

TestNG Data Driven Testing

Press Space for next page →



Agenda

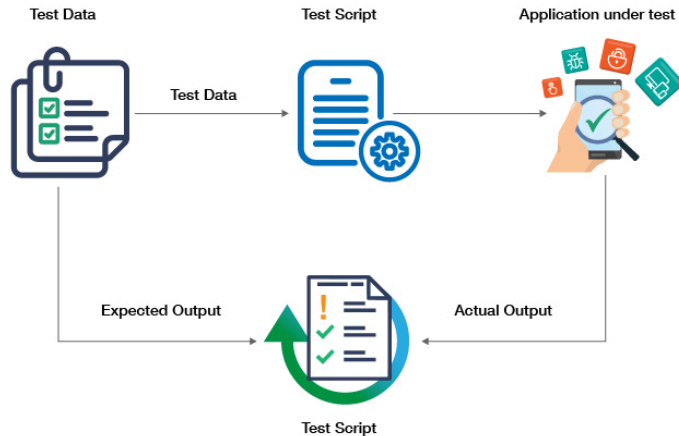
1. TestNG Data Driven Testing
2. Code Example
3. Code Example(Data Driven Testing)
4. Assertion in TestNG
5. Code Example: TestNG Assertion
6. Running the Tests using testng.xml
7. Apache POI
8. Apache POI: Dependency
9. Code Example: Read Excel File
10. Code Example: Load Data from Excel

TestNG Data Driven Testing

Data-driven testing is a test design and execution strategy where test scripts read test data from data sources like Excel, CSV, XML, databases, etc. and use it as test input.

- It can be used when one Test Case has to execute with different set of Data

Example: Let's say we have 20 credentials and we need to test our application with all credentials so we will be writing only 1 test script and test will be passed via Excel sheet.



Code Example

```
1  class DataDrivenTest {
2
3      @Test(dataProvider = "data-provider")
4      public void testMethod(String username, String password) {
5          System.out.println("Username: " + username + " Password: " + password);
6      }
7
8      @DataProvider(name = "data-provider")
9      public Object[][] dataProviderMethod() {
10
11          Object[][] data = new Object[3][2];
12
13          data[0][0] = "username1";
14          data[0][1] = "password1";
15
16          data[1][0] = "username2";
17          data[1][1] = "password2";
18
19          data[2][0] = "username3";
20          data[2][1] = "password3";
21
22          return data;
23      }
24 }
```

Code Example(Data Driven Testing)

```

1  class DataDrivenTest {
2      WebDriver driver;
3      @BeforeMethod
4      public void setup() {
5          driver = new ChromeDriver();
6          driver.get("https://www.saucedemo.com/");
7          driver.manage().window().maximize();
8      }
9      @Test(dataProvider = "sauceLabData")
10     public void login(String username, String password) {
11         driver.findElement(By.id("user-name")).sendKeys(username);
12         driver.findElement(By.name("password")).sendKeys(password);
13         driver.findElement(By.className("submit-button")).click();
14         Thread.sleep(3000);
15         String actual = null;
16         try {
17             actual = driver.findElement(By.xpath("//div[@id='product-grid']//div[@class='product-name']")).getText();
18         } catch (Exception e) {
19             System.out.println(e.getMessage());
20         }
21         Assert.assertEquals(actual, "Products", "Login failed")
22     }
23 }

```

```

1  @AfterMethod
2  public void tearDown() {
3      driver.quit();
4  }
5
6  @DataProvider(name = "sauceLabData")
7  public Object[][] passData() {
8      Object[][] data = new Object[3][2];
9      //1st Set
10     data[0][0] = "standard_user";
11     data[0][1] = "secret_sauce";
12     //2nd Set
13     data[1][0] = "problem_user";
14     data[1][1] = "secret_sauce";
15     //3rd Set
16     data[2][0] = "admin2";
17     data[2][1] = "demo1234";
18     return data;
19 }

```

Assertion in TestNG

- TestNG provides a set of assertion methods to validate the actual and expected results.

Why is Assertion?

- **Verification & Validation:** Ensures that the code behaves as expected and meets the requirements.

