エンドユーザ向けWebアプリケーションフレームワークの提案と試作

Web Application Framework for End-User-Initiative Development

李 静† 中所 武司†
Sei Ree Takeshi Chusho

1 はじめに

インターネットの普及と共に、様々な業務がWebアプリケーション化され、それぞれのWebアプリケーションをゼロから開発するのは負担の大きな作業であるため、Webアプリケーションを効率的に開発するための基盤を提供するフレームワークやコンポーネントを使うことが基本的な技法となっている。また、情報システムは、従来、情報処理の専門家が開発し、限られた人達が利用してきた。しかし、ワークステーションやパソコンの普及およびそれらをつなぐネットワークの普及に伴い、一般の業務の専門家が自ら情報システムやWebアプリケーションを利用するようになってきた。

そこで，我々は、このような業務の専門家が日常的に担当する小さな部門や個人の業務を対象とする中小規模のWeb アプリケーションに関して、低コストで短期間に開発するとともに、頻繁な機能変更を伴う保守に対応することが重要になるという観点から、その分野の業務の専門家主導で開発・保守できるような技法を研究してきた。[1]

2 エンドユーザ主導開発

本研究では、外注する予算の確保が難しい中小規模のWebアプリケーションを対象として、サービスの頻繁な変化にも対応できるようなエンドユーザ主導型フレーワークを提案する。エンドユーザである業務専門家が、自らの有する業務知識に基づき、このフレームワークを用いて自らWebアプリケーションを構築することをめざしている。

そこで、エンドユーザがコーディングなしで、Webアプリケーションを開発できるフレームワークの試作・評価を行うために、次のようなアプローチをとった。まず、不用品交換システム[2]を例題として選定し、ソフトウェアの全体像を把握するために例題アプリケーションを構築する。次に、フレームワーク作成の基盤とするために、AJAX[3]とJSON[4]技術を用いて例題アプリケーションを再構築する。この実装した例題アプリケーションからコンポーネントを抽出し、コンポーネント間をつなげるAPIを作成する。最後に、これらのコンポーネントとAPIによりフレームワークを構築する。

3 例題アプリケーションの構築

3.1 アプリケーションの概要

今回、不用品交換システムを例題としてとりあげた理由は、最近、環境問題が重要視されていることがある。持続可能な社会実現の一環として、IT技術の適用による資源の節約や環境保全の達成（Green-By-IT）が期待されていることから、たとえば、地方自治体や町内会のボランティアで運営している不用品交換ショップを担当者自身がWebサイトとして立ち上げることができれば、大きな効果が期待できる。

本研究では、不用品を提供する人とそれを活用したい人の仲介をするシステムを想定している。ユーザは当システムを用い、PC端末で不用品の情報を登録したり、申請したり、検索したりすることができ、全部で10画面により構成する。図1は不用品交換システムの画面遷移を示し、次のユースケースを備える。

―メールアドレスを登録する。
―ユーザ情報を登録・更新する。
―システムにログインする。
―不用品情報を登録・更新する。
―不用品情報を検索する。
―不用品を申請する。

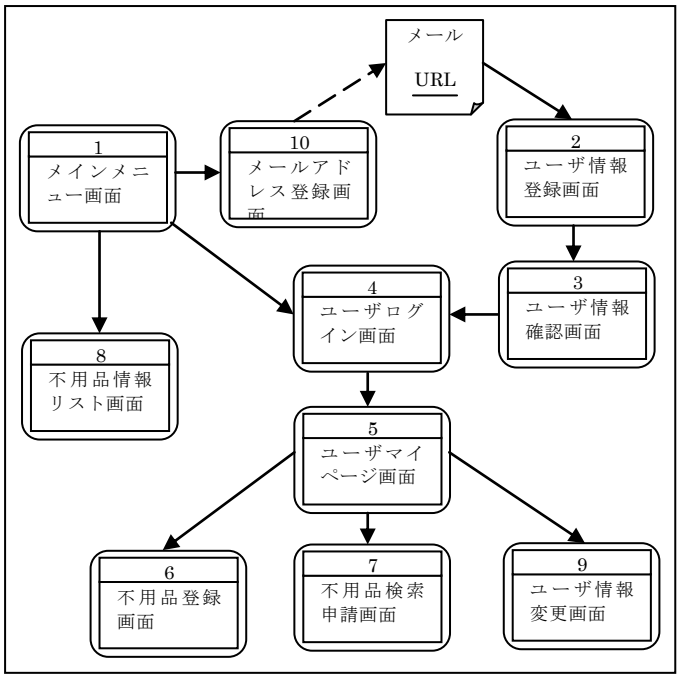

図1. 不用品交換システムの画面遷移図

3.2 基本的アーキテクチャ

システムは、従来の3層アーキテクチャを採用し、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

†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ソフトウェア工学研究室, Meiji University

