既存WebフレームワークへのAjaxの適用

堀場 隆文

Applying Ajax to Existing Web Framework

Takafumi Horiba

Web2.0を実現するクライアントサイドの技術の一つとしてAjax（Asynchronous JavaScript + XML）がある. 本論文ではサーバサイドで動作する既存のWebフレームワークに対してAjaxを適用した場合の課題と解決案を検討する. 日本IBMのIPアセット[1]であるWeb Application Components（WACs）をWebフレームワークの具体例として挙げ, Ajaxの特徴である非同期通信がどのように利用できるかを検討する. 検討した結果として, セッション管理とシステムユーティリティの共用などの具体的な実現方法を提示する.

Ajax (Asynchronous JavaScript + XML) is one of the web technologies used to implement Web 2.0. In this report, I will provide some points to consider when applying Ajax to an existing Web framework and suggest a solution.

I will introduce Web Application Components (WACs), produced by IBM Japan, as one example of a Web framework, and suggest session management design and utility usage on WACs.

Key Words & Phrases： WACs, Ajax, Webシステム, 非同期通信, ユーザ・インターフェース
WACs, Ajax, web system , asynchronous communication, user interface

1. はじめに

Webシステムに対する共通の課題が「ユーザの操作性の改善」である. この課題を解決したWebサイトが出現し始めた. その一例がGoogle Suggest™［1］やGoogle mail™［2］である. その特徴はユーザが意識せずサーバに通信をすること, 通信した結果を操作中の画面に動的に表示することである. これらの機能はクライアントサイド技術の一つであるAjax（Asynchronous JavaScript + XML）を利用することで実現ができる.

Ajaxを適用する際には, 以下の課題を検討する必要がある.

（1）Webフレームワークとの適用性
（2）Ajaxフレームワークの選定
（3）クロスブラウザ問題
（4）パフォーマンスへの影響
（5）セキュリティ

既に企業のWebシステムはWebフレームワークを用いて構築されている. Webフレームワークとは, Webシステムの構築に共通して必要な機能のライブラリ群で, 利用することにより重複開発の回避・品質の向上・アーキテクチャの統一がしやすくなる. Ajaxを適用する場合にもサーバサイドの構築にはWebフレームワークを適用することが多いと考える. よって, Ajaxを適用する際には, サーバサイドで動作するWebフレームワークとの適用性の検討が最も重要な要素となる. Ajaxはクライアントサイドで動作するため, サーバサイドのWebフレームワークに対して影響を与えるのは通信に関する部分である.

Webフレームワークとの適用性の検討以外に, Ajaxを簡易に実現するためのJavaScriptライブラリ群であるAjaxフレームワークを適切に選定する必要や, AjaxはJavaScriptがベースであるため, ブラウザ間（バージョン間も）で動作や機能が異なることがあり（クロスブラウザ問題）対象ブラウザやバージョンの検討および, それらの差を吸収する方法の実現も検討することも必要である. パフォーマンスについては, JavaScriptを処理するクライアントマシンへの負荷や, 通信量の増加によるサーバへの影響も検討する必要があり, セキュリティについてもJavaScriptを利用し, かつ, サー

提出日：2007年3月13日　　再提出日：2007年9月17日

1 IPアセットとは, ベストプラクティスとして再利用可能なライブラリ群やノウハウ.

バに通信するという点で検討する必要がある.

本論文では, 既存のWebフレームワークとしてWACs（Web Application Components）を具体例として挙げ, Ajaxの通信部分, つまり, 非同期通信に注目しAjaxを適用した場合のセッションの効率利用および, 画面遷移制御を含むシステムユーティリティの共同利用方法を検討する. 以下, 2章でAjaxの技術要素を整理し, 3章ではWACsの紹介および, WACsのアーキテクチャや機能について説明する. そして, 4章で, WACsにAjaxを適用するための課題点を整理し, 第5章で課題点の解決方法を示しまとめとする.

2. Ajaxの技術要素と基本機能

Ajaxは「A New Approach to Web Applications」[3]というタイトルでJesse James GarrettによってWebで発表され急速に浸透している. Ajaxはブラウザが標準で備えている既存の技術を組み合わせて利用できることが特徴で, XMLHttpRequest[2]を利用した非同期通信技術, XMLを利用したデータ交換技術, XHTML[3]やJavaScriptを利用した動的な画面構成技術で構成されている.

本章では非同期通信, データ交換, 画面構成の三つのAjaxの基本機能を説明する.

