JSFによるリッチクライアント開発の検証
－JSFはWebアプリケーション開発を容易にする－

上段　左から　和田、越田、村木
下段　左から　清原、伴

技術部
担当部長

伴 達夫

Tatsuo Ban

金融システム開発本部
金融システム第２開発部
ITエンジニア

越田 裕美

Hiromi Koshida

ユビキタスソリューション事業部
ユビキタスソリューション部
ITエンジニア

村木 信也

Shinya Muraki

金融システム開発本部
金融システム第２開発部
ITエンジニア

清原 大心

Daishin Kiyohara

ユビキタスソリューション事業部
ユビキタスソリューション部
ITエンジニア

和田 宗靖

Muneyasu Wada

JSF（JavaServer Faces）は、Webアプリケーション開発におけるプレゼンテーション部分の操作性向上と、開発時の生産性向上を目標に策定された仕様である。この仕様に基き各ベンダーが実装をし、開発環境とあわせた製品を提供しはじめた。今回私達は実際に開発環境を用いてデモシステムを構築し、JSFの実用性を検証した。

この検証は、日本IBMが主催するｊStart（jumpStart）合同研究会に参加するかたちで実施された。参加企業は、当社以外に下記の３社である。このｊStart合同研究会の主旨は、お客様に最新技術をいち早く、かつ実際に検証されたものとして届けることである。

本活動を通して、JSFはお客様への提案において有用であることを検証できた。また、デモシステムの開発により、JSFを使用する際に役立ついくつかのノウハウも得られたので、それも合わせて報告する。

ｊStart合同研究会参加企業　　日本IBM株式会社
株式会社アイ・ティ・フロンティア
アドソル日進株式会社

1. はじめに

昨今、従来のシンクライアント・ファットクライアントに加えて、リッチクライアントという言葉をよく耳にするようになった。JSF（Java Server Faces）はリッチクライアントであるという説明が多いが、JSFをクライアント側からのみ説明するのは間違いである。

JSFは、MVCモデルに沿ったフレームワークであり、JCP（Java Community Process）によって標準化されている。JSFは、主にプレゼンテーション部分の機能が豊富であり、ベンダの提供する開発ツール上でVisual Basicのような直感的な開発をすることが可能である。

JSFからは、以下のような機能が提供されている。

（1）JavaBean管理
（2）ナビゲーション（画面遷移定義）
（3）バリデーション（入力値の妥当性検証）
（4）コンバージョン（入力値の型変換）
（5）イベントハンドリング
（6）国際化

JSFがリッチクライアントといわれる理由の一つは、機能が豊富な標準コンポーネントや標準バリデータを利用することにより、ブラウザ上で動作するWebアプリケーションのEoU(Ease of Use)を高めることができるからである。また、標準コンポーネントで足りない機能は、カスタムコンポーネントを作成することで対応できる。JSFではコンポーネント作成手順が決められているので、標準コンポーネントと同じ構成のコンポーネントを容易に作成することができる。よってEoD(Ease of Development)や生産性・メンテナンス性という面においても大きなメリットが得られる。JSFの出力に関しては、HTMLのみに限らないため、携帯電話やPDAなどに対応することも可能である。今後、システム構築においてプレゼンテーション部分にJSFを利用することが増えてくると考えられる。

今回のｊStart合同研究会では、現実に即したデモシステムを開発し、開発者の視点でJSFがどのように使用でき、今後プロジェクトで利用できるかを検証した。また、今後の開発で役立つと想定できる機能をカスタムコンポーネントとして作成した。

2. JSFの全体構成

Webサイトには、エンドユーザに表示するいろいろな画面が用意されている。

エンドユーザは、画面から必要な入力を行い、システムとの対話を行う。対話の内容は、HTTPリクエストとしてJSFフレームワークが動作しているサーバに送られる。

JSFフレームワークでは、Facesサーブレットと呼ばれる単一サーブレットがすべてのリクエストを受け付けて処理を開始する。この部分をコントローラと呼ぶ。サーバ上では、要求されたリクエストに応じて、JavaBean値の変更や押されたボタンに対応したアクションの実行をする。アクションによっては、バックエンドシステムに接続する場合もある。この部分をモデルと呼ぶ。次に、画面遷移の定義に基づき、画面を生成しHTTPレスポンスとしてエンドユーザに返送する。この部分をビューと呼ぶ。

これらの処理の流れを図１に示す。

開発者は、JSFカスタムタグをJSP(JavaServerPage)内に記述することでJSFを使用することができる。JSFカスタムタグにはJSPに配置するUIコンポーネントや入力データをチェックするバリデータが標準として用意されている。もし、設計要件からそれらで足りない場合は、カスタムコンポーネントやカスタムバリデータとして追加するこ

