Use the Enhancement Package for SAP ERP strategy to develop and enhance Web Dynpro ABAP applications

Part 1 — Developing new Web Dynpro ABAP applications

by Karl Kessler

Karl Kessler
Product Manager,
SAP AG

Karl Kessler studied computer science at the Technical University in Munich, Germany. He joined SAP in 1992. In 1994, Karl became a member of the ABAP Workbench product management group, and in 1996, he became product manager for business programming languages and frameworks. In 2003, Karl assumed responsibility for product management of the SAP NetWeaver platform, including SAP Web Application Server, SAP NetWeaver Developer Studio, SAP Enterprise Portal, Web services, J2EE, and ABAP. You may reach him via email at karl.kessler@sap.com.

SAP's Web Dynpro is a powerful programming model for Web user interfaces (UIs). What makes this tool even more productive is the way that it supports not only the development of new applications, but the enhancement of existing ones.

The overall goal of this two-part article is to show you how to develop flexible Web Dynpro ABAP applications and enhance existing ones using the latest enhancement techniques from SAP. Part 1 begins by explaining the SAP ERP enhancement strategy. Then, it walks you through the development of a custom Web Dynpro application that is built around a BAPI service call. I use a tool that greatly simplifies this process called the Service Wizard. This process includes creating the Web Dynpro context that models the import and export parameters of the service call and thus builds the basis to derive corresponding UI elements that will be displayed by one of the application views that you develop in this article.

Part 2 explains how to enhance your new Web Dynpro application, planning the enhancement and adding a custom view with its own context so the enhanced application accurately extends the original application in a meaningful way.

In addition to these, you find wizards to generate Web Services from function modules and a lot more.

The SAP ERP enhancement strategy

SAP executives have repeatedly stressed the importance of the SAP ERP strategy at many conferences this year. Let me summarize the strategy in the following statements:

- SAP ERP 6.0 is the "go to" release for SAP ERP customers. SAP ERP 6.0 is built on a stable application core, furthermore allowing you to

Note!

The ABAP Workbench is a collection of integrated tools, such as Class Builder, ABAP Dictionary, Screen Painter, and Menu Painter, which you can launch via transaction code SE80. The design time environment is represented in the SAP GUI like any other transaction. The ABAP Workbench includes several wizards, each supporting a particular purpose. For example:

- The Service Wizard, as the name suggests, helps you to build a Web Dynpro UI around a service call (e.g., a BAPI or Web service). The Service Wizard automatically derives all of the type information for you.
- The Web Dynpro Code Wizard makes some of the tedious coding tasks easier. It is more of a utility that makes typing long sequences of code quicker.

bring in innovation in a non-disruptive, continuous manner via enhancement packages.

- Enhancement packages deliver innovations in new UI-centric scenarios in Web Dynpro that simplify business processes running SAP NetWeaver Portal.
- Enhancement packages deliver content for the enterprise service-oriented architecture (enterprise SOA) and SOA-based applications. The Enterprise Services Repository (ESR) contains the enterprise SOA content, which is structured in a process-centric fashion in the form of enterprise services bundles published on SDN (http://sdn.sap.com). This makes it easier to identify all of the services related to a particular business process scenario.
- The industry solutions use enhancement packages to deliver industry-specific business process extensions and process variants that offer unique value and differentiation.
- No customer is forced to implement enhancement packages. The customer, following his or her roadmap of continuous innovation, defines and determines the adoption path. However, if you adhere to the enhancement packages strategy, there is a huge potential to reduce upgrade efforts. It enables you to identify the area in which you want to innovate and then activate the newly delivered functions and content in a pick-and-choose fashion.
- SAP Enhancement Package 3 for SAP ERP 6.0 went to customers during Ramp-Up and met with great success as measured by customer satisfaction. SAP Enhancement Package 4 for SAP ERP 6.0 should go to Ramp-Up by the time this article is published. All of the other members of the SAP Business Suite such as SAP Customer Relationship Management (SAP CRM) and SAP Supplier Relationship Management (SAP SRM) will follow SAP ERP in pioneering the enhancement packages approach.
- The Switch and Enhancement Frameworks residing in the ABAP stack of SAP NetWeaver provide the technical basis of the Enhancement Framework strategy.[1]

Note!

The terms “innovation” and “managing the speed of innovation” were used at several SAP conferences, including SAPPHIRE 2008 Orlando, to distinguish between the accelerated innovation path for topics such as Composition and Business Process Management and a more conservative approach (called *continuous innovation*) to introduce innovation via enhancement packages delivered on a stable core.

[1] See my article, “Introducing the switch and enhancement framework — consolidating industry solutions with the mySAP ERP core” (*SAP Professional Journal*, March/April 2006).

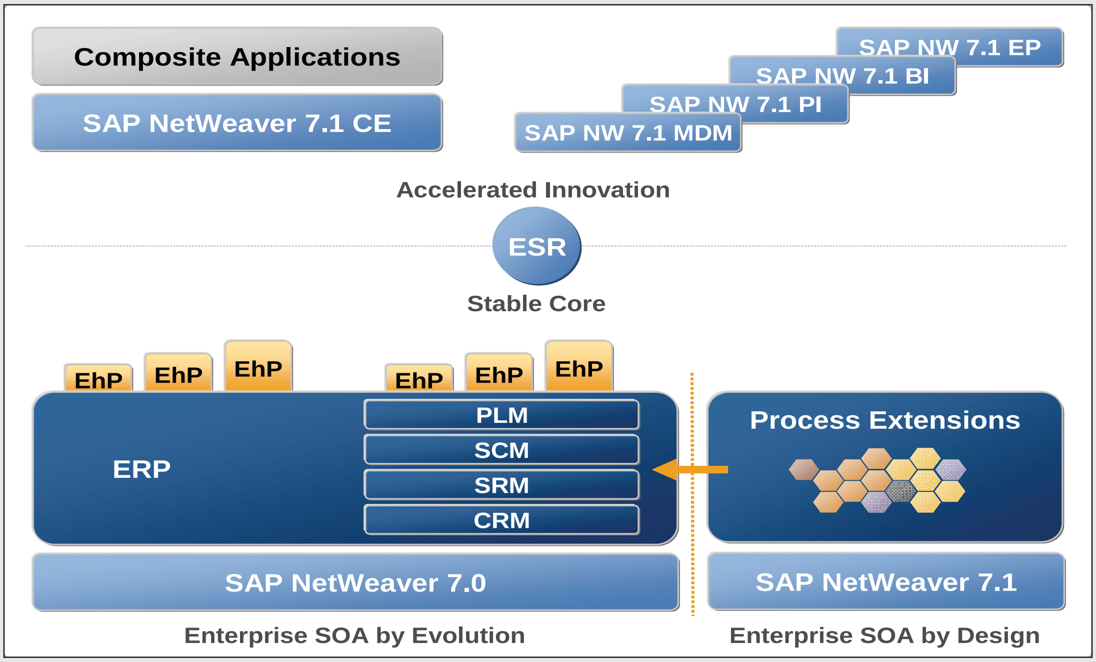


Figure 1 Details of the SAP ERP enhancement strategy

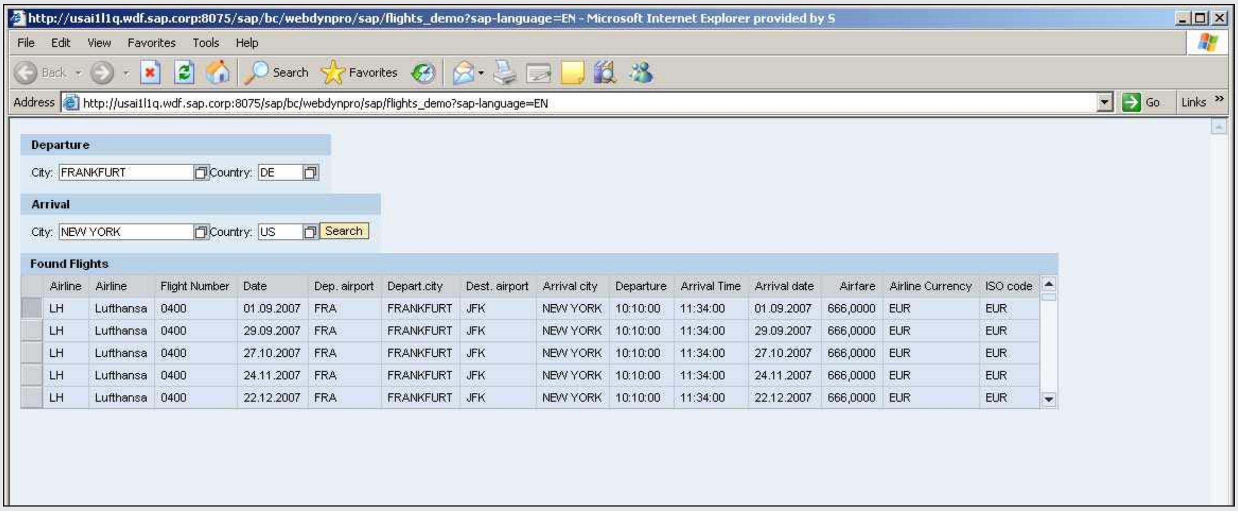
The SAP ERP strategy helps customers to choose the optimal roadmap for innovation in their system landscape and processes. Since a smooth approach to innovating in mission-critical environments, such as SAP ERP, is of paramount importance, enhancement packages are the right choice. They allow you to schedule innovation at your own pace and not be forced to do so by technical implementation constraints. Broad adoption by the SAP Business Suite makes the decision easy because then you can consistently apply the benefits of enhancement packages to all phases of your business process value chain.

Figure 1 illustrates the following facts: SAP ERP sits on top of a stable core with SAP NetWeaver 7.0 and SAP ERP offering optional enhancement packages. In addition, SAP ERP 6.0 is interoperable with standalone SAP NetWeaver 7.1 systems, such as SAP NetWeaver Composition Environment (SAP NetWeaver CE) 7.1 and SAP NetWeaver Process Integration (SAP NetWeaver PI) 7.1. The interoperability allows you to drive innovation at various levels: smoothly on the SAP ERP level in a pick-and-choose approach — only activate what you really need — and more aggressively in the area of composition and process integration to quickly adopt the latest standards in, e.g., Java. The figure shows how you enable innovation with SAP NetWeaver 7.0 and SAP NetWeaver 7.1.[2]

Developing a standard Web Dynpro application

The application you're going to develop is the sample flight application shown in **Figure 2** on the next page. (You may already know this sample flight data model from SAP Internet Demonstration and Evaluation System [SAP IDES] in the SAP R/3 system.)

[2] All figures are © SAP AG.



http://usai1l1q.wdf.sap.corp:8075/sap/bc/webdynpro/sap/flights_demo?sap-language=EN - Microsoft Internet Explorer provided by S
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links

http://usai1l1q.wdf.sap.corp:8075/sap/bc/webdynpro/sap/flights_demo?sap-language=EN

Departure

City: FRANKFURT Country: DE

Arrival

City: NEW YORK Country: US Search

Found Flights

Airline	Airline	Flight Number	Date	Dep. airport	Depart.city	Dest. airport	Arrival city	Departure	Arrival Time	Arrival date	Airfare	Airline Currency	ISO code
LH	Lufthansa	0400	01.09.2007	FRA	FRANKFURT	JFK	NEW YORK	10:10:00	11:34:00	01.09.2007	666,0000	EUR	EUR
LH	Lufthansa	0400	29.09.2007	FRA	FRANKFURT	JFK	NEW YORK	10:10:00	11:34:00	29.09.2007	666,0000	EUR	EUR
LH	Lufthansa	0400	27.10.2007	FRA	FRANKFURT	JFK	NEW YORK	10:10:00	11:34:00	27.10.2007	666,0000	EUR	EUR
LH	Lufthansa	0400	24.11.2007	FRA	FRANKFURT	JFK	NEW YORK	10:10:00	11:34:00	24.11.2007	666,0000	EUR	EUR
LH	Lufthansa	0400	22.12.2007	FRA	FRANKFURT	JFK	NEW YORK	10:10:00	11:34:00	22.12.2007	666,0000	EUR	EUR

Figure 2 Sample of a Web Dynpro flight application

Note!

