IBM DB2 Alphablox

管理者用ガイド

バージョン 8.4

SD88-6488-03
(英文原典：SC18-9432-03)

IBM DB2 Alphablox

管理者用ガイド

バージョン 8.4

SD88-6488-03
(英文原典：SC18-9432-03)

目次

第 1 章 DB2 Alphablox の概要

DB2 Alphablox は、J2EE アプリケーション・サーバー環境で稼働し、Web ベースの分析アプリケーションを作成するためのサービスを提供する製品です。 DB2 Alphablox は、IBM® WebSphere®、BEA WebLogic、Apache Tomcat などの先進的なアプリケーション・サーバーに統合できます。この章では、DB2 Alphablox の概要を示し、J2EE 環境と DB2 Alphablox との関連について解説し、DB2 Alphablox のアーキテクチャーを取り上げます。

DB2 Alphablox の概要

DB2 Alphablox は、企業のインフラストラクチャーに調和し、企業のファイアウォールの内側、および外側の両方のまたがる広範なユーザーを対象にする機能を持ったカスタム・アプリケーションや Web ベースのアプリケーションを短時間で作成する機能を備えています。 DB2 Alphablox プラットフォームで構築したアプリケーションは、Web ブラウザー内において、リアルタイムで高度なカスタマイズが可能な多次元分析を可能にする標準の Web ブラウザーで稼働します。

DB2 Alphablox プラットフォームには、以下の機能が用意されています。

- マルチディメンション・データベースとリレーショナル・データベースのデータにアクセスして対話を行う機能
- リレーショナル・データベースのデータに基づく構造化レポートを作成する機能
- 多彩なチャートの中から選択してデータを表示する機能
- データベースにデータを書き戻すアプリケーションを作成する機能 (特に、「what-if」タイプの財務計画アプリケーションで効果を発揮します)
- マルチディメンション・データ・ソースを使用し、フィルターやドリルダウンなどによってさまざまなレベルにデータを絞り込み、目的のデータだけを抽出した正確なビューを対話式に表示するための機能
- 直感的に操作できるユーザー・インターフェースから、ユーザーが強力なデータ分析を簡単に実行するための機能
- 1 つのアプリケーションから複数のデータ・ソースにアクセスするための機能
- アプリケーション・サーバー (IBM WebSphere や BEA WebLogic) など、さまざまなエンタープライズ・インフラストラクチャー・コンポーネントに統合するための機能

DB2 Alphablox には、開発者がカスタム・アプリケーションを作成するためのさまざまなアプリケーション・プログラマー・インターフェース (API) が用意されています。 DB2 Alphablox の API のプログラム言語は、 Java™ です。アプリケーション開発者がこの API にアクセスするには、サーバーで実行される Java を使用する方法と、ブラウザーで解釈される JavaScript™ を使用する方法があります。

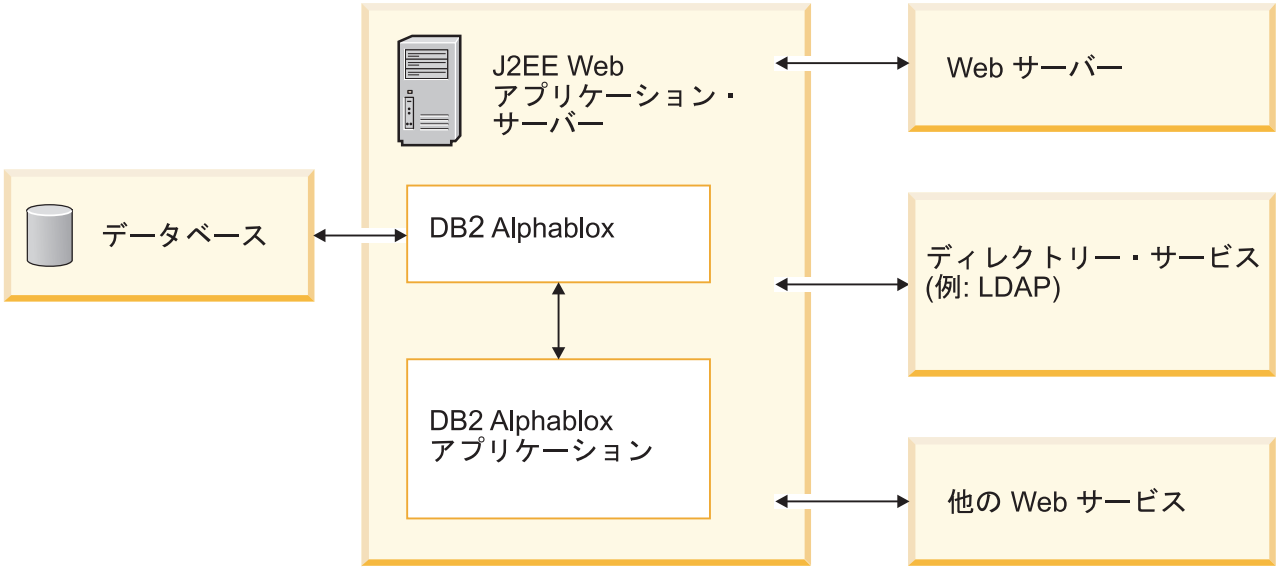
これ以降は、J2EE 環境と DB2 Alphablox との関連、DB2 Alphablox の各種コンポーネント、DB2 Alphablox のアーキテクチャー、DB2 Alphablox のコア・コンポーネントについて解説していきます。 DB2 Alphablox アプリケーションを作成する方

法の詳細については、 *開発者用ガイド* を参照してください。 DB2 Alphablox の API の構文とリファレンス情報については、 *開発者用リファレンス* を参照してください。

J2EE 環境と DB2 Alphablox

DB2 Alphablox は、J2EE アプリケーション・サーバー (IBM WebSphere と BEA WebLogic) や Apache Tomcat の環境の中で稼働します。サポートされているアプリケーション・サーバーとバージョンのリストについては、 *インストール・ガイド* を参照してください。

J2EE 環境と DB2 Alphablox との関連を以下の図にまとめます。



アプリケーション・サーバー内で実行するメリット

アプリケーション・サーバー内で実行することには、いくつかのメリットがあります。

- オープン・スタンダード・ベースの J2EE コンポーネントを DB2 Alphablox 環境に簡単に統合できる
- さまざまな Web サービスによって提供されているサービスにアクセスできる
- Java ランタイム環境に用意されているすべてのサービスや、それぞれのアプリケーション・サーバーに用意されている Java 拡張機能にアクセスできる
- Web サービスの提供やセキュリティーなどをそれぞれの分野の専門ベンダーに任せ、 DB2 Alphablox では分析用のプラットフォームの提供に集中できる
- JavaServer Pages (JSP)、 Java、 JavaBeans™ コンポーネント、XML などの J2EE テクノロジーを使いこなす J2EE 開発者のためのプラットフォームを提供できる

DB2 Alphablox をアプリケーション・サーバーと密に統合することによって、開発者は、上記のすべてのメリットを生かしながら、DB2 Alphablox の豊富な API セットにアクセスできます。

WebSphere と WebLogic のアプリケーション・サーバー構成

DB2 Alphablox は、IBM WebSphere または BEA WebLogic のアプリケーション・サーバーで稼働するように構成できます。 DB2 Alphablox の基本操作は WebSphere の場合も、WebLogic の場合も、Tomcat の場合も同じです。ただし、細かな管理作業にいくつかの小さな違いがあります。例えば、 WebSphere の場合は、DB2 Alphablox で作成した新規アプリケーションをアプリケーション・サーバーに登録する必要があります。そのようにしておけば、アプリケーション・サーバーの始動時にそのアプリケーションも始動します。

WebSphere や WebLogic などの商用のアプリケーション・サーバーと一緒に実行する場合は、それぞれのプラットフォームに用意されているすべてのツールやスケーラビリティーやサービスと、DB2 Alphablox プラットフォームに用意されているすべてのサービスの両方にアクセスできます。

インストール時に、WebSphere、WebLogic、Tomcat の構成オプションを設定します。詳細については、*インストール・ガイド*を参照してください。

DB2 Alphablox アプリケーションは J2EE アプリケーションなので、さまざまな構成で機能します。そのため、Apache Tomcat 構成でアプリケーションの開発とテストを行ってから、社内で使用しているアプリケーション・サーバー構成で展開することも可能です。ただし、プラットフォームが変わると実行時に多少の差異が出ることもあるので、対象の構成でアプリケーションをテストしてから展開する必要があります。それでも、アプリケーションにプラットフォーム固有のサービスがなければ、別の構成に移行するときに解決しなければならない問題はごくわずかです。

Apache Tomcat 構成

1 つのオプションとして、DB2 Alphablox では、Apache Tomcat アプリケーション・サーバーを使用できます。 Apache Tomcat 構成の場合、DB2 Alphablox のインストーラーは、指定の Apache Tomcat サーバー (バージョン 3.2.4 のみ) を使用します。 Apache Tomcat の詳細については、http://jakarta.apache.org/tomcat/ を参照してください。Tomcat バージョン 3.2.4 のコピーは、http://archive.apache.org/dist/tomcat/tomcat-3/archive/v3.2.4/ から入手できます。

DB2 Alphablox と一緒に Apache Tomcat を使用する場合、DB2 Alphablox プラットフォームを使用するために必要な Tomcat 固有の管理作業はありません。 Apache Tomcat 構成は、拡張性の高いソリューションであり、129 ページの『第 16 章 データベース・リポジトリーの使用』にあるクラスタリング・ソリューションも使用できます。

Apache Tomcat 構成には、ユーザー、グループ、役割をサポートする十分なセキュリティー機能が用意されています。 Apache Tomcat 構成では、サポートされている外部 Web サーバー (Microsoft® IIS、Sun iPlanet、Apache など) を使用することも、Tomcat 自身で HTTP 要求を処理することもできます。

ヒント: Apache Tomcat 構成に組み込まれている HTTP サーバーは、開発用のシステムや小規模な実動システムには適していますが、より大規模な実動システムの場合は、外部 Web サーバーを使用するほうが望ましいと言えます。外部 Web サーバーには、多くのキャッシング機能やページ・サービス機能があるので、HTTP のパフォーマンスが向上するからです。

DB2 Alphablox プラットフォームのコンポーネント

DB2 Alphablox プラットフォームを構成するエレメントは、以下のとおりです。それぞれについて、このセクションで詳しく説明します。

- 4 ページの『DB2 Alphablox』
- 8 ページの『DB2 Alphablox アプリケーション』
- 9 ページの『Application Studio』

DB2 Alphablox

DB2 Alphablox には、分析アプリケーションの迅速な開発、展開、使用に特化した堅固なアーキテクチャーがあります。このアーキテクチャーに組み込まれている主な機能は、以下のとおりです。

- 『データ・アダプター』
- 5 ページの『Blox コンポーネント』
- 7 ページの『DHTML クライアント』
- 7 ページの『リモート管理機能』
- 7 ページの『一元的なアプリケーション管理』
- 8 ページの『N 層アーキテクチャー』

データ・アダプター

DB2 Alphablox には、各種データベースとの接続に特化したデータ・マネージャーが用意されています。このデータ・マネージャーは、リレーショナル・データベースとマルチディメンション・データベース (DB2 Alphablox キューブなど) のデータに対するアクセス、ブラウズ、照会、検索を実行します。各データベースには、プラグイン・アダプターによって接続します。リレーショナル・データベースのためのプラグイン・アダプターは、基本的に JDBC ドライバーです。それぞれのアダプターは、データベース固有の接続情報や処理をカプセル化して、追加のデータベースを接続するための労力を大幅に軽減しています。

データ・マネージャーとその関連データ・アダプターには、以下のサポートが用意されています。

- 事前に構成されている名前付きのデータベース接続のコレクション (データ・ソースという) をブラウズする機能
- DB2 Alphablox アプリケーションに対して、各データ・ソース内の使用可能なデータベースへのアクセスを提供する機能
- ある特定のデータ・ソースと互換性のある照会タイプをパブリッシュする機能
- データベースのメタデータの全探索を実行する機能
- ユーザー・セッションのデータベース接続を管理する機能
- 照会オブジェクトを基本となるネイティブの照会言語に変換する機能
- データベースに対して照会を実行する機能
- データベースから結果セットのデータとスキーマを獲得する機能
- 表示、ピボット、拡張、ソート、ドリルによって結果セットを処理する機能

- ユーザー入力を取り込んで、データを基本データベースに書き戻すアプリケーションを作成する機能 (予算アプリケーションの「what-if」タイプのシナリオでよく使用します)

Blox コンポーネント

DB2 Alphablox アプリケーションは、Blox コンポーネントと呼ばれる部品を使用して、標準の Web ブラウザーから社内のデータにアクセスし、リアルタイムでデータを表示します。 Blox コンポーネントは、標準の JSP ページで結合 (アセンブル) される再利用可能なソフトウェア・コンポーネントです。その結果として組み立てられる対話式アプリケーションには、Web ブラウザーから社内のイントラネットまたはインターネットを経由してアクセスできます。

これらのテスト済みのソフトウェア・コンポーネントによって、以下のアプリケーション機能を実現できます。

- データへのアクセス
- データの対話式分析
- データの柔軟な表示
- 詳細な管理情報 (ユーザー、グループ、アプリケーション名など) へのアクセス

例えば、DataBlox コンポーネントは、基本データベースのデータに基づいて、HTML のリストに製品カテゴリーのデータを設定できます。ユーザーがそのリストから 1 つのカテゴリーを選択すると、基本データベースに対して照会が実行されます。結果として、ユーザーが「軽自動車」というカテゴリーを選択した場合と「SUV」というカテゴリーを選択した場合とでは、それぞれ異なるデータ・セットが表示されます。

Blox コンポーネントには、幅広いアプリケーション・プログラマー・インターフェース (API) 呼び出しが用意されており、その API 呼び出しには JSP ファイルからアクセスできます。各種の API 呼び出しによって、アクセス対象のデータを大幅にカスタマイズしたり、ユーザーにどの程度の対話機能を提供するかを制御したり、ユーザーに対するデータ表示をカスタマイズしたりすることが可能になります。

Web ベース・アプリケーションに Blox コンポーネントをアセンブルすることにより、開発者はビジネス情報に対する即時アクセスを素早くユーザーに提供することができます。一群の分析アプリケーションで同じ Blox コンポーネントを使用すれば、アプリケーションの作成、配布、保守にかかる労力を削減できるだけでなく、ユーザーが操作方法を習得するための時間も短縮できます。どのようなデータを表示する場合でも、Blox の動作は、アプリケーションやプラットフォームによって変わることがありません。

さらに、DB2 Alphablox アプリケーションの組み立てに要するスキルは、従来のアプリケーション開発に必要なスキルよりもはるかに少なくてすみます。アプリケーション開発者は、最初に複雑なプログラム言語をマスターしなくても、HTML、JavaScript、マルチメディア・オブジェクト、Blox コンポーネントを使用して、すぐに開発作業を開始できます。これらの部品を使用して、直感的に操作できるユーザー・インターフェースを作成し、強力な機能をユーザー・コミュニティーに提供できます。

DB2 Alphablox には、アプリケーションを組み立てるための部品として以下の Blox コンポーネントが用意されています。

Blox コンポーネント	用途
DataBlox	• サポートされるマルチディメンション・データベースへのアクセスを提供し、データ・セットのクライアント表示を開発します。 • サポートされるリレーショナル・データベースへのアクセスを提供します。 • 照会要求を受け入れて実行します。 • データ表示を担当する Blox コンポーネントに照会の結果セットを提供します。
ChartBlox	• マルチディメンション・データのグラフィカル・ビューを表示します。 • ユーザーがデータをさまざまなチャート形式 (円グラフ、棒グラフ、折れ線グラフなど) で処理することを可能にします。 • ユーザーが階層データを順々にドリルダウンしたり、データ・ビューのピボットを実行したりすることを可能にします。
DataLayoutBlox	• 使用可能なデータ・ディメンションと各ディメンションが存在する軸をグループ化したリストを表示します。 • ユーザーがページ軸、行軸、列軸、「その他」(未使用) の軸の間でディメンションを移動することを可能にします。
GridBlox	• マルチディメンション・データまたはリレーショナル・データを拡張グリッド形式で表示します。 • ユーザーがマルチディメンション・データの分析や操作を行うことを可能にします。 • ユーザーが階層データを順々にドリルダウンしたり、データ・ビューのピボットを実行したりすることを可能にします。
PageBlox	• ページ軸に存在するディメンションのドロップ・リストを表示します (つまり、ChartBlox と GridBlox に表示されるデータをフィルターによって絞り込みます)。 • ユーザーがフィルターによってデータの絞り込みを行う対象のディメンションやメンバーを変更することを可能にします。
ToolbarBlox	• ユーザーが Blox 機能にアクセスするためのボタンを表示します。 – チャートの表示とグリッドの表示を切り替えるための機能 – チャート・タイプを選択するための機能 – アプリケーション・ビューの保管や検索のための機能 – ツールバーを移動してアクセスを容易にするための機能 – Blox を別個のウィンドウで開くための機能 – 行と列のピボットを実行するための機能 • アセンブラーがツールバーに表示するボタンを選択して、Blox 機能へのユーザー・アクセスを調整することを可能にします。
PresentBlox	上記 6 つの Blox コンポーネント (DataBlox、DataLayoutBlox、ChartBlox、GridBlox、PageBlox、および ToolbarBlox) の機能を 1 つの Blox コンポーネントに結合して、アプリケーションの組み立てを単純化し、Web ページ上の資産を保存します。

Blox コンポーネント	用途
RepositoryBlox	アプリケーション・アセンブラーが保管済みのオブジェクト (保管済みのアプリケーション・ビューや、サーバー、アプリケーション、グループ、ユーザーのプロパティーなど) にアクセスすることを可能にします。

DB2 Alphablox に用意されている Blox API の詳細については、 *開発者用リファレンス*を参照してください。 Blox コンポーネントを使用してアプリケーションを作成するためのガイドラインについては、 *開発者用ガイド* を参照してください。

DHTML クライアント

DB2 Alphablox のアプリケーション開発者は、Web ブラウザーからアクセスできるアプリケーションを作成できます。Web ブラウザーは、標準的な DHTML テクノロジーを使用するので、追加のプラグインは不要です。 DB2 Alphablox が要求に沿ったモードでアプリケーションをレンダリングするので、アプリケーション開発者がさらにコーディングを行う必要はありません。

DHTML クライアントは、JavaScript と Cascading Style Sheet (CSS) を利用した Dynamic HTML テクノロジーに基づいており、使いやすくカスタマイズ性に優れたグラフィカル・ユーザー・インターフェースによる幅広いデータ分析機能をサポートしています。プラグインや Java クラス・ファイルのダウンロードは必要ありません。 DHTML クライアントでは、サポートされるバージョンの Mozilla Firefox および Microsoft Internet Explorer を使用する必要があります。

リモート管理機能

DB2 Alphablox には、Web ベースの管理ページ、コンソール・コマンド・ウィンドウ、telnet ウィンドウのいずれかを使用した広範囲のシステム管理機能が用意されています。管理者は、これらの管理インターフェースから以下の作業を実行できます。

- DB2 Alphablox コンソール・ログの作成と、記録するイベントやメッセージのレベルの指定
- DB2 Alphablox とアプリケーションのアクティビティーのモニター
- データ・ソース、ユーザー、グループ、アプリケーションなどの DB2 Alphablox オブジェクトの作成
- DB2 Alphablox オブジェクトの使用状況のモニター
- DB2 Alphablox のサービス、セッション、アプリケーションの開始と停止

詳しくは、25 ページの『第 4 章 基本管理タスク』と 83 ページの『第 13 章 DB2 Alphablox プロパティーの構成』を参照してください。

一元的なアプリケーション管理

DB2 Alphablox には、一元的なアプリケーション管理を行うためのアプリケーション・マネージャーが用意されています。アプリケーション・マネージャーは、以下の情報の追跡管理を行います。

- 使用可能なアプリケーションのリスト
- アクティブなアプリケーションのリスト

- アプリケーション内のアクティビティー
- アプリケーションにログインしたユーザー
- アプリケーションが使用している Blox コンポーネント
- アプリケーションが使用しているデータ・ソース
- アプリケーション・インスタンスの履歴

N 層アーキテクチャー

DB2 Alphablox は、最新 Java テクノロジーを利用して、 Web ベースの N 層アーキテクチャーをインプリメントします。典型的なアプリケーションは、以下の 3 つの層を使用します。

- データを配置して、データベース・サーバー (リレーショナルまたはマルチディメンション) からデータを検索するための層
- アプリケーション Web ページをアプリケーション・サーバー上の Web アプリケーションに配置して、DB2 Alphablox によるサービスを受けるための層
- クライアント・マシンの Web ブラウザーによって Web ページやユーザー・インターフェースを表示するための層

Blox コンポーネントは、イントラネットまたはインターネットを経由して Web サーバーから Web ブラウザーにオンデマンドで渡されます。この Java ベース・アーキテクチャーを使用すれば、クライアント・サイドのアプリケーション・ソフトウェアのインストール、構成、保守の手間を省けます。

DB2 Alphablox アプリケーション

ユーザーにとって、DB2 Alphablox アプリケーションは、他の Web サイトのようにブラウズする一連の Web ページとして表示されます。これらの Web ページは、以下のアプリケーション・コンポーネントのコンテナーとして機能します。

- ユーザー・インターフェースを拡張するための標準的な HTML のタグやページ・エレメント (ロゴ、テキスト、イメージ、アイコン、ビデオ・クリップ、サウンド・クリップ、アニメーションなど)
- 必要なアプリケーション機能やユーザー・インターフェースを配信するために必要なコンポーネント
- 拡張アプリケーションと UI ロジックのための JavaScript または Java スクリプトレット

例えば、販売分析アプリケーションであれば、まず販売地域のイメージ・マップが表示されます。ユーザーがいずれかの地域をクリックすると、その地域のデータのチャートが表示されます。ユーザーはさらに、ユーザー・インターフェースからチャート形式を変更したり、データを対話式のグリッド形式で表示したり、マルチディメンション操作 (ドリルやピボット) を実行したりできます。アプリケーションによっては、Blox 出力を印刷用したり、他のアプリケーション (スプレッドシートなど) にエクスポートしたりするためのレンダリングも可能です。 DB2 Alphablox アプリケーションのタイプとコンポーネントの詳細については、 15 ページの『第 2 章 DB2 Alphablox アプリケーション』を参照してください。

Application Studio

DB2 Alphablox の姉妹製品である Application Studio は、DB2 Alphablox そのものに組み込まれています。この製品を使用すれば、完成版のカスタム分析アプリケーションをユーザーに配信するための時間と労力をさらに削減できます。 Application Studio には、以下のコンポーネントが含まれています。

- DHTML Query Builder。アプリケーション・データ・ソースに対する照会の開発とテストを行ったり、アプリケーションのプロトタイプ作成とテストのためにいくつかのサンプル・データ・ソースにアクセスしたりするために使用できるワークベンチ・ツールです。
- Blox Sampler - DHTML。DB2 Alphablox の機能に焦点を合わせた実用的なコード例が含まれています。開発者が Blox API の理解を深めるためにも役立ちます。

Application Studio を使用すれば、アプリケーションのプロトタイプを短時間で作成できます。 Studio には、テンプレートの構成手順を段階的に説明した詳しい資料も用意されています。 Application Studio と各コンポーネントの詳細については、Application Studio 内の各種テンプレートのオンライン・ヘルプを参照してください。

DB2 Alphablox のアーキテクチャー

ここでは、DB2 Alphablox の以下のコンポーネントについて解説します。

- 10 ページの『サービス・マネージャー』
- 10 ページの『リクエスト・マネージャー』
- 10 ページの『セッション・マネージャー』
- 10 ページの『ユーザー・マネージャー』
- 11 ページの『アプリケーション・マネージャー』
- 12 ページの『データ・マネージャー』
- 12 ページの『リポジトリー・マネージャー』
- 12 ページの『ファイル・マネージャー』
- 12 ページの『コンソール・マネージャー』
- 13 ページの『DB2 Alphablox Cube Manager』
- 13 ページの『クラスター・マネージャー』

DB2 Alphablox を構成するいくつかのサービスは、それぞれが別個に独立して稼働します。また、以下のような広範囲の管理機能も用意されています。

- DB2 Alphablox のリソース (アプリケーション、データ・ソース、ユーザー、グループなど) を管理するための管理インターフェース
- Web ブラウザー、Telnet、またはコマンド・ウィンドウからアクセス可能な管理コンソール
- サービス、セッション、アプリケーションの開始、一時停止、再開、停止のための機能
- DB2 Alphablox とアプリケーションのアクティビティーをモニターする機能
- イベントやメッセージを追跡管理するためのログ

DB2 Alphablox の初期インストール時に、稼働に必要なデフォルトのプロパティー・セットが自動的に構成されます。その後、システム管理者は、DB2 Alphablox のデフォルト・プロパティーの変更や追加を行えます。詳しくは、83 ページの『第 13 章 DB2 Alphablox プロパティーの構成』を参照してください。

サービス・マネージャー

サービス・マネージャーは、他のサービス (必要なサード・パーティー・サービスも含む) の開始、制御、アクセス提供を担当します。また、すべてのサービス要求を処理し、サービス・コンポーネントの配布を管理します。

リクエスト・マネージャー

リクエスト・マネージャーは、アプリケーション・ページとピア・サービスのすべての要求を制御します。 DB2 Alphablox は、それぞれの接続ごとに*ピア* を作成します。このピアは、接続の状態を追跡管理します。リクエスト・マネージャーは、各要求のスレッドの作成、モニター、管理を行い、セッション ID を有効なセッション・リストに照らして確認し、各要求を処理するための適切な要求オブジェクトを作成します。

セッション・マネージャー

セッション・マネージャーは、ユーザーと DB2 Alphablox アプリケーションとの対話を制御し、モニターします。ユーザーがアプリケーションを要求すると、セッション・マネージャーは、まず認証のためにその要求をユーザー・マネージャーに渡します。ユーザーが正しく認証を受けた時点で、セッション・マネージャーは新規セッションを作成します。 1 人のユーザーが複数のセッション (複数のブラウザー・インスタンス) を同時に保有することもできます。セッション・マネージャーは、セッション・オブジェクトの作成と管理を行い、ユーザーがどのアプリケーションを使用しているのかを追跡管理します。

セッション・マネージャーはさらに、各アプリケーションの現在の状態を保管してから休止セッションを終了し、セッション・リソースを解放します。ユーザーがセッションの期限が切れた後にブラウザー・ウィンドウで作業しようとすると、再接続のメッセージが表示されます。ユーザーが再接続を要求すると (つまり、Internet Explorer ブラウザーの「**更新**」ボタン、または Mozilla Firefox の「**再読み込み**」ボタンをクリックすると)、セッション・マネージャーは新規セッションを作成し、前回のセッションの期限が切れた時点でのアプリケーションの状態を復元します。

ユーザー・マネージャー

Alphablox 内のユーザー・マネージャーは、DB2 Alphablox サービスを利用するすべてのユーザーを制御します。アプリケーション・ユーザーと管理ユーザーの認証、ユーザーが使用しているリソースのモニター、データやアプリケーションに対するユーザー・アクセスの管理、アクティブ・ユーザーのリストの保守、各ユーザーに関する情報 (ユーザーがいつからどれくらいの時間にわたってどのアプリケーションを利用したかなどの情報) の保守を行います。

DB2® Alphablox 8.4 およびそれ以前のバージョンでは、ユーザー・マネージャーは、拡張可能ユーザー・マネージャーというパーソナライゼーション・エンジンの基盤の上に存在します。外部ユーザー・マネージャーは Servlet 2.2 に基づいてセキ

ュリティーを処理し、LDAP ベースおよびリポジトリー・ベースのユーザー管理機能を提供し、さらに他の外部リポジトリー用のカスタム・セキュリティーのための API を提供します。 DB2 Alphablox 8.4.1 からは、DB2 Alphablox は Servlet 2.4 仕様をサポートするようになりました。拡張可能ユーザー・マネージャーは推奨されなくなり、Servlet 2.4 に基づく新しいパーソナライゼーション・マネージャーに取って代わられました。このパーソナライゼーション・マネージャーは、ユーザーまたはグループ・オブジェクトが作成、ロード、または削除されたときに、そのことを DB2 Alphablox に通知する方法を提供します。また、それを使用して、ユーザーまたはグループのプロパティーを指定および変更したり、グループに含まれるユーザーとグループの変更を行うこともできます。セキュリティーは別個のモデルになりました。ユーザー認証情報は基礎となるアプリケーション・サーバーで構成されますが、 DB2 Alphablox はユーザー認証のための Java 認証・承認サービス (JAAS) ベースの API を提供します。

ユーザー・マネージャーは、DB2 Alphablox セキュリティー機構のための以下のオブジェクトを認識し、その管理を行います。

- 「ユーザー」は、個々のエンド・ユーザーかアプリケーション管理者のいずれかです。
- 「グループ」は、ユーザーや他のグループの集合であり、管理者が複数のユーザーを 1 つにまとめて管理するための便利なメカニズムです。「グループ」は階層構造になっています。
- 「許可」は、特定ユーザーが役割の中に持つアクセス権限 (アクセス権限なし、読み取り、読み取り/書き込み) を決定するためのオブジェクトです。
- 「役割」は、特定のアクセス許可セットを持つユーザーやグループのリストです。

注: 許可と役割に関して 1 つの注意点があります。1 つの役割の中に同じユーザーが複数回含まれている場合 (例えば、個人のユーザーとして含まれているほかに、グループのメンバーとしても含まれている場合など) は、そのユーザーにすべての項目のすべてのアクセス許可の和集合が与えられます。例えば、そのユーザーが個人として読み取り/書き込みアクセス権限を持っており、そのユーザーが所属しているグループが読み取りアクセス権限を持っている場合、そのユーザーは、読み取り/書き込みアクセス権限を持つことになります。

アプリケーション・マネージャー

アプリケーション・マネージャーは、アセンブラーまたは管理者が DB2 Alphablox ホーム・ページの「**管理**」タブにある「**アプリケーション**」ページでアプリケーションの作成または変更を行った場合にアプリケーション定義を作成します。

アプリケーション・マネージャーはさらに、アプリケーションに関するユーザー要求を受け入れ、ユーザーにアプリケーションへのアクセス権があるかどうかを確認し、それぞれの要求に応じてアプリケーション・インスタンスを作成します。また、すべてのアプリケーション・インスタンスをモニターし、管理者がインスタンスをシャットダウンすることを可能にし、休止インスタンスを自動的に終了し、以下の情報を保守します。

- 使用可能なアプリケーションのリスト
- アクティブなアプリケーションのリスト

- アプリケーション内のアクティビティー
- アプリケーションにログインしたユーザー
- アプリケーションが使用している Blox コンポーネント

データ・マネージャー

データ・マネージャーは、アプリケーションによって使用される各データ・ソースへのアクセスを制御します。また、さまざまなリレーショナル・データ・ソースやマルチディメンション・データ・ソースのデータのブラウズ、照会、検索を可能にします。サポートされているデータ・ソースのタイプについては、このリリースのインストール・ガイドを参照してください。

データ・マネージャーは、データベース・アダプターと連動して、以下の具体的な機能を果たします。

- 事前に構成されている名前付きデータ・ソースのコレクションをブラウズする機能
- 各データ・ソースに使用可能なキューブがあれば、それを公開する機能
- ある特定のデータ・ソースと互換性のある照会タイプをパブリッシュする機能
- データベースのメタデータの全探索を実行する機能
- ユーザー・セッションのデータベース接続を管理する機能
- 照会オブジェクトを基本となるネイティブの照会言語に変換する機能
- データベースに対して照会を実行する機能
- データベースから結果セットのデータとスキーマを獲得する機能
- ピボット、展開、ドリルによって結果セットを変更する機能
- 「what-if」タイプの分析に関してデータベースにユーザー入力を書き戻すアプリケーションを作成する機能

リポジトリー・マネージャー

リポジトリー・マネージャーは、DB2 Alphablox リポジトリーとその内容を制御します。その内容には、保管済みのアプリケーション・ビューと、他のすべてのアプリケーション、グループ、ユーザーのプロパティーが含まれます。 DB2 Alphablox リポジトリーの保管場所としては、オペレーティング・システム・ファイルと、リレーショナル・データベースがあります。詳しくは、129 ページの『第 16 章 データベース・リポジトリーの使用』を参照してください。

ファイル・マネージャー

ファイル・マネージャーは、一時ファイルと DB2 Alphablox 管理ページで使用するファイルを制御します。

コンソール・マネージャー

コンソール・マネージャーには、システム管理者が DB2 Alphablox 環境のモニターと制御を行うための複数のインターフェース (Web ブラウザー、Telnet、コマンド・ウィンドウ) が用意されています。詳しくは、 151 ページの『第 19 章 DB2 Alphablox コンソール・コマンド』と 20 ページの『コンソール』を参照してください。

DB2 Alphablox Cube Manager

DB2 Alphablox Cube Manager には、DB2 Alphablox キューブの停止や開始などの管理作業を実行するためのインターフェースが用意されています。 DB2 Alphablox Cube Server の詳細については、「*DB2 Alphablox Cube Server 管理者用ガイド*」を参照してください。

クラスター・マネージャー

クラスター・マネージャーは、クラスター環境で DB2 Alphablox の複数のノードを実行するときに行われるノード間の通信を制御します。クラスター構成は、DB2 Alphablox システムのスケーラビリティーを向上させて、多数のユーザーをサポートするために使用します。クラスター環境で DB2 Alphablox を実行するためのセットアップ方法については、 147 ページの『第 18 章 クラスター環境の使用』を参照してください。

DB2 Alphablox の使用を開始するための作業のリスト

ここでは、DB2 Alphablox の使用を開始するために必要な基本的な作業を簡単にまとめます。それぞれの作業の詳細については、*相互参照* の欄にある資料やセクションを参照してください。

作業	相互参照
1. DB2 Alphablox の実行環境について理解し、構成の詳細を決定します (例えば、IBM WebSphere、BEA WebLogic、Apache Tomcat のうちのどれを使用するか、など)。	1 ページの『第 1 章 DB2 Alphablox の概要』
2. DB2 Alphablox のインストールと構成を行います。	*インストール・ガイド*
3. DB2 Alphablox ホーム・ページについて理解します (そのホーム・ページから、アプリケーションの作成、ユーザーの作成、セキュリティー・ポリシーのセットアップなどの管理作業を行います)。	19 ページの『第 3 章 DB2 Alphablox ホーム・ページ』
4. データを配置するデータベース環境について理解します。	それぞれのデータベース環境の資料やナレッジ・ワーカー
5. DB2 Alphablox アプリケーションの開発環境について理解します。	15 ページの『第 2 章 DB2 Alphablox アプリケーション』と*開発者用ガイド*
6. アプリケーション定義を作成します。	31 ページの『第 6 章 アプリケーションの定義』
7. JSP テクノロジー、Blox カスタム・タグ・ライブラリー、DB2 Alphablox API を使用してアプリケーションの開発とテストを行います。	*開発者用ガイドと開発者用リファレンス*
8. アプリケーションを社内に展開します。	ユーザー、社内イントラネット、インターネット

第 2 章 DB2 Alphablox アプリケーション

DB2 Alphablox アプリケーションを作成すれば、社内のイントラネット環境やインターネットを利用して、ブラウザーからデータにアクセスして分析を行えます。DB2 Alphablox アプリケーションは、高度にカスタマイズ可能であり多くの型の企業に統合できるため、「DB2 Alphablox アプリケーション」と名づけることができる広範な種類のアプリケーションがあります。この章では、そのようなアプリケーションのタイプや DB2 Alphablox アプリケーションを構成するコンポーネントについて解説します。 DB2 Alphablox アプリケーションの詳細については、*開発者用ガイド*を参照してください。

DB2 Alphablox アプリケーションのタイプ

DB2 Alphablox アプリケーションは、J2EE に準拠した Web アプリケーションであり、DB2 Alphablox 環境で稼働します。 DB2 Alphablox アプリケーションは、Web ブラウザーから実行でき、ほとんどの Web アプリケーション・テクノロジーと組み合わせることができるので、機能や外観は実に多彩です。最も基本的なレベルの DB2 Alphablox アプリケーションは、分析データを対話式で Web ページに表示します。ユーザーは、そのデータから貴重なビジネス情報を入手できます。データは社内のあらゆるデータベースに存在します。

Web アプリケーションの DB2 Alphablox コンポーネントに関して、このセクションでは、DB2 Alphablox アプリケーションの以下のカテゴリーについて簡単に説明します。

- 『データ表示モード』
- 16 ページの『Relational Reporting アプリケーション』

DB2 Alphablox のさまざまな部品を 1 つのアプリケーションとして組み合わせることも可能です。例えば、同じアプリケーションでチャートとスプレッドシートの両方を表示する、といった具合です。アプリケーションで実行できる作業や、そのようなアプリケーションを開発する方法の詳細については、*開発者用ガイド* と *Relational Reporting 開発者用ガイド* を参照してください。

データ表示モード

DB2 Alphablox アプリケーションでは、さまざまな形態でデータを表示できます。データをグリッドで表示することも、チャートで表示することも、その両方で表示することも可能です。さらに、データベースからデータを抽出して、ドロップダウン・リストやメニューなどのアプリケーション部品にそのデータを設定することもできます。

DB2 Alphablox DHTML クライアントは、 HTML、 JavaScript、Cascading Style Sheets (CSS) などの最新の Dynamic HTML テクノロジーを利用して、使いやすくカスタマイズ性に優れたグラフィカル・ユーザー・インターフェースによる幅広いデータ分析機能をサポートしています (グラフィカル・ユーザー・インターフェースには、右クリックによるコンテキスト・メニューもあります)。 DHTML クライ

アントの大きな利点の 1 つは、Blox API と Blox UI モデルによって実現されている拡張性の高さです。 DHTML クライアント・モードを利用するには、最新の Microsoft Internet Explorer ブラウザーが必要ですが、それ以外の Web ブラウザー・プラグインは不要です。クライアント・マシンをインストールすることも、Java クラス・ファイルをダウンロードすることも必要ありません。

PDF への変換

DB2 Alphablox アプリケーションでは、PDF ファイルにレポートを書き出して、高品質の出力を作成することもできます。そのために、「PDF に出力」機能は必要とする全てのエレメント (例えば会社のロゴや定形文面など) を組み込んだ PDF テンプレートを使用して、生のデータを取り込み、高品質のレポートを組み込んだ PDF ファイルを作成します。その PDF ファイルは、印刷したり、E メールで送信したり、Web サイトに配置したりして配布できます。 PDF の機能は、すべてのクライアント・レンダリング・モードで使用できます。

XML を使用したカスタム・レンダリング・モード

DB2 Alphablox には、XML 形式のデータにアクセスするための API が用意されています。開発者はその API を使用して、必要に応じてデータのカスタム・レイアウトを作成したり、カスタム・テクニックを駆使してデータを思いどおりに操作したりできます。これは、他の表示方式では対応できない特殊なニーズがある場合に便利な機能です。

Relational Reporting アプリケーション

ReportBlox コンポーネントを使用すれば、リレーショナル・データベースのデータに基づいてレポートを生成するための DB2 Alphablox アプリケーションを作成できます。このレポートは、カスタマイズ性に優れており、HTML のテーブルに表示できます。Cascading Style Sheets (CSS) のテクノロジーを使用すれば、レポートを簡単にカスタマイズして、外観を柔軟に変更できます。

ReportBlox コンポーネントを使って作成されたレポートは、それらをカスタマイズするためのユーザー・インターフェースを備えた対話モードや、プログラムに従ってそれらを変更するための包括的な API を使うことができます。 ReportBlox コンポーネントの詳細については、 *Relational Reporting 開発者用ガイド* を参照してください。

DB2 Alphablox アプリケーションのコンポーネント

DB2 Alphablox アプリケーションは J2EE アプリケーションなので、J2EE アプリケーションで使用できるあらゆるコンポーネントを利用できます。このセクションでは、DB2 Alphablox アプリケーションの以下の基本的なコンポーネントだけを取り上げます。

- 17 ページの『JavaServer Pages (JSP)』
- 17 ページの『Blox コンポーネント』

DB2 Alphablox アプリケーションのディレクトリー構造の詳細については、 31 ページの『DB2 Alphablox のアプリケーション定義』を参照してください。

JavaServer Pages (JSP)

アプリケーション開発者は、アプリケーションに DB2 Alphablox 機能を提供するために JavaServer Pages (JSP) テクノロジーのメカニズムをよく使用します。対話式 Web ページの開発では、アプリケーションのニーズに応じて、JSP、Blox カスタム・タグ・ライブラリー、HTML、 Java または JavaScript のコードを使用します。JSP ページのコンパイルは、実行時にアプリケーション・サーバーで行われます (パフォーマンス上の理由から、ほとんどのアプリケーション・サーバー構成では、初回のアクセス時にのみページのコンパイルが行われ、それ以降のページ要求については、プリコンパイルされているページが使用されます)。その後、DB2 Alphablox がすべての DB2 Alphablox 要求を処理し、Web サーバーからユーザーに動的コンテンツが戻される、という流れになります。

Blox コンポーネント

Blox コンポーネントとは、データ・ソースへのアクセスや、グリッドやチャートによるデータ表示などの機能を提供する DB2 Alphablox コンポーネントです。 JSP ページに Blox コンポーネントを追加するときには、Blox カスタム・タグ・ライブラリーを使用します。 Blox タグ・ライブラリーを使用すれば、Blox のさまざまな局面を制御できます。例えば、ChartBlox の属性を変更して、表示するチャートのタイプを指定する、といったことが可能です。

Java または JavaScript を使用して、Blox コンポーネントをプログラマチックに操作することもできます。 JavaScript を使用して Blox API にアクセスする場合は、コードがブラウザーで解釈されます。 Java を使用して Blox API にアクセスする場合は、コードが JSP エンジン (アプリケーション・サーバー) によってコンパイルされてから、サーバーで実行されます。

Blox の用途の詳細については、 7 ページの『DHTML クライアント』を参照してください。 DB2 Alphablox アプリケーションで Blox コンポーネントを使用する方法や、DHTML クライアントの使用時に利用できるコンポーネントのタイプの詳細については、 *開発者用ガイドと開発者用リファレンス* を参照してください。

第 3 章 DB2 Alphablox ホーム・ページ

この章では、DB2 Alphablox ホーム・ページについて説明します。コマンド行インターフェースの詳細については、 151 ページの『第 19 章 DB2 Alphablox コンソール・コマンド』を参照してください。

DB2 Alphablox ホーム・ページの概要

DB2 Alphablox は、初期インストール時に自動的に構成されます。管理者は、インストール後に DB2 Alphablox の始動構成、プロパティー、ポートを設定したり変更したりできます。 DB2 Alphablox の管理には、DB2 Alphablox ホーム・ページの「**管理**」タブか、コンソール・コマンド行インターフェースを使用します。 DB2 Alphablox 管理ページにアクセスするには、ブラウザー・ウィンドウで以下の URL アドレスのいずれかを入力します。

```
http://<servername>/AlphabloxAdmin/home/
http://<servername>/AlphabloxAdmin/
http://<servername>/AlphabloxAdmin (iPlanet を除く)
```

<servername> は、DB2 Alphablox が実行されるサーバーの名前とポート番号です。

「アプリケーション」タブ

DB2 Alphablox ホーム・ページを開くと、デフォルトで「アプリケーション」タブが選択されます。

ユーザーが DB2 Alphablox ホーム・ページにアクセスすると、デフォルトで「**アプリケーション**」タブが表示されます。「**アプリケーション**」 タブには、ユーザーによるアクセスが可能なアプリケーションと選択可能な保管済みのアプリケーション状態が表示されます。他のタブの表示やアクセスに必要な権限がないユーザーには、「**アプリケーション**」タブしか表示されません。

アプリケーションを起動するには、該当するリンクをクリックします。アプリケーションは、デフォルトで DHTML モードで起動します。ドロップ・リストから「**デフォルト・アプリケーション状態**」を選択すると、アプリケーション定義でデフォルトの状態として定義されている状態でアプリケーションがロードされます。

その他のアプリケーション

DB2 Alphablox で定義されているその他のアプリケーションもすべて「**アプリケーション**」 タブに表示されます。アプリケーションを起動するには、そのリンクをクリックします。アプリケーションは、デフォルトで DHTML モードで起動します。ドロップ・リストから「**デフォルト・アプリケーション状態**」を選択すると、アプリケーション定義でデフォルトの状態として定義されている状態でアプリケーションがロードされます。

「管理」タブ

DB2 Alphablox ホーム・ページに「**管理**」タブが表示されるのは、認証を受けたユーザーに管理特権がある場合に限られます。そのタブをクリックすると、「**一般**」ページが開き、そのページで管理者は、DB2 Alphablox の外観の定義や変更を行えます。「**管理**」 タブには、以下のセクションが含まれています。

- 「一般」
- 22 ページの『「グループ」』
- 22 ページの『「ユーザー」』
- 22 ページの『「役割」』
- 22 ページの『「アプリケーション」』
- 22 ページの『「データ・ソース」』
- 22 ページの『DB2 Alphablox キューブ』

「一般」

管理者は、「**管理**」 タブの「**一般**」リンクから、DB2 Alphablox のコア機能を構成するためのインターフェースを利用できます。

「一般プロパティー」

「**一般プロパティー**」セクションでは、DB2 Alphablox の始動パラメーターや、DB2 Alphablox Cube Server で使用できる DB2 Alphablox キューブの最大数を設定するパラメーターを構成します。これらの作業を実行する方法の詳細については、83 ページの『DB2 Alphablox の管理タスク』を参照してください。 DB2 Alphablox Cube Server の詳細については、 *DB2 Alphablox Cube Server 管理者用ガイド*を参照してください。

「カスタム・プロパティー」

カスタム・プロパティーを定義する方法の詳細については、 89 ページの『カスタム・プロパティーの定義』を参照してください。

「ランタイム管理」

「**ランタイム管理**」セクションでは、DB2 Alphablox のライセンス、アプリケーション・セッション、コメント集合、PDF レポート (DHTML のみ)、DB2 Alphablox キューブの管理を行います。 DB2 Alphablox キューブの作成と保守の方法については、「*DB2 Alphablox Cube Server 管理者用ガイド*」を参照してください。

コンソール

「**コンソール・セッションの開始 (Start Console Session)**」リンクをクリックすると、コンソール・ウィンドウが開きます。管理者はここからテキスト・コマンドを入力して、通常はユーザー・インターフェースから実行する管理機能のほとんどを実行できます。コンソールの使用方法については、 151 ページの『第 19 章 DB2 Alphablox コンソール・コマンド』を参照してください。

