

Keyword Driven Framework In Uft With Complete Source Code

Keyword Driven Framework in UFT with Complete Source Code

Automating tests is crucial for efficient software development, and UFT (Unified Functional Testing) offers robust capabilities for this. A particularly powerful approach is using a **keyword-driven framework in UFT**, which significantly improves test maintainability, reusability, and collaboration. This article dives deep into building such a framework, providing complete source code examples and exploring its advantages. We will also cover topics like **UFT test**

automation, data-driven testing in UFT, and the overall implementation of a **keyword driven test framework**.

Understanding the Keyword Driven Framework in UFT

A keyword-driven framework, also known as a table-driven framework or action word framework, separates test script logic from test data. Test steps are represented by keywords, which are then mapped to actions within UFT. This decoupling allows business users, with limited coding experience, to design tests by selecting keywords and inputting data, while developers focus on creating robust, reusable functions. This approach dramatically reduces the time and effort needed for test creation and maintenance. For example, instead of writing complex VBScript code for each login attempt, a keyword like "Login" would execute the necessary actions.

Benefits of a Keyword Driven Framework in UFT

Adopting a keyword-driven testing approach in UFT offers several significant advantages:

- **Increased Reusability:** Keywords represent reusable components, significantly reducing redundant code. A single "ClickButton" keyword, for instance, can be used across multiple tests and applications.
- **Enhanced Maintainability:** Changes to the application under test require modifications only in the keyword's implementation, not across multiple test scripts.
- **Improved Collaboration:** Business analysts and testers can design tests using keywords without deep programming knowledge, fostering better collaboration between development and testing teams.
- **Reduced Development Time:** The reusable nature of keywords significantly accelerates test development.
- **Easier Test Data Management:** Test data is separated from the script logic, making it easier to manage and update. This naturally integrates

with **data-driven testing in UFT**, enhancing the framework's flexibility.

Implementing a Keyword Driven Framework in UFT: Step-by-Step Guide

Let's illustrate the implementation with a simple login scenario. We will utilize Excel sheets to store our test data and keywords.

1. Keyword Repository (Excel Sheet):

Create an Excel sheet with columns for "Keyword," "Action," "Object," and "Value."

Keyword	Action	Object	Value
OpenBrowser	OpenBrowser	Browser	Chrome

```
| Navigate | Navigate | URL | https://www.google.com |  
| EnterText | Type | UsernameTextbox | myusername |  
| EnterText | Type | PasswordTextbox | mypassword |  
| ClickButton | Click | LoginButton | |  
| CloseBrowser| CloseBrowser | Browser | |
```

2. UFT Test Script:

```
```vbscript
```

```
Option Explicit
```

```
Function RunKeyword(Keyword, Action, Object, Value)
```

```
On Error Resume Next
```

```
Dim objObject
Set objObject = Description.Create()
objObject("name").Value = Object
Select Case Keyword
Case "OpenBrowser"
Browser("title:=.*").Open Value
Case "Navigate"
Browser("title:=.*").Navigate Value
Case "EnterText"
objObject.Type Value
```

```
Case "ClickButton"
objObject.Click
Case "CloseBrowser"
Browser("title:=.*").Close
Case Else
MsgBox "Keyword not found: " & Keyword
End Select
On Error GoTo 0
End Function
Sub Main
```

```
Dim objExcel, objWorksheet, i, LastRow

Set objExcel = CreateObject("Excel.Application")

objExcel.Visible = True

Set objWorksheet = objExcel.Workbooks.Open("C:\Keywords.xls").Worksheets(1)
'Path to your Excel file

LastRow = objWorksheet.Cells(Rows.Count, 1).End(xlUp).Row

For i = 2 To LastRow 'Start from row 2 to skip header row

RunKeyword objWorksheet.Cells(i, 1).Value, objWorksheet.Cells(i, 2).Value,
objWorksheet.Cells(i, 3).Value, objWorksheet.Cells(i, 4).Value

Next

objExcel.Quit
```



```
Set objExcel = Nothing
```

```
End Sub
```

```
```
```

This script reads the keyword data from the Excel sheet and executes the corresponding actions in UFT. Remember to replace `"C:\Keywords.xls"` with the actual path to your Excel file. This basic example demonstrates the fundamental principles. A more robust implementation would include error handling, logging, and more sophisticated object identification techniques. Furthermore, the usage of external libraries or customized functions can significantly enhance the capabilities of your **keyword driven test framework**.

Extending the Keyword Driven Framework:

Advanced Techniques

This simple framework can be extended considerably. Consider adding features like:

- **Parameterization:** Pass dynamic values to keywords, allowing for greater flexibility in test execution.
- **Reporting:** Integrate with reporting tools to generate detailed test results.
- **Modular Design:** Break down complex keywords into smaller, more manageable sub-keywords.
- **Object Repository:** Utilize UFT's object repository for better object identification and maintenance.
- **Data-Driven Testing Integration:** Seamlessly integrate data from external sources like databases or CSV files.

