

Nextra  
はじめにお読みください

---

Version 6  
1<sup>st</sup> Edition



# 目次

---

<b>第 1 章 はじめに</b> .....	<b>2</b>
オープン分散環境 .....	2
2 層構造(クライアント/サーバ)アーキテクチャ .....	3
3 層構造アーキテクチャ .....	4
Nextra ユーティリティ.....	5
次のステップ .....	8
<b>第 2 章 インストールの方法</b> .....	<b>9</b>
インストールの方法 .....	9
<b>第 3 章 最後に</b> .....	<b>10</b>
お問い合わせ.....	11

# 第 1 章 はじめに

---

3 層ベースの分散アプリケーション・サーバ「Nextra」の世界へようこそ。本書の対象読者は、この新しいアプリケーション環境にこれまで馴染みのなかった方です。

Nextra 開発環境に既に精通しておられる方は、本書の対象ではありません。適切な情報を得るには、『サーバ開発者ガイド』に進んでください。

## オープン分散環境

---

### オープン分散環境

---

Nextra 製品ファミリーは、一連のソフトウェアユーティリティから構成されます。このユーティリティを使用すると、「読者」は、「オープン」な「分散」クライアント／サーバ型アプリケーションを作成できるようになります。先に進む前に、上記で「 」で囲んで示した用語をここで定義します。

### 読者

---

「読者」とは、もちろん、本書を読む人を指しています。アプリケーション開発者、内容を理解する必要のある全ての人が対象読者になります。

### 分散

---

「分散」は、アプリケーションの機能単位への分割を意味します。分散アプリケーションとは、機能単位が別の論理スペース、ほとんどの場合は別のマシン上で実行されるアプリケーションのことです。たとえば、データをメインフレームから取得し、ワークステーションで処理し、そして PC で表示するなどです。この場合、各マシンで実行されるプログラムは全て 1 つの分散アプリケーションの一部品と考えられます。

## クライアント／サーバ

---

クライアントはプログラムであり、サーバも別のプログラムです。クライアントは、サーバに特定の処理を実行するように要求します。サーバにその機能があり、クライアントが適正なアクセス許可を持っているならば、サーバはその処理を実行し、結果をクライアントに返します。クライアント／サーバは、アプリケーションのアーキテクチャを意味する場合があります。詳細については、「[2層構造\(クライアント／サーバ\)アーキテクチャ](#)」を参照してください。大きく見れば、クライアント／サーバコンピューティングには、要求するプログラムと要求を処理するプログラムの双方の利点をいかしたあらゆるアプリケーションが含まれると言えます。

## オープン

---

「オープン」は、コンピュータ業界の専門用語です。オープンコンソーシアム、オープンプロダクト、オープンシステムというように使われます。一般的に、「オープン」とは、特定のベンダーにロックイン (Lock -in) されていない状態を指します。逆に、アプリケーションを実行するのに、ハードウェアとソフトウェアの特定の組合せに依存する状態はロックインされていることとなります。Nextra が、ベンダーロックインの状態からユーザを開放するという意味で、「オープン」という言葉を使用します。オープン分散環境内では、使用中のプログラムの変更をせずに、そのまま別のワークステーションや PC にコピーして使用することが可能です。Nextra を使用すると、手持ちのハードウェアとソフトウェアを使用して、オープン環境を実現することができます。この定義を頭に入れて、もう一度前述の文を見てみましょう。Nextra ファミリーは、一連のソフトウェアユーティリティから構成されます。このユーティリティを使用することにより、読者は「オープン分散アプリケーションシステム」を作成することができます。特定のベンダーにロックインされることなく、最新のコンピューティングテクノロジーによるアプリケーションを作成する手助けとなるように、Nextra は設計されています。

## 2層構造(クライアント／サーバ)アーキテクチャ

---

第1に、クライアント／サーバコンピューティングは、情報システム部門が直面する問題の多くを解決してくれるように思われます。さまざまな機能がありますが、その中でも、安価なデスクトップマシンと管理しやすい LAN をビジネスで使用できるようになり、戦略的コンピューティングのために、実行できるオプションをダウンサイジングできるようになっています。ソフトウェアベンダーは、この新しいアーキテクチャを利用したアプリケーションをリリースし始めました。一般的に、これらのアプリケーションは、次の2つの分離した階層から構成されます。

- クライアント(ユーザインタフェースと大部分のロジックを処理する。)
- サーバ(データを提供し、要求に応じてクライアントに対して処理する。)

はじめにお読みください

クライアントは、サーバに接続してデータへアクセスします。サーバは、要求された情報を返します。クライアントは、情報を操作してデータを表示します。複数のクライアントが単一のサーバにアクセスできます。サーバは、ネットワーク上のどのマシンにも置くことができます。

