Webアプリケーション学習用軽量コンテナの開発
A Development of Lightweight Containers for Learning Web Applications

佐藤英人
Hideto Sato
東京国際大学 商学部 情報システム学科
Department of Information Systems, Faculty of Commerce, Tokyo International Univ.

要旨
Web アプリケーションの世界では、入門レベルとフレームワーク・コンテナを駆使するレベルに大きなギャップがあり、この間を埋める教材やツールが欠けているように思われる。本論文では、大規模業務用とは異なる要件をもつ教育用あるいは小規模業務用のコンテナ及びフレームワークとして開発した Sabaphy を紹介する。これは、言語に PHP5 を使用し、EJB3 を簡略化したオブジェクト永続化とセッション管理の機能等を提供するもので、特別なインストールは不要、簡単なことは簡単に実行できる、段階的に学習できるといった特徴をもっている。

1. はじめに

1990 年代後半以降インターネットの普及とともに、Web アプリケーションがビジネス情報システムの中心的役割を占めるようになってきた。そのアーキテクチャとして 1999 年に提案された J2EE（Java2 Enterprise Edition）の考え方[1][2]は、今日では情報システムの標準的な構成方法として広く受け入れられており、この構成に従った多くのコンテナやフレームワークが登場してきている。

本稿では、この J2EE に代表されるアーキテクチャを教育するためのツール（コンテナやフレームワーク）について考察し、PHP5 を用いて開発した教育用コンテナ Sabaphy の特徴と概要を説明する。

なお本稿では、ライブラリクラスでユーザ作成オブジェクトの操作を行うものを「コンテナ」、依存性注入も行うものを「DI コンテナ」、複数コンテナが協調してプログラムのフロー制御を行うものを「フレームワーク」と呼んでいる。

2. Web アプリケーション向けコンテナの現状

図 1 は J2EE に沿ったアプリケーションの構成例である。核になる部分がドメイン層でここでは永続化サービスを提供する EntityBean コンテナ、セッションサービスを提供する SessionBean コンテナが提供されている。プレゼンテーション層については、Struts などのフレームワークが使用されている。

上で述べた EJB コンテナは複雑で難しいということから、これに代わるものとして、O/R マッピングコンテナ（Hibernate）が登場し、これらのコンテナを組み込んで使用するための DI コンテナ（Spring, Seasar など）が使われるようになってきている[3]。これらの動きを受けて EJB も現在 EJB3 に向けて改訂中である[4]。

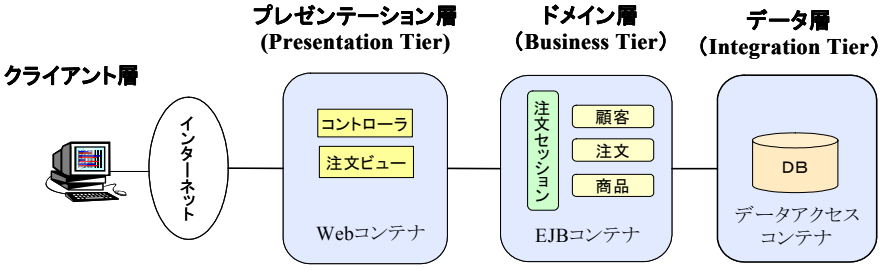

図 1　J2EE アーキテクチャ

3. Web アプリケーション学習と既存コンテナ

ソフトウェアアーキテクチャを学習する上で効果的な方法は、それに沿ったアプリケーションを学習者自身が作成し、実用的に使ってみることである。上記の軽量コンテナは、いずれもオープンソースであり、入手上の障害はないが、初心者である学習者に使わせるには、以下の３つの問題がある。

(1) インストールと初期設定：共用環境である教室での使用や学習者の自宅での利用を考えると、特別なインストールや設定を必要とするものを使用することは難しい。

(2) 難解な概念用語：既存コンテナはいずれも大規模業務用としての使用を想定しており、Web アプリケーション全般について利用者が一通りの知識を持つものとして、概念が構成されている。

(3) Java 言語の問題：軽量コンテナの多くは Java で書かれている。静的型付けを使う Java は初心者には

難しい。また、Web 環境では独特の配備方法やネーミングサービスを理解する必要がある。

4. Sabaphy の特徴

Sabaphy は初心者が教室でも自宅でも Web アプリケーションの学習を一通りできるように開発されたコンテナ群であり、以下の７つの特徴を持っている。

(1) 習得が容易な言語として PHP5 を採用: Web アプリケーションの入門には、PHP や Perl が使われている。2004 年に登場した PHP5 は、それまでの PHP の機能に加え、Java によく似た構文、キーワードをもつオブジェクト指向機能を提供するもので、入門レベルからのステップアップに最適な言語である。

