Macros

What can I use the CALL EXECUTE routine for?

In code which has no Macro Language statements consideration need only be given to the two phases of processing:

- Compilation
- Execution

Code will be executed in a linear fashion, from top to bottom.

When Macro Language statements are included in the code additional timing phases must be considered:

- Macro Language Compilation
- Macro Language Execution
- Compilation
- Execution

Code may no longer execute in a simple linear fashion, as the Macro Language Compilation and Execution phases are performed during DATA step Compilation. (Execution only takes place after the Step Boundary e.g. RUN; is encountered.)

Submitting the code:

```
data _null_ ;
  set sashelp.class ;
    if sex = 'M' then
    do ;
      %put Boy ;
    end ;
    else
    do ;
      %put Girl ;
    end ;
run ;
```

will generate the LOG:

```
1    data _null_ ;
2      set sashelp.class ;
3        if sex = 'M' then
4        do ;
5          %put Boy ;
Boy
6        end ;
7        else
8        do ;
```

```
9          %put Girl ;
Girl
10       end ;
11   run ;
```

Note that each of the %PUT statements are compiled and executed un-conditionally and immediately, leaving the DATA Step syntax:

```
data _null_ ;
  set sashelp.class ;
    if sex = 'M' then
    do ;

    end ;
    else
    do ;

    end ;
run ;
```

to be processed.

The CALL EXECUTE routine allows the Macro Language statements to be executed conditionally at the Step Boundary.

Amending the code:

```
data _null_ ;
  set sashelp.class ;
    if sex = 'M' then
    do ;
      call execute('%put Boy ;') ;
    end ;
    else
    do ;
      call execute('%put Girl ;') ;
    end ;
run ;
```

Will generate the LOG:

```
12   data _null_ ;
13     set sashelp.class ;
14       if sex = 'M' then
15       do ;
16         call execute('%put Boy ;') ;
17       end ;
18       else
19       do ;
20         call execute('%put Girl ;') ;
21       end ;
22   run ;

Boy
Girl
Girl
Girl
Boy
Boy
Girl
Girl
Boy
Boy
Girl
Girl
Girl
Girl
Boy
Boy
Boy
Boy
Boy
```

giving the intended result.

To further demonstrate the order in which statements are executed, consider the code:

```
data code ;
 %let name = Alan ;
 instr  = '%put Name is &name..;' ;
 instr2 = "%put Name is &name..;" ;
 call execute (instr) ;
 %let name = David ;
run ;

proc print ;
run ;
```

generates the LOG:

```
1
2    data code ;
3     %let name = Alan ;
4     instr  = '%put Name is &name..;' ;
5     instr2 = "%put Name is &name..;" ;
Name is Alan.
6     call execute (instr) ;
7     %let name = David ;
8    run ;

Name is David.
```

and the report:

```
 Obs            instr            instr2

   1     %put Name is &name..;
```

Knowing that Macro Language triggers (& and %) are not resolved inside single quotes, but will be resolved inside double quotes, the variable instr is stored with the value exactly as typed, but the Macro Language statement within the double quotes on the second assignment statement is executed immediately generating the Name is Alan. line in the LOG and leaving a missing value for the instr2 variable.

At the Step Boundary (run ;) the syntax stored in the instr variable is returned to the Input Stack to be processed, by which time the value of the Macro Variable is David, and therefore generates the Name is David. line in the LOG.

The CALL EXECUTE routine also allows for conditional calls to a Macro Program, based on dataset values.

Attempting to run the code which uses relative row counts to call the Macro Program with differing parameters:

```
%macro calc_mean (ds,var_vbl,class_vbl) ;
  proc means data = &ds mean ;
```

```
    var &var_vbl ;
    %if &class_vbl ne %then
    %do ;
    class &class_vbl ;
    %end ;
  run ;
%mend calc_mean ;

data _null_ ;
  set sashelp.class end = final ;
  if sex = 'M' then boys + 1 ;
    else girls + 1 ;
  if final then
  do ;
   if (girls = 0 or boys = 0) then
   do ;
     %calc_mean(sashelp.class,age)
   end ;
   else
   do ;
     %calc_mean(sashelp.class,age,sex)
   end ;
  end ;
run ;
```

generates ERRORs as the Macro Programme calls are executed immediately returning PROC step syntax within the DO - END block. Use the CALL EXECUTE routine to return just one of the Macro Programme calls at the Step Boundary, after the DATA Step has completed execution.

```
data _null_ ;
  set sashelp.class end = final ;
  if sex = 'M' then boys + 1 ;
    else girls + 1 ;
  if final then
  do ;
   if (girls = 0 or boys = 0) then
   do ;
     call execute('%calc_mean(sashelp.class,age)') ;
   end ;
   else
   do ;
     call execute('%calc_mean(sashelp.class,age,sex)') ;
   end ;
  end ;
run ;
```

Macros

Unique solution ID: #1016
Author: Alan D Rudland
Last update: 2017-03-27 16:45