Vol.2009-DD-73 No.1
2009/9/25

Web API マッシュアップのための E4X サービス

杉本正勝

東京情報大学・総合情報学部・情報システム学科

複数の Web API (Web サービス) を連携させて新しい Web アプリを作り出すマッシュアップでは，Web サービスの処理結果としてのデータを次の Web サービスにうまくつなげる必要がある．データ形式には XML がよく利用される．XML データを処理する従来の DOM API には多くのプログラミングステップが必要という欠点があった．最近ではより簡潔に処理できる JSON データも現れているが，この場合，XML to JSON データ変換が必要となる．
ここでは，XML データを E4X サービスで扱うことに焦点を合わせ，E4X サービスの実現法の検討と試作システムの一例を紹介する．また，E4X サービスを利用することで手軽にマッシュアップが行えることを示す．

On E4X Service for Mashing-up the Web APIs

Masakatsu Sugimoto
Department of Information Systems
Tokyo University of Information Sciences

In a mashing-up process where plural Web APIs: Web services are linked to establish an Web application, XML data is used frequently. There has been a difficulty in handling XML data. The conventional technologies such as DOM API, XSLT are technologies for experienced programmers. For easy mash-up, we need technology for easy handling of XML. JSON is a solution to this problem. However, we need XML to JSON data conversion. Another problem is that the specification on data sequencing rule for JSON is different from that for XML. Here, we will focus on E4X for handling XML data. We will discuss design and implementation of an E4X service for easing mashing-up. We will show how easily mashing-up is performed in our experimental system.

1. はじめに

複数の Web API（Web サービス）を連携させて Web アプリを作り出すマッシュアップは，新しいソフトウェア設計・作成法として注目を浴びている．SOA（サービス指向アーキテクチャ）の文脈では，ビジネス分野に於ける BPEL のように，ビジネス分野の専門家がその分野の言葉でシステムを記述し，システムを動かす方向に向かっている．しかし，より広い範囲のシステム作りには，まだまだプログラミングの専門家を必要としている．

我々はプログラミングの専門家でなくても利用できるマッシュアップの枠組み作り目指して取り組んでいる．Java や C++といったプログラミング言語を駆使しなくても，容易にアプリを設計・作成することを目標としている．

Web サービスを組合せる手法はその方向に大きく一歩進んだ．しかし，Web サービスの処理結果としてのデータを次の Web サービスにうまくつなげるには問題があった．サービス結果のデータ形式には XML がよく利用されているが，XML データを処理する従来の DOM API では多くのプログラミングステップが必要で，プログラミングの専門家が必要という欠点があった．最近ではより簡潔に行える JSON データも現れているが，JSON を利用するには，XML-JSON データ変換が必要となる．また，XML と JSON には繰返し項目の記述法に差があり，注意を要する．本研究では XML データの取り扱いを容易にし，プログラミングの専門家でなくても取り組めるマッシュアップを実現することに取り組んだ．

先回の発表では，JSON と XML との比較を行なった後，その時点では「プログラミングでの処理の容易さ」の点で，JSON が優れていることをあげ，XML には「連想型データアクセス」機能が必要であると結論づけた [1][2].

E4X はまさにこの点に焦点を合わせた JavaScript 言語の機能拡張であり，大いに期待が持てる．しかし，主要なＷｅｂブラウザーすべてでこの機能が使える訳ではない．Firefox では使えるが，I.E. では使えない[3][4][5].

我々は E4X をより広範に利用できるようなメカニズムに焦点を合わせた．E4X 処理を Web サービスとして実現し，どのブラウザーからでもこの機能が利用できるようにと考えた．

XML データを E4X サービスで扱うことに焦点を合わせ，E4X サービスの実現法の検討と試作システムの一例を紹介する．また，E4X サービスを利用することで手軽にマッシュアップが行えることを示す．

E4X サービスの実現には，Jaxer Web AP サーバーを利用した．このサーバーには Mozilla の

JavaScript 実行機能が内蔵され，E4X の実行機能が含まれている．

Web サービスには SOAP 型と REST 型に二つに分類されるが，ここでは REST 型を利用している[6][7][8][9].

