J 2 EE™ ブループリンツにおける 多層アプリケーションの設計ガイドライン

Design Guideline of J 2 EE™ Blueprints for Multi-Tier Applications

樫 井 一 臣

要約 J 2 EE™ は，多層アプリケーションのために豊富な API を提供しているが，大規模なシステムを短期間で開発するには，適切な設計のガイドラインが必要になってきている．ここでは，J 2 EE ブループリンツが推奨しているいくつかのアーキテクチャやデザインパターンなどについて考察し，その有効性を検証する．

Abstract J 2 EE™ offers rich API's for multi-tier applications. However, adequate design guideline for developing large scale business application systems are needed.

This paper reviews some architecture and design patterns recommended in J 2 EE Blueprints, and examine its usefulness.

1. は じ め に

Java が登場してから 5 年が経過し，インターネットの普及とともにネットワーク・ベースのシステムを開発するにあたってのプログラミング言語としては，標準的な地位を確立したと言っても過言ではないだろう．

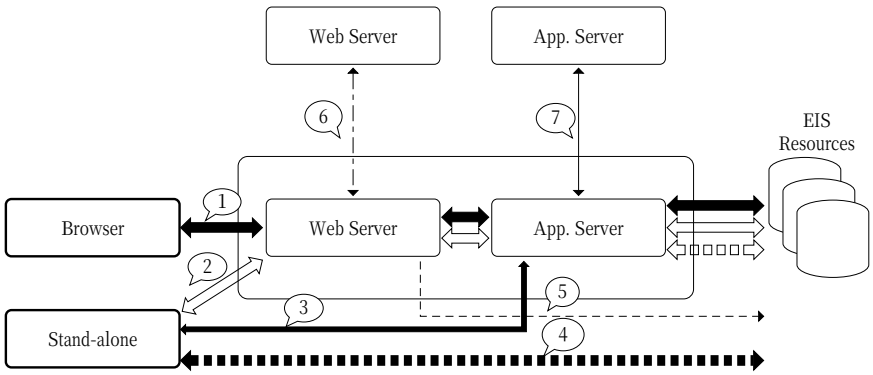

図 1 多層アプリケーションシステム

最近では図 1 のように多層の構成をとる大規模なシステムも多くなってきた．本格的な E コマースのサイトなどはほとんどこのような構成をとっている．特にウェブ・サーバやアプリケーション・サーバのコンポーネントは，Java で実装するのが主流となっている．

JCP（Java Community Process，開発者コミュニティによる Java API 仕様を定義するためのオープンなプロセス）では，これらの API セットを J 2 EE（Java 2 プラ

ットフォーム，エンタープライズ・エディション）として標準化している．

J 2 EE は，J 2 SE（Java 2 プラットフォーム，スタンダード・エディション）をベースにサーバ・コンポーネントのための API や他のシステムと連係するための数多くの API からなる(図 2)． これはアプリケーションの開発者にとっては便利な一方，すべての API を理解するには時間がかり，大規模なシステムにおいて適材適所で J 2 EE のテクノロジを利用するにはかなりの熟練が要求されることになる．また，あるひとつの機能を実装するやり方が複数あることもあり，パフォーマンスやスケーラビリティに大きく影響することもある．

このため，J 2 EE にはブループリンツというドキュメントとサンプル・アプリケーションからなる設計のためのガイドラインがある．

ここでは J 2 EE ブループリンツで推奨しているいくつかのアーキテクチャとデザインパターンを紹介し，その有効性について考察する．

2. Java 2 プラットフォーム，エンタープライズ・エディション（J 2 EE）1.2 の概要

2.1 J 2 EE の API

J 2 EE のアーキテクチャは図 2 のようになっており，J 2 SE をベースにエンタープライズ・アプリケーションを構築するにあたって必要な各種の API をサポートしている．中心となるのはビジネス・ロジックをコンポーネント化するための EJB(Enterprise JavaBeans)と，ウェブ・アプリケーションのための Servlet と JSP(Java Server Pages）である．