注: Telnet コンソール・ユーザー (DB2 Alphablox ホーム・ページの「**管理**」タブの「**一般**」リンクの「**Telnet コンソール (Telnet Console)**」リンクで定義されているユーザー) も、Telnet 端末ソフトウェアからコンソールにアクセスできます。 Microsoft Windows® オペレーティング・システムを実行するするほとんど

のシステムの場合、 Telnet ソフトウェアにアクセスするには、 Windows の「**スタート**」メニューから、「**すべてのプログラム**」 フォルダー、「**アクセサリ**」 グループ、「**Telnet**」ショートカットの順を選択するか、「**スタート**」メニューから「**ファイル名を指定して実行**」ショートカットを選択して、**telnet** と入力します。

ポータル・テーマ・ユーティリティー

注: DB2 Alphablox が WebSphere Portal Server 上にインストールされている場合のみ、「ポータル・テーマ・ユーティリティー (Portal Theme Utility)」オプションがメニューに現れます。

「**ポータル・テーマ・ユーティリティー (Portal Theme Utility)**」リンクは、DB2 Alphablox Blox コンポーネントを含む WebSphere ポータル・アプリケーションで使用するポータル・テーマを作成する場合にクリックします。このユーティリティーを使用すると、指定された WebSphere Portal Server テーマからのスタイル・プロパティーを、指定された DB2 Alphablox からのスタイル・プロパティーとマージすることにより、新しい DB2 Alphablox ポータル・テーマを作成することができます。新規のテーマは、ポータル・アプリケーションで Blox コンポーネント (例えば GridBlox または ChartBlox) を最適に表示するために使用することができます。

ポータル・テーマ・ユーティリティーのオプションのポートレット・バージョンを、ポータル・アプリケーションで使用することができます。このポートレットは、DB2 Alphablox インストールの `installableApps` ディレクトリーにある、`AlphabloxAdminPortlets.war` ファイルで見つけることができます。

ポータル・テーマ・ユーティリティーの使用に関するヘルプについては、このユーティリティーの「ヘルプ」ボタンを押すと表示されるオンライン・ヘルプを参照してください。ポータル・テーマ・ユーティリティーのポートレット・バージョンでは、ヘルプ・モードを選択するとオンライン・ヘルプが開きます。

アプリケーション移行

注: DB2 Alphablox が Apache Tomcat サーバー上にインストールされている場合のみ、「アプリケーション移行ユーティリティー (Application Migration Utility)」オプションがメニューに現れます。

「**アプリケーション移行 (Application Migration)**」リンクをクリックして、Tomcat 5.5 用の Alphablox アプリケーション移行ユーティリティーにアクセスします。このユーティリティーは、選択された DB2 Alphablox アプリケーションの Apache Tomcat 3.2.4 から Apache 5.5 への移行を支援します。移行されるアプリケーションごとに、ユーティリティーは以下のことを行います。

- (必要な場合に) アプリケーションを、現行の Alphablox インストール・システムの *webapps* ディレクトリーにコピーします。
- アプリケーションを Apache Tomcat 5.5 にデプロイします。
- (必要な場合に) アプリケーションを Alphablox にインポートします。
- Alphablox タグ・ライブラリー記述子 (TLD) ファイルを現行リリース用に更新します。
- アプリケーションの *web.xml* ファイルを Servlet 2.4 仕様に更新します。

移行済みアプリケーションは Apache Tomcat 5.5 インストール・システムで DB2 Alphablox と共に適切に稼働するはずです。

「グループ」

「**管理**」タブの「**グループ**」リンクを使用して、グループの定義、グループへのユーザーの割り当てなどを行います。グループをセットアップする方法の詳細については、 59 ページの『第 9 章 グループの定義』を参照してください。

「ユーザー」

「**管理**」タブの「**ユーザー**」リンクを使用して、ユーザーの定義、パスワードのリセット、ユーザーへのグループ・メンバーシップの割り当てなどを行います。ユーザーをセットアップする方法の詳細については、55 ページの『第 8 章 ユーザーの定義』を参照してください。

「役割」

「**管理**」タブの「**役割**」リンクを使用して、役割の作成や変更を行います。役割を使用して、どのアプリケーションに対してどのユーザーやグループにアクセス権限を与えるかを制御できます。役割をセットアップする方法の詳細については、63 ページの『第 10 章 役割の定義』を参照してください。

「アプリケーション」

「**管理**」タブの「**アプリケーション**」リンクを使用して、DB2 Alphablox に存在するアプリケーションの状態の作成や変更を行います。デフォルト状態の定義、役割の設定、「アプリケーション」タブに表示するイメージの定義などを行えます。アプリケーションを定義するために必要な作業の詳細については、 31 ページの『第 6 章 アプリケーションの定義』を参照してください。

「データ・ソース」

「**管理**」タブの「**データ・ソース**」リンクを使用して、DB2 Alphablox で実行する DB2 Alphablox アプリケーションで使用可能なデータ・ソースの定義や変更を行います。 OLAP データベース (IBM DB2 OLAP Server™、 Hyperion Essbase、Microsoft Analysis Services など)、リレーショナル・データベース (IBM DB2 UDB、 Oracle、Sybase、 Microsoft SQL Server など)、DB2 Alphablox キューブにアクセスするためのデータ・ソースを定義できます。データ・ソースの定義方法や変更方法の詳細については、 45 ページの『第 7 章 データ・ソースの定義』を参照してください。

DB2 Alphablox キューブ

「**管理**」タブの「**キューブ**」リンクを使用して、DB2 Alphablox キューブの定義や変更を行います。 DB2 Alphablox キューブは、リレーショナル・データベースに保管されているデータをマルチディメンション形式で表示します。 DB2 Alphablox キューブと DB2 Alphablox Cube Server の詳細については、 *DB2 Alphablox Cube Server 管理者用ガイド* を参照してください。

「アセンブリー」タブ: Application Studio

DB2 Alphablox ホーム・ページの「**アセンブリー**」タブをクリックすると、DB2 Alphablox Application Studio が開きます。 Application Studio の開始ページには、サンプルやツールを含んだページへのリンクが設定されています。開発者は、これらのページを使用して、DB2 Alphablox アプリケーションを短時間で作成できます。 Application Studio のフレームワークには、アプリケーションを簡単に作成するための使いやすいワークベンチ・ツールや実用的なコード例が用意されています。 Application Studio の詳細については、 *開発者用ガイド* を参照してください。

「ワークベンチ」

「**アセンブリー**」タブの「**ワークベンチ**」ページには、**クエリー・ビルダー**へのリンクが設定されています。クエリー・ビルダーは、開発者がさまざまなデータ・ソースに対して対話式に照会を実行し、結果をグリッドやチャートで表示するためのアプリケーションです。

「例」

「**アセンブリー**」タブの「**例**」ページには、JSP アプリケーションの例やコード・サンプルなどへのリンクが設定されています。多くの実用的なコード例を含んだ *Blox Sampler* も用意されています。これらのコード・サンプルは、DB2 Alphablox によるアプリケーション開発の学習に役立つばかりか、独自のカスタム・アプリケーションに直接カット・アンド・ペーストして利用することもできます。

サンプル・データ

DB2 Alphablox には、*Blox Sampler* アプリケーションで使用できるサンプル・データも用意されています。サンプル・データは、「Quality Chocolate Company (QCC)」という名前のデータベースであり、マルチディメンション・データベース用のバージョンとリレーショナル・データベース用のバージョンがあります。サンプル・データをインストールする方法の詳細については、「*インストール・ガイド*」の “サンプル・データのロード”を参照してください。

右上隅にあるリンク

DB2 Alphablox ホーム・ページのそれぞれのタブの右上隅には、以下の 2 つのリンクがあります。

- 『「マイ・プロファイル」』
- 24 ページの『「ヘルプ」』

「マイ・プロファイル」

「**マイ・プロファイル**」リンクをクリックすると、「**プロファイル**」ページが開きます。ユーザーは、このページから、プロファイルの説明、パスワード、E メール・アドレスの変更や、ローカル・クラス (JAR) ファイルのインストールやアンインストールを行えます。

「ヘルプ」

「**ヘルプ**」メニューには、以下のリンクがあります。

- 「**ヘルプ**」。このリンクから、そのページ (つまり、そのボタンをクリックしたページ) のコンテキストに依存したオンライン・ヘルプを利用できます。オンライン・ヘルプによって、各ページのインターフェースの使用方法を確認できます。
- 「**オンライン文書**」。このリンクをクリックすると、新しいブラウザー・ウィンドウ DB2 Alphablox オンライン文書が開きます。オンライン文書には、Web ベースの HTML 版と PDF 版の DB2 Alphablox 資料のセットが含まれています。
- 「**サーバー・サイド API 解説書 (Server-Side API Reference)**」。このリンクをクリックすると、サーバー・サイド Java API の Javadoc が開きます。
- 「**DB2 Alphablox について**」。このリンクから、DB2 Alphablox の著作権情報を確認できます。

注: `AlphabloxAdministrator` 役割のメンバーではないユーザーには、一部のリンクが表示されません。

第 4 章 基本管理タスク

DB2 Alphablox には、サーバー環境を管理するための広範囲の機能が組み込まれています。 DB2 Alphablox ホーム・ページから、サーバーのすべての局面を一元的に管理できます。このセクションでは、DB2 Alphablox の開始、アクセス、構成、停止のための手順を説明します。

DB2 Alphablox の開始

IBM WebSphere や BEA WebLogic などの商用のアプリケーション・サーバーと一緒に DB2 Alphablox を実行している場合は、DB2 Alphablox がそのアプリケーション・サーバーに統合されます。アプリケーション・サーバーを始動すると、DB2 Alphablox も自動的に開始されます。 IBM WebSphere Portal Server の場合、DB2 Alphablox はポータル・サーバーの始動時に開始されます。 WebLogic または WebSphere を始動する方法については、それぞれの製品の資料を参照してください。

DB2 Alphablox へのアクセス

DB2 Alphablox の管理は、DB2 Alphablox ホーム・ページの「**管理**」タブで行います。

ブラウザーから DB2 Alphablox にアクセスする場合は、以下の URL を入力します。

```
http://<serverName>/AlphabloxAdmin/home/
```

`<serverName>` は、DB2 Alphablox が実行されるサーバーの名前とポート番号です。

Windows の「**スタート**」メニューから DB2 Alphablox ホーム・ページの「**管理**」タブにアクセスするには、以下のようにします。

1. Windows の「**スタート**」メニューを開きます。
2. 「**すべてのプログラム**」フォルダーを選択します。
3. 「**DB2 Alphablox**」プログラム・グループを選択します。
4. DB2 Alphablox インスタンスのインスタンス名を選択します (デフォルト値は、**AlphabloxAnalytics**)。
5. 「**DB2 Alphablox ホーム・ページ (DB2 Alphablox Home Page)**」ショートカットを選択します。

DB2 Alphablox の停止

WebSphere アプリケーション・サーバーを使用している場合に DB2 Alphablox を停止するには、そのアプリケーション・サーバーを停止します。 WebSphere Portal Server の場合、ポータル・サーバーを停止すると DB2 Alphabloxも停止します。 Apache Tomcat インストール・システムの場合に DB2 Alphablox を停止するには、

コンソール・ウィンドウで実行している場合はシャットダウン・スクリプトを使用して停止し、Windows のサービスとして実行している場合は「サービス」コントロール・パネルから停止します。

Apache Tomcat インストール・システムでシャットダウン・スクリプトを使用して DB2 Alphablox を停止する

DB2 Alphablox をコンソール・ウィンドウ (Windows の場合: Windows の「**スタート**」メニュー、「**すべてのプログラム**」フォルダー、「**DB2 Alphablox**」グループ、インスタンス名 (デフォルト値は **AlphabloxAnalytics**)、「**DB2 Alphablox のシャットダウン (Shutdown DB2 Alphablox)**」ショートカットから DB2 Alphablox を開始するときに開くウィンドウ; Sun Solaris の場合: サーバーを始動したウィンドウ) で実行しているときに DB2 Alphablox を停止するには、以下の手順を実行します。

Windows システムの場合:

- 「**スタート**」メニュー、「**すべてのプログラム**」フォルダー、「**DB2 Alphablox**」グループ、インスタンス名 (デフォルト値は **AlphabloxAnalytics**)、「**DB2 Alphablox のシャットダウン (Shutdown DB2 Alphablox)**」ショートカットを実行します。

Linux、UNIX システムの場合:

- <db2alphablox_dir>/bin ディレクトリーから以下のコマンドを実行します (<db2alphablox_dir> は、DB2 Alphablox のインストール・ディレクトリーです)。

  ```
  ./StopAlphablox.sh
  ```

Apache Tomcat インストール・システムで「サービス」コントロール・パネルから DB2 Alphablox を停止する

DB2 Alphablox を Windows のサービスとして実行している場合は、以下の手順で DB2 Alphablox を停止します。

1. Windows 「**スタート**」メニュー、「**設定**」、「**コントロール パネル**」ショートカットから Windows のコントロール パネルを開きます。
2. 「**サービス**」コントロール・パネルをダブルクリックします。「サービス」ウィンドウが表示されます。
3. 実行中の DB2 Alphablox サービスごとに、サービスを選択して、「**停止**」ボタンをクリックします。
4. 「**はい**」ボタンをクリックして、サービスの停止を確定します。
5. 「**閉じる**」ボタンをクリックして、「サービス」ウィンドウを閉じます。

Essbase クライアント・ライブラリー・ユーティリティーの使用

DB2 Alphablox を DB2 OLAP Server または Hyperion Essbase データ・ソースとともに使用するには、適切な Essbase クライアント・ライブラリー・ファイルをインストールして、それらを DB2 Alphablox で使用できるようにします。 Essbase クライアント・ライブラリーは、Hyperion の Web サイト (http://support.hyperion.com/) から入手できます。

DB2 Alphablox にインストールされている Essbase クライアント・ライブラリーを変更または更新するには、以下の手順を実行します。

1. DB2 Alphablox で以下のディレクトリーを開きます。

   db2alpahblox_dir/bin/ ここで、`db2alpahblox_dir` は DB2 Alphablox がインストールされている場所です (例えば *c:¥alphablox¥analytics¥*)。
2. ChangeEssbase バッチ・ファイル (Windowsの場合は *ChangeEssbase.bat*、Linux® と UNIX® の場合は *ChangeEssbase.sh*) を見つけて、それを実行します。スクリプト・コンソール・ウィンドウが表示されます。
3. DB2 Alphablox がインストールされている場所の完全修飾パス (例えば、*c:¥alphablox¥analytics¥*) を入力して、**Enter** を押します。
4. 使用可能なクラス・ライブラリー・オプションのリストが表示されます。使用するクライアント・ライブラリーの Essbase バージョンに該当するオプションを選択して、**Enter** を押します。
5. 操作が成功すると、システムが再構成されたことを示すメッセージが表示されます。
6. 変更を有効にするために DB2 Alphablox を再始動します。

その他の管理タスクと情報

その他の管理タスクの詳細については、以下のセクションを参照してください。

- 1 ページの『第 1 章 DB2 Alphablox の概要』
- 151 ページの『第 19 章 DB2 Alphablox コンソール・コマンド』
- 19 ページの『第 3 章 DB2 Alphablox ホーム・ページ』
- 31 ページの『第 6 章 アプリケーションの定義』
- 89 ページの『カスタム・プロパティーの定義』
- 63 ページの『第 10 章 役割の定義』
- 45 ページの『第 7 章 データ・ソースの定義』
- 55 ページの『第 8 章 ユーザーの定義』
- 59 ページの『第 9 章 グループの定義』

第 5 章 クライアントの管理

管理者は、DHTML クライアントから DB2 Alphablox アプリケーションを使用することに関連したいくつかの問題を把握していなければなりません。

DHTML クライアントの管理

DB2 Alphablox の DHTML レンダリング・モードを使用する場合は、サポートされている Mozilla Firefox および Microsoft Internet Explorer Web ブラウザーの組み込み機能によって分析アプリケーションが表示されます。そのため、DHTML ベースのアプリケーションのサポートには、管理上の手間がほとんどかかりません。

サポートされている DHTML クライアント構成

Mozilla Firefox および Microsoft Internet Explorer Web ブラウザーのサポートされている DHTML クライアント構成については、 *インストール・ガイド* の『システム要件』のセクションを参照してください。

DHTML クライアントに関する問題

基本的に、Mozilla Firefox およびMicrosoft Internet Explorer から DB2 Alphablox アプリケーションを使用する場合は、ほとんど問題が発生しません。念のために把握しておくべきいくつかの問題を以下にまとめます。

CSS

まれなケースとして、Mozilla Firefox および Microsoft Internet Explorer Web ブラウザーで使用可能な拡張オプションに含まれている Cascading Style Sheets (CSS) の設定をユーザーが個人的に変更することがあります。 CSS にはカスケードの性質 (つまり、滝の流れのように上位の設定が下位の設定に影響を与える性質) があるので、CSS を幅広く利用している DHTML クライアントでは、これらの設定の変更がブラウザーの Blox コンポーネントの外観に影響を及ぼす可能性があります。

ポップアップ・ウィンドウ

DHTML クライアントの使用時にボタンやリンクをクリックしたときに、詳細情報を示したポップアップ・ウィンドウやダイアログが表示されない場合があります。Web ブラウザーに組み込まれたポップアップ・ウィンドウ・ブロッキング機能が、ポップアップ・ウィンドウの表示を抑止する可能性があります。また、オプションのブラウザー・ツール (例えば、Google ツールバー) およびファイアウォール・ソフトウェア (例えば、Zone Alarm) に、Web ブラウザーのポップアップ・ウィンドウの動作に影響を与える設定が含まれていることもあります。この問題を解決するには、その機能を使用不可にするか、ユーザー設定を変更して DB2 Alphablox アプリケーションのポップアップ・ウィンドウの表示を許可する必要があります。

第 6 章 アプリケーションの定義

この章では、J2EE のアプリケーション構造について解説し、DB2 Alphablox でアプリケーション定義の作成、変更、削除を行うための手順を示します。

DB2 Alphablox のアプリケーション定義

「**管理**」タブの「**アプリケーション**」ページには、DB2 Alphablox アプリケーションのリストが表示されます。新規の DB2 Alphablox アプリケーションを定義するには、「**作成**」ボタンをクリックします。アプリケーション定義の変更または削除を行うには、リストからアプリケーションを選択し、「**編集**」ボタンまたは「**削除**」ボタンを押します。

DB2 Alphablox の「管理」ページからアプリケーションを作成すると、DB2 Alphablox が DB2 Alphablox リポジトリーにアプリケーション定義を作成します。DB2 Alphablox リポジトリーでのアプリケーション名は、エンタープライズ・アプリケーション名と固有 ID (URI) の 2 つの部分から構成されます。例えば、MyApp という名前のアプリケーションを作成する場合、リポジトリーでのアプリケーションの名前は `MyApp_MyApp` になります。WebSphere および WebLogic アプリケーション・サーバーの場合、DB2 Alphablox は、アプリケーション・サーバーにインポートする必要がある、インストール可能なアプリケーションを作成します。

以下は、WebSphere アプリケーション・サーバーにインポートされる典型的な DB2 Alphablox アプリケーションのディレクトリー構造です。

```
<application>.ear
  META-INF/
    application.xml
    MANIFEST.MF
  <application>.war
    META-INF/
      MANIFEST.MF
    WEB-INF
      timeschema.dtd
      timeschema.xml
      web.xml
      classes/
      lib/ (for Alphablox 8.4.1 applications only)
         aastaglibs.jar
      tlds/ (for Alphablox 8.4 applications only)
        blox.tld
        bloxform.tld
        bloxlogic.tld
        bloxui.tld
```

JSP ファイル、 JavaScript ファイル、HTML ファイル、CSS ファイル、およびイメージ・ファイルを含むアプリケーション・ファイルは、通常 `<application>.war` ディレクトリーに配置されています。開発環境によっては、いくつかのファイルは `<application>.war` ディレクトリー内のネストされたフォルダーに入っている場合があります。JavaBeans コンポーネントを含むカスタム Java クラスは、`classes` ディレクトリーにあります。DB2 Alphablox アプリケーションに必要なタグ・ライブラリー記述子 (TLD) ファイルは、 DB2 Alphablox 8.4 アプリケーションでは *tlds*

ディレクトリーにあります。 DB2 Alphablox 8.4.1 アプリケーションでは、TLD ファイルは *aastaglibs.jar* にパッケージされていて、アプリケーションの *lib* ディレクトリーにあります。特殊なアプリケーションによっては時間スキーマ関連ファイル (`timeschema.dtd` および `timeschema.xml`) が必要になります。これらは、*WEB-INF* ディレクトリー内に直接配置されています。DB2 Alphablox がインストール可能なアプリケーションを作成すると、DB2 Alphablox アプリケーションの必要に応じて、デプロイメント記述子ファイル (`web.xml`) も変更されます。

アプリケーション名

DB2 Alphablox リポジトリーで使用されるアプリケーション名は、エンタープライズ・アプリケーション名と固有 ID の両方で構成されます。アプリケーションの作成後、特定のアプリケーションの編集ビューに移動し、自分のアプリケーションの命名情報を表示することができます。このリストには、DB2 Alphablox 名、J2EE アプリケーション名、モジュール名、コンテキスト・パス、および表示名が含まれます。

WEB-INF ディレクトリー

各 Web アプリケーションに、`WEB-INF` ディレクトリーがあります。このディレクトリーは、アプリケーションの作成時に自動生成され、アプリケーションに関するメタデータが入ります。通常の操作では、`WEB-INF` ディレクトリーの下にある内容を変更すべきではありません。アプリケーションの作成の間に、以下のファイルが `WEB-INF` ディレクトリーにインストールされます。

- `web.xml`
- DB2 Alphablox 8.4.1 では `aastaglibs.jar` (*lib* ディレクトリーにある)。 DB2 Alphablox 8.4 では個々の TLD ファイル (例えば *blox.tld*)。
- `timeschema.dtd`
- `timeschema.xml`

アプリケーションのデプロイメント記述子ファイル (`web.xml`) は、すべての J2EE アプリケーションの標準 XML ファイルであり、アプリケーションの属性を記述するマークアップを含みます。アプリケーション・サーバーは、アプリケーションを初期化するときに `web.xml` ファイルを読み取ります。

重要: `web.xml` ファイルに対する変更は、製品のアップグレード中に DB2 Alphablox によって自動的に実行されます。`web.xml` に間違った項目があると、アプリケーションで予期しない動作が発生する場合があるので、変更を加えるときには十分な注意が必要です。

Alphablox タグ・ライブラリー記述子 (TLD) ファイルは、Alphablox 8.4.1 では *aastaglibs.jar* にパッケージされています (Alphablox 8.4 では、ばらばらの TLD ファイル)。これによって開発者は、 Blox コンポーネントと DB2 Alphablox アプリケーションの作成に使用する DB2 Alphablox タグ・ライブラリーにアクセスできます。 2 つの時間関連ファイル (`timeschema.dtd` および `timeschema.xml`) が `WEB-INF` ディレクトリー内に直接保管されており、開発者はこれらのファイルを使用して、会計およびカレンダー期間を扱う特殊なアプリケーションを作成することができます。これらのファイルの詳細については、「*開発者用ガイド*」と「*開発者用リファレンス*」を参照してください。

新規アプリケーションの定義

新規アプリケーションを定義するには、以下のようにします。

注: アプリケーション・サーバーとして WebSphere を使用する場合は、アプリケーションを作成するために追加の手順が必要になります。詳しくは、34 ページの『WebSphere と一緒に実行する場合のアプリケーションの定義』を参照してください。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
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. 「**作成**」ボタンをクリックします。「**アプリケーションの作成**」ページが表示されます。
5. アプリケーションに固有の「**名前**」を入力します。英語以外の言語で実行している場合に使用できる文字は、A-Z、a-z、0-9、下線、特殊文字 (アクセント付き文字など) です。名前の表示では大/小文字の区別がありますが、認証の対象となる実際の名前には大/小文字の区別がありません。 *Public*、*Private*、*Properties* は予約語であり、オブジェクト名としては使用できません。
6. [オプション] 「**記述 (Description)**」フィールドにアプリケーションの簡単な説明を入力します。
7. [オプション] このアプリケーションの最初のページを配置する場所を示す、アプリケーション名への相対 URL アドレス (**ホーム URL**) と、このアプリケーションの識別イメージを指す別の相対 URL アドレス (**イメージ URL**) を入力します。その 2 つの値に基づいて、「**アプリケーション**」タブの下に 1 つの項目が作成されます (ユーザーはその項目からアプリケーションを起動します)。
8. 「**デフォルトの保管状態 (Default Saved State)**」フィールドで、希望する保管状態を指定するパラメーター名を入力します。ユーザーがアプリケーションの保管状態バージョンにアクセスするようにしたい場合は、アプリケーション開発者がパラメーター名を作成します。「デフォルトの保管状態 (Default Saved State)」が指定されない場合、リストアされる状態は、ブラウザーのシャットダウンまたはタイムアウト時のアプリケーション状態です。
9. 「**書き込み特権セキュリティー役割 (Write Privileges Security Role)**」 テキスト・ボックスに、このアプリケーションにアクセスするための既存の役割の名前を入力します。何も入力しないと、デフォルトのアクセス権が適用されます。
10. 「**セッション・タイムアウト**」に J2EE セッションのタイムアウト値 (ユーザー・セッションがタイムアウトになるまでのアイドル時間) を入力します。タイムアウト値は、分単位で指定します。
11. 自動保管されている最新の状態をロードするかどうかを指定するために、「**保管済みアプリケーション状態のリストアが使用可能**」ドロップ・リストから「**はい**」または「**いいえ**」を選択します。

12. 「**アプリケーション・ページでの組み込み (Include on Applications Page)**」オプションは、「アプリケーション」ページのビューで、このアプリケーションがユーザーに表示されるかどうかを決定します。デフォルト値は 「**はい**」 です。
13. [オプション] アプリケーションの「**デフォルトのレンダリング・モード**」を設定します。新規アプリケーションは、デフォルトで DHTML レンダリング・モードをサポートし、新規アプリケーションのディレクトリーに Alphablox タグ・ライブラリー記述子 (TLD) ファイルのコピーをインストールします。
14. [オプション] **ヘッダー・リンク**機能を使用して、グリッドに表示されるメンバーからのリンクをセットアップします (そのリンクから URL を開いたり、JavaScript を実行したりできます)。「**ヘッダー・リンク**」テキスト・ボックスに、固有のメンバー名と URL アドレスまたは他の有効な URL テキストを追加します (以下の構文を使用して各項目の間に改行を挿入します)。

    ```
    Member = URL
    ```

    例えば、IBM という名前のメンバーから `http://www.ibm.com` へのリンクを作成するには、「ヘッダー・リンク」テキスト・ボックスに以下のように入力します。

    ```
    IBM = http://www.ibm.com
    ```
15. 「**保管**」ボタンを押して新規アプリケーションを定義し、「アプリケーション」ページに戻ります。
16. 外部 Web サーバー (Apache HTTP Server、Microsoft IIS、または Sun iPlanet) と一緒に Apache Tomcat 構成を実行している場合は、アプリケーションを使用可能にするために、まず追加の手順を実行しなければなりません。詳しくは、39 ページの『外部 Web サーバーにアプリケーションを登録する』を参照してください。

WebSphere と一緒に実行する場合のアプリケーションの定義

IBM WebSphere Application Server を使用している場合は、アプリケーションを DB2 Alphablox で定義する前か後に、 WebSphere でも定義する必要があります。このセクションでは、WebSphere で Web アプリケーションを作成するために必要な基本手順を説明します。 WebSphere を使用してアプリケーションを作成する方法については、WebSphere の資料を参照してください。

WebSphere と一緒に実行する場合に、DB2 Alphablox アプリケーションを定義する方法は 2 つあります。

- 35 ページの『DB2 Alphablox でアプリケーションを作成してから WebSphere で登録する』
- 36 ページの『DB2 Alphablox に既存の WebSphere アプリケーションをインポートする』

どちらの方法でもかまいません。それぞれに合った便利な方法を選んでください。

DB2 Alphablox でアプリケーションを作成してから WebSphere で登録する

WebSphere と一緒に実行している DB2 Alphablox でアプリケーションを作成する場合、 DB2 Alphablox は、エンタープライズ・アーカイブ (EAR) ファイルにアプリケーション構造を作成します。 EAR ファイルは、以下のディレクトリーに作成されます。

```
<db2alphablox_dir>/installableApps/
```

ファイルの名前は、<application>.ear です (<application> は、アプリケーションの作成時にアプリケーションに指定した名前です)。

新しく作成したアプリケーションを使用する前に、以下の手順を実行して DB2 Alphablox でアプリケーション構造を作成し、それを WebSphere に登録する必要があります。

1. まず33 ページの『新規アプリケーションの定義』の説明に従って DB2 Alphablox で新規アプリケーションを定義します。 <db2alphablox_dir>/installableApps に EAR ファイルが作成されます。
2. WebSphere 管理コンソールを開き、「**アプリケーション**」>「**新規アプリケーションのインストール**」を選択します。
3. 「**アプリケーション・インストールの準備**」画面で「ブラウズ」ボタンをクリックして、以下のパスを選択します。

   ```
   <db2alphablox_dir>/installableApps/<newApplication>.ear
   ```

   <application> は、インストールするアプリケーションの名前です。「次へ」を押します。

   注: クラスター環境のみ: ネットワーク経由でブラウズして <newApplication>.ear ファイルを探す場合は、サーバー・パス設定を使用しなければならない場合があります。
4. 次の画面に「**デフォルトのバインディング・オプション (Default Bindings Options)**」 が表示されます。別のバインディングが必要でない限りデフォルト設定のままにして、「次へ」をクリックします。
5. 「**アプリケーション・セキュリティー警告**」画面が表示されます。このページのボタンをスクロールして、「継続」ボタンをクリックします。
6. 次のセクション (「**新規アプリケーションのインストール**」) には、以下の 5 つの手順がダイアログに表示されます。

   手順 1: インストール実行のためのオプションを指定する

   既存の設定のままにして、「次へ」をクリックします。

   手順 2: 仮想ホストを Web モジュールにマップする

   既存の値を受け入れて、「次へ」をクリックします。

   手順 3: モジュールをアプリケーション・サーバーにマップする

   既存の値を受け入れて、「次へ」をクリックします。

手順 4: セキュリティー役割をユーザー/グループにマップする

AlphabloxAdministrator 役割と AlphabloxUser 役割の 2 つが表示されます。それぞれの役割に少なくとも 1 つのユーザーを追加する必要があります。

AlphabloxAdministrator 役割の場合は、その役割の前にあるチェック・ボックスにチェック・マークを付け、「**ユーザーの検索**」ボタンまたは「**グループの検索**」ボタンをクリックして、管理ユーザーを追加します。少なくとも 1 つのユーザーを選択する必要があります。追加したユーザーまたはグループは、その役割の「マップされたユーザー」または「マップされたグループ」に表示されます。

AlphabloxUser 役割の場合は、その役割の「すべての認証済み (All Authenticated)」欄の下のチェック・ボックスにチェック・マークを付けます。これにより、認証済みユーザーはすべてアプリケーションにアクセスできるようになります。これらの設定を使用して、アプリケーション要件に応じて特定のユーザーまたはグループだけにアクセス権を制限することもできます。

終了したら、「次へ」をクリックします。

手順 5: サマリー

この画面の下部にスクロールして、「終了」ボタンをクリックします。

7. アプリケーションがインストールされ、「アプリケーション <Application> は正常にインストールされました (Application <Application> installed successfully)」というメッセージが表示されます。「**マスター構成に保管 (Save to Master Configuration)**」リンクをクリックします。
8. 「**マスター構成に保管 (Save to Master Configuration)**」ダイアログが表示されます。「保管」ボタンをクリックします。少しの待ち時間の後、「管理コンソール」ホーム・ページが再び表示されます。
9. 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**」を開き、そのアプリケーションを再始動します。

DB2 Alphablox ホーム・ページから内容を追加したり、アプリケーションを使用したり、そのプロパティーを変更したりできるようになります。

DB2 Alphablox に既存の WebSphere アプリケーションをインポートする

既存の WebSphere アプリケーションを DB2 Alphablox にインポートして、必要な DB2 Alphablox 設定とファイルを追加するには、42 ページの『既存の J2EE アプリケーションのインポート』の手順を実行します。

既存のアプリケーション定義の変更

このセクションでは、既存のアプリケーションを変更する方法を説明します。既存のアプリケーションを変更するには、以下のようにします。

1. admin ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。

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. 変更するアプリケーションをアプリケーションのリストから選択し、「**編集**」ボタンをクリックします。「**アプリケーションの編集**」ページが開きます。
5. 該当するプロパティーの値を変更します。

   注: WebSphere および WebLogic 構成: 「**名前**」フィールドを変更することによってアプリケーションの名前を変更することができますが、アプリケーション名を変更すると、DB2 Alphablox リポジトリーのアプリケーション名だけが変更され、関連する WebSphere または WebLogic のアプリケーション名やコンテキスト・パスは変更されません。WebSphere または WebLogic の関連アプリケーション・プロパティーを変更するには、アプリケーション・サーバーの管理コンソールを使用して、基本アプリケーション・コンテキスト・パスの名前を新しい名前に合わせて変更し、基本アプリケーション文書の基本ディレクトリーの名前を変更します。

   注: Apache Tomcat 構成の場合: 「**名前**」フィールドを変更すると、アプリケーション名を変更することができます。アプリケーションの名前を変更すると、DB2 Alphablox リポジトリー内のアプリケーションのディレクトリー名と、そのアプリケーションを参照するすべてのブックマークの名前も変更されます。
6. 「**保管**」ボタンをクリックしてアプリケーションのプロパティーの値を変更し、「**アプリケーション**」ページに戻ります。
7. 外部 Web サーバー (Apache HTTP Server、Microsoft IIS、Sun iPlanet) と一緒に Apache Tomcat 構成を実行している環境で、アプリケーションの名前を変更する場合は、それぞれの構成に合わせて必要な追加手順も実行しなければなりません。詳しくは、39 ページの『外部 Web サーバーにアプリケーションを登録する』を参照してください。

既存のアプリケーション定義の削除

既存のアプリケーションを削除するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
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. 削除するアプリケーションをアプリケーションのリストから選択し、「**削除**」ボタンをクリックします。

   重要: アプリケーションを削除すると、DB2 Alphablox からアプリケーション定義が除去されるので、そのアプリケーション名は、アプリケーションのリストに表示されなくなります。ただし、アプリケーション・フォルダーとその関連ファイルは除去されないので、手動で削除する必要があります。

重要: WebLogic で DB2 Alphablox アプリケーションを削除する場合:
`autoDeploy` が使用可能になっている場合、WebLogic サーバー上で DB2 Alphablox アプリケーションを完全に削除するには、以下の手順を実行します。

a. WebLogic admin コンソールを使用して、autoDeploy 機能を使用不可にします。
b. DB2 Alphablox からアプリケーション定義を削除します。 DB2 Alphablox は、WebLogic サーバー上でアプリケーションの登録を抹消します。
c. アプリケーション・フォルダーを削除します。
d. 必要に応じて、autoDeploy 機能を再び使用可能にします。

autoDeploy が使用可能になっていない場合は、手順 b と c だけを実行してください。

5. 外部 Web サーバー (Apache HTTP Server、Microsoft IIS、Sun iPlanet) と一緒に Apache Tomcat 構成を実行している場合は、アプリケーションを削除した後で、それぞれの構成に合わせて必要な手順を実行しなければなりません。詳しくは、39 ページの『外部 Web サーバーにアプリケーションを登録する』を参照してください。

WebLogic クラスター使用時のアプリケーションの定義

BEA WebLogic クラスター環境で DB2 Alphablox を使用している場合は、以下の手順で新規アプリケーションを作成します。

1. 33 ページの『新規アプリケーションの定義』の説明に従って、アプリケーションの下の「管理」タブから DB2 Alphablox でアプリケーションを定義します。
2. 新しく作成したアプリケーションのディレクトリー全体を WebLogic 管理サーバーにコピーします。望ましいのは、アプリケーション・ディレクトリーと同じレベルのドメイン直下のディレクトリーに配置することです。
3. 必要なすべての JSP ファイルを (*WEB-INF* ディレクトリーと同じレベルで) アプリケーション・ディレクトリーに追加します。
4. WebLogic コンソールで、新しく追加した Web アプリケーションを構成します。 WebLogic サーバーでは、新規アプリケーション・ディレクトリーを表示してから、「**選択**」をクリックするか、適切なレベルにナビゲートします。
5. 展開のターゲットとなるクラスターを選択してから、「**構成と展開 (Configure and Deploy)**」ボタンをクリックします。
6. アプリケーションを展開した後、Web アプリケーションとして AlphabloxServer を選択します。
7. 「展開 (Deploy)」タブを選択し、AlphabloxServer アプリケーションを再展開します。
8. 再展開を実行すると、DB2 Alphablox がいったん停止してから、再始動します。
9. DB2 Alphablox が再始動したら、アプリケーションにアクセスできます。

注: それぞれの管理対象サーバーに JSP ファイルをコピーする必要はありません。アプリケーションを WebLogic に登録するときに、自動的にコピーされるからです。その動作の詳細やその動作を変更する方法については、WebLogic の資料を参照してください。

外部 Web サーバーにアプリケーションを登録する

外部 Web サーバーと一緒に DB2 Alphablox の Apache Tomcat 構成を実行している場合は、アプリケーションの作成または削除を行うたびにいくつかの手順を実行する必要があります。

Web サーバーは静的ページに対して、アプリケーション・サーバーは動的ページに対して、それぞれ効率的にサービスを提供するように設計されているので、Web サーバーとアプリケーション・サーバーの両方のパフォーマンスを最高度に引き出すには、ここで説明する手順を実行する必要があります。その手順を実行しないと、アプリケーション・サーバーは、静的ページの処理を不必要に実行せざるを得なくなります。

ここでは、以下の構成について説明します。

- 39 ページの『Apache HTTP Server でのアプリケーションの登録』
- 40 ページの『Microsoft Internet Information Server (IIS) Web サーバーでのアプリケーションの登録』
- 40 ページの『Alphablox 8.4 での Sun iPlanet Web サーバー上のアプリケーションの登録』

Apache HTTP Server でのアプリケーションの登録

Apache Tomcat 5.5 での DB2 Alphablox 8.4.1

Apache Tomcat 構成を伴う DB2 Alphablox アプリケーション定義を作成または削除するときには、両方のサーバーが正しく停止され、再始動されるまで、変更は Apache HTTP Server に反映されません。 DB2 Alphablox が再始動されるときに、新しい Apache 構成ファイル (`mod_jk.conf`) が Apache HTTP Server に書き込まれます。

DB2 Alphablox アプリケーションの追加または削除を登録するには、以下のようにします。

1. Apache HTTP Server と Apache Tomcat サーバー (DB2 Alphablox を実行している) の両方をシャットダウンします。
2. Apache Tomcat サーバーを再始動します。
3. Apache HTTP Server を再始動します。

両方のサーバーを再始動した後、アプリケーションの変更が Apache HTTP Server で認識されます。

Apache Tomcat 3.2.4 での DB2 Alphablox 8.4

Apache HTTP Server を使用して Apache Tomcat 構成を実行している場合は、DB2 Alphablox アプリケーション定義の作成または削除を行うたびに Apache Web サーバーを再始動しなければなりません。この再始動を行わなければ、DB2 Alphablox アプリケーション定義の作成または削除の時点で Apache 構成ファイル (`mod_jk.conf-alphablox`) に対して DB2 Alphablox が行う変更を Apache は読み取れません。

Apache HTTP Servers の停止と再始動

Apache を停止するには、シャットダウン・コマンドを実行するか (コンソール・ウィンドウで実行中の場合)、「**サービス**」コントロール・パネルからそのサービスを選択し、「**停止**」ボタンをクリックします (Windows のサービスとして実行中の場合)。

Apache を再始動するには、始動コマンドを実行するか (コンソール・ウィンドウで実行中の場合)、「**サービス**」コントロール・パネルからそのサービスを選択し、「**開始**」ボタンをクリックします (Windowsのサービスとして実行中の場合)。

Apache Web サーバーの停止と開始の詳細については、Apache の資料を参照してください。

Microsoft Internet Information Server (IIS) Web サーバーでのアプリケーションの登録

IIS Web サーバーを使用してスタンドアロン構成を実行している場合は、DB2 Alphablox アプリケーション定義の作成または削除を行うたびに以下の 2 つの作業を実行しなければなりません。

- DB2 Alphablox アプリケーションの仮想ディレクトリーの追加または削除を行います。
- Web サーバーを再始動します。

Alphablox 8.4 での Sun iPlanet Web サーバー上のアプリケーションの登録

Sun iPlanet Web サーバーを DB2 Alphablox 8.4 で (8.4.1 ではない) 使用してスタンドアロン構成を実行している場合は、DB2 Alphablox アプリケーション定義の作成または削除を行うたびに以下の 2 つの作業を実行しなければなりません。

- 『文書ディレクトリーの追加』
- 41 ページの『DB2 Alphablox スタイルへの適切な割り当ての追加』

文書ディレクトリーの追加

それぞれの DB2 Alphablox アプリケーションごとに、文書ディレクトリーを Web サーバーに追加する必要があります (Web サーバーは、そのディレクトリーからページを提供します)。 iPlanet Web サーバーにアプリケーション・ディレクトリーを追加する手順は、以下のとおりです。

1. 「iPlanet Web Server 管理 (iPlanet Web Server Administrative)」ページを開きます。デフォルトでは、iPlanet の実行サーバーのポート 8888 からこのページにアクセスできますが、このポートはシステムによって異なる場合があります。
2. ドロップダウン・リストからサーバーを選択し、「**管理**」ボタンをクリックします。
3. 右上隅にある「**クラス・マネージャー (Class Manager)**」リンクをクリックします。
4. 「**コンテンツ管理 (Content Mgmt)**」タブをクリックします。
5. 「**追加文書ディレクトリー (Additional Document Directories)**」リンクをクリックします。

6. 「**URL 接頭部 (URL prefix)**」テキスト・ボックスに、アプリケーションのコンテキスト名を入力します (以下の例のように末尾にスラッシュ (/) を付けます)。

   ```
   MyApplication/
   ```

   重要: スラッシュはコンテキスト名の先頭に付けないでください。そのようにすると、iPlanet 構成ファイルが破壊される場合があります。

7. 「**ディレクトリーにマップ (Map to Directory)**」テキスト・ボックスに、以下の例のようにアプリケーションのディレクトリーを入力します。

   ```
   d:/Alphablox4/webapps/MyApplication
   ```

   ヒント: パス名に使用できるのはスラッシュ (/) だけです。iPlanet には、パス名の円記号 (¥) を正しく処理できないバージョンがあります。

8. 「**スタイルの適用**」ドロップダウン・リストから「**なし**」を選択します。
9. 「**OK**」ボタンをクリックしてから、変更を保管します。

DB2 Alphablox スタイルへの適切な割り当ての追加

それぞれのアプリケーションごとに、3 つの URL 接頭部ワイルドカード文字を **alphablox** スタイルに割り当てる必要があります。この割り当てによって、該当する要求が DB2 Alphablox に転送されます。 **alphablox** スタイルに新規割り当てを追加するには、以下の手順を実行します。

1. 「iPlanet Web Server 管理 (iPlanet Web Server Administrative)」ページを開きます。デフォルトでは、iPlanet の実行サーバーのポート 8888 からこのページにアクセスできますが、このポートはシステムによって異なる場合があります。
2. ドロップダウン・リストからサーバーを選択し、「**管理**」ボタンをクリックします。
3. 「**スタイル**」タブをクリックします。
4. 「**スタイルの割り当て**」リンクをクリックします。
5. ドロップ・リストから `alphablox` スタイルを選択します。
6. 標準的な DB2 Alphablox アプリケーションの場合は、以下の 3 つの **URL 接頭部ワイルドカード文字**を追加して、`alphablox` スタイルに割り当てます。

   ```
   MyApplication/abx/*
   MyApplication/servlet/*
   MyApplication/*.jsp
   ```

   MyApplication は、アプリケーションのコンテキスト名です。

7. 3.x 互換機能を使用するアプリケーションの場合は、上記の **URL 接頭部ワイルドカード**の代わりに以下のワイルドカードを追加して、alphablox スタイルに割り当てます。

   ```
   MyApplication/*
   ```

   MyApplication は、アプリケーションのコンテキスト名です。

8. この変更を保管して、すべての URL 転送要求にその変更を適用します。
9. 割り当てが正しく追加されているかどうかを確認するには、「**割り当てのリスト (List Assignments)**」リンクをクリックして、alphablox スタイルの割り当てリストを調べます。

既存の J2EE アプリケーションのインポート

既存の J2EE アプリケーションをインポートして、それを DB2 Alphablox アプリケーションとして定義できます。 DB2 Alphablox にインポートした J2EE アプリケーションは、Blox レンダリングやブックマークなどのすべての DB2 Alphablox アプリケーション・サービスにアクセスできます。

J2EE アプリケーションのインポートは、WebSphere と一緒に実行している場合にアプリケーションを使用可能にする一般的な方法の 1 つです。 WebSphere と一緒に実行している場合にアプリケーションを定義する方法の詳細については、34 ページの『WebSphere と一緒に実行する場合のアプリケーションの定義』を参照してください。

注: DB2 Alphablox がインストールされる前に作成された既存の Apache Tomcat アプリケーションをインポートするときは、*$CATALINA_HOME/conf/context.xml* ファイルを含む各所に `crossContext=true` を追加する必要もあります。詳しくは、http://tomcat.apache.org/tomcat-5.5-doc/config/context.html を参照してください。

DB2 Alphablox に既存の J2EE アプリケーションをインポートするには、以下の手順を実行します。

1. DB2 Alphablox にインポートする既存の J2EE アプリケーションのアプリケーション・コンテキスト名を確認します。
2. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
3. 「**管理**」タブをクリックします。
4. 「**アプリケーション**」リンクをクリックします。
5. 「**作成**」ボタンをクリックします。
6. 「**J2EE アプリケーションのインポート (Import J2EE Application)**」ドロップ・リストを 「**はい**」に設定します。「既存の J2EE アプリケーションのインポート (Importing Existing J2EE Application)」ウィンドウが表示されます。
7. 「**アプリケーションの検索**」ボタンを押します。「アプリケーションのインポート」ウィンドウが表示されます。
8. ドロップダウン・リストから、「**J2EE Application Name (J2EE アプリケーション名)**」 (Apache Tomcat では選択不可) および「**モジュール名**」を選択します。読み取り専用の **URI** および「**コンテキスト・パス**」フィールドには、選択内容に基づいて値が表示されます。
9. 「**アプリケーションの選択**」ボタンを押します。ダイアログ・ウィンドウが閉じ、「**既存の J2EE アプリケーションのインポート**」フォームが自動的に記入されます。
10. 「**保管**」を押して、選択された J2EE アプリケーションのインポートを完了します。「キャンセル」を押して「アプリケーション」メイン・ページに戻るか、または「**J2EE アプリケーションのインポート**」の下にある「**いいえ**」を選択して「**アプリケーションの作成**」ウィンドウに戻ります。

