

Working with the Function Builder



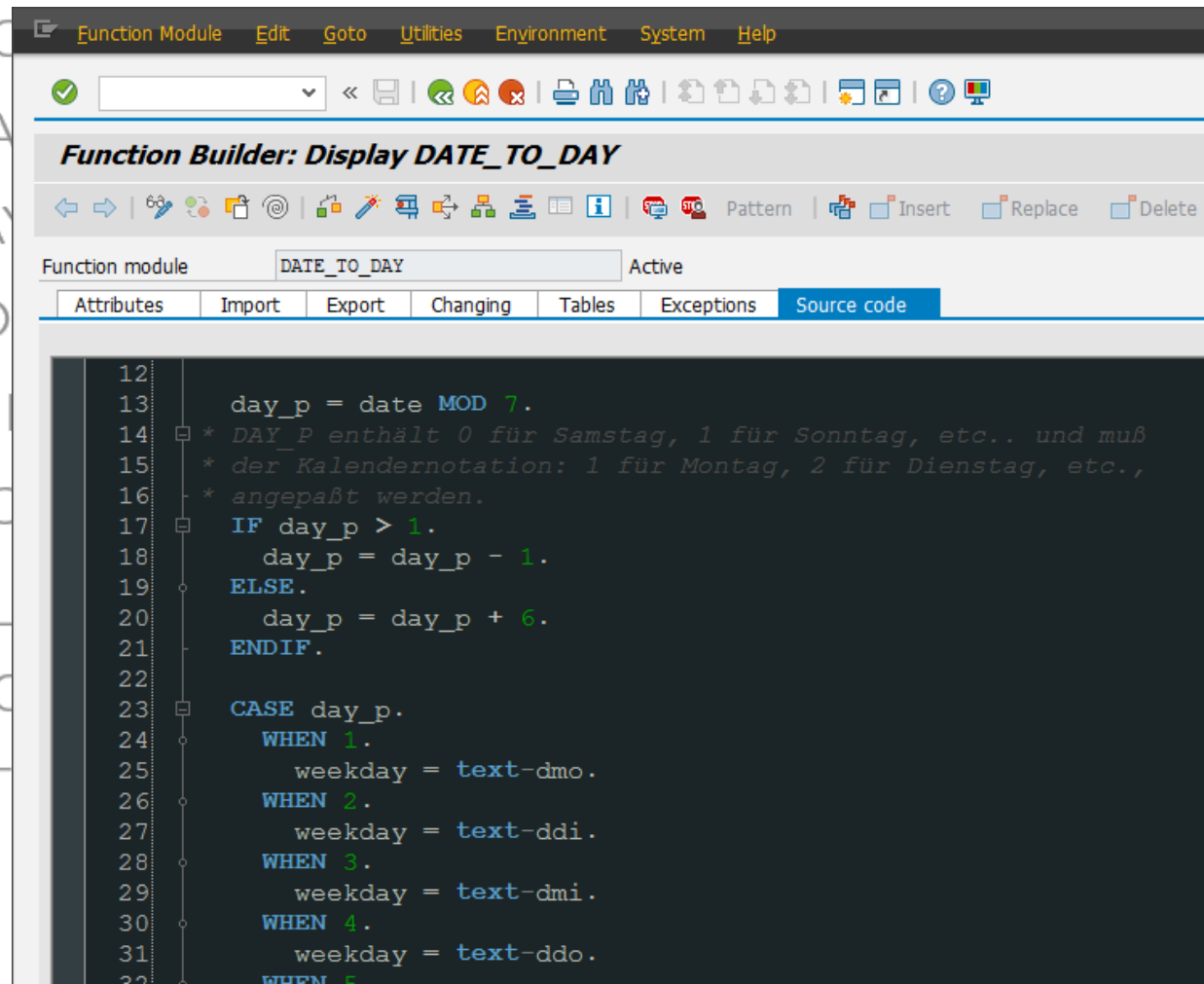
Alex Gönczy

ABAP DEVELOPER

@abapmentor <http://abapmentor.expertise-team.com/>



TRANSACTION_CALL_VIA_RFC | RS_PROGRAM_CHECK_NAME | RS_CORR_INSERT | REPS_O
RS_DELETE_PROGRAM | RS_ACCESS_PERMISSION | CALCULATE_DATE | **DATE_TO_DAY** | DATE
RP_CALC_DATE_IN_INTERVAL | DAY_ATTRIBUTES_GET | MONTHS_BETWEEN_TWO_DATES | E
HR_HK_DIFF_BT_2_DATES | FIMA_DAYS_AND_MO
HRGPBS_HESA_DATE_FORMAT | SD_CALC_DURA
HR_99S_INTERVAL_BETWEEN_DATES | LAST_DAY
DAY_IN_WEEK | SD_DATETIME_DIFFERENCE | HO
DATE_CONVERT_TO_FACTORYDATE | F4_DATE |
REUSE_ALV_GRID_DISPLAY | REUSE_ALV_FIELDC
CHANGEDOCUMENT_READ | CLOI_PUT_SIGN_IN
CONVERSION_EXIT_ALPHA_OUTPUT | READ_EXC
SAPGUI_PROGRESS_INDICATOR | FILENAME_GET



The screenshot shows the SAP Function Builder interface for the **DATE_TO_DAY** function. The title bar indicates it is a **Function Module**. The main title is **Function Builder: Display DATE_TO_DAY**. Below the title bar, there is a toolbar with various icons for navigation and editing. The **Function module** is set to **DATE_TO_DAY** and is marked as **Active**. The **Source code** tab is selected, showing the following ABAP code:

```
12  
13     day_p = date MOD 7.  
14     * DAY_P enthält 0 für Samstag, 1 für Sonntag, etc.. und muß  
15     * der Kalendernotation: 1 für Montag, 2 für Dienstag, etc.,  