(1)非同期通信

AjaxではXMLHttpRequestを利用して通信を行う. 従来のWebシステムの通信は画面遷移を伴う（同期する）通信に限られていた. しかし, XMLHttpRequestは画面遷移と同期せずに通信ができる（以下Ajax通信）. よって, Google Suggest™のようにユーザの操作とは同期せずにサーバと通信を行い結果を画面遷移せずに画面に反映できる.

(2)通信データ

通信データはXMLを利用することを想定されているが, XMLに限定されてはいない. 単純なテキストを送受信することも可能で, サーバサイドとクライアントサイドで通信データのフォーマットを自由に決め, 通信ができる. つまり, 従来の形式でテキストデータを送信することで, サーバサイドのWebフレームワークはAjaxを意識せず構築できる. サーバからデータを受信する際にもXMLで受信することも可能ではあるが, XMLのパースが必要になるためXMLより扱いやすいJSON（JavaScript Object Notation）[4]を利用することでJavaScriptから容易にデータ処理が可能となる.

(3)画面構成技術

画面の構成は, 受信したデータを解析し, DHTML（Dynamic HTML）[4]を利用して行う. 画面の一部を動的に変更することが可能であるため, ユーザの操作を止めることなく画面を更新できる.

3. WACsのアーキテクチャと開発支援ツール

ここではWACsの概要を説明した上で, WACsのアーキテクチャと開発支援ツールを紹介する.

3.1 WACs概要

WACsは日本IBMが提供しているWebフレームワークであり, 多数のプロジェクトで適用された実績あるWebフレームワークである. WACsはMVC（Model-View-Controller）モデル2[5]に基づいた軽量フレームワーク[6]部分と, 多くのWebシステムで共通に必要となる基盤系機能（WACsではシステムユーティリティと定義）および, 開発支援ツールで構成されている.

WACsを適用したWebシステムでは**図1**に示すようにクライアントとサーバで主に以下の技術が利用される.

図1. クライアント・サーバ間の技術

(1)クライアント

Webブラウザを前提とし, Webブラウザ上で動作するHTML, JavaScriptで画面や通信を制御する.

(2)クライアントとサーバ通信

クライアントとサーバの通信はHTTP（HTTPS）を必須としている.

(3)サーバ

サーブレットがクライアントからのリクエストを解析し, リクエストパラメータを変換し業務ロジックに渡す. 通常, 結果はJSPから生成したHTMLを送信しているが, HTTPで送信できるデータであればHTML以外でも可能である.（バイナリデータも送信可能）

3.2 サーバサイドの軽量フレームワーク

WACsは軽量フレームワークとしてWebフレームワー

2 サーバとHTTP通信するためのAPI[5]

3 HTMLを拡張したXML形式のマークアップ言語[6]

4 HTMLによる静的な表示をJavaScriptやCSS（Cascading Style Sheet）[7]を利用して動的に変更するWeb技術

5 MVCモデルとは, ビジネスロジックと表示ロジックを分離することで開発生産性・保守性を高める設計手法. モデル2はそれをWebの各要素, JavaBeans-JSP（Java Server Pages）- Servletを適用したパターン.

6 軽量フレームワークは, Struts同様にWebフレームワークのコア部分のみで構成している

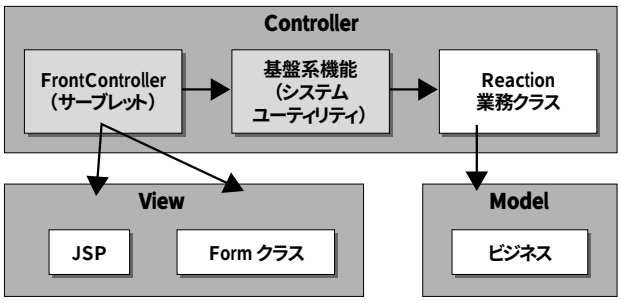

図2.　MVCモデル2に基づくサーバサイドのフレームワーク

クのコア機能を, 主に**図2**に示すControllerとして提供している. このControllerが必要に応じて後述のシステムユーティリティを呼び出すなどのWebに必要な制御を実行する. Controllerの代表的なクラスとしてFrontControllerクラス（StrutsのActionServletの役割）がある.

システムユーティリティは本論文でも扱うセッション管理機能や遷移履歴管理機能のほかに, 入力データチェック機能, 流量制限, メッセージ管理, 多言語対応など機能が豊富にそろえられている.

