Webアプリケーションにおける予約業務フレームワークの実現と再利用性の評価

津久井　浩[††]　中所武司[†]

インターネットやイントラネットに接続されたコンピュータシステムのオフィスへの導入に伴い、現在様々な業務がWebアプリケーション化されている。今回、Webアプリケーション開発技法の研究として、問題領域に特化したフレームワークをとりあげた。対象問題領域に予約業務を選択し、アプリケーションの具体例として開発した会議室予約システムと座席予約システムから共通部分を分析することで、フレームワークの抽出を行なった。また、開発したソースコード自動生成ツールを併用することで、本フレームワークによる予約業務Webアプリケーションの開発負担をより軽減することを実現した。そして、本フレームワークを再利用性の観点で分析するために、例題業務として商品予約と図書予約を選択し適用実験を行なった結果、本フレームワークで定めたデータ処理方式に基づくことで幅広い予約業務に対して高い再利用性を挙げられることを確認した。

Web Application Framework for Reservation Systemes and its Reusability

Hiroshi Tsukui[††] and Takeshi Chusho[†]

How to develop web applications becomes important as the number of end-users using the Internet increases on the inside and outside of offices. A domain-specific framework must be a solution of this problem. In this paper, a web application framework for reservation systems based on 3-tier architecture of the Struts, is developed after two different types of reservation systems, namely, for room reservation and for seat reservation, are built. In addition, an automatic code generation for DB table processing and dynamic pages generation, is developed. The reusability is confirmed by applying the framework and the tool to the other types of reservation systems for book reservation and goods reservation.

1. はじめに

インターネットやイントラネットに接続されたコンピュータシステムがオフィスを中心に積極的に導入され、システムへの頻繁な機能変更を伴うものも多い。最近では、ユーザインタフェースにWebブラウザを利用するWebアプリケーションが増加している。しかしWebアプリケーションをゼロから開発するのは、開発コストが増大するだけでなく、開発方法の習得時間もかかるため、短期開発の要求にこたえられない。

そこで、最近の情報システム構築では、オープンなアーキテクチャとフレームワーク、デザインパターン、コンポーネントなどの構成要素からビジュアルツールを用いてアプリケーションを再帰的に構築していくことが追求されている。特に問題領域に特化したフレームワークは、その領域のアプリケーション構築を行なう時の開発効率の向上を図れることが確かめられている[3)4)]。そこで本研究[7)]では、Webアプリケーション開発にフレームワークを適用することによる開発者の負担軽減について再利用性の観点から分析した。対象問題領域には予約業務を選択した。

本稿では、最初に3層Webアプリケーションおよび、Webアプリケーション開発のためのフレームワークStrutsを適用したシステムアーキテクチャについて述べる。次に、対象問題領域とした予約業務の定義や体系および、抽出した予約業務フレームワークについて述べる。そして、予約業務フレームワークの適用によるWebアプリケーション開発の効率をより向上させるために開発したソースコード自動生成ツールについて述べ、最後に構築したフレームワークの例題業務への適用実験と評価の結果をまとめる。

2. 3層Webアプリケーション

2.1 3層アーキテクチャ

これまで分散システムとして広く使用されてきたク

† 明治大学大学院 理工学研究科 基礎理工学専攻 情報科学系
†† 2003年4月から（株）日立製作所 勤務

ライアント／サーバ（C/S）システムは、クライアント側にアプリケーションをインストールする必要があり、アプリケーションに修正が必要なときには再インストールしなければならないなどデメリットが大きい。一方、Web アプリケーションは、クライアント側はブラウザだけなので、このような問題は生じない利点がある。3 層 Web アプリケーションは、図 1 のようにユーザインターフェースを提供するプレゼンテーション層、アプリケーションの処理を行なうアプリケーション層、データを管理するデータソース層の 3 つの層で構成されており、それぞれが、Web ブラウザ、アプリケーションサーバ、データベースサーバに対応している。

本研究では、アプリケーションサーバに Tomcat、データベースサーバに Oracle を適用した。なお実装言語には、Java 言語と Java のサーバサイド技術であるサーブレット・JSP を適用している。

