BEA WebLogic Commerce Server Components

MyBuyBeans Components Tour

BEA WebLogic Commerce Server 2.0
Document Edition 3.0
April 2000

BEA WebLogic Commerce Server Components - MyBuyBeans Components Tour

Document Edition	Date	Software Version
2.0	January 2000	BEA WebLogic Commerce Server 1.7
2.1	February 2000	BEA WebLogic Commerce Server 1.7.1
3.0	April 2000	BEA WebLogic Commerce Server 2.0

Contents

About This Document

The ultimate learning tool is a working application—and we have provided a way for you to see how it all fits together. This technical tour takes you through the entire development process of extending the sample MyBuyBeans.com site with a custom item.

The tour contains detailed information on each of the tools that you need to create your own component extensions, including the BEA WebLogic Commerce Server™ (WLCS) Smart Generator tool.

This document contains the following topics:

- Chapter 1, “Getting Started.”
- Chapter 2, “Scenario: Beans & Co., The Online Beans Distributor.”
- Chapter 3, “Design.”
- Chapter 4, “Implement.”
- Chapter 5, “Deploy.”
- Chapter 6, “Results - An Enhanced Web Site.”

What You Need to Know

This document is intended mainly for application developers who are interested in using WLCS components as software building blocks for e-commerce Web sites. This document assumes a familiarity with Java programming, Enterprise Java Beans, Rational Rose™, and BEA WebLogic Server (WLS). The WLS product serves as the platform for BEA WebLogic Commerce Server and BEA WebLogic Personalization Server.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the "e-docs" Product Documentation page at http://e-docs.bea.com.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WLCS documentation Home page at http://e-docs.bea.com/wlcs/. A PDF version of this document is also available on your local system if you installed the separate WLCS documentation kit (after installing the WLCS software kit). In the installed WLCS directory, the documentation's default starting location is:

```
\server\public_html\docs\index.htm
```

You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Commerce Server documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

Contact Us!

Your feedback on the BEA WebLogic Commerce Server documentation is important to us. Send us e-mail at **docsupport@beasys.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Commerce Server documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Commerce Server 2.0 release.

If you have any questions about this version of BEA WebLogic Commerce Server, or if you have problems installing and running BEA WebLogic Commerce Server, contact BEA Customer Support through BEA WebSupport at **www.beasys.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	**Item**
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

Convention	Item

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Example:
	`public interface Item extends ConfigurableEntity`
	`{`
	`public ItemValue getItemByValue() throws`
	`RemoteException;`
	`public void setItemByValue(ItemValue value) throws`
	`RemoteException;`
	`//...`
	`}`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Getting Started

This section contains the following topics:

Welcome to the WLCS Components Tour!

WebLogic Commerce Server components are *true* components: easy-to-use, reusable, customizable, standards-based, and extensible. This tour presents a typical e-Commerce retail scenario and shows how WebLogic Commerce Server components can help you meet the development challenges of such a business scenario. Using the fictitious *MyBuyBeans.com* retail site as an example, this tour steps you through the entire component development and deployment process.

What the Tour Includes

This tour presents a scenario in which BEA WebLogic Commerce Server components are leveraged to quickly and efficiently solve the following common business problem:

Consider an online product catalog. It is filled with items that have different descriptions, identifiers, and prices. Because many items have different attributes, they need to be stored differently. However, items do share some common attributes. Now imagine adding a new line of product with new features, attributes, and pricing schemes to the catalog. It is in a case like this where customizability and extensibility play big roles. Enter BEA WebLogic Commerce Server!

We begin by leading you through a design process in which you use Rational Rose to *extend* two WebLogic Commerce Server components: `Item` and `ItemPriceCalculationPolicy`.

Next, we show you how to implement the design. This includes exporting the model, generating EJB source code for the WebLogic Commerce Server components, and adding custom business and presentation logic.

Finally, you will deploy the newly extended WebLogic Commerce Server as a solution to the stated business problem.

Requirements

In order to understand the tour, you should be familiar with the following concepts and technologies:

- HTML
- The Java programming language
- Rational Rose and the Unified Modeling Language (UML). However, you only need to understand a small subset of the Rose features and UML.
- The WLCS Smart Generator

- BEA WebLogic Server

Additionally, we assume you have a Java compiler and Rational Rose installed on your Windows 98/NT machine.

Setting the Stage

The Company

Beans & Company is a regional powerhouse in beans. For more than a decade they have expanded their catalog business to include virtually every variety of bean. They have accomplished this through acquisition of other bean-related businesses. Aggressive marketing strategies have successfully positioned Beans & Co. to expand into the global marketplace!

The Opportunity

It is time for Beans & Company to live up to its sales potential and deliver value to its investors. To reach these goals they need to streamline their operations and quickly reach a worldwide audience. They have leading technology and a skilled technical staff. Now, they must leverage these capabilities across all of their separate business units, and deliver a unified brand image to the consumer.

The Project

Senior management has asked their technical staff to deliver a presence on the Internet. The new site must give customers access to the complete product line. To meet the expected demand, the system must be tightly integrated with the existing logistics and accounting capabilities. To minimize risk and meet the deadline, they need to use their own in-house technical and operations staff.

The Solution

Using enterprise integration, Beans & Co. is able to combine their disparate existing systems with the new technology they need. They are also able to deliver a cohesive presentation to consumers. The technology, Enterprise Java Beans, is being aggressively adopted across all the distinct platforms on which their systems run.

Beans & Co. is using BEA WebLogic Commerce Server. BEA is showing Beans & Co. the power of this new technology. New business and presentation logic is being developed using BEA components. Special implementations are being created to interface with the existing shipping, inventory, and billing systems. They are even able to leverage their extensive customer database and product catalogs.

Reminder about the WebLogic Server SP1

Before you start the WebLogic server and use the demonstration software described in this document, you must add three WebLogic Server 5.1 Service Pack 1 (SP1) files to your CLASSPATH. The files are either included in the installed WebLogic Commerce Server directory, or you can download them from the BEA Download site. The SP1 JAR files are:

- `weblogic510sp1boot.jar`
- `weblogic510sp1.jar`
- `WebLogic_RDBMS.jar`

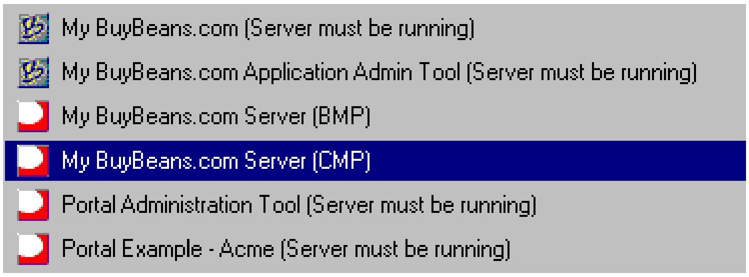
If you do not see these files in a WLCS software directory, please start at the following Web address: http://www.bea.com/download.html and follow the links to the WebLogic Server download page.

Once the files are in a directory on your system, add them to your CLASSPATH. Note that these files must be present before any other items in your CLASSPATH.

Get Started!