There are numerous introductions on how to build a UI in Web Dynpro ABAP. Ulrich Hoffman provides a good basic overview in the article "Get started developing Web-native custom SAP applications with Web Dynpro for ABAP" (*SAP Professional Journal*, July/August 2007). Therefore, I will only briefly sketch how to develop a standard Web Dynpro application with the tools that are part of ABAP Workbench.

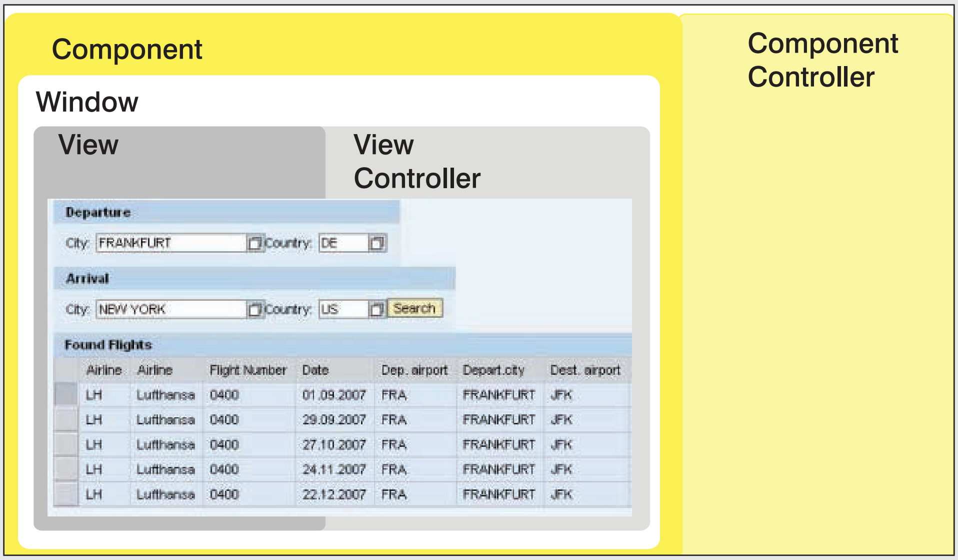
At the top of the screen are several search fields, such as departure and arrival destination. Each field is implemented as a city/country pair with *value help* (if you press the drop-down button attached to the input field, the system displays a list of values from which to choose). Lower on the screen is a table that contains all of the flights that match the search criteria when the user clicks on the Search button.

Figure 3 shows the Web Dynpro blueprint for this flight application.

The Web Dynpro application is made of a Web Dynpro component that contains a window consisting of one Web Dynpro view with the necessary UI elements. Since you can manipulate each Web Dynpro artifact within the application's code, each artifact has a controller attached (i.e., basically an ABAP class offering methods to interact with the artifacts).

When you create a Web Dynpro component in transaction SE80 (Object Navigator), the system asks you to provide a short description, as shown in **Figure 4**.

Then, the system automatically creates a couple of standard Web Dynpro artifacts, including a main window and a view. You can rename the artifacts, but for the sake of simplicity, use the defaults. The system then asks if you have any changes and, if so, what the transport settings that are associated with your Web Dynpro component are. This is a standard ABAP Workbench feature whenever you create or change a program, a class, a Web Dynpro component, etc. When you enter the change mode, the pop-up dialog appears requesting the transport request under which the change is being made (Change recording). If you fill it out once, the system won't ask again until you release the transport request to the consolidation system (i.e., test or production). To keep things



Departure

City: FRANKFURT Country: DE

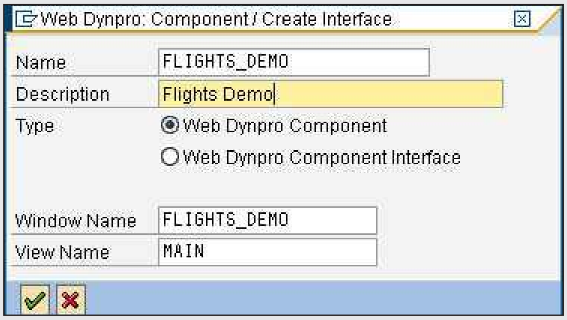
Arrival

City: NEW YORK Country: US

Found Flights

Airline	Airline	Flight Number	Date	Dep. airport	Depart.city	Dest. airport
LH	Lufthansa	0400	01.09.2007	FRA	FRANKFURT	JFK
LH	Lufthansa	0400	29.09.2007	FRA	FRANKFURT	JFK
LH	Lufthansa	0400	27.10.2007	FRA	FRANKFURT	JFK
LH	Lufthansa	0400	24.11.2007	FRA	FRANKFURT	JFK
LH	Lufthansa	0400	22.12.2007	FRA	FRANKFURT	JFK

Figure 3 Structure of the sample Web Dynpro flight application



Web Dynpro: Component / Create Interface

Name	FLIGHTS_DEMO
Description	Flights Demo
Type	● Web Dynpro Component ○ Web Dynpro Component Interface
Window Name	FLIGHTS_DEMO
View Name	MAIN

Figure 4 Creating the Web Dynpro component named FLIGHTS_DEMO

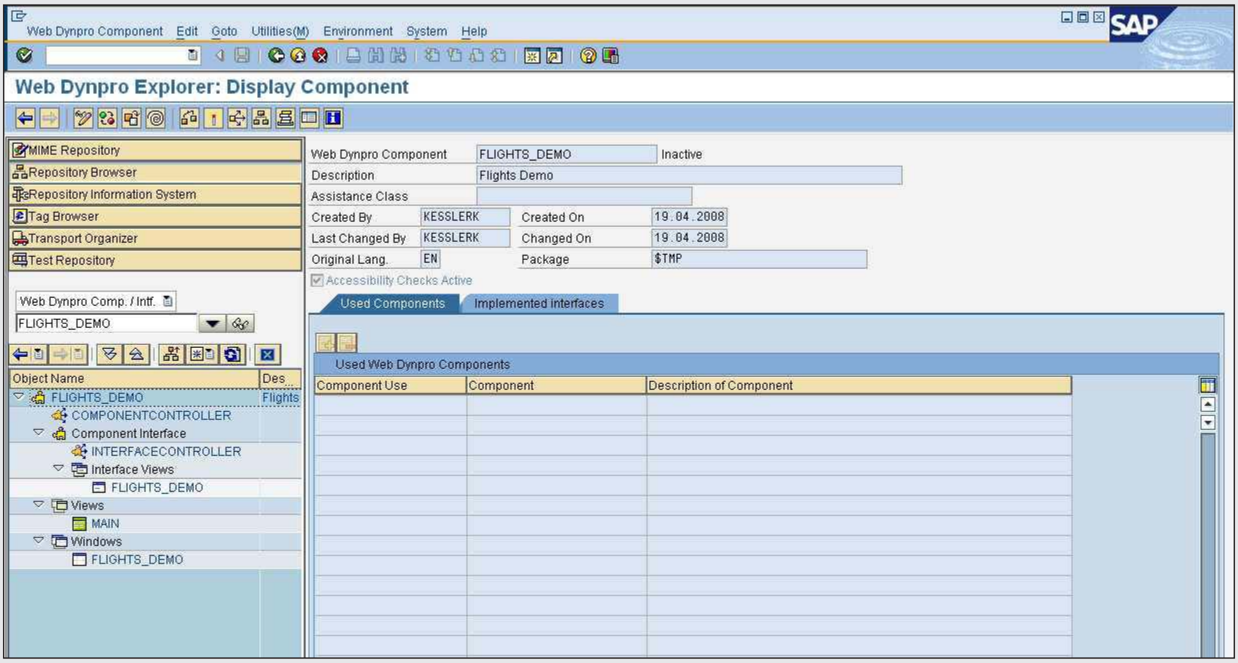
simple, store the new object as a local object. See **Figure 5** on the next page for the initial structure of the Web Dynpro component.

The new Web Dynpro component is shown in Web Dynpro Explorer. I expanded the complete structure of the Web Dynpro application (in the directory tree on the left) so you can see the initial artifacts of the application (the view MAIN and the window FLIGHTS_DEMO). A Web Dynpro application is a URL that enables you to launch the Web Dynpro component. This component is modularized and you can also use it in other situations (e.g., embedded in a higher-order component).

There are several different approaches for building a Web Dynpro UI from scratch. Whenever you can rely on the building blocks already available, such as a service call and the corresponding service implementation, I recommend that you derive as much as possible from the service definition itself. The Service Wizard, which is available in the Web Dynpro tools of the ABAP Workbench, supports this approach.

Using the Service Wizard to develop context

To create the Web Dynpro context you need for the example flight application, you use the Service Wizard. This service uses a BAPI function call, or an ABAP Objects method call, to derive suitable Web Dynpro context based on the service's import and



Web Dynpro Explorer: Display Component

Web Dynpro Component	FLIGHTS_DEMO	Inactive	
Description	Flights Demo		
Assistance Class			
Created By	KESSLERK	Created On	19.04.2008
Last Changed By	KESSLERK	Changed On	19.04.2008
Original Lang.	EN	Package	$TMP

☑ Accessibility Checks Active

Used Components | Implemented Interfaces

Used Web Dynpro Components

Component Use	Component	Description of Component

FLIGHTS_DEMO

Object Name	Des...
FLIGHTS_DEMO	Flights
COMPONENTCONTROLLER	
Component Interface	
INTERFACECONTROLLER	
Interface Views	
FLIGHTS_DEMO	
Views	
MAIN	
Windows	
FLIGHTS_DEMO	

Figure 5 The initial state of the newly created Web Dynpro component

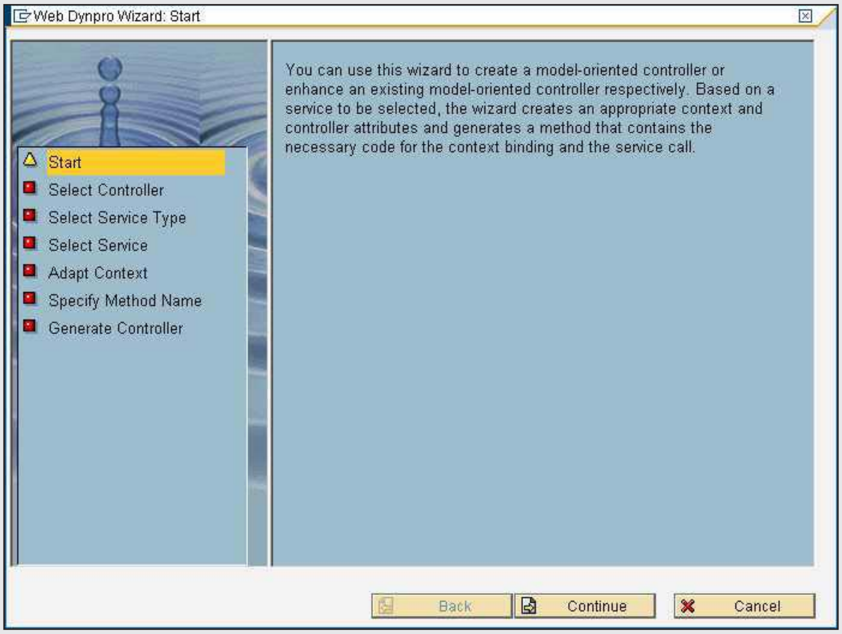
export parameter types. That way you can easily build a UI instead of developing everything from scratch. Follow these steps:

1. To get to the start screen of the Service Wizard, as shown in **Figure 6**, right-click on the FLIGHTS_DEMO root node of your Web Dynpro component (**Figure 5**) to get the context menu, click on Create, and then click on Service Call.

