Keyword Driven Framework In Qtp With Complete Source Code

Keyword Driven Framework in QTP with Complete Source Code

The world of automated software testing is constantly evolving, and one technique that significantly improves efficiency and maintainability is the keyword-driven framework. This article dives deep into creating a keyword-driven framework in QuickTest Professional (QTP), now known as UFT (Unified Functional Testing), providing you with a complete source code example

and a comprehensive understanding of its implementation. We'll explore the benefits, practical usage, and potential challenges, equipping you with the knowledge to build and leverage this powerful testing approach. This framework allows for efficient test automation by separating test scripts from test data, a concept we'll explore in detail, covering topics like **data-driven testing**, **excel integration**, and **function libraries**.

Introduction to Keyword-Driven Frameworks in UFT

A keyword-driven framework in UFT streamlines the testing process by employing keywords to represent specific actions within your application under test (AUT). Instead of writing complex VBScript code for each test case, you define a set of keywords that correspond to these actions (e.g., "Login," "Click Button," "Verify Text"). Test cases are then created by simply sequencing these keywords along with the necessary data. This approach promotes reusability, simplifies maintenance, and allows business users with limited coding

experience to participate in test case design. This makes it a crucial element of successful **test automation strategies** .

Benefits of a Keyword Driven Framework in UFT

Using a keyword-driven approach offers several significant advantages:

- **Increased Reusability:** Keywords encapsulate actions, making them reusable across multiple test cases. This drastically reduces code duplication and development time.
- **Improved Maintainability:** Changes to the AUT require minimal modifications to the test cases. You primarily update the keyword definitions rather than rewriting extensive scripts.
- **Enhanced Collaboration:** The keyword-based approach allows business analysts and testers to easily collaborate on test case design, as they don't need advanced programming skills.
- **Reduced Test Script Complexity:** Test scripts become simpler and more

readable, leading to easier debugging and troubleshooting.

- **Better Data Management:** Separating test data from the scripts allows for efficient management and manipulation of test input values. We'll see how this works with **excel data-driven testing** below.

Implementing a Keyword-Driven Framework in UFT: A Complete Example

Let's build a simple keyword-driven framework in UFT. We'll use an Excel spreadsheet to store our test data and keywords, and a function library to handle the actual actions.

1. Excel Sheet (TestData.xls):

Keyword	Object	Value
OpenBrowser	Browser	"www.google.com"

| TypeText | SearchBox| "Keyword Driven Framework" |

| ClickButton | SearchButton | |

| VerifyText | ResultPage | "Keyword Driven Framework" |

| CloseBrowser | Browser | |

2. UFT Function Library (KeywordLibrary.vbs):

```vbscript

Function OpenBrowser(browserName, url)

Set browser = Browser("name:=" & browserName).Create()

browser.Navigate url

End Function

Function TypeText(objectName, text)

Set obj = Description.Create()

obj("name").Value = objectName

Set obj = Window("title:=.*").ChildObjects(obj)
```

```
obj.SetFocus

obj.Type text

End Function

Function ClickButton(objectName)

Set obj = Description.Create()

obj("name").Value = objectName

Set                  obj                 =
Window("title:=.*").ChildObjects(obj)

obj.Click

End Function

Function          VerifyText(objectName,
expectedText)

Set obj = Description.Create()

obj("name").Value = objectName

Set                  obj                 =
Window("title:=.*").ChildObjects(obj)

If      InStr(obj.GetROProperty("text"),
expectedText) > 0 Then
```

```
Reporter.ReportEvent micPass, "Verification Passed", "Text '" & expectedText & "' found."

Else

Reporter.ReportEvent micFail, "Verification Failed", "Text '" & expectedText & "' not found."

End If

End Function

Function CloseBrowser(browserName)

Set browser = Browser("name:=" & browserName).Object

browser.Close

End Function
```

3. UFT Test Script (TestScript.mts):

```vbscript
'Set the path to your Excel file.
```

```
Set objExcel = CreateObject("Excel.Application")

Set objWorkbook = objExcel.Workbooks.Open("C:\TestData.xls")

Set objSheet = objWorkbook.Worksheets(1)

'Loop through each row in the Excel sheet

For i = 2 To objSheet.UsedRange.Rows.Count

keyword = objSheet.Cells(i, 1).Value

object = objSheet.Cells(i, 2).Value

value = objSheet.Cells(i, 3).Value

'Call the appropriate function based on the keyword

Select Case keyword

Case "OpenBrowser"

Call OpenBrowser(object, value)

Case "TypeText"

Call TypeText(object, value)
```

```
Case "ClickButton"

Call ClickButton(object)

Case "VerifyText"

Call VerifyText(object, value)

Case "CloseBrowser"

Call CloseBrowser(object)

End Select

Next

objWorkbook.Close

objExcel.Quit
```

This example demonstrates a basic implementation. In a real-world scenario, you'd have a more extensive function library and a robust mechanism for error handling and reporting. This example also showcases **excel integration** and its role in a keyword driven framework.

