B-018

問題領域を特化したWebアプリケーションフレームワーク構築方法の実験と評価

Domain-Specific Framework for Web Application and the Evaluation

周 鋒† 中所 武司†
Feng Zhou Takeshi Chusho

1 はじめに

インタネットの普及とデジタルコンテンツの拡大により，掲示板，ショッピングサイトなど，Webアプリケーションの利用が一般化している．

しかし，各アプリケーションをゼロから開発するのは容易ではない．そこで，再利用技術を用いてWebアプリケーションを効率的に開発するために，これらのWebアプリケーションに共通するアーキテクチャやクラス間の連携方法などの仕組みに着目して基盤となるフレームワークを開発する技法が重要になっている．

我々のこれまでの研究では，Struts[1]，Hibernate[2]といった汎用的なフレームワークよりも問題領域を特化した予約業務フレームワークを開発し，適用評価を行った結果，会議室予約では61%，商品予約では71%の再利用率を確認できた[3].

一般には，問題領域を狭い範囲に限定すると再利用率が高くなり，広い範囲を想定すると再利用率が低くなるというトレードオフの関係がある．

今回，問題領域と再利用率のトレードオフの関係を定量的に明確化するため，広い問題領域と考えられる予約業務とそれよりも狭いルーム予約業務の2つのフレームワークを開発し，それぞれ商品販売と教室予約のアプリケーション開発への適用実験を行った．

2 システムアーキテクチャ

従来の3層アーキテクチャ[4]は，ユーザインタ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うアプリケーション層，データを管理するデータ層で構成されている．実装レベルでは，それぞれWeb ブラウザとＷｅｂサーバ，アプリケーションサーバ，データベースサーバに対応する．

但し，JSP/Servletを中心とした開発では，HTMLとJavaコードの混在のため，互いの変更がもう一方に与える影響が大きいことや．画面フローが把握しづらいことなどにより，システム全体を理解しづらく，開発と保守の容易性を低下させてしまうという問題があり，Strutsに代表されるWebアプリケーションフレームワークが開発された．StrutsはMVCモデルに準拠しており，アプリケーションの状態を保持するモデルと，表示・出力を司るビュー，入力を受け取ってその内容に応じてビューとモデルを制御するコントローラの役割が明確に分離されている．

今回は，図1に示すように，3層アーキテクチャにStrutsを適用するとともに，Strutsが対応していないビジネスロジックとデータベースアクセスも加えることにより，より高い保守性と拡張性を実現した．

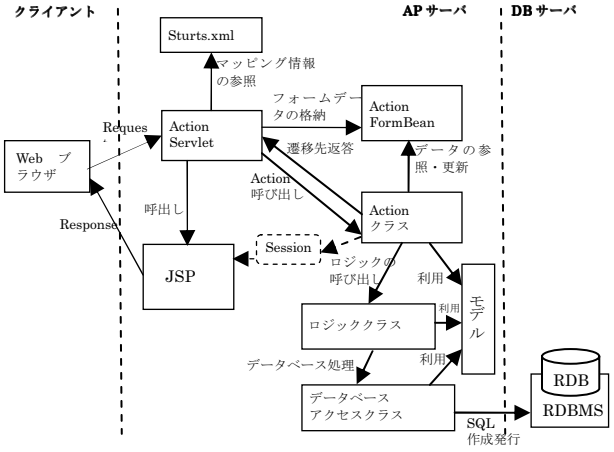

図1.システムアーキテクチャ

3 システム概要

本研究では，対象とする問題領域を予約業務とし，従来のような書類や電話による施設予約やチケット予約などの手続きをWeb上で実現できるようにする．

フレームワークとしては，広い問題領域の予約業務と狭い問題領域のルーム予約業務の2種類を開発した．広い範囲の予約業務フレームワークにおける予約業務の定義は「存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」としたので，施設予約，チケット予約，商品予約といったあらゆる予約業務が含まれる．その機能仕様を次の表1に示す．

表1.予約業務の機能仕様

	一般ユーザー	システム管理者
ユーザー管理	ログイン	ログイン ユーザー登録・変更・削除
資源管理	—	資源登録・変更・削除
予約機能	新規予約・予約照会・変更・削除	新規予約・予約照会・変更・削除

狭い範囲のルーム予約業務フレームワークでは，予約業務の子問題領域として，会議室予約，教室予約などのルームの予約を対象とする．機能としては，表1の予約業務機能に，時間チェックなどルーム予約に特有な機能を追加した．