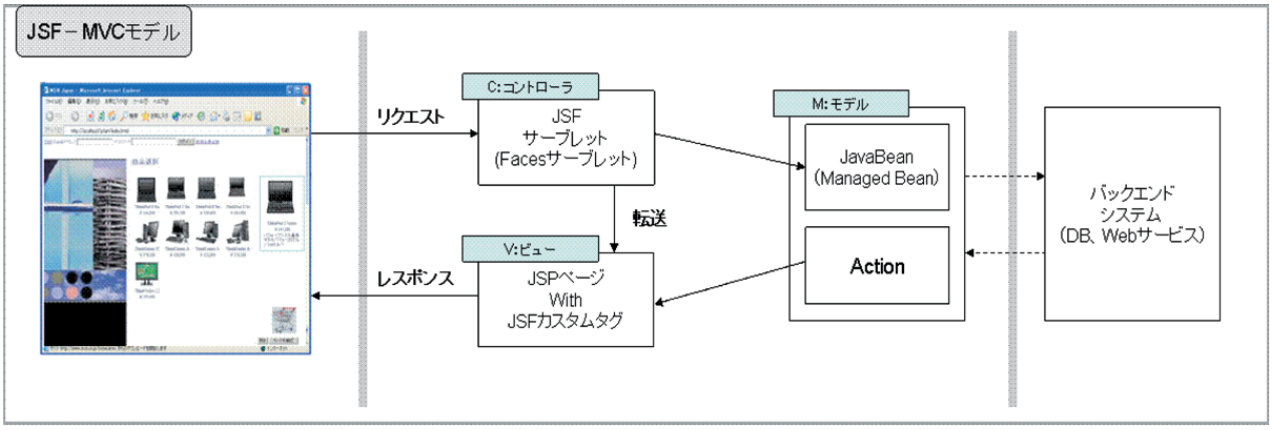

図１　処理の流れ

とができる。IBM WSAD（WebSphere Studio Application Developer）には、標準コンポーネントの拡張としてFacesClientというカスタムコンポーネントも予め用意されている。

標準コンポーネントに加えて、拡張コンポーネントをGUI上で配置・設定できるかどうかは、開発ツールに依存する。今後、各社から、いろいろな工夫の入った開発ツールが提供されると予想される。このことから開発ツールの使いやすさが、JSFの評価に大きく影響する。

フレームワークの処理フローを図２に示す。

アプリケーション開発時に開発者が作成および設定する必要のある構成要素（英語で表記）を以下に示す。

- JSP（画面）
  - JSFカスタムタグを持ったJSP。UIコンポーネントが配置されている。
- Configuration File
  - 画面遷移定義などを記述するXMLファイル。
- ManagedBeans
  - モデル部分に使用するJavaBean。UIコンポーネントとバインドしている。
- Event Action Program
  - イベントに対応して処理を行うJavaプログラム

もし、アプリケーション独自に必要なカスタムコンポーネントやカスタムバリデータがあれば、それらを処理フローに組み込むことができる。カスタムコンポーネントは、JSFカスタムタグを通じて図２のJSF　Libraries/Tagsに追加され、カスタムバリデータは、図２のValidatorsに追加される。

3. JSFのライフサイクル

JSFのライフサイクルについて説明する。JSFでは、画面の構成情報をサーバ内部にコンポーネントツリーと呼ばれるツリー状のデータとして保持している。ライフサイクルの中で、このコンポーネントツリーを使用して、リクエストを受け取ってからレスポンスを返すまでの処理を行う。このJSFのライフサイクルを図３に示す。