しかし、企業が2層構造クライアント／サーバアプリケーションを設計し始めるにつれ、2層構造アーキテクチャは限界に達しました。

この構造では、ほとんどのクライアントが、その対応するサーバにしかアクセスできません。企業は、メインフレームアーキテクチャによるハードウェアロックインから解放された一方で、別の種類のロックイン、つまりソフトウェアのロックインに陥ったのです。

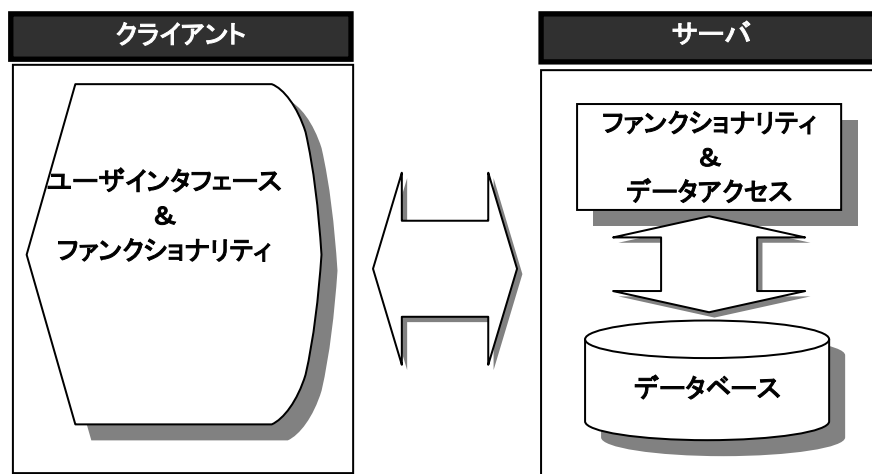


図 1.1: 2層構造アプリケーション

ソフトウェアロックインだけでなく、モジュール性の低さによりアプリケーションの柔軟性が制限されています。コードの再利用は不可能ですし、簡単に修正することもできません。そのような欠点があるにもかかわらず、2層構造クライアント／サーバアプリケーションが、今日の統合された戦略的アプリケーションにとって実行可能な解決策であると考えられる方もいますが、我々は新たな解決策を模索しました。

### 3層構造アーキテクチャ

そこで、2つの層それぞれからビジネスロジックを分離し、ファンクショナリティ層という新しい中間層を作ることによって、クライアント／サーバモデルを飛躍的に発展させました。古いクライアントは完全にプレゼンテーション用ツールになり、古いサーバはDB管理システムやメインフレームのようなデータソースになりました。新しい中間層には、アプリケーションが実行しなければならないビジネスロジックを配置します。たとえば、データ検索、数理計算、またはトランザクション処理などです。

この3層構造アーキテクチャが2層構造より優れているのは何故でしょうか？3層構造アーキテクチャは、2層構造の限界を超越するだけでなく、将来起こり得る未知の問題を解決する可能性を持つアーキテクチャを提供するからです。

キーは中間層にあります。中間層によって、所有する製品を相互運用できます。さまざまな種類のユーザインタフェースを使用して、さまざまな所有 DB とあらゆる種類のリモートシステムにアクセスすることができます。ハードウェアロックインもソフトウェアロックインもありません。このような広範な選択肢というのは、コンピュータ業界では先例のないことであり、情報システム設計の基盤となるパラダイムに質的変化が起こったことを意味します。3層構造分散クライアント/サーバアーキテクチャで構築されたアプリケーションは、モジュール性があり修正が容易です。さまざまな言語を使用し、クライアントやサーバプログラムの追加を分散アプリケーション実行中に行うことができます。さらに、新しいコンポーネントを置く際に、アプリケーションの他の部分を修正する必要もありません。

