esProc

Excel integrate esProc

Issued by Raqsoft

Contents

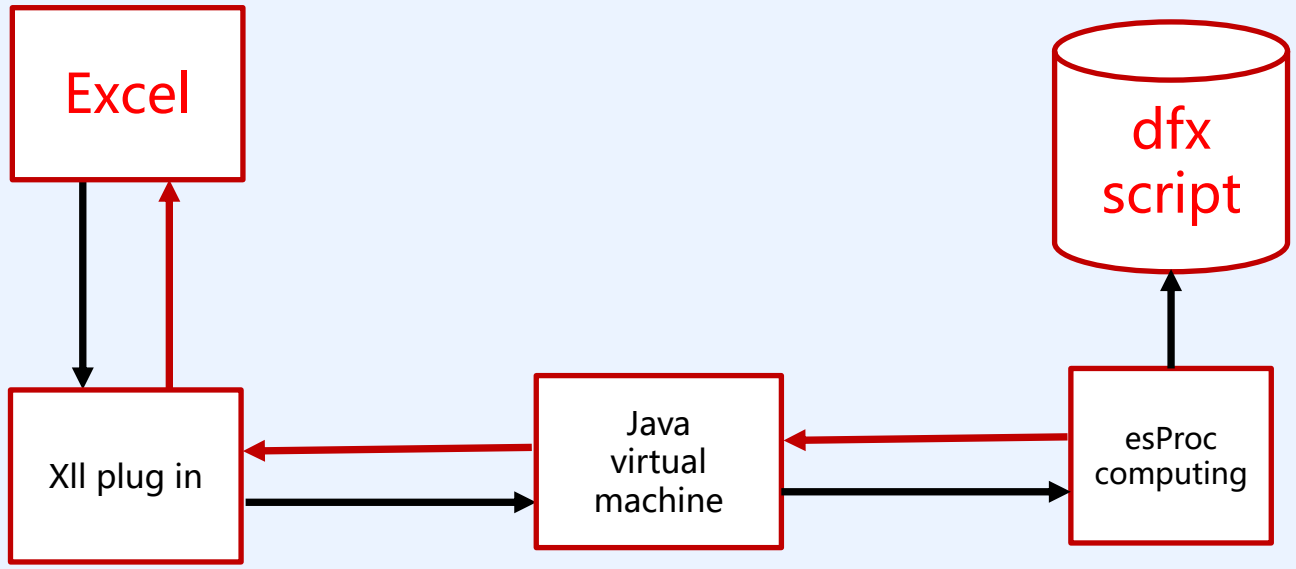
1 Preface

Excel is the leading spreadsheet program in the industry, which is widely used in all professions and trades. It is the most popular personal computer data processing software.

It is a data computing engine for structured processing. It has agile syntax, grid style script and complete debugging function. It is specially designed for structured/semi-structured data processing. It is suitable for multi-step, complex business rules and multiple mixed data sources. It can provide computing services for massive data with complex business logic.

Excel is used as the front-end for users, and the interface, calculation, business logic and data access functions provided by esProc SPL language are used to realize the best combination of the two.

2 Structural relationship



Excel passes the data to Java virtual machine through the XII plug-in, which extends its function. The virtual machine informs the esProc computing engine. After executing the DFX script, the calculation results are returned according to the original path and presented in Excel.

For users, on the basis of understanding the structural relationship between excel and esProc, the focus is on how the front-end excel calls the DFX script to solve the needs of users.

3 Environment setting

A. Java installation

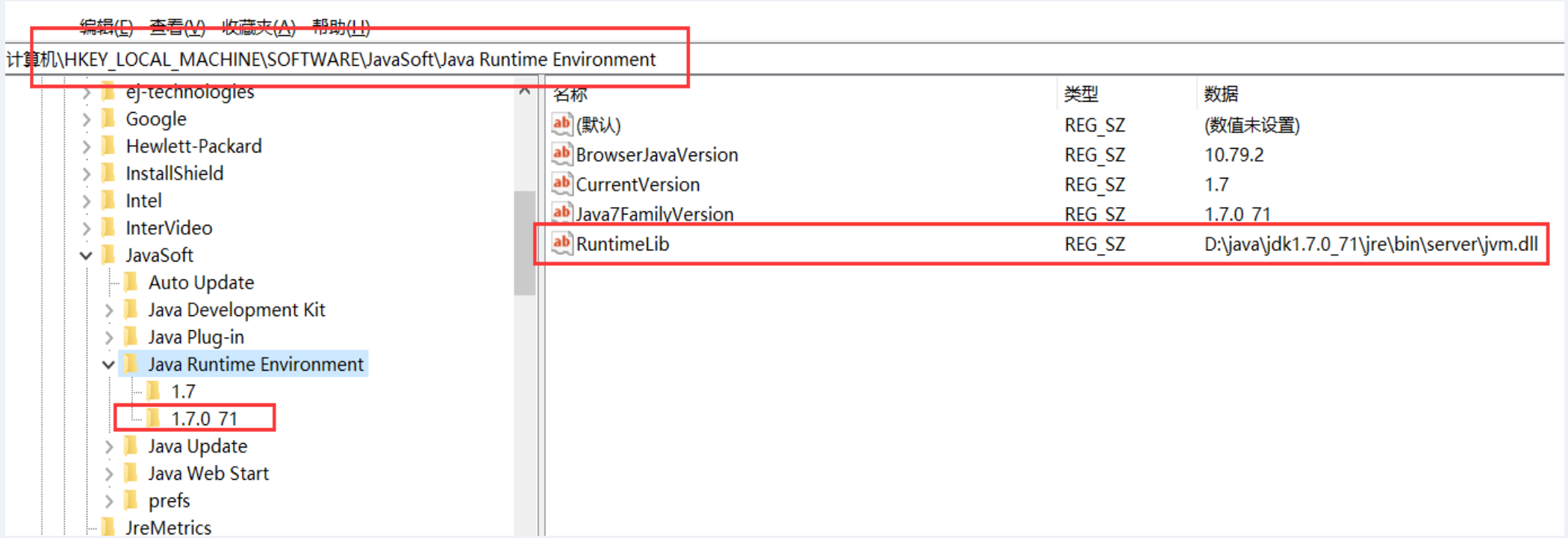
I. The same architecture is required: Excel and Java JDK are the same x86 or x64 architecture, and can't be combined in different forms, otherwise exceptions will occur. Run java -version to know if it is an x64 schema.