タを管理するデータ層で構成される。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。

近年、Webアプリケーションを開発するために、MVCモデルに準拠するStrutsに代表される汎用的フレームワークが利用されている。MVCモデルは、アプリケーションの状態を保持するモデルと，表示・出力を司るビューと、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。その結果、処理フローが把握しやすくなるため、システム全体を理解しやすく、開発と保守の容易性を向上させるという効果がある。

しかし、Strutsなどの汎用的フレームワークはコントローラへの設定が複雑という問題がある。そこで、本稿では、汎用的フレームワークの代わりに、JSP/Servletモデルに基づくシンプルなフレームワークEcoFWを開発した。図2は、フレームワークEcoFWを用いたシステムアーキテクチャを表している。EcoFWでは、URLによりロジッククラスを呼出し、ビューとロジッククラスを分離することが実現できた。さらに、データベースへの操作を容易にする機能も備えた。図2において、APサーバ層の実線部分はドメイン非依存であり、他のドメインにも適用する。点線の箇所はドメインに依存する。

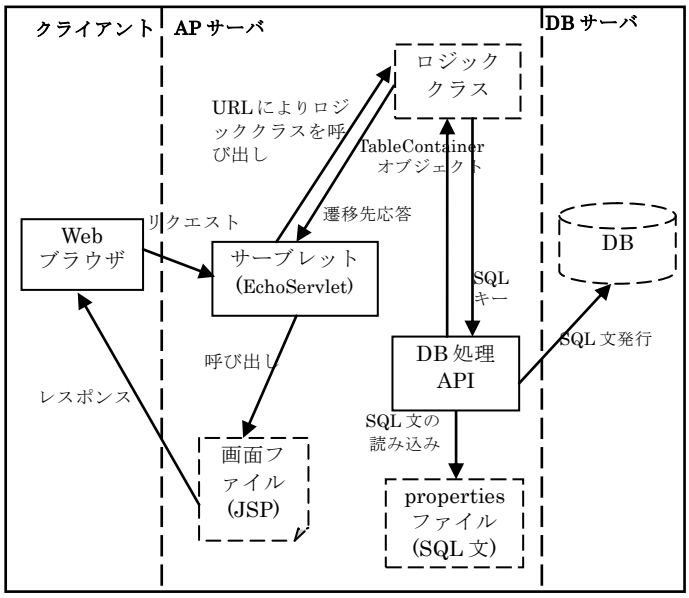

図2. システムアーキテクチャ（バージョン１）

3.3 実装結果

ここまでの研究では、2つのバージョンの例題アプリケーションを実装した。バージョン1は、図２に示すように、EcoFWを用いて不用品交換システムの実装を行った。バージョン2は、バージョン１の例題アプリケーションを書き直し、ドメイン依存部分の画面ファイル(JSP)とロジッククラスがドメイン非依存となるように実装した。

バージョン1のドメイン依存部分においては、1画面に対して、１つのJSPと１つのロジッククラスが対応するというルールで作成した。また、業務用のSQL文を事前に作成し、propertiesファイルに格納した。バージョン1の不用品交換システムの実装内容を表１に示す。

表1. バージョン1の実装内容（ドメイン依存部分）

内容	JSP	ロジック クラス	Properties ファイル
数	10	10	2

バージョン２では、バージョン1のJSPファイルに対して、JavaScriptを使って、HTMLをダイナミックにさせるようにした。そして、ロジックから画面へ渡す情報はJSON形式データを使うことにした。その理由は、JavaScriptのAjax技術を用い、JSON形式データを簡単に解析できるようするためである。このように、AJAX と JSON技術を用いたバージョン2のシステムアーキテクチャを図3に表す。

