動的Ｗｅｂコンテンツ作成を支援するフレームワークの開発

A Framework for Dynamic Web Contents Development

西山 明秀
Akihide NISHIYAMA

富士通株式会社[*1] （E-mail: cae15630@pop16.odn.ne.jp ）

ABSTRACT. We developed a framework for dynamic web contents development. This is a report of developments of a framework on improvements of software reusability in Web Applications using frameworks. In developments of Web applications, using frameworks is useful. There are many source codes that can be shared between several projects in Web application. Because a framework provides implementations of common processes in several Web applications, developers of Web applications only implement specific processes for developing application. In this study, we implemented a framework executing business-logic related on the framework. In case of using this framework, it makes architecture of business-logic fixed, so business-logic as software components can be reused easily. It is important that the way of implementation is standardized. Each developer must share the knowledge about the reused software. Because developers can only use types registered in this framework, the interface of business-logic can be standardized in a framework as a small domain.

１．背景

現在、インターネットは誰でも情報を公開し取得できる情報インフラとして普及している。また、ビジネス用途だけでなく、Web サイトを作成し情報を発信する一般ユーザの数も多い。しかし、作成者が更新した時の情報を保持し続ける静的な Web コンテンツが大半である。リアルタイムに情報が更新されるような動的なコンテンツの作成にはプログラミングを行わなければならない事が多い。しかし、プログラミングに関する知識や熟練をすべての Web コンテンツを作成する人に要求するのは現実的でない。

２．目的

そこで、一般ユーザが容易に動的な Web コンテンツを作成できるようなシステムの作成を目標とする。また、現在、注目されている技術として Web サービスがある。Web サービスとはインターネット上のあらゆる場所に存在するサービスをオンデマンドで利用できるようにする技術である。しかし、Web サービスは発展途上の技術であり、その本格的な利用にはある程度の試行が必要となる。本プロジェクトにより、Web サービスを公開できる場を用意することによって利用ユーザ数を増やし、日本国内での Web サービスの普及の支援を目指す。本プロジェクトに Web サービスの利用・公開する機能を付加することによって、インターネット上における Web サービスの利用できる場の普及の支援を目指す。

３．システム概要

本プロジェクトで開発したフレームワークを higherground と名づけた。フレームワーク(higherground)は図１に示すようにビジネスロジックの実行を行う．Web アプリケーションを MVC(Model View Control)2 モデルで開発することを前提にし，他のフレームワークと連携して動作させることを仮定して開発した．MVC2 モデルとは J2EE(Java2 Enterprise Edition)を用いて Web アプリケーション開発のために提唱された開発モデルである。

クライアントからの要求に応じ, 結果を HTML 文書として返すプレゼンテーションロジックは JSP(Java Server Pages)で実装する．特に今回，JSP を処理する Web コンテナとして Tomcat4.1 を利用した．画面遷移を制御するコントロールロジックは Jakarta Struts を利用した．Struts ではビジネスロジックの実装方法を定めていない．したがって，Struts から投げられたビジネスロジックの処理を今回開発したフレームワークが受け取り，ビジネスロジックの実行を行う．本フレームワークはソフトウェア部品として関連図けられたビジネスロジックを実行する．本フレームワークにおいてもソフトウェア部品に対するインタフェースに対する制約を設けているが実際の実装方法に対する制約はない．ゆえに，目的にあった方法で実装を行うことができる．たとえば，ソフトウェア内でデータベースアクセスや，EJB(Enterprise Java Beans), Web サービスの利用なども行うことができる．

