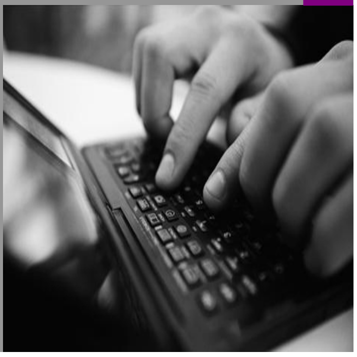


How-to Guide
SAP NetWeaver ‘04
SAP NetWeaver 7.0

# How To… Use Interactive Forms in BSP Applications

Version 1.00 – August 2007

Jeff Gebo – SAP Labs LLC

Applicable Releases:
SAP NetWeaver ’04
SAP NetWeaver 7.0

Requires Adobe Documents Services (ADS) to be installed & configured.

Requires knowledge of BSP Application programming.

# 1 Scenario

You want to use Interactive Forms in ABAP BSP applications.

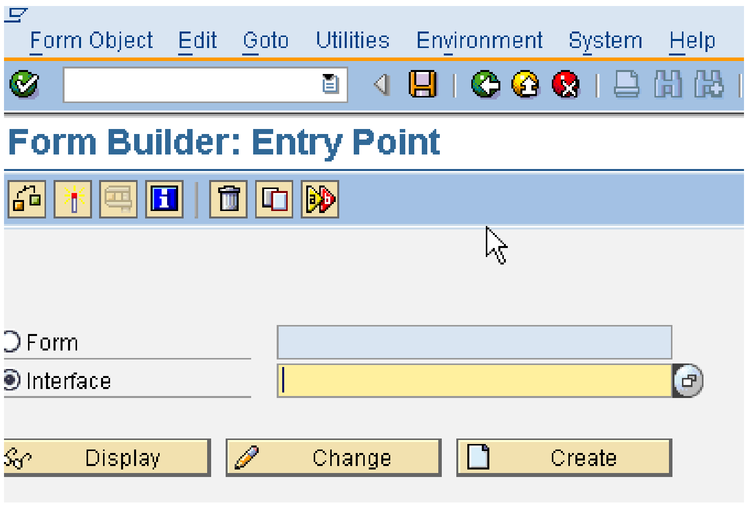
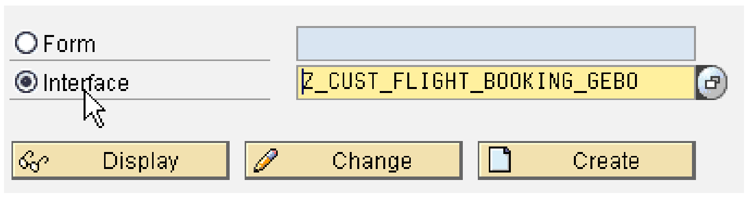
# 2 Introduction

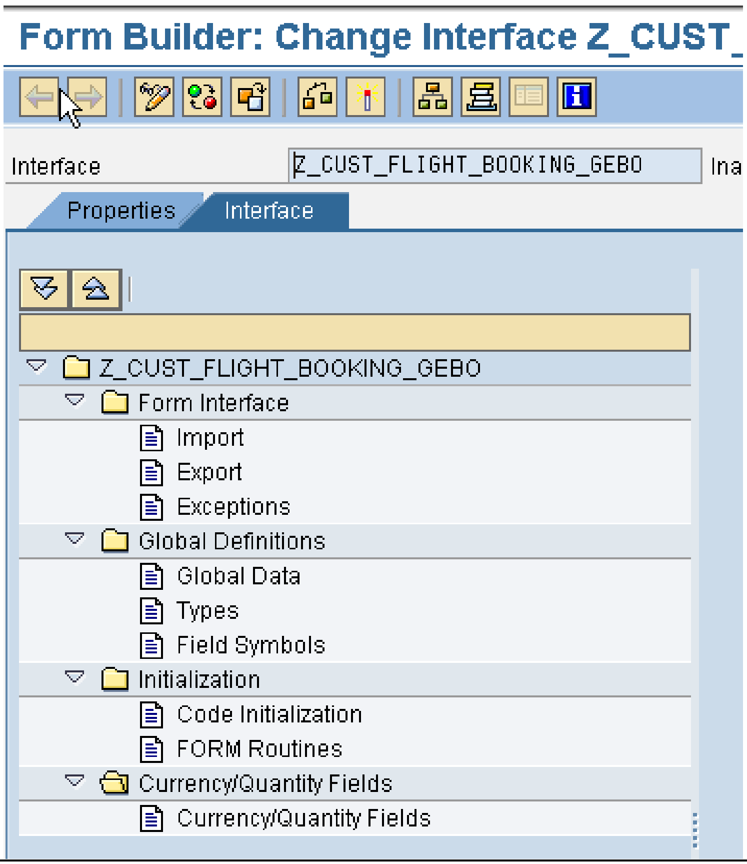
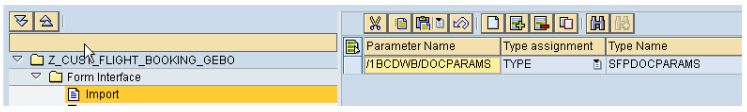
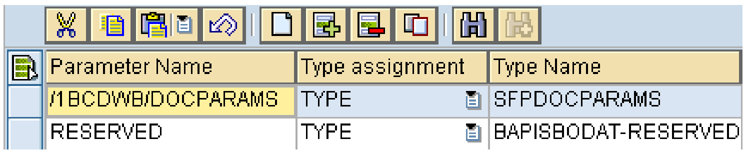
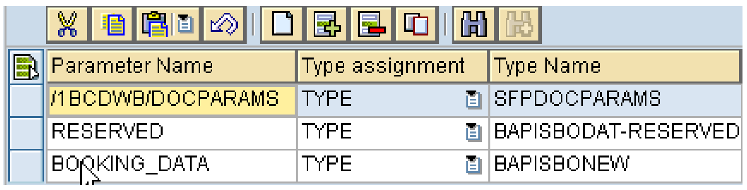
Using Adobe Interactive Forms in BSP applications isn't much different from doing just standard print forms from BSP. It just takes an understanding of a few details as well as the iXML library to parse the result that is passed back from the Interactive Form. The following tutorial will guide you through creating the Interactive Form, creating a BSP application that uses the Form, and parsing the data returned by the form with the iXML API.

