社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

ドメイン知識に基づくエンドユーザ向け Web アプリケーション フレームワークの試作

李 静† 中所 武司‡

明治大学大学院理工学研究科 ソフトウェア工学研究室
E-mail: †libing_anne@hotmail.com, ‡chusho@cs.meiji.ac.jp

あらまし 近年のインターネットの普及により、様々な業務が Web アプリケーション化されている。それに伴い、Web アプリケーションを効率的に開発するための基盤として、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。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開発のための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 Web アプリケーションを構築することをめざしている。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に関する研究アプローチとその試作について報告する。

キーワード エンドユーザ、Web アプリケーション、フレームワーク、ドメイン知識

Web Application Framework for End-User-Initiative Development with Domain Knowledge

Sei Ree† Takeshi Chusho‡

Software Engineering Laboratory, Graduate School of Science and Technology, Meiji University
E-mail: †libing_anne@hotmail.com ‡chusho@cs.meiji.ac.jp

Abstract In recent years, many web applications are used in various business fields. These web applications are often developed on the base of framework with reusable components. Furthermore, it is increasingly required that business professionals build their web application by themselves. In this paper, web application framework for end-user-initiative development is proposed to support small and medium businesses. Then, the web application framework and a visual tool for the end-user are built.

Keyword End-User, Web Application, Framework, Domain Knowledge

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[3]。次に、フレームワーク作成の基盤とするために、Ajax[4]と JSON[5]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。

さらに，このフレームワークを用いて、エンドユーザが作成した外部仕様からアプリケーションに変換する機能を実現するために，エンドユーザを支援するビジュアルツールを開発する.その詳細は第 5 章で述べる。

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。

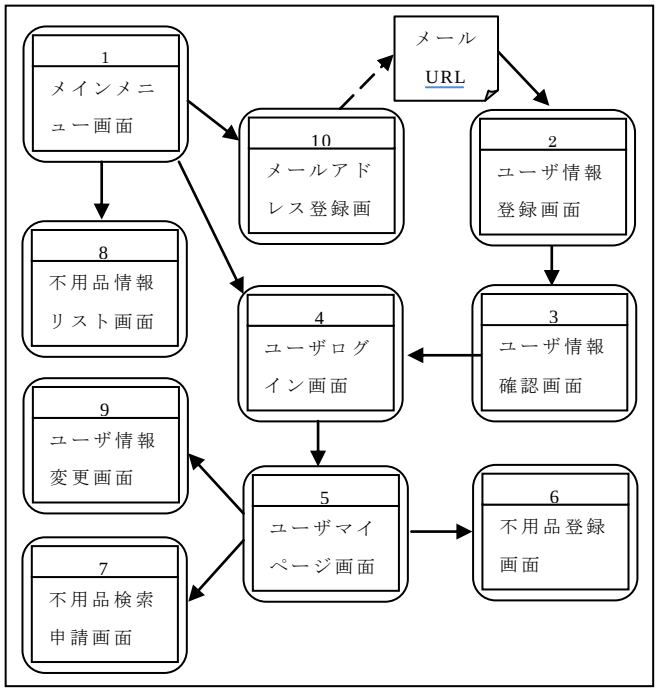

図1. 不用品交換システムの画面遷移図

—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。

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。

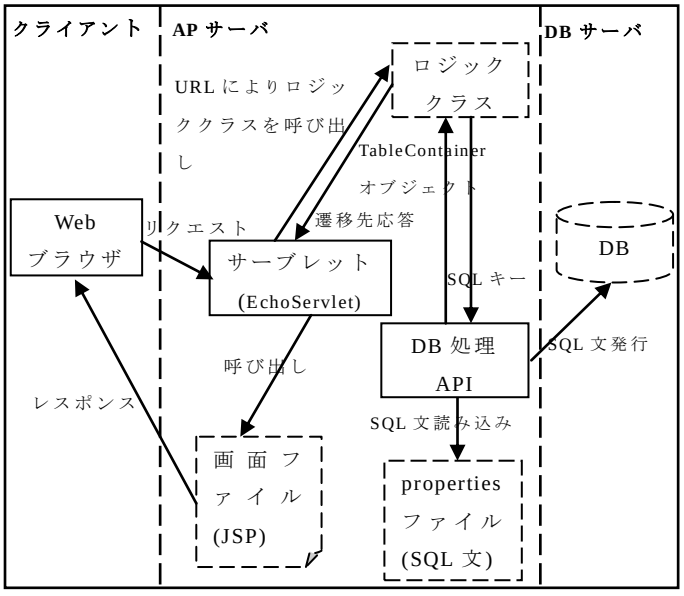

図2. 不用品交換システムの構造(バージョン１)

