IBM DB2 Alphablox

Getting Started

Version 84

GC18-9607-01

IBM DB2 Alphablox

Getting Started

Version 84

GC18-9607-01

Note:
Before using this information and the product it supports, read the information in "Notices" on page 31.

Second Edition (March 2006)

This edition applies to version 8, release 4, of IBM DB2 Alphablox for Linux, UNIX and Windows (product number 5724-L14) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Tutorial: Building your first application 1
Defining your application 1
Accessing data. 2
Creating your application home page 2
Setting a default home page 3
Creating your first analytic view. 4
Structure of your first analytic view page. 7
Creating your second analytic view. 9
Summary 10

Chapter 2. Tutorial: Building your first portlet with Blox components 11
Installing the sample portlets 11
Running the sample portlets. 12
Examining the structure of a portlet JSP page with a Blox component 12
Creating your own portlet JSP page with Blox components 14
Creating a portlet project to use Blox components 16
Configuring a portlet project using Rational Application Developer. 17
Next steps. 18
Portlet development tips 18

Chapter 3. Tutorial: Building applications using Rational Developer tools. 21
Preparing your development environment 21
Installing the DB2 Alphablox Toolkit 21
Installing DB2 Alphablox in WebSphere integrated test environments 22
Creating a WebSphere server instance 23
Creating WebSphere 5.1 server substitution variables 23
Configuring WebSphere 5.1 server instances . . . 24
Adding the guest user to the to DB2 Alphablox Administrators group 24
Creating DB2 Alphablox applications. 25
Creating JSP files with DB2 Alphablox content. . . 25
Accessing DB2 OLAP Server and Essbase data sources 26

Chapter 4. Tutorial: Building a DB2 Alphablox cube using DB2 Cube Views 27
Defining a DB2 relational data source 27
Defining an Alphablox Cube Server Adaptor data source 28
Defining a DB2 Alphablox cube 28
Starting your DB2 Alphablox cube. 29

Notices 31
Trademarks 32

Index 35

Chapter 1. Tutorial: Building your first application

This tutorial teaches you important basics about Blox components and provides steps that you can follow to quickly build a DB2® Alphablox application.

To build DB2 Alphablox applications, you will be using JavaServer Pages (JSP) technology, but knowledge of JSP is not required for this tutorial. This tutorial explains the aspects of JSP that you need to know to begin building analytic applications. To learn more about JSP technology, there are many excellent books available.

Prerequisites

The steps in this tutorial assume that you are using the DB2 Alphablox running on WebSphere® Application Server. If you are using BEA WebLogic or Apache Tomcat as your application server, you might need to modify a few of the following steps to fit particular differences in your server.

Also, before beginning this tutorial, you need to install a DB2 Alphablox cube (qcc_2003) and its underlying relational data source (qcc2003-rdb). The sample database files and instructions (in readme.txt files) on configuring the DB2 Alphablox cube and the relational database required for this tutorial can be found in the sampledata/qcc/acs directory of the DB2 Alphablox Installation CD.

Defining your application

To create an application using the J2EE development approach, you need to create a directory structure with a WEB-INF directory that contains an application descriptor files (web.xml). The simplest way to create this structure in DB2 Alphablox is to create a new application using the Application page within the DB2 Alphablox Admin Pages.

To create your application and folder:

1. Create a new application, called MyApp, in DB2 Alphablox by following the steps described in the Application Definitions section of the *Administrator's Guide*.
   - Enter `MyApp` in the **Name** field .
   - Enter `My App` (with a space) in the **Display Name** field, which defines the label for the application that appears in list on the Applications page.
2. Click the top **Applications** tab at the top left of the page. A list of the available applications opens.
3. Click on the name of your newly created application (My App) in the list of applications. Because you have not create any files yet, the directory of files is empty.
4. Navigate to the new application folder on your application server (for WebSphere, the folder is located in the WebSphere installedApps directory). Notice that the application includes a WEB-INF directory with a web.xml file and the tlds directory. The web.xml file defines application information. The tlds directory includes the Blox tag library descriptor file (blox.tld), which defines the Blox tags that you will be use to create analytic views in this tutorial. The directory contains other tag library descriptor (TLD) files for other

Blox tag libraries (not used in this tutorial), including bloxform.tld, bloxlogic.tld, bloxreport.tld, and bloxui.tld.

You have created a DB2 Alphablox application framework that can now have analytic views added to it.

Accessing data

For this tutorial, you will be using the DB2 Alphablox cube data source you created before beginning this tutorial. Your relational database and the Alphablox Cube Server must be running. Also, the DB2 Alphablox cube being used must be running.

To verify that your required data sources are available and operating:

1. Open your browser to the DB2 Alphablox Administration Pages.
2. Click the **Administration** tab.
3. Click the **Data Sources**.
4. On the Data Sources page, find the qcc2003-rdb data source in the menu on the left side of the page. If it's not there, you need to review the steps in the readme.txt file in the sampledata/qcc/acs directory of your DB2 Alphablox Installation CD.
5. Click on qcc2003-rdb in the list of available data sources and then click the **Test Selected Data Source** button. If the database is running and your data source definition is configured properly, a success message appears. If an error message appears, you need to check the data source definition (selecting the data source and clicking the **Edit** button) or whether the database is properly configured with a user who has access rights to the database.
6. Select the qcc2003-acs data source from the list of data sources and then click the **Test Selected Data Source** button. If the Alphablox Cube Server data source is correctly configured, a success message appears. If an error message appear, review the data source definition.

After successfully testing the data sources, you are ready to begin learning how to build a couple of simple DB2 Alphablox analytic views.

Creating your application home page

For this task, you can create a simple home page using basic web skills.

For the purposes of this tutorial, you only need to create a very simple web application structure. On the home page, you need to create two links for accessing the two analytic views you create in this tutorial.

To create a simple home page:

1. Using a text editor or your favorite development tool, name it `index.html`. For the purposes of this tutorial, you can keep it very simple.
2. Edit the file to specify an application title (for example, "My DB2 Alphablox Application") and two links to the two analytic views you will create.

   Copy and paste, or type, the following code into your index.html file.

```
<html>
<head>
<title>My DB2 Alphablox Application</title>
</head>
```

```
<body>
<h2>My DB2 Alphablox Application</h2>
<p>
<a href="PresentBloxView.jsp">Simple PresentBlox View</a>
</p>
<p>
<a href="ChartBloxView.jsp">Customized ChartBlox View</a>
</p>
</body>
</html>
```

3. Save this file to the `MyApp` directory that you created when you defined the application.
4. Open your web browser to the DB2 Alphablox Home page. By default, the browser displays the **Applications** tab.
5. Click the **My App** link to verify that the file is included in the directory listing. Click the **index.html** link to view your home page. If DB2 Alphablox is installed on an IBM® WebSphere Application Server or a BEA WebLogic server, directory listings must be enabled before you can perform this step. For DB2 Alphablox installations on Apache Tomcat, no additional steps should be required. If you see an HTTP 403 error (directory listings are not allowed by default), you can either modify the server to allow directory listings or just append the file name myapp.html to the end of the URL (for example, http://localhost:9080/MyApp/myapp.html. The new page should be display when using the fully-qualified link address.

You now have a home page with links to the two analytic views you will create next.

Setting a default home page

In previous steps, you accessed the home page directly by clicking on the file name link in the MyApp application directory, or by specifying the home page file, index.html, in the URL. In this step, you specify the default home page for the application.

To set default application home page:

1. Open your browser to the DB2 Alphablox home page.
2. Click the **Administration** tab.
3. Click the **Applications** link.
4. Select **MyApp** in the applications list and click the **Edit** button below the list
5. On the Edit Application page for MyApp, enter `index.html` in the **Home URL** field.
6. Click the **Save** button to save the change you made.
7. Click the main Applications tab, the top left folder tab, to return to the Applications page.
8. Click the **My App** application name. The application now opens directly to the defined home page.

