

# Custom Page Layout with IBM WebSphere Portlet Factory

7/30/09

© Copyright International Business Machines Corporation 2009. All rights reserved.

This article with the accompanying sample shows you how to use IBM® WebSphere® Portlet Factory Version 6.1.1 (hereafter called Portlet Factory or the Factory) to 6.1.2.2.

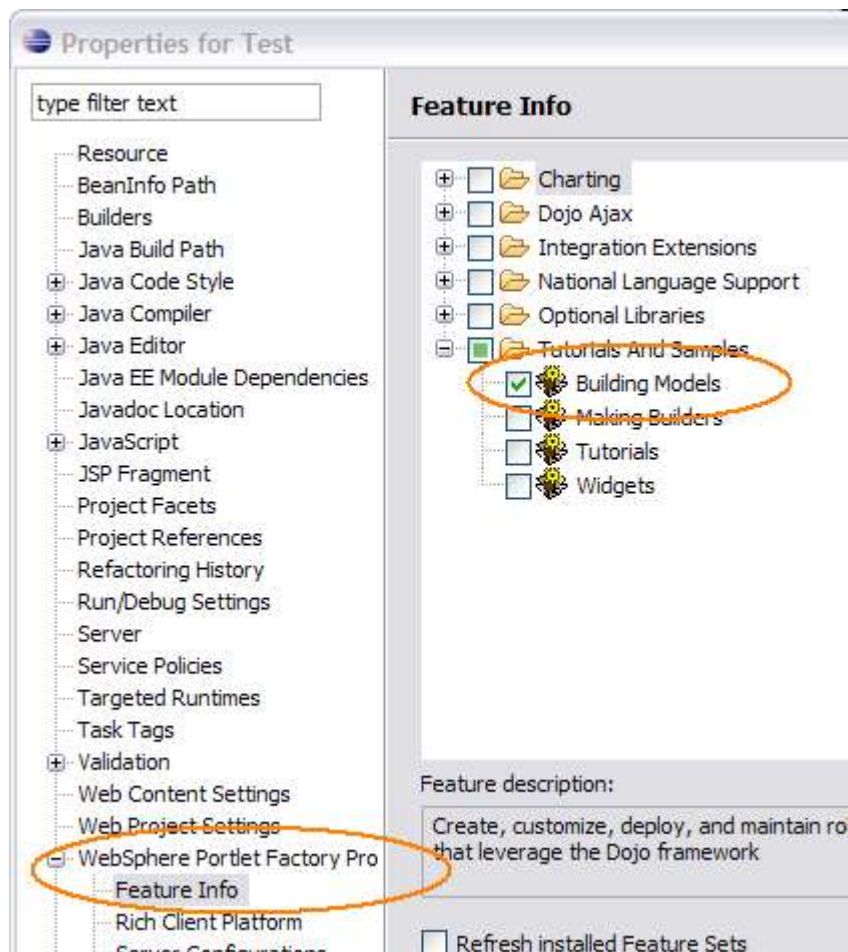
This article is one in a collection of articles and samples that illustrate techniques for developing with Portlet Factory. See the [Portlet Factory Product Documentation](#) page for a complete list of these. For an introduction to developing with Portlet Factory, you may want look at the introductory tutorials that are available both in the product help and on that web site.

## Prerequisites

You should have a basic familiarity with Portlet Factory and be able to create and run Portlet Factory models.

You should also have installed in your project the Feature Set “Building Models” from the “Tutorials and Samples” group. All of the samples are variations of the OrdersServiceConsumer model from that set, and they all depend on the OrdersServiceProvider model.

To install this Feature Set, open Project Properties and match the circled portions of the picture below.



## **Managing the layout of pages built with PageAutomation**

PageAutomation is the technology of the Factory that builds both JSP pages and Java code based on a schema or other data definition. The core access to this technology is the Data Page builder, which can also be accessed through high level builders such as View and Form or Input Form. Part of the power of PageAutomation is that it starts by creating an internal model of how the data will be laid out on the page, and then you can use assorted modifier builders, such as Data Field Modifier or Data Column Modifier, to change the internal representation before the actual work of creating the JSP and Java occurs. This document is focused only on the different techniques you can use to manage the layout of the JSP pages to get the eventual HTML result that is desired.

### **Different Techniques**

There are a number of different techniques for laying out the JSP, with different advantages and disadvantages to each.