J2EE アプリケーションを正常にインポートすると、その新規アプリケーションがアプリケーションのリストに表示されます。そのアプリケーションの構造は、31 ページの『DB2 Alphablox のアプリケーション定義』で説明されているものになります。

第 7 章 データ・ソースの定義

この章では、マルチディメンション・データ・ソースとリレーショナル・データ・ソースの管理について取り上げ、DB2 Alphablox のデータ・ソース定義の作成、変更、削除のための手順を示します。

新規データ・ソースの定義

DB2 Alphablox は、マルチディメンション・データ・ソースおよびリレーショナル・データ・ソースをサポートします。「**管理**」タブの「**データ・ソース**」ページで、1 つ以上の DB2 Alphablox アプリケーションからアクセスするデータ・ソースを定義できます。 DB2 Alphablox アプリケーションは、これらのデータ・ソースを使用して、リレーショナル・データベースやマルチディメンション・データベースにアクセスします。「**管理**」タブの「**データ・ソース**」ページは、定義するデータ・ソースのタイプによって異なります。

新規データ・ソースを定義するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
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。「**データ・ソースの作成 (Create Data Source)**」ページが表示されます。
5. 「**データ・ソース名 (Data Source Name)**」テキスト・ボックスにデータ・ソースの固有の名前 (必須) と説明を入力します。英語以外の言語で実行している場合に使用できる文字は、A-Z、a-z、0-9、下線、特殊文字 (アクセント付き文字など) です。名前の表示では大/小文字の区別がありますが、認証の対象となる実際の名前には大/小文字の区別がありません。 *Public*、*Private*、*Properties* は予約語であり、オブジェクト名としては使用できません。
6. 「**アダプター**」ドロップ・リストから、データ・ソースとの接続に使用するデータベース・アダプターを選択します。

   注: DB2 Alphablox には、クライアントとサーバーの間の接続テストに使用する Canned データ・アダプターが付属しています。この点で問題があると感じる場合は、このアダプターの使用に関して DB2 Alphablox カスタマー・サポートに連絡し、支援を依頼してください。
7. 「**サーバー名 (Server Name)**」テキスト・ボックスに、データ・ソースが存在するホスト・マシン名を指定します。 (データ・アダプターによって名前が異なる場合があります。)
8. データ・ソースへのアクセスに関するその他のアダプター固有の情報をすべて指定します。例えば、「**デフォルト・カタログ**」、「**ポート番号 (Port Number)**」、「**データベース (カタログ) (Database (catalog))**」、 「**スキーマ**」、「**SID**」、「**ユーザー名**」、「**パスワード**」などがあります。

注: WebSphere と WebLogic アプリケーション・サーバーのインストール・システムでは、アプリケーションの認証にこのパスワードが使用されないので、「**パスワード**」は「**データ・ソース・パスワード**」として表示されます。その場合、ユーザー認証は、WebSphere または WebLogic サーバーによって処理されます。「**DB2 Alphablox ユーザー名とパスワードの使用 (Use DB2 Alphablox Username and Password)**」がご使用のアプリケーションで使用可能になっている場合、そのパスワードは、データ・ソースの認証のためだけに使用されるということです。

注: Microsoft SQL Server ユーザー: データベースの名前は、Microsoft SQL Server の ID の形式に関する規則に準拠しています。 ID には、正規 ID と区切り ID という 2 つのクラスがあります。正規 ID の形式の規則に従わない ID はすべて区切り ID であり、常に二重引用符か大括弧で区切らなければなりません。例えば、`test-1` という名前の MS SQL データベースの場合、DB2 Alphablox データ・ソース名の中のデータベース (カタログ) 名は、`[test-1]` (データ・ソース名に大括弧を組み込む場合) または `"test-1"` (データ・ソース名に引用符を組み込む場合) のどちらかです。

9. Microsoft SQL Server データ・ソースを使用している場合は、SQL Server 認証ユーザーとして定義されているデータベース・ユーザーを指定する必要があります。 Windows 認証を使用する SQL Server ユーザーは、DB2 Alphablox で正しく機能しません。
10. このデータ・ソースからの照会結果セットのサイズを制限する場合は、「**最大行数**」と「**最大列数**」を指定します。

    注: この数として、`20` 未満の値を設定しないでください。ほとんどのアプリケーションでは、デフォルトの `1000` で十分です。大きな照会が切り捨てられると、ブラウザーにエラー・ダイアログが表示されます。
11. Oracle データ・ソースを使用している場合は、表示の準備として取得する行数を「**行のプリフェッチ (Row Prefetch)**」フィールドに指定します。デフォルト値は、`100` です。
12. 「**保管**」ボタンをクリックして新規データ・ソースを定義し、「**データ・ソース**」ページに戻ります。

既存のデータ・ソース定義の変更と削除

このセクションでは、既存のデータ・ソースの変更と削除の方法について解説します。

既存のデータ・ソース定義の変更

既存のデータ・ソースを変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
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. 「**編集**」ボタンをクリックします。「**データ・ソースの編集**」ページが表示されます。

5. 該当する値を変更します。
6. 「**保管**」ボタンをクリックして定義を変更し、「**データ・ソース**」ページに戻ります。

既存のデータ・ソース定義の削除

既存のデータ・ソースを削除するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
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。「**データ・ソース**」ページが表示されます。
4. 「**削除**」ボタンをクリックしてデータ・ソース定義を削除し、「**データ・ソース**」ページに戻ります。

   重要: データ・ソースを削除すると、そのデータ・ソースは DB2 Alphablox から永久的に削除されます。

Microsoft Analysis Services データ・ソースに関する Microsoft 認証のセットアップ

Microsoft Analysis Services サーバー (Microsoft SQL Server 2000 Analysis Services およびMicrosoft SQL Server 2005 Analysis Services を含む) では、Windows ベースの認証を使用します。 Microsoft Analysis Services にアクセスする DB2 Alphablox アプリケーションで Windows ベースの認証を使用するには、DB2 Alphablox から Microsoft Analysis Services に渡す `userName` プロパティーと `password` プロパティーを Windows のユーザーとパスワードに合わせなければなりません。 DB2 Alphablox と Microsoft Analysis Services が別々のドメインに存在する場合は、その両方が Windows のトラステッド・ドメインでなければなりません。

DB2 Alphablox のユーザーが Microsoft Analysis Services に正しくログインできるようにするには、 DB2 Alphablox を実行する Windows ユーザー (つまり、DB2 Alphablox の開始時にログインした Windows ユーザー) に「オペレーティング システムの一部として機能」ユーザーの権利が必要です。 DB2 Alphablox を Windows のサービスとして実行している場合は、デフォルトの Windows ユーザーの下で実行するようにサービスを構成できます。このセクションでは、Windows マシンでこの形式の Microsoft Analysis Services 認証を使用するための構成手順を説明します。具体的には、以下の手順を取り上げます。

- 『Windows のユーザー権利のセットアップ』
- 49 ページの『Windows のサービスの構成』
- 50 ページの『Microsoft Analysis Services でのユーザーの構成』

Windows のユーザー権利のセットアップ

このセクションでは、Windows マシンから Microsoft Analysis Services データベース内のデータにアクセスするための構成手順を説明します。

注: クラスター環境で DB2 Alphablox を実行しながら、Microsoft Analysis Services データ・ソースにアクセスする場合は、クラスターの各ノードを同じ Windows ユーザーの下で実行するように構成する必要があります。また、各マシンのセキュリティー・ポリシーをこのセクションの説明に従って設定する必要があります。

Windows のユーザー権利の設定

DB2 Alphablox のユーザーが Microsoft Analysis Services に正しく接続できるようにするには、 DB2 Alphablox を実行する Windows ユーザー (つまり、DB2 Alphablox の開始時にログインした Windows ユーザー、または DB2 Alphablox をサービスとして実行している場合は、そのサービスを実行しているユーザー) に「オペレーティング システムの一部として機能」ユーザーの権利が必要です。

注: Windows ユーザーを Microsoft Analysis Services でセットアップする必要があります。

Windows システムで DB2 Alphablox を実行するユーザーのユーザー権利を有効にするには、以下の手順を実行します。

1. Windows の「**スタート**」メニューから、「**すべてのプログラム**」>「**管理ツール**」>「**ユーザー マネージャ**」を選択します。

   Windowsの「ユーザー マネージャ」ウィンドウが表示されます。
2. 「**原則**」メニューから、「**ユーザーの権利**」を選択します。
3. 「**高度なユーザー権利の表示**」というラベルの付いたチェック・ボックスを選択します。
4. 「**権利**」というラベルの付いたドロップ・リストから、「**オペレーティング システムの一部として機能**」を選択します。
5. 「**追加**」ボタンをクリックします。
6. 「**ユーザーとグループの追加**」ダイアログ・ボックスの「**ドメインまたはコンピュータ**」というラベルの付いたドロップ・リストに、追加するアカウントの正しいドメイン名またはマシン名が表示されていることを確認します。
7. 「**ユーザーの表示**」ボタンをクリックします。
8. 「**名前**」リスト・ボックスで、追加するアカウントを選択します。
9. 「**追加**」ボタンをクリックします。「**追加する名前**」リスト・ボックスにそのユーザー名が追加されます。
10. マシンをリブートします。

注: Microsoft Analysis Services にアクセスする DB2 Alphablox データ・ソースのユーザーは、「オペレーティング システムの一部として機能」ユーザーの権利を持つ Windows ユーザーと一致していなければなりません。

Windows のユーザー権利の設定

Windows システムで DB2 Alphablox を実行するユーザーのユーザー権利を有効にするには、以下の手順を実行します。

1. Windows 「**スタート**」メニューから「**設定**」>「**コントロール パネル**」を選択し、コントロール パネルを起動します。
2. 「**管理ツール**」を開きます。

3. 「**管理ツール**」コントロール・パネル・フォルダーから「**ローカル セキュリティ ポリシー**」文書を開きます。
4. 「**ローカル セキュリティ設定**」ウィンドウで「**ローカル ポリシー**」フォルダーを展開します。
5. 「**ユーザー権利の割り当て**」フォルダーを選択します。
6. 「**ローカル セキュリティ設定**」ウィンドウの右側のペインで、「**オペレーティング システムの一部として機能**」ユーザー権利をダブルクリックします。
7. 「**オペレーティング システムの一部として機能**」ユーザー権利の「**ローカル セキュリティ ポリシー設定**」ウィンドウで「**追加**」ボタンをクリックして、DB2 Alphablox を実行するユーザーを選択します。 DB2 Alphablox をネットワーク・ユーザーの下で実行している場合は、正しいネットワーク・ユーザーを選択してください (例えば、CORPNET というWindows ネットワーク上の *alphablox* という名前のネットワーク・ユーザーであれば、CORPNET¥alphablox になります)。
8. 「**ローカル ポリシー設定**」ボックスにチェック・マークを付けて、「**OK**」ボタンをクリックします。
9. 「**ローカル セキュリティ ポリシー**」ウィンドウを閉じます。
10. マシンをリブートします。

注: Microsoft Analysis Services にアクセスする DB2 Alphablox データ・ソースのユーザーは、「オペレーティング システムの一部として機能」ユーザーの権利を持つ Windows ユーザーと同じでなければなりません。

Windows のサービスの構成

ログインしたユーザーとは異なるユーザーの下で DB2 Alphablox を実行するための構成を行うには、DB2 Alphablox をサービスとして実行する必要があります。 DB2 Alphablox から Microsoft Analysis Services に接続する場合は、Windows ユーザーを Microsoft Analysis Services でセットアップし、そのユーザーに「オペレーティング システムの一部として機能」ユーザーの権利を付与する必要があります。

別の Windows ユーザーの下でサービスを開始するための構成を行うには、以下の手順を実行します。

1. Windows の「スタート」メニューから、「**設定**」を選択します。
2. プルダウン・メニューから「**コントロール パネル**」を選択します。
3. 「**サービス**」アイコンをダブルクリックします。
4. 「**サービス**」ウィンドウで、「**DB2 Alphablox (AlphabloxAnalytics)**」サービスをダブルクリックします。
5. 「**サービス**」ダイアログ・ボックスで、「**アカウント**」ラジオ・ボタンを選択します。
6. 右のテキスト・ボックスにユーザー名を入力します。
7. 「**パスワード**」テキスト・ボックスにパスワードを入力します。
8. 確認のために「**パスワードの確認入力**」テキスト・ボックスにもう一度パスワードを入力します。
9. 「**OK**」ボタンをクリックします。

Microsoft Analysis Services でのユーザーの構成

DB2 Alphablox データ・ソースで構成した場合でも、アプリケーションでユーザー名とパスワードのパラメーター (または関連メソッド) を使用して指定した場合でも、Microsoft Analysis Services に接続する DB2 Alphablox で指定したユーザーはすべて、 Windows ユーザーでなければならず、 Microsoft Analysis Services で正しく構成しておかなければなりません。 Windows のドメイン・グループ Everyone に含まれているのはドメイン・ユーザーだけであり、ローカル・マシン・ユーザーは含まれていません。

注: マシンのローカル・ユーザーとして Microsoft Analysis Services にログインしようとする場合、確実な接続を可能にするために、そのユーザーは Administrators グループのメンバーであると同時に OLAP Administrator でなければなりません。ユーザーが両方の管理者グループのメンバーでない場合、初回のログインは成功しますが、次回以降のログインは失敗し、以下のような例外が発生します。

```
com.alphablox.util.NotAuthorizedException: Provider cannot be
found. It may not be properly installed
```

JDBC データ・ソースの処理

ここには、JDBC データ・ソースに関連した以下のセクションが含まれています。

- 『Sybase JConnect リレーショナル・ドライバー用の環境のセットアップ』
- 51 ページの『JDBC トレース機能のセットアップ』
- 51 ページの『サポートされている JDBC ドライバーを別のバージョンに更新する』
- 52 ページの『JDBC ドライバーの追加』
- 53 ページの『クラスパス設定の変更』

Sybase JConnect リレーショナル・ドライバー用の環境のセットアップ

Sybase JConnect ドライバーを使用して Sybase リレーショナル・データベースに接続する場合は、Sybase データベース・サーバーで SQL スクリプト・ファイルを実行する必要があります。 SQL スクリプト・ファイルは、以下のディレクトリーにあります。

```
<db2alphablox_dir>/tools/sql
```

<db2alphablox_dir> は、DB2 Alphablox のインストール先のディレクトリーです。SQL スクリプトの名前と各スクリプトを実行する Sybase データベース・サーバーを以下の表にまとめます。

SQL スクリプト	スクリプトを実行する Sybase データベース・サーバー
sql_asa.sql	SQL Anywhere 5.5.02 以降
sql_server.sql	SQL Server バージョン 12 より前のバージョン
sql_server12.sql	SQL Server バージョン 12 以降

実行している Sybase データベース・サーバーを判別し、それに該当するスクリプトをそのデータベース・サーバーで実行します。どのスクリプトも、Sybase JConnect JDBC ドライバーが必要とするストアード・プロシージャーとシステム・オブジェクトをセットアップします。 Sybase データベース・サーバー上で SQL スクリプト・ファイルを実行する方法については、Sybase の資料を参照してください。

DB2 Alphablox から Sybase データベース・サーバーに接続するには、サーバー上で適切な SQL スクリプトを実行しなければなりません。

JDBC トレース機能のセットアップ

DB2 Alphablox 内にあるリレーショナル・データ・ソース・ドライバーについては、JDBC トレース機能を使用可能にできます。 JDBC トレース機能を使用可能にすると、JDBC ドライバーは、DB2 Alphablox コンソールとログ・ファイルに通知メッセージを書き出します。 JDBC トレース機能メッセージはすべて、INFO メッセージ・レベルで記録されます。

リレーショナル・データ・ソースで JDBC トレース機能を使用可能にするには、「**JDBC トレース機能を使用可能にする (JDBC Tracing Enabled)**」ドロップ・リストを「**はい**」に設定します。このオプションは、サポートされていリレーショナル・データベース・ドライバーのいずれかを使用するデータ・ソースでのみ使用できます。 JDBC トレース機能を使用可能にした後に、メッセージ・レベルを INFO 以上に設定してください。例えば、「コンソール」ウィンドウで、以下のコマンドを入力します。

```
report info
```

JDBC トレース機能を使用して、リレーショナル・データベースとの接続に関する問題をデバッグします。 JDBC トレース機能が使用可能になっていると、以下の情報が記録されます。

- JDBC ドライバーのログ情報
- DB2 Alphablox からリレーショナル・データベースへの接続に関する情報
- データベースが DB2 Alphablox に戻すそれぞれの結果セットからの列メタデータについての情報

接続時の問題や JDBC トレース機能の詳細については、DB2 Alphablox カスタマー・サポートに連絡してください。

サポートされている JDBC ドライバーを別のバージョンに更新する

DB2 Alphablox がリレーショナル・データベースへのアクセスに使用するサポート対象の JDBC ドライバーのいずれかを更新する場合は、以下の手順を実行します。

重要: DB2 Alphablox でテストが行われているのは、「*インストール・ガイド*」の『システム要件』にリストされているサポート対象の JDBC ドライバーだけです。それ以外のドライバーはテストされていないので、予期しない結果が生じる可能性もあります。詳細については、DB2 Alphablox カスタマー・サポートに連絡してください。

1. 新規 JDBC ドライバーの Java クラスを入手します。通常、これらのファイルの拡張子は、zip か jar のどちらかです (例えば、ojdbc14.jar のようになります)。
2. JDBC ドライバーのインストール時に指定したディレクトリーを探します。
3. 更新する JDBC ドライバー・ファイルの名前を変更し (例えば、<driverName>.zip.old などに変更し)、サポートされているバージョンに戻さなければならない場合のために保存しておきます。
4. 新規ドライバー・ファイルの名前を、サポートされている既存のドライバーのいずれかと同じ名前に変更し、それを <db2alphablox_dir>/lib/ ディレクトリーにコピーします。
5. 以下のセクションの説明に従って、JDBC ドライバーの新規クラスにアクセスできるように、DB2 Alphablox の始動環境を更新します。
6. クラスパス情報を更新した後、変更を有効にするためにサーバーを再始動します。

JDBC ドライバーの追加

インストール時またはアップグレード時にデフォルトで DB2 Alphablox アプリケーションに用意されるリレーショナル・データ・ソースは、jdbcdrivers.xml ファイルに定義されています。このファイルには変更を加えないでください。このファイルは、DB2 Alphablox をアップグレードするときに自動的に上書きされます。追加の DB2 Alphablox データ・ソースの作成が必要な場合は、オプションの jdbcdrivers_additional.xml ファイルを使用できます。

DB2 Alphablox 用に追加のデータ・ソースを作成するには、以下の手順を実行します。

1. 以下のディレクトリーに存在する jdbcdrivers_additional.xml ファイルを検索します。

   <db2alphablox_dir>/repository/servers/
2. このファイルは編集可能ですが、同じディレクトリー内に jdbcdrivers_additional.xml という名前でこのファイルのバックアップ・コピーを作成しておきます。デフォルトの jdbcdrivers_additional.xml ファイルのバックアップ・コピーは、jdbcdrivers_additional.xml.example という名前で同じディレクトリーに存在しています。
3. このファイルに含まれるサンプル JDBC ドライバーを使用して、必要な追加データ・ソースを作成します。その場合の注意点は、以下のとおりです。
   a. 使用可能にするすべての新規 JDBC ドライバー定義で、使用不可になっている属性を true から false に変更します。デフォルトでは、サンプル JDBC ドライバーは、使用不可になっています。
4. これらの変更を有効にするために、アプリケーションを再始動します。
5. 以下のセクション (53 ページの『クラスパス設定の変更』) の説明に従って編集を行い、すべての DB2 Alphablox クラスパスに新規ドライバーを組み込みます。

クラスパス設定の変更

jdbcdrivers_additional.xml ファイルにドライバーを追加したら、新規 JDBC ドライバーが検出されるようにクラスパスの情報も更新しなければなりません。それぞれの構成に応じて、以下の手順を実行してください。

WebSphere

WebSphere で JDBC ドライバーの追加や削除を行う場合、クラスパス設定の変更は必要ありません。追加する場合は、追加の JDBC ドライバーを <websphere_dir>/AppServer/lib/ext ディレクトリーに配置し、削除する場合は、使用しなくなった JDBC ドライバーや古い JDBC ドライバーをそのディレクトリーから削除します。クラスパスの変更を有効にするために、アプリケーション・サーバーを再始動します。

WebLogic

BEA WebLogic アプリケーション・サーバーのインストール・システムの場合は、バッチ・コマンド・ファイル <db2alphablox_dir>/bin/jdbcsetup.bat (Windows) または <db2alphablox_dir>/bin/jdbcsetup.sh (UNIX) でクラスパス情報を変更できます。変更を有効にするために、アプリケーション・サーバーを再始動します。

Tomcat

Apache Tomcat の下で実行している DB2 Alphablox のクラスパス情報を変更するには、以下の手順を実行します。

Windows: Windows コンソールで Tomcat を実行している場合は、テキスト・エディターで <db2alphablox_dir>/bin/jdbcsetup.bat ファイルを開き、必要に応じてクラスパス設定を変更します。

アプリケーション・サーバーを Windows のサービスとして実行している場合は、テキスト・エディターで (DB2 Alphablox ディレクトリーではなく Tomcat ディレクトリーの) <tomcat_dir>/conf/wrapper.properties ファイルを開き、必要に応じてクラスパス設定を変更します。変更を有効にするために、アプリケーション・サーバーを再始動します。

Linux と UNIX: Linux または UNIX のシステムで Apache Tomcat と一緒に DB2 Alphablox を使用している場合は、テキスト・エディターで <tomcat_dir>/bin/analytics.sh ファイルを開き、必要に応じてクラスパス設定を変更します。変更を有効にするために、アプリケーション・サーバーを再始動します。

第 8 章 ユーザーの定義

DB2 Alphablox 「**ユーザー**」管理ページを使用して、ユーザーの作成、変更、削除を行えます。

「**アプリケーション**」ページと「**データ・ソース**」ページに関する情報は、各ページの右上隅にある「**ヘルプ**」リンクからも入手できます。

注: 既存のユーザーを変更する場合は、別のページに移動する前に必ず「**保管**」ボタンをクリックしてください。そうしないと、ページで加えた変更は保管されません。

新規ユーザーの作成

「**管理**」タブの下の「**ユーザー**」リンクを使用して、新規ユーザーの作成と編集のためのページを開きます。インストール時には、2 つのユーザー (管理とゲスト) が作成されます。「**ユーザー**」ページの「**作成**」ボタンをクリックして、追加のユーザーを定義できます。既存のユーザー名を選択し、「**編集**」ボタンまたは「**削除**」ボタンをクリックすれば、グループ名の変更または削除を行えます。ユーザーが損傷を受けた場合は、DB2 Alphablox を再始動することによって修復できます。このセクションでは、DB2 Alphablox ホーム・ページから新規ユーザーを定義するための手順を説明します。

新規ユーザーを定義するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**ユーザー**」リンクをクリックします。「**ユーザー**」ページが表示されます。
4. 「**作成**」ボタンをクリックします。「**ユーザーの作成 (Create User)**」ページが表示されます。

   注: セキュリティー構成によっては、この画面にパスワード情報が含まれない場合もあります。
5. 「**ユーザー名**」テキスト・ボックスにユーザーの ID を入力します (必須)。ユーザー名は、DB2 Alphablox のログインに使用する名前です。英語以外の言語で実行している場合に使用できる文字は、A-Z、a-z、0-9、下線、特殊文字 (アクセント付き文字など) です。ユーザー名の表示では大/小文字の区別がありますが、認証の対象となる実際のユーザー名には大/小文字の区別がありません。*Public*、*Private*、*Properties* は予約語であり、オブジェクト名としては使用できません。
6. (オプション) ユーザーの「**フルネーム**」、「**説明**」、「**E メール・アドレス**」を入力します。
7. 「**パスワード**」テキスト・ボックスにユーザーの初期パスワードを入力します。パスワードは大文字小文字が区別されます。

8. 確認のために「**パスワードの確認**」テキスト・ボックスにパスワードを再入力します。
9. 「**ユーザーによるプロファイルの編集を許可する**」ドロップ・リストで、ユーザーにプロファイルを編集する特権を与えるかどうかを指定します。このプロパティーを「**はい**」に設定すると、ユーザーは、DB2 Alphablox ホーム・ページの「**マイ・プロファイル**」リンクからプロファイルにアクセスして、内容を変更できるようになります。
10. カスタム・プロパティーの値を入力します。(カスタム・ユーザー・プロパティーが 1 つでも定義されている場合は、そのプロパティーが「**ユーザーによるプロファイルの編集を許可する**」ドロップ・リストの下に表示されます。)
11. 「**保管**」ボタンをクリックして新規ユーザーを定義し、「ユーザー」ページに戻ります。

注: 新規ユーザーを作成すると、そのユーザーには自動的に *Public* グループのメンバーシップが割り当てられます。

既存のグループとユーザーの変更と削除

このセクションでは、DB2 Alphablox ホーム・ページからプロパティーを変更したり、既存のユーザーやグループを削除したりするための手順を説明します。

既存のユーザーのプロパティーの変更

既存のユーザーのプロパティーを変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**ユーザー**」リンクをクリックします。「**ユーザー**」ページが表示されます。
4. 「**編集**」ボタンをクリックします。「**ユーザーの編集**」ページが表示されます。
5. 変更する必要のある値を変更します。

   注: WebSphere と WebLogic のインストール・システムでは、アプリケーションの認証にこのパスワードが使用されないので、「パスワード」は「データ・ソース・パスワード」として表示されます。その場合、ユーザー認証は、WebSphere または WebLogic によって処理されます。つまり、アプリケーションで DB2 Alphablox のユーザー名とパスワードを使用する場合、そのパスワードは、データ・ソースの認証のためだけに使用されるということです。
6. 「**保管**」ボタンをクリックして「**一般プロパティー**」パネルの変更を保管し、「**ユーザー**」ページに戻ります。
7. ユーザーのアプリケーション・プロパティーを変更するには、「**アプリケーション・プロパティー**」タブをクリックします。必要な変更を加え、「**保管**」ボタンをクリックします。
8. ユーザーのグループのメンバーシップや役割のメンバーシップを変更するには、「**メンバーシップ**」タブをクリックします。必要な変更を加えます。メンバーシップに加えた変更は、すぐに有効になります。グループのメンバーシップと役割

のメンバーシップの詳細については、57 ページの『ユーザーが所属するグループの変更』と 64 ページの『ユーザーやグループが所属する役割の変更』を参照してください。

既存のユーザーの削除

既存のユーザーを削除するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**ユーザー**」リンクをクリックします。「**ユーザー**」ページが表示されます。
4. ユーザーのリストからユーザーを選択します。
5. 「**削除**」ボタンをクリックしてユーザーを削除します。

重要: ユーザーを削除すると、そのユーザーは DB2 Alphablox から永久的に削除されます。

ユーザーが所属するグループの変更

このセクションでは、ユーザーとグループのグループ・メンバーシップの追加や変更を行うための手順を説明します。

ユーザーが所属するグループを変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**ユーザー**」リンクをクリックします。「**ユーザー**」ページが表示されます。
4. 「**編集**」ボタンをクリックします。「**ユーザーの編集**」ページが表示されます。
5. 「**メンバーシップ**」タブをクリックします。
6. ユーザーが所属する新規グループを追加するには、「**使用可能なグループ**」リストからグループを選択し、左矢印ボタンをクリックして「**グループのメンバーシップ**」リストに移動します。
7. ユーザーが所属するグループを除去するには、「**グループのメンバーシップ**」リストからグループを選択し、右矢印ボタンをクリックして「**使用可能なグループ**」リストに移動します。
8. ユーザーが所属する新規役割を追加するには、「**使用可能な役割**」リストからグループを選択し、左矢印ボタンをクリックして「**役割のメンバーシップ**」リストに移動します。

注: ユーザーのグループや役割のメンバーシップに加えた変更は、すぐに有効になります。

第 9 章 グループの定義

「**グループ**」管理ページを使用して、グループの作成、変更、削除を行えます。

「**グループ**」ページに関する情報は、各ページの右上隅にある「**ヘルプ**」リンクからも入手できます。

注: 既存のグループを変更する場合は、別のページに移動する前に必ず「**保管**」ボタンをクリックしてください。そうしないと、ページで加えた変更は保管されません。

新規グループの作成

「**管理**」タブの下の「**グループ**」リンクを使用して、新規グループの作成と編集のためのページを開きます。インストール時に、2 つのグループ (管理者とパブリック) が作成されています。「**グループ**」ページの「**作成**」ボタンをクリックして、追加のグループを定義できます。既存のグループ名を選択し、「**編集**」ボタンまたは「**削除**」ボタンをクリックすれば、グループ名の変更または削除を行えます。グループが損傷を受けた場合は、DB2 Alphablox を再始動することによって修復できます。このセクションでは、DB2 Alphablox ホーム・ページから新規グループを定義するための手順を説明します。

新規グループを定義するには、以下のようにします。

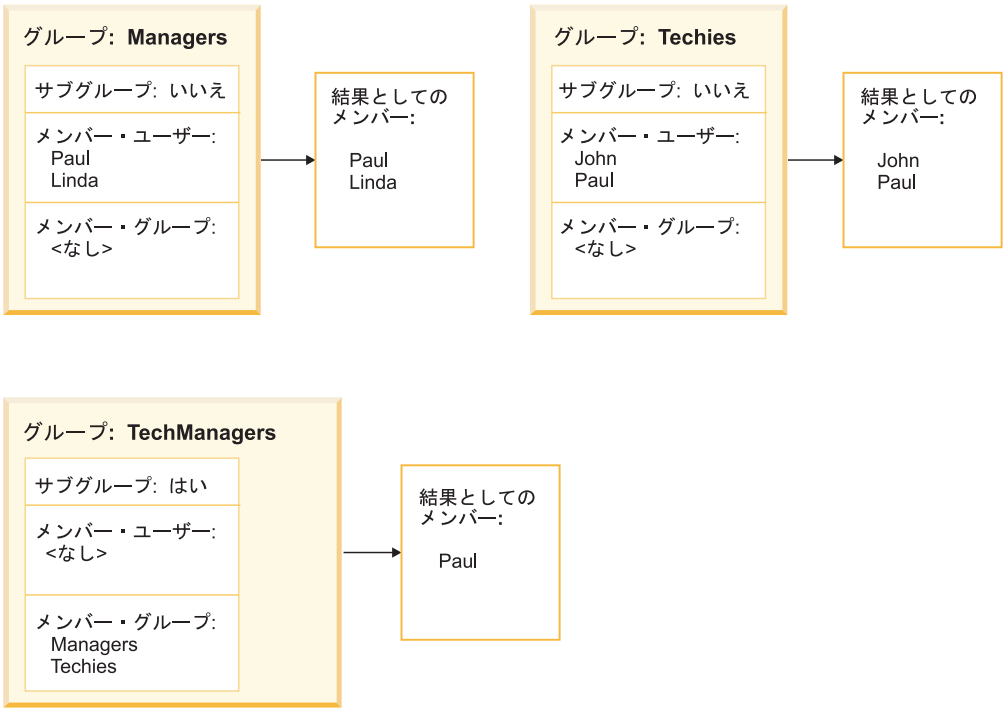
1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**グループ**」リンクをクリックします。「**グループ**」ページが表示されます。
4. 「**作成**」ボタンをクリックします。「**グループの作成**」ページが表示されます。
5. 「**グループ名**」テキスト・ボックスに、グループの名前を入力します (必須)。英語以外の言語で実行している場合に使用できる文字は、A-Z、a-z、0-9、下線、特殊文字 (アクセント付き文字など) です。グループ名の表示では大/小文字の区別がありますが、認証の対象となる実際のグループ名には大/小文字の区別がありません。 *Public*、*Private*、*Properties* は予約語であり、オブジェクト名としては使用できません。
6. 「**説明**」にグループの説明を入力します (オプション)。
7. 「**サブグループ**」ドロップ・リストで、このグループをサブグループにする場合は「**はい**」を、そうでない場合は「**いいえ**」を選択します。サブグループの詳細については、60 ページの『サブグループについて』を参照してください。
8. ユーザーやグループを「**メンバー・ユーザー**」**選択ボックス**や「**メンバー・グループ**」選択ボックスでこのグループに追加します。そのためには、ユーザーやグループのリストをスクロールします。このグループに複数のユーザーやグループを追加するには、Ctrl キーを押したままで選択してください。

9. カスタム・プロパティーの値を入力します。(カスタム・グループ・プロパティーが定義されている場合は、「**メンバー・グループ**」選択ボックスの下に表示されます。)
10. 「**保管**」ボタンをクリックして新規グループを定義し、「**グループ**」ページに戻ります。

サブグループについて

サブグループ とは、複数のグループに所属するユーザーを自動的に別のグループのメンバーにするためのメカニズムです。ある新規グループが「**Yes (はい)**」に設定された「**Subgroup (サブグループ)**」ドロップ・リストを持っている場合、その結果として生ずるユーザー・メンバーは、その新規グループのメンバーとして割り当てられている各グループのユーザーの論理積になります。

サブグループを使用する例を以下の図に示します。この例のグループ *TechManagers* は、サブグループを使用して、*Managers* グループと *Techies* グループの両方に所属するメンバーを自動的に組み込みます。



管理者は、*TechManagers* グループのサブグループを使用することによって、*Managers* グループと *Techies* グループのすべての共通メンバーのセットを自動的に組み込むためのグループを作成します。この例の場合、両方のグループに存在するユーザーは *Paul* だけです。他のユーザーを *Managers* グループと *Techies* グループの両方に追加すれば、そのユーザーは、自動的に *TechManagers* グループのメンバーになります。

注: サブグループを使用している場合、「**メンバー・ユーザー**」選択ボックスで個々のユーザーを選択することによって、手動でそのグループにユーザーを追加することもできます。

既存のグループの変更と削除

このセクションでは、DB2 Alphablox ホーム・ページからプロパティーを変更したり、既存のユーザーやグループを削除したりするための手順を説明します。

既存のグループの変更

既存のグループを変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**グループ**」リンクをクリックします。「**グループ**」ページが表示されます。
4. グループのリストからグループを選択します。
5. 「**編集**」ボタンをクリックします。「**グループの編集**」ページが表示されます。
6. 変更する必要のある値を変更します。
7. 「**保管**」ボタンをクリックして変更を保管し、「**ユーザー**」ページに戻ります。
8. ユーザーのアプリケーション・プロパティーを変更するには、「**アプリケーション・プロパティー**」タブをクリックします。
9. 必要な変更を加えます。
10. 「**保管**」ボタンをクリックします。
11. ユーザーのグループや役割のメンバーシップを変更するには、「**メンバーシップ**」タブをクリックします。
12. 必要な変更を加えます。メンバーシップに加えた変更は、すぐに有効になります。グループのメンバーシップと役割のメンバーシップの詳細については、57 ページの『ユーザーが所属するグループの変更』と 64 ページの『ユーザーやグループが所属する役割の変更』を参照してください。
13. 「**保管**」ボタンをクリックします。

既存のグループの削除

既存のグループを削除するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**グループ**」リンクをクリックします。「**グループ**」ページが表示されます。
4. グループのリストからグループを選択します。
5. 「**削除**」ボタンをクリックしてグループを削除します。

   重要: グループを削除すると、そのグループは DB2 Alphablox から永久的に削除されます。

第 10 章 役割の定義

「**役割**」管理ページを使用して、役割の作成、変更、削除を行えます。「役割」ページは、Apache Tomcat 構成で DB2 Alphablox を実行している場合にのみ表示されます。WebSphere または WebLogic を使用している場合は、アプリケーション・サーバーがこの機能を提供します。ユーザーの役割は通常、Web アプリケーション・サーバーのアプリケーション展開ツールで指定します。そこで役割を定義して展開記述子ファイルに追加したら、DB2 Alphablox の内部で役割を正しく使用できるようになります。

「**役割**」ページに関する情報は、各ページの右上隅にある「**ヘルプ**」リンクからも入手できます。

注: 既存の役割を変更する場合は、別のページに移動する前に必ず「**保管**」ボタンをクリックしてください。そうしないと、ページで加えた変更は保管されません。

新規役割の定義

DB2 Alphablox では、アプリケーションへのアクセスを役割によって制御できます。役割には、ユーザーやグループのリストと、そのリスト内の各項目に関連したアクセス許可が含まれています。

新規役割を作成するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**役割**」リンクをクリックします。「**役割**」ページが表示されます。
4. 「**作成**」ボタンをクリックします。「**役割の作成**」ページが表示されます。
5. 「**役割名**」テキスト・ボックスに、役割の名前を入力します (必須)。英語以外の言語で実行している場合に使用できる文字は、A-Z、a-z、0-9、下線、特殊文字 (アクセント付き文字など) です。名前の表示では大/小文字の区別がありますが、認証の対象となる実際の名前には大/小文字の区別がありません。 *Public*、*Private*、*Properties* は予約語であり、オブジェクト名としては使用できません。
6. 「**説明**」に役割の説明を入力します (オプション)。
7. 「**保管**」ボタンをクリックして新規役割を定義し、「**役割**」ページに戻ります。

既存の役割の変更と削除

既存の役割を変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**役割**」リンクをクリックします。「**役割**」ページが表示されます。

4. 役割のリストから役割を選択します。
5. 「**編集**」ボタンをクリックします。「**役割の編集**」ページが表示されます。
6. 必要に応じて、役割の説明を変更します。
7. 次に進む前に「**保管**」ボタンをクリックします。そうしないと、変更は保管されません。
8. 役割に割り当てるグループを変更するには、「**メンバー・グループ**」タブをクリックします。現時点で役割のメンバーになっているグループのリストと、現時点で役割のメンバーにはなっていない使用可能なグループのリストを示したページが表示されます。左矢印ボタンと右矢印ボタンを使用して、選択した項目を「**使用可能なグループ**」リストから「**メンバー・グループ**」リストに移動したり、「**メンバー・グループ**」リストから「**使用可能なグループ**」リストに移動したりできます。
9. 「**保管**」ボタンをクリックして、役割のグループ・メンバーシップ・プロパティーを保管します。
10. 役割のメンバーになっているユーザーを変更するには、「**メンバー・ユーザー**」タブをクリックします。現時点で役割のメンバーになっているユーザーのリストと、現時点で役割のメンバーにはなっていない使用可能なユーザーのリストを示したページが表示されます。左矢印ボタンと右矢印ボタンを使用して、選択した項目を「**使用可能なユーザー**」リストから「**メンバー・ユーザー**」リストに移動したり、「**メンバー・ユーザー**」リストから「**使用可能なユーザー**」リストに移動したりできます。
11. 「**保管**」ボタンをクリックして、役割のユーザー・メンバーシップ・プロパティーを保管します。

ユーザーやグループが所属する役割の変更

ユーザーやグループに割り当てる役割を変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**ユーザー**」リンクまたは「**グループ**」リンクをクリックします。「**役割**」ページが表示されます。
4. 「**ユーザー**」ページまたは「**グループ**」ページで、ユーザー (またはグループ) のリストからユーザー (またはグループ) を選択します。
5. 「**編集**」ボタンをクリックします。「**ユーザーの編集**」 (または「**グループの編集**」) ページが開きます。
6. 「**メンバーシップ**」タブをクリックします。
7. 「**メンバーシップ**」パネルの下半分には、「**役割のメンバーシップ**」ボックスと「**使用可能な役割**」ボックスがあります。左矢印と右矢印ボタンを使用して、選択した項目を「**使用可能な役割**」リストから「**役割のメンバーシップ**」リストに移動したり、「**役割のメンバーシップ**」リストから「**使用可能な役割**」リストに移動したりできます。

既存の役割の削除

既存の役割を削除するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. 「**役割**」リンクをクリックします。「**役割**」ページが表示されます。
4. 役割のリストから役割を選択します。
5. 「**削除**」ボタンをクリックして役割を削除します。

   重要: 役割を削除すると、その役割は DB2 Alphablox から永久的に削除されます。

第 11 章 セキュリティーと認証

この一連のトピックでは、Apache Tomcat 構成で実行する DB2 Alphablox を使用したセキュリティーと認証について説明します。この構成では、外部の Web サーバーを使用しない場合もあれば、別の Web サーバー (Sun iPlanet、Microsoft IIS、または Apache HTTP Server) を使用する場合もあります。

WebSphere や WebLogic などの市販のアプリケーション・サーバーを使用している場合のセキュリティーと認証の詳細については、それぞれの製品の資料を参照してください。

注: DB2 Alphablox 8.4 と 8.4.1 とでは、サポートする外部 Web サーバーが異なります。 DB2 Alphablox 8.4.1 は Sun iPlanet をサポートしません。 DB2 Alphablox 8.4.1 は Apache HTTP Server 2.0 をサポートし、DB2 Alphablox 8.4 は Apache HTTP Server 1.3 をサポートします。詳しくは、*「インストール・ガイド」*のシステム要件を参照してください。

DB2 Alphablox の認証とセキュリティーのモード

DB2 Alphablox からユーザーに対してページを配信するには、まずそのユーザーが認証を受けている必要があります。この点は、基礎となる Web サーバーが何であっても言えることです。ただし、外部の Web サーバーを使用する場合、DB2 Alphablox からの認証は基本的に透過的に行われます。

デフォルトで、HTML ページは、Web サーバーの背後にある DB2 Alphablox 構成内のアプリケーション・サーバーに経路指定されておらず、実際にアプリケーション・サーバーが JSP ページまたは DB2 Alphablox コンテンツを含むページを受け取るまで認証は行われません。 Web サーバーで HTML ページの認証を行うための構成を行うことはできますが、DB2 Alphablox 構成は、デフォルトでこの認証をサポートしていません。 Web サーバーの背後にある DB2 Alphablox の構成で認証を行う場合は、すべての HTML ページの認証をサポートするように Web サーバーを構成するか、ウェルカム・ページを JSP ファイルにする必要があります。

DB2 Alphablox は、以下の 2 種類のユーザー認証をサポートしています。

認証の種類	説明
アプリケーション・サーバー認証	DB2 Alphablox が外部 Web サーバーなしでアプリケーション・サーバーで作動している場合、認証と許可はアプリケーション・サーバーによって管理されます。
Web サーバー認証	Sun iPlanet、Apache HTTP Server、または Microsoft IIS を Web サーバーとして使用する場合は、Web サーバーが認証を行います。 DB2 Alphablox がユーザーを認証しようとする場合、Web サーバーは、DB2 Alphablox にユーザー ID を渡します。 **注:** Web サーバーがユーザー ID とパスワードを DB2 Alphablox に渡すことはありません。そのようなことは、セキュリティー・ブリーチ (抜け穴) の原因になるからです。

Alphablox 8.4.1 でのセキュリティー・モデル

DB2 Alphablox 8.4.1 は、標準 Servlet 2.4 セキュリティー API に基づく新しいセキュリティー・モデルを提供します。 Alphablox 8.4.1 と 8.4 では、サポートする Apache Tomcat のバージョンが異なるので、セキュリティーのサポートも異なっています。2 つの Tomcat バージョンはそれぞれ異なるセキュリティー・モデルおよびサーブレット仕様をサポートしているからです。 DB2 Alphablox 8.4.1 では、セキュリティー情報は Apache Tomcat の *conf/server.xml* ファイルの中で直接構成されるようになりました。 Tomcat 5.5 でセキュリティーを構成する方法については、構成レルムに関する Apache Tomcat の資料 (http://tomcat.apache.org/tomcat-5.5-doc/realm-howto.html) をご覧ください。

このセキュリティー・モデルをサポートする Alphablox API は、3 つのパッケージで提供されます。

- com.alphablox.security.jaas
- com.alphablox.security.jndi
- com.alphablox.security.ntlm

Java 認証・承認サービス (JAAS) ベースの Alphablox ログイン・モジュール

com.alphablox.security.jaas パッケージは、JAAS ベースの Alphablox ログイン・モジュールを提供しています。 com.alphablox.security.jaas.AlphabloxLoginModule クラスは、DB2 Alphablox リポジトリーの認証情報にアクセスすることにより、ユーザーの認証を行います。 Tomcat 5.5 インストール・ディレクトリー内の *conf/alphablox-jaas.config* ログイン構成ファイルで `debug` オプションが `true` (デフォルト) に設定されている場合、デバッグ・メッセージが Alphablox サーバー・ログに出力されます。

Tomcat 5.5 での JNDI レルム構成

Apache Tomcat JNDI レルムは LDAP 動的グループをサポートしませんが、DB2 Alphablox は Alphablox JNDI レルムを提供しており、それを使用して、LDAP 動的グループからのユーザーおよびグループ情報にアクセスすることができます。 com.alphablox.security.jndi.AlphabloxJNDIRealm クラスは、LDAP 動的グループ内で役割メンバーシップを追加で検索することにより、Tomcat 5.5 における JNDIRealm の基本サポートを拡張します。このクラスは、JNDIRealm で提供されるものに加わる追加の 4 つの属性を定義します。

属性	説明
`userName`	オプション。属性名をユーザーの固有の名前として読み取るように設定します。この値は、JNDI レルムの `userPattern` または `userSearch` ストリングで使用される属性に一致する必要があります。デフォルトは `uid` です。
`memberURL`	オプション。動的グループ URL 属性名を設定します。デフォルトは `memberURL` です。

属性	説明
cacheEnabled	オプション。関連したユーザー名への動的グループのキャッシングを使用可能または使用不可にします。デフォルトは true です。
dynamicGroupSearch	必須。 LDAP 動的グループ用の roleBase の検索に使用される JNDI フィルターを設定します。デフォルトはヌルです。キャッシュを使用可能にする (cacheEnabled="true") と、動的グループを繰り返し照会する必要がなくなるため、パフォーマンスが良くなります。

詳しい API 情報については、DB2 Alphablox インフォメーション・センターにある Javadoc 文書をご覧ください。

Alphablox JNDI レルムを Apache Tomcat 5.5 に追加する

Alphablox JNDI レルムを Tomcat に追加するには、className を com.alphablox.security.jndi.AlphabloxJNDIRealm に設定して Apache Tomcat の *conf/server.xml* ファイルを構成します。

```
<Realm
   className="com.alphablox.security.jndi.AlphabloxJNDIRealm"
   connectionURL="ldap://your_ldap.your_server.com:port"
   userBase="..."
   userSearch="..."
   userRoleName="..."
   roleBase="..."
   roleName="..."
   roleSearch="..."
   dynamicGroupSearch="(&amp; (cn={0}) (objectclass=groupofurls))"
 />
```