Conclusion

Implementing a keyword-driven framework in UFT offers a significant upgrade in your test automation strategy. By separating test logic from test data, you achieve better maintainability, reusability, and collaboration. While the initial setup requires planning, the long-term benefits, including reduced development time and increased efficiency, are substantial. The provided code serves as a foundation; further development and customization will tailor it to your specific needs and testing environment. Remember that robust error handling and comprehensive reporting are essential for a production-ready keyword-driven framework. Utilizing this approach effectively leverages the strengths of UFT for robust and efficient test automation.

FAQ

Q1: What are the limitations of a keyword-driven framework?

A1: While keyword-driven frameworks offer many advantages, they also have

limitations. Complex test scenarios might require intricate keyword combinations, increasing the complexity of the keyword repository. Also, debugging can be more challenging compared to script-based approaches as you're working at a higher level of abstraction. Furthermore, initial setup and maintenance of the keyword repository can be time-consuming.

Q2: How does a keyword-driven framework differ from a data-driven framework?

A2: Keyword-driven frameworks separate test logic (keywords) from test data, while data-driven frameworks primarily focus on separating test data from test scripts. A keyword-driven framework uses keywords to represent actions, while a data-driven framework uses data sets to drive the test execution. Often, a hybrid approach is used, combining the strengths of both.

Q3: Can I use external data sources with my keyword-driven framework?

A3: Absolutely! The power of a keyword-driven framework shines when

combined with external data sources. You can easily read test data from Excel files, databases (like SQL Server or Oracle), CSV files, or even APIs. This allows for a more efficient and flexible testing process.

Q4: How do I handle error conditions within my keywords?

A4: Robust error handling is crucial. Incorporate error-checking mechanisms within each keyword function. Use `On Error Resume Next` and `On Error GoTo 0` statements in VBScript to gracefully handle exceptions. Log error messages for detailed debugging and reporting.

Q5: What are some best practices for designing keywords?

A5: Keywords should be concise, descriptive, and easily understandable. Strive for reusability, making them as generic as possible. Avoid overly complex keywords; break down complex actions into smaller, more manageable units. Maintain a consistent naming convention.

Q6: How can I integrate my keyword-driven framework with a CI/CD

pipeline?

A6: Integrate your UFT tests into your CI/CD pipeline using tools like Jenkins or Azure DevOps. This allows for automated test execution as part of the build process, providing continuous feedback and improving the development cycle.

Q7: What are some alternatives to a keyword-driven framework in UFT?

A7: Other approaches include data-driven frameworks, modular frameworks, and hybrid frameworks combining aspects of various approaches. The best choice depends on the project's complexity and specific needs.

Q8: How can I improve the object identification within my keyword-driven framework?

A8: Employ a robust object repository and explore different object identification techniques, such as using descriptive programming, regular expressions, and smart identification. Consider using image-based recognition for scenarios with challenging object identification.

Keyword Driven Framework in UFT with Complete Source Code: Streamlining Your Test Automation

Automating testing procedures is essential in today's rapid software delivery cycle . Among the plethora of automation frameworks available , the Keyword Driven Framework (KDF) sits out for its adaptability and longevity. This article delves into the implementation of a KDF within UFT (Unified Functional Testing), offering a complete source code example and highlighting its perks.

The core notion behind a KDF includes distinguishing test logic from test details. Test actions are represented as keywords, each relating to a particular function within the application below test (AUT). These keywords are maintained in external data sources like databases, permitting automation specialists to easily alter test cases without modifying the subjacent code. This promotes reusability and substantially reduces maintenance effort .

Think of it like a recipe : the keywords are like the components , and the data sheet is the recipe itself. You can easily replace components (data) without re-writing the whole instruction manual (script).

Implementing a Keyword Driven Framework in UFT:

The subsequent sections outline the implementation of a KDF utilizing UFT. We will hone in on a basic example, but the ideas can be extended to manage more elaborate scenarios .

1. Keyword Definition:

We begin by specifying our keywords. Those keywords will represent frequent actions executed during testing, such as logon , navigate to a particular page, enter data into fields , and confirm results . Each keyword will be linked to a corresponding UFT subroutine .

2. Data Sheet:

Next, we construct an outside data sheet (e.g., an Excel spreadsheet) to hold test data. This data sheet will include columns for each keyword and the associated data necessary for that keyword. For example, for the "login" keyword, we might have columns for "username" and "password".

3. UFT Script:

Our UFT code will then read the data from the outside data sheet and run the corresponding UFT functions based on the keywords existing in the data.