Code Example: TestNG Assertion

```
1  class AssertionTest {
2      @Test
3      public void testMethod() {
4          int actual = 5;
5          int expected = 5;
6          Object obj1 = null;
7          Object obj2 = new Object();
8          Object obj3 = obj2;
9
10         Assert.assertEquals(actual, expected, "Actual and expected values are not equal");
11         Assert.assertNotEquals(actual, 6, "Actual value is equal to 6");
12         Assert.assertTrue(actual == 5, "Actual value is not equal to 5");
13         Assert.assertFalse(actual == 6, "Actual value is equal to 6");
14         Assert.assertNull(obj1, "Object is not null");
15         Assert.assertNotNull(obj2, "Object is null");
16         Assert.assertSame(obj2, obj3, "Objects do not point to the same reference");
17         Assert.assertNotSame(obj1, obj2, "Objects point to the same reference");
18     }
19 }
```

Running the Tests using testng.xml

- TestNG allows you to run multiple test classes using a single testng.xml file.

Documentations: https://testng.org/#_testng_xml

```
1  <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
2  <suite name="Suite1">
3      <test name="Test1">
4          <classes>
5              <class name="TestNGAnnotations" />
6          </classes>
7      </test>
8  </suite>
```


Apache POI

Apache POI is a popular API that allows to create, modify, and display MS Office files using Java programs. It is an open-source library developed and distributed by Apache Software Foundation to design or modify Microsoft Office files using Java program.

Why Apache POI?

- Read, write and modify Excel files

Apache POI: Dependency

- POI and POI-OOXML are the two main dependencies for Apache POI.

Maven Repository: <https://mvnrepository.com/search?q=poi>

Add the following dependencies in your `pom.xml` file:

```
1  <dependency>
2      <groupId>org.apache.poi</groupId>
3      <artifactId>poi</artifactId>
4      <version>5.3.0</version>
5  </dependency>
6
7  <dependency>
8      <groupId>org.apache.poi</groupId>
9      <artifactId>poi-ooxml</artifactId>
10     <version>5.3.0</version>
11 </dependency>
```

Code Example: Read Excel File


```
1  class ReadExcelFile {
2      public static void main(String[] args) throws IOException {
3          File src = new File("SauceLabCredentials.xlsx");
4
5          FileInputStream fileInputStream = new FileInputStream(src);
6          XSSFWorkbook workbook = new XSSFWorkbook(fileInputStream);
7
8          Sheet sheet = workbook.getSheet("Sheet1");
9
10         String data0 = sheet.getRow(1).getCell(0).getStringCellValue();
11         System.out.println("Data from Excel is " + data0);
12
13         String data1 = sheet.getRow(2).getCell(0).getStringCellValue();
14         System.out.println("Data from Excel is " + data1);
15
16         workbook.close();
17         fileInputStream.close();
18     }
19 }
```

Code Example: Load Data from Excel

```
1 public class DataDrivenTest {
2     WebDriver driver;
3     @BeforeMethod
4     public void setup() {
5         driver = new ChromeDriver();
6         driver.get("https://www.saucedemo.com/");
7         driver.manage().window().maximize();
8     }
9     @Test(dataProvider = "sauceLabData")
10    public void login(String username, String password) {
11        driver.findElement(By.id("user-name")).sendKeys(username);
12        driver.findElement(By.name("password")).sendKeys(password);
13        driver.findElement(By.className("submit-button")).click();
14        Thread.sleep(3000);
15        String actual = null;
16        try {
17            actual = driver.findElement(By.xpath("//h1")).getText();
18        } catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21        Assert.assertEquals(actual, "Products", "Login failed");
22    }
23 }
```

```
1 @AfterMethod
2 public void tearDown() {
3     driver.quit();
4 }
5 @DataProvider(name = "sauceLabData")
6 public Object[][] passData() throws IOException {
7     Object[][] data = new Object[0][0];
8     try (FileInputStream inputStream = new FileInputStream("data.xlsx")) {
9         Workbook workbook = new XSSFWorkbook(inputStream);
10        Sheet sheet = workbook.getSheet("Sheet1");
11
12        int rowCount = sheet.getLastRowNum() - sheet.getFirstRowNum() + 1;
13        data = new Object[rowCount][2];
14
15        for (int i = 1; i <= rowCount; i++) {
16            Row row = sheet.getRow(i);
17            for (int j = 0; j < 2; j++) {
18                data[i - 1][j] = row.getCell(j).getStringCellValue();
19            }
20        }
21    } catch (IOException e) {
22        System.out.println(e.getMessage());
23    }
24    return data;
25 }
```

Thank you 

 [qa-june-2024-automation-with-java-slides](#)