ライフサイクル内で実行される個々の処理をプロセスと呼ぶ。プロセスは、コンポーネントツリーを利用して処理をする。各プロセスの処理内容は次のようになる。

（１）Restore View（Viewの復元）
画面に対応するコンポーネントツリーを復元する。

（２）Apply Request Value（リクエスト値の適用）
画面から入力されたデータをコンポーネントツリー

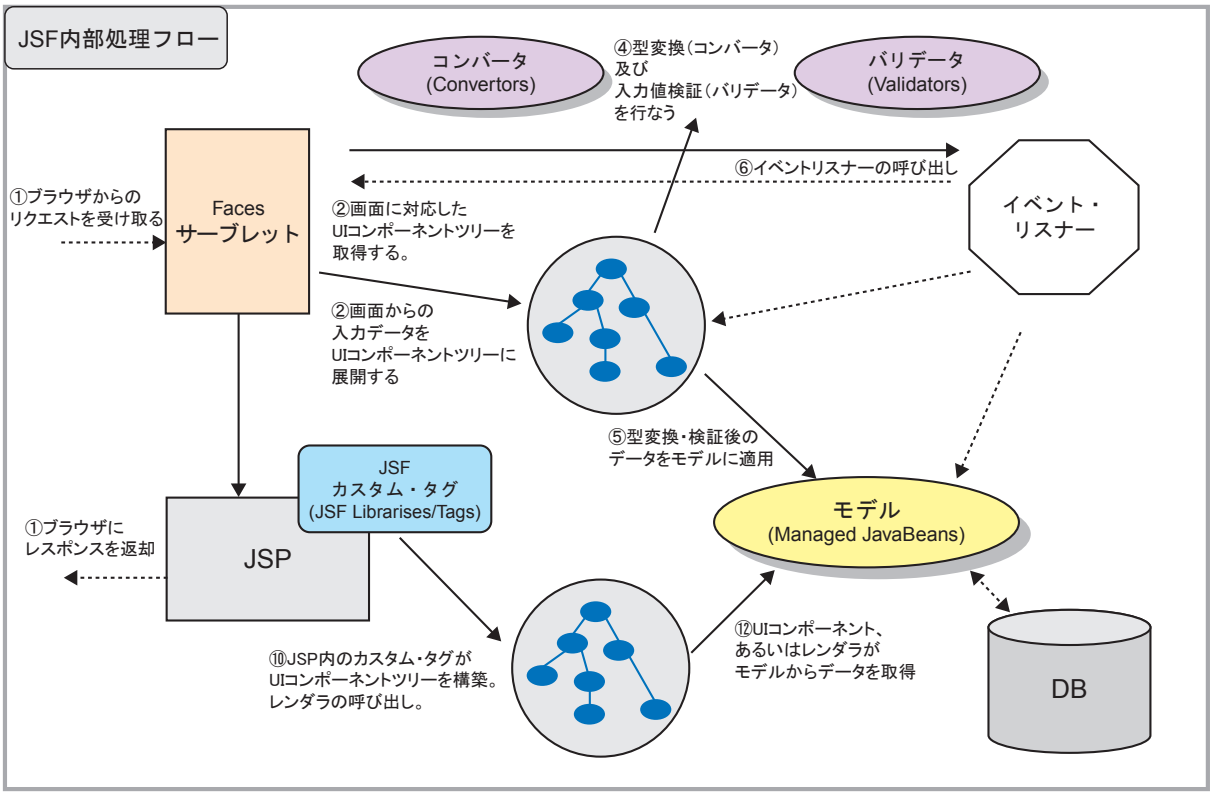

図２　JSF内部処理フロー

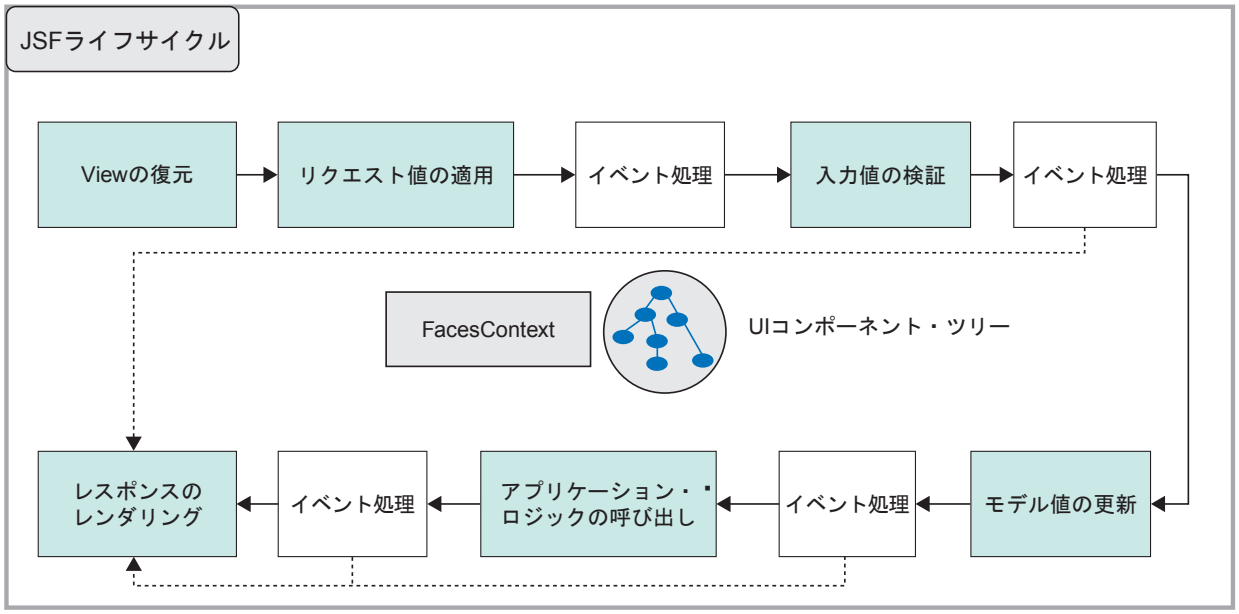

図３　ライフサイクルのプロセス

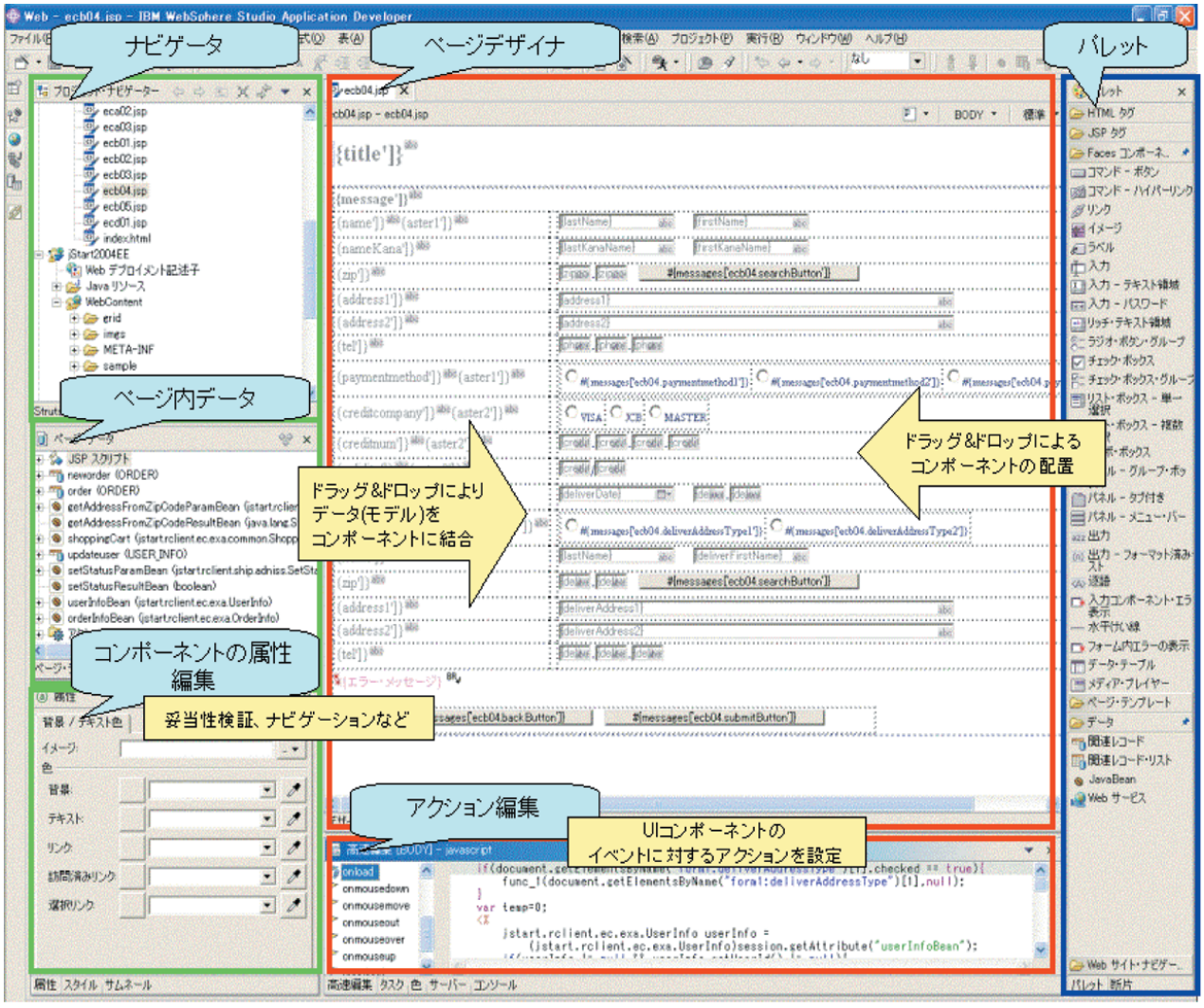

図４　開発ツールの画面

に適用する。

（３）Process Validation（入力値の検証）
入力データに対して､コンポーネントタグで指定したバリデータを呼び出し、妥当性チェックをする。

（４）Update Model Value（モデル値の更新）
コンポーネントツリーのデータをバインドされているモデルに格納する。

（５）Invoke Application（アプリケーション・ロジックの呼び出し）
イベントに対応したアプリケーション・ロジックを呼び出し、遷移先を決定する。

（６）Render Response（レスポンスのレンダリング）
レスポンス画面を生成し、クライアントに返す。

このライフサイクルの中で、標準コンポーネント・バリデータと同様に、カスタムコンポーネントやカスタムバリデータを呼び出すことが可能である。

4. 開発ツールを使用した画面作成

現在、いくつかのベンダからJSFを利用できる開発ツールがリリースされている。今回はIBM社のWSAD 5.1.2を使用した。WSADでの開発画面を図４に示す。

このツールは、以下のような手順でJSFに対応したアプリケーションを開発することができる。

（１）UIコンポーネントを画面へドラッグ＆ドロップ
（２）Wizardを使用してManagedBeanを作成
（３）画面へ配置したUIコンポーネントへManagedBeanをバインド
（４）各UIコンポーネントの属性定義
（５）イベント処理の設定
（６）画面遷移の定義
（７）SDO（Service Data Objects IBMの推奨するDBアクセス手法）の作成
