MVC in ABAP OO, part 1 the Model

Applies to:

All SAP NetWeaver releases. For more information, visit the ABAP homepage.

Summary

Many of my students were not certain on how to use the things they've learnt in the BC401(Abap Objects) course on programming. Here is a practical example of how we can set up data processing in a way that brings out the benefits of OO, I will do that with the MVC design pattern, but in a different manner than other examples that are published. I will (briefly) discuss the structure of the design, for an overall understanding, the code is more detailed.

Author: Robin Fillerup
Company: SAP
Created on: 21 November 2011

Author Bio

Robin Fillerup is in software development for over 20 years. In the early 90s he started working on OO projects. In 1998 he started as a teacher at SAP Netherlands.

Table of Contents

Purpose of MVC

Purpose of MVC is to decouple user interface and data processing. It should make software simpler and thus easier to maintain. It consists of a data processing component(Model) that is 'reusable' because of its independency of the user interface(View). So user interface and data processing are decoupled from each other. They are 'connected' by the controller, a piece of software sitting between user interface and data processing. On the model site we find data retrieval, data processing and data validation. The view is just responsible for displaying the state of the data, and make sure that the user has interaction possibilities, via menu, buttons, etc. The controller is responsible for the interaction between view and model.

General structure/setup

There are examples of the use of MVC in SAPGUI programs that gives the impression that UI is shown by the controller. In those examples a controller object is calling some method (display, show_) of a view object, but as we all know: classes cannot hold screens, so this method in the view object is calling some screen of a function group thereby 'blurring' what really is happening in a SAPGUI program: a global variable is transported to(PBO) and from(PAI) the screen. Often we see that this setup leads to more complexity (simplicity is also an objective in this example).
I will define OO interfaces with the needed definitions (method and event definitions, data types). The implementing class is instantiated via a **factory class**, this is desirable because then we have decoupling in a flexible manner, the controller class only knows the interfaces and doesn't know anything about implementing classes. So if a different implementation should be used (for example a subclass), we can arrange that in the factory class.
The **view** will be a SAPGUI program, in there the declaration of global variables: for the business data, and for screen/ui elements(ALV) and the controller object The SAPGUI screen will call methods of the controller object, this will be done initially, and every PBO/PAI.
The **controller** will have access to the global variables of the SAPGUI program via data references (type ref to data), that makes it possible for the controller to manipulate the content of these variables, and pass user input to the model(s). For each screen the controller will have 2 methods, one for handling PBO the other for handling PAI of the screen. The controller object will have references to model objects, instantiated via a factory class.
The controller object makes the interaction between SAPGUI program and model object, for example: if the user presses the 'Search' button in a screen, the PAI of the screen will call the appropriate PAI method (PAI_100) of the controller object, pass the command code, the controller method will call a 'Search' method of the model object(s), the controller will make sure that the model data is transferred to the view (global variables in SAPGUI program).

The types of models

As stated the model is responsible for data processing, that includes reading, storing, processing and validating. This data is to be shown in an UI. In SAPGUI screens almost always display the contents of either abap structure or abap internal tables, so it make sense that we have 2 different kind of models:

- Single entity model: the model object holds a structure. The controller will access this structure via a setter and getter.
- Multiple entity model, model instance holds an internal table. The controller will have access to this internal table either via data referencing or a getter method.

Both type of models will need generic processing, for example: SAVE, GET_MODEL_STATE, GET_ERRORS, GET_TABLE, SET_TABLE_REF, REVERT_CHANGES, these are defined in 2 interfaces:

- ZIF_RF_SINGLE_ENTITY_MODEL
- ZIF_RF_MULTIPLE_ENTITY_MODEL

These interfaces are implemented in 2 (super) classes:

- ZCL_RF_SINGLE_ENTITY_MODEL
- ZCL_RF_MULTIPLE_ENTITY_MODEL

There will also specific processing, for example, READ_FLIGHTS_BY_AIRLINE, READ_FLIGHTS_BY_CONNECTION, CHECK_STRUCTURE etc, these are also defined in different 2 interfaces, and implemented in 2 (Sub)Classes. In my demo I have 2 specific single entity model interfaces:

- ZIF_RF_CONNECTION_MODEL
- ZIF_RF_FLIGHT_MODEL

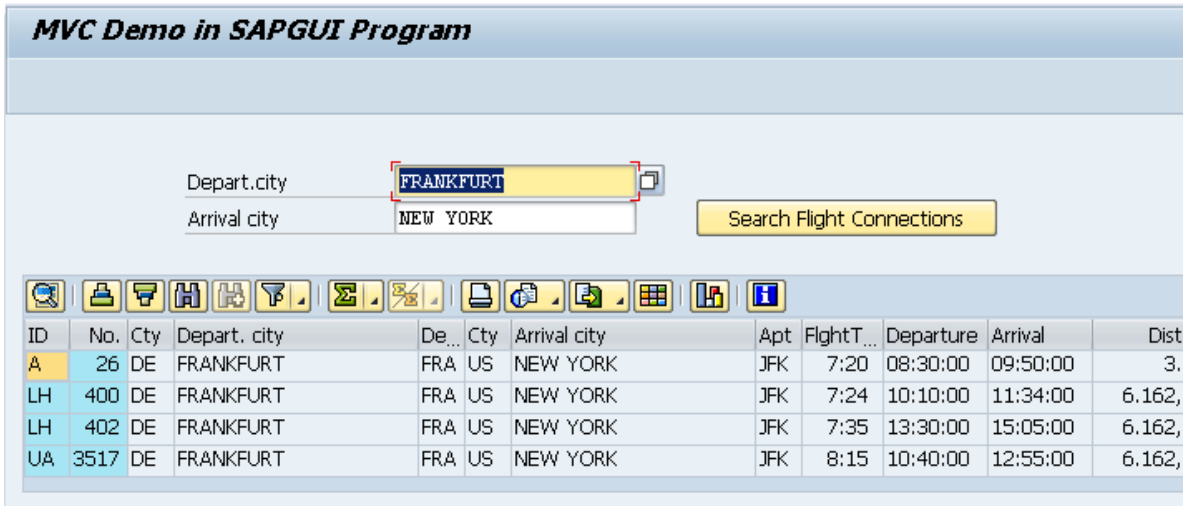
And also 2 specific multiple entity model interfaces:

- ZIF_RF_CONNECTIONS_MODEL
- ZIF_RF_FLIGHTS_MODEL

Model Interaction

Model objects needs interactions, for example if the user double clicks on a line in the ALV, details needs to be displayed either in another ALV or in input fields of a screen (both variants are in the demo). If the content of an input field changes, these changes must also be displayed in the ALV. These interactions are done via methods and events. There are 3 interactions in my demo that will follow.

