BEA WebLogic Components

Hands-On Technical Tour

BEA WebLogic Components 1.7.1
Document Edition 2.1
February 2000

Copyright

Restricted Rights Legend

Trademarks or Service Marks

WebLogic Components

Document Edition	Date	Software Version
2.0	January 2000	BEA WebLogic Components 1.7
2.1	February 2000	BEA WebLogic Components 1.7.1

Contents

6. Results - An Enhanced Web Site

About This Document

The ultimate learning tool is a working application -- and we've 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 My BuyBeans.com site with a custom item.

The tour contains detailed information on each of the tools that you need to create your own WebLogic Application Component extensions, including the SmartGenerator generation tool.

What You Need to Know

This document is intended mainly for application developers who are interested in using BEA WebLogic application components as software building blocks for eBusiness. It assumes a familiarity with Java programming, Enterprise Java Beans, and the BEA WebLogic Application Server, which serves as the platform for BEA WebLogic Application Components.

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.beasys.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 WebLogic Components documentation Home page on the e-docs Web site (and also on the documentation CD). 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 Components 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/.

Related Information

The following BEA WebLogic Enterprise documents contain information that is relevant to using the idltojava compiler and understanding how to implement Java CORBA applications in the WLE system.

For more information in general about Java IDL and Java CORBA applications, refer to the following sources.

- The OMG Web Site at http://www.omg.org/
- The Sun Microsystems, Inc. Java site at http://java.sun.com/

For more reference sites, please see Appendix A.

Contact Us!

Your feedback on the BEA WebLogic Enterprise 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 Components documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Components 1.7 release.

If you have any questions about this version of BEA WebLogic Components, or if you have problems installing and running BEA WebLogic Components, 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.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`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. *Examples*: 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.

1 Getting Started

This topic includes the following sections:

- Overview
- Relevance
- Requirements
- Contents

Overview

Welcome to the WebLogic Components Technical Tour! This tour will navigate you through some of the features of BEA's WebLogic Components. WebLogic Components are "true" components: easy-to-use, reusable, customizable, standards-based, and extensible. This tour will show you details of how WebLogic Components can help you and the steps you need to take to leverage their powerful features.

Relevance

Consider an online product catalog. It is filled with items that have different descriptions, identifiers, and prices. Since 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's WebLogic Components!

Requirements

In order to fully understand the tour, it is necessary for the user to be familiar with a few concepts, including:

- Rational Rose and UML
- BEA WebLogic Components Smart Generator
- HTML
- Java Programming Language
- BEA WebLogic Application Server

Additionally, it is assumed that the user has a Java compiler and Rational Rose installed on his or her Windows 98/NT machine.

Contents

This tour presents a scenario in which WebLogic Components are leveraged to quickly and efficiently solve a common business problem. It starts with a design process in which you will use Rational Rose to extend two WebLogic Components: Item and ItemPriceCalculationPolicy. The tour will next show you how to implement the design. This includes exporting the model, generating WebLogic Components, and adding custom business and presentation logic. Finally, you will deploy the newly extended WebLogic Components as a solution to the stated business problem.

2 Scenario

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 My BuyBeans.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 My BuyBeans.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 My BuyBeans 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 My BuyBeans.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's WebLogic Components**.** BEA's WebLogic Components provided 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 Components Item to add unique properties for beanie hats. You could also leverage the pluggable WebLogic Components methods for pricing flexibility. You slowly being to realize that your assignment is actually quite possible!

3 Design

This topic includes the following sections:

- Overview of Design Considerations
- Prerequisites for Design Work
- Directions for Designing

Overview of Design Considerations

Having received instructions from senior management to extend the My BuyBeans.com site, the My BuyBeans 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 My BuyBeans product catalog must contain information on both of these attributes and both customers and My BuyBeans 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 Components developer, you decide to use WebLogic Components to develop a solution.

Looking at the Rational Rose model of the existing My BuyBeans solution, you quickly realize that all items in the My BuyBeans 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 *Modeling with eBSCs*, a document published by BEA.