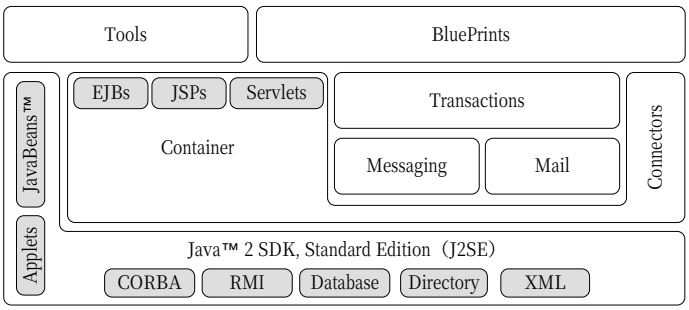

図 2　J 2 EE アーキテクチャ

2.1.1 Enterprise JavaBeans（EJB）

EJB はビジネスロジックをコンポーネント化するための API であり，図 3 のようなアーキテクチャとなっている． EJB コンポーネントは EJB コンテナの中で機能し，クライアントの要求やシステムの状況により必要に応じてコンテナから利用される．リソース管理やトランザクション管理などはコンテナが行うため，コンポーネントにはロジックを記述するだけでよい．

2.1.2 Servlet と Java Server Pages（JSP）

Servlet は，ウェブサーバにおける CGI のようにブラウザからのリクエストに対しダイナミックにレスポンス（通常は，ブラウザに表示する HTML や XML）を生成

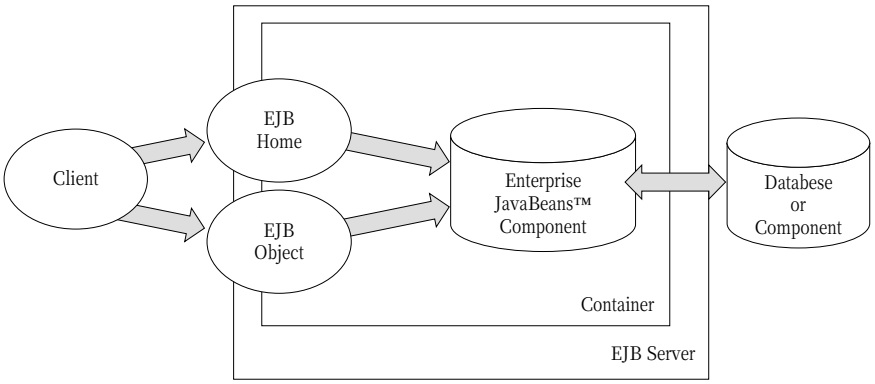

図 3　EJB アーキテクチャ

し返す．ただし，リクエストの都度に新たにリソースを消費することはないのでスケーラブルにシステムを拡張することができる．Servlet は Java 言語で処理を記述することになるので，レスポンスを生成するために print メソッドが多用されることになる．このようにプレゼンテーションとロジックが混在しているため，Web デザイナとプログラマの役割分担が困難である．このために JSP というダイナミック・コンテンツのためのテクノロジがある．JSP では，HTML や XML などの中に Java 言語によるロジックを記述を記述することが可能であり，Web デザイナが利用しやくなっている．実行時には JSP のソースコードがダイナミックに Servlet のソースコードに変換され，Java のクラスファイルにコンパイルされるため上記の Servlet の利点も受け継ぐことになる．

2.1.3　その他の API

その他に，J 2 EE にはトランザクションのための JTA（Java Transaction API）や，データベースにアクセスするための JDBC（Java Database Connectivity），メッセージング・サービスを利用するための JMS（Java Messaging Service）などの API が含まれている．