(2) オープンソースのみを使用: ライセンス管理に煩わされることなく、自宅でも利用できる。

(3) 特別なインストール・設定は不要: ライブラリフォルダをドキュメントルートにコピーするだけで使える。また、設定が不要なシリアル DB（オブジェクトをシリアル化して格納する DB）も利用できる。

(4) 標準的な概念で構成: DI コンテナ、EJB 相当のコンテナ、Hibernate 型 O/R マッピングを採用。

(5) 段階的な学習が可能: 入門レベルから、徐々にコンテナを取り込み、段階的に学習できる。

(6) 可能な限り単純に: 必要最小限の機能に絞り込んでいる。その詳細については後述する。

(7) ある程度実用的に使える: 簡単な設定変更だけで、シリアル DB をリレーショナル DB（MySQL 等）に置き換えることができ、ある程度のものであれば、実用的に動かすことができる。

5. Sabaphy アプリケーションの構成

Sabaphy アプリケーションにはフレームワークを使うものと使わないものの２種類のものがある。図２はフレームワークを使った場合のアプリケーション構成である。クライアントから入力データが送られてくるとサーバ上のメインプログラム（メイン PHP）が立ち上がる。メイン PHP は、DB の所在やコマンド情報などの設定情報を与えて Sabaphy を初期化し、処理をフレームワークに委ねる。Dispatcher と CommandHandler がフレームワークの実装である。**Dispatcher** は入力データから処理すべきコマンドを識別し、処理を CommandHandler に委譲する。

CommandHandler は与えられたコマンド情報に従って、ユーザ作成のワークユニット上のアクションを起動し、ユーザ作成のビューに出力 HTML の生成を依頼する。ここで**ワークユニット**はセッションが維持されている間データを保持し、１組のトランザクションを実行するオブジェクトで、EJB の Stateful SessionBean に相当する。ワークユニットは必要に応じ、エンティティを生成/更新し、それに処理を依頼する。**エンティティ**は DB に永続化されるオブジェクトで、EJB の EntityBean に相当する。