Please note, this tutorial requires that you have access to a SAP NetWeaver '04 (or 2004s) AS ABAP that is configured to communicate with an Adobe Documents Server (ADS). In addition, to complete the tutorial fully it is required that the Flight Data model on the AS ABAP has been populated. If it is not, you can still accomplish most of the tutorial and will still gain an understanding of how to use this technology. Knowledge of BSP Application programming is required to complete this tutorial.

# 3 The Step By Step Solution

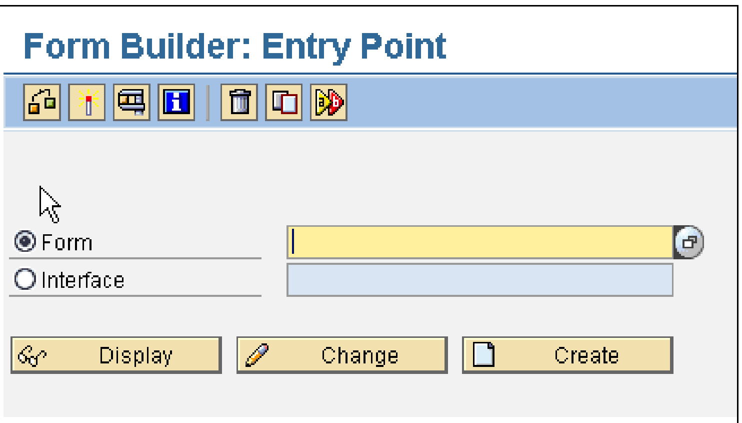
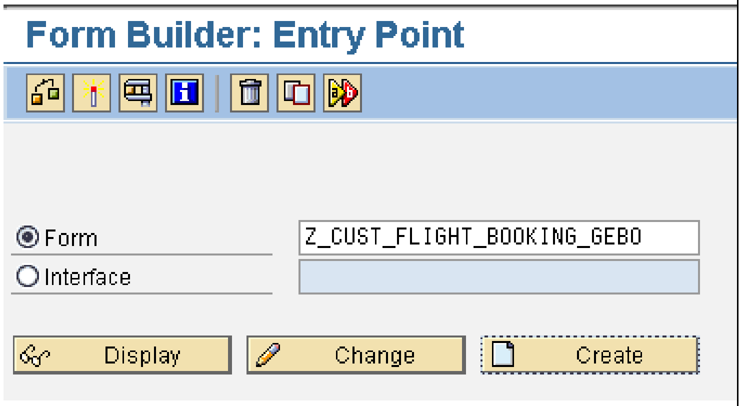
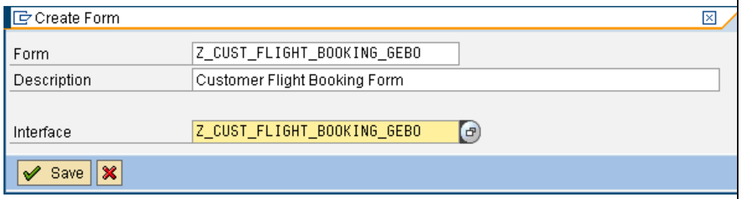
## 3.1 Create the Form Interface. This will be a simple interface that accepts flight booking data to be entered.

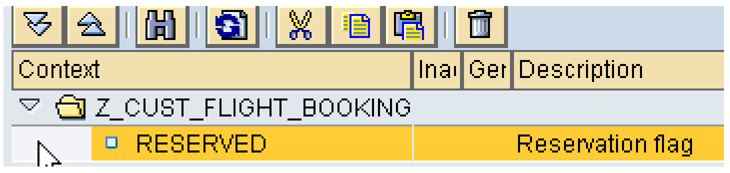
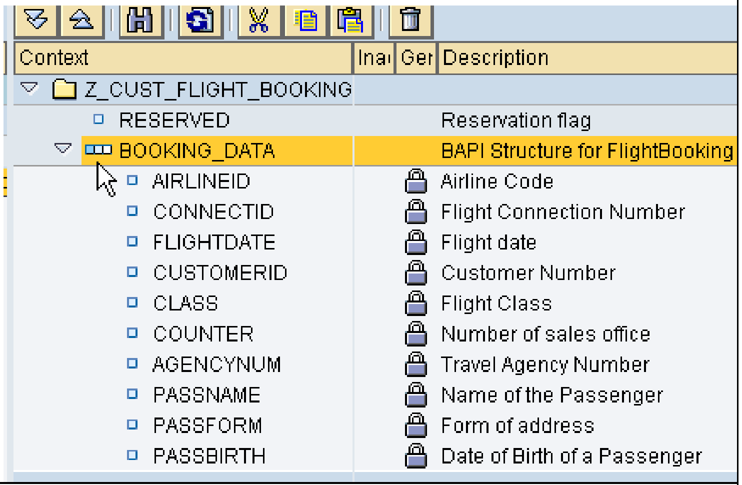
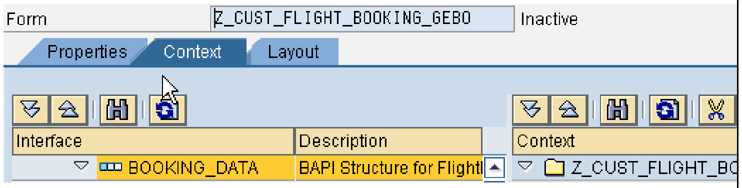
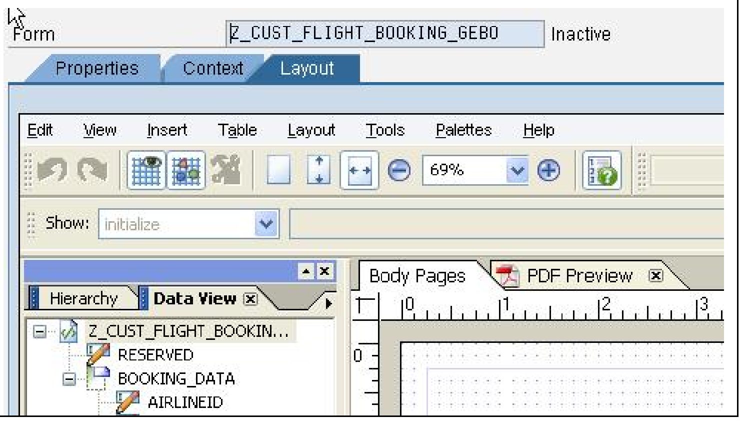
| | |
|---|---|
| 1. Go to transaction SFP. Select the Interface radio button. |  Form Builder: Entry Point<br>● Interface |
| 2. Enter the Interface name in the corresponding input field: Z_CUST_FLIGHT_BOOKING_<USERNAME>.<br><br>3. Click Create, enter a description and save the interface a local object. |  ● Interface Z_CUST_FLIGHT_BOOKING_GEBO |

