI/CD pipelines, which are essential for enterprise-level test automation.

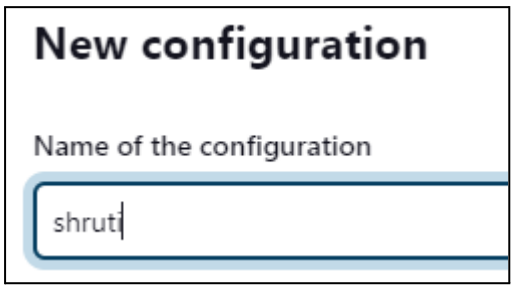
Create and run a test case on Chrome/Firefox browser with selenium IDE addon

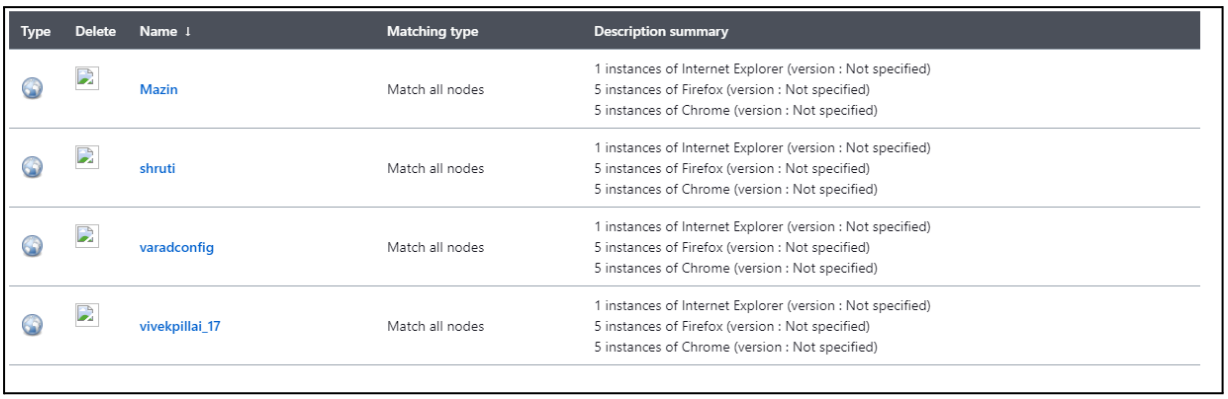


Create a Maven Project in Jenkins and run selenium tests using selenium Grid

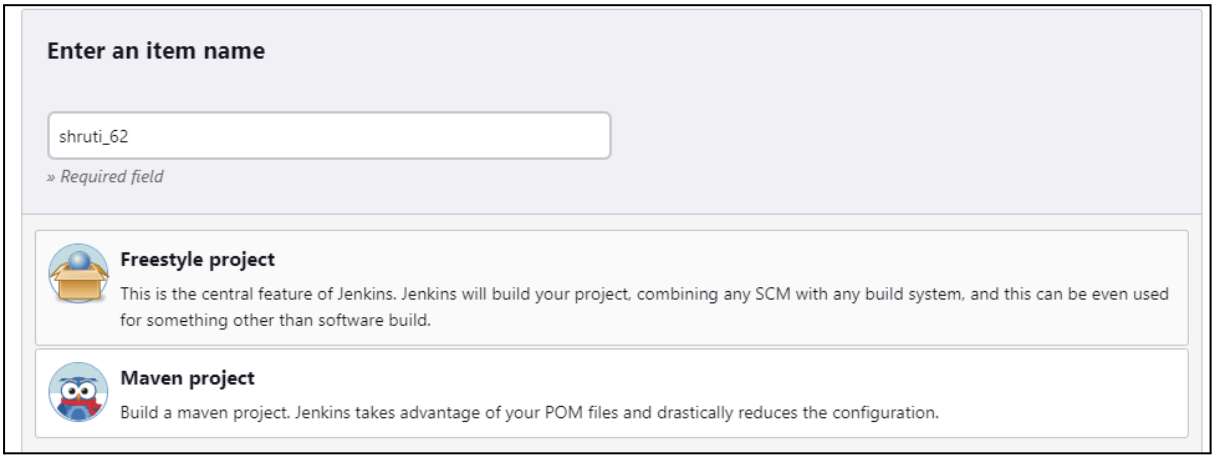


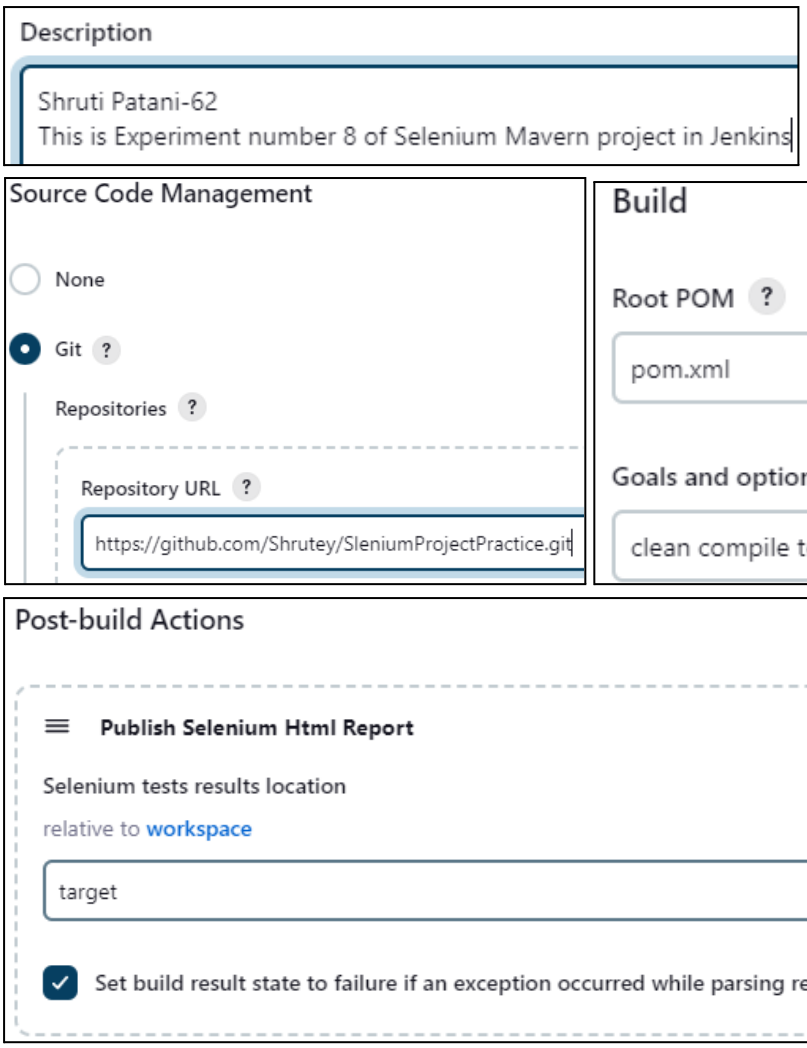






Type	Delete	Name ↓	Matching type	Description summary
		Mazin	Match all nodes	1 instances of Internet Explorer (version : Not specified) 5 instances of Firefox (version : Not specified) 5 instances of Chrome (version : Not specified)
		shruti	Match all nodes	1 instances of Internet Explorer (version : Not specified) 5 instances of Firefox (version : Not specified) 5 instances of Chrome (version : Not specified)
		varadconfig	Match all nodes	1 instances of Internet Explorer (version : Not specified) 5 instances of Firefox (version : Not specified) 5 instances of Chrome (version : Not specified)
		vivekpillai_17	Match all nodes	1 instances of Internet Explorer (version : Not specified) 5 instances of Firefox (version : Not specified) 5 instances of Chrome (version : Not specified)




Description
Shruti Patani-62
This is Experiment number 8 of Selenium Mavern project in Jenkins
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/Shrutey/SleniumProjectPractice.git
Build
Root POM
pom.xml
Goals and options
clean compile test
Post-build Actions
