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

Part 2 — Enhancing existing 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 at karl.kessler@sap.com.

This is the second installment of my two-article series on the development of flexible Web Dynpro ABAP applications and the enhancement of existing ones using the latest techniques from SAP. In the first installment, published in the September/October 2008 issue of *SAP Professional Journal*, I explained the SAP Enhancement Package for SAP ERP 6.0 strategy, which helps customers choose the optimal roadmap for innovation in their system landscape and processes. The focus of that article was on deriving a Web Dynpro user interface (UI) and corresponding application from the signature (input and output parameters) of a BAPI call, effectively using all Web Dynpro development tools.

With a solid understanding of SAP's enhancement strategy and the development of UIs based on Web Dynpro, you are now ready to begin to enhance a given UI in Web Dynpro using the application you developed in the first article. In this installment, you will learn how to enhance a Web Dynpro application, which begins with designing the enhancement so that the changes are logical, accurate, and easy to use. Next you will learn how to implement the enhancement. You will also learn how to create a custom view that will allow you to display additional drill-down data for a selected flight. And finally, you will learn about programming techniques for Web Dynpro *context mapping*.

Enhancing the newly created Web Dynpro application

The most important recommendation when enhancing an application — be it Web Dynpro or a classic UI — is to keep the extensions separate from the original code as much as possible and to add extensions only at a few well-defined places. The reason is quite obvious: Although enhancing

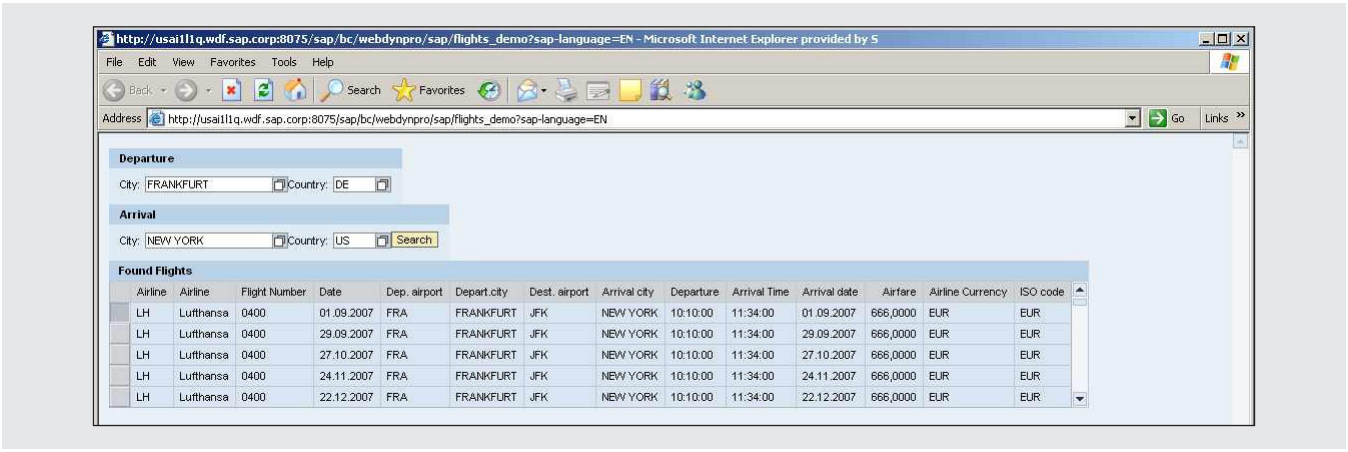

Figure 1 Web Dynpro flight application example

an application is much better in terms of maintenance when compared to traditional modifications, the fact remains that the original code might change over time in subsequent releases or support packages. If you have intertwined your custom code with SAP's original code, you will get long cycles of conflict resolution. Try to keep your code on your side and maintain a few *enhancement points*. Because you rely on implicit enhancement points, SAP is not aware of your extensions. Maintaining a few enhancement points is simple, and adjusting them is easy, if necessary. If you follow this discipline, you can manage your enhancement code much better. Needless to say, the industry solutions from SAP follow a similar approach when they enhance the generic core application to implement a particular industry flavor.

In the first article, you created the Web Dynpro flight application, as shown in **Figure 1**. At the top of the screen, there are search fields for specifying the departure and arrival destinations. Each field is implemented as a city/country pair with value help (that is, the drop-down button attached to the input field so the system displays a list of values from which to choose). The table contains all of the "found" flights — that is, the flights that match the search criteria when the user clicks the Search button.

Before you begin to make changes to your application, you need to understand what changes need to be made and why, and then plan how to make those changes in a logical and uniform way. You need to consider various aspects of this process, such as a clear navigation to additional custom views. You need to carefully decide where to include custom logic by adding a button that triggers the custom action. You need to keep most of the original application untouched so that users can quickly understand and learn your extension if they are already familiar with the standard functionality. You also need to look at the layout aspects of your extensions to better manage the user's overall expectation and experience. You'll develop the enhancement in the following example to learn how this works in practice.

Designing the enhanced application

You want your Web Dynpro application to be able to display bookings for a selected flight when the user clicks a button that retrieves the correct data. Rather than creating a new application, you can enhance the existing one. You will begin by enhancing the original view (MAIN) in the application by adding a button with the name "Get Bookings," which will retrieve booking information from the table SBOOK for the selected flight. You also need to create another view that displays the booking information in a table format. When the user selects a flight in the MAIN view and clicks the Get Bookings button, a second view will display the booking information.

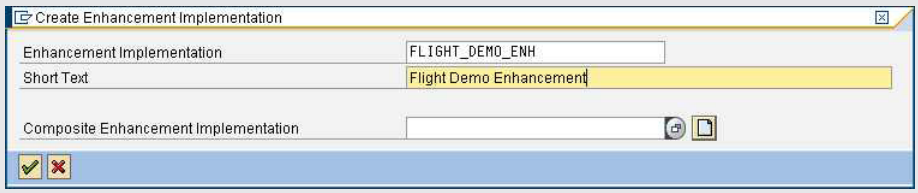

Figure 2 The view editor in enhancement mode

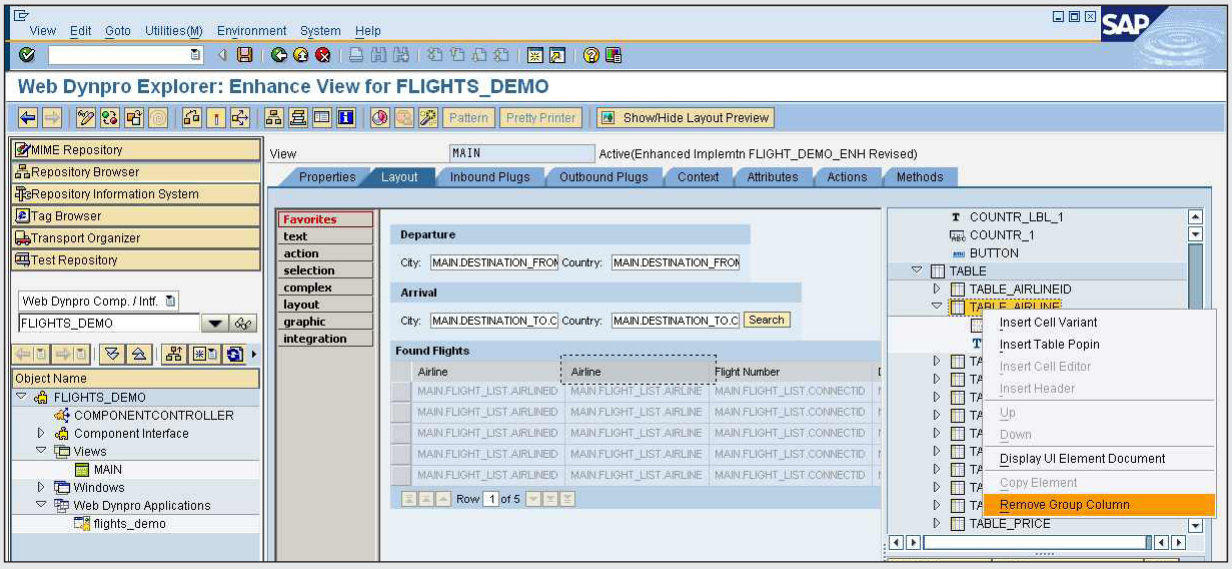

Figure 3 Marking a column for deletion

Implementing an enhancement

To enhance an application, you need to work in *enhancement mode*. The enhancement mode keeps your extensions separate from the original application. In contrast, if you change an application (basically a modification if you change an SAP object), you overwrite the object.

You enter the view editor by clicking the enhancement mode icon, which switches the view editor into enhancement mode from the change mode. As shown in **Figure 2**, the Create Enhancement Implementation dialog signals that you are in the enhancement mode. Here, you enter the enhancement implementation name (FLIGHT_DEMO_ENH, in this case) that will act as a container to collect all of your *enhancement artifacts*. You can later aggregate them to a composite enhancement implementation if you want.

The enhancement mode presents most elements in a protected fashion, but allows you to add custom elements. There is also an option for hiding certain aspects of the original application. For example, if you want to hide the table column that contains the airline names (TABLE_AIRLINE), but keep the column with the airline IDs (TABLE_ARILINEID) in the table, right-click TABLE AIRLINE and select Remove Group Column from the context menu, as shown in **Figure 3**.

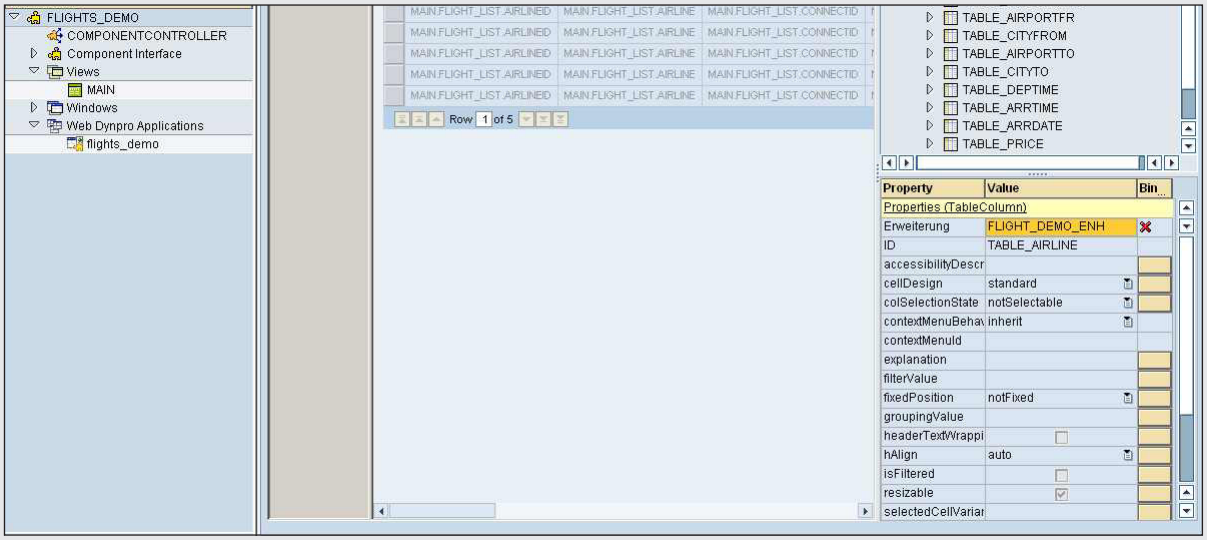

Figure 4 Property sheet of a removed table column

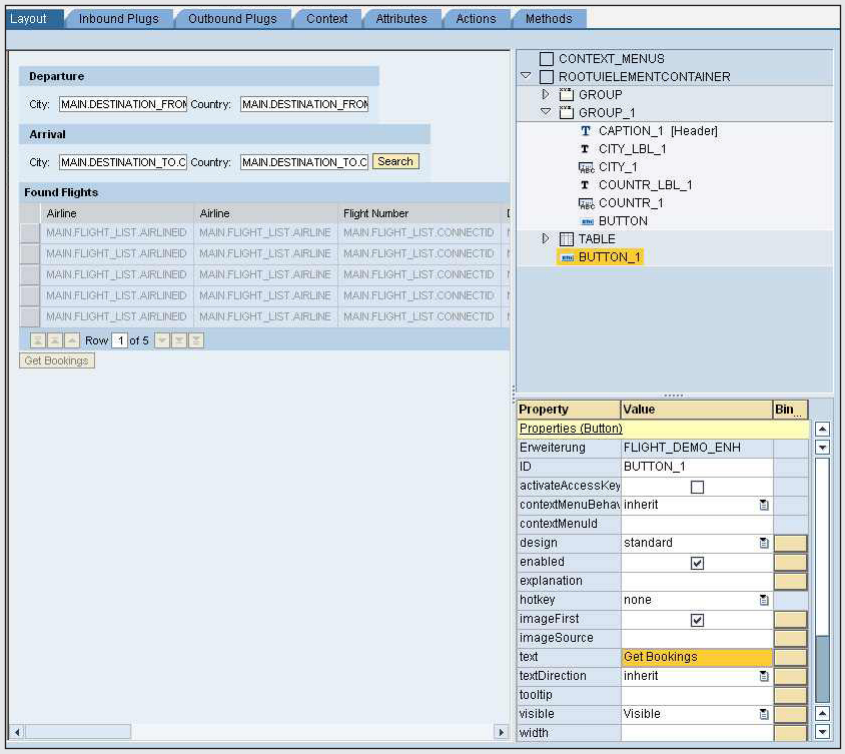

Figure 5 Adding a button to the application while in enhancement mode

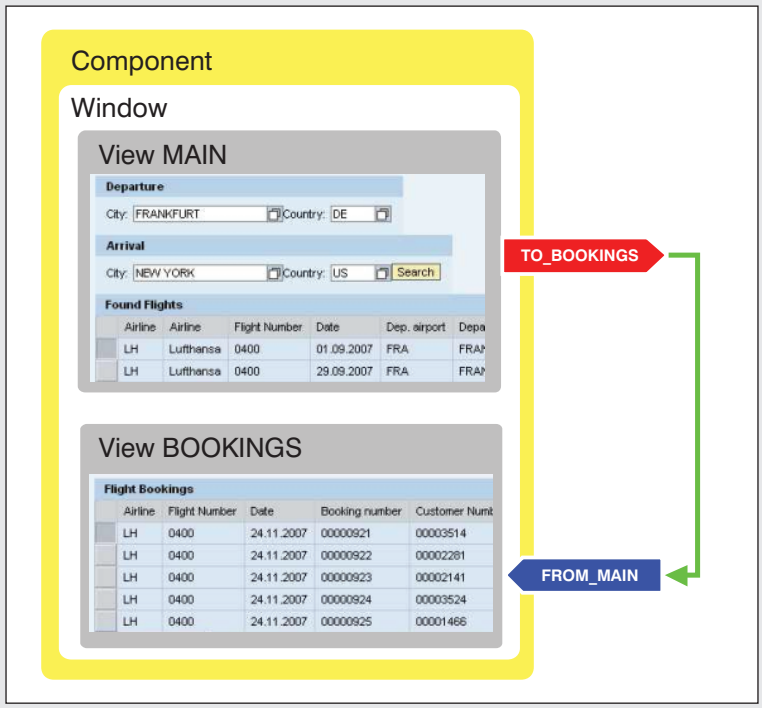

Figure 6 Blueprint of the enhancement for retrieving booking for a selected flight

In the *property window* shown on the right in **Figure 4**, there is an X, which indicates that the column AIRLINE will not be displayed during runtime (it still appears in design time, however, to provide an "anchor" for any further enhancements).

Now that you are in the enhancement mode, you can add the custom button with the label "Get Bookings" in the MAIN view. This button will be the *enhancement point* by which the application will be able to retrieve the bookings information for the selected flight and eventually display that data in a new view. Open the context menu from the MAIN view, select Insert → Element, and add a button to the original control tree under the TABLE node, as shown in **Figure 5**.

Figure 6 provides a blueprint of the enhancement. To be able to retrieve bookings for a selected flight, you need to create a custom view named BOOKINGS and connect the MAIN view with the BOOKINGS view. In Web Dynpro, navigation between views is facilitated by *plugs*. An outbound plug points to potential target views, whereas an inbound plug points to potential source views. If an outbound plug is connected to an inbound plug, a transition can occur between the views that are connected from source to target.

First, you need to extend the MAIN view with an outbound plug, which you'll label TO_BOOKINGS. The TO_BOOKINGS plug will link the MAIN view to the BOOKINGS view that you'll create shortly. Switch to the Outbound Plugs tab, and enter the name of the outbound plug (TO_BOOKINGS), as shown in **Figure 7** on the next page.

The Web Dynpro programming model allows you to associate the navigation plug from one view (displaying flights in the MAIN view) to the other (displaying bookings in the BOOKINGS view). Using the button's property sheet, you can do so by creating an action for the button that triggers the navigation. See **Figure 8** on the next page.

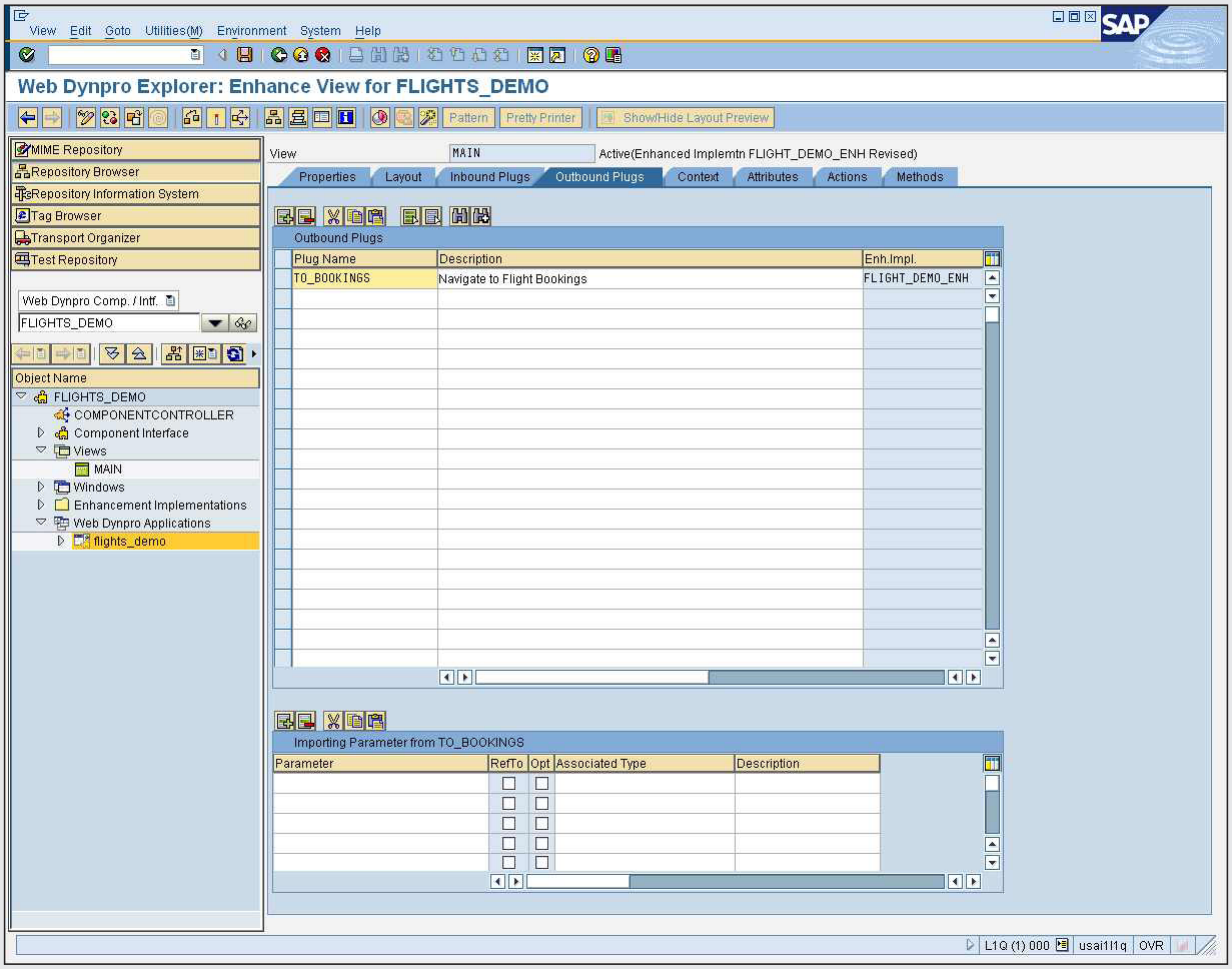

Figure 7 Extend the MAIN view with an outbound plug

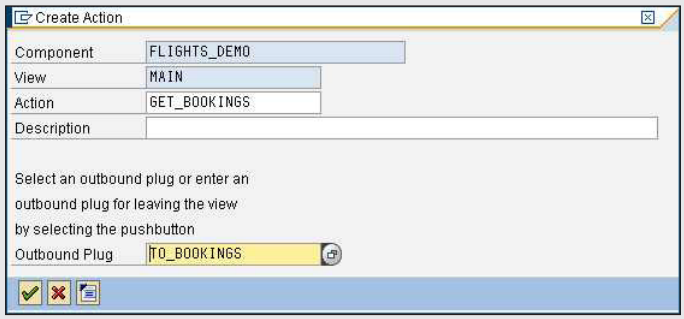

Figure 8 Creating a button that triggers the navigation to the TO_BOOKINGS outbound plug

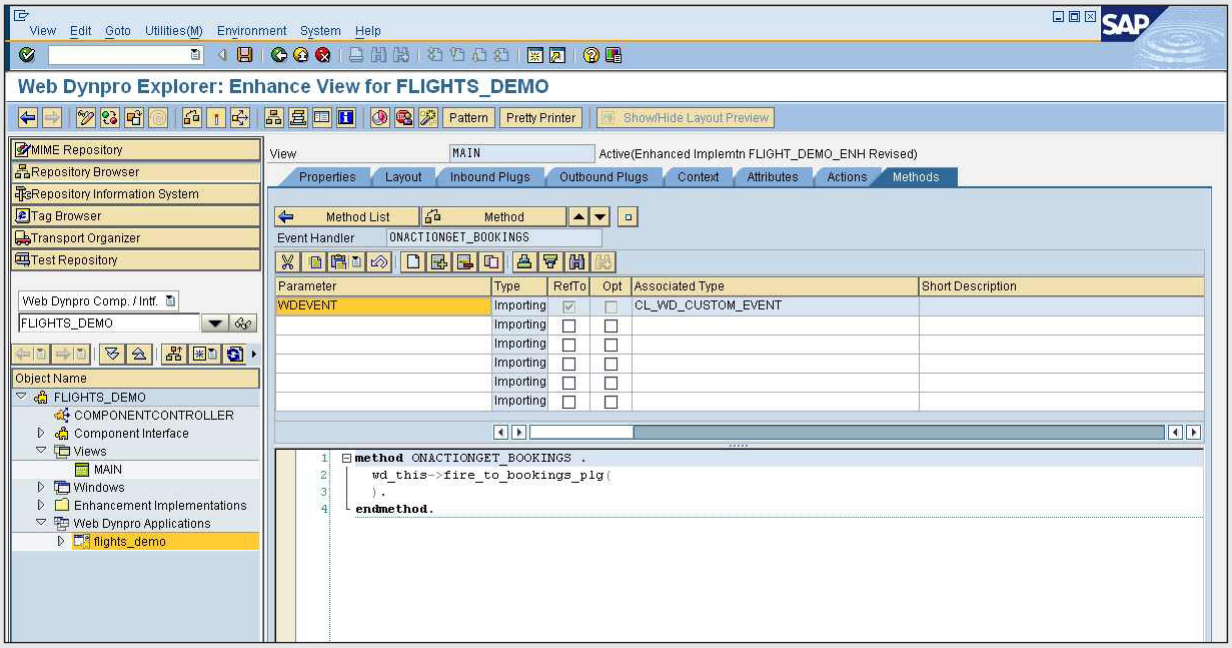

Figure 9 Code generated by the Web Dynpro Code wizard for the action GET_BOOKINGS

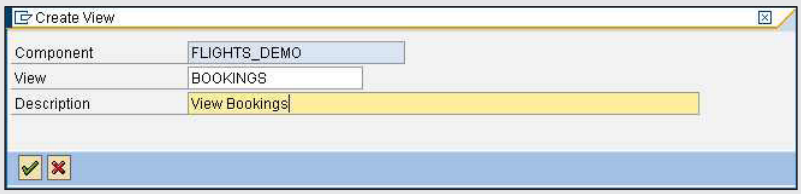

Figure 10 Creating the custom view for the enhancement

When you create the action, the Web Dynpro Code Wizard adds the required "fire plug" code automatically, as shown in **Figure 9**.

Adding the custom view

Now that you've added the enhancement point to the original application, you are ready to choose the enhancement[1] and create a custom view for the BOOKINGS as shown in **Figure 10**, with an inbound plug named "FROM_MAIN," which you will create next.

[1] In principle, the system can manage multiple enhancements in parallel. Therefore you must always select the enhancement when adding custom views to a given application.

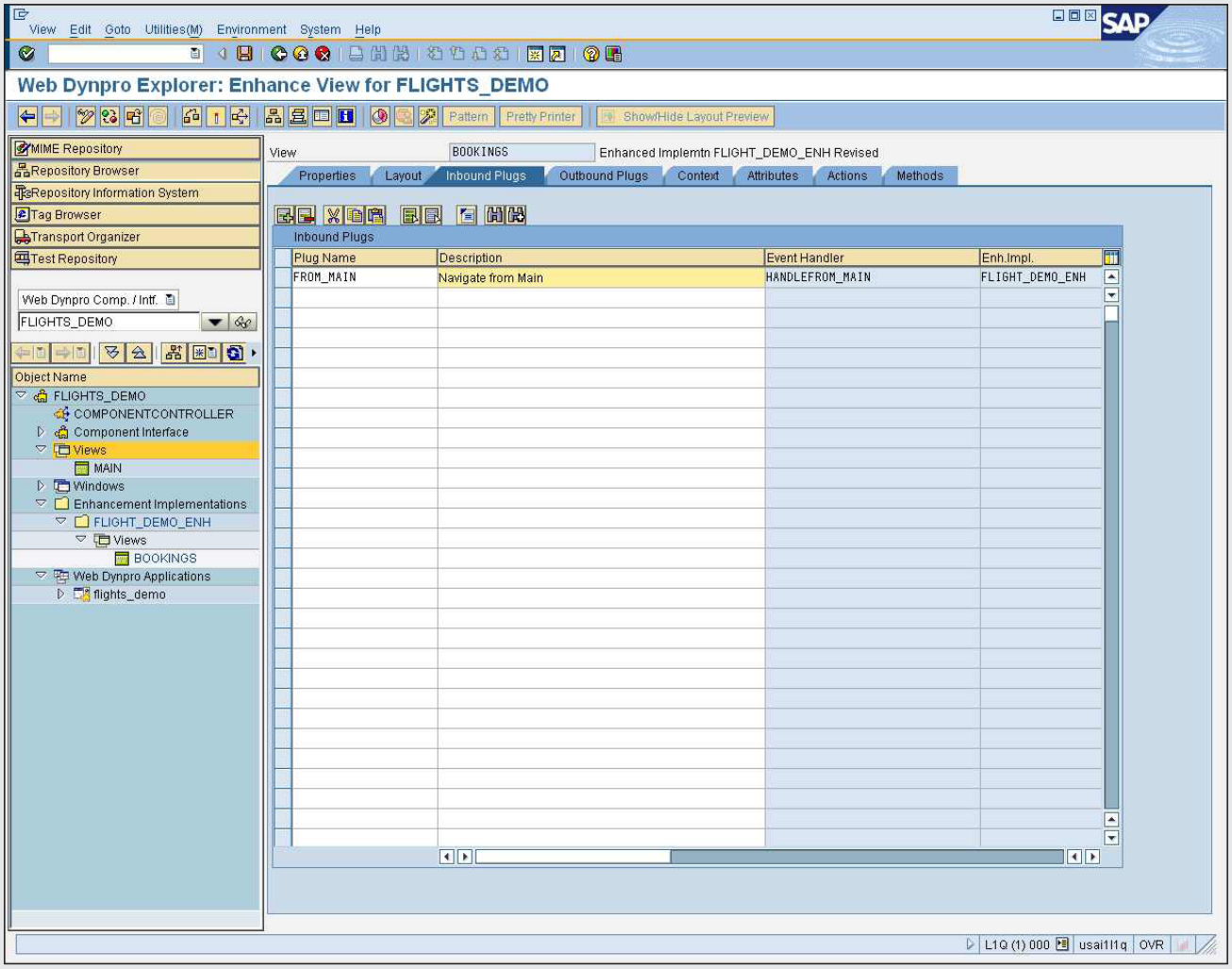

Figure 11 The BOOKINGS view with the FROM_MAIN inbound plug

Double-click the window FLIGHTS_DEMO in the outline, and drag the newly created view into the window hierarchy since the view BOOKINGS is not contained in the hierarchy by default. Next, navigate to the Inbound Plugs tab and enter the name of the plug (e.g., FROM_MAIN), as shown in **Figure 11**.

The BOOKINGS enhancement view is embedded into the FLIGHTS_DEMO window, which contains the original MAIN view, as shown in **Figure 12**. (The implicit containment of MAIN in the window FLIGHTS_DEMO is specified in **Figure 4**, but you must make that connection explicit when implementing your enhancements.)

To make the navigation happen, you need to link the TO_BOOKINGS outbound plug to the FROM_MAIN inbound plug by simply dragging the outbound plug and dropping it onto the inbound plug, which results in the screen shown in **Figure 13**. Click the execute button to create the

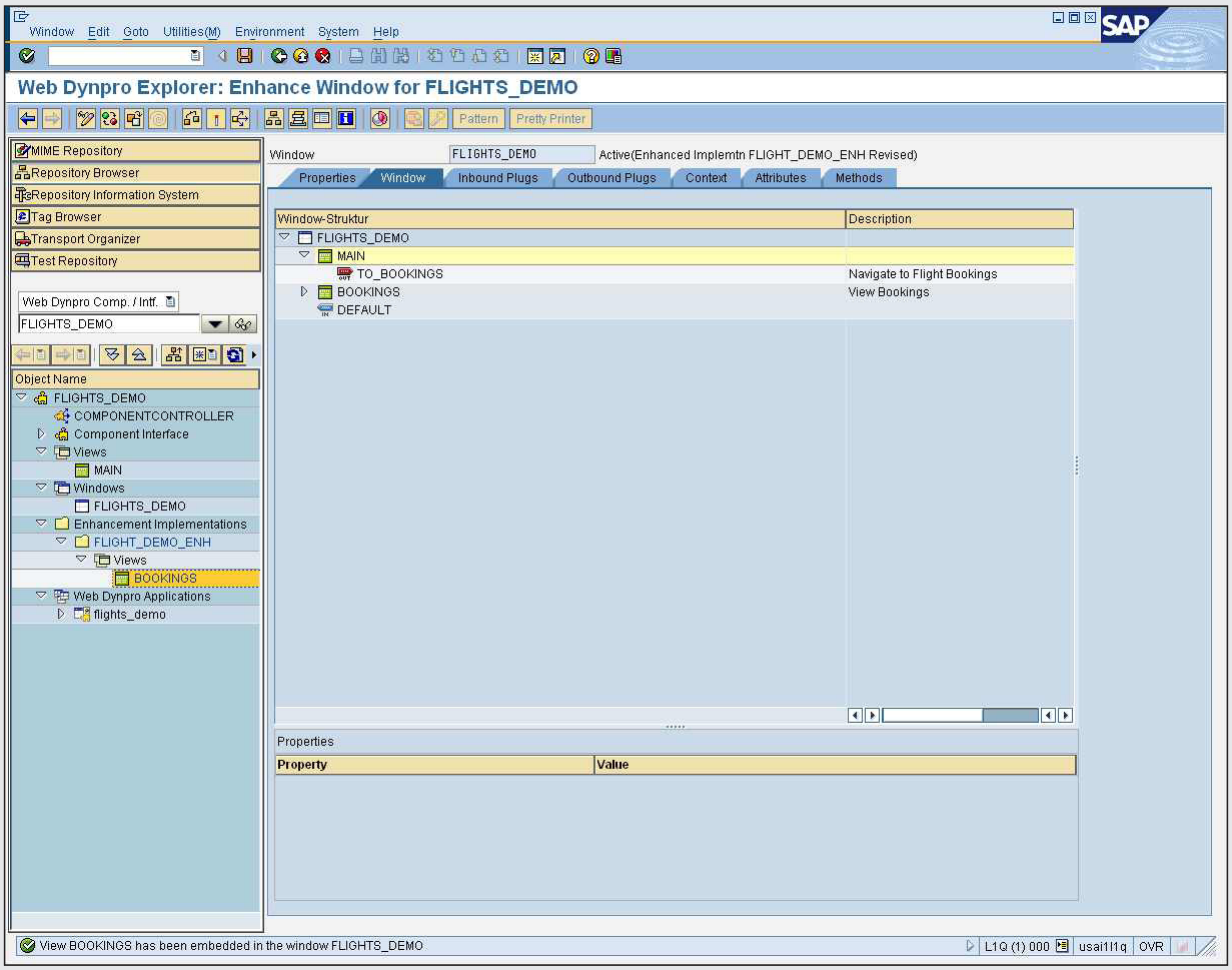

Figure 12 Hierarchy showing the MAIN and BOOKINGS views as part of the FLIGHTS_DEMO application

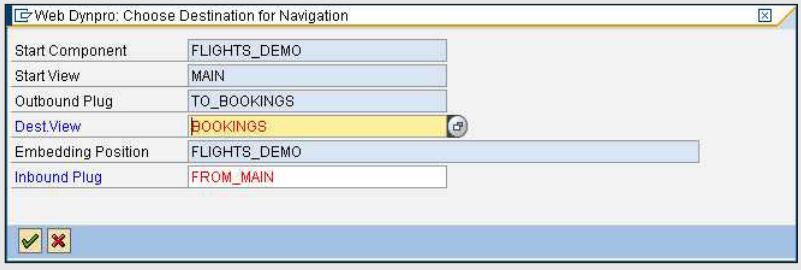

Figure 13 Creating a link between the outbound plug and the inbound plug

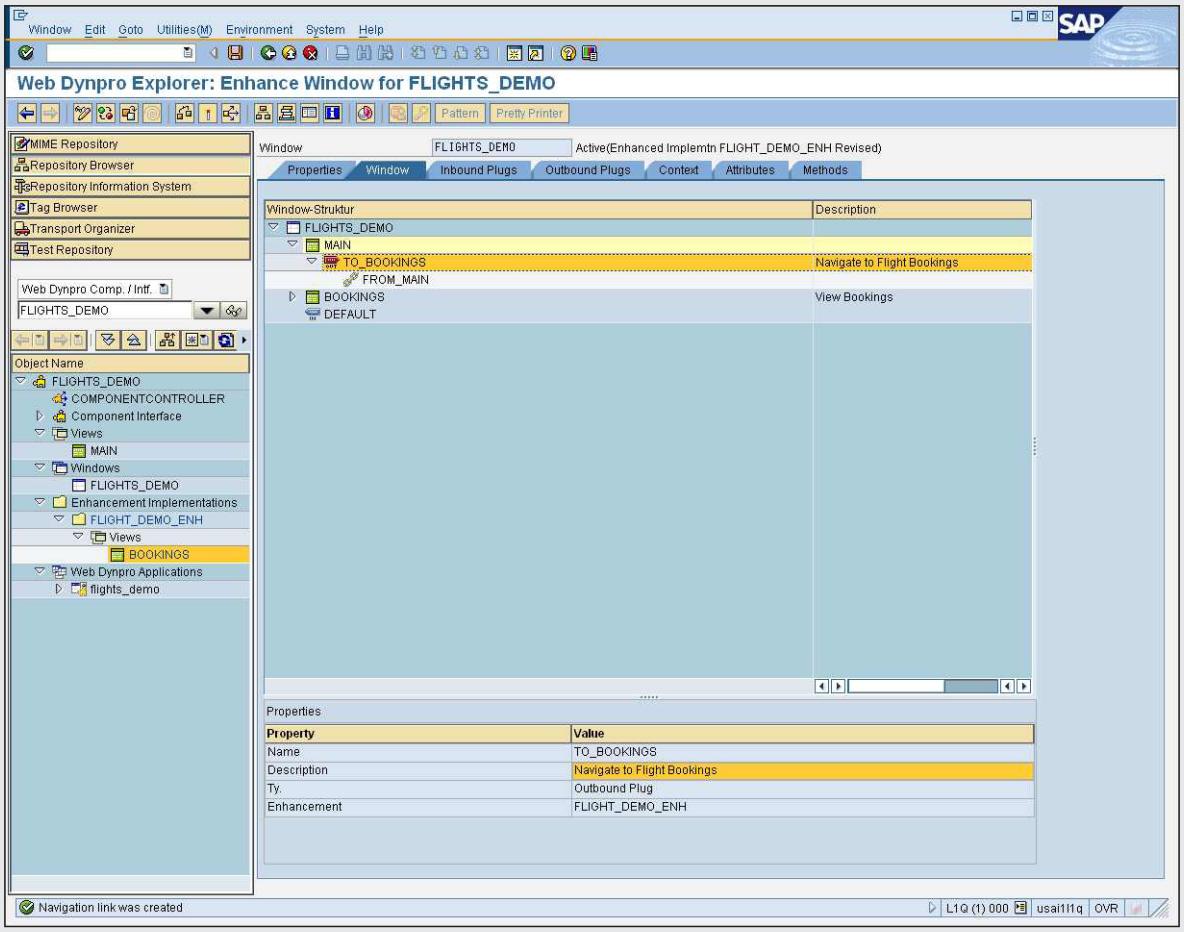

Figure 14 Navigation between the MAIN view and the BOOKINGS view can now occur

link. As shown in **Figure 14**, there is a link between the outbound plug TO_BOOKINGS and inbound plug FROM MAIN, indicated by a chain symbol, so navigation between the two views can now take place.

Now you need to identify the source of the data that can be retrieved using this enhancement.

Context mapping

You could activate the enhanced application at this point in time, but it would display an empty BOOKINGS view. This is because you have implemented the desired navigation, but you haven't defined the data flow. To feed the BOOKINGS view, you need to provide the proper *context mapping* from the component controller to the view controller. **Figure 15** shows the blueprint of the necessary data flow. You need to make the flight list available to the BOOKINGS view as well, and then identify the relevant bookings from the selected flight and display them accordingly in a table view.

Keep in mind is that there is no BAPI available for selecting booking information. Therefore, you need to provide your own custom code here. First, you need to create a custom context node (FLIGHT_LIST) to store the bookings based on the selected flight. Navigate to the Context tab, right-click CONTEXT and then select Create → Node (see **Figure 16**).

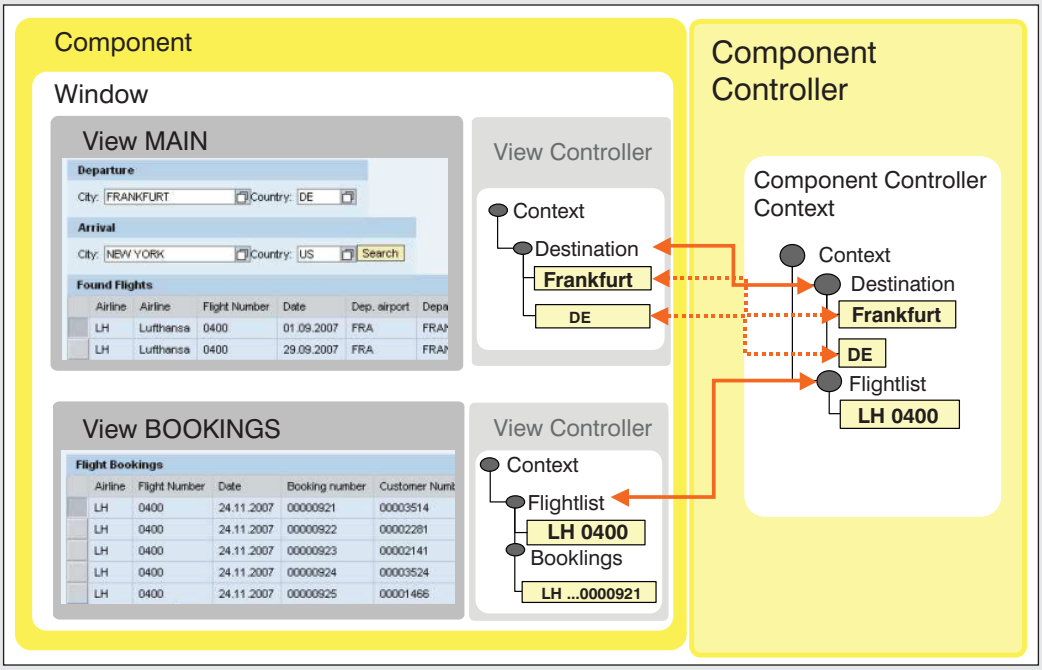

Figure 15 Blueprint extended to show the enhancements

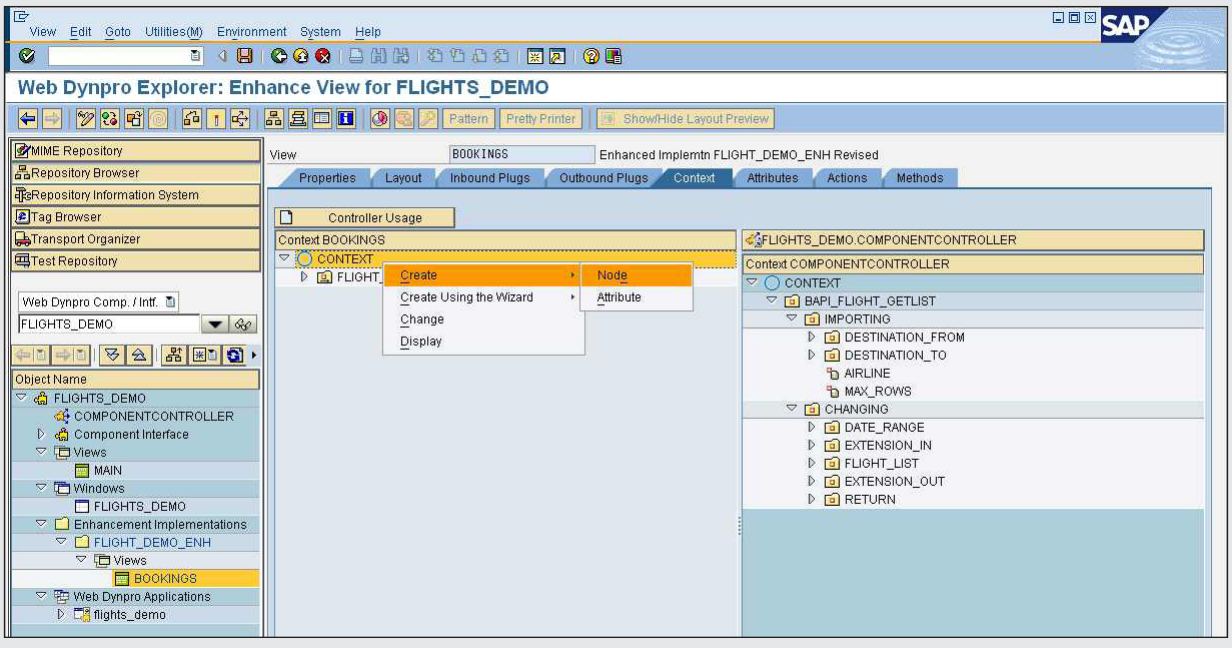

Figure 16 Creating a custom context node

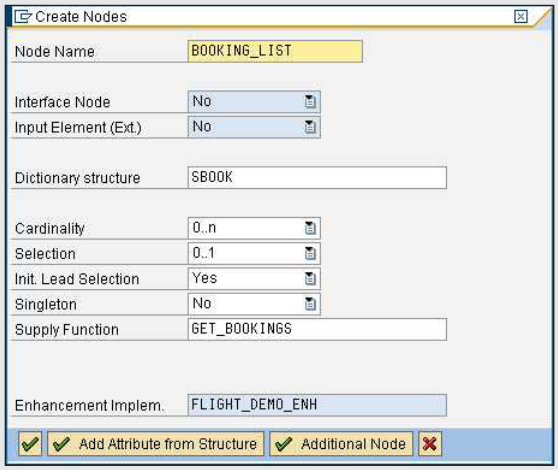

Figure 17 Defining the context node for displaying the flight bookings

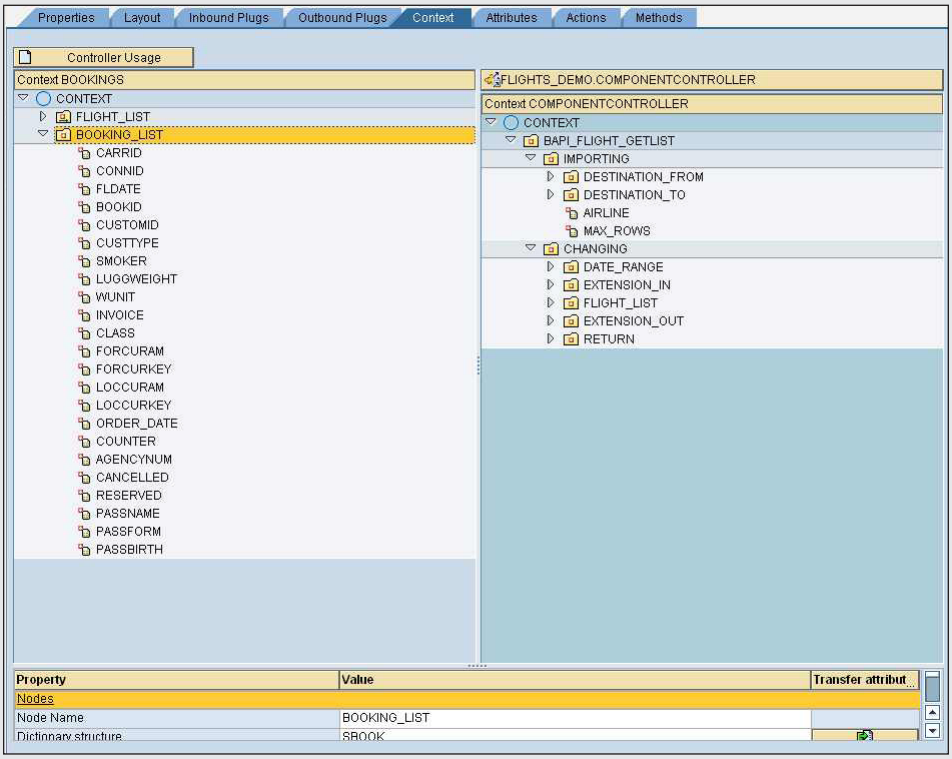

Figure 18 Copying all of the fields from the SBOOK table to the BOOKING_LIST custom context node

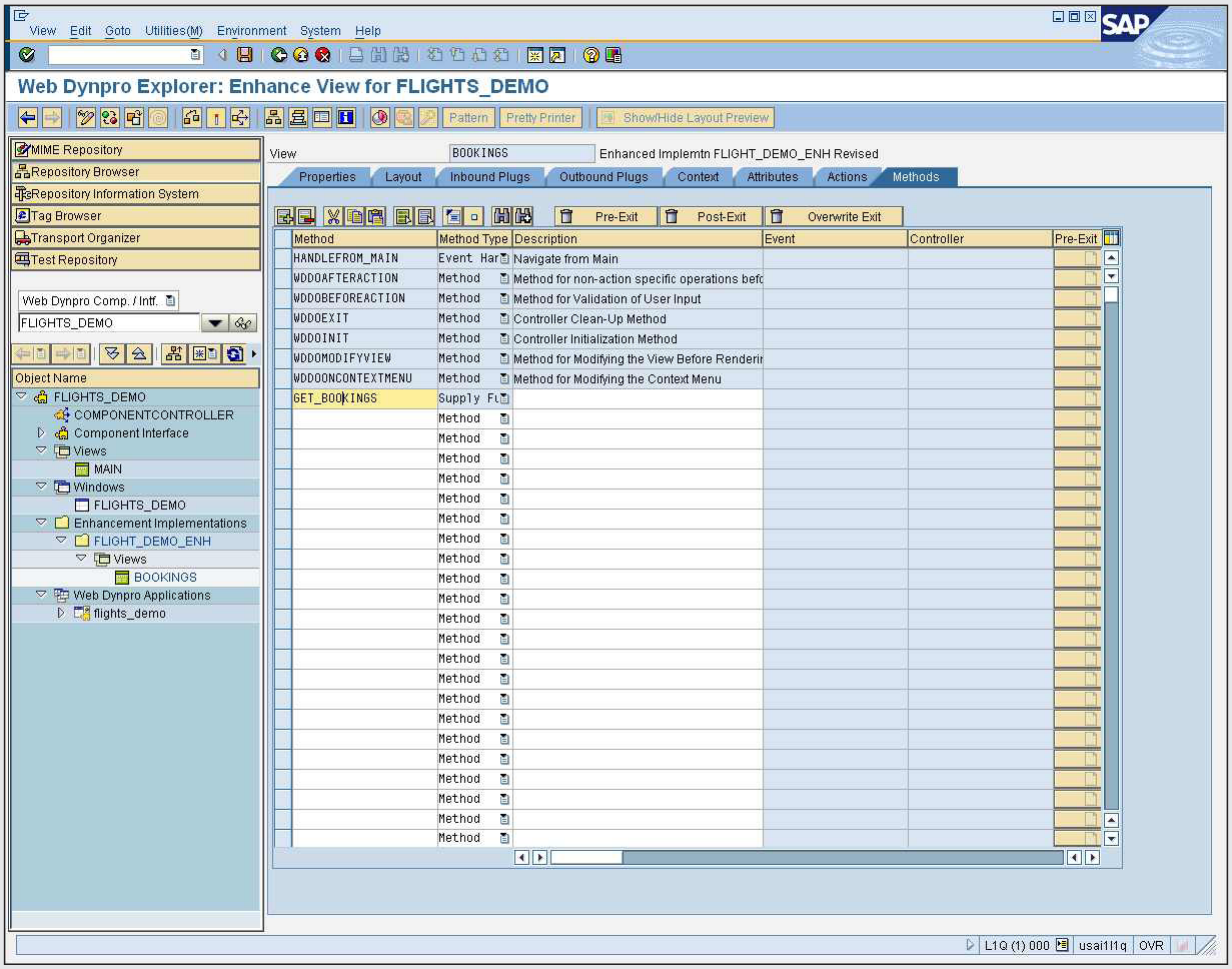

Figure 19 List of methods in the view controller

As shown in **Figure 17**, you need to define the node based on the SBOOK dictionary structure. The table SBOOK is an essential part of the IDES flight data model. You also need to specify the *supply function* named GET_BOOKINGS, which will always be called when the node's data need to be presented.

Next, select all fields of the SBOOK table and copy them to the context, as shown in **Figure 18**.

Next, navigate to the Methods tab to implement the supply function. **Figure 19** shows the list of methods in the view controller.

Figure 20 on the next page shows the code for the skeleton implementation of the supply function that was automatically generated when you defined the BOOKING_LIST node.

The implementation of the supply function reads the selected flight in the helper variables (method GET_STATIC_ATTRIBUTES), feeds these variables into a corresponding SELECT * FROM SBOOK statement, and then binds the internal table containing the resulting flight bookings to the Web Dynpro context. You can imagine the convenience of the creation of a service call demonstrated in the first

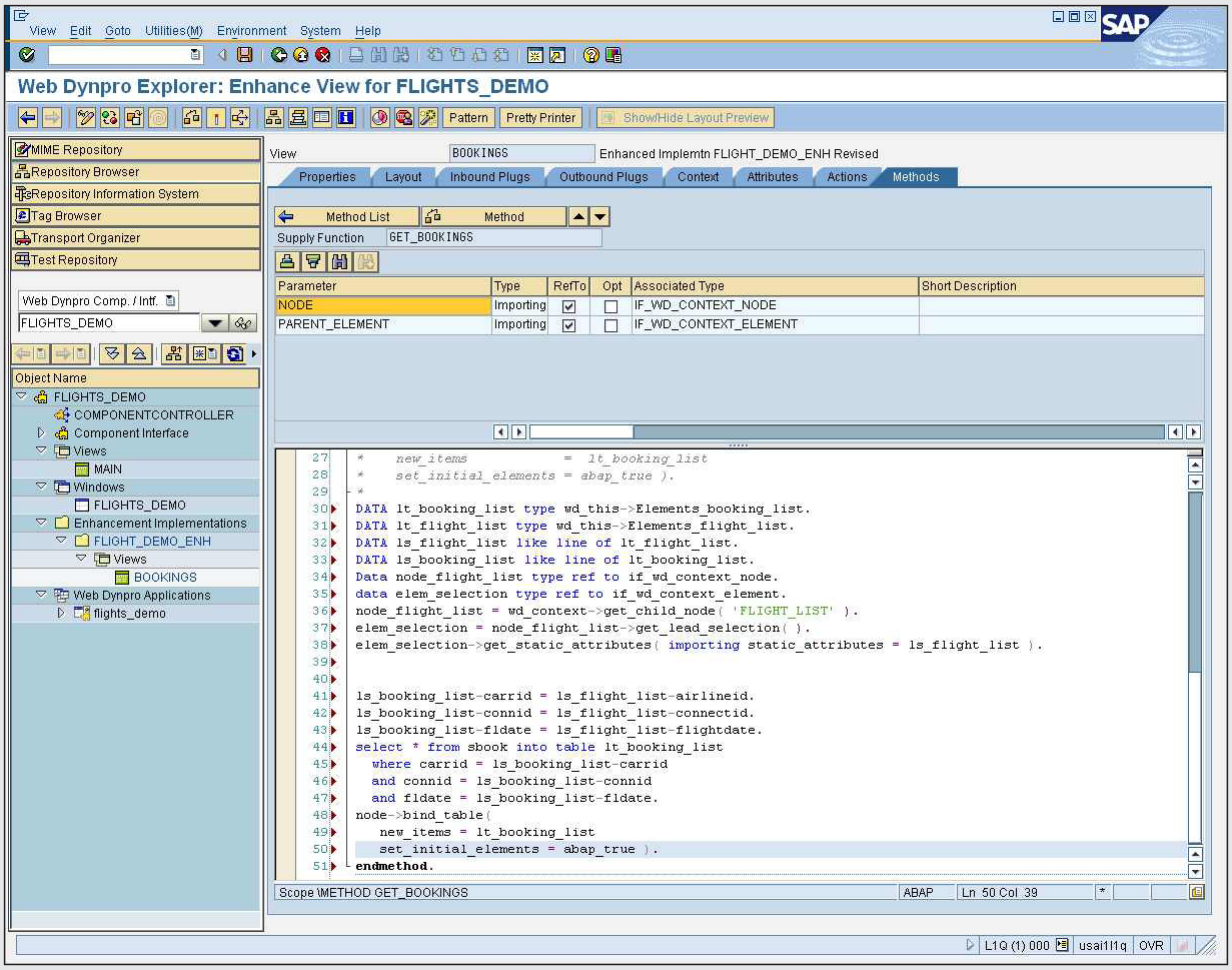

Figure 20 Code generated for implementing the GET_BOOKINGS supply function

part of this article, but you would need a BAPI for that. Because there is no BAPI to retrieve bookings, you need to create a UI table that is bound to the data source of the BOOKING table exactly the same way you implemented the FLIGHT_LIST for the MAIN view. See **Figure 21**.

When you activate and launch your enhancement, it should look similar to **Figure 22**.

By means of the enhancement, you have now extended the original application to include a linked custom view. You have reused context variables of the component controller, but the original application remained almost "untouched" except for the button that triggers the display of the custom view.

