研究ノート

社会科学系学部における情報科学（コンピュータ）教育（その４）
―EJB を用いた情報システム構築―

藤 尾 好 則

目 次

1．はじめに

「社会科学系学部における情報科学（コンピュータ）教育（その３）」[1]では、Servlet と JSP を用いた電子商取引を行う情報システムの構築についてソフトウェアのアーキテクチャに焦点を当てて考察した。

本論文では、企業のアプリケーションのソフトウェア部品を標準化した EJB (Enterprise Java Beans) 及び企業システム構築のためアプリケーションの実装標準である J2EE (Java 2 Platform, Enterprise Edition) を用いた情報システム構築について考察する。この情報システム構築法の考え方は、標準化された基盤上にブロックを積み上げて建物を構築する工法に類似している。

目的の一つは高度なスキルを持つＳＥやプログラマが実施しなければならない情報システム構築ための設計作業を簡略化すること、他の一つはシステムアナリストやシステムの利用者（運用者、管理者、経営者）が携わる業務改革・改善等、情報システム導入の目的である本質的な業務の論理構築に時間を費やして行こうとするものである。

2． EJB を用いた電子商取引の枠組み

アプリケーションは、論文[1]と同様に電子商店での買い物を取り上げる。

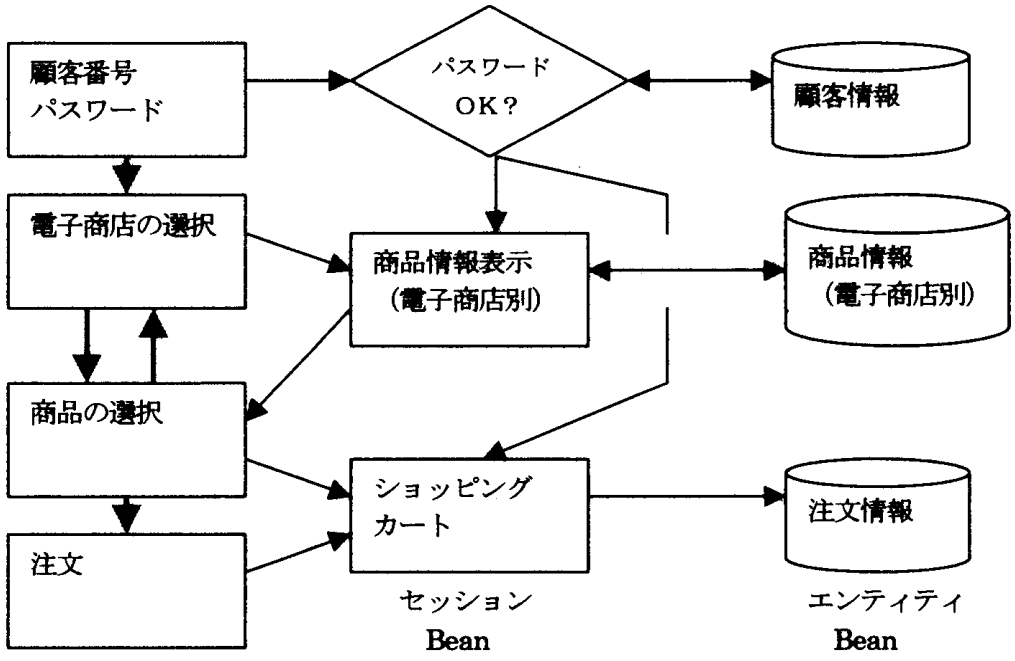

図1　買い物の概要

ヒューマンインタフェース（画面）は JSP 、ビジネスプロセス（業務手順）は EJB 、データベースは Access を用いて設計している。このため論文[1]とは業務処理フローのレベルは類似しているが、詳細なソフトウェア・アーキテクチャでは EJB を使用しているビジネスプロセスの部分が大きく異なっている。