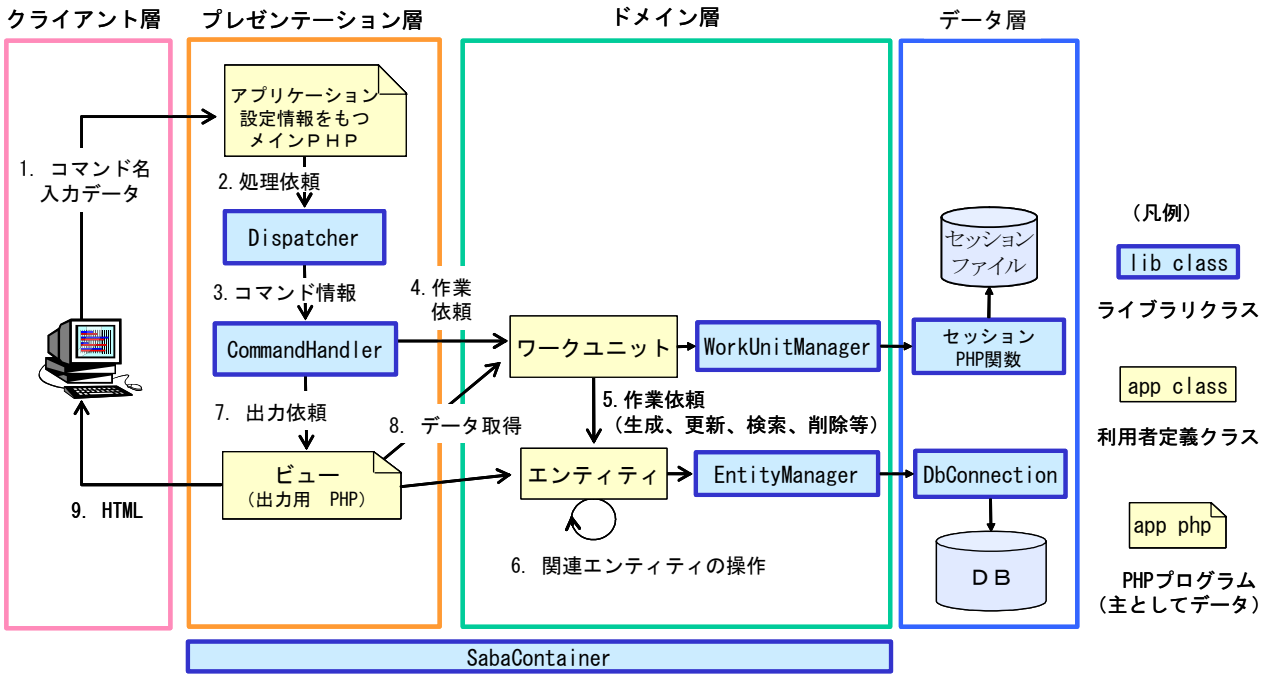

図２　Sabaphy アプリケーションの構成

ワークユニット、エンティティはともに何も継承しない通常のオブジェクト（Java でいえば POJO）である。これらオブジェクトをワークユニットあるいはエンティティとして機能させるものが、**WorkUnitManager, EntityManager** である。EntityManager は Hibernate 型の O/R マッピング[5]を採用しており、EJB3 の EntityManager とほぼ同様である。図２の下部の **SabaContainer** は、DI コンテナ[6]であり、要求に応じてコンポーネント間に依存関係を注入し、コンポーネントのインスタンスを提供する。

6. 利用者作成プログラム

利用者が作成しなければならないプログラムは、メイン PHP、ワークユニット、エンティティ、ビューの４種である。以下では顧客登録のミニサンプルを例として、これらを説明する。

6.1. エンティティ

エンティティは永続オブジェクトであり、その属性データは DB に格納される。図３は顧客クラスの定義例で、3～5 行で id、name、address の３つの属性を宣言している。7～9 行は O/R マッピング情報を与えるメソッドである。

エンティティでは、その生成・更新・削除を、EntityManager を介して DB に通知する。14、18、21 行がそれで、getEntityManager() は EntityManager を取得するグローバル関数である。

```
1)  <?php
2)  class Customer {
3)      public $id;
4)      public $name;
5)      public $address;
6)
7)      function keyName() { return "id"; }        //主キー属性
8)      function tableName() { return "Customer"; } //テーブル名
9)      function initialKeyValue() { return "C0000"; } //自動付番初期値
10)
11)     function init($name, $address) {
12)         $this->name = $name;
13)         $this->address = $address;
14)         getEntityManager()->save($this);    //DBに保存を依頼
15)     }
16)     function changeAddress($newAddress) {
17)         $this->address = $newAddress;
18)         getEntityManager()->update($this);  //DBに更新を依頼
19)     }
20)     function delete() {
21)         getEntityManager()->delete($this);  //DBに削除を依頼
22)     }
23) }
24) ?>
```

図３　エンティティのクラス定義例

6.2. ワークユニット

ワークユニットはセッションデータを保持し、トランザクションを実行するオブジェクトである。図４は顧客の登録を行うワークユニットで、はじめに 6～10 行の receiveInput()メソッドで入力データをキャッシュし、クライアントからこれでよいという確認が得られたとき、11～15 行の fixInput()メソッドで、キャッシュされていたデータを使って顧客インスタンスを生成する。

```
1)  <?php
2)  class InputCustomerWorkUnit {
3)      public $name;
4)      public $address;
5)      // アクションメソッド
6)      function receiveInput() {
7)          //入力データをセッションデータとしてキャッシュ
8)          $this->name = $_REQUEST['name '];
9)          $this->address = $_REQUEST['address '];
10)     }
11)     function fixInput() {
12)         //キャッシュデータを使ってオブジェクトを生成
13)         $customer = new Customer();
14)         $customer->init($this->name, $this->address);
15)     }
16)     function getAllCustomers() {
17)         return getEntityManager()->findAll('Customer');
18)     }
19) }
20) ?>
```

図４　ワークユニットのクラス定義例

6.3. ビュー

ビューは HTML を生成するプログラムである。PHP はそれ自身 HTML テンプレートエンジン（Java の世界の JSP 相当）であるので、Sabaphy では普通に PHP でビューを記述する。

6.4. メイン PHP

メイン PHP はクライアントから呼び出されるプログラムである。図５にあるように、(1)使用するプログラムファイルを指定し、(2) DB ファイルの所在などの設定データを与えて DI コンテナを初期設定し、(3)入力データとして与えられたコマンドに応じた処理を記述する。この図はフレームワークを使用しない場合の例である。このコマンドに応じた処理を定型化し、データ（コマンド情報）として処理内容を記述するように変えたものが、Sabaphy のフレームワークである。

7. Web アプリケーションの段階的学習

Sabaphy を使用することで、入門レベルから、一部プログラムを手直しするだけで連続的に、以下の Web アプリケーション技術を学ぶことができる。詳しくは Sabaphy のホームページを参照されたい[7]。