注:

- "&" 文字は、XML では "&" としてエンコードする必要があります。
- *server.xml* ファイルに変更を加えた後、Tomcat を再始動する必要があります。

管理権限とユーザー権限

DB2 Alphablox をインストールすると、管理権限を持つ 1 つのユーザー・プロファイルが作成されます。このプロファイルの名前は *admin* であり、変更はできません。 *admin* のデフォルトのパスワードは、*password* です。 Web サーバー・セキュリティーを使用する場合は、*admin* という名前のユーザーが Web サーバーの認証方式の下に存在している必要があります。 (IIS や Windows® NTLM を使用する場合、*admin* は、ドメイン・ユーザーではなくローカル・ユーザーになります。)

admin ユーザーは、DB2 Alphablox の「**管理**」 ページからユーザー・プロパティーなどの DB2 Alphablox オブジェクトに対する読み取り/書き込みを行うためのアクセス権限を持っています。 DB2 Alphablox ユーザーは、自身のユーザー・プロパティーに対する読み取り/書き込みアクセス権限だけを持ちます。 *admin* ユーザーや管理者グループのメンバーである別の DB2 Alphablox ユーザーは、1 つ以上のアプリケーションへのアクセス権限を特定のユーザー、ユーザー・グループ、またはすべてのユーザーに付与できます。

アプリケーションへのゲスト・ログイン権限の除去

デフォルトでは、DB2 Alphablox のインストール時に 2 つのデフォルト・ユーザー (ゲストと管理者) が設定されます。ゲスト・ユーザーがアプリケーションにアクセスできないようにするには、アプリケーションの web.xml ファイルを変更し、auth-constraint エレメントと security-role エレメントの両方の role-name エレメントで、AlphabloxUser を AlphabloxAuthenticatedUser に置き換えます。

アプリケーション・サーバーのセキュリティー領域とアプリケーション

開発用のサーバーでアプリケーションを作成し、アプリケーションの展開時に実動用のサーバーにそのアプリケーションを移動することがよくあります。その 2 つのサーバーが同じインスタンス名を共有していない場合は、アプリケーションの移動時に、移行後のアプリケーションの web.xml ファイルに指定されているセキュリティー領域が新しいロケーションのセキュリティー領域と一致しないことがあるので、開発者と管理者はその点に注意しなければなりません。 IBM WebSphere と BEA WebLogic のアプリケーション・サーバーでは、アプリケーションが同じサーバー上の別のセキュリティー領域にあるアプリケーションにアクセスしようとすると、ユーザー認証を求める画面が表示されます。それを防止するには、アプリケーションの web.xml ファイルを変更して、セキュリティー領域を一致させる必要があります。

ユーザーに認証のプロンプトが出される別の状況は、WAR または EAR ファイルとして受け取ったアプリケーションを WebSphere および WebLogic アプリケーション・サーバーにインポートするときです。インスタンス名はチェックされず、異なるインスタンス名である可能性があります。

Web サーバー認証と DB2 Alphablox 認証

DB2 Alphablox では、認証済みのユーザーだけがアプリケーションにアクセスできます。正しく認証されていないユーザーは、データ・ソースに接続することも、データを検索することも、アプリケーション・ページでデータを表示することもできません。 DB2 Alphablox は、ディレクトリー・レベルでアクセスをロックしないので、正しく認証されたユーザーは、ブラウザーを使用して、制限付きのアプリケーションでも開くことができます。ただし、アプリケーション・ページを開くことはできても、そのページにデータは表示されません。

アプリケーションを配置したディレクトリーの下のディレクトリーやファイルをユーザーがブラウズできないようにする方法の詳細については、78 ページの『ディレクトリー・ブラウズを使用不可にする』を参照してください。

デフォルトのインストールの場合、DB2 Alphablox は独自の認証を行います。ただし、別の Web サーバーを使用する場合は、その Web サーバーが認証を行うように DB2 Alphablox をセットアップできます。 Web サーバー認証を行えば、ローカル・エリア・ネットワーク上にすでに存在するユーザー・アカウントに基づいて認証を行えるので、DB2 Alphablox 管理者の管理作業の負担が軽くなります。このセクションでは、Web サーバーが認証を実行するように DB2 Alphablox をセットアップする方法を説明します。

注: WebServerRealm アダプターを選択した場合は、Web サーバーのセキュリティーが他の何よりも優先されます。この場合、DB2 Alphablox のパスワードは意味を持ちません。

Alphablox 8.4 と Apache Tomcat 3.2.4 での Sun iPlanet Web Server セキュリティー・オプションの使用

Sun iPlanet のセキュリティー・モデルは、基本または平文モードの大半のユーザーとブラウザーの組み合わせに適しています。 iPlanet から URL 経由で DB2 Alphablox を管理するには、*admin* というユーザーを Web サーバーのユーザー・データベースに追加します。そうでない場合は、DB2 Alphablox URL に直接アクセスします (例えば、`http://<servername>:<port>/AlphabloxAdmin/home` を使用します)。

iPlanet との併用時に DB2 Alphablox のセキュリティーを使用する場合は、iPlanet Web サーバーのセキュリティーを使用不可にします。そのようにすれば、すべてのユーザーが DB2 Alphablox によって認証されます。

さらに、機能的で安全な環境を確立するために、DB2 Alphablox の設定を正しく行ってください。詳しくは、77 ページの『Web サーバー・ベースのセキュリティーを使用するように DB2 Alphablox を構成する』を参照してください。

IIS NTLM 用の Microsoft セキュリティー・オプションの設定

Microsoft IIS Web サーバーと一緒に DB2 Alphablox を使用する場合は、ユーザーが DB2 Alphablox にログインしたときに、DB2 Alphablox の代わりに IIS が認証を行うように、セキュリティー認証をセットアップできます。 IIS がトラステッド・ドメイン内の Windows ログインを受け入れるように構成されている場合、トラステッド・ユーザーは、追加の認証を行わずに DB2 Alphablox にアクセスできます。つまり、ユーザーがトラステッド・ドメインでログインした状態で DB2 Alphablox にアクセスしているときは、ユーザー名やパスワードを入力する必要はありません。トラステッド・ユーザーとは、通常は大規模な全社レベルの Windows サーバーの Windows トラステッド・ドメインで認証された Windows ユーザーのことです。

この構成では、IIS がすべての要求認証を処理します。 DB2 Alphablox では、認証済みの要求の処理についてもユーザー名が必要です。したがって、Microsoft Internet Explorer のユーザーがブラウザーのデフォルトのセキュリティー設定を (「詳細設定」で) 変更し、ユーザー名とパスワードの入力を求める画面を表示しないようにした場合、DB2 Alphablox はそのブラウザーからの要求を処理できません。

NTLM を使用して、Microsoft Internet Information Services (IIS) と共に DB2 Alphablox が安全に作動するようにセットアップするには、以下の作業を実行します。

- 72 ページの『Microsoft IIS のインストール』
- 72 ページの『DB2 Alphablox をインストールして Microsoft IIS を Web サーバーとして選択する』
- 72 ページの『Microsoft IIS でのセキュリティー設定の構成』
  - 72 ページの『Microsoft IIS でのセキュリティー設定の使用可能化』

– 74 ページの『Microsoft IIS での匿名ユーザー権限の制限』

- 75 ページの『NTLM 用に admin という名前の Windows ローカル・ユーザーを作成する』
- 75 ページの『Alphablox 8.4.1 用に Tomcat 5.5 で NTLM セキュリティーを構成する』
- 75 ページの『DB2 Alphablox へのログイン』

さらに、機能的で安全な環境を確立するために、DB2 Alphablox の設定を正しく行ってください。詳しくは、77 ページの『Web サーバー・ベースのセキュリティーを使用するように DB2 Alphablox を構成する』を参照してください。

Microsoft IIS のインストール

DB2 Alphablox をインストールする*前に* Microsoft IIS をインストールする必要があります。 IIS をインストールする方法の詳細については、Microsoft の資料を参照してください。

DB2 Alphablox をインストールして Microsoft IIS を Web サーバーとして選択する

MicrosoftIIS を Web サーバーとして使用して DB2 Alphablox を実行するには、インストール中に「Web サーバーの構成の選択」ページで、「**新しい Microsoft IIS 構成の作成**」オプションを選択することにより、新しい Microsoft IIS 構成を作成する必要があります。 Web サーバーのルート・ディレクトリーの下に必要なファイルがインストールされます。

IIS と一緒に DB2 Alphablox を使用するには、DB2 Alphablox をインストールする前に IIS をインストールする必要があります。

Microsoft IIS でのセキュリティー設定の構成

Microsoft IIS の Microsoft 管理コンソール (MMC) を使用して、以下の構成を行います。

Microsoft IIS でのセキュリティー設定の使用可能化: 適切なセキュリティー設定を使用可能にするには、以下のようにします。

1. Microsoft IIS 管理コンソールを開きます。
2. 左のウィンドウ・ペインで、該当する Web サーバー (通常は「**既定の Web サイト**」) を右クリックし、「**プロパティ**」を選択します。「既定の Web サイトのプロパティ」ウィンドウが開きます。
3. 「**ディレクトリ セキュリティ**」タブをクリックします。
4. 「**編集**」ボタンを押します。NTLM による Windows トラステッド認証を使用するには、「**Windows 統合認証 (Windows Integrated authentication)**」チェック・ボックスにチェック・マークを付けます (他のボックスにはチェック・マークを付けません)。この方法でログインできるのは、Microsoft Internet Explorer のユーザーだけです。

   この方法は、クライアントのホスト名を指定する場合にのみ有効です。

   注:

- DB2 Alphablox 8.4 で、許可を受けているホスト名を指定するには、DB2 Alphablox ホーム・ページで「**管理**」タブをクリックし、「**一般プロパティー**」セクションの下の「**システム**」リンクをクリックし、「**システム**」ページの「**許可クライアント・リスト**」テキスト・ボックスに、許可を受けているクライアントを入力します。 DB2 Alphablox を実行するマシンのクライアントだけにアクセス許可を制限するには、「**許可クライアント・リスト**」テキスト・ボックスに `localhost` と入力します。
- DB2 Alphablox 8.4.1 では、Tomcat 5.5 に直接、許可クライアントを指定します。 Tomcat 5.5 では、リモート・ホスト・フィルターやリモート・アドレス・フィルターを指定して、許可または拒否する IP アドレスやホスト名を識別することができます。例えば、ローカル・ホスト以外から着信するすべての要求をブロックするには、次のようにします。

  ```
  <Valve
     className="org.apache.catalina.valves.RemoteAddrValve" allow="127.0.0.1" />
  ```

  リモート・ホスト・フィルターおよびリモート・アドレス・フィルターに関する詳細は、http://tomcat.apache.org/tomcat-5.5-doc/config/valve.html をご覧ください。

DB2 Alphablox では、ユーザー名とパスワードの入力を求める画面がユーザーに表示され、そのユーザー名とパスワードに基づいて、DB2 Alphablox による認証が行われます。

セキュリティー・タイプ	**説明**
匿名を許可	このオプションを使用すると、ユーザーは、匿名アカウントを使用してログインできます。 IIS で NTLM セキュリティーを使用している場合は、このオプションにチェック・マークを付けないでください。認証を受けたユーザーは、アプリケーションが配置されている `webapps` ディレクトリーの下のディレクトリーとファイルをブラウズできます。ディレクトリー・ブラウズを防止する方法については、78 ページの『ディレクトリー・ブラウズを使用不可にする』を参照してください。 このチェック・ボックスを選択すると、ユーザーは、リソースに対する無制限のアクセス許可を得ることになります。詳しくは、74 ページの『Microsoft IIS での匿名ユーザー権限の制限』を参照してください。
基本認証 (平文)	Mozilla Firefox とMicrosoft Internet Explorer のブラウザーの場合、このオプションを使用すると、ユーザー ID とパスワードが平文として送信されます。 **注:** Mozilla Firefox ブラウザーから Microsoft IIS 上の DB2 Alphablox アプリケーションにアクセスする環境では、このオプションに必ずチェック・マークを付けてください。 Mozilla Firefox と Internet Explorer の両方のブラウザーから Microsoft IIS 上の DB2 Alphablox アプリケーションにアクセスする環境では、このオプションと「Windows 統合認証 (Windows Integrated authentication)」の両方を選択してください。

セキュリティー・タイプ	説明
Windows 統合認証 (Windows Integrated authentication)	IIS セキュリティーを使用する場合は、この方法をお勧めします。IIS Web サーバーと Microsoft Internet Explorer ブラウザーの組み合わせを使用する環境に限って言えば、このオプションを使用すると、ユーザー ID とパスワードは、平文ではなく Microsoft の独自形式で送信されます。

5. 「**匿名の接続を許可する**」ボックスにチェック・マークを付けた場合 (つまり、トラステッド NTLM セキュリティーを使用しない場合) は、「**匿名の接続を許可する**」チェック・ボックスの右にある「**編集**」ボタンをクリックします。匿名ユーザー名 (通常は IUSR_<HOSTNAME> という形式) が表示されます。匿名ユーザー名を書き留めて、74 ページの『Microsoft IIS での匿名ユーザー権限の制限』の手順に進みます。
6. 「**OK**」ボタンをクリックして、「認証方法」ウィンドウを閉じます。
7. 「**OK**」ボタンをクリックして、「既定の Web サイトのプロパティ」ウィンドウを閉じます。

Microsoft IIS での匿名ユーザー権限の制限: この手順が必要になるのは、トラステッド NTLM セキュリティーを使用しない 場合に限られます。上級の Microsoft IIS および Windows 管理者だけがこの手順を実行すべきです。匿名ユーザーのアクセスを `webapps` ディレクトリーだけに制限しても、Web サーバー上のオブジェクトへのアクセスが必ずしも制限されるわけではありません。それゆえにこの方法が推奨されています。

IIS の下で「匿名を許可」を設定した場合は、以下の手順を実行して匿名ユーザーの権限を制限します。

1. Windows のファイル エクスプローラを開きます。
2. *Inetpub* ディレクトリーを探して右クリックし、メニューを開きます。 *Inetpub* ディレクトリーは、Web サーバーによってアクセス許可が与えられている文書を保管するために IIS が使用するディレクトリーであり、そのデフォルトのディレクトリーは、*C:¥Inetpub* です。
3. ドロップ・リストから「**プロパティ**」を選択して、「**ディレクトリのプロパティ**」ウィンドウを開きます。
4. 「**セキュリティ**」タブをクリックします。
5. そのウィンドウの下部にある「**追加**」ボタンを押して、「**ユーザーとグループの追加**」ウィンドウを開きます。
6. 「**ユーザーの表示**」ボタンをクリックして、ユーザーのリストを表示します。
7. 匿名ユーザーの名前 (IUSR_*hostname* など) を選択します。
8. 「**追加**」ボタンを押して匿名ユーザーを追加します。
9. 「**アクセス権の種類**」リストから、スクロールして「**アクセス権なし**」を選択します。
10. 「**OK**」ボタンをクリックして「ディレクトリのアクセス権」ウィンドウに戻り、「**サブディレクトリのアクセス権を置き換える (Replace Permission on Subdirectories)**」チェック・ボックスを選択します。

11. 「**OK**」ボタンを押します。 Windows は、*InetPub* ディレクトリーの下のディレクトリー・ツリー全体を更新します。これには数分かかります。

NTLM 用に admin という名前の Windows ローカル・ユーザーを作成する

DB2 Alphablox のインストール中に Microsoft IIS 管理アカウントを追加した場合はこの手順をスキップして、ローカル・マシン上に admin アカウントを作成する代わりに、その管理ユーザーを使用することができます。

トラステッド NTLM セキュリティーでは、DB2 Alphablox *admin* ユーザーの Windows アカウントを作成することが必要です。 Microsoft IIS 上の NTLM から認証を受ける場合、デフォルトでは、ユーザー名は、Web サーバーのドメイン (通常はネットワーク・ドメイン) 内のユーザー名に対応します。

ほとんどの場合、DB2 Alphablox *admin* ユーザーは、ネットワーク・ドメイン内のユーザーではありません。したがって、*admin* ユーザーを IIS ホストのローカル・ドメインに追加する必要があります。 IIS の下で *admin* として認証を受ける場合、ユーザー名の形式は、以下のようになります。

machine-name¥admin

Alphablox 8.4.1 用に Tomcat 5.5 で NTLM セキュリティーを構成する

1. Tomcat 5.5 ディレクトリーにある *conf/server.xml* ファイルを開きます。
2. AJP/13 コネクターの定義を探索します。それは以下のようになっています。

```
<Connector port="8009"
   enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

3. tomcatAuthentication 属性を Connector タグに追加し、それを false に設定します。

```
  <Connector port="8009" tomcatAuthentication="false"
    enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

4. 既存の Realm タグをコメントにし、新しい Realm タグを追加することによって、DB2 Alphablox の AlphabloxNTLMRealm を使用するようにタグを置き換えます。

```
<!--
  <Realm className="org.apache.catalina.realm.JAASRealm" debug="10"
  appName="Alphablox"
  userClassNames="com.alphablox.security.jaas.UserPrincipal"
  roleClassNames="com.alphablox.security.jaas.RolePrincipal"
  useContextClassLoader="false" />
-->
   <Realm className="com.alphablox.security.AlphabloxNTLMRealm" />
```

DB2 Alphablox へのログイン

Microsoft IIS を介して *admin* ユーザーとして DB2 Alphablox に接続するときには、以下のようにする必要があります。

- ログイン名に正しい形式を使用します。

  IIS マシンが、*admin* ユーザーが認証を受ける元になるクライアントとは別のドメインにログインしている場合、クライアント・ドメイン名とユーザー名を渡す

必要があります。例えば、*admin* ユーザーが WEBDEV1 ドメインのクライアント・マシンからログインしている場合、ログインの形式は、以下のようになります。

```
WEBDEV1¥admin
```

- ブラウザー認証が正しく設定されていることを確認します。

デフォルトでは、Microsoft Internet Explorer Web ブラウザーは、 Windows のユーザー名とパスワードを使用して、NTLM からユーザーの認証を受けます。別のユーザー (*admin* など) として Microsoft IIS にログインするには、このデフォルトの動作を変更する必要があります。 (*admin* のユーザー名とパスワードは、ユーザー名およびパスワードと一致しないので、ログインしようとしても失敗します。)

Microsoft Internet Explorer ブラウザーでの認証時にユーザー名とパスワードの入力を求める画面を表示するには、以下の手順を実行します。

1. Internet Explorer の「**ツール**」メニューから「**インターネット オプション**」を選択します。
2. 「**セキュリティ**」タブを選択します。
3. イントラネット (LAN) 経由でアプリケーションを使用する場合は、「**ローカル イントラネット ゾーン**」を選択します。
4. 「**レベルのカスタマイズ**」ボタンを押して、セキュリティー・レベルを設定します。
5. 「**ユーザー認証**」セクションの「**ログイン**」セクションで、「**ユーザー名とパスワードを入力してログオンする**」ボタンを選択します。
6. 「**OK**」ボタンをクリックして、「セキュリティの設定」ウィンドウを閉じます。
7. 「**OK**」ボタンをクリックして、「インターネット オプション」ウィンドウを閉じます。

Windows 統合認証では、サーバーからブラウザーに Negotiate ヘッダーが渡され、Kerberos 認証と Windows 統合認証 (以前は NTLM または NT チャレンジ/レスポンスと呼ばれていた) のどちらを使用するかが指定されます。 Negotiate プロセスをスキップするには、以下の手順を実行します。

1. DB2 Alphablox を停止して、コマンド・プロンプトを開き、`Inetpub¥AdminScripts` ディレクトリーに移動します。
2. 以下のコマンドを入力します。

   ```
   cscript adsutil.vbs set w3svc/NTAuthenticationProviders "NTLM"
   ```

3. 認証をチェックするには、以下のように入力します。

   ```
   cscript adsutil.vbs get w3svc/NTAuthenticationProviders
   ```

   デフォルト値は、「Negotiate, NTLM」です。 NTLM だけに設定すると、「NTLM」と表示されます。
4. 次のコマンドを入力します: `iisreset` *`computerName`* `/RESTART` IIS が再始動します。
5. DB2 Alphablox を再始動すると、IIS 経由で DB2 Alphablox にログインできるようになります。

Web サーバー・ベースのセキュリティーを使用するように DB2 Alphablox を構成する

どの Web サーバーを使用している場合でも、利便性とセキュリティーの観点から、以下の DB2 Alphablox 設定を検討してください。

- 『ユーザー・アカウントの自動生成を構成する』
- 78 ページの『IP アドレスのフィルター操作』
- 78 ページの『ディレクトリー権限の設定』
- 78 ページの『ディレクトリー・ブラウズを使用不可にする』

ユーザー・アカウントの自動生成を構成する

ユーザーが正常に Microsoft IIS で認証されていても、Web サーバーが応答できるようになるためには、ユーザーは要求を行うための DB2 Alphablox ユーザー ID を持つ必要があります。 DB2 Alphablox には、すでに IIS や iPlanet に認識されているすべてのユーザーのユーザー・アカウントを個別に作成する代わりに、ユーザー・アカウントを自動的に生成する機能が用意されています。

自動的にユーザー ID を作成するように DB2 Alphablox を構成するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「一般」ページが開きます。
3. 「一般プロパティー」セクションで、「**システム**」リンクをクリックします。
4. 「**ユーザー・プロファイルを自動的に生成**」チェック・ボックスを選択します。
5. 「**保管**」ボタンを押します。

その後、新しいユーザーがログインしようとすると、DB2 Alphablox からユーザー ID とパスワードの入力を求める画面が表示され、その入力に基づいてそのユーザーのアカウントが自動的に作成されます。 Web サーバーとして Microsoft IIS を使用している場合、DB2 Alphablox について Windows 認証を使用するには、71 ページの『IIS NTLM 用の Microsoft セキュリティー・オプションの設定』の説明のとおりに追加の構成を行う必要があります。

制約事項: IIS NTLM セキュリティーを使用している場合に、Mozilla Firefox ブラウザーから DB2 Alphablox を管理する必要があれば、最初のログイン時に必ず Mozilla Firefox ブラウザーからログインするようにしてください。 Mozilla Firefox ブラウザーからログインすると、平文のパスワードでユーザーが作成されます。そうしない場合は、Mozilla Firefox ブラウザーでの認証が失敗します。

重要: 悪意のあるユーザーが Web サーバー・セキュリティーをバイパスするのを防止するために、IP アドレスのフィルター操作を実行できます (この点については、次のセクションを参照してください)。

IP アドレスのフィルター操作

DB2 Alphablox 8.4 (8.4.1 ではない) では、DB2 Alphablox で指定のクライアントからの要求だけを受け入れるための構成を行えます。 Web サーバー・ベースのセキュリティーでは、DB2 Alphablox が Web サーバー・ホスト (つまり、Web サーバー自体または Web サーバー・ホスト・マシンの管理者) からの要求だけを受け入れるように設定を行う必要があります。 DB2 Alphablox ホーム・ページの 「**管理**」タブの「**一般プロパティー**」ページの「**システム**」リンクから、「**許可クライアント・リスト**」プロパティーを使用して、受け入れ可能な DNS 名または IP アドレスのリストを作成できます。ローカル・マシンからの接続だけを受け入れたい場合は、「**許可クライアント・リスト**」を *localhost* に設定します。

DB2 Alphablox 8.4.1 では、Tomcat 5.5 に直接、許可クライアントを指定します。Apache Tomcat 5.5 では、リモート・ホスト・フィルターやリモート・アドレス・フィルターを指定して、許可または拒否する IP アドレスやホスト名を識別することができます。例えば、ローカル・ホスト以外から着信するすべての要求をブロックするには、次のようにします。

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="127.0.0.1" />
```

リモート・ホスト・フィルターおよびリモート・アドレス・フィルターに関する詳細は、http://tomcat.apache.org/tomcat-5.5-doc/config/valve.html をご覧ください。

ディレクトリー権限の設定

Microsoft IIS と一緒に DB2 Alphablox を使用する場合は、Web サーバーの `docroot` または DB2 Alphablox の `webapps` ディレクトリーの下の制限付きのファイルにユーザーがアクセスすることを防止しなければなりません。 IIS でディレクトリー権限を設定する方法の詳細については、IIS の資料を参照してください。

ディレクトリー・ブラウズを使用不可にする

インターネット・ブラウザーのユーザーは、Web サーバー上のディレクトリーにランダムにアクセスできます。ユーザーがこの方法でファイルの検索やアクセスを行うことを防止するには、Web サーバー上でディレクトリー・ブラウズを使用不可にする必要があります。それぞれの Web サーバーの資料を参照してください。

第 12 章 DB2 Alphablox の拡張

この章では、カスタム計算、DHTML クライアント、拡張可能ユーザー・マネージャーのカスタマイズによって、DB2 Alphablox の組み込み機能を拡張する方法について解説します。

概要

DB2 Alphablox には、ほとんどの分析アプリケーションのニーズをサポートする数多くの機能が用意されていますが、開発者がプラットフォームを拡張して、カスタム計算、ユーザー・マネージャー、DHTML クライアントを使用することもできます。カスタム計算は、組み込みの標準計算 API では十分でない場合に必要になることがあります。また、ユーザー・マネージャーの拡張機能によって、認証、許可、パーソナライゼーションのカスタマイズもできます。さらに、DHTML クライアントをカスタマイズすれば、独自のニーズを処理できます。ここでは、計算の拡張機能、ユーザー・マネージャーの拡張機能、DHTML クライアントの拡張機能の概要をそれぞれ説明します。さらに、カスタマイズを処理するための DB2 Alphablox の構成方法についても取り上げます。

計算拡張機能

DB2 Alphablox には、ほとんどの分析アプリケーションの標準的な計算要件をサポートする組み込みの計算 API が用意されています。それでも、組み込み機能によってニーズに十分対応できない場合は、以下の手順によって独自のカスタム計算拡張機能を作成できます。

1. 関数クラスを拡張するために独自の計算拡張機能を作成します。
   a. 以下のメソッドをインプリメントします。

      ```
      public double getResult(double[] variables)
      ```

      `getResult` メソッドは、変数数値のリストに対するユーザー定義計算の結果を戻します。
   b. このファイルが以下のパッケージに属していることを確認してください。

      ```
      com.alphablox.util.calculator
      ```
   c. このファイルの名前は、最初の文字を大文字にして、後続の文字を小文字にしなければなりません。
2. このファイルをコンパイルします。
3. このファイルを JAR ファイルに追加します。
4. その JAR ファイルを以下のディレクトリーに配置します。

   ```
   <alphabloxDirectory>¥lib
   ```
5. 始動ファイルを変更して、この JAR ファイルをクラスパスに組み込みます。このクラスパスの正しい設定方法の詳細については、81 ページの『クラスパスの設定』を参照してください。

カスタム計算の詳細については、「DHTML Client 開発者用リファレンス」の『DataBlox リファレンス』のセクションを参照してください。

ユーザー・マネージャー拡張機能

DB2 Alphablox ユーザー・マネージャーを拡張すれば、DB2 Alphablox にあらかじめ用意されている機能では対応できない、ユーザー認証、許可、パーソナライゼーションのカスタム要件をサポートできます。 DB2 Alphablox ユーザー・マネージャーを拡張するために使用できる Java インターフェースとサーバー・コマンドについては、107 ページの『拡張可能ユーザー・マネージャー』を参照してください。カスタム Java クラスにアクセスするためのクラスパスの定義方法については、81 ページの『クラスパスの設定』を参照してください。

DHTML クライアント拡張機能

DB2 Alphablox DHTML クライアントには、ユーザー用のすぐに使用可能な機能が豊富に用意されています。多くのアプリケーション・ページにとっては、この機能で十分でしょう。しかし、開発者は、DHTML クライアントの背後にある Blox UI モデルを利用して、ページをカスタマイズしたり、カスタム JavaBeans コンポーネントを使用したり、カスタム JSP タグ・ライブラリーを作成したりして、分析アプリケーションをさらに拡張できます。

Blox UI モデルを使用して DB2 Alphablox の機能を拡張する方法の詳細については、「DHTML Client 開発者用ガイド」と「*DHTML Client 開発者用リファレンス*」を参照してください。

カスタム Java クラスをサポートするための DB2 Alphablox の構成

すでに説明したとおり、DHTML クライアント、計算機能、ユーザー・マネージャーのカスタマイズは、カスタム Java クラスによって行えます。カスタム・クラスを作成して使用する場合の注意点を以下にまとめます。

- カスタム・クラス (通常は JAR ファイルとして圧縮) は、サーバー上のどの位置にも格納できます。

  重要: Java クラスを DB2 Alphablox のインストール・ディレクトリー内に格納すると、アップグレード時にファイルが削除される可能性があります。そのため、Alphablox では、カスタム・クラスを DB2 Alphablox ディレクトリーの外部の位置に格納するか、以下のアプリケーション・ディレクトリーに組み込むことを推奨しています。

  ```
  <alphlabloxDirectory>/webapps/<applicationDirectory>
  ```

- DB2 Alphablox 始動バッチ・ファイルに加えた変更は、DB2 Alphablox のアップグレード時に失われます。その変更については、次のセクション (81 ページの『クラスパスの設定』) を参照してください。

クラスパスの設定

DB2 Alphablox でカスタム・クラスを使用するには、始動バッチ・ファイルで指定されている DB2 Alphablox クラスパスにそのクラスのディレクトリーを追加する必要があります。それぞれのアプリケーション・サーバーに応じて、以下の手順を実行してください。

Apache Tomcat インプリメンテーションの場合は、以下のようにします。

1. テキスト・エディターで、<db2alphablox_dir>/appserver/bin/aas.bat ファイル (Windows プラットフォーム) または <db2alphablox_dir>/appserver/bin/aas.sh ファイル (Linux、 UNIX プラットフォーム) を開きます (<db2alphablox_dir> は、DB2 Alphablox のインストール先のディレクトリーです)。
2. クラスパスを設定する行を見つけ、同じ構文を使用してカスタム・クラスの場所を設定する行を追加します。例えば、カスタム・クラスが c:/myclasses/classes directory にある場合は、以下のようにします。

   ```
   set CLASSPATH=%CLASSPATH%;%AAS_LIB%¥xerces.jar
   // Add your class path below
   set CLASSPATH=%CLASSPATH%; c:/myclasses/classes
   ```

   DB2 Alphablox を Windows のサービスとして実行している場合は、<db2alphablox_dir>¥appserver¥conf¥wrapper.properties にあるサービス・パラメーター・ファイルでもクラスパスを設定します。

   ```
   wrapper.class_path=$WRAPPER_AAS_HOME$¥lib¥aasserver.jar
   // Add your class path below
   wrapper.class_path=c:/myclasses/classes
   ```

3. 変更を保管します。
4. 変更を有効にするために DB2 Alphablox を再始動します。

WebLogic アプリケーション・サーバーの場合は、以下の手順を実行します。

1. テキスト・エディターで、<db2alphablox_dir>/bin/aassetup.bat (Windows プラットフォーム) または <db2alphablox_dir>/bin/aassetup.sh (Linux、 UNIX プラットフォーム) を開きます。
2. クラスパスを設定する行を見つけ、同じ構文を使用してカスタム・クラスの場所を設定する行を追加します。例えば、カスタム・クラスが c:/myclasses/classes directory にある場合は、以下のようにします。

   ```
   set CLASSPATH=%CLASSPATH%;%AAS_CP%
   // Add your class path below
   set CLASSPATH=%CLASSPATH%; c:/myclasses/classes
   ```

   DB2 Alphablox を Windows のサービスとして実行している場合は、<db2alphablox_dir>¥bin¥aassetup_nt_service.bat にあるサービス・パラメーター・ファイルでもクラスパスを設定します。

3. WebLogic を再始動します。 DB2 Alphablox を Windows のサービスとして実行している場合は、<BEA>/weblogic700/server/bin の下の installSvc.cmd を再度実行します。

WebSphere を使用している場合は、独自のクラスを含んだ JAR ファイルを作成してから (ルーズ・クラスは作成できません)、その JAR ファイルを

<websphere_dir>/AppServer/lib/ext の下に配置する必要があります。加えた変更を有効にするには、WebSphereを再始動します。

第 13 章 DB2 Alphablox プロパティーの構成

DB2 Alphablox ホーム・ページの「**管理**」タブの下の「**一般**」リンクには、一般的な管理タスクを実行するための一連の Web ページへのリンクが用意されています。これらのすべてのタスクと他のいくつかのタスクは、151 ページの『第 19 章 DB2 Alphablox コンソール・コマンド』の説明のとおり、コンソール・コマンドによって実行することもできます。 DB2 Alphablox にアクセスするには、ブラウザーに以下の URL を入力します。

```
http://<servername>/AlphabloxAdmin/home
```

<servername> は、DB2 Alphablox が実行されるサーバーの名前とポート番号です。この章では、DB2 Alphablox の実行に関するプロパティーを定義するための手順を取り上げ、DB2 Alphablox のログについて解説します。

DB2 Alphablox の管理タスク

「**管理**」タブの下の「**一般**」ページの「**一般プロパティー**」セクションには、DB2 Alphablox の以下の関連タスクを実行するためのインターフェースへのリンクが用意されています。

- 『始動プロパティーの構成』
- 85 ページの『システム・プロパティーの構成』
- 87 ページの『Telnet ポートの指定』
- 88 ページの『DB2 Alphablox Cube Manager の構成』

始動プロパティーの構成

初期インストールと構成のプロセスによって、「**始動**」ページのプロパティーの値がすでに設定されています。このプロパティー・セットは、DB2 Alphablox の始動に必要な絶対最小値に相当します。

注: アプリケーションを展開する前に、85 ページの『システム・プロパティーの構成』の説明のとおりに、DB2 Alphablox のその他のプロパティーについても、正しい値を検討して設定してください。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**一般プロパティー**」セクションで、「**始動**」リンクをクリックし、以下のように DB2 Alphablox の始動プロパティーを表示して変更します。

「**始動**」ページで、以下の始動プロパティーの値を指定できます。

プロパティー名	デフォルト値	説明
インスタンス名	`AlphabloxAnalytics`	この DB2 Alphablox インスタンスの名前です。この名前はインストール時に決定され、DB2 Alphablox を再インストールしない限り変更はできません。 **注:** これは、DB2 Alphablox マシンのドメイン名 (DNS) ではありません。
ログ・ファイル名	`Server.log`	この DB2 Alphablox セッションのログ・ファイルの名前です。 これは、`<repository>/servers/<instanceName>/logs` ディレクトリーにあります。
コマンド・ファイル名	`abc.console`	オプションのコマンド・ファイルの名前です。DB2 Alphablox は、始動時にこの名前でファイルを検索し、このファイルから一連の DB2 Alphablox コマンドを読み取ります。このファイル内のコマンドの構文は、DB2 Alphablox コンソールから入力するコマンドの構文と同じです。
デフォルト・メッセージ・レベル	INFO	表示してログ・ファイルに書き込むメッセージの最小 (重大度が最も低い) レベルです。有効な値を重大度の低いほうから並べると、DEBUG、VERBOSE、INFO、SYSTEM、WARNING、ERROR、FATAL の順になります。メッセージ・レベルの詳細については、159 ページの『メッセージ・レベル』を参照してください。
DB2 Alphablox ログを使用可能にする	はい	DB2 Alphablox ログへのメッセージの書き込みを使用可能にするかどうかを指定します。
DB2 Alphablox アイドル継続時間	15 分	DB2 Alphablox が非アクティブになってから DB2 Alphablox が中断するまでの時間を分単位で指定します。中断とは、DB2 Alphablox がリソースをほとんど使用しない状態のことです。何かの要求があると、サーバーの活動は自動的に再開します。
DB2 Alphablox コンソール・デフォルト・メッセージ・レベル	FATAL	`System.out` に送信するメッセージの最小 (重大度が最も低い) レベルです (DB2 Alphablox の Apache Tomcat インプリメンテーションの場合、出力は Tomcat コンソールに表示されます)。有効な値を重大度の低いほうから並べると、DEBUG、VERBOSE、INFO、SYSTEM、WARNING、ERROR、FATAL の順になります。

プロパティー名	デフォルト値	説明
すべてのユーザー・オブジェクトのロード	使用可能	すべてのユーザー・オブジェクトにメモリーをロードします。デフォルトでは、ユーザー・マネージャーは、ユーザー・オブジェクトを DB2 Alphablox リポジトリーに格納します。多数のユーザーが存在する場合に、システム・プロパティーの「ユーザーのロード保持時間 (分単位)」設定と一緒に使用して、メモリー管理の効率を高め、ユーザー・マネージャーの始動に要する時間を短縮できます。

4. これらのプロパティーに必要な値を入力します。
5. 「**保管**」ボタンをクリックして変更を適用します。

注: これらの変更を有効にするために、DB2 Alphablox をいったん停止してから再始動する必要があります。

システム・プロパティーの構成

初期インストールを行ってからアプリケーションを展開するまでの間に、DB2 Alphablox のシステム・プロパティーを検討して編集します。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**一般プロパティー**」セクションで、「**システム**」リンクをクリックし、以下のように DB2 Alphablox のシステム・プロパティーを表示して変更します。

   「**システム**」プロパティー・ページで、以下のプロパティーの値を指定できます。

プロパティー名	デフォルト値	説明
Web サーバー URL 接頭部	<empty>	正しい値は、使用中の Web サーバーによって異なります。 インストール後にこの値を変更する場合は、まず DB2 Alphablox カスタマー・サポートに連絡してください。
新規ログ開始メッセージ・レベル	INFO	新規ログ・ファイルに書き込む最も重大度の低いレベルを設定します。
新規ログ終了メッセージ・レベル	FATAL	新規ログ・ファイルに書き込む最も重大度の高いレベルを設定します。このプロパティーと「**新規ログ開始メッセージ・レベル**」プロパティーに同じ値 (ERROR など) を設定すると、作成されるログにはそのレベルのメッセージだけが含まれることになります。

プロパティー名	デフォルト値	説明
認証を使用可能にする	はい	ユーザーが DB2 Alphablox リソースと DB2 Alphablox アプリケーションにアクセスする前に認証を必要とするかどうかを設定します。値を「いいえ」にした場合は、すべてのユーザーがパスワードなしでゲストとしてログインすることになります。
ユーザー・プロファイルを自動的に生成	いいえ	ユーザーが DB2 Alphablox にログインした時点で、ユーザー・プロファイルを自動的に作成するかどうかを指定します。「**はい**」に設定すると、パスワードにかかわりなくすべてのユーザー要求が受け入れられ、そのユーザーに DB2 Alphablox へのアクセス許可が与えられます。
許可クライアント・リスト	<empty>	DB2 Alphablox へのアクセスを許可されたユーザーのリストです。この値は、IP アドレスやホスト名をコンマで区切ったリストであり、ワイルドカード文字は使用できません。空のリストを指定した場合は、すべてのユーザーが DB2 Alphablox へのアクセスを許可されます。 **注:** Web サーバー・セキュリティーを使用する場合は、許可クライアントを設定して、その Web サーバー・マシンだけに DB2 Alphablox へのアクセス許可を与える必要があります。
メッセージ履歴サイズ	100	メッセージ履歴領域に保管するメッセージの数を設定します (そのメッセージは、コンソールで表示できます)。その領域が満杯になると、DB2 Alphablox は、先頭に折り返して最も古いメッセージを上書きします (そのメッセージは、ログ・ファイルには保管されたまま残ります)。
終了時に保管	はい	セッションの終了時に、DB2 Alphablox とすべてのプロパティーの現在の状態を保管するかどうかを設定します。管理者やアセンブラーがセッション中に加えた変更を保管していなかった場合でも、この値を「はい」に設定しておけば、その変更が失われることはありません。
デフォルト HTML クライアント・テーマ	coleman	アプリケーション・ページを HTML 形式にレンダリングするときに使用するデフォルトのテーマを設定します。 (詳細については、*開発者用ガイド*を参照してください。)
SMTP サーバー	インストール時に指定	SMTP サーバーの名前を設定し、メールの送信に JavaMail を使用するアプリケーションが正常に機能するようにします。有効な SMTP メール・サーバーの名前を設定してください。

プロパティー名	デフォルト値	説明
ユーザーのロード保持時間 (分単位)	-1	ユーザー・セッションの最後のインスタンスの終了後に、ユーザー・オブジェクトをメモリー内に保持する時間を分単位で設定します。デフォルトの -1 の場合は、DB2 Alphablox が実行している限り、ユーザーが無期限にロードされます。多数のユーザーが存在する場合に、始動プロパティーの「すべてのユーザー・オブジェクトのロード」設定と一緒に使用して、メモリー管理の効率を高め、ユーザー・マネージャーの始動に要する時間を短縮できます。

1. これらのプロパティーに必要な値を入力します。
2. 「**保管**」ボタンをクリックして変更を適用します。

注: プロパティーの変更を有効にするために、DB2 Alphablox をいったん停止してから再始動する必要があります。

Telnet ポートの指定

DB2 Alphablox が通信内容を listen するポートを指定するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**一般プロパティー**」セクションで、「**Telnet コンソール**」リンクをクリックし、以下のように DB2 Alphablox のポートを表示して変更します。

   「**Telnet コンソール (Telnet Console)**」ページで、以下のプロパティー値を変更できます。

プロパティー名	デフォルト値	説明
Telnet コンソール・ポート	23	DB2 Alphablox コンソールの Telnet バージョンのポートです。 DB2 Alphablox がデフォルトの Telnet デバイスになっている場合は、Telnet コンソールにアクセスするときにポート番号を入力する必要はありません。DB2 Alphablox コンソールへの Telnet アクセスを使用不可にするには、この値を 0 (ゼロ) に設定します。
Telnet ユーザー名	admin	Telnet セッションを実行できるユーザーの名前です。
Telnet パスワード	password	Telnet ユーザー名のパスワードです。
Telnet パスワードの確認		「**Telnet パスワード (Telnet Password)**」テキスト・ボックスにパスワードを入力した場合、このテキスト・ボックスでパスワードを確認する必要があります。

プロパティー名	デフォルト値	説明
Telnet タイムアウト	15 分	DB2 Alphablox への Telnet セッションがタイムアウトになるまでの時間を分単位で指定します。Telnet セッションは、タイムアウトになった時点で終了します。
コンソール・マネージャーの再始動	チェックなし	チェック・マークを付けた場合は、「**保管**」ボタンをクリックしたときにコンソール・マネージャーが再始動します。

4. 必要なポート番号を入力します。
5. 「**保管**」ボタンをクリックして変更を適用します。

注: これらの変更を有効にするために、対応するサービス・マネージャーをいったん停止してから再始動する必要があります。サービス・マネージャーを再始動するには、変更したポートの下のボックスにチェック・マークを付けて、「**保管**」ボタンをクリックします。

DB2 Alphablox Cube Manager の構成

DB2 Alphablox Cube Manager のリンクをクリックすると、サーバーにロードできる DB2 Alphablox キューブの数を管理者が制限するためのページが表示されます。DB2 Alphablox の各キューブは大量のシステム・リソースを使用する可能性があるため、管理者は、DB2 Alphablox の実行マシンで使用可能なメモリーに基づいてキューブの数を制限できます。

システムで使用できる DB2 Alphablox キューブの数を制限するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**一般プロパティー**」セクションで、「**DB2 Alphablox Cube Manager**」リンクをクリックし、以下のように DB2 Alphablox のシステム・プロパティーを表示して変更します。
4. 「**最大キューブ数 (Maximum Cubes)**」ボックスにチェック・マークを付けます。
5. 「**最大キューブ数 (Maximum Cubes)**」 テキスト・ボックスに DB2 Alphablox キューブの最大数を入力します。
6. 「**保管**」ボタンをクリックして変更を適用します。

DB2 Alphablox Cube Server を使用して DB2 Alphablox キューブの作成や管理を行う方法の詳細については、「*DB2 Alphablox Cube Server 管理者用ガイド*」を参照してください。

カスタム・プロパティーの定義

カスタム・プロパティーを使用して、ユーザー・ログインの詳細を DB2 Alphablox のユーザー・プロファイルに関連付けることができます。ユーザー・プロファイルの詳細については、55 ページの『第 8 章 ユーザーの定義』を参照してください。 DB2 Alphablox には、カスタム・プロパティーのサンプル・ファイルが用意されており、この作業を簡単に行えるようになっています。

プロパティーの有効値とデフォルト値は、プロパティーの定義から取得できます。開発者は例えば RepositoryBlox の `getUserProperty()` メソッドを使用して、ユーザー・プロパティーをプログラムに従って利用可能です。

注: 値の中にスペースを使用しなければならないカスタム・プロパティーを定義する場合は、スペースを入力する代わりに ` ` を使用します。スペースを入力しても認識されないので、プロパティー値から除去されてしまいます。

新規ユーザー・プロパティーの定義

新規ユーザー・プロパティーを定義するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**カスタム・プロパティー**」セクションで、「**ユーザー定義**」リンクをクリックします。「**ユーザー定義**」ページが表示されます。
4. 「**作成**」ボタンをクリックします。「**ユーザー・カスタム・プロパティーの作成**」ページが表示されます。
5. 「**プロパティー名**」テキスト・ボックスに名前を入力し (必須)、「**表示ラベル**」テキスト・ボックスに説明を入力します。この説明は、「**管理**」タブの下の「**アプリケーション**」リンクからアクセスするページのアプリケーション定義に表示されます。

   注: LDAP ベースのユーザー・マネージャーや独自のエディターを持つ他の外部ユーザー・マネージャー (NTLM など) を使用している場合は、「**プロパティー名**」 の下の「外部リポジトリー」と「ローカル・リポジトリー」のどちらかをまず選択する必要があります。 DB2 Alphablox でカスタム・プロパティーをロードするには、「外部リポジトリー」ラジオ・ボタンをクリックして、ドロップ・リストから該当する項目を選択してください。
6. プロパティーの有効値を複数設定する場合は、「**複数を選択**」ボタンをクリックします。
7. ドロップ・リストから「**ユーザー・アクセス**」レベルを選択します。 (アプリケーション・プロパティーの定義、変更、削除を行うには、常に管理権限が必要です。)
   - **非表示:** このプロパティーは、「新規ユーザーとグループ」、「ユーザーとグループの変更」、「ユーザー・プロファイルの管理」の各ページに表示されません。この値の変更は、JavaScript などのアプリケーションか、このページからしか行えません。

- **表示:** このプロパティーとデフォルト値は、「新規ユーザーとグループ」、「ユーザーとグループの変更」、「ユーザー・プロファイルの管理」の各ページに表示されますが、ユーザーは値を変更できません。
- **編集:** このプロパティーとデフォルト値は、「新規ユーザーとグループ」、「ユーザーとグループの変更」、「ユーザー・プロファイルの管理」の各ページに表示され、変更が可能です。プロパティーに複数の有効値がある場合、ユーザーはリストから値を選択します。プロパティーに有効な値のリストがない場合、ユーザーは値を入力します。

8. 必要に応じて、「**デフォルト値**」にプロパティーのデフォルト値を入力します。
9. 「**値リスト (Value List)**」テキスト・ボックスで、有効なプロパティー値のコンマ区切りストリングを入力します (オプション)。ここに入力した値は、ユーザーが選択を行うためのリストに表示されます。値を入力せずに、「**ユーザー・アクセス**」を「**編集**」に設定すると、それぞれのページでユーザーが値を入力できるテキスト・ボックスが表示されます。
10. 「**保管**」ボタンをクリックして新規プロパティーを保管し、「**一般**」ページに戻ります。

ユーザー・プロパティーの変更

ユーザー・プロパティーを変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**カスタム・プロパティー**」セクションで、「**ユーザー定義**」リンクをクリックします。「**ユーザー定義**」ページが表示されます。
4. 変更する定義を選択して、「**編集**」ボタンをクリックします。「**ユーザー・カスタム・プロパティーの編集**」ページが表示され、入力フィールドに現在の値が表示されます。
5. 必要な変更を加えます。
6. 「**保管**」ボタンをクリックして変更を保管し、「**一般**」ページに戻ります。

ユーザー・プロパティーの削除

ユーザー・プロパティーを削除するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**カスタム・プロパティー**」セクションで、「**ユーザー定義**」リンクをクリックします。「**ユーザー定義**」ページが表示されます。
4. 既存のプロパティーのリストから、削除するプロパティー定義を選択します。
5. 「**削除**」ボタンをクリックしてプロパティーを削除し、「**一般**」ページに戻ります。

   注: プロパティー定義の削除は永久的なものです。削除したプロパティー定義は、回復できません。

新規カスタム・アプリケーション・プロパティーの定義

カスタム・アプリケーション定義を作成するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**カスタム・プロパティー**」セクションで、「**アプリケーション定義**」リンクをクリックします。「**アプリケーション定義**」ページが表示されます。
4. 「**作成**」ボタンをクリックします。「**アプリケーション・カスタム・プロパティーの作成**」ページが表示されます。
5. 「**プロパティー名**」テキスト・ボックスに名前を入力し (必須)、「**表示ラベル**」テキスト・ボックスに説明を入力します。この説明は、「**管理**」タブの下の「**アプリケーション**」リンクからアクセスするページのアプリケーション定義に表示されます。
6. プロパティーの有効値を複数設定する場合は、「**複数を選択**」ボタンをクリックします。
7. ドロップ・リストから「**ユーザー・アクセス**」レベルを選択します。 (アプリケーション・プロパティーの定義、変更、削除を行うには、常に管理権限が必要です。)
   - **非表示:** このプロパティーは、それぞれのページに表示されません。この値の変更は、JavaScript などのアプリケーションか、このページからしか行えません。
   - **表示:** このプロパティーとデフォルト値は、それぞれのページに表示されますが、ユーザーは値を変更できません。
   - **編集:** このプロパティーとデフォルト値は、それぞれのページに表示され、変更が可能です。プロパティーに複数の有効値がある場合、ユーザーはリストから目的の値を選択します。プロパティーに有効な値のリストがない場合、ユーザーは値を入力します。
8. 必要に応じて、「**デフォルト値**」にプロパティーのデフォルト値を入力します。
9. 「**値リスト (Value List)**」テキスト・ボックスで、有効なプロパティー値のコンマ区切りストリングを入力します (オプション)。ここに入力した値は、ユーザーが選択を行うためのリストに表示されます。値を入力せずに、「**ユーザー・アクセス**」を「**編集**」に設定すると、それぞれのページでユーザーが値を入力できるテキスト・ボックスが表示されます。
10. 「**保管**」ボタンをクリックして新規プロパティーを保管し、「**一般**」ページに戻ります。

アプリケーション・プロパティーの変更

アプリケーション・プロパティー定義を変更するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。

3. そのページの「**カスタム・プロパティー**」セクションで、「**アプリケーション定義**」リンクをクリックします。「**アプリケーション定義**」ページが表示されます。
4. 変更する定義を選択して、「**編集**」ボタンをクリックします。「**アプリケーション・カスタム・プロパティーの編集**」ページが表示され、入力フィールドに現在の値が表示されます。
5. 必要な変更を加えます。
6. 「**保管**」ボタンをクリックして変更を保管し、「**一般**」ページに戻ります。

アプリケーション・プロパティーの削除

アプリケーション・プロパティー定義を削除するには、以下のようにします。

1. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
2. 「**管理**」タブをクリックします。「**一般**」ページが表示されます。
3. そのページの「**カスタム・プロパティー**」セクションで、「**アプリケーション定義**」リンクをクリックします。「**アプリケーション定義**」ページが表示されます。
4. 既存のプロパティーのリストから、削除するプロパティー定義を選択します。
5. 「**削除**」ボタンをクリックしてプロパティーを削除し、「**一般**」ページに戻ります。

   注: プロパティー定義の削除は永久的なものです。削除したプロパティー定義は、回復できません。

注: この図では、*UserPropDesc.properties* ファイルの内容に基づいて、「**ユーザーによるプロファイルの編集を許可する**」ドロップ・リストの下にすべての項目が表示されています。

コメント集合の作成と管理

DB2 Alphablox では、マルチディメンション・グリッドのデータ・セルでユーザー・コメントを使用するための管理を行えます。ここでは、DB2 Alphablox「管理」タブの下にある「コメント管理」ページを使用して、特定のデータ・ソースで使用するコメント集合の構成方法について解説します。コメントは、サポートされている Web ブラウザーの DHTML クライアントで表示できます。

注: コメント集合を使用するには、WebSphere に組み込まれているオプションである Microsoft JDBC Driver for SQL Server との連動を実現するための追加の構成手順が必要になります。 DB2 Alphablox カスタマー・サポートに連絡して援助を求めてください。

「コメント管理」ダイアログへのアクセス

新しいウィンドウで開く「コメント管理」ページにアクセスするには、以下の手順を実行します。

1. DB2 Alphablox ホーム・ページを開きます。
2. 「管理」タブをクリックします。

3. 左側のナビゲーション・メニューの「ランタイム管理」の下にある「コメント管理」リンクをクリックします。
4. 「コメント管理」ダイアログが新しいブラウザー・ウィンドウに表示されます。

注: 「コメント管理」ダイアログを使用するには、リレーショナル・テーブルの作成とドロップの権限が必要です。カスタム・コメント・アプリケーションを開発するために CommentsBlox API を使用する場合は、テーブルの選択、挿入、更新、削除、作成、ドロップの権限が必要になります。

データ・ソースの定義とアクセス

データ・ソースを定義してコメント集合の保管に使用するには、以下の手順を実行します。

1. 「データ・ソース」見出しの下にある「名前」選択リストをクリックして、コメント集合の保管先にする定義済みの DB2 Alphablox データ・ソースを選択します。 [注: DB2 Alphablox でデータ・ソースを定義する方法の詳細については、『データ・ソースの定義』を参照してください。]
2. 「**ユーザー名**」と「**パスワード**」の入力フィールドで、データ・ソース内にテーブルを作成する権限を持っているユーザーのログインとパスワードを入力します。 [注: これらのフィールドを空にした場合、DB2 Alphablox のデータ・ソース定義でデータ・ソースのデフォルトのユーザー名とパスワードが設定されていれば、そのユーザー名とパスワードが自動的に送信されます。]
3. 「接続 (Connect)」ボタンをクリックします。定義済みのデータ・ソースとユーザー情報が有効であれば、データ・ソースへの接続が行われ、「コメント管理」ダイアログの「集合」セクションが使用可能になります。

コメント集合の定義

ユーザーがグリッド内のデータ・セルにコメントを追加する前に、まず管理者が関連コメントを格納するためのコメント集合を作成する必要があります。

注: コメント集合内のディメンションは、集合内にコメントが存在しない限り、追加や除去が可能です。

1. 新規コメント集合を作成するには、以下の手順を実行します。
2. 「集合 (Collection)」リスト・ボックスの下にある「作成」ボタンをクリックします。ブラウザー・ウィンドウの右側に 3 つの新規セクションが表示されます。
3. 「名前」セクションにコメント集合の名前を入力します。 [注: 名前にはスペースを使用できません。また、コメントの保管に使用するデータベース内のテーブルの命名要件に準拠する必要があります。]
4. 「フィールド」セクションで、コメント集合内のフィールドを追加したり削除したりできます。デフォルトで、「作成者」、「タイム・スタンプ」、「コメント・テキスト」という 3 つの必要フィールドが組み込まれます。その他のフィールドを追加して、名前を定義し、必要に応じてフィールドの説明を入力します。オプションで CellValue フィールドを追加することもできます。 CellValue フィールドの値は、セルを選択したときに、そのセルの現在の値になり、その値が関連先のコメントに自動的に追加されます。

5. 「ディメンション」セクションで、定義済みの DB2 Alphablox データ・ソースのリストからマルチディメンション・データ・ソースを選択する必要があります。選択したキューブ内で使用可能なディメンションのリストが、リスト内のデータ・ソースの下のテキスト・ボックスに表示されます。
6. ユーザーがコメントを追加する対象のディメンションを選択します。複数のディメンションを選択するには、Control キーを押しながら、それぞれの追加ディメンションをクリックする必要があります。

   注: 選択したディメンションによって、ユーザーが追加するコメントの有効範囲が影響を受けます。その点を以下の 2 つの例で示します。

   - 例 1: 「年」、「製品」、「地域」という 3 つのディメンションを持つキューブがあるとしましょう。「ディメンション」リストからそのうちの 2 つ (「製品」と「年」) だけを選択した場合、「ディメンション」リストで選択していない「地域」ディメンションのメンバーに追加されたコメントは、「地域」の他のすべてのメンバーに表示されます。
   - 例 2: 「年」、「製品」、「地域」という 3 つのディメンションを持つキューブがあるとしましょう。「ディメンション」リストから 3 つのディメンションすべてを選択した場合、コメントは、追加先のセルだけで表示されます。
7. 「保管」ボタンをクリックします。この時点で、新しく定義した集合の名前が集合リストに表示されます。

Microsoft SQL Server または Sybase のデータベースを使用したコメント集合

Microsoft SQL Server または Sybase のデータベースを使用して、非 ASCII 文字を含んだコメントを格納する場合は、以下の手順を実行する必要があります。

1. 定義するコメント集合ごとに、自動生成テーブル内の列のデータ型を `varchar` から `nvarchar` に変更する必要があります。

   以下の 5 つのテーブルは、データ・ソースを使用する定義済みのすべてのコメント集合によって共有されます。

   - 集合: `CollectionName`
   - コメント: `CommentText`
   - コンテキスト: `ContextName`
   - ディメンション: `DimName`
   - フィールド定義: `descr`、`FieldName`
   - フィールド値: `fieldvalue`

   さらに、コメント集合ごとに 1 つの固有のテーブルが作成されます。

   - ABXMBRT_<collectionName>

   <collectionName> は、コメント集合に指定した名前です。以下の列のデータ型を `nvarchar` に変更する必要があります。

   - ABXMBRT_<collectionName>: MemberName
2. アプリケーション内のすべての JSP ページでは、各ページの JSP ページ・ディレクティブで Unicode 文字セットを以下のように指定する必要があります。

