Handling Test Data and Executing It in Multiple Environments for Bus Ticket Booking Application

the process of creating a TestNG framework for testing a bus ticket booking application in multiple environments. We will be using the Page Object Model (POM) approach to organize test cases efficiently. This framework will handle test data and allow tests to be executed in different environments.

- The TestNG framework test Data Management Managing test data for various test scenarios.
- Page Object Model (POM) Organizing test cases using POM for efficient maintenance and readability.
- Maven Integration Managing dependencies and build processes using Maven.

Step-by-Step Implementation

Step 1: Define Pages to Be Tested

Before setting up the framework, determine which pages and functionalities of the bus ticket booking application you want to test. Define a list of pages to be tested, such as the Home Page, Register Page, Login Page.

Step 2: Create a Base Class

Create a Base class to handle the driver session and application details. The Base class is the foundation of the framework and should include:

- Driver session initialization and management.

- WebDriver instance.

- Reporting setup (extent reports).

- Any suite or test-level configurations.

Step 3: Page Class

For each page identified in Step 1, create a corresponding page class. In these page classes:

- Define the elements that need to be tested (buttons, input fields).

- Write locators for these elements.

- Utilize the `@FindBy` annotation from PageFactory to group elements together.

Step 4: Action Methods

In the page classes, create action methods to define the actions to be performed on the elements. These methods should interact with the page elements using WebElements defined earlier.

Step 5: Test Classes

Create test classes for each page or functionality to be tested. In these test classes:

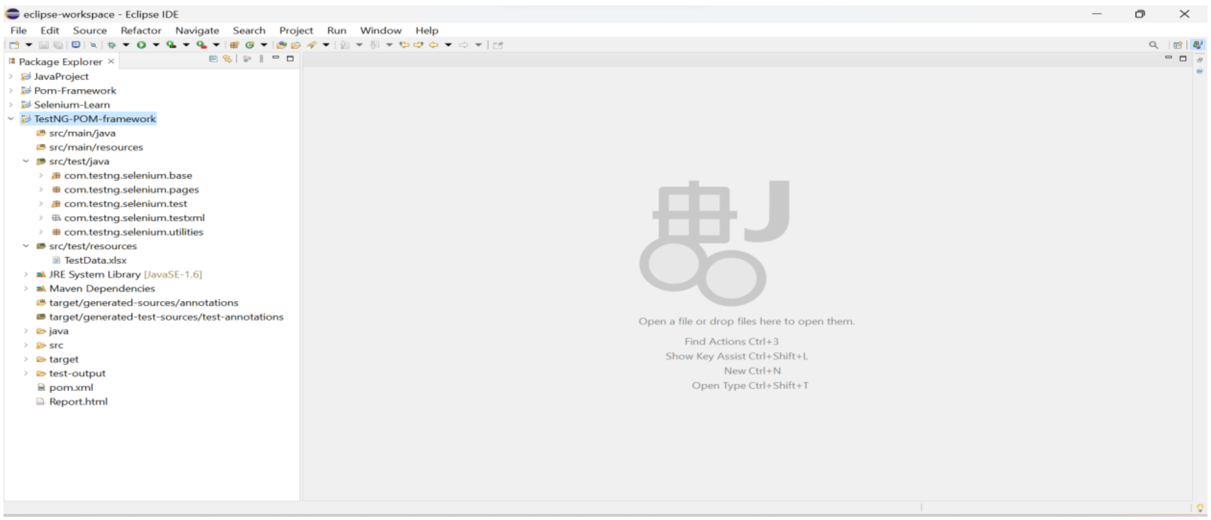
- Extend the Base class to access the driver and browser setup.
- Create objects of the respective page classes.
- Call the action methods defined in the page classes to perform test actions.

Step 6: TestNG and Listeners

Implement TestNG listeners to capture and log events during test execution. This helps in generating detailed test reports and logs.

