7V-8

PHP+Smarty を用いた単一データベースによる複数サイトの構築

北浦 慶尚† 前田 奈緒美† 植村 俊介† 植木 泰博‡ 冬木 正彦† 堀川 和義§ 荒川 雅裕†
関西大学工学部† 関西大学先端科学技術推進機構‡ 株式会社 e-kikai§

1. はじめに

近年,インターネット電子商取引サイト（以下,EC サイト）は企業にとって,重要な手段となっている．さらに,情報掲載数の増加のため,検索のしやすさや商品の表示方法の変更などの,ホームページのデザインを容易に変えることができる“即時性”が求められている.

Web アプリケーション開発の点から見ると，同様の商品を取り扱っていている EC サイトは,同じビジネスロジックが用いられることもある．しかし,デザインが異なれば,個別に実装する必要がある.同時に,Web アプリケーション開発で広く用いられている PHP 言語は「HTML 埋め込みモデル」であるがゆえ,デザインとビジネスロジックの両方の知識が必要であり,デザイナーとプログラマの分業を行うことが困難である．さらに,画面デザイン部分の変更が生じた場合,プログラム全体の構造を把握する必要があり,求められる即時性と相反する結果となり一度作成されたホームページの変更は困難となる.

本研究では,Web アプリケーションの開発に MVC モデルを適用し,単一のデータベース・共通のロジックを用い開発し,その効率・生産性を検証する.

2. システムの検討

2.1 システムの概要

対象とする EC サイトは,中古機械情報を取り扱う”e-kikai”（A）と、同様の商品を取り扱うサイトBである．サイトの機能は,中古機械在庫情報検索・会員によるメール配信機能・2 者会員間の取引機能などである．また,２つのサイトの他に,サイト運営者やデザインは異なるが,これらの機能を持つ EC サイトを８つ（合計 10 サイト）作成できるように設計した.

2.2 データベースの共通化

データベースは,データを構造化して保持するとともに,特定のアプリケーションに依存しない汎用的なアクセスの手段をアプリケーションに提供するものである．本研究ではこの特徴を用い,単一のデータベースを複数のアプリケーションに対応させる．また,データベースに格納された情報は全サイトで使用されるとは限らない．よって,商品をどのサイトで使用するかを識別するために,商品毎にサイトフラグを立てる.

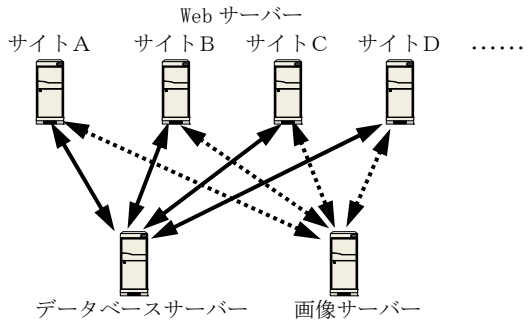

図１ データベースとロジックの共通化の概念

2.3 MVC モデルによるロジックの共通化

ソフトウェア開発の方式として,MVC モデルがある．これは,アプリケーションを「Model（ビジネスロジック）」「View（デザイン）」「Controller（Model と View の制御）」の３つの役割で明確に分離しようとするものである[1]．MVC モデルに基づき,アプリケーション開発言語で「フレームワーク」が作られ,デザイナーとプログラマの分業が進められている.

PHP 言語においてもフレームワーク mojavi・maple 等が作られている．しかし,PHP のフレームワークは熟成されておらず簡単には利用できないことがわかった.

テンプレートエンジン Smarty は,PHP のデザインとロジックの分離を実現している．本システムでは,PHP と Smarty を用いることでデザインとロジックの分離を行い,さらに汎用的なデータベースアクセスクラスを作成することで MVC モデルを実現する.

3. システムの設計

3.1 要素技術

本システムは OS に FedoraCore4,Web サーバーとして Apache2.0.54,データベースサーバーとして PostgreSQL8.0.3,開発言語として PHP5.0.4,テンプレートエンジン Smarty2.6.9 を利用した.

3.2 実装手順

実装手順を以下に示す.

① 現在稼動しているサイトＡのデザインをその

Development of Different Websites on Single Database by PHP and Smarty
† Yoshinao Kitaura, Kansai University
† Naomi Maeda, Kansai University
† Shunsuke Uemura, Kansai University
‡ Yasuhiro Ueki, ORDIST, Kansai University
† Masahiko Fuyuki, Kansai University
§ Kazuyoshi Horikawa, e-kikai
† Masahiro Arakawa, Kansai University