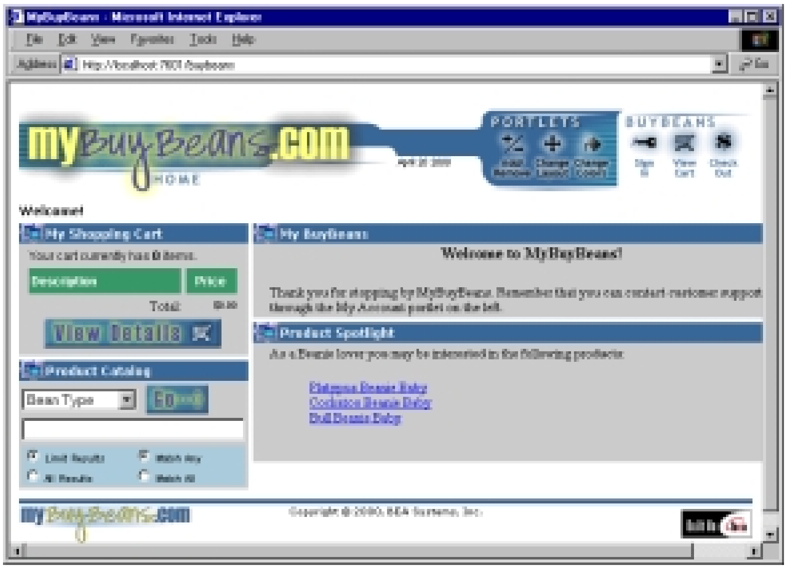
Now that you have installed WebLogic Server 5.1 with SP1 and WebLogic Commerce Server, it's time to get started with the demonstration software.

- First you will want to start the WebLogic Commerce Server on your system. From the Windows Start menu, select Start → Programs → BEA WebLogic Commerce Server → Examples → My BuyBeans.com Server (CMP). The following screen shows this item selected on the menu:


My BuyBeans.com (Server must be running)
My BuyBeans.com Application Admin Tool (Server must be running)
My BuyBeans.com Server (BMP)
My BuyBeans.com Server (CMP)
Portal Administration Tool (Server must be running)
Portal Example - Acme (Server must be running)

- Next you should connect to the site. From the Windows Start menu, select Start → Programs → BEA WebLogic Commerce Server → Examples → My BuyBeans.com (Server must be running). This step will launch the MyBuyBeans site in your default browser and access the following local URL: http://localhost:7601/buybeans.

  For example:



Your cart currently has 0 items.

Welcome to MyBuyBeans!

Thank you for stopping by MyBuyBeans. Remember that you can contact customer support through the My Account portlet on the left.

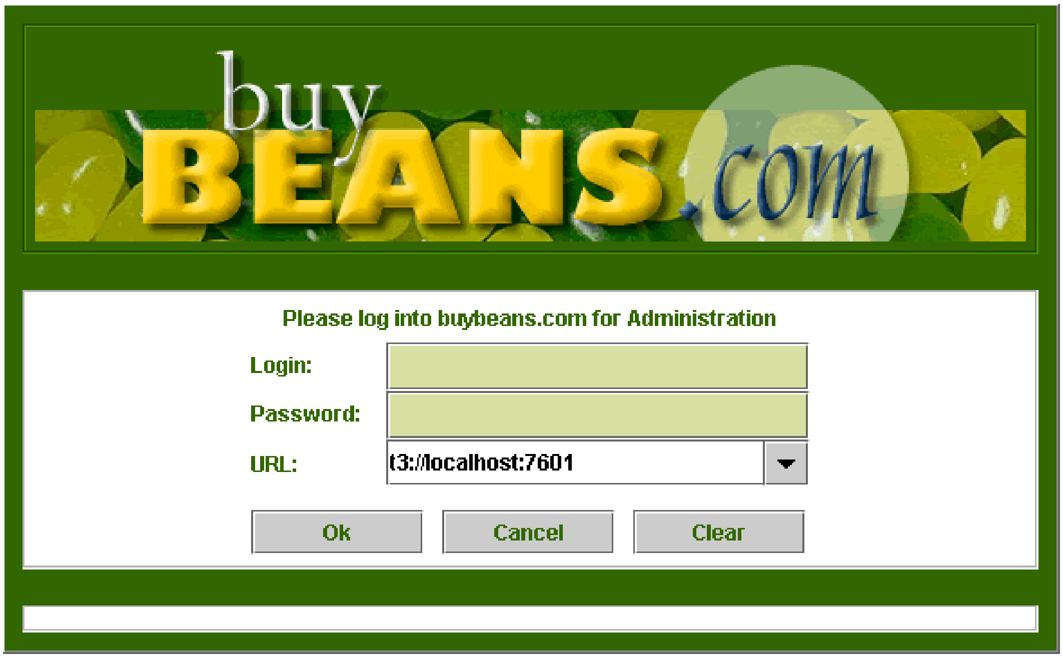
As a Beanie lover you may be interested in the following products:

Platypus Beanie Baby
Cockatoo Beanie Baby
Bull Beanie Baby

Copyright © 2000, BEA Systems, Inc.

You can sign in with username `cool` and password `bean`, or register a new user and shop!

- You can run our Java-based Admin Client to manage the site and get a behind the scenes look. From the Windows Start menu, select Start → Programs → BEA WebLogic Commerce Server → Examples → My BuyBeans.com Application Admin Tool (Server must be running). When the Java application starts, it displays the following login screen:



t3://localhost:7601

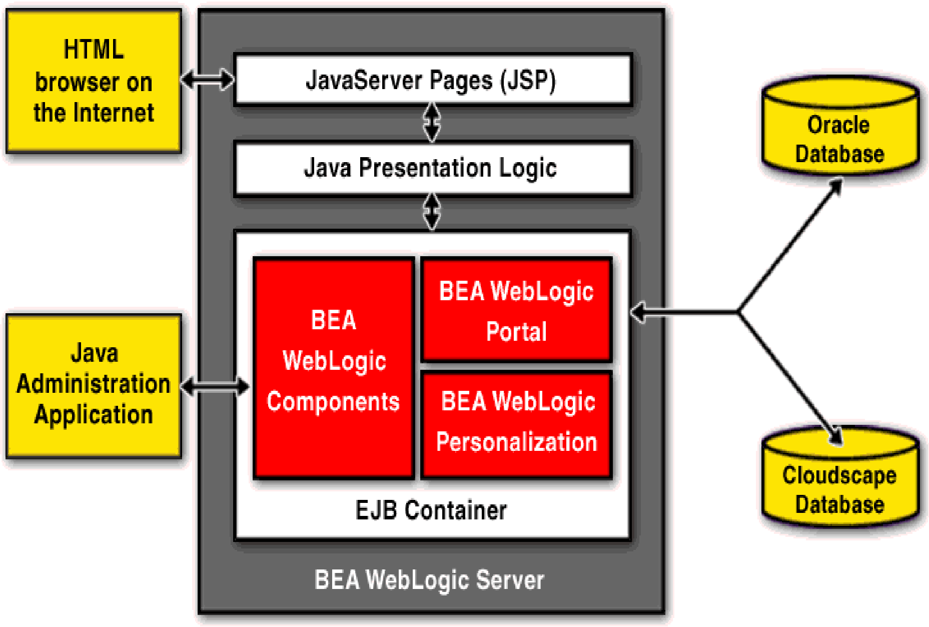
(Leave the Login and Password fields empty, and click the Ok button.)

