Something like a function/method in batch files?

52

8

is there anything that mimicks a method like one knows it from Java, C# etc.? I have 5 lines of commands in a batch file, those 5 lines are used at more than one place inside the batch file. I can't use a goto, because depending on the errorlevel created by those 5 lines I have different actions that follow. I tried putting my 5 lines inside a batch file 5lines.bat, but the original batch file original.bat only calls 5lines.bat and doesn't execute the commands after the call to 5lines.bat): That's how my original.bat looks like:

```
5lines.bat
echo this gets never called, how to make sure this gets called?
```

There's no exit or something like this in 5lines.bat! How can I make sure the line after 5lines.bat gets called?

windows command-line batch-file

asked Apr 13 '12 at 22:37

stefan.at.wpf

6,159 25 102 174

8 Answers

20

Placing the reusable functions into a separate batch file would certainly work to simulate a function.

The catch is that you have to use the `call` command in order to ensure that control returns to the caller after the second batch file finishes executing.

```
call 5lines.bat
echo this will now get called
```

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

69

You could use the call command :

```
call:myDosFunc
```

And then define the function this way :

```
:myDosFunc    - here starts the function
echo.  here the myDosFunc function is executing a group of commands
echo.  it could do a lot of things
goto:eof
```

Source : Batch Functions

answered Apr 13 '12 at 22:42

Erwald
1,586 1 10 19

Thank you too, the call command was indeed what I was missing. – stefan.at.wpf Apr 13 '12 at 23:30

6 But how does it work? What's the magic in that `goto :eof` ? – Calmarius Nov 21 '14 at 16:26

4 `goto :eof` is the same as `exit /b` , both returns just after the calling `CALL` – jeb Jul 1 '15 at 15:26

If I understand correctly, this is like you called this same batch, beginning at the `:myDosFunc` line. – insan-e Nov 30 '17 at 22:08

@jeb it doesn't do what it means? No need to create your own return points? – Sandburg Feb 1 at 11:39

21

Just for completeness, you can also pass parameters to the function:

Function call

```
call :myDosFunc 100 "string val"
```

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
echo. Got Param#2 %~2
goto :eof
```

answered Jan 27 '18 at 11:32

Shital Shah

28k 6 118 103

14

Solution:

```
@ECHO OFF

call:header Start Some Operation

... put your business logic here
... make sure EXIT below is present
... so you don't run into actual functions without the call

call:header Operation Finished Successfully

EXIT /B %ERRORLEVEL%

:: Functions

:header
ECHO ================================================
ECHO %*
ECHO ================================================
EXIT /B 0
```

Important to put EXIT /B at the end of each function, as well as before function definitions start, in my example this is:

EXIT /B %ERRORLEVEL%

answered Jul 1 '15 at 15:23

DmitrySemenov

3.566 3 29 41

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

5

Alternatively, you could put the common lines into another batch file that you call from the main one

answered Apr 13 '12 at 22:43

Attila

23.1k 2 34 47

Great link! (though I ended up simply using call 5lines.bat) – stefan.at.wpf Apr 13 '12 at 23:30

2

Here's a 'hack' that will allow you to have "anonymous" functions in batch files:

```
@echo off
setlocal
set "anonymous=/?"

:: calling the anonymous function
call :%%anonymous%% a b c 3>&1 >nul

:: here the anonymous function is defined
if "%0" == ":%anonymous%" (
  echo(
  echo Anonymous call:
  echo %%1=%1 %%2=%2 %%3=%3
  exit /b 0
)>&3
::end of the anonymous function
```

The anonymous function block should be placed right after the call statement and must end with exit statement

the trick is that `CALL` internally uses `GOTO` and then returns to the line where the `CALL` was executed. With the double expansion GOTO help message is triggered (with `%%/?%%` argument) and then continues the script. But after it is finished it returns to the `CALL` - that's why the if statement is needed.

edited Feb 10 '17 at 16:24

answered Jan 14 '17 at 15:44

npocmaka

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

1

edited Oct 16 '16 at 15:52

answered Apr 22 '14 at 14:47

jwfearn

18.6k 22 82 112

1

I'm not sure if it was obvious from other answers but just to be explicit I'm posting this answer. I found other answers helpful in writing below code.

```
echo what
rem the third param gives info to which label it should comeback to
call :myDosFunc 100 "string val" ComeBack

:ComeBack
echo what what
goto :eof

:myDosFunc
echo. Got Param#1 %~1
echo. Got Param#2 %~2
set returnto=%~3
goto :%returnto%
```

answered Aug 14 '18 at 7:41

pasha

952 10 27

2 Why use `goto :%returnto%` ? `goto :eof` or just "end of file" will automatically return to the line next to `call ...` (that's the very purpose of `call`) and doesn't leave a orphaned JumpBackAddress on the stack. Too many of them will break your script with an errormessage. – Stephan Aug 14 '18 at 8:15

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook