WebSphere™ Application Server

# WebSphere Application Server 概説

*バージョン 4.0*

SC88-8798-00
(英文原典：SC09-4581-00)

WebSphere™ Application Server

# WebSphere Application Server 概説

*バージョン 4.0*

SC88-8798-00
(英文原典：SC09-4581-00)

**ご注意**

本書の情報およびそれによってサポートされる製品を使用する前に、121ページの『特記事項』に記載する一般情報をお読みください。

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

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

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

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

(URL は、変更になる場合があります)


**原　典：** SC09-4581-00
WebSphere™ Application Server
Getting Started with WebSphere Application Server
Version 4.0

**発　行：** 日本アイ・ビー・エム株式会社

**担　当：** ナショナル・ランゲージ・サポート

第1刷 2001.3


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

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

# 目次


**図 . . . . . vii**

**表 . . . . . ix**

**本書について. . . . . xi**
本書の対象読者 . . . . . xi
本書の編成 . . . . . xi
関連情報 . . . . . xii
本書で使用されている規則 . . . . . xii

**第1章 IBM WebSphere ファミリーの紹介 . . 1**
IBM と e-business . . . . . 1
WebSphere ファミリー: e-business ソリューションの提供. . . . . 2
WebSphere Application Server: お客様の多様なニーズに合わせた 3 つの版 . . . . . 2
どの版を使用すべきか ? . . . . . 4
WebSphere Edge Server. . . . . 4
WebSphere Studio. . . . . 6
VisualAge for Java . . . . . 8
WebSphere Application Server に関する詳しい情報の探しかた . . . . . 9
インストール担当者およびシステム管理者 10
アプリケーション開発者およびシステム構築者. . . . . 10

**第2章 分散コンピューティングと WebSphere Application Server. . . . . 13**
3 層のクライアント / サーバー・コンピューティング . . . . . 13
トランザクション: 分散環境におけるデータ整合性とパフォーマンスの保証 . . . . . 15
セキュリティー: 許可使用だけが行われることの保証 . . . . . 16

**第3章 コンポーネント・テクノロジーの概要 19**
オブジェクト. . . . . 19
構成を用いたオブジェクトの作成 . . . . 20
インターフェース対実装 . . . . . 21
クラス . . . . . 21
継承. . . . . 21
ポリモアフィズム . . . . . 22
コンポーネント . . . . . 23
コンポーネント・モデルおよび関連テクノロジー. . . . . 23
Java 2™ Platform Enterprise Edition (J2EE™) . . . . . 24
JavaBeans コンポーネント . . . . . 26
Enterprise beans . . . . . 27
アプレットおよびサーブレット. . . . . 29
JavaServer Pages. . . . . 33
CORBA . . . . . 33
Microsoft COM . . . . . 35
Managed Object Framework . . . . . 36

**第4章 WebSphere Application Server へのビジネス・モデルの適応. . . . . 39**
コンポーネント・テクノロジー. . . . . 39
共通ソフトウェア・アプローチ. . . . . 40
ソフトウェアの可用性. . . . . 40
ソフトウェアの再利用. . . . . 41
ソフトウェア開発タスクの分割. . . . . 41
ツール・セット . . . . . 42
スケーラビリティー . . . . . 42
オープン・スタンダードおよび投資の活用 . . 42

**第5章 WebSphere Application Server スタンダード版およびアドバンスド版の紹介 . . 45**
Application Servers スタンダード版と Application Servers アドバンスド版の相違点 . 45
Application Server アドバンスド版の紹介 . . 46
Application Server アドバンスド版環境 . . 46
アプリケーション・モデル . . . . . 48
WebSphere プログラミング・モデル・エクステンション. . . . . 49
Application Server アドバンスド版における管理モデル . . . . . 50
管理ツール . . . . . 51
拡張マークアップ言語 (XML) . . . . . 52
Application Server アドバンスド版によって使用されるサービス . . . . . 53
命名サービス. . . . . 53
トランザクション・サービス . . . . . 54

セキュリティー・サービス . . . . . 54
作業負荷管理サービス. . . . . 55
開発環境 . . . . . 55
資料. . . . . 56

**第6章 WebSphere Application Server エンタープライズ版. . . . . 57**
Application Server エンタープライズ版を使用する理由 . . . . . 57
Application Server エンタープライズ版製品で使用される下位レベルのサービス . . . . . 58
分散コンピューティング環境 (DCE) . . . 58
共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) . . . . 60
Component Object Model (COM) . . . . 61
Application Server エンタープライズ版で使用可能なその他のツール. . . . . 62

**第7章 Component Broker の紹介 . . . . 65**
Component Broker のフィーチャー . . . . 65
アプリケーション・アーキテクチャー . . . 68
アプリケーション・アダプター. . . . . 69
オブジェクト・サービス . . . . . 69
並行性制御サービス . . . . . 70
イベント・サービス . . . . . 70
通知サービス. . . . . 70
外部化サービス . . . . . 70
識別サービス. . . . . 70
ライフサイクル・サービス . . . . . 71
命名サービス. . . . . 71
セキュリティー・サービス . . . . . 71
トランザクション・サービス . . . . . 72
セッション・サービス. . . . . 72
照会サービス. . . . . 72
キャッシュ・サービス. . . . . 72
作業負荷管理. . . . . 73
システム管理. . . . . 73
開発ツール . . . . . 75
オブジェクト・ビルダー . . . . . 75

**第8章 TXSeries の紹介 . . . . . 79**
TXSeries CICS . . . . . 79
基本 CICS 概念. . . . . 80
CICS アプリケーション・プログラミング・インターフェース. . . . . 81
リレーショナル・データベース・サポート 82
キュー・サービス . . . . . 82
システム間通信 . . . . . 82
CICS SNA サポート . . . . . 83
ユーザーとの通信 . . . . . 84
CICS Transaction Gateway . . . . . 84
CICS 管理 . . . . . 84
TXSeries Encina . . . . . 85
Encina モニター. . . . . 86
回復可能キューイング・サービス (RQS) . 86
構造化ファイル・サーバー (SFS) . . . . 87
対等通信 (PPC) サービス . . . . . 87
DE-Light ゲートウェイ . . . . . 88
Encina ツールキット . . . . . 89
Encina++ . . . . . 89
Windows プラットフォームで使用可能な Encina ツール . . . . . 90
WebSphere Application Server アドバンスド版とのインターオペラビリティー. . . . . 91

**第9章 トポロジーおよび構成の例. . . . . 93**
クライアントのトポロジー . . . . . 93
シック・クライアント. . . . . 94
シン・クライアント . . . . . 94
シンナー・クライアント . . . . . 95
サーバーのトポロジー. . . . . 96
分散サーバー. . . . . 96
複製サーバー. . . . . 98
統合サーバー. . . . . 98
Application Server スタンダード版のトポロジー . . . . . 99
Application Server アドバンスド版のトポロジー . . . . . 99
単純な構成 . . . . . 99
DMZ 構成 . . . . . 100
TXSeries 構成 . . . . . 101
単純な CICS 構成 . . . . . 101
DCE セル内での単純な CICS 構成 . . . 102
単純な Encina モニター・セル構成 . . . 103
Component Broker 構成 . . . . . 105
単純な構成 . . . . . 105
基本作業負荷管理構成 . . . . . 105

**付録. WebSphere Application Server のライブラリー . . . . . 107**

**特記事項 . . . . . 121**

商標およびサービス・マーク . . . . . 124 **索引 . . . . . 127**

# 図


1. 3 層のクライアント / サーバー・アーキテクチャー . . . . . 14
2. bean では、ビジュアルなプログラミングを行える . . . . . 27
3. EJB アーキテクチャー. . . . . 28
4. アプレット・ベースのビジネス・ソリューション . . . . . 31
5. サーブレット・ベースのビジネス・ソリューション . . . . . 32
6. ORB を使用したクライアント / サーバー通信 . . . . . 34
7. Application Server アドバンスド版環境のコンポーネント . . . . . 47
8. Component Broker システム・マネージャー . . . . . 74
9. Encina のアーキテクチャー . . . . . 86
10. Java アプリケーションと Encina/Encina++ サーバー間のインターオペラビリティー . . . . . 92
11. シック・クライアントには、ローカル・ユーザー・インターフェースおよびリモート・ビジネス論理が割り当てられる . 94
12. シン・クライアントには、アプレット・ユーザー・インターフェースおよびリモート・ビジネス論理が割り当てられる . 95
13. サーブレットと一緒に使用するシンナー・クライアント . . . . . 96
14. リソース・マネージャーに接続された分散サーバーのクラスター . . . . . 97
15. 中間層とバックエンド層の物理的な統合 98
16. Application Server スタンダード版の単純な構成 . . . . . 99
17. Application Server アドバンスド版の単純な構成 . . . . . 100
18. Application Server アドバンスド版での DMZ 構成 . . . . . 101
19. 非 DCE セル環境で CICS を使用する単純な分散構成. . . . . 102
20. DCE セル環境で CICS を使用する分散構成 . . . . . 103
21. 単純な Encina Monitor 構成 . . . . 104
22. Component Broker の基本トポロジー 105
23. Component Broker の水平作業負荷管理トポロジー . . . . . 106

# 表

1. 本書で使用する規則 . . . . . xii
2. WebSphere Application Server の版の選択 4
3. Encina++ のコンポーネント . . . . . 90
4. WebSphere Application Server のライブラリー. . . . . 107

# 本書について

本書では、IBM® WebSphere™ Application Server を初めて使用する場合に役立つ情報を紹介します。本書を読めば、この製品を構成するコンポーネントをよく理解することができます。本書は WebSphere Application Server エンタープライズ版を中心に説明していますが、 WebSphere Application Server スタンダード版および WebSphere Application Server アドバンスド版に関する情報も含まれています。

## 本書の対象読者

本書は、インストール担当者、システム管理者、開発者、システム設計者、および WebSphere Application Server に関する知識が必要なその他の情報技術専門家が使用することを前提としています。本書を読むためには、分散コンピューティングおよび Web コンピューティングについての最低限の知識が必要です。

## 本書の編成

本書は以下のように編成されています。

- 1ページの『第1章 IBM WebSphere ファミリーの紹介』では、IBM e-business 戦略の高水準の概要を示し、WebSphere Application Server および関連製品によってこの戦略がどのように実現されるのかを説明します。
- 13ページの『第2章 分散コンピューティングと WebSphere Application Server』では、分散コンピューティング、トランザクション、およびセキュリティーについて説明します。
- 19ページの『第3章 コンポーネント・テクノロジーの概要』では、WebSphere Application Server で使用する各種コンポーネント・モデルおよび関連するテクノロジーについて説明します。
- 39ページの『第4章 WebSphere Application Server へのビジネス・モデルの適応』では、組織における WebSphere Application Server の重要性を説明します。
- 45ページの『第5章 WebSphere Application Server スタンダード版およびアドバンスド版の紹介』では、Application Server アドバンスド版および Application Server スタンダード版について、前者に主眼を置いて、主要コンポーネントと概念の説明を行います。

- 57ページの『第6章 WebSphere Application Server エンタープライズ版』では、Application Server エンタープライズ版および関連概念について説明します。
- 65ページの『第7章 Component Broker の紹介』では、Component Broker について詳しく説明します。
- 79ページの『第8章 TXSeries の紹介』では、TXSeries™ について詳しく説明します。
- 93ページの『第9章 トポロジーおよび構成の例』では、WebSphere Application Server システム構成の例について説明します。
- 107ページの『付録. WebSphere Application Server のライブラリー』では、WebSphere Application Server で利用可能な資料の全リストを紹介します。

## 関連情報

本書で述べられているトピックおよびソフトウェアの詳細については、下記の資料を参照してください。

- *WebSphere ビジネス構築のソリューション*
- *CICS アプリケーション・プログラミング・ガイド*
- Component Broker *アプリケーション開発ツール・ガイド*
- Component Broker *計画、パフォーマンスおよびインストール・ガイド*
- Component Broker *プログラミング・ガイド*
- TXSeries *計画およびインストール・ガイド*
- *Encina アプリケーションの作成*
- *Windows 環境での Encina アプリケーションの作成*
- *Enterprise Beans の作成*

## 本書で使用されている規則

WebSphere Application Server エンタープライズ版の資料では、次の表記上の規則とキー入力の規則が使用されています。

*表 1. 本書で使用する規則*

| 規則 | 意味 |
|---|---|
| **太字** | コマンド名を示します。グラフィカル・ユーザー・インターフェース (GUI) については、メニュー、メニュー項目、ラベル、およびボタンも示します。 |

*表 1. 本書で使用する規則 (続き)*

| 規則 | 意味 |
|---|---|
| モノスペース | コマンド・プロンプトに記述どおりに入力しなければならないテキストと、記述どおりに使用しなければならない値を示します。コマンド、関数、およびリソース定義属性とその値などが含まれます。また、画面上のテキストやコードの例も示します。 |
| *イタリック* | 自分で指定しなければならない可変値を示します (たとえば、 *fileName* の部分にはファイル名を指定する必要があります)。また、強調目的の場合や資料の名称を示す場合もあります。 |
| Ctrl-*x* | 制御文字シーケンスを示します (*x* はキーの名前)。たとえば、Ctrl-c は、 Ctrl キーを押さえたままで c キーを押すことを示します。 |
| Return | Return、Enter、または左矢印が書かれたキーを表します。 |
| % | **root** 特権を必要としないコマンドの UNIX コマンド・シェル・プロンプトを表します。 |
| # | **root** 特権を必要とするコマンドの UNIX コマンド・シェル・プロンプトを表します。 |
| C:¥> | Windows NT® のコマンド・プロンプトを表します。 |
| **>** | メニューの説明で使用されている場合は、一連のメニュー選択項目を示します。たとえば、「**「ファイル (File)」> 「新規作成 (New)」**をクリックします。」は、**「ファイル (File)」**メニューの**「新規作成 (New)」**コマンドをクリックするという意味です。 |
| | ツリー・ビューの説明で使用されている場合は、一連のフォルダーまたはオブジェクトの展開を示します。たとえば、「**「管理ゾーン (Management Zones)」> 「サンプル・セルおよびワークグループ・ゾーン (Sample Cell and Work Group Zone)」> 「構成 (Configuration)」**を展開します。」には、次のような意味があります。<br>1. 「管理ゾーン (Management Zones)」フォルダーを展開します。<br>2. 「サンプル・セルおよびワークグループ・ゾーン (Sample Cell and Work Group Zone)」という名前の管理ゾーンを展開します。<br>3. 「構成 (Configurations)」フォルダーを展開します。<br>**注:** 画面中のオブジェクトの隣に正符号 (**+**) があると、そのオブジェクトを展開できます。オブジェクトを展開すると、正符号が負符号 (**-**) に置き換えられます。 |
| **+** | ツリー構造を展開すると、表示されるオブジェクトの数を多くできます。展開するには、オブジェクトの隣にある正符号 (**+**) をクリックします。 |
| **-** | ツリー構造の分岐を縮小して、その分岐に含まれるオブジェクトが表示されないようにします。ツリー構造の分岐を縮小するには、分岐の先頭のオブジェクトの隣にある負符号 (**-**) をクリックします。 |

*表 1. 本書で使用する規則 (続き)*

| 規則 | 意味 |
|---|---|
| コマンドの入力 | コマンドを「入力する」または「実行する」ように指示される場合には、コマンドをタイプしてから Return を押します。たとえば、「**ls** コマンドを入力する。」という指示があった場合には、 **ls** をコマンド・プロンプトにタイプしてから Return を押します。 |
| [ ] | 構文記述中のオプション項目を囲みます。 |
| { } | 構文記述中で必須選択項目のリストを囲みます。 |
| \| | 構文記述中の中括弧 (**{ }**) で囲まれた選択項目のリスト中の項目を分離します。 |
| ... | 省略記号。構文記述中の省略記号は、前述の項目を 1 回以上繰り返せることを示します。例の中の省略記号は、簡潔にするためにその情報が例から省略されたことを示します。 |
| IN | 関数の説明の中で、データを関数に渡す場合、値が使用されるパラメーターを示します。このパラメーターは、変更されたデータを呼び出し側のルーチンに戻す場合には使用しません。 (作成するコードの中には IN 宣言を含めないでください。) |
| OUT | 関数の説明の中で、変更されたデータを呼び出し側のルーチンに戻す場合、値が使用されるパラメーターを示します。このパラメーターは、データを関数に渡す場合には使用しません。 (作成するコードの中には OUT 宣言を含めないでください。) |
| INOUT | 関数の説明の中で、値が関数に渡され、関数によって変更された後、呼び出し側のルーチンに戻されるパラメーターを示します。このパラメーターは、IN パラメーターとしてかつ OUT パラメーターとして機能します。 (作成するコードの中には INOUT 宣言を含めないでください。) |
| $CICS | CICS プロダクトのインストール先の絶対パス名を示します。たとえば、Windows NT の場合は C:¥opt¥cics、 Solaris の場合は /opt/cics です。環境変数 CICS がプロダクトのパス名に設定されている場合は、記載されているとおりに例を使用することができます。そうでない場合、$CICS 部分はすべて CICS のプロダクトのパス名に置き換えてください。 |
| CICS on Open Systems | サポートされているすべての UNIX プラットフォーム用の CICS プロダクトを指します。 |
| TXSeries CICS | CICS for AIX、CICS for Solaris、および CICS for Windows NT を指します。 |

*表 1. 本書で使用する規則 (続き)*

| 規則 | 意味 |
| --- | --- |
| CICS | CICS on Open Systems プロダクトと CICS for Windows NT プロダクトの総称です。複数の CICS on Open Systems プロダクト間の違いを強調する場合は、特定バージョンの CICS on Open Systems プロダクトに言及します。CICS ファミリー中の他の CICS プロダクトは、オペレーティング・システムによって区別されます。たとえば、CICS for OS/2 や、 ESA、MVS、および VSE プラットフォーム用の IBM メインフレーム・ベース CICS という具合です。 |

# 第1章 IBM WebSphere ファミリーの紹介

この章では、IBM による e-business への取り組み方を検証し、 WebSphere ファミリーの製品によってユーザーの e-business の問題を解決する方法を説明します。また、WebSphere ファミリーで使用可能なその他のツールのうちのいくつかを取り上げ、それらが IBM の全体的なアプローチにどのように適合するのかを示します。この章の最後のセクションでは、WebSphere ファミリーの製品に関する詳しい情報が記載されている資料を紹介します。

## IBM と e-business

World Wide Web (Web) の人気は、個人の間でも企業の間でも急速に高まってきました。個人が Web を使用する場合、その目的はさまざまですが、企業が Web を使用する目的は、主として、顧客、サプライヤー、戦略的パートナー、および従業員に製品、サービス、および情報を提供することを目的としています。

初めて企業が Web に進出したころには、少数の静的な Web ページを用意し、販売する製品とサービスを列挙し、それらの製品およびサービスを注文するための電話番号および住所を表示するだけで十分でした。情報サービスを提供する企業 (ソフトウェア会社など) は、この新規分野におけるパイオニアに属し、情報およびソフトウェア・プロダクトをダウンロードできるようにしました。

新しいテクノロジーが開発されると、静的な Web ページでは不十分になりました。それに応じて、企業はアクティブな Web サイトを構築し、顧客が製品を直接注文したり、顧客やサプライヤーが企業と連絡を取ったり、従業員が相互に連絡を取り合ったりできるようにしました。

多くの企業の Web サイトが急速に変化する一方で、非 Web のビジネス情報システムも、アプリケーション開発がメインフレーム・システムから分散システムに広がるにつれて、大きな変化を遂げました。 Open Group の分散コンピューティング環境 (DCE) および Object Management Group (OMG) の共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) は、これらのタイプのシステムの基盤を提供する、2 つの主要なテクノロジーでした。

最近までは、Web ビジネス情報システムと非 Web ビジネス情報システムの多くの部分が、相互に切り離されたままになっていました。これを変えたのが、

IBM e-business イニシアチブおよび WebSphere ファミリーです。 WebSphere は、非 Web ベースのシステムを Web ベースのシステムに統合し、企業全体の単一ビジネス・システムを形成できるようになりました。

## WebSphere ファミリー: e-business ソリューションの提供

IBM WebSphere ファミリーは、ユーザーが e-business の約束を実現する手助けをするために設計されたものです。 IBM WebSphere は、お客様が高性能な Web サイトを開発および管理し、それらの Web サイトを新規または既存の非 Web ビジネス情報システムと統合できるようにする、ソフトウェア・プロダクトの集まりです。これは、以下の一般的なタイプの企業に焦点を当てています。

- 最新テクノロジーを使用して、強大な Web を構築したり、現行の Web をアップグレードしたい企業
- 分散型の全社的なビジネス・システムおよびアプリケーションを開発したい企業
- Web を既存のコンピューター・システムおよびアプリケーションと統合したい企業

WebSphere ファミリーは、WebSphere Application Server、および WebSphere Application Server と密接に統合されて、そのパフォーマンスを向上させるその他の WebSphere ファミリー・ソフトウェアから成り立ちます。

WebSphere Application Server は、Sun Microsystem の Java™ 2 および Enterprise Edition (J2EE™) 標準と完全な互換性があり、 CORBA、XA、SSL (Secure Sockets Layer)、Kerberos、論理装置 (LU) 6.2、 LDAP (Lightweight Directory Access Protocol) などのその他の一般的な業界標準をサポートしています。

## WebSphere Application Server: お客様の多様なニーズに合わせた 3 つの版

e-business の目標を達成できるようにするために、WebSphere では 次の 3 つの版を提供しています。

- WebSphere Application Server *スタンダード版* (Standard Application Server とも呼ばれます) は、サーバー側のビジネス・アプリケーションの移植性と、Java テクノロジーのパフォーマンスおよび管理容易性を結合させて、Java ベースの Web アプリケーションを設計するための包括的なプラットフォームを提供します。 Sun Microsystems の Enterprise JavaBeans™ 仕様に従

って構築されたアプリケーション用のサーバー機能が組み込まれています。また、Web アプリケーションが企業データベースとトランザクション・システムとの対話を可能にします。

- WebSphere Application Server *アドバンスド版* (Advanced Application Server とも呼ばれます) は、 Application Server スタンダード版をベースとしています。分散サーバー環境をサポートし、分散サーバー環境を管理するための基本サービスが組み込まれています。また、非 Web ビジネス・システムとのより密接な統合を提供します。
- WebSphere Application Server *エンタープライズ版* (Application Server エンタープライズ版とも呼ばれます) は、 Application Server アドバンスド版をベースとして、機能を社内全体の情報システムに拡張します。 IBM の世界規模のトランザクション・アプリケーション環境である TXSeries™ (Encina® と CICS® の両方から成り立ちます) と、 Component Broker の完全分散オブジェクトおよびビジネス・プロセス統合機能を組み合わせたものです。Application Server エンタープライズ版には、Application Server アドバンスド版の完全バージョンが含まれています。

3 つの版はすべて、IBM AIX®、Sun Microsystems Solaris、および Microsoft® Windows NT® のプラットフォームで利用できます。 WebSphere スタンダード版およびアドバンスド版は、 Linux、IBM AS/400®、および Novell Netware のプラットフォームで利用できます。 WebSphere スタンダード版およびアドバンスド版と、エンタープライズ版の TXSeries コンポーネントは、Hewlett-Packard HP-UX プラットフォームで利用できます。サポートされているプラットフォームの完全なリストについては、 WebSphere Application Server の Web サイト (www.ibm.com/software/webservers/appserv) を参照してください。

WebSphere Application Server for OS/390® は、スタンダード版およびエンタープライズ版が用意されています。このプラットフォームでは、スタンダード版に IBM JDK および WebSphere Site Analyzer が添付されています。また、Common Connector Framework を介する OS/390 プラットフォーム上の CICS および IMS™ への接続もサポートしています。エンタープライズ版は、スタンダード版の製品と OS/390 向けの Component Broker から成り立ちます。

WebSphere Application Server は、主に OS/390 プラットフォーム上の実行時環境です。アプリケーションを作成するには、VisualAge™ for Java や Component Broker Object Development Toolkit などの、 Windows および UNIX 対応の WebSphere 開発ツールを使用することができます。

## どの版を使用すべきか ?

WebSphere Application Server の 3 つの版は、さまざまなタイプの顧客のニーズをサポートすることを目的としています。 表2 に、どの版が組織のニーズに合うかについての推奨事項を紹介します。

*表 2. WebSphere Application Server の版の選択*

| 版 | 機能 | 推奨 |
|---|---|---|
| スタンダード | • 単一マシン・アプリケーション・サーバー<br>• JSP™ ページ、サーブレット、Enterprise beans をサポート<br>• Web サイト開発ツールを提供 | トラフィックが比較的低く、単一のアプリケーション・サーバーで取りまとめられる Web サイト。小規模の組織向き。 |
| アドバンスド | • マルチマシン・アプリケーション・サーバー<br>• JSP ページ、サーブレット、Enterprise beans をサポート<br>• 作業負荷管理、高度なセキュリティー、Web サイトおよび Java 開発ツール、バックエンド・リソースへの適切なアクセスを提供 | トラフィックが中〜高で、複数のアプリケーション・サーバーで取りまとめられる Web サイト。中規模〜大規模の組織向き。 |
| エンタープライズ | Application Server アドバンスド版および Component Broker と TXSeries の全機能 | Web サーバーに統合する強力で全社的なトランザクション・サーバーを必要とする組織 |

## WebSphere Edge Server

IBM WebSphere Edge Server には、Web サーバーの輻輳 (ふくそう) を軽減し、コンテンツの可用性を高め、 Web サーバーのパフォーマンスを向上させるための、ソフトウェア・ツールが組み込まれています。この目的は、インターネットまたは社内イントラネットを介して Web ベースのコンテンツにアクセスするユーザーに、より良いサービスを提供することです。 Edge Server という名前から分かるように、ソフトウェアは通常、社内イントラネットとインターネット間の境界に (ネットワーク構成の意味で) 近いマシン上で実行されます。

Edge Server と WebSphere Application Server を一緒に使用することで、 Web サーバーへのクライアント・アクセスを制御することができます。これには、

Caching Proxy と Network Dispatcher という 2 つのコンポーネントがあります。次に、これらのコンポーネントについて説明します。

### Caching Proxy

Caching Proxy コンポーネントは、エンド・ユーザーからのデータ要求を代行受信し、コンテンツを取りまとめているマシンから要求された情報を取得して、エンド・ユーザーに戻します。一般的に、要求とは、Web サーバー・マシン (オリジン・サーバーまたはコンテンツ・ホストとも呼ばれます) に格納されているドキュメントの情報を得ることが目的で、 HTTP を介して転送されます。ただし、Caching Proxy は、ファイル転送プロトコル (FTP) などの他のプロトコルを処理するように構成することもできます。

特定のタイプのコンテンツを取得すると、 Caching Proxy は、要求側に配布する前に、ローカル・キャッシュにそのコンテンツを格納します。キャッシュに格納できるコンテンツの最も典型的な例が、静的 Web ページ (アクセス時に動的に生成される部分を持たないページ) です。キャッシュに格納することにより、Caching Proxy は続けて同じコンテンツに対する要求があったときに、キャッシュから直接配布できるため、コンテンツ・ホストから再度取得するよりも時間を大幅に短縮することができます。

Caching Proxy は、Web アクセス可能コンテンツを取りまとめる場合、およびインターネット・アクセスを提供する場合の両方で役立ちます。

- コンテンツ・ホストで使用する場合、Caching Proxy は、インターネットと社内のコンテンツ・ホスト間の逆方向プロキシーとしてインストールします。プロキシーは、インターネットから来るユーザー要求を代行受信して、適切なコンテンツ・ホストに転送し、戻されたデータをキャッシュに格納して、インターネットを介してユーザーに配布します。
- インターネット・アクセス・プロバイダーで使用する場合、 Caching Proxy は、社内のエンド・ユーザーとインターネット間の順方向プロキシーとしてインストールします。プロキシーは、ユーザーの要求をインターネット上のコンテンツ・ホストに転送し、取得したデータをキャッシュに格納して、ユーザーに配布します。

Caching Proxy は、WebSphere の以前のバージョンでは Web Traffic Express と呼ばれていました。

### Network Dispatcher

Network Dispatcher も、エンド・ユーザーからのデータ要求を代行受信しますが、実際にデータを取得するのではなく、その要求を現時点で最も満たすこと

ができるサーバー・マシンに転送します。言い換えれば、同じタイプの要求を処理するあらかじめ定義されたマシン間で着信要求を分散します。

Network Dispatcher は、HTTP オリジン・サーバーと Caching Proxy マシンの両方を含め、複数のサーバーに要求を分散することができます。必要に応じて、どのサーバーが要求を最も適切に処理できるかを判断する場合に Network Dispatcher が使用する基準を指定する規則を作成することもできます。

Caching Proxy と同様に、Network Dispatcher は Web アクセス可能コンテンツを取りまとめる場合、およびインターネット・アクセスを提供する場合の両方で役立ちます。

- コンテンツ・ホストで使用する場合、 Network Dispatcher はインターネットと社内のバックエンド・サーバー (POP3 プロトコルまたは IMAP プロトコルを処理するコンテンツ・ホスト、 Caching Proxy マシン、またはメール・サーバー・マシン) 間にインストールします。 Network Dispatcher は、需要が高かったり、コンテンツの容量が大きいために、企業が複数のバックエンド・サーバーを使用している場合でも、インターネット上ではその企業で単一のサーバーと見なされます。
- Network Dispatcher の Content Based Routing (CBR) モジュールが Caching Proxy と一緒にインストールされている場合は、 HTTP 要求は URL などの管理者が指定した特性に基づいて分散することもできるため、同じコンテンツをすべてのバックエンド・サーバーに格納する必要がなくなります。
- インターネット・アクセス・プロバイダーで使用する場合、 Network Dispatcher は、社内のエンド・ユーザーと、社内イントラネット内の複数の Caching Proxy マシンの間にインストールして、マシン間での負荷を分散させます。複数の Caching Proxy マシン間で負荷を分散すれば、需要が高くなった場合でもインターネットへのアクセスを確実に保証することができます。また、バックアップの Network Dispatcher をインストールして、主要なもので一時的に障害が発生した場合に機能を引き継ぐようにしておけば、高い可用性を保証することもできます。

### WebSphere Studio

WebSphere Studio は、Web サイト開発のすべての局面を共通インターフェースに組み込むツール・セットです。コンテンツ作成者、グラフィック・アーティスト、プログラマー、および Webmaster のすべてが、各自に必要なファイルにそれぞれアクセスしながら、同じプロジェクトの作業を行うことができます。 WebSphere Studio を使用したほうが、連携して動的な対話式 Web アプリケーションを簡単に作成、アセンブル、発行、および保守することができます。

WebSphere Studio は、Workbench、Page Designer、Remote Debugger、およびウィザードからなり、付属の Web 開発製品と一緒に提供されます。 WebSphere Studio を使用すると、以下のことをはじめとする、進んだビジネス機能をサポートする対話式 Web サイトの作成に必要なことが、すべて行えます。

- Studio のウィザードを使用して、Java bean、データベース照会、および Java サーブレットを作成する。プログラマー以外の人でも、ユーザーの Web サイトに強力な機能を追加する、サーバー側論理を生成することができます。このウィザードを使用すると、入力フォーム、出力ページ、およびそれらすべてを機能させる Java コードを簡単に作成することができます。ユーザーの Web ページは、ただそこに収まっているだけではなく、ビジネスのために活躍してくれるようになります。
- Web サイト・ファイルをプロジェクトおよびフォルダーにグループ化する。どのように編成するのが良いか決めて、それに従ってファイルをグループ化してください。フィルターおよびグローバル・サーチ機能を使用して、必要なファイルを検出することができます。振る舞いが予測できる、使い慣れたオブジェクトを使用して、日常作業の速度と効率を高めることができます。
- ファイルを個別に、あるいは共用バージョン管理システムで保守する。ファイルは、ローカルで、独自のワークステーションで保管することも、ネットワーク内の他のシステムで保管することも、全機能のバージョン管理システム (VCS) で保管することもできます。ユーザーは安心して共同作業を行うことができ、ファイル保全性が確保されます。
- 好きなツールを使用してファイルを編集および更新する。ファイル・タイプごとに、使用する編集プログラムと表示プログラムを選択することができます。 Studio ファイルをオープンするときに、デフォルト選択をすぐに立ち上げたり、代替ツールを選択したりすることができます。
- ファイル関連に高速にアクセスし、中断したリンクを検出する。関連ビューにより、ユーザーのファイルが相互にどのようにリンクされているのかを示すビジュアル表示が行われます。特定のファイルに他のファイルがいくつリンクしているのか、どのファイルのリンクが中断しているのか、またどのファイルがまったくリンクされていないのかを、迅速に調べることができます。
- どの開発段階でも Web サイトを発行できる。Studio Workbench 内で、サイト開発からサイト発行に直接進むことができます。開発の任意の段階で Web サイトの全部または一部を発行し、作業の結果をすぐに調べてテストすることができます。Web サイトをリリースする準備が整ってから、ファイルを簡単に最終段階に転送し、実動サーバーに発行することができます。