Now that you are really impressed, you will want to find out how we did it. Of course this site was made possible by our WebLogic Commerce Server components! The site also uses industry-leading technology like the WebLogic application server and Cloudscape database. We have installed evaluation copies of these products to support this demo and to show you how powerful they are.

We also include Deployment Sets for leading relational databases using a BEA implementation for bean-managed persistence. For details, see the Deployment chapter in the document *WebLogic Commere Server Components Developer's Guide*.

The rest of this tour takes you through the process of implementing the WLCS Components solution!

MyBuyBeans.com Architecture

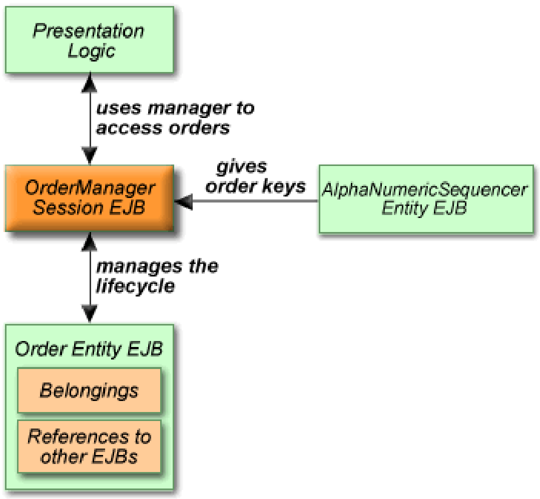


MyBuyBeans.com is an n-tier distributed system entirely built using BEA WebLogic technology. The system consits of a client front-end, a middle tier of WebLogic Enterprise JavaBeans™ (EJB) components, and a back-end database. The middle tier also contains the BEA WebLogic Portal framework and Personalization engine. All components of the middle tier run within an instance of the BEA WebLogic Application Server.

Customers access MyBuyBeans.com using a HTML Web browser. The Web browser uses HTTP to connect to a set of Java Server Pages (JSP) on the application server. A JSP consists of static HTML and embedded Java code that generates dynamic HTML contents. When accessed for the first time, each JSP is compiled by the WebLogic Application server into a Java Servlet. The output of the Java Servlet is the static HTML contained in the JSP along with the dynamic HTML that is generated by the execution of the embedded Java code.

The use of JSPs both simplifies and streamlines the process of dynamic HTML generation. It allows a Web designer to design and implement a Web page using the normal HTML design tools that he or she is familiar with, while at the same time, freeing an Application developer to embed the necessary Java code for the dynamic content of the page. The JSP source is provided so that you can get a feel for just how easy it is to use and modify.

Using the embeded Java code found in the JSP, the generated Servlets make calls to presentation logic written in Java. The MyBuyBeans.com presentation logic is written as a set of Java classes. These classes encapsulate data access to the middle tier of WebLogic Components, thus making the JSPs simpler and easier to maintain. The presentation logic objects use the Java Naming and Directory Interface (JNDI) to locate the appropriate WebLogic Components. Once a reference is obtained to a WebLogic Component, it can be used as if it were a local resource.



Component Interaction Example

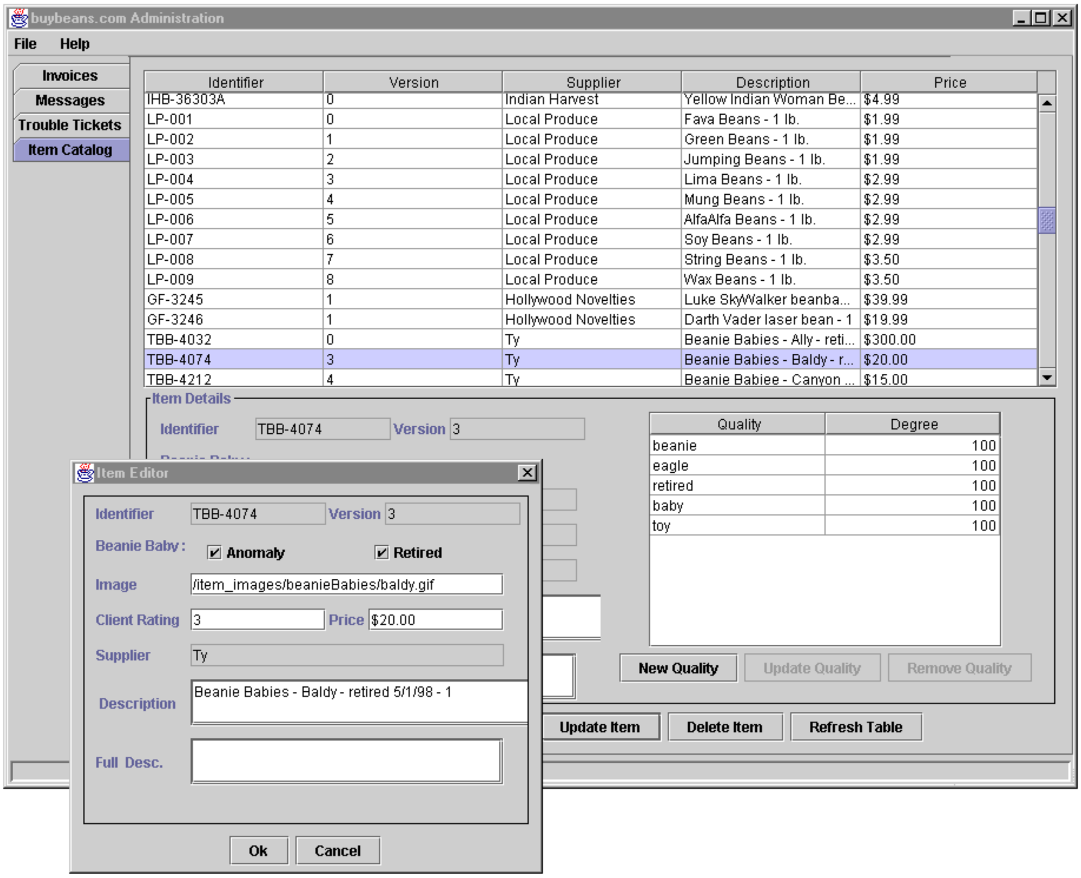
At the heart of the system are WebLogic Commerce Server, a set of 80 EJBs that provide most of the functionality of essential e-business. These beans run inside an "EJB Container" on the application server and expose the MyBuyBeans back-end systems. Since the components can be found using the Java Naming and Directory Interface (JNDI), the can be accessed from anywhere on the network. To simplify delivery of this demonstration, all of the components and servlets are running together in one instance of WebLogic Server. When deploying this application on an enterprise level, the various components will run on different machines and, potentially, on different architectures.

