IBM DB2 Alphablox

概説およびインストール

バージョン 8.4

GD88-6692-01
(英文原典：GC18-9607-01)

IBM DB2 Alphablox

概説およびインストール

バージョン 8.4

GD88-6692-01
(英文原典：GC18-9607-01)

お願い

本書および本書で紹介する製品をご使用になる前に、 37 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM DB2 Alphablox for Linux, UNIX and Windows (製品番号 5724-L14) バージョン 8 リリース 4 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： GC18–9607–01
IBM DB2 Alphablox
Getting Started
Version 8.4

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2006.3

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

第 1 章 チュートリアル: 最初のアプリケーションを構築する

このチュートリアルでは、Blox コンポーネントについての重要な基礎を学習し、DB2® Alphablox アプリケーションをすばやく構築するためのステップを解説します。

DB2 Alphablox アプリケーションを構築するには、JavaServer Pages (JSP) テクノロジーを使用します。しかし、このチュートリアルでは、JSP の知識は必要ありません。このチュートリアルでは、分析アプリケーションの構築を開始するために知っておく必要のある JSP の特徴について説明します。 JSP テクノロジーについて詳しく学習するために、多数の優れた資料を入手できます。

前提条件

このチュートリアルの各ステップでは、WebSphere® Application Server で稼働する DB2 Alphablox を使用していることを前提としています。アプリケーション・サーバーとして BEA WebLogic または Apache Tomcat を使用している場合、サーバーでの特有の違いに適合するように、いくつかのステップを変更する必要があるかもしれません。

また、このチュートリアルを開始する前に、DB2 Alphablox キューブ (qcc_2003) およびその基礎となるリレーショナル・データ・ソース (qcc2003-rdb) をインストールする必要があります。このチュートリアルに必要な DB2 Alphablox キューブおよびリレーショナル・データベースを構成するためのサンプル・データベース・ファイルと説明 (readme.txt ファイル内) は、DB2 Alphablox Installation CD の sampledata/qcc/acs ディレクトリーにあります。

アプリケーションを定義する

J2EE 開発アプローチを使用してアプリケーションを作成するには、 WEB-INF ディレクトリーにアプリケーション記述子ファイル (web.xml) が入ったディレクトリー構造を作成する必要があります。 DB2 Alphablox でこの構造を作成する最も簡単な方法は、DB2 Alphablox 管理ページの「Application」ページを使用して、新規アプリケーションを作成することです。

アプリケーションとフォルダーを作成するには、以下のようにします。

1. 「*管理者用ガイド*」の『アプリケーションの定義』のセクションで説明されているステップに従って、MyApp という新規アプリケーションを DB2 Alphablox で作成します。
   - `MyApp` を「**Name**」フィールドに入力します。
   - `My App` (スペースあり) を「**Display Name**」フィールドに入力します。これは、「Applications」ページのリストに表示されるアプリケーションのラベルを定義します。
2. ページの左上にある先頭の「**Applications**」タブをクリックします。使用可能なアプリケーションのリストが開きます。

3. アプリケーションのリストで、新規に作成したアプリケーションの名前 (My App) をクリックします。ファイルをまだ作成していないため、ファイルのディレクトリーは空です。
4. アプリケーション・サーバー上の新規のアプリケーション・フォルダーにナビゲートします (WebSphere の場合、フォルダーは WebSphere `installedApps` ディレクトリーにあります)。アプリケーションには、`web.xml` ファイルが入った `WEB-INF` ディレクトリーと、`tlds` ディレクトリーが含まれることに注意してください。 `web.xml` ファイルはアプリケーション情報を定義します。 `tlds` ディレクトリーには、Blox タグ・ライブラリー記述子ファイル (`blox.tld`) が入っています。これは、このチュートリアルで分析ビューを作成するために使用する Blox タグを定義します。ディレクトリーには、その他の Blox タグ・ライブラリー用のその他のタグ・ライブラリー記述子 (TLD) ファイルもあります (このチュートリアルでは使用されません)。それには、`bloxform.tld`、`bloxlogic.tld`、`bloxreport.tld`、および `bloxui.tld` が含まれています。

これで、DB2 Alphablox アプリケーション・フレームワークが作成され、分析ビューを追加することができます。

データにアクセスする

このチュートリアルでは、これを開始する前に作成した DB2 Alphablox キューブ・データ・ソースを使用します。リレーショナル・データベースと Alphablox Cube Server が稼働している必要があります。また、使用する DB2 Alphablox キューブも稼働している必要があります。

必要なデータ・ソースが使用可能であり、稼働していることを確認するには、以下のようにします。

1. ブラウザーを開いて、DB2 Alphablox 管理ページにアクセスします。
2. 「**管理**」タブをクリックします。
3. 「**データ・ソース**」をクリックします。
4. 「データ・ソース」ページの左端のメニューで、qcc2003-rdb データ・ソースを見つけます。これがメニューにない場合、DB2 Alphablox Installation CD の sampledata/qcc/acs ディレクトリーにある readme.txt ファイル内のステップを検討する必要があります。
5. 使用可能なデータ・ソースのリストで qcc2003-rdb をクリックしてから、「**選択したデータ・ソースのテスト (Test Selected Data Source)**」ボタンをクリックします。データベースが稼働しており、データ・ソース定義が正しく構成されている場合には、成功したというメッセージが表示されます。エラー・メッセージが表示される場合には、データ・ソース定義を調べるか (データ・ソースを選択して、「**編集**」ボタンをクリックする)、またはデータベースに対するアクセス権を持つユーザーでデータベースが正しく構成されているかどうかを調べます。
6. データ・ソースのリストから qcc2003-acs データ・ソースを選択してから、「**選択したデータ・ソースのテスト (Test Selected Data Source)**」ボタンをクリックします。 Alphablox Cube Server データ・ソースが正しく構成される場合は、成功したというメッセージが表示されます。エラー・メッセージが表示される場合には、データ・ソース定義を確認してください。

データ・ソースのテストが正常に行われたら、いくつかの単純な DB2 Alphablox 分析ビューを構築する方法を学習する準備ができたことになります。

アプリケーション・ホーム・ページを作成する

この作業では、基本的な Web スキルを使用して単純なホーム・ページを作成できます。

このチュートリアルの目的としては、非常に単純なWeb アプリケーション構造を作成するだけで十分です。ホーム・ページ上に、このチュートリアルで作成する 2 つの分析ビューにアクセスするための 2 つのリンクを作成する必要があります。

単純なホーム・ページを作成するには、以下のようにします。

1. テキスト・エディターまたは任意の開発ツールを使用して`index.html` という名前を付けます。このチュートリアルの目的としては、非常に単純なもので済みます。
2. ファイルを編集して、アプリケーション・タイトル (たとえば、「My DB2 Alphablox Application」) と、作成する 2 つの分析ビューへの 2 つのリンクを指定します。

   以下のコードをコピーして index.html ファイルに貼り付けるか、または入力します。

   ```
   <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="ChartView.jsp">Customized Chart View</a>
   </p>
   </body>
   </html>
   ```
3. このファイルを、アプリケーションを定義したときに作成した `MyApp` ディレクトリーに保管します。
4. Web ブラウザーを開いて、DB2 Alphablox ホーム・ページにアクセスします。デフォルトでは、ブラウザーは「**アプリケーション**」タブを表示します。
5. 「**My App**」リンクをクリックして、ファイルがディレクトリー・リストに含まれていることを確認します。 **index.html** リンクをクリックして、ホーム・ページを表示します。 DB2 Alphablox が IBM WebSphere Application Server または BEA WebLogic サーバーにインストールされている場合、このステップを実行する前にディレクトリー・リストを使用可能にする必要があります。DB2 Alphablox が Apache Tomcat にインストールされている場合、追加のステップは必要ありません。HTTP 403 エラー (ディレクトリー・リストがデフォルトで許可されていない) が表示される場合、ディレクトリー・リストを許可するようにサーバーを変更するか、またはファイル名 `myapp.html` を URL の末尾に追加することができます (たとえば、http://localhost:9080/MyApp/myapp.html)。完全修飾されたリンク・アドレスを使用すると、新規ページが表示されます。

これで、次に作成する 2 つの分析ビューへのリンクを持つホーム・ページが作成されました。

デフォルトのホーム・ページを設定する

前のステップで、MyApp アプリケーション・ディレクトリー・リストのファイル名リンクをクリックするか、または URL にホーム・ページ・ファイル index.html を指定して、ホーム・ページに直接アクセスしました。このステップでは、アプリケーションにデフォルト・ホーム・ページを指定します。

デフォルト・アプリケーション・ホーム・ページを設定するには、以下のようにします。

1. ブラウザーを開いて、DB2 Alphablox ホーム・ページにアクセスします。
2. 「**管理**」タブをクリックします。
3. 「**アプリケーション**」リンクをクリックします。
4. アプリケーション・リストで **MyApp** を選択し、リストの下の「**編集**」ボタンをクリックします。
5. MyApp の「Edit Application」ページで、`index.html` を「**ホーム URL (Home URL)**」フィールドに入力します。
6. 「**保管**」ボタンをクリックして、変更を保管します。
7. メインの「アプリケーション」タブ (左上フォルダーのタブ) をクリックして、「アプリケーション」ページに戻ります。
8. 「**My App**」アプリケーション名をクリックします。 これで、アプリケーションは定義されたホーム・ページを直接開きます。