Directions for Designing

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

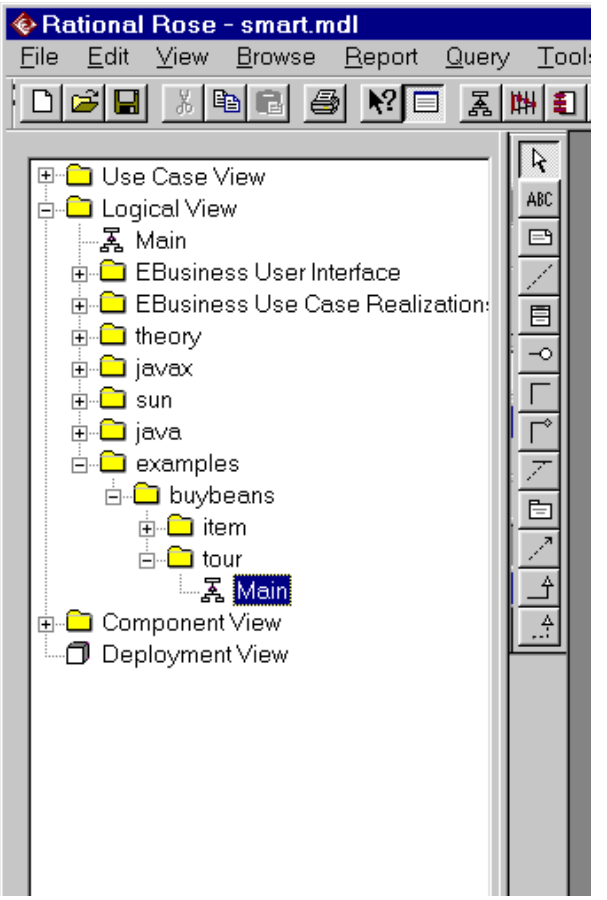
1. Navigate to the *model* directory found under the WebLogic Components installation directory.
2. Create a new subdirectory called *tour*. Be sure to name this new subdirectory "*tour*" exactly as given here (all lower-case), or else the rest of the technical demo will not work.
3. Create a copy of the Rational Rose model *BEA WebLogicAC.mdl* (found in the *BEA WebLogicAC* 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 Components installation directory.
7. Double click on *BEA WebLogicAC.mdl*. This will open Rational Rose.

Step 2. Create a new *My BuyBeans 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:



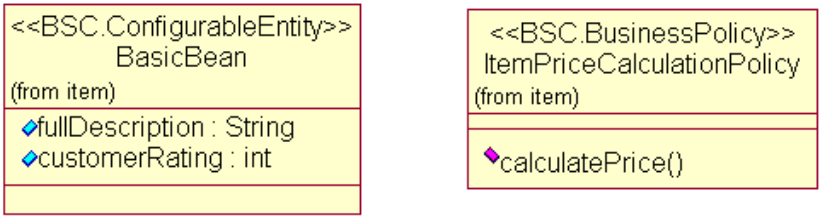
Step 4. Add a BasicBean Component to the class diagram.

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:



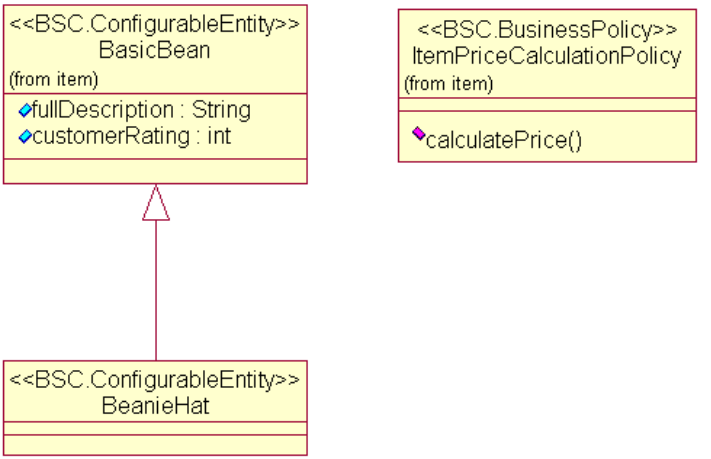
Step 6. Create a new BeanieHat Component that extends the BasicBean Component.