2. Click on Continue to get to the wizard's Select Controller screen, as shown in **Figure 7**.

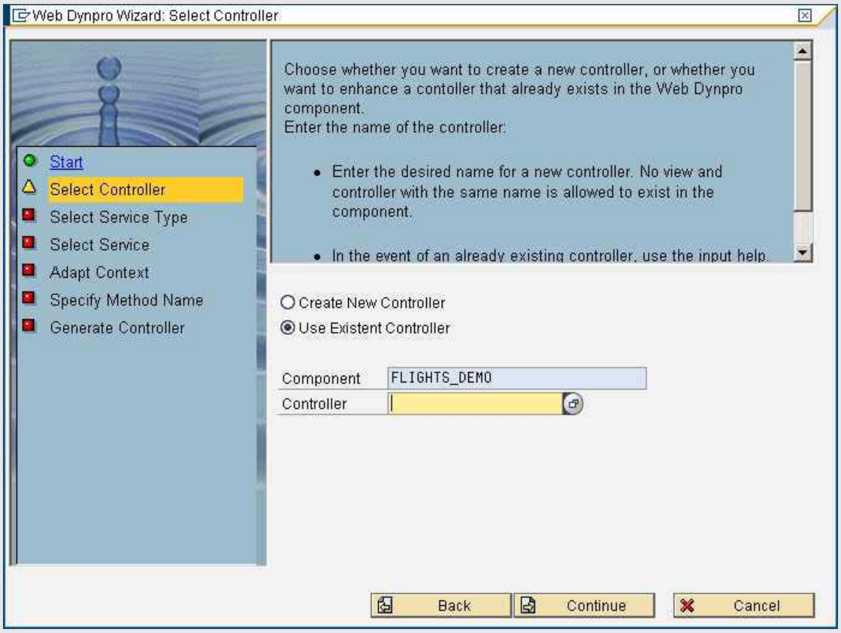
3. You can choose to create a controller for the application from scratch or use an existing one, which you can modify to meet the needs of the application you are developing. For this example, select the Use Existent Controller option. Then, you can either click on the drop-down button to the right of the Controller field or press F4 to display a list of existing controllers. In the current Web Dynpro component, there is only one predefined controller, COMPONENTCONTROLLER. In a more sophisticated application, you may want to introduce additional levels of abstraction by using custom controllers, but you won't need them for this application. Select the predefined controller, and click on Continue to get to the wizard's Select Service Type screen, as shown in **Figure 8** on page 10.

4. Next, you need to specify the type of service. In this case, select Function Module since you call local functionality on the application server. You could use Class Method (a method of an ABAP class) or Web Service Proxy, but the basic procedure remains the same. (When you call a Web service, you typically cross a machine boundary and invoke a service on another machine. That's the first step toward compiling a simple composite application that consumes services.) Click on Continue to get to the wizard's Select Service screen, as shown in **Figure 9** on page 11.



You can use this wizard to create a model-oriented controller or enhance an existing model-oriented controller respectively. Based on a service to be selected, the wizard creates an appropriate context and controller attributes and generates a method that contains the necessary code for the context binding and the service call.

Figure 6 Starting the Service Wizard



Choose whether you want to create a new controller, or whether you want to enhance a contoller that already exists in the Web Dynpro component.
Enter the name of the controller:

- Enter the desired name for a new controller. No view and controller with the same name is allowed to exist in the component.
- In the event of an already existing controller, use the input help.

○ Create New Controller
● Use Existent Controller

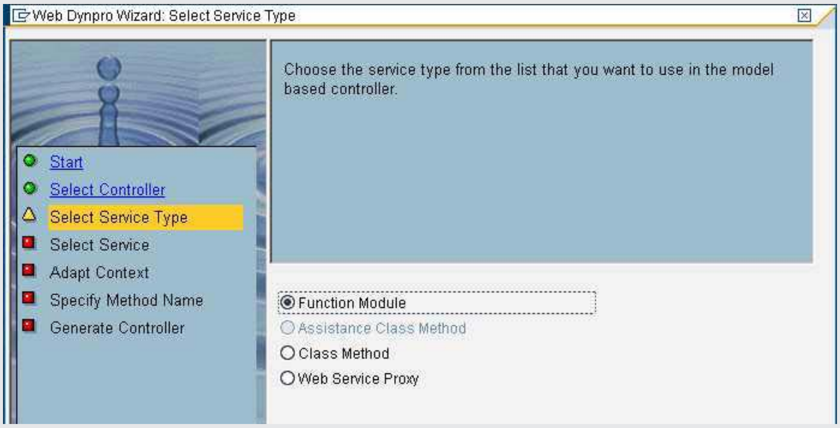
Component FLIGHTS_DEMO
Controller

Figure 7 Using an existing Web Dynpro component

Definition!

The Web Dynpro program model is based on a hierarchy of variables — called the *Web Dynpro context.* These variables are attached to the layered components of a Web Dynpro application and ultimately are used to display data in the window or receive input from the end user. You can map context variables to other context variables, defining how the data flows between them at runtime.

Web Dynpro context variables are bound to a controller. Ultimately, context variables are variables that need to be defined in the scope of an ABAP class. The controller is such a class. There are different kinds of controllers, including window controllers and view controllers. View controllers manage views, which in turn have input and output fields. Windows manage views and their relationships. The browser displays the window which acts as a container for the views (**Figure 3**). The art of Web Dynpro programming mainly consists of carefully designing the right portions of interaction and establishing a suitable controller hierarchy (which leads to the previously discussed context hierarchy).



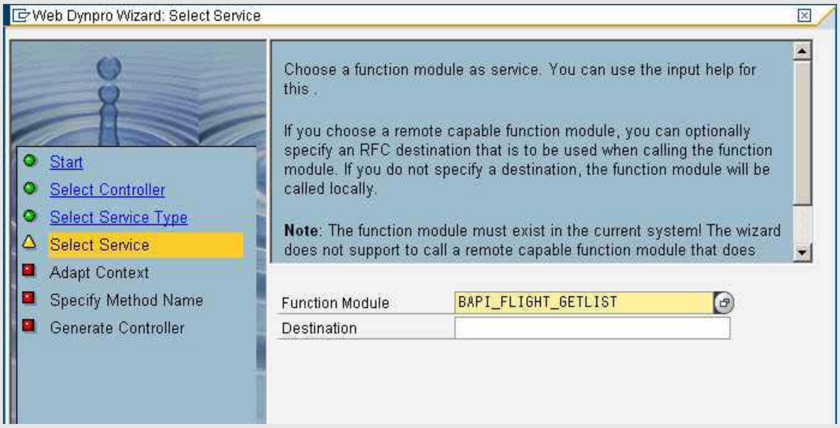
Choose the service type from the list that you want to use in the model based controller.

◉ Function Module
○ Assistance Class Method
○ Class Method
○ Web Service Proxy

Figure 8 Specifying the service type

5. Next, select the function module BAPI_ FLIGHT_GETLIST. You can find this BAPI in transaction SE37 (ABAP Function Modules) and test it there to see whether the flight tables are filled in your development system. Leave the Destination field empty so the flight application will run on your local application server. Then, click on Continue to get to the wizard's Adapt Node Name screen, as shown in **Figure 10**.

6. Here, you see the parameters of the BAPI_ FLIGHT_GETLIST function call. You need to decide which parameters you want to use as controller attributes or context nodes (e.g., the parameters for departure and arrival destinations). It's essential to switch the object type to context node for all the parameters so that the system recognizes the data as content. To change the object type, click on the drop-down icon in the ObjectType column, and select Context (Node/



Choose a function module as service. You can use the input help for this .

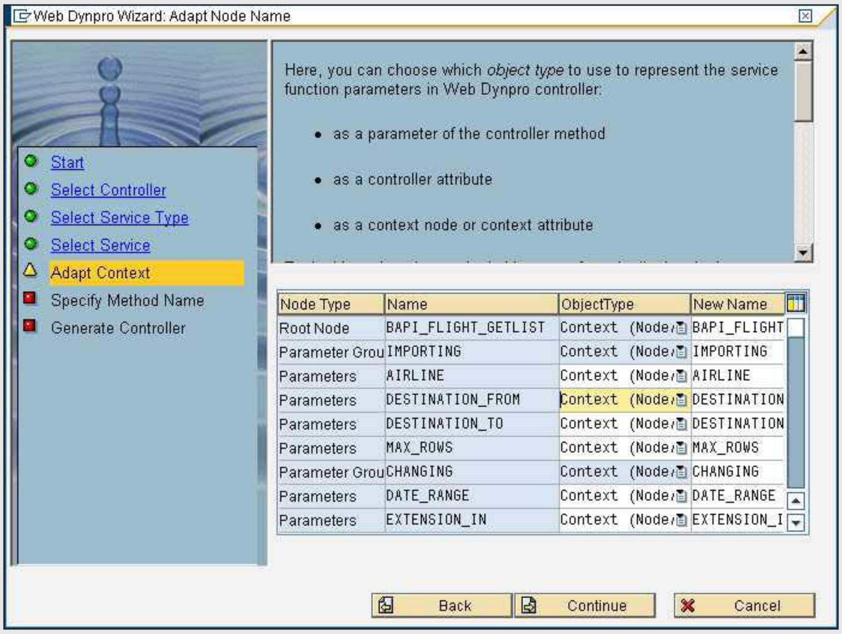
If you choose a remote capable function module, you can optionally specify an RFC destination that is to be used when calling the function module. If you do not specify a destination, the function module will be called locally.

Note: The function module must exist in the current system! The wizard does not support to call a remote capable function module that does

Function Module BAPI_FLIGHT_GETLIST

Destination

Figure 9 Selecting the function module BAPI_FLIGHT_GETLIST



Here, you can choose which *object type* to use to represent the service function parameters in Web Dynpro controller:

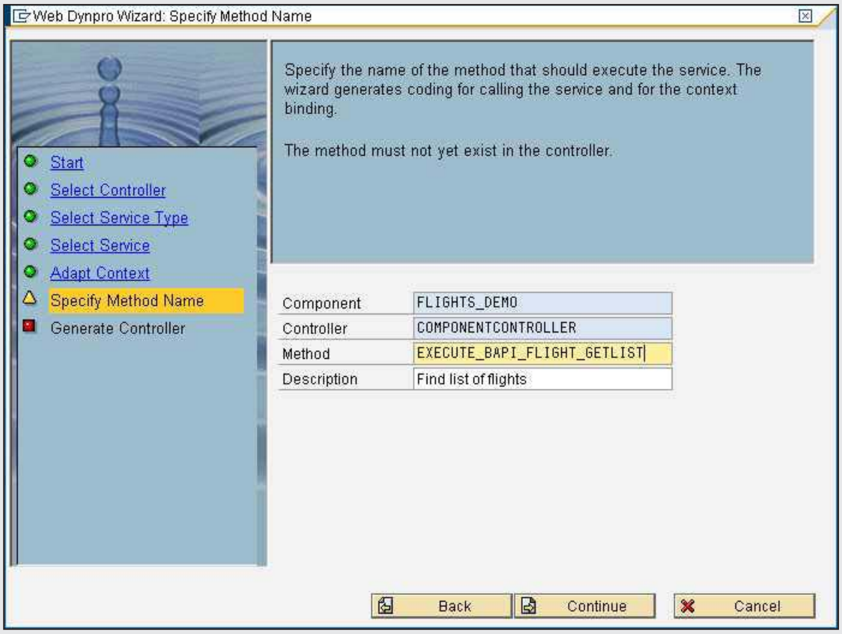
- as a parameter of the controller method
- as a controller attribute
- as a context node or context attribute

Node Type	Name	ObjectType	New Name
Root Node	BAPI_FLIGHT_GETLIST	Context (Node/	BAPI_FLIGHT
Parameter Grou	IMPORTING	Context (Node/	IMPORTING
Parameters	AIRLINE	Context (Node/	AIRLINE
Parameters	DESTINATION_FROM	Context (Node/	DESTINATION
Parameters	DESTINATION_TO	Context (Node/	DESTINATION
Parameters	MAX_ROWS	Context (Node/	MAX_ROWS
Parameter Grou	CHANGING	Context (Node/	CHANGING
Parameters	DATE_RANGE	Context (Node/	DATE_RANGE
Parameters	EXTENSION_IN	Context (Node/	EXTENSION_I

Figure 10 Selecting the parameters for use as controller attributes or context nodes

Attribute). Then, click on Continue to move to the wizard's Specify Method Name screen, as shown in **Figure 11** on the next page.