これでホーム・ページが定義されたので、サーバーはアプリケーションの開始点を認識し、ユーザーが URL でホーム・ページを指定しなくても、アプリケーションにこのページで開くように指示します。このようにして、ユーザーがブラウザーのアドレス・バーに http://*yourServerName*/MyApp/ を入力すると、アプリケーションは定義されたホーム・ページ index.html を自動的に開きます。

この後のトピックでは、DB2 Alphablox Query Builder アプリケーションを使用して生成された簡単な Blox タグを使用して、DB2 Alphablox 分析ビューの作成を始めます。

最初の分析ビューを作成する

この作業では、グリッドとチャートを PresentBlox コンポーネントに結合して表示する DB2 Alphablox 分析ビューを作成します。

前提条件: 「*開発者用ガイド*」の序文のセクションをお読みになり、DB2 Alphablox とそのコンポーネントについて学習してください。ここでは、DB2 Alphablox 機能について、および JSP テクノロジーを使用した分析アプリケーションの構築に Blox カスタム・タグ・ライブラリーを使用する方法についての基本的な理解が得られます。

作成する最初の分析ビューでは、DB2 Alphablox 照会ビルダー・アプリケーションを使用して、PresentBlox コンポーネントについて定義した JSP カスタム・タグを生成します。このコンポーネントは、データを提示するために使用される最も一般的な Blox コンポーネントです。 PresentBlox コンポーネントには、グリッドおよびチャート・ビュー、データ・レイアウト・パネル、ツールバー、およびその他のネストされたコンポーネントを含めることができます。グリッドおよびチャート・ビューは、同じデータの同期化された代替ビューを表示します。

照会ビルダーを使用して最初の分析ビューを作成するには、以下のようにします。

1. ブラウザーを開いて、DB2 Alphablox 管理ページにアクセスします。デフォルトでは、「アプリケーション」タブが表示されます。
2. DB2 Alphablox 照会ビルダーをクリックして、アプリケーションを開きます。
3. 「**接続の設定**」ボタンをクリックします。「データベース接続」ウィンドウが表示されます。
4. qcc2003-acs データ・ソースを選択してから、「**接続**」ボタンを押します。ウィンドウが閉じると、「照会ビルダー」ページの「データベース状況」セクションに「接続済み」と表示されるはずです。

   注: 「**スキーマ**」、「**カタログ**」、「**ユーザー名**」、および「**パスワード**」フィールドには値を入力する必要はありません。これらは選択したデータ・ソースのデータ・ソース定義から自動的に取得されます。また、「**デフォルト照会の実行**」チェック・ボックスは選択しないでください。
5. ページで、「状況」セクションの下にある「照会」ページを見つけます。以下の重要な部分に注意してください。
   - このチュートリアルでは、DB2 Alphablox キューブ (qcc2003-acs) を使用します。「照会」セクション・ヘッダーのすぐ下に、使用可能な (現在実行中の) キューブのリストが表示されます。 qcc2003-acs キューブがリストされているはずです。
   - このデータ・ソースに関連したデフォルトの照会ストリングがテキスト・ウィンドウに表示されます。データ・ソースは DB2 Alphablox キューブであるため、アルファベット順にリストされた最初のキューブが照会ステートメントに追加されます (たとえば、″select from [*datasourceName*].″)。

     注: DB2 Alphablox Cube Server で接続プール機能が使用可能になっていると、「**デフォルト照会の取得**」ボタンを使用して、DB2 Alphablox キューブのデフォルト照会を取得することはできません。
   - ページの下部にある PresentBlox にデータが表示されている場合、標準のユーザー・インターフェースを使用して、軸の交換、ドリルアップおよびドリルダウン、軸の間でのディメンションの移動などを行います。
   - 照会の結果を表示するには、「照会の実行」ボタンをクリックします。結果セットは、ページの下部にある PresentBlox に表示されます。
6. デフォルト照会の大括弧の中に `qcc2003-acs` を入力し (たとえば、`select from [qcc2003-acs]`)、「**照会の実行**」ボタンを押します。

   PresentBlox セクションに、照会に基づいたデフォルト・ビューが表示されます。

7. PresentBlox ユーザー・インターフェースを使用して、新規のビューを作成します。

   たとえば、左側の「データ・レイアウト」パネルで「製品 (Products)」ディメンションを「行」軸にドラッグできます。次に、「時間 (カレンダー) (Time (Calendar))」ディメンションを「列」軸にドラッグできます。さらに、「メジャー (Measures)」ディメンションを「ページ (Page)」軸にドラッグすることもできます。その結果、表示域の上部の「ページ (Page)」パネルに「メジャー (Measures)」メニューが表示され、ページ・フィルターが表示されます。他の 2 つのディメンションを選んで、それらを「ページ (Page)」軸に追加し、3 つのページ・フィルターが画面上部に表示されるようにします。 DB2 Alphablox ユーザー・インターフェースに慣れていない場合は、PresentBlox ツールバーから「**ヘルプ**」を選択して、インターフェースの使用方法を学習してください。
8. 新規のビューを作成し終えたら、「照会」セクションに表示されている照会ステートメントに注目します。照会が動的に生成されていない場合 (「照会の自動更新」オプションがチェックされていないとき)、「**現行照会の取得**」ボタンをクリックして、現行照会を取得できます。テキスト・ボックスに、現行のデータ・ビューの開発に必要な照会ステートメントが表示されます。照会ステートメントを「照会」フィールドからコピーして貼り付けて、DB2 Alphablox アプリケーションの開発に使用することができます。
9. 「**Blox タグの生成**」ボタンをクリックします。 PresentBlox レイアウトおよび結果セットを複製するための、タグ (およびネストされたタグ) を表示したダイアログが開きます。
10. テキスト・エディターで新規ファイルを開き、以下のテキストをファイルに追加してから、ファイルを PresentBloxView.jsp として保管します。次のトピックでは、このページ上のコードの多くの重要なエレメントについて説明します。

```
<%@ 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. 保管したファイル (PresentBloxView.jsp) をサーバー上の MyApp ディレクトリーにコピーし、MyApp ホーム・ページから Simple PresentBlox View ページへのリンクをクリックして、新規の分析ビューをテストします。あるいは、ブラウザー・アドレスに URL を直接入力することもできます (たとえば、http://localhost:9080/MyApp/PresentBloxView.jsp)。表示されるビューは、DB2 Alphablox 照会ビルダーで扱った PresentBlox に似ているはずです。

Simple PresentBlox View ページ (PresentBloxView.jsp) の完全なコード例を以下に示します。

```
<%@ 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>
```

上記のコード例では、読みやすくするための改行が照会ステートメントに挿入されています。このページを正しく実行するには、(上記の query および selectableSlicerDimensions 属性などの) 属性値に含まれている改行を削除する必要があります。 DB2 Alphablox 照会ビルダーで生成されたコードでは、組み込まれるタグの多くは不要です。ただし、ここにあるすべてのタグを見れば、Blox タグが親タグおよびネストされたタグとしてどのように動作するのかを洞察することができま

す。また、照会ビルダーの PresentBlox で操作した結果として追加されたタグ属性の使用法を見ることもできます。 Blox タグ、タグ属性、および DB2 Alphablox アプリケーションの開発方法を詳しく学習するには、DB2 Alphablox インフォメーション・センターを参照できます。

最初の分析ビュー・ページの構造

このトピックでは、前のトピックで作成した最初の分析ビュー (PresentBloxView.jsp) の構造を要約します。

前のトピックで作成したページの主要部分のいくつかについて、簡潔な概要を以下に示します。コードに関してより詳しく理解するには、DB2 Alphablox インフォメーション・センターで興味のあるトピックを参照する必要があります。

PresentBloxView.jsp ページは、次の行で始まります。

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

この行は JSP taglib ディレクティブで、Blox タグ・ライブラリーを使用する予定であることをサーバーに知らせます。 `uri` は、タグ・ライブラリー記述子ファイルへのポインターです。 `blox` として定義された `prefix` 値は、このページ上で `blox` で始まるタグを探し、Blox タグ・ライブラリーを使用して、タグ・ライブラリー記述子ファイルで定義された内容を処理するようにサーバーに指示します。

taglib ディレクティブのすぐ下で、PresentBlox コンポーネントは以下のタグとそれらのタグ属性で指定されます。

