**Keyword-Driven Testing: Overview**

Keyword-Driven Testing (KDT) is a methodology in automated testing where test scripts are created using keywords representing actions or operations. These keywords are mapped to functions in the test framework, making it easier for non-programmers to contribute to test creation.

**How It Works**

1. **Define Keywords**:
   * Keywords are human-readable terms representing specific actions (e.g., "Login", "ClickButton", "VerifyText").
   * Each keyword corresponds to a function or method in the code.
2. **Prepare a Test Data File**:
   * Use a structured file format like Excel, CSV, or a database.
   * Each row typically represents a test step, with columns for:
     + **Keyword**: The action to perform (e.g., "EnterText").
     + **Locator**: The UI element to interact with (e.g., XPath or CSS selector).
     + **Input Data**: Values to input (e.g., username or password).
     + **Expected Result**: What to verify after the step.
3. **Develop a Test Framework**:
   * Implement a library of reusable functions (mapped to keywords) in the chosen programming language (e.g., Python, Java, or JavaScript).
   * A driver script interprets the test data and executes corresponding functions.
4. **Execute Tests**:
   * The framework reads the test data file, performs actions defined by the keywords, and logs results.

**Advantages**

* **Low Programming Requirement**: Testers focus on creating test data files, reducing the need for programming skills.
* **Reusability**: Functions tied to keywords can be reused across multiple tests.
* **Flexibility**: Supports modular and scalable test design.

**Implementation in Python**

Here’s an example of a Keyword-Driven Testing framework in Python:

**Keyword Mapping**

def open\_browser(url):

driver.get(url)

def enter\_text(locator, value):

element = driver.find\_element\_by\_xpath(locator)

element.send\_keys(value)

def click\_button(locator):

driver.find\_element\_by\_xpath(locator).click()

def verify\_text(locator, expected):

actual = driver.find\_element\_by\_xpath(locator).text

assert actual == expected, f"Expected {expected}, got {actual}"

keyword\_map = {

"OpenBrowser": open\_browser,

"EnterText": enter\_text,

"ClickButton": click\_button,

"VerifyText": verify\_text,

}

**Test Data (e.g., Excel or CSV)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Keyword** | **Locator** | **Input Data** | **Expected Result** |
| OpenBrowser |  | [https://example.com](https://example.com/) |  |
| EnterText | //input[@name='user'] | test\_user |  |
| ClickButton | //button[@id='login'] |  |  |
| VerifyText | //div[@id='welcome'] |  | Welcome, User! |

**Driver Script**

import csv

# Example: Reading test data from CSV

with open("test\_data.csv", "r") as file:

reader = csv.DictReader(file)

for row in reader:

keyword = row["Keyword"]

locator = row["Locator"]

input\_data = row["Input Data"]

expected\_result = row["Expected Result"]

# Execute corresponding function

if keyword in keyword\_map:

if expected\_result:

keyword\_map[keyword](locator, input\_data, expected\_result)

else:

keyword\_map[keyword](locator, input\_data)

**Python Example**

**Test Framework**

python

Copy code

from selenium import webdriver

# Initialize WebDriver

driver = webdriver.Chrome()

# Keyword Function Definitions

def open\_browser(url):

driver.get(url)

def enter\_text(locator, value):

driver.find\_element\_by\_xpath(locator).send\_keys(value)

def click\_button(locator):

driver.find\_element\_by\_xpath(locator).click()

def verify\_text(locator, expected):

actual = driver.find\_element\_by\_xpath(locator).text

assert actual == expected, f"Expected {expected}, but got {actual}"

# Keyword Map

keyword\_map = {

"OpenBrowser": open\_browser,

"EnterText": enter\_text,

"ClickButton": click\_button,

"VerifyText": verify\_text,

}

**Test Data (CSV Example)**

csv

Copy code

Keyword,Locator,Input Data,Expected Result

OpenBrowser,,,https://example.com

EnterText,//input[@id='user'],test\_user,

ClickButton,//button[@id='login'],,

VerifyText,//h1[@id='welcome'],,Welcome, User!

**Driver Script**

python

Copy code

import csv

# Read Test Data

with open('test\_data.csv', 'r') as file:

reader = csv.DictReader(file)

for row in reader:

keyword = row['Keyword']

locator = row['Locator']

input\_data = row['Input Data']

expected\_result = row['Expected Result']

# Execute the corresponding function

if keyword in keyword\_map:

if expected\_result:

keyword\_map[keyword](locator, expected\_result)

elif locator and input\_data:

keyword\_map[keyword](locator, input\_data)

elif input\_data:

keyword\_map[keyword](input\_data)

else:

keyword\_map[keyword]()

**Implementation in Java**

Using frameworks like Selenium and Apache POI for Excel data handling:

**Keyword Mapping**

public void openBrowser(String url) {

driver.get(url);

}

public void enterText(String locator, String value) {

driver.findElement(By.xpath(locator)).sendKeys(value);

}

public void clickButton(String locator) {

driver.findElement(By.xpath(locator)).click();

}

public void verifyText(String locator, String expected) {

String actual = driver.findElement(By.xpath(locator)).getText();

Assert.assertEquals(actual, expected, "Verification Failed!");

}

**Driver Code**

// Example: Reading test data from Excel (using Apache POI)