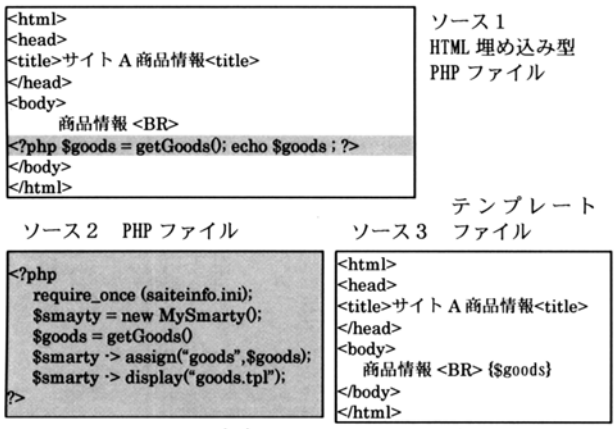

図２　デザインとロジックの分離

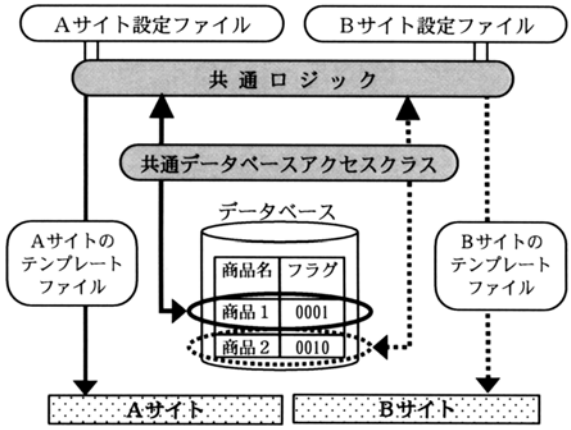

図３　データベースとロジックの共通化

ままに,デザインとロジックの分離を行う.

② サイトＡのデザインリニューアルを行う.

③ サイトＢを,サイトＡの持つ機能の中からいくつかの機能を用いて,デザインと共にリニューアルを行う.

実装手順①において行った,デザインとロジックの分離・データベースとロジックの共通化の仕組みを以下に示す.

3.3 デザインとロジックの分離

図２ソース１の HTML 埋め込み型の PHP ファイルを,PHP のみで構成したファイルと HTML 構文のみで構成したファイル（テンプレートファイル）に分離する．PHP ファイルでは,動的に出力する変数をテンプレート変数として格納し,表示するテンプレートファイルを呼び出す（図２ソース２）．呼び出すテンプレートファイルでは,{$変数名}と表記した部分に格納された変数が代入されて,ブラウザへの表示が行われる（図２ソース３）.

3.4 データベースとロジックの共通化

クライアントからの要求に応じて,Web サーバー上の共通ロジックは,共通データベースアクセスクラスを呼び出す．共通データベースアクセスクラスでは,設定ファイルで格納されたサイト情報を元に,必要な情報を取り出す（図３）．このとき必要な情報は,管理者情報・会員情報・在庫商品情報・取引情報などである．これらすべての情報にサイトフラグを設定し,サイト毎の情報を識別する．これにより,複数のサイトに共通する情報（たとえば、サイトＡとサイトＢの２つに入会している会員情報）は一括で管理することができる．さらに商品情報等に紐付けされる動画・画像は,NFS を用いて画像サーバーから呼び出される.

4. システムの評価および考察

実装手順①前後・実装手順②後での比較を表１に示す.

図２より,デザインとロジックの分離を行うと,ファイル数が多くなり,ファイルサイズも大きくなる.しかし表１より,実装手順①後のファイルサイズが小さくなった．これは共通データベースアクセスクラスを用いることにより汎用性が進んだ結果である.

表１ 実装前後のファイルサイズ・実装時間の比較

	合計 ファイルサイズ	実装時間
元サイト	1,895,072 byte	-
元サイト MVC 化	1,386,601 byte	延べ 450 時間
サイトリニューアル	1,249,619 byte	延べ 80 時間

また,実装手順②は延べ 80 時間であった．変数の出力をテンプレートファイルに依存させ,ロジックとデザインを明確に分離させることにより,デザインの変更が容易になったことが確かめられた．これにより，デザイナーとプログラマの分業を行うことも可能になった．さらに,今後のプログラムの保守やロジックの変更も容易であると考えられる.

サイトＢは,サイトＡと共通のビジネスロジックを用いるため,ロジックを先行開発したことに相当する．このため,実装手順③のサイトＢリニューアルに要する時間は,短縮される．結果として,デザインに依らないロジックを先行開発することは,開発期間短縮に対して有効であると考えられる．あわせて,商用 Web アプリケーションにとって重要なデザインを,ユーザーのニーズに即座に合わせ短期間で開発できると考えられる.

5. まとめ

本システムで扱った中古機械情報のように,複数の独立したサイトでも同じ商品情報を扱う EC サイトは多く存在する．それらに対しても,本研究を適応し,ユーザーのニーズに直結した Web アプリケーション開発をしていきたい.

参考文献

[1] 山田祥寛;Smarty 入門 PHP+テンプレートエンジンでつくる MVC アプリケーション,翔泳社(2005)