Conclusion

Web Dynpro is a powerful way to create and enhance applications designed for the Web. Because of the importance of the SAP Enhancement Package for SAP ERP 6.0 strategy, Web Dynpro plays a vital role in both developing custom applications and

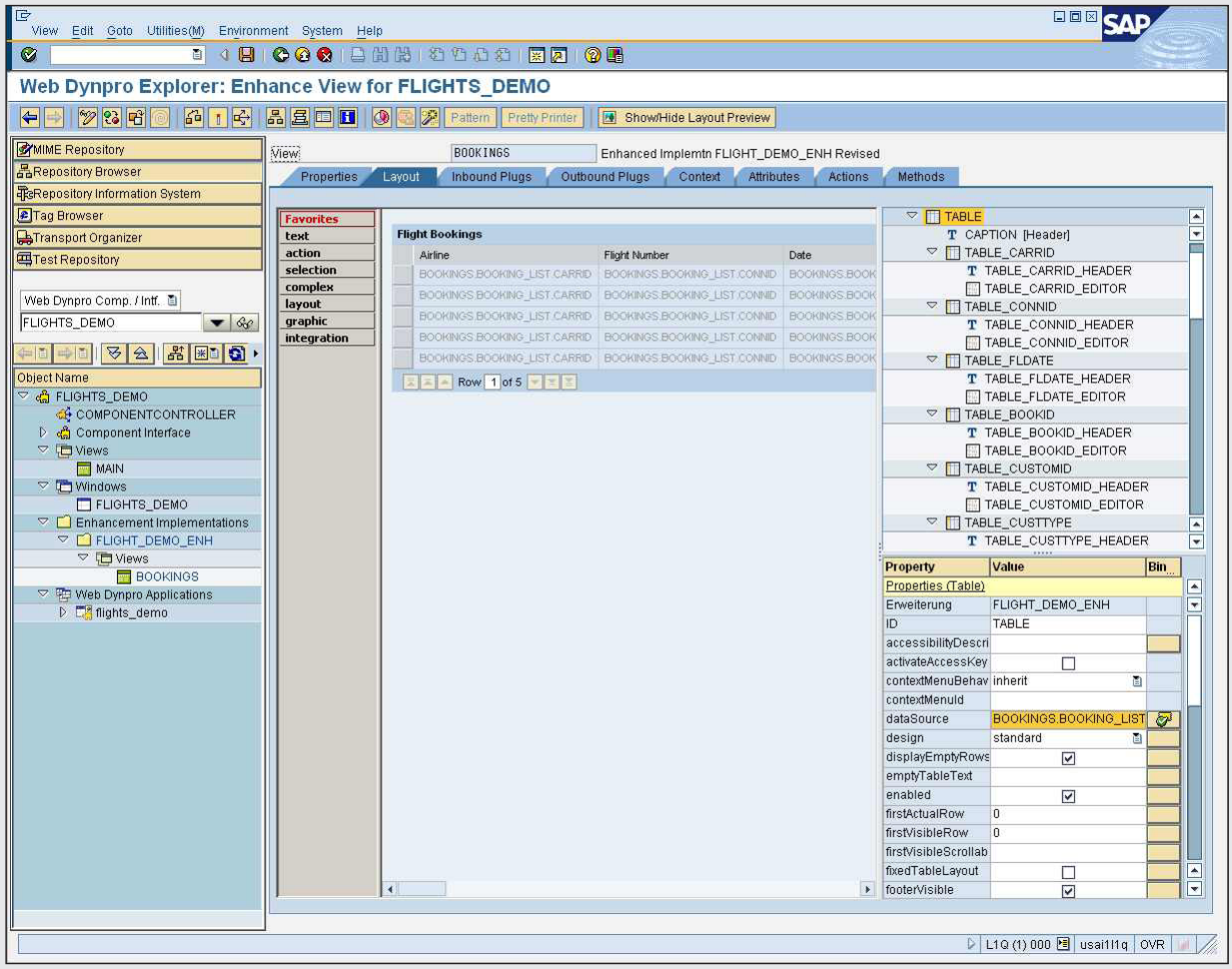

Figure 21 Creating the UI table and binding it to the BOOKING_LIST data source

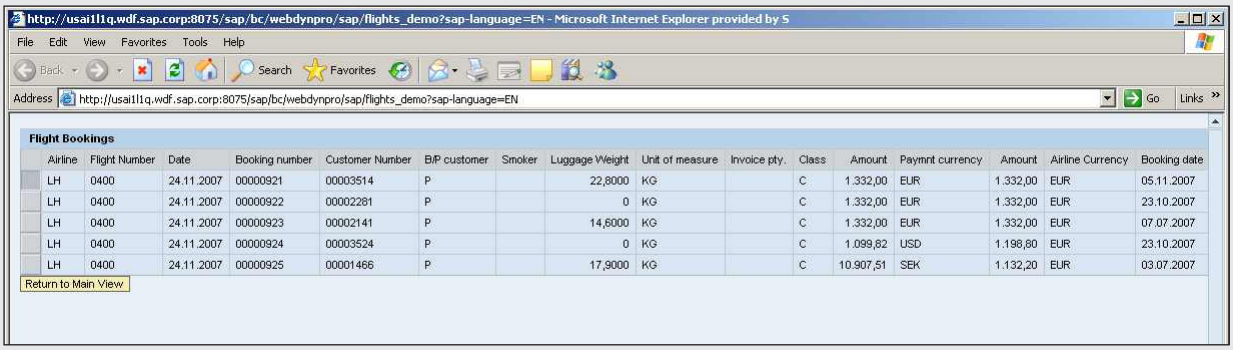

Figure 22 Implementation of the enhancement

enhancing prepackaged applications delivered by SAP as part of their Business Suite solutions. The enhancement mode present in all Web Dynpro development tools properly separates your extensions from the original applications, thereby preserving your investment in SAP solutions. If you follow a few enhancement guidelines, your work can easily be reused in subsequent enhancement packages for SAP ERP as well — even if the original applications evolve over time.

United States Postal Service Statement of Ownership, Management, and Circulation 1.Publication Title SAP Professional Journal. 2.Publication Number #0020-789. 3.Filing Date 10/1/08. 4.Issue Frequency Bi-monthly. 5.Number of Issues Published Annually 6. 6.Annual Subscription Price $895.00. 7.Complete Mailing Address of Known Office of Publication Two Washingtonian Center 9737 Washingtonian Blvd, Suite 100, Gaithersburg, Montgomery County, MD 20878-7364. Contact Person Renee Dudash Telephone (301) 287-2212. 8.Complete Mailing Address of Headquarters or General Business Office of Publisher Same as #7. 9.Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor, Editor-in-Chief Bonnie Penzias, Publisher Michael Nadeau, Managing Editor Jane Pedicini, All located at 20 Carematrix Drive, Dedham, MA 02026. 10.Owner Full Name UCG Technology, LLC Bruce Levenson, Edwin Peskowitz, Nancy Becker, Daniel Brown, Todd Foreman. Complete Mailing Address All Located At Two Washingtonian Center, 9737 Washingtonian Blvd, Suite 100, Gaithersburg, MD 20878-7364. 11.Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities None. 12. n/a. 13.Publication Title SAP Professional Journal. 14.Issue Date for Circulation Data Jul/Aug 2008. 15.Extent and Nature of Circulation 15a.Total Number of Copies (*Net Press Run)* Average No. Copies Each Issue During Preceding 12 Months 1,806, No. Copies of Single Issue Published Nearest to Filing Date 1,771; 15b.Paid Circulation (1)Mailed Outside-County Paid Subscriptions Stated on PS Form 3541 547(average), 531(actual); (2)Mailed In-County Paid Subscriptions Stated on PS Form 3541 0(average), 0(actual); (3)Paid Distribution Outside the Mails Including Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid Distribution outside USPS 0(average), 0(actual); (4)Paid Distribution by Other Classes of Mail Through the USPS 558(average), 530(actual); 15c.Total Paid Distribution 1,105(average), 1,061(actual); 15d.Free or Nominal Rate Distribution (1)Free or Nominal Rate Outside-County Copies Included on PS Form 3541 21(average), 22(actual); (2)Free or Nominal Rate In-County Copies Included on PS Form 3541 0(average), 0(actual); (3)Free or Nominal Rate Copies Mailed at Other Classes Through the USPS 14(average), 15(actual); (4)Free or Nominal Rate Distribution Outside the Mail 598 (average), 480 (actual) 15e.Total Free or Nominal Rate Distribution 633(average), 517(actual); 15f.Total Distribution 1,738(average), 1,578(actual); 15g.Copies Not Distributed 68(average), 193(actual); 15h.Total 1,806(average), 1,771(actual); 15i.Percent Paid 64%(average), 67%(actual). 16.Publication of Statement of Ownership If the publication is a general publication, publication of this statement is required. Will be printed in the Nov/Dec 2008 issue of this publication. Signature and Title of Editor, Publisher, Business Manager, Or Owner Renee M. Dudash, Director of Central Production.