Studio Workbench は、ユーザーの Web サイトのアプリケーションやファイルを管理および保守するのに役立ちます。次の機能を提供します。

- あるプロジェクト内のファイル間のリンク関連をグラフィカル表示する。
- ファイルの変更または移動時に、リンクを自動的に更新する。
- 複数のオーサリング・ツールを登録し、Web サイト・ファイルを編集するたびに選択を行えるようにする。
- Web サイトの実動サイクルのステージを設定し、各種のステージを異なる (そして複数の) サーバーに向けて発行する。
- 既存サイト・コンテンツの Studio プロジェクトへの直接転送を単純化するインポート・ウィザード。
- Web サイトまたはサブサイトを単一ファイルにすぐにアーカイブする方法。
- サード・パーティー製のツールを Workbench 環境に簡単に統合する機能。
- IBM VisualAge Team Connection®、Microsoft SourceSafe®、PVCS、Rational ClearCase、および Lotus® Domino™ など、人気の高いソース制御管理ソフトウェアの統合による、進行中の作業を表示する共通ビューを備えた拡張チーム環境。

Studio Page Designer は、JavaServer Pages (JSP)、Java サーブレット、およびその他の Java ベースの Web ツールを作成できるようにするための、ビジュアル設計環境を備えています。たとえば、このビジュアル環境を使用して、Java bean を JSP アプリケーションにドラッグ・アンド・ドロップすることができます。Studio Page Designer は、DHTML および HTML ページの作成に使用することもでき、HTML または DHTML ソースとブラウザー・ビューを簡単に編集し、切り替えるための機能を備えています。この機能は、新しい IBM HomePage Builder 製品に基づいています。

Studio Remote Debugger は、Studio 環境で JSP ファイルおよび Java サーブレットのソース・レベル・デバッグを行います。リモート・デバッグは、WebSphere Application Server 3.0 またはそれ以上がインストールされている任意のマシンで行うことができます。

## VisualAge for Java

VisualAge for Java は、Java プログラム開発の完全なサイクルをサポートする統合開発環境です。VisualAge for Java は、WebSphere Application Server と密接に統合されています。この統合により、VisualAge 開発者は VisualAge を終了しないで Java プログラムを開発、配置、およびテストすることができます。また、企業環境に共通の複雑さを管理して、定形ステップを自動化することのできる環境を開発者に提供します。

VisualAge for Java のビジュアル・プログラミング・フィーチャーを使用して、Java のアプレットおよびアプリケーションを短時間で開発することができます。ビジュアル・コンポジション・エディターでは、ポイント・アンド・クリックによって以下の作業を行うことができます。

- ユーザーのプログラム向けのユーザー・インターフェースの設計
- ユーザー・インターフェースエレメントの振る舞いの指定
- ユーザー・インターフェースとプログラムの他の要素との間の関連の定義

VisualAge for Java は、ユーザーがビジュアル・コンポジション・エディターでビジュアルに指定したものを実装するための、Java コードを生成します。多くの場合、Java コードをまったく書かずに、完全なプログラムを設計して実行することができます。

VisualAge for Java は、ビジュアル・プログラミング・フィーチャーのほかに、アプレット、サーブレット、アプリケーション、JavaBeans コンポーネント、および Enterprise JavaBeans (EJB) 仕様に従って作成された Enterprise beans の作成を含め、多くの作業を迅速に行えるように案内する SmartGuides も備えています。また、基礎になるファイル・システムの要求に応じて、既存のコードをインポートしたり、コードをエクスポートしたりすることもできます。

VisualAge for Java は、業界最先端のコードを開発するために必要なプログラミング・ツールを提供します。特に、以下のことが行えます。

- 完全に統合されたビジュアル・デバッガーを使用して、実行中のコードを検査および更新する。
- Java bean を構築、変更、および使用し、また Enterprise beans を構築、変更、配置、およびテストする。
- プロジェクト、パッケージ、クラス、またはメソッドのレベルでコードをブラウズする。

VisualAge for Java には、プログラムの複数の版の更新を簡易化する、コード管理システムも備わっています。

## WebSphere Application Server に関する詳しい情報の探しかた

この資料の残りの部分では、WebSphere Application Server の 3 つの版について説明します。また、このセクションの残りの部分に記載された文書を読むことによって、詳しい情報を得ることができます。

すべての WebSphere 資料は、WebSphere Application Server のライブラリー・ページ (www.software.ibm.com/webservers/appserv/library.html) から入手できます。これらの資料は、ハイパーテキスト・マークアップ言語 (HTML) ファイル版と Adobe PDF ファイル版が用意されています。

### インストール担当者およびシステム管理者

特定バージョンの WebSphere Application Server をインストール、構成、または管理する必要がある場合には、以下のうちの 1 つまたは複数をお読みください。

- Application Server スタンダード版およびアドバンスド版のインストールおよび構成の基本を学習したい場合は、インストールしたい製品の「インフォセンター」を参照してください。このオンライン文書センターは、システムを短時間で簡単に稼働および実行したいとお考えになっている、初心者ユーザー向けに書かれています。どちらの版の「インフォセンター」とも、WebSphere Application Server のライブラリー・ページ (www.software.ibm.com/webservers/appserv/library.html) から入手できます。
- Application Server スタンダード版およびアドバンスド版の管理を学習したい場合は、次の資料を参照してください。
  - WebSphere Administrative Console 内で利用できるオンライン・ヘルプ。
  - スタンダード版またはアドバンスド版「インフォセンター」のオンライン資料。
- Application Server エンタープライズ版と共にシステムをインストール、構成、および管理する方法を学習したい場合は、最初に Component Broker 用の「*計画、パフォーマンスおよびインストール・ガイド*」または TXSeries 用の「*計画およびインストール・ガイド*」を参照してください。

### アプリケーション開発者およびシステム構築者

特定バージョンの WebSphere Application Server を使用してビジネス・システムを設計したり、アプリケーションを開発したりする必要がある場合には、以下の文書のうちの 1 つまたは複数をお読みください。

- Application Server スタンダード版およびアドバンスド版でアプリケーションを計画、開発、およびトラブルシューティングする方法を学習したい場合は、 WebSphere Application Server のライブラリー・ページ (www.software.ibm.com/webservers/appserv/library.html) から入手できる、スタンダード版インフォセンターおよびアドバンスド版インフォセンターを参照してください。
- Sun Microsystems Enterprise JavaBeans 仕様に従った Enterprise beans および関連するコンポーネント開発の基本を学習したい場合は、最初に「*Enterprise*

*Beans の作成*」を参照してください。この文書には、Application Server アドバンスド版と Application Server エンタープライズ版の両方で Enterprise beans を開発するための手順が示されています。

- WebSphere ファミリーでのシステムおよびアプリケーションの設計と開発に関連する広範な問題について学習したい場合は、「*WebSphere ビジネス構築のソリューション*」を参照してください。「*WebSphere Application Server 概説*」に記載されているテクノロジー、問題、トポロジーの多くは、「*WebSphere ビジネス構築のソリューション*」で詳しく説明されています。
- Component Broker でのアプリケーションの開発について学習したい場合は、最初に「*アプリケーション開発ツール・ガイド*」を読み、次に Component Broker の「*プログラミング・ガイド*」を参照してください。 OS/390 のプログラマーは、「*WebSphere Application Server エンタープライズ版 for OS/390 Component Broker アプリケーション・アセンブル・ガイド*」も参照してください。
- TXSeries CICS でのアプリケーション開発について学習したい場合は、まず「*CICS アプリケーション・プログラミング・ガイド*」を参照してください。
- TXSeries Encina でのアプリケーション開発について学習したい場合は、まず「*Encina アプリケーションの作成*」(一般的な開発向け) または「*Windows 環境での Encina アプリケーションの作成*」(Microsoft Windows プラットフォームでの開発向け) を参照してください。

Application Server エンタープライズ版に関して使用可能な資料の完全なリストが 107ページの『付録. WebSphere Application Server のライブラリー』に示されていますので、参照してください。

# 第2章 分散コンピューティングと WebSphere Application Server

WebSphere Application Server は、オープンな分散コンピューティングのための環境を提供します。WebSphere によって提供される機能を使用して、さまざまなプラットフォーム上のユーザーおよびプロセスが相互作用することができます。 Application Server アドバンスド版と Application Server エンタープライズ版は、ともに分散コンピューティング環境を提供します。

このセクションでは、分散コンピューティング、トランザクション処理、およびセキュリティーに関係する基本概念の概要を紹介します。これらの概念をよくご存じの場合は、19ページの『第3章 コンポーネント・テクノロジーの概要』 または 39ページの『第4章 WebSphere Application Server へのビジネス・モデルの適応』 に進むことができます。

## 3 層のクライアント / サーバー・コンピューティング

分散システムで稼働するソフトウェアを編成するための共通の方法として、機能をクライアントとサーバーの 2 つの部分に分けることができます。クライアント は、サーバー と呼ばれる他のプログラムによって提供されるサービスを使用するプログラムです。クライアントがサービスを要求し、サーバーがそのサービスを実行します。サーバー機能は、何らかの種類のリソース管理を伴うことが多く、その場合、サーバーがリソースへのアクセスを同期化および管理し、クライアント要求に対してデータまたは状況情報で応答します。クライアント・プログラムは、一般に、ユーザー対話を処理し、ユーザーのためにデータを要求するか、あるいは何らかのデータ変更を開始します。

たとえば、クライアントは、ユーザー (Web ブラウザーを使用している人など) が製品の発注を入力するために使用できる、フォームを提供することができます。クライアントはこの発注情報をサーバーに送信し、サーバーは製品データベースを調べて、発送および請求に必要な作業を実行します。一般には、1 つのサーバーが複数のクライアントによって使用されます。たとえば、数 10 または数 100 のクライアントが、データベース・アクセスを制御する数台のサーバーと相互作用することができます。

一般的な設計のクライアント / サーバー・システムは、ユーザーと相互作用するクライアント、アプリケーションのビジネス論理を含むアプリケーション・

サーバー、およびデータを保管するリソース・マネージャーの、3 つの層からなります。この方法が 図1 に示されています。このモデルでは、クライアントは、実際のリソース・マネージャーについて何も知る必要がありません。使用しているデータベースを変更した場合、サーバーを変更しなければならないことはありますが、クライアントは変更する必要がありません。サーバーのコピーはクライアントのコピーよりも少ないのが普通であり、またサーバーは更新しやすい位置 (たとえば、ユーザーのデスクで実行される PC ではなく、中央のマシン) にあることが多いため、更新手順も単純化されます。さらに、この方法では追加のセキュリティーが提供されます。リソース・マネージャーによって制御されるデータにアクセスする必要があるのはサーバーだけで、クライアントはアクセスする必要がありません。

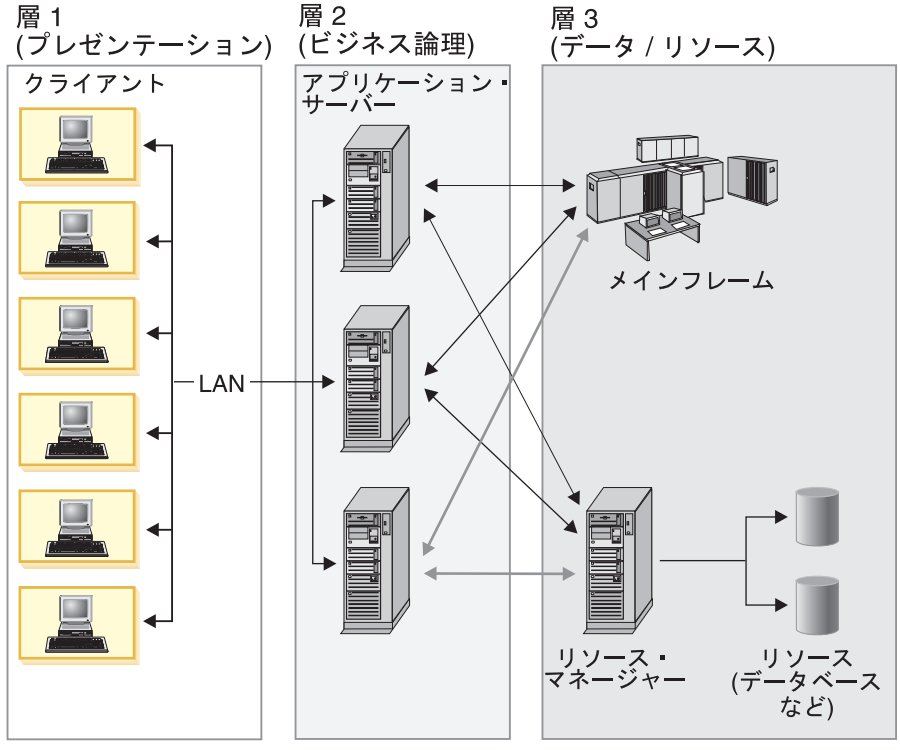


*図 1. 3 層のクライアント / サーバー・アーキテクチャー*

WebSphere Application Server は、このアーキテクチャーにおける中間層を提供し、クライアント (アプレット、Visual Basic® クライアント、C++ クライアント、その他) がデータ・リソース (リレーショナル・データベース、MQSeries®、その他)、および既存アプリケーションと相互作用できるようにし

ます。このアーキテクチャーは、Application Server エンタープライズ版の 2 つの主要コンポーネント、Component Broker および TXSeries でも使用されます。

## トランザクション: 分散環境におけるデータ整合性とパフォーマンスの保証

*トランザクション* は、ある整合状態から別の整合状態にデータを変換する、一連のオペレーションです。この一連のオペレーションは目に見えない作業単位であり、コンテキストによっては、トランザクションは*作業論理単位* (LUW) と呼ばれます。トランザクションは、障害シナリオを単純化する、分散システム・プログラミング用のツールです。

トランザクションは *ACID プロパティー* を提供します。

- *原子性* : トランザクションの変更はアトミックです。つまり、トランザクションの部分であるオペレーションは、すべて行われるか、どれも行われないかのいずれかです。
- *一貫性* : トランザクションは一貫性の保たれた状態間でデータを移動します。
- *独立性* : トランザクションは並行して実行することができますが、どのトランザクションからも、進行中の別の作業は見えません。トランザクションは、逐次実行しているように見えます。
- *耐久性* : トランザクションが正常に完了すると、その変更内容は、その後で障害が発生しても失われずに残ります。

たとえば、ある口座から別の口座にお金を振り替えるトランザクションについて考えてみます。このような振替を行うには、ある口座からお金を引き出し、別の口座にそれを入金する必要があります。ある口座からのお金の引き出しと、別の口座への入金は、1 つのアトミック・トランザクションの 2 つの部分であり、両方を完了させることができない場合には、いずれも行われないようにする必要があります。ある口座に対する複数の要求が同時に処理される場合、それらの要求は*独立* していて、その口座に影響を与えられるのは一時点で 1 つのトランザクションだけでなければなりません。この振替の直後に銀行のサーバーが故障しても、システムが再び使用可能になったときに正しい残高が示されなければなりません。つまり、変更は*耐久性* がなければなりません。*一貫性* はアプリケーションの機能であることに注意してください。ある口座から別の口座にお金を振り替える場合、アプリケーションはその口座から、別口座に加算するのと同じ金額を引き出す必要があります。

トランザクションは、2 つの方法のうちのいずれかによって完了させることができます。つまり、コミットによって完了させることも、ロールバックによっ

て完了させることもできます。正常に完了したトランザクションは、コミットされます。正常に完了しなかったトランザクションは、*ロールバック* されます。ロールバック・トランザクションによって行われたデータ変更は、完全に取り消す必要があります。上の例で、ある口座からお金が引き出されたにもかかわらず、障害のためにお金が別の口座に入金されなかった場合には、最初の口座に対して行われた変更は完全に取り消さなければなりません。次回に送信元が口座残高を照会したときには、正しい残高が表示されなければなりません。

*分散トランザクション* は、通常は複数のマシンで、マルチプロセスで実行されるトランザクションです。各プロセスはトランザクションのために機能します。

分散トランザクションは、ローカル・トランザクションと同じように、ACID プロパティーに従わなければなりません。ただし、分散トランザクションでは、どのプロセスでも障害が発生する可能性があり、そのような障害の際にも、トランザクションのためにすでに行われた作業を各プロセスで取り消す必要があるため、これらのプロパティーの保守は非常に複雑になります。

分散トランザクション処理システムは、次の 2 つのフィーチャーを使用することによって、分散トランザクションで ACID プロパティーを保守します。

- *回復可能プロセス* : 回復可能プロセスは、そのアクションのログを記録するため、障害が発生した場合に前の状態を復元することができます。
- *コミット・プロトコル* : コミット・プロトコルは、複数のプロセスがトランザクションのコミットまたは打ち切りを調整できるようにします。最も一般的な、そしてまた WebSphere Application Server 全体を通じて使用されるコミット・プロトコルは、2 フェーズ・コミット・プロトコルです。

## セキュリティー: 許可使用だけが行われることの保証

エンタープライズ・コンピューティングが、情報システム (IS) サイトに置かれた少数の強力なメインフレームだけによって行われていたときには、許可ユーザーだけがコンピューター・サービスおよび情報にアクセスできるようにすることは、かなり容易な作業でした。分散コンピューティング・システムでは、ユーザー、アプリケーション・サーバー、およびリソース・マネージャーが世界中に分散している可能性があり、コンピューター・システム・リソースの保護はきわめて複雑な作業となりました。

分散コンピューティング・システムでのセキュリティーの提供に関連した問題はたくさんありますが、基本的な問題はあまり変わりません。良好なセキュリティー・サービスは、認証と許可の 2 つの主要機能を提供します。

*認証* は、*プリンシパル* (ユーザーまたはコンピューター・プロセス) が最初にコンピューティング・リソースへのアクセスを試みたときに行われます。その時点で、セキュリティー・サービスはプリンシパルに対し、本人であることの証明を行うように要求します。人間のユーザーの場合には、一般にユーザー ID とパスワードを入力して証明を行い、プロセスの場合には、通常は暗号化された鍵を提供します。パスワードまたは鍵が有効な場合、セキュリティー・サービスはユーザーに、そのプリンシパルを識別し、そのプリンシパルが認証されていることを表す、*トークン* あるいは*チケット* を与えます。

認証を受けたプリンシパルは、そのセキュリティー・サービスによって保護されるコンピューター・システムの境界内にある任意のリソースの使用を試みることができます。ただし、特定のコンピューティング・リソースの使用だけを許可されたプリンシパルは、それらのリソースだけを使用することができます。*許可* は、認証を受けたプリンシパルがリソースの使用を要求したときに行われ、セキュリティー・サービスが そのユーザーがそのリソースを使用する特権を得ているかどうかを判別します。一般的には、許可は、どのユーザーまたはプロセス (あるいはユーザーまたはプロセスのグループ) がリソースの使用を許可されているのかを定義するアクセス制御リスト (ACL) を、そのリソースと関連付けることによって処理されます。そのプリンシパルが許可されている場合、プリンシパルは当該リソースにアクセスすることができます。

分散コンピューティング環境では、プリンシパルとリソースは、両者が真正であることを証明するまでは、相互に相手側の識別を疑う必要があります。このようなことが必要なのは、プリンシパルが識別を偽ってリソースにアクセスしたり、リソースが「トロイの木馬」であって、そのプリンシパルから大切な情報を得ようとしたりする恐れがあるためです。この問題を解決するために、セキュリティー・サービスには、*信頼される第三者* として機能するセキュリティー・サーバーが含まれていて、プリンシパルおよびリソースを認証し、それらのエンティティーが相互に識別を証明し合えるようになっています。

# 第3章 コンポーネント・テクノロジーの概要

ビジネス情報システムは、市場、組織の構造、企業の方法の変化に対応できるようにフレキシブルでなければなりません。信頼性が高く、管理しやすいものでなければなりません。会社の規模が大きくなった場合には拡大することができ、幅広い種類のハードウェアとソフトウェアのプラットフォームで実行することができなければなりません。そして特に、迅速に設計し、コード化し、テストし、実動化しなければなりません。

コンポーネント・テクノロジーとは、IBM WebSphere プロダクト・ファミリーのベースとなるものです。このテクノロジーにより、企業は前述の情報テクノロジーの目標を満たすことができます。コンポーネント・テクノロジーは、アプリケーション開発に良識ある構築ブロック・アプローチを提供します。アプリケーションは、特定の機能を実行するソフトウェアの再使用可能なブロックから作成します。コンポーネント・テクノロジーを使用することで、企業はアプリケーションをより速く開発し、保守を簡易化することができます。

このセクションでは、コンポーネント・テクノロジーの基本概念について説明します。次のトピックから構成されます。

- 『オブジェクト』
- 23ページの『コンポーネント』
- 23ページの『コンポーネント・モデルおよび関連テクノロジー』

この章は、オブジェクト指向プログラミングおよびコンポーネント・テクノロジーの経験がまったく、またはほとんどない方を対象としています。これらの概念をよくご存じの場合は、39ページの『第4章 WebSphere Application Server へのビジネス・モデルの適応』 に進むことができます。

## オブジェクト

オブジェクトを使用して、概念エンティティーおよび物理エンティティーをモデル化します。オブジェクトは、ユーザー・インターフェース、組み込まれているシステム、および複雑なビジネス・システムなどのその他の多くの場所で使用します。モデルは、ソフトウェア・パケット内のエンティティーの状態と振る舞いの両方、つまりオブジェクト を取り込まなければなりません。オブジェクトは、エンティティーに関することを認識し、エンティティーに関連付けられたアクションを実行できなければなりません。オブジェクトが認識してい

るものを、オブジェクトの*状態*、*プロパティー*、または*属性* と言います。オブジェクトが実行することを、オブジェクトの*振る舞い*、*動作*、またはメソッドと言います。まとめると、オブジェクトが認識し、実行することから、アプリケーションのその他の部分に対する*インターフェース* が構成されます。

オブジェクトがインターフェースを実装する方法は、内部の問題です。そのため、インターフェースの実装は、ソフトウェア・システムのその他の部分から隠される、つまり*カプセル化* されます。実装の詳細を隠す、たとえば、オブジェクトの状態を内部で表す方法を隠すことにより、システムのその他の部分が、その詳細に依存しないようにすることができます。これにより、システムの他の部分に悪影響を与えることなく、オブジェクトの実装を変更することが可能になります。

*ビジネス・オブジェクト* は、データと、ビジネス・システムの個々の部分が実行するタスクから成り立ちます。ビジネス・オブジェクトを利用することで、ソフトウェア開発者は、配布する必要があるビジネス情報と実行する必要があるタスクを直接反映するソフトウェア・システムを設計することができます。その結果、非常に重要なビジネス要件を見落とすようなことがなくなります。正しく使用すれば、ビジネス・オブジェクトは、情報システムの管理を簡易化して、情報システムの寿命を伸ばすことができます。

ビジネス・オブジェクトは、独立して存在することはありません。それらは、協力し合って、組織の円滑な運営に必要な操作を実行します。オブジェクト指向システム内の各オブジェクトには、満たさなければならない独自の一連の責任が割り当てられます。これらの責任を定義および割り振るのは、システム設計者の仕事です。

オブジェクトには、相互関係もあります。これは、あるオブジェクトが他のオブジェクトを着信要求の一部または要求の結果として認識する、一時的な関係である場合や、オブジェクトが互いに協力し合うように特に設計された、構造的な関係である場合もあります。

### 構成を用いたオブジェクトの作成

オブジェクト指向システムは、一般に、*構成*、つまり他のオブジェクトからオブジェクトを作成するプロセスを使用して、ビジネス・モデルを実装します。構成を用いたオブジェクトの作成は、ブロックから建物を作る方法に似ています。個々のブロック (オブジェクト) は、さまざまなコンテキストで再使用することができます。古いブロックは新しいブロックに置き換えることができます。ブロックを結合して、より大きいユニットを作成することができます。これらのユニットも、さまざまな方法で使用することができます。

オブジェクト・テクノロジーの約束の 1 つは、構築ブロックの能力と簡易性を使用して、ビジネス・アプリケーションを開発し、保守できるようにすることです。情報システムは、組織と共に拡大および展開する再使用可能なビジネス・オブジェクトの集まりとして設計することができます。

## インターフェース対実装

オブジェクト指向システムの最も強力な機能の 1 つは、オブジェクト・インターフェースの概念です (19ページの『オブジェクト』を参照してください)。オブジェクトのインターフェース (もっと特定すれば、オブジェクトのクラスが提供するインターフェース) は、オブジェクトに適用する特定の必要条件と保証を指定するものです。

特定の前提条件を満たさなければ、オブジェクトのクライアントがそのオブジェクトのインターフェースを使用できないようにすることができます。前提条件には、メソッドを呼び出す順序に関する制限事項や、クライアントが提供する入力データのタイプに関する必要条件を含めることができます。

クライアントにサービスを提供するオブジェクトは、受け取ったメッセージに応えて、特定のタスクを実行しなければなりません。インターフェースで指定されるのは、考慮すべき例外を含め、確実に発生することだけです。オブジェクトがタスクを実行する方法については、何も指定されません。

インターフェースの*実装* では、オブジェクトが要求をどのように満たすかを指定します。実装は常にプライベートです。インターフェースの実装は変更することができますが、インターフェースそのものには影響しません。

## クラス

オブジェクト指向システムには、オブジェクトを作成する際にテンプレートの役割を果たすクラス というメカニズムが用意されています。クラスは、オブジェクトのインターフェース、そのインターフェースをサポートする実装の詳細、およびオブジェクトの状態を示す特定の情報を表すものです。そして、それぞれのオブジェクトは、クラスのインスタンス です。

## 継承

常に、システムにはインターフェースの複数の実装がある可能性があります。クライアントは、インターフェースの各実装と一様かつ透過的に対話することができます。この概念は、オブジェクト指向システムに固有のものです。これを可能にしているのが、*継承* という考えです。継承により、クラスは、他のクラスのインターフェースと実装を共有、すなわち継承することができます。他のクラスに属性と振る舞いを渡すクラスのことを*親クラス* または*基本クラス*

と言います。属性と振る舞いを継承するクラスのことを*子クラス* または*派生クラス* と言います。子クラスは、1 つ以上の親クラスから継承することができます。

継承を使用することで、追加による設計を行うことができます。関連するタイプのファミリーの属性と振る舞いをパッケージした共通の親クラスを作成できます。子クラスは、親のインターフェースを継承し、また独自の属性と振る舞いを追加することもできます (属性とメソッドは、親インターフェースから引き出すことはできません)。

## ポリモアフィズム

継承を使用することで、オブジェクトを繰り返し使用することができます。インターフェースの再使用のことを、*インターフェースの継承* と言います。インターフェースの実装の再使用のことを、*実装の継承* と言います。インターフェースは再使用できますが、インターフェースの実装は必ずしも再使用できるとは限りません。たとえば、残高を調整する方法は、当座預金と貯蓄預金で異なることがあります。実際、インターフェース設計の再使用は、実装コードの再使用よりもはるかに大切です。プライベートな実装の詳細は、システム内の他のオブジェクトに影響しませんが、パブリック・インターフェースの詳細は影響します。

*ポリモアフィズム* は、インターフェースを再使用するための強力なツールです。これにより、継承によって関連している各種クラスのオブジェクトを、同じ関数呼び出しに対して固有の方法で応答させることが可能になります。インターフェースすべてに同じメソッドが含まれていますが、メソッドの実装はオブジェクトによって異なります。子クラスでのメソッドの実装は、親クラスの元の実装をオーバーライドすることができます。メソッドが呼び出されると、各クラスの派生オブジェクトは、特定の実装で定義された方法で応答します。

ポリモアフィズムを利用すれば、クライアント・コードの再使用も可能です。クライアントは、自分のインターフェースを介した場合のみオブジェクトを参照します。つまりクライアントは、そのインターフェースを実装したすべてのオブジェクトと通信することができます。クライアントからの問い合わせを受けた各オブジェクトは、独自の実装に適した方法で応答します。クライアントの観点から見ると、あるオブジェクトを、システム内の他のオブジェクトの代わりとして使用することができます。

オブジェクト・コードも再使用できます。一般に、簡単なインターフェースを持つオブジェクトは、それだけ再使用できる確率も高くなります。たとえば、Account オブジェクトのインターフェースは、そのオブジェクトを、照会やレ

ポート、決算処理や監査処理、月次明細書の利子計算など、さまざまなコンテキストで使用できるように設計することができます。

再使用を実現する場合には、複数のプロジェクトの過程で数多くの設計反復が必要になることがあります。ソフトウェア開発者は、オブジェクト指向システムで再使用を実現できる可能性がある場合には、再使用オブジェクトの設計に高い優先順位を割り当てなければなりません。

## コンポーネント

コンポーネントは、大規模な情報システムで機能することができる、追加可能なオブジェクトです。これらの追加可能な機能には、次のタスクを実行する機能が含まれます。

- オブジェクトの作成と破棄
- オブジェクトへの分散ネットワーク内の場所の提供
- オブジェクトのシステム同一性の確立
- リソース・マネージャーへのオブジェクトの状態の格納
- オブジェクトの活動化および非活動化 (またはページング) の処理
- オブジェクトの状態変換へのトランザクションの意味体系のマップ
- オブジェクトのロックと、そのベースとなるデータ・ストアとの調整
- オブジェクトへのアクセスの制御
- オブジェクトの検索
- オブジェクトへのイベント発生の通知
- 分散ネットワークでのオブジェクトの状態の転送
- オブジェクトへの人間が判読できる名前の割り当て

コンポーネントに変換したオブジェクトは、分散処理、リモート・プロシージャー呼び出し、パーシスタンス管理、統合セキュリティー、トランザクション、並行性制御、および動的照会機能をサポートするので、従来のクライアント / サーバー・アプリケーション環境で再使用できます。

## コンポーネント・モデルおよび関連テクノロジー

コンポーネント・モデル、すなわちコンポーネント・アーキテクチャー では、オブジェクトが含まれるソフトウェア・システム内でオブジェクトを 1 つにまとめる方法を指定します。オブジェクトの相互運用性に対するモデルです。このセクションでは、 WebSphere Application Server でビジネス・アプリケーシ

ョンを設計して実装する場合に使用できるコンポーネント・モデルおよび関連テクノロジーについて説明します。次のトピックから構成されます。

- 『Java 2™ Platform Enterprise Edition (J2EE™)』
- 26ページの『JavaBeans コンポーネント』
- 27ページの『Enterprise beans』
- 29ページの『アプレットおよびサーブレット』
- 33ページの『JavaServer Pages』
- 33ページの『CORBA』
- 35ページの『Microsoft COM』
- 36ページの『Managed Object Framework』

## Java 2™ Platform Enterprise Edition (J2EE™)

WebSphere Application Server は、Java™ 2 Platform, Enterprise Edition (J2EE™) 仕様に準拠しています。 J2EE では、多層アプリケーションの設計と実装の標準アーキテクチャーが定義されています。次のエレメントから成り立ちます。

### J2EE アプリケーション・コンポーネントおよび実行時環境

J2EE プラットフォームは、J2EE アプリケーションを取りまとめる標準プラットフォームです。実行時環境は、次のものから成り立ちます。

- アプリケーション・コンポーネント。J2EE 対応プログラムを作成する場合の構築ブロックです。 J2EE プログラミング・モデルは、アプリケーションがサポートしなければならない 4 タイプのコンポーネントを定義しています。
  - アプリケーション・クライアント。一般的にデスクトップ・コンピューター上で実行される Java プログラムです。
  - アプレット。Web ブラウザーで実行され、J2EE アプリケーションのユーザー・インターフェースを提供します。
  - サーブレットおよび JSP ページ。 Web サーバー上で実行され、アプリケーション・クライアントからの HTTP 要求に応えます。
  - Enterprise beans。 EJB サーバーで実行され、アプリケーションのビジネス論理を含み、トランザクションを処理します。
- コンテナー。アプリケーション・コンポーネントの実行時サポートを提供します。トランザクション管理、セキュリティー、状態管理などのサービスをアプリケーション・コンポーネントに提供します。
- リソース・マネージャー・ドライバー。 J2EE アプリケーションを外部リソースに接続します。

- データベース。ビジネス・データを格納します。

### J2EE アプリケーション・モデルおよび開発チームの役割

J2EE アプリケーション・モデルでは、多層のシン・クライアント・サービスを開発する場合の標準アプリケーション・モデルを定義します。アプリケーションは、サーブレット、JSP ページ、Enterprise beans などのコンポーネントからアセンブルされて、 J2EE 実行時プラットフォームに配置されます。 J2EE アプリケーション開発チームの役割は次のとおりです。

- プロダクト・プロバイダー — コンポーネント・コンテナー、 J2EE プラットフォーム API などの J2EE 機能を含め、 J2EE プロダクトを実装します。
- システム管理者 — 配置した J2EE アプリケーション、および対応するコンピューティングとネットワーク・インフラストラクチャーを構成し、管理します。
- ツール・プロバイダー — アプリケーション・コンポーネントを開発し、パッケージするためのツールを提供します。
- アプリケーション・コンポーネント・プロバイダー — J2EE アプリケーション・コンポーネントを設計し、実装します。
- アプリケーション・アセンブラー — アプリケーション・コンポーネントを 1 つの完全な J2EE アプリケーションにアセンブルします。
- デプロイヤー — アプリケーションを配置します。

### J2EE の標準サービス

J2EE の標準サービスには、次のサービスおよびプロトコルが含まれます。

- HTTP および HTTPS
- Java Transaction API (JTA)
- RMI-IIOP。Java 固有の RMI プロトコルおよび CORBA IIOP プロトコルの両方がサポートされています。
- JavIDL
- Java Database Connectivity (JDBC™) API
- Java Messaging Service (JMS) API
- Java Naming and Directory Interface™ (JNDI)
- JavaMail™
- JavaBeans™ Activation Framework (JAF)
- Java API for XML Parsing (JAXP)
- J2EE Connector アーキテクチャー