2．1 ヒューマンインタフェース

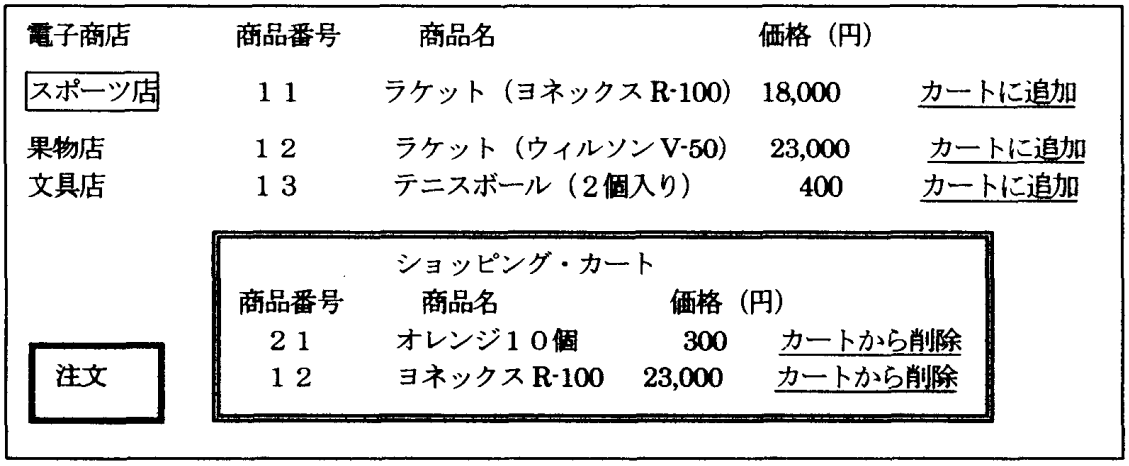

図2 買い物の画面

買い物の概要（図1）に示すように顧客がログイン画面から顧客コードとパスワードを入力する。顧客のパスワードチェックが“ＯＫ”であれば買い物の画面（図2）を表示する。この画面には電子商店の選択、選択した電子商店の商品、顧客が選択した商品を入れるショッピング・カートの表示が含まれる。顧客は表示商品を“カートに追加”をクリックして商品をカートに追加し、カートにある商品を“カートから削除”をクリックして商品を削除して元に戻すことができる。買い物が終了した時点で“注文”ボタンをクリックしてカートに入れた商品を注文する。[2]