Now that the home page is defined, the server will know where to start the application, and will direct the application to open at this page, even if the user fails to specify the home page in their URL. Thus, if a user enters http://*yourServerName*/MyApp/ into the browser address bar, the application will automatically open to the defined home page, index.html.

In the upcoming topics, you will begin to create DB2 Alphablox analytic views using simple Blox tags generated using the DB2 Alphablox Query Builder application.

Creating your first analytic view

In this task, you will create a DB2 Alphablox analytic view displaying a grid and a chart combined into the PresentBlox component.

Prerequisite: Read the introductory sections of the *Developer's Guide* to learn about DB2 Alphablox and its components. This will give you a basic understanding of DB2 Alphablox functionality and how the Blox custom tag libraries can be used in building analytic applications using JSP technology.

For the first analytic view you will create, you will use the DB2 Alphablox Query Builder application to generate a JSP custom tag defined for a PresentBlox component, the most popular Blox component used for presenting data. A PresentBlox component can include grid and chart views, a data layout panel, toolbars, and other nested components. The grid and chart views represent synchronized alternative views of the same data.

To create your first analytic view using Query Builder:

1. Open a browser to the DB2 Alphablox Admin Pages. By default, the Applications tab is visible.
2. Click on DB2 Alphablox Query Builder to open the application.
3. Click the **Connection Settings** button. The Database Connection window appears.
4. Select the qcc2003-acs data source and then press the **Connect** button. The window closes and you should see "Connected" appear in the Database Status section of the Query Builder page.

   Note: You are not required to fill in the values for the **Schema**, **Catalog**, **Username**, and **Password** fields since they are retrieved automatically from the data source definition for the data source you select. Also, do not select the **Execute Default Query** check box.
5. Find the Query section on the page, below the Status section. Note the following features:
   - For this tutorial, you are using a DB2 Alphablox cube (qcc2003-acs). Just beneath the Query section header, you should see a listing of cubes that are available (currently running). The qcc2003-acs cube should be listed.
   - A default query string associated with this data source appears in the text window. Since the data source is a DB2 Alphablox cube, the first cube, listed alphabetically, is added into the query statement (for example, `"select from [datasourceName]"`).

     Note: When the connection pooling feature in the DB2 Alphablox Cube Server is enabled, you will not be able to use the **Get Default Query** button to get a default query for a DB2 Alphablox cube.
   -
   - When data appears in the PresentBlox at the bottom of the page, use the standard user interface to swap axes, drill up or down, move dimensions between axes, and so forth.

- To view the results of the query, click the Execute Query button. The result set appears in the PresentBlox at the bottom of the page.

6. Type `qcc2003-acs` between the square brackets of the default query (for example, `select from [qcc2003-acs]`) and then press the **Execute Query** button.

   The PresentBlox section should display a default view based on the query.

7. Use the PresentBlox user interface to create a new view.

   For example, you can drag the Products dimension to the Row axis in the Data Layout panel on the left. Then, you can drag the Time (Calendar) dimension to the Column axis. You might then want to drag the Measures dimension into the Page axis, resulting in a Measures menu appearing in the Page panel, which displays page filters, above the display area. Pick two other dimensions and add them to the Page axis so that three page filters appear above the display. If you are new to using the DB2 Alphablox user interface, select **Help** from the PresentBlox toolbar to learn more about how to use the interface.

8. After you are finished creating your new view, note the query statement that appears in the Query section. If the query is not being dynamically generated (when the **Automatically Update Queries** option is unchecked), you can click the **Get Current Query** button to get the current query. The text box displays the query statement required to develop the current data view. You can copy and paste query statements from the **Query** field to be used in developing DB2 Alphablox applications.

9. Click the **Generate Blox Tag** button. A dialog opens displaying the tag (and nested tags) to reproduce the PresentBlox layout and result set.

10. Open a new file in your text editor and add the following text to the file and then save the file as PresentBloxView.jsp. The next topic will explain many of the important elements of the code on this page.

```
<%@ taglib uri="bloxtld" prefix="blox" %>

//Copy and paste the tag information from the generated Blox tag
//immediately below this line, removing any line breaks from
//query statement. You can also remove this comment after adding
//the code from DB2 Alphablox Query Builder.

<html>
<head>
<blox:header/>
</head>
<body>
<h2>Simple PresentBlox View</h2>
<p>

<blox:display bloxRef="MyPresentBloxView"/>

</p>
</body>
</html>
```

11. Copy the saved file (PresentBloxView.jsp) to the MyApp directory on your server and test your new analytic view by clicking on the link to this page, Simple PresentBlox View, from your MyApp home page. Alternatively, you can enter the URL directly into a browser address (for example, http://localhost:9080/MyApp/PresentBloxView.jsp). The view that you see should look similar to the PresentBlox you manipulated in the DB2 Alphablox Query Builder.

A code example of a complete version of the Simple PresentBlox View page (PresentBloxView.jsp) is shown below:

```
<%@ taglib uri="bloxtld" prefix="blox" %>

<blox:present
    id="queryBuilder4_present"
    height="500"
    visible="false"
    width="100%">
    <blox:grid/>
    <blox:chart/>
    <blox:page/>
    <blox:data
        dataSourceName="qcc2003-acs"
        onErrorClearResultSet="true"
        query=" SELECT DISTINCT({[qcc_2003].[Time (Calendar)].[All Time
(Calendar)], [qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2000],
[qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2001],
[qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2002],
[qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2003]} )
ON AXIS(0), DISTINCT( {[qcc_2003].[Products].[All Products],
[qcc_2003].[Products].[All Products].[100  Truffles],
[qcc_2003].[Products].[All Products].[200  Chocolate Blocks],
[qcc_2003].[Products].[All Products].[300  Chocolate Nuts],
[qcc_2003].[Products].[All Products].[400  Specialties]} )
ON AXIS(1) FROM [qcc_2003] WHERE ([qcc_2003].[Measures].[Sales],
[qcc_2003].[Time (Fiscal)].[All Time (Fiscal)],
[qcc_2003].[Date Opened].[All Date Opened],
[qcc_2003].[Has Nuts].[All Has Nuts],
[qcc_2003].[Chocolate Type].[All Chocolate Type],
[qcc_2003].[Ounces Per Package].[All Ounces Per Package],
[qcc_2003].[Pieces Per Package].[All Pieces Per Package],
[qcc_2003].[Date Introduced].[All Date Introduced],
[qcc_2003].[Seasonal].[All Seasonal],
[qcc_2003].[Scenario].[Actual],
[qcc_2003].[Locations].[All Locations])"
        selectableSlicerDimensions="[qcc_2003].[Measures],
[qcc_2003].[Locations],[qcc_2003].[Chocolate Type]"
        useAliases="true"/>
    <blox:toolbar/>
    <blox:dataLayout/>
    <bloxui:calculationEditor />
</blox:present>

<html>
<head>
<blox:header/>
</head>
<body>
<h2>Simple PresentBlox View</h2>

<blox:display bloxRef="queryBuilder4_present"/>

</p>
</body>
</html>
```

In the code example above, the query statement has line breaks in it for readability. To run this page properly requires the line breaks in the attribute values (like the `query` and `selectableSlicerDimensions` attributes above) to be removed. In the code generated by the DB2 Alphablox Query Builder, many tags that are included are not necessary. Seeing all of the tags here, though, should give you insight into how the Blox tags work as parent and nested tags. Also, you can see the use of the tag attributes that are added as a result of the manipulations you make in Query

Builder's PresentBlox. To learn more about the details of using Blox tags, tag attributes, and how to develop DB2 Alphablox applications, you can refer to the DB2 Alphablox Information Center.

Structure of your first analytic view page

This topic summarizes the structure of the first analytic view (PresentBloxView.jsp) that you created in the previous topic.

Below is a brief overview of some of the highlights from the page created in the previous topic. For a more complete understanding of the code, you will have to read topics of interest in the DB2 Alphablox Information Center.

The PresentBloxView.jsp file begins with the following line:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
```

This line is a JSP taglib directive that informs the server that you intend to use the Blox tag library. The `uri` is a pointer to the tag library descriptor file. The `prefix` value, defined as `blox`, tells the server to look for any tags on this page that begin with `blox`, then process the contents using the Blox tag library as defined in the tag library descriptor file.

Just below the taglib directive, the PresentBlox component is specified by the following tags and their tag attributes:

```
<blox:present
    id="queryBuilder4_present"
    height="500"
    visible="false"
    width="100%">
    <blox:grid/>
    <blox:chart/>
    <blox:page/>
    <blox:data
      dataSourceName="qcc2003-acs"
      onErrorClearResultSet="true"
      query="SELECT DISTINCT({[qcc_2003].[Time (Calendar)].[All Time
(Calendar)], [qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2000],
[qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2001],
[qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2002],
[qcc_2003].[Time (Calendar)].[All Time (Calendar)].[2003]} )
ON AXIS(0), DISTINCT( {[qcc_2003].[Products].[All Products],
[qcc_2003].[Products].[All Products].[100  Truffles],
[qcc_2003].[Products].[All Products].[200  Chocolate Blocks],
[qcc_2003].[Products].[All Products].[300  Chocolate Nuts],
[qcc_2003].[Products].[All Products].[400  Specialties]} )
ON AXIS(1) FROM [qcc_2003] WHERE ([qcc_2003].[Measures].[Sales],
[qcc_2003].[Time (Fiscal)].[All Time (Fiscal)],
[qcc_2003].[Date Opened].[All Date Opened],
[qcc_2003].[Has Nuts].[All Has Nuts],
[qcc_2003].[Chocolate Type].[All Chocolate Type],
[qcc_2003].[Ounces Per Package].[All Ounces Per Package],
[qcc_2003].[Pieces Per Package].[All Pieces Per Package],
[qcc_2003].[Date Introduced].[All Date Introduced],
[qcc_2003].[Seasonal].[All Seasonal],
[qcc_2003].[Scenario].[Actual],
[qcc_2003].[Locations].[All Locations])"
      selectableSlicerDimensions="[qcc_2003].[Measures],
[qcc_2003].[Locations],[qcc_2003].[Chocolate Type]"
      useAliases="true"/>
```

```
    <blox:toolbar/>
    <blox:dataLayout/>
    <bloxui:calculationEditor />
</blox:present>
```

The <blox:present> tag specifies that you want a PresentBlox to appear here with an id attribute value of queryBuilder4_present. The id attribute allows you to identify this particular Blox for scripting purposes. Note that the <blox:present> tag includes a visible attribute set to false. When set to false, the PresentBlox will not be rendered until the <blox:display> tag is encountered by the compiler in the page.

There are many attributes available for each Blox defined using tags, but unless you need to specify attributes values that are different than the default values, you do not need to include them in the tag. In this example, a height value of 500 and a width value of 100%. Height and width can be defined in pixels or in percentages.

A nested DataBlox is included, which defines the dataSourceName attribute as qcc2003-acs. If no data source is specified, you will see a ″No data available″ message when the page is displayed. The query attribute includes the MDX query statement generated by DB2 Alphablox Query Builder. When you understand the MDX query language used with the DB2 Alphablox Cube Server, you can simplify the query, making it much shorter. The query statement here, though, is complete and will generate the view you created in the Query Builder application.

The <head> section of the page contains a special Blox tag for adding important code to the page before the page is rendered:

```
<blox:header/>
```

This tag is used by DB2 Alphablox to automatically add required HTML, JavaScript™ and CSS code into the head section of the page. When a page is rendered by the server, this tag adds CSS links to defined HTML themes and a meta tag for preventing caching. Remember to enter this tag into every JSP page that use Blox components. If this tag is not included, the page will not render properly.

In the <body> section of the page is the following line:

```
<blox:display bloxRef="MyPresentBloxView"/>
```

This Blox tag results in the PresentBlox specified above being rendered here. This enables you to make changes to a Blox component or its nested components in one place, where you may also have other Blox tags and JSP scriptlets controlling the behavior of the Blox components. In complex JSP files, and as you become more familiar with DB2 Alphablox, you may elect to have a DataBlox component separate from the PresentBlox tags and to add JSP scriptlets that modify the Blox components and their behavior before the Blox component is rendered in the web browser.

To summarize, the most important Blox tags used on this page include: <blox:header/>, <blox:present>, and <blox:display>. These three tags, and nested tags, specify an analytic view without requiring any Java™ code on the page. The complexity of the presentation logic is being managed by these Blox tags. By adding nested Blox and modifying the attribute values, you can customize views based on your business requirements.

For more complex views and applications, additional JavaBeans™ components and Java code may be required.

Creating your second analytic view

In this task, you will create an analytic view displaying just a chart with page filters, based on code from the first analytic view in this tutorial.

In many applications data is represented using just a grid or a chart. While the PresentBlox component combines both the GridBlox and the ChartBlox as nested components that display together within the PresentBlox component, you can disable or hide the nested components that you do not want to be seen. Using a PresentBlox component to create a chart view with page filters is relatively simple, accomplished by modifying the tag attributes on some of the nested Blox tags.

In this task, you will reuse the PresentBlox tag from the first analytic view to create a second customized analytic view.

To create an analytic view showing a chart with page filters:

1. Open the previously created PresentBloxView.jsp file in your editor and save it as ChartBloxView.jsp.
2. In the PresentBlox tag's nested components listed below, add a `visible` attribute to the tags as shown here and set their values to `false`. Adding the attribute will cause the specified nested components to not display, leaving a chart view with a page filter bar.

   ```
   <blox:grid visible="false"/>
   <blox:chart/>
   <blox:toolbar visible="false"/>
   <blox:page/>
   <blox:dataLayout visible="false"/>
   ```

3. Modify the PageBlox tag (`<blox:page>`), adding the `labelPlacement` attribute and setting the value to `top`. By default, the label for page filters appears to the left of the menus. Adding this attribute and setting it to `top` overrides the default behavior and will place the labels above the page filters.

   ```
   <blox:page labelPlacement="top" />
   ```

4. Modify the name of the `id` attribute on the `<blox:present>` tag to the following value: `chartview`. Each Blox component displayed, or rendered, to a page needs a unique ID in order to prevent errors.
5. Save your changes and add the file to your MyApp directory on the server:
6. Open your web browser and test your new page, either by clicking on the link from your home page or accessing it directly using the URL (for example, http://localhost:9080/MyApp/ChartBloxView.jsp)

You have now created two analytic views, making use of the DB2 Alphablox Query Builder application to help you get started. Also, you learned that by modifying or adding attributes on existing tags, you can control the appearance and behavior of the Blox and their nested components.

Summary

If you finished all of the tutorial tasks, you have learned how to build a basic DB2 Alphablox application using the DB2 Alphablox Admin Pages, the DB2 Alphablox Query Builder application and Blox tags. If you have already taken a look at the *Developer's Reference*, you are aware that there are many properties and methods available for defining and manipulating Blox on JSP pages. So what should you do next?

If you want to see your corporate data in an DB2 Alphablox view, you now know how to create an application from scratch and add analytic views to your application. If you want to see something right away, you could make some simple modifications to the MyApp application. To display data from your corporate databases instead of using the qcc2003-acs data source, you can create data sources that point to your corporate databases, modify the `dataSourceName` attributes to point to the newly defined data source, then add an appropriate `query` attribute. Using the DB2 Alphablox Query Builder, you can often quickly create usable query statements and Blox tags. For details on creating data sources, see the *Administrator's Guide*. To learn how to create appropriate queries and much more about building DB2 Alphablox applications, see Retrieving Data in the *Developer's Guide* and the DataBlox section of the *Developer's Reference*.

Chapter 2. Tutorial: Building your first portlet with Blox components

In this tutorial, you will learn how to add Blox components to your portlets. You will:

1. Install a prebuilt sample portlet. With this step, you can quickly see how a Blox component is added to a portlet JSP page and how it displays in on a portal page.
2. Write your own JSP page with a GridBlox.

The tasks in this tutorial do not provide details about general portlet development. This tutorial focuses on DB2 Alphablox-specific tasks and assumes some familiarity with the general concepts of the portal environment and portlet development.

Prerequisites

- DB2 Alphablox must be installed under a WebSphere Portal Version 5.1 server. See the *Installation Guide* for details about installation.
- Your WebSphere Portal server must be started.
- You must have administrative access to your WebSphere Portal server.
- You must have familiarity with the administrative functions and user interface in WebSphere Portal.
- You should have basic knowledge of Java and JSP.
- You should have a JSP editor installed.

Although you can use any JSP editor or even a text editor for this tutorial, when you develop your own portlets, use a development tool recommended by WebSphere Portal, such as Rational® Application Developer.

This tutorial uses a predefined data source that was installed with DB2 Alphablox. You can use this data source to quickly develop a basic application. You do not need to worry about configuring a custom data source for this tutorial.

Installing the sample portlets

The best way to learn how to add a Blox to your portlet is to install the sample portlets provided in DB2 Alphablox and load them into a portal page. This allows you to examine the basic structure of the JSP code and match that to the output on your portal.

Install the sample portlets provided in DB2 Alphablox:

1. Open your browser and log in to your portal as an administrative user (the URL is in the form of `http://<yourPortalServer>:<port>/wps/portal`).
2. Click the **Administration** button.
3. Under the Portlet Management section, click **Web Modules**. The Manage Web Modules page appears on the right.
4. Click **Install**. You are prompted for the Web Module to install.
5. Click the **Browse** button and navigate to the `installableApps` directory under your DB2 Alphablox installation directory.

6. Choose `AlphabloxSamplePortlets.war` and click **Next**. The DB2 Alphablox Sample Portlets application with a portlet called DB2 Alphablox JSP Page Sample Portlet is displayed in the Portlet application table.
7. Click **Finish**.

The DB2 Alphablox Sample Portlets application and the included portlets are now installed. Check the `installedApps` directory under your WebSphere Portal installation. The name for the newly created directory starts with `DB2 Alpha` and ends with a dynamically generated portlet ID in the pattern of `_PA_x_x_xx.ear`.

Running the sample portlets

To run the sample portlets on your portal page:

1. Go to your portal page.
2. Create or edit an existing page. You can create a new page for testing this sample portlet, or you can click an existing page to edit the page.
3. Click one of the **Add Portlets** buttons on the portal layout page.
4. Type `DB2` in the Search field and press **Search**. The **DB2 Alphablox JSP Page Sample Portlet** check box appears.
5. Select the check box and click **OK**.
6. Click **Done**.

When the portal page refreshes, a PresentBlox appears in your portal page. This PresentBlox has:

- A menu bar on top
- Two toolbar panels underneath the menu bar
- A data layout panel on the left that lets you move dimensions around the different axes
- A grid that shows the data in tabular format
- A three-dimensional bar chart on the right

This `present.jsp` file is the default page to load as specified in the BloxJSPPagePortlet servlet included in the sample. The source code for this sample servlet is available in the `WEB-INF/src/` directory.

You have successfully installed the sample portlets and added a portlet with Blox to your portal page. The next task is to examine the code structure in this JSP page.

Examining the structure of a portlet JSP page with a Blox component

In this task, you review the code structure of a JSP page that contains a Blox. All JSP pages must have the same key elements.

To open a JSP file:

1. Navigate to the `installedApps/` directory under your WebSphere Portal installation, and locate the newly created application folder that starts with DB2 Alpha.
2. Navigate to the `PA_x_x_xx.war/jsp/html/` directory.
3. Open `present.jsp` in your JSP or Java editor.
4. Examine the following code and note the key elements:

```
<%@ page contentType="text/html"%>

<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>

<portletAPI:init/>

<%
  String bloxName = portletResponse.encodeNamespace("presentBlox");
%>

<head>
  <blox:header />
</head>

<blox:present id="presentBlox" bloxName="<%= bloxName %>" width="800">
  <blox:data dataSourceName="canned" />
</blox:present>
```

This block of code contains six key elements:

1. The first line tells the browser the output is HTML:

   ```
   <%@ page contentType="text/html"%>
   ```

2. The next set of code specifies the two JSP tag libraries used in this page:

   ```
   <%@ taglib uri="bloxtld" prefix="blox"%>
   <%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
   ```

   The `uri` is a pointer to the directory location where the tag library descriptor file is located. The `prefix` values, defined as `blox` and `portletAPI`, tell the server to:

   - Look for any tags on this page that begin with `blox`, and then process the contents using the Blox Tag Library as defined in the tag library descriptor file.
   - Look for any tags on this page that begin with `portletAPI`, and then process the contents using the Portlet Tag Library as defined in the tag library descriptor file.

3. Next, a portlet initialization tag is added:

   ```
   <potletAPI:init/>
   ```

   This tag provides access to the PortletRequest, PortletResponse, and PortletConfig objects. With PortletResponse, you can invoke the `encodeNamespace()` method to ensure that the name of your Blox does not clash with other objects on other portlets that run on the same page.

4. The next tag encodes the namespace for the Blox to add to the page:

   ```
   <%
     String bloxName = portletResponse.encodeNamespace("presentBlox");
   %>
   ```

   This allows you to later create a Blox and assign this unique name to it.

5. The next block of code adds the Blox header tag that is required for Blox rendering and server-client communication:

   ```
   <head>
     <blox:header />
   </head>
   ```

   This tag is used by DB2 Alphablox to automatically add the required HTML, JavaScript, and CSS code into the head section of the page. When a page is rendered by the server, this tag results in the inclusion of CSS links to defined

HTML themes and a meta tag to prevent caching. This tag must be added to every JSP page that contains Blox components, or the components will not render properly.

6. Add a PresentBlox using tags provided in the Blox Tag Library:

```
<blox:present id="presentBlox" bloxName="<%= bloxName %>" width="800">
  <blox:data dataSourceName="canned" />
</blox:present>
```

   - This code adds a PresentBlox with an `id` of `presentBlox` and a `bloxName` of `xx_x_x_xxx_presentBlox`, which is the result of namespace encoding.
   - This Blox has a width of 800 pixels and the height of 400® pixels (the default).
   - A nested DataBlox is included, with the `dataSourceName` tag attribute set to `canned`.

   The `id` tag attribute of the PresentBlox is required. It specifies the Java scripting name to use on the JSP page. The `bloxName` attribute specifies the object name on the server. An encoded `bloxName` ensures that the instance is unique on the server.

   Note: If no data source is specified, you will typically see the **No data available** message in the grid. The canned data source is predefined during installation and includes a small amount of sample data. It does not require installation and configuration of a real external database, and can be used for troubleshooting and learning. Because we don't need to specify a query, there is no `query` attribute in the code.

This page does not have an outmost `<html>` tag because this JSP page is displayed on a portal page with other portlets. No additional `<html>` or `<body>` tag are needed.

Now that you have examined the JSP structure and learned the essential code to include in your portlet JSP pages, in the next task, you will create a new JSP page with a different Blox and specify some common Blox attributes to have the page fit better on your portal page.

Creating your own portlet JSP page with Blox components

In this task, you will create a new JSP page with a GridBlox and set some of its properties. The purpose is for you to become familiarize with the general Blox tag construct while creating a GridBlox that fits well in a typical portal page.

A GridBlox has a default size of 400x400 pixels. It also comes with a menubar and a toolbar. You will set some of the common used GridBlox properties so the grid only takes up 100 pixels in height, with both its menubar and toolbar turned off. These are done by setting the following GridBlox properties:

- `height` : set to 100 pixels
- `menubarVisible`: set to false
- `toolbarVisible`: set to false

Make sure you have met the requirements specified in Chapter 2, "Tutorial: Building your first portlet with Blox components," on page 11 and installed the sample portlet application as described in the section on "Installing the sample portlets" on page 11.

Follow the steps below:

1. In the browser window where the PresentBlox is displayed, click the portlet's edit button (the button with a pencil icon). You are presented with a selection drop list.
2. In the drop list, select ″Grid Blox″ and click **OK**.

   As the page refreshes, you should see a 400x400 GridBlox. This GridBlox, by default, has its menubar and toolbar turned on. You will change its size to 400x100, with both the menubar and toolbar turned off.
3. Navigate to the PA_*x_x_xx*.war/jsp/html/ directory under the DB2 Alphablox Sample Portlets application installed previously.
4. Open grid.jsp in your JSP editor. This page looks almost identical to present.jsp, except that:
   - the <blox:present> tag now says <blox:grid> and the value for id is different:

     ```
     <blox:grid id="gridBlox" bloxName="<%= bloxName %>" width="400">
       <blox:data dataSourceName="canned" />
     </blox:grid>
     ```
   - the bloxName value is different:

     ```
     <%
       String bloxName = portletResponse.encodeNamespace("gridBlox");
     %>
     ```
5. Set the height of this GridBlox to 100 pixels by adding the height attribute and setting its value to 100:

   ```
   <blox:grid id="gridBlox" bloxName="<%= bloxName %>" width="400" height="100">
     <blox:data dataSourceName="canned" />
   </blox:grid>
   ```
6. Turn off the menubar and toolbar on the top by setting the menubarVisible and toolbarVisible attributes to false:

   ```
   <blox:grid id="gridBlox" bloxName="<%= bloxName %>" width="400" height="100"
      menubarVisible="false" toolbarVisible="false" >
     <blox:data dataSourceName="canned" />
   </blox:grid>
   ```

   Make sure you enter the attribute names correctly, including the cases (uppercase ″V″ in both attribute names). Also make sure the attributes are added before the ending angle bracket (″>″).
7. Change the namespace to myFirstGrid:

   ```
   <%
     String bloxName = portletResponse.encodeNamespace("myFirstGrid");
   %>
   ```

   Changing the namespace for this GridBlox is to ensure that the changes you just made will be reflected as you load this JSP into your portal. Since you have previously loaded this page, there is already an instance of this GridBlox running on the server for this session. Unless you change the namespace, the changes you just make will not be reflected even if you refresh the page. Changing the namespace is a quick way to test your changes in a development environment. Alternatively, you can also open a new browser window so a new object is created on the server for the new session.
8. Save the file.

You are now ready to test this file on WebSphere Portal.

To test the changes you just made:

1. Go back to the portal page.
2. Click the browser's Refresh button to reload the page.

You should see a 400x100 GridBlox with no menubar or toolbar.

Note: This JSP selection drop list is not a built-in functionality of WebSphere Portal, but created by this sample portlet. Check the `edit.jsp` file and the Java source files in the `WEB-INF/src/` directory.

Creating a portlet project to use Blox components

When creating your portlet project in your development tool, make sure the AlphabloxServer servlet mapping and the tag library references needed for Blox components are added to your project's `web.xml` file, and the `.tld` files for DB2 Alphablox Tag Libraries are copied over to your project.

1. Modify your project's `web.xml` file to include the following lines:
   - For servlet definition and servlet mapping:

```
<servlet>
  <servlet-name>AlphabloxServer</servlet-name>
  <servlet-class>com.alphablox.server.webapps.server.AlphabloxServer
     </servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>AlphabloxServer</servlet-name>
  <url-pattern>/abx/*</url-pattern>
</servlet-mapping>
```

   - For tag library references:

```
<taglib>
  <taglib-uri>bloxtld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/blox.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>bloxformtld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/bloxform.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>bloxlogictld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/bloxlogic.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>bloxreporttld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/bloxreport.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>bloxportlettld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/bloxportlet.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>bloxuitld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/bloxui.tld</taglib-location>
</taglib>
```

2. Copy the `.tld` files for DB2 Alphablox Tag Libraries into your project's `WEB-INF/tlds/` directory. These files are located under

   `<db2alphablox_dir>/bin/`

   where `<db2alphablox_dir>` is your DB2 Alphablox installation directory.

Configuring a portlet project using Rational Application Developer

WebSphere Portal recommends that you use Rational Application Developer for portlet development. In particular, Rational Application Developer Version 6.0 is required for WebSphere Portal Version 5.1. Rational Application Developer provides a wizard that guides you through the setup and creation of a portlet project. Based on your selection, it sets up the appropriate structure and deployment descriptor file, and automatically creates the necessary Java classes for your controller and JSP pages for your portlet view. As you follow along the New Portlet Project wizard, make sure you have the following configured correctly:

- Choose "WebSphere Portal v5.1 stub" for your target server. This is specified on the wizard's first screen by clicking the **Show Advanced >>** button.
- Add the servlet mapping for DB2 Alphablox to the `web.xml` file created under `WebContent/WEB-INF/`. This is done by entering the following servlet definition and mapping code in your project's `web.xml`:

```
<servlet>
  <servlet-name>AlphabloxServer</servlet-name>
  <servlet-class>com.alphablox.server.webapps.server.AlphabloxServer
      </servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>AlphabloxServer</servlet-name>
  <url-pattern>/abx/*</url-pattern>
</servlet-mapping>
```

- Add the Alphablox Tag Libraries to your project's `web.xml`. Follow the steps below:
  1. With `web.xml` open, click the **Variables** tab.
  2. Scroll to the Tag Libraries References section at the bottom.
  3. Click **Add**.
  4. For **URL** and **Location**, enter the values based on the following table:

URL	Location
bloxtld	/WEB-INF/tlds/blox.tld
bloxformtld	/WEB-INF/tlds/bloxform.tld
bloxlogictld	/WEB-INF/tlds/bloxlogic.tld
bloxreporttld	/WEB-INF/tlds/bloxreport.tld
bloxportlettld	/WEB-INF/tlds/bloxportlet.tld
bloxuitld	/WEB-INF/tlds/bloxui.tld

For each pair of URL and Location, you will need to click **Finish** to add the tag library reference and then click **Add** to add the next pair.

Note: For details on the different tag libraries and their uses, see Using JavaServer Pages and Blox .

Finally, remember to copy the `.tld` files for DB2 Alphablox Tag Libraries into your project's `WEB-INF/tlds/` directory. These files are located under

`<db2alphablox_dir>/bin/`

where `<db2alphablox_dir>` is your DB2 Alphablox installation directory.

Next steps

You have learned how to create a portlet with Blox components and the essential code structure. In order to display data from your databases, you need to:

1. Define a new data source to DB2 Alphablox that points to your data source.
2. Modify the `dataSourceName` attribute of DataBlox to point to the defined data source.
3. Add a `query` attribute with your query string to DataBlox.

For details on creating data sources, see the *Administrator's Guide*. To learn how to create appropriate queries, see Retrieving Data in the Developer's Guide and the DataBlox section of the *Developer's Reference*.

Once you have your own data appearing in Blox views, you can also begin exploring the many properties that can be set by using the Blox tags.

Portlet development tips

As you use the information provided in the DB2 Alphablox documentation and continue your portlet development, there are some portlet-specific topics, concepts, and general guidelines that you need to be aware of. The following list puts together these general development guidelines as well as pointers to specific sections in the documentation that are of interest to you.

- Always encode the name of your Blox using the portlet's namespace. A namespace ensures a unique Blox name for the current J2EE session.
- Always set the width and height of a Blox using pixels. Outside of the portal environment, you can set the width and the height to a percentage value such as "50%" or "100%". Percentage values do not work in the portal environment because multiple portlets coexist on the page.
- Do not use relative URLs to invoke resources within a portlet. URLs should be encoded using the `encodeURL()` method of the PortletResponse class. In places where the documentation shows the use of relative URL, you should always encode the URL.
- To ensure that your Blox portlet uses a theme similar to your portal theme, use the Portal Theme Utility. This utility is available from two sources:
  - The DB2 Alphablox home page.

    Under the **Administration** tab, click the **General** link. The Utility is listed under the Portal section. See the *Administrator's Guide* and the online help for details.
  - The `AlphabloxAdminPortlets.war` file.

    This is a portlet version of the Portal Theme Utility. You can install this portlet to run it through your portal without having to log in separately to the DB2 Alphablox home page.

  This utility will combine the styles from your portal environment and the styles in DB2 Alphablox into one so that Blox will be displayed in similar colors and fonts as other portlets on the page.
- Check the Blox Portlet Tag Library topic in the *Developer's Guide* for issues you need to take into consideration during the planning phase.
- DB2 Alphablox has a powerful Blox UI Model. The UI Model includes a ClientLink object that lets you load a specified URL when a Blox component on the page is clicked. For portlet development, use the Blox Portlet Tag Library to

create the ClientLink. The tags will handle URL formation and handling dynamically so that the links do not turn stale after the page is refreshed. See the topic, Planning for portlet development, in the *Developer's Guide* for details.

- For portlet and action URIs, use the Blox Portlet Tag Library to create the portlet link or action link. You can then utilize the portal's Portlet API to process the action or portlet link. See the Blox Portlet Tag Library topic in the *Developer's Guide* for details.

Chapter 3. Tutorial: Building applications using Rational Developer tools

You can use Rational Developer tools with the DB2 Alphablox Toolkit to more quickly develop applications using DB2 Alphablox components and the DB2 Alphablox Java APIs.

Rational Developer tools (Rational Application Developer or Rational Web Developer) can be used with the DB2 Alphablox Toolkit, an Eclipse-based set of plug-ins, to develop and test applications built using DB2 Alphablox technology. This tutorial will guide you in configuring your Rational integrated development environment to enable the use of method and tag completion as well as custom enhancements added with the DB2 Alphablox Toolkit.

To begin configuring your Rational Developer tool with the DB2 Alphablox Toolkit to develop and test DB2 Alphablox applications.

Preparing your development environment

Before you can install and begin using the DB2 Alphablox Toolkit, prepare your development environment by making sure that you have all of the required software.

To prepare your development environment:

1. Install a Rational Developer tool (either Rational Application Developer or Rational Web Developer) on your workstation.
2. Install any required updates to ensure that you have at least Rational Application Developer or Rational Web Developer version 6.0.0.1.
3. If you plan to use WebSphere 5.1 integrated test environments, make sure that you have updated your Rational Developer version to include this optional installation.

   Note: Default installations of the Rational Developer tools do not include the WebSphere 5.1 integrated test environments.

You are now ready to perform the tasks in this tutorial.

Installing the DB2 Alphablox Toolkit

Using the DB2 Alphablox Toolkit with the Rational Developer tool will make it easier for you to begin developing web-based applications with DB2 Alphablox content.

The DB2 Alphablox Toolkit requires the use of Rational Application Developer or Rational Web Developer version 6.0.0.1.

Installing the DB2 Alphablox Toolkit into a Rational Developer tool saves having from having to perform many manual steps to enable the use of content completion for Java methods and JSP custom tags. Also, custom wizards and cheat sheets can be used guide you in creating applications with DB2 Alphablox content and defining WebSphere server instances.

To install the DB2 Alphablox Toolkit into Rational Application Developer or Rational Web Developer:

1. Insert the DB2 Alphablox Installation Disk into the CD drive on your workstation.
2. In the plugin directory, locate the subdirectory named UpdateSite.
3. Copy the UpdateSite directory to a convenient location on your hard disk. For example, copy the directory to C:\DB2Alphablox\UpdateSite.
4. Start the Rational Developer tool.
5. On the menu bar, select **Help > Software Updates > Find and Install**.
6. In the Install window that opens, select the **Search for new features to install** option and then click **Next**.
7. On the Update sites to visit window, click the **New Local Site** button and browse to the location of the UpdateSite directory.
8. Click **Next**, select the **DB2 Alphablox Toolkit** feature, and click **Next** again.
9. In the windows that follow, accept the license agreement and select the location where the DB2 Alphablox Toolkit feature will be installed.
10. After the installation is complete, restart the Rational Developer tool

After you restart the Rational Developer tool, the DB2 Alphablox Toolkit features will be available.

Installing DB2 Alphablox in WebSphere integrated test environments

In this task, you install DB2 Alphablox on the WebSphere integrated test environments in a Rational Developer tool.

Prerequisites: The supported WebSphere integrated test environment must be installed in the Rational Developer tool. You need access to the DB2 Alphablox Installation Disk. Review and perform any required preinstallation steps described in the Preinstallation Tasks section of the *DB2 Alphablox Installation Guide*.

DB2 Alphablox can be installed on standalone WebSphere application servers or on the WebSphere integrated test environments available within the Rational Developer tool. The summary of steps outlined below explain the differences in installation that must be performed when installing DB2 Alphablox on the WebSphere runtimes available within the Rational Developer tool.

To install DB2 Alphablox in WebSphere integrated test environments

Install DB2 Alphablox following the steps in the Installation section of the *DB2 Alphablox Installation Guide*, with the following exceptions:

1. In the Configure WebSphere window of the installer, in the **WebSphere Root Directory** field, specify the location of the WebSphere runtime that you want to use as your integrated test environment server.

   For example, to install DB2 Alphablox to the WebSphere 6 integrated test environment on the default installation, select the path to the base_v6 directory. For a typical Rational Application Developer installation, the path would be similar to the following:

   ```
   C:\Program Files\IBM\Rational\SDP\6.0\runtimes\base_v6
   ```
2. In the WebSphere Settings window, specify a name and password for the WebSphere administrator. These entry values are not used in the WebSphere integrated test environment, but are required by the DB2 Alphablox installer.