1. Data Column Modifier for re-ordering (plus!)
2. Export of the HTML base page and hand-editing
3. Data Hierarchy Modifier for grouping and re-ordering
4. Forms Layout Builder
5. Manipulation of the HTML Template













## Data Column Modifier for Re-ordering (Plus!)

For a page with a table on it, you can use the Data Column Modifier to re-order the fields, and to make some other layout changes to the table. You can add a counter column, a delete column (of one of three styles), and you can add an extra column. Any extra column will initially be blank, but you could use a Data Field Modifier, or a Text, Link, or Button builder, for instance, to add additional functionality in that column.

Here is some sample inputs, followed by the display that this creates. Note that the columns have been reordered from the default, and a counter and an extra column have been added (though nothing was yet added to the extra column).

Add Counter Column	 <input checked="" type="checkbox"/>
Add Columns	 <input type="text" value="Extra"/>
Manage Columns	 <input checked="" type="checkbox"/>

Tip: You can reorder the fields as they appear on the page by dragging and dropping the rows in the table.

	Column Name	Status	Column Heading	Column
	.Row_Counter	Counter Column	##	
	.ORDER_ID	Do Not Change	[Do Not Change]	
	.DATE_ORDERED	Do Not Change	[Do Not Change]	
	.QUANTITY	Do Not Change	[Do Not Change]	
	.AMOUNT	Do Not Change	[Do Not Change]	
	.BILLING	Do Not Change	[Do Not Change]	
	.STATUS	Do Not Change	[Do Not Change]	
	.DATE_SHIPPED	Do Not Change	[Do Not Change]	
	.SHIPPED	Hide	[Do Not Change]	
	.STATE	Do Not Change	[Do Not Change]	
	.Extra	Do Not Change	[Do Not Change]	
	...		[Blank]	

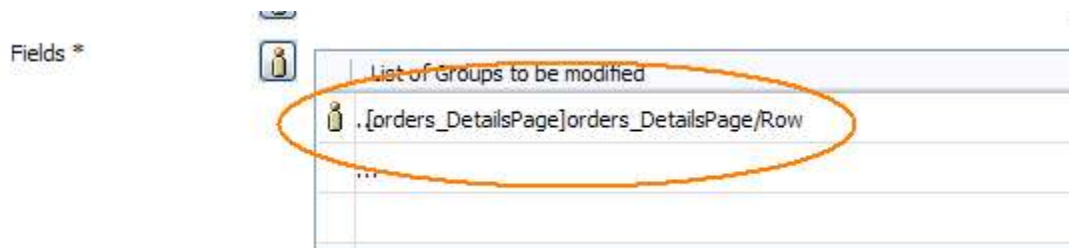
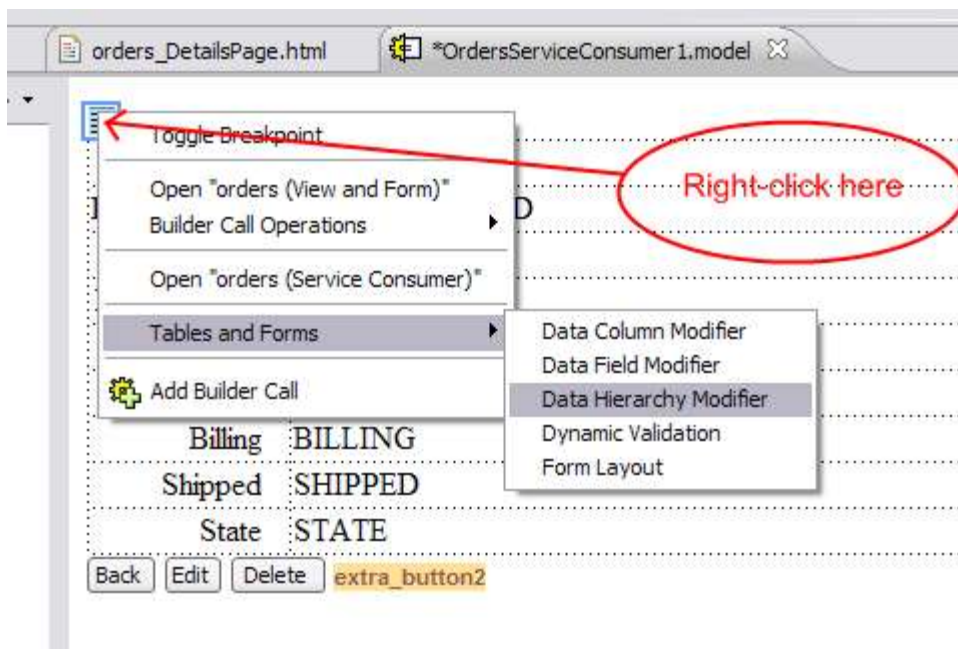
##	Order ID	Date Ordered	Quantity	Amount	Billing	Status	Date Shipped	State	Extra
1	<a href="#">008731</a>	2002-06-16	76	790.98	PO	Shipped	2002-06-22	New Hampshire	
2	<a href="#">000715</a>	2000-02-12	4	457.72	Credit	Out of Stock	-	Massachusetts	
3	<a href="#">004912</a>	2005-08-03	27	163.18	COD	Shipped	2005-08-04	Missouri	
4	<a href="#">005978</a>	2001-05-01	38	287.51	Credit	In Process		Vermont	
5	<a href="#">001629</a>	2006-11-16	100	512.39	PO	Shipped	2006-12-15	California	