2．2 ビジネスプロセス

クラスを抽出してEJBを用いた設計を考慮したUML記述のクラス図（図3）を作成する[2]。

このクラス図を基に画面やデータベースとの注文処理をシーケンス図（図4）に図示する。

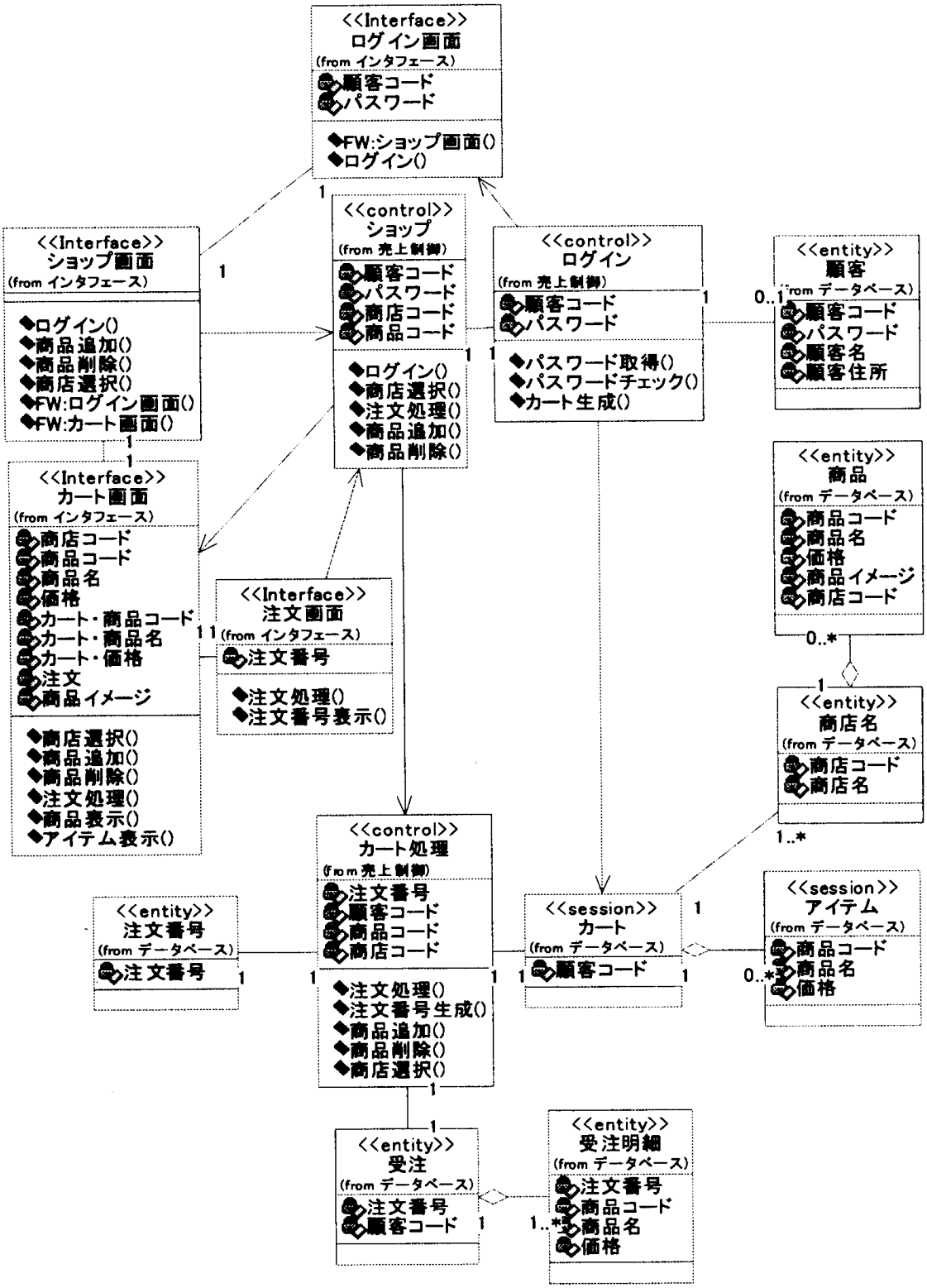
<<Interface>>
ログイン画面
(from インタフェース)
顧客コード
パスワード
◆FW:ショップ画面()
◆ログイン()
<<Interface>>
ショップ画面
(from インタフェース)
◆ログイン()
◆商品追加()
◆商品削除()
◆商店選択()
◆FW:ログイン画面()
◆FW:カート画面()
<<control>>
ショップ
(from 売上制御)
顧客コード
パスワード
商店コード
商品コード
◆ログイン()
◆商店選択()
◆注文処理()
◆商品追加()
◆商品削除()
<<control>>
ログイン
(from 売上制御)
顧客コード
パスワード
◆パスワード取得()
◆パスワードチェック()
◆カート生成()
<<entity>>
顧客
(from データベース)
顧客コード
パスワード
顧客名
顧客住所
<<Interface>>
カート画面
(from インタフェース)
商店コード
商品コード
商品名
価格
カート・商品コード
カート・商品名
カート・価格
注文
商品イメージ
◆商店選択()
◆商品追加()
◆商品削除()
◆注文処理()
◆商品表示()
◆アイテム表示()
<<Interface>>
注文画面
(from インタフェース)
注文番号
◆注文処理()
◆注文番号表示()
<<entity>>
商品
(from データベース)
商品コード
商品名
価格
商品イメージ
商店コード
<<entity>>
商店名
(from データベース)
商店コード
商店名
<<control>>
カート処理
(from 売上制御)
注文番号
顧客コード
商品コード
商店コード
◆注文処理()
◆注文番号生成()
◆商品追加()
◆商品削除()
◆商店選択()
<<entity>>
注文番号
(from データベース)
注文番号
<<session>>
カート
(from データベース)
顧客コード
<<session>>
アイテム
(from データベース)
商品コード
商品名
価格
<<entity>>
受注
(from データベース)
注文番号
顧客コード
<<entity>>
受注明細
(from データベース)
注文番号
商品コード
商品名
価格