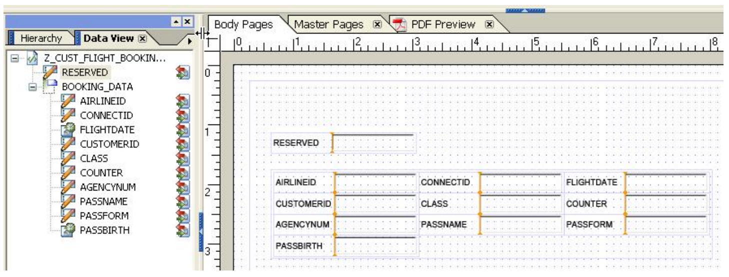
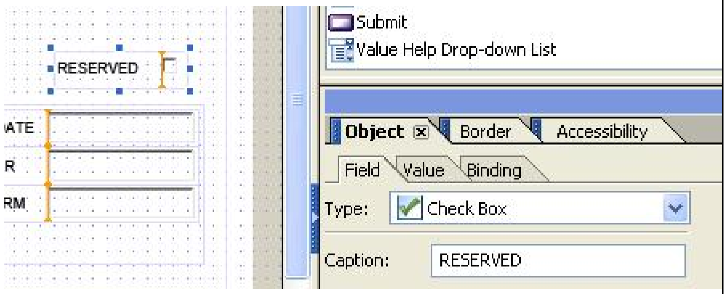
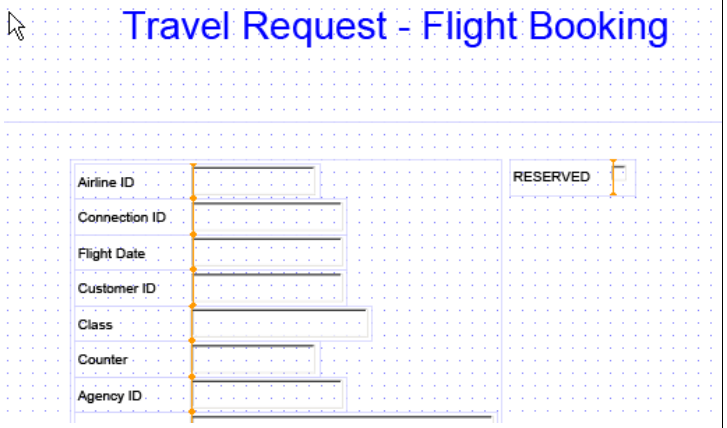
| | |
|---|---|
| |  Form Builder: Change Interface Z_CUST_<br>Interface Z_CUST_FLIGHT_BOOKING_GEBO Ina<br>Properties Interface<br>Z_CUST_FLIGHT_BOOKING_GEBO<br>Form Interface<br>Import<br>Export<br>Exceptions<br>Global Definitions<br>Global Data<br>Types<br>Field Symbols<br>Initialization<br>Code Initialization<br>FORM Routines<br>Currency/Quantity Fields<br>Currency/Quantity Fields |
| 4. Double click on the Import item shown on the left hand pane…this will allow you to enter new elements for the form interface. |  Z_CUST_FLIGHT_BOOKING_GEBO<br>Form Interface<br>Import<br>Parameter Name / Type assignment / Type Name<br>/1BCDWB/DOCPARAMS / TYPE / SFPDOCPARAMS |
| 5. Click the create icon on the right hand pane to insert a new row into the import interface.<br>6. Enter the parameter name "Reserved", with type BAPISBODAT-RESERVED. |  Parameter Name / Type assignment / Type Name<br>/1BCDWB/DOCPARAMS / TYPE / SFPDOCPARAMS<br>RESERVED / TYPE / BAPISBODAT-RESERVED |
| 7. Click the create icon again to insert another new row into the import interface.<br>8. Enter the parameter name "BOOKING_DATA", with type BAPISBONEW. |  Parameter Name / Type assignment / Type Name<br>/1BCDWB/DOCPARAMS / TYPE / SFPDOCPARAMS<br>RESERVED / TYPE / BAPISBODAT-RESERVED<br>BOOKING_DATA / TYPE / BAPISBONEW |

|   |   |
|---|---|
| 9. Save and Active the interface. You now have a simple interface that takes a reserved flag and new flight booking data structure. The next step is to use this form interface in a form. |   |

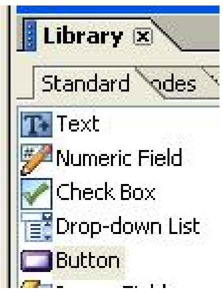
## 3.2 Create the Form, using the Form Interface that you created in the last step. This will be a simple form that will allow users to enter information for booking a flight.