```
<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>
```

<blox:present> タグは、id 属性値を queryBuilder4_present として、PresentBlox がここに表示されるように指定します。スクリプト記述の目的では、id 属性によって、この特定の Blox を識別できます。 <blox:present> タグでは、visible 属性が false に設定されて組み込まれていることに注意してください。 false に設定すると、ページ内のコンパイラーが <blox:display> タグを見つけるまで、PresentBlox はレンダリングされません。

タグを使用して定義された Blox ごとに多くの属性を使用できます。しかし、デフォルト値とは異なる属性値を指定するのでなければ、それらをタグに含める必要はありません。この例では、height 値は 500 で、width 値は 100% です。高さと幅はピクセルまたはパーセントで定義できます。

ネストされた DataBlox が組み込まれます。dataSourceName 属性は qc2003-acs に定義されます。データ・ソースを指定しない場合、ページが表示されるときに「使用可能なデータはありません」というメッセージが表示されます。 query 属性には、DB2 Alphablox 照会ビルダーで生成された MDX 照会ステートメントが含まれます。 DB2 Alphablox Cube Server で使用するMDX 照会言語を理解していると、ステートメントをより短くし、単純な照会にすることができます。ただし、ここに載せられている照会ステートメントは完全なものなので、照会ビルダー・アプリケーションで作成したビューを生成します。

ページの <head> セクションには、ページがレンダリングされる前に重要なコードをページに追加するための特殊な Blox タグが含まれています。

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

このタグは、必要な HTML、JavaScript™ および CSS コードをページのヘッド・セクションに自動的に追加するために、DB2 Alphablox によって使用されます。ページがサーバーによってレンダリングされる場合は、このタグにより、定義済み HTML テーマへの CSS リンクと、キャッシングを防ぐためのメタ・タグが追加されます。このタグは、Blox コンポーネントを使用するすべての JSP ページに忘れずに入力してください。このタグが含まれていない場合、ページは正しくレンダリングしません。

ページの <body> セクションには、次の行があります。

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

この Blox タグにより、上で指定した PresentBlox はここでレンダリングされます。これにより、Blox コンポーネントまたはそのネストされたコンポーネントを 1 個所で変更することができます。さらに、その他の Blox タグおよび JSP スクリプトレットで Blox コンポーネントの動作を制御することもできます。複雑な JSP ファイルでは、DB2 Alphablox に精通しているならば、DataBlox コンポーネントを PresentBlox タグから分離させ、Blox コンポーネントが Web ブラウザーでレンダリングされる前に、その Blox コンポーネントとそれらの動作を変更する JSP スクリプトレットを追加することを選択できます。

要約すると、最も重要な次の Blox タグがこのページで使用されました。<blox:header/>、<blox:present>、および <blox:display>。これらの 3 つのタグ、およびネストされたタグは、ページ上に Java™ コードを必要とすることなく分析ビューを指定します。表示ロジックの複雑さは、これらの Blox タグによって管理されることになります。ネストされた Blox を追加し、属性値を変更することにより、ビジネスの要件に基づいてビューをカスタマイズできます。

より複雑なビューおよびアプリケーションの場合、追加の JavaBeans コンポーネントおよび Java コードが必要になることがあります。

2 番目の分析ビューを作成する

この作業では、このチュートリアルの最初の分析ビューからのコードに基づいて、ページ・フィルター付きのチャートを表示する分析ビューを作成します。

多くのアプリケーションでは、データはグリッドまたはチャートだけを使用して表現されます。 PresentBlox コンポーネントは、GridBlox と ChartBlox の両方を、PresentBlox コンポーネント内で同時に表示するネストされたコンポーネントとして結合しますが、表示したくないネストされたコンポーネントを使用不可にするか、または非表示にすることができます。 PresentBlox コンポーネントを使用してページ・フィルター付きのチャート・ビューを作成することは比較的簡単です。これは、ネストされた Blox タグのいくつかでタグ属性を変更することによって行います。

この作業では、最初の分析ビューからの PresentBlox タグを再利用して、2 番目のカスタマイズ済み分析ビューを作成します。

ページ・フィルター付きのチャートを表示する分析ビューを作成するには、以下のようにします。

1. 以前に作成した PresentBloxView.jsp ファイルをエディターで開いて、それを CustomizedChartBlox.jsp として保管します。
2. 以下にリストされた PresentBlox タグのネストされたコンポーネントで、visible 属性をそれぞれのタグに追加し、値を false に設定します。 この属性を追加すると、これらのネストされたコンポーネントは表示されなくなりますが、ページ・フィルター・バー付きのチャート・ビューは残ります。

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

3. labelPlacement 属性を追加し、値を top に設定して、PageBlox タグ (<blox:page>) を変更します。 デフォルトでは、ページ・フィルターのラベルはメニューの左側に表示されます。この属性を追加し、それを top に設定すると、デフォルトの動作がオーバーライドされ、ラベルはページ・フィルターの上部に配置されます。

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

4. <blox:present> タグの id 属性の名前を、chartview に変更します。 エラーを防ぐために、ページに表示される Blox コンポーネント、またはレンダリングされる Blox コンポーネントには、それぞれ固有の ID が必要です。
5. 変更を保管し、ファイルをサーバー上の MyApp ディレクトリーに追加します。

6. Web ブラウザーを開いて、新規ページをテストします。これを行うには、ホーム・ページからリンクをクリックするか、URL (たとえば、http://localhost:9080/MyApp/CustomizedChartView.jsp) を使用して、ページに直接アクセスします。

これで、作業を開始するのに役立つ DB2 Alphablox 照会ビルダー・アプリケーションを使用して、2 つの分析ビューを作成できました。また、既存のタグの属性を変更または追加することによって、Blox とそれらのネストされたコンポーネントの外観および動作を制御できることも学習しました。

要約

チュートリアルの作業をすべて完了したら、DB2 Alphablox 管理ページ、DB2 Alphablox 照会ビルダー・アプリケーション、および Blox タグを使用して、基本的な DB2 Alphablox アプリケーションを構築する方法を習得できたことになります。「*開発者用リファレンス*」をすでに一読している場合は、JSP ページ上で Blox を定義および操作するために使用可能なプロパティーとメソッドがたくさんあることにお気付きでしょう。それでは、次に何をしたらよいでしょうか。

企業データを DB2 ビューで見たい場合は、アプリケーションを最初から作成し、分析ビューをアプリケーションに追加する方法をすでに知っています。何かを早く見たいのであれば、MyApp アプリケーションに単純な変更をいくつか加えることができます。 qcc2003-acs データ・ソースを使用する代わりに、企業データベースからデータを表示するには、企業データ・ソースを指すデータ・ソースを作成し、`dataSourceName` 属性を新規に定義したデータ・ソースを指すように変更してから、適切な `query` 属性を追加することができます。 DB2 Alphablox 照会ビルダーを使用すると、しばしば、使用可能な照会ステートメントおよび Blox タグをすばやく作成することができます。データ・ソースの作成について詳しくは、「*管理者用ガイド*」を参照してください。適切な照会を作成する方法、および DB2 Alphablox アプリケーションの構築方法の詳細を学習するには、「*開発者用ガイド*」の『データの取り出し』および「*開発者用リファレンス*」の『DataBlox』のセクションを参照してください。

第 2 章 チュートリアル: Blox コンポーネントで最初のポートレットを構築する

このチュートリアルでは、Blox コンポーネントをポートレットに追加する方法を学習します。以下の事柄を行います。

1. 事前構築されたサンプル・ポートレットをインストールします。このステップを使用して、Blox コンポーネントをポートレット JSP ページに追加する方法と、それがポータル・ページにどのように表示されるかをすばやく理解することができます。
2. GridBlox を含む独自の JSP ページを作成します。

このチュートリアルの作業では、一般的なポートレット開発について詳しく説明していません。このチュートリアルは DB2 Alphablox 特有のタスクに焦点を当てており、ポータル環境およびポートレットの開発の一般的な概念にある程度通じていることを前提としています。

前提条件

- DB2 Alphablox が WebSphere Portal Version 5.1 サーバーにインストールされていなければならない。インストールについて詳しくは、「*インストール・ガイド*」を参照してください。
- WebSphere Portal サーバーが開始済みでなければならない。
- WebSphere Portal サーバーに対する管理アクセスがなければならない。
- WebSphere Portal の管理機能およびユーザー・インターフェースに精通していなければならない。
- Java および JSP の基礎知識がなければならない。
- JSP エディターがインストール済みでなければならない。

このチュートリアルには任意の JSP エディターまたはテキスト・エディターを使用できますが、独自のポートレットを開発するときには、WebSphere Portal で推奨される開発ツール (Rational® Application Developer など) を使用してください。

このチュートリアルでは、DB2 Alphablox とともにインストールされた定義済みデータ・ソースを使用します。このデータ・ソースを使用して、基本的なアプリケーションをすばやく開発できます。このチュートリアル用のカスタム・データ・ソースの構成について心配する必要はありません。

サンプル・ポートレットのインストール

Blox をポートレットに追加する方法を学習するには、DB2 Alphablox で提供されるサンプル・ポートレットをインストールし、それをポータル・ページにロードするのが最適です。これにより、JSP コードの基本構造を調べて、それをポータルでの出力と突き合わせることができます。

DB2 Alphablox で提供されるサンプル・ポートレットをインストールします。

1. ブラウザーを開き、管理ユーザーとしてポータルにログインします (URL は http://<yourPortalServer>:<port>/wps/portal の形式です)。
2. 「**管理**」ボタンをクリックします。
3. 「ポートレット管理 (Portlet Management)」セクションで、「**Web モジュール (Web Modules)**」をクリックします。「Web モジュールの管理 (Manage Web Modules)」ページが右側に表示されます。
4. 「**インストール (Install)**」をクリックします。インストールする Web モジュールを入力するように指示されます。
5. 「**ブラウズ**」ボタンをクリックし、DB2 Alphablox インストール・ディレクトリーの下の installableApps ディレクトリーにナビゲートします。
6. AlphabloxSamplePortlets.war を選択して、「**次へ**」をクリックします。 DB2 Alphablox JSP Page Sample Portlet というポートレットを使用する DB2 Alphablox Sample Portlets アプリケーションが、ポートレット・アプリケーション・テーブルに表示されます。
7. 「**完了**」をクリックします。

DB2 Alphablox Sample Portlets アプリケーションとそこに含まれるポートレットがインストールされました。 WebSphere Portal インストールの下の installedApps ディレクトリーを調べます。新規に作成されたディレクトリーの名前は DB2 Alpha で始まり、動的に生成されたポートレット ID (パターンは _PA_*x_x_xx*.ear) で終わります。

サンプル・ポートレットの実行

サンプル・ポートレットをポータル・ページで実行するには、以下のようにします。

1. ポータル・ページに進みます。
2. 既存のページを作成または編集します。このサンプル・ポートレットをテストするための新規ページを作成するか、または既存のページをクリックしてページを編集することもできます。
3. ポータル・レイアウト・ページで「**ポートレットの追加 (Add Portlets)**」ボタンのいずれかをクリックします。
4. 「検索」フィールドに DB2 を入力して、「**検索**」を押します。「**DB2 Alphablox JSP ページ・サンプル・ポートレット (DB2 Alphablox JSP Page Sample Portlet)**」チェック・ボックスが表示されます。
5. チェック・ボックスを選択して、「**OK**」をクリックします。
6. 「**完了**」をクリックします。

ポータル・ページが最新表示されると、ポータル・ページに PresentBlox が表示されます。この PresentBlox には以下のものがあります。

- 上部にメニュー・バー
- メニュー・バーの下に 2 つのツールバー・パネル
- 左側にデータ・レイアウト・パネル (いろいろな軸方向にディメンションを移動させる)
- 表形式でデータを表示するグリッド
- 右側に三次元の棒グラフ

present.jsp ファイルはデフォルトのページで、サンプルの中にある BloxJSPPagePortlet サーブレットで指定されているとおりにロードするためのものです。このサンプル・サーブレットのソース・コードは WEB-INF/src/ ディレクトリーで入手可能です。

これで、サンプル・ポートレットを正常にインストールし、Blox を持つポートレットをポータル・ページに追加しました。次の作業は、この JSP ページ内のコード構造を調べることです。

Blox コンポーネントを含むポートレット JSP ページの構造を調べる

この作業では、Blox を含む JSP ページのコード構造を調べます。すべての JSP ページは同じキー・エレメントを持っている必要があります。

JSP ファイルを開くには、以下のようにします。

1. WebSphere Portal インストールの下の installedApps/ ディレクトリーにナビゲートし、DB2 Alpha で始まる新しく作成したアプリケーション・フォルダーに位置指定します。
2. PA_*x_x_xx*.war/jsp/html/ ディレクトリーにナビゲートします。
3. JSP または Java エディターで present.jsp を開きます。
4. 以下のコードを調べて、キー・エレメントに注目します。