Model interaction part 1:



ID	No.	Cty	Depart. city	De...	Cty	Arrival city	Apt	FlghtT...	Departure	Arrival	Dist
A	26	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:20	08:30:00	09:50:00	3.
LH	400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,
LH	402	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:35	13:30:00	15:05:00	6.162,
UA	3517	DE	FRANKFURT	FRA	US	NEW YORK	JFK	8:15	10:40:00	12:55:00	6.162,

The ‘Search Flight Connections’ button triggers PAI.

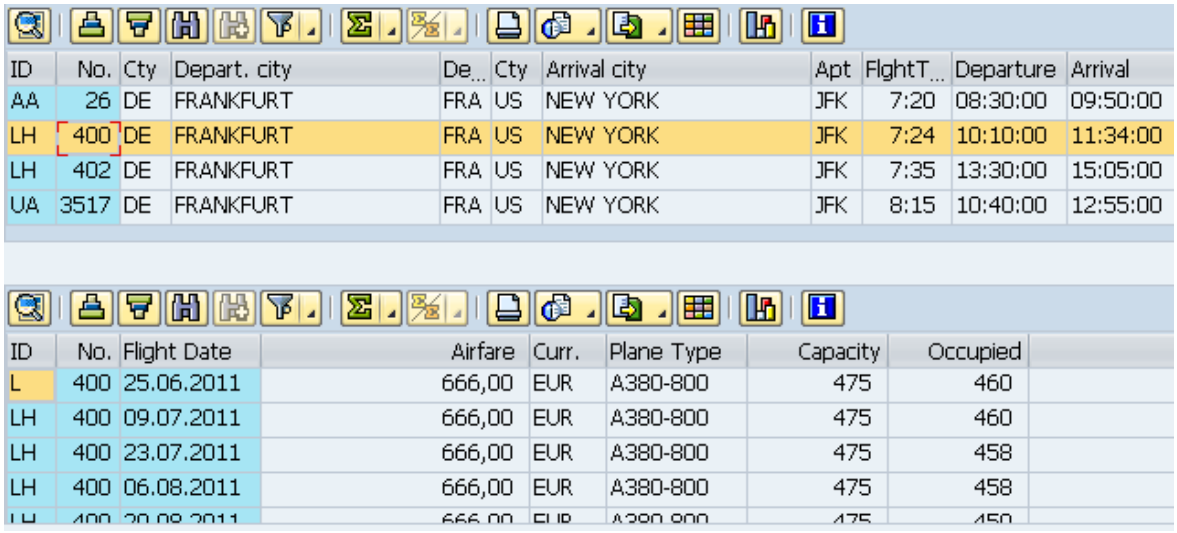
PAI:

```
  go_controller->pai_0100( ok_code ).

pai_0100( ):
    "me->mr_flight refs to SFLIGHT screen structure(TABLES)in Prog
    me->mo_flight_model->set_flight_structure(
                           me->mr_flight->* ). "Screen fields to Model!

    CASE iv_ok_code.
      WHEN 'BACK' OR 'RW' OR '%EX'.
        LEAVE PROGRAM .
      WHEN 'SEARCH_CONNECTIONS'.
       "me->mr_connection refs to SPFLI screen structure(TABLES)in Prog
        me->mo_connections_model->read_connections_by_dep_dest(
           EXPORTING iv_cityfrom = me->mr_connection->cityfrom
                     iv_cityto = me->mr_connection->cityto ).
        me->mo_alv_connection->refresh_table_display( ).
      WHEN 'SAVE_FLIGHT'.
*     To Be Done later..
    ENDCASE.
```

Model interaction part 2:



ID	No.	Cty	Depart. city	De...	Cty	Arrival city	Apt	FlghtT...	Departure	Arrival
AA	26	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:20	08:30:00	09:50:00
LH	400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00
LH	402	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:35	13:30:00	15:05:00
UA	3517	DE	FRANKFURT	FRA	US	NEW YORK	JFK	8:15	10:40:00	12:55:00

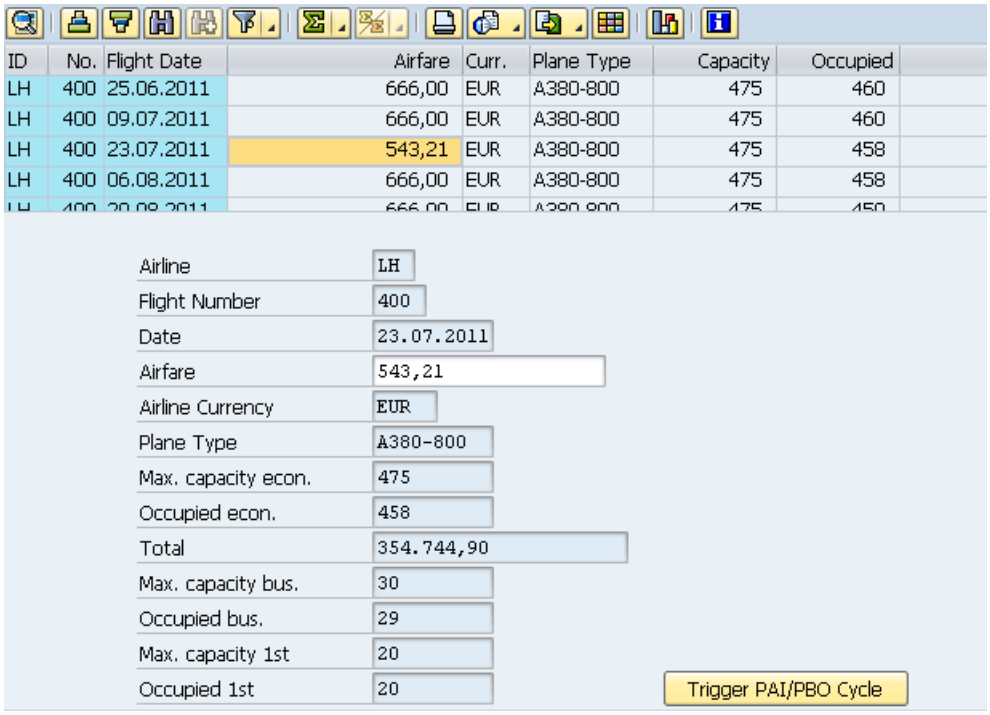
ID	No.	Flight Date	Airfare	Curr.	Plane Type	Capacity	Occupied
L	400	25.06.2011	666,00	EUR	A380-800	475	460
LH	400	09.07.2011	666,00	EUR	A380-800	475	460
LH	400	23.07.2011	666,00	EUR	A380-800	475	458
LH	400	06.08.2011	666,00	EUR	A380-800	475	458