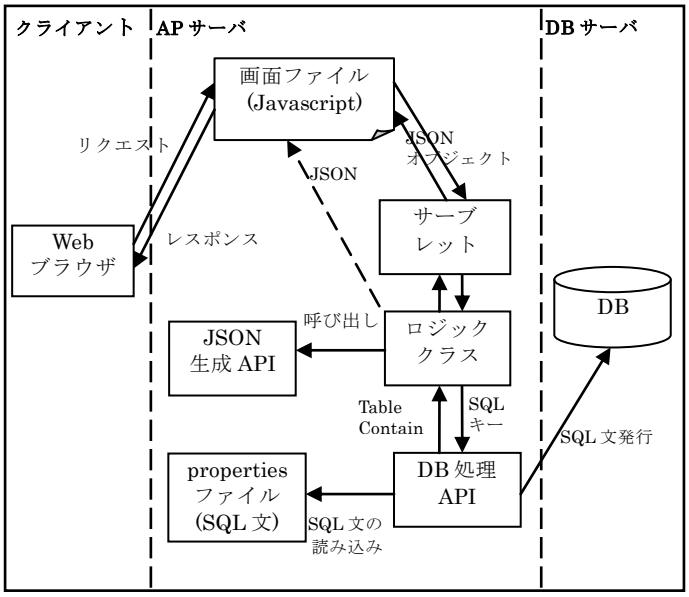

図3. システムアーキテクチャ（バージョン２）

従来、情報の表示・出力を司るビューを構築するのには、HTML言語を用い、<table>、<input>などのHTMLタグを使ってきた。バージョン2の実装では、JavaScript言語を用いてHTMLタグを自動的に生成することにより、HTMLタグを使わなくてよいようにした。

次の例は、各画面が備えるメニュー画面へのリンクに関するバージョン１とバージョン２の実装を示す。バージョン１の実装例で記述されている"メニュー画面へ"、"menuLink"、"menu.jsp"、"body"を、バージョン2でのコンポーネント_LINK_Cの引数として渡して実行すると、バージョン1と同じようなHTMLタグを生成することができる。バージョン２で作成したコンポーネントはJSファイルに格納しており、他の画面にも共通に使われる。

実装の例：

・バージョン1でのHTMLタグ

```
<body>
 <a href="menu.jsp" name="menuLink ">
                              メニュー画面へ</a>
</body>
```

・バージョン2でのHTMLタグを生成するコンポーネント

```
var _LINK_C = function(value,name,url,parentE){
    this.init = function(){
      var dom=document.createElement("a")
     var elementLink = parent.appendChild(dom)
     elementLink.href = url
      var txt=new _LABEL_C(value,elementLink)
      txt.init()
    }
}
```

ロジッククラスでは、データベースへの操作結果を整理して、画面へ渡すのが主な処理である。バージョン１では、データベースへの操作結果を固定形式のJavaオブジェクトに整理した。バージョン2では、JavaオブジェクトをJSONオブジェクトに変換し、画面に渡すようにした。JSONオブジェクトを生成するAPIが各画面で共通に使われるように作成した。表２はバージョン2で作成したコンポーネントとAPIの数を示す。

表2. バージョン2の実装内容

内容	コンポーネント	API	ロジッククラス	Propertiesファイル
数	13	1	10	2

実際の不用品交換システムで実装された画面の例として、ユーザマイページの画面を図４に示す。ユーザがログインしたら、マイページ画面へ遷移する。この画面では、ユーザが登録した不用品情報および申請した不用品が表示され、変更・取消の操作を行うことができる。

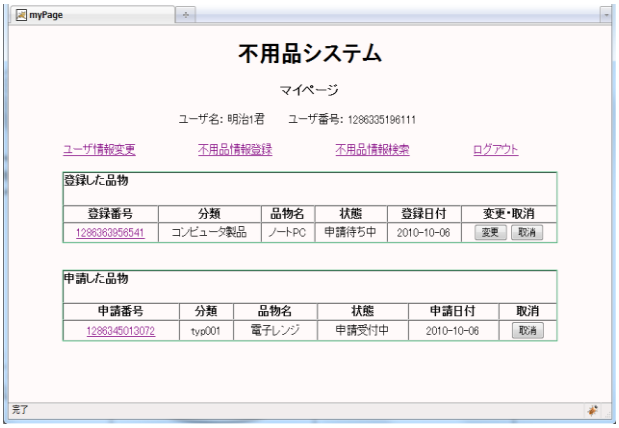

図4. 不用品交換システムのユーザマイページ画面

3.4　アプリケーションに関する考察

例題アプリケーションのバージョン１はシンプルであるが、上述したシステム機能を全部備えており、現在実用性について試用評価している。汎用的フレームワークの代わりにEcoFWを開発し、ビューとモデルを分離したことにより、システムアーキテクチャを理解しやすくなり、保守の容易性もアップさせたと考えられる。

