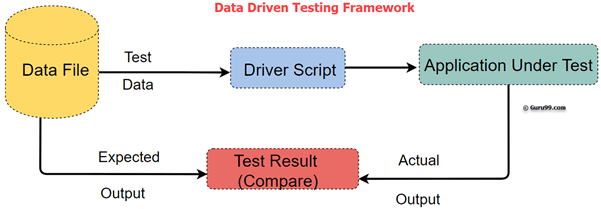
In Data-driven test automation framework, input data can be stored in single or multiple data sources like xls, XML, csv, and databases.



**Why Data Driven Testing?**

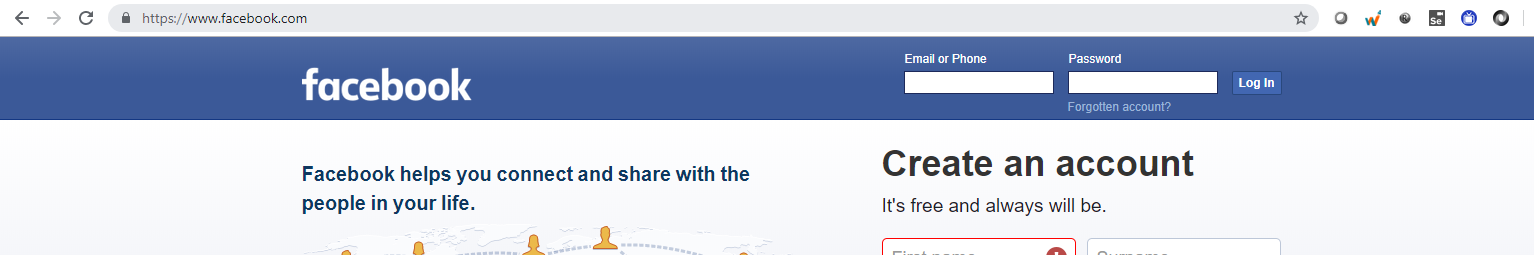
Data Driven Testing framework resolves this problem by keeping the data separate from Functional tests. The same test script can execute for different combinations of input test data and generate test results.

**Example:**

For example, we want to test the login system with multiple input fields with 1000 different data sets.

To test this, you can take following different approaches:

**How to create a Data Driven Automation Framework**



Consider you want to Test Login functionality of an application.

**Step 1)** Identify the Test Cases

* Input Correct username and password – Login Success
* Input incorrect username and correct password – Login Failure
* Input correct username and incorrect password - Login Failure

**Step 2)** Create detailed test Steps for above 3 Test Cases

**Step 3)** Create Test Script

// This is Pseudo Code

// Test Step 1: Launch Application

driver.get("URL of the Appliation");

// Test Step 2: Enter Password

txtbox\_username.sendKeys("valid");

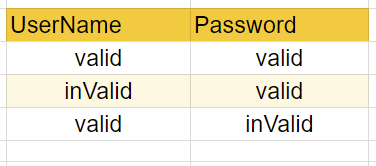
// Test Step 3: Enter Password

txtbox\_password.sendKeys("invalid");

// Test Step 4: Check Results

If (Next Screen) print success else Fail

**Step 4)** Create an excel/csv with the Input Test Data

[](https://www.guru99.com/images/1/032318_1019_WhatisDataD3.png)

**Step 5)** Step Modify the Script to Loop over Input Test Data. The input commands should also be parameterized

// This is Pseudo Code

// Loop 3 Times

for (i = 0; i & lt; = 3; i++) {

// Read data from Excel and store into variables

int input\_1 = ReadExcel(i, 0);

int input\_2 = ReadExcel(i, 1);

// Test Step 1: Launch Application

driver.get("URL of the Application");

// Test Step 2: Enter Password

txtbox\_username.sendKeys(input\_1);

// Test Step 3: Enter Password

txtbox\_password.sendKeys(input\_2);

// Test Step 4: Check Results

If(Next Screen) print success

else Fail

}

Above are just 3 test cases. The test script can be used to loop over following test cases just by appending test data values to Excel

* Input incorrect username and incorrect password – Login Fail
* Input correct username and password blank – Login Fail
* Input blank username and blank password– Login Fail

And so on

**Best practices of Data Driven testing:**

Below given are Best testing practices for Data-Driven testing:

* It is ideal to use realistic information during the data-driven testing process
* Test flow navigation should be coded inside the test script
* Use Data to Drive Dynamic Assertions
* Test positive as well as negative outcomes

**Advantages of Data-Driven testing**

Data-Driven offer many advantages some of them are:

1. Allows to test application with multiple sets of data values during Regression testing
2. Test data and verification data can be organized in just one file, and it is separate from the test case logic.
3. Base on the tool, it is possible to have the test scripts in a single repository. This makes the texts easy to understand, maintain and manage.
4. Actions and Functions can be reused in different tests.
5. Some tools generate test data automatically. This is useful when large volumes of random test data are necessary, which helps to save the time.
6. Data-driven testing can perform any phase of the development. A data-driven test cares are generally merged in the single process. However, it can be used in multiple test cases.
7. Allows developers and testers to have clear separation for the logic of their test cases/scripts from the test data.
8. The same test cases can be executed several times which helps to reduce test case and scripts.
9. Any changes in the test script do not affect the test data

**Disadvantages of Data Driven testing:**

Some Drawbacks of Data Driven Automation Testing method are:

1. Quality of the test is depended on the automation skills of the Implementing team
2. Data validation is a time-consuming task when testing large amount of data.
3. Maintenance is a big issue as large amount of coding needed for Data-Driven testing.
4. High-level technical skills are required. A tester may have to learn an entirely new scripting language.
5. There will be more documentation. Mostly related to scripts management tests infrastructure and testing results.
6. A text editor like Notepad is required to create and maintain data files.

### Interface In POI

One of the most remarkable features of **Apache POI** is that it supports read and write operations on both .xls and .xslx files.

Below mentioned are some of the **interfaces of POI**.

* **XSSFWorkbook:** Represents workbook in xlsx file.
* **HSSFWorkbook:** Represents workbook in xls file.
* **XSSFSheet:** Represents a sheet in XLSX file.
* **HSSFSheet:** Represents a sheet in XLS file.
* **XSSFRow:** Represents a row in a sheet of XLSX file.
* **HSSFRow:** Represents a row in a sheet of XLS file.
* **XSSFCell:** Represents a cell in a row of XLSX file.
* **HSSFCell:** Represents a cell in a row of XLS file.

**Fields available in a cell:**

* **CELL\_TYPE\_BLANK:** Represents a blank cell.
* **CELL\_TYPE\_BOOLEAN:** Represents a Boolean cell (true or false).
* **CELL\_TYPE\_ERROR:** Represents an error value in a cell.
* **CELL\_TYPE\_FORMULA:** Represents a formula result on a cell.
* **CELL\_TYPE\_NUMERIC:** Represents numeric data in a cell.
* **CELL\_TYPE\_STRING:** Represents string in a cell.