If the user double clicks on the top alv, the double_click event is handled in the on_double_click_connection method of the controller object.

```
on_double_click_connection( ):
    DATA lv_tabix TYPE sytabix.
    lv_tabix = e_row-index.
    "Multi entity Model delivers the single entity model:
    me->mo_connection_model ?=
             me->mo_connections_model->get_connection_model( lv_tabix ).
    "single entity model delivers the multiple entity model
    "(here the Flights of one Connection):
    me->mo_flights_model = me->mo_connection_model->get_flights_model( ).
   "me->mr_flights is referencing to gt_sflight in  sapgui program
    me->mo_flights_model->set_table_ref( me->mr_flights ).
    me->mo_alv_flight->refresh_table_display( ).
```

In case of change the single entity model raise its EV_SINGLE_ENTITY_CHANGED event, this is handled in the corresponding multiple entity model in the ON_SINGLE_ENTITY_MODEL_CHANGED method, so the right line of the internal table can be modified. The multiple entity model then raises than its EV_MULTIPLE_ENTITY_MODEL_CHANGED event, this is handled by the controller in its ON_MULTI_ENTITY_MODEL_CHANGED, this handler then calls the REFRESH_TABLE_DISPLAY method of the appropriate ALV.

Model interaction part 3:



If the user double click a line in the second alv, the contents of this line are shown in input fields. This is done via the on_double_click_flight method of the controller.

```
on_double_click_flight( ):
    DATA lv_tabix TYPE sytabix.
    lv_tabix = e_row-index.
    me->mo_flight_model = me->mo_flights_model->get_flight_model( lv_tabix ).
    me->mr_flight->* = me->mo_flight_model->get_flight_structure(   ).
    "Trigger PAI/PBO for data transport to screen
    cl_gui_cfw=>set_new_ok_code( 'DUMMY' ).
```

The user than changes contents of the price field, and presses the ‘Trigger PAI/PBO Cycle’.

```
PAI:
  go_controller->pai_0100( ok_code ).

pai_0100( ):
    "me->mr_flight refs to SFLIGHT screen structure(TABLES)in Prog
    me->mo_flight_model->set_flight_structure(
                          me->mr_flight->* ). "Screen fields to Model!
set_flight_structure( ):
* This method can be made generic in the superclass!
  FIELD-SYMBOLS <fs_structure> TYPE any.

  IF is_flight <> me->ms_flight.
    ASSIGN me->mr_structure_in_previous_state->* TO <fs_structure> .
    <fs_structure> = me->ms_flight.
    me->ms_flight = is_flight. "TODO! Error Handling! Next Blog? Vote please..
    RAISE EVENT ev_single_entity_model_changed
           EXPORTING es_structure = me->ms_flight.
  ENDIF.
```

Event ev_single_entity_model_changed is handled in on_single_entity_model_changed of zcl_rf_multiple_entity_model.

```
on_single_entity_model_changed( ):
  FIELD-SYMBOLS <fs_table> TYPE ANY TABLE.

  ASSIGN me->mr_table->* TO <fs_table>.
  MODIFY TABLE <fs_table> FROM es_structure.
  RAISE EVENT EV_MULTI_ENTITY_MODEL_CHANGED.

Event ev_multi_entity_model_changed is handled in on_multi_entity_model_changed of the
controller object (class: zcl_rf_sapgui_controller).
on_multi_entity_model_changed( ):
  DATA ls_stable TYPE  lvc_s_stbl.

  ls_stable-row = abap_true.

  IF sender = me->mo_flights_model.
    me->mo_alv_flight->refresh_table_display( EXPORTING is_stable = ls_stable   ).
  ENDIF.
```

Conclusion and outlooks

What we have seen so far is a flexible decoupling of user interface and model, via a factory class. All data processing is done, or will be done in future, at the model side, that includes reading, storing, association to other models, validating.

Of course this is just the first step, there are some issues that I will address in future blogs:

- Advanced Controller topics: error handling
- Transaction handling

I will (depending on remarks etc) rework the code, and make the transport of all code available later.

```
*---------------------------------------------------------------------*
*       CLASS ZCL_RF_MODEL_FACTORY DEFINITION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_model_factory DEFINITION
  PUBLIC
  CREATE PUBLIC .

  PUBLIC SECTION.
*"* public components of class ZCL_RF_MODEL_FACTORY
*"* do not include other source files here!!!

    CLASS-METHODS get_flight_model
      RETURNING
        value(ro_flight_model) TYPE REF TO zif_rf_flight_model .
    CLASS-METHODS get_flights_model
      RETURNING
        value(ro_flights_model) TYPE REF TO zif_rf_flights_model .
    CLASS-METHODS get_connection_model
      RETURNING
        value(ro_connection_model) TYPE REF TO zif_rf_connection_model .
    CLASS-METHODS get_connections_model
      RETURNING
        value(ro_connections_model) TYPE REF TO zif_rf_connections_model .
  PROTECTED SECTION.
  PRIVATE SECTION.
*"* private components of class ZCL_RF_MODEL_FACTORY
*"* do not include other source files here!!!
ENDCLASS.                    "ZCL_RF_MODEL_FACTORY DEFINITION



*---------------------------------------------------------------------*
*       CLASS ZCL_RF_MODEL_FACTORY IMPLEMENTATION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_model_factory IMPLEMENTATION.


  METHOD get_connections_model.

    CREATE OBJECT ro_connections_model TYPE zcl_rf_connections_model.

  ENDMETHOD.                    "GET_CONNECTIONS_MODEL


  METHOD get_connection_model.

    CREATE OBJECT ro_connection_model TYPE zcl_rf_connection_model.

  ENDMETHOD.                    "get_connection_model


  METHOD get_flights_model.
```

```
    CREATE OBJECT ro_flights_model TYPE zcl_rf_flights_model.
  ENDMETHOD.                    "GET_FLIGHTS_MODEL


  METHOD get_flight_model.

    CREATE OBJECT ro_flight_model TYPE zcl_rf_flight_model.
  ENDMETHOD.                    "GET_FLIGHT_MODEL


ENDCLASS.                    "ZCL_RF_MODEL_FACTORY IMPLEMENTATION



*"* components of interface ZIF_RF_SINGLE_ENTITY_MODEL
INTERFACE zif_rf_single_entity_model
  PUBLIC .


  TYPES ty_edit_mode TYPE char12 .
  TYPES ty_model_state TYPE char12 .

  CONSTANTS co_edit_mode_change TYPE ty_edit_mode VALUE 'CHANGE'. "#EC NOTEXT
  CONSTANTS co_edit_mode_display TYPE ty_edit_mode VALUE 'DISPLAY'. "#EC NOTEXT
  CONSTANTS co_model_state_new TYPE ty_model_state VALUE 'NEW'. "#EC NOTEXT
  CONSTANTS co_model_state_notchanged TYPE ty_model_state VALUE 'NOTCHANGED'. "#EC NOTEXT
  CONSTANTS co_model_state_changed TYPE ty_model_state VALUE 'CHANGED'. "#EC NOTEXT
  CONSTANTS co_model_state_saved TYPE ty_model_state VALUE 'SAVED'. "#EC NOTEXT

  EVENTS ev_single_entity_model_changed
    EXPORTING
      value(es_structure) TYPE data .

  METHODS read_by_key
    IMPORTING
      !is_structure TYPE data .
  METHODS get_errors
    RETURNING
      value(rt_model_error) TYPE zcl_rf_model_error=>ty_t_model_error .
  METHODS check_structure .
  METHODS save .
  METHODS revert .
  METHODS set_edit_mode
    IMPORTING
      !iv_edit_mode TYPE ty_edit_mode .
  METHODS get_model_state
    RETURNING
      value(re_model_state) TYPE ty_model_state .
  METHODS get_struct_descr
    RETURNING
      value(ro_structdescr) TYPE REF TO cl_abap_structdescr .
ENDINTERFACE.                    "ZIF_RF_SINGLE_ENTITY_MODEL
```

```
*"* components of interface ZIF_RF_MULTIPLE_ENTITY_MODEL
INTERFACE zif_rf_multiple_entity_model
  PUBLIC .


  EVENTS ev_multi_entity_model_changed .

  METHODS get_table_ref
    RETURNING
      value(rr_table) TYPE REF TO data .
  METHODS get_table_descr
    RETURNING
      value(ro_tabledescr) TYPE REF TO cl_abap_tabledescr .
  METHODS on_single_entity_model_changed
    FOR EVENT ev_single_entity_model_changed OF zif_rf_single_entity_model
    IMPORTING
      !es_structure
      !sender .
  METHODS get_table
    EXPORTING
      value(et_table) TYPE data .
  METHODS set_table_ref
    IMPORTING
      !ir_table TYPE REF TO data .
ENDINTERFACE.                    "ZIF_RF_MULTIPLE_ENTITY_MODEL



*"* components of interface ZIF_RF_CONNECTION_MODEL
INTERFACE zif_rf_connection_model
  PUBLIC .

  INTERFACES zif_rf_single_entity_model .

  TYPES ty_s_connection TYPE spfli .

  METHODS get_flights_model
    RETURNING
      value(ro_flights_model) TYPE REF TO zif_rf_flights_model .
  METHODS get_carrid
    RETURNING
      value(rv_carrid) TYPE s_carr_id .
  METHODS get_connid
    RETURNING
      value(rv_connid) TYPE s_conn_id .
ENDINTERFACE.                    "ZIF_RF_CONNECTION_MODEL

*"* components of interface ZIF_RF_CONNECTIONS_MODEL
INTERFACE zif_rf_connections_model
  PUBLIC .


  INTERFACES zif_rf_multiple_entity_model .

  TYPES ty_s_connection TYPE spfli .
  TYPES ty_t_connection TYPE STANDARD TABLE OF spfli
```

```
                        WITH NON-UNIQUE KEY carrid connid .

  METHODS read_connections_by_dep_dest
    IMPORTING
      iv_cityfrom TYPE spfli-cityfrom
      iv_cityto TYPE spfli-cityto .
  METHODS get_connection_model
    IMPORTING
      iv_index TYPE sytabix
    RETURNING
      value(ro_connection_model) TYPE REF TO zif_rf_connection_model .
ENDINTERFACE.                    "ZIF_RF_CONNECTIONS_MODEL


*"* components of interface ZIF_RF_FLIGHT_MODEL
INTERFACE zif_rf_flight_model
  PUBLIC .


  INTERFACES zif_rf_single_entity_model .