*1 現所属。本成果は著者の工学院大学大学院在籍時のものである。

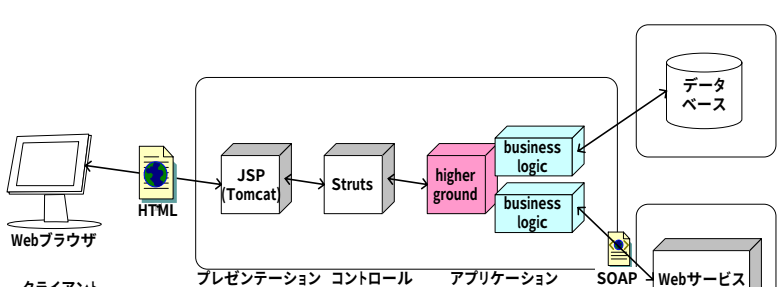

図１　Web アプリケーション全体から見た higherground

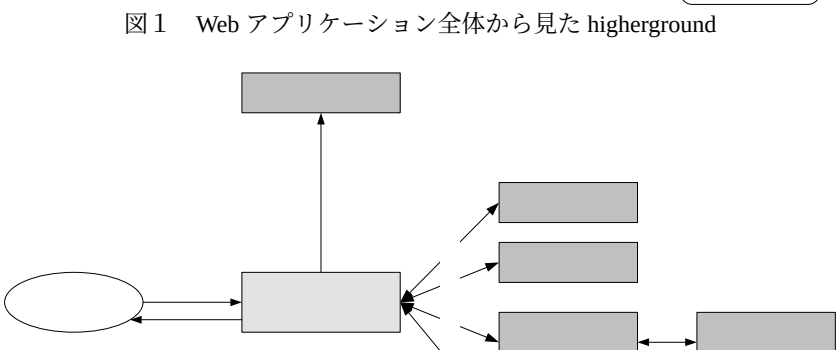

図２　higherground の実行概要

４．システムの実行概要

本フレームワークはプロセス実行の要求が来ると，あらかじめ Web アプリケーション開発者が配置したビジネスロジックの実行手順定義に基づいて，フレームワークに関連付けられているビジネスロジックを実行していく．図２に本フレームワークの実行概要を示す．フレームワークが直接コールするメソッドは普通の Java クラスで実装されていなければならない．EJB により実装されたビジネスロジックや他のサーバ上で Web サービスとして公開されたビジネスロジックを実行したい場合，通常の Java クラスで実装されたラッパークラスを通して実行する．これにより，フレームワークはビジネスロジックの実装方法に依存しないことになり，幅広い実装方法や新技術への対応も図ることができる．

５．システムの内部動作概要

本フレームワークはビジネスロジック間で授受するデータの管理，実行すべきビジネスロジックの決定，実行を行う．図３に内部動作概要を示す．本フレームワークに登録できるビジネスロジックは処理の結果の状態情報を戻り値として返すと定義した．本フレームワークはこの戻り値とワークフロー定義ファイルの情報を基にビジネスロジックの実行を繰り返す．

６．XML 定義ファイルの種類

本フレームワークは XML 文書による定義ファイルに記述された内容に基づいて動作する．本フレームワークで利用する XML 文書は次の 4 種類ある．

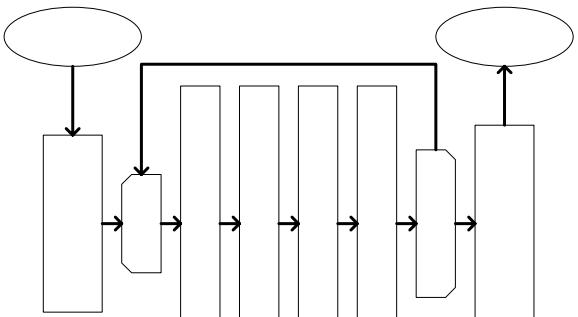

図３　higherground の内部動作概要

（1）バインディング情報定義ファイル

本フレームワークでは型情報を言語から独立して管理する．Web サービスにおいてインタフェースを定義する WSDL(Web Service Definition Language)では XML Scheme で定義された型情報を利用している．そのため，Web サービスとの親和性を高めるために，本フレームワークにおいても XML Scheme で定義された型情報を採用した．その XML Scheme の型と Java 言語の型のバインディング情報をこの定義ファイルで格納する．

（2）型情報定義ファイル

ビジネスロジックのインタフェースで用いる型を定義する．型情報定義ファイルを拡張することで新たな型を作り出すことができる．図４に示すように型は階層関係で定義される．したがって，すでにある型を 1 対 1 で拡張するか，組み合わせて拡張することができる．しかし，型の種類が増えると型の管理が困難となるため，すでに定義されている型を用いることを推奨する．

1. ビジ
の実

（3）サービス定義ファイル
ビジネスロジックを実装する，または，ビジネスロジックを実装するクラスに委譲する Java クラスの情報を格納する．

（4）ワークフロー定義ファイル
ビジネスロジックの実行手順を格納する．

すべての XML 文書では名前空間の利用ができる．名前空間を利用することにより複数の XML 文書間で情報を共有することができる．そのため，複数のファイルに情報を分散して保存できる．