- Java Authentication and Authorization Service (JAAS)

### J2EE 互換性テスト・スイートおよび参照の実装

J2EE 互換性テスト・スイートは、アプリケーションが J2EE プラットフォーム標準に準拠していることを検証する、一連の互換性テストです。

J2EE 参照の実装では、J2EE の可能性を示し、 J2EE プラットフォームの動作定義が提供されます。

## JavaBeans コンポーネント

コンポーネントの実装には、数多くのプログラム言語を使用することができます。ただし、クロス・プラットフォーム・コンポーネント開発には、 Java プログラム言語が最適です。 Java は、Sun Microsystems が作成したオブジェクト指向言語です。 C++ などの古いオブジェクト指向プログラム言語よりも、使いやすくなっています。

*JavaBeans* テクノロジーは、 Java アプリケーション環境向けの Sun のコンポーネント・アーキテクチャーです。一度作成すればどこでも実行できるという考えを、再使用可能なコンポーネントの開発まで拡張しています。 *bean* は、JavaBeans 仕様に従って作成された再使用可能なソフトウェア・コンポーネントです。ビジュアルなプログラミング・ツールを使って、複数の bean を組み合わせて 1 つのアプリケーションを作成することができます。

bean は、普通の Java オブジェクトからインターフェースの追加セットを実装します。 bean に固有のインターフェースは次のとおりです。

- **イントロスペクション** — ビジュアルなプログラミング・ツールで bean の動作を分析できるため、開発者は bean を接続することができます。
- **カスタマイズ** — 開発者は、ビジュアルなプログラミング・ツールに用意されているプロパティー・シートを使用して、 bean の外観と振る舞いをカスタマイズすることができます。
- **イベント** — 通知メカニズムを使用することで、 bean は互いに通信することができます。
- **プロパティー** — bean の属性を表し、属性が変更されたときに通知をブロードキャストします。
- **パーシスタンス** — 開発者は、bean をカスタマイズして、 Java Archive (JAR) ファイルにパッケージすることができます。シリアライゼーション とも言います。

27ページの図2 に、簡単なアプレットの開発時に 2 つの JavaBeans コンポーネントをどのように結合できるかを示します。この例では、ユーザーが

「**Close**」ボタンを押すと、イベントが発生します。イベントは、メソッドを呼び出す AccountProxy bean に伝えられます。ビジュアルな Java 開発ツールを使用して、bean をリンクすることができます。

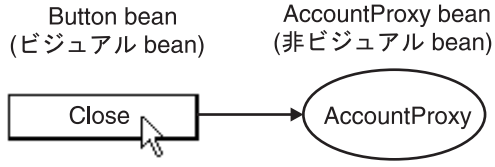


*図 2. bean では、ビジュアルなプログラミングを行える*

bean は、ActiveX コントロールと相互運用することもできます (ActiveX は、Microsoft のダウンロード可能、相互運用可能なオブジェクトのバージョンです)。ブリッジ・ソフトウェアを利用すれば、 bean を Internet Explorer、Visual Basic®、Microsoft Word などの ActiveX 環境に配置することができます。

## Enterprise beans

*Enterprise JavaBeans* (EJB) 仕様に従うことで、開発者は Java でサーバー側のビジネス・コンポーネントを作成することができます。 Enterprise beans とは、Java プログラム言語で分散型のオブジェクト指向ビジネス・アプリケーションを開発するための標準コンポーネント・アーキテクチャーです。サーバー上にインストールして、リモート・クライアントからアクセスするように作られています。 Enterprise beans は、プラットフォームに依存せず、 EJB 仕様をサポートするどのような実行時環境にも配置することができます。

このセクションは、次の Enterprise bean 関連のトピックから構成されます。

- 『EJB アーキテクチャー』
- 28ページの『パーシスタンス』
- 29ページの『Enterprise beans の配置』

### EJB アーキテクチャー

28ページの図3 に、 EJB 仕様の詳細を示します。

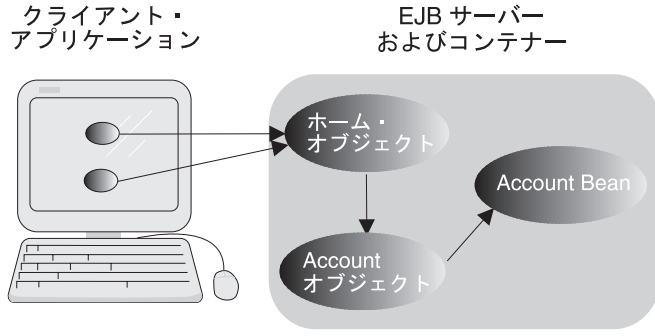


*図 3. EJB アーキテクチャー*

クライアント・アプリケーションは、ホーム というリモート・オブジェクトと通信します。ホーム・オブジェクトを使用して、リモート・クライアントからアクセスするビジネス・オブジェクト (たとえば、Account オブジェクト) を作成します。このオブジェクトが、口座に関するすべての属性と振る舞い (口座残高を調整したり、口座を開いたりするメソッドなど) を提供します。

Account オブジェクトは、受け取った要求を、実際のビジネス実装を提供する Enterprise bean に転送します。着信要求を解釈することによって、 Account オブジェクトは Enterprise bean へのアクセスを制御し、ビジネス・メソッドを実行するための要件を処理します。たとえば、Account オブジェクトが適切なトランザクション・コンテキストを設定しなければ、口座からお金を出し入れすることはできません。

図3 のサーバー側で、コンテナー は EJB 環境の実行時コンポーネントです。コンテナーは、トランザクション・サポート、セキュリティー・サポート、メモリー管理などの重要なサービスを提供します。 Account オブジェクトは、タスクを実行するためにコンテナーと共同作業を行います。

ホーム・コレクションとコンテナーは、 WebSphere Application Server エンタープライズ版だけの機能です。 C++ ビジネス・オブジェクトと Enterprise beans の同時サーバー・サポートを提供します。実行時サービスを階層化する必要はありません。

### パーシスタンス

Enterprise beans は、2 つの異なるタイプのパーシスタンス を示します。

- セッション *bean* は、タスクと操作を表します。一時的なもので、永続的ストレージ内のデータとは対応しません。ただし、セッション bean は、存続

時間で実行されたことがあるアクションのリストなど、状態情報を保持することができます。セッション bean に関連付けられた状態情報は、クライアント間では共有されません。

- エンティティー *bean* は、永続的データを表します。各エンティティー bean は、他の Enterprise beans と共有するデータ・ストアにマップされます。たとえば、エンティティー beans はリレーショナル・データベース内の列に対応することができます。多くの場合、エンティティー bean には、なんらかのトランザクション方式でアクセスしなければなりません。エンティティー bean のインスタンスは固有であって、複数のユーザーからアクセスすることができます。エンティティー beans は、状態情報を格納しないので、クライアント間で共有することができます。

エンティティー bean の永続的状態を管理する方法は 2 つあります。

- *bean 管理パーシスタンス* (BMP) では、 bean は永続的ストレージに直接アクセスします。永続的ストレージへのアクセスは、ソフトウェア開発者が実装しなければなりません。
- *コンテナー管理パーシスタンス* (CMP) では、 bean はコンテナーに依存して、永続的ストレージへの透過的なアクセスを提供します。ソフトウェア開発者は、永続的ストレージへのアクセスを明示的に実装する必要はありません。

### Enterprise beans の配置

Enterprise beans は、実行時サービスを提供するコンテナーに配置します。 *配置記述子* を使用して、コンテナーが Enterprise bean を管理する方法を指定します。配置記述子は、アプリケーションのアセンブリー時または配置時に設定します。これは、Enterprise beans のライフサイクル、パーシスタンス、トランザクション、およびセキュリティーの設定を定義するものです。

さらに、 Enterprise bean の*環境プロパティー* を利用すれば、開発者はアプリケーションのニーズに基づいて bean をカスタマイズすることができます。たとえば、環境プロパティーでデータベースの場所を指定することができます。配置記述子と環境プロパティーを使用することにより、ベースとなるソース・コードにアクセスすることなくアプリケーションをカスタマイズできます。

## アプレットおよびサーブレット

Java アプレット およびサーブレット は、 beans および Enterprise beans を補うテクノロジーです。それ自身はコンポーネントではありませんが、 Java プログラムは、特定のクライアント / サーバー・タスクを実行するように調整されます。

### アプレット

アプレットは、クライアント・マシンの Java 対応ブラウザーで実行されます。アプレット内のビジネス論理を中央の World Wide Web サーバーからクライアントにダウンロードして実行することにより、クライアント・マシンが常に最新のコードを取得することを保証します。管理コストは、アプリケーションを手作業で更新しなければならない従来のデスクトップ環境よりも低くなります。アプレットのダウンロード可能な性質により、ハードウェア要件も削減されます。これらの理由から、多くの企業が内部に Java アプレットを配置しています。

アプレットは、Java2™ Software Development Kit (SDK) 内の標準パッケージを使用して設計することも、 Java Foundation Classes (JFC) のコンポーネントを使用して設計することもできます。アプレットは、Web ブラウザーが提供する Java Virtual Machine (JVM) を使用します。 JVM は、ベースとなるオペレーティング・システムに従ってコンパイル後の Java プログラムを解釈する実行時環境です。 JVM はほとんどのオペレーティング・システムで開発されているので、アプレットは、特別な移植要件なしでさまざまなオペレーティング・システムで実行することができます。

また、アプレットを使用することで、より多くのクライアントまでビジネスを広げることができます。インターネットにより、顧客に企業への新しい入り口が提供され、企業エレクトロニック・コマース (電子商取引) を行うチャンスが開かれます。アプレットは、いつでもインターネットを介してダウンロードできるため、顧客は発注や情報の取得など、さまざまなタスクを実行できるようになります。

31ページの図4 に、アプレットを使用してデータベースにアクセスする場合のシナリオを示します。 Web サーバーは、ハイパーテキスト・マークアップ言語 (*HTML*) の 1 ページの情報を Web ブラウザーにダウンロードします (1)。このページには、*Java Archive* (JAR) ファイルにパッケージされた Java アプレットをサーバーからダウンロードするコマンドが含まれています (2)。アプレットは、Java の *Remote Method Invocation* (RMI) プロトコルを使用して、サーバーに常駐する他のオブジェクトと通信します (3)。そして、これらのオブジェクトがデータベースにアクセスします (4)。

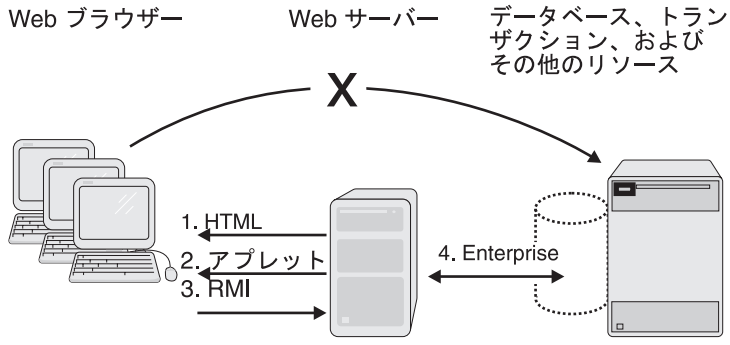


*図4. アプレット・ベースのビジネス・ソリューション*

アプレットは、リソースに直接アクセスできません。 Java セキュリティーにより、アプレットは、ブラウザーが実行されているマシン上のファイル・システムや、サーバー以外のマシンと直接やり取りできないようになっています。

アプレットには、2 つの欠点があります。

- アプレットのサイズとネットワーク接続のタイプによって、ダウンロードにかなり時間がかかることがあります。
- ブラウザーの JVM が、アプレット内の Java コードの機能をサポートしていない場合は、アプレットは正しく機能できません。適正な JVM をブラウザーに動的に接続することはできますが、これでも、基本的な互換性問題は解決されず、アプレットのパフォーマンス低下を招くことがあります。

### サーブレット

Java サーブレットは、アプレットを補うテクノロジーです。多くの場合、アプリケーションではアプレットとサーブレットの両方を使用します。サーブレットは、WWW サーバー上で実行される Java プログラムです。特別なインターフェースを利用することで、 HTML 形式からのユーザーの実行依頼などの要求を Web ブラウザーから受け取ることができます。サーブレットは一般に、着信要求に応答して企業リソースにアクセスし、新しい HTML ページを動的にフォーマットして、ブラウザーに戻します。 HTML 生成を簡易化するため、サーブレットは *Java Server Side Include* (JSSI) スクリプトと *JavaServer Page* (JSP) スクリプトを使用することができます (ただし、使用できるスクリプトはこれらだけではありません)。

サーブレットは、Java Servlet API とそれに関連するクラスおよびメソッドを使用します。また、サーブレットは、Java Servlet API を拡張して機能を追加する Java クラス・パッケージも使用します。アプレットと同様、サーブレットは Java 標準の一部であり、各種プラットフォームで実行できる設計になっ

ています。ただし、アプレットと異なり、Java サーブレットはブラウザーがサポートする JVM を使用しません。サーブレットの実行環境は、企業内で制御される Web サーバーです。これにより、サーブレットの振る舞いは信頼でき、予測できるものになります。

図5 に、サーブレットだけを使用してデータベースにアクセスする方法を示します。サーブレットで生成された HTML ファイルには、組み込み *JavaScript* コマンドが含まれています (1)。 JavaScript は、クライアント・マシン上で実行する簡単なプログラムを作成する場合に使用するスクリプト言語です。JavaScript に JVM は必要ないので、ダウンロード時間はアプレットよりも短くなります。ブラウザーは HTTP を使用して、サーブレットと直接やり取りします (2)。サーブレットはデータベースやその他のリソースにアクセスすることができます (3)。

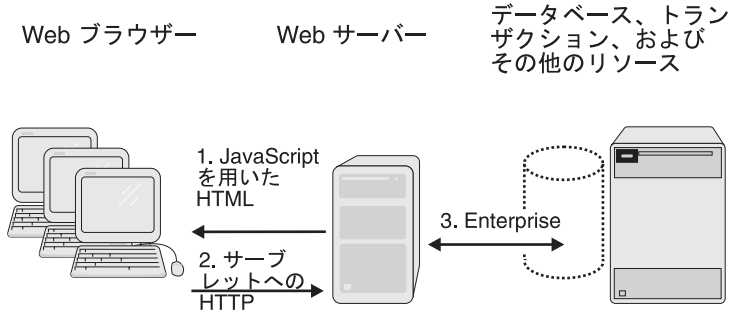


*図5. サーブレット・ベースのビジネス・ソリューション*

サーブレット・ベースのアプリケーションには、インターネットにアクセスできる機能と、クライアントでローカルに論理を処理できる機能があります。管理コストとクライアント・ハードウェア・コストが削減されます。
31ページの図4 のアプレット・ベースのアプリケーションと比べて、サーブレット・ベースのアプリケーションでは、アプレットのダウンロードにかかるオーバーヘッドが回避されます。 JVM が必要ないので、クライアント・ブラウザーの機能に関する問題も減ります。また、ファイアウォールを介した通信に関する要件もなくなります。

一般に使用されている CGI プログラムでは、プロセス全体がユーザー要求を処理する必要がありますが、サーブレットは、それとは異なり、スレッドを使用してユーザー要求を処理することができます。この機能があるため、サーブレットは CGI プログラムに比べて大幅に効率が高くなっています。

サーブレットは、Web サーバーの開始時に自動的にロードすることも、クライアントがサービスを最初に要求したときにロードすることもできます。ロードされたサーブレットは、引き続き実行され、さらにクライアント要求を待ちます。

## JavaServer Pages

WebSphere Application Server は、動的 Web ページ・コンテンツへの強力なアプローチである JavaServer Pages (JSP) をサポートします。 Application Server における JSP 機能は、Sun Microsystems の JavaServer Pages Specification に基づいたものです。

JSP ファイルは、ある意味では、静的 HTML におけるサーバー・サイド・インクルードと似ています。これは、サーブレット機能が Web ページに組み込まれる点が共通しているためです。ただし、サーバー・サイド・インクルードでは、サーブレットの呼び出しは特別なサーブレット・タグに組み込まれ、JSP ページでは、Java サーブレット・コード (またはその他の Java コード) は HTML ページに直接組み込まれます。

JSP ページの多くの利点のうちの 1 つは、HTML コーディングを Web ページ内のビジネス論理と効率良く分離できることです。 JSP ページを使用して、サーブレット、Java bean、Enterprise beans、Java ベースの Web アプリケーションなどの、再使用可能コンポーネントにアクセスすることができます。

## CORBA

共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) 仕様は、分散オブジェクトと統合オブジェクト・サービス間の通信を定義するものです。 CORBA は、コンポーネントを使用するアプリケーション開発に共通のフレームワークを提供します。オブジェクト同士が分散コンピューター・ネットワークを介して透過的に通信することを可能にする、ソフトウェア・サービスを定義します。 CORBA は、言語にもプラットフォームにも依存していないため、さまざまなベンダー・パッケージに実装され、各種マシンに配置され、異なるプログラム言語でコード化され、各種オペレーティング・システムで実行されるオブジェクトを処理することができます。 CORBA 対応アプリケーションは、TXSeries と Component Broker のどちらを使用しても作成できます。

### 分散オブジェクト通信

分散オブジェクトは、*オブジェクト・リクエスト・ブローカー* (*ORB*) を使用して通信します。 ORB は、*IIOP* (*Internet Inter-ORB Protocol*) を使用して、ネットワークを介してローカル・クライアント要求を送信します。このプロト

コルは、CORBA 定義のメッセージ交換機能を持つ TCP/IP ベースのプロトコルです。 IIOP により、ORB は互いに通信し合い、分散オブジェクト通信にインターネットを使用することができます。

図6 に、ORB を使用したクライアント / サーバー通信を示します。クライアントは、ネットワークの別の場所に置かれているリモート・オブジェクトを表すプロキシー・オブジェクト を使用します。プロキシー・オブジェクトがリモート・オブジェクトとの通信を処理するので、クライアントは実際のオブジェクトがどこに常駐しているかを知る必要はありません。プロキシー・オブジェクトは、ORB を利用して、クライアント要求を、ネットワークを介して送信できる形式に変換します。サーバー上のオブジェクト・アダプター が、リモート・オブジェクトを見つけて、処理対象の要求をディスパッチします。これは、ORB とオブジェクト実装コード間の一次インターフェースです。結果は、オブジェクト・アダプターからプロキシー・オブジェクトに、プロキシー・オブジェクトから要求元のクライアントに戻されます。

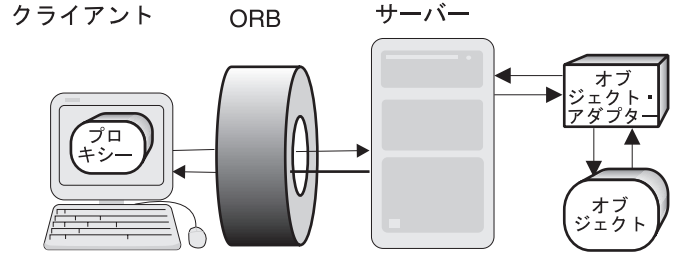


*図 6. ORB を使用したクライアント / サーバー通信*

プロキシー・オブジェクトと実際のオブジェクトは、 *Interface Definition Language* (*IDL*) を使用することで同期します。 IDL は、リモート・オブジェクトが提供する振る舞いと属性を指定するものです。アプリケーション開発時に、インターフェースの IDL 指定がコンパイルされて、クライアントとサーバーのリモート・インターフェース を実装するコードが生成されます。クライアントは、このリモート・インターフェースを介してリモート・オブジェクトと通信するので、このインターフェースをサポートするすべてのリモート・オブジェクトとやり取りすることができます。コンポーネント・インターフェースは、CORBA IDL ファイルで作成されて、 CORBA の **idl** コンパイラーを使用してコンパイルされます。これにより、分散アプリケーションの作成に必要なほとんどのスタブおよびスケルトン・ファイルが作成されます。

CORBA のビジネス・オブジェクトは、プロキシーが表すインターフェースを実装します。クライアントのプロキシー・オブジェクトは、抽象的なインター

フェースを表します。このオブジェクトのサーバー実装では、このインターフェースを継承して、オブジェクトのビジネス論理を実装します。

#### リモート呼び出し

CORBA リモート呼び出しを使用すると、クライアント・プロキシー・オブジェクトとサーバー側オブジェクトの間の通信が可能になります。CORBA サーバーは、クライアント・プロキシー・オブジェクトがサービスを得るためにバインドする実装オブジェクトをエクスポートします。トランザクションに関与したり他のオブジェクトに対してトランザクション要求を行ったりするオブジェクトは、トランザクション・オブジェクト と呼ばれます。

CORBA におけるリモート呼び出しは、クライアントがプロキシー・オブジェクトを作成し、そのオブジェクトを使用して、サーバーにある対応の実装オブジェクト上のメソッドを呼び出したときに行われます。このリモート呼び出しは、多くの点で RPC に似ています。

#### 共通オブジェクト・サービス

CORBA では、分散コンポーネントの管理に使用する、共通オブジェクト・サービスが指定されています。これらのサービスには、命名、セキュリティー、ライフサイクル、イベント、外部化、トランザクション、パーシスタンスなどが含まれます。 CORBA 仕様には、特定のドメインや共通機能をサポートするインターフェースも含まれています。

### Microsoft COM

Microsoft Component Object Model (COM) は、バイナリーのソフトウェア・コンポーネントからアプリケーションを作成することを可能にする、コンポーネント・アーキテクチャーです。 COM を使用することで、Windows アプリケーションを、単一のエンティティーとしてではなく、複数のコンポーネントとして開発することができます。各アプリケーションの個々のコンポーネントは、透過的および個別にアップグレードできるため、アプリケーションはより分散しやすくなります。 COM では、どのようなプログラム言語でもアプリケーションを作成することができます。

COM コンポーネントは、アプリケーションの 1 つのアクションを実行する、コンパイルされたコードの一部分です (これに対し、他のオブジェクト指向プログラミング・モデルにおけるコンポーネントは、アプリケーションのソース・コードで定義されます)。 COM コンポーネントは、クライアントでインスタンス化されると、そのクライアントのプロキシーとして機能し、データを

マーシャルおよびアンマーシャルしてサーバーとやり取りし、データと状況情報をクライアントに戻します。これにより、データはカプセル化されて処理から保護されます。

COM は、次のものを提供します。

- 標準化されたコンポーネント・インターフェース
- コンポーネント間の通信
- コンポーネント間の共用メモリー管理
- エラーおよび状況のレポート
- コンポーネントの動的ロード

## Managed Object Framework

Component Broker には独自のオブジェクト・モデルがあります。サーバー・オブジェクトは、Managed Object Framework (MOFW) から派生したものです。MOFW は、複数のオブジェクトが 1 つとなって単一コンポーネントとして動作できるようにするものです。各コンポーネントは、アプリケーション・アダプターによって管理される、一連のオブジェクトからなります。クライアントから見ると、コンポーネントは、オブジェクトの集合としてサーバーに実装されている場合でも、単一オブジェクトとして機能します。

コンポーネントは、ビジネス・オブジェクト、管理下のオブジェクト、データ・オブジェクトの、 3 タイプの基本オブジェクトから成り立ちます。永続的なコンポーネントは、4 つ目のオブジェクト、つまり、リソース・マネージャーやデータ・ストアにアクセスするためのコードを提供する、永続オブジェクトを追加します。キー・オブジェクトとコピー・ヘルパー・オブジェクトが、コンポーネントの位置付けと作成をサポートします。

次に、コンポーネントを構成するオブジェクトについて説明します。

### ビジネス・オブジェクト

*ビジネス・オブジェクト* は、コンポーネントのインターフェース、つまりビジネス論理を含む属性とメソッドを定義するものです。ビジネス・オブジェクトのインターフェースは、CORBA Interface Definition Language (IDL) によって定義されます。 IDL は、オペレーティング・システムおよびプログラム言語とは独立して、オブジェクトのインターフェースを指定します。

ビジネス・オブジェクトは、C++ でも Java プログラム言語でも実装することができます。ビジネス・オブジェクトは Managed Object Framework から派生するものであるため、ユーザーが提供する必要があるコードは、ユーザーが定義したアプリケーション固有メソッドの実装だけです。 Component Broker ツ

ールを使用してビジネス・オブジェクトを作成すると、フレームワークが拡張されます。オブジェクトの状態の一部である属性は、キャッシュに格納したりデータ・オブジェクトに委任したりすることができます。

### データ・オブジェクト

データ・オブジェクト は、コンポーネントの重要な状態情報 (状態データ) を管理するものです。これらのオブジェクトは、ビジネス・オブジェクトが状態データを獲得および設定するためのインターフェースを提供します。

データ・オブジェクトにより、ビジネス・オブジェクトは次のことを実行せずに済みます。

- どのデータ・ストアを使用するかを知ること
- データ・ストアへのアクセス方法を知ること
- データ・ストアへのアクセスを管理すること

### 管理下のオブジェクト

*管理下のオブジェクト* では、アプリケーション・アダプターによってコンポーネントを管理することができます。この管理能力は別のオブジェクトによって提供されるため、コンポーネント・インターフェースに影響を与えることなく、提供されるサービスのタイプを変更することができます。

管理されるサービスの例として、以下のものがあります。

- パーシスタンス、トランザクション、およびセキュリティー
- 複数のサーバーでの作業負荷管理およびアベイラビリティー管理
- 既存のデータベースおよびアプリケーションへのオブジェクト指向アクセス

### 永続オブジェクト

*永続オブジェクト* は、コンポーネントの状態をデータ・ストアに保管するメカニズムを提供する C++ オブジェクトです。それぞれの永続オブジェクトには、データ・ストア内の対応するレコードを見つけるために使用される ID、すなわちキーがあります。永続オブジェクトには、主として次の 2 つの種類があります。

- データベース・データへのアクセス (データベース・スキーマへのマッピング) に使用されるもの
- 手続き型データへのアクセス (CICS または IMS bean、または Procedural Adaptor bean (PA bean) へのマッピング) に使用されるもの

### キー・オブジェクトおよびコピー・ヘルパー・オブジェクト

コンポーネントのキー・オブジェクト は、サーバー上のコンポーネントの特定インスタンスを探す場合に使用する属性を定義するものです。キーは 1 つまたは複数のビジネス・オブジェクト属性からなります。この属性には、インスタンスを一意的に識別するために十分な情報が含まれていなければなりません。

コピー・ヘルパー・オブジェクト は、クライアント・アプリケーションがサーバー上のコンポーネントの新規インスタンスを作成するための効果的な方法を提供します。このオブジェクトはオプションで使用できます。コピー・ヘルパーには、ビジネス・オブジェクト (またはそのサブセット) と同じ属性が含まれています。コピー・ヘルパーを使用しないと、クライアントは、それぞれの新規インスタンスごとにサーバーへ多くの呼び出しを行わなければならない可能性があります。つまり、インスタンスを作成するために 1 つの呼び出しが必要となり、さらにインスタンスのそれぞれの属性ごとに呼び出しが必要になることがあります。コピー・ヘルパーを使用すると、クライアントは、コピー・ヘルパーのローカル・インスタンスを作成し、その属性の値を設定してから、サーバー・コンポーネントを作成してコピー・ヘルパーを渡すことにより、1 回の呼び出しでその属性を初期化することができます。

### コンポーネントのアセンブル

コンポーネントのアセンブルは、ビジネス・オブジェクト、データ・オブジェクト、またはスキーマから始めることができます。オブジェクトを 1 つのコンポーネントに構成するには、サーバー上の特定のコンポーネントを形成するオブジェクトを選択しなければなりません。配置済みのコンポーネントには、クライアント・アプリケーション、またはサーバー・データへのアクセスを必要とするその他のコンポーネントから、管理下のオブジェクトを介してアクセスします。

### コンポーネントのインスタンス化および実行

各コンポーネントは、ホーム・オブジェクトを使用することにより、インスタンス化され、その後で探し出されます。ホーム・オブジェクト は、クライアントおよびその他のコンポーネントが特定タイプのコンポーネントを見つけたり作成したりできるようにする、サーバー・オブジェクトです。コンポーネントは、作成されると、コンテナー内で実行されます。コンテナーはコンポーネントのアプリケーション・アダプターとして機能し、管理下のオブジェクトを介してそのコンポーネントに管理サービスを提供します。呼び出しプログラムからは、ビジネス・オブジェクト・インターフェース (これは、コンポーネント全体へのインターフェースとして機能します) しか見えません。管理下のオブジェクトさえ、隠れて見えません。

# 第4章 WebSphere Application Server へのビジネス・モデルの適応

人材が企業において最も貴重なリソースであるのと同様に、ソフトウェアもまた企業にとって大切なものです。他のビジネス・リソースに対する場合と同じように、その期待に応えなければなりません。法規の改正、新しいテクノロジー、競合製品などの外部影響が引き金となって起こる変更は、ビジネス・ポリシーとビジネス・プロセスの点から分析することができます。これで、対応する情報システム・モデルを設計し、適応することができます。

## コンポーネント・テクノロジー

ソフトウェアは、過剰に適応させることなく、既存のビジネス・プロセス内で適切に機能しなければなりません。既存のビジネス・プロセスを作り直すことは、生産性の低下と同様、望ましいことではありません。コンポーネントは、ビジネス・モデルでソフトウェア実装を制御できる、分散アプリケーションの新しい作成アプローチです。理由は、次のとおりです。

- ソフトウェア・モデルとソフトウェア実装はビジネスの観点から表される。
- コンポーネントにビジネス名を割り当て、ビジネスの役割と一貫性がある振る舞いと状態を埋め込む。
- コンポーネント間の関係をビジネス・プロセスで設定する。
- モデルは、ビジネス・ドメインのエキスパートが情報テクノロジーの専門家のサポートを受けて、協力して作成および保持する。

コンポーネント・ベース・システムのこれらの特性から、今日の変化の激しいビジネス環境に必要な柔軟性を得ることができます。ビジネス目標における変化をシステム・モデルにそのまま当てはめ、新しいコンポーネントおよび関係をより簡単に導入できる場合のメリットを考えてみましょう。

- 既存のコンポーネントを不必要に変更することなく新しいコンポーネントを導入できる。
- 既存の (まだ有効な) 関係を変更することなく新しい関係を確立できる。
- システム全体を作成し直すことなく関係を変更できる。
- コンポーネント・ベース・アプリケーションを既存のビジネス情報システムとインターフェース接続して、テクノロジー上の既存の投資を活かして新しい開発を進めることができる。

## 共通ソフトウェア・アプローチ

高レベルのプログラミング・モデルを利用することで、ソフトウェア開発者は、システムで直接使用できる有効なビジネス・コンポーネントを作成することができます。他のビジネスが同じプログラミング・モデルに従っていれば、組織は、より高いレベルの会社間システム統合を実現して、コラボレーション契約、併合、買収をサポートすることができます。オブジェクト指向の設計および実装の共通アプローチは、多大の利益をもたらし、ソフトウェア投資を無駄にしないためには、多くの面で欠かすことができません。

次のような重要な業界傾向によって、共通ソフトウェア・アプローチの普及は支えられています。

- コンポーネント・ベース・アーキテクチャーの業界全体のサポート
- Java プログラム言語、インターネット、社内イントラネットの急速な普及
- 堅固な企業サーバーでサポートされる分散ソフトウェア・トポロジー

これらの傾向が収束することにより、新しいコンピューティング・モデルをベースとした前例のないチャンスが生まれました。同時に、企業は、既存のアプリケーション・システムやデータへの実質的な投資を活かさなければなりません。最初から作成し直すには、あまりにもコストがかかりすぎます。

## ソフトウェアの可用性

IBM は、40 年近くに渡ってコンピューティング・インフラストラクチャーの業界トップを維持し、トランザクション、コンパイラー、オペレーティング・システム、通信、データベースなどのテクノロジーで非常に多くのことを経験してきました。この技術的な知識が、WebSphere で活かされています。WebSphere は、一般的な、業界全体のプラットフォームで利用することができ、幅広い範囲のオペレーティング・システム、通信プロトコル、データベースなどのリソースをサポートしています。

企業は、予想外の障害にも対応できるようにスタッフのクロス・トレーニングを行うことで、信頼性を向上させています。同様に、システムは、冗長性を組み込むことで信頼性を向上させ、利用可能なリソースに作業を移行することで予想外の障害に対応することができます。 WebSphere には、システムの単一の障害点を取り除き、システムの可用性を維持する、自動化作業負荷管理機能とフェイルオーバー機能が組み込まれています。これにより、可用性が高く、障害に強いシステムを開発することができます。

## ソフトウェアの再利用

コンポーネント・ベース・システムでは、個々のコンポーネントを作り直さずに繰り返し使用できるという、再使用性が約束されています。ただし、この約束は、当初はオブジェクト・システムに必要なサービスを提供するインフラストラクチャーがなかったために、大抵は実現されないままになっていました。その結果、オペレーティング・システム、データ管理メカニズム、および通信メカニズムのサポートは、アプリケーションに明示的にコーディングしなければなりませんでした。

WebSphere は、再使用可能コンポーネントのサポートに必要な分散オブジェクト・インフラストラクチャーを提供します。これで、開発者は、ビジネス機能を実装したコンポーネントの開発に専念することができます。コンポーネントは、さまざまなプラットフォーム、オペレーティング・システム、データベース、通信プロトコル、プログラム言語で再使用することができます。