  TYPES ty_s_flight TYPE sflight .

  METHODS get_bookings_model .
  METHODS get_passengers_model .
  METHODS set_flight_structure
    IMPORTING
      is_flight TYPE ty_s_flight .
  METHODS get_flight_structure
    RETURNING
      value(rs_flight) TYPE ty_s_flight .
ENDINTERFACE.                    "ZIF_RF_FLIGHT_MODEL

*"* components of interface ZIF_RF_FLIGHTS_MODEL
INTERFACE zif_rf_flights_model
  PUBLIC .

  INTERFACES zif_rf_multiple_entity_model .

  TYPES ty_s_flight TYPE sflight .
  TYPES:
    ty_t_flight TYPE STANDARD TABLE OF sflight
              WITH NON-UNIQUE KEY carrid connid fldate .

  METHODS read_flights_by_airline
    IMPORTING
      im_carrid TYPE s_carr_id .
  METHODS read_flights_by_connection
    IMPORTING
      iv_carrid TYPE s_carr_id
      iv_connid TYPE s_conn_id .
  METHODS get_flight_model
    IMPORTING
      iv_index TYPE numeric
    RETURNING
      value(ro_flight_model) TYPE REF TO zif_rf_flight_model .
ENDINTERFACE.                    "ZIF_RF_FLIGHTS_MODEL
```

```
CLASS zcl_rf_single_entity_model DEFINITION
  PUBLIC
  ABSTRACT
  CREATE PROTECTED

  GLOBAL FRIENDS zcl_rf_model_factory .

  PUBLIC SECTION.
*"* public components of class ZCL_RF_SINGLE_ENTITY_MODEL
*"* do not include other source files here!!!

    INTERFACES zif_rf_single_entity_model
        ABSTRACT METHODS read_by_key .

    METHODS constructor .
  PROTECTED SECTION.
*"* protected components of class ZCL_RF_SINGLE_ENTITY_MODEL
*"* do not include other source files here!!!

    DATA mv_edit_mode TYPE ty_edit_mode .
    DATA mv_model_state TYPE ty_model_state .
    CLASS zcl_rf_model_error DEFINITION LOAD .
    DATA mt_model_error TYPE zcl_rf_model_error=>ty_t_model_error .
    DATA mr_structure TYPE REF TO data .
    TYPE-POOLS abap .
    DATA mv_needs_checking TYPE abap_bool .
    DATA mo_structdescr TYPE REF TO cl_abap_structdescr .
    DATA mr_structure_in_previous_state TYPE REF TO data .

    METHODS connect_spec_gen_structures
      CHANGING
        !cs_structure TYPE data .
  PRIVATE SECTION.
*"* private components of class ZCL_RF_SINGLE_ENTITY_MODEL
*"* do not include other source files here!!!
ENDCLASS.

CLASS ZCL_RF_SINGLE_ENTITY_MODEL IMPLEMENTATION.
  METHOD connect_spec_gen_structures.

    GET REFERENCE OF cs_structure INTO me->mr_structure.
    me->mo_structdescr ?= cl_abap_typedescr=>describe_by_data_ref( me->mr_structure ).
    CREATE DATA me->mr_structure_in_previous_state TYPE HANDLE me->mo_structdescr.

  ENDMETHOD.                    "connect_spec_gen_structures
  METHOD constructor.
    me->mv_model_state =  co_model_state_notchanged.


  ENDMETHOD.                    "constructor
  METHOD zif_rf_single_entity_model~check_structure.

    FIELD-SYMBOLS <fs> TYPE any.
```

```
    ASSERT me->mr_structure IS BOUND.  "No Connection between Generic and Specific
Structure!!
    ASSIGN me->mr_structure->* TO <fs>.

    IF me->mv_needs_checking = abap_true.
      RAISE EVENT ev_single_entity_model_changed
        EXPORTING es_structure = <fs>.
      me->mv_needs_checking = abap_false.
    ENDIF.
  ENDMETHOD.                    "ZIF_RF_SINGLE_ENTITY_MODEL~CHECK_STRUCTURE
  METHOD zif_rf_single_entity_model~get_errors.
    rt_model_error = me->mt_model_error.
  ENDMETHOD.                    "ZIF_RF_SINGLE_ENTITY_MODEL~GET_ERRORS


  METHOD zif_rf_single_entity_model~get_model_state.

    re_model_state = me->mv_model_state.

  ENDMETHOD.                    "zif_RF_single_entity_model~get_model_state


  METHOD zif_rf_single_entity_model~get_struct_descr.

    ASSERT me->mr_structure IS BOUND.  "No Connection between SpecificAnd Generic
Structure
    ro_structdescr ?= cl_abap_structdescr=>describe_by_data_ref( me->mr_structure ).
  ENDMETHOD.                    "ZIF_RF_SINGLE_ENTITY_MODEL~GET_STRUCT_DESCR

  METHOD zif_rf_single_entity_model~is_changed_state.

    IF me->mv_model_state = co_model_state_changed OR
       me->mv_model_state = co_model_state_new.
      rv_is_changed_state = abap_true.
    ELSE.
      rv_is_changed_state = abap_false.
    ENDIF.

  ENDMETHOD.                    "ZIF_RF_SINGLE_ENTITY_MODEL~IS_CHANGED_STATE
  METHOD zif_rf_single_entity_model~revert.

    FIELD-SYMBOLS <fs> TYPE any.
    FIELD-SYMBOLS <fs_previous> TYPE any.
    ASSERT me->mr_structure IS BOUND.  "GEneric Structur not boundto  specific
structure!
    ASSIGN me->mr_structure->* TO <fs>.
    me->read_by_key( EXPORTING is_structure =  <fs> ).
    ASSIGN me->mr_structure_in_previous_state->* TO <fs_previous>.

    <fs_previous> =  <fs>.

  ENDMETHOD.                    "zif_RF_single_entity_model~revert



  METHOD zif_rf_single_entity_model~save.
```

```
    me->mv_model_state =   co_model_state_saved .
  ENDMETHOD.                    "ZIF_RF_SINGLE_ENTITY_MODEL~SAVE


  METHOD zif_rf_single_entity_model~set_edit_mode.

    ASSERT iv_edit_mode = co_edit_mode_change
          OR  iv_edit_mode = co_edit_mode_display.

    me->mv_edit_mode = iv_edit_mode.

  ENDMETHOD.                    "zif_RF_single_entity_model~set_edit_mode


ENDCLASS.
*---------------------------------------------------------------------*
*       CLASS ZCL_RF_MULTIPLE_ENTITY_MODEL DEFINITION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_multiple_entity_model DEFINITION
  PUBLIC
  ABSTRACT
  CREATE PUBLIC

  GLOBAL FRIENDS zcl_rf_model_factory .

  PUBLIC SECTION.
*"* public components of class ZCL_RF_MULTIPLE_ENTITY_MODEL
