

Pruebas basadas en datos con TestNG y Excel

Las pruebas basadas en datos con TestNG son un enfoque potente que permite ejecutar el mismo caso de prueba con múltiples conjuntos de datos. Al usar fuentes de datos externas como archivos Excel o CSV, puede administrar y mantener fácilmente sus datos de prueba, lo que aumenta la eficiencia y la fiabilidad de su proceso de prueba.

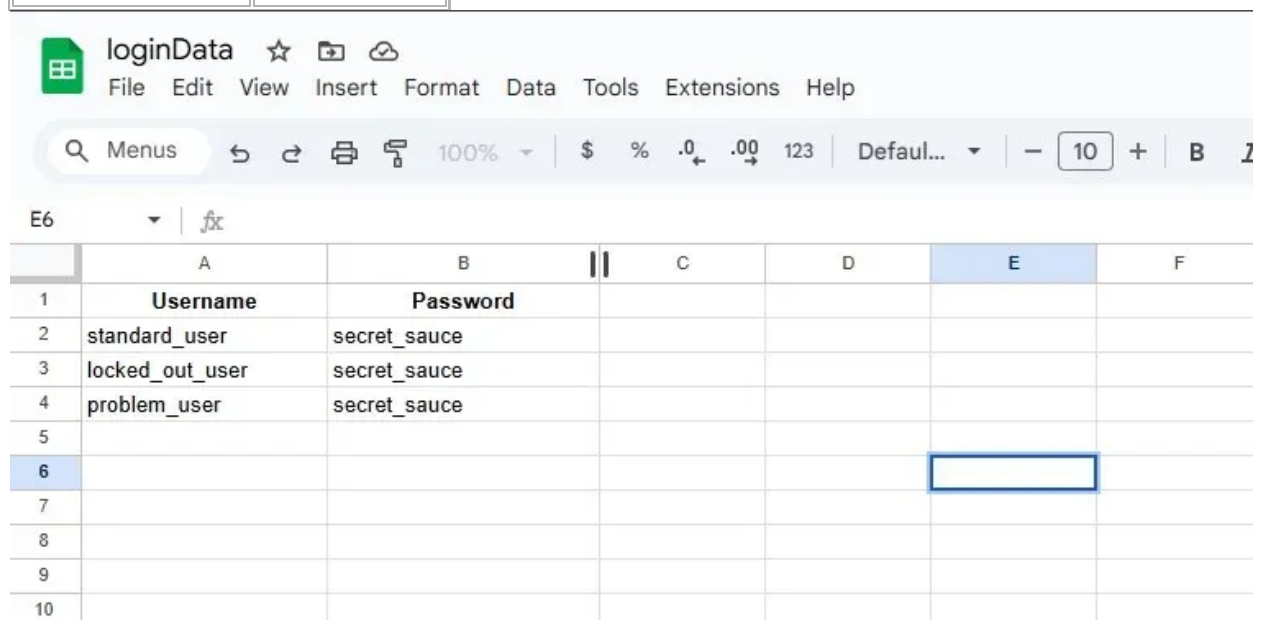
Estos son los pasos de las pruebas basadas en datos que utilizan DataProvider de TestNG en combinación con archivos Excel para proporcionar detalles de inicio de sesión a un caso de prueba de Selenium.

Paso 1: Prepare su archivo de Excel

Cree un archivo de Excel que contenga varios conjuntos de credenciales de inicio de sesión. Cada fila representará un nuevo conjunto de datos.

loginData.xlsx

Username	Password
standard_user	secret_sauce
locked_out_user	secret_sauce
problem_user	secret_sauce



The screenshot shows an Excel application window titled 'loginData'. The ribbon includes File, Edit, View, Insert, Format, Data, Tools, Extensions, and Help. The formula bar shows 'E6'. The spreadsheet has columns A through F and rows 1 through 10. The data is as follows:

	A	B	C	D	E	F
1	Username	Password				
2	standard_user	secret_sauce				
3	locked_out_user	secret_sauce				
4	problem_user	secret_sauce				
5						
6						
7						
8						
9						
10						

Paso 2: Agregar las dependencias requeridas

Asegúrese de tener las dependencias necesarias en su pom.xml si está utilizando Maven:

```
<dependencies>
```

```
<dependency>
```

```

    <groupId>org.apache.poi</groupId>

    <artifactId>poi-ooxml</artifactId>

    <version>5.2.3</version>
</dependency>

<dependency>

    <groupId>org.testng</groupId>

    <artifactId>testng</artifactId>

    <version>7.4.0</version>

    <scope>test</scope>
</dependency>

<dependency>

    <groupId>org.seleniumhq.selenium</groupId>

    <artifactId>selenium-java</artifactId>

    <version>4.0.0</version>
</dependency>
</dependencies>

```

Paso 3: Crear una utilidad para leer datos de Excel.

