
🧪 Automation Testing of OrangeHRM using Selenium & TestNG (Maven Project)

📑 Table of Contents

1️⃣ Project Title

Automation Testing of OrangeHRM Login Page using Selenium WebDriver and TestNG

2️⃣ One-Line Project Summary

This project automates the validation of the OrangeHRM login page by verifying the **page title**, **URL**, and **login header text** using **Selenium WebDriver with TestNG** in **Eclipse IDE**, managed through **Maven**.

3️⃣ Aim of the Project

The main aim of this project is to:

- Understand **real-time web automation testing**
- Use **TestNG assertions** for validation
- Automate basic but critical UI validations
- Implement a **Maven-based Selenium framework**
- Follow an **industry-standard project structure** that recruiters expect

---

4️⃣ Tools & Technologies Used

Tool / Technology	Purpose
Java	Programming language
Selenium WebDriver	Browser automation
TestNG	Test execution & assertions
Maven	Dependency management
Eclipse IDE	Development environment
Chrome Browser	Test execution
OrangeHRM Demo Site	Application under test

5️⃣ Project Folder Structure

The project follows a **standard Maven structure**, which is highly preferred in real-time automation projects:

```
Artifact_id
│
├── src/test/java
│   └── _Package
│       └── _Java_Class.java
│
├── src/test/resources
```

```
├── Maven Dependencies
├── JRE System Library
├── pom.xml
├── target
└── test-output
```

- This structure improves **maintainability, scalability, and readability**.

---

6 Test Scenarios Covered

The following test scenarios are automated:

1. Validate the **page title** of OrangeHRM
2. Validate the **URL** after page load
3. Validate the **login header text (Login)**

Each scenario is independent and executed using **TestNG priorities**.

---

7 Automation Approach & Design

- Used **TestNG annotations** (`@BeforeMethod`, `@Test`, `@AfterMethod`)
- Browser setup and teardown handled centrally
- **Implicit Wait** implemented to handle synchronization
- Assertions used to validate expected vs actual results
- Tests executed in a **priority-based order**

This approach mimics **real industry automation frameworks**.

---

8 Code Implementation

📌 pom.xml (Maven Configuration)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
  <modelVersion>4.0.0</modelVersion>

  <groupId>Artifact_id</groupId>
  <artifactId>Artifact_id</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>

    <!-- Selenium WebDriver -->
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>4.39.0</version>
    </dependency>

    <!-- TestNG -->
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>7.11.0</version>
      <scope>test</scope>
    </dependency>

  </dependencies>

</project>
```

📌 Test Class (_Java_Class.java)

```
package _Package;

import java.time.Duration;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
```

```
public class _Java_Class {

    WebDriver driver;

    @BeforeMethod
    public void setup() {
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
        driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");
    }

    @AfterMethod
    public void end_test() throws InterruptedException {
        Thread.sleep(4000);
        driver.close();
    }

    @Test(priority = 1)
    public void check_title() {
        String actual_Title = driver.getTitle();
        String expected_Title = "OrangeHRM";
        Assert.assertEquals(actual_Title, expected_Title, "Title mismatch");
    }

    @Test(priority = 2)
    public void check_url() {
        String actual_URL = driver.getCurrentUrl();
        String expected_URL =
"https://opensource-demo.orangehrmlive.com/web/index.php/auth/login";
        Assert.assertEquals(actual_URL, expected_URL, "URL mismatch");
    }

    @Test(priority = 3)
    public void check_h5() {
        WebElement head5 = driver.findElement(By.xpath("//h5"));
        String actual_text = head5.getText();
        Assert.assertEquals(actual_text, "Login");
    }
}
```

9️⃣ Assertions & Validation Strategy

- **Assert.assertEquals()** is used to compare:
  - Actual title vs Expected title
  - Actual URL vs Expected URL
  - Actual UI text vs Expected UI text
- Assertion failures immediately mark the test as **FAILED**
- Clear failure messages improve debugging

🔟 Challenges Faced & Solutions

🔴 Challenge 1: Synchronization Issues

Solution: Implemented **Implicit Wait** to handle dynamic loading elements.

🔴 Challenge 2: Test Dependency

Solution: Used **TestNG priorities** to ensure logical execution order.

🔴 Challenge 3: Browser Stability

Solution: Proper setup and teardown using `@BeforeMethod` and `@AfterMethod`.

1️⃣1️⃣ Execution Flow

1. Browser launches
2. OrangeHRM login page opens
3. Title validation executes
4. URL validation executes
5. Login header text validation executes
6. Browser closes after each test

1️⃣2️⃣ Key Learning Outcomes

- Hands-on experience with **Selenium WebDriver**
- Practical understanding of **TestNG framework**

- Maven dependency management
- Writing clean and maintainable automation code
- Understanding **real-world automation testing workflow**

1️⃣3️⃣ Conclusion

This project demonstrates my ability to design and implement a **basic yet industry-relevant automation framework** using Selenium and TestNG. It reflects my understanding of **UI validation, test structure, assertions, and Maven configuration**, making it suitable for **entry-level SDET / Automation Tester roles**.