MyBuyBeans.com uses both Session and Entity WebLogic Components. Component access to the MyBuyBeans back-end is facilitated by the WebLogic Application Server which provides a JDBC connection pool to the MyBuyBeans database. The Entity beans in this application use "container-managed" persistence. This means that WebLogic Server manages storing the contents of an Entity EJB to the database when the bean's persistable attributes change. The demo MyBuyBeans.com application uses a Cloudscape database. The database is lightweight, and runs embedded within the WebLogic Server, so we do not need to start it separately.

MyBuyBeans.com uses the concept of pluggable methods to set up pricing policies for its products and to calculate the prices based on these policies. Three ItemPriceCalculationPolicies are used, each holding business logic specific to the pricing of a different class of Items. We have included the UML diagram and documentation for these items and their pricing policies.

In addition to the WebLogic Components, MyBuyBeans.com takes advantage of the features of the WebLogic Portal framework and WebLogic Personalization engine. The WebLogic Portal framework allows an Internet portal site to be quickly assembled from a collection of *portlet applications*. Additionally, the Portal framework provides mechanisms for portal administration and user personalization. Portlet applications (also referred to as Portlets) are JSP and HTML pages that present dynamic content in a portal application. The content is laid out in the portal page by the Portal framework according to a user's personalized information as stored by the WebLogic Personalization Server.

Just so you don't think that this is browser-only technology, we have also created a back-office administration application implemented as a standalone Java application. This application provides a unified interface to commonly performed tasks at MyBuyBeans.com (such as managing the item catalog, or dealing with trouble tickets from customers). This application accesses remote objects directly to accomplish its goals.



Identifier	Version	Supplier	Description	Price
IHB-36303A	0	Indian Harvest	Yellow Indian Woman Be...	$4.99
LP-001	0	Local Produce	Fava Beans - 1 lb.	$1.99
LP-002	1	Local Produce	Green Beans - 1 lb.	$1.99
LP-003	2	Local Produce	Jumping Beans - 1 lb.	$1.99
LP-004	3	Local Produce	Lima Beans - 1 lb.	$2.99
LP-005	4	Local Produce	Mung Beans - 1 lb.	$2.99
LP-006	5	Local Produce	AlfaAlfa Beans - 1 lb.	$2.99
LP-007	6	Local Produce	Soy Beans - 1 lb.	$2.99
LP-008	7	Local Produce	String Beans - 1 lb.	$3.50
LP-009	8	Local Produce	Wax Beans - 1 lb.	$3.50
GF-3245	1	Hollywood Novelties	Luke SkyWalker beanba...	$39.99
GF-3246	1	Hollywood Novelties	Darth Vader laser bean - 1	$19.99
TBB-4032	0	Ty	Beanie Babies - Ally - reti...	$300.00
TBB-4074	3	Ty	Beanie Babies - Baldy - r...	$20.00
TBB-4212	4	Ty	Beanie Babiee - Canyon ...	$15.00

Item Details

Identifier TBB-4074 Version 3

Quality	Degree
beanie	100
eagle	100
retired	100
baby	100
toy	100

Item Editor

Identifier TBB-4074 Version 3

Beanie Baby: ☑ Anomaly ☑ Retired

Image /item_images/beanieBabies/baldy.gif

Client Rating 3 Price $20.00

Supplier Ty

Description Beanie Babies - Baldy - retired 5/1/98 - 1

Full Desc.

About the MyBuyBeans.com Databases

The MyBuyBeans.com application can use a Cloudscape database or an Oracle database.

The Cloudscape database is lightweight, and runs embedded in the application server, so you don't need to start it separately. As you run the application, it is important to keep in mind that because this evaluation version of Cloudscape supports database-level locks only, it handles only a single user. **While the Cloudscape database is locked by one user, another user does not have access to the database.**

The MyBuyBeans.com database contains the following tables corresponding to the beans deployed for the MyBuyBeans application:

TABLE NAMES	
ALPHANUMERICSEQUENCER	ITEM
BASICBEAN	ITEMINVENTORY
BEANIEBABY	ITEMQUALITIES
BUSINESSSMARTWORKAREA	ITEMSBYQUALITY
COFFEEBEAN	JELLYBEAN
CUSTOMER	MAILBOX
CUSTOMERPROFILE	PACKINGLIST
EBUSINESSSESSION	SHIPPINGMETHOD
INVENTORYRECORD	TORDER
INVOICE	TROUBLETICKET

The following examples assume that `WL_Commerce_Home` is the directory in which you installed the WebLogic Commerce Server software.

To create the database tables, run the `WL_Commerce_Home\db\Oracle\BuyBeansOracle805.sql` file, or the `WL_Commerce_Home\db\Cloudscape\BuyBeansCloudscape (.bat or .sh)` procedure.

With the WebLogic server running, run the DataLoader procedure (.bat or .sh) to populate the databases.

To run `DataLoader.bat` on NT systems, use:

```
WL_Commerce_Home\bin\win32\DataLoader.bat
```

To run `DataLoader.sh` on Solaris 7 systems use:

```
WL_Commerce_Home/bin/solaris2/win32/DataLoader.sh
```

The procedure will populate the database with all the items for the application. The database is ready to run after you complete these steps. You can now add users and other data by using the WebLogic Personalization Server Administration Tools.

If you make any structural changes to the MyBuyBeans database, you will need to make the appropriate changes in the deployment descriptor of the particular beans. Sample XML deployment descriptor files already exist for the MyBuyBeans tour. See the following files that reside under the `WL_Commerce_Home/src/examples/buybeans/tour/Meta_Inf` directory:

- `ejb-jar.xml`
- `BeanieHat-cmp-rdbms-jar.xml`
- `weblogic-ejb-jar.xml`

2 Scenario: Beans & Co., The Online Beans Distributor

It’s Monday morning. You are called into a meeting and told that Beans & Co. is adding a new line to its many products: beanie hats! Since the beanie hat market is expected to be very profitable and Marketing plans to start running beanie hat advertisements next week, this new product line needs to be available on-line at the MyBuyBeans.com site "yesterday". Furthermore, Management has come up with a unique pricing policy for beanie hats: propellers increase the base price of a beanie hat by two dollars each. As the lead architect of the MyBuyBeans.com site, you have been chosen to come up with a solution.

After convincing Marketing and Management that "yesterday" is not a feasible due date, you agree to finish the development and testing by the end of the week. As you are leaving the meeting, one of the VPs mentions that this is crucial to the success of the company:

> *We know you can even do it in less than a week. No one here has forgotten what your team accomplished in the original launch of our MyBuyBeans site!*

You leave the room thinking, "No problem. I’ll just run into a telephone booth, and transform into a super-human being! Everything will be done on time!" At the same time, you feel proud of your team’s accomplishments in developing the MyBuyBeans.com site. As you walk back to your desk, you start thinking about the effort involved in creating the site.

You recall that you implemented the site using BEA WebLogic Commerce Server**.** This product provides a significant portion of the business functionality and they were easy-to-use, reusable, customizable, standard-based, and extensible. As a result, you and your team finished the site in 1/3 the expected time. Aaah…yes! Maybe you can extend the basic WebLogic Commerce Server Item to add unique properties for beanie hats. You could also leverage the pluggable WebLogic Commerce Server methods for pricing flexibility. You slowly being to realize that your assignment is actually quite possible!