```
<%@ 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>
```

このコードのブロックには 6 つのキー・エレメントが含まれています。

1. 最初の行は、出力が HTML であることをブラウザーに知らせるものです。

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

2. 次のコードのセットは、このページで使用される 2 つの JSP タグ・ライブラリーを指定します。

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

   uri は、タグ・ライブラリー記述子ファイルが置かれているディレクトリー位置へのポインターです。 prefix 値 (blox および portletAPI として定義) は、サーバーに次のことを指示します。

- このページで、blox で始まるタグを検索し、Blox タグ・ライブラリーを使用して、タグ・ライブラリー記述子ファイルで定義された内容を処理する。
- このページで、portletAPI で始まるタグを検索し、ポートレット・タグ・ライブラリーを使用して、タグ・ライブラリー記述子ファイルで定義された内容を処理する。

3. 次に、ポートレット初期化タグが追加されます。

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

このタグにより、PortletRequest、PortletResponse、および PortletConfig オブジェクトにアクセスできます。 PortletResponse を使用して、encodeNamespace() メソッドを呼び出して、同じページで実行する他のポートレット上の他のオブジェクトと Blox の名前が競合していないことを確認できます。

4. 次のタグでは、ページに追加する Blox のネーム・スペースをエンコードします。

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

これにより、Blox を後で作成し、Blox にこの固有の名前を割り当てることができます。

5. コードの次のブロックを指定すると、Blox レンダリングとサーバー・クライアント間の通信に必要な Blox ヘッダー・タグが追加されます。

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

このタグは、必要な HTML、JavaScript、および CSS コードをページのヘッド・セクションに自動的に追加するために、DB2 Alphablox によって使用されます。ページがサーバーによってレンダリングされる場合は、このタグにより、定義済み HTML テーマへの CSS リンクと、キャッシングを防ぐためのメタ・タグが含められます。このタグを Blox コンポーネントを含むすべての JSP ページに追加する必要があります。そうしないと、コンポーネントが正しくレンダリングしません。

6. Blox タグ・ライブラリーで提供されるタグを使用して、PresentBlox を追加します。

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

- このコードにより、presentBlox が id で、xx_x_x_xxx_presentBlox が bloxName の PresentBlox が追加されます。これがネーム・スペースのエンコードの結果です。
- この Blox の幅は 800 ピクセル、高さは 400 ピクセル (デフォルト) です。
- ネストされた DataBlox が組み込まれます。dataSourceName タグ属性は canned に設定されています。

PresentBlox の id タグ属性が必要です。これは、JSP ページで使用する Java スクリプト名を指定します。 bloxName 属性は、サーバー上のオブジェクト名を指定します。エンコードされた bloxName により、インスタンスはサーバー上で固有になります。

注: データ・ソースを指定しない場合、通常、グリッドに「**使用可能なデータはありません**」というメッセージが表示されます。 canned データ・ソースはインストール時に事前定義され、少量のサンプル・データが含まれています。それは実外部データベースのインストールおよび構成を必要としません。また、トラブルシューティングや学習に使用できます。照会を指定する必要がないため、コード内に `query` 属性はありません。

このページには、最も外部の `<html>` タグがありません。なぜなら、この JSP ページは他のポートレットとともにポータル・ページに表示されるからです。追加の `<html>` または `<body>` タグは必要ありません。

JSP 構造を調べて、ポートレット JSP ページに含める不可欠なコードについて学習したので、次の作業では、別の Blox で新規の JSP ページを作成し、そのページがご使用のポータル・ページに適するように、いくつかの共通の Blox 属性を指定しましょう。

Blox コンポーネントを含む独自のポートレット JSP ページを作成する

この作業では、GridBlox を含む新規の JSP ページを作成し、そのプロパティーをいくつか設定します。その目的は、標準的なポータル・ページにうまく適合する GridBlox を作成しながら、一般的な Blox タグ構成に精通することです。

GridBlox のデフォルトのサイズは 400x400 ピクセルです。また、メニュー・バーとツールバーが付属しています。メニュー・バーとツールバーの表示をオフにして、グリッドの高さが 100 ピクセルだけを占めるように、共通に使用される GridBlox プロパティーのいくつかを設定します。これらを行うには、次の GridBlox プロパティーを設定します。

- `height` : 100 ピクセルに設定
- `menubarVisible`: false に設定
- `toolbarVisible`: false に設定

13 ページの『第 2 章 チュートリアル: Blox コンポーネントで最初のポートレットを構築する』で指定した要件を満たしており、13 ページの『サンプル・ポートレットのインストール』のセクションの説明に従ってサンプル・ポートレット・アプリケーションをインストールしてあることを確認します。

以下のステップに従います。

1. PresentBlox が表示されているブラウザー・ウィンドウで、ポートレットの編集ボタン (鉛筆のアイコンの付いたボタン) をクリックします。選択ドロップ・リストが表示されます。
2. ドロップ・リストで、「Grid Blox」を選択し、「**OK**」をクリックします。

   ページを最新表示すると、400x400 の GridBlox が表示されます。この GridBlox は、デフォルトではメニュー・バーとツールバーの表示がオンになっています。サイズを 400x100 に変更し、メニュー・バーとツールバーの表示をオフにします。
3. 前にインストールした DB2 Alphablox Sample Portlets アプリケーションの下の `PA_x_x_xx.war/jsp/html/` ディレクトリーにナビゲートします。

4. JSP エディターで grid.jsp を開きます。このページは、以下の点を除いて、present.jsp とほぼ同一です。
   - <blox:present> タグが <blox:grid> となっており、id の値が異なる。

     ```
     <blox:grid id="gridBlox" bloxName="<%= bloxName %>" width="400">
       <blox:data dataSourceName="canned" />
     </blox:grid>
     ```

   - bloxName 値が異なる。

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

5. height 属性を追加し、その値を 100 に設定して、この GridBlox の高さを 100 ピクセルに設定します。

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

6. menubarVisible と toolbarVisible 属性を false に設定して、上部にあるメニュー・バーとツールバーの表示をオフにします。

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

   大/小文字を含めて、属性名が正しく入力されているかを確認します (どちらの属性名も大文字の ″V″ を使用します)。また、末尾の不等号括弧 (″>″) の前に属性が追加されているかを確認します。

7. ネーム・スペースを myFirstGrid に変更します。

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

   この GridBlox のネーム・スペースを変更すると、この JSP をポータルにロードするときに、行った変更が確実に反映されます。このページは以前にロードしてあるため、このセッション用のサーバーで実行する、この GridBlox のインスタンスがすでに存在します。ネーム・スペースを変更しないと、ページを最新表示しても、行った変更は反映されません。ネーム・スペースを変更することが、開発環境で変更をテストするための速い方法です。あるいは、新規のブラウザー・ウィンドウを開いて、新規オブジェクトが新規セッション用のサーバーで作成されるようにすることもできます。

8. ファイルを保管します。

これで、このファイルを WebSphere Portal でテストする準備ができました。

行った変更をテストするには、以下のようにします。

1. ポータル・ページに戻ります。
2. ブラウザーの「リフレッシュ」ボタンをクリックして、ページを再ロードします。

メニュー・バーもツールバーも表示されていない 400x100 の GridBlox が表示されます。

注: この JSP 選択ドロップ・リストは WebSphere Portal の組み込み機能ではありませんが、このサンプル・ポートレットによって作成されます。 WEB-INF/src/ ディレクトリーの edit.jsp ファイルと Java ソース・ファイルを調べます。

Blox コンポーネントを使用するポートレット・プロジェクトを作成する

ポートレット・プロジェクトを開発ツールで作成している場合、Blox コンポーネントに必要な AlphabloxServer サーブレット・マッピングとタグ・ライブラリー参照がプロジェクトの web.xml ファイルに追加され、DB2 Alphablox タグ・ライブラリー用の .tld ファイルがプロジェクトにコピーされていることを確認してください。

1. 以下の行を含むように、プロジェクトの web.xml ファイルを変更します。
   - サーブレット定義とサーブレット・マッピングについて:

     ```
     <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>
     ```

   - タグ・ライブラリー参照について:

     ```
     <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. DB2 Alphablox タグ・ライブラリー用の .tld ファイルをプロジェクトの WEB-INF/tlds/ ディレクトリーにコピーします。これらのファイルは、

   ```
   <db2alphablox_dir>/bin/
   ```

   の下にあります。ここで、<db2alphablox_dir> は DB2 Alphablox がインストールされているディレクトリーです。

Rational Application Developer を使用してポートレット・プロジェクトを構成する

WebSphere Portal では、ポートレット開発に Rational Application Developer を使用することが推奨されています。特に、WebSphere Portal バージョン 5.1 には Rational Application Developer バージョン 6.0 が必要です。 Rational Application Developer には、ポートレット・プロジェクトのセットアップおよび作成をガイドするウィザードが用意されています。選択に基づいて、適切な構造およびデプロイメント記述子ファイルがセットアップされ、ご使用のコントローラーおよびポートレット・ビュー用の JSP ページに必要な Java クラスが自動的に作成されます。「新規のポートレット・プロジェクト (New Portlet Project)」ウィザードに従いながら、以下の事柄が正しく構成されていることを確認してください。

- ターゲット・サーバーに「WebSphere Portal v5.1 スタブ」を選択します。これは、「**拡張表示 (Show Advanced >>)**」ボタンをクリックすると、ウィザードの最初の画面で指定されます。
- DB2 Alphablox のサーブレット・マッピングを `WebContent/WEB-INF/` の下に作成された `web.xml` ファイルに追加します。これは、プロジェクトの `web.xml` で以下のサーブレット定義およびマッピング・コードを入力して行います。