II. JDK with SPL: if the jvm.dll of Java JDK in regedit of the system registry is not installed or invalid, excel will load the JDK in esProc(under raqsoft\common\jre\bin\server), that is, the JDK in the registry takes precedence over the JDK with esProc.

III. user installation of JDK: if the user wants to use the specified version of Java, for the Java JDK is the installation version or the green version, it is necessary to ensure that the information of JDK in the registry is valid.

A. Java installation

Let's take win10 jdk1.7.0.71 x86 as an example.
JDK location in the registry: Under HKEY_LOCAL_MACHINE\SOFTWARE
X86：SOFTWARE\\JavaSoft\\Java Runtime Environment
X64：SOFTWARE\\WOW6432Node\\JavaSoft\\Java Runtime Environment
In version 1.7.0.71, the runtimelib value is the location of the jvm.dll file.



When you install multiple versions of Java on your computer, you need to pay attention to the conflict between JDK versions, which may cause Excel to fail to load plug-ins.

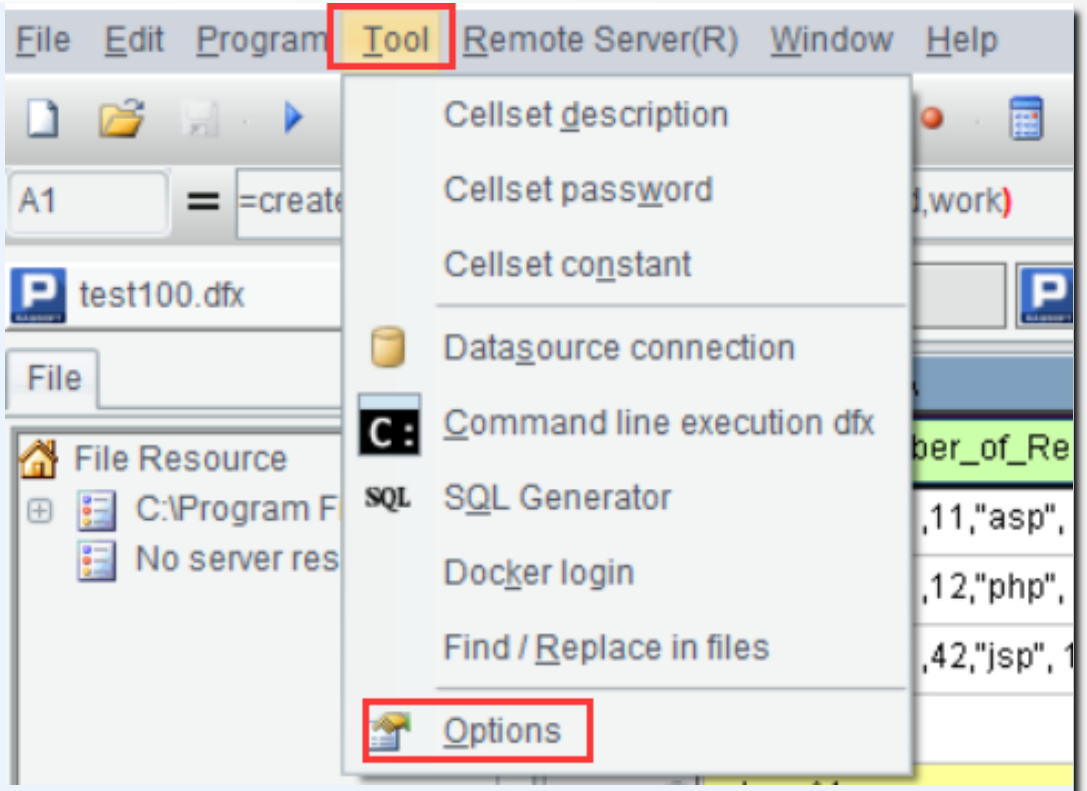
B. esProc setting

I. excel related files:

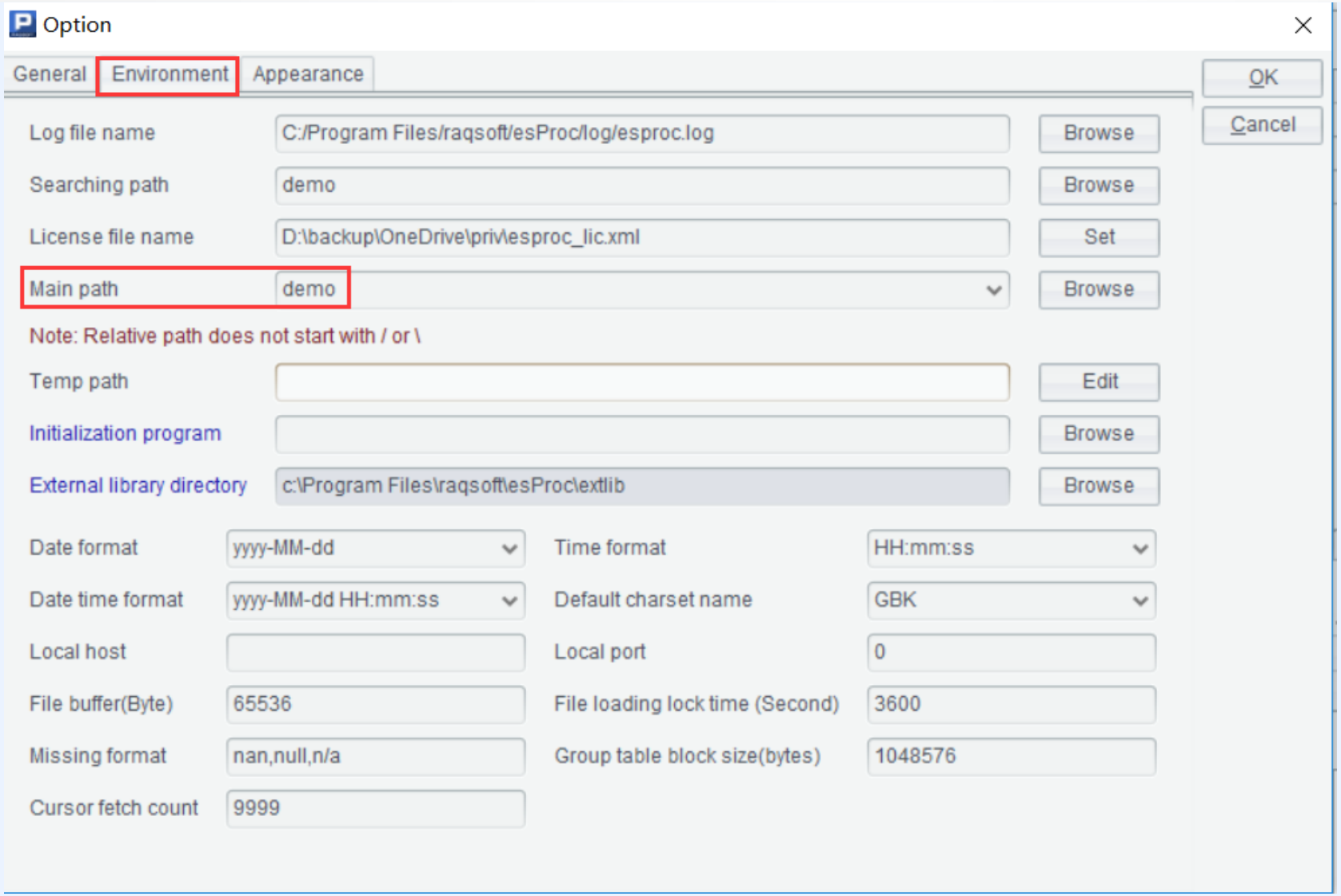
After esProc is installed, the two files related to excel are:
esProc\lib\EsprocXll.jar,
esProc\bin\ExcelRaq.xll。
Excelraq.xll is the excel plug-in of esProc, and their location cannot be changed, otherwise, the plug-in fails to load.

II. Location setting of DFX file:

Set the storage location of DFX script file to facilitate the call of SPL. The operation of setting the DFX file path in esProc interface is as follows:
Menu—>Tool—>Options—>Environment—>
Main path



