

Implementing Business Objects in CAF and Developing Web Dynpro Application



Applies to:

SAP Net Weaver CE 7.1.1 EHP1 (SP0, SP1, SP2 sand SP3). For more information, visit the [Web Dynpro Java homepage](#).

Summary

This article explains Implementing Business Objects in CAF and developing Web Dynpro application using Enterprise Java Bean (EJB) model.

Author: Sunil Kumar S S

Company: SAP Labs India Pvt. Ltd.

Created on: 04 May 2009

Author Bio



Sunil Kumar S S works as a Quality Engineer within the Web Dynpro Tools team. He has been associated with SAP Labs since July 2007. His areas of expertise include Web Dynpro Java, Data Dictionary, Web Services, CAF and NWDI.

Table of Contents

Prerequisite.....	3
Introduction	3
CAF Business Objects	3
Enterprise Java Bean (EJB) Model.....	3
Modeling Business Objects	3
Creating Composite Application Development Component.....	3
Creating Business Objects.....	4
Implementing Permission Checks for Business Objects	4
Generated EJB Implementation Class for Business Objects.....	4
Create a Web Dynpro Application consuming Business Objects Implemented in CAF.....	5
Creating a Web Dynpro Project	5
Creating the Application Unit	5
Importing Enterprise Java Bean (EJB) Model.....	6
Creating a User Interface to access data from web service	6
Build and Deploy DCs.....	8
Run Application.....	8
Output	9
Related Content.....	10
Copyright.....	11

Prerequisite

- SAP NetWeaver Developer Studio 7.1

Introduction

CAF Business Objects

A business object is a semantic entity that represents the smallest data unit to take part in a business scenario.

You use a business object to define persistable data types and to generate database tables. You can specify relations between business objects, persistency and permissions, implement or map basic methods for reading, writing, updating, deleting or querying data.

A business object is a clearly identifiable unit.

Ex: For a car rental system, you can create the following business objects:

- Car – with attributes such as license plate number, car model, number of seats, maximum speed, mileage, color
- Customer – with attributes such as the first name, last name, credit card number, address, phone, fax, e-mail, date of registration
- Order – with attributes such as the date, invoice number, or price.

You should create associations with the other business objects to specify details about the customer and car.

Enterprise Java Bean (EJB) Model

The Web Dynpro model importer supports the Common Model Interface (CMI). CMI is a layer of indirection between a framework like Web Dynpro Java, which uses a model, and a framework which implements a model, in this case the Enterprise Java Bean model. Both the using and the implementing framework are independent. Nevertheless they can exchange data via data binding, for instance.

Regarding the model, the layer grants a framework like Web Dynpro access to the business logic. It exposes business data and metadata via the CMI to the framework that sits on top of the CMI. For typed models, the structure is at least partially known at design time, and design time meta data is available. You can bind context nodes to model classes at design time and program in a type safe manner against these models. There is no urgent need for CMI metadata at runtime. For the generic models it is typical that the structure is unknown or at least not stable at design time. The CMI metadata must be available at runtime. There is only a dynamic binding between context node and model class at runtime, and no generation step is needed. The Enterprise Java Bean model allows mixtures between these two model types.

Modeling Business Objects

Creating Composite Application Development Component

Prerequisites

1. Launch the SAP NetWeaver Developer Studio.
2. Configure Application Server (AS) Java: Windows -> Preference -> SAP AS JAVA.
3. Choose Window -> Open Perspective -> Other.
4. Select Composite Application from the list.
5. Choose OK.

Procedure

1. Choose File -> New ->Project...
2. Select Development Infrastructure -> Development Component and choose Next.
3. Select Composite Application and click next
4. Select the software component for the new DC to be created in and Click next.

The default software component is Local Development -> My Components.

5. Fill the required data as follows:

Vendor : demo.sap.com

Name : Enter the name of the DC.

Caption : Enter a single line caption.

Language : Choose the language. (**Default value**: American English)

Domain : Choose the domain. (**Default value** : Basis)

Support Component : BC

6. Choose Next, then choose Finish. The application you have created is added to the Composite Application Explorer.

Creating Business Objects

1. Click on the default package 'modeled' to create the business object with the secondary mouse button and choose New Business Object.
2. Enter a name for the business object.
3. Choose where you want to put your business object node, choose *Create a new structure*.
4. Choose Finish.