```
<%@ page contentType="text/html; charset=UTF-8" %>
```

コメント集合定義の表示

既存の集合の定義を表示するには、以下の手順を実行します。

1. 定義済みのデータ・ソースに接続している場合は、事前定義のコメント集合のリストが「集合」テキスト・ボックス内に表示され、「作成」、「表示」、「削除」の各ボタンが使用可能になります。接続していない場合は、まず前述の『データ・ソースの定義とアクセス』の手順を実行して、データ・ソースに接続してください。
2. 「集合 (Collection)」テキスト・ボックスに表示されているコメント集合のいずれかをクリックしてから、「表示」ボタンをクリックします。集合定義がウィンドウの右半分に表示されます。

コメント集合の削除

定義済みのコメント集合を除去するには、以下の手順を実行します。

1. 定義済みのデータ・ソースに接続します (前述の『データ・ソースの定義とアクセス』を参照してください)。
2. 「集合 (Collection)」テキスト・ボックスで、除去するコメント集合の名前をクリックします。
3. 「削除」ボタンをクリックします。集合名がリストから除去されます。

コメントの追加と表示

コメント集合が定義されていれば、ユーザーは、コメントが使用可能になっているグリッドのデータ・セルにコメントを追加できます。コメントが使用可能になっているデータ・セルをユーザーが右クリックすると、メニューが表示されます。ユーザーは、「コメント」サブメニューで「コメントの追加」か「コメントの表示」を選択できます。

CommentsBlox の詳細については、「*開発者用リファレンス*」の『CommentsBlox』のセクションを参照してください。

リモート PDF プロセッサーの作成

パフォーマンスやメモリー管理の効率を高めたり、複数の DB2 Alphablox ホストで PDF 処理を共有したりするために、専用のリモート・サーバーで PDF エンジンを実行することもできます。

リモート PDF プロセッサーの構成

内部サーバーの代わりにリモート PDF プロセッサーを使用するには、以下の手順を実行します。

1. リモート・サーバー上に *AlphabloxPDFRemote* という名前のルート・ディレクトリーを作成します。
2. このルート・ディレクトリーで、2 つの新規ディレクトリー、*lib* と *repository* を作成します。

3. *repository* ディレクトリー内で、2 つのディレクトリー、*font* と *theme* を作成します。
4. DB2 Alphablox インストール・システムの *db2alphablox_dir/lib/* ディレクトリーから、以下の JAR ファイルをリモート・サーバーの *lib* ディレクトリーにコピーします。
   - bforeport.jar
   - pdfserver.jar
   - xalan.jar
   - xercesImpl.jar
   - xml-apis.jar
   - icu4j_3_4_1.jar
5. すべてのファイルを alphablox *db2alphablox_dir/repository/theme* ディレクトリーから、リモート・サーバー上の *theme* (*repository/theme*) ディレクトリーにコピーします。
6. すべてのファイルを alphablox *db2alphablox_dir/repository/font* ディレクトリーから、リモート・サーバー上の *font* (*repository/font*) ディレクトリーにコピーします。
7. DB2 Alphablox インストール・システムの *db2alphablox_dir/bin/* ディレクトリーから、StartPDFRemoteServer スクリプト・ファイル (Windows システムの場合は *StartPDFReportServer.bat*、Linux および UNIX システムの場合は *StartPDFReportServer.sh*) をリモート・サーバー上の *AlphabloxPDFRemote* ディレクトリーにコピーします。
8. StartPDFRemoteServer スクリプト・ファイルを編集して、以下の変更を加えます。
   - *LIB* 設定を、リモート・サーバーの *lib* ディレクトリーを指すように変更する。
   - JAVA 設定が有効な JRE または JDK を指していることを確認する。
   - 必要に応じて、*LISTEN_PORT* 設定を変更する。
   - スクリプトが現行ディレクトリーで開始することを確認します (デフォルトでは、スクリプト・ファイルは 1 つの上にディレクトリーになります)。
9. このバッチ・ファイルを実行します。リモート PDF プロセッサーが使用可能になります。
10. これを DB2 Alphablox アプリケーションから使用するには、次のセクションの手順を実行して、DB2 Alphablox でリモート・サーバーを使用するための構成を行う必要があります。

リモート PDF レポート管理の構成

PDF レポート処理は、DB2 Alphablox 管理ページの「PDF レポート・ランタイム管理 (PDF Reports Runtime Management)」の設定値によって管理します。リモート・サーバーを構成するには、以下の手順を実行します。

1. DB2 Alphablox 管理ページを開き、「管理」タブをクリックします。
2. ページの左側にある「ランタイム管理」の下にある「PDF レポート (DHTML) (PDF Reports (DHTML))」リンクをクリックして、「PDF レポート (PDF Reports)」ダイアログを開きます。

3. リモート PDF サーバーを構成するために、以下の変更を行います。
   a. 「編集」ボタンをクリックします。
   b. サーバー設定を「内部サーバーの使用 (Use Internal Server)」から「PDF レポート・サーバーの使用 (Use PDF Report Server)」に変更します。
   c. リモート・サーバー用のホスト名とホスト・ポートを定義します。
   d. 必要があれば「圧縮 (Compression)」オプションを選択します。必要がなければ、デフォルトの「圧縮なし (No Compression)」のままにします。これらのオプションは、JDK で定義されている圧縮設定に基づいています。実際には、低と高の設定で圧縮の違いはほとんどありません。低い設定でもファイルはかなり圧縮されます。最も高い設定の場合は、最も低い設定よりもわずかに圧縮率が高くなります。
   e. 変更を保管します。リモート PDF サーバーが正常に構成されていれば、「使用可能 (Available)」という状況になります。

DB2 Alphablox のログ・ファイル

DB2 Alphablox では、DB2 Alphablox のさまざまなイベントやエラーがログ・ファイルに記録されます。 DB2 Alphablox コンソールに書き出されるメッセージはすべて、ログ・ファイルにも書き込まれます。ログ・ファイルは、削除や移動を行わない限りその場所に存在し続けるので、DB2 Alphablox のアクティビティーの履歴情報を保持するための便利な機能です。履歴情報は、管理監査のときや、DB2 Alphablox カスタマー・サポートの援助を受けるときに役立ちます。

ログ・ファイルの循環間隔の設定

デフォルトでは、DB2 Alphablox によって、新しいログ・ファイルが毎日作成されます。さらに、アプリケーション・サーバーの毎回の始動時にも、新しいログ・ファイルが作成されます。管理者は、ログ・ファイルの循環設定を変更できます。そのためには、DB2 Alphablox 管理ページを使用するか、server.properties ファイルを直接編集します。

DB2 Alphablox 管理ページで間隔を設定するには、ブラウザーを開き、DB2 Alphablox 管理ページの「管理」タブに進み、「一般プロパティー」の下の「システム」をクリックします。デフォルトで、「新規ログの循環間隔 (New Log Rollover Interval)」は、毎日循環するように設定されています。この値を編集して、循環しないように設定することもできれば、分、時間、日、週、キロバイトのいずれかを指定することもできます。

ログ・ファイルの循環間隔を server.properties ファイルで設定するには、以下のディレクトリーにある server.properties ファイルを開きます。

```
<repository_dir>/servers/<instanceName>/
```

LogRollOverInterval の設定を検索し、以下の値を使用して目的の設定に変更します。

以下の値のいずれか (デフォルトは 1D) に設定します。

#M - (#) 分後に循環する

#H - (#) 時間後に循環する

#D - (#) 日後に循環する

#W - (#) 週間後に循環する

#K - (#) K バイトが書き込まれたら循環する

NONE - 循環しない

ログ・ファイル名

ログ・ファイルは、以下のディレクトリーにあります。

<*repository_root*>/servers/<*instance_name*>/logs

アクティブ・ログ・ファイルのデフォルト名は、Server.log です。この名前を構成するためのページにアクセスするには、「**管理**」タブ、「**一般プロパティー**」セクション、「**始動**」リンクの順にクリックします。

DB2 Alphablox では、毎回の始動時に新しいログ・ファイルが作成されます。 DB2 Alphablox が始動すると、前のログ・ファイルの名前が以下のように変更されます。

Server<*timestamp*>.log

<*timestamp*> は、ログ内の最後の項目の時刻です。タイム・スタンプは、以下の形式になります。

YYMMDD_HHmmSS

例えば、DB2 Alphablox が最後にシャットダウンされたのが 2000 年 6 月 3 日の 6:22:35 PM であれば、次回の始動時には、古いログ・ファイルの名前が *Server000603_182235.log* に変更され、現在のログ項目を格納する Server.log という新しいログ・ファイルが作成されます。タイム・スタンプの時刻は 24 時間形式であることに注意してください (この例の *18* は、午後 6 時という意味です)。「**管理**」ページでログ・ファイルの名前を変更した場合は、その新しい名前にタイム・スタンプが追加されます。

注: アクティブ・ログ・ファイルには、タイム・スタンプが追加されません。その名前は、「**管理**」ページで指定した名前と全く同じです。

ログ・ファイルの管理

古いログ・ファイルを無期限に保存しておいても何も問題はありませんが、特定の時間が経過した時点で古いファイルをアーカイブに格納するためのプロセスを確立することも可能です。古いログ・ファイルをいつまでも保存しておくと、ディスク・スペースが取られるというデメリットがあるからです。ログ・ファイルが占有するディスク・スペースの量は、DB2 Alphablox のアクティビティーと、「管理」ページで設定するデフォルト・メッセージ・レベルによって決まります。メッセージ・レベルを DEBUG または VERBOSE に設定している場合は、ログ・ファイルがすぐに大きくなっていきます。

古いログ・ファイルは、リネームされて名前にタイムスタンプが付いているので、そのタイム・スタンプを利用して、ログ・ファイルの管理に役立つユーティリティ

ーを作成できます。例えば、ログ・ファイルの古さが 3 か月を超えた時点でそのファイルを削除したり移動したりするスクリプトを作成する、といった具合です。

第 14 章 ユーザー・マネージャーとパーソナライゼーション (Alphablox 8.4.1)

パーソナライゼーション・マネージャーは、アプリケーション・コンテンツをカスタマイズするためのパーソナライゼーション機能を管理します。 DB2 Alphablox 8.4.1 では、ユーザー・マネージャーに用意されているセキュリティー機能とパーソナライゼーション機能が 2 つのモデルに分けられています。

DB2 Alphablox 8.4.1 は、Servlet 2.4 規格と Java 認証・承認サービスの規格に基づくセキュリティー・モデルを提供しています。このセキュリティー・モデルについては、68 ページの『Alphablox 8.4.1 でのセキュリティー・モデル』で説明されています。

パーソナライゼーションについては、新しいパーソナライゼーション・マネージャーが、ユーザーまたはグループが作成、ロード、または削除されたときにそのことを DB2 Alphablox に通知するためのインターフェースを提供します。さらに、パーソナライゼーション・マネージャーにより、ユーザーまたはグループのプロパティーを指定および変更したり、グループに含まれているユーザーとグループの変更を行ったり、独自のユーザーまたはグループを作成して Alphablox リポジトリーに追加することができます。

パーソナライゼーション・マネージャー

DB2 Alphablox 8.4.1 の新しいパーソナライゼーション・マネージャーにより、外部ユーザー・リポジトリーから、ユーザーおよびグループのプロパティーを読み取ることができます。 com.alphablox.personalization パッケージ内のパーソナライゼーション・マネージャー・インターフェースには、ユーザーまたはグループ・オブジェクトが作成、ロード、または削除されたときに、そのことが通知されます。また、この PersonalizationManager オブジェクトにより、ユーザー・リポジトリー内に定義されるプロパティーをもって、DB2 Alphablox ユーザーおよびグループは増強されます。

パーソナライゼーション・マネージャーを使用する場合、*config.xml* ファイルを *alphablox_dir/repository/servers/instance* ディレクトリーに追加してください。 DB2 Alphablox のスタートアップの際に、それは以下のことを行います。

1. リポジトリー内で *config.xml* ファイルを探します。
2. PersonalizationManager オブジェクトのインスタンスを作成します。
3. 提供されたプロパティーを設定します。
4. PersonalizationManager オブジェクトをユーザー・マネージャー・サービスに割り当てます。

以下に示すのは、<Server> タグの例です。

```
<Server>
    <Service name="User Manager">
    <PersonalizationManager
```

```
      className="myPackage.CustomPersonalizationManager"
    />
    </Service>
</Server>
```

パーソナライゼーション・マネージャーには、以下の特性および動作があります。

- 外部ユーザー・リポジトリー内のユーザーまたはグループのプロパティーが DB2 Alphablox 管理ページで使用可能になるのは、同じカスタム・プロパティーを DB2 Alphablox 管理ページで定義してからのことです。
- カスタム・プロパティーと同じプロパティーがいったん定義されると、DB2 Alphablox は読み取り操作だけを実行します。 DB2 Alphablox 管理ページで行われた変更は、メモリー内のみで DB2 Alphablox に対してローカルであり、外部ユーザー・リポジトリーに書き戻されることはありません。それらの値は、次回ユーザー・オブジェクトがロードされるときに上書きされます。
- 接続情報や *config.xml* ファイル内の他の属性を変更する必要がある場合は、Apache Tomcat を再始動する (それによって DB2 Alphablox が再始動する) 必要があります。

JNDI ユーザーおよびグループ・プロパティーへのアクセス

このセクションで提供される情報は DB2 Alphablox 8.4.1 に適用されます。

JNDI ベースのリポジトリー内のユーザーおよびグループのプロパティーは、DB2 Alphablox で使用でき、RepositoryBlox API を介してアクセス可能です。これらのプロパティーを使用可能にするためには、*alphablox_dir/repository/servers/instance/config.xml* ファイル内で以下のようにプロパティーを指定する必要があります。

1. *config.xml* ファイルを *alphablox_dir/repository/servers/instance* ディレクトリーに追加します。

```
<Server>
    <Service name="User Manager">
     <PersonalizationManager
    className="com.alphablox.personalization.jndi.JNDIPersonalizationManager"
        connectionURL="ldap://machineName:port"
        userBase="ou=People,dc=company,dc=com"
        userSearch="(&amp; (uid={0}) (objectclass=person))"
      userProperties="Mail, Phone, Fax"
         groupBase="ou=Groups,dc=company,dc=com"
        groupSearch="(&amp; (cn={0})(objectclass=groupofuniquenames))"
     groupProperties="Description"
      />
    </Service>
</Server>
```

以下の表には、JNDI ユーザーおよびグループのプロパティーにアクセスするために使用可能な属性が記載されています。

属性	説明
connectionURL	必須。 JNDI 接続 URL。例: ldap://machine1.ibm.com:10456

属性	説明
contextFactory	オプション。初期ディレクトリー・コンテキストを作成するときに使用するコンテキスト・ファクトリー。デフォルトは com.sun.jndi.ldap.LdapCtxFactory です。
groupBase	必須。グループを検索するときに使用する JNDI ベース。
groupProperties	必須。 JNDI リポジトリーから DB2 Alphablox ユーザー・オブジェクトに割り当てるユーザー・プロパティー。このプロパティーは、DB2 Alphablox 管理ページで定義されたすべてのカスタム・ユーザー・プロパティーをオーバーライドします。
groupSearch	必須。グループを検索するときに使用する JNDI 検索ストリング。グループ検索には MessageFormat 置換構文 {0} が含まれることがあります。これは検索対象のグループの固有の名前に置き換えられることになります。
groupSubtree	オプション。グループを検索するときに groupBase のサブツリーが使用されるかどうかを判別します。
password	オプション。接続 URL に接続するときに使用するパスワード。
userBase	必須。ユーザーを検索するときに使用する JNDI ベース。
userName	オプション。接続 URL に接続するときに使用するユーザー名。
userProperties	必須。 JNDI リポジトリーから Alphablox ユーザー・オブジェクトに割り当てるユーザー・プロパティー。このプロパティーは、DB2 Alphablox Admin Pages で定義されたすべてのカスタム・ユーザー・プロパティーをオーバーライドします。
userSearch	必須。ユーザーを検索するときに使用する JNDI 検索ストリングを設定します。ユーザー検索には MessageFormat 置換構文 {0} が含まれることがあります。これは検索対象のユーザーの固有の名前に置き換えられます。
userSubtree	オプション。ユーザーを検索するときに使用する userBase のサブツリー。

RepositoryBlox API を使用して JSP ファイル内のユーザー・プロパティーの値にアクセスするには、以下のようにします。

1. DB2 Alphablox 管理ページを使用して、同じカスタム・プロパティーを追加します。「**管理**」 タブ > 「**一般プロパティー**」ページ > 「**カスタマー・プロパティー (Customer Properties)**」セクションに進みます。 これらの手順の詳細については、89 ページの『カスタム・プロパティーの定義』を参照してください。

2. ご使用の JSP ファイルで、RepositoryBlox API を介してこのカスタム・プロパティーとその値にアクセスします。

```
<%@ taglib uri="bloxtld" prefix="blox" %>
...
<blox:repository id="myRepository" />
...
<html>
<head>
    <blox:header />
</head>
<body>
Your region is: <%= myRepository.getUserProperty("region") %>
...
```

3. Apache Tomcat を再始動して (それによって DB2 Alphablox が再始動される)、*config.xml* ファイルへの変更が有効になるようにします。

第 15 章 ユーザー・マネージャー (Alphablox 8.4)

この章では、LDAP 統合のための DB2 Alphablox ユーザー・マネージャー機能の構成と使用の方法や、拡張可能ユーザー・マネージャー・パーソナライゼーション・エンジンを拡張してカスタム・セキュリティーをインプリメントする方法について解説します。

DB2 Alphablox ユーザー・マネージャーの概要

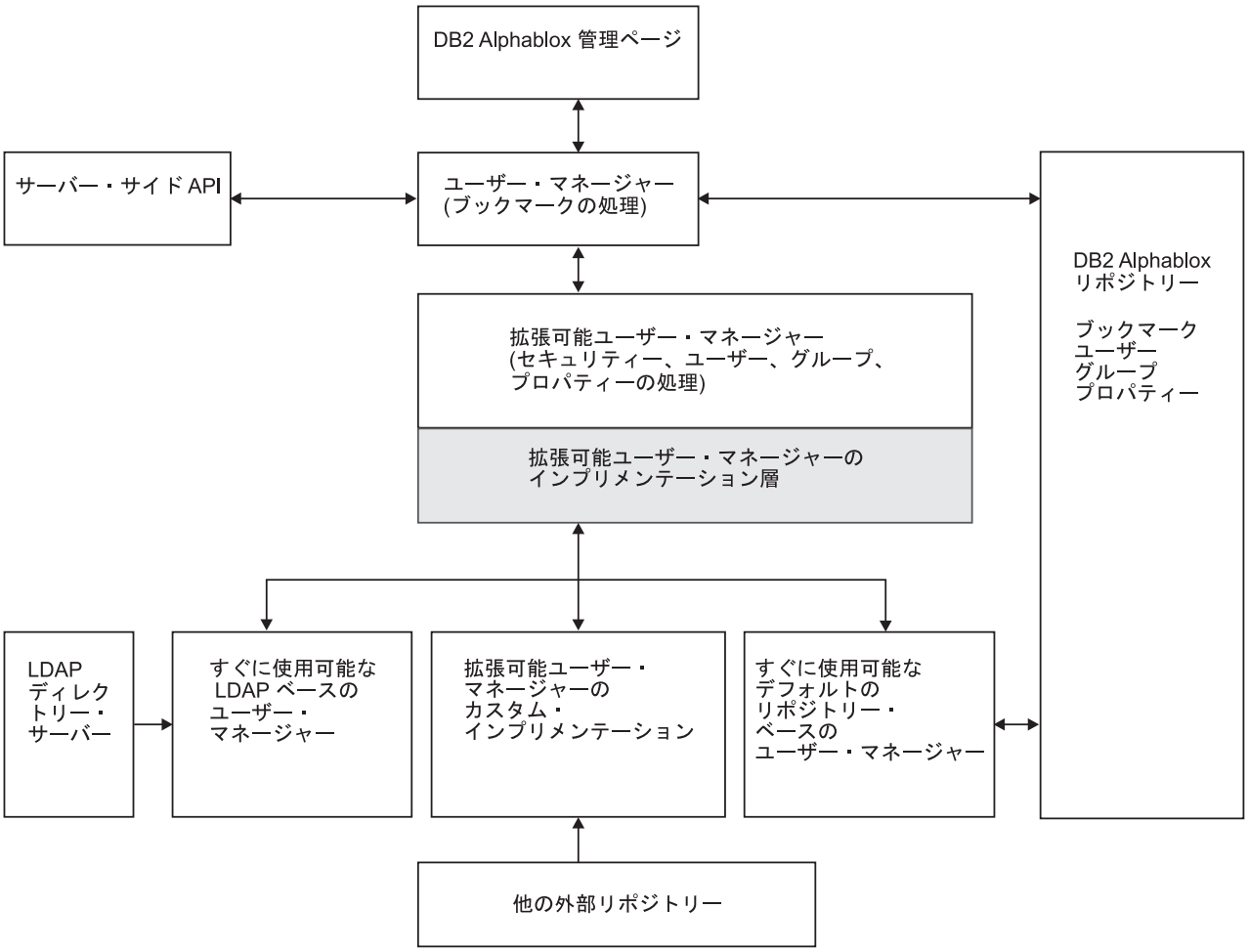
注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

DB2 Alphablox ユーザー・マネージャーでは、ユーザーの認証と許可を管理します。また、アプリケーション・コンテンツをカスタマイズするためのパーソナライゼーション機能も活用できます。 DB2 Alphablox は、デフォルトでは、DB2 Alphablox リポジトリーと J2EE セキュリティー API を使用してユーザー情報とグループ情報を管理します。ユーザーの身元と役割の確認のために、2 つの J2EE セキュリティー・メソッド (`isUserInRole()` と `getUserPrinciple()`) を使用します。

DB2 Alphablox には、すぐに使用可能な Lightweight Directory Access Protocol (LDAP) 統合ソリューションも用意されています。 DB2 Alphablox では、このソリューションに基づいて、LDAP ディレクトリー・サーバーで DB2 Alphablox のユーザー、グループ、カスタム・プロパティーを認識し、ユーザーの認証と許可を行います。

ユーザー・マネージャーは、拡張可能ユーザー・マネージャーというパーソナライゼーション・エンジンの基盤の上に存在します。カスタム・セキュリティーが望ましい環境では、拡張可能ユーザー・マネージャー・パーソナライゼーション・エンジンのインターフェースを使用して、すぐに使用可能な 2 つのセキュリティー・ソリューション (DB2 Alphablox リポジトリーと LDAP) のいずれかを拡張します。あるいは、NTLM などの別の外部ユーザー・マネージャーや既存の Enterprise JavaBeans (EJB) に接続することも可能です。

ユーザー・マネージャーと拡張可能ユーザー・マネージャー・パーソナライゼーション・エンジンのアーキテクチャーを以下の図に示します。



この図で押さえておきたいのは、以下の点です。

- ユーザーやグループや関連プロパティーの情報には、RepositoryBlox を使用したサーバー・サイド API によってプログラマチックにアクセスできます。
- DB2 Alphablox には、セキュリティーとパーソナライゼーションのためのすぐに使用可能な 2 つのソリューション (DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーと LDAP ベースのユーザー・マネージャー) が用意されています。
- デフォルトのリポジトリー・ベースのユーザー・マネージャーは、DB2 Alphablox リポジトリーに対して読み書きを行います。
- すぐに使用可能な LDAP ベースのユーザー・マネージャーは、LDAP サーバーに対して読み取り操作のみを実行します。
- 拡張可能ユーザー・マネージャー API を使用して、NTLM などの別の外部リポジトリーや既存の Enterprise JavaBeans (EJB) を使用することも可能です。
- 使用するリポジトリー (DB2 Alphablox、LDAP、その他の外部ソース) にかかわらず、RepositoryBlox API を使用すれば、いつでもユーザー・プロパティーにアクセスできます。

拡張可能ユーザー・マネージャー

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

DB2 Alphablox ユーザー・マネージャーの中心に位置するのが、拡張可能ユーザー・マネージャー・パーソナライゼーション・エンジンです。この機能によって、以下のようなことが可能になります。

- あらかじめ用意されているユーザー認証や許可をカスタマイズすること
- 外部ユーザー・マネージャーに接続して、DB2 Alphablox リポジトリーの外部にある他のソースのユーザー/グループ情報にアクセスすること
- J2EE セキュリティー API 以外のセキュリティー設定を使用すること

セキュリティーとパーソナライゼーションのためのすぐに使用可能な 2 つの DB2 Alphablox インプリメンテーション (DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーと LDAP ベースのユーザー・マネージャー) は、両方ともこのエンジンを基盤としています。 LDAP ベースのユーザー・マネージャーを使用するには、幾つかの構成手順が必要になります。これらの手順については、107 ページの『LDAP ベースのユーザー・マネージャー』を参照してください。カスタム・セキュリティーをインプリメントしたり、すぐに使用可能な 2 つのセキュリティー・インプリメンテーションのいずれかをカスタマイズしたりするために、拡張可能ユーザー・マネージャーを拡張することも可能です。拡張可能ユーザー・インターフェースを拡張するための詳細については、112 ページの『拡張可能ユーザー・マネージャーのインターフェース』を参照してください。

LDAP ベースのユーザー・マネージャー

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

すぐに使用可能な形で DB2 Alphablox に用意されている拡張可能ユーザー・マネージャーの 1 つのインプリメンテーションでは、LDAP ディレクトリー・サーバーを使用して、DB2 Alphablox のユーザー、グループ、カスタム・プロパティーを認識します。

- DB2 Alphablox は、LDAP サーバーに対して読み取り操作のみを実行します。
- 管理者は、LDAP サーバーですでに定義されている DB2 Alphablox のユーザーとグループだけを作成できます。
- 管理者は、DB2 Alphablox ホーム・ページの「**管理**」タブから名前やメンバーシップを変更できません。
- DB2 Alphablox ホーム・ページの「**管理**」タブからユーザーやグループを削除しても、その項目は基本リポジトリーから除去されず、LDAP サーバーにも影響はありません。
- “admin” ユーザーは、LDAP ユーザーでない限りデフォルトでは使用できなくなります。
- “guest” ユーザーと “public” グループは、LDAP に存在しない場合でも使用可能です。

DB2 Alphablox で LDAP ユーザー・マネージャーを使用するための構成

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

LDAP ベースのユーザー・マネージャーは、WebSphere、WebLogic、Tomcat の各アプリケーション・サーバーと共に使用できます。 DB2 Alphablox ですぐに使用可能な LDAP ディレクトリー・サーバーとの統合を利用するための構成を行うには、以下の手順を実行します。

1. ExtUserManager Telnet コンソール・コマンドを使用して、108 ページの『LDAP ベースのユーザー・マネージャーのプロパティーの設定』の説明のとおりに、LDAP を使用するサーバーを設定します。
2. 以下の Telnet コンソール・コマンドを使用して、サーバー・プロパティー autoCreateUsers を true に設定します。

   ```
   set server autoCreateUsers true
   ```

   注: autoCreateUsers のデフォルト値は、false です。 LDAP サーバーから正しく認証されたユーザーを自動的に作成する機能を DB2 Alphablox で使用可能にするには、このプロパティーを true に設定する必要があります。

   注: 外部ユーザー・マネージャーでは、“public” グループと “guest” ユーザーを定義しておく必要があります (ダミー・インスタンスでもかまいません)。

Apache Tomcat 構成の追加要件

1. LDAP ディレクトリー・サーバーで AlphabloxAdministrator グループを作成します。
2. 少なくとも 1 人のユーザーを AlphabloxAdministrator グループに追加するか、少なくとも 1 人のユーザーが含まれる既存のグループをサブグループとして AlphabloxAdministrator グループに追加します。

LDAP ベースのユーザー・マネージャーのプロパティーの設定

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

1. DB2 Alphablox に対する Telnet コンソール接続を開きます。

   注: DB2 Alphablox ホーム・ページの「**管理**」タブから Telnet コンソールにアクセスした場合は、変更を有効にするために DB2 Alphablox を再始動する必要があります。標準の Telnet コンソールを使用していれば、DB2 Alphablox を再始動する必要はありません。

2. 以下のコマンドを入力します。

   ```
   ExtUserManager setToDefaults ldap "<ldapProps>"
   ```

   <ldapProps> は、プロパティーと値のペアをセミコロンで区切ったリストです。例えば、以下のとおりです。

   ```
   "host:localhost;port:389:admin:cn=DirectoryManager;
   password:password;debug:false;base:dc=alphablox,dc=com"
   ```

注: 上のリストは、改行なしで 1 行に入力する必要があります。

ExtUserManager コマンドの実行後、DB2 Alphablox は、デフォルトの DB2 Alphabloxリポジトリー・ベースのユーザー・マネージャーから LDAP ベースのユーザー・マネージャーに切り替えます。

注: この切り替えによって、現在のすべてのユーザーが切断されます。

<ldapProps> ストリングで定義できるすべての必須プロパティーとオプション・プロパティーを以下の表にまとめます。

プロパティー	説明	例
host	LDAP ディレクトリー・サーバーのホスト	host:localhost
port	LDAP ディレクトリー・サーバーのポート	port:389
admin	LDAP ディレクトリー・サーバーの管理者ユーザー名	admin:cn=Directory Manager
password	LDAP ディレクトリー・サーバーの管理者パスワード	password:myPassword
base	LDAP ディレクトリー・サーバーの検索ベース	base:o=myCompany.com
debug	[オプション] デバッグ・モード設定 (デフォルトは false)	debug:true

ヒント: LDAP の検索ベースでは、検索の開始点を指定します。ディレクトリー階層内の項目の識別名を指定することによって、検索の効率を高めることができます。

カスタム・ユーザー・プロパティーへのアクセス

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

アプリケーションにさらに進んだパーソナライゼーションを備えるために、カスタム・ユーザー・プロパティーを定義することができます。カスタム・ユーザー・プロパティーを定義しておけば、各ユーザーのプロパティーにそれぞれ異なる値を割り当て、そのプロパティー値に RepositoryBlox の API からプログラマチックにアクセスできます。 LDAP ベースのユーザー・マネージャーまたは他の外部ユーザー・マネージャーを使用する場合、DB2 Alphablox は、外部プロパティーとして定義されているカスタム・ユーザー・プロパティーだけをロードします。

カスタム・ユーザー・プロパティーは、DB2 Alphablox ホーム・ページの「**管理**」タブで定義できます。カスタム・プロパティー定義ページで、「**外部プロパティー**」のボックスに必ずチェック・マークを付けてください。詳細な手順については、89 ページの『カスタム・プロパティーの定義』を参照してください。カスタム・ユーザー・プロパティーを外部プロパティーとして定義しておけば、外部カスタム・ユーザー・プロパティーの値に関する情報が (RepositoryBlox.getUserProperty("myExternalUserProp") 呼び出しなどによって)

要求された時点で、DB2 Alphablox ユーザー・マネージャーは、その値を取り込むために自動的に外部ソースにアクセスします。

実行時の動作

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

LDAP ベースのユーザー・マネージャーの実行時の動作を以下にまとめます。

- DB2 Alphablox は、実行時にグループを自動作成しません。管理者は、DB2 Alphablox ホーム・ページの「**管理**」タブを使用して、LDAP グループを明示的に登録する必要があります。
- ユーザーが LDAP サーバーから正しく認証を受けており、autoCreateUsers サーバー・プロパティーが Telnet コマンドで true に設定されている場合 (その方法については、108 ページの『DB2 Alphablox で LDAP ユーザー・マネージャーを使用するための構成』を参照)、DB2 Alphablox は、ユーザーを自動作成します。
- DB2 Alphablox は、外部カスタム・プロパティーとして定義されている LDAP のユーザー・プロパティーとグループ・プロパティーだけをロードします。

拡張可能ユーザー・マネージャーの Telnet コンソール・コマンド

`ExtUserManager` telnet コンソール・コマンドで、ユーザーやグループを管理するために別のクラスを使用したり、あるいは外部リポジトリーを使用するように、DB2 Alphablox に指示することができます。

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

ExtUserManager の一般的な構文は、以下のとおりです。

```
ExtUserManager <Property> <Value>
```

`Property` は、以下のいずれかです。

有効なプロパティー	説明	値
`umclassname`	ユーザー・マネージャー・クラス名	値は、`IUserManager` インターフェースをインプリメントするクラスパスに入っている有効なクラス名でなければなりません。そうでない場合は、エラーになります。詳しくは、112 ページの『カスタム・セキュリティーのインプリメンテーション』を参照してください。
`userclassname`	ユーザー・クラス名	値は、*IUser* インターフェースをインプリメントするクラスパスに入っている有効なクラス名でなければなりません。そうでない場合は、エラーになります。詳しくは、112 ページの『カスタム・セキュリティーのインプリメンテーション』を参照してください。

有効なプロパティー	説明	値
groupclassname	グループ・クラス名	値は、*IGroup* インターフェースをインプリメントするクラスパスに入っている有効なクラス名でなければなりません。そうでない場合は、エラーになります。詳しくは、112 ページの『カスタム・セキュリティーのインプリメンテーション』を参照してください。
customstartupproperty	ユーザー・マネージャーの始動時に値を読み取るために、開発者がコード内で使用できるカスタム始動プロパティー。	このプロパティーの値。
setToDefaults	使用するデフォルト・リポジトリー	次のセクション (『デフォルト・リポジトリーの設定』) を参照してください。

注: 新規カスタム始動プロパティーを設定した場合は、DB2 Alphablox をいったん停止してから再始動する必要があります。

デフォルト・リポジトリーの設定

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

デフォルトの LDAP ベースのユーザー・マネージャーを使用するようにすべてのプロパティーを設定するには、以下の Telnet コンソール・コマンドを使用します。

```
ExtUserManager setToDefaults ldap <ldapProps>
```

ldapprops は、LDAP への接続に必要なすべての情報が含まれているプロパティーです。 DB2 Alphablox を再始動する必要はありませんが、現在実行中のすべてのユーザーが切断されます。 LDAP ベースのユーザー・マネージャーを使用するための構成手順や ldapprops プロパティーの構文の詳細については、107 ページの『LDAP ベースのユーザー・マネージャー』を参照してください。

デフォルトの DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーを使用するようにすべてのプロパティーをリセットするには、以下の Telnet コンソール・コマンドを使用します。

```
ExtUserManager setToDefaults repository
```

このコマンドを実行すると、ユーザー・マネージャーは、最新のユーザー情報を DB2 Alphablox リポジトリーから入手するために、いったん停止してから再始動します。 DB2 Alphablox を再始動する必要はありませんが、現在実行中のすべてのユーザーが切断されます。

外部ユーザー・リポジトリーで使用されなくなったユーザーとグループの除去

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

以下のコマンドを実行すると、拡張可能ユーザー・マネージャーは、外部ユーザー・リポジトリー (LDAP や NTLM など) で使用されなくなったユーザーとグループをリポジトリーから除去します。

```
ExtUserManager clean
```

拡張可能ユーザー・マネージャーのインターフェース

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

拡張可能ユーザー・マネージャーのフレームワークには、IUserManager、IUser、IGroup という 3 つの主要なインターフェースがあります。それぞれのインターフェースの目的とメソッドを以下の表にまとめます。また、各インターフェースをインプリメントしたクラスも示します。

インターフェース	説明	インプリメントしたクラス
IUserManager	実行時のユーザーやグループの作成や検索を受け持ちます。	AbstractUserManager
IUser	基盤となるリポジトリーに定義されているユーザーの認証、許可、ユーザー・プロパティーの読み取り専用ビューを提供します。	AbstractUser
IGroup	基盤となるリポジトリーに定義されているメンバーシップ情報やグループ・プロパティーの読み取り専用ビューを提供します。	AbstractGroup

これらのインターフェースとそれぞれをインプリメントしたクラスは、com.alphablox.personalization パッケージに入っています。このパッケージの Javadoc は、以下のディレクトリーから入手できます。