16     * angepaßt werden.  
17     IF day_p > 1.  
18         day_p = day_p - 1.  
19     ELSE.  
20         day_p = day_p + 6.  
21     ENDIF.  
22  
23     CASE day_p.  
24         WHEN 1.  
25             weekday = text-dmo.  
26         WHEN 2.  
27             weekday = text-ddi.  
28         WHEN 3.  
29             weekday = text-dmi.  
30         WHEN 4.  
31             weekday = text-ddo.  
32         WHEN 5.  
33             weekday = text-dmi.  
34         WHEN 6.  
35             weekday = text-dmo.  
36     ENDCASE.
```

The Goal of This Module

ABAP Editor: Display Report Z_FLIGHT_BOOKING_LIST

Report Z_FLIGHT_BOOKING_LIST

BEFORE

```
23
24
25 " $. Region Determine the name of the selected day
26
27 DATA(day_of_the_week) = pfldate MOD 7.
28
29 IF day_of_the_week > 1.
30     day_of_the_week = day_of_the_week - 1.
31 ELSE.
32     day_of_the_week = day_of_the_week + 6.
33 ENDIF.
34
35 DATA(name_of_the_day) = SWITCH string(
36     day_of_the_week WHEN 1 THEN 'Monday'
37                     WHEN 2 THEN 'Tuesday'
38                     WHEN 3 THEN 'Wednesday'
39                     WHEN 4 THEN 'Thursday'
40                     WHEN 5 THEN 'Friday'
41                     WHEN 6 THEN 'Saturday'
42                     WHEN 7 THEN 'Sunday'
43 ).
44
45 "-----
46 " $. Endregion
47 "-----
```