CHAPTER

3 Design

This section contains the following topics:

Overview of Design Considerations

Having received instructions from senior management to extend the MyBuyBeans.com site, the MyBuyBeans technical staff sits down to plan their course of action. Being the leader of the newly assembled team, you decide to first model a solution to the problem.

Senior management has given your team the beanie hat attributes that are of importance to customers: color and the number of propellers. The MyBuyBeans product catalog must contain information on both of these attributes and both customers and MyBuyBeans staff must be able to search for beanie hats using these properties. In addition to these requirements, management has expressed interest in a specialized pricing policy for beanie hats. Being a knowledgeable WebLogic Commerce Server developer, you decide to use WebLogic Commerce Server to develop a solution.

Looking at the Rational Rose model of the existing MyBuyBeans solution, you quickly realize that all items in the MyBuyBeans catalog are represented by a **Configurable Entity** Component. You also notice that each item has a corresponding **Business Policy** Component for implementing specialized pricing policies. As such, you decide to extend the existing model with two new Components: BeanieHat and BeanieHatPricePolicy.

Prerequisites for Design Work

In order to fully understand this section of the tour, it is recommended that you have a general understanding of Rational Rose and UML. For information on these two topics, please see the Development Process chapter in the *BEA WebLogic Commerce Server Components Developer's Guide*.

Directions for Designing

Step 1. Open a copy of BEA WebLogic Commerce Server Rational Rose model.

1. Navigate to the *model* directory found under the WebLogic Commerce Server installation directory.
2. Create a new subdirectory called *tour*. Be sure to name this new subdirectory "*tour*" exactly as given here (all lower-case), or else the rest of the technical demonstration will not work.

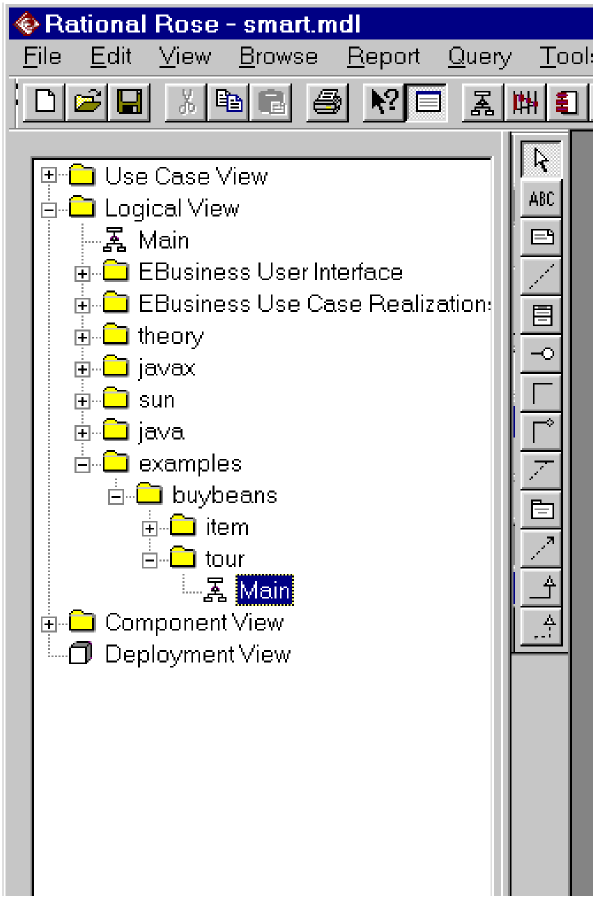
3. Create a copy of the Rational Rose model *WebLogicCommerce.mdl* (found in the *WebLogicCommerce* directory) in the new *tour* directory.
4. Start Rational Rose.
5. Click on the **Open** menu item in the **File** menu.
6. Navigate to the *model\tour* directory found under the WebLogic Commerce Server installation directory.
7. Double click on *WebLogicCommerce.mdl*. This will open Rational Rose.

Step 2. Create a new MyBuyBeans tour package for the new components.

1. Expand the *Logical View* folder in the Rational Rose navigation window.
2. Expand the *examples* folder.
3. Right click on the *buybeans* folder.
4. Highlight the **New** menu item and then click on the **Package** menu item.
5. Name the new package *tour*.

Step 3. Add a new class diagram to the tour package.

1. Right click on the *tour* folder.
2. Highlight the **New** menu item and then click on the **Class Diagram** menu item.
3. Name the class diagram *Main*.
4. Double click on the *Main* class diagram icon.
5. Your Rational Rose navigation window should now look something like the following:

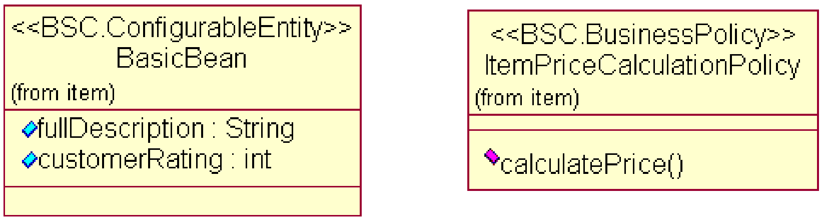


Step 4. Add a BasicBean Component to the class diagram.

1. Expand the *examples* directory in the *Logical View* folder.
2. Expand the *buybeans* directory.
3. Expand the *item* directory.
4. Click and drag the **BasicBean** class icon onto the Main class diagram window.

Step 5. Add an ItemPriceCalculationPolicy Component to the class diagram.

1. Expand the *theory* directory in the *Logical View* folder.
2. Expand the *smart* directory.
3. Expand the *ebusiness* directory. Expand the *item* directory.
4. Click and drag the **ItemPriceCalculationPolicy** class icon onto the Main class diagram window.
5. Your Main class diagram window should now contain something like the following:



Step 6. Create a new BeanieHat Component that extends the BasicBean Component.

1. Click on the Class button on the center toolbar. This button looks like the following:

2. Create a new class by clicking on the Main class diagram window.
3. Name the new class **BeanieHat**.

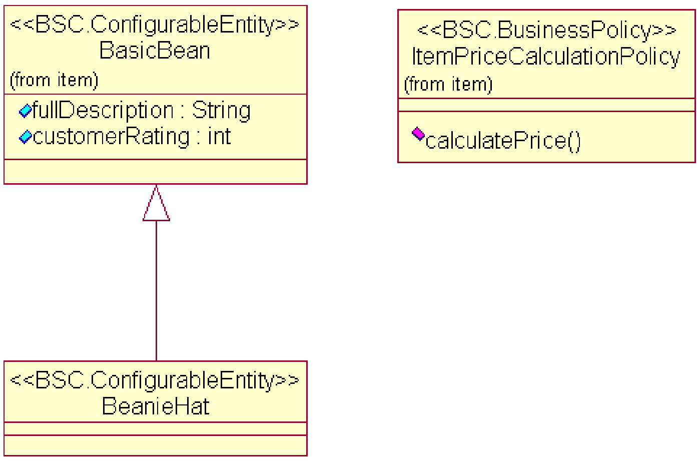
4. Right click on the **BeanieHat** class and then click on the **Open Specification** menu item.