例題アプリケーションのバージョン２では、システム機能は拡張しなかったが、バージョン１に比べてドメイン非依存の割合を増加させた。表３は、例題アプリケーションの実装におけるバージョン１とバージョン２のステップ数及びバージョン１に対するバージョン２のステップ数の減少量の割合を示す。表からわかるように、バージョン２はバージョン１と比べると、ドメイン依存のJSPの記述ステップ数が51%減少し、ロジッククラスも43%の削減量になった。

表3. 例題アプリケーション実装におけるステップ数

	ドメイン依存			ドメイン非依存		
	JSP	ロジッククラス	Properties ファイル	EcoFW	JS ファイル	API
バージョン１	624	1447	422	1438	0	0
バージョン２	308	831	422	1438	769	389
減少量	316	616	0	-	-	-
減少量の割合	51%	43%	0	-	-	-

4　フレームワークの抽出

4.1　ドメイン非依存部分の抽出

本研究では、バージョン２で実装した例題アプリケーションをコンポーネント化し、ビュー部分とモデル部分からドメイン非依存部分を抽出した。

ビュー部分に関しては、上述したように各画面でHTMLタグを自動的に生成するコンポーネントを作成した。そして、フレームワークを構築するために、作成したコンポーネントの汎用化を試みた。汎用化の方法としては、コンポーネントに次の３つの属性を付加してコンポーネントの実体と属性を分離し、実体が呼び出される時に属性を設定できるようにした。

コンポーネントの属性：

・data
　→コンポーネントに差し込むデータ

・css
　→コンポーネントの位置、サイズなどの情報

・events
　→コンポーネントのイベント

例えば、HTMLの<input type="text"/>タグを生成するコンポーネント_INPUT_TEXT_Cの場合、以下の属性

css属性：
　width: 100px　height: 30px

data属性：
　value: 入力欄テスト

events属性：
　onchage:changColor(red)

を設定して実行すれば、次の HMTL タグ

```
<input type="text" width="100px " height="30px"
  value="入力欄テスト" onchange=" changColor(red)"/>
```

を生成することができる。

モデル部分に関しては、バージョン 1 での各ロジッククラスが、データベースへの操作結果を JSON オブジェクトの形式で画面へ返すようにした。各ロジッククラスで SQL 文発行の結果である Java オブジェクト TableContainer を JSON オブジェクトへ変換した部分を抽出し、共通に使われる API を作成した。

4.2 フレームワーク化の考察

ビュー部分をコンポーネント化することによって画面ファイル(JSP)のステップ数が大幅に削減したことを確認した。さらに、コンポーネントを汎用化することで、その他のドメインの画面構築にも適用可能とした。

・業務 GUI(画面)とデータベースをマッピングする。

5 おわりに

本稿では、エンドユーザ向け Web アプリケーションフレームワークの研究の一環として、不用品交換システムを例題とするアプリケーションを試作し、考察及び評価を行った。バージョン 1 の例題アプリケーションは、基本的なシステム機能を実現したうえで、フレームワーク EcoFW をドメイン非依存部分として分離した。さらに、バージョン 2 の例題アプリケーションを実装することによって、画面ファイル(JSP)をコンポーネント化させ、ドメイン非依存性を実現した。そして、JSON オブジェクトを使うことによって、ソース量をかなり削減できることを確認した。

今後は、ロジッククラスで残しているドメイン依存部分のさらなる非依存化を検討するとともに、図 5 に示したエンドユーザへの支援方式を実現するためのビジュアルツールを構築していく。