2. E4X

XML を処理するには DOM API が利用されてきた．DOM API は DOM ツリーを辿ることで，所定のデータ要素にアクセスする汎用的な手法である．

E4X は，ECMAScript for XML の略であり，ECMA が標準化した JavaScript 言語向けの XML 処理用拡張機能である．JavaScript にネイティブ XML オブジェクトを導入するとともに，XML リテラルを XML オブジェクトへ取り込むための機能等を持っている[3][4][5].

次の例は，ネイティブ XML オブジェクトを変数 pt へ設定している[4].

```
// Create an XML object
var pt =
    <periodictable>
        <element id='1'><name>Hydrogen</name></element>
        <element id='2'><name>Helium</name></element>
        <element id='3'><name>Lithium</name></element>
    </periodictable> ;
```

次の例は，XML オブジェクトへ要素を追加する．

```
pt.element += <element id='4'><name>Beryllium</name></element> ;
```

また，{ と }の括弧でくくることで XML の中に JavaScript の式を直接書くこともできる．

```
pt = <periodictable></ periodictable > ;
var elements = [ 'Hydrogen', 'Helium', 'Lithium' ] ;
// Create XML tags using array contents.
for( var n = 0 ; n < elements.length ; n++ ){
    pt.element += <element id = { n + 1} ><name>{ elements[ n ] }</name></element> ;
}
```

文字列の XML をパースして利用することも可能である．

```
pt.element += new XML( '<element id = "5"><name>Boron</name></element>' ) ;
```

XML の部分文字列を扱うときには，次に示すように XML()の代わりに XMLList()を使う．

```
pt.element += new XMLList( '<element id = "6"><name>Carbon</name></element>' +
                           '<element id = "7"><name>Nitrogen</name></element>' ) ;
```

E4X では，XML データのデータ要素に対するアクセスも大変簡潔になる．表 1 には，次の XML データに対するアクセスを，従来の DOM API で行なった場合と，E4X の場合とのアクセス式の比較を示す[5].

```
var authorsXML =
<allAuthors>
    <publisher>Friendly Books</publisher>
    <publishYear>2008</publishYear>
    <author authorID="1">
        <authorFirstName>Alison</authorFirstName>
        <authorLastName>Ambrose</authorLastName>
        <books>
            <book ID="1">
                <bookName>Shopping for profit and pleasure
                </bookName>
                <bookPublishYear>2002</bookPublishYear>
                <bookCost>14.99</bookCost>
            </book>
            ---
        </books>
    </author>
    <author authorID="2">
        ---
    </author>
</allAuthors> ;
```

E4X の式は DOM API に比較して，直感的であり，アクセス式は簡潔になる．それ故，我々は Web API のマッシュアップのおける XML データの処理には E4X を利用することとした．

	DOM API	E4X
1	authorsXML.child(“publisher”)	authorsXML.publisher
2	authorsXML.child(“author”)	authorsXML.author
3	authorsXML.child(“author”)[0]	authorsXML.author[0]
4	authorsXML.child(“author”)[0]. child(“books”).child(“book”)[1]	authorsXML.author[0]. books.book[1]
5	authersXML.children()	authersXML.*
6	authorsXML.child(“author”)[1].children()	authorsXML.author[1].*
7	authorsXML.children()[1]	authorsXML.*[1]
8	authorsXML.child(“author”). attribute(“authorID”)	authorsXML.author.@authorID
9	authorsXML.child(“author”)[1]. child(“books”).child(“book”)[0]. attribute(“ID”)	authorsXML.author[1]. books.book[0].@ID
10	authorsXML.child(“author”)[0]. attributes()	authorsXML.author[0].@*
11	authorsXML.descendants(“book”)	authorsXML..book
12	authorsXML.child(“author”)[0]. child(“books”).descendants(“*”)	authorsXMLauthor”[0].books..*

表 1　DOM API と E4X 比較

3.　E4Xサービスとマッシュアップ

上記のように XML データ処理を簡潔に行なえる E4X であるが，現状ではその機能が使える場所が限定されている．ブラウザーでは，Firefox では使用できるが，I.E.では使用できない．また，サーバーでは，Aptana 社の Jaxer など少数のみで利用できる．