3.3　基盤系機能（システムユーティリティ）

WACsでは, Webシステムに共通で必要な基盤系機能をシステムユーティリティとして提供している. 当節ではクライアントとサーバ間の制御に関わるWACsのシステムユーティリティの一部を説明する.

（1）セッション管理機能

リクエストされたイベントに応じてセッションの有無のチェックや生成・無効化や論理的な分割を実現する機能である. セッションの無効化はWAS（WebSphere® Application Server）[8] はJ2EE（Java™ 2 Enterprise Edition）に準拠するため, セッション無効化機能を提供しているが[9], WACsでも以前は同様の機能をWASとは別に提供していた.

（2）遷移履歴管理機能

ブラウザの戻るボタンによる遷移やお気に入り・URLの直接入力などによる想定外の遷移など不正な遷移を禁止するためのチェック機能である. 定義ファイルに定義されたイベントID（Strutsの場合はアクションパス）およびイベント種別に基づいてリクエストの正当性を検査する.

3.4　開発支援ツール

WACsは開発生産性を上げるために画面に関連する以下の開発支援ツールを提供している. 開発支援ツールを使用することで品質の高いソースや共通ロジックを自動生成する.

（1）コード自動生成ツール

HTMLのフォームの項目の定義を行うことで, それに対応するJSP, Reactionクラス, Formクラスを自動生成し開発生産性を高める. SQLやデータベースの定義と組み合わせることでデータベースアクセスのロジックも自動生成される. なお, 基本的な入力チェックロジックもFormクラスの中に自動生成される.

（2）JSP自動生成ツール

コード自動生成ツールと画面（XHTML）を組み合わせて自動的にJSPを生成し開発生産性を高める. JSP自動生成ツールを利用することでXHTMLに対して画面項目のJSPタグが自動的に埋め込まれるため生産性が高い.

4．WACsにAjaxを適用するための課題

4.1　セッションの無効化

通常, Webシステムではメモリのクリーンアップやセキュリティの観点から一定時間アクセスがないセッションは自動的に無効化される. しかし, Ajax通信を利用した場合, 従来どおりセッションの無効化ができないケースがある. 例えば, タイマー起動でサーバに自動的に通信する画面は定期的にサーバにアクセスをするためセッションの無効化を行えない. 従って, 想定の時間以上, セッションが維持されメモリ不足やパフォーマンスの低下, 利用可能セッション数（Cookieの生成数）の不足によりサービスレベルが低下することも考えられる. よって, メモリを効率利用するために効果的なセッションの無効化の実施方法の確立が必要である.

4.2　システムユーティリティの共同利用

WACsのシステムユーティリティが利用するシステムパラメータはHTMLのフォーム内にhidden項目として自動的に埋め込まれる. Strutsでは, Transaction Token [10] がこれと似たパラメータの一つである. これはリクエスト時にサーバサイドでシステムユーティリティが動作するために必要なパラメータである. システムユーティリティには前述のセッション管理・遷移履歴管理などがあり, hiddenに埋め込まれたシステムパラメータを利用してリクエストの整合性をチェックする. Ajax通信をする場合, WACsのシステムユーティリティを正しく動作させるためにフォームに埋め込まれたシステムパラメータを取得し送信する必要がある. 送信しなければエラーとしてリクエストは受け付けられない. Ajax通信後に従来の通信を行う場合も同様である. システムユーティリティを両方の通信で正しく利用するためには, システムパラメータを共同利用する必要がある.

5．課題点の解決方法

4章で挙げた二つの課題に対する解決案を述べる．

5.1　セッション無効化機能の改善

セッションの効率利用をするためにセッション無効化機能の改善について述べる．

Ajaxの場合，サーバサイドでセッションを作成せず，クライアントサイドで全ての状態を保持する考え方もある［11］．この方法はJavaScriptの変数やフォームのhiddenとしてクライアントサイドで状態を保持する方法である．しかし，この方法は既存のWebフレームワークがセッションを利用することを想定して作られていることや，既存の開発・設計ノウハウを活用しにくいという点から今回は解決案として検討はしない．よって，既存の仕組みと同じ環境でセッションを維持する場合の効果的な実現方法を検討する．

5.1.1　改善の概要

クライアントからのアクセスがない場合のセッション無効化はWASなどのJ2EEサーバが自動的に実施する．J2EEサーバによるセッションの無効化はタイマー起動によるAjax通信をした場合，効果的に機能しない．そこで定義ファイルに基づいた無効化の判定および無効化の実施機能を設計することにした．