なお，今回開発したフレームワークでは，表1の機能のうち，予約業務の核となる一般ユーザーの予約機能とログイン機能の2つの機能を対象とした．

†明治大学大学院理工学研究科基礎理工学専攻
ソフトウェア工学研究室

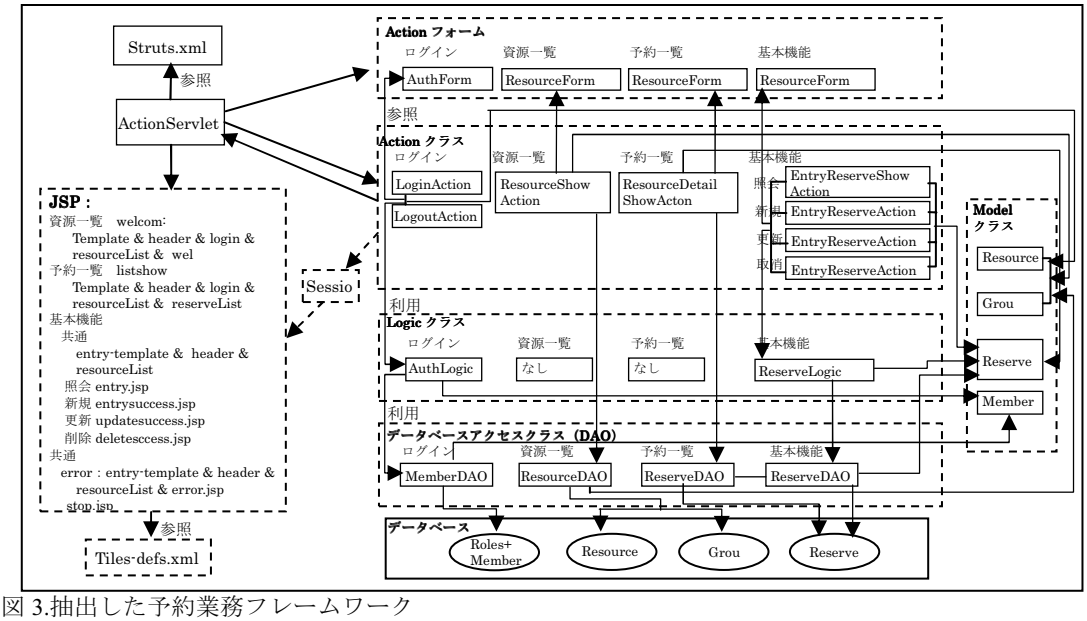

図 3.抽出した予約業務フレームワーク

4　構築実験

4.1　フレームワークの構築方法

今回のフレームワークの構築と評価は，「機能分析と設計」→「例題開発」→「フレームワーク抽出」→「適用評価」という流れとなっている．即ち，3 節の機能仕様に基づいて典型的な例題システムを開発し，この例題システムをベースに予約業務の共通部分を抽出して，フレームワークを構築した．

4.2　例題開発

3 節の機能仕様に基づいて，予約業務フレームワークとルーム予約業務フレームワークの両方を抽出するための会議室予約システムを開発した．

この例題システムでは，図 2 のように，最初に，ヘッダ，ログイン画面，会議室一覧，説明画面からなる初期画面が表示され，会議室を選択すると月間予約状況照会でき，さらにログインした後に個々の予約詳細の照会・変更・取消ができるというようになっている．

図 2.会議室予約システムの画面例

4.3　予約業務フレームワークの抽出

広い問題領域の予約業務フレームワークの抽出は，それぞれ新規予約，変更などの機能ごとに行った．各機能に対応するクラス，設定ファイル，データベースのうち，予約業務としての共通部分をフレームワークとして抽出した．結果を図 3 に示す．

以下，図の主要な抽出部分の詳細について述べる．

データベース：予約対象とする資源を保存する Resource テーブル，資源分類の Grou テーブル，ログインのための member，roles テーブル，予約情報を管理する Reserve テーブルが共通となる．各テーブル内共通の列もそれぞれ抽出しているが，詳細の説明は省略する．

Action クラス： ActionServlet から呼び出され，必要な初期化を行い，適切なロジッククラスを呼び出し，結果を受けてリンク先に遷移する．資源表示の ResourceShowAction，予約一覧表示ための ResourceDetailShowAction，予約・照会・変更・取消のための EntryReserveAction と EntryReserveAction を共通として抽出した．

Action フォーム：ユーザから送られた値をアクションへ転送する役割を担う．ResourceForm などのフォームクラスと Struts.xml 設定中で定義したダイナミックフォームを抽出した．

Model クラス：Resource，Grou，Reserve，Member を抽出した．

Logic クラス：ビジネスロジックを処理するクラスである．ReserveLogic というクラスで予約の各処理を行う．

データベースアクセスクラス：資源のアクセスのための ResourceDAO と予約のアクセスのための ReserveDAO を抽出した．

ビュー：表示する JSP ファイル，レイアウトを管理するテンプレートページおよびその設定ファイルを抽出した．

Struts.xml：アクションマッピングの設定として共通の部分を抽出した．

結果として，例題のステップ数 1444 のうち，抽出した予約業務フレームワークは 895 ステップとなった．

4.4　ルーム予約業務フレームワークの抽出

次に，問題領域を狭くしたルーム予約業務フレームワークの抽出も行った．ルーム予約業務は予約業務の子問題領域であるため，抽出したフレームワークのクラスとファイル類については，4. 3 節の予約業務フレームワーク