```
<db2alphablox_dir>/system/documentation/javadoc/blox/index.html
```

<db2alphablox_dir> は、DB2 Alphablox のインストール先のディレクトリーです。

カスタム・セキュリティーのインプリメンテーション

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

拡張可能ユーザー・マネージャーには、認証と許可に使用するユーザー、グループ、ユーザー・マネージャー・クラスを識別するための 3 つの DB2 Alphablox プロパティーがあります。

- umclassname

  DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーのデフォルト値は、com.alphablox.personalization.repository.RepUserManager です。 LDAP ベースのユーザー・マネージャーのデフォルト値は、com.alphablox.personalization.ldap.LDAPUserManager です。

- `userclassname`

  DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーのためのこのプロパティーのデフォルト値は、`com.alphablox.personalization.repository.RepUser` です。 LDAP ベースのユーザー・マネージャーのデフォルト値は、`com.alphablox.personalization.ldap.LDAPUser` です。

- `groupclassname`

  DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーのためのこのプロパティーのデフォルト値は、`com.alphablox.personalization.repository.RepGroup` です。 LDAP ベースのユーザー・マネージャーのデフォルト値は、`com.alphablox.personalization.ldap.LDAPGroup` です。

独自のユーザー/グループ/ユーザー・マネージャー・クラスを作成してから、Telnet コンソール・コマンド `ExtUserManager` を使用して、そのユーザー/グループ/ユーザー・マネージャー・クラスを指すようにこれらの DB2 Alphablox プロパティーに新しい値を設定できます。

あらかじめ用意されているセキュリティー機構の一部またはすべてをカスタマイズする場合に拡張する必要のあるクラスを以下の表にまとめます。

目標	解決策
デフォルトの DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーによるカスタム認証または許可	`AlphabloxUser` クラスの拡張
デフォルトの LDAP ベースのユーザー・マネージャーによるカスタム認証または許可	`ldapUser` クラスの拡張
シングル・サインオン	`AlphabloxUser` クラスの `getPassword()` メソッドのインプリメント
DB2 Alphablox または LDAP 以外のリポジトリーからのユーザー/グループ・プロパティーの読み取り	`AlphabloxUser` クラスまたは `AlphabloxGroup` クラスの拡張
DB2 Alphablox または LDAP 以外のリポジトリーからの動的グループ・メンバーシップ情報の読み取り	`AlphabloxUser` クラスまたは `AlphabloxGroup` クラスの拡張
NTLM サポート	拡張可能ユーザー・マネージャーのインターフェースの完全インプリメンテーション

注: DB2 Alphablox でカスタム・クラスを使用するには、始動バッチ・ファイルで指定されている DB2 Alphablox クラスパスにそのクラスのディレクトリーを追加する必要があります。クラスパスの設定方法の詳細については、81 ページの『クラスパスの設定』を参照してください。

シングル・サインオン

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

DB2 Alphablox データ・ソースが DB2 Alphablox のユーザー名とパスワードを使用するように構成されている場合、シングル・サインオンをインプリメントするために、`AlphabloxUser` クラスの `getPassword()` メソッドを使用して、base64 エンコードのパスワードを取得できます。さまざまな状況でのシングル・サインオンのソリューションを以下の表にまとめます。

ユーザー・マネージャー	LDAP を使用したデータ・ソースか?	認証方式
LDAP ベースのユーザー・マネージャー	はい	DB2 Alphablox の基本認証 (ブラウザー・チャレンジの認証)、アクションは必要なし
LDAP ベースのユーザー・マネージャー	いいえ	`getPassword()` の使用
その他のリポジトリー	どちらの場合もある	`getPassword()` の使用

カスタム・セキュリティーの例

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

カスタム・セキュリティーのために拡張可能ユーザー・マネージャーを拡張する 3 つの例を以下にまとめます。それぞれの例で、以下の点を示します。

- デフォルトの DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーによって使用されるデフォルトのユーザー/グループ/ユーザー・マネージャー・クラスを拡張する方法
- Telnet `ExtUserManager` コマンドを使用して新規クラス名を設定する方法

外部ユーザー・マネージャーの完全インプリメンテーションの詳細な例については、以下のディレクトリーにあるシンプル・ユーザー・マネージャーの例を参照してください。

```
<db2alphablox_dir>/system/documentation/admin/Examples/
```

Java ソース・ファイルが用意されています。

例 1: DB2 Alphablox で外部ユーザー・マネージャーを使用するための設定

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

この例では、デフォルトの DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーをカスタマイズする方法を示します。この場合、ユーザー・マネージャーの始動時に、ファイルからカスタム・プロパティーのリストを読み取ることもできます。手順は、以下のようになります。

1. パッケージにカスタム・ユーザー・マネージャー・クラスを作成します (この例では、 com.myCompany.user というパッケージに MyUserManager というクラスを作成します)。
   - このクラスは、DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーで使用される AlphabloxUserManager クラスを拡張したものです。
2. 以下の Telnet コンソール・コマンドを使用して、umclassname サーバー・プロパティーに新しい値を設定します。

   ```
   ExtUserManager umclassname com.myCompany.user.MyUserManager
   ```

簡単なカスタム MyUserManager クラスの例は、以下のようになります。

```
package com.myCompany.user;
import com.alphablox.personalization.alphablox.*;
import com.alphablox.personalization.*;
import java.util.*;

public class MyUserManager extends AlphabloxUserManager {
   void start(Properties props) throws PEngineException  {
      super.start(props);
      String myXmlFile = (String) prop.get("customstartupprop");
      // read the xml file and do other things
   }
}
```

例 2: DB2 Alphablox で別のユーザー・クラスを使用するための設定

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

この例では、デフォルトの DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーでユーザー許可をカスタマイズする方法を示します。この場合は、DB2 Alphablox またはアプリケーションにアクセスして、*IUser* インターフェースの isUserInRole() メソッドを上書きする許可をすべてのユーザーに与えます。手順は、以下のようになります。

1. パッケージにカスタム・ユーザー・クラスを作成します (この例では、com.myCompany.user というパッケージに MyUser というクラスを作成します)。
   - このクラスは、DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーで使用されるデフォルトの AlphabloxUser クラスを拡張したものです。
2. 以下の Telnet コンソール・コマンドを使用して、userclassname サーバー・プロパティーに新しい値を設定します。

   ```
   ExtUserManager userclassname com.myCompany.user.MyUser
   ```

   簡単な MyUser の例は、以下のようになります。

   ```
   package com.myCompany.user;
   import com.alphablox.personalization.repository.*;
   import com.alphablox.personalization.*;
   import java.util.*;
   ```

```
public class MyRepositoryUser extends AlphabloxUser {
   public boolean isUserInRole(HttpServletRequest req,
   String [] roles) throws PEngineException  {
      // Everybody can get in
      return true;
   }
}
```

例 3: DB2 Alphablox で別のグループ・クラスを使用するための設定

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

この例では、デフォルトの DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーをカスタマイズして、グループ・メンバーシップの確認が行われるたびにシステム・メッセージを出力する方法を示します。手順は、以下のようになります。

1. パッケージにカスタム・グループ・クラスを作成します (この例では、`com.myCompany.user` というパッケージに `MyGroup` というクラスを作成します)。
   - このクラスは、DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーで使用されるデフォルトの `AlphabloxGroup` クラスを拡張したものです。
2. 以下の Telnet コンソール・コマンドを使用して、`groupclassname` サーバー・プロパティーに新しい値を設定します。

   ```
   ExtUserManager groupclassname com.myCompany.user.MyGroup
   ```

簡単な `MyGroup` の例は、以下のようになります。

```
package com.myCompany.user;
import com.alphablox.personalization.alphablox.*;
import com.alphablox.personalization.*;
import java.util.*;

public class MyGroup extends AlphabloxGroup {
   public boolean containsUser(IUser user,
          boolean checkSubGroups) throws PEngineException {
      boolean exists =           super.containsUser(user,checkSubGroups);
      System.out.println("User membership checked.");
      return exists;
   }
}
```

インターフェース・メソッドの相互参照

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

このセクションは、拡張可能ユーザー・マネージャーの 3 つのインターフェースのリファレンス資料です。インターフェースのメソッドごとに、説明と構文と使用上の注意を示します。

カスタマイズ版のユーザー・マネージャーの詳細な例については、以下のディレクトリーにあるシンプル・ユーザー・マネージャーの例を参照してください。

<db2alphablox_dir>/system/documentation/admin/Examples/

インターフェース	メソッド
IUserManager	findGroup() findUser() getExternalProperties() getPrincipleUserName() hasExternalEditor() resume() setCaseSensitiveGroups() setCaseSensitiveUsers() start() stop() suspend()
IUser	authenticate() authorize() getName() getPassword() getPropertiesSubset() isUserInRole() refresh()
IGroup	containsGroup() containsUser() getName() getPropertiesSubset() refresh()

IUserManager インターフェース

注: このセクションは DB2 Alphablox 8.4 だけに適用されます (8.4.1 ではサポートされません)。

AbstractUserManager クラスは、*IUserManager* インターフェースをインプリメントしたものであり、実行時にユーザー情報とグループ情報を検索して、ユーザーの身元を確認します。このクラスを拡張するには、以下のインポート・ステートメントをコードに追加します。

```
import com.alphablox.personalization.*;
```

AbstractUserManager クラスを拡張する例の詳細については、114 ページの『例 1: DB2 Alphablox で外部ユーザー・マネージャーを使用するための設定』を参照してください。

findGroup()

グループを検索して、該当する *IGroup* のインスタンスを戻します。

構文

```
IGroup findGroup(String id, boolean fromCache);
        // throws PEngineException
```

説明:

引数	説明
id	グループ ID
fromCache	キャッシュからグループを検索するかどうか

使用法

このメソッドは、以下の場合にヌルを戻します。

- *fromCache* を true に設定したときに、グループがメモリー内に存在しない場合
- *fromCache* を true に設定したときに、グループがメモリー内に存在せず、グループがユーザー・マネージャーの基本リポジトリー内の有効なグループではない場合

findUser()

ユーザーを検索して、該当する *IUser* のインスタンスを戻します。

構文

```
IUser findUser(String id, boolean fromCache);
        // throws PEngineException
```

説明:

引数	説明
id	ユーザー ID
fromCache	キャッシュからユーザーを検索するかどうか

使用法

このメソッドは、以下の場合にヌルを戻します。

- *fromCache* を true に設定したときに、ユーザーがメモリー内に存在しない場合
- *fromCache* を true に設定したときに、ユーザーがメモリー内に存在せず、ユーザーがユーザー・マネージャーの基本リポジトリー内の有効なユーザーではない場合

getExternalProperties()

外部エディターで定義されたプロパティーのストリング配列を戻します。

構文

```
String[] getExternalProperties();  //throws PEngineException
```

getPrincipleUserName()

この要求に関連したユーザー名を戻します。

構文

```
String getPrincipleUserName(HttpServletRequest request);
        // throws PEngineException
```

説明:

引数	説明
request	現在の HTTP 要求

使用法

DB2 Alphablox は、新規 DB2 Alphablox セッションの作成時にユーザー名を判別するためにこのメソッドを使用します。管理者は、このメソッドを使用して、DB2 Alphablox のデフォルトの動作をオーバーライドできます。

hasExternalEditor()

外部ユーザー・マネージャー・エディターがあるかどうかを戻します。

構文

```
boolean hasExternalEditor();
```

使用法

外部ユーザー・マネージャーが独自のエディターを持っている場合は、true を戻します。例えば、LDAP ベースのユーザー・マネージャーを使用している場合、このメソッドは、true を戻します。デフォルトの DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーの場合は、false を戻します。外部エディターを使用した拡張可能ユーザー・マネージャーの完全インプリメンテーションを対象にする場合以外は、true を戻す状況でこのメソッドをインプリメントしないでください。

resume()

ユーザー・マネージャーを再開します。

構文

```
void resume();    // throws PEngineExceptionn
```

使用法

DB2 Alphablox ユーザー・マネージャーが自身のサービスを再開するときに、このメソッドが呼び出されます。 DB2 Alphablox ユーザー・マネージャーのサービスの中断後にこのメソッドを呼び出すと、ユーザー・マネージャーが再開します。

例

以下のコードは、ユーザー・マネージャーが再開したことを示すメッセージをログに記録します。

```
import com.alphablox.personalization.*;

public class MyUserManager extends AbstractUserManager {
...
   public void resume() throws PEngineException {
      System.out.println("Resumed");
   }
...
}
```

参照箇所

121 ページの『suspend()』

setCaseSensitiveGroups()

グループ名で大/小文字を区別するかどうかを指定します。

構文

```
void setCaseSensitiveGroups(boolean caseSensitive);
        // throws PEngineException
```

説明:

引数	説明
caseSensitive	true - グループ名で大/小文字を区別します。

使用法

このメソッドを true に設定すると、グループ名で大/小文字が区別されることになり、ユーザー・マネージャーもその大/小文字を区別します。

setCaseSensitiveUsers()

ユーザー名で大/小文字を区別するかどうかを指定します。

構文

```
void setCaseSensitiveUsers(boolean caseSensitive);
        // throws PEngineException
```

説明:

引数	説明
caseSensitive	true - ユーザー名で大/小文字を区別します。

使用法

このメソッドを true に設定すると、ユーザー名で大/小文字が区別されることになり、ユーザー・マネージャーもその大/小文字を区別します。

start()

ユーザー・マネージャーを始動します。

構文

```
void start(java.util.Properties props);
        // throws PEngineException
```

説明:

引数	説明
props	パーソナライゼーションに関連した DB2 Alphablox プロパティーです。有効なプロパティーは、umclassname、groupclassname、custompropery、ldapprops です。詳しくは、110 ページの『拡張可能ユーザー・マネージャーの Telnet コンソール・コマンド』を参照してください。

使用法

このメソッドは、DB2 Alphablox ユーザー・マネージャーの毎回の始動時に呼び出されます。このメソッドの使用によっては、該当するリポジトリーとの接続が確立され、必要に応じてユーザーやグループのオブジェクトがインスタンス化されます。 DB2 Alphablox リポジトリー・ベースのユーザー・マネージャーの場合は、リポジトリー内のすべてのユーザーとグループがインスタンス化されます。 LDAP ベースのユーザー・マネージャーでは、DB2 Alphablox に登録されているユーザーとグループだけを作成する必要があります。このため、findUser() メソッドや findGroup() メソッドを呼び出す場合など、必要に応じてユーザーやグループを作成するほうが望ましいと言えます。

stop()

ユーザー・マネージャーを停止して、すべてのリソースを解放します。

構文

```
void stop();     // throws PEngineException
```

使用法

DB2 Alphablox ユーザー・マネージャーが自身のサービスを停止するときに、このメソッドが呼び出されます。

参照箇所

120 ページの『start()』、121 ページの『suspend()』

suspend()

外部ユーザー・マネージャーを中断して、データベース接続などの未使用リソースを解放します。

構文

```
void suspend();     // throws PEngineException
```

使用法

DB2 Alphablox ユーザー・マネージャーが自身のサービスを中断するときに、このメソッドが呼び出されます。

参照箇所

119 ページの『resume()』

IUser インターフェース

このクラスを拡張するには、以下のインポート・ステートメントをコードに追加します。

```
import com.alphablox.personalization.*;
```

AbstractUser クラスを拡張する例の詳細については、115 ページの『例 2: DB2 Alphablox で別のユーザー・クラスを使用するための設定』を参照してください。

authenticate()

HTTP 要求に指定されているユーザーとパスワードが、リポジトリーに保管されているユーザーとパスワードと同じかどうかをチェックします。

構文

```
boolean authenticate(HttpServletRequest request,
                     String authorizationHeader);
                   // throws PEngineException
```

説明:

引数	説明
request	HTTP 要求
authorizationHeader	キャッシュからグループを検索するかどうか

使用法

ユーザー保管のパスワードがパラメーターのパスワードと同じ場合は、true を戻します。このメソッドを適用できるのは、DB2 Alphablox Apache Tomcat 構成 (Apache Tomcat; Microsoft IIS なし) だけです。この場合、DB2 Alphablox は、基本認証 (ブラウザー・チャレンジの認証など) を使用します。 DB2 Alphablox Tomcat インターセプターは、ユーザーが初めてセッションにアクセスしたときに、このメソッドを呼び出します。このメソッドには、要求オブジェクトとエンコード許可ヘッダーという 2 つのパラメーターがあります。このヘッダーに、ユーザー名とパスワードが含まれています。そのヘッダーをデコードするには、以下のメソッドを使用します。

```
AbstractUserManager.getDecoder().decode(authorizationHeader);
```

パスワードを取得した後に、getPassword() メソッドの戻り値としてそのパスワードを使用して、データ・ソースに対するシングル・サインオンを許可する場合は、そのパスワードをオブジェクト・メモリーに保管することも可能です。

authorize()

指定した役割リストにユーザーが入っているかどうかをチェックします。

構文

```
boolean authorize(HttpServletRequest request,
                  String[] roles);
          // throws PEngineException
```

説明:

引数	説明
request	HTTP 要求
roles	役割リスト (ユーザーがいずれかの役割に入っているかどうかをチェックします)

使用法

指定した役割のいずれかにユーザーが入っている場合は、true を戻します。

getEmail()

ユーザーの E メールを取得します。

構文

```
String getEmail();    // throws PEngineException
```

使用法

DB2 Alphablox が E メール名を外部リポジトリーから読み取るのではなく、むしろ内部で保持している場合は、ヌルを戻します。 LDAP の場合、ユーザーの E メールとフルネームは LDAP から取り込まれ、DB2 Alphablox 管理ページでその値を編集することはできません。その場合に、このメソッドを使用すれば、ユーザーの E メールを取得できます。

getFullName()

ユーザーのフルネームを取得します。

構文

```
String getFullName();    // throws PEngineException
```

使用法

DB2 Alphablox がフルネームを外部リポジトリーから読み取るのではなく、むしろ内部で保持している場合は、ヌルを戻します。 LDAP の場合、ユーザーの E メールとフルネームは LDAP から取り込まれ、DB2 Alphablox 管理ページでその値を編集することはできません。その場合に、このメソッドを使用すれば、ユーザーのフルネームを取得できます。

getName()

ユーザー名を取得します。

構文

```
String getName();      // throws PEngineException
```

使用法

このユーザー名は、有効な DB2 Alphablox ユーザー名でなければなりません。

getPassword()

ユーザーのパスワードを取得します。

構文

```
String getPassword();     // throws PEngineException
```

使用法

ユーザーの base64 エンコード・パスワードです。データ・ソースが DB2 Alphablox のユーザー名とパスワードを使用する設定になっている場合、DB2 Alphablox は、リポジトリーに保管されているエンコード・パスワードを使用して、認証やデータ・ソースへのアクセスを行います。データ・ソースへのシングル・サインオンなどのカスタム・セキュリティーの場合は、このメソッドを使用します。

このメソッドで保管されたパスワードは、メモリー内で一時的に使用されます。このパスワードは、呼び出し側に戻す前に、以下のコードを使用してエンコードしなければならないことに注意してください。

```
AbstractUserManager.getEncoder().encode(password);
```

それから、呼び出し側に戻します。

参照箇所

114 ページの『シングル・サインオン』

getPropertiesSubset()

目的のユーザーだけに限定したプロパティーのサブセットを取得します。

構文

```
java.util.Properties getPropertiesSubset(String[] propList);
                // throws PEngineException
```

説明:

引数	説明
propList	プロパティー・サブセットのストリング配列

使用法

引数とした渡したユーザー・プロパティーのサブセットのプロパティー・オブジェクトを戻します。その値は、開発者が変更しなければ、メモリーから取り込まれると仮定されます。

isUserInRole()

指定した役割のいずれかにユーザーが属しているかどうかを識別します。

構文

```
boolean isUserInRole(HttpServletRequest request,
                     String[] roles);
            // throws PEngineException
```

説明:

引数	説明
request	HTTP 要求
roles	ユーザーが属する役割

使用法

ユーザーがリスト内の役割のいずれかに属している場合は、`true` を戻します。役割のインプリメンテーションは、開発者の仕事です。このメソッドのインプリメンテーションは、標準 J2EE API の `Request.isUserInRole(String`*role*`)` メソッドに基づいています。 DB2 Alphablox は、このメソッドを使用して、管理機能やブックマークへの書き込みに関するユーザー・アクセスを判別します。

LDAP ベースのユーザー・マネージャーを使用している場合、このメソッドは、IGroup.containsUser() メソッドを呼び出します。役割は、LDAP のグループに相当するからです。

参照箇所

125 ページの『containsGroup()』

refresh()

メモリー内にキャッシュされているすべての情報をリフレッシュして、基本リポジトリーから最新の情報を取得します。

構文

```
void refresh();       // throws PEngineException
```

使用法

メモリー内に保管されているユーザー情報を強制的にリフレッシュして、外部リポジトリーから最新の情報を取得します。

IGroup インターフェース

このクラスを拡張するには、以下のインポート・ステートメントをコードに追加します。

```
import com.alphablox.personalization.*;
```

AbstractGroup クラスを拡張する例の詳細については、116 ページの『例 3: DB2 Alphablox で別のグループ・クラスを使用するための設定』を参照してください。

containsGroup()

このグループにサブグループが含まれているかどうかをチェックします。

構文

```
boolean containsGroup(IGroup group, boolean checkSubGroups);
                  // throws PEngineException
```

説明:

引数	説明
group	IGroup のインスタンス
checkSubGroups	true - サブグループをチェックします。false-- サブグループをチェックしません。

使用法

このグループにサブグループが含まれている場合は、true を戻します。

containsUser()

このグループに指定のユーザーが含まれているかどうかをチェックします。

構文

```
boolean containsUser(IUser user, boolean checkSubGroups);
                  // throws PEngineException
```

説明:

引数	説明
user	IUser のインスタンス
checkSubGroups	true -- サブグループをチェックします。false -- サブグループをチェックしません。

使用法

このグループに指定のユーザーが含まれている場合は、true を戻します。

getName()

グループ名を取得します。

構文

```
String getName();      // throws PEngineException
```

使用法

このグループ名は、有効な DB2 Alphablox グループ名でなければなりません。

getPropertiesSubset()

目的のグループだけに限定したプロパティーのサブセットを取得します。

構文

```
java.util.Properties getPropertiesSubset(String[] propList);
                // throws PEngineException
```

説明:

引数	説明
propList	プロパティーを含んだストリング配列

使用法

少なくとも渡されたグループ・プロパティーのリストのための Properties オブジェクトを戻します。その値は、開発者が変更しなければ、メモリーから取り込まれると仮定されます。

refresh()

メモリー内にキャッシュされているすべての情報をリフレッシュして、基本リポジトリーから最新の情報を取得します。

構文

```
void refresh();       // throws PEngineException
```

使用法

メモリー内に保管されているグループ情報を強制的にリフレッシュして、外部リポジトリーから最新の情報を取得します。

第 16 章 データベース・リポジトリーの使用

この章では、DB2 Alphablox リポジトリーとしてリレーショナル・データベースを使用するための DB2 Alphablox の構成方法について解説します。

DB2 Alphablox リポジトリーの概要

DB2 Alphablox リポジトリーは、アプリケーション、ユーザー、グループ、ブックマークなどの情報を追跡管理するために DB2 Alphablox が使用するオブジェクトの格納場所です。このリポジトリーは、オペレーティング・システムのファイル・システムにも、リレーショナル・データベースにも配置できます。クラスター環境で DB2 Alphablox を実行するために構成をセットアップする場合は、リレーショナル・データベースにリポジトリーを配置する必要があります。リレーショナル・データベースの場合は、複数のサーバー・ノードがリポジトリーに対する読み書きを行っても、リポジトリー内のデータの整合状態を維持できます。

DB2 Alphablox 環境内のリポジトリー

DB2 Alphablox の内部では、リポジトリー・マネージャーがリポジトリーへのアクセスを制御します。リポジトリー・マネージャーは、リポジトリーがファイル・システムにある場合でも、リレーショナル・データベースにある場合でも、Java Naming and Directory Interface (JNDI) を使用してリポジトリーと通信します。JNDI 層では、リポジトリーのタイプ (DB2 Alphablox のファイル・システムなのか、データベースなのか) に応じて、それぞれ異なるサービス・プロバイダーを使用します。

リポジトリーに対応しているデータベースは、データ・ソース・アクセスに対応しているデータベースと同じです。詳細については、*インストール・ガイド*を参照してください。 DB2 Alphablox のアーキテクチャーの詳細については、9 ページの『DB2 Alphablox のアーキテクチャー』を参照してください。

リポジトリー変換ユーティリティーは、ファイル・システムとファイル・システムの間、ファイル・システムとデータベースの間、データベースとファイル・システムの間、データベースとデータベースの間でリポジトリーを変換するための Java プログラムです。この変換ユーティリティーは、インストール・プロセスの中で自動実行することも、手動で実行することもできます。この変換ユーティリティーの詳細については、130 ページの『リポジトリー変換ユーティリティーの使用』を参照してください。

リレーショナル・リポジトリーのメリット

DB2 Alphablox リポジトリーとしてリレーショナル・データベースを使用することには、2 つの大きなメリットがあります。

- リレーショナル・リポジトリーには複数のサーバーからアクセスできるので、DB2 Alphablox のクラスター環境を使用して実質的に無限の拡張性を実現できます。

- リレーショナル・リポジトリーの場合は、トランザクションによるデータ保全、バックアップと復元の操作、整合状態へのロールバック、データベースのレプリケーションなど、データベース環境ならではの強力なツール群を活用できます。

リポジトリーには、DB2 Alphablox の操作にとって非常に重要なオブジェクトを格納するので、データ保全のために用意されているデータベースのツール群をすべて活用して、システムの耐久性と信頼性を高めることができます。

DB2 Alphablox リポジトリーの構成

DB2 Alphablox リポジトリーの初期状態は、インストール・プロセスで確立されます。 DB2 Alphablox リポジトリーにアクセスするために、リポジトリーのタイプに応じて、**DB2 Alphablox ファイル・システム・サービス・プロバイダー**と **DB2 Alphablox データベース・サービス・プロバイダー**という 2 つのサービス・プロバイダー が用意されています。デフォルトの状態では、**DB2 Alphablox ファイル・システム・サービス・プロバイダー**を使用します。 **DB2 Alphablox データベース・サービス・プロバイダー**は、DB2 Alphablox に対応しているすべてのリレーショナル・データベースをサポートしています。インストール・プロセスでリポジトリーのタイプを指定する方法や、サポートされているリレーショナル・データベースの詳細については、 *インストール・ガイド*を参照してください。

リポジトリー・タイプの確認

サーバーでどちらのリポジトリー・サービス・プロバイダーが使用されているかを確認するには、以下の手順を実行します。

1. DB2 Alphablox が稼働中であることを確認します。
2. *admin* ユーザーまたは管理者グループのメンバーとして DB2 Alphablox ホーム・ページにログインします。
3. 「**管理**」タブをクリックします。「管理」タブの下に「**一般**」ページが表示されます。
4. そのページの「**一般プロパティー**」セクションで、**リポジトリー・マネージャー**のリンクをクリックして、DB2 Alphablox リポジトリーのプロパティーを表示します。

   リポジトリー・サービス・プロバイダーの項目には、どの JNDI サービス・プロバイダーが使用されているかが表示され、**データベース・アダプター**の項目には、どの JDBC ドライバーが使用されているかが表示され、その他の項目には、データ・ソースの構成情報が表示されます。ファイル・システムのサービス・プロバイダーが使用されている場合、**リポジトリー・ロケーション**の項目には、リポジトリー・ファイルの格納先のディレクトリーが表示されます。

注: リポジトリー・マネージャーのプロパティーのページでリポジトリーのタイプを変更することはできません。リポジトリーのタイプを変更するには、リポジトリー変換ユーティリティーを使用します。

リポジトリー変換ユーティリティーの使用

リポジトリーのタイプを (例えば、ファイル・システムから Oracle データベースに) 変更するには、リポジトリー変換ユーティリティーを実行する必要がありま

す。リポジトリー変換ユーティリティーは、コマンド行ウィンドウ (Windows システムの MS-DOS ウィンドウや、 Linux、 UNIX システムの xterm や他のコマンド・ウィンドウ) で実行する Java プログラムです。このユーティリティーは、DB2 Alphablox リポジトリーを 1 つの場所から別の場所に移動するために必要なテーブルやファイルを作成します。リポジトリー変換ユーティリティーを使用して、ファイル・システムからファイル・システムに、ファイル・システムからデータベースに、データベースからファイル・システムに、データベースからデータベースにリポジトリーを移動できます。

リポジトリー変換ユーティリティーの開始

リポジトリー変換ユーティリティーは、DB2 Alphablox をインストールした Windows システムの「**スタート**」メニューから開始することもできれば、 Windows システムで以下のファイルを実行することによって開始することもできます。

重要: リポジトリー変換ユーティリティーを実行する前に、必ず DB2 Alphablox をシャットダウンしてください。

```
<db2alphablox_dir>/Tools/convert/ConvertRepository.exe
```

Linux、UNIX システムでは、以下のファイルを実行することによって開始できます。

```
<db2alphablox_dir>/Tools/convert/ConvertRepository
```

`<db2alphablox_dir>` は、DB2 Alphablox のインストール先のディレクトリーです。

リポジトリー変換ユーティリティーの対話式コマンド行オプション

リポジトリー変換ユーティリティーのメインメニューの対話式コマンド行オプションを以下の表にまとめます。

オプション		説明
1	Set DB2 Alphablox File Manager Root	ファイル・システム・リポジトリーで使用するリポジトリー・ファイルの場所としてディレクトリーを設定します。このディレクトリーは、有効な DB2 Alphablox ディレクトリーでなければなりません。そうでない場合は、その操作はエラーと共に失敗します。
2	Set DB2 Alphablox Instance Name	リポジトリー変換ユーティリティーでアクセスする DB2 Alphablox のインスタンスを選択します。指定するインスタンスは、有効な DB2 Alphablox インスタンスでなければなりません。そうでない場合は、その操作はエラーと共に失敗します。

オプション		説明
3	Convert One Repository to Another	既存のリポジトリーを別のリポジトリーに変換し、すべての必要なデータを 1 つのリポジトリーから別のリポジトリーに移動します。ファイルからデータベース、データベースからファイル、ファイルからファイル、データベースからデータベースというリポジトリー変換オプションがあります。リポジトリーを変換するときには、以下のいずれかのオプションを選択するためのプロンプトが表示されます。 • COPY: 1 つのリポジトリーの内容を別のリポジトリーにコピーし、元のリポジトリーをそのまま残します。 • MOVE: 1 つのリポジトリーから別のリポジトリーに内容を移動します。元のリポジトリーは、削除します。 さらに、宛先のリポジトリーに関して、以下のいずれかのオプションを選択するためのプロンプトが表示されます。 • NEW: 宛先に新しいリポジトリーを作成します。 • UPDATE: 宛先のリポジトリーを新しいリポジトリーで置き換える代わりに、ソースのリポジトリーに基づいてデータを更新します。 • OVERWRITE: テーブルを再作成し、宛先のリポジトリーのすべてのデータをソースのリポジトリーの情報で置き換えます。 OVERWRITE オプションの場合、宛先のリポジトリーのデータはすべて消えます。
4	Create an Empty Database Repository	DB2 Alphablox リポジトリーに必要なテーブルをデータベース内に作成します。このオプションの場合は、初期内容の入ったテーブルを作成するだけで、テーブルにリポジトリー・オブジェクトを設定することはしません。
5	Verify and Repair a Repository	リポジトリーに問題がないかどうかを検査して、問題が見つかればそのレポートを生成します。ファイルのリポジトリーとデータベースのリポジトリーの両方の検査のためのオプションが含まれています。データベースのリポジトリーの場合は、このオプションによって一部の問題を訂正することもできます。
6	Change DB2 Alphablox to use a different Repository	DB2 Alphablox のインスタンスをリダイレクトして、別のリポジトリーを参照するようにします。変更先のリポジトリーは、実際に存在していなければならず、DB2 Alphablox マシンからアクセスできるものでなければなりません。

オプション	説明
7 Conversion Utility Options	このユーティリティーの詳細メッセージと通常メッセージを切り替えるオプションや、変換ユーティリティーが入力情報の記憶のために使用する履歴を格納したバッファーを消去するオプションがあります。リポジトリー・テーブルを作成するために別の DDL スキーマ・ファイルを指定するオプションもあります。別の DDL ファイルは、徹底的にテストした後でない限り指定しないでください。
8 Exit	リポジトリー変換ユーティリティーを終了します。

リポジトリー変換ユーティリティーは、変換ユーティリティー・セッションのすべてのアクティビティーを保管するために、repositoryconvert.log というログ・ファイルを保持します。 repositoryconvert.log ファイルは、変換ユーティリティーと同じディレクトリー (<db2alphablox_dir>/Tools/convert) にあります。

ファイル・システムからデータベースへの変換

ファイル・システムからデータベースにリポジトリーを変換する前に、以下の情報を収集してください。

- ファイル・システム・リポジトリーの絶対パス (d:¥alphablox¥repository など)
- データベースの接続情報

ファイル・システムからデータベースに DB2 Alphablox リポジトリーを変換するために必要な手順は、以下のとおりです。

1. DB2 Alphablox をシャットダウンします。
2. リポジトリー変換ユーティリティーを開始します (詳細については、131 ページの『リポジトリー変換ユーティリティーの開始』を参照してください)。
3. オプション 3 **Convert one repository to another** を選択するために、**3** を入力して、Enter キーを押します。
4. オプション 1 **Convert file to database** を選択するために、**1** を入力して、Enter キーを押します。
5. リポジトリーのルート・ディレクトリーを確認して、Enter キーを押します。

   注: デフォルトとして表示されているディレクトリーがリポジトリーのディレクトリーでない場合は、DB2 Alphablox の正しいインスタンスにアクセスしているかどうかを確認してください。リポジトリー変換ユーティリティーがアクセスするデフォルトのインスタンスは、インストールされているインスタンスであり、デフォルトで AlphabloxAnaltyics という名前になります。別のインスタンス名になっている場合は、現在のシーケンスをいったん終了して、メインメニューのオプション 2 から正しいインスタンス名を設定してください。
6. すべてが正しければ、**Continue** を選択するために、**1** を入力して、Enter キーを押します。

7. ご使用のデータベース・サーバーに対応するデータベースを選択します。例えば、Oracle 8.1.7 を使用している場合は、**2** を入力します。
8. プロンプトに従って、データベースの構成情報を入力します。
9. すべてが正しければ、**Continue** を選択するために、**1** を入力して、Enter キーを押します。
10. COPY か MOVE を入力します。 COPY の場合は、古いリポジトリーをそのまま残して、そのコピーを宛先のリポジトリーに作成します。MOVE の場合は、古いリポジトリーを削除して、新しいリポジトリーを宛先のリポジトリーに作成します。
11. NEW か UPDATE か OVERWRITE を入力します。既存のリポジトリーが存在せず、新しいテーブルの入った新しいリポジトリーを作成する場合は、NEW を使用します。リポジトリーの既存のテーブル構造をそのまま残し、ソースのリポジトリーのデータを組み込むことによってリポジトリーを更新する場合は、UPDATE を使用します。古いデータとデータベース・テーブルをすべて削除し、新しいテーブルとデータを再作成する場合は、OVERWRITE を使用します。
12. DB2 Alphablox インスタンスで新しいリポジトリーを使用する場合は、**Update DB2 Alphablox to use the New Repository** プロンプトに **Y** を入力します。
13. DB2 Alphablox インスタンスのすべての必要なプロパティーを更新する場合は、**Update DB2 Alphablox Properties** プロンプトで ALL を選択します。ALL の場合は、すべてのサーバー・プロパティーを変換します。SPECIFIC の場合は、ローカル・マシンに固有のサーバー・プロパティーだけを変換します (クラスターのプロパティーは変換しません)。GLOBAL の場合は、クラスター内で共有されているプロパティーだけを変換します (ローカル・マシンの項目は変換しません)。 NONE の場合は、DB2 Alphablox インスタンスのプロパティーを変更しません。
14. すべてが正しければ、**Continue** を選択するために、**1** を入力して、Enter キーを押します。
15. 変換が終了すると、メインメニューが再び表示されます。 **8** を入力して、リポジトリー変換ユーティリティーを終了します。

データベースからファイル・システムへの変換

データベースからファイル・システムにリポジトリーを変換する前に、以下の情報を収集してください。

- データベース・リポジトリーの接続情報
- ファイル・システム・リポジトリーの配置先の絶対パス
  (`d:¥alphablox¥Repository` など)

データベースからファイル・システムに DB2 Alphablox リポジトリーを変換するために必要な手順は、以下のとおりです。

1. DB2 Alphablox をシャットダウンします。
2. リポジトリー変換ユーティリティーを開始します (詳細については、131 ページの『リポジトリー変換ユーティリティーの開始』を参照してください)。
3. オプション 3 **Convert one repository to another** を選択するために、**3** を入力して、Enter キーを押します。

4. オプション 2 **Convert database to file** を選択するために、**2** を入力して、Enter キーを押します。
5. ご使用のデータベース・サーバーに対応するデータベースを選択します。例えば、Oracle 8.1.7 を使用している場合は、**2** を入力します。
6. プロンプトに従って、データベースの構成情報を入力します。
7. すべてが正しければ、**Continue** を選択するために、**1** を入力して、Enter キーを押します。

   リポジトリー変換ユーティリティーがデータベースへの接続を試みます。
8. 宛先のリポジトリー・ルートのディレクトリーを入力して、Enter キーを押します。
9. すべてが正しければ、**Continue** を選択するために、**1** を入力して、Enter キーを押します。
10. COPY か MOVE を入力します。 COPY の場合は、古いリポジトリーをそのまま残して、そのコピーを宛先のリポジトリーに作成します。MOVE の場合は、古いリポジトリーを削除して、新しいリポジトリーを宛先のリポジトリーに作成します。
11. NEW か UPDATE か OVERWRITE を入力します。既存のリポジトリーが存在せず、新しいリポジトリーを作成する場合は、NEW を使用します。リポジトリーの既存の構造をそのまま残し、ソースのリポジトリーのデータによってリポジトリーを更新する場合は、UPDATE を使用します。古いデータと構造をすべて削除し、新しい構造とデータを再作成する場合は、OVERWRITE を使用します。
12. DB2 Alphablox インスタンスで新しいリポジトリーを使用する場合は、**Update DB2 Alphablox to use the New Repository** プロンプトに **Y** を入力します。
13. DB2 Alphablox インスタンスのすべての必要なプロパティーを更新する場合は、**Update DB2 Alphablox Properties** プロンプトで ALL を選択します。ALL の場合は、すべてのサーバー・プロパティーを変換します。SPECIFIC の場合は、ローカル・マシンに固有のサーバー・プロパティーだけを変換します (クラスターのプロパティーは変換しません)。GLOBAL の場合は、クラスター内で共有されているプロパティーだけを変換します (ローカル・マシンの項目は変換しません)。 NONE の場合は、DB2 Alphablox インスタンスのプロパティーを変更しません。
14. すべてが正しければ、**Continue** を選択するために、**1** を入力して、Enter キーを押します。
15. 変換が終了すると、メインメニューが再び表示されます。 **8** を入力して、リポジトリー変換ユーティリティーを終了します。

インスタンスで既存のリポジトリーを使用するための構成

DB2 Alphablox インスタンスで既存のリポジトリーを使用するための構成を行う前に、接続先のデータベース・リポジトリーの接続情報を収集してください。

DB2 Alphablox インスタンスで既存の DB2 Alphablox リポジトリーを使用するための構成に必要な手順は、以下のとおりです。

1. DB2 Alphablox をシャットダウンします。

2. リポジトリー変換ユーティリティーを開始します (詳細については、131 ページの『リポジトリー変換ユーティリティーの開始』を参照してください)。
3. オプション 6 **Change DB2 Alphablox to use a different repository** を選択するために、**6** を入力して、Enter キーを押します。
4. オプション 2 **Database repository is the target** を選択するために、**2** を入力して、Enter キーを押します。
5. ご使用のデータベース・サーバーに対応するデータベースを選択します。例えば、Oracle 8.1.7 を使用している場合は、**2** を入力します。
6. プロンプトに従って、データベースの構成情報を入力します。
7. すべてが正しければ、**Continue** を選択するために、**1** を入力して、Enter キーを押します。

   リポジトリー変換ユーティリティーがデータベースへの接続を試み、指定のリポジトリーを使用するように DB2 Alphablox インスタンスを更新します。
8. **8** を入力して、リポジトリー変換ユーティリティーを終了します。

コマンド行構文

ほとんどの場合は、これまでの説明のとおりに対話モードでリポジトリー変換ユーティリティーを使用できます。ただし、コマンド行オプションで別のデータベース DDL ファイルを指定したり、自動スクリプトの一部として変換ユーティリティーを実行したりすることも可能です。リポジトリー・ユーティリティーの基本的な構文は、以下のとおりです。

```
java -cp [class_path] com.alphablox.util.convert.Convert operation
     destination [source] [arguments]
```

説明:

class_path
: DB2 Alphablox インスタンスの Java クラスパスです。 *class_path* の例については、DB2 Alphablox の始動ファイル (例えば、Windows プラットフォームの場合は `AnalysisServer.bat`、 Linux、UNIX プラットフォームの場合は `AnalysisServer.sh`) をご覧ください。

operation
: 実行するリポジトリー変換ユーティリティーの操作です。それぞれの使用可能な操作の説明については、137 ページの『操作の説明』を参照してください。

destination
: 宛先のリポジトリーを記述したプロパティー・ファイルの相対パスまたは絶対パスです。宛先のサンプル・ファイルについては、138 ページの『ソースと宛先のサンプル・プロパティー・ファイル』を参照してください。

source
: ソースのリポジトリーを記述したプロパティー・ファイルの相対パスまたは絶対パスです。 *source* 引数の指定は、COPY 操作と MOVE 操作の場合に必須になります。宛先のサンプル・ファイルについて

は、138 ページの『ソースと宛先のサンプル・プロパティー・ファイル』を参照してください。

arguments 137 ページの『引数』の表にある 1 つ以上の引数です。

operation、*destination*、*source*、*arguments* のいずれかを指定しない場合は、リポジトリー変換ユーティリティーの実行が対話モードになります。

操作の説明: リポジトリー変換ユーティリティーで使用できる操作を以下の表にまとめます。

操作	説明
HELP	コマンド行ヘルプを表示します。
CHANGE	アクティブなリポジトリーを宛先のリポジトリーに変更します。
COPY	ソースのリポジトリーをそのまま残して、そのコピーを宛先のリポジトリーに作成します。
DELETE	宛先のリポジトリーを永久的に削除します。
MOVE	ソースのリポジトリーを削除して、新しいリポジトリーを宛先のリポジトリーに作成します。
NEW | CREATE	宛先に新しいリポジトリーを作成して、デフォルトのリポジトリー値を設定します。
VERIFY	宛先のリポジトリーで検査操作を実行して、結果のレポートを生成します。

引数: リポジトリー変換ユーティリティーの引数を以下の表にまとめます。

引数	説明
DEBUG	問題の診断に役立つ追加のデバッグ情報を出力します。
LOG:*file*	変換ユーティリティーのすべてのアクティビティーを記録するログ・ファイルの名前を指定します。*file* のデフォルト名は、*repositoryconvert.log* です。
SERVER:*InstanceName*	DB2 Alphablox インスタンスのインスタンス名を指定します。 SERVER 引数は必須です。
OVERWRITE	NEW、MOVE、COPY、CHANGE のいずれかの操作で使用します。OVERWRITE 引数を指定する場合は、宛先のリポジトリーの古いデータと構造をすべて削除し、新しい構造とデータを再作成します。この引数を指定しない場合、リポジトリー変換ユーティリティーは、宛先に既存のリポジトリーを検出した時点で停止します。
PROPS:*option*	SERVER:*InstanceName* 引数と一緒に使用して、宛先のリポジトリーでどのサーバー・プロパティーを更新するかを指定します。 *option* の値は、以下のとおりです。

- ALL: すべてのサーバー・プロパティーを変換します。
- GLOBAL: クラスター内で共有されているプロパティーだけを変換します (ローカル・マシンの項目は変換しません)。
- SPECIFIC: ローカル・マシンに固有のサーバー・プロパティーだけを変換します (クラスターのプロパティーは変換しません)。

UPDATE
MOVE 操作または COPY 操作で、宛先のリポジトリーの内容を置き換えるのではなく、ソースのリポジトリーの情報によって宛先のリポジトリーを更新することを指定します。

DDL:*file*
データベース・リポジトリーのテーブル、索引、初期内容を作成するためのデフォルトの DDL スキーマ・ファイルをオーバーライドします。

重要: この引数の使用には十分な注意が必要です。

USEDEST
指定した場合は、サーバー・プロパティーを変更して、宛先のリポジトリーを使用します。指定しない場合は、サーバーのリポジトリー・プロパティーを変更しません。

ソースと宛先のサンプル・プロパティー・ファイル: ここでは、リポジトリー変換ユーティリティーで使用するソースと宛先のサンプル・ファイルの内容を示します。プロパティー・ファイルでは、リポジトリーのタイプ、リポジトリーの接続情報、DB2 Alphablox を実行するコンピューター上のリポジトリー・プロパティー・ファイルの場所、DDL ファイルの名前を指定します。

oracle817 というサーバーの Oracle データベースに配置するリポジトリーのプロパティー・ファイルのサンプルを以下に示します。

```
RepositoryTarget=JDBCTarget
java.naming.factory.initial=com.alphablox.jndisp.AlphabloxContextFactory
java.naming.provider.url.server=oracle817
java.naming.provider.url.port=1521
java.naming.provider.url.sid=orcl817
database_driver=oracle.jdbc.driver.OracleDriver
fileroot=C:¥alphablox¥analytics¥repository¥servers¥
commandfile=oracle.dmlsql
user=user
password=password
```

ファイル・システムに配置するリポジトリーのプロパティー・ファイルのサンプルを以下に示します。