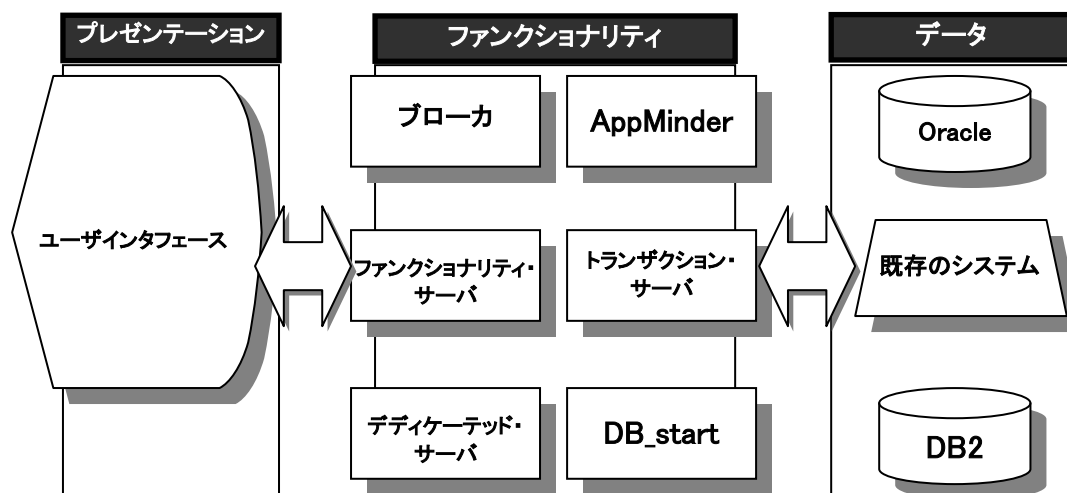


図 1.2: 3層構造アプリケーション

3層構造クライアント/サーバモデルは、Nextra オープン分散環境の基盤となります。Nextra アプリケーションは、堅牢性、拡張性、高パフォーマンスを提供します。

Nextra ユーティリティを使用すると、アプリケーションの構築と修正が容易になります。また、必要に応じて、既存のプラットフォームや新規のテクノロジーを統合することができます。

## Nextra ユーティリティ

Nextra ユーティリティは、既存のハードウェア、ソフトウェアおよびネットワーク上で実行される分散アプリケーションの開発を容易にします。

はじめにお読みください

ここでは、開発プロセスを容易にする一連のユーティリティをご紹介します。開発パッケージに同梱されているユーティリティに必要なファイルを用意すれば、すぐに分散アプリケーションを作成することができます。

## RPCMake

---

分散された各層(プレゼンテーション層、ファンクショナルリティ層、DB 層)プロセス間の通信インフラ(スタブ・スケルトン)を自動生成します。開発者は、通信部分に関わる作業から解放されます。IDL (Interface Definition Language) ファイルをインプットとし、クライアントプログラム用のスタブ (Stub)、サーバプログラム用のスタブ (別名:スケルトン:Skelton)を自動生成します。

## SQLMake

---

Nextra が提供する DB アクセス・サーバ (DB\_start) を呼出すクライアントプログラム用スタブの自動生成を行います。DB\_start は、SQLMake で使用された SQL ファイルをインプットとし、SQLMake で生成されたクライアント・スタブを持つクライアントプログラムと通信を行うことができます。

## TPMake

---

トランザクションの整合性を侵すことなく、さまざまな RDBMS 下で実行される複数 DB にアクセスできるトランザクション・サーバプログラムの開発をサポートします。開発者は、TPMake で自動生成されたコードを使用してトランザクション制御のロジックを記述します。TPMake に関しては、SQL ファイルだけでなく、リソースファイルも作成しなければなりません。リソースファイルは、各 SQL ファイルを適切な DB にマップするテキストファイルです。

## RPCDebug

---

サーバプログラム単体テストツールです。RPCDebug を使用すると、特定サーバの全ての RPC をテストすることができます。よって、クライアントプログラムの作成を待たずに、サーバプログラム内の RPC 関数のテストを行えます。

## RPC Developer

---

RPC Developer は、前述の全てのユーティリティを統合的に取り扱う GUI ツールです。  
RPC Developer は、コマンド”rpcdev”を使用します。



図 1.3: RPC Developer の GUI 画面

## Broker

---

Nextra におけるネーミングサーバの役割を果たします。サーバプロセスの位置情報を把握します。クライアントは、ネットワーク上に分散されたサーバの物理的な位置情報をブローカより取得します。マスタブローカ、サブブローカというようにブローカの階層を形成することも可能です。

## DB\_start

---

DB アクセス・サーバとも呼ばれ、バイナリ実行モジュールとして提供されます。開発者は煩わしい DB 操作言語を記述することなく、SQL ステートメントを記述するだけで、クライアントからの DB 操作が可能になります。Oracle、DB2、SQL Server、HiRDB がサポートされています。