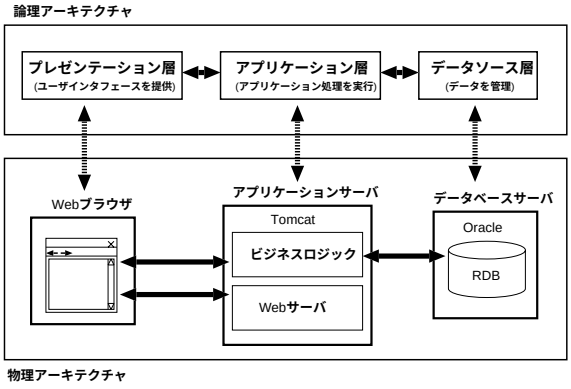

図 1 3層アーキテクチャ

2.2 Struts の適用

サーブレット・JSP を中心とするサーバサイド Java テクノロジーは Web アプリケーションの実行速度や拡張性の面における課題を改善した。しかし、今日のように業務仕様の変更や機能拡張などの要求が頻繁に発生することに伴い、より一層の開発期間の短縮や信頼性のさらなる向上が課題となっている。このような背景から Web アプリケーション・フレームワークである Struts[1] が登場した。Struts は、特定の問題領域に特化せず、Web アプリケーションを開発するためのテンプレートを提供する。今回、下記のような理由から Struts を適用することにした。

- 開発容易性
  従来の Web アプリケーション開発で開発者自身が行う必要のあった、HTTP リクエストの管理や、フォームデータの格納、画面フローの制御などを、Struts が機能として提供している。特に画面フローの制御方法は、プログラムとは別に、外部に XML で定義するという方式をとっているので、階層的構造でフローを理解しやすい、フローの変更にプログラムをなぞる必要がないなど、頻繁に発生する画面周りの仕様変更に柔軟に対応できる。
- 保守性・拡張性
  Struts は、アプリケーションの状態を保持するモデル、モデルに対する表示形式を提供するビュー、クライアントからのリクエストを受け付けるコントローラーから構成される Model-View-Controller(MVC) パタンに準拠している。このため、各々のサブシステムの役割が明確に分担されており保守性・拡張性が高い。

Struts を適用したシステムアーキテクチャを図 2 に示す。 それぞれのサブシステムの役割について述べる。アクション・サーブレットは、アプリケーションの中に 1 つだけ存在し、すべての HTTP リクエストを管理する。送られてきたリクエストを最初に受け取り、その内容を解析し、その他のサブシステムへ指示を送る。アクション・フォームは、Web ページ上でのフォーム情報を扱うクラスで、Struts が提供するクラスを拡張することが前提となる。ユーザが入力したフォームの値を格納しておく。アクション・マッピングは、URL とアクションのマッピング情報を保持する XML 文書ファイルである。アクション・クラスは、ビジネスロジックを呼び出したり、記述したりするクラスである。Java クラス（モデル）は、アクション・クラスからの処理依頼を受け、ビジネスロジックを実行するクラスである。SQL 文の作成や発行を行ない、必要に応じて結果をアクション・クラスに返す。Java クラス（動的 HTML 文書生成）は、表示情報の動的生成を伴う HTML 文書の作成を行なうクラスである。例えば、空き状況を示すためのカレンダーや座席表などの HTML 文書を作成する。

3. 予約業務フレームワークの抽出

3.1 フレームワークの対象問題領域の定義

フレームワークの対象問題領域として選択した予約業務は、施設の予約やチケットの予約などが挙げられるが、紙や電話による従来の手続きを Web ブラウザ上で実現することで、手間や人件費の削減など利用者と運営者の両方に高い利益をもたらす業務の 1 つである。本研究では、予約業務における予約の定義を「存