我々は E4X の機能をネット中のどこからでも利用できるように，E4X 機能を Web サービス化する．

E4X サービスの基本は，サービス呼出し側は E4X を含む JavaScript プログラムの断片を用意し，そのプログラム断片の実行をサービス側に依頼する．このプログラム断片のことをここではマッシュアップ・プログラム・セグメント(msps)と呼ぶ．

msps に指定できるのは，原則として E4X と JavaScript の全機能とする．しかし，サービス側で msps を実行するときの「実行のセキュリティ」を考慮して，機能制限する必要が生じる．

また，マッシュアップを容易にするために表 2 に示す機能を追加した．

ms_create_XML/1 は，パラメータとして与えられる XML 文字列の中から先頭の xml 宣言を削除した上で，E4X の XML オブジェクトを作り出す．

```
function ms_create_XML( p ){
    // Filetering: '<?xml          ?>'を削除
    var s = p.replace( /<¥?(.*?)¥?>/ , '' ) ;
    var xml_01 = new XML( s ) ;
    return xml_01 ;
}
```

ms_service_result/1 は，マッシュアップ処理中の任意のデータをマッシュアップ処理後，サービス呼出し側で利用できるように用意した．サービス呼出し側では，ms_service_result オジェクトから ID を指定してデータを得ることができる．

ms_put/1, ms_put_nl/1 は，マッシュアップ動作の検証用に用意している．

以上のように，E4X サービスを単に『XML の処理の実行』だけではなく，Web API 呼出し，マッシュアップ処理, 動作検証を含めて行なえるように, 追加機能を用意したことがポイントである．

図 1 には，一つのサービス（Yahoo!の日本語形態素解析サービス）を呼出し，そのサービス結果としての XML データを E4X の機能で処理する msps の例を示す．この例では，E4X の機能である namesapace/0 と default xml namespace 文を用い, XML の名前空間に対処している．

サービスを呼ぶ側のプログラムは次のようになる．msps に「url-encoding」を施した後，サービス側に渡している．本サービスは JSONP 型で実現しているので，パラメータとして callback 関数名も指定している．

```
text01 = encodeURIComponent( msps ) ;
url01 =   E4X サービスの url ;
parms01 = '?input=' + text01 + '&output=xml&callback=' + callback 関数名 ;
url02 = url01 + parms01 ;
JSONP_service_call( url02 ) ;
```

Vol.2009-DD-73 No.1
2009/9/25

	機　能	関数名とパラメータ数	説　明
1	Web API を呼出す	(1) ms_service_call/2	第一引数は，url 第二引数は，JSON形式のパラメータ指定 例：　{ 'v' : '1.1' , 'q' : '東京情報大学' }
		(2) ms_service_call02/1	第一引数は，urlの後ろにパラメータ結合した形式 結合されるパラメータの例 : ?v=1.1&q=東京情報大学
2	文字列から E4X の XML オブジェクトを作る	(1) ms_create_XML/1	引数は，文字列 例：　ms_create_XML(resp01)
		(2) ms_create_XMLList/1	引数は，文字列 例：　ms_create_XMLList(resp01)
3	サービス結果を ID 付きで戻す	ms_service_result/1	ms_service_result ('result_ID') = value ;　形式の代入文を利用する
4	サービス結果をデフォルトの ID: 'ms_put'で戻す	(1) ms_put/1	ms_putのサービス結果が文字結合される(累積する). この機能は動作検証等に使用できる
		(2) ms_put_nl/1	ms_putと同様な動作をする．違いは文字列の終わりに改行コードを足す

(注)　ms_service_result/1 は，マッシュアップ処理中の任意のデータをマッシュアップ処理後，呼出し側で利用できるように用意した．呼出し側では，ms_service_result オブジェクトから ID を指定してデータを得ることができる．