Usage and Implementation Strategies

Successful implementation of a keyword-driven framework requires careful planning and execution. Key steps include:

- **Keyword Identification:** Define a comprehensive set of keywords that covers all the required actions in your AUT.
- **Function Library Development:** Create well-structured and reusable functions for each keyword.
- **Data Management:** Establish a clear system for organizing and managing test data, often using Excel or a database. This will be a core part of your **data-driven testing** strategy.
- **Test Case Design:** Create test cases by simply sequencing keywords and input data.
- **Execution and Reporting:** Implement a mechanism for executing test cases and generating comprehensive reports.

Conclusion

The keyword-driven framework in UFT offers a highly efficient and maintainable approach to automated software testing. By separating test logic from test data and using reusable keywords, this framework significantly reduces development time, enhances collaboration, and improves overall test quality. While the initial setup requires careful planning, the long-term benefits far outweigh the initial investment. Remember to focus on choosing meaningful keywords and building a well-structured function library to truly reap the rewards of this powerful testing methodology.

FAQ

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

A1: While highly beneficial, keyword-driven frameworks have limitations. For very complex scenarios with intricate logic, they can become less efficient than a data-driven or hybrid approach. Furthermore, extensive keyword creation

and maintenance can be challenging for large-scale projects. Proper planning and a modular design are key to mitigating these challenges.

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

A2: A keyword-driven framework focuses on actions (keywords) that are executed based on data provided (often from an external source like Excel), while a data-driven framework primarily focuses on iterating through test data to execute the same set of actions repeatedly. They are not mutually exclusive and can be combined effectively in a hybrid approach, further enhancing testing efficiency.

Q3: Can I use a database instead of Excel for test data?

A3: Absolutely! Using a database like SQL Server or MySQL offers several advantages, particularly for larger projects with extensive test data. You'd need to adjust your script to interact with the database instead of Excel, but this provides more robust data management capabilities.

Q4: How do I handle errors and exceptions in a keyword-driven framework?

A4: Robust error handling is crucial. Your functions should include try-catch blocks to gracefully handle exceptions. You should also implement detailed logging and reporting mechanisms to record error details and aid in debugging. The UFT reporting features are excellent for this purpose.

Q5: What are some best practices for designing keywords?

A5: Use clear, concise, and descriptive names. Keep keywords self-contained and avoid dependencies on other keywords wherever possible. Aim for modularity – break down complex actions into smaller, more manageable keywords. Follow consistent naming conventions throughout your framework.

Q6: How can I integrate this with continuous integration/continuous deployment (CI/CD) pipelines?

A6: You can integrate your keyword-driven framework with CI/CD pipelines by automating the execution of your test

scripts using tools like Jenkins or Azure DevOps. The results can then be integrated into your CI/CD pipeline to trigger appropriate actions based on test success or failure.

Q7: What is the role of object repository in this framework?

A7: While not explicitly shown in the simplified example, a well-maintained object repository remains essential. The object repository stores information about the objects in your application under test. This allows for improved maintainability and reduces the risk of test script breakage when UI elements change slightly. You should reference objects from the repository within your keyword functions.

Q8: Is this approach suitable for all types of testing?

A8: Keyword-driven frameworks are particularly well-suited for functional testing and regression testing. They can be adapted for other types of testing, but the effectiveness may vary depending on the specific testing approach and the complexity of the application. They are

less often directly used for performance or security testing, for example, though the underlying principles of modularity and reusability can still be applied.

Keyword Driven Framework in QTP with Complete Source Code: A Comprehensive Guide

Automating validation procedures is essential in today's fast-paced software creation landscape. Traditional testing is inefficient and susceptible to mistakes , while automation presents significant advantages in terms of velocity , correctness, and re-application. One robust approach to software test automation is the Keyword Driven Framework (KDF), and this article will delve into its execution using QuickTest Professional (QTP), now known as UFT (Unified Functional Testing), providing a complete source code example.