2.2　J 2 EE ブループリンツ

上記のように J 2 EE には大規模なサーバアプリケーションをサポートするために多くの API が含まれており，適切に API を使用するためには設計のガイドラインが必要であると考えられる．また，あるひとつの機能を実装するにあたっても様々なやり方が考えられる．例えば，多層アプリケーションにおいてセッション情報を保持することを考えた場合，クライアント（ウェブ・ブラウザ），ウェブサーバ，アプリケーション・サーバ，データベース・サーバのどの層に保持することも可能である．このような場合でもガイドラインを参考にすることにより，短期間で適切な設計を行うことができるだろう．

このようなニーズに応えるため，J 2 EE ではブループリンツとよばれる設計ガイドラインを提供し，開発者に便宜を図っている．ブループリンツは，「Designing Enterprise Applications with the Java (TM) 2 Platform, Enterprise Edition」という 340 ページのドキュメントと，Java Pet Store というオンライン・ショッピング・サイト

のサンプル・アプリケーションからなっている（図 4）.

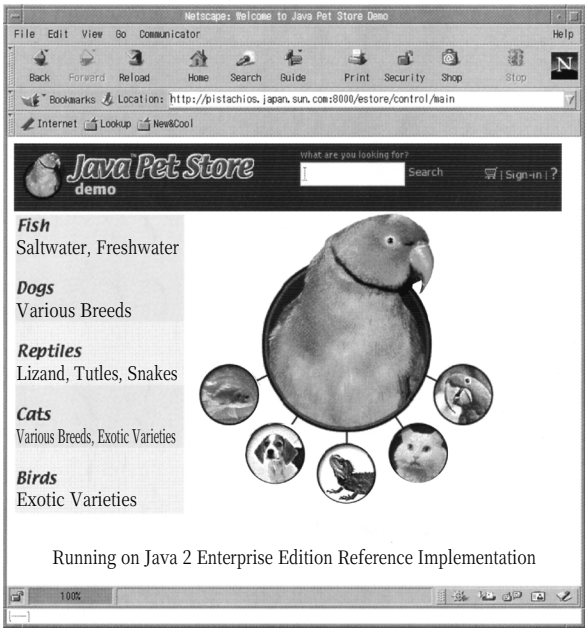

図 4　サンプルアプリケーション：Java Pet Store

2.3　多層アプリケーション

Java Pet Store は，図 5 のような多層アプリケーションとして設計されている．また，バリエーションとしては，クライアントを Java アプレットや Java スタンドアロン・アプリケーションとし，直接 EJB 層にアクセスしたり，ウェブ層から JDBC を用いて，直接 EIS 層にアクセスするといったシナリオが考えられる（図 1）．以下では，典型的な Java Pet Store のような構成をベースにして話を進める．

3.　J2EE ブループリンツにおける設計ガイドライン

ここでは，J2EE ブループリンツが推奨するアーキテクチャやデザインパターン，その実装方法のうち，最も典型的ないくつかの例を紹介しその有用性について考察する．

3.1　MVC アーキテクチャと J2EE での実装

3.1.1　MVC アーキテクチャ

MVC は GUI ツールキットなどでよく使用されるアーキテクチャであり，図 6 のような構造になっている．

モデル（Model）は，プレゼンテーションには依存しない本質的なデータとそれを扱うロジックからなる．一方，ビュー（View）はモデルの視覚的な表現を行う．コントローラ（Controller）は，主にイベントによりこれらをコントロールするオブジェクトであり，モデルのデータを変更したりビューに通知を行ったりする．これにより各オブジェクトの役割分担が明確になり，イベント・ドリブンでシンプルな構造を