とほぼ同じである．以下，予約業務フレームワークとの異なるところのみを述べる．

ロジッククラス：予約したい時間帯に既存の予約があるかどうかのチェックのための例外処理クラス

DAO クラス：ロジッククラスでの時間チェックのための SQL 発行処理

ビュー：予約一覧表示の月間画面はルーム予約業務ではカスタマイズなしで共通になった；予約業務にて一部共通として抽出したが，テンプレートとその設定は完全に共通になった；予約時間が既存の予約と重複している場合の例外表示ページも追加した．

Struts.xml：例外のマッピング設定も予め決められるため，アクションマッピングの設定については，カスタマイズなしで完全共通になった．

結果として，抽出したルーム予約業務フレームワークのステップ数は 1130 となり，予約業務より 235 ステップ増加した．

5　適用実験と評価

本研究で抽出した広い領域の予約業務フレームワークと狭い領域のルーム予約業務フレームワークをシステム開発に適用する実験を行い，再利用率の視点で評価を行った．

5.1　予約業務フレームワークの適用試験

予約業務の「存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」の定義の範囲内で，フレームワークの抽出用に用いた会議室予約システムとかなり性質の異なる書籍販売システムを適用実験の対象として選択した．書籍販売システムは，典型的な予約業務ではないが，購入したい商品を永久的に占有することを宣言するという形で予約業務の一種と考えられる．

開発した書籍販売システムにおける予約業務フレームワークの割合は，表 2 に示す．

表 2.予約業務適用実験におけるフレームワークの割合

	Action クラス	Action Form	モデル	ロジッククラス	DAO クラス	ビュー	アクションマッピング	合計
全体	268	84	168	84	343	402	70	1419
フレームワーク	234	47	84	49	245	175	61	895
業務固有	34	37	84	35	98	227	9	524
フレームワークの割合	87%	56%	50%	58%	71%	43%	87%	63%

システム全体の 1419 行のソースコードの中の 895 行がフレームワークで提供されているので，63%の手間を省くことができたことになる．よって，予約業務フレームワークの場合，かなり性質の異なる予約業務も対応できるという点が確認できた．但し，ほぼ全クラスとファイルのカスタマイズが必要となっている．

5.2　ルーム予約業務フレームワークの適用試験

ルーム予約業務は，問題領域を狭く取り，ルームの予約を対象とするため，適用実験では，学校の教室予約システムを選択した．

フレームワーク抽出に用いた会議室システムの予約単位は何時何分から何時何分までという通常の時間指定になっているが，教室予約システムでは，あらかじめ決められた時限単位で予約を行う．そのほか，予約に必要な情報についても異なる部分がある．

開発した教室予約システムにおけるルーム予約業務フレームワークの割合は，表 3 に示す．

表 3.ルーム予約業務適用実験におけるフレームワークの割合

	Action クラス	Action Form	モデル	ロジッククラス	DAO クラス	ビュー	アクションマッピング	合計
全体	279	98	126	78	292	465	63	1401
フレームワーク	234	47	84	64	266	372	63	1130
業務固有	45	51	42	14	26	93	0	271
フレームワークの割合	83%	48%	66%	82%	91%	80%	100%	81%

この適用実験で 81%の再利用率を達成できたことからわかるように，かなり性質の近い業務に適用する場合，再利用率が大幅に向上できるといえる．問題領域を狭くすると処理する業務も明確になるため，フレームワークを通じて設計の部分もある程度再利用できるようになることを確認できた．

5.3　総合評価

適用実験により，以下の結論を得た．

(1) トレードオフ関係の定量的な確認
問題領域と再利用率の間にトレードオフの関係があるという定性的な認識があるが，今回その関係を定量的に確認できた．すなわち，広い領域の実験では 63%の再利用率であったのに対し，狭い領域の場合は，81%の再利用率が達成された．よって，システム開発する際，出来るだけ一番適用できる狭いフレームワークを選択すべきであるといえる．

(2) 拡張性・保守性
開発したフレームワークは，MVC モデルをベースに，ビジネスロジック層，データベースアクセス層も加えたため，全体を理解しやすくなり，拡張・保守しやすくなるという利点もある．

参考文献

[1] The Apache Software Foundation
http://struts.apache.org/

[2] Red Hat, Inc.
http://www.hibernate.org/

[3] 中所武司，津久井浩，予約業務を例題とした Web アプリケーション用フレームワークの再利用性の評価，電子情報通信学会 和文論文誌 D-I 分冊，Vol.J88-D-I, No.5, pp.930-939 (May. 2005)

[4] 中所武司，藤原克哉　Java による Web アプリケーション入門，サイエンス社(Feb. 2005)