7．ビジネスロジックのインタフェース規約

直接フレームワークから呼び出されるビジネスロジックを実装したサービスのメソッドは次のフレームワークの提供する規約に沿っていなければならない．

（1）メソッドの入力値と出力値は各 1 個のみとする
メソッドの第一引数は入力値が入れられる Java Beans への参照，第二引数は出力値を代入する Java Beans への参照である．そして，各引数の型は DynaBean クラスにしなければならない．DynaBean クラスは Jakarta Commons プロジェクトの beansutil に含まれるクラスである．DynaBean クラスはフレームワーク開発のために作られたクラスであり動的に JavaBeans クラスを生成するユーティリティクラスである．動的にメンバ変数を生成し値を代入できる．DynaBean クラスでは動的な Bean 生成を行うため，誤った Bean のメンバにアクセスしようとすると，コンパイル時ではなくランタイムで例外が発生する．そのため，このクラスの利用には注意が必要となる．

（2）入力値と出力値で用いる変数の型はフレームワークに登録された型のみ利用できる．
複数のビジネスロジックを連携させて動作させたい場，各ビジネスロジック間のデータ連携をどのように行えばよいのかという問題がある．本フレームワークでは，利用できる型を制限することにより連携するデータを取り扱いやすくする．そのため，利用できる型は，定義ファイルに記述された型のみ利用できる．しかし，前述のように入力値と出力値はともに1つしか持つことが許されていないので，新規に型を定義しなければならない場合もある．そこで，定義ファイルにその新たな型情報を書き込めば，新たな型を定義することはできる．しかし，無駄にその方の種類を増やすことは望ましくない．

（3）メソッドの戻り値はビジネスロジックの結果の状態を示す．
ビジネスロジックの実行により得られる結果は前述のように，引数で与えられた変数に代入する．ビジネスロジックを実装したメソッドはそのビジネスロジックを実行した結果の状態を String 型に代入して返す．この戻り値はそのビジネスロジックを実行した後にフレームワークが何を実行すべきか，という情報の一部となる．フレームワークは戻り値と定義ファイルに記述された情報を基に次に実行するビジネスロジック，または，終了すべきかを決定する．

8．カスタムタグの利用

本フレームワークを利用するためには，サーブレットや Struts の Action Class のように Java コードから呼び出す方法が扱いやすい．しかし，短期間で開発される小規模なシステムの場合，JSP からも本フレームワークが呼び出せると便利な場合もある．JSP では Java のコードをスクリプトレットによって直接埋め込むことがきる．しかし，JSP のコードに Java のコードを埋め込むということは, タグの中に Java のコードが入ることになり非常に判読しにくくなる．また，JSP を記述する Web デザイナに Java 言語に関する知識も要求してしまう．

そこで，本フレームワークでは JSP のカスタムタグを利用することでこの問題を解決している．カスタムタグは JSP で使用することのできる新たなタグを開発者が独自に作り出すことのできる機能である．図５に本フレームワークで定義したカスタムタグの使用例を示す．

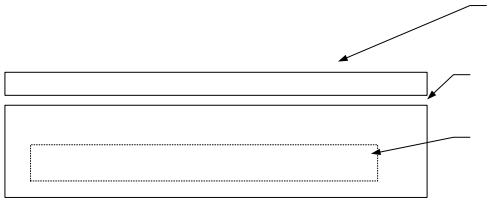

図５　カスタムタグを利用したフレームワークへのアクセス方法

9．サンプルアプリケーション（図書管理システム）について

今回，開発したフレームワークの効果を調べるために，フレームワークの Web アプリケーション開発事例として，「図書管理システム」を開発した．システムの処理内容はまったく同様でフレームワークを利用したシステムと利用しないシステムの 2 つを作成した．この 2 つのシステムを用いてフレームワークの効果について検討した．

「図書管理システム」には管理者と一般利用者のユーザが存在する．管理者は，新規図書の登録と，新規ユーザの登録が行える．新規図書の登録では，管理者がすべてのデータをフォームから入力する方法と，Amazon.com の提供する Web サービスを利用しデータを取得する方法がある．一般利用者はデータベースに登録された図書を検索でき，また，図書の貸し出し記録や，図書の返却として貸し出し記録の削除ができる．図６に「図書管理システム」のユースケースを示す．

「図書管理システム」は図７のように構成される．Web コンテナとして Jakarta プロジェクトの Tomcat4.1，画面遷移制御のフレームワークとして Struts，Web サービスを利用するための API として Apache AXIS，データベースとして postgreSQL を用いた．「図書管理システム」の一画面を図８に示す．