ms_put/1, ms_put_nl/1 は，マッシュアップ動作の検証用に用意している．

表 2　E4X サービスで使用できる追加機能

```
var api_id = ' application-ID here ' ;
var url_01 = 'http://jlp.yahooapis.jp/MAService/V1/parse' ;
var text01 = 'この文章をYahoo！Japanの日本語形態素解析サービスで解析し，文章の『読み』を求めてみます. ' ;
var text01_encoded = encodeURIComponent( text01 ) ;
var parms_01 = { 'appid'      : api_id ,
                 'sentence' : text01_encoded ,
                 'results'    : 'ma,uniq' ,
                 'response' : 'surface,reading,pos,baseform,feature'
               } ;
var service01 = {} ;
service01.resp = ms_service_call( url_01 ,
                                  parms_01
                                ) ;
var xml_01 = ms_create_XML( service01.resp ) ;
var ns = xml_01.namespace() ;
default xml namespace = ns ;
var word_list01 = xml_01.ma_result.word_list ;
ms_put( '¥n' ) ;
ms_put( 'テキスト:  ' + text01 + '¥n' ) ;
ms_put( '語カウント:  ' + xml_01.ma_result.word_list.word.length() + '¥n' ) ;
ms_put( '読みは:  ' ) ;
for( var i = 0 ; i < word_list01.word.length() ; i ++ ) {
    ms_put( word_list01.word[ i ].reading ) ;
}
```

(注)　E4X の機能である namesapace/0 と default xml namespace 文を用い，XML の名前空間に対処

図 1　msps の例

4. E4Xサービスを用いたマッシュアップ

図2にE4Xサービスを用いて二つのWeb APIをマッシュアップする例を示す．図3はマッシュアップ結果を示している．

```
var service01 = {} ;   // 第一番目のサービスの呼出し
service01.resp = ms_service_call( 'http://www.geocoding.jp/api/' ,
                                  { 'v' : '1.1' , 'q' : '東京情報大学' }
                                ) ;
var xml_01 = ms_create_XML( service01.resp ) ;
var ns01 = xml_01.namespace() ;
default xml namespace = ns01 ;
ms_put( 'xml_01:¥n' + xml_01 + '¥n'  ) ;
ms_service_result[ 'url_01' ] = xml_01.url.toString() ; // url 値を保存
var service02 = {} ;   // 第二番目のサービスの呼出し
service02.resp = ms_service_call( 'http://express.heartrails.com/api/xml' ,
                                  { 'method' : 'getStations' ,
                                    'x' : xml_01.coordinate.lng ,
                                    'y' : xml_01.coordinate.lat
                                  }
                                ) ;
xml_02 = ms_create_XML(service02.resp ) ;
var ns02 = xml_02.namespace() ;
default xml namespace = ns02 ;
var stations = xml_02.station ;
for ( var i = 0 ; i < stations.length() ; i++ ) {
        ms_put_nl(    stations[ i ].name + ': '  + stations[ i ].line  + ': '  + stations[ i ].distance  ) ;
}
```

図 2 二つのWeb APIマッシュアップ（例）

図2の第一番目のサービスは，「geocodingサービス」で，ランドマーク名（この場合は，例として東京情報大学）の位置座標を求める．第二番目のサービスは「HeartRails Expressサービス」で，位置座標からその地点への路線情報を取得するサービスである[7][8]．この二つをマッシュアップして，『東京情報大学への最寄り駅名（複数）と大学・駅間の距離』を求めている．

```
url_01:
 http://www.geocoding.jp/?q=%E6%9D%B1%E4%BA%AC%E6%83%85%E5%A0%B1%E5%A4%A7%E5%AD%A6
ms_put:
 xml_01:
<result>
  <version>1.1</version>
  <address>東京情報大学</address>
  <coordinate>
    <lat>35.637358</lat>
    <lng>140.203357</lng>
    <lat_dms>35,38,14.489</lat_dms>
    <lng_dms>140,12,12.085</lng_dms>
  </coordinate>

<url>http://www.geocoding.jp/?q=%E6%9D%B1%E4%BA%AC%E6%83%85%E5%A0%B1%E5%A4%A7%E5%AD%
A6</url>
  <needs_to_verify>yes</needs_to_verify>
  <google_maps>私立東京情報大学</google_maps>
</result>
千城台: 千葉都市モノレール 2 号線 2140m
千城台北: 千葉都市モノレール 2 号線 2220m
小倉台: 千葉都市モノレール 2 号線 3040m
```

図 3　図2のマッシュアップの実行結果

5. E4Xサービス実現の機構

E4X サービスを実現する機構を図 4 に示す.