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）

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をダイナミックに作成できるようにした。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。

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。

具体的な例として，各画面が備えるメニュー画面へのリンク機能に関するバージョン１とバージョン２の実装を以下に示す。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でリンク機能を果たす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でリンク機能を果たすコンポーネント

```
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。

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、ユーザのマイページ画面を図３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。

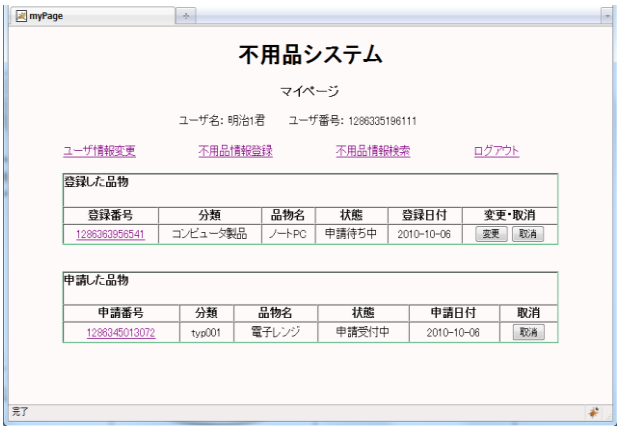

図3. 不用品交換システムのマイページ画面

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２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2. 例題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　top:10px　left:10px
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" style="width:100px;top:10px;left:10px"
 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。図４は、不用品交換システムのバージョン２のアーキテクチャを表す。図４で示すように、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するようにした。また、バージョン２の各ロジッククラスがバージョン１より軽量になるため、Java オブジェクトから JSON へ変換する部分を抽出して共通に使われる JSON 生成 API を作成した。

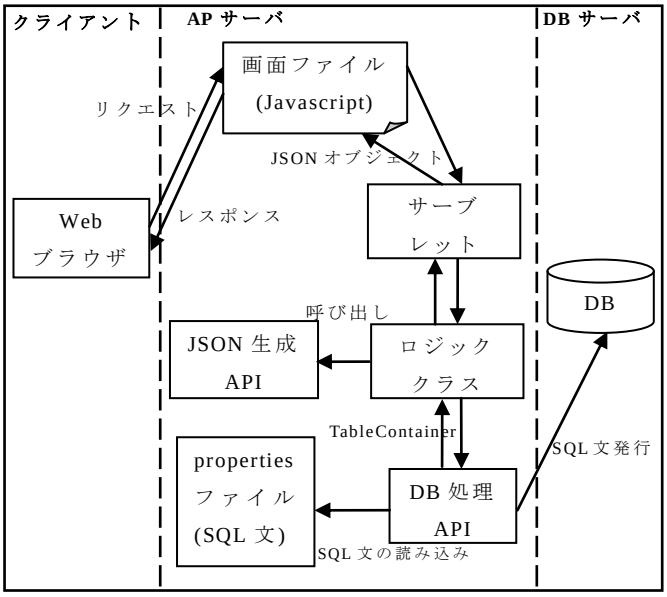

図4. 不用品交換システムの構造(バージョン２)

4.2. フレームワーク化の考察

ドメイン非依存部分の抽出によって、フレームワークの構造をおおむね決定できた。

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。