（８）Webサービス接続の作成

5. 開発のシナリオ

JSFの仕様では、フレームワークや開発ツールの製作者も含めてJSFとの係わり方を開発シナリオとして例示している。

Webサイトを構築する場合にJSFを使用することを想定して次の役割分担が示されている。

- Page Author（ページ製作者）：開発ツールを使用してページを作成する人
- Component Writer（コンポーネント作成者）：カスタムコンポーネントの作成者も含む
- Application Developer（開発者）：バックエンドシステムを開発し、ＪＳＦフレームワークとの接続仕様を決める人
- Tool Provider（開発ツール提供者）
- JSF Implementor（JSFのフレームワーク製作者）

この例からJSFの目標とするところは、ページ製作者が開発ツールを使用してＷｅｂアプリケーションを開発できることである。従って、次のような推測も可能である。

「ＪＳＦによって開発者にＷｅｂ技術がなくても、ページ製作者にＪａｖａ技術がなくても、Ｗｅｂアプリケーションを開発することができる。」

しかし、現状ではその目標が実際の開発に反映されているとは言いがたい。

先にも述べたが、JSFは仕様であり、実装は各ベンダに任されているのが実状である。そのため、開発ツールとフレームワークの製作者は同一の役割をしていると考えられる。また、開発ツールはプログラムの開発環境に統合されていることから画面作成とコンポーネント作成も同一になることが考えられる。明らかにJSFの掲げる目標に比べて開発者の役目が多くなっている。その関係を図５に示す。

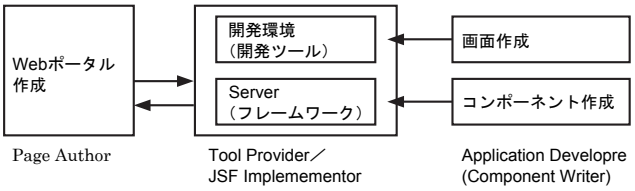

図５　関係者役割分担図

実際の作業分担としては以下のような状況が想定できる。

- Page Author：ページツールを使用してページを作成する人。システムとの接続部分は除く。
- Component Writer：カスタムコンポーネントの数によるが、少なければ開発者の仕事になる。
- Application Developer：バックエンドのシステムと接続をするロジックをプログラム開発環境で作成。そのために必要な画面を含むWebアプリケーション作成。
- Tool Provider／JSF Implementor: 各ベンダが両方をあわせて提供する。