## ソフトウェア開発タスクの分割

WebSphere プログラミング・モデルでは、アプリケーション開発を複数の異なる役割に分割します。各開発者が果たす役割は、開発するアプリケーションのタイプによって異なります。これらの役割により、開発チームのメンバーは、システムの他の部分の低レベルの技術の詳細な問題にわずらわされることなく、企業にとって一番役に立つタスクに専念することができます。たとえば、Java プログラマーは、Enterprise beans、サーブレットなどのタイプのコンポーネント開発に専念することができ、これらのコンポーネントがブラウザー・ベースのクライアントでどのように使用されるかを心配する必要はありません。同様に、Web サイトの開発者は、これらのコンポーネントの参照を Web ページにコーディングすることに専念することができ、実装の詳細に悩む必要はありません。

役割ベースの開発により、チームの各メンバーがシステムの分析、設計、開発、テスト、配置に専念できる、生産性の高いチームを作ることができます。各メンバーは、企業の利益に直結する役割の部分に専念することができます。

## ツール・セット

分散オブジェクト・システムには、再帰的実装パターンがあります。これらのパターンは、数の面では比較的少ないけれども、繰り返しが多く、ほとんど固定されています。このため、これらのパターンを使用してシステムを生成するツール・セットを作成することができます。これらのパターンから、プログラミング・モデルを作り、分散オブジェクト・システム開発に「ベスト・プラクティス」を定義します。

WebSphere 開発ツールは、開発者の生産性の向上に役立ちます。利用されているパターンに基づいてどの情報が必要であるかを判断し、目的に合った情報を探し出して管理し、この情報を使用してシステムを生成、配置、管理することができます。このため、ソフトウェア開発者はツールに格納されていない情報、つまりビジネス論理の設計および実装に専念することができます。

## スケーラビリティー

生産性は、成長によって低下する可能性があります。企業は、成長するにつれて、増加した作業負荷を処理する人を雇用しなければなりません。同じことは、成長した企業をサポートするコンピューター・システムにも当てはまります。コンピューター・システムが飽和状態になると、企業に提供されていたサービス・レベルは停滞状態に達し、場合によっては実際に低下することもあります。これにより、企業の成長に悪影響が及ぶこともあるので、できるだけ迅速に対応しなければなりません。

WebSphere では、コンピューティング力を追加することによって、つまり、コンピューターを追加するか、機能をより強力なプラットフォームに移行することによって、システムを成長させることができます。悪影響が及ぶ前に飽和状態のサインを見つけられるように、実行中のシステムをモニターします。また、動的な構成を行えるため、システムを利用不能にさせずに条件に対応することができます。これは、システムの可用性が一番重要なときに問題が発生した場合に特に重要です。

## オープン・スタンダードおよび投資の活用

他のシステムと通信できないシステムや、他のシステムで大規模な変更を必要とするシステムは、企業にとって大きな問題となります。このために、生産性が低下したり機能していないシステムへのこれまでの投資が無駄になる、という問題が生じることもあります。

WebSphere は、他のシステムにほとんど、あるいはまったく影響を与えることなく、さまざまな方法で他のシステムと通信することができます。これにより、WebSphere ベースのシステムの生産性をできる限り向上させ、また他のシステムも生産性を維持できるようになります。

オープン・スタンダードとは、コンピューター業界のさまざまなセグメントが共通機能の実装に同意した方法を定義したものです。 WebSphere は、Java 2、Enterprise Edition (J2EE) 仕様、 Common Object Request Broker Architecture (CORBA)、XA、 SSL (Secure Sockets Layer)、Kerberos、論理装置 (LU) 6.2、など、業界全体の標準をサポートしています。

企業は、情報技術システムに大きな投資を行っています。分散オブジェクト・システムは、既存のシステムと一緒に動作するように設計しなければなりません。 WebSphere は、既存のシステムと通信し、相互運用する方法を提供することで、これまでの情報技術への投資を活かします。

# 第5章 WebSphere Application Server スタンダード版およびアドバンスド版の紹介

IBM の WebSphere Application Server は、新世代のビジネス・アプリケーションを設計、実装、配置、および管理するための、ミドルウェアの総合セットをお客様にお届けします。これらのアプリケーションは、簡単な Web サイト・ストアフロントから、組織のコンピューティング・インフラストラクチャー全体の変更まで、多岐に渡ります。スタンダード版、アドバンスド版、エンタープライズ版の 3 つが用意されています。

## Application Servers スタンダード版と Application Servers アドバンスド版の相違点

Application Server スタンダード版とアドバンスド版は、機能的によく似ており、どちらを使用しても、強力なビジネス情報システムを作成することができます。ただし、Application Server スタンダード版と Application Server アドバンスド版には、大きな違いがいくつかあります。

- Application Server スタンダード版は、単一マシン・サーバー環境しかサポートしていません。 Application Server アドバンスド版は、複数マシン・サーバー環境をサポートし、命名、トランザクション、作業負荷管理などのサービスの基本サポートが組み込まれています。ただし、どちらの版も複数のクライアント・マシンからのアクセスをサポートしています。
- Application Server アドバンスド版は、複数のノードで EJB サーバーのクローンを簡単に作成し、作業負荷管理を行い、可用性を向上させることができる、EJB サーバーの複製をサポートしています。 Application Server スタンダード版では、複製は行えません。
- 2 つのアプリケーション・サーバーは機能に違いがあるため、それぞれへの管理インターフェースは若干異なります。Application Server アドバンスド版へのインターフェースを使用して Application Server スタンダード版環境を管理することはできず、また Application Server スタンダード版へのインターフェースを使用して Application Server アドバンスド版を管理することはできません。Application Server アドバンスド版は、データベース・リポジトリーに構成情報を格納しますが、 Application Server スタンダード版は、XML ファイルに格納します。

Application Server エンタープライズ版には Application Server アドバンスド版が含まれています。したがって、Application Server エンタープライズ版を購入すると、3 つの Application Servers のうちの任意の製品を使用して e-business ソリューションを実装することができます。

Application Server アドバンスド版には、 Application Server スタンダード版のすべての要素と、その他の要素が含まれているため、この章の残りの部分では、Application Server アドバンスド版に焦点を絞って説明します。

## Application Server アドバンスド版の紹介

WebSphere Application Server アドバンスド版は、以下の主要な機能を備えています。

- Java サーブレットおよび JavaServer Pages (JSP) を使用して実際の Web サイトを開発するためのツール。この機能は、Application Server スタンダード版でも使用可能です。
- EJB 仕様に従って作成された Enterprise beans を開発および配置するためのツールとの密接な統合。 Enterprise beans は、ユーザーの Web サイトとユーザーの非 Web コンピューター・システムとの間のブリッジとして機能することができます。
- Application Server アドバンスド版環境のコンポーネントを管理するためのグラフィカル・ユーザー・インターフェース (GUI) である WebSphere Administrative Console。この機能は、Application Server スタンダード版でも使用可能です。
- 拡張マークアップ言語 (XML) 文書を生成、検証、構文解析、および提示するための、一連のアプリケーション・プログラミング・インターフェース (API) (この機能は、 Application Server スタンダード版にも含まれています)。

### Application Server アドバンスド版環境

47ページの図7 に、 Application Server アドバンスド版を構成するコンポーネントを示します。

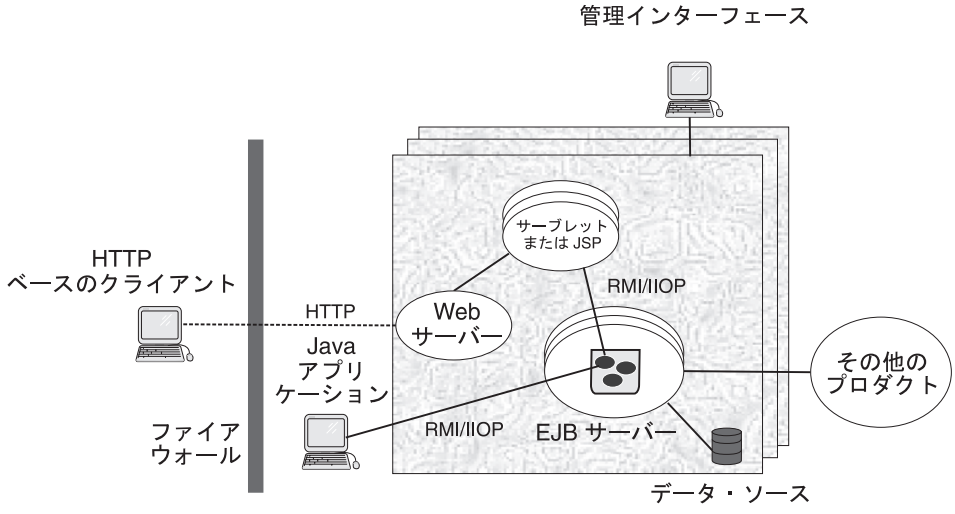


*図 7. Application Server アドバンスド版環境のコンポーネント*

これらのコンポーネントを組み合わせることにより、ユーザーの Web サイトに重点を置いた、強力な Java 中心の 3 層システムを作成することができます。次に、Application Server アドバンスド版の各部分について説明します。

### 管理サーバー

*管理サーバー* および*管理インターフェース* を利用することで、アプリケーション・サーバーおよびアプリケーション・プロセスを中央でモニターし、制御することができます。詳細については、50ページの『Application Server アドバンスド版における管理モデル』を参照してください。

### ブラウザー・ベース・クライアント

Application Server アドバンスド版で実行されるアプリケーションのクライアントは、一般に Java 対応ブラウザーで実行されます。これらのクライアントは、HTTP を使用して、Web サーバーとの間で情報をやり取りします。ブラウザー・ベース・クライアントには、アプレット (29ページの『アプレットおよびサーブレット』 を参照)、および JavaServer Pages (JSP) (33ページの『JavaServer Pages』 を参照) を含めることもできます。

### Web サーバー

組み込み Java セキュリティーの制限を受けるスタンドアロンの Java アプレットを除き、ブラウザー・ベース・クライアントのアプリケーションを使用するためには、ユーザーの Application Server アドバンスド版環境内の少なくとも 1 つのマシンに Web サーバーがインストールされている必要があります。

Web サーバーは、ブラウザー・ベースのアプリケーションと Application Server アドバンスド版の他のコンポーネントとの間の通信リンクを提供します。

WebSphere Application Server は、人気の高い Web サーバーの多くをサポートしています。 Application Server アドバンスド版 には、Apache サーバーの変更バージョンである IBM HTTP Server が添付されています。サポートされている Web サーバーについては、 IBM WebSphere Application Server のサイト (www.ibm.com/software/webservers/appserv) を参照してください。

### サーブレット・エンジン

Application Server アドバンスド版には、ユーザーの Web サーバーおよびその基礎になるオペレーティング・システムの両方と独立した、Java ベースのサーブレット・エンジンが含まれています。 Java サーブレットは、Web を介して要求および応答サービスを提供するためのフレームワークを作成することにより、 Web サーバー の能力を拡張します。サーブレットの詳細については、29ページの『アプレットおよびサーブレット』を参照してください。

### Enterprise beans

アドバンスド版は、Enterprise beans を完全サポートしています。 Enterprise bean は、他の Enterprise bean やその他の Java コンポーネントと組み合わせて 3 層の分散アプリケーションを作成することのできる、Java コンポーネントです。 *EJB* サーバー は、Enterprise beans の実行時環境を提供し、トランザクション管理、命名、セキュリティーなどの基本的なレベルのプログラミング・タスクを処理します。 Java アプリケーションは、Internet Inter-ORB Protocol を介した Java リモート・メソッド呼び出し (RMI/IIOP) を使用することで、 EJB サーバーと直接対話することができます。

Enterprise beans、EJB サーバー、およびコンテナーの詳細については、27ページの『Enterprise beans』 を参照してください。

## アプリケーション・モデル

Application Server アドバンスド版のアプリケーションは、データ・ストレージにリレーショナル・データベース・システムを使用するオブジェクト指向のビジネス論理から成り立ちます。アプリケーションは通常、Web クライアント (シック・クライアントまたはシン・クライアントのいずれか) と統合します。また、アプリケーション・サーバーで実行されている既存の手続き型アプリケーションと統合することもできます。

アプリケーションは、それぞれ異なる機能を実行する、次のコンポーネントから成り立ちます。

- HTML ページおよび JSP ページは、ユーザー・インターフェースとプログラムの流れを提供します。
- Enterprise beans には、アプリケーションのビジネス論理が含まれています。トランザクション処理とデータベース・アクセスを処理します。
- サーブレットは、アプリケーションの他のコンポーネントとの間で作業を調整します。また、Web ページのコンテンツを動的に生成することもできます。
- JavaBeans コンポーネントにより、他のタイプのコンポーネントと一緒に動作することができます。
- リレーショナル・データベースにより、Enterprise beans のパーシスタンスと照会機能が実装されます。アプリケーションでは、新規データベース、既存のデータベースのどちらでも使用できます。

## WebSphere プログラミング・モデル・エクステンション

プログラミング・モデル・エクステンションは、 Java プログラムに再使用可能なビジネス論理を提供します。 Application Server アドバンスド環境には、Java プログラマー向けのツール・セットが 2 つ用意されています。コマンド・パッケージと分散例外パッケージです。これらのツールについては、「*Enterprise Beans の作成*」を参照してください。ただし、これらのツールは、Enterprise beans 以外でも使用できます。

コマンド・パッケージは、分散アプリケーションがリモート要求を 1 つにまとめて、個々のリモート呼び出しの数を削減するための方法です。リモート呼び出しはリソースを消費するので、コマンド・パッケージを利用することで分散アプリケーションのパフォーマンスを向上させることができます。また、コマンド・パッケージは、要求を作成する汎用的な方法でもあります。パッケージは、ローカルやリモートでのコマンド実行と、サーバーの実装に依存しないコマンド実行に共通の方法となります。サーバー (Enterprise bean、JDBC サーバーなど) が、コマンドのターゲットになることもあります。

分散例外パッケージは、分散アプリケーションでの例外管理に役立ちます。複雑な分散アプリケーションを作成していると、例外の処理で選択が求められます。 1 つの選択肢は、各例外を明示的に管理し、それぞれ名前でキャッチして、再スローする方法です。この場合、オリジナルの例外に関する情報がなくなることはありませんが、例外の数が増加すると、コードの収拾がつかなくなる可能性があります。もう 1 つの選択肢は、例外のグループをキャッチしたら、1 つの例外をスローする戦略を採用する方法です。この方法では、複数の例外の管理が可能となりますが、例外はアプリケーションを通るので情報を失うことになります。分散例外パッケージを利用すれば、例外のシーケンスをチ

ェーニングして、 1 つのスロー可能なオブジェクトにすることができます。例外チェーンにより、直前の例外を失うことなく、ある例外に応答する例外をスローすることができます。また、チェーンから例外を検索することもできます。

## Application Server アドバンスド版における管理モデル

WebSphere Application Server は、EJB サーバーおよびその他のリソースを中央で管理できるようにします。管理サーバーは、サーブレット、JSP ファイル、Enterprise beans、および EJB サーバーを管理します。この管理は、管理サーバーのインターフェースである WebSphere Administrative Console を使用して、 WebSphere Application Server 管理者が指示して行います。

WebSphere Application Server における*管理ドメイン* は、管理ノードと呼ばれるホスト・マシンの集合です。各管理ノードは、*管理サーバー* を実行します (管理サーバーは EJB サーバーでもあります)。ノードの管理サーバーは、そのノードにあるリソースの構成、モニター、および管理の責任を負います。リソースには、EJB サーバー、コンテナー、配置された bean、JSP ファイル、Java サーブレット、アプリケーションなどの動的なオブジェクトが含まれます。リソースにはまた、ドメイン内のリソースのセキュリティーを定義するために使用されるメソッド・グループやポリシーなどのオブジェクトも含まれています。

リソース bean は、コンテナー管理された永続 (CMP) エンティティー bean です。リソースに関連する永続データ (たとえば EJB サーバーの名前、現行状態、実行可能ファイル) は、中央のデータ・リポジトリーに保管されます。管理サーバーは、リポジトリー内に保管されたリソース情報のアクセス、定義、および変更を行うために、リポジトリー・サーバーと通信します。管理サーバーはまた、タスクを代行したり要求に応答したりするために、他の (リモート) 管理サーバーとも通信します。Application Server アドバンスド版とともにパッケージされている IBM DB2 リレーショナル・データベースは、リポジトリー・サーバーとして機能します。ほかに、Oracle、Sybase、InstantDB などのデータベースも使用できます。

管理は、リポジトリー・サーバー内のリソース bean へのメソッド呼び出しを介して行われます。WebSphere Administrative Console は、管理サーバーに対して、ドメイン内のリソースのアクセスや変更を行うように要求します。管理サーバー内では、セッション bean がリソース bean 内のメソッドを起動します。各リソース bean には、属性値を獲得および設定するためのメソッドを含む属性クラスが関連付けられています。

1 つのドメイン内のすべての管理サーバーは、そのドメイン内のリソース用に中央記憶装置を共用します。実行されているノードとは無関係に、どの管理サーバーも、他のノード上のリソースの特性または状況を表示したり変更したりすることができます。ある管理サーバーが、別のリモート・ノードで実行されているリソース上のメソッドを呼び出すと、ローカル管理サーバーに代わってリモート管理サーバーがそのメソッドを代行するようになります。

リソースは、オブジェクト・タイプを相互に関連付けるオブジェクト・タイプ階層をモデルとして作られます。その他のオブジェクト・タイプは、サーバー・グループなどのエンティティーを表します。関連オブジェクトは、ツリー階層でその上位にあるオブジェクトからメソッドを継承します。

EJB サーバーなど、管理ドメイン内にあるオブジェクトの中には、コピー (*モデル化*) することによって、複製元のオブジェクトと同一の機能を果たす複製 (*クローン*) を作成できるものがあります。これにより管理者は、複数のノードでサーバー機能を複製し、可用性と効率を向上させることができます。リソースを複製した後でモデルを変更すると、その変更はすべての複製に自動的に伝えられます。リソースのモデルを管理することにより、サーバーなどのリソースの複数の複製を効率良く管理することができます。

複製できるリソースは次のとおりです。

- アプリケーション・サーバー
- EJB コンテナー
- Enterprise beans
- サーブレット
- サーブレット・エンジン
- Web アプリケーション

## 管理ツール

WebSphere Administrative Console は、Application Server アドバンスド版への管理インターフェースです。これは、リソースの構成やセキュリティー・ポリシーの設定から、サーバーの開始や bean の配置、さらにシステム障害の識別と応答、あるいは使用パターンのモニターまで、さまざまな管理作業に使用することができます。 WebSphere Administrative Console によってサポートされるタスクは、構成、オペレーション、セキュリティー、障害追及、パフォーマンス、およびデータ保管の 6 つのカテゴリーに分類されます。

WebSphere Administrative Console は、ある管理ドメイン内のすべてのリソースの中央化された階層ビュー、管理オペレーションを実行するための指示、リソ

ースの属性を表示および変更するためのフォーム、JAR ファイルのための中央ブラウズ機能、クリティカルなイベントをモニターするためのメッセージ・ウィンドウ、およびコンテキストに依存したヘルプを提供します。WebSphere Administrative Console は、ユーザー・コマンドに応じてリポジトリー内の情報を変更し、管理ドメインの構成および状況に対する変更を反映します。

## 拡張マークアップ言語 (XML)

XML は、インターネットを介した文書 (およびその他の構造化情報) を交換しやすくすることを目的として作成されたものです。簡単に言えば、文書マークアップ言語を定義するための標準です。マークアップ言語は、次の機能のうちの 1 つ以上を備えたエレメント (多くの場合、タグと呼ばれます) の集合です。

- 文書の構造を記述する。
- 文書の内容を記述する。
- 文書をユーザーに提示する方法を制御する。

HTML は Web ベースの文書のマークアップ言語として最も広く使用されていますが、多くの制限があります。 HTML タグは、Web ページのビジュアルな表示を表すものであって、論理構造を実際に指定するものではありません。HTML ユーザーが使用できるタグは比較的限られ、また市販の Web ブラウザーは HTML 標準に含まれないタグをサポートしないため、ユーザー独自のタグを作成することもできません。

表示を制御するタグが文書内容を記述するタグと同じファイルに入っていることにより、HTML はさらに制約を受けます。HTML 4 および Cascading Style Sheets により、HTML 作成者は文書内容を表示から分離して扱うことができますが、HTML 4 は文書の内容を記述する機能が制限されています。

XML にこのような制限はありません。 XML ユーザーは、独自のカスタム・タグ・セットを定義したり、一般に使用されている文書タイプ定義 (DTD) のタグを使用したりすることができます。 XML タグは、文書の内容と構造を指定するものです。表示は、文書の内容から切り離されます。

Application Server アドバンスド版に含まれている *XML 文書構成サービス* を利用することで、サーバー側の XML 処理を実装するサーブレットおよびアプリケーションを開発することができます。このサービスには、管理インターフェースを使用しないでサーブレット構成パラメーターを設定するための、一連

の API が組み込まれています。この代替方式では、以下のものを含む XML サーブレット構成ファイル (これは servlet_instance_name.servlet という名前の XML 文書です) が作成されます。

- サーブレット・クラス・ファイルの名前。
- サーブレットの記述。
- サーブレット初期化パラメーター。
- そのサーブレットが呼び出すことのできる各 JSP ファイルの汎用リソース ID (URI) を含むページ・リスト。このページ・リストには、デフォルト・ページ、エラー・ページ、および、HTTP 要求に名前が表示されたときにロードされる 1 つまたは複数のターゲット・ページを含めることができます。

## Application Server アドバンスド版によって使用されるサービス

Application Server アドバンスド版は、主としてユーザーのビジネスのうちの Web による部分を扱いますが、EJB サーバーは、Web アプリケーションと非 Web アプリケーションを接続してお客さまのビジネス・システムのすべてを網羅するための、ブリッジの役割を果たします。このセクションでは、分散アプリケーションを開発および使用できるようにするために行う必要のある一般作業のうちのいくつかを説明します。また、Application Server アドバンスド版でこれらの各作業に取り組むために使用されるツールについても説明します。

### 命名サービス

オブジェクト指向の分散コンピューティング環境では、クライアントは、クライアントとオブジェクトがすべて同じマシンにある場合と同様にオブジェクトを見付けて識別できるメカニズムを備えていなければなりません。命名サービスは、このメカニズムを提供します。EJB サーバー環境では、命名サービスへの共通フロントエンドを用意するために、 Java Naming and Directory Interface (JNDI) が使用されます。

JNDI は Java アプリケーションに命名およびディレクトリー機能を提供しますが、API は、命名およびディレクトリー・サービスの特定インプリメンテーションとは独立しています。この独立性があるため、JNDI API の背後でアクセスすることにより、異なる命名およびディレクトリー・サービスを使用できるようになります。したがって、Java アプリケーションは、Lightweight Directory Access Protocol (LDAP) や DNS (Domain Name System) などの、既存の多くの命名およびディレクトリー・サービスを使用することができます。

## トランザクション・サービス

*トランザクション* は、ある整合状態から別の整合状態にデータを変換する、一連のオペレーションです。EJB サーバーは、Java Transaction API (JTA) で定義されたメカニズムを使用することにより、EJB アプリケーションのトランザクションを管理します。

ほとんどの用途では、Enterprise bean の開発者は、トランザクションの管理に関連するタスクを EJB サーバーに代行させることができます。開発者は、トランザクションの配置記述子属性を設定することによって、この委任を行います。 Enterprise bean のコード自体には、トランザクション論理を含める必要はありません。

トランザクションの詳細については、15ページの『トランザクション: 分散環境におけるデータ整合性とパフォーマンスの保証』を参照してください。

## セキュリティー・サービス

Application Server アドバンスド版環境におけるセキュリティー・サービスの主要コンポーネントは、セキュリティー Enterprise bean を含む EJB サーバーです。システム管理者は、セキュリティー・サービスを管理するときには、セキュリティー bean を操作します。

ある EJB クライアントが認証されると、そのクライアントは、操作する Enterprise bean にあるメソッドの呼び出しを試みることができます。そのメソッド呼び出しに関連するプリンシパルが、メソッドの呼び出しに必要なアクセス権を所有している場合、メソッドは正常に呼び出されます。これらのアクセス権は、アプリケーションごとに設定することができ (管理者が定義した Web およびオブジェクト・リソースのセット)、またメソッド・グループごとに設定することができます (管理者が定義した Java のインターフェースとメソッドのペアのセット)。 1 つのアプリケーションに複数のメソッド・グループを含めることができます。

一般に、メソッドを呼び出すプリンシパルは、複数の Web サーバーおよび EJB サーバーを通じて、その呼び出しと関連付けられます (この関連は*委任* と呼ばれます)。この方法でメソッド呼び出しを委任すると、EJB クライアントのユーザーは、認証を 1 回行うだけで済みます。ユーザーの認証情報を複数の Web サーバーに伝えるためには、 HTTP Cookies (クライアント・コンピューターのハード・ディスクに格納されている識別ファイル) が使用されます。これらの Cookie の存続時間は、ブラウザー・セッションの存続時間と同じになります。

## 作業負荷管理サービス

作業負荷管理 (WLM) は、WebSphere Application Server 環境でのタスク処理の分散を最適化します。着信要求は、最も効果的に要求を処理できるアプリケーション・サーバーなどのオブジェクトに分散されます。

作業負荷管理サービスにより、クライアント要求はシステム内の各マシンの能力に従って分散され、より高いクライアント負荷に対応できるようにシステムを簡単に拡張できるようになります。また、要求をリダイレクトすることでフェイルオーバーをサポートし、保守および管理を簡易化します。

WebSphere Application Server 環境には、作業負荷を管理する方法が複数用意されています。

- 個々の Enterprise beans やサーブレットからアプリケーション・サーバー全体に至るまで、アプリケーション・オブジェクトの複製およびモデルを作成することができます。 Application Server アドバンスド版の WLM サービスは、 Enterprise bean と EJB サーバーの複製間で処理要求を分散します。
- サーブレットのリダイレクトを使用して、サーブレットの複製にクライアント要求を転送します。
- EJB サーバーを組み合わせて、EJB サーバー・グループにまとめることができます。クライアントは、単一のサーバーであるかのように、これらの EJB サーバー・グループにアクセスします。
- WLM 対応管理サーバーが、管理要求のフェイルオーバーと負荷分散を行います。

## 開発環境

WebSphere Studio は、WebSphere Application Server アプリケーション開発環境です。これを使用して、個人的な Web ページから、e-business アプリケーションのフロントエンドの役割を果たす Web サイトに至るまで、あらゆるアプリケーションを作成することができます。 WebSphere Studio には、HTML コンテンツを開発するツール・セットが用意されており、他のコンテンツ開発ツールと統合することができます。

VisualAge for Java は、Java プログラム開発の完全なサイクルをサポートする統合開発環境です。VisualAge for Java は、正式には Application Server スタンダード版またはアドバンスド版 の一部ではありませんが、 WebSphere Application Server 環境と密接に統合されています。この統合により、VisualAge 開発者は VisualAge プログラムを離れることなく Java プログラムを開発、配置、およびテストすることができます。また、複雑な企業環境を管理できる環境が提供され、ルーチン・ステップを自動化することができます。

## 資料

スタンダード版およびアドバンスド版のインフォセンターは、 WebSphere Application Server のインストールと構成、およびアプリケーションの計画、開発、配置、トラブルシューティングに役立つ情報を提供します。スタンダード版およびアドバンスド版のインフォセンターはどちらも、 WebSphere Application Server ライブラリー・ページ (www.software.ibm.com/webservers/appserv/library.html) からアクセスすることができます。

# 第6章 WebSphere Application Server エンタープライズ版

この章では、WebSphere Application Server エンタープライズ版の内容について簡単に説明します。また、Application Server エンタープライズ版が実行される環境、および Application Server エンタープライズ版にパッケージされている追加製品についても説明します。

## Application Server エンタープライズ版を使用する理由

Application Server エンタープライズ版には、アドバンスド版に含まれるすべての製品が含まれ、さらに以下の主要な製品コンポーネントが追加されています。

- Component Broker。これは、分散コンピューティングのためのエンタープライズ・ソリューションであり、コンポーネント・ベースの分散ソリューションを開発および配置するためのスケーラブルで管理可能な実行時環境を提供します。これは、Object Management Group (OMG) の共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) に含まれているオープン・スタンダードを、統合された形で完全に実装したものです。さらに、Component Broker には EJB Specification が別個に実装されています。これは、アドバンスド版に含まれている実装と一緒に、あるいはそれに代えて使用することができます。
- TXSeries。これは、多様で複雑なネットワーク内の複数プラットフォームにまたがるトランザクション・アプリケーションの作成をサポートし、単純化する、2 つの一般的なミドルウェア・パッケージ (CICS および Encina) を含んでいます。 TXSeries アプリケーションは、エンタープライズ相互の統合を提供するほかに、高水準のスケーラビリティー、可用性、保全性、長寿命、およびセキュリティーを提供します。

Application Server エンタープライズ版には、顧客指向でしかもサプライヤーも重視した現代ビジネスのすべての局面に渡るアプリケーションを構築するための、完全なツール・セットが含まれています。強力な Web サイトを構築したい場合にも、異種の非 Web ビジネス・コンピューティング・リソースを結び合わせる分散型のトランザクション・アプリケーションを作成したい場合にも、Web システムと非 Web システムを統合したい場合にも、あるいはこれらのすべての目標を達成したい場合にも、Application Server エンタープライズ版はユーザーのお役に立ちます。

次に、Application Server エンタープライズ版が実行される基礎になっている、主要な環境およびサービスについて説明します。また、Application Server エンタープライズ版と一緒に使用するためにライセンス付与される、その他の製品についても簡単に説明します。次に続く 2 つの章では、Application Server エンタープライズ版のコンポーネントを紹介し、これらの各コンポーネントの役割を説明します。

## Application Server エンタープライズ版製品で使用される下位レベルのサービス

Application Server エンタープライズ版製品の多くは、以下のサービスのうちの 1 つまたは複数に依存して、セキュリティー、命名、およびリモート・プロシージャー呼び出しなどの下位レベル・タスクを処理します。

- Open Group の分散コンピューティング環境 (DCE)。詳細については、『分散コンピューティング環境 (DCE)』を参照してください
- The Object Management Group (OMG) の Component Object Request Broker Architecture (CORBA)。詳細については、60ページの『共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA)』を参照してください
- Microsoft Corporation の Component Object Model (COM)。詳細については、61ページの『Component Object Model (COM)』を参照してください

Application Server エンタープライズ版には、Component Broker 製品に組み込まれた EJB Specification (および関連の Java 仕様) の実装も含まれています。この実装に関する情報は、Component Broker に関する一般的な説明で提供されています。詳細については、65ページの『第7章 Component Broker の紹介』を参照してください。

### 分散コンピューティング環境 (DCE)

DCE は、Component Broker または TXSeries を使用する分散トランザクション処理環境を、ハードウェア、オペレーティング・システム、ネットワーク・トランスポート、およびアプリケーション・ソフトウェアが異なる複数のマシン間で、シームレスに実行できるようにします。DCE 層は、個々のマシンの基本オペレーティング・システムを拡張して、分散コンピューティングのための共通のインフラストラクチャーを提供します。DCE によって提供される標準インターフェースを使用することにより、アプリケーションを他の DCE プラットフォームと相互に作用したり、それらのプラットフォームに移植したりできるようになります。以下のセクションでは、Application Server エンタープライズ版製品によって使用される DCE サービスについて説明します。

### リモート・プロシージャー呼び出し (RPC)

DCE サポートの中核となっているのは、*RPC* です。RPC は分散システム内のプロセス間で、ネットワーク上の透過な通信を提供します。複数のプロセスは、RPC を使用することにより、それらがセルと呼ばれる管理単位内の同一マシンにある場合にも、別のマシンにある場合にも、まったく同様に通信することができます。DCE セキュリティー・サービスを使用して RPC を認証することができます。*認証済み RPC* に損傷がないかどうかを検査したり、プライバシー保護のためにこれを暗号化したりすることができます。DCE は、1 つのクライアントがサーバーとの間で複数の RPC 会話を並行して行ったり、1 つのサーバーが多数の並行したクライアント要求を処理したりできるように、*マルチスレッド化* を使用します。

### セル・ディレクトリー・サービス (CDS)

CDS は、名前だけを使用してネットワーク・リソースにアクセスできるようにするために、ネーム・スペース を提供します。アプリケーションは、リソースのアドレスを知っている必要がありません。(代表的なネットワーク・リソースとして、サーバー、ユーザー、ファイル、ディスク、または印刷キューがあります。) さらに、リソースが移動しても、同じ名前で突き止めることができ、アプリケーション・コードを変更する必要はありません。

CDS サーバーは*クリアリングハウス* と呼ばれるデータベースを管理します。このクリアリングハウスには、DCE セル内のネットワーク・リソースの名前と属性 (ロケーションも含まれます) が入っています。あるネットワーク・リソースに対して要求が出されると、 CDS サーバーがリソースを探し出します。

DCE ディレクトリー・サービスは、セルの外側にあるリソースを識別するためのグローバル名もサポートします。

### DCE セキュリティー・サービス

DCE セキュリティー・サービスは、DCE セル内で、セキュア (機密保護された) 通信と、ネットワーク・リソースへの制御されたアクセスを行えるようにします。このサービスは、DCE *プリンシパル* (ユーザー、サーバー、および DCE 対応クライアント) の識別を検証し、それらのプリンシパルが、使用を許可されたネットワーク・リソースだけにアクセスできるようにします。DCE セキュリティー・サービスは、以下のことを行います。

- セルのセキュリティー・データベースに入っているセキュリティー情報の、中央ソースを管理する。
- ユーザーなどの対話式プリンシパルの識別を検証し、そのプリンシパルが DCE にログインできるようにする。これは、ログイン・コンテキストの確立と呼ばれます。

- プリンシパルおよびサービスにチケット を付与し、セキュアな通信が行えるようにする。
- プリンシパルの資格を証明し、リソースへのプリンシパルのアクセス権限を制御する。
- CICS 領域などの非対話式プリンシパルの識別を検証し、そのプリンシパルが対話式ユーザー・ログインに相当する操作を行えるようにする。この方法により、非対話式プリンシパルは、それを開始したプリンシパルの識別のもとで実行する代わりに、独自のログイン・コンテキストを確立することができます。
- DCE セル内のネットワーク・リソースへのアクセスに関してプリンシパルに付与する許可を制御する。 DCE セル内の各オブジェクトには、どのユーザーがどの操作を実行できるかを指定する、*アクセス制御リスト (ACL)* が関連付けられています。 ACL は、ファイル、ディレクトリー、およびレジストリー・オブジェクトと関連付けることができ、内部オブジェクトへのアクセスを制御するために任意のアプリケーションによって実装することができます。

DCE セキュリティー・サービスはセキュリティー・サーバーとして実装されます。このサーバーは、セキュリティー・データベース (DCE *レジストリー・データベース* とも呼ばれます) へのネットワーク・リソースのセキュリティー情報の保管を保守します。

#### 分散タイム・サービス (DTS)

システム・クロックの変動を補正するために、DCE DTS は、DCE セル内のサーバーのすべてのシステム・クロックが同期するようにします。これは、サーバーが異なる時間帯に属している場合には、特に重要です。タイム・サービスは、認証および許可サービスのオペレーションの信頼性を保証するためにも欠くことができません。

### 共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA)

OMG は、ネットワークを介したオブジェクト指向アプリケーションの開発を簡単にするために、CORBA 仕様を作成しました。 CORBA オブジェクトは、任意のオブジェクト指向プログラム言語に実装される標準ソフトウェア・オブジェクトです。オブジェクト・リクエスト・ブローカー (ORB) は、クライアント・プログラムとオブジェクトとの間でのメッセージ転送を仲介します。クライアント・プログラムがオブジェクト上のメソッドを呼び出すと、ORB はその要求を代行受信し、そのメソッドを実装しているオブジェクトを探します。

メソッド呼び出しの結果は、ORB によってクライアント・プログラムに戻されます。プログラマーからは、すべての作業が 1 つのコンピューター・システムで行われているように見えます。

Internet Inter-ORB Protocol (IIOP) は、異なる ORB インプリメンテーション間での通信を可能にします。IIOP は TCP/IP を基礎にしていて、CORBA によって定義された追加のメッセージ交換プロトコルを含んでいます。

各種の Application Server エンタープライズ版製品は、以下の ORB のうちのいずれかをサポートします。

- IONA Orbix ORB
- IBM Component Broker ORB
- IBM Java ORB

詳細については、33ページの『CORBA』 を参照してください

## Component Object Model (COM)

Microsoft COM は、透過的かつ個別にアップグレードすることのできる個々のコンポーネントからなるアプリケーションを開発するためのモデルです。COM を使用すると、Windows アプリケーションを単一エンティティーではなく複数のコンポーネントとして開発することができ、アプリケーションの分散および保守が容易になります。

TXSeries Encina も Component Broker も、どちらも Windows NT システムの COM 機能を提供します。

- Encina アプリケーションは、COM コンポーネントを使用して、 Encina クライアントの機能を透過的に処理することができます。クライアントは、Encina COM コンポーネントのインスタンスを作成し、これらのコンポーネント上の標準メンバー機能を呼び出すことで、 Encina サーバーに問い合わせたり、トランザクションを管理するなどのタスクを実行することができます。

  COM コンポーネントは、標準 Encina Transactional Interface Definition Language (TIDL) ファイルから作成できます。生成されるダイナミック・リンク・ライブラリー (DLL) と、 Encina COM DLL やその他の Encina 以外の COM DLL を一緒に使用して、 Encina クライアントを開発することができます。
- Component Broker アプリケーションは、COM コンポーネントを使用して、管理下のオブジェクトにアクセスすることができます。管理下のオブジェクトは、オブジェクトにアクセスする場合のプロキシーとなる COM コンポー

ネントによって実行時にラップされます。また、COM コンポーネントを使用して、リモートの CORBA オブジェクトにアクセスすることもできます。

COM コンポーネントは、標準 Interface Definition Language (IDL) ファイルから作成できます。生成される DLL と、その他の COM DLL とを一緒に使用して、Component Broker クライアントを開発することができます。一部の Component Broker オブジェクト・サービス用の COM コンポーネントも用意されています。

COM コンポーネントの詳細については、35ページの『Microsoft COM』 を参照してください。

## Application Server エンタープライズ版で使用可能なその他のツール

Application Server エンタープライズ版には、Application Server エンタープライズ版の主要なツールによって必要とされる (あるいは併用することが推奨される)、以下の追加製品が組み込まれています。

- IBM DB2 — DB2 は分散リレーショナル・データベースであり、TXSeries および Component Broker と一緒に、リソース・マネージャーとして使用することができます。DB2 は、Application Server アドバンスド版に含まれている EJB 管理サーバーで使用することができ、また、Component Broker に含まれている EJB 管理サーバーでは、必ず使用する必要があります。また、Application Server アドバンスド版、エンタープライズ版の両方で、コンテナー管理パーシスタンス (CMP) エンティティー bean に関連した永続データを保管するために使用することもできます。
- IBM HTTP Server — IBM HTTP Server は、一般に使用されている Apache Web サーバーに基づいた、強力な Web サーバーです。 Web サーバー機能をすべて提供するほかに、セキュア・トランザクションのための拡張 SSL も提供します。 Application Server アドバンスド版には、人気の高いほとんどの Web サーバーに対応するプラグインが用意されているため、ユーザーの Web サーバーを Java アプリケーション・サーバーに拡張することができます。
- IBM MQSeries — IBM MQSeries® に属する製品では、メッセージとキューを使用することにより、アプリケーション・プログラムの非シリアルの非同期方式通信を実現することができます。 MQSeries の中核となっているのは Message Queue Interface (MQI) です。これは、さまざまなプラットフォームでアプリケーションが透過的に通信できるようにする、高レベルのプログラミング・インターフェースです。MQI はネットワーク・インターフェースを管理し、メッセージの送達を保証し、通信プロトコルを処理し、システムの問題からの回復を処理します。

- IBM VisualAge for Java Enterprise Edition — VisualAge for Java は、 Java ベースのビジネス情報システムを構築するための多くの機能が組み込まれた、強力な統合開発環境 (IDE) です。この強力な IDE は、Enterprise JavaBeans および JavaBeans の仕様に従って作成された Java アプリケーションおよびコンポーネントの開発およびテストをサポートします。詳細については、8ページの『VisualAge for Java』 を参照してください
- IBM VisualAge C++ Professional Edition — VisualAge C++ は、マルチプラットフォームのオブジェクト指向アプリケーション開発のための、豊富な環境およびツール・セットを提供します。この開発環境は、高性能なパフォーマンスおよび計算が重点を占めるアプリケーションのためには、特に大切です。その Open Class® Library は、 AIX および Windows NT で強力なアプリケーションを構築するための、拡張クラス・ライブラリーおよびフレームワークを提供します。

# 第7章 Component Broker の紹介

Component Broker は、分散コンポーネント・ベースのソリューションを開発および配置するためのスケーラブルで管理可能な環境を提供します。これは、Object Management Group (OMG) の共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) イニシアチブに含まれているオープン・スタンダードの統合実装です。 Component Broker の使いやすいフレームワークには、CORBA インターフェースの下位レベルの詳細の多くが隠されています。Windows NT、AIX、Solaris、および OS/390 の各プラットフォームで利用できます。

この概説では、以下の Component Broker フィーチャーについて紹介します。


- 『Component Broker のフィーチャー』
- 68ページの『アプリケーション・アーキテクチャー』
- 69ページの『アプリケーション・アダプター』
- 69ページの『オブジェクト・サービス』
- 73ページの『システム管理』
- 75ページの『開発ツール』


## Component Broker のフィーチャー

Component Broker は、基本的にはオブジェクト・サーバーです。これには、Component Broker サーバーで実行されるビジネス・オブジェクトを作成するために最適化された、開発環境が備わっています。このサーバーは、 Component Broker Connector (CBConnector) というランタイム・パッケージと、 VisualAge Component Development Toolkit (CB Tools) の両方から成り立ちます。実行時パッケージにはサーバーが備わっています。このサーバー内でビジネス・オブジェクト・コンポーネントが実行され、一連の管理ツールによって管理されます。

オブジェクト・サーバーとして、Component Broker は、クライアントがオブジェクト指向のミドルウェアを介してバックエンド・システムにアクセスできるようにするアプリケーション環境を提供します。このシステムは、デスクトップからメインフレームの最大規模のクラスターまで、あらゆるものを組み込むことができる、スケーラブルなインフラストラクチャーを提供します。

Component Broker の実行時環境は、CORBA モデル、または Sun Microsystems の Enterprise JavaBeans (EJB) 仕様のモデルに準拠する、 C++ および Java ベ

ースのビジネス論理の実行をサポートします。 Enterprise JavaBeans 仕様は、移送可能な、プラットフォームから独立した、再使用可能コンポーネント・アーキテクチャーを提供します。これらのコンポーネントは、多様なサービスおよび機能に簡単にアクセスするための、強力なマルチスレッド・サーバーで実行されます。

Component Broker のオブジェクト・サービスは、OMG モデル、および構成可能かつ拡張可能なオブジェクト・インターフェースの集合である Component Broker Managed Object Framework (MOFW) に基づいて、統合形式で使用することができます。提供される主要オブジェクト・サービスとしては、データ・キャッシュおよびプリフェッチ、並行性制御、イベント、外部化、オブジェクト識別、ライフサイクル、命名、照会、セキュリティー、およびトランザクションがあります。アプリケーション・アダプターは、これらのインターフェースを拡張および専門化して、管理下のオブジェクトに、オブジェクト指向データベースに似たホームとコンテナーを提供します。

Component Broker には、 CORBA 2.0 に準拠したオブジェクト・リクエスト・ブローカー (ORB) が組み込まれています。この ORB は、Managed Object Framework と、Component Broker が提供するオブジェクト・サービスによって、大きなカプセルにされています。ORB は、他の Internet Inter-ORB Protocol (IIOP) およびクライアントとの相互運用を容易にします。Component Broker は、Internet Inter-ORB Protocol (IIOP) を直接に使用することにより、あるいは Remote Method Invocation over IIOP (RMI/IIOP) を使用することにより、クライアント・アクセスをサポートします。これにより、Java クライアント (アプレット、アプリケーション、またはサーブレット)、C++ クライアント、および Microsoft ActiveX® または Visual Basic クライアントが、そのサーバーで実行されているビジネス・オブジェクトにアクセスできるようになります。

Component Broker アプリケーションは、 DB2、Oracle、CICS、IMS、および MQSeries にあるリソースにアクセスすることができます。これらのリソース・マネージャーによってデータ・オブジェクトが戻されたビジネス・オブジェクトは、分散トランザクションに参加することができます。Component Broker は、CORBA Object Transaction Service (OTS) の実装を介して、これらのリソース・マネージャーのすべてに関する外部コミット・コーディネーターとして機能します。

管理者は、システム管理ツールを使用して、分散オブジェクト・コンピューティング環境を制御することができます。これらのツールを使用すると、配置構成をモデル化し、分散オブジェクト・パラダイムによって導入された抽象の実際の管理を行うことができます。管理者は、サーバーおよびサーバー・グルー

プを追加することにより、構成を拡大することができます。また、コンテナー管理によってサービス品質を更新したり、追加のコンピューティング・リソースをオブジェクト・サーバー・プールに追加して環境を拡張したりすることができます。

Component Broker で実行されるアプリケーションの開発方法としては、次の 3 つの方法があります。

- Component Broker には、CORBA ベースのアプリケーションを作成するためのオブジェクト・ビルダー・ツールが用意されています。システムの設計は、Rational Rose ビジュアル・モデル化ツールからオブジェクト・ビルダーにインポートできます。これらの設計をインポートしたら、実装用のテンプレートを使用することができます。開発者は、実装フレーム内にビジネス論理を埋め込むだけです。それ以外のことは、オブジェクト・ビルダーが行います。オブジェクト・ビルダーは、 Component Broker サーバーでアプリケーションをテストするために必要なコード、 MAKE ファイル、およびアプリケーション構成情報を生成します。また、オブジェクト・ビルダーは、開発者チームによる大規模なシステム開発もサポートします。
- 他のツール (VisualAge for Java など) で作成された Enterprise beans も、Component Broker に配置して、EJB サーバーとして使用することができます。開発者は、EJB プログラミング・モデルを使用することで Enterprise アプリケーションを作成できます。オブジェクト・ビルダーは、Component Broker 実行時に実行する Enterprise beans の配置ツールとして使用できるので、コンテナー管理パーシスタンス (CMP) を持つエンティティー beans をデータベースや既存のアプリケーションにマッピングしやすくなります。このサポートは、CORBA ベースのビジネス・オブジェクトを既存のリソース・マネージャーにマップするために使用されるものと同じテクノロジーに基づいて行われます。Enterprise beans は、CORBA ベースのビジネス・オブジェクトで使用可能なものと同じ、オブジェクト・サービスの実装を利用します。
- オブジェクト・ビルダー・ツールと Component Broker ランタイムを使用して、 Enterprise beans と Component Broker ビジネス・オブジェクトを組み合わせた分散オブジェクト・アプリケーションを簡単に作成、テスト、および配置することができます。

Component Broker アプリケーションは、 WebSphere プログラミング・モデル・エクステンションも利用できます。詳細については、49ページの『WebSphere プログラミング・モデル・エクステンション』 を参照してください。

## アプリケーション・アーキテクチャー

Component Broker アプリケーションは、3 層アプリケーションとして設計されます (13ページの『3 層のクライアント / サーバー・コンピューティング』を参照)。各層の内容を以下に要約します。

- *第 1 層* — クライアント・アプリケーションは、ユーザーが直接対話できる C++ プログラム、Java アプレット、または Visual Basic プログラムにすることも、また独自のクライアントを持つ Web サーバーまたはアプリケーション・サーバーにすることもできます。クライアントは、CORBA 準拠の ORB 上のプロキシー・オブジェクトを介して、サーバー上のコンポーネントにアクセスすることができます。 Component Broker クライアントのプログラミング・モデルは、 ActiveX/COM オブジェクトなどのプログラミング・モデルを使用するクライアントからコンポーネントへのアクセスもサポートします。
- *中間層* — アプリケーションのビジネス論理が入っているコンポーネントを Component Broker サーバーに配置するときの実行時環境。コンポーネントは、CORBA ベースのビジネス・オブジェクトでも、Enterprise beans でも可能です。サーバーのコンポーネントは、1 つのマシンに置くことも、多くの異なるマシンに置くこともできます。
- *第 3 層* — DB2、Oracle、CICS、 IMS など、各種リソース・マネージャー内のデータ。中間層のアプリケーション・アダプターにより、コンポーネントは第 3 層のリソースにアクセスすることができます。第 3 層は、多くの異なる物理ホストで実行することができ、それ自体が追加の物理層を備えているアプリケーション論理を使用することができます。

3 層アプリケーションの開発は、Managed Object Framework によって実現されます。このフレームワークは、開発ツールの組によってサポートされます。この組により、ユーザーはフレームワークを利用して、継承およびフレームワークの実装の詳細を指定することなく、コンポーネントを作成することができます。

コンポーネントは中間層に配置し、第 1 層 (クライアント) を第 3 層 (データベースおよび他のリソース) に結び付けます。これらのコンポーネントは、CORBA ベースのビジネス・オブジェクトまたは Enterprise beans として実装されます。そして、アプリケーション論理を高性能なサーバーで実行できるようにし、各種リソース・マネージャーの混在による複雑さからクライアント・アプリケーションを隔離します。クライアントは CORBA に準拠した ORB を介してコンポーネントを操作し、コンポーネントは、ターゲット・リソース・マネージャー (たとえば、TCP/IP) によってサポートされる任意の通信プロトコルを使用して、リソース・マネージャーを操作します。 Component Broker

のコンポーネント・アーキテクチャーの概要については、 36ページの『Managed Object Framework』 を参照してください。

Component Broker のサーバー・プロセスは、完全な実行環境を備え、管理下のオブジェクトの部分的な実装を提供します。メソッド要求は ORB からサーバーに送られます。サーバーは、コンテナーおよびアダプターと一緒に働いて、メソッド要求のディスパッチを保証します。

詳細については、Component Broker の「*プログラミング・ガイド*」を参照してください。

## アプリケーション・アダプター

アプリケーション・アダプターは、オブジェクト指向のデータベースに似ています。データベース・システムがデータやレコードの場所を提供するのと同じように、コンポーネントの管理下のオブジェクトの場所を提供します。アプリケーション・アダプターは、管理下のオブジェクトにシステム機能 (たとえば、識別、キャッシング、パーシスタンスなど) を提供することを担当します。 Component Broker が提供するアプリケーション・アダプターは次のとおりです。

- 手続き型アプリケーション・アダプター (PAA)。このアプリケーション・アダプターにより、 Component Broker アプリケーションは、CICS、IMS、SAP などの手続き型リソースにアクセスすることができます。
- MQSeries アプリケーション・アダプター。Component Broker アプリケーションと MQSeries アプリケーションを統合します。
- データベース・アプリケーション・アダプター。 DB2、Oracle、Informix などのリレーショナル・データベースへのアクセスに使用します。

Component Broker で使用可能なアダプターの使用方法については、次の資料を参照しくください。

- *MQSeries アプリケーション・アダプター開発ガイド*
- *手続き型アプリケーション・アダプター開発ガイド*
- *データベース・アプリケーション・アダプター開発ガイド*

## オブジェクト・サービス

Component Broker アプリケーション・サーバーは、ビジネス・オブジェクトまたは Enterprise beans にさまざまなサービスを提供します。これらのオブジェクト・サービスの中には、本質的に管理的なものもあり、そうしたサービスの振る舞いは、管理ツールを用いて構成します。また、インターフェースとして

提供されるものや、インフラストラクチャーに組み込んで、ビジネス論理の代わりに機能するものもあります。これらのサービスの詳細については、Component Broker の「*上級プログラミング・ガイド*」を参照してください。

### 並行性制御サービス

並行性制御サービスは、トランザクション環境で使用することを目的としています。このサービスは、アプリケーションが複数のトランザクションまたはスレッドによる共用リソースへのアクセスを調整できるようにする、一連のインターフェースからなります。複数のトランザクションまたはスレッドが同時に 1 つのリソースにアクセスしようとした場合、競合するアクションが調整され、リソースの整合状態が維持されます。

### イベント・サービス

イベント・サービスにより、オブジェクトは互いに非同期で通信できるようになります。定義された役割は、サプライヤー・オブジェクトとコンシューマー・オブジェクトの 2 つです。サプライヤーは、イベントを作成し、コンシューマーはイベントを処理します。 Component Broker はイベント・サービスを使用してイベントのオカレンスを一意的に識別します。イベント・メッセージは、イベント・インスタンスが発生したということ以外、イベント・インスタンスに関する情報は伝えません。

### 通知サービス

通知サービスにより、オブジェクトは特定のイベントにそのオブジェクトの関与を登録 (または登録抹消) することができます。これには、サプライヤー・オブジェクトが、低レベルのオブジェクト・プロセスに影響を与えずに、コンシューマー・オブジェクトと非同期で通信することを可能にするイベント・チャネルが含まれます。

### 外部化サービス

外部化サービスにより、オブジェクトは非オブジェクト形式で状態を保存および復元することができます。これにより、オブジェクトの状態を、オブジェクト自体とは独立に存在させられるようになります。この状態は、元のオブジェクトまたはそれが入っていた ORB が存続しているかどうかにかかわらず、いつまでも保守することができます。

### 識別サービス

識別サービスは、オブジェクトごとの固有の識別情報を提供します。コンテナー、サーバー、ホスト、およびドメイン内のオブジェクトの位置を示す相対情報から、オブジェクトの識別を引き出します。

### ライフサイクル・サービス

ライフサイクル・サービスは、分散環境でのオブジェクトの作成、コピー、移動、および削除の操作を提供します。クライアントは、さまざまな場所でオブジェクトのライフサイクル操作を実行する必要があります。

### 命名サービス

Component Broker の命名サービスでは、命名の階層を作成して、オブジェクトを探しやすくすることができます。クライアントは、このサービスを他のサービスと併用することにより、さまざまな命名コンテキスト・ツリーをナビゲートして、特定のオブジェクトを見つけることができます。Component Broker の命名サービスは、絶対パスと相対パスの両方を処理します。

### セキュリティー・サービス

Component Broker はさまざまなセキュリティー・サービスをサポートします。分散システムのセキュリティーを確保するためのメカニズムおよびテクノロジーを提供します。ただし、ユーザー自身で行った場合ほどには、分散システムを保護することはできません。 Component Broker の実行時環境 (特に、オブジェクトの構成で想定されているアプリケーション・アダプター) は、そのオブジェクトの永続データに関してデータ・システムで行われた要求の認証性を確立する責任を負います。これらのメカニズムには、次のものが含まれます。

- 認証
- メッセージ保護
- 許可

セキュリティー・サービスは、主として、エンド・ユーザーが、使用を許可されていない情報およびリソースにアクセスできないようにするために使用されます。これは、主に分散ビジネス・オブジェクトを対象にしていますが、拡張することによって、それらのビジネス・オブジェクトによって使用される他の非オブジェクト指向または非分散ソースからの情報およびリソースも対象に含めることができます。

Component Broker は、多くの場合、ビジネス・アプリケーションや企業データなどの既存の情報システム・リソースを保護するために使用されます。それらのリソースは、中央化されて、物理的な機密保護機能のある環境に保管されるか、あるいは、アクセス・チャネルの制御を介してアクセス制限されているのが普通でした。

### トランザクション・サービス

トランザクション・サービスは、プログラマーがトランザクションを実装する場合に使用できる、標準の分散オブジェクト・インターフェースを提供します。 Component Broker は、トランザクション・サービスを使用して、矛盾することがないようにトランザクション・データを更新します。アプリケーションがトランザクション・サービスと並行性サービスを一緒に使用していれば、これらの更新は、他のタスクに対して実行される更新の影響を受けません。

トランザクションの詳細については、15ページの『トランザクション: 分散環境におけるデータ整合性とパフォーマンスの保証』を参照してください。

### セッション・サービス

セッション・サービスにより、アプリケーションはセッションの範囲を制御することができます。そのセッションの有効範囲内の関連するアプリケーション・プロファイルおよびセッション・プロパティーに関する情報を提供します。

セッションの有効範囲は、そのセッションが開始された時点から終了した時点までの間に存在するように定義します。各セッションは、トランザクション・コンテキストの外側で、オブジェクトの特定インスタンスを他のセッションから分離する機能を備えています。各セッションには、オブジェクト・データの独自のビューを割り当てることができます。

### 照会サービス

照会サービスは、オブジェクト指向構造化照会言語 (OOSQL) で記述された一連の条件に基づいて、 Component Broker コレクション内でオブジェクトを探します。 OOSQL は、SQL を拡張したもので、照会ステートメント内のオブジェクト・コレクション、オブジェクト属性、およびメソッドを処理する機能があります。複雑な検索基準をサポートします。

照会サービスは、オブジェクト参照子のリストまたはオブジェクト属性値のリストのどちらでも戻すことができます。ベースとなっているデータベースの検索機能と索引を使用して、オブジェクトを効率良く検索します。

### キャッシュ・サービス

キャッシュ・サービスは、データのオプティミスティック・キャッシュとペシミスティック・キャッシュをサポートして、並行性およびパフォーマンスを強化します。オプティミスティック・キャッシュでは、頻繁に使用されるデータは Component Broker サーバーのメモリーにキャッシュされ、トランザクションごとにデータベースから再読み取りされることがなくなります。キャッシュ

されたデータは、タイムアウト値に基づいて無効にされます。ペシミスティック・キャッシュは、アプリケーションで現行データが使用されることを保証する必要があり、またトランザクションの逐次性を保証するためにアプリケーションで高度な分離を行う必要がある場合に使用されます。キャッシュ・モードは、各オブジェクト・タイプでシステム管理エンド・ユーザー・インターフェースを使用して構成してください。

### 作業負荷管理

あるアプリケーションを使用するクライアントの数が多くなると、作業量およびサーバーの負荷が増大します。作業負荷管理機能は、Component Broker の実行時環境が、要求を処理するためのアプリケーション・サーバーを動的に割り振れるようにします。これにより、分散ネットワーク内のオブジェクト間のルーティングが削減されて、クライアント要求の応答時間が最小限に抑えられ、サーバーのディスパッチが最大化されます。 Component Broker における作業負荷管理のかぎは、サーバーのグループを使用して、共通構成で複数のアプリケーション・サーバーを定義することです。

## システム管理

Component Broker は、幅広いシステム管理機能を提供します。 UNIX プラットフォームおよび Windows NT プラットフォームでは、システム管理はグラフィカル・ユーザー・インターフェース (GUI) を用いて行います。この GUI を使用すると、ユーザーの企業を簡単にしかも効率よく管理および運営することができます。システム・マネージャーのユーザー・インターフェースは、ユーザーの企業のエンティティーをシステム管理オブジェクトとして表示し、操作します。 OS/390 プラットフォームでは、システム管理は、Administration Application および Operations Application を使用して行います。

74ページの図8 に示すように、システム・マネージャーは、Component Broker ネットワーク内のアプリケーション、サーバー、およびクライアントを制御します。サーバー、クライアント、アプリケーション、およびそのリソースはすべて、システム管理オブジェクトとして管理されます。それらのオブジェクトに関する情報は、システム・マネージャーによって保守される中央構成データに保管されます。システム・マネージャーは、そのネットワーク内のそれぞれの管理下のホストにあるシステム管理オブジェクトと、当該管理下のホストで実行されるシステム・マネージャー・エージェントを介して対話します。システム・マネージャーは、システム・マネージャー・ユーザー・インターフェースを介して、そのネットワーク内のオブジェクトに関するデータを管理者に提供します。

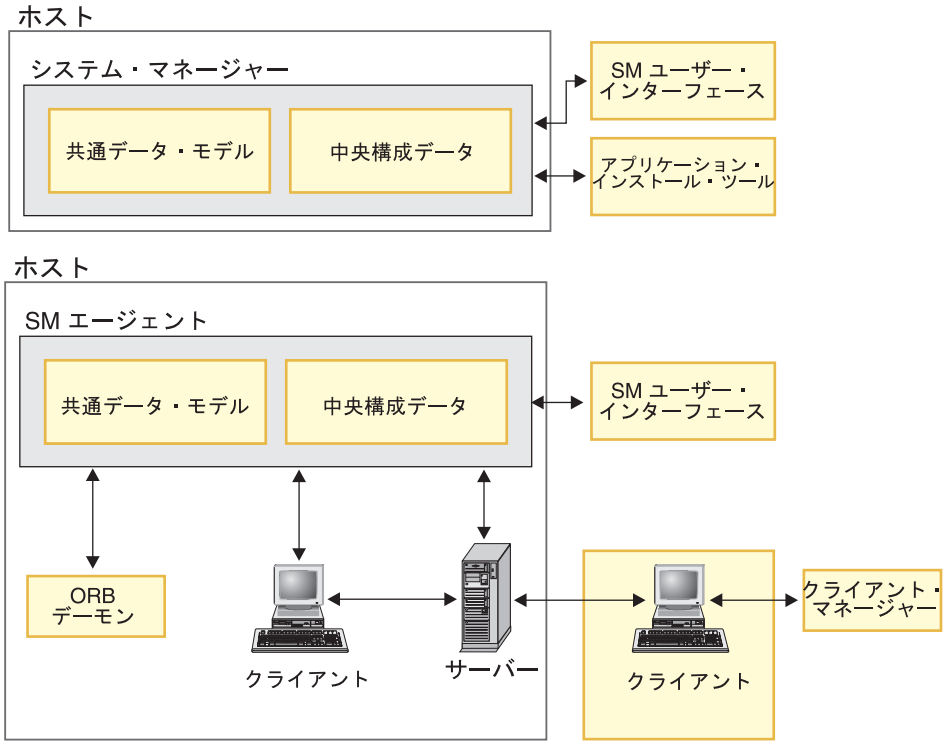


*図 8. Component Broker システム・マネージャー*

システムを効率よく管理するためには、システムおよびリソースを、ユーザーのビジネス・ニーズに合わせて編成してください。システム・マネージャーを使用することにより、たとえば、次のことが行えます。

- ホスト・コンピューターをセルおよび作業グループにグループ化する。
- ユーザーの企業を、別個の単位として管理される 1 つまたは複数の管理ゾーンとして構成する。ユーザーの企業内のホストは、1 つの管理ゾーン、すなわちネットワーク・ゾーンに自動的に構成されます。ビジネス・アプリケーションおよびそのクライアントとサーバーは、 1 つ以上の別のアプリケーション管理ゾーンに構成することができます。
- 各管理ゾーンの代替構成を作成して、ビジネス・ニーズの変化に対応できるようにする。たとえば、別のアプリケーションまたは地域をサポートし、別のサーバーを使用し、さらには別のホスト・コンピューターを使用する構成を用意することができます。
- 制御されたサーバー・グループ全体にわたるアプリケーションの作業負荷管理を構成する。

- 構成が完全で、適切に定義され、活動開始の準備が整っていることを検証する。
- ある管理ゾーンの構成全体を 1 つのアクションで活動開始し、事業活動の全部または一部を更新する。

上記のすべてを、システム管理オブジェクトの最小定義によって行うことができます。構成をアクティブにできるかどうかを検証する場合、システム・マネージャーは、どのシステム管理オブジェクトおよび関連が必要であるのかを判別し、可能な場合には、それらのオブジェクトおよび関連を作成します。

また、Component Broker システム管理を使用して、社内のシステムを操作したり、ランタイム・オブジェクトの機能を実行したりすることもできます。たとえば、以下のことが行えます。
- 個々のサーバーおよびアプリケーションを開始および停止する。
- 管理下のオブジェクトの状況を表示および操作する。
- 管理下のオブジェクトの属性および関連を表示および変更する。
- サーバー・グループ全体に渡って作業負荷管理を変更する。

## 開発ツール

CB ツールと VisualAge for Java および VisualAge C++ を一緒に使用して、多層アプリケーションを生成することができます。

### オブジェクト・ビルダー

オブジェクト・ビルダーは Component Broker 向けの開発環境です。オブジェクト・ビルダーは、以下の用途に使用することができます。
- 新規アプリケーションの開発
- 既存のアプリケーションの累積
- 既存のアプリケーションへの新規機能の追加
- アプリケーションのパッケージ
- Component Broker 実行時で実行する Enterprise beans の作成

オブジェクト・ビルダーは、アプリケーション開発に、最初から最後まで使用されます。また、Rational Rose で設計してから、その設計を Object Builder にインポートして、最終的なオブジェクトおよびプログラム論理を追加することもできます。オブジェクト・ビルダーは、Java と C++ の両方で実装された IDL を使用して、CORBA プログラミング・モデルをサポートします。単体テスト・バージョンおよびサーバー・セットアップ・スクリプトが完備したクライアント / サーバーのフル・パッケージを含む、完全に作動するアプリケーションを生成することができます。

CB ツールは、オブジェクト・ビルダーと統合されて、コードおよびエミッターをコンパイルするための入力になります。この統合により、オブジェクト・ビルダーが Component Broker 製品用の開発環境として定義されます。これを使用して、アプリケーションを最初から最後まで開発することも、アーキテクチャー・エンジニアリング設計をオブジェクト・ビルダーにインポートして、そこで最終的なオブジェクトおよびプログラム論理を追加することもできます。

アプリケーションを生成するためのプラットフォームは、選択することができます。選択したそれぞれのプラットフォームごとに、アプリケーション・プロジェクトの作業ディレクトリーに同等なサブディレクトリーが追加されます。コードを生成するたびに、選択したすべてのプラットフォーム用のコードを生成します。ただし、再作成する必要があるコードは、最後のコード生成以降にモデルに変更が加えられたために生成する必要が生じたものだけです。

開発したコンポーネントをユーザーのターゲット・プラットフォームに配置できるようにするために、制約を定義することができます。これらの制約は、オブジェクトの作成時に設定することも、後でプロパティーを編集して設定することもできます。たとえば、プラットフォーム制約が AIX および OS/390 に設定されている場合には、そのオブジェクトの OS/390 専用バージョンを開発するために、OS/390 制約だけを適用することを選択できます。

オブジェクト・ビルダーのユーザー・インターフェースを使用すると、アプリケーションのさまざまなビューにアクセスできるようになります。オブジェクト・ビルダーで定義するアプリケーション DLL ファイルを作成するためには、 Component Broker Server ソフトウェア開発キット (SDK)、および前提条件のアプリケーション開発ソフトウェアをインストールしておく必要があります。