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。

5. エンドユーザへの支援

5.1. 概要

例題アプリケーションのバージョン１を実装するために，その外部仕様の設計を行った。業務用のテーブル定義書、図３に示したような画面の詳細設計書、

図１に示した画面遷移図、図２の properties ファイルに格納するビジネスロジック用の SQL 文を作成した。エンドユーザ主導開発では、これらの外部仕様をエンドユーザ自身が作成しなければならないので、そのためのビジュアルツールを作成することとした。このビジュアルツールは、業務用の DB 設計、GUI 仕様設計、ビジネスロジック作成の三つの基本機能を備える必要がある。図５はエンドユーザへの支援方式を示す。次節で、ビジュアルツールのそれぞれの機能を説明する。

出力などの情報を定義した。エンドユーザがビジュアルツールを使う場合は、ページごとにコンポーネントを組み込み，ドラッグ&ドロップの操作で定義する方式とする。そして、コンポーネントごとに属性などの情報を設定する。設定した情報は、JSON ファイルの形式で保存する。即ち、エンドユーザが構築したアプリケーションを運用するときに JSON ファイルが呼ばれ、画面が描画される。ビジネスロジック設計に使用することを考慮して、標準コンポーネントの仕様に基づいてページごとに図５の固有 DB に保存する。

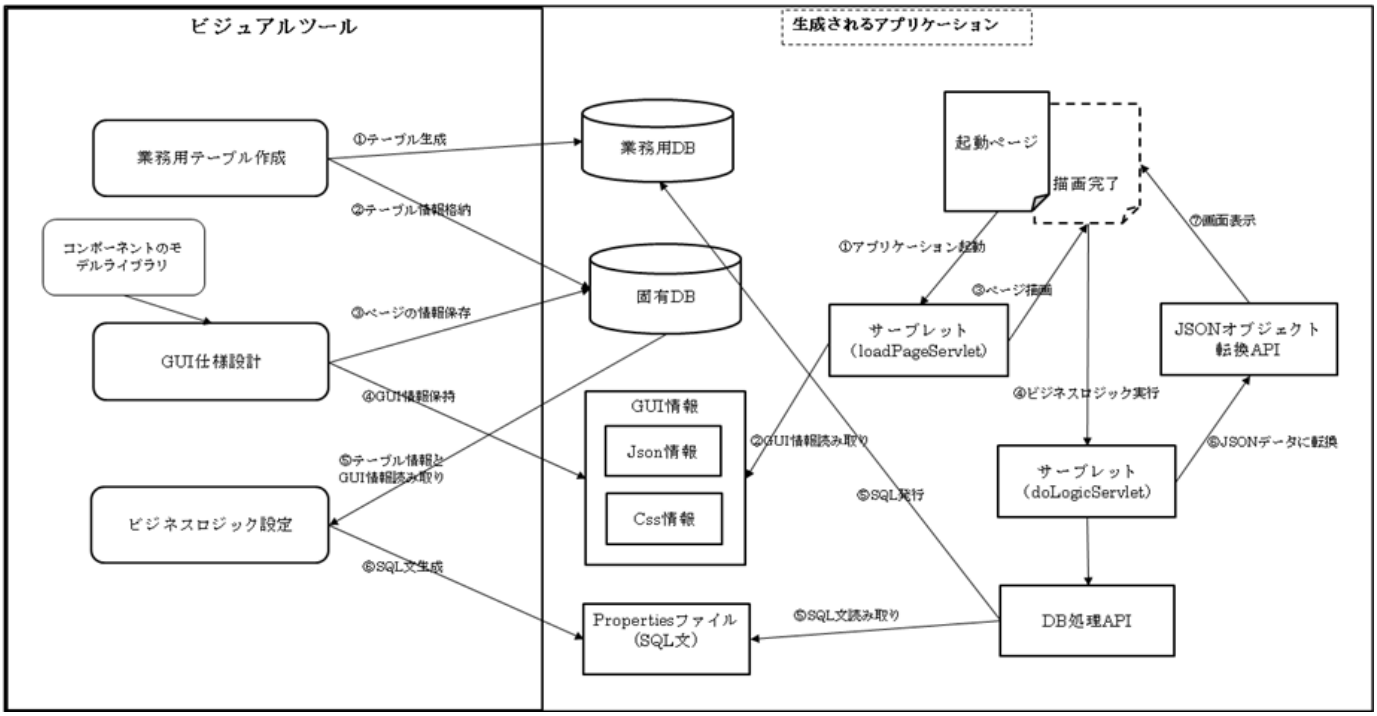

図5. エンドユーザへの支援方式

5.2. DB 自動生成

本研究では、アプリケーションの管理対象に応じて業務用のテーブルを作成する。不用品交換システムの場合は、ユーザ、不用品、不用品申請の３つの管理対象があり、それぞれに対して、必要な管理項目を備えたテーブルを作成した。エンドユーザはビジュアルツールを用いて管理対象を設定して、管理項目の追加および型の設定ができる。管理項目の型は使用している RDBMS の型により提供される。エンドユーザの設定情報に基づいて SQL テーブル操作コマンドを使ってテーブルの生成、削除、更新を行い、DB 自動生成の機能を実現する。同時に、テーブル情報を図５の固有 DB に保持して、ビジネスロジックに使うことになる。

5.3. GUI 仕様設計

例題アプリケーションの画面詳細設計書では、画面ごとに，該当画面の項目、項目属性、表示形式、入力/

また、不用品交換システムの例題アプリケーションの開発を通じて、多くの機能を備えた標準コンポーネントを抽出して実装した。図６は、標準コンポーネントのイメージを示す．パラメタ設定、イベント配置、情報表示の３つの機能により構成され、必要に応じて取捨選択すればよい。

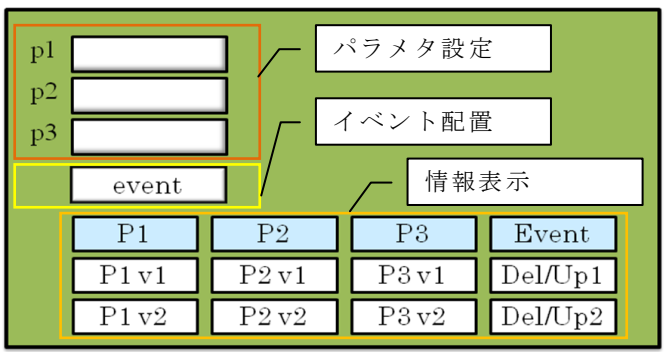

図6. 標準コンポーネントのイメージ

5.4. ビジネスロジック仕様実現