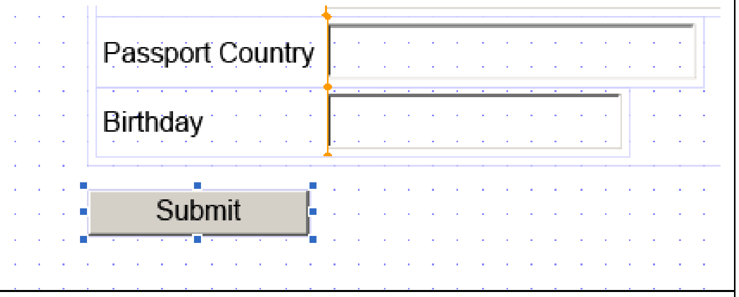
| | |
|---|---|
| 1. Go to transaction SFP. Select the Form radio button. |  Form Builder: Entry Point<br>[✓ Form] Interface<br>Display Change Create |
| 2. Enter the Interface name in the corresponding input field: Z_CUST_FLIGHT_BOOKING_<USERNAME>.<br>3. Click Create. |  Form Builder: Entry Point<br>[✓ Form] Z_CUST_FLIGHT_BOOKING_GEBO<br>Interface<br>Display Change Create |
| 4. Enter a description and the name of the interface you created in the last section (in this case it is the same name as the form you are creating).<br>5. Click Save, and save the Form as a local object. |  Create Form<br>Form: Z_CUST_FLIGHT_BOOKING_GEBO<br>Description: Customer Flight Booking Form<br>Interface: Z_CUST_FLIGHT_BOOKING_GEBO<br>Save |

| | |
|---|---|
| 6. Drag and Drop the “RESERVED” element found under Import on the left hand pane, the context node on the right hand pane.<br><br>7. Do the same with the BOOKING_DATA element. Since BOOKING_DATA is a structure you will see it expanded as shown to the right.<br><br>8. Save your form. | <br>Context \| Inac \| Ger \| Description<br>Z_CUST_FLIGHT_BOOKING<br>RESERVED \| Reservation flag<br><br><br>Context \| Inac \| Ger \| Description<br>Z_CUST_FLIGHT_BOOKING<br>RESERVED \| Reservation flag<br>BOOKING_DATA \| BAPI Structure for FlightBooking<br>AIRLINEID \| Airline Code<br>CONNECTID \| Flight Connection Number<br>FLIGHTDATE \| Flight date<br>CUSTOMERID \| Customer Number<br>CLASS \| Flight Class<br>COUNTER \| Number of sales office<br>AGENCYNUM \| Travel Agency Number<br>PASSNAME \| Name of the Passenger<br>PASSFORM \| Form of address<br>PASSBIRTH \| Date of Birth of a Passenger |
| 9. Select the Layout tab of your form. This will launch the Adobe LiveCycle Designer. | <br>Form Z_CUST_FLIGHT_BOOKING_GEBO Inactive<br>Properties \| Context \| Layout<br>Interface \| Description<br>BOOKING_DATA \| BAPI Structure for Flight<br>Context<br>Z_CUST_FLIGHT_BO<br><br><br>Form Z_CUST_FLIGHT_BOOKING_GEBO Inactive<br>Properties \| Context \| Layout<br>Edit View Insert Table Layout Tools Palettes Help<br>69%<br>Show: initialize<br>Hierarchy \| Data View<br>Body Pages \| PDF Preview<br>Z_CUST_FLIGHT_BOOKIN...<br>RESERVED<br>BOOKING_DATA<br>AIRLINEID |

| | |
|---|---|
| 10. In the LiveCycle Designer, drag and drop the BOOKING_DATA node from the Data View to the form.<br><br>11. Do the same for the RESERVED data element. |  Hierarchy<br>Data View<br>Body Pages<br>Master Pages<br>PDF Preview<br>Z_CUST_FLIGHT_BOOKIN...<br>RESERVED<br>BOOKING_DATA<br>AIRLINEID<br>CONNECTID<br>FLIGHTDATE<br>CUSTOMERID<br>CLASS<br>COUNTER<br>AGENCYNUM<br>PASSNAME<br>PASSFORM<br>PASSBIRTH |
| 12. Change the RESERVED UI Element into a checkbox. To do this, select the UI element, then in the Object tab in the object palette on the right; select the Field sub-tab. Change the Type dropdown to be Check Box. |  RESERVED<br>Submit<br>Value Help Drop-down List<br>Object<br>Border<br>Accessibility<br>Field<br>Value<br>Binding<br>Type: Check Box<br>Caption: RESERVED |
| 13. Modify the rest of the form as you see fit. Add a header and a logo on the Master page. Change the BOOKING_DATA format. Make the labels more readable. |  Travel Request - Flight Booking<br>Airline ID<br>Connection ID<br>Flight Date<br>Customer ID<br>Class<br>Counter<br>Agency ID<br>RESERVED |

14. Add a UI element “Button” to the form from the Standard Library.

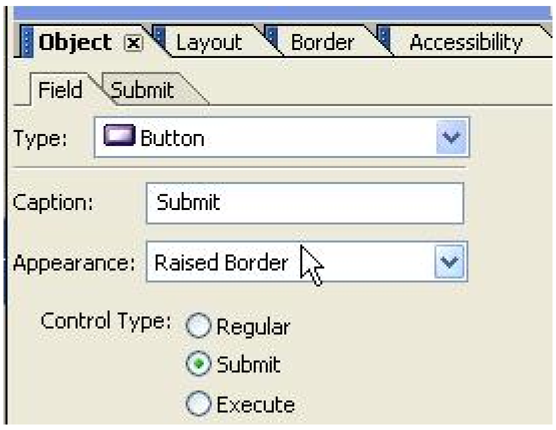




15. Set the proper properties on the button to call a BSP controller and to pass XML data.

Select the Submit Button, and then go to the Object > Field tab on the right hand side. Set the Caption to “Submit”. Set the Control Type to submit.