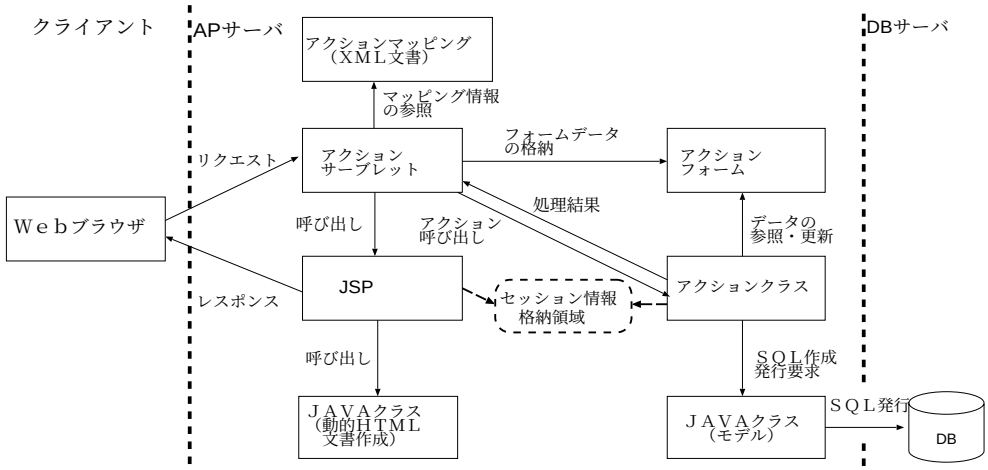

図 2 Struts を適用したシステムアーキテクチャ

在する資源を、一定期間占有することをあらかじめ宣言すること」とする。また、本フレームワークが対象とする予約業務の基本機能は「予約」「取消」「照会」の３つとする。この機能はサービスを受ける側である利用者を対象としている。システムの管理者もしくは資源の保持者が使用する機能については本フレームワークの対象としない。

次に、予約業務における概念モデルを図 3 に示す。

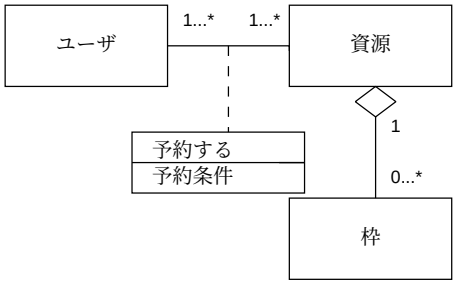

図 3 予約業務の概念モデル

予約業務に存在する概念は大きく、ユーザ（利用者）・資源・枠の３つにわけられる。ユーザは実際に予約・取消・照会手続きを行なう利用者のことを指す。資源の種類には、商品予約や図書予約に代表される物、施設予約や座席予約に代表される場所、病院の診療予約に代表される人などが挙げられる。枠は定義された資源によっては、存在する場合と存在しない場合の両方が考えられる。例えば、会議室予約に代表される施設予約系は、１つ１つの施設が資源となり、決められた単位時間が枠という形で区切られている。しかし、ある商品を資源とした場合は、その商品の在庫が複数あったとしても１つ１つの在庫を ID などで区別していない限り枠という形でわけることはできない。

次に、予約業務を体系化した様子を図 4 に示す。予約業務では枠の概念が、施設予約などに代表される時間、座席予約に代表される空間、商品予約などの時間・空間どちらに該当しないものの３つに、大きくわけることができると考えられる。そして、その枠が単一か複数かでわけられる。例えば、チケット予約において、利用者が予約できる枠が S・A・B 席にわけられており、各々の席が複数ある場合は、枠の数が複数となる。この場合、図 3 の資源は会場に対応し、枠は S・A・B の３つが対応する。そして、各々の席の数は、枠が持つ属性として扱う。もう一つのケースとして、利用者が予約する枠が S1,S2,,,B3 と１座席ずつ区別されていた場合、枠の数は単一となる。概念モデルの枠は、会場にある座席の数だけ存在することになる。

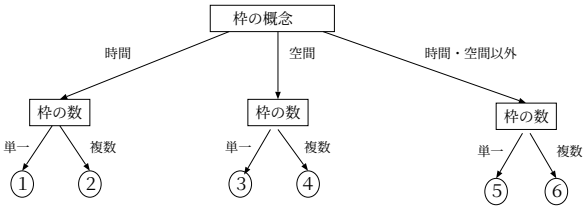

図 4 予約業務の体系

3.2 フレームワークの抽出方法

フレームワーク抽出のために、最初にアプリケーションの具体例を２つ選択し、それを実際に構築することで共通部分の分析および抽象化を行なった。例題システムには、会議室予約システムと座席予約システムを選択した。例として会議室予約システムにおける予約画面を図 5 に示す。 会議室予約は、図 4 の体系