ビジネスロジックに関しては、バリデーション機能、DB 操作機能、画面遷移とメール送信などのそれ以外の機能を分けて実現する。ビジュアルツールによる支援機能を検討した結果、ビジネスロジックの支援が DB 自動生成と GUI 仕様設計支援より難しいことが判明した。よって、本研究では、ビジネスロジックに関して２段階に分けて実装することとした。第 1 段階では、バリデーション、DB 操作、画面遷移を実装する。また、第 2 段階でメール送信などのドメイン依存の機能を加える。

本研究では、関連項目のリレーションシップの設定を行うことによってビジネスロジックを実現する。エンドユーザがビジュアルツールを用いて、コンポーネントのパラメタとチェックタイプのリレーションシップを設定してバリデーション機能を実現する。DB 操作機能の場合は、イベント、パラメタ、テーブルカラムの関連付けを行う。設定した情報は SQL 文の形式で保持される。図５で示すように、構築されたアプリケーションを運用する時にはパラメタを SQL 文に差し込んで DB 操作機能を実行する。画面遷移は、イベントとページの関係を設定する。

5.5. 考察

エンドユーザへの支援方式を示す図５は、左側がアプリケーション開発支援のビジュアルツールで、右側が生成されたアプリケーションの部分である。アプリケーション開発支援方式では、ビジュアルツールを用いて、アプリケーションを実現するための DB 設計、GUI 仕様設計、ビジネスロジックの設定を行う。エンドユーザの操作しやすさを考慮して、ドラッグ&ドロップで GUI 仕様を設計するようにする。そして、GUI 標準コンポーネントを抽出することによってエンドユーザが GUI 仕様設計をしやすくなる。リレーションシップを設定する形式でバリデーションと DB 操作機能を実現することにより、エンドユーザが理解しやすくなる。

図５の右部分は、エンドユーザが構築したアプリケーションを実行する環境である。アプリケーションは、起動ページにより起動され、システムの保持情報を利用しながら、アプリケーションの各機能を実行する。即ち、フレームワークは、エンドユーザが設定した情報を解釈することによってアプリケーションの実行を行うといえる。また、生成されたアプリケーションの保守の多くは、保持した情報だけの変更でよいので、アプリケーションをはじめから作り直す必要はない。

6. 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。

また、エンドユーザを支援するためのビジュアルツールを検討した。第 1 段階で各機能の実現方式を定めた上で実装を行っている。

今後は、ビジュアルツールの第 1 段階の実装を終わらせるとともに、第 2 段階のドメイン依存のビジネスロジックについて検討していく。

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] 李静, 中所武司, “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,” FIT2011 第 10 回情報科学技術フォーラム, B-013, Sep. 2011.

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

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