WWW.TECHNODIVINE.COM

REFACTORING KEYWORD DRIVEN APPROACH WITH FUNCTIONAL POINTERS

For Automation Geeks| By Bharat Kakkar

Although, a developer always tries to write best solution in the code, refactoring is something, which can never be ignored.

A problem may have multiple solutions. However, implementing the best is always on priority. Refactoring is a technique to restructure the code to have best possible solution implemented. Thus, if you are in process of maintaining the code by refactoring the same, one of the best practices is to find a smallest way with minimal maintenance.

Let us discuss one of those solutions here, which may help you to develop such an implementation. Specially, if you have a keyword driven framework with function names as your keywords, you use a lot many select cases.

Let us consider one of the aforesaid examples wherein you have a situation to call a specified function based on a conditional statement (function names are your keywords and user specifies which function to be called in an external datasheet). One of the alternatives to this solution will be to have a **select statement** implemented, however what if your conditions are increasing gradually or you are adding more functions to your framework and these function are to be called based certain condition (user selection in our case). So, every time you introduce a new function you need to update the select condition. Additionally, for every new condition you will be adding three lines of code

1. Case <value>
2. Function call
3. Break

Now, think about the solution which will not need any updates to be made in the code for every new keyword and all you need to do is to add new function (**And that's it**).

The solution is "**Functional Pointers**". Let us first understand what functional pointer is?

Functional pointers are reference to a memory location where a piece of code is stored or a function stored in your application/script which may be executed on the fly.

Note:

- When a functional pointer is called and the code stored within the memory location it is called direct functional pointer
- When a functional pointer is called and the code stored within the Application/script it is called In-direct functional pointer

Now, let me show how we can implement "functional pointers" in VBScript.

This would be done using `getref` function

Definition: It points to the address of a procedure to be executed. Returns a reference to a procedure that can be bound to an event

Implementation:

```
'Getting the values from user
FunctionParameter1=InputBox("Enter the Parameter value (valid Values
TestFunction/TestFunctionOne", "Technodivine.com Example","10")
FunctionName=InputBox("Enter the Parameter value", "Technodivine.com
Example","TestFunction")

'Calling function based on user input
Set FuntionPointer=getref(FunctionName)
Call FuntionPointer (FunctionParameter1)

'Function defination
Function TestFunction(FunctionParameter1)
    msgbox "Called TestFunction with parameter value : "& FunctionParameter1
End Function

Function TestFunctionOne(FunctionParameter1)
    msgbox "Called TestFunctionOne with parameter value : "&
FunctionParameter1
End Function
```

Applying Error handling similar to case default (in our case, user call a function which was never defined):

```
If FuntionPointer Is Nothing Then
    MsgBox FunctionName &" is not defined!" 'FunctionExists = False
End if

                OR

If Err.number <> 0 Then
          MsgBox FunctionName &" is not defined!" 'FunctionExists = False
End If
```

Write one of the above codes before you call the functional Pointe, i.e. `Call FuntionPointer (FunctionParameter1).`