5. Select *BSC.ConfigurableEntity* in the **Stereotype** pull down menu. Press the **OK** button.

6. Click on the Generalization button on the center toolbar. This button looks like:

7. Place the cursor over the **BeanieHat** class. Click and drag the cursor over the **BasicBean** class and then release the mouse button.

8. Your Main class diagram window should now contain something like the following:

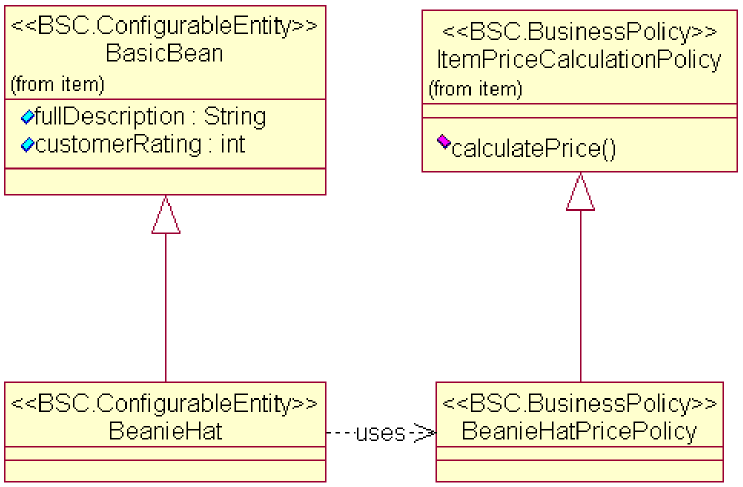


Step 7. Create a new BeanieHatPricePolicy Component that extends the ItemPriceCalculationPolicy Component.

1. Click on the Class button on the center toolbar.

2. Create a new class by clicking on the Main class diagram window.

3. Name the new class **BeanieHatPricePolicy**.
4. Right click on the **BeanieHatPricePolicy** class and then click on the **Open Specification** menu item.
5. Select *BSC.BusinessPolicy* in the **Stereotype** pull down menu. Press the **OK** button.
6. Click on the Generalization button on the center toolbar.
7. Place the cursor over the **BeanieHatPricePolicy** class. Click and drag the cursor over the **ItemPriceCalculationPolicy** class and then release the mouse button.
8. Click on the Dependency button on the center toolbar. This button looks like:
9. Place the cursor over the **BeanieHat** class. Click and drag the cursor over the **BeanieHatPricePolicy** class and then release the mouse button. Right click on the resulting dotted line and then click on the **Open Specification** menu item. Name the dependency *uses*. Press the OK button.
10. Your Main class diagram window should now contain something like the following:



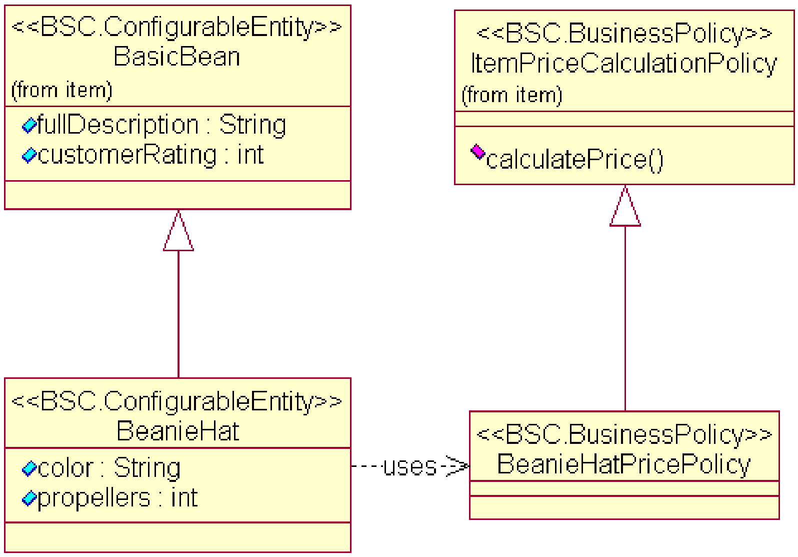
Step 8. Add attributes to the BeanieHat Component.

1. Right click on the **BeanieHat** class in the Main class diagram.
2. Click on the **Open Specification** menu item.
3. Click on the **Attributes** tab in the **Class Specification** dialog.
4. Right click on the attributes list and select the **Insert** menu item. Name the new attribute *color.*
5. Double click on the *color* attribute. This should open an **Attribute Specification** dialog.
6. Type *String* in the **Type** field and select the **Public** radio button in the **Export Control** panel. Press the **OK** button.
7. Right click on the attributes list and select the **Insert** menu item. Name the new attribute *propellers.*

8. Double click on the *propellers* attribute. This should open an **Attribute Specification** dialog.
9. Type *int* in the **Type** field and select the **Public** radio button in the **Export Control** panel. Press the **OK** button.
10. Press the **OK** button in the **Class Specification** dialog.

Step 9. Save and review the final Rational Rose model.

1. Click on the **Save** menu item in the **File** menu.
2. Your final Main class diagram should contain something like the following:



CHAPTER

4 Implement

This section contains the following topics:

Overview of Implementation

Having finished using Rational Rose to model two new components that will serve as the basis for the MyBuyBeans site modifications, you are now prepared to implement your solution. You assemble your team and plan your next steps.

Because you are a knowledgeable WebLogic Commerce Server developer, you know that the implementation process involves exporting the Rational Rose model using the **BEA Rational Rose Plugin**, generating Enterprise Java Bean (EJB) implementations of your WebLogic Commerce Server using **the WLCS Smart Generator**, and then adding any necessary custom business logic to the EJBs. You also realize that presentation logic for the new BeanieHat Component must be developed.

Prerequisites for Implementing

In order to fully understand this section of the tour, it is recommended that you have a general understanding of BEA Smart Generator application, HTML, and the Java programming language. You must also have successfully completed all steps of the Design section of the tour.

Directions for Implementing

Step 1. Open the copy of the WLCS Rational Rose model.

1. Start Rational Rose.
2. Click on the **Open** menu item in the **File** menu.
3. Navigate to the *model\BEA WebLogicCommerce* directory found under the WebLogic Components installation directory.
4. Double click on *BEA WebLogicCommerce.mdl*.

Step 2. Export the model using the WLCS Rational Rose Plug-In.

1. Click on the **WebLogicCommerce** menu item in the **Tools** menu.
2. Click on the **Export Model As** submenu item.
3. Navigate to the *src* directory found under the WebLogic Components installation directory.
4. Name the file *bbtour* and then click on the **Save** button.
5. Click the **OK** button in the **Success** dialog.

Step 3. Create a new SmartGenerator project.