図3　クラス図

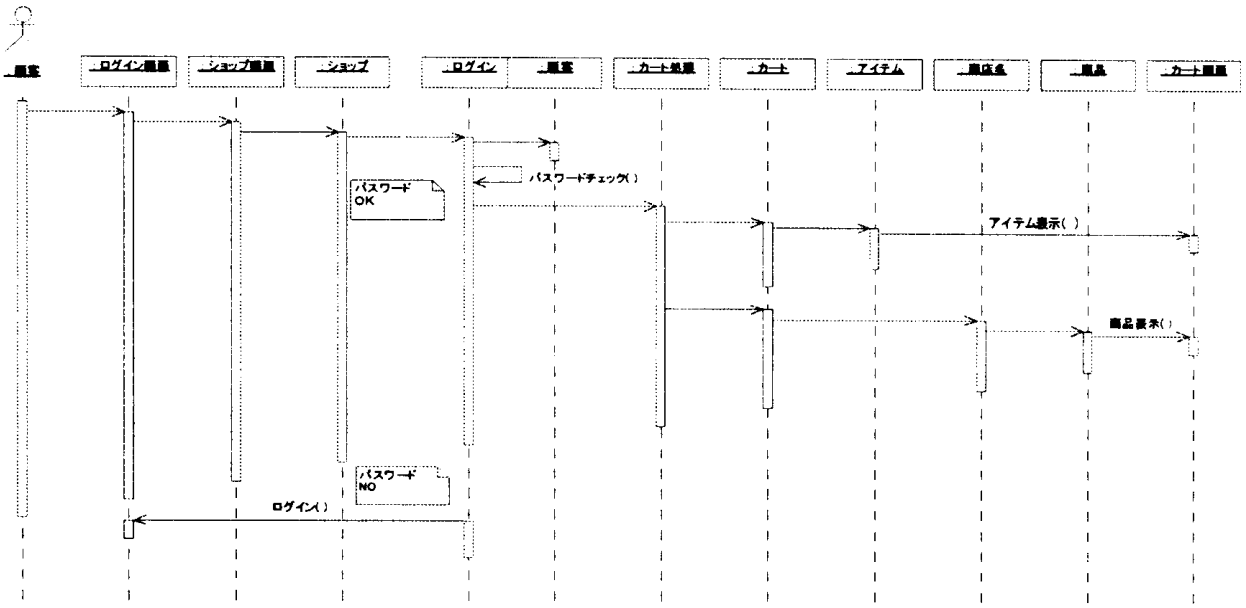

図4　シーケンス図（パスワードチェック）

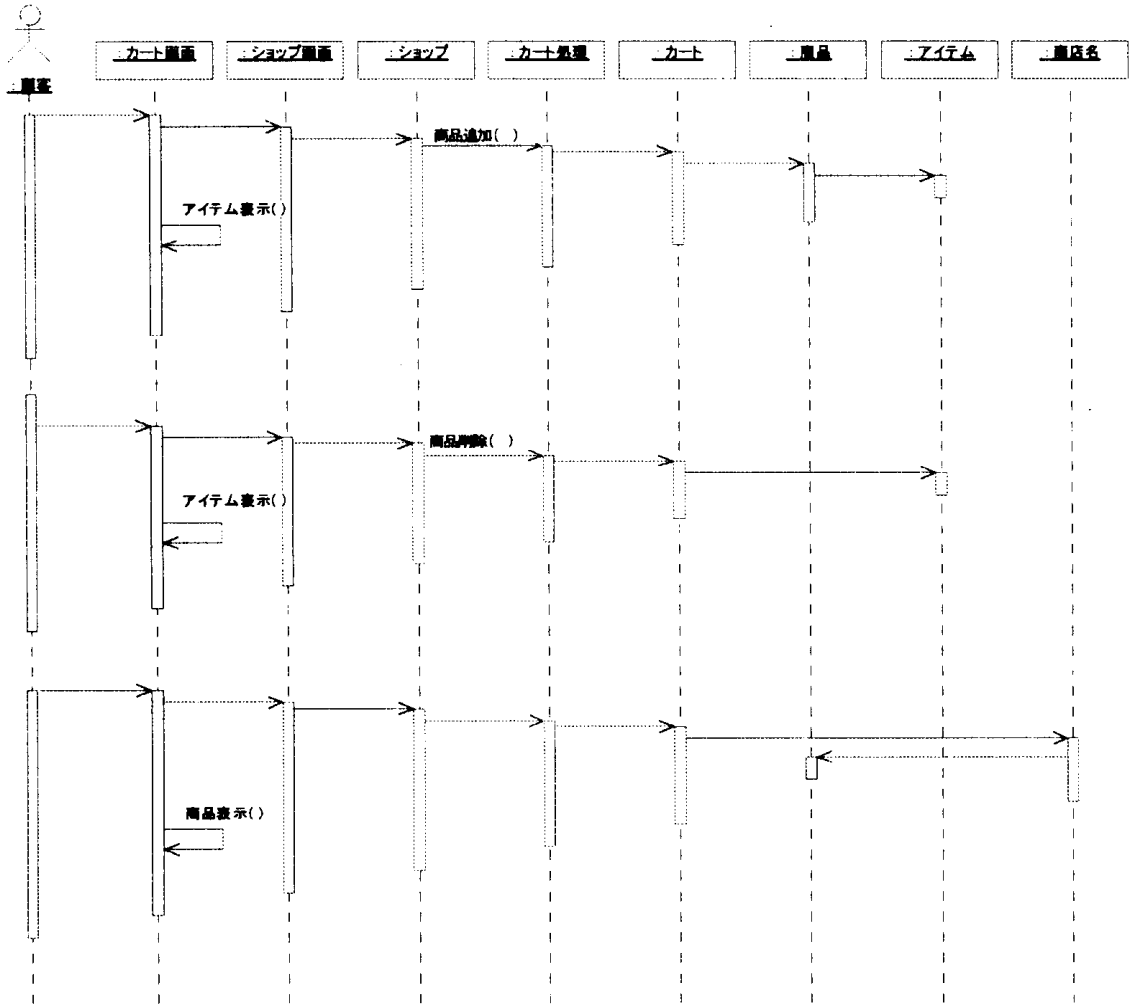

図5　シーケンス図（カートへの商品追加、カートから商品削除、電子商店選択）

2．3　データベース

注文処理に関する表のリレーション（関係）を設計する。このデータベースの設計は全てのシステム（その2～4）[1][6][7]において共通していて、同じデータベースを使用する。

3．EJBが必要な理由

2層のクライアント／サーバモデルでは各クライアントにアプリケーションのロジックがインストールされているため管理が困難である。一方、多層アプリケーションの場合は、クライアントにユーザインタフェースだけがあり、サーバの中間層にアプリケーションのロジックを配置し一括してクライアントの処理を行う。この構成ではアプリケーションのロジック変更は、この個所に限定して行うため管理が簡略になる。

信頼性が高く、安全で、管理の容易なアプリケーションの作成は難しい。しかしEJB仕様[3]に従ったコンポーネットを使って分散システムを構築すると負担を軽減できる。理由は開発作業を複数の作業に分割しそれぞれの業務の専門家に割り当てることができること、開発者はEJBサーバとトランザクションやセキュリティーサービスを自動的に行うコンテナサービスを利用できること、作成したEJBに可搬性を持たせることができるためである。

3．1　EJBのアーキテクチャ

多層分散アプリケーションは、ローカルマシンで実行されるクライアント、サーバの中間層に配置されたビジネスロジック、バックエンド層のデータベースや既存のアプリケーション等がある。