The Keyword Driven Framework in QTP arranges test cases around keywords, which signify specific actions or

functions within the application being tested . These keywords are held in a centralized repository, typically an Excel spreadsheet or a database, alongside linked data, such as input values and anticipated outcomes . The framework then reads these keywords and data, running the corresponding QTP scripts to robotize the verification cycle.

This approach offers several considerable merits:

- **Increased Reusability**: Keywords can be recycled across multiple test cases, reducing repetition and upkeep .
- **Improved Maintainability**: Changes to the test scripts are isolated , making it simpler to update and sustain the framework.
- **Enhanced Collaboration**: The separation of test creation from test scripting permits for better collaboration between test engineers and developers.
- **Easier Understanding**: The keyword-driven technique makes it less difficult for non-programmers to understand and modify test cases.

Let's investigate a practical example. Imagine validating a login feature . The keywords might include: "OpenBrowser", "TypeText", "ClickButton", "VerifyText". The data sheet would contain information such as the address of the website , the login , the passphrase, and the expected greeting.

The QTP script would then access this data, execute the appropriate keywords, and match the obtained outcomes against the anticipated outcomes .

Complete Source Code Example (Illustrative):

This example uses VBScript within QTP. Remember that this is a simplified illustration and a full-fledged framework requires more elaborate error handling, logging, and reporting mechanisms.

```vbscript

' Read data from Excel sheet (replace "testData.xls" with your file)

Set objExcel = CreateObject("Excel.Application")
```

```
Set objWorkbook = objExcel.Workbooks.Open("testData.xls")

Set objWorksheet = objWorkbook.Worksheets(1)

' Loop through each test case

For i = 2 To objWorksheet.UsedRange.Rows.Count

keyword = objWorksheet.Cells(i, 1).Value

value1 = objWorksheet.Cells(i, 2).Value

value2 = objWorksheet.Cells(i, 3).Value

Select Case keyword

Case "OpenBrowser"

Browser("title:=.*").Page("title:=.*").WebEdit("na
value1

Case "TypeText"

Browser("title:=.*").Page("title:=.*").WebEdit("na
value2

Case "ClickButton"

Browser("title:=.*").Page("title:=.*").WebButton("
```

```
Case "VerifyText"

' Add your verification logic here

End Select

Next

objWorkbook.Close

objExcel.Quit

Set objWorksheet = Nothing

Set objWorkbook = Nothing

Set objExcel = Nothing
```

This code snippet demonstrates the fundamental idea of reading keywords and data from an Excel sheet and then using them to drive the QTP script's execution. A strong framework would require more advanced error handling, logging, and reporting capabilities, but this example provides a foundation for building a complete Keyword Driven Framework in QTP. Remember to configure the necessary components and adapt the code to suit your specific system and

testing needs .

Conclusion:

The Keyword Driven Framework in QTP provides a structured and sustainable method to automate software testing . By separating test development from implementation, it improves collaboration , reusability , and sustainability. While this article provides a simplified example, executing a full-fledged KDF requires careful planning, design, and attention to detail. The rewards, however, are substantial in terms of efficiency and excellence of the testing process .

Frequently Asked Questions (FAQs):

1. **Q: What are the limitations of a Keyword Driven Framework?**

A: While KDFs offer many benefits, they can become intricate to manage for large endeavors. Careful planning and development are vital to avert superfluous complexity .

2. **Q: Can I use other stores besides Excel?**

A: Absolutely . Databases such as Oracle are often used as substitute data sources for a Keyword Driven Framework.

3. **Q: What scripting languages can I use with QTP?**

A: QTP (UFT) primarily uses VBScript, but it also supports other languages through add-ins .

4. **Q: How do I handle mistakes in my Keyword Driven Framework?**

A: Robust error handling is crucial . Implement error-handling blocks in your QTP scripts to gracefully manage errors and record them appropriately. This enables problem-solving and locating the root cause of malfunctions .

https://dokument.biblioteka.traefik.light.krakow.p
of__dental__anatomy-and__surgery.pdf
https://dokument.biblioteka.traefik.light.krakow.p
rebel_t2i__manuals.pdf
https://dokument.biblioteka.traefik.light.krakow.p
integral_equations__boundary__problems__of_functio
their_application_to_mathematical_physics-
n__i_muskhelishvili.pdf
https://dokument.biblioteka.traefik.light.krakow.p
design-
and_implementation_solution__manual.pdf