Publish Selenium Html Report
Selenium tests results location
relative to workspace
target
Set build result state to failure if an exception occurred while parsing result files

The Selenium test reports.

Result	Name	Tests total	Tests passes	Tests failures	Commands passes	Commands failures	Commands errors	Duration
	Summen	0	0	0	0	0	0	0 s = 0.0 min and 0 s

Console Output

```
Started by user admin
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\shruti_62
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Shrutey/SleniumProjectPractice.git
 > C:\Program Files\Git\bin\git.exe init C:\ProgramData\Jenkins\.jenkins\workspace\shruti_62 # timeout=10
Fetching upstream changes from https://github.com/Shrutey/SleniumProjectPractice.git
 > C:\Program Files\Git\bin\git.exe --version # timeout=10
 > git --version # 'git version 2.34.1.windows.1'
 > C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/Shrutey/SleniumProjectPractice.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/Shrutey/SleniumProjectPractice.git # timeout=10
 > C:\Program Files\Git\bin\git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision e4d6ed8719a8f959346555f9a86feb825768cabf (refs/remotes/origin/master)
 > C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
 > C:\Program Files\Git\bin\git.exe checkout -f e4d6ed8719a8f959346555f9a86feb825768cabf # timeout=10
Commit message: "adding the new developed scripts"
First time build. Skipping changelog.
Parsing POMs
```