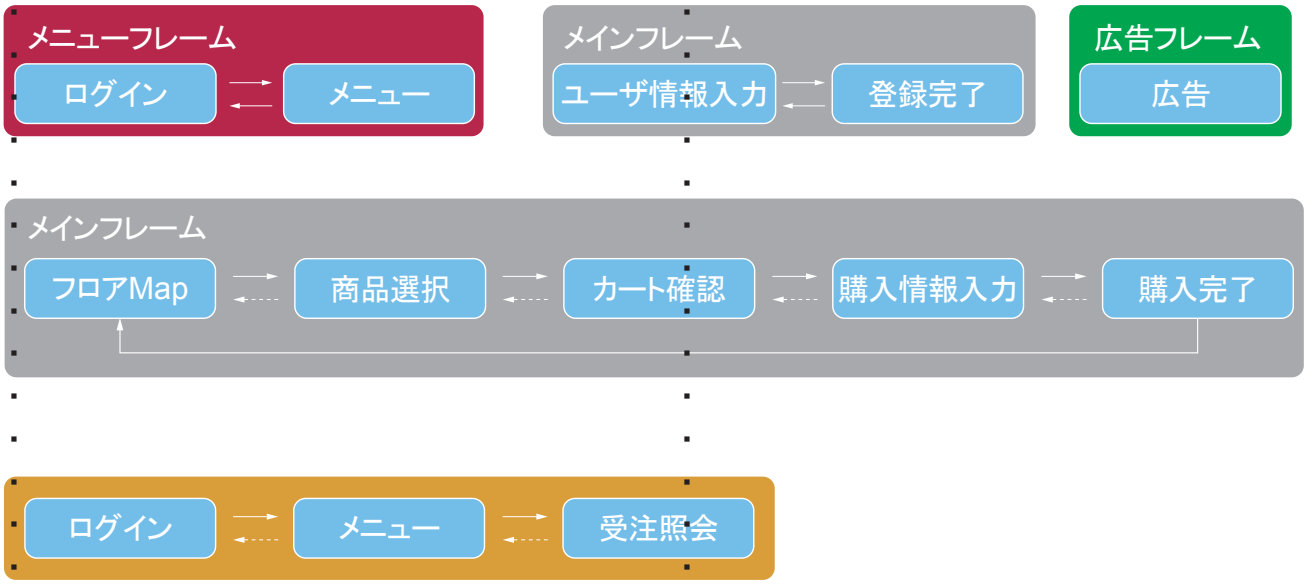

図６　画面一覧

今回のデモシステム開発においては、この開発シナリオの役割分担を想定し、開発者の視点で評価をした。画面設計ができていれば、開発ツール上でページへコンポーネントを配置し、その属性に値を設定し、作成したページの画面遷移を決定するだけでプレゼンテーション部分が開発できることを期待した。

6. デモシステムの作成を通じて

6.1. デモシステム作成の目的

今回の研究会で開発するデモシステム作成の目的は以下のとおりである。

（１）JSFについて知見深化
（２）リッチクライアントとしてのEoUの検証
（３）WSADの提供するEoDの検証
（４）生産性向上の可能性追及
（５）カスタムコンポーネント作成ノウハウの習得
（６）開発ノウハウの蓄積

これらの目的を満たすため、ECサイト構築を選択した。

6.2. 全体構成（画面）

デモシステムの構成は、大きくコンシューマ向けサイトとショップ店員向けサイトに分類される。コンシューマ向けサイトは、ドラッグ＆ドロップなどの入力支援や非同期広告配信などの見た目にもリッチなサイト構成とした。一方ショップ店員向けサイトは、キーボード入力にフォーカスしたサイト構成とした。

表１　画面構成要素一覧

画面内容	WSAD	カスタムコンポーネント	カスタムバリデータ	Webサービス
フレームセット	－			
ログイン	○		使用	
メニュー	○			
ユーザ情報入力	×		使用	使用
登録完了	○			
フロアMap	－			
商品選択	○	使用		
カート確認	○			
購入情報入力	×		使用	使用
購入完了	○			
広告	－			
店員ログイン	○	使用		
店員メニュー	○			
受注照会	○	使用		使用

○：JSF＋JSF IBM拡張のみ
×：JSF＋JSF IBM拡張＋JavaScript
－：HTMLのみ

図６に作成した画面の一覧を示す。また表１には各画面がどのような要素で構成されているかを示す。

表１の中にあるWSADの項目は、WSADの標準的な機能と必要最低限のコードのみで作成できたか否かを表している。本来、すべての画面で○になることが期待されるが、今回のシステムでは×になる画面がいくつか出てきた。原因としては、ライフサイクルの処理プロセス内で、値を保持してほしくないところが保持されていたり、バリデートチェックしてほしくないところでバリデートチェックが走

ったりしたためである。それらを回避するために直接処理とは関係ないJavaScriptを作成する必要が発生した。

6.3. カスタムコンポーネントの作成

JSFでは、UIコンポーネントを継承することにより、容易にカスタムコンポーネントを作成することができる。今回のデモシステムの中で今後利用できそうな機能をカスタムコンポーネント化することにした。作成したのは以下の3つである。

- ドラッグ＆ドロップコンポーネント（商品選択）
- 広告コンポーネント（広告表示）
- NestedDataTableコンポーネント（受注照会）

（1）ドラッグ＆ドロップコンポーネント

DBから取得したデータをコンポーネントに渡すことで商品を一覧表示し、カートにドロップできるようなJavaScriptコードを出力するコンポーネントである。（図7）