(1) Web アプリケーション入門（標準 PHP のみを使用）、(2)オブジェクト永続化とセッション管理の導入、(3)MVC パターンの適用、(4)フレームワークの導入、(5)個別機能の高度化：リレーショナル DB の利用、ユーザ認証の利用など。

8. 簡略化している機能

Sabaphy では一通り Web アプリケーションを構築できるが、業務用コンテナに比べて、以下のように多くの機能を省略あるいは簡略化している。

省略している機能： 分散オブジェクト機能、コネクションプール、二相コミット、AOP（ただしコマンドチェーンの利用で類似のことは可能）など。

簡略化している機能

(1) 主キーは単純キーのみ（複合キーは未サポート）、(2) DB の列名はオブジェクトの属性名と同じと仮定、(3) 1 度に使用できる DB 接続は 1 本のみ、(4) 排他制御とコミットだけの単純化したトランザクション管理、(5) セッション管理は PHP の標準機能を使用、(6) 入力データは PHP の標準リクエストデータを使用。

```
1) <?php
2) // (1) 使用するプログラムファイルの指定
3) ini_set('include_path', '.;../..');
4) require_once 'sabaphy/SabaContainer.php';
5) require_once 'InputCustomerWorkUnit.class.php';
6) require_once 'Customer.class.php';

7) // (2) 初期設定
8) session_start();
9) $cnt = new SabaContainer();
10) $cnt->setup(array('dbConnection'=>'SdbConnection', 'file'=>'db/database'));

11) // (3) 処理の振り分け
12) $cmd = null;
13) if (isset($_REQUEST['_COMMAND'])) $cmd = $_REQUEST['_COMMAND'];
14) if (($cmd == null) || ($cmd == '戻る')) {
15)     require 'newCustomer.html.php';
16) } elseif ($cmd == '送信') {
17)     $workunit = new InputCustomerWorkUnit();
18)     $workunit->receiveInput();
19)     getWorkUnitManager()->save($workunit);
20)     require 'confirmCustomer.html.php';
21) } elseif ($cmd == 'この内容で保存') {
22)     $workunit = getWorkUnitManager()->restore('InputCustomerWorkUnit');
23)     $workunit->fixInput();
24)     require 'newCustomer.html.php';
25)     getEntityManager()->commit();
26)     getWorkUnitManager()->delete($workunit);
27) } elseif ($cmd == '入力終了') {
28)     require 'showCustomers.html.php';
29) }
30) ?>
```

図５　メイン PHP の例　（フレームワーク未使用時）

9. むすび

Sabaphy は 2005 年 4 月から情報システム学科の筆者の演習（２、３年生）で使用している。まだ成果を評価できる段階ではないが、Web アプリケーションに関心を持つ層を広げることができたといえる。

「より簡単に（Ease of Development）、よりはやく」というのは、システム開発における時代の流れである。スクリプト言語（PHP）を使用した軽量コンテナという Sabaphy は、この流れに沿った１つの解の提案でもある。しかし、Sabaphy では、８章で述べたように多くの単純化を行っている。これらの単純化が妥当なものであるかどうか、今後多くの方々の意見を拝聴していきたいと考えている。

なお、Sabaphy の設計に当たっては、Hibernate, Spring をはじめ多くのオープンソースのコンテナ・フレームワークを参考にさせていただいた。これらの開発者の方々に謝意を表します。

参考文献

[1] Singh, I., Stearns, B. and Johnson, M., *Designing Enterprise Applications with the J2EE Platform*, Second Edition（J2EE アプリケーション設計ガイド 第２版）, Sun Microsystems, 2002.
[2] Alur, D., Crupi, J. and Malks, D., *Core J2EE Patterns Best Practices and Design Strategies*, Second Edition, Prentice Hall, 2003.
[3] 岡本隆史他, *Light Weight Java,* 毎日コミュニケーションズ, 2005.
[4] EJB 3.0 Expert Group, *JSR220: Enterprise JavaBeans, Version 3.0, EJB 3.0 Simplified API*, Sun Microsystems, 2005.
[5] Elliott, J., *Hibernate*, (佐藤直生訳, Hibernate, オライリージャパン, 2004), O'Reilly Media Inc., 2004.
[6] Fowler, M., “Inversion of Control Containers and the Dependency Injection pattern”, http://www.martinfowler.com/articles/injection.html, （角谷訳, http://www.kakutani.com/trans/fowler/injection.html）, 2004.1.
[7] 佐藤英人, “Sabaphy ホームページ”, http://satolab.tiu.ac.jp/SabaphyHome/, 2005.8.
[8] Johnson, R., *Expert one-to-one J2EE Design and Development* (今野睦監訳, 実践 J2EE システムデザイン、ソフトバンク、2003）, Wrox Press, 2002.