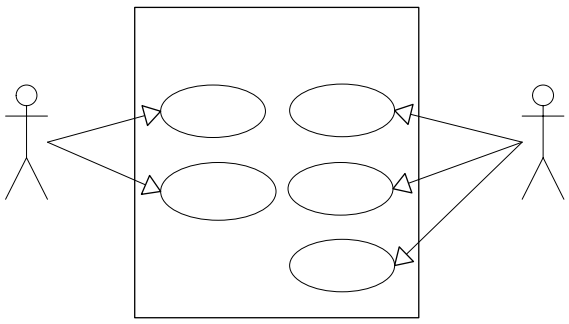

図６　図書管理システムのユースケース図

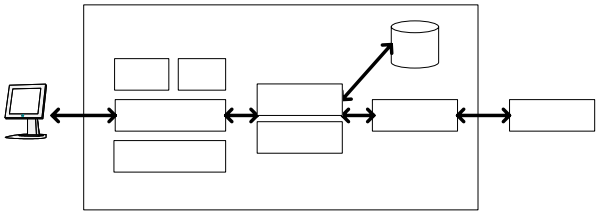

図７　「図書管理システム」のシステム概要図

図８　図書管理システムの図書検索結果画面

１０．本フレームワークを用いることの効果についての考察

本フレームワークを使用することの利点と問題点は次のように考えられる．

（1）利点

a）開発における役割の区分をより明確にできる

本フレームワークを利用することで，ビジネスロジックとプレゼンテーションロジックの区分をより明確に分けることができた．これにより，多人数による開発時に開発者はそれぞれの役割に集中することがき，また，役割の区分が曖昧になることにより生じる混乱を避けることができる．

b）ビジネスロジックの実装の隠蔽ができる

本論文で用いたサンプルアプリケーションのように，ビジネスロジックにおいてデータベースなどのセキュリティが要求される部分へのアクセスを記述する場合，そのコードはあまり多くの人の目に触れないほうが好ましい．本フレームワークではビジネスロジックを開発する者以外は，そのビジネスロジックを利用するのみでそのコードを見ることはできない．そのため，ビジネスロジックの記述に対する機密性を高めることができる．

（1）問題点

a）システムによってはコード量が増える

あらかじめサーバ上に登録されたビジネスロジックを再利用できるので，Web アプリケーション開発者はビジネスロジックのコーディング量を減らすことができる．

既存のコードからビジネスロジック部分を抽出しフレームワーク上に移行するために，フレームワークを用いないときに比べ一つの Web アプリケーション内で記述する定義ファイル，フレームワークへアクセスするためのコード，フレームワークから呼び出されるためのコードなどが必要となる．したがって，ファイル数が増えコード量が一般的に増える．しかし，ビジネスロジックの再利用率が上がれば，その分，コード量は減る．

b）フレームワークの信頼性

他の開発者が開発したコードが本当に信頼できるかという問題がある．この問題は，ソフトウェアの再利用を図る上で常に生じる問題である．

c）フレームワークについての知識が必要となる

フレームワークを利用するためには，その利用するフレームワークに対する知識が必要となる．そのため，開発者に対する教育の時間が増える．とくに，フレームワークでは，そのフレームワークのアーキテクチャにしたがって実装を行わなければそのフレームワークの持つ特徴を十分に引き出すことが難しくなる．そのため，正しいフレームワークに関する知識が必要となる．この問題を解決するためには GUI ツールによるサポートが重要である．つまり，GUI ツールによってフレームワーク特有のアーキテクチャを隠蔽する努力が必要である．

一般的にフレームワーク開発において，システムが大きくなるにつれて設定ファイルが増大するという問題がある．今回開発したフレームワークでも同様であり，サービスの量が増えると定義ファイルが大きくなる．フレームワークでは設定ファイルの文書形式を，標準化された文書として XML 形式を利用するケースが多い．XML 文書はテキスト形式の文書であるが，機械が情報を読み取るための文書なので，そのサイズが大きくなると人間が解析することが難しくなる．そのため，多くの情報でも視覚的に整理して表示されるような GUI ビューアも必要である．そこで，本フレームワークにおいても定義ファイルを生成できる GUI エディタを用意した．GUI エディタのスナップショットを図９に示す．

このように，フレームワークにおいて GUI ツールは実際の運用時には必要ないものの重要な要因を含んでいる．ある特定のフレームワークに関する初心者は，そのフレームワークの独自性から，そのフレームワークの利用方法を習得するまでに多くの時間を必要とする．したがって，フレームワークにはそのアーキテクチャを深く知らなくても直感的にそのフレームワーク利用がはかれるようなツールが必要とされている．しかし，一般的にフレームワーク開発よりも GUI ツール開発のほうが困難である．