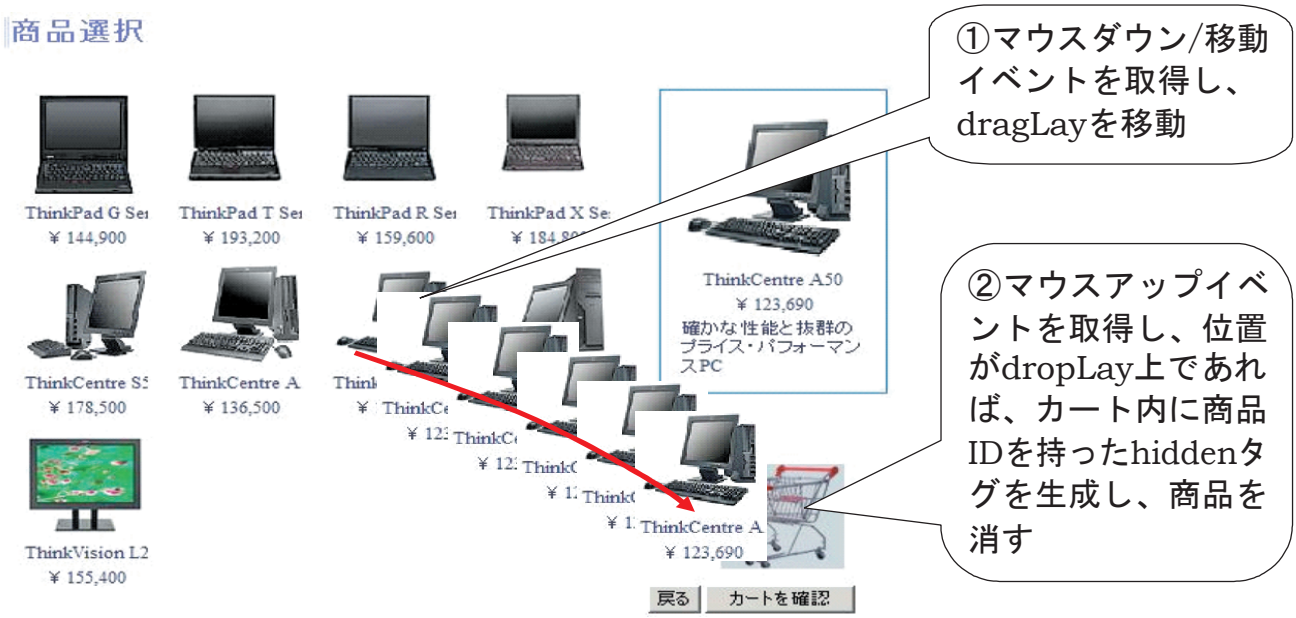

図7　ドラッグ＆ドロップ画面

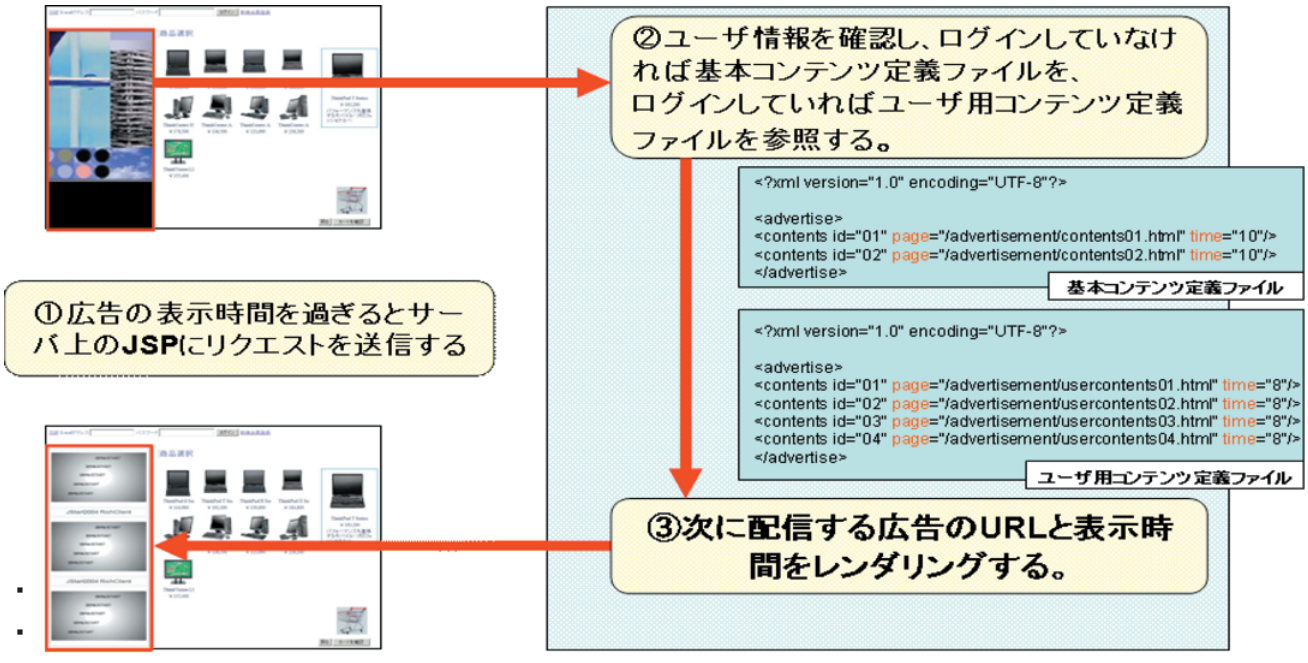

図8　広告表示画面

ドラッグ＆ドロップの機能は、JavaScriptで実装しているため、全ての処理がクライアント側で実行され・れる。これにより商品選択ごとのサーバアクセスが発生せず、遅延のないスムーズな動作が実現できる。