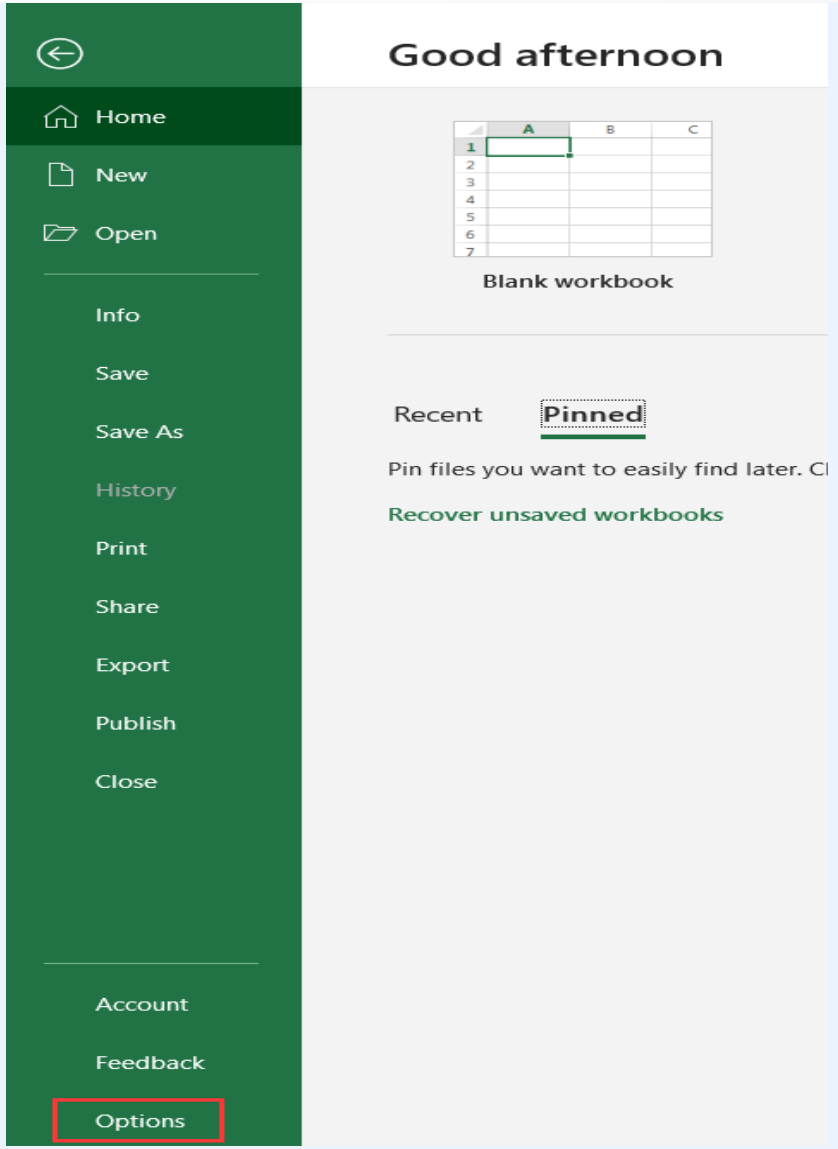
B. esProc setting



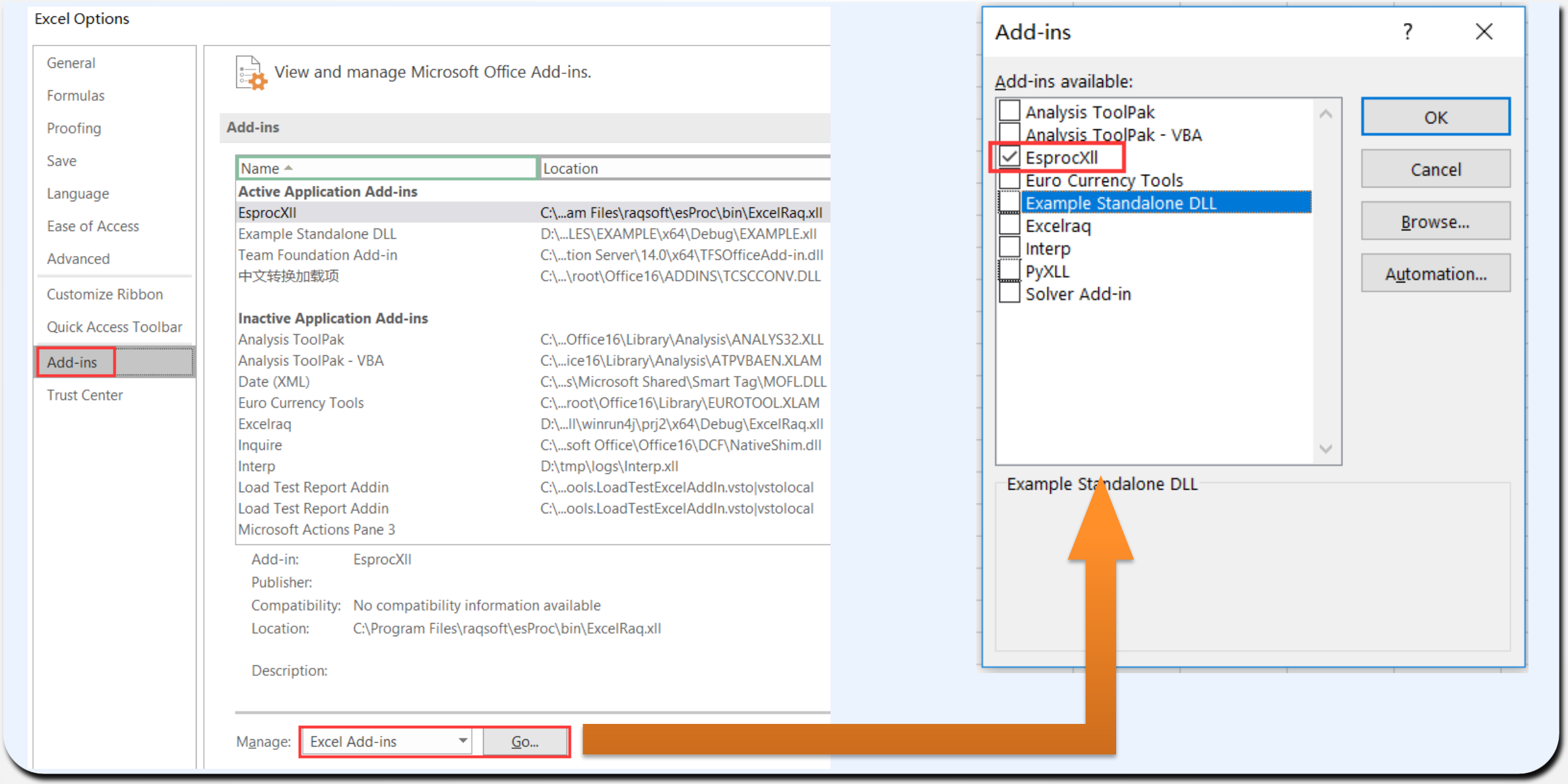
By default, the location of the DFX file is set in esProc installation directory raqsoft\esProc\demo.

C. load plug-in

After opening Excel software, you can select excelraq.xll file through file - > Options - > add in - > go to - > Browse - > add in, check esprocxll in add in, and excelraq file is enabled.



C. load plug-in



4. Operation instructions

DFX interface call Description:
dfx (fmt, arg1,...) Function interface.
Parameter fmt: If it is a DFX file name without suffix, it means calling DFX script file; If it is an expression string starting with "=", it means dynamically parsed and evaluated expression. When it is an expression, the number of? included is the same as that of Arg and corresponds one by one.
Parameter arg1,...: Input parameters can be empty, one or more, but up to 50 parameters, separated by commas. Parameter types can be string, int, float, double, one-dimensional array, two-dimensional array, etc.
Return type: array structure.

Cell value operation: After selecting a cell, enter similar =dfx("tsum", A1:C3) in the input field, and then click enter to display the return value in this cell only.

Table filling operation: Select the grid to be filled, enter similar =dfx("demo" ,A1:C3) in the input field, and then press Ctrl + Shift + enter to achieve automatic filling.

Copy operation: When the DFX script uses the clipboard (s) function and excel calls DFX for execution, you can select another cell to copy the calculation results in the clipboard.