1 [2](#) [3](#) [>](#) [>>](#)

## Tip: Creating Modifier Builders

Many of the elements on a page can be modified using different modifier builders, and some builders are appropriate for some page elements, and some are appropriate for others. The best way to look at the appropriate modifier builders for an element is to right click that element on the page (or in the WebApp tree) and choose from among the builders offered in the categorized section.

For PageAutomation elements, the groups are represented in the Design View by the small blue square at the top of the section. If you right click on one of these, the only modifier builders available are in the “Tables and Forms” category, and you can see all the appropriate modifiers there to choose from. When you create a new modifier builder this way, then the appropriate field to select the group you wanted to modify will already be filled in.



## Export of the HTML base page and hand-editing

One of the features of the PageAutomation system is that you can completely take over the creation of the HTML which makes the framework for the JSP page that PageAutomation will create. This has both benefits and limitations. The benefit is that you can control precisely how the resulting page will look, which still using PageAutomation to generate the Java code that manages the flow of data (including validation). The disadvantage to this approach is that it means hand-edited work for each page on which you use it, and that page has now lost the flexibility to modify itself if there is a change in the schema, query, or service upon which the page is based.

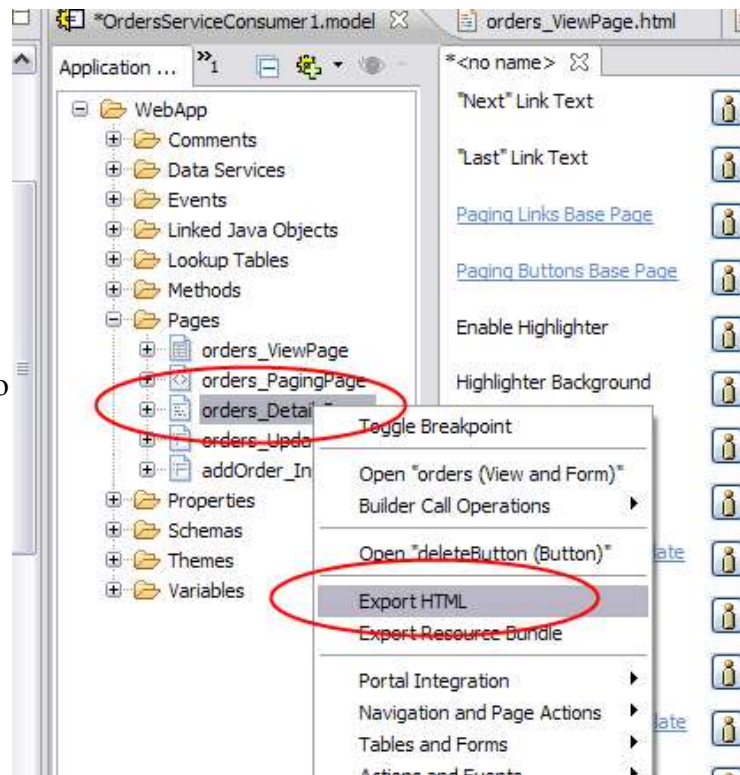
There are several steps you have to go through to manage this process.