```
RepositoryTarget=ABXTarget
java.naming.factory.initial=com.alphablox.jndisp.AlphabloxContextFactory
java.naming.provider.url=C:¥alphablox¥analytics¥repository¥
fileroot=C:¥alphablox¥analytics¥repository¥servers¥
```

第 17 章 接続プールの使用

この章では、マルチディメンション・データ・ソースとリレーショナル・データ・ソースで接続プールを使用するための DB2 Alphablox の構成方法について解説します。

接続プール - 概要

Web ベース・アプリケーションは、マルチディメンション・データベースやリレーショナル・データベースと対話する必要があるときに、まずデータベースに接続しなければなりません。接続を確立し、維持し、不要になったときに解放するには、リソースを必要とするので、毎回の接続にはオーバーヘッドが伴います。 Web ベース・アプリケーションの場合、データベースとのユーザー対話は基本的に短く、多くの場合、要求そのものの処理に必要な時間よりも、データベースとの接続や切断に要する時間のほうが長くなってしまいます。

このようなデータベース対話の処理効率を高めるために、アプリケーション・サーバーやデータベースには通常、接続プールという機能が用意されています。接続プールとは、事前に確立されたデータベース接続のグループであり、このグループをアプリケーション間で共有します。接続プールを使用すれば、必要なデータベース対話を実現できるばかりか、毎回の接続にそれほど多くの時間とリソースを費やさずにすみます。

接続プールが使用可能であり、正しく構成されていれば、DB2 Alphablox でも接続プールを活用してアプリケーションのパフォーマンスを改善できます。 DB2 Alphablox で接続プールを使用する場合は、アプリケーション・サーバー (IBM WebSphere や BEA WebLogic) のリレーショナル・データベース接続プールを使用するか、 IBM DB2 OLAP Server の展開サービスまたは Hyperion Essbase の展開サービスの接続プール・サポートを使用できます。

MDB の接続プール

マルチディメンション・データベースの接続プールは、リレーショナル・データ・ソースだけに対応している標準の JDBC 接続の場合には使用できません。 DB2 OLAP Server または Hyperion Essbase を使用する場合、 DB2 Alphablox では、IBM DB2 OLAP Server の展開サービスまたは Hyperion Essbase の展開サービスに用意されている接続プールのインプリメンテーションを使用できます。

DB2 OLAP Server と Hyperion Essbase の接続プール

IBM DB2 OLAP Server の展開サービスまたは Hyperion Essbase の展開サービスについては、接続プールをサポートするための構成を行えます。それぞれの接続プールの定義では、DB2 OLAP Server または Essbase で使用するユーザー名とパスワードを指定します。接続プールのアクセスは、すべてのユーザーを許可するためのオプションによってすべてのユーザーに対して定義することも、指定したユーザーとグループのリストに対して定義することもできます。 DB2 Alphablox データ・ソー

スで IBM DB2 OLAP Server の展開サービスまたは Hyperion Essbase の展開サービスのアダプターを指定し、IBM DB2 OLAP Server の展開サービスまたは Hyperion Essbase の展開サービスで接続プールを正しく構成すれば、DB2 Alphablox でそれらの接続プールを使用できるようになります。 Essbase の接続プールを構成するための詳細については、 IBM DB2 OLAP Server の展開サービスまたは Hyperion Essbase の展開サービスの資料を参照してください。

Microsoft Analysis Services および接続プール

Microsoft Analysis のインプリメンテーションで接続プールを使用すると、分析アプリケーションのパフォーマンスの向上に効果があります。デフォルトでは、接続プールは使用不可になっています。

接続プールを使用可能にしているときに、DataBlox *disconnect()* メソッドが呼び出されると、DataBlox は切断されますが、ADO 接続はクローズされず、再利用できるように接続プールに置かれます。それ以降の DataBlox 接続で、プール内にあるものと一致する接続が要求されると、その接続はプールから除去されて DataBlox に与えられます。そうでない場合、新規の接続が作成されます。プールされた接続は、データ・ソースのユーザー名、パスワード、プロバイダー・ストリング、カタログ (Microsoft Analysis Services データベース名)、およびスキーマ (Microsoft Analysis Services では使用されないため、通常はヌルになっている) に対してキーが付けられます。 DataBlox コンポーネントがプール内の接続の 1 つを使用するためには、5 つのプロパティーがすべて一致している必要があります。異なるデータ・ソースでプロバイダー・ストリングを変更すると、プール内で異なる固有の接続になります。固有の接続のそれぞれが、基本的に接続プール・キーです。例えば、user1 として database1 への 2 つの接続があり、これらがデフォルトのプロバイダー・ストリング "MSOLAP" を使用します。また、同じユーザーおよびデータベースとしてもう 1 つの接続があり、それがプロバイダー・ストリング *"MSOLAP;Default Isolation Mode=1;Execution Location=3;Client Cache Size=0;"* を使用するとします。その場合、2 つの異なる接続プール・キーを使用する 3 つの接続があることになります。

接続プールの使用可能化

MSOLAP 接続プールは、デフォルトでは使用不可になっています。これを使用可能にするには、MSOLAPEnableConnectionPool サーバー・プロパティーを true に設定する必要があります。

接続プールを使用可能にするには、以下のようにします。

1. Telnet コンソールを開き DB2 Alphablox サーバーに接続します (またはDB2 Alphablox 管理ページで使用可能な「管理コンソール (Admin Console)」ウィンドウを使用します)。
2. "*set MSOLAPEnableConnectionPool true*" と入力し、MSOLAPEnableConnectionPool サーバー・プロパティーを設定して、Enter キーを押します。 "Server property 'Enable pooling of the MSOLAP connection pool' set to true." というメッセージが表示されます。
3. "save" と入力して、Enter キーを押します。 "Server properties saved" メッセージが表示されます。

注: プロパティーを保管しない場合、サーバー・プロパティーは現行サーバー・セッション中のみ適用されます。

接続プールを使用不可にする手順は、上記のステップの繰り返しになりますが、MSOLAPEnableConnectionPool プロパティーは false に設定してください。

接続プールの使用

DB2 Alphablox Microsoft Analysis Services 接続プールを使用する際に、すべての DataBlox autoDisconnect プロパティーが true に設定されていることを確かめる必要があります。そうしない場合、ユーザー・セッションが終了するまで、接続は使用可能なプールに戻りません。また、clearClientCache() メソッドは、接続プールを使用しない場合とは異なる仕方で動作します。接続プールが使用可能になっているときに clearClientCache() メソッドが呼び出される場合、使用可能なプールから、接続プール・キーが DataBlox 接続と一致するすべての接続が消去されます。さらに、ODBOBridgeJavaAPI.clearPool() メソッドを使用して、接続プール全体を消去することもできます。このメソッドでは、使用可能なプールにあるすべての ADO 接続が消去されます。

接続プールの制約

並行性が非常に高く、固有のプール接続 (ユーザー名) がほとんどない環境では、プール・サイズを制限することにより、メモリー消費量とパフォーマンスの両方を改善できます。 DB2 Alphablox が接続プール・キーごとに開いていたアクティブな MSOLAP 接続の数を制限する MSOLAPConnectionPoolLimit サーバー・プロパティーが追加されました。これには、プール内にある接続が含まれます。この機能は慎重に使用する必要があります。これは、ここで説明される固有の環境での使用だけを目的としています。 MSOLAPConnectionPoolLimit プロパティーは、接続プール・キーごとに接続の数を制限します。 MSOLAPConnectionPoolLimit サーバー・プロパティーを使用して、接続の最大数を設定してください。

例えば、MSOLAPConnectionPoolLimit を 4 に設定すると、接続プール・キーごとに開く MSOLAP 接続の最大数は 4 に設定されます。この例では、MSAS への接続がユーザー名プロパティー以外は同じであり、3 つのユーザー名 (Exec、Manager、および Admin) のみが接続していることを想定しています。 DataBlox コンポーネントがユーザー ID Admin を使用して接続しようとしており、これが 5 番目の接続である場合、そのユーザー名の接続が解放されて (または閉じられて) プールに戻されるまで待機し、開いている接続の数が制限を下回るようにする必要があります。

接続プールの調整

DB2 Alphablox Microsoft Analysis Services 接続プールを使用すると、メモリー使用量とパフォーマンスの両方を最適化するようにプールを調整できます。考慮すべきいくつかの要因があります。

Microsoft Analysis Services クライアント・サイド・キャッシュ: 接続について一般に認識されていない概念の 1 つに、Microsoft Analysis Services クライアント・サイド・キャッシュの影響があります。 MSAS は、DB2 Alphablox ホスト上で多くのものをキャッシュに入れることができます。例えば、算出されるメジャーは常に、Alphablox ホスト上の ADO 層で計算されてキャッシュに入れられます。情報が Alphablox セッションの間で絶えず再利用される場合、その情報を含んでいる ADO 接続を絶えず開いておくと効果的です。

接続プールの考慮事項: 接続プールは、接続に使用される接続プール・キーの数が限定されている場合 (少数の特殊ユーザー名) や、DB2 Alphablox セッションに、同じ接続プール・キーを使用する複数の Blox 接続がある場合に役立ちます。

(固有の接続プール・キーを必要としている) すべてのユーザーが固有であり、接続プール・キーごとに単一の DataBlox だけを持っているか、または多数の接続プール・キーを持っており、ユーザー・セッションの終了時にそれらを消去できない場合には、接続プールを使用しないでください。

autoDisconnect プロパティーの使用: 接続プールを使用するときには、たいていの場合 DataBlox autoDisconnect プロパティーを使用する必要があります。そうしないと、ADO 接続は再利用のためにプールに戻されません。

clearClientCache() メソッドの使用: 多数の接続プール・キーを使用している (つまり、Microsoft Analysis Services へのすべての接続が固有の名前を持つユーザーである) 場合、ユーザーがセッションを処理しているときに、固有の接続プール・キーを持つすべての DataBlox コンポーネントに対して DataBlox clearClientCache() メソッドを呼び出すことができます。これは一般に、ユーザーが明示的にログアウトするか、またはブラウザー・ウィンドウを閉じるときに呼び出されるログアウト・ページを使用して行われます。

clearPool() メソッドの使用: 多くの場合、接続プールを定期的に消去するのが最善です。ほとんどの環境では、プールを毎晩消去する単純なタイマー・サーブレットが適しています。

接続プールのモニターおよび管理: 以下の静的メソッドは、接続プールのモニターおよび管理に使用するために、ODBOBridgeJava クラスで使用可能になっています。

ODBOBridgeJavaAPI.poolSize()
使用可能なプールにある接続の数を戻します。

ODBOBridgeJavaAPI.clearPool()
使用可能なプールにある接続を切断します。

ODBOBridgeJavaAPI.getMaximumNumberOfConnections()
開くことができる MSAS 接続の最大数を取得します。

ODBOBridgeJavaAPI. getNumberOfOpenConnections()
開いている MSAS 接続の現行数を取得します。

開いている ADO 接続の数は、使用可能なプールにある接続の数と使用中の接続の数を加えたものに等しくなります。

以下の JSP スクリプトレットの例では、上述のメソッドを使用して、接続プールを管理およびモニターします。

```
<% ODBOBridgeJavaAPI.setMaximumNumberOfConnections(15);
  int count = ODBOBridgeJavaAPI.poolSize();
  out.write("Number of connections in the pool: "+count+"<br>");
  count = ODBOBridgeJavaAPI.getNumberOfOpenConnections();
  out.write("Number of connections active: "+count+"<br>");
  count = ODBOBridgeJavaAPI.getMaximumNumberOfConnections();
  out.write("Connection limit: "+count+"<br>");
%>
```

RDB の接続プール

RDB の接続プールは、リレーショナル・データベースに対する同一の JDBC 接続の名前付きのグループであり、接続プールをアプリケーション・サーバーに登録した時点で作成されます。 Web ベース・アプリケーションは、このプールから接続を「借りて」、データベース対話にその接続を使用してから、その接続を閉じてプールに接続を戻します。

接続プールを使用するメリットを以下の表にまとめます。

メリット	説明
応答時間の短縮	リソースは、データベースにアクセスするたびに、そのデータベースに接続する必要があります。接続プールを開始すると、接続プールによって指定の数の物理データベース接続が作成されるので、各アプリケーションのデータベース接続を作成するためのオーバーヘッドが無くなります。接続プールを使用すれば、接続の作成、維持、解放にかかわるオーバーヘッドが無くなるので、結果セットの取り出しにかかる応答時間が大幅に短縮されます。
基盤となるデータベースの抽象化	接続プールは、基盤となるデータベースに対する抽象層としての役割を果たすので、ベンダー固有の SQL 例外を処理する必要がなくなり、アプリケーション・データベースの切り替えが容易になります。この場合に処理しなければならないのは、アプリケーション・サーバーからの接続プール例外だけです。
並行接続の管理	ライセンスの制約によってデータベース接続の数が限られている場合は、並行データベース接続の数を管理する目的で接続プールを利用できます。

DB2 Alphablox による RDB 接続プールの使用

WebSphere や WebLogic のアプリケーション・サーバーと一緒に DB2 Alphablox を使用する場合は、接続プールを以下の機能から使用できます。

- DB2 Alphablox データ・ソース
- データベースを基盤とする DB2 Alphablox リポジトリー
- RDB キューブ
- ReportBlox
- CommentsBlox
- JDBC 接続 Bean

接続プールが使用可能で、正しく構成されている場合、上記の DB2 Alphablox 機能のうち、DB2 Alphablox データ・ソースとリレーショナル・データベースを基盤とする DB2 Alphablox リポジトリー以外はすべて、RDB 接続プールを自動的に活用します。

DB2 Alphablox データ・ソースと RDB 接続プール

WebSphere や WebLogic のアプリケーション・サーバーと一緒に DB2 Alphablox を使用する場合、RDB 接続プールを活用するには、DB2 Alphablox のデータ・ソース定義でアプリケーション・サーバー・データ・ソース・アダプターのオプションを選択する必要があります。このアダプターのオプションは、DB2 Alphablox で、

対応する WebSphere と WebLogic のアプリケーション・サーバーを使用するための構成を行っている場合にのみ表示されます。

注: DB2 Alphablox で、WebSphere や WebLogic の接続プールを使用してデータ・ソースにアクセスする場合は、それらのアプリケーション・サーバーでサポートされているすべてのリレーショナル・データベースに対応できます。

注: DB2 Alphablox の Apache Tomcat 構成は、RDB 接続プールをサポートしていません。

DB2 Alphablox リポジトリーと RDB 接続プール

リレーショナル・データベースを基盤とする DB2 Alphablox リポジトリーで RDB 接続プールを使用可能または使用不可にするには、以下の手順を実行します。

使用可能にする手順

1. アプリケーション・サーバー上で定義されている接続プールのための DB2 Alphablox データ・ソースを作成します。詳細については、上記の指示を参照してください。
2. リポジトリー変換ユーティリティーを実行します。

   Windows:

   ```
   <serverDirectory>/tools/convert/ConvertRepository.exe
   ```

   Linux と UNIX:

   ```
   <serverDirectory>/tools/convert/ConvertRepository.bat
   ```
3. DB2 Alphablox で WebSphere または WebLogic を使用している場合は、オプション 8 (“Configure Web Application Server Connection Pooling”) が表示されます。このオプションを選択して、Enter を押します。
4. オプション 1 (“Turn On connection pooling”) を選択して、Enter を押します。
5. アプリケーション・サーバーで定義されているデータ・ソースの JNDI 名を入力します。

使用不可にする手順

1. リポジトリー変換ユーティリティーを実行します。
2. DB2 Alphablox で WebSphere または WebLogic を実行している場合は、オプション 8 (“Configure Web Application Server Connection Pooling”) が表示されます。このオプションを選択して、Enter を押します。
3. オプション 1 (“Turn Off connection pooling”) を選択します。

BEA WebLogic の接続プールの構成

リレーショナル・データ・ソースに対して BEA WebLogic の接続プールを使用する場合は、定義されている接続プールにアクセスするために、BEA WebLogic で各データ・ソースの WebLogic ユーザーを作成する必要があります。

DB2 Alphablox で WebLogic の接続プールを使用するための構成を正しく行うには、DB2 Alphablox データ・ソース定義のデフォルト・ユーザー名とデフォルト・

パスワードを WebLogic ユーザーで使用しなければなりません。接続プールを使用しない場合は、リレーショナル・データ・ソースにアクセスするために WebLogic ユーザーを作成する必要はありません。

第 18 章 クラスター環境の使用

この章では、分析アプリケーションの拡張性を高めるために、クラスター環境で DB2 Alphablox を構成して使用する方法を解説します。ここでは、2 つのクラスター環境を取り上げます。つまり、 WebSphere のクラスター環境と WebLogic のスタンドアロン・クラスター環境です。

クラスター環境の概要

拡張性や可用性を高めるために、複数のサーバーからなるクラスター環境で DB2 Alphablox を実行することもできます。クラスター環境内のサーバーの数を増やせば、システムでそれだけ多くのユーザーをサポートできます。クラスターには主に 2 つの機能があります。

- **拡張性:** クラスターの容量は、1 台のマシンに限定されません。クラスターに新しいサーバーを追加して、容量を動的に拡張できます。
- **高可用性:** クラスターでは、複数のサーバーを使用して冗長性を確保できるので、障害が発生してもクライアントに影響を与えずにすみます。

注: クラスター環境で DB2 Alphablox を実行するために構成をセットアップする場合は、リレーショナル・データベースにリポジトリーを配置する必要があります。 DB2 Alphablox リポジトリーとリレーショナル・データベースによるリポジトリー構成の詳細については、129 ページの『DB2 Alphablox リポジトリーの概要』を参照してください。

WebSphere のクラスター環境

WebSphere のクラスター環境で DB2 Alphablox を構成してインストールするための詳細については、 *インストール・ガイド*の『*WebSphere クラスター環境での DB2 Alphablox の使用*』を参照してください。 WebSphere のクラスターを構成して使用するための詳細については、 WebSphere Server の資料を参照してください。

WebLogic のクラスター環境

WebLogic のクラスターは、連動する WebLogic サーバーのグループであり、1 台のサーバーだけの環境では得られない拡張性と信頼性を実現できます。クラスターは、クライアントには 1 台のサーバーのように見えますが、実際には複数のサーバーのグループです。このような WebLogic クラスターは、同じ物理マシン上の複数インスタンスの場合もあれば、複数の物理マシンにインストールした複数の WebLogic サーバーの場合もあります。

各 DB2 Alphablox ホストは、クラスター内の他のすべてのノードと通信し、リポジトリー内に何かの変更があれば、互いにそのことを通知し合います。リポジトリーの変更 (新規ユーザーなど) はリポジトリー内に直接格納され、リポジトリーの読み取りや変更は、各ノードから実行できます。

WebLogic のクラスター環境での DB2 Alphablox の構成とインストール

WebLogic のクラスター環境で DB2 Alphablox を構成してインストールするための詳細については、*インストール・ガイド*の『*WebLogic クラスター環境での DB2 Alphablox の使用*』を参照してください。 WebLogic のクラスターを構成して使用するための詳細については、WebLogic Server の資料を参照してください。

WebLogic のクラスター環境での新規アプリケーションの作成

WebLogic のクラスター環境で新しい DB2 Alphablox アプリケーションを定義する方法については、38 ページの『WebLogic クラスター使用時のアプリケーションの定義』を参照してください。

WebLogic の垂直クラスターの使用

垂直クラスター構成を使用すれば、DB2 Alphablox アプリケーションと Microsoft Analysis Services を併用する環境の拡張性を高めたり、効率の向上とコストの削減を目指して 1 台のマシンを複数のサーバー・インスタンスのホストとして設定したりできます。

Microsoft Analysis Services のユーザーは、DB2 Alphablox アプリケーションで WebLogic の垂直クラスターを使用し、1 台のマシンで複数の DB2 Alphablox インスタンスを実行することによって、Windows オペレーティング・システム上の Microsoft Analysis Services 2000 プロセスに関する 2 GB という制限を回避できます。その結果、Microsoft Analysis Services の拡張性が高くなります。

IT 投資を有効活用し、複数マシンの使用に伴う保守関連の問題を減らすためにも、1 台の強力なマシンで垂直クラスターを使用できます。

WebLogic のクラスター環境で DB2 Alphablox を構成してインストールするための詳細については、*インストール・ガイド*の『*WebLogic クラスター環境での DB2 Alphablox の使用*』を参照してください。 WebLogic のクラスターを構成して使用するための詳細については、WebLogic Server の資料を参照してください。

クラスターに関するコンソール・コマンド

「**クラスター・オプション (Cluster Options)**」ページに情報を入力する作業には、管理ユーザー・インターフェースだけでなく、DB2 Alphablox コンソールも使用できます。クラスター・プロパティーの現在の設定を表示するには、DB2 Alphablox コンソール・ウィンドウに以下のコマンドを入力します。

```
get service cluster
```

コンソールに対する画面出力は、以下のようになります。

```
get service cluster
Service Cluster Manager Properties:

IsClustered ........ false
  (Is clustering enabled [true|false])
LeadHost ...........
  (Name or IP address of the lead AAS host in the cluster)
PortNum ............ 7855
  (The port number on which the lead host listens for cluster messages)
MaxHosts ........... 10
  (The maximum number of AAS hosts in the cluster)
StartupTime ........ 60
  (Time, in seconds, each normal node waits while connecting to
   the cluster)
LiveIsClustered .... false
  (Clustering is currently enabled)
LiveLeadHost .......
  (Current name or IP address of the lead AAS host.)
LivePortNum ........ 7855
  (Current port number the lead host is listening on.)
LiveMaxHosts ....... 10
  (Current maximum number of AAS hosts allowed in the cluster)
LiveStartupTime .... 60
  (Current time each normal node waits while connecting to the
   lead host)
```

クラスターに関するコンソール・コマンドとそれぞれの機能の説明を以下の表にまとめます。

コマンド構文	説明
`cluster shutdown`	DB2 Alphablox クラスター内のすべてのインスタンスをシャットダウンします。 `CLUSTER SHUTDOWN` コマンドは、クラスター内のどのホストのコンソールからでも実行できます。
`get service cluster`	クラスター・プロパティーの現在の設定をリストします。先頭が “Live” のプロパティー (`LiveIsClustered` など) は、クラスター・マネージャーが使用している実際の値を戻します。ユーザーがその値を構成することはできません。その他のプロパティーの値は、この下の `SET SERVICE CLUSTER` コマンドによって設定できます。

`set service cluster <property> <value>`	クラスター・プロパティーの値を指定の値に変更します。
	
	property の有効な項目は、以下のとおりです。
	`IsClustered`
	`LeadHost`
	`PortNum`
	`MaxHosts`
	`StartupTime`
	
	新しい設定は、クラスター・マネージャーを再始動した時点で (通常は、DB2 Alphablox を再始動したときに) 有効になります。

コマンド構文	説明
`set service cluster IsClustered <True|False>`	クラスターの状態を*使用可能* または*使用不可* に設定します。 true に設定すると、クラスタリングが使用可能になります。 false に設定すると、クラスタリングが使用不可になります。
`set service cluster LeadHost <lead_host>`	リード・ホスト名または IP アドレス (*lead_host*) を設定します。この名前または IP アドレスは、ネットワーク上でリード・ホストがインストールされているマシンを決定するものでなければなりません。
`set service cluster PortNum <port_number>`	リード・ホストがクラスターの通信を listen するためのポート番号 (*port_number*) を設定します。このポートは、リード・ホスト上で使用可能なものでなければなりません。そうでない場合は、クラスターの開始が失敗します。
`set service cluster MaxHosts <maximum_hosts>`	クラスター内で許容できるホストの最大数 (*maximum_hosts*) を設定します (リード・ホストも含めた数です)。 *maximum_hosts* の値を超えた後にクラスターに参加しようとする通常ホストは、クラスターへの参加を拒否されます。
`set service cluster StartupTime <startup_time>`	開始時に通常ノードがクラスターに参加しようとする秒数 (*startup_time*) を設定します。 *startup_time* の秒数を超えると、通常ノード・ホストは自動的にシャットダウンします。
`show hosts`	現時点でクラスターに接続しているすべてのホストをリストします。クラスタリングが使用不可になっている場合、SHOW HOSTS は何も戻しません。

第 19 章 DB2 Alphablox コンソール・コマンド

DB2 Alphablox の管理は、コンソールから行うことも、DB2 Alphablox ホーム・ページの「**管理**」タブの下にある Web ページから行うこともできます。 DB2 Alphablox ホーム・ページのユーザー・インターフェースから構成するほとんどの管理アクティビティー (ユーザーやグループの作成や編集など) は、コンソール・コマンドによって構成することも可能です。この章では、コンソールの使用方法を示し、使用可能なコンソール・コマンドを取り上げます。

コンソールへのアクセス

DB2 Alphablox コンソールにアクセスするには 2 つの方法があります。

- 『HTML コンソール』
- 『Telnet コンソール』

HTML コンソール

DB2 Alphablox ホーム・ページからコンソールにアクセスするには、「**管理**」タブ、「**一般**」リンク、「**DB2 Alphablox コンソールの起動 (Launch DB2 Alphablox Console)**」リンクをクリックします。コンソールの HTML ページが開きます。

管理者は、Telnet セッションでコンソールを開くこともできます。その場合は、Windows の「**スタート**」メニューから DB2 Alphablox を開始したときに立ち上がる「コンソール」ウィンドウか、 Linux、 UNIX システムで DB2 Alphablox を開始したときに表示されるウィンドウを使用します。

Telnet コンソール

Telnet コンソールにアクセスするには、DB2 Alphablox を実行しているマシンで Telnet セッションを開始します。その際に、「**Telnet コンソール**」管理ページで構成されている Telnet ポートを指定します (デフォルトでは 23)。例えば、*pastrami* というマシンで DB2 Alphablox を実行している場合は、コマンド・プロンプトで以下のように入力します。

```
telnet pastrami 23
```

Windows マシンでは、インストール・プロセスによって、Telnet コンソールのショートカットが「**スタート**」メニューの「**すべてのプログラム**」グループの「**DB2 Alphablox**」フォルダーのインスタンス名フォルダー (デフォルト値は **AlphabloxAnalytics**) の「**コンソール**」の中に入ります。

コンソールのユーザー名とパスワードを入力して、Telnet コンソールに対する認証を行います。

Telnet コンソール・セッションを終了するには、以下のコマンドを入力します。

```
release
```

コマンド構文

DB2 Alphablox のコマンドは、以下の構文を使用します。

```
COMMAND object [value(s)] COMMAND object [value(s)] object [value(s)]
```

説明:

- object は、以下のいずれかです。
  - SERVER (デフォルト)
  - CONSOLE *console ID* (*console ID* は特定のアクティブ・コンソールの ID。例えば、C1)
  - サーバー・オブジェクトの名前。英語以外の言語で実行している場合に使用できる文字は、A-Z、a-z、0-9、下線、特殊文字 (アクセント付き文字など) です。名前の表示では大/小文字の区別がありますが、認証の対象となる実際の名前には大/小文字の区別がありません。 *Public*、*Private*、*Properties* は予約語であり、オブジェクト名としては使用できません。
- value によってさらにオブジェクトを修飾します。

以下の行を例に取ると、*get* はコマンド、*user* はサーバー・オブジェクトの名前、*admin* は特定のユーザー・オブジェクトの名前です。

```
get user admin
```

コマンドの後にオブジェクトだけを入力した場合は、そのコマンドに必要な値を示すメッセージが表示されます。例えば、以下のコマンドを入力したとしましょう。

```
create data source
```

この場合は、以下のメッセージが表示されます。

```
Create data source takes more parameters: data_source_name adapter_name
SERVER server_name [property value]...
```

プロパティー名は、該当するユーザー・インターフェースのページに表示される名前に対応しています。例えば、DB2 OLAP Server または Hyperion Essbase のデータ・ソースを作成している場合、プロパティー名は、*application*、*database*、*username*、*password*、*maxrows*、*maxcols*、*useaasuserauth* となります。

コマンド省略形

コマンドの入力速度を上げるために、コンソールではコマンド省略形がサポートされています。例えば、H は HELP コマンドの省略形です。同じ文字で始まるコマンドが複数ある場合は、省略形が固有のものになるだけの長さが必要です。例えば、START コマンドと STATISTICS コマンドはいずれも STA で始まるので、その後に 1 文字を追加することによって固有の省略形を作ることができます (つまり、STAR は START、STAT は STATISTICS です)。

ヒント: コマンドの使用に関する注意点を以下にまとめます。

- コンソールでは、コマンド行構文全体の省略形もサポートされています。例えば、H C によって CREATE コマンドのヘルプを表示したり、G U L によって名前の先頭が L のユーザーのプロパティーを取り出したりできます。
- 値のストリングを引用符で囲む必要はありません。ただし、データベース・アダプターは**例外**です (例えば、“ibm db2 olap server” のように入力します)。

- 一般プロパティーの省略形を使用する場合は、プロパティーの説明ではなく名前の最初の文字を使用します。例えば、DefaultMessageLevel という名前のプロパティーの説明が Initial Console Message Level だとすれば、省略形は DEFAULTM のようになります。

コンソール・コマンドのリスト

ほとんどのコマンドは、「コンソール」画面 (「**管理**」タブの下にある「**一般**」ページの「**DB2 Alphablox コンソールの起動 (Launch DB2 Alphablox Console)**」リンクから表示) と DB2 Alphablox コマンド画面のどちらからでも入力できます。ただし、その片方からしか実行できないコマンドもいくつかあります。さらに、すべてのコマンドが Telnet でサポートされているわけではありません。 DB2 Alphablox Cube Server のコンソール・コマンドのリストについては、「*DB2 Alphablox Cube Server 管理者用ガイド*」を参照してください。クラスタリング固有のコマンドのリストについては、148 ページの『クラスターに関するコンソール・コマンド』を参照してください。拡張可能ユーザー・マネージャー固有のコマンドのリストについては、110 ページの『拡張可能ユーザー・マネージャーの Telnet コンソール・コマンド』を参照してください。

注: 以下に示すコンソール・コマンドは DB2 Alphablox にのみ適用されます。例えば、WebSphere サーバーにインストールされている DB2 Alphablox からアプリケーションを削除する場合、そのアプリケーションのリストは DB2 Alphablox からのみ削除されます。そのアプリケーションを WebSphere サーバーから削除するには、WebSphere Application Server の管理コンソールを使用して削除する必要があります。

コマンドと例	目的と説明
ADD object value object value add group admin user radams	最初のオブジェクトに 2 番目のオブジェクトを追加します。 admin というグループに radams というユーザーを追加します。
CREATE object value(s) create user radams haggis create data source TBC ibm db2 olap server adapter server db2erver1 application demo database basic username user1 password password1 maxrows 1000 maxcols 1000 useaasuserauth false	ユーザーやグループなどのオブジェクトを作成します。 名前が radams でパスワードが haggis という値の新しいユーザーを作成します。 *db2server1* という名前の IBM DB2 OLAP Server の TBC (demo/basic) アプリケーションにアクセスする *TBC* というデータ・ソースを作成します。
DELETE object value delete user radams delete outlinecache...	DB2 Alphablox と DB2 Alphablox リポジトリーからオブジェクトを削除します。 DB2 Alphablox からアプリケーションを削除しても、そのアプリケーションは WebSphere サーバーから削除されません。 DB2 Alphablox から radams というユーザーを削除します。 DB2 OLAP Server または Essbase のキャッシュから 1 つの項目を削除します。

コマンドと例	目的と説明
EXIT	DB2 Alphablox を停止して、「コンソール」画面を終了します。このコマンドは、DB2 Alphablox コマンド画面からのみ実行できます。DB2 Alphablox ホーム・ページからアクセスする「コンソール」画面からは実行できません。
ExtUserManager	拡張可能ユーザー・マネージャーのコンソール・コマンドについては、110 ページの『拡張可能ユーザー・マネージャーの Telnet コンソール・コマンド』を参照してください。
GET object value get get user radams	オブジェクトの情報を表示します。 サーバーの情報を表示します (Server がデフォルト・オブジェクトなので)。 radams というユーザーのプロパティーを表示します。
HELP object help help create	指定したコマンドのヘルプ情報を表示します。 すべてのコマンドのリストを表示します。 CREATE コマンドの説明と構文を表示します。
KILL object kill 1046976892932726235	指定したオブジェクト (通常はユーザー・セッション) の現在のインスタンスを強制終了します。 1046976892932726235 という ID のセッションを強制終了します。
LOAD object value load user <userName> load theme load license	サーバー・ワークスペースにオブジェクトをロードします。 サーバー・ワークスペースに <userName> というユーザーをロードします。 サーバー・ワークスペースにすべての使用可能なテーマをロードします。 (このコマンドは、指定したテーマだけのロードはサポートしていません。) DB2 Alphablox の `bin` ディレクトリーにあるライセンス・ファイル (`license.xml`) をロード (更新) します。
LOCK	サーバーに対するローカル・アクセスをロックします。このコマンドを入力した後で DB2 Alphablox にアクセスするには、ユーザー名とパスワードの入力が必要です。このコマンドは、DB2 Alphablox の「コンソール」画面と Telnet からのみ実行できます。

コマンドと例	目的と説明
MESSAGE object value(s) message review 20 message review debug 20 message C2 Testing	他のコンソールやログ・ファイルにシステム・メッセージを送信します。 現在のコンソールに最新の 20 個のメッセージを送信します。 現在のコンソールに最新の 20 個の DEBUG メッセージを送信します。 コンソール C2 に “Testing” というメッセージを送信します。
OBJECTS object objects objects user	管理可能オブジェクトの情報を表示します。 管理可能オブジェクトの階層を表示します。 ユーザーの情報を表示します。
RELEASE	このコマンドを入力したリモート・サーバー・コンソールを解放します。このコマンドは、DB2 Alphablox の「コンソール」画面からのみ実行できます。
REMOVE object object REMOVE group admin user radams	ターゲット・オブジェクトからソース・オブジェクトを除去します。 admin というグループから radams というユーザーを除去します。
REPORT object report debug	現在のコンソールのメッセージ・レベルを DEBUG、VERBOSE、INFO、SYSTEM、WARNING、ERROR、FATAL のいずれかに設定します。 メッセージ・レベルを DEBUG に設定します。
RESUME object resume user	指定したサービスを中断状態から開始します。 ユーザー・マネージャーを前の中断状態から開始します。
RUN object run abc.console run *createUsers.txt*	ターゲットとして指定したコマンド・ファイルを実行します。 abc.console コマンド・ファイルを実行します。 *createUsers.txt* ファイルを実行します。 (このファイルは、新しいユーザーを作成して初期のパスワードを設定するための一連の create user コマンドが入ったバッチ・ファイルである可能性があります。例については、注 2 を参照してください。)

コマンドと例	目的と説明
SAVE object value	ターゲットのプロパティーを保管します。
save save user radams	現時点のすべての「一般」プロパティーを保管します (「一般」がデフォルト・オブジェクトであるため)。コンソールで入力したすべての変更は、現在のセッションでのみ有効であり、このコマンドを実行するか、DB2 Alphablox の「終了時に保管」プロパティーを「はい」に設定しない限り失われてしまいます。 radams というユーザーのプロパティーを保管します。
SET object value object value	オブジェクトに値を設定します。
set user radams password castle set smtpserver mail set resolveAliasesToBaseMembers	radams というユーザーのパスワードを castle に設定します。 SMTP サーバーを *mail* というメール・サーバーに設定します。
SHOW object	ターゲットの情報を取得します。
show topics show server show user radams show theme show outlinecache show hosts show	入手可能な情報が存在するすべてのトピックを表示します。 すべてのサーバー・トピック (ユーザーを含む) を表示します。 radams というユーザーの情報を表示します。 テーマの名前と説明を表示します。 DB2 OLAP Server または Essbase のキャッシュにあるすべての項目を表示します。 クラスタリングが使用可能な場合に、クラスター内の各マシンの名前を表示します。 オブジェクトのリストを表示します。 DB2 OLAP Server または Hyperion Essbase に対して照会を実行するアプリケーションやテスト接続の使用によってクライアント・ライブラリーがまだロードされていない場合に SHOW コマンドを実行すると、そのクライアント・ライブラリー API がロードされ、そのバージョンが表示されます。
START object	サービスを停止状態から開始します。
start user	ユーザー・マネージャーを前の停止状態から開始します。
STATISTICS object value	ターゲットの入手可能な統計を表示します。
statistics user	すべてのユーザーの統計を表示します。
statistics user radams	radams というユーザーの統計を表示します。

コマンドと例	目的と説明
STOP object stop user	稼働状態または中断状態のサービスを停止します。サービスを停止状態から開始するには、START を使用します。 **注:** 実行中の DB2 Alphablox キューブがすべて停止しているのでない限り、STOP コマンドを使用してキューブ・マネージャーを停止しないでください。 ユーザー・マネージャーを停止します。
SUSPEND object suspend user	サービスを中断します。例えば、ユーザー・マネージャーを中断すると、新しいユーザー・セッションのインスタンス化ができなくなります。サービスを中断状態から開始するには、RESUME を使用します。 ユーザー・マネージャーを中断します。

Essbase 固有のコンソール・コマンド

DB2 Alphablox は、パフォーマンスを高めるために、DB2 OLAP Server や Hyperion Essbase の一括表示をメモリー内にキャッシュします。ここでは、一括表示キャッシュの管理に使用する 2 つのコマンドと、別名を基本メンバー名に解決するためのコマンドの構文を解説します。取り上げるのは、以下のコマンドです。

- 157 ページの『RESOLVEALIASESTOBASEMEMBERS コマンド』
- 158 ページの『SHOW OUTLINECACHE コマンド』
- 158 ページの『DELETE OUTLINECACHE コマンド』

RESOLVEALIASESTOBASEMEMBERS コマンド

DB2 OLAP Server や Essbase のデータベースでは、メンバー名の別名を設定できます。 RESOLVEALIASESTOBASEMEMBERS というサーバー・プロパティーは、その時々で別々の名前に解決される可能性がある別名で照会を保管するブックマークに対して使用します。例えば、「この四半期」という別名を参照する照会にブックマークを設定している場合、その照会の参照先になる DB2 OLAP Server や Essbase のデータベース内の実際のメンバーは、現在と 6 か月前とでは異なります。そのブックマークでこの四半期の現行データを取得するには、RESOLVEALIASESTOBASEMEMBERS サーバー・プロパティーを TRUE に設定する必要があります。

RESOLVEALIASESTOBASEMEMBERS コンソール・コマンドの構文は、以下のとおりです。

```
set resolvealiasestobasemembers true | false
```

RESOLVEALIASESTOBASEMEMBERS サーバー・プロパティーの現在の状態を戻すコマンドは、以下のとおりです。

```
get resolvealiasestobasemembers

ResolveAliasesToBaseMembers .... false
  (ResolveAliasesToBaseMembers)
```

この機能を使用可能にするには、以下のコンソール・コマンドを入力します。

```
set resolvealiasestobasemembers true
```

ヒント: このコマンドの省略形を使用する場合は、固有の名前になるように、以下のようにして十分な数の文字を入力します。

```
set resolve true
```

SHOW OUTLINECACHE コマンド

DB2 OLAP Server または Hyperion Essbase のデータ・ソースのための SHOW OUTLINECACHE コマンドの構文は、以下のとおりです。

```
show outlinecache [essbasecachemanager [entry [entryname]]]
```

このコマンドによって、キャッシュ・マネージャーまたはその管理対象の項目に関する情報を表示できます。例えば、DB2 OLAP Server または Essbase のキャッシュ・マネージャーの管理対象になっているすべての項目を表示するには、以下のコマンドを使用します。

```
show outlinecache essbasecachemanager entry
```

システムからは以下のような応答があります。

```
Entry
  Entry MDB1.Financial.Current
  ....Total accesses: 7
  ....Current accessors: 1
```

この応答によれば、DB2 OLAP Server または Essbase のキャッシュ・マネージャーが現時点で管理しているのは、MDB1.Financial.Current という 1 つの項目だけです。このキャッシュは、合計で 7 回 (累積回数) アクセスされており、現時点で 1 つの接続によって使用されています。

DELETE OUTLINECACHE コマンド

DB2 OLAP Server または Hyperion Essbase のデータ・ソースのための DELETE OUTLINECACHE コマンドの構文は、以下のとおりです。

```
delete outlinecache [essbasecachemanager [entry [entryname]]]
```

このコマンドによって、指定した項目をキャッシュから除去できます。 DB2 Alphablox は、現在の接続数 (accessors の値) が 0 の場合にのみ、その項目を除去します。項目を除去すると、その項目に関連するすべてのメモリー・リソースが解放されます。例えば、キャッシュから MDB1.Financial.Current という項目を削除するには、以下のコマンドを使用します。

```
delete outlinecache essbasecachemanager entry MDB1.Financial.Current
```

システムからは以下のような応答があります。指定した項目が確かに削除されています。

```
MDB1.Financial.Current
Cache entry deleted.
```

その項目がキャッシュ内にもう存在しないことを確かめるには、以下のコマンドを実行して、応答を確認します。

```
show outlinecache essbasecachemanager

EssbaseCache Manager
  Total entries: 0
```

コンソール・コマンドに関する注意点

ここでは、各種の動作について解説し、DB2 Alphablox コンソールを使用するためのヒントを示します。取り上げるのは、以下のトピックです。

一般プロパティーの表示

DB2 Alphablox コンソールを使用して、一般プロパティーやオブジェクトを表示できます。ほとんどのプロパティー変更は、すぐに有効になります。そうでない場合は、変更を有効にするために DB2 Alphablox の再始動が必要であるというメッセージが表示されます。 DB2 Alphablox の現在のプロパティーの完全リストを確認するには、GET コマンドを使用します。

コンソールの下部にあるリンクから、頻繁に使用する情報にアクセスできます。

- 「**Help**」からは、DB2 Alphablox のコマンド・リストを表示できます。
- 「**Status**」からは、DB2 Alphablox の状況情報を表示できます。
- 「**Messages**」からは、DB2 Alphablox のログのスクロール可能ビューを表示できます。
- 「**Sessions**」からは、この DB2 Alphablox インスタンスのすべての現行セッションの状況情報を表示できます。
- 「**Users**」からは、ユーザーのリストを表示できます (ユーザーがログオンしているかどうかは無関係)。
- 「**Services**」からは、DB2 Alphablox のコンポーネントの状況を表示できます。サービスのリストについては、9 ページの『DB2 Alphablox のアーキテクチャー』を参照してください。
- 「**Settings**」からは、DB2 Alphablox の各プロパティーの現在の設定を表示できます (開始と拡張の両方)。
- 「**Show**」からは、DB2 Alphablox のトピックの階層リストを表示できます。
- 「**History**」からは、このコンソールから実行されたコマンドの履歴を表示できます。

コンソールにアクセスする方法については、151 ページの『コンソールへのアクセス』を参照してください。

メッセージ・レベル

DB2 Alphablox から生成されるメッセージは、すべてのアクティブ・コンソールとアクティブ・ログ・ファイルに送信できます。メッセージには、以下の情報が含まれています。

- 日時
- メッセージ・レベル
- メッセージ・テキスト
- メッセージ送信元 (サービスまたはユーザーの名前)

典型的なメッセージを以下に示します。

```
01/9/99 12:25:33 PM [VERBOSE] Request 4223: Processing request
'/servlet/AnalysisServer/console/consolestyle.css' [Session 41, User radams]
```

重大度の順に (低いほうから高いほうに) メッセージ・レベルを並べると、以下のようになります。

- DEBUG: デバッグ情報です。
- VERBOSE: すべてのシステム・メッセージです。
- INFO: 管理者の処置を必要としない小さなシステム・イベントです。
- SYSTEM: ユーザーの作成などの通常のシステム・イベントです。
- WARNING: 回復可能なエラーです (ただし、管理者としては調査しておきたいエラーです)。
- ERROR: 回復不能なエラーです。
- FATAL: サーバーの強制終了を引き起こすエラーです。

コンソールからのテキスト・ファイルの実行

コンソールから *run* コマンドを使用して、任意の数の DB2 Alphablox コマンドを含んだプレーン・テキスト・ファイルを実行できます。例えば、以下の行を含んだ *d:¥CreateUser.txt* というファイルがあるとしましょう。

```
create user radams blue
create user sadams green
create user lplanting purple
create user klawrence yellow
create user dmessink orange
```

コンソールから *run* コマンドを使用してこのファイルを実行すれば、5 つの新しいユーザーを作成し、それぞれの初期パスワードを設定できます。そのためには、コンソールから以下のコマンドを実行します。

```
run d:/CreateUser.txt
```

注: ファイル名は、完全修飾名にする必要があります。

DB2 Alphablox のログ・メッセージ

DB2 Alphablox のログ・ファイル (デフォルトでは Server.log) には、以下のような無害なメッセージが書き出されることがあります。