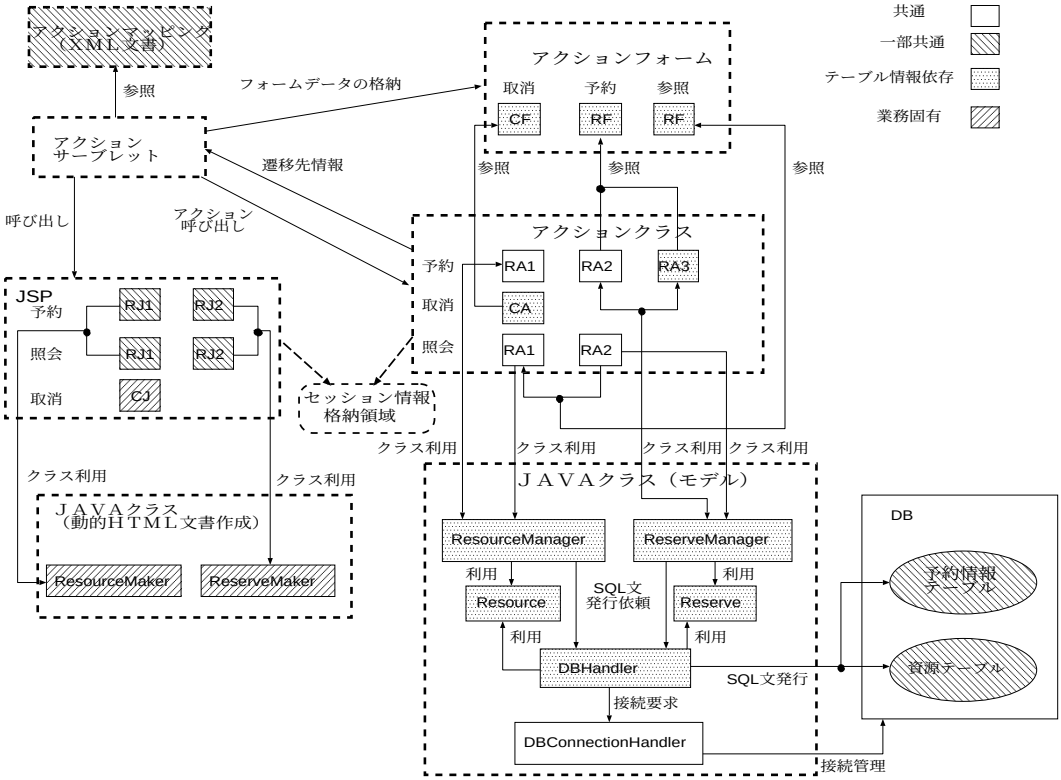

図 6 抽出したフレームワーク

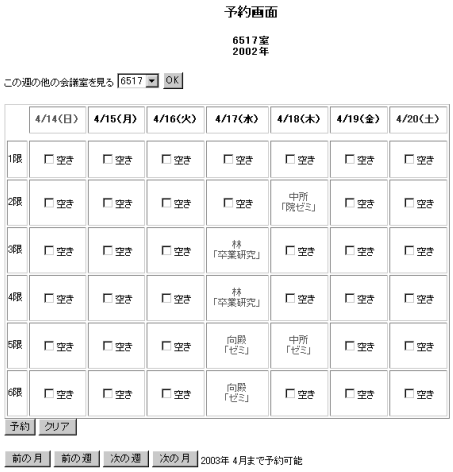

図 5 会議室予約システムの予約画面

１に該当し、予約単位が時間（時限）となるのに対し、座席予約は体系３に該当し、予約単位が空間（座席）となる。このように予約単位の異なる２つの業務のプログラム上での共通箇所を抽象化することで、抽出するフレームワークの対象問題領域を広く設定できるように考えた。

3.3 抽出したフレームワーク

抽出したフレームワークを図６に示す。点線の囲みは、図２のシステムアーキテクチャ上の各サブシステムに対応する。その中の各クラスは、再利用の観点で、以下の４種類に分類できる。

- 共通
- 一部共通
- ＤＢテーブル情報依存
  テーブル設計の内容に依存するが、その内容が決まれば自動的に決定するクラス
- 業務固有
  クラス内で宣言されているメソッドは共通であるが、その処理内容はアプリケーションに固有であったクラス。

次に各サブシステムの構成を述べる。

ＤＢテーブルは、予約の対象となる資源を管理する資源テーブルと、予約のために必要な情報を管理する予約情報テーブルの２つから構成される。両システムのテーブルを図７に示す。 資源テーブルには、会議室あるいはイベントに関する情報が格納される。そして、資源を特定するための列が主キーとして設定されている。予約情報テーブルは、誰がどの資源のどの枠