Complete Source Code Example:

(Note: This is a simplified example. A real-world implementation would be significantly far thorough .)

```
`` `vbscript
```

```
' UFT Script
```

```
Option Explicit

Sub Main()

Dim objExcel, objWorksheet, strKeyword, strData

Dim iRow, iCol, lastRow

' Create Excel object

Set objExcel = CreateObject("Excel.Application")

Set objWorksheet =
objExcel.Workbooks.Open("C:\testData.xls").Worksheets("Sheet1") 'Change path

' Get last row

lastRow = objWorksheet.Cells(Rows.Count, 1).End(xlUp).Row
```

```
' Loop through each row in the Excel sheet  
  
For iRow = 2 To lastRow 'Start from row 2 to skip header  
  
    strKeyword = objWorksheet.Cells(iRow, 1).Value  
  
    Select Case strKeyword  
  
        Case "Login"  
  
            Call Login(objWorksheet.Cells(iRow, 2).Value, objWorksheet.Cells(iRow,  
3).Value)  
  
        Case "NavigateToPage"  
  
            Call NavigateToPage(objWorksheet.Cells(iRow, 2).Value)  
  
        Case "EnterData"
```

```
Call EnterData(objWorksheet.Cells(iRow, 2).Value, objWorksheet.Cells(iRow, 3).Value)
```

```
Case "VerifyResult"
```

```
Call VerifyResult(objWorksheet.Cells(iRow, 2).Value)
```

```
Case Else
```

```
Reporter.ReportEvent micFail, "Keyword not found:", strKeyword
```

```
End Select
```

```
Next iRow
```

```
' Clean up
```

```
objWorksheet.Close
```

```
objExcel.Quit

Set objWorksheet = Nothing

Set objExcel = Nothing

End Sub

' Function to perform login

Sub Login(strUsername, strPassword)

' UFT code to perform login using strUsername and strPassword

' ...your login code here...

Reporter.ReportEvent micPass, "Login successful", "User: " & strUsername

End Sub
```

' ...other functions for NavigateToPage, EnterData, VerifyResult...

...

testData.xls (Example):

| Keyword | Data1 | Data2 |
|---------|-------|-------|
|---------|-------|-------|

| Keyword | Data1 | Data2 |
|---------|-------|-------|
|---------|-------|-------|

| | | |
|-------|-------|-------|
| Login | user1 | pass1 |
|-------|-------|-------|

| | | |
|----------------|-----------|--|
| NavigateToPage | /homePage | |
|----------------|-----------|--|

| | | |
|-----------|--------------|------|
| EnterData | txtFirstName | John |
|-----------|--------------|------|

| | | |
|--------------|-----------------|--|
| VerifyResult | Success Message | |
|--------------|-----------------|--|

Advantages of Keyword Driven Framework:

- Enhanced reusability of test components .
- Streamlined test maintenance .
- Increased cooperation between testers and product stakeholders .
- Reduced test development time.
- Improved test coverage .

Conclusion:

The Keyword Driven Framework offers a robust and efficient approach to test automation . By separating test reasoning from test data, it significantly better maintainability and minimizes maintenance overhead . The illustration source code provides a basic framework that can be enlarged and customized to accommodate different testing needs.

Frequently Asked Questions (FAQs):

Q1: What are the limitations of a Keyword Driven Framework?

A1: While strong, KDFs can grow elaborate to handle for highly large and

intricate projects. The starting setup can also necessitate significant work.

Q2: Can I integrate a Keyword Driven Framework with other testing tools?

A2: Yes, KDFs are not restricted to UFT. The subjacent principles can be utilized via other QA tools and languages .

Q3: How do I process errors in a Keyword Driven Framework?

A3: Robust error handling is essential . You can execute exception-handling blocks within your UFT functions to seize and process errors gracefully .

Q4: What are some best practices for designing a Keyword Driven Framework?

A4: Follow clear naming standards for keywords. Keep keywords brief and focused . Properly document your keywords and their purpose . Use a revision control system to track changes to your structure .

https://www.backpackingmatt.com/lib/tk/548T38K/DOC/modern-analytical_chemistry_david_harvey_solutions_manual.pdf
https://www.backpackingmatt.com/pub/906M93J/421M1213J5/EPUB/microeconomics_robert_pindyck_8th_solution-manual.pdf
https://www.backpackingmatt.com/short/iv/2015928IV7260909/EPDF/2008__klr650_service_manual.pdf
https://www.backpackingmatt.com/ppt/uz092609/123Z65502U132856/PPT/81-z250-kawasaki_workshop_manual.pdf
https://www.backpackingmatt.com/book/45069R314N/52464RN/PPT/atr__72__600-systems-guide.pdf
https://www.backpackingmatt.com/pub/37274QJ/132856090926/EPDF/papoulis-and_pillai-solution_manual.pdf
https://www.backpackingmatt.com/text/132856/82406BP/PDF/top-notch_1__workbook_answer-key_unit_5.pdf
https://www.backpackingmatt.com/journal/132856/1CQ5147/RTF/2015__audi_owners_manual.pdf
https://www.backpackingmatt.com/edu/l8D2703/132856090926/KINDLE/fish_by__stephen_lundin.pdf

https://www.backpackingmatt.com/ref/bc/4C730B5294260909/TEXT/fourth_grade_math_pacing-guide-hamilton_county.pdf