5. Use of DFX

The following describes how excel calls the DFX script through different operation modes of DFX.

A. Cell value operation

Let's start with two ways to rank:

I RANK:

For example, if there are five numbers 100,100, 80, 80, 60, 50, the result is

First: 100,100
Second: empty
Third: 80
Fourth: 60
Fifth: 50

It can be seen from the above ranking that if there is a parallel ranking situation , several of them are the same, and the following positions will be vacated.

II、China style rank

The result of the same five numbers ranking is

First 100,100
Second: 80
Third: 60
Fourth: 50

From the above result, we can see that the Chinese ranking will not break the ranking because of the same number ranking, but will be continuous.

A. Cell value operation

For the first type of ranking, excel provides the corresponding interface rank (). For Chinese ranking, VBA script is used. The code is as follows:

```
Function cnrank(ByVal nm, ByVal rng As Range)
    Set d = CreateObject("scripting.dictionary")
    For Each rn In rng
        If VBA.IsNumeric(rn.Value) And Len(rn) > 0 Then
            d(rn.Value) = ""
        End If
    Next rn
    arr = d.keys
    d.RemoveAll
    For j = 0 To UBound(arr)
        d(WorksheetFunction.Large(arr, j + 1)) = j + 1
    Next j
    If VBA.IsNumeric(nm) Then
        pm = d(nm * 1)
    Else
        pm = ""
    End If
End Function
```

A. Cell value operation

Compared with the implementation of VBA code, SPL code is much simpler, as follows:

	A	B	C	D	E
1	Name	score	order		
2	Tom1	99	1		
3	Tom2	99	1		
4	Tom3	25	19		=dfx("=[?].rank@z(?)", B2:B21, 28)
5	Tom4	26	18		16
6	Tom5	12.3	20		
7	Tom6	28	16		
8	Tom7	29	15		
9	Tom8	30	14		=dfx("=[?].rank@zi(?)", B2:B21, 28)
10	Tom9	31	13		14
11	Tom10	32	12		
12	Tom11	33	11		
13	Tom12	28	16		
14	Tom13	35	10		
15	Tom14	36	9		
16	Tom15	37	8		
17	Tom16	38	7		
18	Tom17	39	6		
19	Tom18	40	5		
20	Tom19	41	4		
21	Tom20	42	3		

The first type Rank

Chinese ranking

B. Table filling operation

Next, the clothing table will be classified and summarized, and the calculation results will be displayed in Excel. Load the data of clothing.xls, and calculate the sales volume of each branch by type and category.

clothing.xls data:

type	category	store1	store2	store3	store4	store5
man	A	100	200	300	400	350
man	B	300	400	500	600	522
man	A	800	900	1000	1100	450
man	B	200	300	400	500	300
man	C	400	500	600	700	891
woman	B	600	700	800	900	257
woman	C	500	600	700	800	880
woman	A	700	800	900	1000	750
woman	C	900	1000	1100	1200	440
woman	E	280	0	0	360	600

B. Table filling operation

VBA code is implemented as follows:

```
Sub clothing()
    Set d = CreateObject("scripting.dictionary")
    Application.ScreenUpdating = False

    arr = Sheets(1).[a1].CurrentRegion
    c = UBound(arr, 2)
    For j = 3 To UBound(arr)
        For i = 3 To UBound(arr, 2)
        d(arr(j, 1) & "##" & arr(j, 2) & "##" & arr(2,
i)) = d(arr(j, 1) & "##" & arr(j, 2) & "##" & arr(2,
i)) + arr(j, i)
        Next i
    Next j
    Sheets(2).UsedRange.ClearContents
    Sheets(1).Rows(2).Copy Sheets(2).[a1]
    r = 2
    With Sheets(2)
        For j = 0 To d.Count - 1
            arr = Split(d.keys()(j), "##")
            If d.exists(arr(0) & arr(1)) Then
                a = d(arr(0) & arr(1))
            Else
                a = r
                .Cells(a, 1) = arr(0)
                .Cells(a, 2) = arr(1)
                d(arr(0) & arr(1)) = a
                r = r + 1
            End If
            For i = 3 To c
                .Cells(a, i) = d(arr(0) & "##" & arr(1) & "##"
& .Cells(1, i))
            Next i
        Next j
    End With
    Application.ScreenUpdating = True
End Sub
```

B. Table filling operation

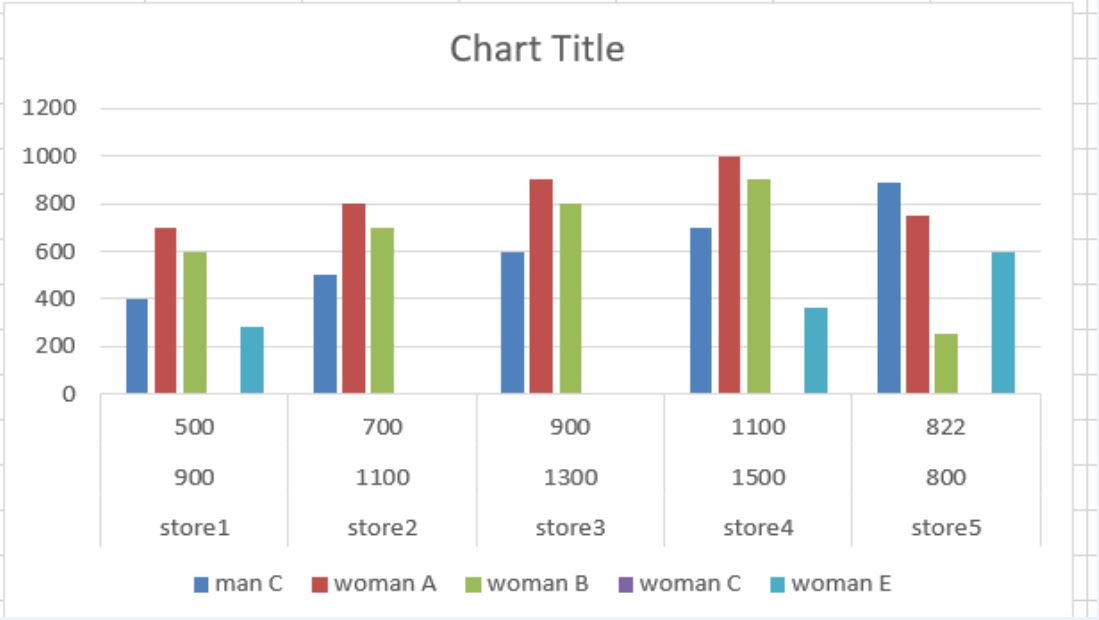
Due to the lack of grouping interface in VBA, the code details need to be implemented by itself. However, SPL is much more comprehensive and easy to implement for such problems. The script code of using clothing.dfx is as follows:

	A
1	=file("D:/dev/clothing.xls").xlsopen()
2	=A1.xlsimport@t(;A1(1).stname, 1)
3	=A2.group(type, category)
4	=A3.new(type, category, ~.sum(#3):store1, ~.sum(#4):store2,~.sum(#5):store3,~.sum(#6):store4,~.sum(#7):store5)
5	>A1.xlsclose()
6	return A4

B. Table filling operation

G8 {=dfx("clothing")}

	A	B	C	D	E	F	G
1	type	category	store1	store2	store3	store4	store5
2	man	A	900	1100	1300	1500	800
3	man	B	500	700	900	1100	822
4	man	C	400	500	600	700	891
5	woman	A	700	800	900	1000	750
6	woman	B	600	700	800	900	257
7	woman	C	1400	1600	1800	2000	1320
8	woman	E	280	0	0	360	600



Operating instructions: In Excel, select A1: G8 area first, and then input =dfx("clothing") in the input box, and press Ctrl + Shift + enter at the same time to realize automatic filling, as shown in the left figure:

The histogram can also be inserted to make the calculation results more intuitive.

C. Copy operation

Still use the above clothing test data, only the DFX script adds the copy function.
The script code of clipboard.dfx is as follows:

	A
1	=file("D:/dev/clothing.xls").xlsopen()
2	=A1.xlsimport@t(;A1(1).stname, 2)
3	=A2.group(type, category)
4	=A3.new(type, category, ~.sum(#3):store1, ~.sum(#4):store2,~.sum(#5):store3, ~.sum(#6):store4,~.sum(#7):store5)
5	>A1.xlsclose()
6	=clipboard(export@t(A4))

Copy calculation result to clipboard

Operating instructions: In Excel, first select cell A1, and then input in the input box: =dfx("clipboard") , click enter, and then paste from A2 with ctrl+v. The result is as follows:

A1 =dfx("clipboard")

	A	B	C	D	E	F	G
1	TRUE						
2	type	category	store1	store2	store3	store4	store5
3	man	A	900	1100	1300	1500	800
4	man	B	500	700	900	1100	822
5	man	C	400	500	600	700	891
6	woman	A	700	800	900	1000	750
7	woman	B	600	700	800	900	257
8	woman	C	1400	1600	1800	2000	1320
9	woman	E	280	0	0	360	600

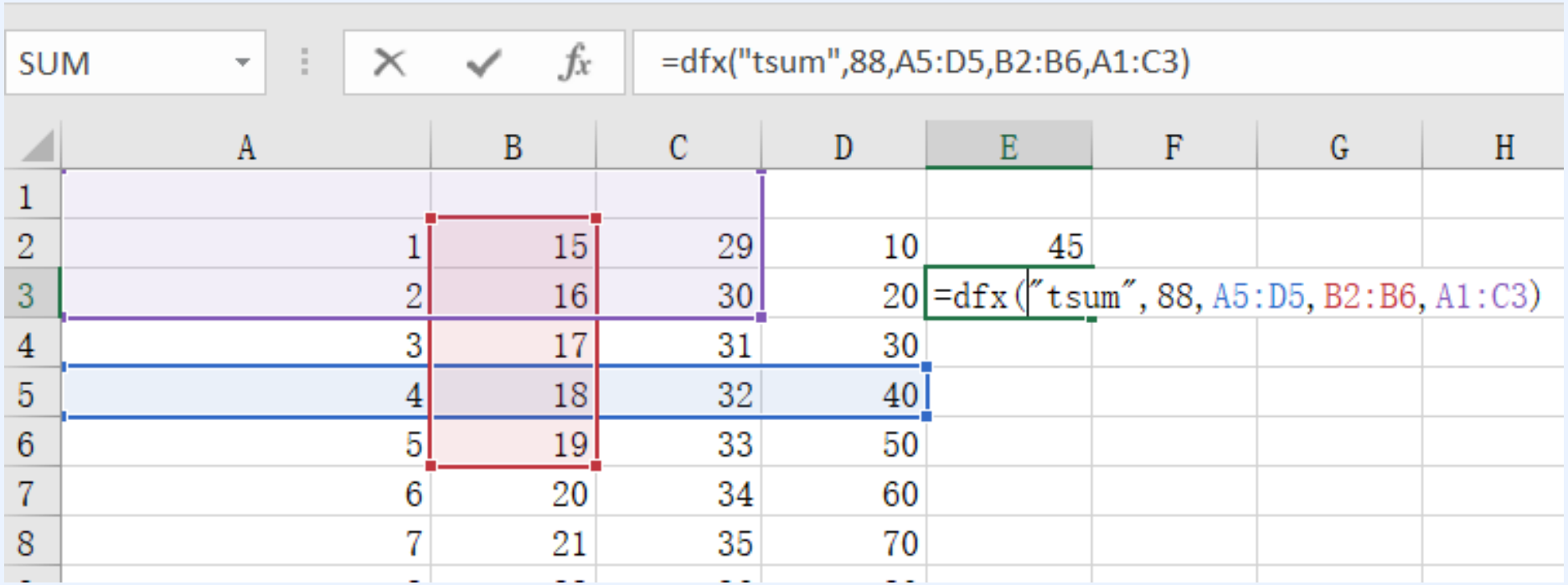
D. Pass parameters to DFX

Pass different types of parameters to the DFX script. Use the tsum.dfx script code as follows:

	A	B	C	D	E
1	0	0	0	0	
2	if (ifa(arg1))	>A1=arg1.sum()	else	if(ifnumber(arg1))	>A1=arg1
3				else	>A1=len(arg1)
4	if (ifa(arg2))	>B1=arg2.sum()	else	if(ifnumber(arg2))	>B1=arg2
5				else	>B1=len(arg2)
6	if (ifa(arg3))	>C1=arg3.sum()	else	if(ifnumber(arg3))	>C1=arg3
7				else	>C1=len(arg3)
8	if (ifa(arg4))	for arg4.count()			
9			if (ifa(arg4(B8)))	>D1=D1+arg4(B8).sum()	
10	return A1+B1+C1+D1				

D. Pass parameters to DFX

The parameters passed are integer, single row multi value, single column multi value and multi row multi column value.



For the date type, you need to convert it to a string. The code is as follows:
=dfx("testDate",23.33,13,"2011-02-03")
=dfx("testDate",A1,C1,TEXT(B1,"yyyy-mm-dd")), where B1 is date type data 。

6. Example of VBA calling DFX interface

The SPL interface is invoked in the VBA script to realize their interaction. The tdemo.dfx script code is as follows:

	A
1	=create(age, name,pid,work)
2	>A1.insert(0,11,"asp", 100, "techer")
3	>A1.insert(0,12,"php", 200, "manager")
4	>A1.insert(0,42,"jsp", 300, "java")
5	>A1.insert(0,43,"java", 400, "help")
6	=clipboard(export@t(A1))
7	return A1

The functions Run() ExecuteExcel4Macro() of application are mainly used.

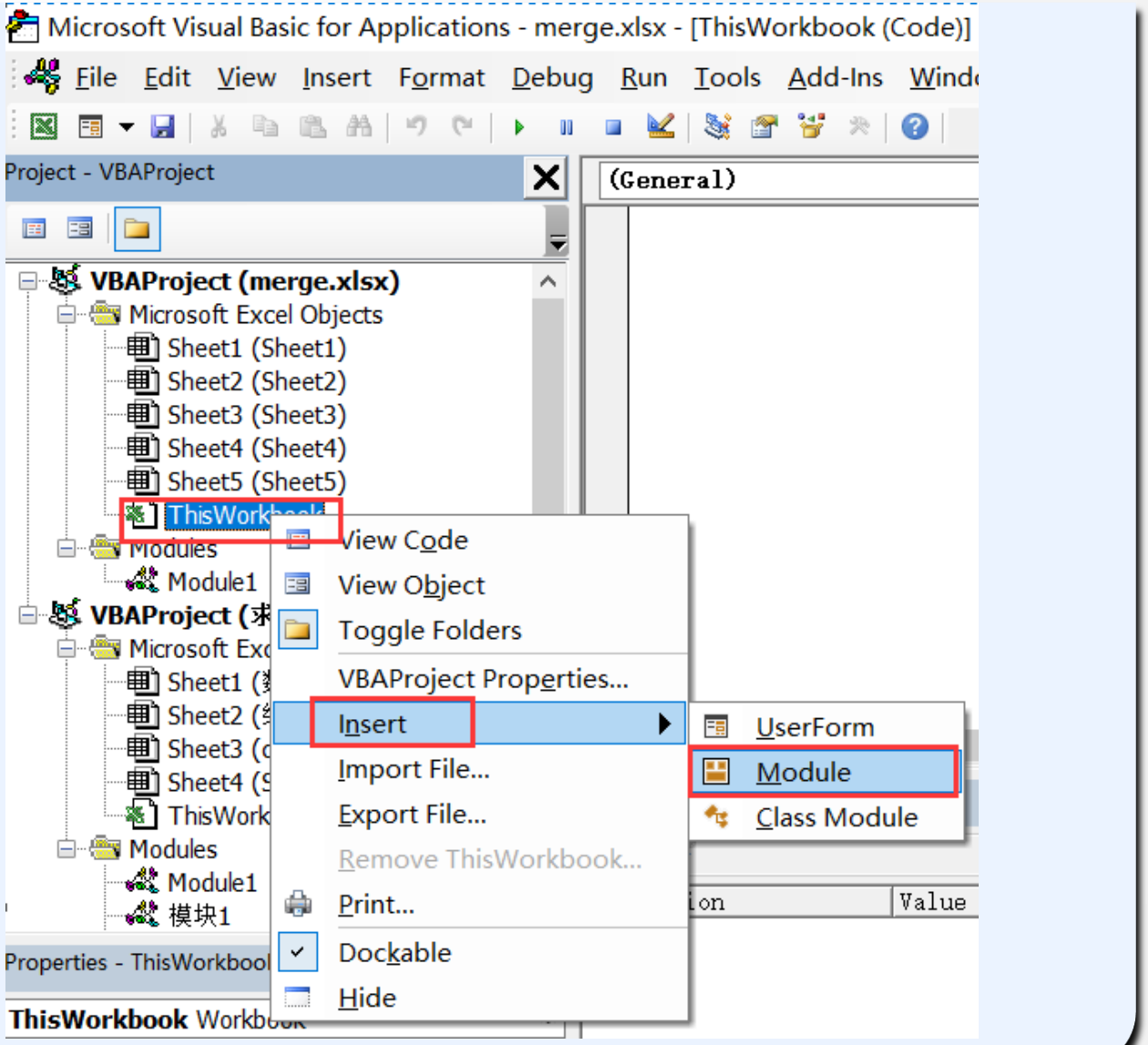
A. Application.Run() invocation

Operation steps:

A: Alt+F11 open VBA compiler, right-click on ThisWorkbook and select insert module.