Switch to the Submit tab, and enter processBookingForm.do into the URL input field. Set the Submit Format to be XML Data (XML).

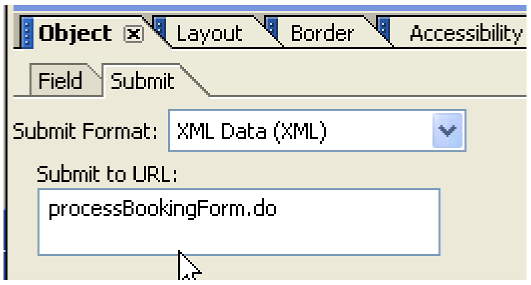


Type: Button

Caption: Submit

Appearance: Raised Border

Control Type: Regular [✓ Submit] Execute

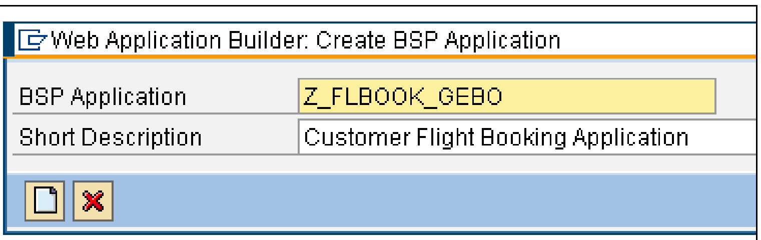
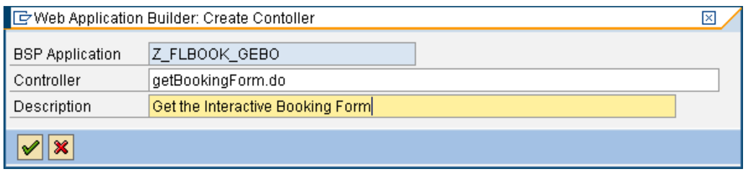


Submit Format: XML Data (XML)

Submit to URL: processBookingForm.do

| | |
|---|---|
| 16. Save and Active your form. | |

## 3.3 Create a new BSP Application & Controller to display the Form.

| | |
|---|---|
| 1. Go to transaction SE80 and create a new BSP Application.<br><br>Note: Keep the name of the application under 15 characters. | <br>Web Application Builder: Create BSP Application<br>BSP Application: Z_FLBOOK_GEBO<br>Short Description: Customer Flight Booking Application |
| 2. Create a BSP Controller named getBookingForm.do with BSP Controller Class named Z_GET_BOOKING_FORM. | <br>Web Application Builder: Create Contoller<br>BSP Application: Z_FLBOOK_GEBO<br>Controller: getBookingForm.do<br>Description: Get the Interactive Booking Form |

| | |
|---|---|
| 3. Redefine the DO_REQUEST method of the Z_GET_BOOKING_FORM Class. | <pre>method DO_REQUEST.<br>*CALL METHOD SUPER->DO_REQUEST<br>*     .<br><br><br>endmethod.</pre> |
| 4. Add the lines of code to remove the header fields that are set by default in BSP response headers.<br><br>Note: without doing this your PDF will not show up in the browser! | <pre>runtime->server->response->delete_header_field(<br>name = 'Cache-Control' )<br>.<br>runtime->server->response->delete_header_field(<br>name = 'Expires' ).<br>runtime->server->response->delete_header_field(<br>name = 'Pragma' ).</pre> |
| 5. Add the call to lookup the Form function module name. You must declare a variable of type funcname for the function name to be stored in. | <pre>data: l_name TYPE funcname.<br><br>try.<br>    call function 'FP_FUNCTION_MODULE_NAME'<br>      exporting<br>        i_name                     =<br>'Z_CUST_FLIGHT_BOOKING_GEBO'<br>      importing<br>        e_funcname                 = l_name<br>      exceptions<br>        others                     = 0.<br>  catch cx_fp_api_repository. "#EC NO_HANDLER<br>  catch cx_fp_api_usage.      "#EC NO_HANDLER<br>  catch cx_fp_api_internal.   "#EC NO_HANDLER<br><br>endtry.</pre> |

6. Add a call to start the form processing so that we can generate the Interactive PDF.

Be sure to use the Flags set here! Most notably the one to get the PDF returned back from the Function call. We will need this so that we can return to it to the client in the response object.

```
data: l_outputparams TYPE sfpoutputparams.

*Start Form Processing.
l_outputparams-getpdf = 'X'.

call function 'FP_JOB_OPEN'
    changing
      ie_outputparams       = l_outputparams
    exceptions
      cancel                = 1
      usage_error           = 2
      system_error          = 3
      internal_error        = 4
      others                = 5.
if sy-subrc <> 0.
    message id sy-msgid type sy-msgty number sy-
msgno
            with sy-msgv1 sy-msgv2 sy-msgv3 sy-
msgv4.
endif.
```

7. Add a call to the Form's function module. Notice the settings on fp_docparams that allow for the generation of an Interactive form. Without these settings the form will be generated as standard print form.