7. Next, the wizard suggests a method name, EXECUTE_BAPI_FLIGHT_GETLIST, which will call the function module BAPI_FLIGHT_



Web Dynpro Wizard: Specify Method Name

Specify the name of the method that should execute the service. The wizard generates coding for calling the service and for the context binding.

The method must not yet exist in the controller.

Component	FLIGHTS_DEMO
Controller	COMPONENTCONTROLLER
Method	EXECUTE_BAPI_FLIGHT_GETLIST
Description	Find list of flights

Figure 11 Naming the method that calls the function module

GETLIST. Verify the information displayed and click on Continue.

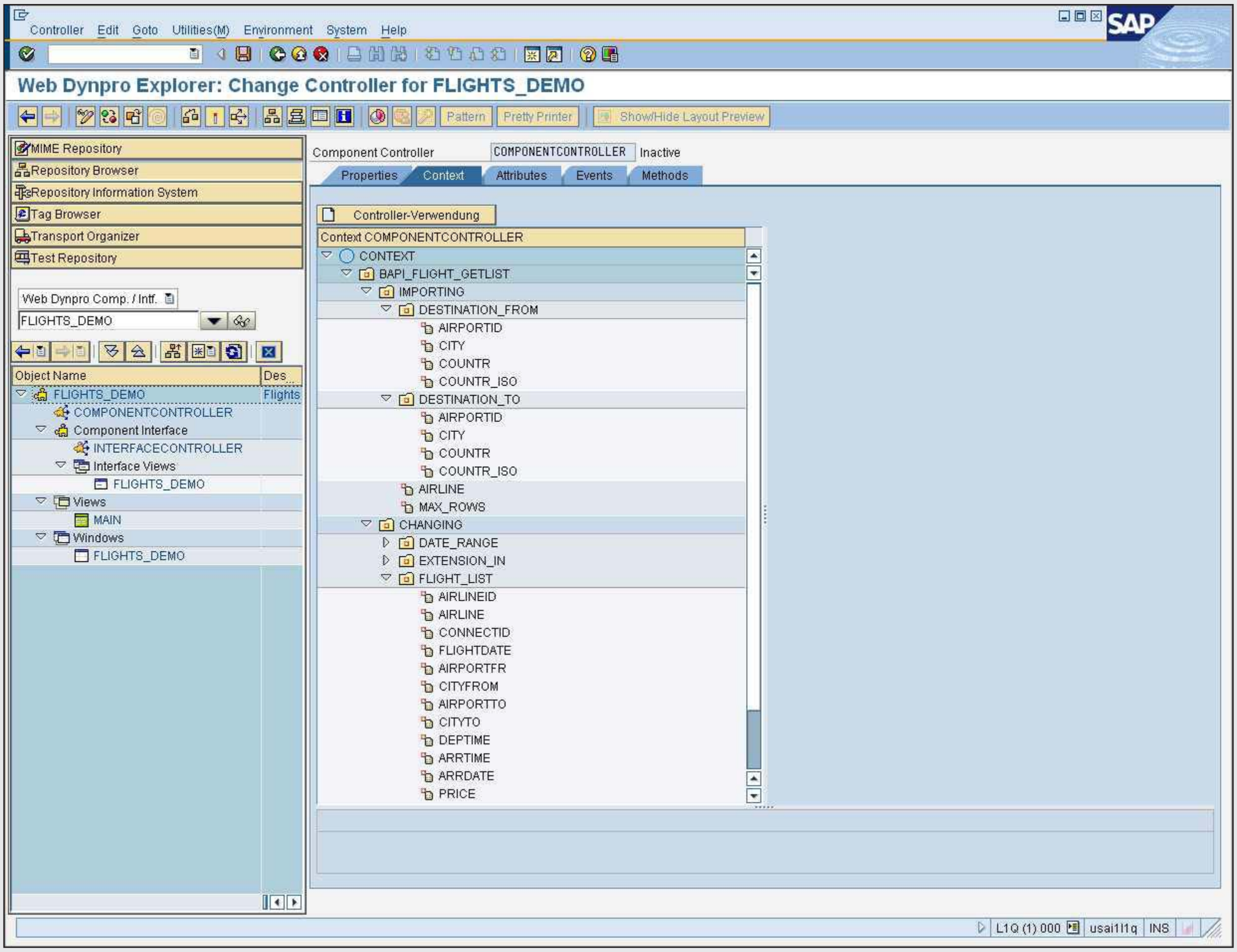
8. On the last screen, the wizard's Generate Controller, you have the final choice to complete the wizard or to cancel the wizard. The result of the wizard is shown in **Figure 12**.

The context hierarchy displayed is completely derived from the import/export parameters structure of the underlying BAPI. All of the type information of the BAPI is now present in the context definition of the corresponding Web Dynpro component. That information serves as a starting point when you define the Web Dynpro views that make certain parts of the component controller's context visible. Now, define the context for the UI.

Create the view context

You need to refine the context to fit your UI. The chosen BAPI has a lot of parameters, but you only need a few of them on the UI level, such as CITY and COUNTR fields for the departure and arrival destination (the BAPI's interface supports ISO codes, but we don't use them here). On the result side, not all parameters that are returned need to be considered. In our example, we focus on the flight list that fulfills the search criteria. The other parameters such as date range and extension are not used. **Figure 13** shows the next milestone in the Web Dynpro blueprint for this application.

You need to map the nodes in the component controller context (which basically are variables that store the export variables inside the Web Dynpro component before the BAPI is called and the result sets once the BAPI has returned) to the nodes of the view controller context that serve as input and output fields for the UI. The nodes of the view controller context are then bound to UI elements by associating the context variables to properties of the UI controls, such as a table or search input field. The context variables of the underlying Web Dynpro component are



Web Dynpro Explorer: Change Controller for FLIGHTS_DEMO

Web Dynpro Comp. / Intf.: FLIGHTS_DEMO

Component Controller: COMPONENTCONTROLLER Inactive

```
Object Name                                   Des...
▽ FLIGHTS_DEMO                                Flights
    COMPONENTCONTROLLER
  ▽ Component Interface
      INTERFACECONTROLLER
    ▽ Interface Views
        FLIGHTS_DEMO
  ▽ Views
      MAIN
  ▽ Windows
      FLIGHTS_DEMO
```

```
Context COMPONENTCONTROLLER
▽ CONTEXT
  ▽ BAPI_FLIGHT_GETLIST
    ▽ IMPORTING
      ▽ DESTINATION_FROM
          AIRPORTID
          CITY
          COUNTR
          COUNTR_ISO
      ▽ DESTINATION_TO
          AIRPORTID
          CITY
          COUNTR
          COUNTR_ISO
        AIRLINE
        MAX_ROWS
    ▽ CHANGING
      ▷ DATE_RANGE
      ▷ EXTENSION_IN
      ▽ FLIGHT_LIST
          AIRLINEID
          AIRLINE
          CONNECTID
          FLIGHTDATE
          AIRPORTFR
          CITYFROM
          AIRPORTTO
          CITYTO
          DEPTIME
          ARRTIME
          ARRDATE
          PRICE
```

L1Q (1) 000 usai1l1q INS

Figure 12 The Web Dynpro context for the BAPI call

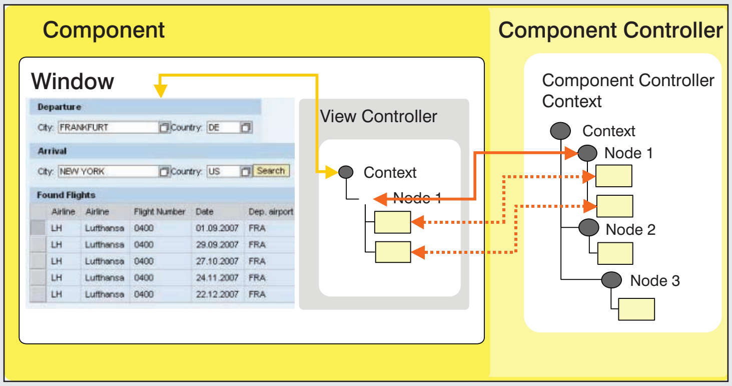
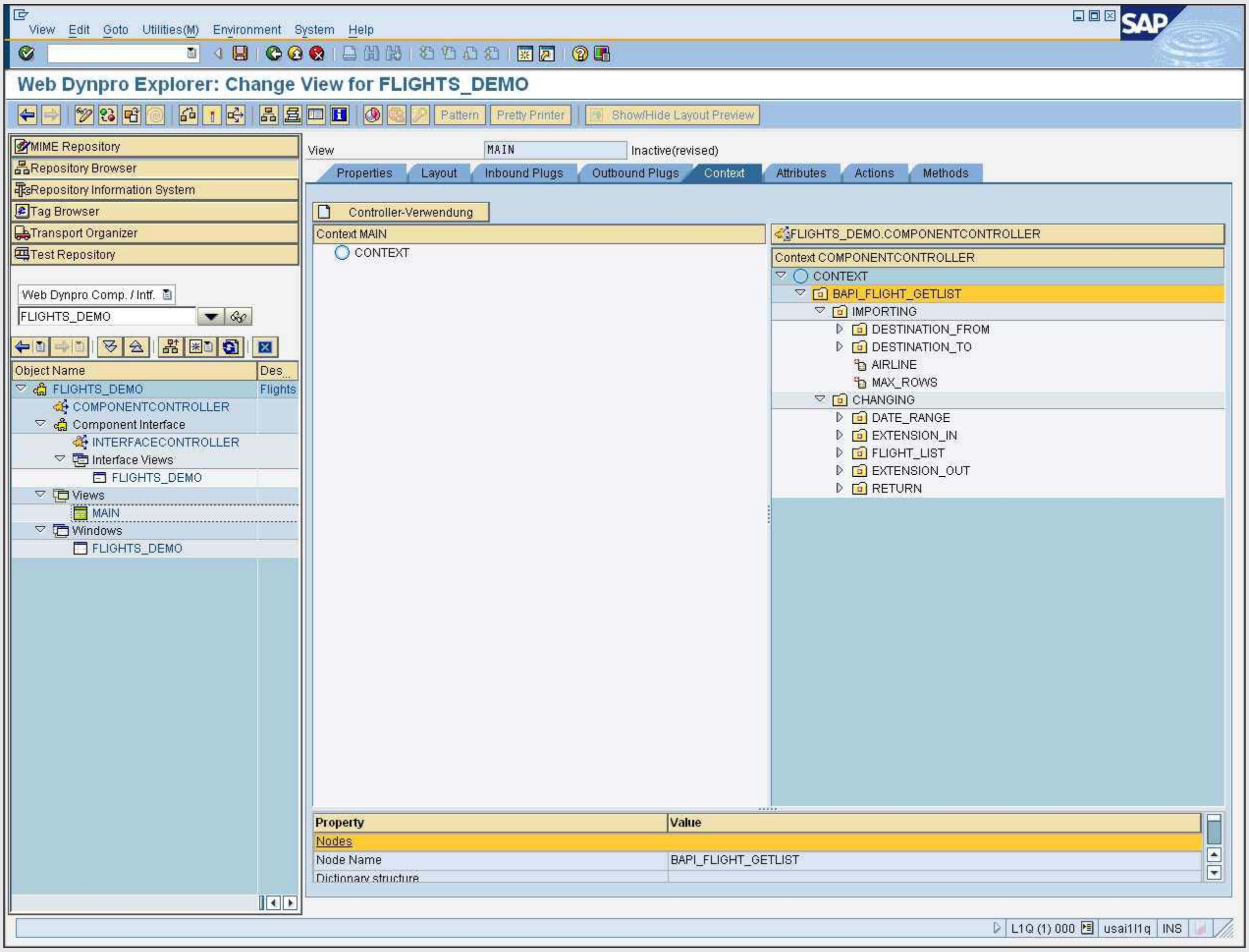