*"* do not include other source files here!!!

    INTERFACES zif_rf_multiple_entity_model .
  PROTECTED SECTION.
*"* protected components of class ZCL_RF_MULTIPLE_ENTITY_MODEL

    DATA mr_table TYPE REF TO data .
    DATA mo_tabledescr TYPE REF TO cl_abap_tabledescr .

    METHODS connect_spec_gen_table
      CHANGING
        !ct_table TYPE ANY TABLE .
    METHODS call_connect_spec_gen_table
    ABSTRACT
      CHANGING
        !ct_table TYPE ANY TABLE .
  PRIVATE SECTION.
*"* private components of class ZCL_RF_MULTIPLE_ENTITY_MODEL

ENDCLASS.                    "ZCL_RF_MULTIPLE_ENTITY_MODEL DEFINITION
```

```
*---------------------------------------------------------------------*
*       CLASS ZCL_RF_MULTIPLE_ENTITY_MODEL IMPLEMENTATION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_multiple_entity_model IMPLEMENTATION.


  METHOD connect_spec_gen_table.


    GET REFERENCE OF ct_table INTO me->mr_table.
    me->mo_tabledescr ?= cl_abap_typedescr=>describe_by_data_ref( me->mr_table ).


  ENDMETHOD.                    "connect_spec_gen_table


  METHOD zif_rf_multiple_entity_model~get_table.

    FIELD-SYMBOLS <fs_table> TYPE ANY TABLE.

    ASSIGN me->mr_table->* TO <fs_table>.

    et_table = <fs_table>.

  ENDMETHOD.                    "zif_rf_multiple_entity_model~get_table


  METHOD zif_rf_multiple_entity_model~get_table_descr.
    ASSERT  me->mr_table IS BOUND.
    ASSERT  me->mo_tabledescr IS BOUND.

    ro_tabledescr = me->mo_tabledescr.

  ENDMETHOD.                    "ZIF_RF_MULTIPLE_ENTITY_MODEL~GET_TABLE_DESCR


  METHOD zif_rf_multiple_entity_model~get_table_ref.


    rr_table = me->mr_table.

  ENDMETHOD.                    "ZIF_RF_MULTIPLE_ENTITY_MODEL~GET_TABLE_REF


  METHOD zif_rf_multiple_entity_model~on_single_entity_model_changed.

    FIELD-SYMBOLS <fs_table> TYPE ANY TABLE.

    ASSIGN me->mr_table->* TO <fs_table>.
    MODIFY TABLE <fs_table> FROM es_structure.
    RAISE EVENT ev_multi_entity_model_changed.

  ENDMETHOD.
"ZIF_RF_MULTIPLE_ENTITY_MODEL~ON_SINGLE_ENTITY_MODEL_CHANGED
```

```
  METHOD zif_rf_multiple_entity_model~set_table_ref.
  ENDMETHOD.                    "ZIF_RF_MULTIPLE_ENTITY_MODEL~SET_TABLE_REF
ENDCLASS.                    "ZCL_RF_MULTIPLE_ENTITY_MODEL IMPLEMENTATION
*---------------------------------------------------------------------*
*       CLASS ZCL_RF_CONNECTION_MODEL DEFINITION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_connection_model DEFINITION
  PUBLIC
  INHERITING FROM zcl_rf_single_entity_model
  CREATE PROTECTED

  GLOBAL FRIENDS zcl_rf_model_factory .

  PUBLIC SECTION.
*"* public components of class ZCL_RF_CONNECTION_MODEL

    INTERFACES zif_rf_connection_model .

    METHODS constructor .

    METHODS zif_rf_single_entity_model~read_by_key
      REDEFINITION .
  PROTECTED SECTION.
*"* protected components of class ZCL_RF_CONNECTION_MODEL

    DATA ms_connection TYPE ty_s_connection .

  PRIVATE SECTION.
*"* private components of class ZCL_RF_CONNECTION_MODEL

ENDCLASS.                    "ZCL_RF_CONNECTION_MODEL DEFINITION

*---------------------------------------------------------------------*
*       CLASS ZCL_RF_CONNECTION_MODEL IMPLEMENTATION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_connection_model IMPLEMENTATION.


  METHOD constructor.

    super->constructor( ).
    me->connect_spec_gen_structures( CHANGING cs_structure = me->ms_connection ).
  ENDMETHOD.                    "CONSTRUCTOR


  METHOD zif_rf_connection_model~get_carrid.
    rv_carrid = me->ms_connection-carrid.

  ENDMETHOD.                    "zif_rf_connection_model~get_carrid
```

```
  METHOD zif_rf_connection_model~get_connid.
    rv_connid = me->ms_connection-connid.
  ENDMETHOD.                    "ZIF_RF_CONNECTION_MODEL~GET_CONNID


  METHOD zif_rf_connection_model~get_flights_model.

    ro_flights_model = zcl_rf_model_factory=>get_flights_model( ).
    ro_flights_model->read_flights_by_connection(    iv_carrid = me->ms_connection-
carrid
                                                     iv_connid = me->ms_connection-
connid ).

  ENDMETHOD.                    "ZIF_RF_CONNECTION_MODEL~GET_FLIGHTS_MODEL


  METHOD zif_rf_single_entity_model~read_by_key.

    FIELD-SYMBOLS <fs> TYPE ty_s_connection.

    ASSIGN is_structure TO <fs> CASTING.
    SELECT SINGLE * FROM spfli INTO CORRESPONDING FIELDS OF me->ms_connection
      WHERE carrid = <fs>-carrid
      AND  connid = <fs>-connid.

    ASSERT sy-subrc = 0.
  ENDMETHOD.                    "ZIF_RF_SINGLE_ENTITY_MODEL~READ_BY_KEY
ENDCLASS.                    "ZCL_RF_CONNECTION_MODEL IMPLEMENTATION


*---------------------------------------------------------------------*
*       CLASS zcl_rf_connections_model DEFINITION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_connections_model DEFINITION
  PUBLIC
  INHERITING FROM zcl_rf_multiple_entity_model
  CREATE PUBLIC

  GLOBAL FRIENDS zcl_rf_model_factory .

  PUBLIC SECTION.
*"* public components of class ZCL_RF_CONNECTIONS_MODEL


    INTERFACES zif_rf_connections_model .

    ALIASES read_connections_by_dep_dest
      FOR zif_rf_connections_model~read_connections_by_dep_dest .
    ALIASES ev_multi_entity_model_changed
      FOR zif_rf_multiple_entity_model~ev_multi_entity_model_changed .
    ALIASES ty_s_connection
      FOR zif_rf_connections_model~ty_s_connection .
    ALIASES ty_t_connection
```

```
      FOR zif_rf_connections_model~ty_t_connection .