E4X を含む msps を受取る．Filter 部でプログラム・セグメント中のエラーやセキュリティ条件に合わない命令を検出し，排除する．その後のプログラム・セグメントを Evaluation 部で実行し，実行結果を「url encoding」後，返信する.

図 4 の機構は，Aptana 社の Jaxer Web AP サーバー上で実現した．Jaxer サーバーには，Mozilla の JavaScript 実行機構が内蔵され，E4X 実行機能も含まれている.

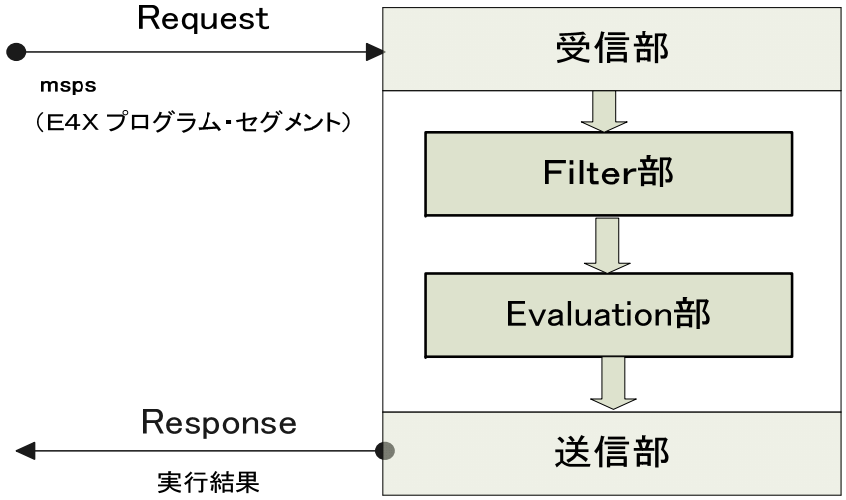

図 4　E4X サービスを実現する機構

E4X サービス・テスター

図 5 には，この E4X サービスの動作を検証する目的の E4X サービス・テスターの画面を示す．E4X code エリアに実行したい msps（E4X プログラム・セグメント）を置き，Service call ボタンを押すと，サービス実行結果が結果エリアに表示される.

なお，E4X code エリアの直後のエリアには，サービス結果をもとにクライアント側で実行したい動作を JavaScript のプログラム・セグメントの形式で置くことができる．このエリアを設けることで，小規模なマッシュアップ機能を，クライアント動作を含めて検証することが可能となる[10].