```
[VERBOSE] Request 36: File not found '...¥resources¥basic_en_US.class'
[VERBOSE] Request 37: File not found '...¥resources¥basic_en_US.properties'
[VERBOSE] Request 38: File not found '...¥resources¥basic_en.class'
[VERBOSE] Request 39: File not found '...¥resources¥basic_en.properties' ...
[VERBOSE] Request 45: File not found '...¥resources¥metal_en_US.class'
[VERBOSE] Request 46: File not found '...¥resources¥metal_en_US.properties'
[VERBOSE] Request 47: File not found '...¥resources¥metal_en.class'
[VERBOSE] Request 48: File not found '...¥resources¥metal_en.properties'
...
```

この例では、JVM がクライアント・ロケールの言語と地域に最適なリソース・バンドルを検出しようとしています。これは、DB2 Alphablox の正しい実行に特に必要というわけではないので、この種のメッセージは、無視して差し支えありません。

第 20 章 Alphablox FastForward アプリケーションの管理

この章では、Alphablox FastForward アプリケーションを管理する方法を取り上げ、特に構成の方法と基本的な管理タスクについて解説します。

概要

Alphablox FastForward は、DB2 Alphablox にあらかじめインストールされているサンプル・アプリケーション・フレームワークです。このフレームワークを使用すれば、カスタム分析ビューを開発し、展開し、ビジネス組織全体で共有する作業を短時間で実行できます。 FastForward フレームワークには、一般的なアプリケーション・サービス (セキュリティー、コラボレーション、カスタマイズ、パーソナライゼーションなど) がすぐに使用できる形で用意されています。アプリケーション管理者 (特に OLAP 管理者) は、新しいバージョンの FastForward アプリケーションを作成し、レポートをパブリッシュし、コードを見ることもなくその新しいアプリケーションを展開できます。レポートをパブリッシュする作業も、レポートのテンプレートを選択して、レポートのパラメーターを構成するだけで行えます。しかも、このアプリケーション・フレームワークは、柔軟性と拡張性が高いので、JSP 開発者は、変更や拡張を行って、アプリケーション管理者が構成して展開するための新しいカスタム・レポート・テンプレートを作成できます。

FastForward アプリケーション・フレームワークには、レポートや分析のためのアプリケーションに一般的に装備されている機能が用意されています。例えば、以下の機能があります。

- Microsoft Excel へのエクスポート
- 印刷可能なビューの生成
- データのパーソナル・ビューを簡単に保管して共有する機能
- 他のユーザーに E メールでビューを送信する機能
- ビュー間の簡単なナビゲーション

FastForward ユーザーの役割

Alphablox FastForward ユーザーには、アプリケーション管理者、テンプレート開発者、ユーザーという 3 種類の大きな役割があります。 FastForward アプリケーションの成功には、この 3 つのグループの協力関係が欠かせません。この 3 つの役割について、これから簡単に説明します。

アプリケーション管理者

アプリケーション管理者 (通常は OLAP 管理者) は、いくつかの設定を定義して新しいバージョンの FastForward アプリケーションを作成し、用意されているレポート・テンプレートを基にレポートを作成し、ユーザーに対してソリューションを迅速に展開する必要があります。既存のレポート・テンプレートがユーザーの要件を満たしていなければ、アプリケーション管理者はテンプレート開発者と協力して、新しいレポート・テンプレートを作成します。アプリケーション管理者が作業を行

うときには、OLAP データベースに関する自分自身の経験のほかに、本書や FastForward アプリケーションの管理タスク・モードで利用できるオンラインの管理ヘルプも参考にできます。

テンプレート開発者

テンプレート開発者とは基本的に、カスタム・レポート・テンプレートの作成を主に担当する JSP 開発者です。アプリケーション管理者がエンド・ユーザーからの要請に応じてレポートを構成するときに、既存のテンプレートを使用できない場合に、そのようなカスタム・テンプレートが必要になります。テンプレート開発者は、アプリケーション管理者やユーザーとよく打ち合わせながら、新しいレポート・テンプレートを作成する必要があります。既存のレポート・テンプレートを変更するだけですむ場合もあれば、まったく新しいテンプレートを作成しなければならない場合もあります。

テンプレート開発者は、Blox のタグ・ライブラリー、サーバー・サイド Java API、クライアント・サイド JavaScript API のほかに、Web プログラミングに関する自分自身の経験も活用しながら、ほとんどのニーズに対応したテンプレートを作成しなければなりません。さらに、テンプレート開発者、DB2 Alphablox のアプリケーションやビューの作成方法だけでなく、FastForward のユーザー・ヘルプ (ユーザー・モードの「ヘルプ」ボタンからアクセス) や管理者ヘルプ (管理タスク・モードの「ヘルプ」ボタンからアクセス) についても熟知している必要があります。

「ユーザー」

ユーザー (通常は企業や団体の中のビジネス・アナリストや他の業務ユーザー) は、FastForward アプリケーションにログインして、パブリッシュ済みのレポートを基に業務上の問題を分析する必要があります。各 FastForward アプリケーションの対話機能にもよりますが、ユーザーは、データの操作、データ階層内のドリルダウン/ドリルアップ、チャート・タイプの変更、コメントの追加などを行えます。ユーザーは、業務上の問題に対応するためにビューを変更した後に、その現在のビューを保管できます。後からの使用に備えて「プライベート」タブの下に保管用のレポートを作成することも、アプリケーション・ユーザーの定義済みのグループを対象とした「グループ」タブの下に配置して共有することも可能です。

各レポートには通常、ユーザーがレポートの上部にあるアプリケーション・ツールバーから使用できる他のオプションがいくつか用意されています。オンライン分析のためにレポートを保管するオプションのほかに、Excel にエクスポートするためのオプションもあります。このオプションを使用すれば、後からオフラインで分析を行うために Microsoft Excel のスプレッドシートにビューをエクスポートできます。ユーザーはさらに、印刷プレビューのオプションを使用して、特定のビューのコピーを印刷することもできます。また必要に応じて、現在のビューへのリンクを含んだ E メール・メッセージを開き、コメントを追加してから、その E メールを他のアプリケーション・ユーザーに送信することも可能です。

必要なレポートがアプリケーションに用意されていない場合、ユーザーは通常、アプリケーション管理者に対して直接新しいレポートを要請します。

FastForward アプリケーションのシステム要件

「インストール・ガイド」の『システム要件』のセクションに示した要件のほかに、Alphablox FastForward には、DHTML クライアントとのみ連動するという要件があります。

Alphablox FastForward アプリケーションの作成

新しいバージョンの Alphablox FastForward アプリケーションを作成するには、管理者権限のあるユーザーとしてログインしてから、以下の手順を実行します。

1. Web ブラウザーで DB2 Alphablox の管理ページを開きます。デフォルトでは、「アプリケーション」ページが表示されます。
2. 「アプリケーション」ページで、Alphablox FastForward アプリケーションを起動します。
3. アプリケーション・ツールバーで、「管理タスク」ボタンをクリックします。
4. 「テンプレート・アプリケーション」ダイアログ・ウィンドウに「作成」と「続行」という 2 つのオプションが表示されます。「作成」ボタンをクリックします。
5. 「新規アプリケーション」ウィンドウで、以下のフィールドを設定します。

入力フィールド	説明
コンテキスト名	ディレクトリー名として使用するアプリケーションのコンテキスト名を入力します。この名前にはスペースを使用できません。
表示名	「アプリケーション」ページに表示される表示名です。この名前にはスペースを使用できます。
説明	オプション。アプリケーションの簡単な説明です。
管理者役割	必須。アプリケーションの config.xml ファイル (アプリケーションの WEB-INF ディレクトリー内) の AdministratorRole 値を設定します。デフォルト設定は、`AlphabloxAdministrator` です。

重要: 既存の Alphablox FastForward アプリケーションを構成して使用するのではなく、必ず新しいバージョンを作成してください。このサンプル・アプリケーションは、サーバーの後続アップグレードのたびに上書きされていくので、変更は失われてしまいます。

管理者役割の変更

Alphablox FastForward アプリケーションでは、デフォルトで `AdministratorRole` が `AlphabloxAdministrator` に設定されます。特定のアプリケーションの管理者役割を変更するには、上記の「新規アプリケーション」ダイアログで設定を行うか、以下の手順を実行します。

1. 以下の場所にある FastForward アプリケーションの config.xml ファイルを開きます。

<applicationDirectory>/WEB-INF/config.xml

2. config.xml ファイルの以下のセクションで、割り当てられている役割を変更し、AlphabloxAdministrator の代わりにそのアプリケーションの管理者として割り当てる新しい役割を設定します。

```
<param>     <param-name>AdministratorRole</param-name>
<param-value><![CDATA[AlphabloxAdministrator]]></param-value> </param>
```

役割の定義と新しい役割の作成方法については、63 ページの『第 10 章 役割の定義』を参照してください。

FastForward アプリケーションの管理

FastForward 管理アプリケーションには、大まかに分けると以下の 3 つの機能があります。

- パブリッシュ済みのレポートの管理
- FastForward アプリケーションの基本的なユーザー・インターフェース様式の管理
- アプリケーション・ログの表示とクリア

管理タスク・モードでログインしている場合は、アプリケーション・ツールバーの「ヘルプ」ボタンをクリックして、FastForward アプリケーションの管理に関する情報を表示することもできます。

レポート・アクセス・カテゴリーとセキュリティー

FastForward アプリケーションには、ユーザーに関して、パブリッシュ済み、プライベート、グループという 3 つのレポート・アクセス・カテゴリーがあります。

パブリッシュ済みのレポート

パブリッシュ済みのレポートの管理は、上記のように管理画面から行います。パブリッシュ済みのレポートは、特に管理を行わなければ、FastForward アプリケーションにアクセスできるすべてのユーザーから見える状態になります。しかし、管理者は、パブリッシュ済みのレポートへのアクセスを制御するために、レポートのフォルダーに対してセキュリティーを設定できます。フォルダーに対するアクセス許可を持っているユーザーは、そのフォルダー内のすべてのレポートにアクセスできます。管理者は、フォルダーに対するアクセス許可を割り当てるために、フォルダーを 1 つ以上の DB2 Alphablox グループに関連付けます。

プライベートのレポートとグループのレポート

プライベート・アクセスとグループ・アクセスのレポートは、パブリッシュ済みのレポートを基にユーザーが作成します。プライベート・アクセスのレポートは、そのレポートを作成したユーザーだけが利用できます。グループ・アクセスのレポートは、DB2 Alphablox ユーザー・グループのメンバーであるユーザーだけが利用できます。また、そのグループのメンバーは、そのグループだけがアクセスできるレポートを作成できます。

レイアウトとコントロール

すべての Alphablox FastForward アプリケーションは、同じようなレイアウト構造になります。例えば、左側にはナビゲーション・メニューがあり、右上にはアプリケーション・ツールバーがあります。さらに、アプリケーション・ツールバーの下にあるレポート・ウィンドウには、使用可能なレポートが表示されます。

ナビゲーション・メニュー

ナビゲーション・メニューには以下の選択項目があります。

ボタン	説明
フォルダーの作成	「新規フォルダー (New Folder)」という名前の新しいフォルダーを作成します。フォルダーに変更を加えるための「フォルダーの編集」画面がロードされます。
レポートの作成	新しいレポートを編集するための「レポートの編集」画面をロードします。左側のレポート・ツリーにレポートを表示するには、そのレポートを正常に保管する必要があります。
レポートの削除	レポートの削除を確認するためのダイアログをポップアップ表示します。
コピー	レポートのコピーを作成し、レポート名の先頭に「Copy of」というストリングを付けます。ファイル名は、変更が可能です。

レポートの管理

Alphablox FastForward アプリケーションの管理タスク・モードのユーザー・インターフェースには、基本的なレポート管理機能が用意されています。

レポートの作成

レポートを作成するには、ツールバーの「レポートの作成」ボタンをクリックします。「レポートの編集」画面がロードされるので、その画面で以下の設定値を構成できます。

フィールド	説明
名前	ツリーに表示するレポート名です。
説明	ユーザーがツリー内のレポートの上にマウスを置いたときや、レポートをロードしたときに表示される簡略説明です。
テンプレート	テンプレートの名前です。テンプレートのドロップダウン・リストには、アプリケーションで使用できるすべてのテンプレートの名前が表示されます。レポート・テンプレートは、アプリケーションの `templates` ディレクトリーに格納されます。レポート・テンプレートの種類によっては、他のパラメーターも選択する必要があります。

保管	レポートを保管します。レポート・パラメーターの検証ができない場合は、エラーが表示されます。
プレビュー	新しいブラウザー・ウィンドウでレポートを開きます。パラメーターの検証ができない場合は、エラーが表示されます。

レポートの変更

レポートを編集するには、ツリー内の既存のレポートを左クリックします。「レポートの編集」画面がロードされます。

レポートの削除

レポートを削除するには、削除するレポートを左クリックしてから、ツールバーの「レポートの削除」ボタンをクリックします。

レポートの移動

レポートを移動するには、以下のようにします。

1. 移動するレポートの横にあるレポート・アイコンの上にマウス・ポインターを置き、左マウス・ボタンを押したままにします。
2. マウス・ボタンを押したまま、ツリー内の目的の宛先にポインターを移動し、マウス・ボタンを放します。

フォルダーの管理

フォルダーの管理作業は、アプリケーションのユーザー・インターフェースから簡単に実行できます。

フォルダーの作成

フォルダーを作成するには、ツールバーの「フォルダーの作成」ボタンをクリックします。「フォルダーの編集」画面がロードされ、以下の編集可能なフィールドが表示されます。

フィールド	説明
名前	ツリーに表示するレポート名です。
説明	ユーザーがツリー内のフォルダーの上にマウスを置いたときに表示される値です。
アクセス・グループ	使用可能な DB2 Alphablox ユーザー・グループのリストです。複数のユーザー・グループを選択できます。グループを選択しない場合は、そのレポートにすべてのユーザーがアクセスできるようになります (警告が表示されます)。
保管	フォルダーを保管します。

フォルダーの変更

フォルダーを編集するには、ツリー内の既存のレポートを左クリックします。「フォルダーの編集」画面がロードされ、その画面で変更を行えます。

フォルダーの削除

レポートを削除するには、削除するレポートを左クリックしてから、ツールバーの「レポートの削除」ボタンをクリックします。

フォルダーの移動

フォルダーを移動するには、以下のようにします。

1. 移動するフォルダーの横にあるフォルダー・アイコンの上にマウス・ポインターを置き、左マウス・ボタンを押したままにします。
2. マウス・ボタンを押したまま、ツリー内の目的の宛先にそのフォルダーをドラッグし、マウス・ボタンを放します。

アプリケーション・プロパティーの管理

アプリケーション・プロパティーを管理するには、ページの右上隅にある「グローバル設定」ボタンをクリックします。以下の設定を変更できます。

フィールド	説明
タイトル	アプリケーションのタイトルです。これは、ユーザーのアプリケーションの上部と管理ページの上部に表示されます。
ロゴ	ユーザーのアプリケーションのロゴとして表示されるファイル名です。
ロゴ・イメージのアップロード	ローカル・マシンで使用可能なファイル (GIF 形式か JPEG 形式) から新しいロゴ・イメージをアップロードします。ただし、「保管」をクリックするまで実際にアップロードは行われません。
テーマ	インストール済みのテーマを選択するためのドロップダウン・リストです。ただし、サンプル・アプリケーションでは、`fastforward` テーマしか使用できません。
フッター	ユーザーのアプリケーション・ページの下部に表示するテキストを変更します。
管理 E メール	アプリケーションのフィードバックに使用する E メール・アドレスを設定します。このアドレスを設定しない場合は、ユーザーのアプリケーション・ページでフィードバック・ボタンが使用不可になります。
保管	上記の編集フィールドで加えた変更を保管します。

アプリケーション・ログの使用

FastForward アプリケーション・フレームワークには、アプリケーション・レベルの情報や異常を報告するための専用のログ・ファイルがあります。このログ・ファイルを表示したりクリアしたりするには、「ログ・ファイルの表示 (View Log File)」ボタンをクリックします。

付録. OLAP の用語と概念

このセクションでは、マルチディメンション分析 (MDA) とオンライン分析処理 (OLAP) の理解に欠かせない重要な用語と概念について説明します。まず、2 ディメンション分析の用語を取り上げてから、MDA で拡張されている用語の定義を示します。その後、OLAP データベースで使用される関連用語について説明します。

OLAP の他の用語の定義については、DB2 Alphablox の用語集や OLAP Council の用語集も参照してください。

- 『2 ディメンション分析』
- 『マルチディメンション分析』
- 171 ページの『OLAP データベースの用語』

2 ディメンション分析

スプレッドシートやレポートなどのツールを使用して 2 ディメンション・データ分析を実行するユーザーにとってなじみ深い用語を以下にまとめます (その後の表に例を示します)。

- **行**には、関連データのセットが含まれています。以下の表には、2 つのデータ行が含まれています。
- **行ラベル**は、データ値の左に表示されます。行ラベルは紺色です。
- **列**にも、関連データのセットが含まれています。以下の表には、4 つのデータ列が含まれています。
- **列ラベル**は、データ値の上に表示されます。列ラベルは栗色です。
- **データ・ポイント** (**セル**ともいう) は、行と列の交点です。データ・ポイントの背景は灰色です。
- **データ値**は、特定のデータ・ポイントに存在するエレメントです。 10 から 30 までの数値がデータ値です。

2 ディメンションの売り上げ表

	第 1 四半期のユニット数	第 2 四半期のユニット数	第 3 四半期のユニット数	第 4 四半期のユニット数
ダイエット・コーラ	10	15	20	25
コーラ	12	18	24	30

マルチディメンション分析

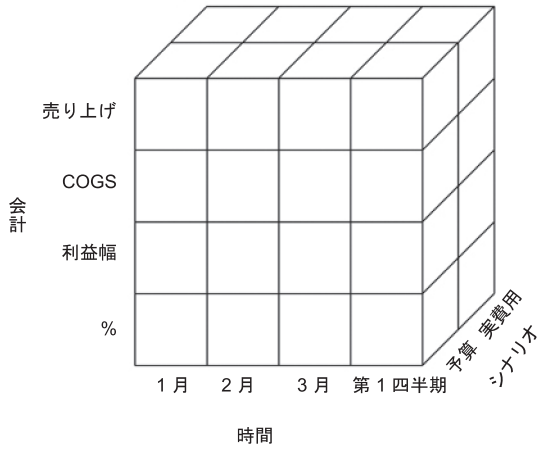
マルチディメンション分析では、これらの用語と概念はさらに複雑になっています。 MDA にかかわっているのは、行と列とその交点だけではありません。マルチディメンション・データを表示するメディアは、2 ディメンションのグリッドである場合が多いですが、MDA の機能には、行、列、データ・ポイントのほかに、ディメンション、階層、メンバー、タイトル、値、インスタンスが含まれています。

マルチディメンション・データに関する用語の定義を以下に示します (その後のマルチディメンション・マトリックスに例を示します)。

- **ディメンション**は、データ・キューブの構造属性であり、関連する階層メンバーで構成されています。例えば、「時間」ディメンションには、「年」、「四半期」、「月」、「週」などのメンバーを組み込めます。「地理」ディメンションには、「地域」、「国」、「市区町村」などのメンバーを組み込めます。
- **ディメンション・メンバー**は、ディメンション内のエレメントです。例えば、「四半期」や「月」は、「時間」ディメンションのメンバーになります。
- **ディメンション階層**は、ディメンション・メンバーを親子関係に編成したものです。例えば、「月」は「四半期」の子になり (つまり、「四半期」に属し)、「四半期」は「年」の子になる、といった具合です。
- **ディメンション・タイトル**は、ディメンションの識別名 (「時間」や「地理」など) です。
- **ディメンション・メンバー・タイトル**は、ディメンション・メンバーの識別名 (「月」や「地域」など) です。
- **ディメンション・メンバー値**は、ディメンション・メンバーのインスタンスです。例えば、1998 は、ディメンション・メンバー「年」の値です。
- **データ・ポイント**は、複数のディメンションの交点です。下のマルチディメンション売り上げマトリックスのデータ・ポイントは、背景が灰色になっています。
- **データ値**は、データ・ポイントに存在する値です。例えば、マルチディメンション売り上げマトリックスの灰色の領域にあるそれぞれの数値がデータ値です。

データ・キューブ

キューブに編成したデータの例を以下の図に示します。ディメンションが 3 つあり、各ディメンションにいくつかのメンバーがあります。



ディメンション	メンバー
会計	売り上げ、売り上げ原価 (COGS)、利益幅、利益幅率
時間	第 1 四半期 (1 月、2 月、3 月)
シナリオ	予算、実費用

マルチディメンション売り上げマトリックス

売り上げデータの階層マルチディメンション・ビューを以下の図に示します。列軸に「時間」ディメンションの 2 つのメンバー (「年」と「四半期」) を表示し、行軸に「在庫」ディメンションの 2 つのメンバー (「カテゴリー」と「製品」) を表示しています。

		列軸					
		第 1 四半期					
		1 月		2 月		3 月	
	会計	予算	実費用	予算	実費用	予算	実費用
行軸	売り上げ	300	350	300	325	325	325
	売り上げ原価	175	185	175	175	185	190
	利益幅	125	165	125	150	140	135
	利益幅率	42	47	42	46	43	42

OLAP データベースの用語

OLAP データベース内のデータの識別、編成、検索に役立つ用語を以下にまとめます。例えば、ユーザーが第 1 四半期 (Qtr1) からドリルダウンした場合、OLAP アクションは、Qtr1 (1 月 (Jan)、2 月 (Feb)、3 月 (Mar)) の「子」を検索します。

サンプル・ディメンション階層	用語と定義
⊟ Year ⊟ Qtr1 ⊞ Jan ⊞ Feb ⊞ Mar ⊟ Qtr2 ⊞ Apr ⊞ May ⊞ Jun ⊟ Qtr3 ⊞ Jul ⊞ Aug ⊞ Sep ⊟ Qtr4 ⊞ Oct ⊞ Nov ⊞ Dec	**ルート:** 階層内の最上位。この例では、ルートは「年」です。 **子孫:** ルートよりも下の世代のメンバー。この例にあるその他のすべてのディメンションは、ルートの子孫です。「1 月」は、「第 1 四半期」と「年」の子孫でもあります。 **子:** 他のメンバーよりも 1 つ下の世代のメンバー。「1 月」は、「第 1 四半期」の子になります。 **親:** 他のメンバーよりも 1 つ上の世代のメンバー。「第 1 四半期」は、「1 月」の親になります。 **兄弟:** 他のメンバーと同じ世代のメンバー。「1 月」、「2 月」、「3 月」は、兄弟になります。 **祖先:** 他のメンバーよりも上の世代のメンバー。「第 1 四半期」と「年」は、両方とも「1 月」の祖先になります。

注: これらの用語に大文字の「I」を接頭部として使用する場合 (つまり、IPARENT、ICHILD などを使用する場合)、その「I」は「包含」という意味です (英語の「inclusion」)。例えば、「親」のドリルアップでは、結果として親のデータだけが表示されます。「IPARENT」のドリルアップでは、結果として親と子の両方のデータが表示されます。

用語集

この用語集では、DB2 Alphablox に関連した用語を取り上げます。 169 ページの『OLAP の用語と概念』の説明も参照してください。さらに広範な用語集がインターネット上の「A Web of Online Dictionaries and WhatIs」などのサイトにあります。

［ア行］

アプリケーション・データベース. 特定のアプリケーションをサポートするためにサーバー・マシン上に存在するデータベース。実動データベースからアプリケーション・データベースを抽出して、アプリケーションに関連データへの高速アクセスを提供することもできます。DB2 Alphablox は、アプリケーション・データのソースとして、マルチディメンション・データベースとリレーショナル・データベースを使用します。

アプリケーション・ページ. DB2 Alphablox アプリケーションの一部である 1 つの HTML ページ。

アプレット. 「Java アプレット」を参照してください。

インスタンス化. 特定のクラスのオブジェクトを作成すること。例えば、ユーザーが DB2 Alphablox アプリケーションを呼び出すと、DB2 Alphablox は、そのアプリケーションをインスタンス化します (つまり、そのアプリケーションのインスタンスを作成します)。

インターネット. ネットワーキングとソフトウェアの共通プロトコル・セットを使用する世界的なコンピューター・ネットワーク。インターネットのユーザーは、Web ページに掲載されたディジタル情報を多数のプラットフォームの間で共有できます。「エクストラネット」と「イントラネット」も参照してください。

イントラネット. インターネットと同様のサービスを提供する内部用の社内ネットワーク。「エクストラネット」も参照してください。

永続性. プログラム実行とプログラム実行の間で、作成したオブジェクトや変数が存在し続け、それぞれの値を保持し続けるという、プログラム言語のプロパティー。

エクストラネット. 外部の世界 (インターネット) にアクセスできる社内 Web サイト。ただし、登録パスワードなどの特権メカニズムの使用が前提になります。エクストラネットを使用して、パートナーや顧客など、事前定義の関係を持った他者との対話を促進できます。

親. 他のディメンション・メンバーよりも 1 つ上の世代のメンバー。例については、『OLAP の用語と概念』の「親」を参照してください。

［カ行］

階層. ディメンションのメンバーを親子関係に編成したもの。通常は、子メンバーを統合したものが親メンバーに相当します。例えば、「時間」ディメンションの階層に、「年」、「四半期」、「月」というメンバーがあるとします。それぞれの「月」は特定の「四半期」の子であり、それぞれの「四半期」は特定の「年」の子になります。

拡張. サマリー値のサポート詳細を表示すること。「縮小」と対比してください。

カタログ. データ・ソースの構造における 1 つの編成レベル。 DB2 OLAP Server や Essbase では、これを「アプリケーション」といいます。一般に、データのキューブをグループ分けしたものがスキーマであり、スキーマをグループ分けしたものがカタログです。

行軸. マルチディメンション・データを表示するための 1 つの方向。行軸は、表示可能な 3 つの軸のうちの 1 つです (他の軸は、ページ軸と列軸)。行では、ディメンション・メンバーのデータ値を表示します。ユーザーは、行軸にディメンションを配置して、そのディメンションのメンバーのデータ値を表示できます。例えば、ユーザーは、行軸に「製品」ディメンションを配置して、メンバーの ID (「製品 ID」や「別名」など) と関連データ値 (売上額など) を表示できます。

兄弟. 他のディメンション・メンバーと同じ世代のメンバー。例については、『OLAP の用語と概念』の「兄弟」を参照してください。

グリッド. 行、列、ページの軸を使用した情報のマルチディメンション表示。グリッドのユーザーは、データ表示のドリルダウン (ドリルアップ)、スライス、ピボットを行えます。 GridBlox は、データをグリッドで表示します。

グループ. 複数の行項目や列項目の見出しを繰り返さないための手段。グループは、データ項目の合計値を自動生成するための手段にもなります。プログラマーは、

SQL の GROUP BY 文節によってリレーショナル・データベース内にグループを定義できます。プログラマーがマルチディメンション・データベース内にグループを定義するときには、データベース階層の定義を使用します。

グループ化. 数値データを表示するときに、そのデータを (通常はコンマやドットによって) 桁数ごとのグループ (1000 単位など) に分けて書式設定すること。

子. 他のディメンション・メンバーよりも 1 つ下の世代のメンバー。例については、『OLAP の用語と概念』の「子」を参照してください。

[サ行]

軸. データの編成と表示のための座標系。 1 つの軸によって、1 つ以上のディメンションを表示できます。マルチディメンション・データの場合、1 つの軸は、キューブの 1 辺としても定義できます。

- 折れ線グラフや棒グラフでは、水平 (X) 軸と垂直 (Y) 軸に基づいてデータを表示します。
- グリッドでは、行、列、ページの軸を使用してマルチディメンション・データを提示します。さらに、DB2 Alphablox では、現在使用されていないディメンションを示す「その他」という軸を表示することもできます。「その他」軸にディメンションを配置すれば、そのディメンションを使用可能な状態にしておきながら、現在のビューからそのディメンションを除去できます。

子孫. ルート (ディメンション階層の最上位) よりも下の世代のディメンション・メンバー。例については、『OLAP の用語と概念』の「子孫」を参照してください。

縮小. 詳細情報を 1 つのサマリー値に統合すること。「拡張」と対比してください。

シン・クライアント. 常駐のアプリケーション・ソフトウェアを持たずに重要なアプリケーション処理を実行するクライアント。ソフトウェアは、必要な時にダウンロードします。 Java 対応の Web ブラウザーは、シン・クライアントです。このブラウザーは、Web ページを表示して、埋め込まれているアプレットを実行できます。「ファット・クライアント」と対比してください。

推奨されない. パラメーターやクラスやメソッドに関して、将来廃止される可能性があることを示す表現。クラスが発展して、そのクラスの API が変化していくにつれ、一貫性を保持するためにパラメーターやメソッドの名前を変更したり、新しいパラメーターやメソッドを追加したりするときに、「推奨されない」API が出てくることになります。アセンブラーが新しい API に移行するまで古い API が残るのはやむを得ませんが、「推奨されない」API を新たに使用するべきではありません。DB2 Alphablox では、「推奨されない」項目が検出されると、クライアント・コンソールにメッセージが書き出されます。

スキーマ. データ・ソースの構造における 1 つの編成レベル。 DB2 OLAP Server や Essbase では、これを「データベース」といいます。一般に、データのキューブをグループ分けしたものがスキーマであり、スキーマをグループ分けしたものがカタログです。

スライス. ページ軸に 1 つ以上のディメンションを配置して、データのセットをフィルターに掛けて抽出すること。例えば、売上データのキューブの行軸に「製品」、列軸に「月」、ページ軸に「年」があるとします。「年」を選択してデータをスライスすれば、その年のデータだけを表示できます。

スレッド. プログラム内でそれぞれ独立して実行できる部分。マルチスレッド化をサポートするオペレーティング・システムを対象とする場合、開発者は、スレッド化した各部分を並行して実行するプログラムを設計できます。

スレッド・セーフ. インターリーブまたはネストによる複数の同時呼び出しを相互干渉なしで実行できるコード、またはある種の相互排他機能によって複数の同時実行から保護されているコード。

層. コンピューティング階層における 1 つのレベル。インターネット/イントラネット環境では、多くの場合、3 層の階層をインプリメントします。つまり、第 1 層は、クライアント・マシン上で実行するブラウザー、第 2 層は、別のマシン上で実行する Web (HTTP) サーバー、第 3 層は、さらに別のマシン上で実行するデータベース・サーバーです。

祖先. 他のディメンション・メンバーよりも上の世代のメンバー。例については、『OLAP の用語と概念』の「祖先」を参照してください。

[タ行]

ダイナミック・アプリケーション・アセンブリー. 明確に定義した動作に基づいて、コンポーネントのセットからアプリケーションを迅速にアセンブルするための手法。

多層アクセス. 「N 層アクセス」を参照してください。

タプル. DB2 Alphablox では、キューブのサブセットを定義するメンバー (各ディメンションから 1 つずつ) のセットを指します。以下のグリッドでは、列軸のタプルに「南部」、「東部」、「西部」が含まれ、行軸のタプルに「VCR 第 1 四半期」、「TV 第 1 四半期」、「TV 第 2 四半期」が含まれています。

		南部	東部	西部
VCR	第 1 四半期	10	20	30
TV	第 1 四半期	20	30	40
TV	第 2 四半期	30	40	50

データウェアハウス. エンタープライズ・レベルのビジネス・システムによって収集したデータの中央リポジトリー。データウェアハウスは通常、エンタープライズ・サーバーに配置します。各種のオンライン・トランザクション処理 (OLTP) アプリケーションからデータを収集し、分析アプリケーションやユーザー照会で使用するためにそのデータを抽出して編成します。

データ項目. マルチディメンション分析で、複数のディメンションの交点に位置する値。

データベース・コネクター. ネットワーク経由でデータベースにアクセスするための共通言語を提供するプログラミング・インターフェース。例えば、Java インプリメンテーション用の JDBC (Java Database Connectivity) や、Microsoft や Apple Macintosh アプリケーション用の ODBC (Open Database Connectivity) があります。

データマート. 分析と表示に関するナレッジ・ワーカーのニーズを満たすために、各種の操作や他のソースから収集したデータのリポジトリー。このデータは、エンタープライズ・レベルのデータベースやデータウェアハウスから抽出できます。基本的に、特定のグループのナレッジ・ワーカーになじみ深い用語でデータを提示します。データマートは、アクセスの容易さや、販売分析などの特定の目的に合わせた使いやすさに重点を置いています。

テーマ. HTML ページに適用できる設計エレメント (フォントやイメージなど) の集合体。複数のページに同じテーマを使用すれば、見た目の統一感を簡単に実現できます。

ディメンション. キューブの階層定義に役立つキューブの構造上の属性。ディメンションのメンバーはすべて、ユーザーから同じようなタイプのデータであると見なされます。例えば、「時間」ディメンションには、「年」、「四半期」、「月」、「週」などのメンバーを組み込めます。さらに、「市場」ディメンションには、「国」、「地域」、「都道府県」、「市区町村」などのメンバーを組み込めます。ディメンションは、マルチディメンション・データ・ソースのデータを編成して、マルチディメンション分析を可能にするするために役立ちます。

ドリルアップ. ディメンション・メンバーの階層を上に向かって移動すること。例えば、ユーザーは、シリコン・バレーの財務データを参照してから、カリフォルニア北部、カリフォルニア、西部諸州、アメリカという具合にドリルアップできます。

ドリルダウン. ディメンション・メンバーの階層を下に向かって移動すること。例えば、ユーザーは、北アメリカの財務データを参照してから、アメリカ、西部諸州、カリフォルニア、カリフォルニア北部、シリコン・バレーという具合にドリルダウンできます。

[ナ行]

認証. 人またはプロセスの身元を確認するプロセス。認証を受けるには、通常、ユーザーが有効な ID とパスワードのペアを入力することが必要です。

ネイティブ SQL. 各データベースに用意されているリレーショナル・データベース・プログラム言語。ネイティブ SQL には通常、そのデータベースのために最適化された機能が含まれています。「SQL」を参照してください。「ODBC SQL」と対比してください。

[ハ行]

ハッシュ・テーブル. キーと値を対応付ける Java データ構造。以下の例は、JavaSoft Java API から取ったものです。最初の例では、数字のハッシュ・テーブルを作成し、アルファベット文字をキーとして使用して、A が 1 に対応し、B が 2 に対応する、といった具合に対応付けを行います。 2 番目の例では、キー (B) を使用して特定の値 (2) を取得します。

```
Hashtable letters = new Hashtable();
numbers.put("A", new Integer(1));
numbers.put("B", new Integer(2));
numbers.put("C", new Integer(3));

Integer n = (Integer)numbers.get("B");
if (n != null) {
    System.out.println("B = " + n);
}
```

ピボット. 1 つの軸から別の軸にディメンションを移動すること。アセンブラーは、Essbase のレポート仕様や JavaScript によってピボットを定義できます。ユーザーは、ユーザー・インターフェースからピボットを実行できます。

ファット・クライアント. クライアント自体にインストールしなければならないアプリケーション・ソフトウェアを使用して、重要な処理を実行するクライアント。「シン・クライアント」と対比してください。

フィルター. 照会から戻されるデータを制限する手段。例えば、フィルターによって、「年」に特定の値が入っているデータだけを照会から戻すように指定できます。

フレーム・セット. ブラウザー・ウィンドウ内の複数の独立制御可能なセクション (フレーム) のセットであり、各セクションに別々の HTML ファイルを表示できます。 1 つのフレームに表示するファイルに、別の (または同じ) フレームに表示するファイルへのリンクを組み込めます。 1 つのフレームにナビゲーションや選択のためのユーザー・ツールを組み込み、別のフレームにユーザー・アクションの結果を表示する、という使い方をよくします。

プロキシー・サーバー. クライアントとサーバーの間に存在するサーバー。プロキシー・サーバーは、サーバーに対するすべての要求をインターセプトして、自身でその要求を処理すべきかどうかを判断します。自身で処理すべきでない要求は、サーバーに転送します。インターネット/イントラネット環境では、頻繁に要求されるページに関する要求をプロキシー・サーバーがインターセプトし、そのページを保存して提供することによって、Web サーバーのパフォーマンスを大幅に改善できます。

ページ軸. マルチディメンション・データを表示するための 1 つの方向。ページでは、マルチディメンション・データベースのスライスを定義します。ユーザーは、ページ軸にディメンションを配置して、そのディメンションのメンバーを選択し、そのメンバーのデータだけを表示できます。例えば、ユーザーは、ページ軸に「市場」ディメンションを配置して、そのディメンションから「ニューヨーク」を選択し、ニューヨークのデータだけを表示できます。

ベクトル. エレメントと呼ばれるオブジェクトの拡張可能な配列。ベクトル内のエレメントの数は、プログラム・ロジックに基づいて動的に増減できます。エレメントの数は、ベクトルのサイズ、つまりベクトルのストレージの最大容量によって定義されます。

[マ行]

マルチ SQL. 複数の SQL ステートメントをチェーンのように結合して、1 つの複雑な照会にしたもの。関連する用語としては、相関照会、副照会、ネスト照会などがあります。

マルチディメンション・データベース. 「OLAP データベース」を参照してください。

ミドルウェア. 異種混合ネットワーク上のアプリケーション同士の対話を管理するソフトウェア。例えば、JDBC ドライバーは、ベンダー独自のインターフェースを使用しないで、プログラムからデータベースにアクセスするためのミドルウェアです。

メタデータ. 他のデータを記述したデータ。データベース・スキーマやデータ・ディクショナリーには、メタデータが含まれています。

[ラ行]

リポジトリー. オブジェクトを識別し、オブジェクトを再利用のために提供する情報データベース。 DB2 Alphablox のリポジトリーには、ユーザー・オブジェクト、アプリケーション・オブジェクト、アクセス・コントロール情報、構成パラメーターを格納します。

ルート. ディメンション階層内の最上位。例については、『OLAP の用語と概念』の「ルート」を参照してください。

レガシー・データベース. 通常、メインフレームに置かれているデータベースであり、多くの場合、ベンダー独自のテクノロジーを使用しています。基本的に、エンタープライズ・レベルの重要な情報ボールトになっています。

列軸. マルチディメンション・データを表示するための 1 つの方向。列軸は、表示可能な 3 つの軸のうちの 1 つです (他の軸は、ページ軸と行軸)。列では、ディメンション・メンバーのデータ値を表示します。ユーザーは、列軸にディメンションを配置して、そのディメンションのメンバーのデータ値を表示できます。例えば、ユーザーは、列軸に「時間」ディメンションを配置して、メンバーの ID (「年」や「四半期」など) と関連データ値 (売上額など) を表示できます。

レポート仕様. DB2 OLAP Server や Hyperion Essbase では、アプリケーションに提供する結果セットの定義を指します。レポート仕様には、データベースから取得するデータを定義したデータ抽出コマンドと、グリッド内のページ軸、列軸、行軸に表示するディメンションを定義した書式設定コマンドが含まれています。

[数字]

2D. 行と列による情報の 2 ディメンション表示のこと。例えば、単純リスト、レポート、2 ディメンション・スプレッドシートなどがあります。

3D. ページ、行、列のディメンションによる情報のマルチディメンション表示のこと。 GridBlox は、そのような表示を実現します。

A

ACL. アクセス・コントロール・リスト。特定のサービスへのアクセスを許可されたユーザーのリストです。

API. アプリケーション・プログラム・インターフェース。プログラマーがソース・コードを記述するときに使用するインターフェースです。 API は、ライブラリー呼び出しの名前と、それぞれのライブラリー呼び出しの引数の数とタイプで構成されています。 DB2 Alphablox の API には、Java と JavaScript に公開されている Blox のメソッドとプロパティーが含まれています。

B

Blox. 各種のアプリケーションを迅速にアセンブルして、イントラネット/インターネット・ベースのエンタープライズ・レベルの分析ソリューションを構築するための再利用可能なソフトウェア・コンポーネント。 Blox をアセンブルして 1 つの完全なアプリケーションを構築する方法については、*開発者用ガイド*を参照してください。各 Blox の詳細については、*開発者用リファレンス*を参照してください。

C

Cascading Style Sheets (CSS). Web ページ開発者が HTML ページの各種エレメントの表示方法を定義するために使用するテクノロジー。ワード・プロセッサー・プログラムで使用するスタイル・シートのようなものです。詳細については、「Cascading Style Sheets: Designing for the Web」(Hakon Wium Lie、Bert Bos 共著、Addison Wesley Longman、1997) または http://www.w3.org/Style/ を参照してください。

CGI. Common Gateway Interface。 Web (HTTP) サーバー上でプログラムを実行するためのプログラミング・インターフェースです。 CGI には、サーバーからサーバーのゲートウェイ・プログラム (実際の処理を行うプログラム) にデータを渡し、ゲートウェイ・プログラムから Web サーバーを経由して要求元のクライアントに結果を戻すための構造が用意されています。「ISAPI」や「NSAPI」と対比してください。

Cookie. クライアント・マシン上に格納されているユーザー・セッションについての情報 (セッション ID など) を含んだオブジェクト。

CSS クラス. エレメントの通常のスタイルに対する拡張やオーバーライドを定義するために使用する CSS テクノロジー。

CSV. コンマ区切り値。各行を改行文字で区切り、行内の各データ値をコンマで区切ったファイル形式です。

D

DAA. 「ダイナミック・アプリケーション・アセンブリー」を参照してください。

DB2 Alphablox アプリケーション. (1) Blox を含んだ JSP ファイル、(2) アプリケーションが使用するデータのアクセス情報、(3) ユーザーにページを表示する方法を定義したプロパティーのコレクション、を組み合わせたもの。アプリケーションは、Web サーバー上に存在し、ユーザー要求への応答としてブラウザー・ウィンドウに表示されます。

DLL. ダイナミック・リンク・ライブラリー。アプリケーションのプロセスとアドレス・スペースの一部として実行するアプリケーション・ファイル。 DLL ファイルは、アプリケーションの開始時にロードされ、必要な限りそのまま保持されるので、DLL を使用することによって、CGI アプリケーションの反復的な取り出しに要する時間を短縮できます。

DSS. 意思決定支援システム。さまざまな情報源を活用して、随時照会、分析、予測をサポートする情報計画システム。

G

GIF イメージ. Graphics Interchange Format。 HTML によってサポートされているグラフィックス形式の 1 つです。

H

HSB. 色調、彩度、輝度の観点から色を記述したカラー・モデル。

HTML. ハイパーテキスト・マークアップ言語。ブラウザー・ウィンドウにテキストを表示する方法を指定するためのテキスト・マークアップ言語です。 HTML では、画面のテキストと、インターネット上の別の場所にある他のテキストやイメージやオブジェクト (Java アプレットなど) を接続できます。

HTML タグ. HTML 文書に埋め込む言語コード。この言語コードによって、文書の内容を表示する方法をブラ

ウザーに対して指定します。 HTML タグには、名前とオプションの属性を設定します。

HTML ページ. 標準的な HTML マークアップ、テキスト、グラフィックス、マルチメディア・オブジェクト、Blox を組み込めるテキスト文書。 HTML ページは、Web ブラウザーで表示します。

HTTP. Hypertext Transfer Protocol。 World Wide Web またはイントラネット上でファイル (テキスト、イメージ、音声、ビデオ、マルチメディア) をやり取りするためのルール・セット。

I

ISAPI. Internet Server Application Program Interface。 Microsoft Internet Information Server に固有の Microsoft Windows プログラム呼び出しのセット。このインターフェースを使用して、CGI (Common Gateway Interface) アプリケーションよりも実行速度の高い Web サーバー・アプリケーションを作成できます。

J

Java アプレット. Java (Sun Microsystems が開発した言語) で記述したプログラム。 Java アプレットを Web ページに埋め込むと、ブラウザーがそのページを表示したときに、アプレットが実行されます。

JavaScript. Web アプリケーションの機能とユーザー・インターフェースを拡張するためのオブジェクト・ベースのスクリプト言語。 Java アプレットと同じように、JavaScript も Web ページの中に埋め込めます。

JDBC. Java Database Connectivity。 Java でリレーショナル・データベースにアクセスするための基本的な API。 JDBC は、構造化照会言語 (SQL) のコール・レベル・インターフェースに基づいています。この機能を使用すれば、ベンダー独自のインターフェースを使用しなくても、SQL 要求によってプログラムからデータベースにアクセスすることが可能になります。それぞれのデータベースごとに用意されているモジュールまたはドライバーが SQL 要求を処理し、その要求を各データベース・システムが認識できる要求に変換します。

JPEG イメージ. Joint Photographic Experts Group。 HTML によってサポートされているグラフィックス形式の 1 つです。ファイル拡張子は 3 文字までという DOS の要件に準拠するために、JPEG ファイルの拡張子は通常、.eps か .jpg になります。

JRE. JavaSoft Java ランタイム環境。この環境を構成しているのは、 Java 仮想マシン、 Java コア・クラス、サポート・ファイルです。この環境は、J2SE Software Development Kit (JDK) のランタイム・コンポーネントでもあります。この JRE は、標準的な Java プラットフォームを構成する実行可能ファイルや他のファイルの最小セットです。

JVM. Java 仮想マシン。 Java コードの解釈と実行を担当するエンティティーです。

M

MDA. マルチディメンション分析。照会、ドリルダウン、スライス、ピボットによって、データベース内をナビゲートしてサブセットを表示し、そのデータのための計算を定義することによって、データを分析する機能。

MIME. Multipurpose Internet Mail Extensions。インターネット・メール標準を使用して、音声、グラフィックス、ビデオ、テキストの各形式のマルチメディア情報をコンピューター・システム間でやり取りするための方式。

N

N 層アクセス. 複数のコンピューティング層をまたいでデータにアクセスするプロセス。単純なクライアント/サーバー・アーキテクチャーでは、クライアントがサーバー上に格納されているデータを要求し、サーバーがクライアントに結果を渡すという、2 層アクセスを使用します。イントラネット・アーキテクチャーでは、多くの場合、クライアント・ブラウザーが Web サーバーにデータを要求し、Web サーバーがその要求をデータベース・サーバーに渡すという、3 層アクセスを使用します。「多層アクセス」ともいいます。

NSAPI. Netscape Application Program Interface。 CGI (Common Gateway Interface) アプリケーションよりも実行速度の高い Web サーバー・アプリケーションを作成するためのインターフェース。

O

ODBC. Open Database Connectivity。どのデータベース管理システム (DBMS) でデータを処理するかにかかわりなく、あらゆるアプリケーションからあらゆるデータにアクセスすることを可能にするデータベース・プログラミング・インターフェース。 ODBC では、アプリケーションと DBMS の間にデータベース・ドライバーと呼ばれる中間層を挿入します。この中間層は、アプリケーションのデータ照会を DBMS が認識できるコマンドに変換します。

ODBC SQL. ネットワーク上のリレーショナル・データベースにアクセスするためのオープンな共通プログラム言語。ほとんどすべてのデータベースに対するアクセスを提供するために、ODBC SQL には、データベースのネイティブ SQL の機能がすべて含まれているわけではありません。「SQL」を参照してください。「ネイティブ SQL」と対比してください。

OLAP データベース. オンライン分析処理データベース。この種のデータベースは、特別な索引作成技法を使用して、サマリー・データの高速照会処理とマルチディメンション・ビューを提供します。 DB2 OLAP Server は、OLAP データベースの一例です。

R

RGB. 光の 3 原色である赤、緑、青という観点から色を記述するカラー・モデル。この 3 つの色を混合することで、他のあらゆる色を作り出すことができます。

S

SQL. 構造化照会言語。リレーショナル・データベース管理システム (RDBMS) にアクセスして、データを取得するためのユーザー・インターフェースを提供する言語です。「ODBC SQL」と「ネイティブ SQL」も参照してください。

T

TCP/IP. 伝送制御プロトコル/インターネット・プロトコル。インターネット上のホスト・マシン同士を接続するために使用する基本的な通信言語 (プロトコル) です。「トランスポート制御プロトコル/インターネット・プロトコル」ともいいます。

U

URL. Uniform Resource Locator。インターネット上のファイルやニュースグループなどのオブジェクトを指定するためのドラフト標準。 HTML 文書では、URL アドレスによってハイパーリンクのターゲットを指定します。

W

Web サーバー. リモート・ブラウザーから送られてきた要求への応答として、HTML ページを送信するサーバー・プロセス。この語は、そのプロセスを実行するマシンを指すこともあります。

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。

〒106-8711
東京都港区六本木 3-2-12
IBM World Trade Asia Corporation
Intellectual Property Law & 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商標または登録商標です。

Intel® および Pentium® は、Intel の米国およびその他の国における商標です。

Microsoft、Windows、Windows NT、および Windows ロゴは、 Microsoft Corporation の米国およびその他の国における商標です。

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。

[ア行]

[カ行]

［サ行］

［タ行］

［ナ行］

［ハ行］

［マ行］

［ヤ行］

［ラ行］

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

W

プログラム番号: 5724-L14

Printed in Japan

SD88-6488-03

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

Spine information:

IBM DB2 Alphablox 管理者用ガイド バージョン 8.4