**Hint**: Get the form functional module's generated name by testing it in transaction SFP, and then use the Pattern wizard to insert the function call in your code.

```
data: fp_docparams TYPE sfpdocparams,
      reservedonly TYPE BAPISBODAT-RESERVED,
      booking_data TYPE BAPISBONEW,
      fp_result TYPE fpformoutput.

fp_docparams-fillable = 'X'.
fp_docparams-langu = sy-langu.

CALL FUNCTION l_name
  EXPORTING
    /1BCDWB/DOCPARAMS          = fp_docparams
    RESERVED                   = reservedonly
    BOOKING_DATA               = booking_data
  IMPORTING
   /1BCDWB/FORMOUTPUT         = fp_result
  EXCEPTIONS
    USAGE_ERROR                = 1
    SYSTEM_ERROR               = 2
    INTERNAL_ERROR             = 3
    OTHERS                     = 4
          .
IF SY-SUBRC <> 0.
  MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
          WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.
```

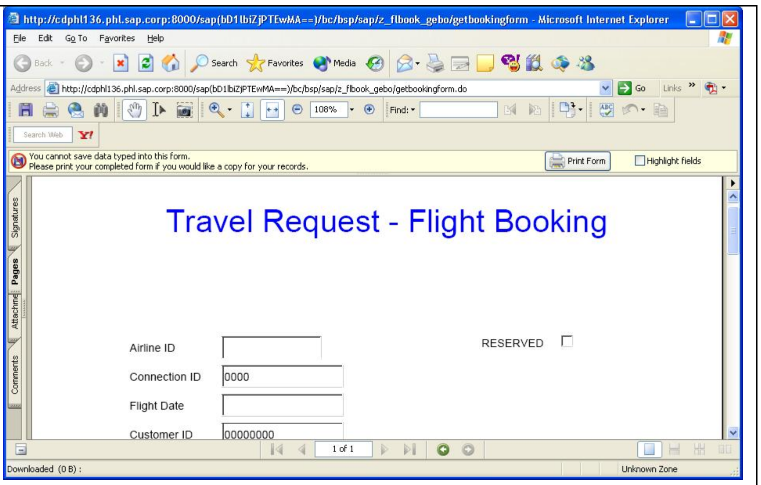
8. Get the returned PDF, and store it in a fpformoutput-pdf type variable.

```
data: pdf type fpformoutput-pdf.

pdf = fp_result-pdf.
```

| | |
|---|---|
| 9. Close the form processing session by calling the FP_JOB_CLOSE function. | <pre>call function 'FP_JOB_CLOSE'<br>    exceptions<br>      usage_error          = 1<br>      system_error         = 2<br>      internal_error       = 3<br>      others               = 4.<br>if sy-subrc <> 0.<br>    message id sy-msgid type sy-msgty number sy-msgno<br>            with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.<br>endif.</pre> |
| 10. Set the content type to be of “application/pdf”. This way, the client will know what kind of format the response is, and will start the adobe reader to display it. | <pre>CALL METHOD RESPONSE->SET_CONTENT_TYPE<br>  EXPORTING<br>    CONTENT_TYPE = 'application/pdf'.</pre> |
| 11. Put the PDF into the response. | <pre>CALL METHOD RESPONSE->SET_DATA<br>  EXPORTING<br>    DATA   = pdf.</pre> |

12. Save and activate your BSP controller. Then test your application…you should see the Interactive form display in the Browswer.



http://cdphl136.phl.sap.corp:8000/sap(bD1lbiZjPTEwMA==)/bc/bsp/sap/z_flbook_gebo/getbookingform - Microsoft Internet Explorer

Address http://cdphl136.phl.sap.corp:8000/sap(bD1lbiZjPTEwMA==)/bc/bsp/sap/z_flbook_gebo/getbookingform.do

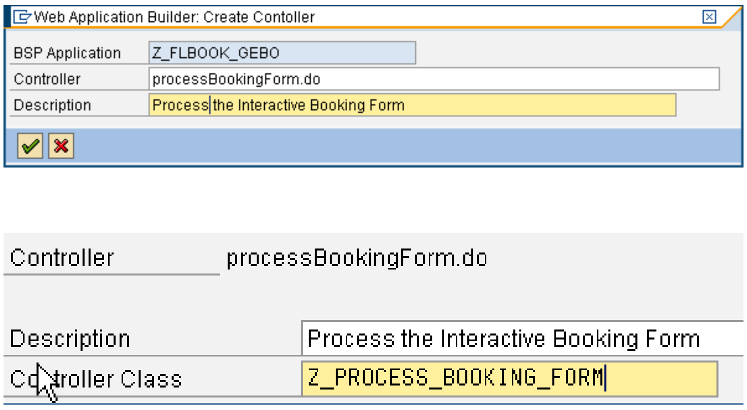
You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

Travel Request - Flight Booking

| | |
|---|---|
| Airline ID | |
| Connection ID | 0000 |
| Flight Date | |
| Customer ID | 00000000 |
| RESERVED | ☐ |