3層からなるEJB多層分散アプリケーション[4]を図6に示す。

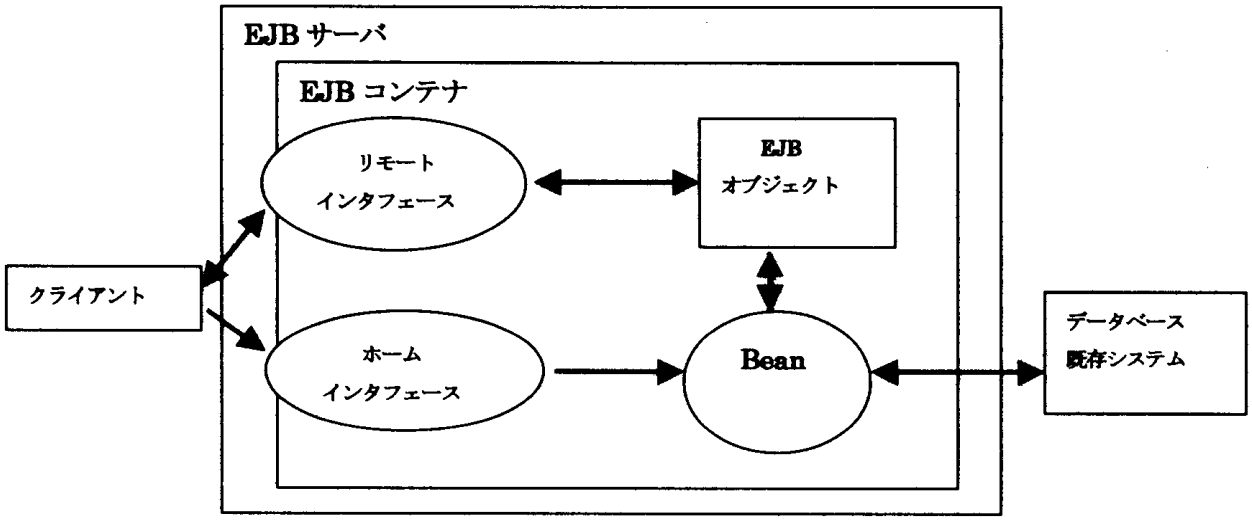

図6　EJB の環境

EJB サーバは、エンタープライズ Bean にシステムサービスを提供し、エンタープライズ Bean が実行されるコンテナを管理する。 JNDI を使ってアクセス可能なネーミングサービスとトランザクションサービスは必須である。EJB コンテナは、エンタープライズ Bean と EJB サーバを結び付け、トランザクション、セキュリティ、及びネットワーク分散を管理する。

エンタープライズ Bean には次の 3 つの機能がある。

①ホームインタフェースは、クライアントがエンタープライズ Bean のインスタンスを作成、検索、廃棄するために使用するメソッドを定義する。

②リモートインタフェースは、エンタープライズ Bean に実装するビジネスメソッドを定義する。クライアントはリモートインタフェースを介してこれらのメソッドにアクセスする。

③エンタープライズ Bean クラスは、この Bean のビジネスロジックを実装する。クライアントはリモートインタフェースを介してこれらのメソッドにアクセスする。

エンタープライズ Bean にはセッション Bean とエンティティ Bean がある。

セッション Bean はクライアントの状態を保持し、クライアントがセッションを終了するとセッション Bean はクライアントによって削除される。

セッション Bean にはステートレスセッション Bean とステートフルセッション Bean の2種類があり、ステートフルセッション Bean は特定のクライアントがデータを必要とする間（クライアントがセッションを保持している期間）存続する。一方、ステートレスセッション Bean は特定のクライアントの状態を保持しないため複数のクライアントをサポートできる。

エンティティ Bean はデータベース内のデータをオブジェクトとして表現し、通常はリレーショナルデータベースのテーブルの行を表す。複数のクライアントによって使用され、データがデータベースにある限り存続する。これら Bean の永続性をコンテナで管理する。

3．2　ホームインタフェースの検索とリモートインタフェースの取得

クライアントは JNDI 名を使ってエンタープライズ Bean のホームインタフェースを検索する。エンタープライズ Bean のホームインタフェースを取得したら、そのエンタプライズ Bean のリモートインタフェースへのリファレンスを取得することができる。[2][4]

コード例では、まず javax.naming.InitialContext オブジェクトをインスタンス化する。次にクライアントはコンテキストの lookup()メソッドを使用してホームインタフェースを検索するためリモートインタフェースの名前（ここでは CustomerEntity) を ctx.lookup()メソッドに渡す。その検索結果をホームインタフェース型 (CustomerEntityHome) にキャストし、Bean のインスタンスを作成する。

さらにホームインタフェースのメソッド findByPrimaryKey()を呼び出し、このメソッドに顧客コード (customerCode) を渡して Bean のリモートインタフェース (CustomerEntity) へのリファレンスを受け取る。

【コード例】

```
Public Cart login (String customerCode, String password) {
  Try {
    //ホームコンテキストを取得する
    javax.naming.Context ctx = new javax.naming.InitialContext();
```

```
  //JNDI 名 (CustomerEntity) を使用して、 JNDI コンテキスト取得す
  //る
  Object customerEntityRef = ctx.lookup("CustomerEntity");
  //検索結果をホームインタフェース型 (customerEntityHome) にキャス
  //トする
  CustomerEntityHome customerEntityHome = (CustomerEntity-
      Home) javax.rmi.PortableRemoteObject.narrow (Customer-
          EntityRef, CustomerEntityHome.class);
Try {
  //顧客コード (customerCode) をホームインタフェースのメソッド find-
  //ByPrimaryKey()に渡し、
  // CustomerEntity リモートインタフェースへのリファレンスを取得する
  CustomerEntity customerEntity = customerEntityHome.find-
      ByPrimaryKey(customerCode);
  //リモートインタフェースのメソッド checkPassword()を使用して、
  //パスワードが一致しているかチェックしする
  if (!customerEntity.checkPassword(password))
  //一致していない場合 null 値を返す
      return null;
  }
    ...
  //又、顧客コードと一致する顧客が見つからない場合も null 値を返す。
  //パスワードが一致した場合は、次のカートを作成する処理を実行する。
    ...
//JNDI 名(Cart)を使用して、 JNDI コンテキストを取得する
Object cartRef = ctx.lookup ("Cart");
//検索結果をホームインタフェース型 (CartHome) にキャストする
CartHome cartHome = (CartHome) javax.rmi.PortableObject.nar-
    row (CarRef, CartHome.class);
```

```
//create()は顧客コード (customerCode) パラメータを受け取り
//この顧客の Cart リモートインタフェースへのリファレンスを返す
Cart cart = cartHome.create (customerCode);
return cart;
```