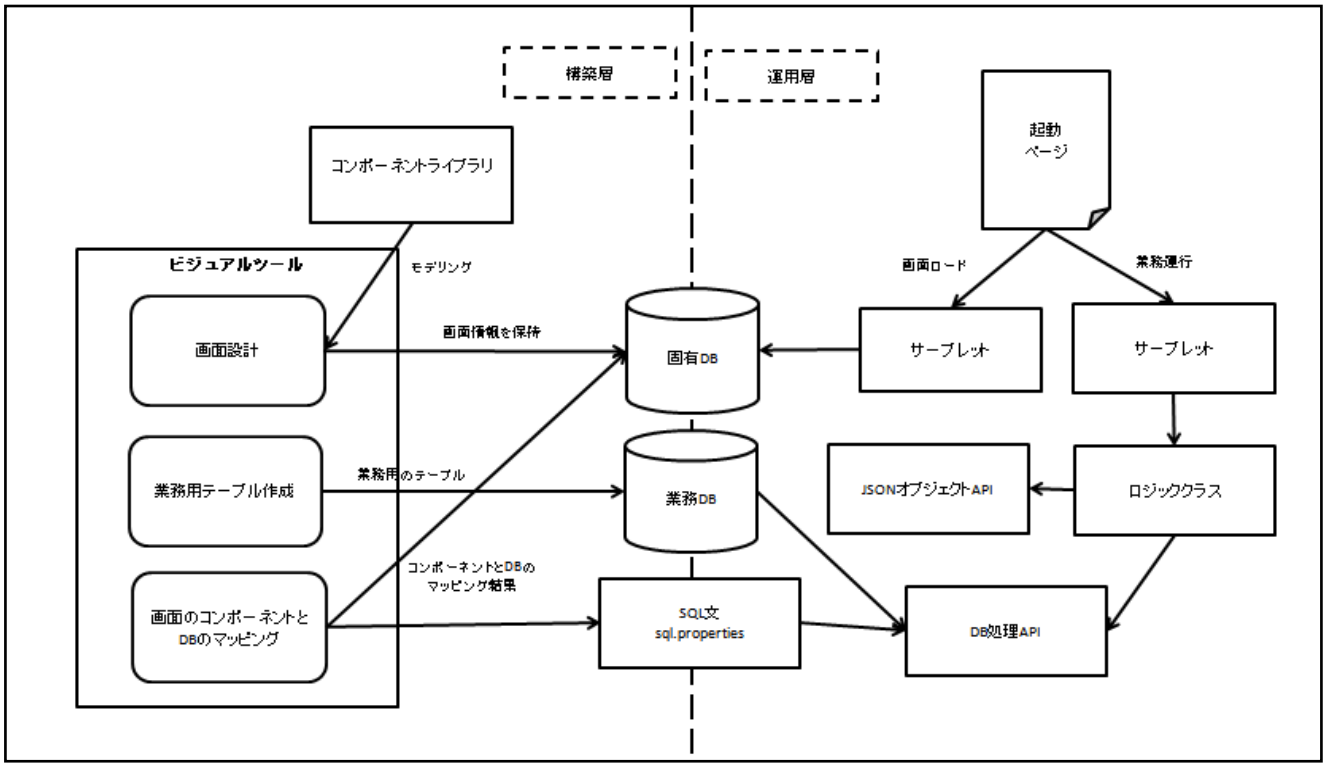

図5. エンドユーザへの支援方式

モデル部分に関しては、一部のビジネスロジックを JSON オブジェクトへ変換することにしたので、ロジッククラスがより軽くなった。なお、ドメインに依存しているロジックを非依存にするのは難しい面もあるが、ロジック系のコンポーネントの作成も検討している。

さらに、上述したコンポーネントと API を結合してエンドユーザを支援する方式については、以下の項目に関して検討した。その結果を図 5 に示す。

・業務 GUI(画面)を設計する。

・業務用のデータベースを作成する。

参考文献

[1]中所武司, “業務の知識を有するエンドユーザ主導のアプリケーション開発技法　～フレームワーク・ドメインモデル・サービス連携～”, 電子情報通信学会 技術研究報告 Vol.107, No.331, 知能ソフトウェア工学研究会 KBSE2007-30, 19-24(Nov. 2007).

[2]中所武司, “抽象フォームを用いたエンドユーザ主導の要求定義法”, 情報処理学会 ウィンターワークショップ 2008・イン・道後 論文集、シンポジウムシリーズ Vol.2008, No.3, pp.63-64（Jan. 2008）

[3]“Ajax”, http://ja.wikipedia.org/wiki/Ajax,（2011.6.27 参照）.

[4]“JSON の紹介”, http://www.json.org/,（2011.6.27 参照）