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:

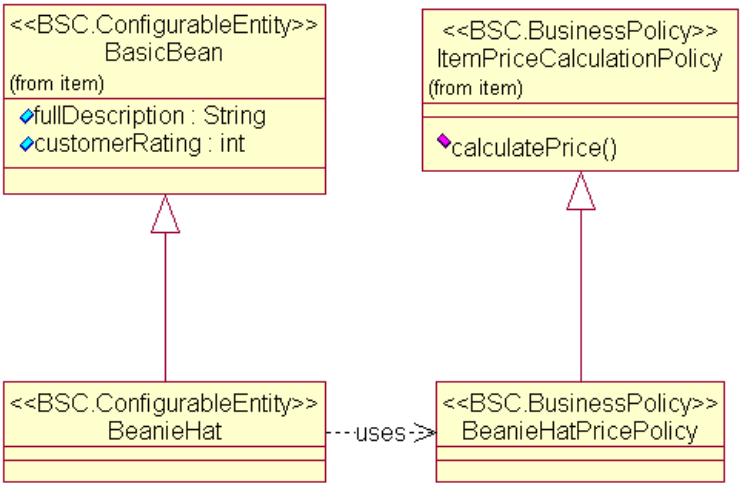
7. Place the cursor over the **BeanieHat** class. Click and drag the cursor over the **BasicBean** class and then release the mouse button.
8. Your Main class diagram window should now contain something like the following:



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:



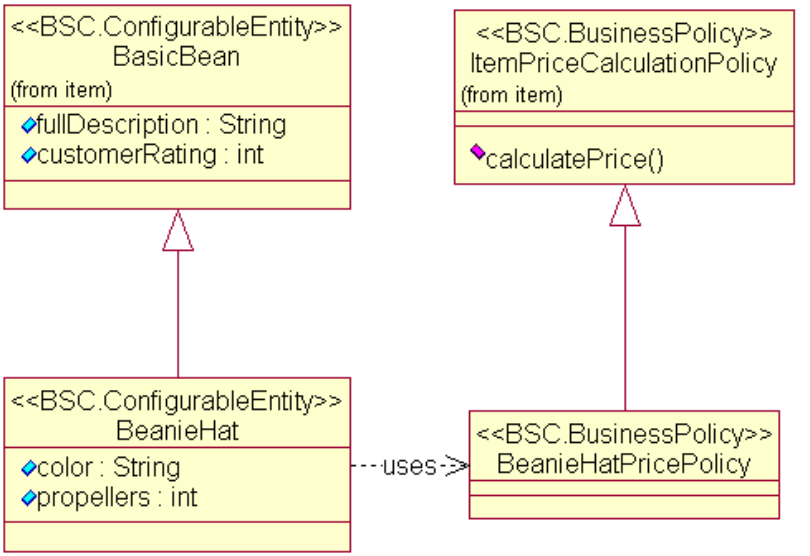
Step 8. Add attributes to the BeanieHat Component.

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:



4 Implement

This topic includes the following sections:

- Overview of Implementation
- Prerequisites for Implementing
- Directions for Implementing

Overview of Implementation

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

Since you are a knowledgeable WebLogic Components developer, you know that implementation with WebLogic Components involves exporting the Rational Rose model using **BEA's Rational Rose Plugin**, generating Enterprise Java Bean (EJB) implementations of your WebLogic Components using **BEA's 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 BEA WebLogic Components Rational Rose model.

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

Step 2. Export the model using BEA's Rational Rose Plug-In.

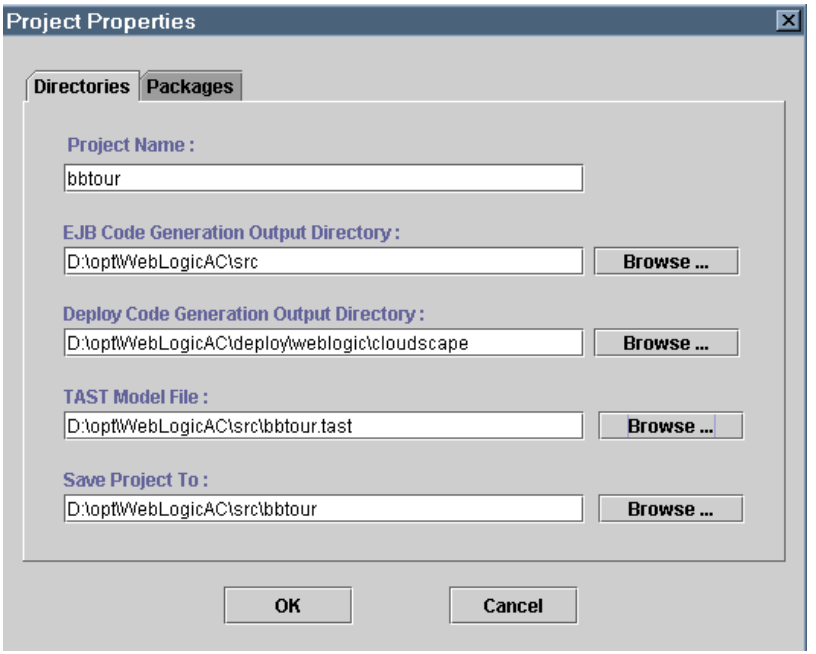
1. Click on the **WebLogicAC** menu item in the **Tools** menu.
2. Click on the **Export Model As** submenu item.
3. Navigate to the *src* directory found under the WebLogic Components installation directory.
4. Name the file *bbtour* and then click on the **Save** button.