    METHODS constructor .

    METHODS zif_rf_multiple_entity_model~set_table_ref
      REDEFINITION .
  PROTECTED SECTION.
*"* protected components of class ZCL_RF_CONNECTIONS_MODEL


    DATA mr_connections TYPE REF TO ty_t_connection .

    METHODS call_connect_spec_gen_table
      REDEFINITION .
  PRIVATE SECTION.
*"* private components of class ZCL_RF_CONNECTIONS_MODEL


    ALIASES on_single_entity_model_changed
      FOR zif_rf_multiple_entity_model~on_single_entity_model_changed .
ENDCLASS.                    "zcl_rf_connections_model DEFINITION
*----------------------------------------------------------------------*
*       CLASS ZCL_RF_CONNECTIONS_MODEL IMPLEMENTATION
*----------------------------------------------------------------------*
*
*----------------------------------------------------------------------*
CLASS zcl_rf_connections_model IMPLEMENTATION.


  METHOD call_connect_spec_gen_table.

    me->connect_spec_gen_table( CHANGING ct_table = ct_table ).

  ENDMETHOD.                    "CALL_CONNECT_SPEC_GEN_TABLE


  METHOD constructor.
    super->constructor( ).
    CREATE DATA me->mr_connections TYPE ty_t_connection.
  ENDMETHOD.                    "constructor


  METHOD zif_rf_connections_model~get_connection_model.

    DATA ls_connection TYPE ty_s_connection.
    READ TABLE me->mr_connections->* INTO ls_connection
          INDEX iv_index.

    ro_connection_model = zcl_rf_model_factory=>get_connection_model( ).
    ro_connection_model->read_by_key( ls_connection ).

  ENDMETHOD.                    "zif_rf_connections_model~get_connection_model


  METHOD zif_rf_connections_model~read_connections_by_dep_dest.
```

```
    SELECT * FROM spfli INTO CORRESPONDING FIELDS OF TABLE me->mr_connections->*
      WHERE cityfrom = iv_cityfrom
      AND   cityto = iv_cityto.

  ENDMETHOD.                    "zif_rf_connections_model~read_connections_by_dep_dest
  METHOD zif_rf_multiple_entity_model~set_table_ref.

    FIELD-SYMBOLS <fs_connections> TYPE ty_t_connection.

    IF me->mr_connections->* IS NOT INITIAL.   "We Are Going To Referencing mod,
      "Data has been read, needs to be transferred
      "to Table in UI ( Via Controller)
      ASSIGN  ir_table->* TO <fs_connections>.
      <fs_connections> = me->mr_connections->*.
    ENDIF.

    me->mr_connections ?= ir_table.
    me->call_connect_spec_gen_table( CHANGING ct_table = me->mr_connections->* ).


  ENDMETHOD.                    "ZIF_RF_MULTIPLE_ENTITY_MODEL~SET_TABLE_REF
ENDCLASS.                    "ZCL_RF_CONNECTIONS_MODEL IMPLEMENTATION


*---------------------------------------------------------------------*
*       CLASS ZCL_RF_FLIGHT_MODEL DEFINITION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_flight_model DEFINITION
  PUBLIC
  INHERITING FROM zcl_rf_single_entity_model
  CREATE PROTECTED

  GLOBAL FRIENDS zcl_rf_model_factory .

  PUBLIC SECTION.
*"* public components of class ZCL_RF_FLIGHT_MODEL
    INTERFACES zif_rf_flight_model .

    METHODS constructor .

    METHODS zif_rf_single_entity_model~read_by_key
      REDEFINITION .
  PROTECTED SECTION.
*"* protected components of class ZCL_RF_FLIGHT_MODEL


    ALIASES ty_s_flight
      FOR zif_rf_flight_model~ty_s_flight .

    DATA ms_flight TYPE ty_s_flight .
  PRIVATE SECTION.
*"* private components of class ZCL_RF_FLIGHT_MODEL

ENDCLASS.                    "ZCL_RF_FLIGHT_MODEL DEFINITION
```

```
*---------------------------------------------------------------------*
*       CLASS ZCL_RF_FLIGHT_MODEL IMPLEMENTATION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_flight_model IMPLEMENTATION.


  METHOD constructor.
    super->constructor( ).
    connect_spec_gen_structures( CHANGING cs_structure =  me->ms_flight ).
  ENDMETHOD.                    "CONSTRUCTOR
  METHOD zif_rf_flight_model~get_bookings_model.
  ENDMETHOD.                    "ZIF_RF_FLIGHT_MODEL~GET_BOOKINGS_MODEL


  METHOD zif_rf_flight_model~get_flight_structure.
    rs_flight = me->ms_flight.
  ENDMETHOD.                    "ZIF_RF_FLIGHT_MODEL~GET_FLIGHT_STRUCTURE


  METHOD zif_rf_flight_model~get_passengers_model.
  ENDMETHOD.                    "ZIF_RF_FLIGHT_MODEL~GET_PASSENGERS_MODEL
  METHOD zif_rf_flight_model~set_flight_structure.
* This method can be made generic in the superclass!
    FIELD-SYMBOLS <fs_structure> TYPE any.

    IF is_flight <> me->ms_flight.
      ASSIGN me->mr_structure_in_previous_state->* TO <fs_structure> .
      <fs_structure> = me->ms_flight.
      me->ms_flight = is_flight.
      me->check_structure( ). "To Do Error Handling
      RAISE EVENT ev_single_entity_model_changed
             EXPORTING es_structure = me->ms_flight.
    ENDIF.

  ENDMETHOD.                    "ZIF_RF_FLIGHT_MODEL~SET_FLIGHT_STRUCTURE


  METHOD zif_rf_single_entity_model~read_by_key.

    FIELD-SYMBOLS <fs> TYPE ty_s_flight.
    ASSIGN is_structure TO <fs> CASTING.
    SELECT SINGLE * FROM sflight INTO CORRESPONDING FIELDS OF me->ms_flight
       WHERE carrid = <fs>-carrid
       AND  connid = <fs>-connid
       AND fldate = <fs>-fldate.
    ASSERT sy-subrc = 0.

    ASSIGN me->mr_structure_in_previous_state->* TO <fs> CASTING.

    <fs> = me->ms_flight.
    me->mv_model_state = co_model_state_notchanged .
```

```
  ENDMETHOD.                    "ZIF_RF_SINGLE_ENTITY_MODEL~READ_BY_KEY
ENDCLASS.                    "ZCL_RF_FLIGHT_MODEL IMPLEMENTATION