4. J2EE プラットホーム

J2EE プラットホーム[5]の目的はプログラミングの生産性を向上することである。この生産性とはアプリケーション開発に関することで、開発チームに多層アプリケーションに必要なサービス標準や各種クライアントをサポートする標準を提供して、アプリケーションの開発にすばやく対応し、柔軟性を増すものである。

企業の成長に合わせたアプリケーションの性能を維持する。このため複数クライアントからの相互作用の処理、システムリソースの効果的な管理、データベースへの接続やトランザクションのサービスを行うメカニズムがある。既存システムとの統合のため、データベース管理システムやトランザクション・モニタ等の中間層やバックエンドサービスへのアクセス標準。さらに、開発者はサーバ、ツールからコンポーネントに至るまで自由に選択できることやセキュリティの維持に対応できる機能を提供している。

4. 1 プラットホームの概要

企業アプリケーションの実装、配布標準である。

企業の多層アプリケーションのサーバ側とクライアント側をサポートするように設計されている。代表的なアプリケーションはユーザインタフェースを提供するクライアント層、クライアントのサービスやアプリケーションのビジネスロジックを提供する単数／多数の中間層、データ管理を提供するバックエンド情報システム層から構成される。図7にその環境を示す。

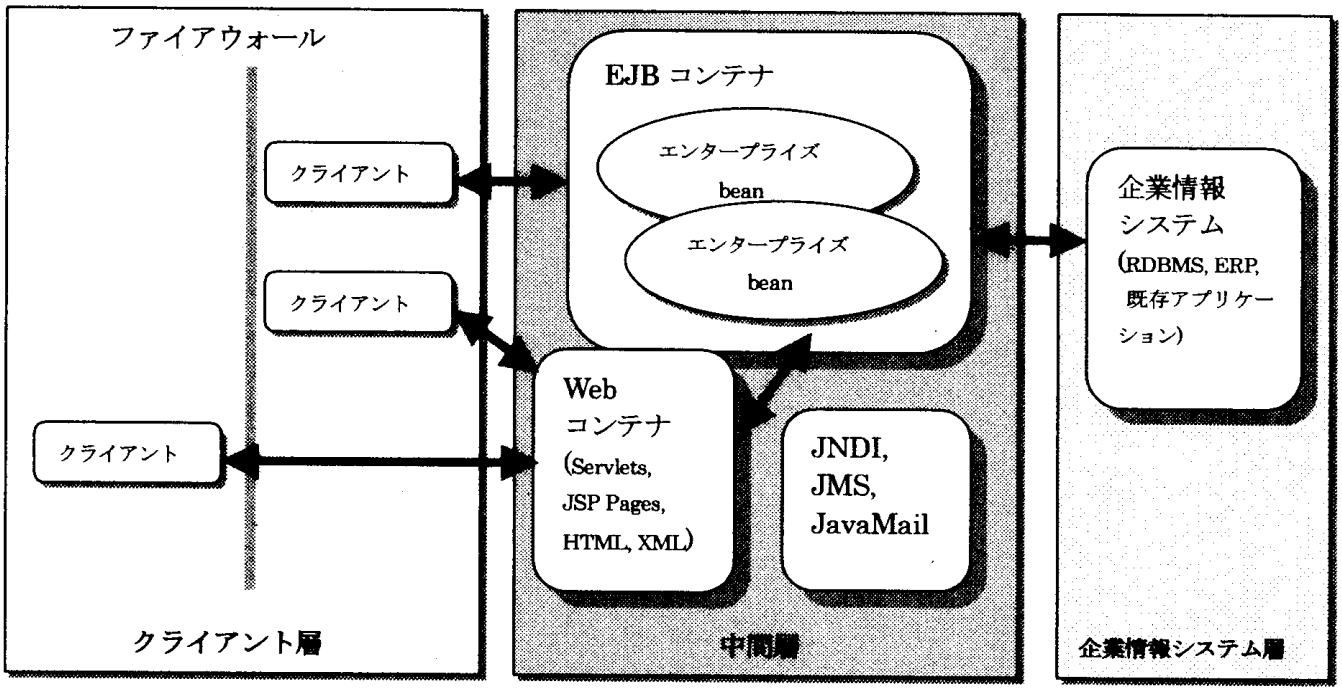

図7　J2EE の環境

クライアント層はファイヤウォール内外の各種クライアントをサポートする。中間層は Web 層の Web コンテナを通してクライアントに対するサービス、EJB 層の EJB コンテナを通してビジネス・コンポーネント・サービスを実行する。企業情報システム(EIS)層は標準 APIs を使って既存の情報システムにアクセスする。

J2EE におけるコンポーネント・ベース開発の中心はコンテナである。Web コンテナはクライアントの要求に対応するための実行支援、即ち JSP の起動やサーブレット処理等の要求実行処理、クラインとへ結果を返す処理を行う。 EJB コンテナは EJB コンポーネントのトランザクション自動実行、ライフサイクル管理、 bean 検索等の処理を行う。又、コンテナは企業情報システムに、 JDBC API を介して RDBMS アクセスを実行する。さらにコンテナは配布記述の XML タグを用いて、アッセンブルや配布時にアプリケーションの振舞いを選択するメカニズムを提供している。このためコンポーネントコードを変更してアプリケーションの振舞いの違い吸収する必要はない。

クライアント層のクライアントは HTML 、 JSP 技術で作成されるダイナ