```
<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>
```

- Alphablox タグ・ライブラリーをプロジェクトの `web.xml` に追加します。以下のステップに従います。
  1. `web.xml` を開いて、「**変数**」タブをクリックします。
  2. 下部にある「タグ・ライブラリー参照 (Tag Libraries References)」セクションにスクロールします。
  3. 「**追加**」をクリックします。
  4. **URL** と**ロケーション**については、以下の表に基づいて値を入力します。

URL	ロケーション
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

URL とロケーションのペアごとに、「**完了**」をクリックしてタグ・ライブラリー参照を追加してから、「**追加**」をクリックして次のペアを追加します。

注: 様々なタグ・ライブラリーとその使用法について詳しくは、『Using JavaServer Pages and Blox』を参照してください。

最後に、DB2 Alphablox タグ・ライブラリーの .tld ファイルをプロジェクトの WEB-INF/tlds/ ディレクトリーに忘れずにコピーしてください。これらのファイルは、

<db2alphablox_dir>/bin/

の下にあります。ここで、<db2alphablox_dir> は DB2 Alphablox がインストールされているディレクトリーです。

次のステップ

Blox コンポーネントを含むポートレットの作成方法と必須のコード構造について学習しました。データベースのデータを表示するには、以下の事柄を行う必要があります。

1. データ・ソースを指す DB2 Alphablox に対して新規のデータ・ソースを定義します。
2. DataBlox の dataSourceName 属性を、定義済みのデータ・ソースを指すように変更します。
3. 照会ストリングを持つ query 属性を DataBlox に追加します。

データ・ソースの作成について詳しくは、「*管理者用ガイド*」を参照してください。適切な照会を作成する方法を学習するには、「開発者用ガイド」の『データの取り出し』および「*開発者用リファレンス*」の『DataBlox』のセクションを参照してください。

独自のデータを Blox ビューに表示したら、Blox タグを使用して設定できる多数のプロパティーを探索することもできます。

ポートレット開発のヒント

DB2 Alphablox の資料にある情報を使用してポートレット開発を続ける際には、知っておく必要のあるいくつかのポートレット固有のトピック、概念、および一般的なガイドラインがあります。以下のリストは、それらの一般的な開発ガイドラインおよび資料中の関心のある特定のセクションを指すポインターをまとめています。

- ポートレットのネーム・スペースを使用して、常に Blox の名前をエンコードします。ネーム・スペースにより、現在の J2EE セッションに固有の Blox 名が確保されます。
- ピクセルを使用して Blox の幅と高さを常に設定します。ポータル環境の外部では、幅と高さを ″50%″ や ″100%″ などのパーセント値に設定できます。しかし、複数のポートレットがページ上に共存するため、パーセット値はポータル環境では無効です。
- ポートレット内のリソースを呼び出すために、相対 URL を使用しないでください。 URL は、PortletResponse クラスの encodeURL() メソッドを使用してエンコードする必要があります。資料で相対 URL の使用を示している箇所では、常に URL をエンコードする必要があります。

- Blox ポートレットがポータル・テーマに類似したテーマを使用するようにするには、ポータル・テーマ・ユーティリティーを使用してください。このユーティリティーは、以下の 2 つのソースから入手できます。
  - DB2 Alphablox ホーム・ページ。

    「**管理**」タブの下で、「**一般**」リンクをクリックします。「ポータル (Portal)」セクションにユーティリティーがリストされます。詳しくは、「*管理者用ガイド*」とオンライン・ヘルプを参照してください。
  - `AlphabloxAdminPortlets.war` ファイル。

    これはポータル・テーマ・ユーティリティーのポートレット・バージョンです。このポートレットをインストールすると、DB2 Alphablox ホーム・ページに単独でログインしなくても、このポートレットをポータルから実行できます。

  このユーティリティーは、ポータル環境のスタイルと DB2 Alphablox のスタイルを 1 つに結合して、Blox がページ上の他のポートレットと同様の色とフォントで表示されるようにします。
- 計画の段階で考慮する必要のある問題については、「*開発者用ガイド*」の『Blox Portlet Tag Library』のトピックを調べてください。
- DB2 Alphablox には強力な Blox UI モデルがあります。 UI モデルには ClientLink オブジェクトが含まれており、これにより、ページ上の Blox コンポーネントがクリックされると、指定した URL がロードされます。ポートレット開発には、Blox ポートレット・タグ・ライブラリーを使用して ClientLink を作成します。ページが最新表示された後でリンクが不整合にならないように、タグは URL 形成と処理を動的に扱います。詳しくは、「*開発者用ガイド*」の『ポートレット開発の計画』のトピックを参照してください。
- ポートレットおよびアクション URI については、Blox ポートレット・タグ・ライブラリーを使用して、ポートレット・リンクまたはアクション・リンクを作成します。次に、ポータルの Portlet API を使用して、アクション・リンクまたはポートレット・リンクを処理することができます。詳しくは、「*開発者用ガイド*」の『Blox Portlet Tag Library』のトピックを参照してください。

第 3 章 チュートリアル: Rational Developer ツールを使用してアプリケーションを構築する

Rational Developer ツールを DB2 Alphablox ツールキットとともに使用して、DB2 Alphablox コンポーネントおよび DB2 Alphablox Java API を使用するアプリケーションを、より短時間で開発することができます。

Rational Developer ツール (Rational Application Developer または Rational Web Developer) を DB2 Alphablox ツールキット (Eclipse ベースのプラグインのセット) とともに使用して、DB2 Alphablox テクノロジーを使用して構築されたアプリケーションを開発およびテストすることができます。このチュートリアルでは、Rational 統合開発環境を構成することにより、メソッドおよびタグ完結の使用を可能にし、DB2 Alphablox ツールキットで追加したカスタム機能強化を可能にすることについて説明します。

Rational Developer ツールを DB2 Alphablox ツールキットとともに構成して、DB2 Alphablox アプリケーションを開発してテストするには、以下のようにします。

開発環境を準備する

DB2 Alphablox ツールキットをインストールして使用を開始する前に、必要なソフトウェアがすべて整っていることを確認して、開発環境を準備してください。

開発環境を準備するには、以下のようにします。

1. Rational Developer ツール (Rational Application Developer か Rational Web Developer のいずれか) をワークステーションにインストールします。
2. 少なくとも Rational Application Developer または Rational Web Developer バージョン 6.0.0.1 を確保するように、必要な更新をインストールします。
3. WebSphere 5.1 統合テスト環境の使用を計画している場合、このオプションのインストールを組み込むように Rational Developer のバージョンを更新したことを確認します。

   注: Rational Developer ツールのデフォルト・インストールには、WebSphere 5.1 統合テスト環境は含まれません。

これで、このチュートリアルにある作業を実行する準備ができました。

DB2 Alphablox ツールキットのインストール

DB2 Alphablox ツールキットを Rational Developer ツールとともに使用すると、DB2 Alphablox コンテンツを含む Web ベース・アプリケーションの開発を簡単に開始できます。

DB2 Alphablox ツールキットでは、Rational Application Developer または Rational Web Developer バージョン 6.0.0.1 を使用する必要があります。

DB2 Alphablox ツールキットを Rational Developer ツールにインストールすると、Java メソッドおよび JSP カスタム・タグのためのコンテンツ完結の使用を可能にするために、たくさんのステップを手動で実行しなくてもすみます。また、DB2 Alphablox コンテンツを使用したアプリケーションの作成、および WebSphere サーバー・インスタンスの定義をガイドするために、カスタム・ウィザードや参考文献を使用することもできます。