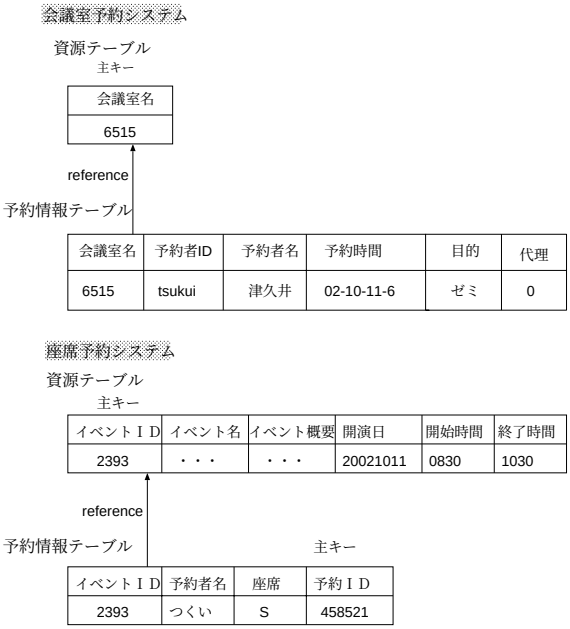

図 7 両システムにおける抽出した DB テーブルの構成

を予約したかを管理するもので、会議室予約では予約者 ID・会議室名・予約時間が、座席予約では予約者名・イベント ID・座席が対応する。それ以外に存在する列は各業務に固有なものである。

次に、Java クラス（モデル）は以下の構成から成る。

Reserve:予約情報を定義したデータ型クラス
ReserveManager:Reserve オブジェクトを利用して、予約処理などを行なうクラス
Resource:資源情報を定義したデータ型クラス
ResourceManager:Resource オブジェクトを利用して新しい資源の追加や削除を行なうクラス
DBHandler:SQL 文を発行するクラス
DBConnectionHandler: DB へのコネクションを管理するクラス

Java クラス（動的 HTML 文書生成）は以下の構成から成る。

ResourceMaker: 資源の一覧を表示し選択させる HTML 文書を作成するクラス
ReserveMaker:予約状況を表示し枠を選択させる HTML 文書を作成するクラス

アクション・クラスは以下の構成から成る。

RA1:ResourceManager から資源情報を取ってくるクラス
RA2:ユーザが選択した資源の予約情報を ReserveManager から取ってくるクラス
RA3:予約に必要な情報を集め ReserveManager に予約依頼するクラス
CA:取消に必要な情報をリクエストから受け取り ReserveManager に取消依頼するクラス

JSP は以下の構成から成る。

RJ1:どの資源を予約するかを選択するための Web ページを構築するクラス
RJ2:対象資源の予約状況を表示し、枠を選択するための Web ページを構築するクラス
CJ:予約取消のために必要な情報を表示し、ユーザが選択もしくはテキストによる入力を行なう Web ページを構築するクラス

アクションフォームは以下の構成から成る。

RF 予約手続きの開始から完了までの間にユーザから受け付けた入力情報を保持するクラス
CF:取消手続きの開始から完了までの間にユーザから受け付けた入力情報を保持するクラス

アクションマッピングは、予約手続きのページフローである資源の選択→枠の選択→予約処理が共通しているので、この記述箇所から構成している。

アクション・サーブレットは両システムとも Struts で提供された 1 クラスから構成している。

3.4 システムの流れ

抽出したフレームワークを適用した際のシステムの処理の流れを示す。例として予約手続きのなかで、ユーザが予約する資源を選択してからその資源の予約状況を表示する Web ページの作成までの流れを図 8 に示す。 具体的には、以下のような手順となる。