ミック HTML 又はアプレットを用いた Web ブラウザを介して Web 層と通信する。又、クライアントは Java 言語のアプリケーションを介して EJB 層と直接通信する形態もある。中間層のビジネスロジックは EJB コンポーネントとして実装される。この EJB コンポーネント・モデルが J2EE プログラミング・モデルの中心である。

5．おわりに

信頼性が高い、大規模な企業情報システムの構築には、標準化システム基盤に J2EE, 企業パッケージに EJB が用いられる。本論文では J2EE や EJB を用いたシステム設計のポイントを記述した。しかし、学生がこの J2EE, EJB を用いたシステムの構築をとおして、基本的なシステム基盤の設計方法を修得することは難しい。理由は、 J2EE や EJB が自動的に処理する部分が多く、3階層の画面、業務ロジック、データベースの具体的な結びつきの理解が困難なためである。

しかし、 J2EE や EJB を用いた基盤システムを構築していて、すでに稼動していることを前提にした教育を行うことは可能である。例えば、ビジネスモデルを調査・研究し、その結果を基盤システムの画面や業務設計を行って情報システムを完成させる等、業務設計やシステム周辺部分の設計の教育である。

社会科学系学部における情報科学（コンピュータ）教育（その2～4）[1][6][7]において、それぞれの情報システム設計の特徴と教育について考察してきた。次回は基盤システムの進化と業務コンテンツの設計を中心に、システム設計の教育について総括する予定である。

参考文献

[1]藤尾好則：「社会科学系学部における情報科学（コンピュータ）教育（その3）」、アドミニストレーション、第7巻1～2合併号、 pp.209-223、熊本県立大学総合管理学会(2001)

[2]藤井等： Jbuilder4によるインターネットアプリケーション構築入門、カットシステム(2001)

[3]Tom Valesky: Enterprise JavaBeans, ADDISON - WESLEY, 1999
[4]Inprise Corporation: エンタープライズアプリケーション開発者ガイド、インプライズ(2000)
[5]Nicholas Kassem and the Enterprise Team: Designing Enterprise Applications with the Java2 Platform, Enterprise Edition, ADDISON - WESLEY, 2000
[6]藤尾好則：「社会科学系学部における情報科学（コンピュータ）教育」、アドミニストレーション、第5巻4号、 pp.39 - 49、熊本県立大学総合管理学会(1999)
[7]藤尾好則：「社会科学系学部における情報科学（コンピュータ）教育（その2）」、アドミニストレーション、第6巻2～3合併号、 pp.1 - 12、熊本県立大学総合管理学会(2000)