この機能では，セッションの無効化をユーザの非操作時間が一定時間経過した場合に実行する．ユーザの非操作時間は，Ajax通信でない従来通信による最後のアクセスから経過した時間を基に計算する．これにより，従来の通信は従来どおりの動作をし，Ajax通信については，有効期間を延長させないためにJ2EEサーバの無効化とは関連せずに破棄を実行できるようになる．当機能の設計は以下のとおりである．

5.1.2　セッションの無効化判定の詳細

図3に示す二つのクラスに無効化判定に必要な機能を設計した．

(1) AjaxSessionControlTableクラス

Ajax通信と従来の通信を判別は，通信時に送信されるイベントIDを利用する．当クラスはイベントIDを定義したファイルのロードおよび，判定をする．定義ファイルのロード以外に以下のメソッドを実装する．

①isUserConnectionメソッド
　定義ファイルに定義されているイベントIDがリクエストされた場合は，Ajax通信と判定する．

(2) AjaxDefaultContextクラス

当クラスはDefaultContextクラスを継承する．リクエスト受信時に，AjaxSessionControlTableクラスを利用してAjax通信の判定を行い，判定結果に応じて，セッションの有効期限の延長・セッションの無効化を実施する．

①isUserConnectionメソッド
　リクエスト内のイベントIDを利用してAjaxSessionControlTableクラスに判定を依頼する．
②updateLastAccessTimeメソッド
　isUserConnectionメソッドでユーザの操作による通信と判定された場合に最終アクセス時間を更新し有効期間を延ばす．
③isValidateメソッド
　有効期間内のアクセスか判定する．有効期間を経過してアクセスの場合，HTTPセッションオブジェクトの無効化を実行する．

AjaxDefaultContextクラスの機能の実行場所は，WACsの標準の場所であるが．Strutsの場合であれば，RequestProcessorのprocessメソッドの先頭が該当する箇所である．

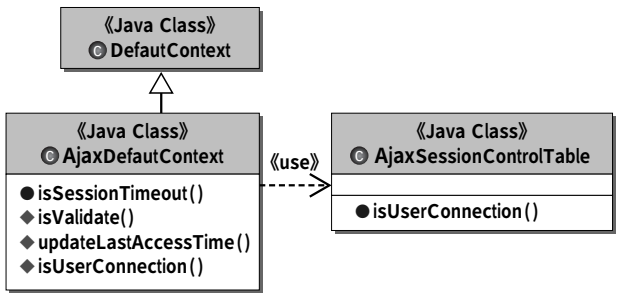

図3.　セッション管理クラスのクラス図

Ajax通信している場合でもイベントIDを定義ファイルに定義しないことで，ユーザのセッション有効期間の延長を実施することも可能である．これは，従来のWeb画面における課題の一つを解決している．セッションの無効化はユーザの操作中も，サーバ通信をしなければ発生する．入力項目が多い画面で発生しやすい．説明した機能を利用し，ユーザが操作している間はAjax通信を行うことでセッションの有効期限を延長することが可能となり課題を解決できる．

5.2　システムユーティリティの共同利用

システムユーティリティが実施する各種のチェック機能をAjax通信・従来の通信の両方で正しく動作させるためにWACsのシステムパラメータの共有および，チェック機能の共同利用が必要である．

5.2.1　システムパラメータの共有

システムパラメータは前述のとおりHTMLのフォームのhidden項目として追加される．よって，Ajax通信時にはhiddenとしてフォームに埋め込まれたシステムパラメータを取得し送信する必要がある．取得時の考慮

点は画面のフォームの数である. 複数フォームある場合, どのフォームのシステムパラメータを利用するか制御する必要があり実装が複雑になる. よってフォームが一つになるように画面設計を推奨する. 取得をしたパラメータはサーバに送信され処理されることで更新される. つまり, 次の通信では, 新たなパラメータをAjax通信時にも従来の通信時にも使用する必要がある. Ajax通信時には, レスポンス内にシステムパラメータを埋め込む設計にし, それをフォームのhiddenに反映することで後続のリクエストが正常に動作する.

また, WACs（WAS）はCookieを用いてクライアントを識別するため, Ajaxによる通信を行う際にもCookieを送付する必要がある.