Para leer los datos del archivo de Excel, crearemos un método de utilidad con Apache POI. Este método leerá el archivo de Excel y devolverá los datos en una `Object[][]` matriz 2D, requerida por el `DataProvider` de TestNG.

ExcelUtils.java

```

package io.learn.datadriven;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class ExcelUtils {

```

```

    public static Object[][] getExcelData(String filePath, String sheetName) throws
IOException, InvalidFormatException {

        FileInputStream file = new FileInputStream(new File(filePath));

        Workbook workbook = WorkbookFactory.create(file);

        Sheet sheet = workbook.getSheet(sheetName);

        int rowCount = sheet.getPhysicalNumberOfRows();

        int colCount = sheet.getRow(0).getPhysicalNumberOfCells();

        Object[][] data = new Object[rowCount - 1][colCount]; // To ignore the header row

        for (int i = 1; i < rowCount; i++) { // Start from 1 to skip the header row
            for (int j = 0; j < colCount; j++) {
                data[i - 1][j] = sheet.getRow(i).getCell(j).toString();
            }
        }

        workbook.close();

        return data;
    }
}

```

Cerciorarse:

- **getExcelData()** : este método lee datos del archivo Excel y los almacena en una matriz 2D (Object[][]).
- **Ruta del archivo** : debe proporcionar la ruta del archivo Excel que desea leer.
- **Nombre de la hoja** : especifique el nombre de la hoja de la que desea extraer datos.

Paso 4: Crear el método DataProvider

Tenemos una utilidad para leer datos de Excel. Podemos usar este método en nuestro DataProvider. Actualice nuestra DataProviderExampleclase para leer datos del archivo de Excel y suministrarlos a la prueba.

Ejemplo de proveedor de datos.java

```
package io.learn.datadriven;
```

```

package io.learn.datadriven;

import org.testng.annotations.DataProvider;
import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import java.io.IOException;

public class DataProviderExample {

    @DataProvider(name = "loginData")

    public Object[][] provideLoginData() throws IOException, InvalidFormatException {

        // Provide the path to the Excel file and the sheet name

        return ExcelUtils.getExcelData("change the path of excel file in your local
storage\\loginData.xlsx", "Sheet_Name which we give at the bottom in Excel File");

    }

}

```

El método DataProvider loginData se actualiza para llamar al ExcelUtils.getExcelData() método para obtener los datos de inicio de sesión del loginData.xlsx archivo.

Paso 5: Crear la clase de prueba de Selenium

En el siguiente paso, cree la clase de prueba de Selenium que usa loginDataDataProvider para la prueba de inicio de sesión. Realizaremos el inicio de sesión con cada conjunto de nombre de usuario y contraseña proporcionados por el archivo de Excel.

Prueba de inicio de sesión.java

```

package io.learn.datadriven;

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 LoginTest {

    WebDriver driver;

    @BeforeMethod
    public void setUp() {
        // Initialize the WebDriver (ensure chromedriver path is correct)
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\change the path
of chromedriver path\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.get("https://www.saucedemo.com/"); // URL of the login page
    }

    @Test(dataProvider = "loginData", dataProviderClass = DataProviderExample.class)
    public void testLogin(String username, String password) {
        // Find elements for username, password, and login button
        WebElement usernameField = driver.findElement(By.id("user-name"));
        WebElement passwordField = driver.findElement(By.id("password"));
        WebElement loginButton = driver.findElement(By.id("login-button"));

        // Clear any existing text from fields
        usernameField.clear();
        passwordField.clear();

        // Input login credentials from DataProvider
        usernameField.sendKeys(username);
        passwordField.sendKeys(password);
        loginButton.click();

        System.out.println("The DataDriven test executed successfully");
    }
}

```

```

// Validate the login behavior by checking the title of the page

Assert.assertTrue(driver.getTitle().contains("Swag Labs"));

}

```

@AfterMethod

```

public void tearDown() {

    driver.quit(); // Close the browser after each test

}
}