FileInputStream fis = new FileInputStream("test\_data.xlsx");

Workbook workbook = new XSSFWorkbook(fis);

Sheet sheet = workbook.getSheetAt(0);

for (Row row : sheet) {

String keyword = row.getCell(0).getStringCellValue();

String locator = row.getCell(1).getStringCellValue();

String inputData = row.getCell(2).getStringCellValue();

String expected = row.getCell(3).getStringCellValue();

switch (keyword) {

case "OpenBrowser":

openBrowser(inputData);

break;

case "EnterText":

enterText(locator, inputData);

break;

case "ClickButton":

clickButton(locator);

break;

case "VerifyText":

verifyText(locator, expected);

break;

}

}

**Java Example**

**Test Framework**

java

Copy code

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class KeywordFramework {

static WebDriver driver;

public static void openBrowser(String url) {

driver = new ChromeDriver();

driver.get(url);

}

public static void enterText(String locator, String value) {

WebElement element = driver.findElement(By.xpath(locator));

element.sendKeys(value);

}

public static void clickButton(String locator) {

driver.findElement(By.xpath(locator)).click();

}

public static void verifyText(String locator, String expected) {

String actual = driver.findElement(By.xpath(locator)).getText();

if (!actual.equals(expected)) {

throw new AssertionError("Expected " + expected + " but found " + actual);

}

}

}

**Test Data (Excel Example)**

| **Keyword** | **Locator** | **Input Data** | **Expected Result** |
| --- | --- | --- | --- |
| OpenBrowser |  | <https://example.com> |  |
| EnterText | //input[@id='user'] | test\_user |  |
| ClickButton | //button[@id='login'] |  |  |
| VerifyText | //h1[@id='welcome'] |  | Welcome, User! |

**Driver Script**

java

Copy code

import org.apache.poi.ss.usermodel.\*;

import java.io.FileInputStream;

public class TestExecutor {

public static void main(String[] args) throws Exception {

FileInputStream fis = new FileInputStream("test\_data.xlsx");

Workbook workbook = WorkbookFactory.create(fis);

Sheet sheet = workbook.getSheetAt(0);

for (Row row : sheet) {

String keyword = row.getCell(0).getStringCellValue();

String locator = row.getCell(1) != null ? row.getCell(1).getStringCellValue() : "";

String inputData = row.getCell(2) != null ? row.getCell(2).getStringCellValue() : "";

String expected = row.getCell(3) != null ? row.getCell(3).getStringCellValue() : "";

switch (keyword) {

case "OpenBrowser":

KeywordFramework.openBrowser(inputData);

break;

case "EnterText":

KeywordFramework.enterText(locator, inputData);

break;

case "ClickButton":

KeywordFramework.clickButton(locator);

break;

case "VerifyText":

KeywordFramework.verifyText(locator, expected);

break;

}

}

}

}

**Implementation in JavaScript (Cypress Example)**

Cypress doesn’t natively support KDT but can mimic it with modular functions:

**Keyword Mapping**

const keywords = {

OpenBrowser: (url) => cy.visit(url),

EnterText: (locator, value) => cy.get(locator).type(value),

ClickButton: (locator) => cy.get(locator).click(),

VerifyText: (locator, expected) => cy.get(locator).should('have.text', expected),

};

**Driver Code**

Cypress.Commands.add("executeKeyword", (keyword, locator, inputData, expected) => {

if (keywords[keyword]) {

if (expected) {

keywords[keyword](locator, inputData, expected);

} else {

keywords[keyword](locator, inputData);

}

}

});

// Example: Define test case

cy.executeKeyword("OpenBrowser", null, "https://example.com");

cy.executeKeyword("EnterText", "input[name='user']", "test\_user");

cy.executeKeyword("ClickButton", "button#login");

cy.executeKeyword("VerifyText", "div#welcome", "Welcome, User!");

**JavaScript Example (Cypress)**

**Keyword Map**

javascript

Copy code

const keywords = {

OpenBrowser: (url) => cy.visit(url),

EnterText: (locator, value) => cy.get(locator).type(value),

ClickButton: (locator) => cy.get(locator).click(),

VerifyText: (locator, expected) => cy.get(locator).should('have.text', expected),

};

**Test Case (JSON Example)**

json

Copy code

[

{ "Keyword": "OpenBrowser", "Locator": "", "InputData": "https://example.com" },

{ "Keyword": "EnterText", "Locator": "input#user", "InputData": "test\_user" },

{ "Keyword": "ClickButton", "Locator": "button#login", "InputData": "" },

{ "Keyword": "VerifyText", "Locator": "h1#welcome", "InputData": "", "ExpectedResult": "Welcome, User!" }

]

**Driver Script**

javascript

Copy code

Cypress.Commands.add("executeKeyword", (keyword, locator, inputData, expected) => {

if (keywords[keyword]) {

if (expected) {

keywords[keyword](locator, expected);

} else if (inputData) {

keywords[keyword](locator, inputData);

} else {

keywords[keyword](locator);

}

}

});

// Test Execution

cy.fixture("test\_case.json").then((testSteps) => {

testSteps.forEach((step) => {

cy.executeKeyword(step.Keyword, step.Locator, step.InputData, step.ExpectedResult);

});

});