Jenkins – Session 4

Learn CICD Tool Jenkins from Scratch

Integrate Maven Project with Jenkins Job

Trainer – Haradhan Pal

Agenda

- Introduction of Maven and Installation
- Creating a Maven Project in Eclipse
- How to Execute Selenium Script through Maven and TestNG
- Apache-Maven installation and Maven Home Variable Set Up
- Maven Commands
- Key Difference between Maven and Jenkins
- Installation of Maven Plugin in Jenkins
- Setup Java and Maven Path in Jenkins
- Execute Maven Project using Jenkins

Introduction of Maven and Installation

- Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
- Maven is used to define project structure, dependencies, build, and test management.
- Using pom.xml user can configure dependencies needed for building testing and running code.

- Execute program using Maven user need to follow the below steps:
  - Install Maven
  - Add dependency
  - Execute the framework

- Follow below steps install maven in eclipse, but if user are using luna or above version, they might have maven by default installed with their eclipse, if not below steps need to be followed:
  - 1. Click on the Help menu and choose Eclipse Market place.
  - 2. Search for Maven with Eclipse; user will get below software result. As I have installed already, so I will not get an install option; for other case, they will have to click the install option.
  - 3. Accept the agreement and restart the eclipse which might take sometimes.

Creating a Maven Project in Eclipse

- Steps for creating Maven project:
- 1. In Eclipse IDE, create a new project by selecting File | New | Other from Eclipse menu.
- 2. On the New dialog, select Maven | Maven Project and click Next
- 3. On the New Maven Project dialog select the Create a simple project and click Next
- 4. Enter Group Id: and Artifact Id: and click finish. The groupId is the id of the project's group. Generally, it is unique amongst an organization. The artifactId is the id of the project. It specifies the name of the project.
- 5. Select pom.xml from Project Explorer. pom.xml file will Open in Editor section
- 6. Add the Selenium-Java and TestNG maven dependencies to pom.xml in the <project> node (Maven Repository Path: (https://mvnrepository.com/).

How to Execute Selenium Script through Maven and TestNG

- ❑ Create a Package and Create a New TestNG Class. Add some code in the class.

Sample Code:

```
@Test
  public void openFacebookPage() {
  System.setProperty("webdriver.chrome.driver","C:\\Users\\XX\\Desktop\\chromedriver.exe");
  WebDriver driver = new ChromeDriver();
  driver.manage().window().maximize();
  driver.get("https://www.facebook.com/");
  System.out.println("Facebook page opened successfully");
  System.out.println(driver.getTitle());
  driver.quit();
}
```

- ❑ Right-click on the TestNG file, and select Run As TestNG Test
- ❑ User need to add the below plugins in the pom xml file
- ❖ Maven surefire plugin
- ❖ Maven complier plugin
- ❑ Right-click on the MavenProject → Run As → Maven Test

How to Execute Selenium Script through Maven and TestNG

```
 <build>
   <pluginManagement>
    <plugins>
     <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.0.0-M5</version>
         </plugin>

    <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.0</version>
            <configuration>
                <source>10</source>
                <target>10</target>
            </configuration>
        </plugin>

  </plugins>
 </pluginManagement>
</build>
```

Apache-Maven installation and Maven Home Variable Set Up

- Go to http://maven.apache.org/download.cgi to download installation file for Maven
- Ensure JAVA_HOME environment variable is set and points to user JDK installation
- Unzip apache-maven-3.8.6-bin.zip
- Select My Computer and select Properties by Right-Clicking on My Computer. Click on Advanced system settings. Click on 'Advanced' tab and then click on 'Environment Variables..'.
- Click on New button under 'System Variables..'
- Write 'MAVEN_HOME' in the Variable name box then enter apache-maven-3.8.6 Maven path in the Variable value box and click OK.
- Add the bin directory of the created directory apache-maven-3.8.6 to the Path environment variable.

Maven Commands

- **Maven Command:**

- Open a command prompt window (Press Windows + R at the same time. Then enter cmd in the Run window and click on the Ok button).
- mvn -version in command prompt to check the version of the Maven already installed user system
- Go to the directory where user saved the maven project (cd location of the project)
- mvn clean: This command cleans the maven project by deleting the target directory
- mvn complie: It's used to compile the source Java classes of the project.
- mvn test: This command is used to run the test cases of the project using the maven-surefire-plugin.
- mvn package: This command builds the maven project and packages them into a JAR, WAR, etc.
- mvn help: This command prints the maven usage and all the available options for us to use.

Key Difference between Maven and Jenkins

Comparison Parameter	*Jenkins*	*Maven*
Purpose	Jenkins is a CI/CD tool, which uses various other tools/plugins to automate the complete build cycle.	Maven is a build lifecycle management tool that helps in dependency management and creating automated builds.
Programming Models	Jenkins uses groovy for pipeline creation or uses the GUI for standalone jobs.	Maven works on an XML based structure for performing various actions in a build lifecycle.
Dependency Management	Jenkins can provide dependency between various jobs, but this is more of an action dependency.	Maven provides capabilities for resource dependency management, including third-party dependencies for builds.
Scope	Jenkins can do much more than build and deploy, such as server management, database management. It uses integration with various tools for this purpose.	Maven is more restricted around code and builds. This serves as a plugin in Jenkins for building Maven-based java projects.

Installation of Maven Plugin in Jenkins

- The Maven Plugin is a plugin that provides the capabilities to configure, build, and run Maven-based projects in Jenkins. This is a must pre-requisite for the integration of Maven with Jenkins.
- Click on the Manage Jenkins link in the left menu bar of Jenkins window
- Under the System Configuration section, click on the Manage Plugins link.
- Under the Plugin Manager, click on the Available tab and search for the maven plugin.
- Select the checkbox in front of the Maven Integration plugin and click on the Download now and install after restart button.

Setup Java and Maven Path in Jenkins

- Click on the Manage Jenkins link in the left menu bar of Jenkins window.
- Now click on the Global Tool Configuration link.
- Set the JDK path in the JAVA_HOME textbox (Either JDK 1.8 or JDK 11 only).
- Enter the path of Maven in the MAVEN_HOME textbox.
- Click on Apply and then Save button.

Execute Maven Project using Jenkins

- Click on New Item link to create a job on Jenkins
- Enter suitable an Item name, select Maven project and click on OK button.
- Describe the project in the description section.
- Enter the Project pom.xml file path location of the project in the Root POM text box under Build section.
- Enter "clean install" in the Goals and options textbox in same Build section.
- Click on ‛Apply‛ and ‛Save‛.
- Click on ‛Build Now‛ button. It will invoke pom.xml.
- Right click on Build Number and click on Console Output to see the result.