1. Start the **WLCS Smart Generator**. On Windows NT, you can do this by selecting the following from the Windows Start menu:

   Start —> Programs —> WebLogic Commerce Server —> Smart Generator

2. Click on the **New Project** menu item in the **File** menu. This should open a **Project Properties** dialog.

3. Name the project *bbtour.*

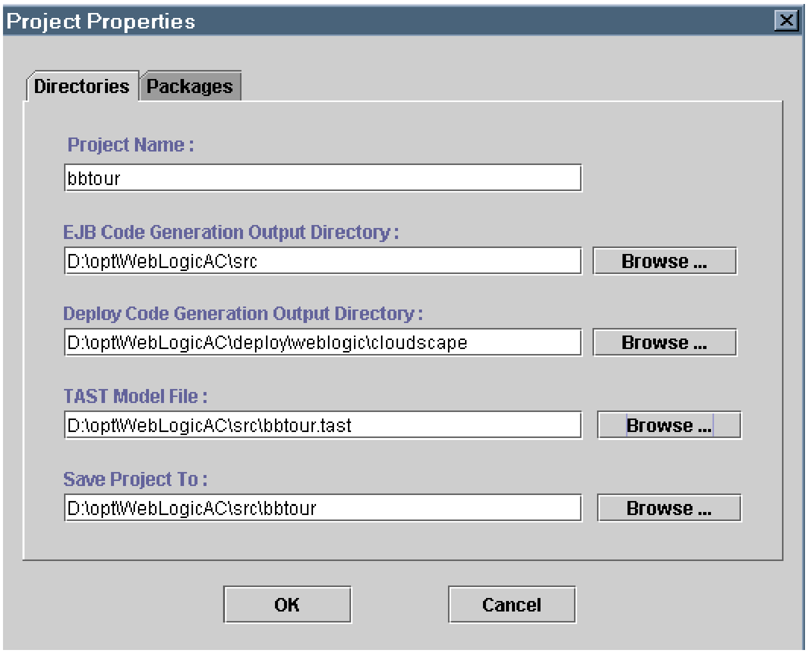
4. Designate the *src* directory found under the WebLogic Components installation directory as the **EJB Code Generation Output Directory**.

5. Designate the *deploy\weblogic\cloudscape\src* directory found under the WebLogic Components installation directory as the **Deploy Code Generation Output Directory**.

6. Designate the *bbtour.tast* file generated in Step 2 as the **TAST Model File**.

7. The **Project Properties** dialog should now look something like the following:

Note: This assumes that the WebLogic Components installation directory is *D:\opt\WebLogicCommerce*:



Project Name : bbtour

EJB Code Generation Output Directory : D:\opt\WebLogicAC\src

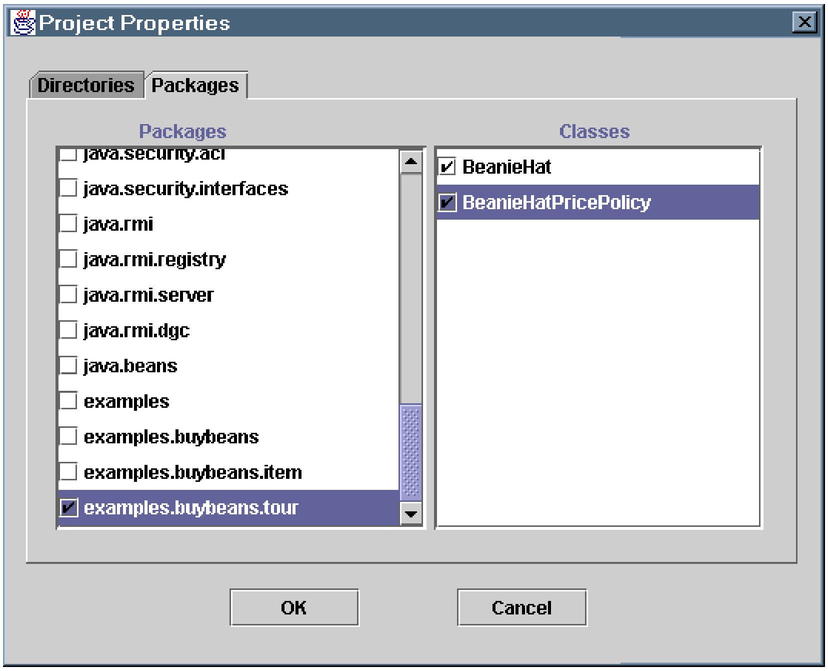
Deploy Code Generation Output Directory : D:\opt\WebLogicAC\deploy\weblogic\cloudscape

TAST Model File : D:\opt\WebLogicAC\src\bbtour.tast

Save Project To : D:\opt\WebLogicAC\src\bbtour

Step 4. Configure the project.

1. Click on the **Packages** tab.
2. Scroll down in the **Packages** pane until the *examples.buybeans.tour* entry is visible, then click on this entry.
3. In the **Classes** pane, double-click on the boxes next to the **BeanieHat** and **BeanieHatPricePolicy** classes. This should check both of the boxes.
4. The **Project Properties** dialog should now look something like the following:



Packages:
- [ ] java.security.acl
- [ ] java.security.interfaces
- [ ] java.rmi
- [ ] java.rmi.registry
- [ ] java.rmi.server
- [ ] java.rmi.dgc
- [ ] java.beans
- [ ] examples
- [ ] examples.buybeans
- [ ] examples.buybeans.item
- [✓] examples.buybeans.tour

Classes:
- [✓] BeanieHat
- [✓] BeanieHatPricePolicy

5. Click on the **OK** button.

Step 5. Generate EJB implementations of the BeanieHat and BeanieHatPricePolicy Components.

1. In the **Project** pane, highlight the project entry that was created in Step 4 by clicking on it.
2. Click on the **Generate** button. Your EJB implementations of the **BeanieHat** and **BeanieHatPricePolicy** WebLogic Components are now being generated.
3. Wait for the Smart Generator to finish and then click on the **Exit** button.

Step 6. Implement custom business logic in the BeanieHatPricePolicy Component

1. Using your favorite editor, open the file *BeanieHatPricePolicyImpl.java* located in the *src\examples\buybeans\tour* directory under the WebLogic Components installation directory. This file is one of several generated by the BEA Smart Generator.

2. Locate the `calculatePrice` function and find the following line of code:

```
return null;
```

3. This is the default implementation of the pricing policy associated with the **BeanieHat** Component. We would like to implement the custom pricing policy for a beanie hat (as outlined by Management), so **REPLACE** the above line of code with the following lines, and then save the file:

```
// Policy: Price = base price + $2.00*propellers
// Use values object for speed! This way we don’t need to
// make a heavy remote + database call every time.

try
{
  // Get Value Object BeanieHatValue
  BeanieHatValue bhv = ((BeanieHat) item).getBeanieHatByValue();


  // Get item’s base price
  theory.smart.axiom.units.Price p = bhv.price;


  // Calculate the price of the beanie hat
  double newValue = p.getValue() + 2.00*bhv.propellers;


  // Return a new price. Do NOT re-set the item’s price.
  // That should always be the base price

  theory.smart.axiom.units.Price newPrice =
theory.smart.axiom.units.PriceHome.create();
  newPrice.setValue(newValue);

  return newPrice;
}
```

```
catch (ClassCastException ce)
{

  // If we are here then the caller did not pass us a BeanieHat

  throw new java.rmi.RemoteException("BeanieHatPricePolicy: item is not a
BeanieHat");
}
```

Step 7. Implement presentation logic for the new BeanieHat Component.