Scope: \REGION Prepare

ABAP Editor: Display Report Z_FLIGHT_BOOKING_LIST

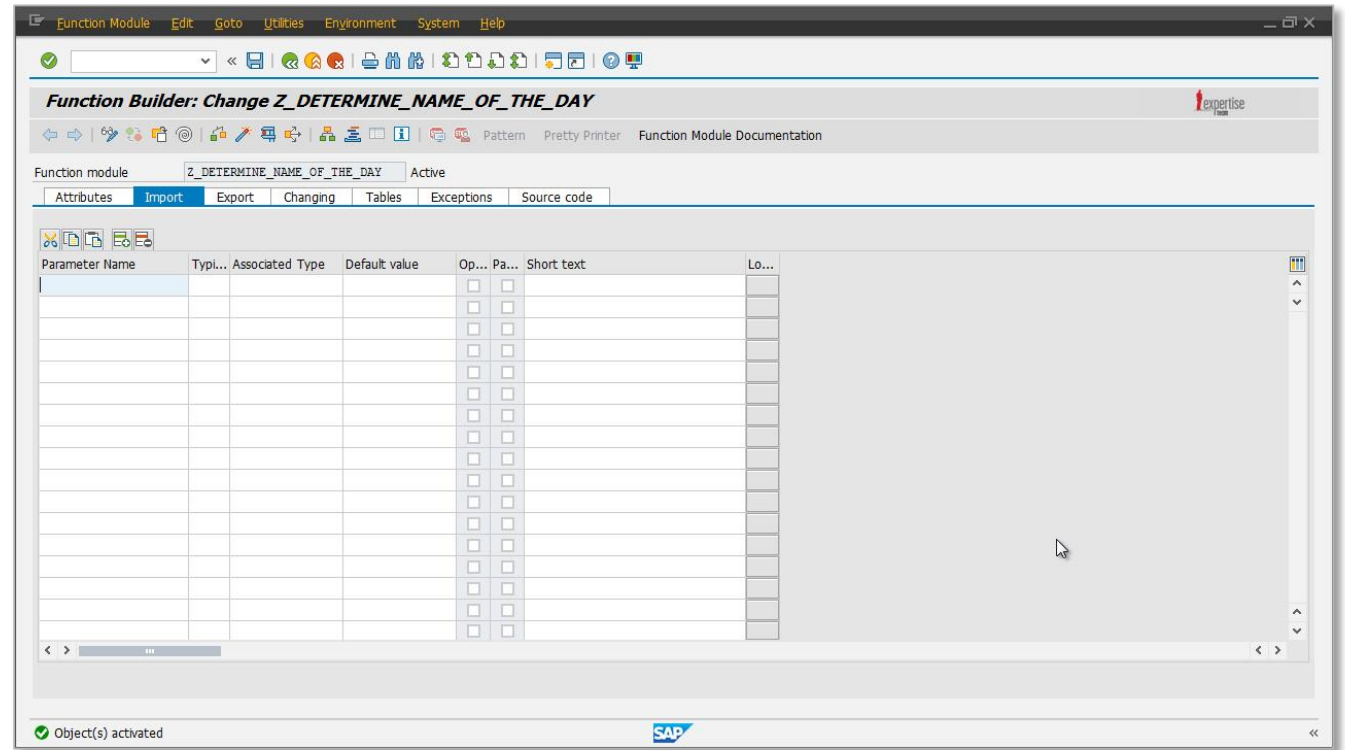
Report Z_FLIGHT_BOOKING_LIST

AFTER

```
23
24
25 " $. Region Determine the name of the selected day
26
27 DATA name_of_the_selected_date TYPE string.
28
29 CALL FUNCTION 'Z_DETERMINE_NAME_OF_THE_DAY'
30     EXPORTING
31         date = pfldate
32     IMPORTING
33         name_of_the_day = name_of_the_selected_date.
34
35 "-----
36 " $. Endregion
37 "-----
38
39 "-----
40 " $. Region Prepare the title of the ALV
41
42 DATA(formatted_date) = |{ pfldate(4) }-{ pfldate+4(2) }|
43 DATA(title_of_alv) = |Flight Bookings on { formatted_date }|
44
45 "-----
46 " $. Endregion
47 "-----
```



Creating the Function Module



Implementing the Function Module

A screenshot of the SAP Function Builder interface. The title bar reads "Function Builder: Change Z_DETERMINE_NAME_OF_THE_DAY". Below the title bar, there are tabs for "Attributes", "Import", "Export", "Changing", "Tables", "Exceptions", and "Source code". The "Source code" tab is active, displaying the following ABAP code:

```
7  * " REFERENCE(NAME_OF_THE_DAY) TYPE STRING
8  * " -----
9
10 DATA(day_of_the_week) = date MOD 7.
11
12 IF day_of_the_week > 1.
13   day_of_the_week = day_of_the_week - 1.
14 ELSE.
15   day_of_the_week = day_of_the_week + 6.
16 ENDIF.
17
18 name_of_the_day = SWITCH string(
19   day_of_the_week WHEN 1 THEN 'Monday'
20                   WHEN 2 THEN 'Tuesday'
21                   WHEN 3 THEN 'Wednesday'
22                   WHEN 4 THEN 'Thursday'
23                   WHEN 5 THEN 'Friday'
24                   WHEN 6 THEN 'Saturday'
25                   WHEN 7 THEN 'Sunday'
26 ).
27
28 ENDFUNCTION.
```