Important: Do not perform the post-installation steps described in the *DB2 Alphablox Installation Guide*.

After installing DB2 Alphablox into the WebSphere runtime, you can create WebSphere server instances for testing DB2 Alphablox applications within the Rational Developer tool.

Creating a WebSphere server instance

To run applications or JSP files that contain DB2 Alphablox content, you need to create a WebSphere server instance that can access required DB2 Alphablox services and Java classes.

Prerequisites: Configure the Rational Developer tool. Install the DB2 Alphablox Toolkit. Install DB2 Alphablox in the WebSphere integrated test environments.

To create a WebSphere server instance for testing applications and JSP files containing DB2 Alphablox content:

1. Open the Rational Developer tool.
2. Click the **Servers** tab (if the tab is not visible, select **Window > Show View > Servers**).
3. Right-click inside the view window and select **New > Server**. The New Server window opens.
4. Enter `localhost` as the host name and select the server type. Click **Next**.
5. Enter the server port number and click **Next**.
6. Add the available projects that you want to run in this server instance to the list of configured project. Click **Finish**. Your new server instance opens in the **Servers** view.

If you created a WebSphere 5.1 server instance, you need to perform the next step, ″Creating WebSphere 5.1 server substitution variables.″ For WebSphere 6 server instances, proceed to ″Adding guest to the DB2 Alphablox Administrators group.″

Creating WebSphere 5.1 server substitution variables

In this task, you will modify server substitution variables for WebSphere 5.1 server instances to properly run DB2 Alphablox within WebSphere integrated test environments.

1. Open the **Servers** view and double-click on the WebSphere 5.1 server instance you want to modify.
2. Click the **Configuration** tab for the server instance.
3. Select **Enable administration console** and deselect **Enable universal test client**.
4. Click the **Variables** tab for the server instance. The **Substitution Variables** window opens.
5. Add two new variables using the **Add** button (located next to the **Node Settings** list).
   a. Add a variable named `WS_EAR_AlphabloxPlatform` and set the value to `$(APP_INSTALL_ROOT)/localhost/AlphabloxPlatform.ear`.
   b. Add a second variable named `WS_EAR_AlphabloxStudio` and set the value to `$(APP_INSTALL_ROOT)/localhost/ApplicationStudio.ear`
6. Save your changes.

For WebSphere 5.1 server instances, you need to follow the steps in ″Configuring WebSphere 5.1 server instances″ to finish modifying your server instance.

Configuring WebSphere 5.1 server instances

For WebSphere 5.1 server instances, you need to configure the server instances to run DB2 Alphablox applications and files.

Prerequisites: Create the required WebSphere 5.1 server substitution variables.

To configure WebSphere 5.1 server instances:

1. Open the **Servers** view in the Rational Developer tool and start the server instance that you want to configure.
2. Right-click the server instance and select **Run Administrative Console**.
3. Do not enter a value in the **ID** field. Click **OK**.
4. Click the **OK** button below the configuration table.
5. Open the **Enterprise Applications** navigation view and click the **ApplicationStudio** application name.
6. Set **Application Binaries** to `$(WS_EAR_ApplicationStudio)`.
7. Select **Use Metadata from Binaries**.
8. Click the **OK** button below the configuration table.
9. Click the **Save** button on the WebSphere Administrative Console.
10. In the Save to Master Configuration window, click **Save**.
11. Start the AlphabloxPlatform and ApplicationStudio applications in the WebSphere Administrative Console under **Applications > Enterprise Applications**.

The server instance is configured. Next you need to do the ″Add guest to DB2 Alphablox Administrators group″ step.

Adding the guest user to the to DB2 Alphablox Administrators group

Adding the guest user to the DB2 Alphablox Administrators group allows you to access the DB2 Alphablox Administration Pages while using Rational Developer tool.

Prerequisites: Create the WebSphere server instance. Configure the WebSphere server instance.

To add the guest user to the DB2 Alphablox Administrators group:.

Important: Do not give the guest user administrator rights on your production WebSphere servers.

1. Log into the DB2 Alphablox console by using the following telnet command: `telnet localhost` *`portNumber`* . where *portNumber* is the port specified during your DB2 Alphablox installation.
2. At your telnet console prompt, type the following DB2 Alphablox console command: `set Administrators guest` and press Enter.
3. In the telnet console, type `save` and then press Enter.
4. Close the telnet session.

Your WebSphere server instance is now ready for use.

Creating DB2 Alphablox applications

When you create new applications in Rational Developer, you need to add required DB2 Alphablox content for the applications to run properly.

Prerequisites: Install the DB2 Alphablox Toolkit on the Rational Developer tool.

1. On the Rational Developer tool menu bar, select **File > New Project**. The **New Project** wizard opens.
2. Expand the **Web** option, select **Dynamic Web Project**, and then click **Next**.
3. Type a name for your project and click the **Show Advanced** button. Additional options appear.
4. Select the appropriate servlet version and target server.
5. Click **Next**. The Features options window opens.
6. Select the **DB2 Alphablox Content** option and then click **Finish**.

Your new web application project is now DB2 Alphablox-enabled. DB2 Alphablox tag libraries and Blox Java API are available, and the application descriptor file (web.xml) has been modified to include required DB2 Alphablox information. You can now begin adding JSP files containing DB2 Alphablox content to your project.

Creating JSP files with DB2 Alphablox content

In this task, you create new JSP files with access to Blox tag libraries using the Rational Developer tool with the DB2 Alphablox Toolkit.

Prerequisites: Install the DB2 Alphablox Toolkit on the Rational Developer tool.

1. In the Rational Developer's **Project Explorer** view, select **File > New > JSP File**.
2. Type a file name in the **File Name** field.
3. Click **Configure advanced options** and press **Next**.
4. Add the DB2 Alphablox tag libraries that you will use in your JSP file.
   a. Click the **Add** button to open the Add Tag Libraries window.
   b. Select the DB2 Alphablox tag libraries you plan on using in your JSP file.
   c. Click **Next**.
5. Select **ISO 10646/Unicode(UTF-8)** from the **Encoding** list. The UTF-8 encoding is required for DB2 Alphablox applications to run properly.
6. Click **Finish**. Your new JSP file appears in the project listing.
7. Double-click the file name to open the file in the JSP editor window.
8. Place your cursor inside the HTML `<head>` tag, but after the DB2 Alphablox JSP taglib directives that you added above.
9. Enter the Blox header tag by typing the following on the new line: `<blox:header/>` The Blox header tag is needed to add required DB2 Alphablox JavaScript and CSS files when you run your JSP file.

Your new JSP file is now enabled to access the DB2 Alphablox tag libraries that you selected. With the Rational Developer tool's Content Assist feature, you can insert Blox tags and tag attributes for the selected tag libraries.

Accessing DB2 OLAP Server and Essbase data sources

In order to access DB2 OLAP Server™ or Essbase data sources in the Rational Developer tool when using WebSphere 5.1 server instances, you need to create a startup batch file that loads required client libraries.

Prerequisites: Install the DB2 Alphablox Toolkit. Install DB2 Alphablox to the WebSphere 5.1 integrated test environments. Configure the WebSphere 5.1 server instances. Install IBM DB2 OLAP Server or Hyperion Essbase on your development machine.

Due to limitations in the WebSphere 5.1 integrated test environments in the Rational Developer tool, you need to create a startup batch file for the Rational Developer tool to access DB2 OLAP Server or Hyperion Essbase data sources. When you start the Rational Developer tool using the batch file, required Essbase client libraries are added to the Java library path.

To create a batch file to enable access to DB2 OLAP Server or Essbase with WebSphere 5.1 server instances:

1. Using a text editor, create and a new text document.
2. Add a line of code to invoke the DB2 Alphablox aassetup.bat file, located in your DB2 Alphablox installation directory. For example, the following code will run the aassetup.bat file located in the specified DB2 Alphablox installation:

   ```
   call C:\alphablox\analytics\bin\aassetup.bat
   ```
3. Add a second line to invoke the RAD rationalsdp.exe file, which will start up RAD. For example, the following code will run the Rational Developer tool:

   ```
   call C:\Program Files\IBM\Rational\SDP\6.0\rationalsdp.exe
   ```
4. Save this file as startRAD.bat to your workstation desktop (or another convenient location).

When you double-click on the startRAD.bat file, the DB2 Alphablox aassetup.bat runs and sets the required paths and environment variables, then Rational Application Developer is started. The version of the Essbase client libraries used with DB2 Alphablox needs to match the version of the DB2 OLAP Server (or Hyperion Essbase) version you are using as a data source. To modify the Essbase client libraries used with DB2 Alphablox, run the DB2 OLAP Server / Essbase Client Library Utility (ChangeEssbase.bat), which can be found in the *db2_alphablox*\analytics\bin directory, where *db2_alphablox* is the root directory of your DB2 Alphablox installation. This utility modifies one of the batch files run when you start RAD using the startRAD batch file you created.

Below is a complete example startRAD.bat file described above:

```
call C:\alphablox\analytics\bin\aassetup.bat
call C:\Program Files\IBM\Rational\SDP\6.0\rationalsdp.exe
```

Whenever you intend to access DB2 OLAP Server or Essbase data sources, the startupRAD.bat file must be used to start the Rational Developer tool.

Chapter 4. Tutorial: Building a DB2 Alphablox cube using DB2 Cube Views

The tutorial guides you through the creation of a DB2 Alphablox cube built using a DB2 Cube Views™ sample database.

The tasks in the tutorial do not provide details about building custom DB2 Alphablox cubes. Instead, the goal is to show you how to quickly create a DB2 Alphablox cube that can be used in exploring the capabilities of DB2 Alphablox Cube Server. The resulting data source can also be used for testing and building DB2 Alphablox applications.

Prerequisites:

- Install DB2 Alphablox. See the *Installation Guide* for details about installation.
- Gain access rights to a supported DB2 Cubes Views implementation which has the DB2 Cube Views CVSAMPLE sample database installed. See the *Installation Guide* for details about supported versions of DB2 Cube Views.

In the tutorial, you will learn how to build a DB2 Alphablox cube based on the DB2 Cube Views CVSAMPLE sample database. During the tutorial, you will learn the following tasks:

Defining a DB2 relational data source

In this task, you will define a data source definition in DB2 Alphablox for a DB2 database.

Prerequisites: Required DB2 JDBC drivers must be accessible to DB2 Alphablox.

A DB2 Alphablox cube requires that the underlying relational data source is predefined as a DB2 Alphablox data source. DB2 Alphablox cubes are generated using the metadata and data available in relational databases.

To define a DB2 Alphablox data source for a DB2 database:

1. Log into the DB2 Alphablox Administration Pages as the admin user (or as a user with administrator rights).
2. Click the **Administration** tab.
3. Click the **Data Sources** link.
4. Click the **Create** button.
5. From the **Adapter** menu, select the appropriate IBM DB2 JDBC driver for your database server.

   Select **IBM DB2 JDBC Type 4 Driver** or **IBM DB2 UDB on iSeries Driver**.
6. In the **Data Source Name** field, type `CVSAMPLE` as the name to be used for your data source.
7. Enter the appropriate values for the **Server Name**, **Port Number**, and the **Database Name** (this should be `CVSAMPLE`).

   Note: If you need help determining the correct values for these fields, contact your database administrator.
8. Enter a **Default Username** and **Default Password**.

The username and password must be valid on the relational database. The default username and password are always used when a DB2 Alphablox cube accesses the relational database. The database user you use should have read access rights to the database.

Note: The value for **Use DB2 Alphablox Username and Password** is ignored when the specified relational data source is being used to populate a DB2 Alphablox cube. Access control lists (ACLs) can be used to restrict access to DB2 Alphablox cubes. For information about ACLs, see the *Administrator's Guide*.

9. The **Maximum Rows** and **Maximum Columns** values are ignored when the data source is being used to populate a DB2 Alphablox cube. You can still enter values and they will be used when other applications use the data source.
10. Set the **JDBC Tracing Enabled** value to **No**, unless you want to write JDBC logging information to the DB2 Alphablox log file. Enable JDBC tracing only if you are experiencing problems and you need to debug their causes.
11. Click the **Save** button to save the data source definition.

You have now defined the DB2 Alphablox data source definition for CVSAMPLE. Now you can now create a DB2 Alphablox cube definition for accessing the Cube Views cubes metadata in this DB2 data source.

Defining an Alphablox Cube Server Adaptor data source

In this task, you will define a DB2 Alphablox data source definition that uses the Alphablox Cube Server Adapter.

Prerequisites: Create the DB2 Alphablox data source definition for your DB2 CVSAMPLE database.

To define a DB2 Alphablox Cube Server Adapter data source:

1. Log into the DB2 Alphablox Administration Pages as the admin user (or as a user with administrator rights).
2. Click the **Administration** tab.
3. Click the **Data Sources** link.
4. Click the **Create** button.
5. From the **Adapter** menu, select the **Alphablox Cube Server Adapter** option.
6. In the **Data Source Name** field, enter `DB2AlphabloxCubes` as the name to be used for your data source.
7. Click the **Save** button to save your data source definition.

You have defined a DB2 Alphablox data source that can be used for accessing DB2 Alphablox cubes. Next you need to define a DB2 Alphablox cube that you can access.

Defining a DB2 Alphablox cube

In this task, you will define a DB2 Alphablox cube based on the metadata from a DB2 Cube Views CVSAMPLE cube.

Prerequisites: Define the DB2 relational data source. Define a DB2 Alphablox Cube Server Adapter data source.

To define the general properties of a DB2 Alphablox cube:

1. Log into the DB2 Alphablox Administration Pages as the admin user (or as a user with administrator rights).
2. Click the **Administration** tab.
3. Click the **Cubes** link.
4. Click the **Create** button. The Cube Administration window opens.
5. Define the new cube:
   a. In the **DB2 Alphablox Cube Name** field, type `CVSales`.
   b. Select the **Enabled** option, located next to the **DB2 Alphablox Cube Name** field. Selecting this option results in your cube starting automatically when your server restarts.
   c. From the **Relational Data Source** menu, select `MyDB2`, the relational data source you created for this tutorial.
   d. Leave the **Security Role** option unselected. This option can be used to limit the users able to access a particular cube.
6. Enable DB2 Cube Views settings and specify the metadata to be used:
   a. Select the **Enable DB2 Cube Views Settings** option.
   b. From the **Cube Model** menu, select **CVSAMPLE.Sales**.
   c. From the **Cube** menu, select the **General Sales** cube,
   d. Select the **Use Business Names** radio button to specify the names. Selecting this option uses the typically more meaningful and readable member names.
   e. Click the **Import Cube Definition** button. Using this option allows you to import a cube definition and pre-populate measures and dimensions in your DB2 Alphablox cube. The cube definition that is imported reflects the DB2 Cube Views cube as closely as DB2 Alphablox can match, based on support of DB2 Cube Views metadata. When you have more experience in working with cubes, you can modify measures and dimensions to meet your needs.
   f. Click on the **Show Import Log** button to see a log specifying information and debugging messages related to the import operation. For this tutorial, this step is included just to help familiarize you with this feature.
   g. Select the **Import cube definition on start, rebuild, and edit** option. This option will result in your DB2 Alphablox cube using the latest DB2 Cube Views definition each time your DB2 Alphablox cube is started, rebuilt, or opened for edit. With more familiarity with DB2 Alphablox and DB2 Cube Views, you can import the cube definition and customize it.
7. Click the **Save** button to save the DB2 Alphablox cube definition.

You have now created a DB2 Alphablox cube definition. Now you can start up your new CVSales cube.

Starting your DB2 Alphablox cube

In this task, you start the CVSales cube using the DB2 Alphablox Administration Pages.

Prerequisites: Define the DB2 relational data source. Define a DB2 Alphablox Cube Server Adapter data source. Define the DB2 Alphablox cube.

To start the CVSales cube:

1. Log into the DB2 Alphablox Administration Pages as the admin user (or as a user with administrator rights).
2. Click the **Administration** tab.
3. Under the **Runtime Management** section, click the **Cubes** link.
4. From the **DB2 Alphablox Cubes** list, select the DB2 Alphablox cube you want to start.
5. Click the **Start** button. When the cube has started, the status field displays **Running**.

You now have a sample DB2 Alphablox cube running. Now you can begin building applications using the DB2 Alphablox cube you have just created. You can perform a quick check on your new cube by using the Query Builder application, selecting your cube as the data source and running MDX queries against the cube.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation, Licensing, 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation, J46A/G4, 555 Bailey Avenue, San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

DB2	DB2 OLAP Server	DB2 Universal Database
IBM	WebSphere	

Alphablox and Blox are trademarks or registered trademarks of Alphablox Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux® is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Index

P

portlet
  JSP structure 12
  sample 11

Q

queries
  generating using Query Builder 4
Query Builder
  using 4
Query Builder, DHTML
  using 4

T

tutorials
  application development 1

Program Number: 5724-L14

Printed in USA

GC18-9607-01