Component
Component Controller
Window
Departure
City: FRANKFURT
Country: DE
Arrival
City: NEW YORK
Country: US
Search
Found Flights
Airline Airline Flight Number Date Dep. airport
LH Lufthansa 0400 01.09.2007 FRA
LH Lufthansa 0400 29.09.2007 FRA
LH Lufthansa 0400 27.10.2007 FRA
LH Lufthansa 0400 24.11.2007 FRA
LH Lufthansa 0400 22.12.2007 FRA
View Controller
Context
Node 1
Component Controller Context
Context
Node 1
Node 2
Node 3

Figure 13 The component controller context of the Web Dynpro application



Property	Value
Nodes	
Node Name	BAPI_FLIGHT_GETLIST
Dictionary structure	

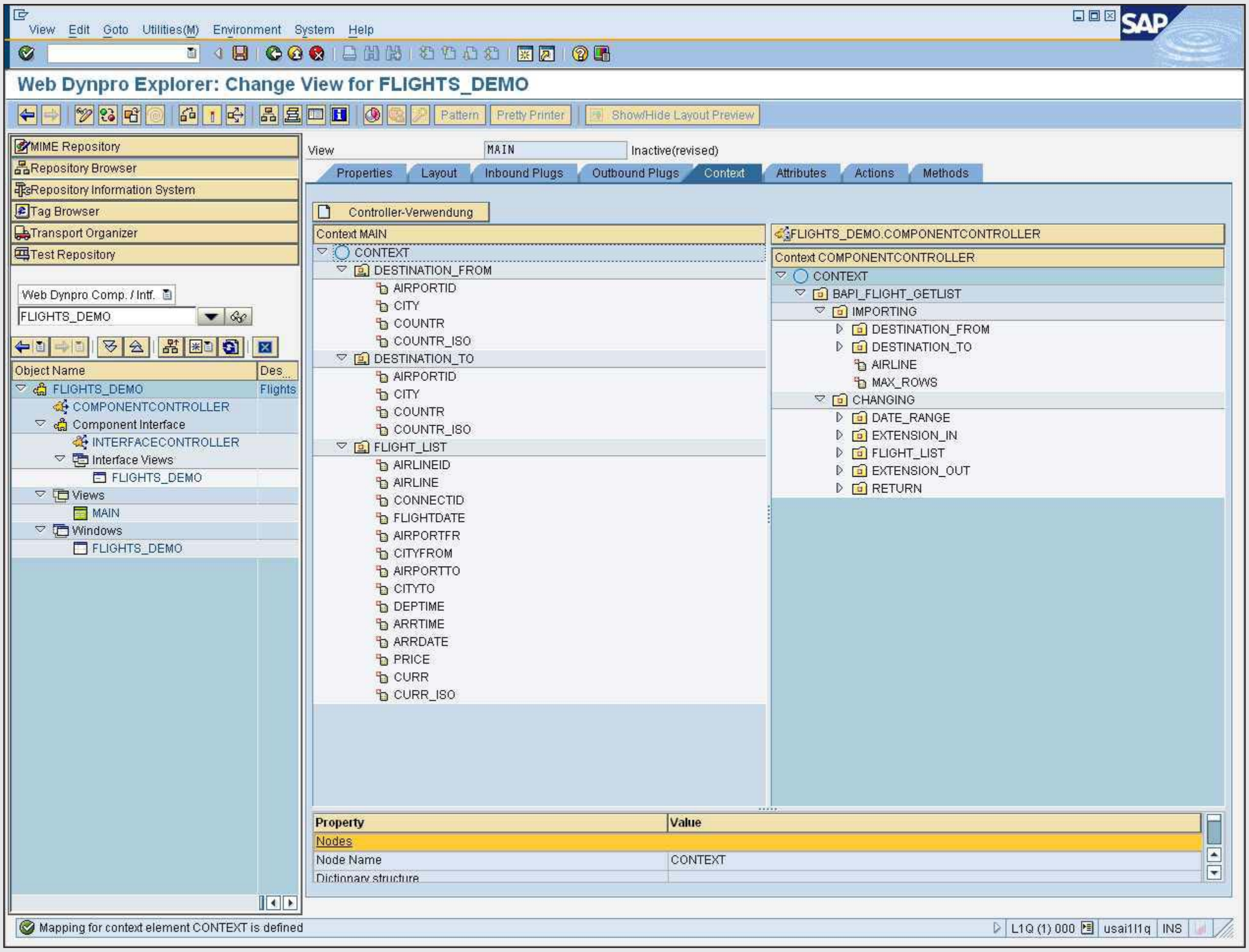
Figure 14 The final context for the MAIN view

first mapped to a suitable view context. The view context in turn is mapped to the UI elements visible in the browser screen.

First, double-click on the view MAIN shown on the left in **Figure 14**, and then click on the Context tab shown on the right. The context of the component controller is visible on the right.

The actual mapping is easy. Just drag the context nodes you want to use in the UI from the COMPONENTCONTROLLER's context column to the view MAIN's context column. A pop-up screen appears asking whether you want to copy and map the context nodes, which is exactly what you want. The node is copied from the component controller to the view controller and the two nodes are mapped making the data transfer possible during runtime. Click on Yes. The final view context is shown in **Figure 15**.

At this point, you can see that the nodes you dragged were removed from the COMPONENT-CONTROLLER's context and now appear in the



Web Dynpro Explorer: Change View for FLIGHTS_DEMO

View MAIN Inactive(revised)

Property	Value
Nodes	
Node Name	CONTEXT
Dictionary structure	

Mapping for context element CONTEXT is defined

Figure 15 The final context view for the UI

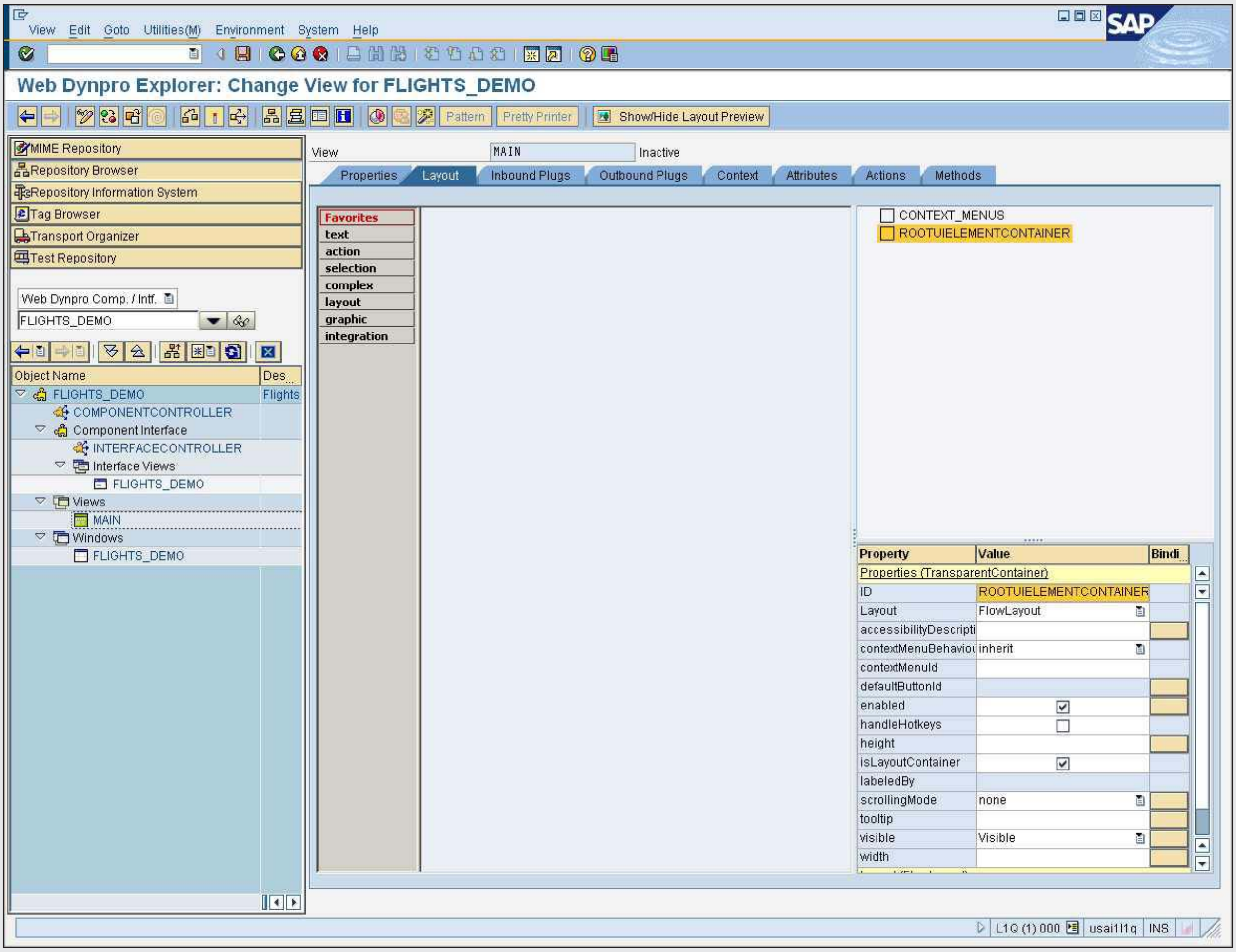
view controller MAIN's context. This enables you to quickly design the corresponding UI.

With the Web Dynpro context for the application ready, you can now create your actual UI.

Create a Web Dynpro UI

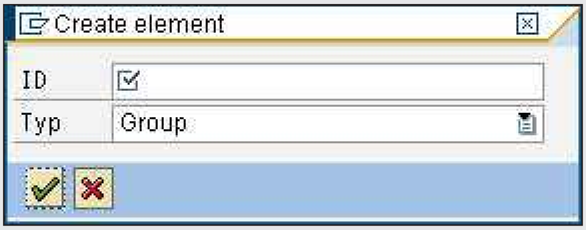
The UI of the flight application is composed of two UI element groups — each consisting of a pair of search fields, a table containing the results, and a button to trigger the BAPI call. Once you create one UI, you can use it as the basis of another UI. You create the UI using the WYSIWYG view designer in Web Dynpro Explorer on the Layout tab. Creating the UI is a 10-step process.