（2）広告コンポーネント

コンシューマのユーザ情報により、最適な広告を配信する仕組みを提供するコンポーネントである。広告は表示時間を過ぎるとサーバにリクエストを送り、サーバ上の定義ファイルに基づき次に表示する広告が決定される。（図8）

この機能はWebページでのバナー広告等に利用可能だと思われる。広告としては、FlashやWindowsMediaなどブラウザ上で表現できるものは、形式に限定なく使用できる。

（3）NestedDataTableコンポーネント

ECサイトにおけるオーダ情報は、複数の明細を持ち階層化されたデータとして存在する。JSF標準のデータコンポーネントでは、このように階層化されたデータを一度に表示することができない。これを実現するために、NestedDataTableコンポーネントを作成した。（図９）

このコンポーネントは、JSFのDataTableコンポーネントのレンダラを拡張することで実現した。

6.4. 開発環境WSAD（WebSphere Studio Application Developer）の評価

今回の開発において実際にWSADを使用した結果、次の点が利点として挙げられる。

（1）ページ上にコンポーネントをドラッグ&ドロップすることで配置できる。

（2）BeanやDB項目（SDOを使用）と画面コンポーネントとのバインドができる。

（3）フレームワークで使用する設定ファイル類も、ツール上の操作で自動生成・編集される。

以上のことから、画面作成の簡便さについて優れていると評価できる。ほとんどの画面についてコンポーネントを使用したGUI操作で開発が可能であるので、システムのプロトタイプ作成には適している。

また、実際の開発でも画面を効率的に開発できる。

一方、今後改善されることも推測されるが、次の点に注意が必要である。

（1）一画面で複数DBのテーブルを使用すると、手動で作成しなければいけないコードが増える。

（2）コードの自動生成・自動編集のタイミング、対象範囲が明確でないため、ページ内でコンポーネントの配置や削除を繰り返すとコードが不正になり手動で

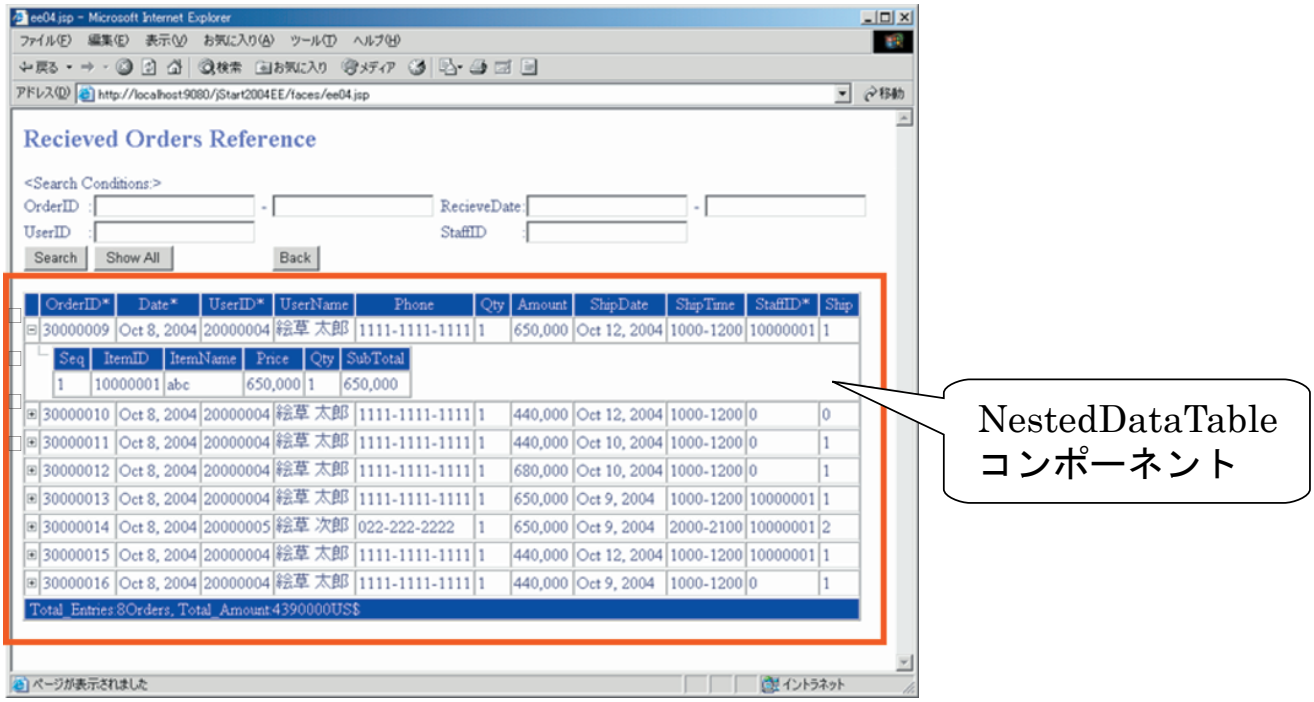

図９　NestedDataTable表示画面

修正するか、一から作り直す必要がでてくる。

（3）PCのスペックによって､動作が遅くなったりフリーズしたりする。PCのスペックがWSADの作業効率に大きな影響を与える。

6.5. 開発の注意点

実際に開発をした結果、JSFの仕様上注意すべき点がいくつか確認された。また、WSADのクセによって意図したとおりの処理が行えない場面もあった。例えば、「GUIで配置したコンポーネントが意図したとおりに動作しない」「画面遷移時に前回入力したデータが意図せずに表示されてしまう」「違うフレームの画面遷移が上手く行えない」といった事象である。このためJavaScriptを記述する必要が生じた。