1. Export the page (or part of the page – only available in version 6.1.5 or later)
2. Hand-edit the page, retaining the important named elements
3. Cause your new hand-edited version to be used instead of a generated page

The way to accomplish the final step is different depending on what builders you used to create the page in the first place, and they are different if you are using a version of the Factory before or after version 6.1.5. Each of these instructions is below, so be sure to choose the one that applies to your case.

### Export the page

To export the page (or a portion of the page) right-click the page element in the WebApp Tree and choose “Export HTML.” This will bring up a dialog to choose the filename, and then the file will be exported.




## Edit the page

This file is the HTML framework that the PageAutomation system is creating so far, for your page, using the information from the existing Data Page and modifier builders, including the HTML Template currently selected. The only difference from the JSP page which is generated is that the individual control builders (such as the Text builder for Display fields or builders like Text Input for Data Entry fields) have not yet been applied.

If you are using a version of the Factory before 6.1.5, you will see many HTMLWRAPPER elements on the page. This is a special tag that is used by the Factory as a placeholder, but it does not create any tag in the resulting JSP, and therefore does not appear in the resulting HTML when the application is running. You can safely delete any of these that do NOT include one of the significant names.

## Significant Names

The way that PageAutomation will use this page is it will look for named elements on the page, and match up the Groups and Fields that it creates from the data definition to those named elements. There are three types of significant names.

1. A Field name: In the example below, see element with the *name="ORDER\_ID"* attribute circled. This element will be used to display the contents of the field ORDER\_ID. (This name comes directly from the schema.)
2. A Field name with "Label" appended: In the example below, see element with the *name="ORDER\_IDLabel"* attribute circled. This element will be used to display the label of the field ORDER\_ID, but only if the selection in the Data Page builder "Translate HTML-based Labels" is checked, which is in the "Label Translation Settings" section of the Data Page Builder.

The screenshot shows the 'Label Translation Settings' section. It includes a title bar, a subtitle 'Settings for managing the translation of labels in a way that can b', and four rows of controls: 'Localized Resource' with an 'Add a new Localized Resource' button, 'Create Resource Bundle' with a 'Create Resource Bundle' button, 'Resource Bundle Name' with an input field, and 'Translate HTML-based Labels' with a checked checkbox. The checkbox and its label are circled in orange.
3. The name of a Group or Table element with "\_Repeat" appended to it. This element is significant only if the corresponding element in the schema is a repeating element, although it will usually appear whether or not the schema element repeats. The significance of this HTML element is that it is where the Repeated Region builder will be placed on the page. If you want to make a custom layout for a table, perhaps with two or more TR tags for every record, then you will have to make sure that the "\_Repeat" element surrounds all of the pieces in the JSP that you want to appear for every row of the data.

```

    <HTMLWRAPPER name="PageContentsContainer">
<DIV name="DisplayGroupWrapper" class="vf_fields">
    <HTMLWRAPPER name="RowLabelContainer"></HTMLWRAPPER>

    <TABLE name="GroupContainer" class="displayPageTable">
<HTMLWRAPPER name="DisplayGroup">
    <!-- Begin: Data display fields -->
    <HTMLWRAPPER name="Row_Repeat">
<TR name="DisplayField" class="DisplayFieldRow">
    <TD class="labelCell" align="right" valign="top" nowrap="">
        <LABEL for="ORDER_ID_field"><SPAN name="ORDER_IDlabel" class=
        </TD>
    <TD width="100%" class="outputDataCell" valign="top" nowrap="">
    <SPAN name="ORDER_ID" class="outputData" value="Field Value" id="
    </TD>
    </TR>

<TR name="DisplayField" class="DisplayFieldRow">
    <TD class="labelCell" align="right" valign="top" nowrap="">
        <LABEL for="DATE_ORDERED_field"><SPAN name="DATE_ORDEREDlabel
        </TD>
    <TD width="100%" class="outputDataCell" valign="top" nowrap="">
    <SPAN name="DATE_ORDERED" class="outputData" value="Field Value" :
    </TD>
    </TR>

```

You then can edit the page with any HTML editor, laying it out as you like, as long as you preserve the significant named elements. For example, the sample lays out the view and details pages to look like the following. These pages appear in the files

WebContent/samples/custom\_page/orders\_ViewPage.html  
 WebContent/samples/custom\_page/orders\_DetailsPage.html