はじめにお読みください

## AppMinder

---

ネットワーク上に分散されたサーバプロセスを定期的に応答管理し、稼動状況を監視します。同時に、自動立ち上げやサーバプロセスダウン時には自動起動を行います。

## 次のステップ

---

『サーバ開発者ガイド』にお進みください。

## 第 2 章 インストールの方法

---

### インストールの方法

---

製品添付の Install.txt を参照してください。

## 第 3 章 最後に

この章では、Nextra のドキュメントで使用される用語の定義を説明します。

表 3.1: 用語集

用語	定義
Application (アプリケーション)	各分散アプリケーションは、正しく定義されたリモートインタフェースを通じて通信するクライアントとサーバの組で構成されます。
Cell (セル)	ブローカを中心に構成されたサーバプロセスの集まりを、セルと呼びます。1 つのセルには必ずマスタブローカが存在し、サブブローカやサーバプロセスがそのマスタブローカに登録します。
Clustering (クラスタリング)	アプリケーション・クラスタリングとも言います。2 つ以上のセル構成を持つものをクラスタリングと言います。
Client (クライアント)	ユーザに代わって要求を行う分散アプリケーションの構成要素です。各クライアントは 1 つ以上のサービスを使用します。
DB Access Server (DB アクセス・サーバ)	「DB_start」を参照してください。
DB Access Modules (DB アクセス・モジュール)	Nextra が提供する、DB にアクセスするモジュール群。DB アクセス・サーバとトランザクション・サーバがあります。
Dedicated Server (デディケイテッド・サーバ)	通常のサーバと異なり、デディケイテッド・サーバは各クライアントからのアクセス毎に子プロセスを作成し、1 クライアントに 1 子プロセスを割り当てます。これによって、各クライアントは子サーバプロセスを専有できます。
Functionality Server (ファンクショナルリテイ・サーバ)	3 層アーキテクチャの中間層に位置し、基本的に DB 層にアクセスしないサーバの一般名称。
IDL (インタフェース定義言語)	Interface Definition Language
RPC (リモート・プロシージャ・コール)	異なるプログラムでプロシージャを起動し、OS やハードウェアの違いによらず、データを交換する関数呼び出し。呼び出しシンタックスは、ローカル関数呼び出しと同じです。
Server (サーバ)	クライアントの要求に応答し、共有リソースへのアクセスを提供する分散アプリケーションの構成要素です。
Service (サービス)	分散アプリケーション中の関数に論理的に関連するセットをインプリメントする 1 つ以上のインタフェースの集まり。各サーバは 1 つのサービスを提供します。

Skeleton (スケルトン)	サーバプログラムの一部で、データマーシャリングと通信を実行します。RPCMake によって自動生成されます。以前は、スケルトンをサーバ・スタブと表現していました。
Stub (スタブ)	クライアント (またはサーバプログラム) の一部で、データマーシャリングと通信を実行します。RPCMake によって自動生成されます。
Transaction Server (トランザクション・サーバ)	トランザクションの整合性を侵すことなく、さまざまな RDBMS 下で実行される複数 DB にアクセスできるサーバプログラムです。「TPMake」を参照してください。
Variable Named Server (バリエブル・ネームド・サーバ)	同じサーバプログラムの実行ファイル名 (インタフェース名) を変えて複数起動することができるサーバのことです。IDL ファイルにはインタフェース名として変数を指定し、実行時に各サーバ名を指定します。同じ構造を持つ複数のデータベースにアクセスする際に、この種のサーバが役に立ちます。個々のデータベースごとに独自のサーバをコーディングする必要がないからです。

## お問い合わせ

お問い合わせフォームが、CD 中に「QAForm.txt」として同梱されています。お問い合わせの際は、こちらをご利用ください。

はじめにお読みください

## ご注意

### 商標権に関する注意

Nextra 製品は、全て Inspire International Inc. の商標または登録商標です。その他記載のブランドおよび製品名は、該当する会社の商標または登録商標です。

### 著作権に関する注意

インスパイア インターナショナル株式会社の書面による許可なく、このマニュアルの内容の全部、もしくは一部を複写、複製、写真によるコピー、製本、翻訳、もしくは電子メディア化ないしは機械読み取りが可能な形態に変換することは固く禁じます

なお、本マニュアルの内容、連絡先などについては、弊社の都合により予告なく変更することがございます。あらかじめご了承ください。

特に記載がない限り、この製品に含まれるソフトウェアおよびドキュメントの著作権は Inspire International Inc. が所有しています。

Nextra

はじめにお読みください

---

2011年 9月 15日	v6 1 <sup>st</sup> Edition
2008年 10月 15日	v5 2 <sup>nd</sup> Edition
2007年 4月 18日	第3版発行
2006年 11月 21日	インストール後の環境設定への追加
2004年 9月 27日	インストーラの記述についての修正
2004年 7月 19日	第2版発行
2003年 4月 18日	初版発行

著者: Inspire International Inc.

---

Copyright © 1998–2011 Inspire International Inc.  
Printed in Japan