DB2 Alphablox ツールキットを Rational Application Developer または Rational Web Developer にインストールするには、以下のようにします。

1. DB2 Alphablox インストール・ディスクを、ご使用のワークステーションの CD ドライブに挿入します。
2. plugin ディレクトリーで、UpdateSite という名前のサブディレクトリーを見つけます。
3. UpdateSite ディレクトリーをハード・ディスク上の便利な位置にコピーします。 たとえば、そのディレクトリーを C:¥DB2Alphablox¥UpdateSite にコピーします。
4. Rational Developer ツールを開始します。
5. メニュー・バーで、「**ヘルプ**」 > 「**ソフトウェア更新 (Software Updates)**」 > 「**検索およびインストール (Find and Install)**」を選択します。
6. 開いた「インストール (Install)」ウィンドウで、「**インストールする新規機能の検索 (Search for new features to install)**」オプションを選択してから、「**次へ**」をクリックします。
7. 「アクセスするサイトの更新 (Update sites to visit)」ウィンドウで、「**新規のローカル・サイト (New Local Site)**」ボタンをクリックし、UpdateSite ディレクトリーの位置をブラウズします。
8. 「**次へ**」をクリックし、**DB2 Alphablox ツールキット**機能を選択してから、「**次へ**」をもう一度クリックします。
9. 続くウィンドウで、ご使用条件を受諾し、DB2 Alphablox ツールキット機能がインストールされる位置を選択します。
10. インストールが完了したら、Rational Developer ツールを再始動します。

Rational Developer ツールを再始動した後で、DB2 Alphablox ツールキット機能が使用可能になります。

DB2 Alphablox を WebSphere 統合テスト環境にインストールする

この作業では、Rational Developer ツールの DB2 Alphablox を WebSphere 統合テスト環境にインストールします。

前提条件: サポートされる WebSphere 統合テスト環境を Rational Developer ツールにインストールする必要があります。 DB2 Alphablox インストール・ディスクにアクセスできる必要があります。「*DB2 Alphablox インストール・ガイド*」の『インストール前の作業』のセクションで説明されている、必要なプリインストールのステップを検討して実行してください。

DB2 Alphablox は、スタンドアロン WebSphere アプリケーション・サーバー、または Rational Developer ツール内で使用可能な WebSphere 統合テスト環境にインスト

ールすることができます。以下に略述されているステップの要約は、DB2 Alphablox を、Rational Developer ツール内で使用可能な WebSphere ランタイムにインストールするときに実行しなければならないインストールでの相違点を説明しています。

DB2 Alphablox を WebSphere 統合テスト環境にインストールするには、以下のようにします。

「*DB2 Alphablox インストール・ガイド*」の『インストール』のセクションにあるステップに従って、DB2 Alphablox をインストールします。ただし、以下の例外があります。

1. インストーラーの「WebSphere の構成」ウィンドウで、「**WebSphere ルート・ディレクトリー**」フィールドに、統合テスト環境サーバーとして使用する WebSphere ランタイムの位置を指定します。

   たとえば、DB2 Alphablox を WebSphere 6 統合テスト環境にデフォルト・インストールでインストールするには、base_v6 ディレクトリーへのパスを選択します。標準的な Rational Application Developer インストールでは、パスは以下のようになります。

   ```
   C:¥Program Files¥IBM¥Rational¥SDP¥6.0¥runtimes¥base_v6
   ```

2. 「WebSphere 設定 (WebSphere Settings)」ウィンドウで、WebSphere 管理者の名前とパスワードを指定します。 これらの入力値は WebSphere 統合テスト環境では使用されませんが、DB2 Alphablox インストーラーで必要とされます。

重要: 「*DB2 Alphablox インストール・ガイド*」で説明されているポストインストールのステップは実行しないでください。

DB2 Alphablox を WebSphere ランタイムにインストールした後、Rational Developer ツール内で DB2 Alphablox アプリケーションをテストするための WebSphere サーバー・インスタンスを作成することができます。

WebSphere サーバー・インスタンスを作成する

DB2 Alphablox コンテンツを含むアプリケーションまたは JSP ファイルを実行するには、必要な DB2 Alphablox サービスおよび Java クラスにアクセスできる WebSphere サーバー・インスタンスを作成する必要があります。

前提条件: Rational Developer ツールを構成します。 DB2 Alphablox ツールキットをインストールします。 DB2 Alphablox を WebSphere 統合テスト環境にインストールします。

DB2 Alphablox コンテンツを含むアプリケーションおよび JSP ファイルをテストするための WebSphere サーバー・インスタンスを作成するには、以下のようにします。

1. Rational Developer ツールを開きます。
2. 「**サーバー**」タブをクリックします (タブが表示されていない場合は、「**ウィンドウ**」> 「**ビューの表示 (Show View)**」 > 「**サーバー**」を選択します)。
3. ビュー・ウィンドウ内を右クリックして、「**新規**」>「**サーバー**」を選択します。 「新規サーバー」ウィンドウが開きます。

4. ホスト名として localhost を入力し、サーバー・タイプを選択します。「**次へ**」をクリックします。
5. サーバー・ポート番号を入力して、「**次へ**」をクリックします。
6. このサーバー・インスタンスで実行する使用可能なプロジェクトを、構成済みプロジェクトのリストに追加します。「**完了**」をクリックします。 新規のサーバー・インスタンスが「**サーバー**」ビューで開きます。

WebSphere 5.1 サーバー・インスタンスを作成した場合は、次のステップ、「WebSphere 5.1 サーバー置換変数を作成する」を実行する必要があります。WebSphere 6 サーバー・インスタンスの場合、「DB2 Alphablox 管理者グループにゲスト・ユーザーを追加する」に進みます。

WebSphere 5.1 サーバー置換変数を作成する

この作業では、WebSphere 5.1 サーバー・インスタンス用にサーバー置換変数を変更して、WebSphere 統合テスト環境内で DB2 Alphablox が正しく稼働するようにします。

1. 「**サーバー**」ビューを開いて、変更する WebSphere 5.1 サーバー・インスタンスをダブルクリックします。
2. サーバー・インスタンスについて「**構成**」タブをクリックします。
3. 「**管理コンソールを使用可能にする (Enable administration console)**」を選択し、「**ユニバーサル・テスト・クライアントを使用可能にする (Enable universal test client)**」を選択解除します。
4. サーバー・インスタンスについて「**変数**」タブをクリックします。 「**置換変数 (Substitution Variables)**」ウィンドウが開きます。
5. 「**追加**」ボタン (「**ノード設定 (Node Settings)**」リストの右側にある) を使用して 2 つの新規変数を追加します。
   a. WS_EAR_AlphabloxPlatform という名前の変数を追加し、値を $(APP_INSTALL_ROOT)/localhost/AlphabloxPlatform.ear に設定します。
   b. WS_EAR_AlphabloxStudio という名前の 2 番目の変数を追加し、値を $(APP_INSTALL_ROOT)/localhost/ApplicationStudio.ear に設定します。
6. 変更を保管します。

WebSphere 5.1 サーバー・インスタンスの場合は、「WebSphere 5.1 サーバー・インスタンスを構成する」のステップに従って、サーバー・インスタンスの変更を完了する必要があります。

WebSphere 5.1 サーバー・インスタンスを構成する

WebSphere 5.1 サーバー・インスタンスの場合、DB2 Alphablox アプリケーションおよびファイルを実行するためのサーバー・インスタンスを構成する必要があります。

前提条件: 必要な WebSphere 5.1 サーバー置換変数を作成します。

WebSphere 5.1 サーバー・インスタンスを構成するには、以下のようにします。

1. Rational Developer ツールの「**サーバー**」ビューを開いて、構成するサーバー・インスタンスを開始します。
2. サーバー・インスタンスを右クリックして、「**管理コンソールの実行 (Run Administrative Console)**」を選択します。