<a href="#">008731</a> Shipped	Ordered: 2002-06-16 Shipped: (true) 2002-06-22
76 units	\$790.98 (PO) New Hampshire
<a href="#">000715</a> Out of Stock	Ordered: 2000-02-12 Shipped: (false) -
4 units	\$457.72 (Credit) Massachusetts
<a href="#">004912</a> Shipped	Ordered: 2005-08-03 Shipped: (true) 2005-08-04
27 units	\$163.18 (COD) Missouri
<a href="#">005978</a> In Process	Ordered: 2001-05-01 Shipped: (false)
38 units	\$287.51 (Credit) Vermont
<a href="#">001629</a> Shipped	Ordered: 2006-11-16 Shipped: (true) 2006-12-15
100 units	\$512.39 (PO) California

1

[Create Order](#)

## Order ID 000715

**Date Ordered** 2000-02-12      **Date Shipped** -  
**Quantity** 4    **Amount** 457.72      **Shipped** false  
**Status** Out of Stock    **Billing** Credit      **State** Massachusetts

[Back](#)   [Edit](#)   [Delete](#)

### Cause your new hand-edited version to be used instead of a generated page

This step is is different depending on whether or not you are using themes, and whether or not you have created the page using a simple Page or Imported Page builder and then applied Data Page to it, or if you have used View and Form to consolidate these steps.

#### Page or Imported Page and Data Page Builder

If this is the situation, you can just change the source of the Page or Imported Page builder to be the new HTML that you have hand-edited. If it is a Page Builder, you can either convert the builder to be Imported Page and then point it at your HTML page you have edited, or else just paste the contents into the builder. The advantage of keeping it in the separate file is that it is easier to maintain, but the advantage of having it all in the

model, without a separate file, is that it is easier to transport the model. For lightweight, sample models, you might prefer to keep it all in the model.

### View and Form

If the page was created by the View and Form, then you will have to change the appropriate “Base Page” input to point to your new page. This becomes an issue if you are using themes, and you are once again faced with two choices: You can turn off the theme for this builder call, or you can override the theme value. If you turn off the theme, then you will be turning it off for all of the View and Form builder, which might not be what you want if you are still using the theme for the other pages that it generates.

If you want to keep using the theme for the other pages, then you override this value using a theme builder. The theme builder will present for you all the values that it is managing, and allow you to override specific ones. The one restriction is that your theme builder **must appear before the View and Form builder in the Builder Call List**. Otherwise it will not be able to modify that input.

#	Name	Type
1	Model Comment	Comment
2	Copyright	Comment
3	overrides	Theme
4	Service Consumer	Comment
5	orders	Service Consumer
6	List, Details, and U...	Comment
7	orders	View and Form
8	Delete	Comment
9	deleteOrder	Action List
10	deleteButton	Button
11	Add Order Page	Comment
12	addOrder	Input Form
13	newOrder	Action List
14	addOrder	Button


Theme builder is before V&F

Below is part of the Theme Builder from the sample. It also has the Details Base Page overridden.

## Theme

Use this Builder to change the Theme that a Model uses.




### ▶ Properties

Name	Overrides
<a href="#">Theme File</a>	 /WEB-INF/factory/themes/blue.uitheme
Override Theme Properties	 <input checked="" type="checkbox"/>
<input type="button" value="Create Theme file"/>	

### ▶ Data Page

#### ▼ View and Form

Theme values that affect the View and Form Builder.

<a href="#">View Base Page</a>	 /sample/pages/orders_ViewPage.html
<a href="#">View Page HTML Template</a>	
<a href="#">View Page Style Sheet</a>	

## Data Hierarchy Modifier for Grouping and Re-ordering

The Data Hierarchy Modifier can be used to make logical groups with the fields, including a header for each group. As with the Data Column Modifier builder, you can also hide fields and change their labels in this builder. To create groups, start by entering the new group names in the “New Group Names” input, separating them with commas. When you navigate off of this input, start and end tags for your new groups will be inserted into the table below. Now you can use drag and drop to put the fields in the groups. Here is a sample of the builder inputs, followed by a display of how the details page would look with these inputs.