5.2.2 遷移履歴管理のチェック

3章3節でも紹介したが, WACsの遷移履歴管理はいくつかのイベント種別に基づいて動作する. 従来の通信については従来どおり定義すればよいが, Ajax通信では考慮が必要である. イベント種別の設定次第で, 前節で述べたシステムパラメータの共有方法が変わる. イベント種別に normalを指定する場合, 前節で述べたシステムパラメータの共有を双方向（Ajaxからフォームへの反映, Ajaxがフォームから取得）で行う必要がある. また, Ajax通信が一つの画面で複数箇所から発生する場合, 同時にリクエストを飛ばせないため, 実装が複雑になる. 一方, ignoreを指定した場合はサーバ送信時に共有するのみで, 受信結果を反映する必要はない. 特にAjax通信が複数同時に発生するときにはignoreの指定が必要となる. uncheckは不用意に利用すると想定外の遷移が起こりえるため, 筆者はAjax通信のイベントにはignoreの指定を推奨する. ignoreを指定することで受信時にパラメータをフォームに反映する必要がなくなること, また, 複数のAjax通信が同一フォームから発生しても実装がシンプルなことから推奨する. なお, ignoreイベントとuncheckイベントの違いは以下のとおりである.

（1）ignoreイベント

不正な遷移ではないかどうかのチェックは行われるが, サーバサイドで保持している遷移履歴の更新やタイムスタンプの更新を行われないため, 後続の従来の通信も正常に動作する. サーバサイドでパラメータが更新されないため, クライアントサイドでもパラメータの反映をする必要がない.

（2）uncheckイベント

遷移に関するチェックは何もされないため不正な遷移がされたとしてもエラーにはならない. 具体的にはURLの直接入力（お気に入りの利用）による操作が可能となる.（タイムスタンプが不適切でも遷移可能）. よって, 正しく利用しないと不正な遷移の温床となる可能性が高くなる.

なお, 全通信がAjaxである場合, 画面遷移という考え方がないため, 遷移履歴管理そのものを利用しないという選択肢もある.

6. おわりに

本論文では, 既存のWebフレームワークを利用したWebシステムに対してAjaxを適用する場合の課題と解決案を検討し, WACsを具体例としセッション管理とシステムユーティリティの共用などの具体的な実現方法を提示した.

セッションの無効化の解決方法ならびに, システムユーティリティのパラメータの共有などの考え方はWACsを利用した場合だけではなく, 一般的なWebシステムにも適用できる.

Ajaxを利用することでユーザ・インターフェースが改善される可能性はある. しかし, 検討したような既存のWebフレームワークとの機能面での適用性, あるいは, 1章で挙げた検討項目を考慮した上で適用可否を判断する必要がある.

謝辞

本論文を執筆するにあたり日本アイ・ビー・エム・サービス株式会社の林誠氏にはAjaxの適用に関する検討項目についてご助言をいただいた. ここに深く感謝の意を示したい.

参考文献

[1] Google: *Google Suggest* http://www.google.co.jp/webhp?complete=1
[2] Google: *Google Mail* http://mail.google.com
[3] Jesse James Garrett: *Ajax: A New Approach to Web Applications,* http://www.adaptivepath.com/publications/essays/archives/000385.php（2005.2.18）.
[4] IEFF: *RFC 4627, The application/json Media Type for JavaScript Object Notation,* http://www.ietf.org/rfc/rfc4627.txt
[5] W3C: *XMLHttpRequest* http://www.w3.org/TR/XMLHttpRequest
[6] W3C: XHTML http://www.w3.org/TR/xhtml11/
[7] W3C: *Cascading Style Sheet* http://www.w3.org/Style/CSS/
[8] IBM Corporation: *WebSphere Application Server* http://www.ibm.com/software/webservers/appserv/was/index.html
[9] IBM Corporation: HTTP セッション使用の最良実例

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rprs_sess.html
［10］The Apache Software Foundation : *Struts Token Processor*
http://struts.apache.org/1.3.8/apidocs/index.html
［11］Bill Higgins, 没入型 Web アプリケーションでの Ajax/REST アーキテクチャー・スタイルの利点
http://www.ibm.com/jp/developerworks/web/library/wa-ajaxarch/

日本アイ・ビー・エム・サービス株式会社
中部アプリケーション・サービス部
ITスペシャリスト

堀場 隆文 Takafumi Horiba

［プロフィール］
2001年日本アイビーエム中部ソリューション（現日本アイ・ビー・エム・サービス）に入社．Webシステムの開発に従事し，2003年以降，ITスペシャリストとしてWebシステムの設計・開発を行う．近年はStrutsやWACsなどのWebフレームワークを利用したWebシステムの構築や，それらの技術支援を担当している．
horiba@jp.ibm.com