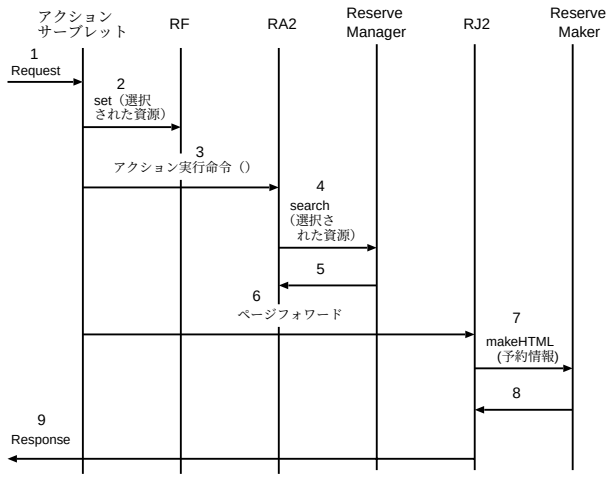

図 8 システムの流れ

(1) アクションサーブレットがユーザリクエストを

受け取る
(2) 選択された資源が RF にセットされる
(3) アクションサーブレットが RA2 にアクション実行命令を行なう
(4) RA2 が Java クラス（モデル）の ReserveManager に選択された資源の予約情報 (Reserve オブジェクトの格納された配列) を全てとってくるように依頼する
(5) RA2 は予約情報を受け取り、セッション情報に格納する
(6) アクション・サーブレットがアクション・マッピングを参照し、RJ2 にフォワードする
(7) RJ2 はセッションに格納されている予約情報を受け取り、ReserveMaker に渡す
(8) ReserveMaker はその予約情報をもとに HTML 文書を作成し、RJ2 に返す
(9) RJ2 が HTML 文書をクライアントに返す

3.5 ステップ数によるフレームワークの割合

抽出したフレームワークが、会議室予約システムと座席予約システム内で占めたステップ数の割合を、サブシステムごとに図 9 に示す。フレームワークのステップとして含めるのは、
(1) 共通クラス
(2) 一部共通クラスの共通記述箇所
(3) テーブル情報依存クラス
である。業務固有クラス、及び片方のシステムのみで作成・利用されたクラスについては、フレームワークのステップに含めない。なお、フレームワークの占めるステップ数が 2 つのシステムで異なるのは、テーブル情報依存クラスのステップ数がテーブルの列の数などによって変化するためである。

	サブシステム	Javaクラス（モデル）	アクションクラス	アクションフォーム	JSP	Javaクラス（動的HTML文書生成）	アクションマッピング	合計
会議室予約	全体	740	370	170	320	550	100	2250
	フレームワーク	460	280	140	190	0	60	1130
	業務に固有	280	90	30	130	550	40	1120
	フレームワークが占める割合	62%	76%	82%	59%	0%	60%	50%
座席予約	全体	550	300	90	260	130	90	1420
	フレームワーク	480	270	90	190	0	60	1090
	業務に固有	70	30	0	70	130	30	330
	フレームワークが占める割合	87%	90%	100%	73%	0%	67%	77%

図 9 ステップ数におけるフレームワークの割合

フレームワークの占める割合の合計が、座席予約の７７%に対し、会議室予約が５０%と低くなった原因は、以下の 3 つである。
(1) Java クラス（動的 HTML 文書生成）のカレンダーを作成する ReserveMaker でのステップ数が大きかった。特に、休日を表示するための計算や、次の月や前の週などの表示期間変更のための計算処理部分の記述量が大きかった。
(2) Java クラス（モデル）において、休日管理や時間の計算処理のためのクラスが存在した。
(3) アクション・クラスにおいて、予約手続きの操作性向上のための処理や、予約確認画面の挿入に伴う処理が必要であった。

すなわち、会議室予約では時間という枠の概念に伴う休日管理や時間の計算処理が発生したことと、予約確認画面の挿入の影響がそのページの処理を行なうアクション・クラスにも及んだことが再利用率の低下を招いたといえる。。

一方、座席予約では、Java クラス（モデル）とアクション・クラスとアクション・フォームの部分のみでは、平均して 90 %近い部分がフレームワークによって構築されている。これは、フレームワークに沿ったページフローであったこと、ビジネスロジックが比較的単純であったことに起因する。

4. ソースコード 自動生成ツール

4.1 概　　要

抽出したフレームワークには、テーブル情報依存クラスが存在する。これらは、抽出した資源テーブルと予約情報テーブルの設計によって、クラスの内容が決定できる。また、抽出したフレームワークの Java クラス（動的 HTML 文書生成）内の 2 クラスは共に業務固有クラスであるが、空き状況を示すカレンダーや座席表などの動的な HTML 文書は通常予約業務において必要となるものである。

そこで、テーブル設計、テーブル情報依存クラス、Java クラス（動的 HTML 文書生成）に関して必要な情報を定義することで、ソースコードを自動生成できるツールを開発した。これにより開発者は、テーブルを作成するための SQL 文の作成・実行や、動的な情報を表示するためのコーディングをゼロから開発する必要がなくなる。