1. Start the view designer by clicking on the Layout tab. The view designer, as shown in **Figure 16** on the next page, is organized like many other tools. On the left, it has a palette of UI controls grouped into various categories (Favorites, text, action,



Property	Value	Bindi
Properties (TransparentContainer)		
ID	ROOTUIELEMENTCONTAINER	
Layout	FlowLayout	
accessibilityDescripti		
contextMenuBehavioı	inherit	
contextMenuId		
defaultButtonId		
enabled	☑	
handleHotkeys	□	
height		
isLayoutContainer	☑	
labeledBy		
scrollingMode	none	
tooltip		
visible	Visible	
width		

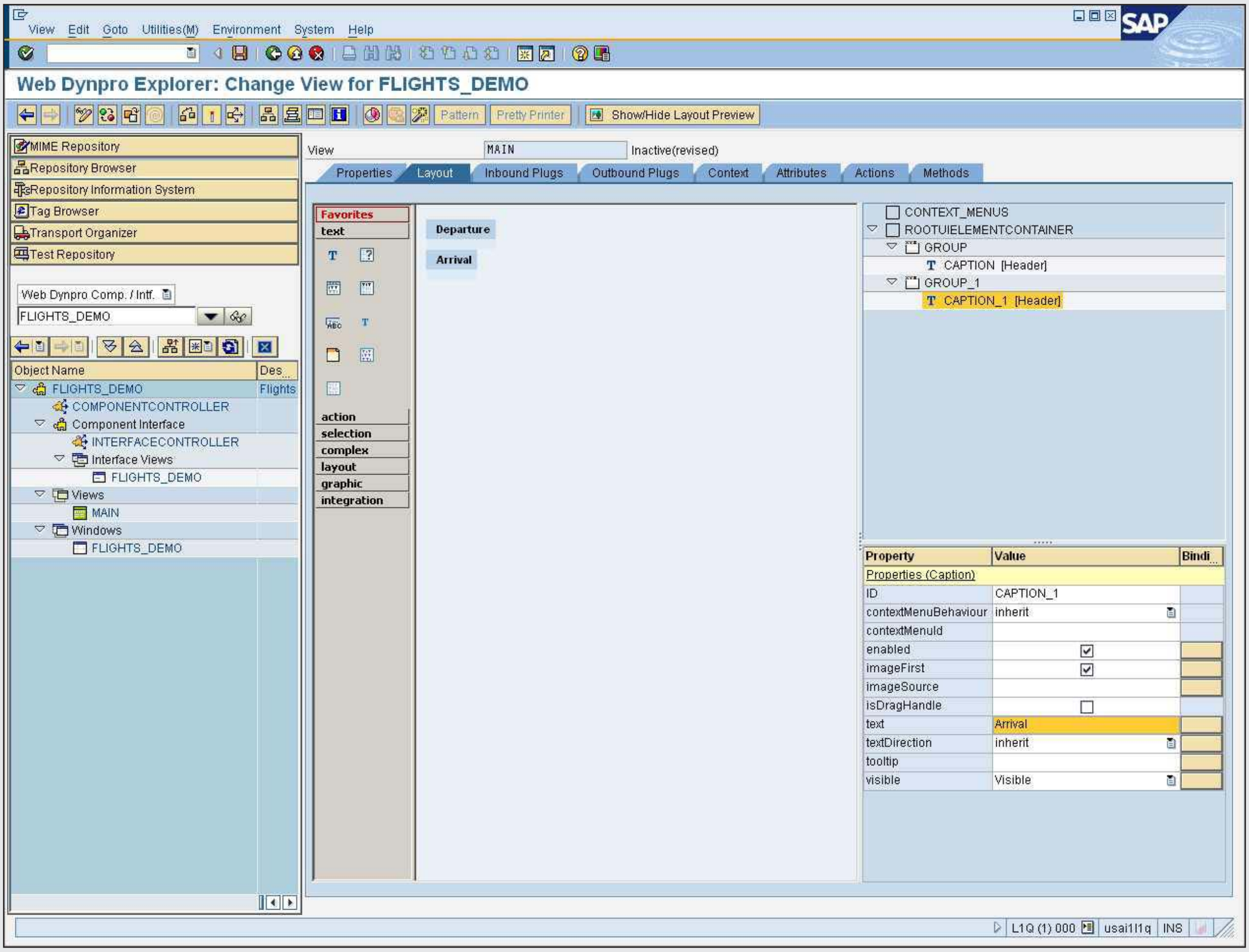
Figure 16 The view designer in Web Dynpro Explorer


Create element

ID	
Typ	Group

Figure 17 Creating a Group element

etc.). The middle section is technically an Internet browser control embedded in the ABAP Workbench screen that displays the view, but also allows you to drag and drop UI elements in a WYSIWYG fashion. Since it's a WYSIWYG area, you have a visual representation of what the UI looks like and how it functions. In the upper right is the control tree of the view (which is currently empty since you haven't added anything), and in the lower right is the Property box, which displays the attributes and properties for a given control.



Property	Value	Bindi...
Properties (Caption)		
ID	CAPTION_1	
contextMenuBehaviour	inherit	
contextMenuId		
enabled	☑	
imageFirst	☑	
imageSource		
isDragHandle	□	
text	Arrival	
textDirection	inherit	
tooltip		
visible	Visible	

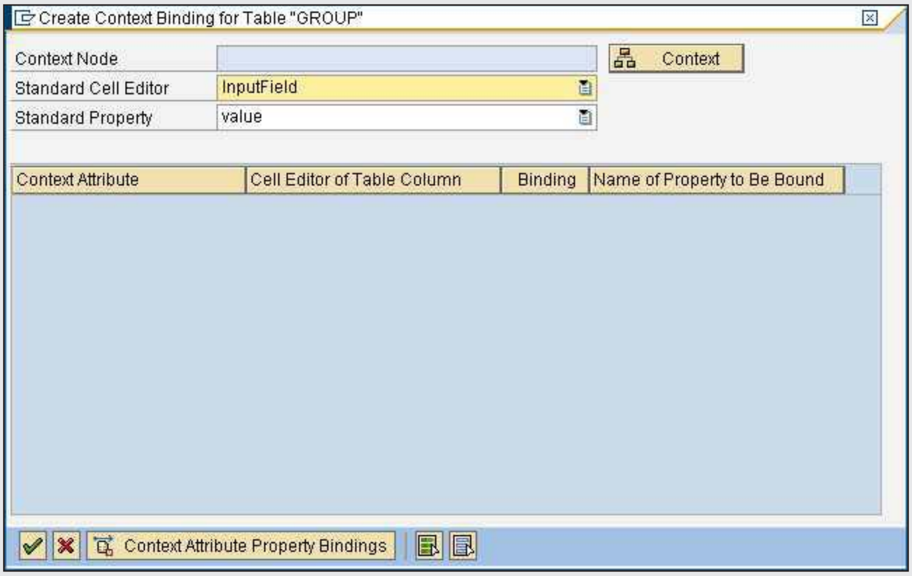
Figure 18 Setting up GROUP_1 for the UI

2. Right-click ROOTUIELEMENTCONTAINER in the control tree (upper right), and then select Insert → Element from the context menu. A dialog window appears to specify the type of control. Choose Group, as shown in **Figure 17**. (Each group contains a header element with a text attribute that is displayed as the group's title in the browser and the actual elements that belong to the group.) In the Property window for the GROUP header element, enter "Departure" as the text, which displays in the application UI.

3. Repeat Step 2 to create a second group, keeping the default group title (GROUP_1), but enter "Arrival" as the display text, as shown on the lower right of **Figure 18**.

Now you need to define the input fields for the search criteria and map them to the context nodes of the MAIN view:

4. Add a container form with the first pair of search fields using the context menu of the first group. **Figure 19** displays the window in which you define the context binding. The

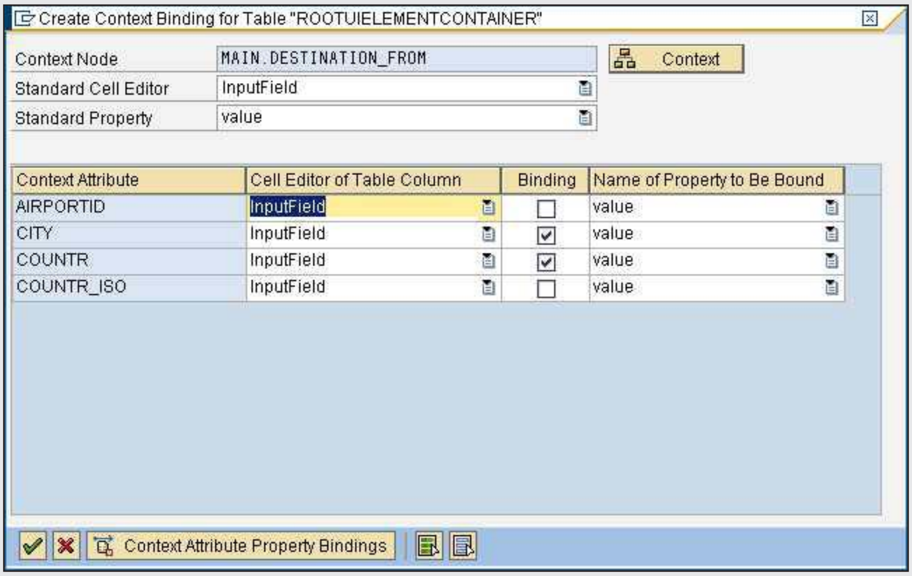


Create Context Binding for Table "GROUP"

Context Node	
Standard Cell Editor	InputField
Standard Property	value

Context Attribute	Cell Editor of Table Column	Binding	Name of Property to Be Bound

Figure 19 Context binding screen before defining the binding

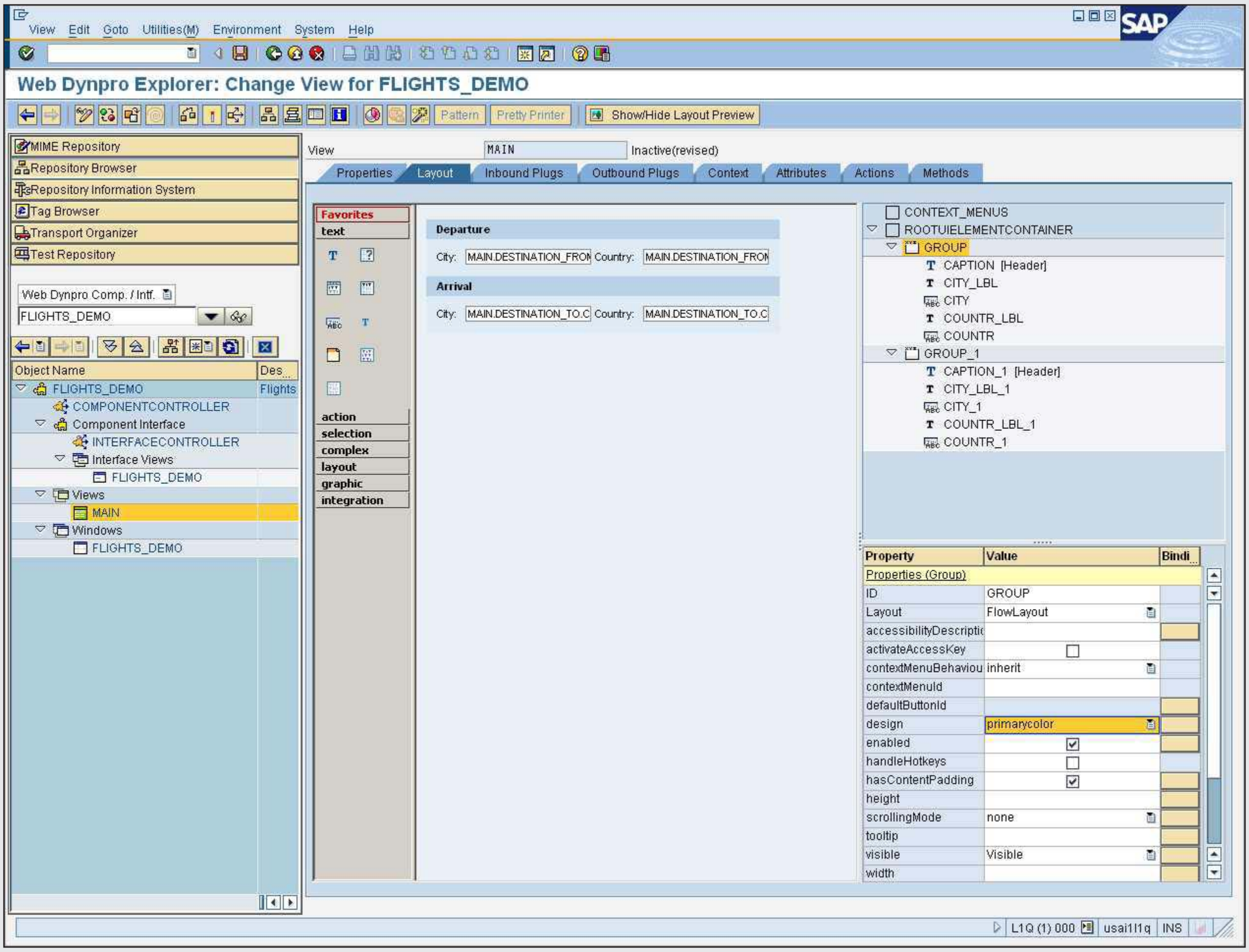


Create Context Binding for Table "ROOTUIELEMENTCONTAINER"