Implementing Permission Checks for Business Objects

1. Double Click on the Business Object created.
2. Open the Permissions tab and enable the *Permission checks enabled* indicator.

Generated EJB Implementation Class for Business Objects

1. Double Click on the Business Object created.
2. Open the Implementation tab and click on the EJB Implementation Class hyperlink.
3. Save, generate, build and deploy your application.

Create a Web Dynpro Application consuming Business Objects Implemented in CAF

Creating a Web Dynpro Project

To manage local development objects of an application, you first need an appropriate project in the Developer Studio. Once you have set up the initial project framework, you can create the specific components of the Web application, define them, and implement the necessary source code. There is a wizard which generates an initial structure for your new Web Dynpro project. The Web Dynpro Explorer provides a logical view of the structure of a Web Dynpro project then. With this project view, you can access all Web Dynpro units. In addition, each Web Dynpro project provides access to local dictionaries, with the help of which you can create user-defined dictionary types. Additionally, you can add required projects resources, such as separate Java help classes, to the Web Dynpro project through the resources node.

Procedure

1. Select Development Infrastructure -> Development Component and choose Next.
2. Select Web Dynpro and Click next.
3. Select the software component for the new DC to be created in and Click next.
The default software component is Local Development -> My Components.
4. In the window that appears, enter a name for the project and specify the settings for Project contents and Project language.
5. Finishing the wizard you are asked to open the perspective, based on the type of the project to be created is associated with the Web Dynpro perspective.

Creating the Application Unit

The project structure you have created does not yet include all the elements that enable you to define the specific functions of the whole Web Dynpro application. These additional elements are encapsulated in a Web Dynpro component that can contain any number of windows and views for visual representation and their corresponding controllers. Moreover, you also need an object that can be used to address and display the Web Dynpro component in the Web browser. This is why you create the logical unit application as an additional object. As an independent program unit it connects to an URL that can be accessed from the user via a window within the Web Dynpro component.

Procedure

1. Expand the Web Dynpro node in the structure of the Demo project.
2. To start the relevant wizard, open the context menu of the Applications node, and choose Create Application.
3. Enter a name for the application unit, such as unit *DemoApp*, and specify the package name (example: *com.sap.demoapp*) in which the generated Java classes will be stored. Choose Next.
4. Assign a Web Dynpro component to which the application is linked. Since a Web Dynpro component has not yet been created, the wizard creates a component at this point. Accept *Create a new component* and choose Next again.
5. In the next screen you specify some general values for the component. Enter the name *DemoComp* for your Web Dynpro component and specify the package name *com.sap.demoapp.comp*. Accept the other

suggested settings and choose Finish.

Result

The application wizard performs several generation routines. Once it has finished, it adds the new application object to the structure. Additionally, it inserts a substructure for the new Web Dynpro component. Furthermore the initial *DemoCompView* is automatically created for you; you find this view under the Views node in your Web Dynpro project.

Furthermore, a window unit with name <myproject>Window, in this example with name *DemoWindow*, has been generated; including the needed plug for application starting purposes.

Importing Enterprise Java Bean (EJB) Model

The Web Dynpro model importer supports the Common Model Interface (CMI). CMI is a layer of indirection between a framework like Web Dynpro Java, which uses a model, and a framework which implements a model, in this case the Enterprise Java Bean model.

Procedure

1. To start the relevant wizard, open the context menu of the Models node, and choose *Create Model*.
In the model import wizard, choose the Enterprise Java Bean Model option and Click next.
2. Enter a name for the model, such as unit *Demomodel*, and specify the package name (example: com.sap.demomodel).
3. Click on Add Dependency, select “caf/runtime/ear” and “<cafproject>/ear”, Click Next, Finish.
4. Select “<cafproject>/ear” and click Next.
5. Select the Enterprise Java Bean Class and Click Next.
6. If proposed model class name clashes with another name then rename the model class Else Click Next.
7. If there are unresolved relations then resolve them by selecting a valid Target Model Class and Click Next.
8. Click Finish. Model is created and visible in the WD explorer.

Creating a User Interface to access data from web service

Apply Service Controller Template

Creates a controller that wraps a service. This service is either a Remote Function Call or a Web service. The wizard then creates the required contexts and an optional method for triggering the call.