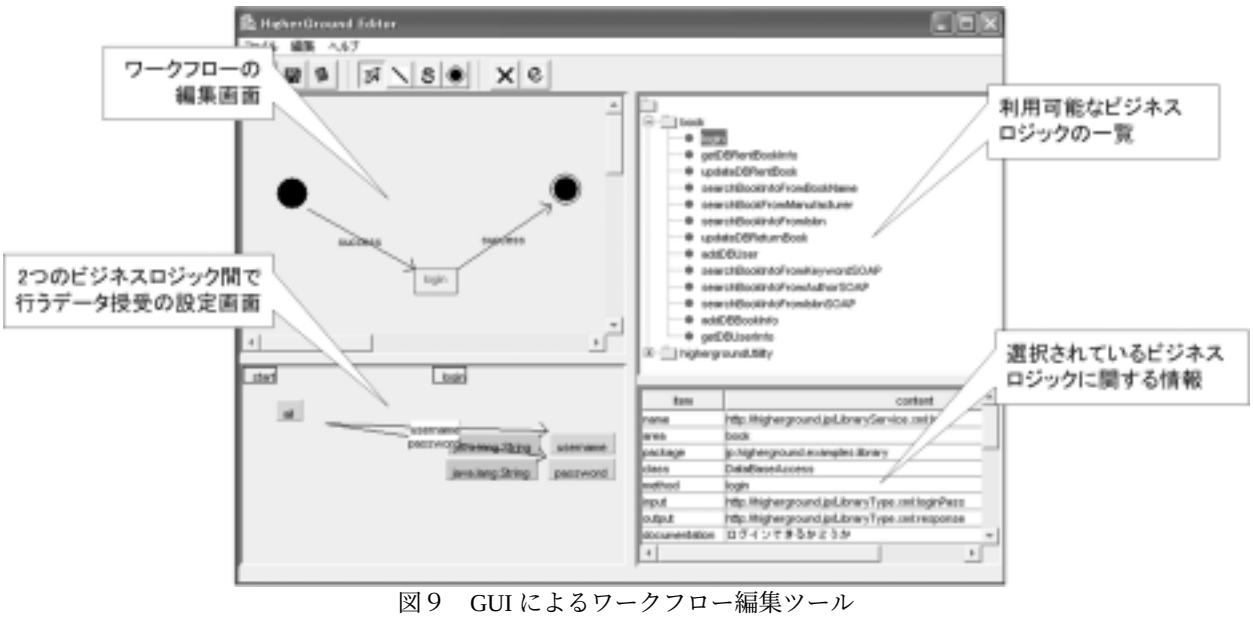

図９　GUI によるワークフロー編集ツール

１２．まとめ

ソフトウェアの再利用性向上を図る技術の一つとしてフレームワークは有用である．なぜならば，ソフトウェア開発に対するアーキテクチャの統一が行え，システム開発にかかわる者同士で知識の統一を図ることができるからである．しかし，ソフトウェアの再利用性向上のためには，規定されたアーキテクチャに従ったソフトウェアが必要であるが，そのソフトウェアを作成するためには手間が増えるという代償がある．つまり，その部品化を行ったソフトウェアが全く使われない，もしくは，利用頻度が低いなどの場合，ソフトウェアの部品化を行うほうのコストが高くなってしまう可能性がある．そのため，本当にそのソフトウェアは再利用されうるものなのかを判断していく必要がある．

本論文では，システム内でのみという狭い範囲でのインタフェースの強制的な標準化を行いソフトウェアの部品化を図るフレームワークを開発した．インタフェースにはオントロジーの問題があるが，この強制的な標準化を行うことでオントロジーの問題が幾分解消したのではないかと考えられる．

今後，Web サービスの分野では，業界ごとにインタフェースを標準化していく動きもある．しかし，標準化がすぐに行えるとは考えにくく，また，Web サービス技術において，標準化に準拠しないインタフェースも作り出すことができる．したがって，本論文で提案したような，インタフェースを強制的に標準化していく手法も有効ではないかと考えられる．

１３．謝辞

プロジェクト管理組織として担当していただき、また、さまざまなサポートをしていただきました株式会社オープンテクノロジーズの佐野 元之様に感謝いたします．

本プロジェクトに関してさまざまなアドバイスをくださいました明治大学の疋田 輝雄教授に感謝いたします。

そして、本事業の機会を与えて下さった竹内 郁雄プロジェクトマネージャーに感謝致します．

１４．参加企業及び機関

特になし

１５．参考文献

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissifers: Design Patterns, Addison-Wesley (1995)

[2] Inderjeet. Singh, Beth. Stearns, Mark. Johnson: Designing En-terprise Applications with the J2EE Platform, Second Edition, Addison-Wesley (2002）