Context Node	MAIN.DESTINATION_FROM
Standard Cell Editor	InputField
Standard Property	value

Context Attribute	Cell Editor of Table Column	Binding	Name of Property to Be Bound
AIRPORTID	InputField	☐	value
CITY	InputField	☑	value
COUNTR	InputField	☑	value
COUNTR_ISO	InputField	☐	value

Figure 20 Binding context nodes to UI elements



Web Dynpro Explorer: Change View for FLIGHTS_DEMO

View MAIN Inactive(revised)

Departure

City: MAIN.DESTINATION_FROM Country: MAIN.DESTINATION_FROM

Arrival

City: MAIN.DESTINATION_TO.C Country: MAIN.DESTINATION_TO.C

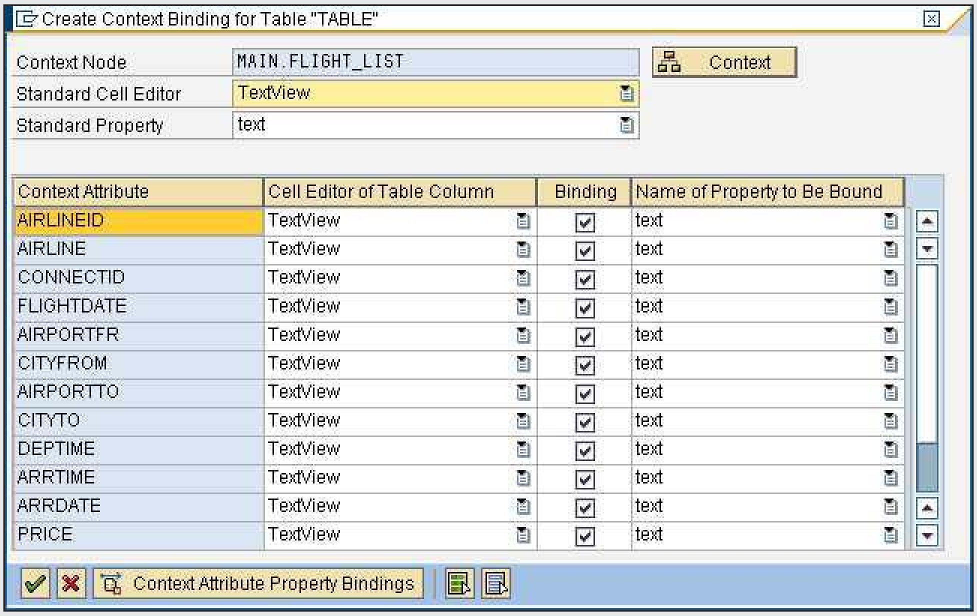
Property	Value	Bindi...
Properties (Group)		
ID	GROUP	
Layout	FlowLayout	
accessibilityDescriptic		
activateAccessKey	☐	
contextMenuBehaviou	inherit	
contextMenuId		
defaultButtonId		
design	primarycolor	
enabled	☑	
handleHotkeys	☐	
hasContentPadding	☑	
height		
scrollingMode	none	
tooltip		
visible	Visible	
width		

Figure 21 UI element groups shown in Web Dynpro View Designer

Context button lets you pick arbitrary context fields from the view context (**Figure 14**). The table is filled when you press the Context button and pick a Context Node.

I chose the context node DESTINATION_FROM, as shown in **Figure 20**. For each of the node's attributes, such as AIRPORTID, CITY, COUNTR, and COUNTR_ISO, the system suggests a standard input field as the default UI element. You could choose other UI element types (check buttons, radio buttons, Web links, etc.) using the drop-down icon, but the InputField UI element is exactly what you need here. The default property of an input field is its "value" property, meaning the default is fine here too. I checked binding for only the CITY and COUNTRY fields because they are the only fields you need to properly call the BAPI.

5. Repeat Step 4 for the search fields for the Arrival group and bind the fields as you did for GROUP. **Figure 21** shows the layout of the application to this point.


Create Context Binding for Table "TABLE"
Context
Context Attribute Property Bindings

Context Node: MAIN.FLIGHT_LIST
Standard Cell Editor: TextView
Standard Property: text

Context Attribute	Cell Editor of Table Column	Binding	Name of Property to Be Bound
AIRLINEID	TextView	☑	text
AIRLINE	TextView	☑	text
CONNECTID	TextView	☑	text
FLIGHTDATE	TextView	☑	text
AIRPORTFR	TextView	☑	text
CITYFROM	TextView	☑	text
AIRPORTTO	TextView	☑	text
CITYTO	TextView	☑	text
DEPTIME	TextView	☑	text
ARRTIME	TextView	☑	text
ARRDATE	TextView	☑	text
PRICE	TextView	☑	text

Figure 22 Binding the context node FLIGHT_LIST to the UI element table

6. Next, you insert a table element to display the search results. Create the Table element in the same way that you created the GROUP element, keeping the default name of TABLE.

7. Then, you bind the table as you did the search fields. **Figure 22** shows the development of the application to this point.

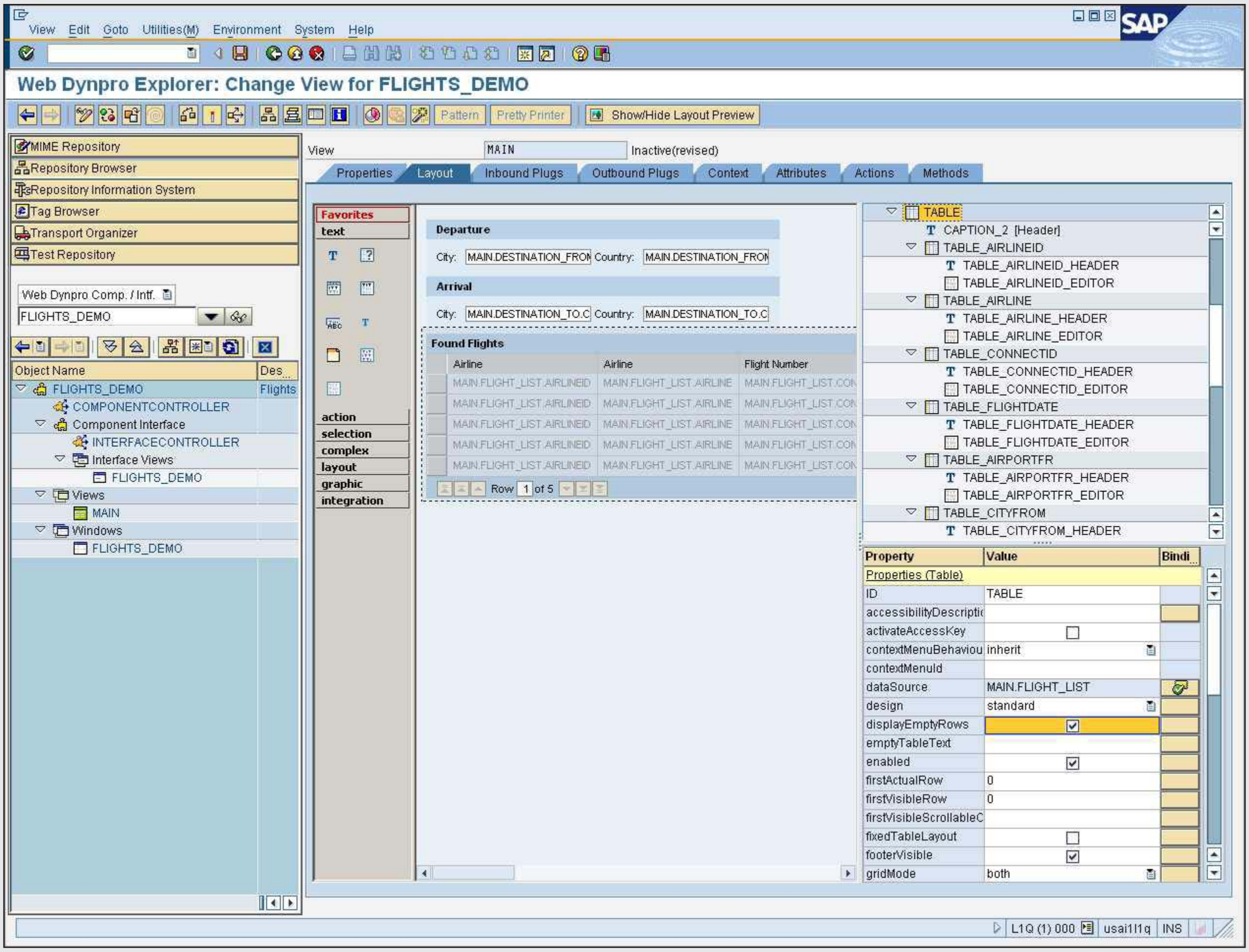
Since tables are typically used to display data, the default UI element type here is TextView, rather than InputField, and the default property that the system assigns is text rather than value.

The result of the table binding is shown in **Figure 23**. The view designer shows the table in the preview area. Each cell displays the name of the bound context variable. The structure of the table (columns, column headers) is visible in the control tree.

8. Now you need to add the button element, which the user can click on to retrieve flights based on the search fields (CITY and COUNTRY), and add some event-handling logic in native ABAP code to trigger the BAPI call. You simply add the button to the control tree by invoking the context menu INSERT → ELEMENT right after the search fields. Choose Button instead of Group (**Figure 16**).

9. Create the action GET_FLIGHTS from the button's Property sheet, in the Events section using the create action button (see **Figure 24** on page 22).