## 3.4 Create another BSP Controller to process the form data.

1. Create a BSP Controller named processBookingForm.do with BSP Controller Class named Z_PROCESS_BOOKING_FORM.

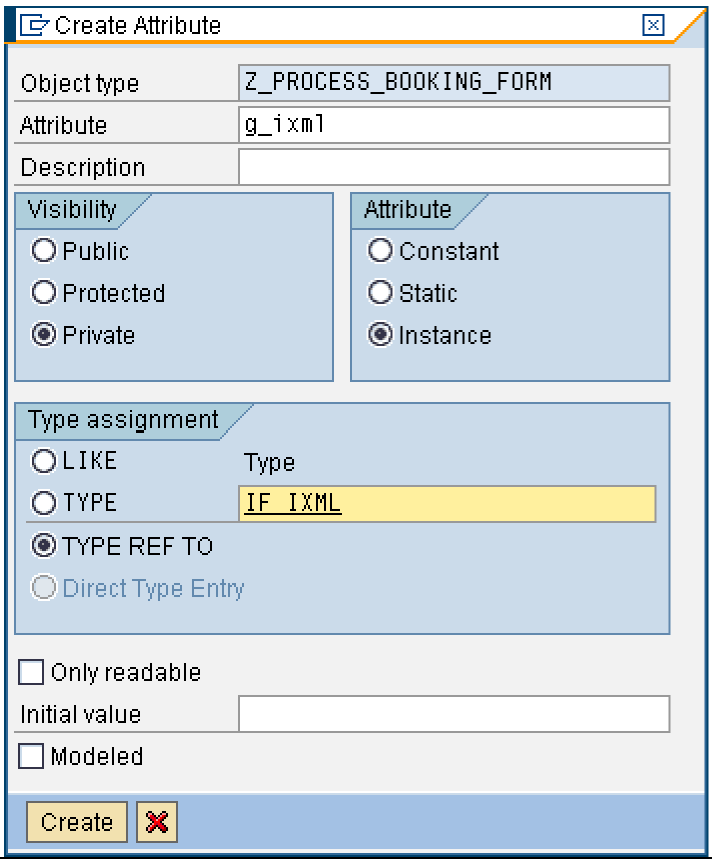


Web Application Builder: Create Contoller

| | |
|---|---|
| BSP Application | Z_FLBOOK_GEBO |
| Controller | processBookingForm.do |
| Description | Process the Interactive Booking Form |

| | |
|---|---|
| Controller | processBookingForm.do |
| Description | Process the Interactive Booking Form |
| Controller Class | Z_PROCESS_BOOKING_FORM |

2. Create an Attribute for your class of type IF_IXML – this will be needed to parse the XML from the Interactive Form.



Create Attribute

| | |
|---|---|
| Object type | Z_PROCESS_BOOKING_FORM |
| Attribute | g_ixml |
| Description | |

Visibility: Public, Protected, [✓ Private]

Attribute: Constant, Static, [✓ Instance]

Type assignment: LIKE, TYPE, [✓ TYPE REF TO], Direct Type Entry

Type: IF_IXML

Only readable ☐

Initial value:

Modeled ☐

3. Redefine the DO_INIT method of the Z_PROCESS_BOOKING_FORM Class.

```
method DO_INIT.
*CALL METHOD SUPER->DO_INIT
*     .

endmethod.
```

4. Add code to get a reference to the IXML service.

```
type-pools: ixml.

class cl_ixml definition load.

g_ixml = cl_ixml=>create( ).
```

5. Redefine the DO_REQUEST method of the Z_PROCESS_BOOKING_FORM Class.

```
method DO_REQUEST.
*CALL METHOD SUPER->DO_REQUEST
*     .

endmethod.
```

6. Add code to get the booking data XML from the request.

Note: the form passes back its data in XML format.

```
data: bookingxml type string.

bookingxml = request->get_cdata( ).
```

| | |
|---|---|
| 7. To process the XML we must create a xml stream out of the xml we received from the request. To do this we need to create a Stream Factory and then a Stream with the Stream Factory. | `data: streamFactory type ref to if_ixml_stream_factory.`<br>`data: iStream type ref to if_ixml_istream.`<br><br>`streamFactory = g_ixml->create_stream_factory( ).`<br><br>`iStream = streamFactory->create_istream_string( bookingxml ).` |
| 8. To process the XML we also must create a Document class. | `data: document type ref to if_ixml_document.`<br><br>`document = g_ixml->create_document( ).` |
| 9. Now create a Parser class so that we can get to the data in the XML document. | `data: parser type ref to if_ixml_parser.`<br><br>`parser = g_ixml->create_parser( stream_factory = streamFactory`<br>`istream = iStream`<br>`document = document ).` |
| 10. Call the parse( ) method on the Parser object.<br><br>The Parser object stores the parsed XML into the Document object we created earlier. Now we can get the data using the Document object.<br><br>**Note**: This method returns an I Type, if it is not equal to 0, you can then process the errors…in this exercise we will not check for errors, but in a production application you should! See the documentation on iXML parser to see how this can be done. | `parser->parse( ).` |
| 11. Create a Node object; this is what is returned from the Document object when we search for a node. | `data: node type ref to if_ixml_node.` |
| 12. Call the find_from_name method on the Document object, user Name = 'RESERVED' to find the value set for the reserved field. And then store the value in a String variable. | `data: strChecked type string.`<br><br>`node = document->find_from_name( name = 'RESERVED' ).`<br><br>`strChecked = node->get_value( ).` |

| | |
|---|---|
| 13. A check box in an Interactive Form returns 1 for checked and 0 for not checked, we need to check this value you set it to 'X' or '' depending if it is checked or not…this is what the BAPI we will call later requires. Create variable of type BAPISBODAT-RESERVED to hold that value.<br><br>Also note that we create a variable of type I, and move the strChecked variable to it. This makes it easy to use the "if" condition on it. | <pre>data: reserved type BAPISBODAT-RESERVED.<br>data: checked type i.<br><br>move strChecked to checked.<br>if ( checked = 1 ).<br>  reserved = 'X'.<br>else.<br>  reserved = ''.<br>endif.</pre> |
| 14. Look up the rest of the values using the Document object. Store the values in a structure type BAPISBONEW<br><br>**Note**: we convert the date into the SAP internal representation. | <pre>data: custbook type BAPISBONEW.<br><br>node = document->find_from_name( name = 'AIRLINEID'<br>).<br>custbook-airlineid = node->get_value( ).<br><br>node = document->find_from_name( name = 'CONNECTID'<br>).<br>custbook-connectid = node->get_value( ).<br><br>data: fd type BAPISBONEW-FLIGHTDATE.<br>data: strFd type string.<br>node = document->find_from_name( name =<br>'FLIGHTDATE' ).<br>strFd = node->get_value( ).<br><br>CALL FUNCTION 'CONVERT_DATE_TO_INTERNAL'<br>  EXPORTING<br>    DATE_EXTERNAL                  = strFd<br>*   ACCEPT_INITIAL_DATE            =<br> IMPORTING<br>   DATE_INTERNAL                  = custbook-<br>flightdate<br> EXCEPTIONS<br>   DATE_EXTERNAL_IS_INVALID       = 1<br>   OTHERS                         = 2<br>          .<br>IF SY-SUBRC <> 0.<br>* Do something<br>ENDIF.<br><br>node = document->find_from_name( name =<br>'CUSTOMERID' ).<br>custbook-customerid = node->get_value( ).<br><br>node = document->find_from_name( name = 'CLASS' ).<br>custbook-class = node->get_value( ).<br><br>node = document->find_from_name( name = 'AGENCYNUM'<br>).<br>custbook-agencynum = node->get_value( ).<br><br>node = document->find_from_name( name = 'PASSNAME'<br>).<br>custbook-passname = node->get_value( ).<br><br>data: pb type BAPISBONEW-PASSBIRTH.<br>data: strPb type string.<br>node = document->find_from_name( name = 'PASSBIRTH'<br>).</pre> |

```
strPb = node->get_value( ).
CALL FUNCTION 'CONVERT_DATE_TO_INTERNAL'
  EXPORTING
    DATE_EXTERNAL                       = strPb
*   ACCEPT_INITIAL_DATE                 =
 IMPORTING
   DATE_INTERNAL                       = custbook-
passbirth
 EXCEPTIONS
   DATE_EXTERNAL_IS_INVALID            = 1
   OTHERS                              = 2
          .
IF SY-SUBRC <> 0.
* Something
ENDIF.
```

