1. What is Maven

Maven as a tool that helps software developers manage their projects more easily. It does three main things:

Project Management: It tells you how to structure your project. Imagine you're building a house. Maven helps you decide where to put the bedrooms, the kitchen, and the bathroom, so everything is organized and makes sense.

Maven provides a structured and standardized way to manage the entire project lifecycle, from project creation to deployment.

Dependency Management: When you build software, you often need other pieces of code or libraries. Maven makes sure you have all those pieces you need. It's like having a shopping list to get all the materials for your house - wood, nails, paint, etc. Maven can automatically download and manage these dependencies.

Build Process: Maven helps you build your project. Building means taking all the pieces of your project and putting them together in the right way. It's like assembling all the parts of a toy. With Maven, you just press a button, and it knows how to do it. You don't have to figure out the exact order or steps; Maven does that for you.

Maven automates the build process, making it easier to compile, test, package, and distribute a project. It provides predefined build phases and goals, reducing the need for manual configuration. Top of Form

**Plugin Ecosystem:** Maven supports a rich ecosystem of plugins that extend its functionality. These plugins can be used for various tasks, such as code analysis, reporting, and integration with other tools.

**Integration with IDEs:** Many popular Integrated Development Environments (IDEs) have built-in Maven integration, simplifying project development and management.

1. Read data from excel file

public static String TESTDATA\_SHEET\_PATH = currentDir

+"//src//main//java//com//makemytrip//qa//testdata//MakeMyTrip\_TestData.xlsx";

static Workbook book;

static Sheet sheet;

static JavascriptExecutor js;

public static Object[][] getTestData(String sheetName) {

FileInputStream file = null;

try {

file = new FileInputStream(TESTDATA\_SHEET\_PATH);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

try {

book = WorkbookFactory.create(file);

} catch (InvalidFormatException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

sheet = book.getSheet(sheetName);

Object[][] data = new Object[sheet.getLastRowNum()][sheet.getRow(0).getLastCellNum()];

// System.out.println(sheet.getLastRowNum() + "--------" +

// sheet.getRow(0).getLastCellNum());

for (int i = 0; i < sheet.getLastRowNum(); i++) {

for (int k = 0; k < sheet.getRow(0).getLastCellNum(); k++) {

data[i][k] = sheet.getRow(i + 1).getCell(k).toString();

// System.out.println(data[i][k]);

}

}

return data;

}

1. We have multiple tests in our class.. and we have multiple classes as well... how we will execute certain test cases from these
2. Using groups concepts

<suite name=””>

<test name=””>

<groups>

<run>

<include name=”smoke”/>

</run>

</groups>

</classes>

<class name=”1”/>

<class name=””/>

<class name=””/>

<class name=””/>

<class name=”100”/>

</classes>

</test>

</suite>

**@Test(groups={“smoke”})**

1. how to execute failed test cases in TestNG

Go to test-output folder>>testng-failed.xnl>>right click run as testng suite

1. What is loading

in Java, when your program starts running, it needs to access classes (which are like predefined tools or blueprints for objects) and resources (like data files or configuration files) to perform its tasks. Loading involves bringing these classes and resources into memory so that the program can use them

1. Mvn package will package create the jar or war file to share the code across
2. Method overloading and overriding

Method Overloading Example:

Suppose you have a Selenium project, and you want to click on web elements in various ways. You can use method overloading to create different click methods that accept different parameters. This allows you to click elements based on different criteria.

In this example, we've created an overloaded click method to click web elements in different ways. You can use this CustomWebDriver class to interact with elements by passing an element, a By locator, or the text contained in the element.

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

public class CustomWebDriver {

private WebDriver driver;

public CustomWebDriver(WebDriver driver) {

this.driver = driver;

}

// Overloaded click method to click by element

public void click(WebElement element) {

element.click();

} // Overloaded click method to click by By locator

public void click(By locator) {

WebElement element = driver.findElement(locator);

element.click();

} // Overloaded click method to click by element's text

public void click(String elementText) {

By locator = By.xpath("//\*[contains(text(), '" + elementText + "')]");

WebElement element = driver.findElement(locator);

element.click();

}

}

Method Overriding Example:

Suppose you want to customize the behavior of the findElement method in Selenium to handle exceptions and take screenshots in case of element not found. You can create a custom WebDriver class that overrides the findElement method.

In this example, the CustomWebDriver class extends the original WebDriver class and overrides the findElement method. When an element is not found, it captures a screenshot and then re-throws the exception. This way, you've customized the behavior of the findElement method to suit your specific needs in your Selenium project.

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.remote.Augmenter;

public class CustomWebDriver extends WebDriverWrapper {

public CustomWebDriver(WebDriver driver) {

super(driver);

}

@Override

public WebElement findElement(By by) {

try {

return super.findElement(by);

} catch (org.openqa.selenium.NoSuchElementException e) {

// Handle the exception (e.g., take a screenshot)

TakesScreenshot tsDriver = (TakesScreenshot) driver;

byte[] screenshot = tsDriver.getScreenshotAs(OutputType.BYTES);

// Save the screenshot or log the error

throw e; // Re-throw the exception

}

}

}

1. Describe static method/why main method is static

The main method is the entry point of a Java program, where the program starts running. It's like the "launch button" for your program. Here's why it's static:

1. **No Object Required**: When you run a Java program, there is no object created from the class where the main method resides. In other words, the program starts without creating an instance of the class. Static methods don't require an object to be called, so they are the natural choice for the entry point.
2. **Consistency**: Making main method static ensures that it's available for execution even before any objects of the class are created. This consistency is important because you don't always want to create an instance just to start your program.

So, in plain language, the main method is static because it doesn't need an object to get things started, and it ensures that your program can run right from the beginning without any extra steps. It's like the "power button" for your Java program.

1. WebDriver driver = new ChromeDriver() explanation

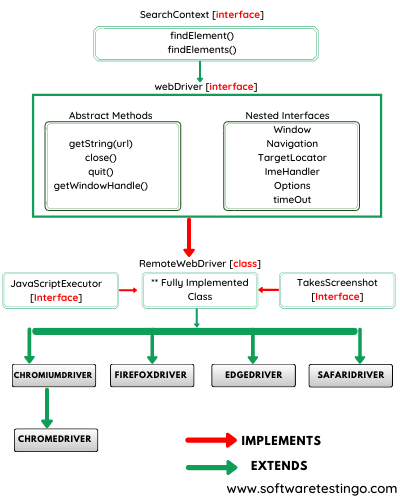
ChromeDriver is a class

ChromeDriver() is the default constructor of ChromeDriver class

New ChromeDriver()=We are creating object for ChromeDriver class

driver = new ChromeDriver() –We need to store instance created in a variable to perform actions. above created instance is stored in “driver” local variable.

WebDriver= return type . Here we can give ChromeDriver also as return type. Since chromeDriver class is implementing WebDriver Interface we can give WebDriver also as return type



1. How to prevent overloading and overriding

**Preventing Overloading:**

Overloading occurs when you have multiple methods with the same name but different parameters in the same class. To prevent overloading for a particular method, simply don't create multiple methods with the same name in that class. Only use unique method names for methods that should not be overloaded.

**Prevent Overriding a Method:**

If you want to prevent a subclass from overriding a method, mark the method in the superclass as final. For example:

public final void myMethod() { // Your method implementation } }

Subclasses won't be able to override the myMethod because it's marked as final.

**Prevent Overriding a Class:**

To prevent a class from being subclassed, use the final keyword for the class itself. For example:

public final class MyFinalClass { // Class members and methods }