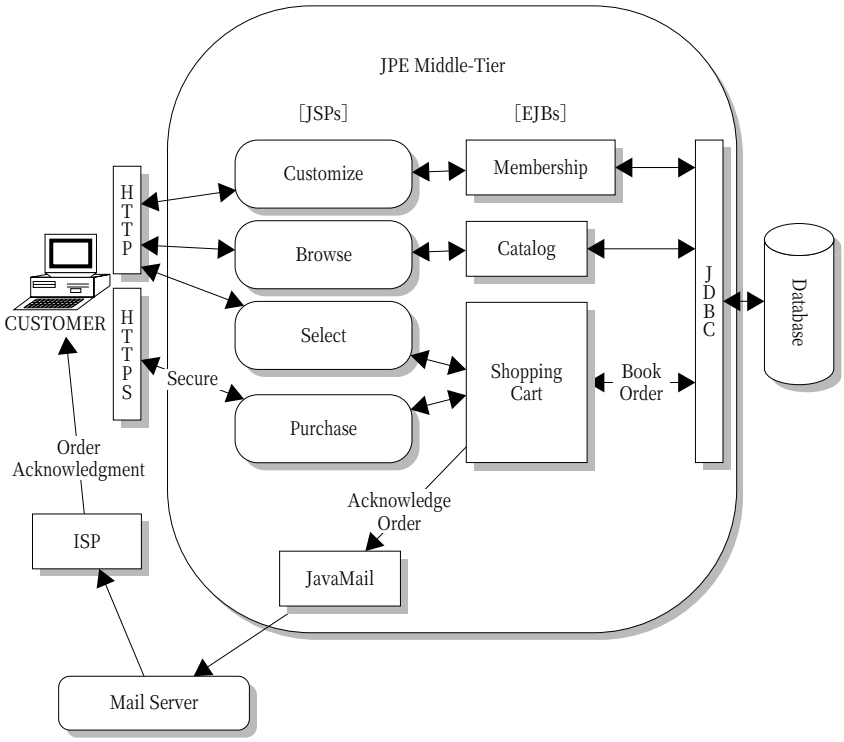

図 5　Java Pet Store の構造

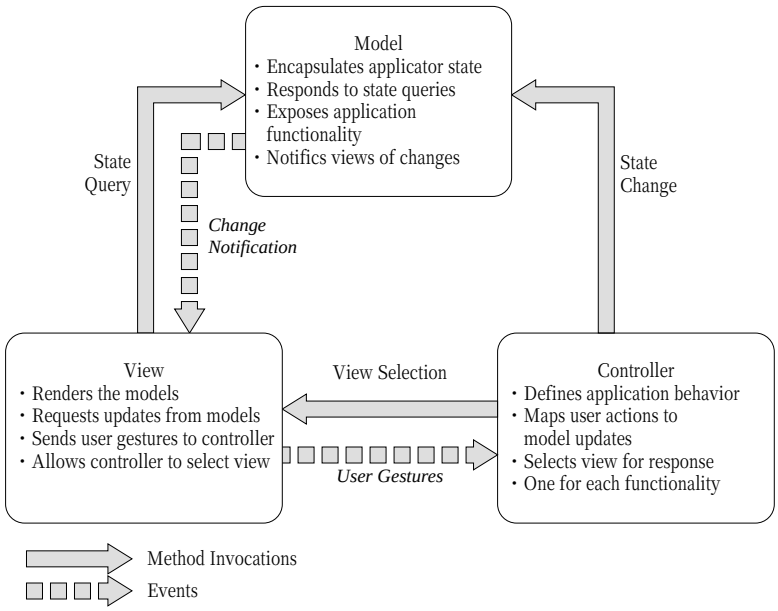

図 6　MVC アーキテクチャ

保つことができる．

3.1.2　EJB 層におけるモデルとセッション・ビーンとエンティティ・ビーンの使用法

モデルはカプセル化されたデータと，公開されたメソッドからなるため，オブジェクトとして実装するのが自然である．また，アプリケーションの性格上，永続性が必要な場合が多く，トランザクションも重要である．このため，J 2 EE ブループリンツでは，モデルは，エンタープライズ・ビーンとして EJB 層に実装することを推奨している．