3. **ID** フィールドに値を入力しないでください。「**OK**」をクリックします。
4. 構成テーブルの下の「**OK**」ボタンをクリックします。
5. 「**エンタープライズ・アプリケーション (Enterprise Applications)**」ナビゲーション・ビューを開き、**ApplicationStudio** アプリケーション名をクリックします。
6. 「**アプリケーション・バイナリー (Application Binaries)**」を `$(WS_EAR_ApplicationStudio)` に設定します。
7. 「**バイナリーからメタデータを使用する (Use Metadata from Binaries)**」を選択します。
8. 構成テーブルの下の「**OK**」ボタンをクリックします。
9. WebSphere 管理コンソールで「**保管**」ボタンをクリックします。
10. 「マスター構成に保管する (Save to Master Configuration)」ウィンドウで、「**保管**」をクリックします。
11. WebSphere 管理コンソールで、「**アプリケーション**」>「**エンタープライズ・アプリケーション (Enterprise Applications)**」の下の AlphabloxPlatform および ApplicationStudio アプリケーションを開始します。

サーバー・インスタンスが構成されます。次に、「DB2 Alphablox 管理者グループにゲストを追加する」必要があります。

DB2 Alphablox 管理者グループにゲスト・ユーザーを追加する

ゲスト・ユーザーを DB2 Alphablox 管理者グループに追加すると、Rational Developer ツールを使用している間に、DB2 Alphablox 管理ページにアクセスすることができます。

前提条件 WebSphere サーバー・インスタンスを作成します。 WebSphere サーバー・インスタンスを構成します。

ゲスト・ユーザーを DB2 Alphablox 管理者グループに追加するには、以下のようにします。

重要: ゲスト・ユーザーには、実動 WebSphere サーバーに対する管理者権限を与えないでください。

1. 次の Telnet コマンドを使用して、DB2 Alphablox コンソールにログインします。`telnet localhost` *`portNumber`*。 ここで、*portNumber* は、DB2 Alphablox のインストール時に指定したポートです。
2. Telnet コンソール・プロンプトで、次の DB2 Alphablox コンソール・コマンドを入力します。 `set Administrators guest`。そして、Enter を押します。
3. Telnet コンソールで、`save` を入力してから、Enter を押します。
4. Telnet セッションを閉じます。

これで、WebSphere サーバー・インスタンスが使用できるようになりました。

DB2 Alphablox アプリケーションを作成する

Rational Developer で新規アプリケーションを作成するときには、アプリケーションを正しく実行するために必要な DB2 Alphablox コンテンツを追加する必要があります。

前提条件: DB2 Alphablox ツールキットを Rational Developer ツールにインストールします。

1. Rational Developer ツールのメニュー・バーで、「**ファイル**」>「**新規プロジェクト (New Project)**」を選択します。 「**新規プロジェクト (New Project)**」ウィザードが開きます。
2. **Web** オプションを展開して、「**動的 Web プロジェクト (Dynamic Web Project)**」を選択してから、「**次へ**」をクリックします。
3. プロジェクトの名前を入力して、「**拡張表示 (Show Advanced)**」ボタンをクリックします。 追加のオプションが表示されます。
4. 該当するサーブレット・バージョンとターゲット・サーバーを選択します。
5. 「**次へ**」をクリックします。 「機能オプション (Features options)」ウィンドウが開きます。
6. 「**DB2 Alphablox コンテンツ (DB2 Alphablox Content)**」オプションを選択してから、「**完了**」をクリックします。

これで、新規の Web アプリケーション・プロジェクトが DB2 Alphablox で使用可能になります。 DB2 Alphablox タグ・ライブラリーおよび Blox Java API が使用可能であり、アプリケーション記述子ファイル (web.xml) は、必要な DB2 Alphablox 情報を含むように変更されました。これで、DB2 Alphablox コンテンツを含む JSP ファイルをプロジェクトに追加することができます。

DB2 Alphablox コンテンツを含む JSP ファイルを作成する

この作業では、Rational Developer ツールを DB2 Alphablox ツールキットともに使用して、Blox タグ・ライブラリーに対するアクセスを持つ新規の JSP ファイルを作成します。

前提条件: DB2 Alphablox ツールキットを Rational Developer ツールにインストールします。

1. Rational Developer の「**Project Explorer**」ビューで、「**ファイル**」>「**新規**」>「**JSP ファイル (JSP File)**」を選択します。
2. ファイル名を「**ファイル名**」フィールドに入力します。
3. 「**拡張オプションの構成 (Configure advanced options)**」をクリックして、「**次へ**」を押します。
4. JSP ファイルで使用する DB2 Alphablox タグ・ライブラリーを追加します。
   a. 「**追加**」ボタンをクリックして、「タグ・ライブラリーの追加 (Add Tag Libraries)」ウィンドウを開きます。
   b. JSP ファイルで使用することを計画している DB2 Alphablox タグ・ライブラリーを選択します。
   c. 「**次へ**」をクリックします。

5. 「**エンコード (Encoding)**」リストから「**ISO 10646/Unicode(UTF-8)**」を選択します。 DB2 Alphablox アプリケーションが正しく稼働するには、UTF-8 エンコードが必要です。
6. 「**完了**」をクリックします。 新規の JSP ファイルがプロジェクト・リストに表示されます。
7. ファイル名をダブルクリックして、JSP エディター・ウィンドウでファイルを開きます。
8. カーソルを HTML `<head>` タグの内部、ただし上記で追加した DB2 Alphablox JSP taglib ディレクティブの後に置きます。
9. 新規行に次のように入力して、Blox ヘッダー・タグを入力します。 `<blox:header/>` JSP ファイルを実行するときに、必要な DB2 Alphablox JavaScript および CSS ファイルを追加するには、Blox ヘッダー・タグが必要です。

これで、新規の JSP ファイルが、選択した DB2 Alphablox タグ・ライブラリーにアクセスできるようになりました。 Rational Developer ツールの Content Assist 機能を使用して、選択したタグ・ライブラリーの Blox タグとタグ属性を挿入できます。

DB2 OLAP Server および Essbase データ・ソースにアクセスする

WebSphere 5.1 サーバー・インスタンスを使用しているときに、Rational Developer ツールで DB2 OLAP Server™ または Essbase データ・ソースにアクセスするには、必要なクライアント・ライブラリーをロードする始動バッチ・ファイルを作成する必要があります。

前提条件: DB2 Alphablox ツールキットをインストールします。 DB2 Alphablox を WebSphere 5.1 統合テスト環境にインストールします。 WebSphere 5.1 サーバー・インスタンスを構成します。 IBM® DB2 OLAP Server または Hyperion Essbase を開発マシンにインストールします。

Rational Developer ツールでは、WebSphere 5.1 統合テスト環境での制限のために、DB2 OLAP Server または Hyperion Essbase データ・ソースにアクセスするには、Rational Developer ツール用の始動バッチ・ファイルを作成する必要があります。バッチ・ファイルを使用して Rational Developer ツールを開始するときには、必要な Essbase クライアント・ライブラリーが Java ライブラリー・パスに追加されます。

WebSphere 5.1 サーバー・インスタンスを使用する DB2 OLAP Server または Essbase にアクセスできるようにするためのバッチ・ファイルを作成するには、以下のようにします。

1. テキスト・エディターを使用して、新規のテキスト文書を作成します。
2. DB2 Alphablox aassetup.bat ファイルを呼び出すコード行を追加します。このファイルは、DB2 Alphablox インストール・ディレクトリーにあります。 たとえば、次のコードは、指定した DB2 Alphablox インストール・ディレクトリーにある aassetup.bat ファイルを実行します。