The status bar at the bottom indicates "Scope: \FUNCTION Z_DETERMINE_NAME_OF_THE_DAY" and "ABAP | Ln 18 Col 16". The SAP logo is visible in the bottom right corner.

Calling the Function Module

A screenshot of the SAP ABAP Editor interface. The title bar reads "ABAP Editor: Change Report Z_FLIGHT_BOOKING_LIST". The editor shows the source code for report Z_FLIGHT_BOOKING_LIST. The code includes comments in German, data declarations, and a function call. The status bar at the bottom indicates "ABAP | Ln 43 Col 89".

```
23
24
25 " $. Region Determine the name of the selected day
26 "
27 DATA name_of_the_selected_date TYPE string.
28
29 CALL FUNCTION 'Z_DETERMINE_NAME_OF_THE_DAY'
30   EXPORTING
31     date          = pfldate
32   IMPORTING
33     name_of_the_day = name_of_the_selected_date.
34
35 "
36 " $. Endregion
37 "
38
39 "
40 " $. Region Prepare the title of the ALV
41 "
42 DATA(formatted_date) = |({ pfldate(4) }-({ pfldate+4(2) }-({ pfldate+6(2) })|.
43 DATA(title_of_alv) = |Flight Bookings on { formatted_date }, { name_of_the_selected_date }|.
44
45 "
46 " $. Endregion
47 "
```



Next

Creating the Function Module

A screenshot of the SAP ABAP code editor. The window title is 'Report' and the file name is 'Z_FLIGHT_BOOKING_LIST'. The code is written in ABAP and includes comments in English. It defines a data type for the selected date, calls a function module to determine the day name, and prepares the title for the ALV grid.

```
23
24
25 *2. Begin: Determine the name of the selected day
26
27 DATA name_of_the_selected_date TYPE string.
28
29 CALL FUNCTION 'Z_DETERMINE_NAME_OF_THE_DAY'
30   EXPORTING
31     date           = pflightdate
32   IMPORTING
33     name_of_the_day = name_of_the_selected_date.
34
35
36 *1. Endregion
37
38
39
40 *1. Begin: Prepare the title of the ALV
41
42 DATA(formatted_date) = |({ pflightdate}) |-{ pflightdate+({2}) }-{ pflightdate+({7}) }|.
43 DATA(title_of_alv) = |Flight Bookings on | formatted_date |, | name_of_the_selected_date ||.
44
45
46 *2. Endregion
47
```

Creating the Function Module



Architecture of the Function Modules

Main Program **SAPLZ_DATE_FUNCTIONS**

Global Data Declarations **LZ_DATE_FUNCTIONSTOP**

```
FUNCTION-POOL Z_DATE_FUNCTIONS.  
TYPES ...  
DATA ...  
CONSTANTS ...
```

Function Modules **LZ_DATE_FUNCTIONSUXX**

```
INCLUDE LZ_DATE_FUNCTIONSU01.  
    " Z_DETERMINE_NAME_OF_THE_DAY  
INCLUDE LZ_DATE_FUNCTIONSU...  
    " Next Function Module
```

```
FUNCTION Z_DETERMINE_NAME_OF_THE_DAY.
```

```
  ...
```

```
ENDFUNCTION.
```

```
ENDFUNCTION.
```

```
...
```



Implementing the Function Module



Calling the Function Module



Summary



Working with the Function Builder

- How to create Function Modules
- How to call Function Modules

What are the benefits?

- No need to implement it one more time
- Reduces the development costs
- You can change it on a central place
- Improves the consistency
- Makes your source code more readable



Last Module

Working with the Class Builder



What are the benefits?

- No need to implement it one more time
- Reduces the development costs
- You can change it on a central place
- Improves the consistency
- Makes your source code more readable