15. Call the BAPI_FLBOOKING_CREATEFROMDATA function to create a flight booking. You will need the variables defined here to get the returned booking number, and any errors that may have occurred.

**Note**: If you do not have any Flight test data in your system, do not call this BAPI, but just print out the values you that have been returned to you from the Interactive Form (Skip this step, and steps 16 & 17).

```
data: bn type BAPISBOKEY-BOOKINGID.
data: return type bapiret2_tab.
data: errmsg type string.

CALL FUNCTION 'BAPI_FLBOOKING_CREATEFROMDATA'
  EXPORTING
    RESERVE_ONLY          = reserved
    BOOKING_DATA          = custbook
*   TEST_RUN              = ' '
  IMPORTING
*   AIRLINEID             =
    BOOKINGNUMBER         = bn
*   TICKET_PRICE          =
  TABLES
*   EXTENSION_IN          =
    RETURN              = return.
```

16. Loop at the return table that may contain errors, put any error messages you find into the response.

```
data: wa type bapiret2.

loop at return into wa.

  if wa-type = 'E'.
    move wa-message to errmsg.
    response->append_cdata( errmsg ).
    response->append_cdata( ‘<BR>’ ).
  endif.

endloop.
```

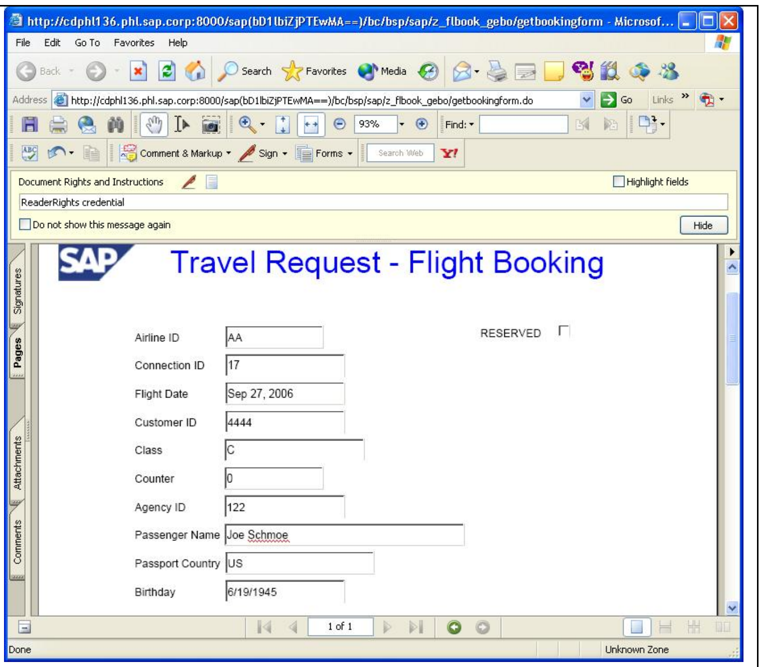
17. Append the booking number to the response.

```
response->append_cdata( 'Flight booked, booking
reference number: ').

data: strBn type string.
move bn to strBn.

response->append_cdata( strBn ).
```

18. Save and Activate the Controller Class and BSP Controller. Run the form, add some valid data into the form and click Submit.

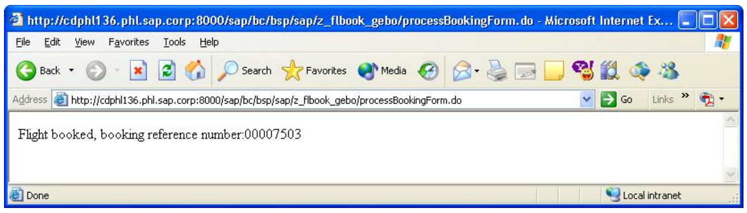

http://cdphl136.phl.sap.corp:8000/sap(bD1lbiZjPTEwMA==)/bc/bsp/sap/z_flbook_gebo/getbookingform.do

Document Rights and Instructions
ReaderRights credential

Travel Request - Flight Booking

| | |
|---|---|
| Airline ID | AA |
| Connection ID | 17 |
| Flight Date | Sep 27, 2006 |
| Customer ID | 4444 |
| Class | C |
| Counter | 0 |
| Agency ID | 122 |
| Passenger Name | Joe Schmoe |
| Passport Country | US |
| Birthday | 6/19/1945 |

RESERVED ☐


http://cdphl136.phl.sap.corp:8000/sap/bc/bsp/sap/z_flbook_gebo/processBookingForm.do

Flight booked, booking reference number:00007503 www.sap.com/netweaver