```
call C:¥alphablox¥analytics¥bin¥aassetup.bat
```

3. RAD rationalsdp.exe ファイルを呼び出す 2 番目の行を追加します。このファイルは、RAD を開始します。 たとえば、次のコードは Rational Developer ツールを実行します。

   ```
   call C:¥Program Files¥IBM¥Rational¥SDP¥6.0¥rationalsdp.exe
   ```

4. このファイルを startRAD.bat としてワークステーションのデスクトップ (または別の便利な位置) に保管します。

startRAD.bat ファイルをダブルクリックすると、DB2 Alphablox の aassetup.bat が実行し、必要なパスおよび環境変数を設定します。そして、Rational Application Developer が開始されます。 DB2 Alphablox とともに使用される Essbase クライアント・ライブラリーのバージョンは、データ・ソースとして使用している DB2 OLAP Server (または Hyperion Essbase) のバージョンと一致している必要があります。 DB2 Alphablox とともに使用される Essbase クライアント・ライブラリーを変更するには、DB2 OLAP Server / Essbase Client Library Utility (ChangeEssbase.bat) を実行します。これは、*db2_alphablox*¥analytics¥bin ディレクトリーにあります。ここで、*db2_alphablox* は、DB2 Alphablox インストールのルート・ディレクトリーです。このユーティリティーは、作成した startRAD バッチ・ファイルを使用して RAD を開始するときに、実行されるバッチ・ファイルの 1 つを変更します。

上で説明した startRAD.bat ファイルの全体例は、以下のとおりです。

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

DB2 OLAP Server または Essbase データ・ソースにアクセスしようとするときには、必ず startupRAD.bat ファイルを使用して、Rational Developer ツールを開始する必要があります。

第 4 章 チュートリアル: DB2 Cube Views を使用して DB2 Alphablox キューブを構築する

このチュートリアルでは、DB2 Cube Views サンプル・データベースを使用して構築される DB2 Alphablox キューブの作成をガイドします。

このチュートリアルの作業では、カスタム DB2 Alphablox キューブの構築について詳しく説明していません。その代わりに、DB2 Alphablox Cube Server の機能を探るために使用できる DB2 Alphablox キューブをすばやく作成する方法を示すことを目的としています。また、結果として得られるデータ・ソースは DB2 Alphablox アプリケーションのテストおよび構築に使用できます。

前提条件:

- DB2 Alphablox をインストールします。インストールについて詳しくは、「*インストール・ガイド*」を参照してください。
- DB2 Cube Views CVSAMPLE サンプル・データベースがインストールされている、サポートされた DB2 Cubes Views インプリメンテーションに対するアクセス権限を取得します。 DB2 Cube Views のサポートされるバージョンについて詳しくは、「*インストール・ガイド*」を参照してください。

チュートリアルでは、DB2 Cube Views CVSAMPLE サンプル・データベースに基づいて DB2 Alphablox キューブを作成する方法を学習します。チュートリアル全体で、以下の作業を学習します。

DB2 リレーショナル・データ・ソースを定義する

この作業では、DB2 Alphablox で DB2 データベース用のデータ・ソース定義を定義します。

前提条件: 必要な DB2 JDBC ドライバーが DB2 Alphablox にアクセス可能になっている必要があります。

DB2 Alphablox キューブでは、基礎となるリレーショナル・データ・ソースが DB2 Alphablox データ・ソースとして事前定義されていることを必要とします。 DB2 Alphablox キューブは、リレーショナル・データベースで使用可能なメタデータおよびデータを使用して生成されます。

DB2 データベース用の DB2 Alphablox データ・ソースを定義するには、以下のようにします。

1. DB2 Alphablox 管理ページに管理ユーザー (または管理者権限を持つユーザー) としてログインします。
2. 「**管理**」タブをクリックします。
3. 「**データ・ソース**」リンクをクリックします。
4. 「**作成**」ボタンをクリックします。

5. 「**アダプター**」メニューから、ご使用のデータベース・サーバーに該当する IBM DB2 JDBC ドライバーを選択します。

   IBM DB2 JDBC Type 4 Driver または **IBM DB2 UDB on iSeries Driver** を選択します。
6. 「**データ・ソース名**」フィールドに、データ・ソースに使用する名前として CVSAMPLE を入力します。
7. 「**サーバー名**」、「**ポート番号**」、および「**データベース名**」 (これは CVSAMPLE です) に適切な値を入力します。

   注: これらのフィールドの正しい値を判断する際に助けが必要な場合は、データベース管理者に連絡してください。
8. 「**デフォルト・ユーザー名**」および「**デフォルト・パスワード**」を入力します。

   ユーザー名とパスワードはリレーショナル・データベースで有効でなければなりません。 DB2 Alphablox キューブがリレーショナル・データベースにアクセスするときには、デフォルトのユーザー名とパスワードが常に使用されます。使用するデータベース・ユーザーはデータベースに対する読み取りアクセス権を持っている必要があります。

   注: 指定されたリレーショナル・データ・ソースを使用して DB2 Alphablox キューブにデータを入れる場合は、「**DB2 Alphablox ユーザー名とパスワードを使用する (Use DB2 Alphablox Username and Password)**」の値は無視されます。アクセス制御リスト (ACL) を使用して、DB2 Alphablox キューブに対するアクセスを制限することができます。 ACL について詳しくは、「*管理者用ガイド*」を参照してください。
9. データ・ソースを使用して DB2 Alphablox キューブにデータを入力する場合は、「**最大行数**」および「**最大列数**」の値は無視されます。値を入力することもできます。それらの値は他のアプリケーションがデータ・ソースを使用するときに使用されます。
10. JDBC ロギング情報を DB2 Alphablox ログ・ファイルに書き込まない場合は、「**JDBC トレースが使用可能 (JDBC Tracing Enabled)**」の値を「**いいえ**」に設定します。問題が発生して、その原因をデバッグする必要がある場合に限り、JDBC トレースを使用可能にします。
11. 「**保管**」ボタンをクリックして、データ・ソース定義を保管します。

これで、CVSAMPLE 用の DB2 Alphablox データ・ソース定義を定義しました。この DB2 データ・ソース内の Cube Views™ キューブ・メタデータにアクセスするための DB2 Alphablox キューブ定義を作成することができます。

Alphablox Cube Server Adaptor データ・ソースを定義する

この作業では、Alphablox Cube Server Adapter を使用する DB2 Alphablox データ・ソース定義を定義します。

前提条件: DB2 CVSAMPLE データベース用の DB2 Alphablox データ・ソース定義を作成します。

DB2 Alphablox Cube Server Adapter データ・ソースを定義するには、以下のようにします。

1. DB2 Alphablox 管理ページに管理ユーザー (または管理者権限を持つユーザー) としてログインします。
2. 「**管理**」タブをクリックします。
3. 「**データ・ソース**」リンクをクリックします。
4. 「**作成**」ボタンをクリックします。
5. 「**アダプター**」メニューから、**Alphablox Cube Server Adapter** オプションを選択します。
6. 「**データ・ソース名**」フィールドに、データ・ソースに使用する名前として `DB2AlphabloxCubes` を入力します。
7. 「**保管**」ボタンをクリックして、データ・ソース定義を保管します。

DB2 Alphablox キューブにアクセスするために使用できる DB2 Alphablox データ・ソースを定義しました。次に、アクセス可能な DB2 Alphablox キューブを定義する必要があります。

DB2 Alphablox キューブを定義する

この作業では、DB2 Cube Views CVSAMPLE キューブからのメタデータに基づいて、DB2 Alphablox キューブを定義します。

前提条件: DB2 リレーショナル・データ・ソースを定義します。 DB2 Alphablox Cube Server Adapter データ・ソースを定義します。

DB2 Alphablox キューブの一般プロパティーを定義するには、以下のようにします。

1. DB2 Alphablox 管理ページに管理ユーザー (または管理者権限を持つユーザー) としてログインします。
2. 「**管理**」タブをクリックします。
3. 「**キューブ**」リンクをクリックします。
4. 「**作成**」ボタンをクリックします。 「キューブ管理 (Cube Administration)」ウィンドウが開きます。
5. 新規キューブを定義します。
   a. 「**DB2 Alphablox キューブ名 (DB2 Alphablox Cube Name)**」フィールドに、`CVSales` を入力します。
   b. 「**DB2 Alphablox キューブ名 (DB2 Alphablox Cube Name)**」フィールドの隣にある「**使用可能**」オプションを選択します。 このオプションを選択すると、サーバーが再始動するときに、キューブが自動的に開始します。
   c. 「**リレーショナル・データ・ソース (Relational Data Source)**」メニューで、`MyDB2` を選択します。これは、このチュートリアル用に作成したリレーショナル・データ・ソースです。
   d. 「**セキュリティー役割 (Security Role)**」オプションは選択しないでおきます。 このオプションは、ユーザーが特定のキューブにアクセスするのを制限するために使用できます。

6. DB2 Cube Views 設定を使用可能にして、使用するメタデータを指定します。
   a. 「**DB2 Cube Views 設定を使用可能にする (Enable DB2 Cube Views Settings)**」オプションを選択します。
   b. 「**キューブ・モデル**」メニューから、**CVSAMPLE.Sales** を選択します。
   c. 「**キューブ**」メニューから、「**一般売り上げ**」キューブを選択します。
   d. 「**ビジネス名を使用する (Use Business Names)**」ラジオ・ボタンを選択して、名前を指定します。 このオプションを選択するときには、通常、分かりやすく読みやすいメンバー名を使用します。
   e. 「**キューブ定義のインポート (Import Cube Definition)**」ボタンをクリックします。 このオプションを使用すると、キューブ定義をインポートし、メジャーおよびディメンションを DB2 Alphablox キューブに事前に入力することができます。インポートされるキューブ定義は、DB2 Cube Views メタデータのサポートに基づいて、DB2 Alphablox がマッチングできる限り厳密に DB2 Cube Views キューブを反映します。キューブでの作業経験が豊富な場合は、必要に合うようにメジャーおよびディメンションを変更することができます。
   f. 「**インポート・ログの表示 (Show Import Log)**」ボタンをクリックして、インポート操作に関連したログ指定の情報およびデバッグ・メッセージを表示します。 このチュートリアルでは、この機能の理解を助ける目的でのみ、このステップが含まれています。
   g. 「**開始、再構築、および編集時にキューブ定義をインポートする (Import cube definition on start, rebuild, and edit)**」オプションを選択します。 このオプションを選択すると、DB2 Alphablox キューブが開始、再構築、または編集用に開かれるたびに、DB2 Alphablox キューブが最新の DB2 Cube Views 定義を使用することになります。 DB2 Alphablox および DB2 Cube Views に十分精通している場合は、キューブ定義をインポートしてカスタマイズすることができます。
7. 「**保管**」ボタンをクリックして、DB2 Alphablox キューブ定義を保管します。

これで、DB2 Alphablox キューブ定義を作成しました。新規の CVSales キューブを開始できます。

DB2 Alphablox キューブを開始する

この作業では、DB2 Alphablox 管理ページを使用して CVSales キューブを開始します。

前提条件: DB2 リレーショナル・データ・ソースを定義します。 DB2 Alphablox Cube Server Adapter データ・ソースを定義します。DB2 Alphablox キューブを定義します。

CVSales キューブを開始するには、以下のようにします。

1. DB2 Alphablox 管理ページに管理ユーザー (または管理者権限を持つユーザー) としてログインします。
2. 「**管理**」タブをクリックします。
3. 「**ランタイム管理**」セクションで、「**キューブ**」リンクをクリックします。

4. 「**DB2 Alphablox キューブ (DB2 Alphablox Cubes)**」リストから、開始したい DB2 Alphablox キューブを選択します。
5. 「**開始**」ボタンをクリックします。 キューブが開始されると、状況フィールドに「**実行中**」と表示されます。

これで、サンプル DB2 Alphablox キューブが稼働しています。作成した DB2 Alphablox キューブを使用してアプリケーションの構築を始めることができます。Query Builder アプリケーションを使用して、データ・ソースとしてご使用の新規キューブを選択し、そのキューブに対して MDX 照会を実行することにより、その新規キューブへのクイック・チェックを実行できます。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

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

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

商標

以下は、IBM Corporation の商標です。

DB2	DB2 OLAP Server	DB2 Universal Database
IBM	WebSphere	

Alphablox および Blox は、Alphablox Corporation の米国およびその他の国における登録商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[サ行]

[タ行]

[ハ行]

プログラム番号: 5724-L14

Printed in Japan

GD88-6692-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12