```

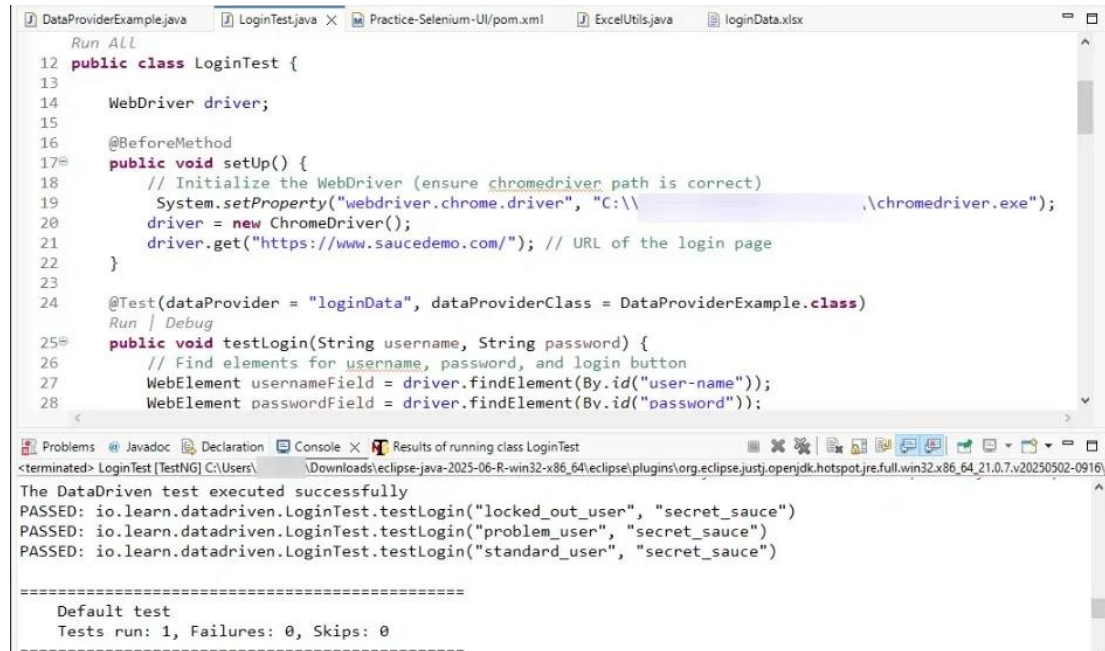
Paso 5: Ejecución de las pruebas

Para ejecutar las pruebas, ejecute el siguiente comando en la terminal o utilice el ejecutor de pruebas de su IDE:

mvn test

Si está utilizando su IDE, puede ejecutar la prueba directamente.

Producción:



The screenshot shows an IDE with the following components:

- Editors:**
 - `DataProviderExample.java`
 - `LoginTest.java` (active)
 - `Practice-Selenium-UI/pom.xml`
 - `ExcelUtils.java`
 - `LoginData.xlsx`
- Code in LoginTest.java:**

```

12 public class LoginTest {
13
14     WebDriver driver;
15
16     @BeforeMethod
17     public void setUp() {
18         // Initialize the WebDriver (ensure chromedriver path is correct)
19         System.setProperty("webdriver.chrome.driver", "C:\\...\\chromedriver.exe");
20         driver = new ChromeDriver();
21         driver.get("https://www.saucedemo.com/"); // URL of the login page
22     }
23
24     @Test(dataProvider = "loginData", dataProviderClass = DataProviderExample.class)
25     public void testLogin(String username, String password) {
26         // Find elements for username, password, and login button
27         WebElement usernameField = driver.findElement(By.id("user-name"));
28         WebElement passwordField = driver.findElement(By.id("password"));

```
- Console Output:**

```

<terminated> LoginTest [TestNG] C:\Users\...
The DataDriven test executed successfully
PASSED: io.learn.datadriven.LoginTest.testLogin("locked_out_user", "secret_sauce")
PASSED: io.learn.datadriven.LoginTest.testLogin("problem_user", "secret_sauce")
PASSED: io.learn.datadriven.LoginTest.testLogin("standard_user", "secret_sauce")

=====
Default test
Tests run: 1, Failures: 0, Skips: 0
=====

```

DataProvider en TestNG con datos de una salida de archivo de Excel

Al usar DataProvider en TestNG con datos de un archivo Excel, puede ejecutar pruebas de manera eficiente con múltiples conjuntos de datos sin codificar los datos de prueba.