これらを回避するために設計に関して次の点に注意が必要である。

（1）フレームをまたいだ画面遷移は行わない。

（2）ライフサイクルの流れに逆らう動作はしない。

また、JSFの国際化機能を利用してページを国際化したとき、IEブラウザのロケール設定は、「言語-国」で設定しないと正しく処理されないことにも注意が必要である。（○: ja-jp, en-us　×: ja, en）▪

基本的には、「JSFをしっかり理解し、JSFの仕様に沿った画面設計」をすることが重要になる。またJSFはツールを用いての開発を前提とした仕様であるため、開発ツールに依存する部分が大きい。よって「開発ツールのクセをよく理解する」ことも重要である。

6.6. デモシステム作成の結果

ECサイトを例とするデモシステム作成の結果を、本章の最初に挙げた目的に沿って整理すると次のようになる。

（1）JSFについて知見深化

仕様では良くわからなかったことが、実装された開発ツールや実行フレームワークを使用することで明確になった。その結果、JSFはJava標準として、今後成長、発展していくとの確信が持てた。

（2）リッチクライアントとしてのEoUの検証

既存のＷｅｂアプリケーションと比較して、JSFのUIコンポーネント、バリデータ、レンダラ、SDOを利用または拡張、もしくは仕様にのっとったカスタム化を施して機能豊富な画面を構築できる。

（3）WSADの提供するEoDの検証

ドラッグ&ドロップ操作によって画面作成が可能である。また、データ、DBの項目、Webサービスとの接続も同様の操作で行える。さらにイベントとプログラムの接続もできるなど簡単に開発ができる。しかし、６．４で示したように、開発ツールとしての機能強化は必要である。

（4）生産性向上の可能性追及

Java標準であることから、仕様に述べられている役目分担により、他者による開発の成果を利用することができる。また、ページ編集者と開発者の作業分担により互いの作業効率が向上する。

（5）カスタムコンポーネント作成ノウハウの習得

６．３で示したように、新規や既存コンポーネントの拡張としてカスタムコンポーネントを作成できる。

（6）開発ノウハウの蓄積

６．５で示したように、今回の開発でもいくつかのノウハウを得た。今後、より大きなシステム開発ではさらに増えると想定できる。▪

7. 終わりに

今回使用したIBMのWSAD5.1.2.以外にも、Sunを始めとするベンダはJSFを実装した開発ツールを発表している。今後これらのEoDに注力した開発ツールは、Webアプリケーション開発の主流になる。既に述べたとおり、これらの開発ツールでは、GUI操作で画面の開発が行えるものになっている。このことはJSFが推奨している開発時の分業化をより推進させ、開発効率の向上に役立つものになる。

また、クライアントの視点のみからWindowsなどのリッチなUIと比較することは望ましくない。再利用性や移植性も考慮したシステム構築全体の視点からUIを考えるとき、J2EE標準であるJSFは優位性があると考えられる。

JSFとよく似たものとしてJakarta Strutsが挙げられる。MVCモデルのコントローラ機能が優れるStrutsはWebアプリケーション開発の現在の主流であり、信頼性も高い。しかし、Strutsの開発者であり、JSFの仕様作成者の１人であるCraig Mcclanahan氏が次のように言っている。

（1）新規開発ならJSFを使用する。

（2）既存のStrutsは､JSFへのマイグレーションを考える。

JSFとStrutsは二者択一の選択肢ではない。しかしUI開

発に関しては、ツールのサポートも含めJSFがスタンダードになる可能性はある。

今回のデモシステム作成を終えて、今すぐお客様にJSFを使ったシステム構築を提案できるかといえば、まだ課題は多い。よってJSFに関する教育は当然として、実際の開発をしながら課題を解決して行く必要がある。

以下に課題の一例を挙げる。

（１）標準コンポーネント使用時の注意点や意図した動作にならなかった時の対処ノウハウ蓄積

（２）お客様が要求されるＵＩをカスタムコンポーネントとして作成できる技術ノウハウ蓄積

（３）運用時のチューンアップ技術

（４）開発ツールと実行フレームワークは別会社のもので可という仕様だが、実際には、開発ツールに依存する部分もある。それを回避するために相互運用のノウハウ蓄積

（５）JSFを使用する上での、設計上の注意点

（６）開発に使用する開発ツールの選定・習熟

JSF自身も発展途上（執筆時点の最新仕様はJSF1.1）であり、上記の課題のいくつかはJSFの仕様、または開発ツールの改善などで解決される可能性もある。これらの課題を解決しノウハウとして蓄積していくことにより、さらにJSFはユーザのクライアントプログラムへの要求を満たし、かつ開発効率を向上させる有効な技術となる。

参考文献

（図書）

1) 若尾正樹ほか "はじめてのJSF", 日経BP社, 2004, 300P

2) ハンス バーグステン "JavaServer Faces完全ガイド", オライリージャパン2004, 663P,

3) 川崎克己 "JSFによるWebアプリケーション開発", 秀和システム, 2004, 446P

4) イー・ベンチャーサポート "徹底解説！ ＪＳＦのすべて", 秀和システム, 2004, 182P

（参照サイト）

1) JSFに関するサイト
http://java.sun.com/j2ee/javaserverfaces/index.jsp

2) Craig Mcclanahan氏のコメント
http://www.theserverside.com/news/thread.tss?_id=29068

＜問い合わせ先＞
技術部
Tel 044-540-2128　伴 達夫
E-mail: tatsuo-ban@exa-corp.co.jp