5. Click the **OK** button in the **Success** dialog.

Step 3. Create a new Smart Generator project.

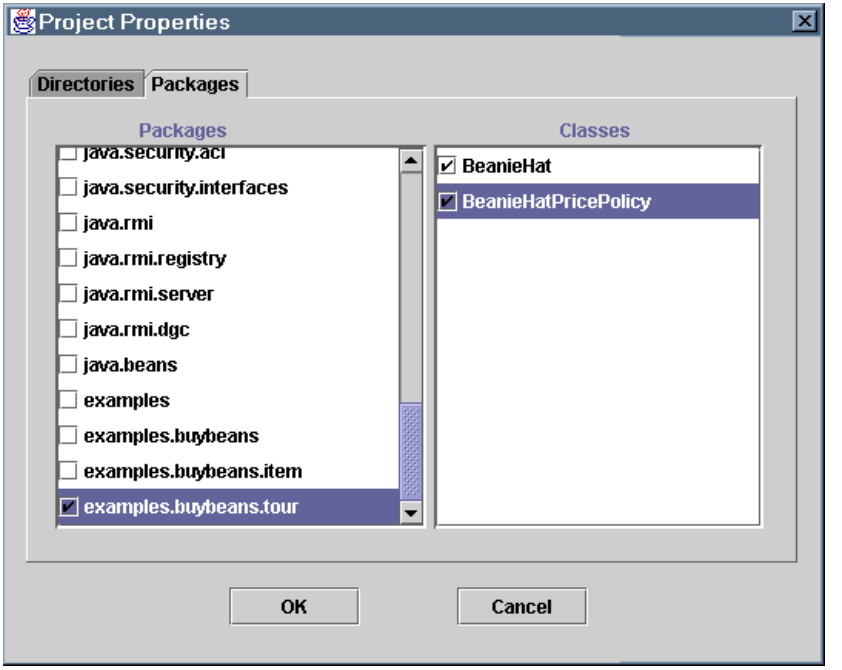
1. Start the **Smart Generator**. You can do this by first selecting the **WebLogicAC** menu item in the **Tools** menu and then clicking the **Smart Generator** submenu item.
2. Click on the **New Project** menu item in the **File** menu. This should open a **Project Properties** dialog.
3. Name the 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\WebLogicAC*:



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:



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 topic includes the following sections:

- Overview of Deployment
- Prerequisites for Deployment
- Directions for Deploying

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 Components**.** You once again assemble your team.

You explain to the team members that deploying BEA WebLogic Components 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 Components 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 documentation. You must also have successfully completed all steps of the Implement section of the tour.

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 Components 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 deployment descriptors for the BeanieHat Component and deploy the WebLogic Components.

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

Step 3. Configure the target Application Server (in this case the BEA WebLogic Application server).

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

```
WEBLOGICAC_HOME/deploy/weblogic/cloudscape/classes/examples/exten
ding/AlphaNumericSequencerExtensionImplDD.ser
```

4. Replace this line with the following lines (again, where `WEBLOGICAC_HOME` is the WebLogic Components installation directory):

```
WEBLOGICAC_HOME/deploy/weblogic/cloudscape/classes/examples/exten
ding/AlphaNumericSequencerExtensionImplDD.ser,\
WEBLOGICAC_HOME/deploy/weblogic/cloudscape/classes/examples/buybe
ans/tour/BeanieHatImplDD.ser
```

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 `deploy/weblogic/cloudscape/misc` directory located under the WebLogic Components installation 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. Go to **Start > Programs > WebLogic Application Solutions > Examples** and choose **My BuyBeans.com Server** 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 WebLogicAC installation directory.
5. Type `tour-loader` at the command line and hit Enter.

6 Results - An Enhanced Web Site

This topic includes the following sections:

- Overview of Results
- Prerequisites for Running the Web Site
- Directions for Viewing the Web Site

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 Components**. Without them, I'm not sure this would have been possible.*

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?