Once you click on the create action button, a pop-up window appears, as shown in **Figure 25** on page 22. Type the name of the action "GET_FLIGHTS" and then press the Enter key. This



Web Dynpro Explorer: Change View for FLIGHTS_DEMO

View MAIN Inactive(revised)

Departure
City: MAIN.DESTINATION_FROM Country: MAIN.DESTINATION_FROM

Arrival
City: MAIN.DESTINATION_TO.C Country: MAIN.DESTINATION_TO.C

Found Flights

Airline	Airline	Flight Number
MAIN.FLIGHT_LIST.AIRLINEID	MAIN.FLIGHT_LIST.AIRLINE	MAIN.FLIGHT_LIST.CON
MAIN.FLIGHT_LIST.AIRLINEID	MAIN.FLIGHT_LIST.AIRLINE	MAIN.FLIGHT_LIST.CON
MAIN.FLIGHT_LIST.AIRLINEID	MAIN.FLIGHT_LIST.AIRLINE	MAIN.FLIGHT_LIST.CON
MAIN.FLIGHT_LIST.AIRLINEID	MAIN.FLIGHT_LIST.AIRLINE	MAIN.FLIGHT_LIST.CON
MAIN.FLIGHT_LIST.AIRLINEID	MAIN.FLIGHT_LIST.AIRLINE	MAIN.FLIGHT_LIST.CON

Row 1 of 5

Property	Value	Bindi...
Properties (Table)		
ID	TABLE	
accessibilityDescriptic		
activateAccessKey	☐	
contextMenuBehaviou	inherit	
contextMenuId		
dataSource	MAIN.FLIGHT_LIST	✓
design	standard	
displayEmptyRows	☑	
emptyTableText		
enabled	☑	
firstActualRow	0	
firstVisibleRow	0	
firstVisibleScrollableC		
fixedTableLayout	☐	
footerVisible	☑	
gridMode	both	

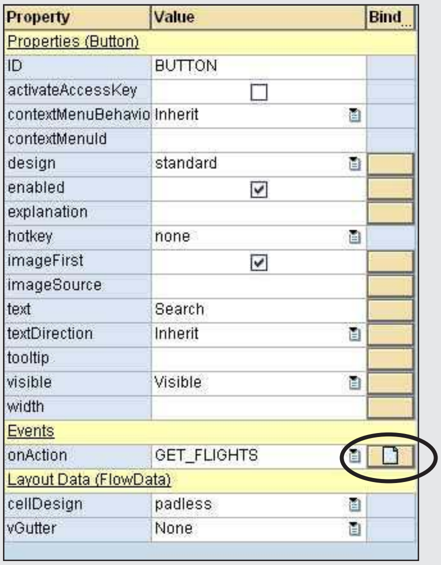
Figure 23 Web Dynpro table with context binding

opens the code editor where you can place your event-handling code. To create some of the code automatically, use the Web Dynpro Code Wizard.

10. Choose the Web Dynpro Code Wizard from the push-button bar. A pop-up window appears with various options for generating code, as shown in **Figure 26** on page 23.

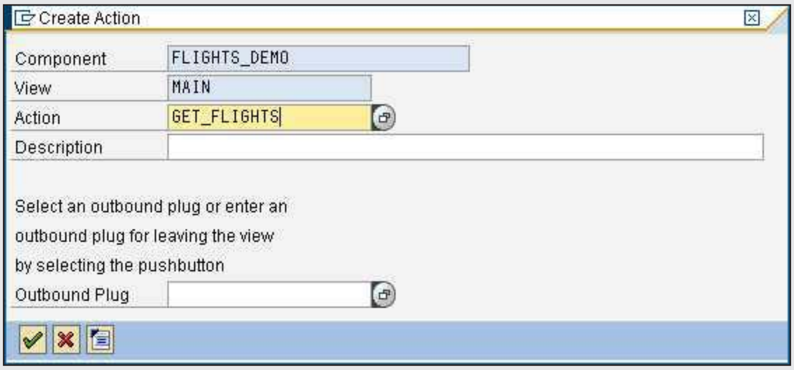
You want to issue a method call to the BAPI, so select the option Method Call in Used Controller. The system suggests the BAPI call created when you launched the Service Call Wizard. **Figure 27** on page 24 shows the automatically generated code.

You've created a complete Web Dynpro component containing search field groups, a result table, a button to trigger the search logic, and the underlying Web Dynpro context definition so that you're ready to call the application in the browser. The only thing missing is a Web Dynpro applica-



Property	Value	Bind
Properties (Button)		
ID	BUTTON	
activateAccessKey	□	
contextMenuBehavio	Inherit	
contextMenuId		
design	standard	
enabled	☑	
explanation		
hotkey	none	
imageFirst	☑	
imageSource		
text	Search	
textDirection	Inherit	
tooltip		
visible	Visible	
width		
Events		
onAction	GET_FLIGHTS	
Layout Data (FlowData)		
cellDesign	padless	
vGutter	None	

Figure 24 Property sheet for UI element button



Create Action

Component	FLIGHTS_DEMO
View	MAIN
Action	GET_FLIGHTS
Description	

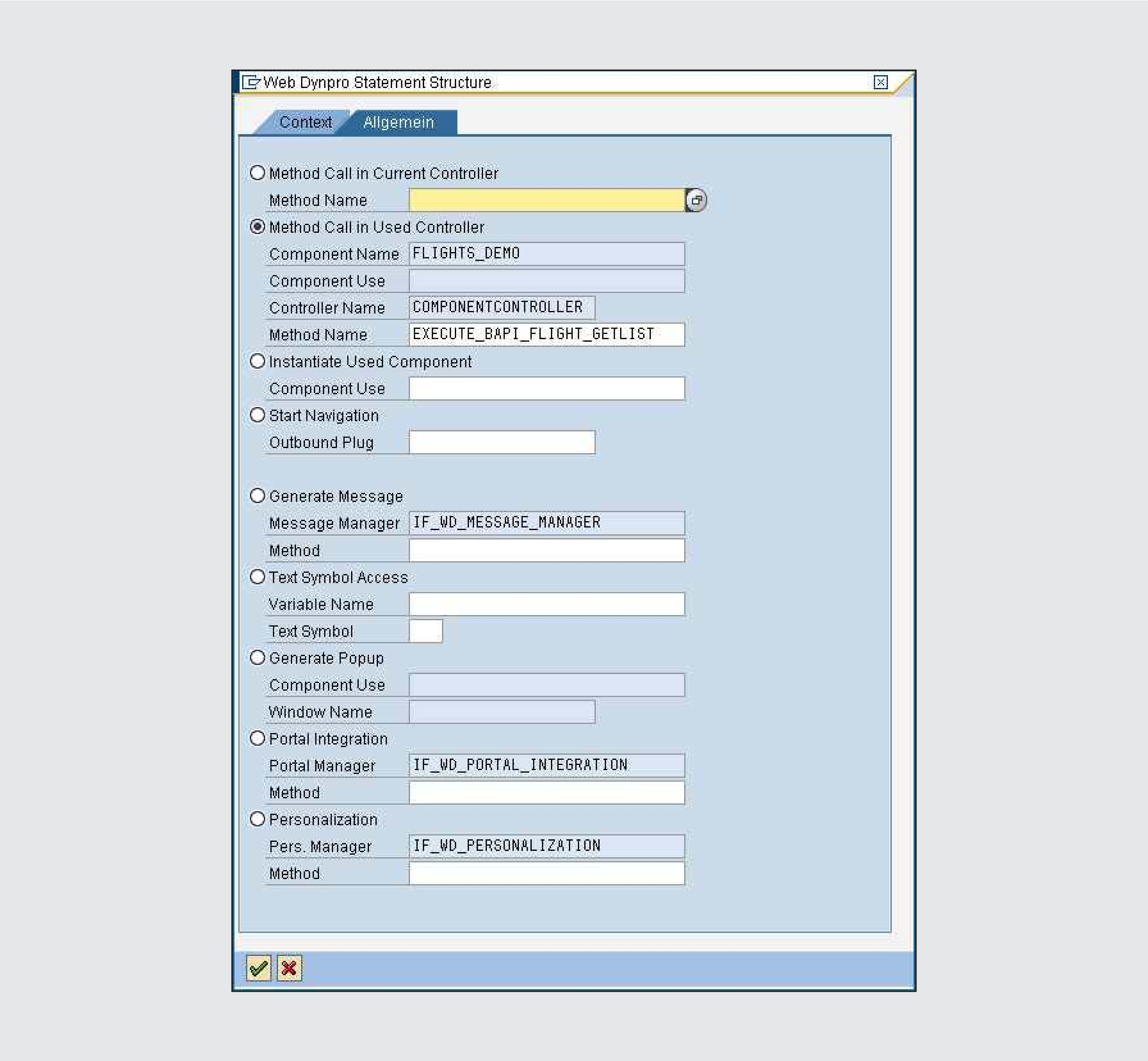
Select an outbound plug or enter an
outbound plug for leaving the view
by selecting the pushbutton

Outbound Plug

Figure 25 Creating the action for the button

tion (basically, an application name from which a URL is composed to launch the application).

From the outline of your Web Dynpro component (which is always visible in the navigation area on the left of the screen), you create a Web Dynpro application with the name flights_demo, as shown in **Figure 28** on page 24. Next, you activate this Web Dynpro application from the context menu, and then you launch it from that context menu as well. You should see results that are similar to those shown in **Figure 2**.



○ Method Call in Current Controller

Method Name

● Method Call in Used Controller

Component Name: FLIGHTS_DEMO

Component Use

Controller Name: COMPONENTCONTROLLER

Method Name: EXECUTE_BAPI_FLIGHT_GETLIST

○ Instantiate Used Component

Component Use

○ Start Navigation

Outbound Plug

○ Generate Message

Message Manager: IF_WD_MESSAGE_MANAGER

Method

○ Text Symbol Access

Variable Name

Text Symbol

○ Generate Popup

Component Use

Window Name

○ Portal Integration

Portal Manager: IF_WD_PORTAL_INTEGRATION

Method

○ Personalization

Pers. Manager: IF_WD_PERSONALIZATION

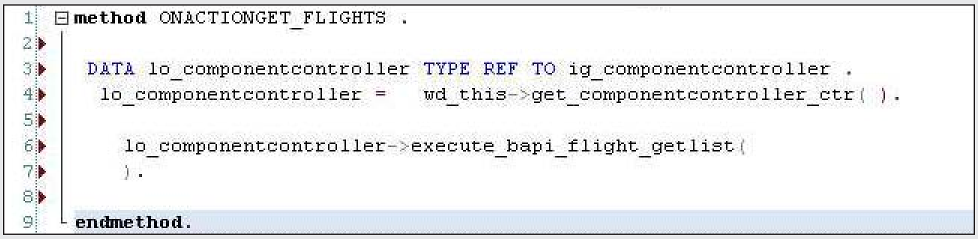
Method

Figure 26 Various options of the Web Dynpro Code Wizard

So far, you have developed a fully functional custom Web Dynpro application using the tools of the ABAP Workbench. In Part 2 of this article, you assume that such a custom application was delivered to you, but you want to enhance and extend the application with further capabilities.

Conclusion

Web Dynpro is a powerful way to create and enhance applications designed for the Web. Due to the importance of the Enhancement Packages strategy for SAP NetWeaver ERP 6.0, Web Dynpro plays a vital role



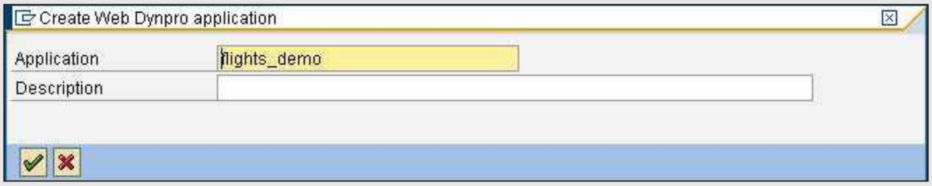
```
method ONACTIONGET_FLIGHTS .

  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .
   lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).

     lo_componentcontroller->execute_bapi_flight_getlist(
     ).

endmethod.
```

Figure 27 Automatically generated code for the Web Dynpro application



Create Web Dynpro application

Application	flights_demo
Description	

Figure 28 Web Dynpro flights_demo application

both in developing custom applications and in enhancing prepackaged applications that SAP delivers as part of their Business Suite solutions. This article has covered how to develop new Web Dynpro ABAP applications. The next article, Part 2, will discuss how to enhance existing Web Dynpro ABAP applications. The enhancement mode present in all Web Dynpro development tools properly separates your extensions from their original applications, thereby preserving your investment in your SAP solutions.