また，複数のクライアントにサービスを提供することや永続性のサポートを考慮すればエンティティ・ビーンとして実装するのが最適であると考えられる．ただし，J 2 EE 1.2 における EJB 1.1 では，エンティティ・ビーンの仕様が十分ではないため，場合によってはステートレス・セッション・ビーンの使用も推奨している．例えば，ひとつのビーンがデータベースの複数のレコードを操作する場合などである．これについては，次期バージョンの EJB 2.0 で改善される見込みである．

3.1.3　ウェブ層におけるビューの実装

ビューは，ウェブ層において Servlet または JSP テクノロジを用いることを推奨している．特に，ウェブのデザインは，アプリケーションのロジックよりも頻繁に変更される可能性が高いのでロジックと分離して，HTML などデザイナが扱いやすい方がよい．このため，クライアントの GUI となるレスポンス（HTML や XML）の生成には JSP が適していると考えられる．なお，バイナリ・コンテンツを生成したり，ウェブ層におけるサービスを実装するには，Servlet を利用することができる．

3.1.4　ウェブ層と EJB 層におけるコントローラの役割

コントローラに関しては，ウェブ層と EJB 層に分かれるが，できるだけ早い時期に Servlet のリクエストをイベントとして EJB 層に伝えることを推奨している．このため，可能な限りセッション管理が可能なステートフル・セッション・ビーンにコントローラの機能をもたせるのがよい．サンプル・アプリケーションでは，すべてのリクエストを一括して受け付けるフロント・コントローラを実装してロジックを論理的に簡素化している．またレスポンスに関しては，ウェブ層のコントローラである Servlet が，表示する JSP を選択しフォワードしている．

3.2　ウェブ層におけるバリューオブジェクトによるモデル・データのキャッシング

上記においては，EJB 層におけるモデルの実装を推奨したわけであるが，欠点としては，リモートアクセスによるパフォーマンスの低下があげられる．特にモデルが多くのプロパティをもっている場合，ウェブ層から個別にプロパティにアクセスするとプロパティの数だけリモートアクセスが発生することになる．このため，バリューオブジェクトとよばれるすべてのプロパティをもつシリアライズ可能なオブジェクトを定義して，ウェブ層からは 1 回の呼び出しでバリューオブジェクトを得ることにより，大幅にパフォーマンスを向上することができる．特にリード・オンリーでアクセスする場合には，キャッシュの一貫性について問題となることは少なく，またエンタープライズ・ビーンにプロパティをセットする必要もないために有効な手段である．

逆に，モデル・データの変更が必要なときには，もしパフォーマンスが大きな問題とならない場合は複雑性が増加するキャッシングを使用せず，エンティティ・ビーン

のメソッドを直接呼び出すのがよい．

3.3 セッション情報の保存

3.3.1 クライアント層におけるセッション情報の保存

ウェブ・ブラウザにクッキーとしてセッション情報を保存することができる．Servlet からクッキーを扱う API は次のようになる．

```
HttpServletRequest req ;
Cookie cookie＝req.getCookie() ;
HttpServletResponse res ;
res.addCookie(cookie) ;
```

長所としては，セッション情報はセッションごとに各ブラウザに保存されるためサーバの資源は必要としないことである．このためクライアント数が増大しても，サーバはスケーラブルに対応できることになる．一方，クライアントからウェブサーバへのリクエストのたびにセッション情報が送られるため，データ転送量が増加し，またセキュリティ上のリスクもある．また，クッキーを扱うための API は，ローレベルであるため使いにくい．

3.3.2 ウェブ層におけるセッション情報の保存

ウェブサーバのセッション・オブジェクトに，属性としてセッション情報を保存することができる．Servlet での API は次のようになる．

```
HttpSession session ;
String key ;
Object value＝session.getAttribute(key) ;
…
session.setAttribute（key, value）;
```