アプリケーションのモデルは、コンポーネントから構成されます。オブジェクト・ビルダーを使用して、Windows NT、AIX、Solaris、または OS/390 サーバーに配置するコンポーネントを開発することができます。ほとんどの開発オプションは、すべてのプラットフォームで同じになっています。主要な相違は、コンポーネントのためのコードを生成するときに現れます。たとえば、選択したプラットフォーム用の継承オプション、フレームワーク・メソッド、およびフレームワーク・メソッド実装をフィルターに掛けることができます。すべてのプラットフォーム・ビューに関する情報は、同じプロジェクト・モデルに保管されます。ビューの切り換えは、いつでもできます。

Component Broker は、移送可能な Enterprise beans を、Component Broker ビジネス・オブジェクトとして実行するように配置するためのツールも備えてい

ます。セッション bean とエンティティー bean の両方の配置がサポートされます。オブジェクト・ビルダーまたは VisualAge for Java から Enterprise bean を配置するためのサポートを含む、bean 配置のためのコマンド行インターフェースと GUI インターフェースが提供されます。bean 管理下のパーシスタンスを備えたセッション bean およびエンティティー bean の配置は、バッチ・モードで、あるいは MAKE ファイルから、不在で行うことができます。 CMP を備えたエンティティー bean を配置するためには、オブジェクト・ビルダーを使用して、その bean の CMP フィールドと永続データ・ストアとの間のマッピングを定義する必要があります。このマッピングは、レガシー・データ・ストアを使用して行うことも、新規データベースを定義して行うこともでき、また、Component Broker によってサポートされる各種の持続バックエンドを利用することもできます。このマッピングは、 Component Broker によって提供されるデータベース・マッピング・ヘルパーの豊富なセットを利用して行うこともできます。

オブジェクト・ビルダーの詳細については、Component Broker の「*アプリケーション開発ツール・ガイド*」を参照してください。

# 第8章 TXSeries の紹介

IBM TXSeries は、サーバーを調整および統合し、ネットワーク全体にわたって高性能なアプリケーションとデータ・ソースを管理する、拡張トランザクション処理ソリューションです。IBM の業界をリードする顧客情報管理システム (CICS) のテクノロジーと、IBM の Encina トランザクション処理製品のテクノロジーを結合したものです。これを使用することにより、オンライン・トランザクション処理に必要な信頼性、可用性、およびデータ保全性を備えた分散クライアント / サーバー環境を作成することができます。

TXSeries はインターネットへの標準プロトコルおよびゲートウェイをサポートします。ユーザーのトランザクション環境を、グループウェアおよびデータベース管理のためのキー・アプリケーションにリンクすることができます。TXSeries を使用すると、インターネットを介して安全性および信頼性を確保しながら商取引を行うこともできます。例えば、注文入力、顧客情報の更新、在庫メンテナンス・アプリケーションを実行することができます。

TXSeries は、DB2、Oracle、Microsoft SQL Server、Informix、および Sybase リレーショナル・データベースに保管されたデータへの、トランザクション・ベースのアクセスを提供します。

## TXSeries CICS

CICS は IBM の汎用オンライン・トランザクション処理ソフトウェアです。これは、デスクトップから最大規模のメインフレームまで、広範囲のオペレーティング・システムで実行されるアプリケーション・サーバーです。 TXSeries CICS は、WebSphere エンタープライズ版の一部であり、AIX、HP-UX、Solaris、および Windows NT で実行されます。そのほかのバージョンの CICS は、OS/390、AS/400、VMS などの他のプラットフォームで実行されます。

CICS はセキュリティー、データ保全性、およびリソース・スケジューリングを取り扱います。また、オンライン・トランザクション処理アプリケーションのために必要な基本ビジネス・ソフトウェア・サービスを内蔵しています。CICS を使用する代表的なトランザクション処理アプリケーションは、以下のとおりです。

- 小売店用分散システム
- 銀行、保険、証券会社用システム

- 受注および注文処理システム
- 一般元帳システム
- 給与計算システム
- 自動預金支払機 (ATM)
- 航空座席予約システム
- プロセス制御システム

このセクションでは、CICS の高レベルな概要を示し、基本 CICS コンポーネントについて説明します。詳細については、TXSeries の資料「*概説および計画ガイド*」を参照してください。

## 基本 CICS 概念

CICS システムのインスタンスは CICS *領域* と呼ばれます。領域は、1 つの単位として構成および管理され、リソースの共通セットを制御します。同じシステム上で複数の CICS 領域を実行することもよくあります。これらの領域は相互に独立させることも (たとえば、会計用、在庫管理用などに分けて使用します)、相互に密接に結び付けることもできます。CICS は、領域間 (システム間とも呼ばれます) 通信のためにいくつかの機能を提供します。

ユーザーと CICS 領域との対話では、 1 つまたは複数のトランザクションが発生します。 CICS 用語では、*トランザクション* は、ユーザーに提供される基本オペレーションです。たとえば、銀行用アプリケーションには、照会トランザクション、引き出しトランザクション、振り込みトランザクションなどが含まれます。この用語の使用法は、他のコンテキストにおける (たとえば、Encina での) 使用法とは若干異なります。CICS では、アトミック作業単位として実行する必要があり、また持続可能で回復可能なアクションのグループを、*作業論理単位* (*LUW*) と呼びます。

アプリケーションを構成するトランザクションは、CICS アプリケーション開発者によって作成されます。管理者は、どのようなトランザクションがユーザーに提供されるのかを、領域のトランザクション定義で指定します。CICS では、いくつかの事前定義されたトランザクションが用意されています。これらのトランザクションでは、ユーザーのサインオンとサインオフが許されます。また、ユーティリティー、管理、およびデバッグ機能も備わっています。

CICS は、ファイル・マネージャー (Encina 構造化ファイル・サーバー (SFS) または DB2) を使用して、一時データと仮想記憶アクセス方式 (VSAM) ファイルを保管します。

## CICS アプリケーション・プログラミング・インターフェース

CICS は、開発者がトランザクション・アプリケーション・プログラムを作成できるように、強力なアプリケーション・プログラミング・インターフェース (API) を提供します。この API は、いくつかの CICS コマンドからなります。これらのコマンドは、高水準言語 (COBOL、C、C++、PL/I、または Java) で書かれたアプリケーション・プログラムに組み込まれます。開発者は、次の例で示すように、CICS コマンドの前に EXEC CICS という語句を指定するだけで済みます。

```
EXEC CICS READ FILE('ORDER') INTO(RECORD)
```

これにより、プログラム・ソース・ファイルは、プログラム言語用のコンパイラー (COBOL コンパイラー、C コンパイラー、その他) によって処理される前に、プリコンパイラーによって処理されるようになります。

CICS API コマンドは、以下のタイプの機能を実行するために使用できます。

- ファイルの読み取り、書き込み、および更新
- メモリーの割り振りおよび解放
- CICS プログラム間での制御の受け渡し
- 一時記憶域キューとの間での読み取りおよび書き込み
- 一時データ・キューとの間での読み取りおよび書き込み
- タイマーの使用
- 監査証跡、変更記録などのジャーナルの作成
- 3270 端末からのデータの送受信
- SNA LU 6.2 通信を使用する通信
- CICS 内部 ディスパッチャーの制御
- 作業論理単位の取り扱い
- セキュリティーおよび認証
- バッチ・データ交換
- モニターおよび診断

その他の機能を実行するコマンドも用意されています。プラットフォームによっては、CICS API コマンドは C++ および Java メソッドとしても実装されます。

CICS は、*ユーザー出口* も提供します。これは、CICS がユーザー作成のプログラム (ユーザー出口プログラム) に制御権を移動することができる、 CICS モジュール内の場所です。ユーザーの出口プログラムが作業を終了すると、

CICS が制御を再開します。ユーザー出口を使用することにより、ユーザー独自の要件に従って CICS システムの機能を拡張し、カスタマイズすることができます。

### リレーショナル・データベース・サポート

CICS は、いくつかのリレーショナル・データベースの使用をサポートします。これらのデータベースを使用すると、CICS アプリケーションによって使用される情報を保管することができます。この情報には、組み込まれた構造化照会言語 (SQL) ステートメントも含めることができます。このデータベースは、必要に応じて完全な 2 フェーズ・コミット・プロセスを使用して、 CICS LUW に完全に参加することができます。 CICS は、以下のリレーショナル・データベース管理システムをサポートします。

- DB2 UDB
- Oracle
- Informix
- Sybase
- Microsoft SQL Server

CICS はまた、VSAM の使用もサポートします。

### キュー・サービス

キューは、単一 CICS 領域または相互接続された CICS 領域のシステム内にあるグローバルなリソースとしての、順次記憶機能です。つまり、ファイルやデータベースと同じように、キューは特定のタスクとは関連付けられません。どのタスクもキューの読み取り、書き込み、または削除を行うことができ、また、キューに関連付けられたポインターはすべてのタスクで共用されます。

CICS では、一時データ・キューと一時記憶域キューの、2 つのタイプのキューが提供されます。これらの名称からは一時的なキューであることが連想されますが、CICS キューは永続記憶域です。主記憶装置に保持される一時記憶域キューを除き、CICS キューは、コールド・スタート時に明示的に廃棄されないかぎり、CICS の実行が終了しても存続します。持続キューは、CICS ファイル・マネージャー (Encina 構造化ファイル・サーバー (SFS) または DB2) によって保管されます。

### システム間通信

複数システム環境では、CICS 領域は、他の領域と通信して、ローカル領域のユーザーにリモート・システムのサービスを提供したり、リモート・システム

のユーザーにローカル領域のサービスを提供したりできます。データとアプリケーションの両方を共用することができます。

CICS 相互通信機能は、分散システムのオペレーションを単純化します。一般に、このサポートは標準の CICS 機能 (ファイルおよびキューの読み取りや書き込みなど) を拡張して、アプリケーションまたはユーザーが、リソースの位置を知らなくてもリモート・システムにあるリソースを使用することができるようにします。以下の CICS 相互通信機能が使用可能です。

- *分散プログラム・リンク (DPL)* は EXEC CICS LINK コマンドの用途を拡張し、CICS アプリケーション・プログラムが、別の CICS システムにあるプログラムにリンクできるようにします。
- *機能シップ* により、アプリケーション・プログラムは、別の CICS システムに属するファイル、一時データ・キュー、および一時記憶域キューにアクセスすることができます。
- *トランザクション・ルーティング* により、トランザクションをリモート・システム上で実行することができます。このトランザクションは、ユーザーのローカル・システムで実行されているかのように、ユーザーの端末で情報を表示することもできます。
- *非同期処理* は、EXEC CICS START コマンドを拡張して、アプリケーションが別の CICS システムで実行するトランザクションを開始できるようにするものです。標準の EXEC CICS START 呼び出しの場合と同様に、START コマンドで要求されたトランザクションは、その START コマンドを発行したアプリケーションとは独立して実行されます。
- *分散トランザクション処理 (DTP)* は、追加の EXEC CICS コマンドを使用して、異なるシステムで実行されている 2 つのアプリケーションが相互に情報を受け渡せるようにします。これらの EXEC CICS コマンドは、SNA アーキテクチャーで定義されている LU 6.2 マップ式会話 Verb にマップされます。

  DTP を使用して、*拡張プログラム間通信機能 (APPC)* プロトコルを使用する CICS 以外のアプリケーションと通信することができます。

## CICS SNA サポート

CICS 領域と Encina PPC ベースのアプリケーションは、SNA を介して、APPC をサポートする任意のシステム (たとえば、IBM メインフレーム・ベースの CICS および APPC ワークステーション) と通信することができます。

TXSeries は、次の 2 つの方式による SNA 通信を使用することができます。

- CICS ローカル SNA サポート。これは同期レベル 0 および 1 をサポートします。
- Encina PPC Gateway サーバー。これは、同期レベル 0、1、および 2 をサポートします。

いずれの方式も、他の CICS 領域との間のすべての CICS 相互通信機能をサポートし、また非 CICS 領域 (たとえば、Encina) への DTP をサポートします。さらに、CICS は、ローカル SNA サポートを使用して IBM CICS ユニバーサル・クライアントと通信することもできます。

## ユーザーとの通信

ユーザーは、クライアントを介して CICS 領域と通信します。クライアントは一般的に、サーバーとの通信、およびユーザーとそのアプリケーション・プログラムへのインターフェースの提供を行うための専用製品です。クライアントは、ある範囲のプラットフォーム (たとえばラップトップ・コンピューターからオープン・システム・ワークステーションまで) で実行されます。 1 つの CICS クライアントは複数の CICS 領域と通信することができます。

## CICS Transaction Gateway

CICS Transaction Gateway は、CICS Internet Gateway の機能と、 CICS Gateway for Java の機能を統合したものです。このゲートウェイにより、Web ブラウザー、ネットワーク・コンピューター、またはインターネット対応のコンシューマー向け装置は、 CICS サーバーで実行されているビジネス・アプリケーションにアクセスすることができます。 CICS Transaction Gateway を使用した場合は、 Web ブラウザーを使用して CICS 3270 端末セッション、Java アプレット、または CORBA ベースのクライアントを実行することができます。

CICS Transaction Gateway は、Windows NT、AIX、Solaris、および OS/2 オペレーティング・システムで稼働します。さらに、Windows 95 および 98 オペレーティング・システム用のアプリケーション開発がサポートされます。

## CICS 管理

CICS のシステム管理は、CICS 環境を構成して CICS 領域を開始可能にすること、実行中の領域をモニターすること、領域をシャットダウンすること、および問題から回復することからなります。CICS の管理は、 SFS、DB2、分散コンピューティング環境 (DCE) などの他のコンポーネントに影響を与える手順を伴います。

CICS の構成および管理に使用される管理ツールは、使用されるオペレーティング・システムによって異なります。たとえば、Windows NT では TXSeries 管理ツールを使用することができ、AIX では CICS 用のシステム管理インターフェース・ツール (SMIT) を使用することができます。このツールにより、管理手順は単純化され、自動化されます。 CICS コマンドやトランザクションなどの、他のツールを使用することもできます。

CICS 管理ツールは、1 台のマシンの CICS 環境 を管理するために設計されています。これらのツールを使用するためには、そのマシンにシステム管理者としてログインしてから、使用したいツールを起動してください。複数のマシンで CICS 環境を管理したい場合には、標準的な技法によってリモートで各マシンにログインして、それらのマシンでツールを使用することができます。たとえば、あるマシンを単一の制御点として使用し、別のマシンでツールを実行するためにセッションをセットアップすることができます。管理ツールへのアクセスは、このマシンへのアクセスを制御することによって制御できます。

## TXSeries Encina

Encina は、オープン分散システムの開発と管理を支援するために設計されています。 Encina は、大規模な分散クライアント / サーバー・システムを構築および実行するためのソフトウェア・プロダクトのファミリーであり、DCE および CORBA によって提供される機能を使用し、拡張します。Encina は、複数のリソース・マネージャーを使用して、複数のプラットフォームでの分散トランザクションをサポートします。また、CICS などの他のトランザクション処理製品と対話し、複数のプラットフォームおよびリソース・マネージャーにまたがるトランザクションを調整します。

Encina は AIX、Sun Solaris、HP-UX、および Windows NT で実行されます。Encina DE-Light Gateway などの機能、および CORBA や Microsoft COM などの標準に関する Encina によるサポートを使用することにより、他のいくつかのプラットフォーム、および Web ブラウザーでもクライアントを実行することができます。

86ページの図9 に、Encina アーキテクチャーの概要を示します。次に、各コンポーネントについて説明します。

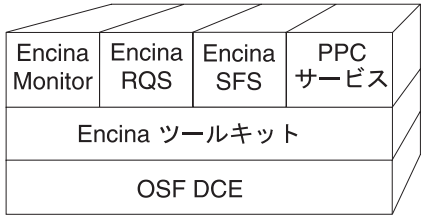


*図 9. Encina のアーキテクチャー*

## Encina モニター

Encina モニターは、トランザクション処理アプリケーションを開発、実行、および管理するためのインフラストラクチャーを提供します。Encina モニターは、リソース・マネージャーと協調して、大量のデータを整合性のある状態で保守するための環境を提供し、定義されたサーバーを介して特定の方法で特定のデータにアクセスできるユーザーおよびクライアントを制御します。このモニターは、IBM メインフレームなどの既存のコンピューティング・リソースと相互作用する、スケーラブルな、オープンのモジュラー・システムを提供します。モニターには、以下のものが組み込まれています。

- プログラマーを分散コンピューティングの複雑さから隔離する、全機能の API。
- 異種環境でのロード・バランシング、スケジューリング、および耐障害性を提供して、ハイパフォーマンスとトランザクション上の保全性を確保できるようにする、信頼性の高い実行環境。
- 広範囲に分散したモニター・ベースのシステムを、単一の論理的に定義されたシステムとして管理できるようにする、包括的な管理環境。

モニターおよびそのプログラミング・インターフェースについては、「*Encina モニター・プログラミング・ガイド*」で詳しく説明されています。Encina モニターへの管理インターフェースの詳細については、「*Encina 管理ガイド 第 1 巻: 基本管理*」を参照してください。

## 回復可能キューイング・サービス (RQS)

回復可能キューイング・サービス (RQS) を使用すると、アプリケーションは、後で処理するためにトランザクション作業をキューに入れることができます。これにより、このアプリケーションは、キューに入れられた作業が後でトランザクションによって完了できるように確保して、トランザクションをコミットすることができます。

RQS は、タスクの一部または全部を後で処理するためにオフロードする必要があるトランザクション・アプリケーションをサポートします。アプリケーションは、タスクに関連するデータをキューに保管することができます。その後、このデータを別のプログラムで処理することができます。あるリソースの使用が、ピーク使用時には所要時間の関係で受け入れがたい場合、トランザクションの一部が他の部分よりも長い時間を要する場合、またはあるリソースが一時的に使用不能になっている場合には、このようなオフロードを利用すると便利です。たとえば、販売の確認をリアルタイムで完了させ、それに関連したデータは RQS キューに保管しておいて、あとで処理することができます。

RQS プログラミングについては、「*Encina RQS プログラミング・ガイド*」を参照してください。 RQS 管理については、「*Encina 管理ガイド 第 2 巻: サーバー管理*」を参照してください。

### 構造化ファイル・サーバー (SFS)

SFS は、トランザクションの保全性、ログに基づく回復、および広範なスケーラビリティーを提供する、レコード単位のファイル・システムです。多くのオペレーティング・システムは、データへのバイト・ストリーム・アクセスのみをサポートします。この場合、すべての入出力データは、その送信元とは無関係に、不定様式のバイトのストリームとして扱われます。SFS は、レコードから構成される*構造化ファイル* を使用します。これらのレコード自体は、フィールドからなります。たとえば、レコードごとに 1 人の従業員に関する情報を収め、その従業員の名前、番号、および給与を表すフィールドを設けることができます。

SFS ファイル内のすべてのデータは、SFS サーバーによって管理されます。このデータにアクセスする必要のあるプログラムは、そのサーバーに要求を発信しなければなりません。サーバーは要求されたデータを検索したり、指定されたオペレーションを実行したりします。

SFS プログラミングについては、「*Encina SFS プログラミング・ガイド*」を参照してください。 SFS 管理については、「*Encina 管理ガイド 第 2 巻: サーバー管理*」を参照してください。

### 対等通信 (PPC) サービス

Encina 対等通信サービス (PPC サービス) は、 Encina トランザクション処理システムが、システム・ネットワーク体系 (SNA) LU 6.2 通信インターフェースを備えたシステムと相互操作できるようにします。これにより、メインフレーム環境と Encina 環境の統合が実現されます。

PPC サービスを使用すると両方向通信が可能になり、メインフレームと Encina がアプリケーションとデータの両方を共用し、どちら側からも通信を開始できるようになります。通信は、DCE セルと SNA ネットワークの両方に含まれているゲートウェイ・サーバーを介して経路指定され、 Encina とメインフレーム・アプリケーションの間に仮想リンクが確立されます。

PPC サービスがサポートするインターフェースは次のとおりです。

- 分散プログラム・リンク (DPL)
- X/Open 共通プログラミング・インターフェース・コミュニケーション (CPI-C) と IBM システム・アプリケーション体系 (SAA) CPI-C の両方
- SAA 共通プログラミング・インターフェース・リソース回復 (CPI-RR)

これらの通信インターフェースとこの分散トランザクション処理モデルは、Encina 環境内で作動します。

PPC プログラミングについては、「*Encina PPC サービス・プログラミング・ガイド*」を参照してください。 PPC 管理については、「*Encina 管理ガイド 第 2 巻: サーバー管理*」を参照してください。

## DE-Light ゲートウェイ

DCE Encina Lightweight Client™ (DE-Light) は、API のセットとゲートウェイ・サーバーから構成されるもので、これを使用して、DCE および Encina の能力を、 DCE クライアントとして実行されていないパーソナル・コンピューターなどのシステムにも拡張することができます。

DE-Light を使用すると、DCE はサポートしないが Java はサポートするシステムから DCE および Encina にアクセスすることができます。 DE-Light クライアントは、作成作業がほとんど必要なく、管理者にとっては扱いやすく、また DCE クライアントや Encina クライアントで必要なネットワーク・リソースも使用しません。しかも、これらの DE-Light クライアントは、従来は完全な DCE および Encina クライアントでしか使用できなかった機能を利用することができます。

DE-Light は、次のコンポーネントから成り立ちます。

- スタンドアロン Java アプリケーション用の Java クライアントの開発に使用する Java API。 DE-Light Java クライアントは、TCP/IP および HTTP を介してゲートウェイと通信します。
- Microsoft Windows NT および Windows 98 環境用のクライアントの開発に使用する C API。 DE-Light C クライアントは、TCP/IP を使用して、既知のエンドポイントにあるゲートウェイと通信します。

- DE-Light クライアントと DCE または Encina との間の通信を可能にする DE-Light Gateway サーバー。

DE-Light Java は、次の 2 つのタイプのクライアントをサポートします。

- Web サーバー上に常駐しているハイパーテキスト・マークアップ言語 (HTML) 文書に組み込まれている Java アプレット
- スタンドアロン Java アプリケーション

DE-Light プログラミングについては、「*Encina DE-Light プログラミング・ガイド*」を参照してください。 DE-Light Gateway 管理については、「*Encina 管理ガイド 第 2 巻: サーバー管理*」を参照してください。

## Encina ツールキット

Encina ツールキットは、大規模な分散クライアント / サーバー・システム開発に必要な機能を提供する、モジュール、ライブラリー、およびプログラムの集合です。ツールキットのモジュールには、ログおよび回復サービス、トランザクション・サービス、およびトランザクション・リモート手続き呼び出し (TRPC、これは DCE RPC テクノロジーの拡張版です) が含まれます。これらのモジュールは、分散トランザクションの保全性を透過的に保証します。このツールキットは、C プログラム言語をトランザクション向けに拡張した Transactional-C (Tran-C) も提供します。

ツールキットの詳細については、「*Encina ツールキット・プログラミング・ガイド*」を参照してください。

## Encina++

Encina++ はオブジェクト指向の Encina 用 API です。これは、多数の Encina コンポーネントにアクセスする複数のクラスからなります。 Encina++ は、DCE (Encina++/DCE)、CORBA (Encina++/CORBA)、および DCE と CORBA の混合 (Encina++/CORBA-DCE) に基づく、オブジェクト指向アプリケーションの開発をサポートします。最後のタイプのアプリケーションは、多くの場合、混合アプリケーションと呼ばれます。

90ページの表3 に、Encina++ を構成するインターフェースをリストします。Encina++/DCE でしか使用できないものや、Encina++/CORBA でしか使用できないもの、またその両方で使用できるものがあります。

表 3. *Encina++* のコンポーネント

| インターフェース | 目的 | 言語 | 使用 |
|---|---|---|---|
| Encina++ | クライアント / サーバー・アプリケーションを作成および管理する。 | C++ | DCE および CORBA |
| Transactional-C++ (Tran-C++) | 分散トランザクション処理。 | C++ | |
| オブジェクト管理グループ・オブジェクト・トランザクション・サービス (OMG OTS) | 分散トランザクション処理; OMG オブジェクト・トランザクション・サービス 仕様 (*OMG 文書 94.8.4*) を実装する。 | C++ | |
| RQS++ | RQS サーバーでトランザクションを用いてデータをエンキューおよびデキューする。 | C++ | DCE のみ |
| SFS++ | SFS サーバーに格納されているデータを操作する。 | C++ | |
| オブジェクト並行性制御サービス (OCCS) | 複数のクライアントが、共用リソースへのアクセスを調整することができる。 OMG *並行性制御サービス案* を実装する。 | C++ | CORBA のみ |
| Java OTS クライアント | Java クライアントが分散トランザクションを使用することができる。 OMG オブジェクト・トランザクション・サービス 仕様を実装する。 | Java | |

各種 Encina++ インターフェースの詳細については、次の資料を参照してください。

- *Encina オブジェクト指向プログラミング・ガイド*
- *Encina Transactional プログラミング・ガイド*
- *Encina RQS++ および SFS++ プログラミング・ガイド*

## Windows プラットフォームで使用可能な Encina ツール

Windows NT システムと Windows 95/98 システムの両方で、Encina とともに、他のプラットフォームでは利用不能な追加のプログラミングおよび診断ツールが提供されます。これらのツールを使用して Encina アプリケーションを作成する方法については、「*Windows 環境での Encina アプリケーションの作成*」を参照してください。

### プログラミング・ツール

以下の Encina ツールは、分散クライアント / サーバー・アプリケーションを作成する際の開発者の作業を支援します。

- Encina サーバー・ウィザード — このウィザードは Encina および Encina++ サーバーを作成するために使用できるもので、サーバー用の標準初期化コードの多くを生成し、そのコードをプロジェクトに編成し、サーバーを構築するために必要な、適切な Encina およびシステム・ライブラリーを関連付けます。
- Encina COM ウィザード — このウィザードは、Encina TIDL インターフェースから Encina COM コンポーネントを (DLL ファイルの形で) 作成するために使用されます。 DLL ファイルを任意の言語で作成されたクライアントに組み込むことにより、そのクライアントは、DLL ファイルで定義されたインターフェースをエクスポートする任意の Encina サーバーにアクセスできるようになります。

### WinTrace

WinTrace ツールは、分散クライアント / サーバー・アプリケーションをデバッグする際の開発者の作業を支援します。この Encina 特定ツールは、アプリケーション出力ファイルと Encina トレース・ファイルを形式設定および表示したり、エラー・コードおよびトレース ID を変換したりするために使用します。また、プロセスの実行中に出力を表示するための Encina トレース listener サーバーを開始するためにも使用することができます。このツールの使用法については、ツールのオンライン・ヘルプを参照してください。

## WebSphere Application Server アドバンスド版とのインターオペラビリティー

Encina アプリケーションは、アドバンスド版のアプリケーションと相互運用することができます。 Java ベースのアドバンスド版アプリケーションは、Encina サーバーのクライアントとして動作できます。 Encina サーバーに接続するには、 Java ベースのシステムと Encina/DCE システムまたは Encina++/DCE システム (あるいはその両方) との間のブリッジの役割をする特別なモニター・アプリケーション・サーバーを使用します。このブリッジ・サーバーが、クライアントで発生したトランザクションをサーバーに伝えます。

92ページの図10 に、ブリッジ・サーバーを使用する分散システムの一般的なアーキテクチャーを示します。

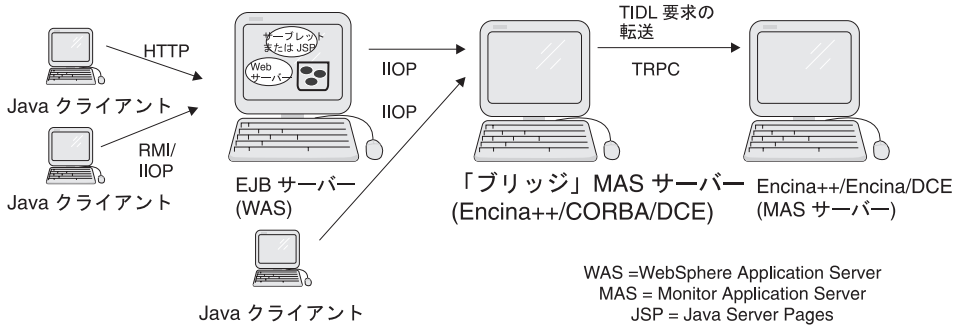


*図 10. Java アプリケーションと Encina/Encina++ サーバー間のインターオペラビリティー*

Java ベースのクライアントは、最小限の開発を加えるだけで Encina/Encina++/DCE アプリケーションにアクセスすることができます。Encina は、TIDL ファイルを入力として受け取るツール (**wstidl** コマンド) を提供し、必要な CORBA IDL インターフェース、Java クラス、サーバー・メイン機能、および接続を確立するために使用するその他のファイルを生成します。生成された Java bean または Enterprise bean は、 Application Server アドバンスド版アプリケーションの一部として使用できます。たとえば、WebSphere EJB サーバー内の Enterprise bean は、生成された bean (または生成された Enterprise bean) を使用して、Encina アプリケーションと通信します。

Encina ブリッジ・サーバーを使用するアプリケーションの例については、「*WebSphere ビジネス構築のソリューション*」を参照してください。

# 第9章 トポロジーおよび構成の例

ここでは、WebSphere Application Server の一般的なトポロジー例および構成例について説明します。目的は、WebSphere Application Server スタンダード版、アドバンスド版、およびエンタープライズ版を使用して設定できる構成の例を紹介することであって、構成をくまなく説明することではありません。

含まれている例は次のとおりです。

- 『クライアントのトポロジー』
- 96ページの『サーバーのトポロジー』
- 99ページの『Application Server スタンダード版のトポロジー』
- 99ページの『Application Server アドバンスド版のトポロジー』
- 101ページの『TXSeries 構成』
- 105ページの『Component Broker 構成』

WebSphere Application Server のその他のトポロジーの詳細については、「*WebSphere ビジネス構築のソリューション*」を参照してください。

## クライアントのトポロジー

従来からのクライアント / サーバー・モデルでは、物理的な 1 つ目の層の上に論理的な 2 つ目の層があります。つまり、クライアントは、アプリケーションのビジネス論理の一部またはすべてを実装します。このアーキテクチャーでは、他の分散アーキテクチャーでクライアントが必要とするものよりも大規模で、数も多い、強力なデスクトップ・マシンが必要となります (このタイプのクライアントのことをファット・クライアントとも言います)。高価な装置が必要になるだけでなく、ビジネス論理をユーザーのデスクトップ上に直接置いて維持することは、管理面でも難しく、費用がかかる可能性があります。

ビジネス情報システムの設計担当者の多くは、重要なビジネス論理をもっと中央で維持するコンピューティング・モデルを実装しています。このアプローチの方が、管理しやすく、情報システムの更新も簡単です。 WebSphere Application Server を使用すれば、ビジネス・プロセスを中央で処理する、さまざまなタイプのクライアントを実装することができます。

## シック・クライアント

シック・クライアントは、ユーザー・インターフェースとしてのローカル・アプリケーションに依存します。ビジネス論理は、分散コンピューティング環境内のリモート・サーバーで処理します。たとえば、シック・クライアントは、ワード・プロセッサーやスプレッドシートなどのデスクトップ・アプリケーションを使用して、ユーザー・インターフェースを提供することができます。

図11 に、典型的なシック・クライアントを示します。この図では、アーキテクチャーのユーザー・インターフェース部分とビジネス論理部分しか示されていません。

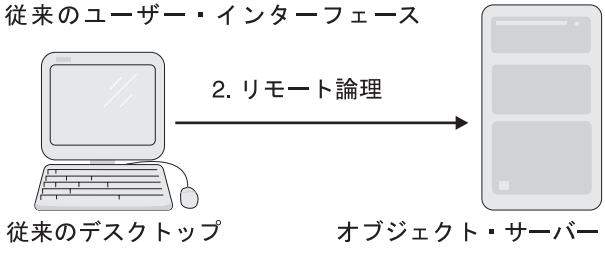


*図 11. シック・クライアントには、ローカル・ユーザー・インターフェースおよびリモート・ビジネス論理が割り当てられる*

シック・クライアントを使用したビジネス・ソリューションを実装すると、従来のクライアントを使用した場合よりも管理オーバーヘッドが減ります。アプリケーションのビジネス論理は、中央に配置されるので、維持しやすくなります。ただし、シック・クライアントには、比較的強力なデスクトップ・コンピューターが必要となります。

## シン・クライアント

クライアントの容量は、Java アプレットを使用して削減することができます。95ページの図12 に、このタイプのシン・クライアントを実装する方法を示します。この例で、アプレットは社内イントラネットを介してブラウザーにダウンロードされ、クライアント・マシン上でローカルに実行されます。