B. Enter the following sub test() function code, which displays the result in Excel.

C: Execute function test(), VBA script automatically fills the result in A1: D5.

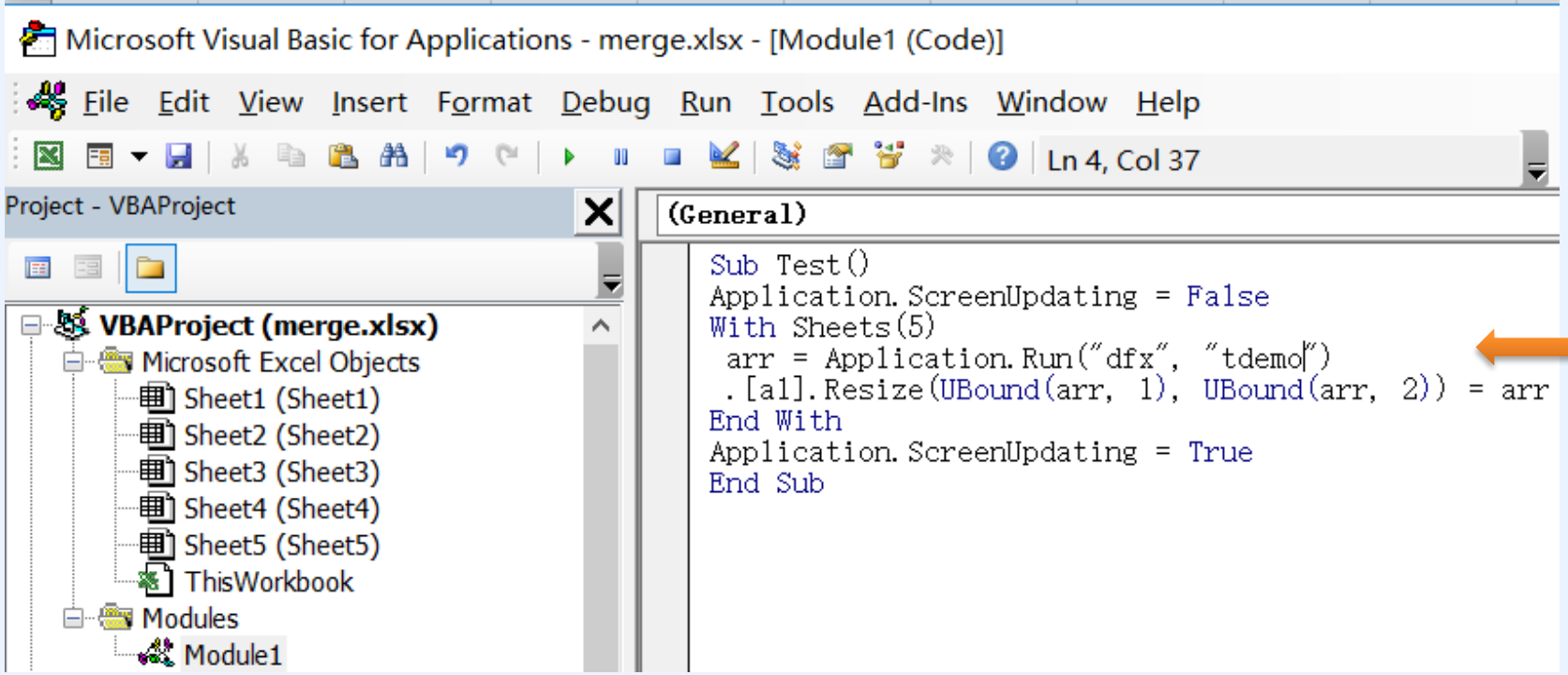


A. Application.Run() invocation

VBA script calls DFX script Application.Run()， and outputs the result to grid:

	A	B	C	D
1	age	name	pid	work
2	11	asp	100	techer
3	12	php	200	manager
4	42	jsp	300	java
5	43	java	400	help

Display result



Call dfx script

B. Application.ExecuteExcel4Macro() invocation

The return value of the DFX script called above is a list. If the return value of DFX is a single value, the ExecuteExcel4Macro interface can be used. The VBA script implementation code is as follows:

```
Sub Test()
  With Sheets(5)
    Cells(6, 1) = Application.ExecuteExcel4Macro("dfx(""tdemo"")")
  End With
End Sub
```

Note: The quotation mark in the passing parameter string of DFX interface needs to be replaced by double quotation mark.

7 Summary

After excel integrates esProc, users can implement DFX scripts with different functions through SPL language according to their own business requirements. Excel calls these DFX scripts, and SPL returns the calculated data, which can be further processed in Excel or presented after report processing, and can realize different application requirements interactively.

In the face of various complex business logic, different data source processing, big data computing, etc., esProc has its own unique features, which can also make up for the shortcomings of Excel in these aspects. Compared with VBA development, SPL language has stronger adaptability, more professional business processing and easier to use. At the same time, with the help of the ease of use of Excel operation, reporting and visualization functions, the two can realize complementary advantages, give full play to their own advantages, and make the data presentation more professional, intuitive and colorful.