Procedure

1. Add the created model to the list of Used models node under Component with the secondary mouse button, choose the created model and click Ok.
2. Select the Component Controller in Web dynpro explorer with the secondary mouse button, choose Template -> Apply
3. Select Service Controller and Click Next.
4. Select “*Request_<businessObjectName><ServiceLocal>_Create*” Model Class and Click next.
5. Select all the nodes, attributes under *Request_Create* and Click next.
6. Enable Generate method for model execution and Click on Finish.

View Context Mapping

1. Double click on the view in Web Dynpro explorer.
2. Go to Context tab, Select Context and with the secondary mouse button, Choose New > Node > Mapping.
3. Click Next, Choose Component Controller in popup Required Controllers and Click OK.
4. Select “*Request_<businessObjectName><ServiceLocal>_Create*” and click Next.
5. Select all the nodes, attributes under *Request_Create* and Click Finish.

Apply Action Button Template

The wizard for the Action Button template allows you to easily create a button, an action, an event handler, and the code for calling a method and triggering an event.

Procedure

1. Select the View in Web dynpro explorer with the secondary mouse button, choose Template -> Apply
2. Choose the option *ActionButton*. Enter a name for the template or use the default.
3. Choose *Next*. In the next window, you use the default for action, and event. For the label, Enter *Create*.
4. Choose *Next* to proceed to the wizard window for defining the methods and plugs.
5. Select the check box Call Method and for controller drop down field select the View Controller.

Corresponding method is also filled in Method drop down field.

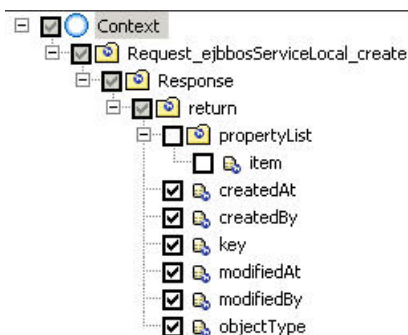
6. Close the wizard by choosing *Finish* and save your changes.

Apply Form Template

You can use the form template wizard to create a form layout for a view, based on the view context. For designing and implementing this form layout, you can use controller context data or access the data of a declared Web Dynpro model directly.

Procedure

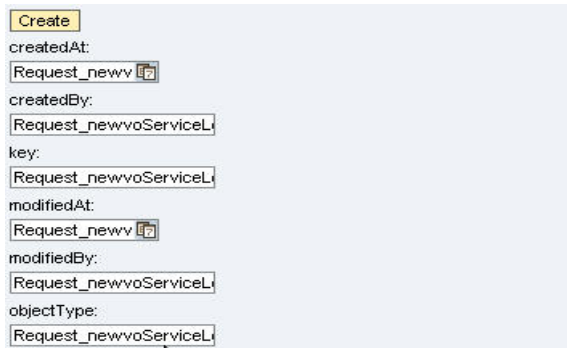
1. Select the View in Web dynpro explorer with the secondary mouse button, choose Template -> Apply
2. Choose the option Form. Enter a name for the layout template or use the default.
3. Choose Next. You see a window in which you can choose the context attributes you require for the form.
4. Select the nodes and attributes displayed in the image below.



5. Close the wizard by choosing Finish and then choose Save All to save your changes.

UI Design

1. Double Click on the view designer.
2. Go to outline View and Select Transparent Container and go to properties View and change the property value layout to Grid Layout.
3. Select each label field in outline view and go to properties View and change the property value text as shown below.



Build and Deploy DCs

Build

1. On the web dynpro project with the secondary mouse button choose Development Component > Build.
2. Select all the DCs and Click OK. Make sure that Fore Build Check box is active.

Deploy

1. On the web dynpro project with the secondary mouse button choose Deploy.
2. Enter the J2ee Engine Username and password and click OK.
3. Click OK on Deploy wizard once deployment is successful.

Run Application

To deploy the logical Application unit which is also displayed in the Web Dynpro Explorer view, you select Deploy new Archive and Run. As you can see, you would also trigger the execution process for the WD application.

Output

Create

createdAt:
4/30/2009

createdBy:
J2EE_GUEST

key:
5b777020-3555-11de-ce3

modifiedAt:
4/30/2009

modifiedBy:
J2EE_GUEST

objectType:
demo.sap.com/http://www

Related Content

[User Interface Technology](#)

[Web Dynpro Java](#)

[Composite Application Framework](#)

Copyright

© Copyright 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C->, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.