4.2 定 義 手 順

本ツールは、
1 資源テーブル
2 予約情報テーブル
3 Web 上での表示方法
の順に定義を行なう。

資源テーブルと予約情報テーブルの定義は、通常の DB テーブル設計と同じで各列の定義と一意な列の定義からなる。ただし予約情報テーブルは、資源テーブ

ルの１列に外部キーを設定する。また、定義した各列の値をユーザがいつ入力するかを定義する。この入力方法は、３つに分けられる。

(1) 予約時に入力
(2) 予約時以外に入力
(3) 不要

予約時以外に入力する情報とは、ログイン時など予約を行なう前に別の機能で既にユーザから入力されている情報をいう。不要とは、システム内部の処理によって決定される情報をいう。この定義は、予約処理に関するソースコード部分を自動的に生成するために必要なものになる。例として、座席予約における資源と予約情報テーブルの定義を図 10 に示す。

予約情報	イベントID	予約者名	座席	予約ID
一意				○
外部キー	○			
入力方法	予約時	ログイン時	予約時	不要

図 10 座席予約の予約情報の定義

手順３の Web 上の表示方法の定義は、Java クラス（動的 HTML 文書生成）で作成する HTML 文書を、ツール側が定義した表示形式の分類から選択し、それに応じた必要な情報をツールに入力していく。

以上の定義を行なうことで、ツールはアプリケーションのソースコードを自動生成する。生成方法は、通常の Java コードの記述と同様に、フレームワークを構成する各クラスの中にソースコードを出力していく。開発者は、生成された各クラスを継承によって拡張したり、新規にクラスを作成することで Web アプリケーションを構築していく。

4.3 ツールによる再利用性の向上

本ツールをフレームワークの抽出に利用した会議室予約システムに適用した結果、Java クラス（動的 HTML 文書生成）を構成する２クラスの自動生成できる割合は図 11 のようになった。Java クラス（動的 HTML 文書生成）内で 44 %が自動生成され、システム全体としては、フレームワークの割合が 11 %上昇することを確認した。

5. 適用実験と評価

5.1 適 用 例 題

実験の例題として、図書予約システムと商品予約システムの２つを選択した。図書予約は、本の貸借を時間によって管理するので、図 4 の体系１に該当する。

会議室予約		Resource Maker	Reserve Maker	合計	システム全体
ツール未使用	全体	70	480	550	2250
	フレームワーク	0	0	0	1130
	業務に固有	70	480	550	1120
	フレームワークが占める割合	0%	0%	0%	50%
ツール使用	全体	70	480	550	2250
	フレームワーク	40	200	240	1370
	業務に固有	30	280	310	880
	フレームワークが占める割合	57%	42%	44%	61%

図 11 ツールの使用による再利用率の変化

しかし、会議室予約のように予約する時間という枠の概念が本には存在しない点が異なる。また、資源が解放（返却）された時点で、次の予約が可能となり、予約の定義である「占有をあらかじめ宣言」とは異なる。このような業務を本フレームワークに適用することで、どの程度の再利用率が得られるかを確認する。商品予約は、図 4 の体系５・６に該当し、時間管理と空間管理の予約業務から抽出したフレームワークがこのような業務にも適用可能かどうかを確認する。なお、どちらの例題システムも単純なビジネスロジックのみを実装する。

5.2 実 験 結 果

フレーワークと業務に固有の記述箇所をサブシステムごとにステップ数によって比較した結果を図 12 に示す。また、適用実験とフレームワークの抽出に用いた合計４つの業務における全体の再利用率を図 13 に示す。なお、会議室予約と座席予約の値は、商品予約と図書予約と同条件で比較を行なうために自動生成ツールを使用した結果を示している。

	サブシステム	Javaクラス（モデル）	アクションクラス	アクションフォーム	JSP	Javaクラス（動的HTML文書生成）	アクションマッピング	合計
図書予約	全体	410	290	120	270	180	70	1340
	フレームワーク	310	260	120	190	100	60	1040
	業務に固有	100	30	0	80	80	10	300
	フレームワークが占める割合	76%	90%	100%	70%	56%	86%	78%
商品予約	全体	510	310	160	300	200	70	1550
	フレームワーク	350	280	160	190	60	60	1100
	業務に固有	160	30	0	110	140	10	450
	フレームワークが占める割合	69%	90%	100%	63%	30%	86%	71%

図 12 適用例題におけるフレームワークの割合

図書予約の場合、新規に追加したクラスは、Java クラス（モデル）で１クラスと、JSP に２ページ（予約完了ページと取消完了ページ）のみであった。合計の再利用率は７８%と高い値を示した。資源を時間で管理する業務の中では特殊な業務であるが、フレームワークの用意したコードを全く修正することがなくアプリケーションを構築することができた。