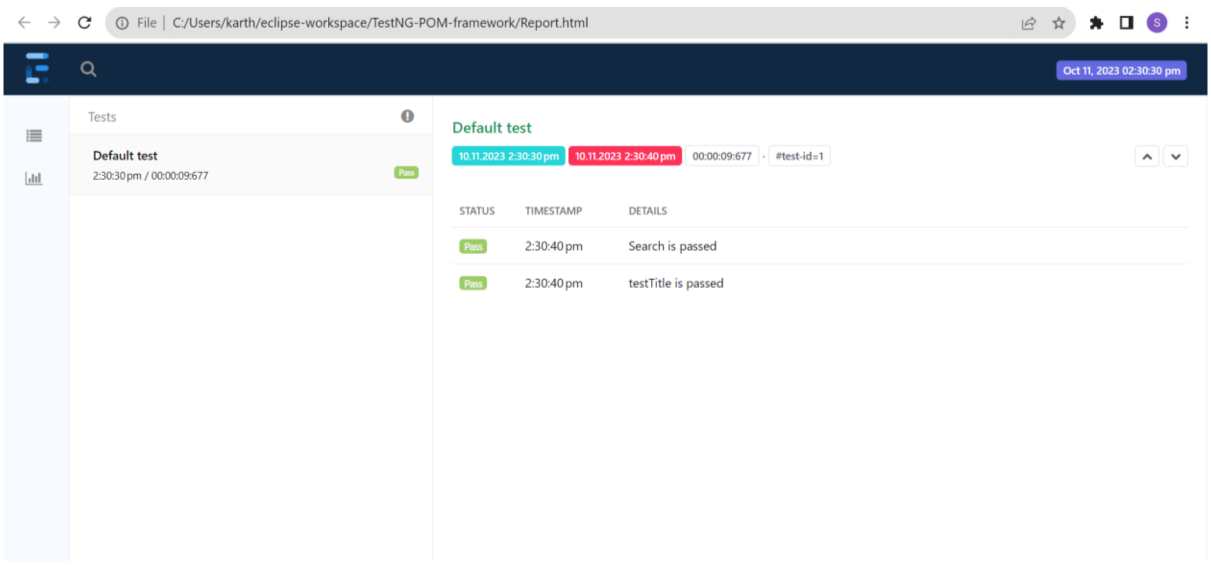
Step 7: Maven Integration

Use Maven to manage dependencies and build processes. Define dependencies, plugins, and repositories in your `pom.xml` file to facilitate the build and execution of your tests.


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer

```
> JavaProject
> Pom-Framework
> Selenium-Learn
v TestNG-POM-framework
    src/main/java
    src/main/resources
  v src/test/java
    > com.testng.selenium.base
    > com.testng.selenium.pages
    > com.testng.selenium.test
    > com.testng.selenium.testxml
    > com.testng.selenium.utilities
  v src/test/resources
      TestData.xlsx
  > JRE System Library [JavaSE-1.6]
  > Maven Dependencies
    target/generated-sources/annotations
    target/generated-test-sources/test-annotations
  > java
  > src
  > target
  > test-output
    pom.xml
    Report.html
```

Open a file or drop files here to open them.
Find Actions Ctrl+3
Show Key Assist Ctrl+Shift+L
New Ctrl+N
Open Type Ctrl+Shift+T