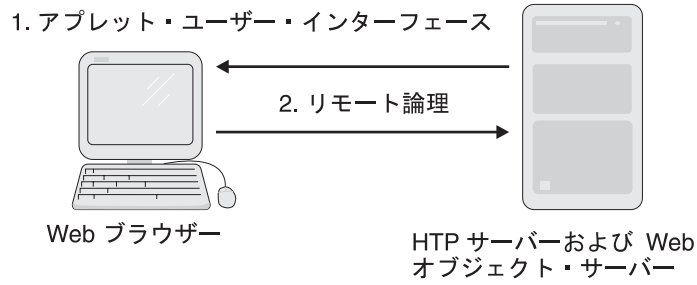


*図 12. シン・クライアントには、アプレット・ユーザー・インターフェースおよびリモート・ビジネス論理が割り当てられる*

シン・クライアントでは、クライアントの実行に数多くの高性能のマシンは必要ないので、コンピューター・リソースの使用効率は良くなります。また、シック・クライアントよりも管理オーバーヘッドも減ります。アプレットにアクセスするたびにアプレットの最新コピーがダウンロードされ、クライアントの更新は自動化されます。これは、多くのマシンが 1 つのクライアントを実行している環境では大きなメリットです。シン・クライアントも、強力なデスクトップ・コンピューターで実行する必要はありません。唯一の要件は、クライアント・マシンのブラウザーがアプレットで必要となる Java Virtual Machine (JVM) レベルをサポートしていることです。ただし、クライアント・ユーザー・インターフェースをネットワークを介してダウンロードしなければならないので、他のタイプのクライアントよりも、シン・クライアントのアプレットはダウンロード時間が長くなる可能性があります。

## シンナー・クライアント

シン・クライアントは、HTML と JavaScript を一緒に使用することで、さらに細く (シンナー) にすることができます (96ページの図13 を参照)。ユーザー要求は Java サーブレットに実行依頼されます。サーブレットは、Web サーバーからリモート処理要求を開始し、必要に応じて新しい HTML ページを動的に生成します。

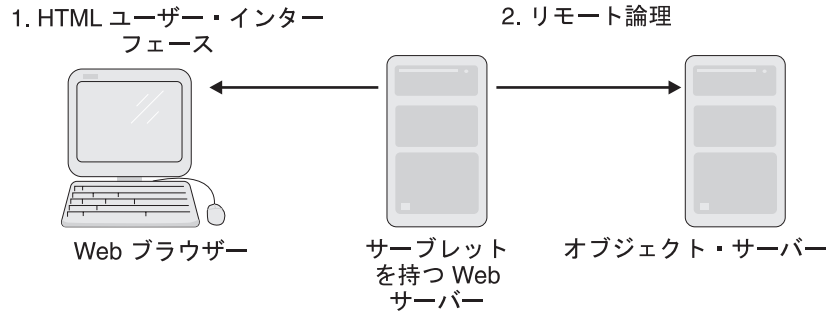


*図 13. サーブレットと一緒に使用するシンナー・クライアント*

シンナー・クライアントは、図13 に示すように、 Java アプレットを使って実装したシン・クライアントよりもダウンロード時間が短くなります。また、シンナー・クライアントは JVM も使用しないので、この面での非互換性問題も解消されます。ハードウェア要件が減り、クライアント管理が減るというアプレットのメリットを保持しています。同時に、クライアントのパフォーマンスと信頼性も向上します。また、ブラウザーと Web サーバーの間で HTTP を使用することにより、クライアントはファイアウォールを利用してサーブレットへ安全にアクセスすることができます。

シンナー・クライアントの使用の欠点は、クライアント機能のうち簡単なものしか提供されないということです。特に、Java の高度なユーザー・インターフェース機能は使用できません。ただし、複雑なユーザー・インターフェースを必要としないクライアントには、適切な選択肢です。

## サーバーのトポロジー

シック・クライアント、シン・クライアント、およびシンナー・クライアントの例は、 3 層分散コンピューティング・アーキテクチャーの 1 つ目の層と 2 つ目の層に焦点を当てていました。 WebSphere Application Server には、 2 つ目の層と 3 つ目の層のサーバーおよびリソースを編成できるモデルがいくつかあります。これらのサーバー・モデルのすべてで、 93ページの『クライアントのトポロジー』 で説明したクライアント・トポロジーを使用することができます。

### 分散サーバー

Application Servers アドバンスド版およびエンタープライズ版は、分散サーバーをサポートしています。 97ページの図14 に、物理的に分散されたネットワーク内の中間層マシンとバックエンド・マシンを示します。この例で、中間層

のマシンのプロセッサーは、3 つ目の層のものよりも小さく、低速で、費用もかかっていません。これらの小型サーバーの数が十分にあり、1 つにクラスター化されていれば、中間層は、その作業負荷を十分に処理できる処理能力を獲得することができます。これで、分散ネットワーク・トラフィックによって発生するオーバーヘッドの負担も軽くなります。また、クラスターの 1 つのメンバーがオフラインになっても他のマシンがその処理を引き継ぐことができるため、クラスターでは、サーバーの可用性も向上します。そのほかに、クラスター・サーバーには、クライアントに悪影響を与えずにコンピューティング能力を増分できるというメリットもあります。

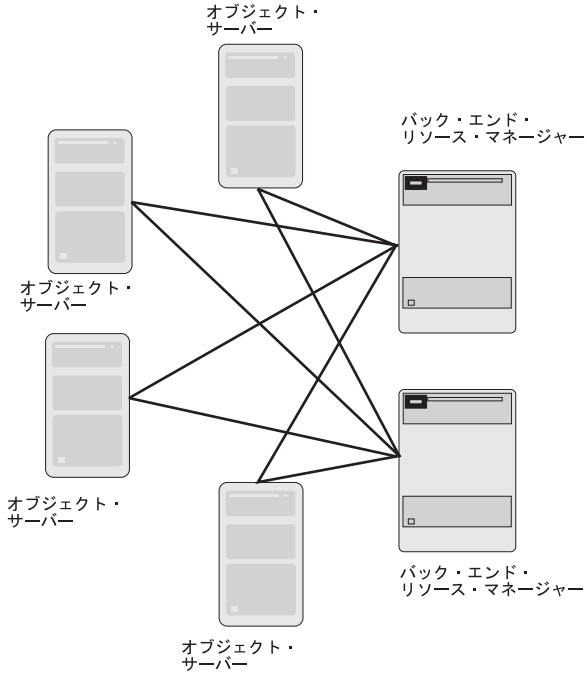


*図 14. リソース・マネージャーに接続された分散サーバーのクラスター*

このアプローチを使用する場合は、サーバーの複製が簡単で、またサーバーに透過的で、管理可能なロード・バランシングと障害サポートが用意されていなければなりません。さらに、中間層環境で実行するビジネス論理は、クラスター全体で整合性が取れていなければなりません

## 複製サーバー

このトポロジーでは、中間層内の各マシンが、サーバーの正確なレプリカ (すなわち複製) を取りまとめます。サーバーの 1 つがオフラインになっても他のマシンがその処理を引き継ぐことができるため、サーバーの複製では、サーバーの可用性が向上します。

Application Servers アドバンスド版およびエンタープライズ版は、作業負荷管理サービスを提供して、サーバーの複製間で処理要求を均等に分散させ、サーバーが予想外にオフラインになった場合には、フェイルオーバー・サービスを提供します。作業負荷管理は、管理サーバーだけでなくアプリケーション・サーバーにも適用されるので、アプリケーション・サーバー・クラスターの管理に悪影響が及ぶことはありません。

## 統合サーバー

多くの会社では、 3 層論理アーキテクチャーを実装しているかどうかにかかわらず、物理アーキテクチャーは 2 層に制限しています。 3 層の物理アーキテクチャーの運用コストは、かなり高額になる可能性があるからです。 図15 では、2 つ目の層と 3 つ目の層が物理的に統合されています。

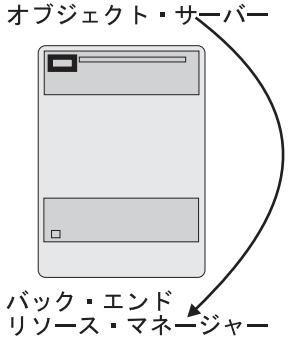


*図 15. 中間層とバックエンド層の物理的な統合*

この構成では、リモート通信を削減し、中間層のアプリケーション・サーバーと、その関連するバックエンド・システムの間のネットワーク・トラフィックのオーバーヘッドをなくすことで、パフォーマンスを向上させています。この中央実装の方が、クラスター・サーバーよりも管理が簡単です。

統合サーバー・アーキテクチャーを実装するシステムの拡大は、実装環境であるコンピューティング・プラットフォームによって異なります。このアーキテ

クチャーを使用するハイエンドなプラットフォーム (IBM OS/390 など) は、簡単に大規模なビジネス・システムに拡大することができます。

## Application Server スタンダード版のトポロジー

Application Server スタンダード版は、 WebSphere Application Server の単一マシン・バージョンです。 図16 に、Application Server スタンダード版の構成方法の例を示します。

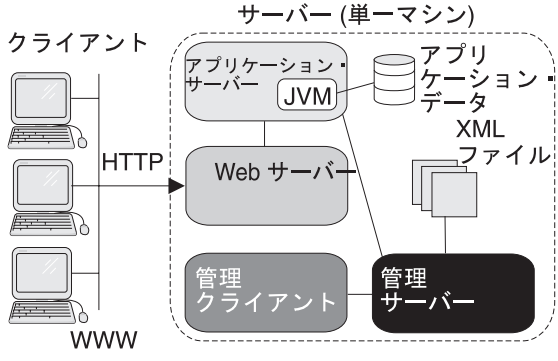


*図 16. Application Server スタンダード版の単純な構成*

クライアントは、WWW 上で HTTP 要求を介して Web サーバーにアクセスします。要求の処理は、アプリケーション・サーバーに依頼され、アプリケーション・サーバーは、同じマシン上にあるデータベースに問い合わせます。管理サーバーおよび管理クライアントが、Application Server スタンダード版環境を管理します。

## Application Server アドバンスド版のトポロジー

Application Server アドバンスド版は、組織のニーズに応じてさまざまなトポロジーで構成することができます。次に、Application Server アドバンスド版のトポロジー例をいくつか紹介します。

### 単純な構成

Application Server アドバンスド版の単純な構成を 100ページの図17 に示します。

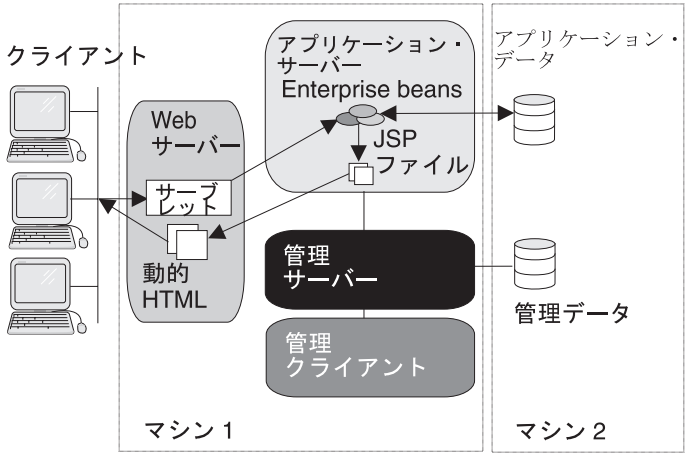


*図 17. Application Server アドバンスド版の単純な構成*

クライアントは、Application Server アドバンスド版で取りまとめられている Web ベースのビジネス・アプリケーションにアクセスします。アプリケーションは複数のサーバーで取りまとめることができますが、分かりやすいように、この構成ではアプリケーション・サーバーは 1 つとします。

クライアントは、Web サーバー上で実行されているサーブレットにアクセスします。サーブレットは、クライアント処理要求を中継して、別のマシン上のデータベースに透過的にアクセスする Enterprise beans に送ります (管理データも、このマシン上のデータベースに格納されます)。この動作の結果は、JavaServer Pages (JSP) ファイルを介して動的に生成される HTML を使用して、クライアントに表示されます。

## DMZ 構成

非武装地帯 (DMZ) トポロジー (101ページの図18 を参照) では、インターネットからアクセスされるビジネス・アプリケーションのセキュリティーが拡張されます。

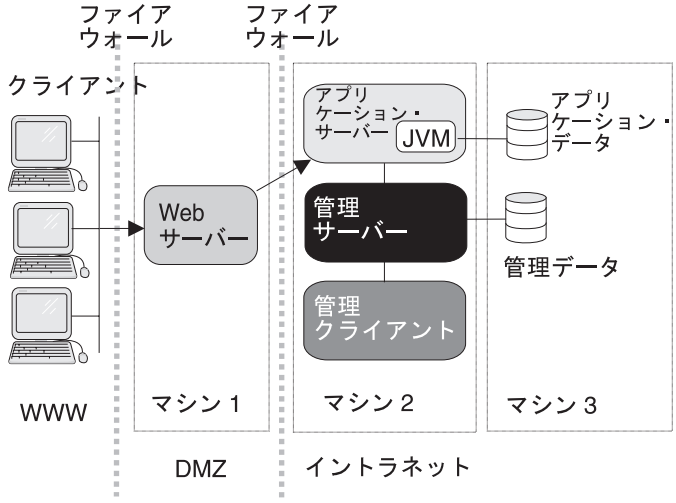


*図 18. Application Server アドバンスド版での DMZ 構成*

この構成では、Web サーバーは、クライアントと、アプリケーション・サーバー、管理サーバー、およびシステムが使用するバックエンド・データ・ストアの両方からファイアウォールによって隔離されます。インターネットからアプリケーションにアクセスしているクライアントが、アプリケーション・サーバー、データベースなど、組織が保護しているイントラネット上のリソースにアクセスするには、 DMZ 内のセキュア Web サーバーを通らなければなりません。これにより、機密性の高い企業リソースに対するリスクが最小限に抑えられます。

## TXSeries 構成

以下のセクションでは、CICS および Encina の構成の例をいくつか示します。

### 単純な CICS 構成

単純な分散 CICS 環境では、102ページの図19 に示すように、1 台のマシンに CICS クライアントが置かれ、もう 1 台のマシンに CICS 領域が置かれます。この構成は、インストール、テスト、および保守が最も容易なため、初めての CICS インストールにはこの構成をお勧めします。

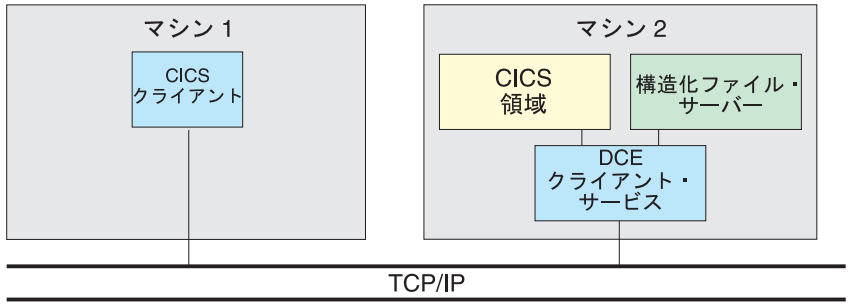


*図 19. 非 DCE セル環境で CICS を使用する単純な分散構成*

この構成例は、RPC 専用環境における CICS を表しています。RPC 専用環境では、内部 CICS セキュリティーおよびディレクトリー・サービスが使用されます。使用される DCE サービスは、トランザクション・データのエンドポイント・マッピングと転送のための DCE RPC だけです。

この構成例は、以下のことを表しています。

- SFS サーバーは、CICS 領域ファイルおよびキューに使用します。また、ユーザー・データの格納にも使用できます。
- CICS 領域と SFS サーバー間の通信には、DCE RPC デーモンが提供する DCE RPC を使用します。それ以外の DCE クライアント・サービスは使用されません。
- CICS クライアントは、CICS 領域への即時 3270 端末アクセスおよびプログラム・アクセスを提供します。

## DCE セル内での単純な CICS 構成

分散コンピューティング環境 (DCE) セル内での単純な分散 CICS 環境は、103ページの図20 に示すように、 1 台のマシン上のクライアントと、別のマシン上で実行されている CICS 領域から成り立ちます。この構成のためには、DCE セル内のマシンに DCE CDS および DCE セキュリティー・サービスをインストールする必要があります。

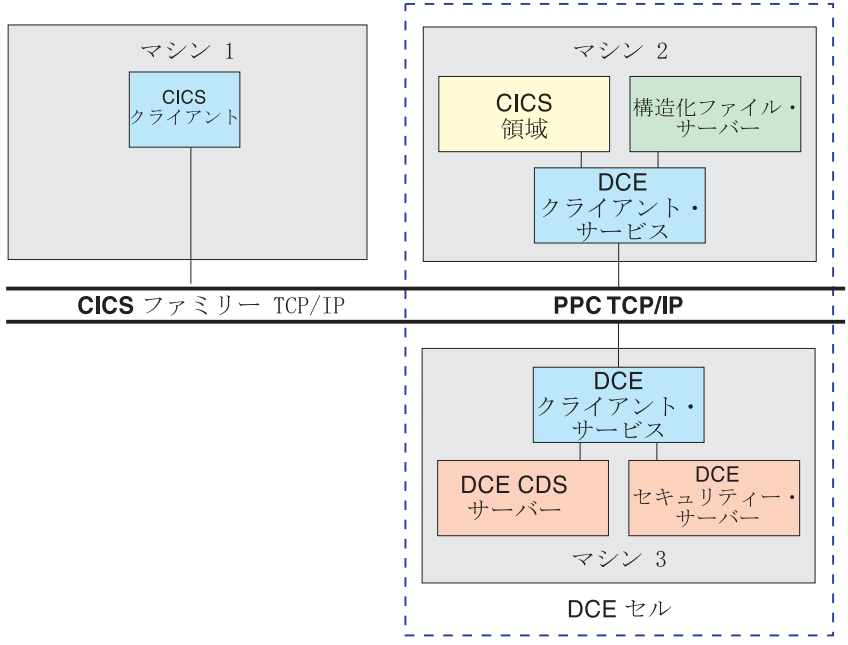


*図 20. DCE セル環境で CICS を使用する分散構成*

この構成例は、以下のことを表しています。

- サーバー・ロケーションおよびセキュリティー・サービスは DCE によって提供されます。
- SFS サーバーは、CICS 領域ファイルおよびキューに使用します。また、ユーザー・データの格納にも使用できます。
- CICS 領域 と SFS サーバーの間の通信に使用される DCE RPC は、 DCE セキュリティー・サーバーで認証することができます。
- CICS クライアントは、CICS 領域への即時 3270 端末アクセスおよびプログラム・アクセスを提供します。

**注:** CICS クライアント・マシンは、DCE セル・サービスを使用する CICS 領域を実行する場合を除き、DCE セルの一部である必要はありません。

## 単純な Encina モニター・セル構成

104ページの図21 に示されている単純な Encina モニター・セル構成は、以下のものを含みます。

- セル・マネージャー。これは、セルにおけるノード・マネージャーの活動を調整します。

- ノード・マネージャー。これは、そのノード (マシン) で実行されているすべてのサーバーの活動を制御します。
- モニター・アプリケーション・サーバー。これは Encina アプリケーションのビジネス論理を提供するもので、SFS サーバーに保管されたデータを扱います。
- モニター・クライアント。これは、Encina アプリケーションの表示論理を提供します。

モニター・セルは DCE セルの一部であり、DCE CDS および DCE セキュリティー・サービスがインストールされた別のマシンを含んでいます。

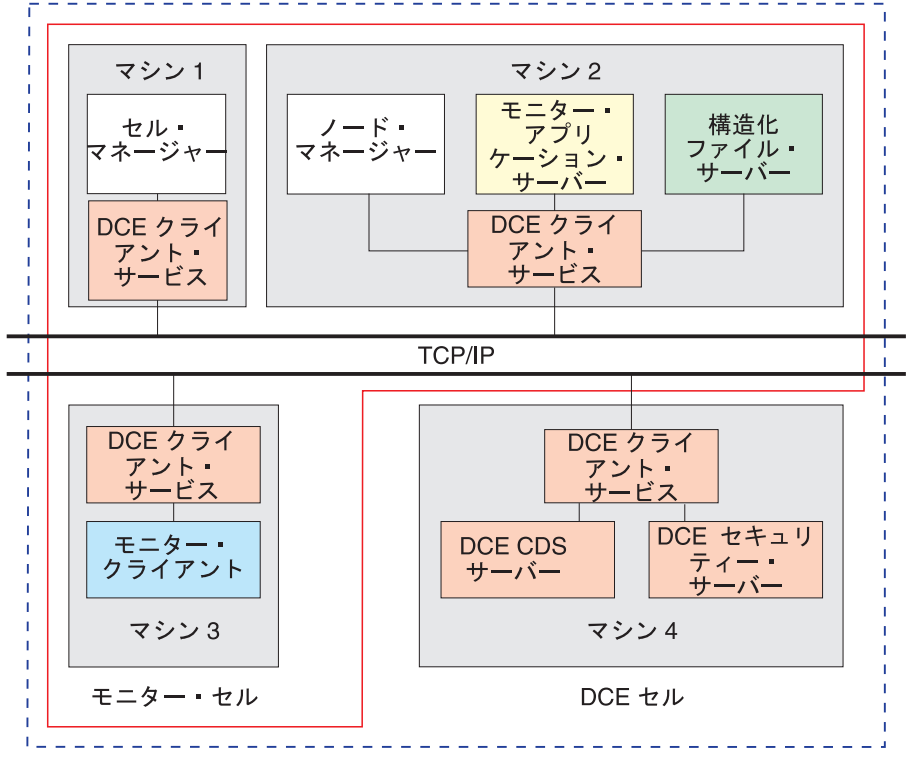


*図 21. 単純な Encina Monitor 構成*

サーバー・ロケーションおよびセキュリティー・サービスは DCE によって提供されます。 Encina サーバーとクライアントの間の通信に使用される DCE RPC は、DCE セキュリティー・サービスによって認証することができます。

## Component Broker 構成

次に、Component Broker の構成例を紹介します。

### 単純な構成

Component Broker な単純なトポロジー (図22 を参照) では、単一のアプリケーション・サーバーを使用して、同一のホスト上ですべてのビジネス・オブジェクトが実行されます。

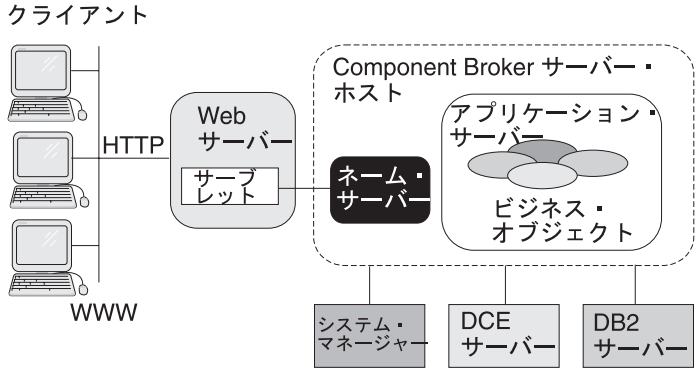


*図 22. Component Broker の基本トポロジー*

Component Broker ホスト・サーバーのほかに、このトポロジーには 3 つのサポート・サーバーが必要です。システム・マネージャーが、Component Broker のインストールを実行し、アプリケーション・サーバーを取りまとめる Component Broker サーバーを管理します。 DCE サーバーは、基本サービスを提供します。 DB2 サーバーは、使用頻度の高い 3 つ目の層のリソース・マネージャーを表しています。この層で構成できるその他のリソース・マネージャーには、IMS および CICS が含まれます。

### 基本作業負荷管理構成

106ページの図23 に示すトポロジーは、基本作業負荷管理構成を表しています。ビジネス・オブジェクトは 1 つのアプリケーション・サーバー上で実行されますが、このサーバーの複数のコピーは、ネットワーク内の別個のマシン上で実行されています。

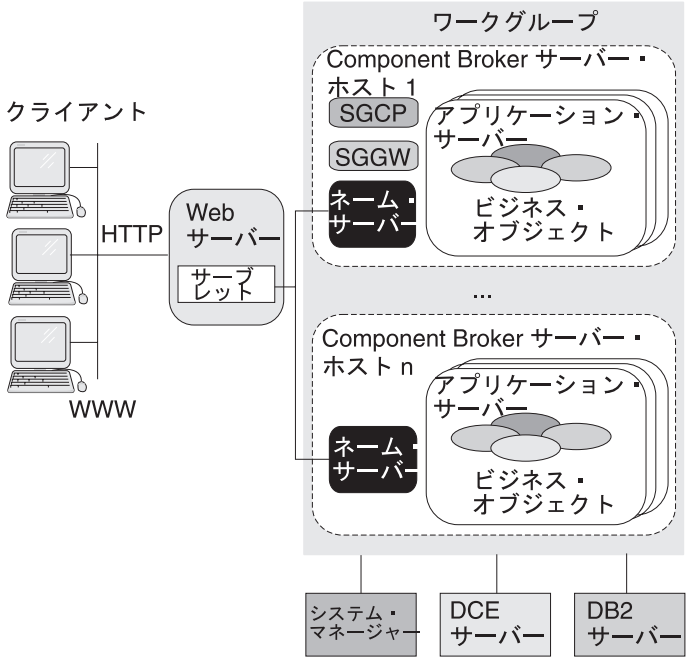


*図 23. Component Broker の水平作業負荷管理トポロジー*

2 つの Component Broker サーバーが 1 つのワークグループに構成されています。各 Component Broker サーバーでは、複数のアプリケーション・サーバーが実行されています。

アプリケーション・サーバーの数を増加すると、アプリケーションをサポートするネットワークの能力が高くなります。クライアント要求は、作業負荷管理グループ内のアプリケーション・サーバー間で分散できるため、クライアントは、サーバー・グループを単一の論理サーバーと考えます。

制御下のサーバー・グループを使用して作業負荷管理を使用可能にする場合は、ネットワーク内のホストの 1 つに、さらに 2 つのサーバーを置かなければなりません。

- サーバー・グループ制御点 (SGCP)
- サーバー・グループ・ゲートウェイ (SGGW)

これらのサーバーは、Component Broker の作業負荷管理サービスの基本サポートを提供します。

# 付録. WebSphere Application Server のライブラリー

次の表に WebSphere Application Server の完全な文書ライブラリーをリストします。すべての文書は、WebSphere Application Server ライブラリー・ページ (www.software.ibm.com/webservers/appserv/library.html) から入手できます。

*表 4. WebSphere Application Server のライブラリー*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| **WebSphere Application Server の共通資料 (サポートされるすべてのプラットフォーム用)** | | |
| SC88-8798 | *WebSphere Application Server 概説* | WebSphere Application Server のすべての版、および各版の内容について、ファミリー全体に共通する概要を示しています。以前は、*Introduction to WebSphere Application Server* というタイトルでした。 |
| SD88-7343 | *Enterprise Beans の作成* | WebSphere Application Server のアドバンスド版およびエンタープライズ版で Enterprise JavaBeans™ コンポーネントを使用して行うプログラミングを紹介しています。 |
| SD88-7362 | *WebSphere ビジネス構築のソリューション* | WebSphere Application Server ファミリーにおけるアプリケーション開発および推奨されるプログラミング慣行を示す、プログラミングの例とシナリオを提供しています。WebSphere ファミリーのその他の製品も説明されています。 |
| GC88-8940 | *パフォーマンス・モニタリング・インフラストラクチャー・クライアント・パッケージの使用ガイド* | パフォーマンス・モニタリング・インフラストラクチャー (PMI) インターフェースを使用して、 WebSphere サーバーからパフォーマンス情報を集め、処理し、表示する WebSphere Application Server クライアントを作成する方法を紹介します。 |
| GC88-8941 | *JRas メッセージ・ロギングおよびトレース機能の使用ガイド* | WebSphere JRas インターフェースを使用して、メッセージ・ロギング機能を WebSphere Application Server アプリケーションに組み込む方法を紹介します。 |
| GC88-8614 | *Workarea 機能の使用ガイド* | WorkArea 機能を使用して、WebSphere Application Server アプリケーションで有効範囲内のコンテキストを設定、取得、および管理する方法を紹介します。 |
| **WebSphere Application Server スタンダード版およびアドバンスド版の資料 (サポートされるすべてのプラットフォーム用)** | | |
| | インフォセンター | WebSphere Application Server スタンダード版およびアドバンスド版のインストール、構成、保守、およびプログラミングに関する情報を提供しています。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| **Component Broker の共通資料 (サポートされるすべてのプラットフォーム用)** | | |
| SD88-7372 | *プログラミング・ガイド* | サポートされるすべてのプラットフォームで、サポートされるすべてのプログラム言語によって行う、Component Broker アプリケーションの開発について説明しています。ビジネス・オブジェクトやデータ・オブジェクトを含むプログラミング・モデルについて説明し、さまざまな基本的なプログラミングの問題に関する情報を記載しています。 |
| SD88-7373 | *上級プログラミング・ガイド* | Component Broker による、共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) オブジェクト・サービス、 Component Broker オブジェクト・リクエスト・ブローカー (ORB)、キャッシュ、通知、照会、セッション、その他のサービス、言語間オブジェクト・モデル (IOM)、および作業負荷管理の実装について説明しています。 |
| SD88-7374 | *MQSeries アプリケーション・アダプター開発ガイド* | MQSeries アプリケーション・アダプターを使用する Component Broker アプリケーションの開発に関する情報を提供します。 |
| SC88-8792 | *手続き型アプリケーション・アダプター開発ガイド* | 手続き型アプリケーション・アダプターを使用する Component Broker アプリケーションの開発に関する情報を提供します。 |
| SC88-8793 | *データベース・アプリケーション・アダプター開発ガイド* | DB2、Oracle、および Informix のアプリケーション・アダプターを使用する Component Broker アプリケーションの開発に関する情報を提供します。 |
| SD88-7375 | *システム管理ガイド* | サポートされるすべてのプラットフォームで Component Broker および Component Broker アプリケーションを管理するための情報を提供しています。また、System Manager インターフェースの一般的な使用法についても説明しています。 |
| SC88-8794 | *プログラミング・リファレンス* | サポートされるすべてのプログラム言語で使用可能な、完全な Component Broker アプリケーション・プログラミング・インターフェース (API) について記述しています。 |
| SD88-7378 | *アプリケーション開発ツール・ガイド* | Component Broker ツールキット・コンポーネントに関して、継承やチーム開発などの共通開発シナリオに焦点を当てて説明します。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| SD88-7379 | *問題判別ガイド* | 管理者およびプログラマーが Component Broker および Component Broker アプリケーションに関する問題を識別し、解決するために必要な情報を提供します。インストールの問題、実行時エラー、アプリケーションのデバッグ、およびログ・メッセージの分析に関する情報が記載されています。 |
| SD88-7380 | *用語集* | Component Broker 資料で一般的に使用されている用語をリストし、定義しています。 |
| GD88-7419 | *Component Broker リリース情報* | 最新の機能変更、認識されている制限事項など、Component Broker に関するプラットフォーム固有およびリリース固有の情報と、制限に対する適切な対処方法 (可能な場合) を提供しています。 |
| | *Distributed Debugger for Workstations* | ワークステーション上の Distributed Debugger プログラムに関する情報を紹介します。 |
| | *Distributed Debugger for OS/390* | OS/390 上の Distributed Debugger プログラムに関する情報を紹介します。 |
| | *Object Level Trace* | Object Level Trace に関する情報を紹介します。 |
| **Component Broker for AIX の資料** | | |
| SD88-7367 | *計画、パフォーマンスおよびインストール・ガイド (AIX)* | AIX 上で最新バージョンの Component Broker をインストール、構成、およびアップグレードを行うための、完全な手順を示しています。 |
| **Component Broker for Solaris の資料** | | |
| SD88-7368 | *計画、パフォーマンスおよびインストール・ガイド (Solaris)* | Solaris 上で最新バージョンの Component Broker をインストールし、構成するための、完全な手順を示しています。 |
| **Component Broker for Windows システムの資料** | | |
| SD88-7366 | *計画、パフォーマンスおよびインストール・ガイド (Windows システム)* | Windows システム で Component Broker の最新バージョンのインストール、構成、およびアップグレードを行うための、完全な手順を示しています。 |
| **Component Broker for OS/390 の資料** | | |
| GA88-8513 | *WebSphere Application Server エンタープライズ版 for OS/390 Component Broker 計画およびインストール・ガイド* | OS/390 プラットフォームにおける Component Broker の計画およびインストールの注意点について説明しています。 |

表 4. WebSphere Application Server のライブラリー (続き)

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| GA88-7007 | *WebSphere Application Server エンタープライズ版 for OS/390 Component Broker アプリケーション・アセンブル・ガイド* | OS/390 で Component Broker を使用して行うアプリケーションのアセンブルに関する情報を記載しています。 |
| GA88-7009 | *WebSphere Application Server for OS/390 ご使用になる前に* | OS/390 プラットフォームでの WebSphere Application Server の計画に関する考慮事項を説明しています。 |
| GA88-7008 | *WebSphere Application Server エンタープライズ版 for OS/390 Component Broker 運用および管理ガイド* | OS/390 プラットフォームでの Component Broker の操作および管理に関する情報を記載しています。 |
| GA22-7329 | *OS/390 Component Broker Messages and Diagnosis* | OS/390 プラットフォーム上の Component Broker に関する診断情報を提供し、メッセージを説明しています。 |
| SC33-6587 | *OS/390 Component Broker System Management User Interface* | OS/390 プラットフォーム上の Component Broker で提供される、システム管理ユーザー・インターフェースについて説明しています。 |
| GA22-7478 | *OS/390 System Management Scripting Application Programming Interface* | OS/390 プラットフォーム上の Component Broker で提供される、システム管理スクリプト・インターフェースについて説明しています。 |
| GA88-8514 | *プログラム・ディレクトリー WebSphere Application Server for OS/390* | OS/390 プログラム・ディレクトリーに対する WebSphere Application Server について説明しています。 |
| GA22-7462 | *WebSphere Application Server for OS/390 Licensed Program Specifications* | OS/390 ライセンス・プログラム仕様に対する WebSphere Application Server について説明しています。 |
| **TXSeries (CICS および Encina) の共通資料 (サポートされるすべてのプラットフォーム用)** | | |
| SC88-8799 | *概説および計画ガイド* | TXSeries 製品を紹介し、トランザクション処理、CICS、および Encina について高レベルの説明を行っています。 |
| GD88-7420 | *TXSeries リリース情報* | プラットフォームおよびリリースに特有な TXSeries に関する情報を提供します。この中には、TXSeries README ファイルよりも完全な新規機能に関する説明、製品資料には記載できなかった機能または変更に関する情報、製品の最新リリース以降に修正された障害の説明、および TXSeries についてすでに知られている制約事項と (可能な場合には) その適切な回避手段に関する情報が含まれています。 |