New Group Names

Name	Status	Label
..*** Order	Do Not Change	Order
.ORDER_ID	Do Not Change	Order ID
.DATE_ORDERED	Do Not Change	Date Ordered
.QUANTITY	Do Not Change	Quantity
.AMOUNT	Do Not Change	Amount
.BILLING	Do Not Change	Billing
..**** END Order	Do Not Change	
..*** Shipped	Do Not Change	Shipped
.STATUS	Do Not Change	Status
.DATE_SHIPPED	Do Not Change	Date Shipped
.SHIPPED	Do Not Change	Shipped
.STATE	Do Not Change	State
..**** END Shipped	Do Not Change	
...		

**Order**

Order ID 005978

Date Ordered 2001-05-01

Quantity \* 38

Amount \* 287.51

Billing Credit ▾

---

**Shipped**

Status In Process

Date Shipped

Shipped

State Vermont ▾

Cancel Submit

In the one builder, this has been applied to both addOrder\_InputPage and orders\_UpdatePage. This is possible because both of these pages are based on the same schema.

The resulting display is shown on the right.

## Forms Layout Builder

This builder is useful when you have a simple details or data entry page which you want to display in two or three columns. Add this builder to the group element, specify how many columns you want, and you can either instruct the builder to break up the fields automatically or you can manage exactly which columns go in which fields by dragging and dropping the fields into the different groups. As a convenience, you can also hide fields and change their labels in this builder.

Here is a sample of the builder input with the fields arranged by dragging and dropping them, after choosing “Explicit” field Management. Following that is a sample of the output you might expect.

Number of columns \*

Field Management \*

Name	Status	Label
*** Column1	Do Not Change	Column1
.ORDER_ID	Do Not Change	Order ID
.DATE_ORDERED	Do Not Change	Date Ordered
.QUANTITY	Do Not Change	Quantity
.AMOUNT	Do Not Change	Amount
**** END Column1	Do Not Change	
*** Column2	Do Not Change	Column2
.STATUS	Do Not Change	Status
.BILLING	Hide	Billing
.SHIPPED	Do Not Change	Shipped
.DATE_SHIPPED	Do Not Change	Date Shipped
.STATE	Do Not Change	State
**** END Column2	Do Not Change	
...		

<b>Order ID</b> 004912	<b>Status</b> Shipped
<b>Date Ordered</b> 2005-08-03	<b>Shipped</b> true
<b>Quantity</b> 27	<b>Date Shipped</b> 2005-08-04
<b>Amount</b> 163.18	<b>State</b> Missouri

[Back](#) [Edit](#) [Delete](#)

## **Manipulation of the HTML Template**

This is an advanced subject and is less about laying out the data on the page as it is about the specific details of how the pieces of the automatically-generated layout are created, individually. Therefore, this subject is not covered in this paper. A more complete look at HTML Templates can be found at

[http://www.ibm.com/developerworks/websphere/library/techarticles/0607\\_odonnell/0607\\_odonnell.html](http://www.ibm.com/developerworks/websphere/library/techarticles/0607_odonnell/0607_odonnell.html)

# Sample

Table 1. Sample package contents

Filename and location	Description
WebContent/WEB-INF/models/samples/CustomPage.model	Copy of the OrderServiceConsumer model from the Feature Set “Building Models,” but with the changes discussed in this document
WebContent/samples/custom_page/orders_DetailsPage.html	Hand-edited page used for the details page
WebContent/samples/custom_page/orders_ViewPage.html	Hand-edited page used for the view page
WebContent/samples/custom_page/gridtable_blue.css	Modified version of the css file that is part of the blue theme

## Instructions for running the sample

To run the sample application:

1. Download the sample ZIP file and import it into a project using the File, Import, WebSphere Portlet Factory Archive command. You will also need to have the Feature Set “Building Models” from the “Tutorials and Samples” group. (See Prerequisites section.)
2. Open the model and run it.

## Resources

WebSphere Portlet Factory product documentation

<http://www.ibm.com/developerworks/websphere/zones/portal/portletfactory/proddoc.html>

WebSphere Portlet Factory support

<http://www.ibm.com/software/genservers/portletfactory/support/>

developerWorks forums

[http://www.ibm.com/developerworks/forums/wsdd\\_forums.jsp](http://www.ibm.com/developerworks/forums/wsdd_forums.jsp)

## Trademarks

1. DB2, IBM, Lotus, Tivoli, Rational, and WebSphere are trademarks or registered trademarks of IBM Corporation in the United States, other countries, or both.
2. Windows and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.
3. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
4. Other company, product, and service names may be trademarks or service marks of others.