*---------------------------------------------------------------------*
*       CLASS ZCL_RF_FLIGHTS_MODEL DEFINITION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_flights_model DEFINITION
  PUBLIC
  INHERITING FROM zcl_rf_multiple_entity_model
  CREATE PROTECTED
  GLOBAL FRIENDS zcl_rf_model_factory .

  PUBLIC SECTION.
*"* public components of class ZCL_RF_FLIGHTS_MODEL

    INTERFACES zif_rf_flights_model .
    METHODS constructor .
    METHODS zif_rf_multiple_entity_model~set_table_ref
      REDEFINITION .
  PROTECTED SECTION.
*"* protected components of class ZCL_RF_FLIGHTS_MODEL
    INTERFACE zif_rf_flights_model LOAD .
    DATA mr_flights TYPE REF TO zif_rf_flights_model=>ty_t_flight .
    METHODS call_connect_spec_gen_table
      REDEFINITION .
  PRIVATE SECTION.
*"* private components of class ZCL_RF_FLIGHTS_MODEL

ENDCLASS.                    "ZCL_RF_FLIGHTS_MODEL DEFINITION



*---------------------------------------------------------------------*
*       CLASS ZCL_RF_FLIGHTS_MODEL IMPLEMENTATION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
CLASS zcl_rf_flights_model IMPLEMENTATION.

  METHOD call_connect_spec_gen_table.

    me->connect_spec_gen_table( CHANGING ct_table = ct_table ).

  ENDMETHOD.                    "CALL_CONNECT_SPEC_GEN_TABLE


  METHOD constructor.

    super->constructor( ).
    CREATE DATA me->mr_flights TYPE ty_t_flight.
    me->connect_spec_gen_table( CHANGING ct_table = me->mr_flights->*  ).

  ENDMETHOD.                    "constructor
```

```
  METHOD zif_rf_flights_model~get_flight_model.


    DATA ls_flight TYPE ty_s_flight.
    READ TABLE me->mr_flights->* INTO ls_flight INDEX iv_index.
    ro_flight_model = zcl_rf_model_factory=>get_flight_model( ).
    SET HANDLER me->on_single_entity_model_changed FOR ro_flight_model.
    ro_flight_model->read_by_key( ls_flight ).
  ENDMETHOD.                    "zif_rf_flights_model~get_flight_model
  METHOD zif_rf_flights_model~read_flights_by_airline.

    SELECT * FROM sflight INTO CORRESPONDING FIELDS OF TABLE me->mr_flights->*
      WHERE carrid = im_carrid.

  ENDMETHOD.                    "zif_rf_flights_model~read_flights_by_airline


  METHOD zif_rf_flights_model~read_flights_by_connection.

    SELECT * FROM sflight INTO CORRESPONDING FIELDS OF TABLE me->mr_flights->*
      WHERE carrid = iv_carrid
      AND   connid = iv_connid.

  ENDMETHOD.                    "zif_rf_flights_model~read_flights_by_connection


  METHOD zif_rf_multiple_entity_model~set_table_ref.

    FIELD-SYMBOLS <fs_flights> TYPE ty_t_flight.
    IF me->mr_flights->* IS NOT INITIAL.   "We Are Going To Referencing mod,
      "Data has been read, needs to be transferred
      "to Table in UI ( Via Controller)
      ASSIGN  ir_table->* TO <fs_flights>.
      <fs_flights> = me->mr_flights->*.
    ENDIF.

    me->mr_flights ?= ir_table.
    me->call_connect_spec_gen_table( CHANGING ct_table = me->mr_flights->* ).

  ENDMETHOD.                    "zif_rf_multiple_entity_model~set_table_ref
ENDCLASS.                   "ZCL_RF_FLIGHTS_MODEL IMPLEMENTATION
*&---------------------------------------------------------------------*
*& Report  Z_RF_SAPGUI_PROGRAM
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

REPORT  z_rf_sapgui_program.

* Declaration Global Variables needed for Gui Screen 100
DATA gt_flight TYPE zcl_rf_sapgui_controller=>ty_t_flight.
DATA gt_connection TYPE zcl_rf_sapgui_controller=>ty_t_connection.
TABLES spfli.
```

```
TABLES sflight.
DATA go_cont_connection TYPE REF TO cl_gui_custom_container.
DATA go_cont_flight TYPE REF TO cl_gui_custom_container.
DATA go_alv_connection TYPE REF TO cl_gui_alv_grid.
DATA go_alv_flight TYPE REF TO cl_gui_alv_grid.


* Declaration Other Variables
DATA go_controller TYPE REF TO zcl_rf_sapgui_controller.
DATA  ok_code TYPE syucomm.

START-OF-SELECTION.

  CALL SCREEN 100.
*&---------------------------------------------------------------------*
*&      Module  STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE status_0100 OUTPUT.
  SET PF-STATUS space.
  SET TITLEBAR 'TITLE_100'.

ENDMODULE.                 " STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
*&      Module  INIT_0100  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE init_0100 OUTPUT.

  IF go_controller IS  BOUND.
    RETURN.
  ENDIF.
  CREATE OBJECT go_controller.

  go_controller->init_0100(
    CHANGING
      ct_flight          = gt_flight
      ct_connection      = gt_connection
      cs_connection      = spfli
      cs_flight          = sflight
      co_cont_connection = go_cont_connection
      co_cont_flight     = go_cont_flight
      co_alv_connection  = go_alv_connection
      co_alv_flight      = go_alv_flight
         ).



ENDMODULE.                 " INIT_0100  OUTPUT
*&---------------------------------------------------------------------*
*&      Module  PBO_0100  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
```

```
MODULE pbo_0100 OUTPUT.

  go_controller->pbo_0100( ).


ENDMODULE.                 " PBO_0100  OUTPUT
*&---------------------------------------------------------------------*
*&      Module  PAI_0100  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE pai_0100 INPUT.

  go_controller->pai_0100( ok_code ).
ENDMODULE.                 " PAI_0100  INPUT
```

Related Content

Model/View/Controller

The MVC design pattern in ABAP for practical use - part 1

For more information, visit the ABAP homepage

Copyright