商品予約の場合も、図書予約と同様に再利用率７１

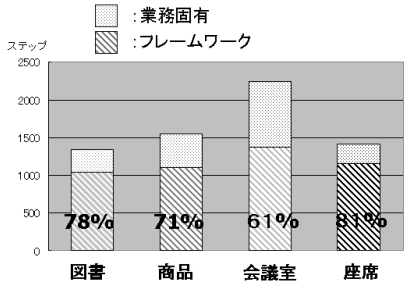

図 13 4つの予約業務における全体の再利用率

%と高い数値を得ることができた。新規に追加したクラスも図書予約と同様で、Java クラス（モデル）に１クラス、JSP に２ページである。この商品予約はフレームワークの抽出に用いた時間管理の会議室予約や空間管理の座席予約とは体系上は異なる位置づけとした。しかし DB テーブルの構成を同様の形で設計すれば、本フレームワークに沿った形でアプリケーションを構築できることを確認した。

5.3 総 合 評 価

本フレームワークによるアプリケーション構築では、以下のことがいえる。

5.3.1 適用可能範囲

本フレームワークは、DB テーブルにおいて資源テーブルと予約情報テーブルという２つのテーブルをもとにしている。そして、

(1) 予約処理のときは、予約情報テーブルへレコードを追加
(2) 取消処理のときは、予約情報テーブルから該当するレコードを消去

という設計に従うことで、本フレームワークに沿った形でアプリケーションを構築することができ、図４の体系で分類したあらゆる予約業務に対して、幅広く適用することが可能である。

5.3.2 拡張性・保守性

アプリケーションを構築する際に、フレームワークのどの箇所を拡張すればよいかを理解しやすいことは重要である。本フレームワークは Struts フレームワークの上に構築すると共に、MVC モデルに基づいてサブシステムの役割が明確化されているので、どの部分を拡張すればよいか分かりやすい。

5.4 再利用率の低下要因の考察

本フレームワークは予約業務に特化していることから、予約手続きを資源の選択→枠の選択→予約処理という流れに定めている。しかし実際には、枠の選択と予約処理の間に予約確認画面などを挿入するケースが考えられる。また、DB テーブルについては、資源テーブルと予約情報テーブルの２テーブルからのみ構成されており、本フレームワークが提供する機能内にそれ以外の DB テーブルを使用するケースも考えられる。それらのテーブルについては、開発者自身が作成し、それに伴う処理の記述もフレームワーク内に組み込む必要がある。会議室予約の再利用率が他の業務に比べて低くなった要因の１つに、これらが実際に該当していたことが言える。このような拡張を行なう場合には、フレームワーク内部の詳細をある程度知る必要がでてくる。また、多くのサブシステムに影響範囲が及ぶとも考えられる。よって、ページ遷移の追加・変更や DB テーブルの追加などは、実際のアプリケーション構築で発生することを前提に考え、フレームワークを拡張することが今後の課題となる。

6. おわりに

本稿では、ニーズの高まっている Web アプリケーションにフレームワークを適用することによって開発負担を軽減する方式を提案した。構築したフレームワークの再利用性検証のため、例題システムへの適用を行なった結果、フレームワークの指定した DB 設計に基づけば、幅広い予約業務に対して高い生産性をあげられることを確認した。

参 考 文 献

1) JakartaProject:The Struts Web Application framework
http://jakarta.apache.org/struts/
2) 熊崎敦司, 野呂昌満, 張漢明, 蜂巣吉成:Web 情報システムのソフトウェアアーキテクチャ, オブジェクト指向最前線 2002,pp93-96(2002)
3) 島田圭，中所武司：フレームワークによる 3 層 Web アプリケーション構築法，情報処理学会研究報告 2000-SE-129，pp.1-8(2000).
4) 藤原克哉，中所武司：窓口業務を例題としたエンドユーザ向き分散アプリケーションフレームワーク wwHww の開発と適用評価，情報処理学会論文誌，Vol.41，No.4，pp.1202-1211(2000).
5) ジム・コナレン：UML による Web アプリケーション開発 ピアソン・エデュケーション (2000)
6) 原田洋子:Java サーバサイドプログラミング, 技術評論社 (2001)
7) 津久井浩, 中所武司：Web アプリケーションにおける予約業務フレームワークの抽出実験と再利用性の検証情報処理学会 研究報告「ソフトウェア工学」No.139 - 007(2002)