表 4. WebSphere Application Server のライブラリー (続き)

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| **TXSeries for AIX の資料** | | |
| SD88-7382 | *計画およびインストール・ガイド (AIX)* | AIX 上で TXSeries (CICS および Encina) の最新バージョンのインストール、構成、およびアップグレードを行うための、完全な手順を示しています。 |
| **TXSeries for Solaris の資料** | | |
| SD88-7383 | *計画およびインストール・ガイド (Solaris)* | Solaris 上で TXSeries (CICS および Encina) の最新バージョンのインストール、構成、およびアップグレードを行うための、完全な手順を示しています。 |
| **TXSeries for Windows システムの資料** | | |
| SD88-7381 | *計画およびインストール・ガイド (Windows システム)* | Windows 上で TXSeries (CICS および Encina) の最新バージョンのインストール、構成、およびアップグレードを行うための、完全な手順を示しています。 |
| **TXSeries for HP-UX の資料** | | |
| SC88-8800 | *計画およびインストール・ガイド (HP-UX)* | HP-UX 上で TXSeries (CICS および Encina) の最新バージョンのインストール、構成、およびアップグレードを行うための、完全な手順を示しています。 |
| **CICS の資料 (特に明記されていない限り、サポートされるすべてのプラットフォーム用)** | | |
| SC88-8804 | *CICS 管理ガイド (オープン・システム)* | UNIX で CICS および CICS アプリケーションを管理するためのガイド情報を提供しています。 CICS 用の Tivoli インターフェースに関する情報も記載しています。また、CICS 用語集も含まれています。 |
| SD88-7385 | *CICS 管理ガイド (Windows システム)* | Windows システムで CICS および CICS アプリケーションを管理するためのガイド情報を提供しています。 CICS 用の Tivoli インターフェースに関する情報も記載しています。また、CICS 用語集も含まれています。 |
| SD88-7388 | *CICS 管理リファレンス* | サポートされるすべてのプラットフォームで CICS を管理するために使用されるコマンドに関する、完全な参照情報を記載しています。 |
| SD88-7389 | *CICS アプリケーション・プログラミング・ガイド* | サポートされるすべてのプラットフォームで、サポートされるすべてのプログラム言語によって行う、CICS ベースのアプリケーションの開発について説明しています。 |
| SD88-7390 | *CICS アプリケーション・プログラミング・リファレンス* | サポートされるすべてのプラットフォームにおける、サポートされるすべてのプログラム言語の、 CICS アプリケーション・プログラミング・インターフェースに関する参照情報を記載しています。 |

表 4. WebSphere Application Server のライブラリー (続き)

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| SD88-7391 | *CICS 相互通信ガイド* | CICS 領域とその他のシステム (たとえば、UNIX または Windows マシン上にある別の CICS 領域、またはメインフレームなどのシステム上の別のアプリケーション) との間での通信を実装する方法を説明しています。 |
| SC09-4589 | *CICS Messages and Codes* | TXSeries CICS システムによって発行される可能性のあるすべてのメッセージおよびコードをリストし、説明しています。 |
| SD88-7394 | *CICS 問題判別ガイド* | 管理者が CICS システムまたはアプリケーションの問題を識別および診断するのを支援します。問題の症状とその可能な原因を記載しています。 |
| SD88-7395 | *CICS ワークロード管理の使用ガイド* | CICS 作業負荷管理ユーティリティーについて説明しています。 |
| SC09-4585 | *CICS IIOP および Java プログラミング・ガイド* | CICS Internet Inter-ORB Programming (IIOP) インターフェースについて説明します。このインターフェースは、CICS アプリケーションが共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) およびオブジェクト・リクエスト・ブローカー (ORB) と通信できるようにするものです。また、CICS 用の Java アプリケーション・プログラミング・インターフェースを使用する方法についても説明しています。 |
| SD88-7397 | *Windows NT の CICS フロントエンド・プログラミング・インターフェース* | Windows NT 用の CICS フロントエンド・プログラミング・インターフェース (FEPI) を使用するアプリケーションの開発に関する情報を提供します。 |
| SD88-7398 | *CICS 環境での IBM Communication Server for AIX の使用ガイド* | IBM Communications Server for AIX で提供されるシステム・ネットワーク体系 (SNA) パッケージとともに CICS を使用するための情報を提供します。 |
| SD88-7399 | *CICS 環境での IBM Communications Server for Windows Systems の使用ガイド* | IBM Communications Server for Windows システムで提供されるシステム・ネットワーク体系 (SNA) パッケージとともに CICS を使用するための情報を提供します。 |
| SD88-7400 | *CICS 環境での Microsoft SNA Server の使用ガイド* | Microsoft for Windows システムで提供されるシステム・ネットワーク体系 (SNA) パッケージとともに CICS を使用するための情報を提供します。 |
| SD88-7401 | *CICS 環境での SNAP-IX for Solaris の 使用ガイド* | Solaris 用に Data Connection Limited (DCL) によって提供されるシステム・ネットワーク体系 (SNA) パッケージとともに CICS を使用するための情報を提供します。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| SC88-8803 | *CICS 環境での HP-UX SNAplus2 の使用ガイド* | Hewlett-Packard で提供される SNAplus2 システム・ネットワーク体系 (SNA) パッケージとともに CICS を使用するための情報を提供します。 |
| **Encina 管理ガイド資料 (サポートされるすべてのプラットフォーム用)** | | |
| SD88-7402 | *Encina 管理ガイド 第 1 巻: 基本管理* | サポートされるすべてのプラットフォームにおける Encina および Encina アプリケーションの管理に関する基本情報を提供します。本書には、Enconsole 管理インターフェースの使用法が説明され、完全な Encina 用語集が含まれています。また、Encina 用の Tivoli インターフェースに関する情報も記載しています。 |
| SD88-7403 | *Encina 管理ガイド 第 2 巻: サーバー管理* | 構造化ファイル・サーバー (SFS)、回復可能キューイング・サービス (RQS)、対等通信 (PPC) Gateway、および DCE Encina Lightweight Client (DE-Light) Gateway の各サーバーの管理を説明しています。 |
| SD88-7404 | *Encina 管理ガイド 第 3 巻: 拡張管理* | Encina オブジェクト階層および Encina とともに使用することのできるコマンド行スクリプト・インターフェース **enccp** などの、拡張管理トピックを説明しています。また、すべての Encina コンポーネントによって使用される環境変数を説明する付録も含まれています。 |
| **Encina プログラミング・ガイド資料 (サポートされるすべてのプラットフォーム用)** | | |
| SD88-7405 | *Encina COBOL プログラミング・ガイド* | Encina で使用可能な COBOL アプリケーション・プログラミング・インターフェースについて説明しています。 |
| SD88-7406 | *Encina モニター・プログラミング・ガイド* | Encina モニター・アプリケーション・プログラミング・インターフェースを使用してプログラムを書く方法を説明しています。 |
| SD88-7407 | *Encina オブジェクト指向プログラミング・ガイド* | 分散コンピューティング環境 (DCE) または共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) 環境でアプリケーションを開発するために使用される、 Encina オブジェクト指向 (Encina++) アプリケーション・プログラミング・インターフェースによるプログラミングについて説明しています。 |
| SD88-7408 | *Encina RQS++ および SFS++ プログラミング・ガイド* | 構造化ファイル・サーバー (SFS) および回復可能キューイング・サービス (RQS) コンポーネント用に Encina オブジェクト指向アプリケーション・プログラミング・インターフェースを使用してプログラムを書く方法について説明しています。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| SD88-7409 | *Encina DE-Light* プログラミング・ガイド | DCE Encina Lightweight Client (DE-Light) アプリケーション・プログラミング・インターフェースに従ってプログラムを書く方法を説明しています。これらのインターフェースは、 DE-Light Gateway サーバーと通信し、分散コンピューティング環境 (DCE) や Encina のようなリソースを大量に消費するシステムをサポートできない小型機から、DCE および Encina アプリケーションにアクセスするための、C および Java ベースの手段です。 |
| SD88-7410 | *Encina PPC* サービス・プログラミング・ガイド | Encina Peer-to-Peer Communications (PPC) エグゼクティブ・アプリケーション・プログラミング・インターフェースに従ってプログラムを書く方法を説明しています。 |
| SD88-7411 | *Encina RQS* プログラミング・ガイド | Encina 回復可能キューイング・サービス (RQS) アプリケーション・プログラミング・インターフェースに従ってプログラムを書く方法を説明しています。 |
| SD88-7412 | *Encina SFS* プログラミング・ガイド | Encina 構造化ファイル・サーバー (SFS) アプリケーション・プログラミング・インターフェースに従ってプログラムを書く方法を説明しています。 |
| SD88-7413 | *Encina* ツールキット・プログラミング・ガイド | 各種の Encina Client (エグゼクティブ) および Server Core アプリケーション・プログラミング・インターフェースに従ってプログラムを書く方法を説明しています。 |
| SD88-7414 | *Encina Transactional* プログラミング・ガイド | Encina Transactional-C (Tran-C) アプリケーション・プログラミング・インターフェースに従ってプログラムを書く方法を説明しています。 |
| SD88-7415 | *Encina* アプリケーションの作成 | Encina アプリケーションの作成を紹介しています。本書はチュートリアルに似た方法を使用して、各種の Encina アプリケーション・プログラミング・インターフェースの開発を説明しています。 Encina の用語およびアプリケーション開発に関する一般情報が記載されています。 |
| SD88-7416 | *Windows* 環境での *Encina* アプリケーションの作成 | Windows システムで TXSeries アプリケーション開発ツールキット (ADK) を使用して Encina アプリケーションを開発するための情報を提供します。 ADK は Microsoft COM インターフェースもサポートしますが、本書では、それについても説明しています。 |
| SC09-4488 | *Encina Messages and Codes* | Encina によって発行される可能性のあるすべてのメッセージおよび状況コードをリストし、説明しています。本書の情報は、管理者と開発者が Encina または Encina アプリケーションの問題を理解し、訂正できるように支援するために構成されています。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| **Encina の管理参照資料 (サポートされるすべてのプラットフォーム用)** | | |
| GD88-7421 | *管理の紹介ページ* | Encina 管理インターフェースを紹介しています。各種のインターフェース、その使用、および構文規則に関する高レベルな説明を行っています。 |
| GD88-7422 | *drpcadmin* コマンド・ページ | **drpcadmin** 管理コマンドの組について説明しています。これらのコマンドは、DCE Encina Lightweight Client (DE-Light) Gateway サーバーを管理するために使用されます。 |
| GD88-7423 | *emadmin* コマンド・ページ | **emadmin** 管理コマンドの組について説明しています。これらのコマンドは、Encina で提供されたオブジェクト階層を操作するために、以前に使用されていたものです。これらの機能は **enccp** インターフェースによって取って代わられたため、これらのコマンドは Encina の今回のリリースおよび将来のリリースでは使用されなくなりました。 |
| GD88-7424 | *enccp* 紹介ページ | Encina 制御プログラム (**enccp**) コマンド行およびスクリプト・インターフェースを紹介しています。**enccp** インターフェースにおけるコマンドを紹介し、コマンドの共通構文を説明し、他の **enccp** 解説書のページで提供された情報を一般的に補足しています。 |
| GD88-7425 | *enccp* サンプル・ページ | **enccp** インターフェースにおけるコマンドの例を示しています。 |
| GD88-7426 | *enccp* オブジェクト・ページ | Encina モニター・オブジェクト階層を構成するオブジェクトを説明しています。この階層は、**emadmin** コマンドの組によって操作されていた従来のオブジェクトのセットに取って代わるものです。 |
| GD88-7427 | *enccp* オペレーション・ページ | **enccp** インターフェースで使用可能なオペレーション (コマンド) について説明しています。これらのコマンドは、Encina モニター・オブジェクト階層を操作するために使用されます。 |
| GD88-7428 | *各種管理リファレンス・ページ* | Encina で使用可能な、各種の (組になっていない) コマンドを説明しています。これには、**ecm** (Encina セル・マネージャー) 始動コマンド、**enm** (Encina ノード・マネージャー) 始動コマンド、**rqs**、**sfs**、**ppcgwy**、および **drpcgwy** 始動コマンド、Encina 始動スクリプト、および Encina コマンド行トレース・ユーティリティーが含まれます。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| GD88-7429 | *otsadmin* コマンド・ページ | **otsadmin** 管理コマンドの組について説明しています。これらのコマンドは、Encina オブジェクト・トランザクション・サービス (OTS) プログラミング・インターフェースに基づいて、ツールキットに似たサーバー管理を行うために使用されます。 |
| GD88-7430 | *ppcadmin* コマンド・ページ | **ppcadmin** 管理コマンドの組について説明しています。これらのコマンドは、対等通信 (PPC) Gateway サーバー、およびこのサーバーのシステム・ネットワーク体系 (SNA) との対話を管理するために使用されます。 |
| GD88-7431 | *rqsadmin* コマンド・ページ | **rqsadmin** 管理コマンドの組について説明しています。これらのコマンドは、回復可能キューイング・サービス (RQS) サーバーとその中のデータを管理するために使用されます。 |
| GD88-7432 | *sfsadmin* コマンド・ページ | **sfsadmin** 管理コマンドの組について説明しています。これらのコマンドは、構造化ファイル・サーバー (SFS) サーバーとその中のデータを管理するために使用されます。 |
| GD88-7433 | *tkadmin* コマンド・ページ | **tkadmin** 管理コマンドの組について説明しています。これらのコマンドは、Encina ツールキット・サーバーを管理するために使用されます。これらのコマンドを使用すると、データおよびログ・ボリューム、サーバー・トレース、トランザクション活動、および ツールキット・サーバーのその他の局面を管理することができます。 |
| **Encina プログラミング参照資料 (サポートされるすべてのプラットフォーム用)** | | |
| GD88-7434 | *C プログラミングの紹介ページ* | Encina で使用可能な C プログラム言語のアプリケーション・プログラミング・インターフェースを紹介しています。また、C インターフェースに従うプログラミングに関連した、高レベルの事項についても説明しています。 |
| GD88-7435 | *アボート機能プログラミングのページ* | Encina で使用可能な C 言語ベースのアボート機能アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースはツールキット・エグゼクティブの一部です。 |
| GD88-7436 | *DE-Light C プログラミングのページ* | DCE Encina Lightweight Client (DE-Light) で使用可能な C 言語ベース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ついて説明しています。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| GD88-7437 | *分散トランザクション・サービス (TRAN) プログラミングのページ* | Encina で使用可能な C 言語ベースの分散トランザクション・サービス (TRAN) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースはツールキット・エグゼクティブの一部です。 |
| GD88-7438 | *EMA プログラミングのページ* | Encina で使用可能な C 言語ベースの Encina モニター管理 (EMA) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは、Encina で提供されたオブジェクト階層を操作するために、以前に使用されていたものです。これは、Encina の今回のリリースおよび将来のリリースでは使用されなくなりました。 |
| GD88-7439 | *ロック・サービス (LOCK) プログラミングのページ* | Encina で使用可能な C 言語ベースのロック・サービス (LOCK) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは Toolkit Server Core の一部です。 |
| GD88-7440 | *ログ・サービス (LOG) プログラミングのページ* | Encina で使用可能な C 言語ベースのログ・サービス (LOG) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは Toolkit Server Core の一部です。 |
| GD88-7441 | *各種 C プログラミングのページ* | Encina で使用可能な各種の C 言語ベースの関数について説明しています。これには、トランザクション・インターフェース定義 (**tidl**) コンパイラーおよび各種変換関数が含まれます。 |
| GD88-7442 | モニター・プログラミングのページ | Encina で使用可能な C 言語ベースのモニター・アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7443 | *PPC エグゼクティブ・プログラミングのページ* | Encina で使用可能な C 言語ベースの対等通信 (PPC) エグゼクティブ・アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7444 | *リカバリー・サービス (REC) プログラミングのページ* | Encina で使用可能な C 言語ベースのリカバリー・サービス (REC) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは Toolkit Server Core の一部です。 |
| GD88-7445 | *リスタート・サービス・プログラミングのページ* | Encina で使用可能な C 言語ベースの再始動サービス・アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは Toolkit Server Core の一部です。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| GD88-7446 | *RQS* プログラミングのページ | Encina で使用可能な C 言語ベースの回復可能キューイング・サービス (RQS) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7447 | *SFS* プログラミングのページ | Encina で使用可能な C 言語ベースの構造化ファイル・サーバー (SFS) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7448 | *T-ISAM* プログラミングのページ | Encina の構造化ファイル・サーバー (SFS) コンポーネントとともに使用することのできる C 言語ベースのトランザクション索引順次アクセス方式 (T-ISAM) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7449 | *ThreadTid* プログラミングのページ | Encina で使用可能な C 言語ベースの Thread-to-Tid マッピング・サービス (ThreadTid) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースはツールキット・エグゼクティブの一部です。 |
| GD88-7450 | *TM-XA* プログラミングのページ | Encina で使用可能な C 言語ベースのトランザクション・マネージャー XA サービス (TM-XA) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは Toolkit Server Core の一部です。 |
| GD88-7451 | トレース機能プログラミングのページ | Encina で使用可能な C 言語ベースのトレース機能アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7452 | *Transactional-C* プログラミングのページ | Encina で使用可能な C 言語ベースの Transactional-C (Tran-C) アプリケーション・プログラミング・インターフェースについて説明しています。のインターフェースは、C プログラム言語にトランザクション向けの拡張および構成を提供するものです。 |
| GD88-7453 | *TranLog* プログラミングのページ | Encina で使用可能な C 言語ベースのトランザクション状態ログ (TranLog) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは Toolkit Server Core の一部です。 |
| GD88-7454 | *TRDCE* プログラミングのページ | Encina で使用可能な C 言語ベースの Transarc Encina 分散コンピューティング環境 (TRDCE) ユーティリティーについて説明しています。このアプリケーション・プログラミング・インターフェースは、ツールキット・エグゼクティブの一部です。 |

表 4. *WebSphere Application Server* のライブラリー *(続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| GD88-7455 | *TRPC* プログラミングのページ | Encina で使用可能な C 言語ベースのトランザクション・リモート・プロシージャー呼び出し (TRPC) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7456 | *TX* インターフェース・プログラミングのページ | Encina で使用可能な C 言語ベースの Encina X/Open TX アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7457 | ボリューム・サービス *(VOL)* プログラミングのページ | Encina で使用可能な C 言語ベースのボリューム・サービス (VOL) アプリケーション・プログラミング・インターフェースについて説明しています。このインターフェースは Toolkit Server Core の一部です。 |
| GD88-7458 | オブジェクト指向プログラミングの紹介ページ | Encina で使用可能な各種のオブジェクト指向 (C++ および Java) のアプリケーション・プログラミング・インターフェースを紹介しています。また、Encina オブジェクト指向インターフェースに従うプログラミングに関連した、高レベルの事項についても説明しています。 |
| GD88-7459 | *DE-Light Java* プログラミングのページ | DCE Encina Lightweight Client (DE-Light) で使用可能な Java 言語ベース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ついて説明しています。 |
| GD88-7460 | *Encina++* プログラミングのページ | C++ 言語ベースの Encina C++ (Encina++) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7461 | 各種オブジェクト指向プログラミングのページ | Encina に組み込まれているデータ定義言語 (**ddl**) コンパイラーと各種 C++ および Java クラスについて説明しています。 |
| GD88-7462 | *OCCS* プログラミングのページ | Encina で使用可能な C++ 言語ベースのオブジェクト並行性制御サービス (OCCS) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7463 | *OTS* 管理プログラミングのページ | Encina で使用可能なオブジェクト・トランザクション・サービス (OTS) アプリケーション・プログラミング・インターフェースにおける C++ 言語ベースの管理クラスおよび関数について説明しています。 |
| GD88-7464 | *OTS C++* プログラミングのページ | Encina で使用可能なオブジェクト・トランザクション・サービス (OTS) アプリケーション・プログラミング・インターフェースにおける C++ 言語ベースのクラスおよび関数について説明しています。 |

*表 4. WebSphere Application Server のライブラリー (続き)*

| 資料番号 | 資料名 | 資料の説明 |
|---|---|---|
| GD88-7465 | *OTS Java プログラミングのページ* | Encina で使用可能なオブジェクト・トランザクション・サービス (OTS) アプリケーション・プログラミング・インターフェースにおける Java ベースのクラスおよび関数について説明しています。 |
| GD88-7466 | *OTS 同期クラス・プログラミングのページ* | Encina で使用可能なオブジェクト・トランザクション・サービス (OTS) アプリケーション・プログラミング・インターフェースにおける C++ 言語ベースの同期クラスおよび関数について説明しています。 |
| GD88-7467 | *RQS++ プログラミングのページ* | Encina で使用可能な C++ 言語ベースの回復可能キューイング・サービス (RQS) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7468 | *SFS++ プログラミングのページ* | Encina で使用可能な C++ 言語ベースの構造化ファイル・サーバー (SFS) アプリケーション・プログラミング・インターフェースについて説明しています。 |
| GD88-7469 | *Transactional-C++ プログラミングのページ* | Encina で使用可能な Transactional-C++ (Tran-C++) アプリケーション・プログラミング・インターフェースにおける C++ 言語ベースのクラス、関数、および構造体について説明しています。このインターフェースは、C++ プログラム言語をトランザクション向けに拡張するものです。 |
| GD88-7470 | *COBOL プログラミングのページ* | Encina COBOL プログラミング・インターフェースに組み込まれている呼び出し (関数) について説明しています。 |

# 特記事項

本書はアメリカ合衆国で提供されている製品およびサービス用に作成されたものであり、本書に記載の製品、サービス、またはフィーチャー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、およびフィーチャーについては、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、機能的に同等な製品、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 製以外の製品と組み合わせた場合、その操作の評価と検証については、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。実施権の許諾については、下記の宛先に、書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP事業所
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。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適確な表現や誤植など) 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。

**Component Broker については、**
IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

**TXSeries については、**
IBM Corporation
ATTN: Software Licensing
11 Stanwix Street
Pittsburgh, PA 15222
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。また、IBM 以外の製品に関するパフォーマンスの正確性、互換性、またはその他の要求は確証できません。 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。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は現れない場合があります。

## 商標およびサービス・マーク

以下の用語は米国 IBM Corporation の商標または登録商標です。

Advanced Peer-to-Peer Networking
AFS
AIX
APPN
AS/400
CICS
CICS OS/2
CICS/400
CICS/6000
CICS/ESA
CICS/MVS
CICS/VSE
CICSPlex
DB2
DCE Encina Lightweight Client
DFS
Encina
IBM
IBM System Application Architecture
IMS
IMS/ESA
Language Environment
MQSeries
MVS/ESA
NetView
Open Class
OS/2
OS/390
OS/400
Parallel Sysplex
PowerPC
RACF
RAMAO
RMF
RISC System/6000
RS/6000
S/390
SecureWay
TeamConnection
Transarc
TXSeries
VSE/ESA
VTAM
VisualAge
WebSphere

Domino、Lotus、および LotusScript は、 Lotus Development 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。

Tivoli は、Tivoli Systems, Inc. の米国およびその他の国における登録商標です。

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。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

本書の一部は、下記著作権を保持する Object Management Group の資料を元にしています。

Copyright 1995, 1996 AT&T/NCR
Copyright 1995, 1996 BNR Europe Ltd.
Copyright 1991, 1992, 1995, 1996 by Digital Equipment Corporation
Copyright 1996 Gradient Technologies, Inc.
Copyright 1995, 1996 Groupe Bull
Copyright 1995, 1996 Expersoft Corporation
Copyright 1996 FUJITSU LIMITED
Copyright 1996 Genesis Development Corporation
Copyright 1989, 1990, 1991, 1992, 1995, 1996 by Hewlett-Packard Company
Copyright 1991, 1992, 1995, 1996 by HyperDesk Corporation
Copyright 1995, 1996 IBM Corporation
Copyright 1995, 1996 ICL, plc
Copyright 1995, 1996 Ing. C. Olivetti &C.Sp
Copyright 1997 International Computers Limited
Copyright 1995, 1996 IONA Technologies, Ltd.
Copyright 1995, 1996 Itasca Systems, Inc.
Copyright 1991, 1992, 1995, 1996 by NCR Corporation
Copyright 1997 Netscape Communications Corporation
Copyright 1997 Northern Telecom Limited
Copyright 1995, 1996 Novell USG
Copyright 1995, 1996 02 Technolgies
Copyright 1991, 1992, 1995, 1996 by Object Design, Inc.
Copyright 1991, 1992, 1995, 1996 Object Management Group, Inc.
Copyright 1995, 1996 Objectivity, Inc.
Copyright 1995, 1996 Oracle Corporation
Copyright 1995, 1996 Persistence Software
Copyright 1995, 1996 Servio, Corp.
Copyright 1996 Siemens Nixdorf Informationssysteme AG
Copyright 1991, 1992, 1995, 1996 by Sun Microsystems, Inc.
Copyright 1995, 1996 SunSoft, Inc.
Copyright 1996 Sybase, Inc.
Copyright 1996 Taligent, Inc.
Copyright 1995, 1996 Tandem Computers, Inc.
Copyright 1995, 1996 Teknekron Software Systems, Inc.
Copyright 1995, 1996 Tivoli Systems, Inc.
Copyright 1995, 1996 Transarc Corporation
Copyright 1995, 1996 Versant Object Technology Corporation

# 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]

アプリケーション・アダプター 69
アプリケーション・モデル 48
アプレット 30
イベント・サービス 70
インターフェース 19, 21
永続オブジェクト 37
エンティティー bean 29, 48
  BMP を使う 29
  CMP を使う 29
オブジェクト 19
  クラス 21
  継承 21
  構成 20
  コラボレーション 20
  ビジネス 20
  分散 33
  ポリモアフィズム 22
オブジェクト・ビルダー 67, 75

## [カ行]

外部化サービス 70
環境プロパティー 29
管理下のオブジェクト 37
キー・オブジェクト 38
許可 17
クラス 21
継承 21
構成 20
コピー・ヘルパー・オブジェクト 38
コマンド・パッケージ 49
コンテナー
  管理パーシスタンス 29
コンテナー *(続き)*
  Enterprise beans の 28
コンテナー管理パーシスタンス (CMP) 29
コンポーネント 23
コンポーネント・アーキテクチャー 23
コンポーネント・テクノロジー 19

## [サ行]

サーブレット 31, 46
作業負荷管理 98
  Application Server アドバンスド版 55
  Component Broker 73
識別サービス 70
シック・クライアント 94
実装 21
シンナー・クライアント 95
シン・クライアント 94
セキュリティー 16
  Application Server アドバンスド版 54
  Application Server エンタープライズ版 59
  Component Broker 71
セッション bean 28, 48
セル・ディレクトリー・サービス 59
属性 19

## [タ行]

通知サービス 70
データ・オブジェクト 37
統合サーバー 98
ドキュメンテーション 9
トポロジー
  クライアント 93
  サーバー 96
トポロジー *(続き)*
  Application Server アドバンスド版 99
  Application Server スタンダード版 99
  DMZ (Application Server アドバンスド版) 100
  TXSeries 101
トランザクション 15
  分散 16
  Application Server アドバンスド版 54
  Component Broker 72

## [ナ行]

認証 17

## [ハ行]

パーシスタンス 28
  コンテナー管理 29
  bean 管理 29
配置記述子 29
ビジネス・オブジェクト 20, 36, 65, 66, 67
ファット・クライアント 93
複製サーバー 98
プラットフォーム
  Component Broker 76
  TXSeries CICS 79
  TXSeries Encina 85
  WebSphere Application Server 3
プログラミング・モデル・エクステンション 49, 67
  コマンド・パッケージ 49
  分散例外パッケージ 49
分散オブジェクト 33
分散コンピューティング 13
分散サーバー 96
分散タイム・サービス (DTS) 60
分散トランザクション 16

分散例外パッケージ 49
文書 107
並行性制御サービス 70
ホーム
  管理下のオブジェクトの 38
  Enterprise beans の 28
ポリモアフィズム 22

## [マ行]

命名
  Application Server アドバンスド版 53
  Component Broker 71

## [ラ行]

ライフサイクル・サービス 71
リモート呼び出し 35
リレーショナル・データベース 49

## [数字]

3 層アーキテクチャー 13

## A

ACID プロパティー 15
ActiveX 27
Apache Web Server 47, 62
Application Server アドバンスド版 2, 45, 46
  アプリケーション設計 48
  アプリケーション・モデル 48
  オンライン文書 107
  単純な構成 99
  DMZ 構成 100
Application Server エンタープライズ版 2, 57
  オンライン文書 107
Application Server スタンダード版 2, 45
  オンライン文書 107
  トポロジー 99

## B

bean 管理パーシスタンス (BMP) 29
beans 26

## C

Caching Proxy 5
CICS 57
  管理 84
  サポートされるプラットフォーム 79
  システム間通信 82
  単純な構成 101
  トランザクション 80
  ユーザー出口 81
  領域 80
  リレーショナル・データベース・サポート 82
  API 81
  CICS 提供のトランザクション 80
  DCE セル内での単純な構成 102
  SNA サポート 83
  Transaction Gateway 84
COM 35, 61, 68
Component Broker 57, 65
  アーキテクチャー 68
  アプリケーション・アダプター 69
  イベント・サービス 70
  永続オブジェクト 37
  オブジェクト・サービス 69
  オブジェクト・ビルダー 67, 75
  開発ツール 75
  外部化サービス 70
  管理下のオブジェクト 37
  キー・オブジェクト 38
  キャッシュ・サービス 72
  コピー・ヘルパー・オブジェクト 38
  作業負荷管理 73
  作業負荷管理構成 105
  識別サービス 70
  システム管理 73
  照会サービス 72
  セキュリティー・サービス 71
  セッション・サービス 72
  単純な構成 105
  通知サービス 70
  データ・オブジェクト 37
Component Broker 57, 65 *(続き)*
  トランザクション・サービス 72
  ビジネス・オブジェクト 36
  並行性制御サービス 70
  命名サービス 71
  ライフサイクル・サービス 71
  リソース 66
  Enterprise beans 67
  Managed Object Framework 36
CORBA 33, 60, 65
  オブジェクト・サービス 35
  IDL 36
CPI-C 87
CPI-RR 87
C++ 62, 65, 75

## D

DB2 62
DCE 58
DE-Light ゲートウェイ 88

## E

EJB サーバー 46, 48, 67
EJB 仕様 27
Encina 57, 85
  サーバー・ウィザード 91
  サポートされるプラットフォーム 85
  ツールキット 89
  トレース・ツール 91
  モニター 86
  モニター・セル構成 103
  COM ウィザード 91
  DE-Light ゲートウェイ 88
  PPC 87
  RQS 86
  SFS 87
  SNA サポート 87
Encina++ 89
Enterprise beans 27, 48, 65, 67
  コンテナーへの配置 29
  Application Server アドバンスド版の 49
e-business 1, 2

## H

HTML 30

## I

IBM HTTP Server 47, 62
IDL 34
IDL (CORBA) 36
IIOP 33
InfoCenter 10

## J

J2EE 24
JAR ファイル 30
Java 30, 62, 65, 67
Java プログラム言語 26
JavaBeans コンポーネント 26
　および ActiveX 27
JavaScript 32
JavaServer Pages 31, 33
JFC 30
JNDI 53
JSP ファイル 33, 46
JSP ページ
　Application Server アドバンスド版の 48
JSSI 31

## L

LUW 80

## M

Managed Object Framework 36
MQSeries 62

## N

Network Dispatcher 5

## O

ORB 33, 60, 66

## R

RMI 30
RPC 59
RQS 86
RQS++ 90

## S

SFS 87
SFS++ 90
SNA 83, 87

## T

Tran-C 89
TXSeries 57, 79
　構成の例 101
　CICS 79
　Encina 85

## V

VisualAge C++ 62, 75
VisualAge for Java 8, 55, 62, 67, 75

## W

Web サーバー 47
WebSphere Administrative Console 46, 50, 51
WebSphere Application Server 2, 39
　および J2EE 24
　サポートされるプラットフォーム 3
　ドキュメンテーション 9
　トランザクション 15
　分散コンピューティング 13
　文書 107
　3 層アーキテクチャー 13
WebSphere Edge Server 4
WebSphere Studio 6, 55
WebSphere ファミリー 2
WebSphere プログラミング・モデル・エクステンション 49, 67
　コマンド・パッケージ 49
　分散例外パッケージ 49

## X

XML 46, 52
XML 文書構成サービス 52

部品番号: CT6PYJA

Printed in Japan

(1P) P/N: CT6PYJA

SC88-8798-00

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:

WebSphere

# WebSphere Application Server 概説

バージョン *4.0*

SC88-8798-00