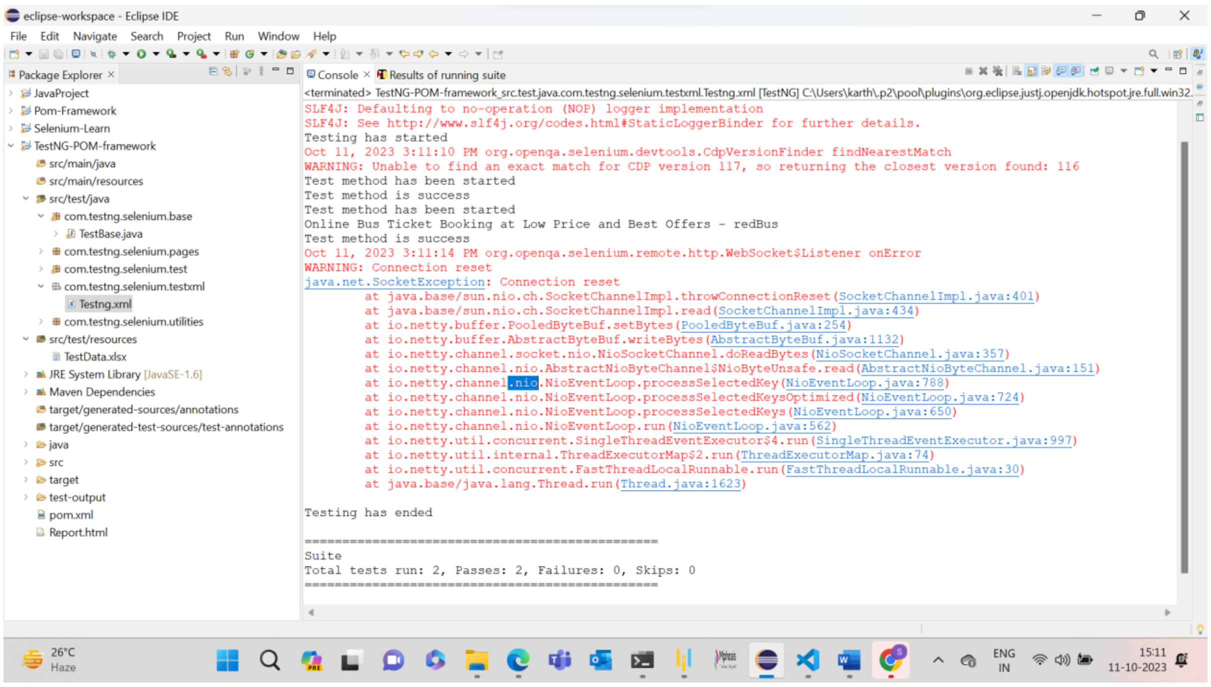

File | C:/Users/karth/eclipse-workspace/TestNG-POM-framework/Report.html
Oct 11, 2023 02:30:30 pm
Tests

Default test
2:30:30 pm / 00:00:09:677 Pass

Default test

10.11.2023 2:30:30 pm · 10.11.2023 2:30:40 pm · 00:00:09:677 · #test-id=1

STATUS	TIMESTAMP	DETAILS
Pass	2:30:40 pm	Search is passed
Pass	2:30:40 pm	testTitle is passed



```
<terminated> TestNG-POM-framework_src.test.java.com.testng.selenium.testxml.Testng.xml [TestNG] C:\Users\karth\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Testing has started
Oct 11, 2023 3:11:10 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 117, so returning the closest version found: 116
Test method has been started
Test method is success
Test method has been started
Online Bus Ticket Booking at Low Price and Best Offers - redBus
Test method is success
Oct 11, 2023 3:11:14 PM org.openqa.selenium.remote.http.WebSocket$Listener onError
WARNING: Connection reset
java.net.SocketException: Connection reset
	at java.base/sun.nio.ch.SocketChannelImpl.throwConnectionReset(SocketChannelImpl.java:401)
	at java.base/sun.nio.ch.SocketChannelImpl.read(SocketChannelImpl.java:434)
	at io.netty.buffer.PooledByteBuf.setBytes(PooledByteBuf.java:254)
	at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:1132)
	at io.netty.channel.socket.nio.NioSocketChannel.doReadBytes(NioSocketChannel.java:357)
	at io.netty.channel.nio.AbstractNioByteChannel$NioByteUnsafe.read(AbstractNioByteChannel.java:151)
	at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:788)
	at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:724)
	at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:650)
	at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:562)
	at io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:997)
	at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
	at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
	at java.base/java.lang.Thread.run(Thread.java:1623)

Testing has ended

===============================================
Suite
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================
```

GitHub repository : https://github.com/Karthick-Office/Project01.git

My "TestNG-POM-framework" Project is present under the "TestNG-POM-framework" folder in the GitHub repository

For Handling Test Data and Executing It in Multiple Environments for Bus Ticket Booking Application