# LoginTest Class - Step-by-Step Documentation

## 📂 Class Overview

```java

public class LoginTest extends BaseTest {

```

- `LoginTest` is the main test class for login functionality.

- It inherits from `BaseTest`, which handles WebDriver setup and teardown before and after each test.

## ✅ Test Method with DataProvider

```java

@Test(dataProvider = "loginCredentials")

public void testLogin(String username, String password) {

```

- Annotated with `@Test` and linked to a `@DataProvider` named `loginCredentials`.

- The method runs once for each set of credentials supplied by the data provider.

## 🧱 Page Object Usage

```java

LoginPage loginPage = new LoginPage(driver);

loginPage.login(username, password);

```

- Instantiates the `LoginPage` POM class with the current WebDriver instance.

- Calls the `login()` method to perform login actions (enter username, password, click login).

## ✅ Assertion Check

```java

String expectedUrl = "https://example.com/dashboard";

Assert.assertEquals(driver.getCurrentUrl(), expectedUrl, "Login failed for user: " + username);

```

- Verifies if login was successful by checking the URL after login.

- If the URL does not match the expected dashboard URL, the test fails and logs the failure with the username used.

## 📊 Data Provider Method

```java

@DataProvider(name = "loginCredentials")

public Object[][] getLoginData() {

return new Object[][] {

{"admin", "admin123"},

{"testuser1", "testpass1"},

{"testuser2", "wrongpass"}

};

}

```

- Supplies test data as a two-dimensional `Object[][]` array.

- This results in three iterations of the `testLogin` method:

1. `admin / admin123`

2. `testuser1 / testpass1`

3. `testuser2 / wrongpass`

## 🔁 Execution Summary

| Iteration | Username | Password | Expected Outcome |

|-----------|-------------|------------|-----------------------------|

| 1 | admin | admin123 | ✅ Success - Correct URL |

| 2 | testuser1 | testpass1 | ✅/❌ Depends on credentials |

| 3 | testuser2 | wrongpass | ❌ Failure - Invalid login |

## 🔚 Summary

This setup allows for robust, repeatable testing of the login functionality using different sets of data.

It promotes modularity and reusability by using:

- \*\*Page Object Model (POM)\*\* for UI actions

- \*\*TestNG DataProviders\*\* for data-driven testing

- \*\*Assertions\*\* for validation

## 📌 Optional Enhancements

- Integrate \*\*Excel-based data\*\* using Apache POI for external test data.

- Add \*\*reporting tools\*\* like ExtentReports for detailed test execution logs.

- Include \*\*additional assertions\*\* to verify login success (like welcome text or logout button).

# LoginTest Class - Step-by-Step Documentation

This document explains the functionality and flow of the LoginTest class used in a TestNG framework with a Page Object Model (POM) approach for data-driven testing.

## 📂 Class Overview

public class LoginTest extends BaseTest {

* LoginTest is the main test class for login functionality.
* It inherits from BaseTest, which handles WebDriver setup and teardown before and after each test.

## ✅ Test Method with DataProvider

@Test(dataProvider = "loginCredentials")

public void testLogin(String username, String password) {

* Annotated with @Test and linked to a @DataProvider named loginCredentials.
* The method runs once for each set of credentials supplied by the data provider.

## 🧱 Page Object Usage

LoginPage loginPage = new LoginPage(driver);

loginPage.login(username, password);

* Instantiates the LoginPage POM class with the current WebDriver instance.
* Calls the login() method to perform login actions (enter username, password, click login).

## ✅ Assertion Check

String expectedUrl = "https://example.com/dashboard";

Assert.assertEquals(driver.getCurrentUrl(), expectedUrl, "Login failed for user: " + username);

* Verifies if login was successful by checking the URL after login.
* If the URL does not match the expected dashboard URL, the test fails and logs the failure with the username used.

## 📊 Data Provider Method

@DataProvider(name = "loginCredentials")

public Object[][] getLoginData() {

return new Object[][] {

{"admin", "admin123"},

{"testuser1", "testpass1"},

{"testuser2", "wrongpass"}

};

}

* Supplies test data as a two-dimensional Object[][] array.
* This results in three iterations of the testLogin method:
  1. admin / admin123
  2. testuser1 / testpass1
  3. testuser2 / wrongpass

## 🔁 Execution Summary

| **Iteration** | **Username** | **Password** | **Expected Outcome** |
| --- | --- | --- | --- |
| 1 | admin | admin123 | ✅ Success - Correct URL |
| 2 | testuser1 | testpass1 | ✅/❌ Depends on credentials |
| 3 | testuser2 | wrongpass | ❌ Failure - Invalid login |

## 🔚 Summary

This setup allows for robust, repeatable testing of the login functionality using different sets of data.  
It promotes modularity and reusability by using:

* **Page Object Model (POM)** for UI actions
* **TestNG DataProviders** for data-driven testing
* **Assertions** for validation

## 📌 Optional Enhancements

* Integrate **Excel-based data** using Apache POI for external test data.
* Add **reporting tools** like ExtentReports for detailed test execution logs.
* Include **additional assertions** to verify login success (like welcome text or logout button).