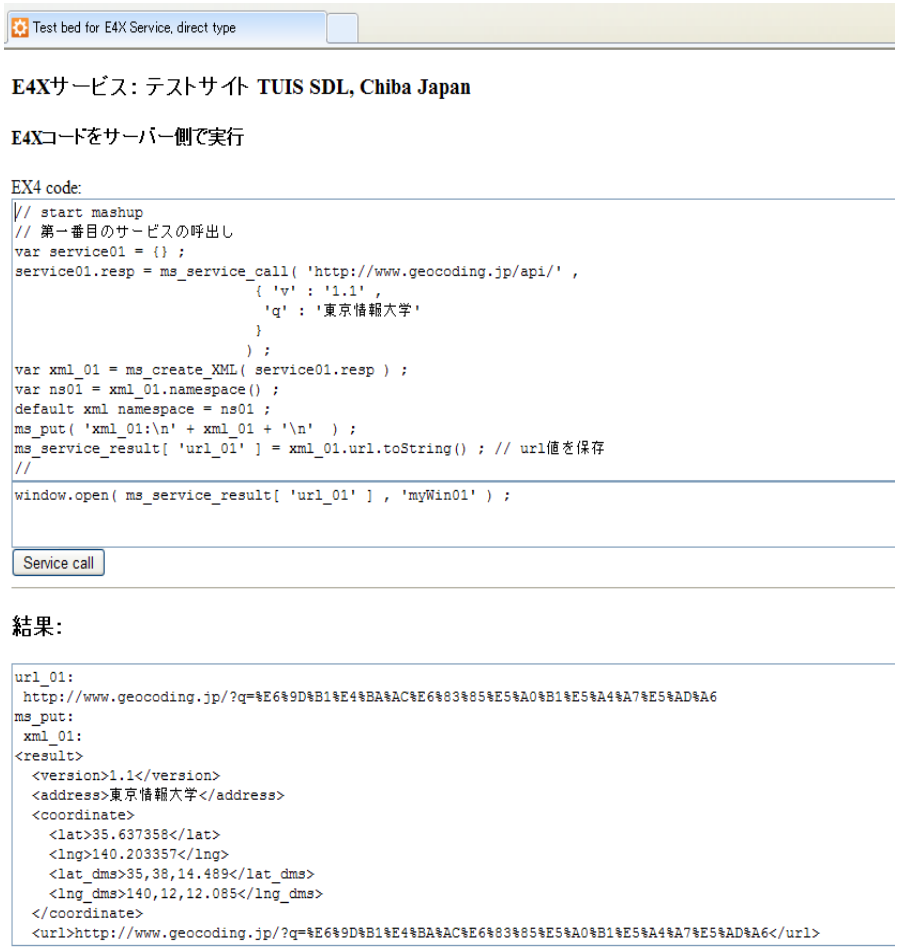

図 5　E4X サービス・テスター利用画面

6. おわりに

Web API マッシュアップをより容易に行なえるように，E4X 処理機能を Web サービスとして実現することを検討した.

Web サービスとして実現することで，E4X に対応していない Web ブラウザーや Web AP サーバーであっても，E4X 機能を利用できることになる.

E4X サービスの実現法の検討，試作，試作システムの利用を行いこの考え方を検証した.

E4X サービスは単に「XML データの処理」にとどまることなく，「マッシュアップ実行サービス」へと大いに拡張できることを見出した．また，そのために必要となる追加機能を抽出できた.

更に、試作した E4X サービス・テスターについて，次のことが分かった.

E4X サービス・テスターは,

① 個別の Web API の動作確認に役立つ
② 複数の Web API のマッシュアップ動作の確認に役立つ
③ XML データや JSON データを処理するプログラムの動作確認に役立つ

クラウドコンピューティングの時代に入り，サービス指向の考え方でソフトウェアを設計・作成することがますます重要となってくる[11][12]． Web API のマッシュアップを更に容易にするために，E4X サービスをより広く利用できるよう努力するとともに，マッシュアップの枠組みをより洗練されたものとしたい.

謝辞 本研究を支援していただきました学校法人東京農業大学の松田藤四郎理事長，東京情報大学の新沼勝利学長はじめ関係の方々に深謝致します.

参考文献

1) 杉本正勝 : Web ページにおける動的サブツリー・ローディング，情報処理学会,研究会報告, DD -61 (2007)
2) 杉本正勝 : Web タッチ shk の設計・構築とデータ構造，情報処理学会,研究会報告, DD-66 (2008)
3) ECMAScript for XML(E4X) Specification ECMA-357 2nd Edition/December (2005)
4) David Flanagan: JavaScript The Definitive Guide, O'Reilly (2006)
5) Sas Jacobs: XML and E4X for Flash and Flex, friend of ed (2009)
6) 横山昌平，的野晃整，サイド ミルザ パレビ，小島 功 : Web 2.0 における JavaScript コードのモジュール化とマッシュアップの枠組み，日本データベース学会，Letters Vol. 5, No. 3, pp. 1-4 (2006)
7) 新谷虎松,大囿忠親 : 知的 Web のためのマッシュアッププログラミング，情報処理, Vol. No. 5, pp. 444-453 (2009)
8) 本田正純 : マッシュアップかんたん A to Z，C&R 研究所 (2007)
9) 池田宗平，草野直樹，長嶺貴一，鎌田十三郎 : 要求駆動によるマッシュアップと AJAX Widget による効率的な閲覧，コンピュータソフトウェア，Vol. 26, No. 3, pp. 20-33 (2009)
10) E4X サービス・テスター : http://202.26.158.57:8081/ms/ms_Tester/ms_Tester01.html
11) Leitner P. Rosenberg F. and Dustdar S.:Daios: Efficient Dynamic Web Service Invocation, IEEE Internet Computing, May/June, pp.72-80 (2009)
12) Yarimagan Y. and Dogac A.: A Semantic-Based Solution for UBL Schema Interoperability, IEEE Internet Computing, May/June, pp. 64-71 (2009)