これは，クッキーによるクライアント層における短所のいくつかを解決している．すなわち，クッキーを無効にしてもよい，ブラウザとのデータのやりとりが少ない，ウェブサーバ内におけるローカルなメソッド呼び出しのため，効率的である，などである．一方，ウェブサーバの資源を使用するため，クライアント数の増加に従いサーバに負荷がかかるため，スケーラビリティに欠ける．また，ローレベルで使いにくい API である点は変わらない．

3.3.3 EJB 層におけるセッション情報の保存

ステートフル・セッション・ビーンを使用して，セッション情報を保存することができる．ウェブ層からはセッション・ビーンのメソッド呼び出しで扱えるため，オブジェクト指向のアプローチが可能である．またセッション・ビーンは，EJB コンテナによりリソース管理されるため，クライアント数が増加してもスケーラブルに対応することが可能である．短所としては，リモート呼び出しになるため，ローカルな呼び出しに較べ若干のパフォーマンス低下が見込まれるが通常は問題になることはあまりないものと思われる．このため，J 2 EE ブループリンツではこの方法を推奨している．

3.3.4 データベースにおけるセッション情報の保存

クッキーには，クライアントの ID のみを保存し，他の情報は，それをプライマリ・キーとして，データベースに保存することができる．これも，EJB 層における方法と同様にスケーラブルではあるが，基本的には，JDBC によるローレベルな API を用いたり，リクエストごとにデータベース・アクセスが発生するなどの短所がある．また，データベースごとに SQL の仕様に違いがあるなど，ポーティング時に問題が発生しやすい懸念もある．

	API の使いやすさ	データ転送量 セキュリティ	スケーラビリティ	リモートアクセス の オーバーヘッド
クライアント層	× Servlet Cookie	×	○	○
ウェブ層	× Servlet HttpSession	○	× ウェブサーバが ボトルネックとなる 可能性	○
EJB 層 (*推奨)	○ SessionBean	○	○ コンテナによる リソース管理	△ ValueObject による キャッシングで解決
データベース	× JDBC	○	○	× データベースアクセスが発生

4. おわりに

MVC アーキテクチャを中心にセッションの保存方法など，具体的な手法まで検討してきた．これらの例を通して，ブループリンツの設計ガイドラインの有効性について述べてきた．

また，コーディングにおいても Java Pet Store サンプル・アプリケーションはソースコードが公開されているため，具体的なプログラミング技術が参考として使えるようになっている．

市場で競争力を得るため，大規模なシステムを短期間で開発する必要性がますます高くなってきている．E コマースのサイトを数ヵ月で立ち上げるといった事例も日常茶飯事である．このような状況の中で，J 2 EE ブループリンツの設計ガイドラインは非常に役に立つと考えられる．

J 2 EE の仕様自体は，次期バージョンの 1.3（執筆現在，最終ドラフト）において，EJB　2.0（執筆現在，最終ドラフト）などを取込みさらに改善していく予定である．また，ブループリンツも B 2 B 相互運用を視野に入れ，されに多くのシナリオが追加される見込みであるし，UML による形式的なアプローチも検討されている．今後，開発者にとって，ますます有益なガイドラインになるものと思われる．

参考文献　［1］　http : //java.sun.com/j 2 ee/blueprints/

執筆者紹介　樫　井　一　臣（Kazuomi Kashii）

1958 年生．1981 年東京工業大学理学部数学科卒業．富士通 FIP，ソニーテクトロニクスを経て，1987 年サンマイクロシステムズ(株)入社．システムエンジニアとして，ソフトウェア開発環境, 分散オブジェクトなどをサポート．1997 年から 2000 年夏までは，Java センターにて，Java テクノロジ全般，特に EJB を中心とした J 2 EE や，Jini に関するプロジェクトを中心に活動してきた．現在は，ドットコム・テクノロジ・センタにて，Java テクノロジを応用した大規模システムや，デジタル・メディア関連のプロジェクトにおいて，コンサルティングやサポートを行っている．