1. Using your favorite editor, open the file *productDetails.jsp* located in the *server\public_html\portals\buybeans\portlets* directory under the WebLogic Components installation directory. This file contains the presentation logic for each beanie item WebLogic Component.

2. Locate the following line of code:

```
<%@ page import="examples.buybeans.item.*" %>
```

3. Insert the following line of code **AFTER** the above line of code:

```
<%@ page import="examples.buybeans.tour.*" %>
```

4. Locate the following line of code:

```
<%-- COFFEE BEAN SPECIFIC PRODUCT INFORMATION     --%>
```

5. Insert the following lines of code **BEFORE** the above line of code (and **AFTER** any code that preceeds the above line of code) and save the file. You can get a little creative here if you are familiar with HTML:

```
<%-- BEANIEHAT SPECIFIC PRODUCT INFORMATION     --%>
<%
   if (myItemValue instanceof BeanieHatValue)
   {
   %>
  <tr>
    <td>
      <table width="95%" border="0" cellpadding="3" bgcolor="#FFFFFF"
align="center">
       <tr bgcolor="#339966">
          <td> <font face="Arial, Helvetica, sans-serif" size="2"
color="#FFFFFF"><b>Color:
            <%= ((BeanieHatValue) myItemValue).color %></b></font>
          </td>
          <td> <font face="Arial, Helvetica, sans-serif" size="2"
color="#FFFFFF"><b>Propellers:
            <%= ((BeanieHatValue) myItemValue).propellers %></b></font>
          </td>
        </tr>
        <tr bgcolor="#CCCCCC">
          <td colspan=3> <font face="Arial, Helvetica, sans-serif" size="2"
color="#000000">Each propeller raises the base price of a beanie hat by
$2.00.</font> </td>
        </tr>
      </table>
    </td>
  </tr>
<%
   }
%>
```

CHAPTER

5 Deploy

This section contains the following topics:

Overview of Deployment

After a short implementation phase, your team has now finished coding the My BuyBeans site solution. It is now time to finally deploy the new customized BEA WebLogic Commerce Server**.** You once again assemble your team.

You explain to the team members that deploying BEA WebLogic Commerce Server requires a few easy steps. First, you must compile the newly created components. Next, you must create a EJB deployment descriptor (DD) for the generated EJB implementation of the BeanieHat Component. This serialized object describes declarative information (i.e. information that is not included directly in the EJB code) that an Application Server uses to deploy the EJB. The Application Server in which the new WebLogic Commerce Server are to deployed then needs to be configured.

Finally, BeanieHat Component instances must be created using beanie hat data given to your team by Marketing. With these four easy steps now in mind, your team proceeds.

Prerequisites for Deployment

In order to fully understand this section of the tour, it is recommended that you have a general understanding of the BEA WebLogic Application Server and its property file settings. For more information on this topic, please see the WebLogic Server (WLS) documentation at http://www.weblogic.com/docs51/resources.html. You must also have successfully completed all steps of the Implement section of the tour.

Also, before starting the deployment steps below, be sure to remove the custom "Finder Methods" manually from the source files (`.java`) that are generated from the model by the Smart Generator.

Directions for Deploying

Step 1. Compile the BeanieHat and BeanieHatPricePolicy Components.

Note: Ensure the BuyBeans server is shut down prior to compiling the tour.

1. Open a command prompt.
2. Navigate to the *bin\win32* directory under the WebLogic Commerce Server installation directory.
3. Type *tour-build* at the command line and hit Enter.
4. The *tour-build* file contains a `javac` command to compile and install both Components. For more information, you might want to look at the contents of this file.

Step 2. Create all stubs and skeletons for the BeanieHat Component.

1. Open a command prompt (if one is not already open).
2. Navigate to the *bin\win32* directory under the WebLogic Commerce Server installation directory.
3. Type *tour-deploy* at the command line and hit Enter.
4. The *tour-deploy* file contains the weblogic.ejbc commands to create all stubs and skeletons for the BeanieHat Component. For more information, you might want to look at the contents of this file.

Step 3. Configure the target application server (BEA WebLogic Application Server).

1. If the MyBuyBeans.com WebLogic Application Server is running, shut it down.
2. Using your favorite editor, open the file `weblogic.properties` located in the WebLogic Commerce Server installation directory.
3. Locate the following line in the file (where `WL_Commerce_Home` is the WebLogic Commerce Server installation directory):

   ```
   WL_Commerce_Home\lib
   ```
4. Add the following line (again, where `WL_Commerce_Home` is the WebLogic Commerce Server installation directory):

   ```
   WL_Commerce_Home\lib\Tour.jar
   ```
5. This statement tells the WebLogic Application Server the location of the deployment descriptor for the BeanieHat Component so that it can deploy the Component.

Step 4. Create instances of the BeanieHat Component based on Marketing beanie hat data.

1. The beanie hat data is stored in a Microsoft Excel worksheet named BeanieHats.xls. It can be found in the `WL_Commerce_Home\db` directory. Please feel free to add or remove any beanie hat data in this file. Just be sure to export the worksheet as a tab delimited file named `BeanieHats.txt`. This file must be saved in the same directory as the `BeanieHats.xls` file. If you do add any new beanie hat data, you must include all data declared as required in the Microsoft Excel worksheet.
2. From the Windows Start menu, select Start → Programs → BEA WebLogic Commerce Server → Examples → My BuyBeans.com Server (CMP) to start the My BuyBeans.com WebLogic application server, and wait for it to initialize.
3. Open a command prompt (if one is not already open).
4. Navigate to the `bin\win32` directory under the WebLogic Commerce Server installation directory.
5. Type `tour-loader` at the command line and hit Enter.

CHAPTER

6 Results - An Enhanced Web Site

This topic includes the following sections:

Overview of Results

Having finished deploying the My BuyBeans site solution, it is now time to see the results of your hard work. Fortunately for you and your team, everything goes off without a hitch. The site is a success and Beans & Co. makes millions selling beanie hats. As a result, you and your team are heralded as miracle workers and given huge raises. Sitting at your desk (wearing a beanie hat, of course) you think to yourself:

> *Thank goodness for BEA WebLogic Commerce Server components. Without them, I'm not sure this would have been possible. Even the coffee seems to taste better now.*

The End.

Prerequisites for Running the Web Site

In order to fully understand this section of the tour, you must have successfully completed all steps of the Deploy section of the tour.

Directions for Viewing the Web Site

Step 1. Start the My BuyBeans.com WebLogic Application Server.

Step 2. Browse the site and check out Beans & Co. newest offering: the beanie hat!

1. Click here to visit the site.
2. In the search portlet, type *red* in the text box and click the **Go** image.
3. Several red beanie hat selections should appear. Click on one of them.
4. Check out how your beanie hat presentation logic looks. Also, check out the price of the beanie hat. It should be in accordance with the customized pricing policy that you implemented. Cool, eh?

Index