社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－DD－69（4）
2008 / 11 / 27

コマンドラインインターフェイスと直感的なXML操作

荒本 道隆†

†アドソル日進（株） 生産技術部 〒108-0075 東京都港区港南4－1－8 リバージュ品川
E-mail: †Aramoto.Michitaka@adniss.jp

あらまし Internet上や企業内に存在するWebサービスや，データファイルとして多くの種類のXMLデータがあり，それらをマッシュアップすることでさらに新しい価値を持ったXMLデータを作成することができる．

すでに様々なマッシュアップツールが出てはいるが，GUI(Graphical User Interface)を使った物が多く，CUI(Character-based User Interface)でのUNIXのコマンドを駆使したような操作感を得ることは難しい．

そこで本試作では，UNIXのシェルによるコマンドラインの操作性を目指し，XMLデータを自由に操作するためのブラウザ上で動作するコマンドラインインターフェイスを持ったWebShellを構築する．

キーワード XML, XPath, ユーザーインターフェイス，コマンドライン, WebShell

Command line interface and intuitive XML operation

Michitaka ARAMOTO†

† Productive Engineering Dept., Ad-Sol Nissin Corp. 4-1-8 Kounan Riverge, Shinagawa, Minato-ku, Tokyo 108-0075 Japan
E-mail: †Aramoto.Michitaka@adniss.jp

Abstract There is a lot of kinds of XML data as Web service and a data file that exists in on Internet and the enterprise, and they are made and the XML data with newer value in mashup tool can be made.

There are a lot of using GUI(Graphical User Interface), and it is difficult to obtain the operation feeling to make good use of the command of UNIX in CUI(Character-based User Interface) .

Then, the purpose of this making for trial purposes aims at the operativeness of the command line with the shell of UNIX, and constructs WebShell with the command line interface that operates on a browser to operate the XML data freely.

Keyword XML, XPath, User Interface, Command Line, WebShell

1. はじめに

1.1. 開発動機

Internet上では，SOAPやRESTfulなどのWebサービスにより，様々な情報をXMLで送受信することが当たり前になっている．また，Webサイト上のHTMLもXHTMLに適合していればXMLとしても扱えるので，実に多くのXMLデータが利用可能である．パソコン上でドキュメントを作成する場合でも，Microsoft製品ではOpen XML Document形式で保存され，OpenOfficeではOpen Document Format形式で保存されるが，いずれもXMLである．そして，それらのXMLデータをマッシュアップすることで，新たな価値を持ったXMLデータを作成できる可能性がある．

ただし，それぞれのXMLデータはすべて異なるスキーマで記述されている．利用する数だけスキーマを理解する必要があるし，スキーマが公開されていないこともありえる．しかも，XMLデータのフォーマットの規定にはXML Schema, DTD(Document Type Definition), RELAX NG(RELAX Next Generation)といった異なるスキーマ定義言語があり，使いたい定義言語以外で提供されている場合もある．そこで今回は，スキーマを利用せず，利用者がデータの内容を見て操作することを前提とする．

すでに無償・有償のさまざまなマッシュアップツールが存在しているが，ほとんどがGUIのツールであり，CUIのツールはまったく見かけられない．しかし，XMLデータの操作を何度も試行錯誤しつつ目的のXMLデータを完成させる過程においては，CUIの方が効率的であり，直感的な操作が可能であると思われる．そこで本試作では，CUIをベースとしたコマンドラインでの実行環境をWebShellとして構築する．

2. WebShellの特徴

2.1. 実行環境

WebShellの動作するインフラとしてはブラウザを使い，開発言語としてJavaScriptを採用する．ブラウザを利用することで，ネットが利用できる環境であればどこでも使え，JavaScriptで開発することでOSに依

存しない実装となる．また，JavaScript は通常の Web サイト構築でも広く使われている言語なので，多くのライブラリをそのまま利用できる．

2.2. GUI と CUI の比較

GUI のアプリケーションでは，主にマウスを使ってボタンやメニューを利用して操作するので，初めてそのアプリケーションを触る場合でも何となく操作でき，初心者にとっては利用し易い．それに対して CUI のアプリケーションでは，キーボードのみを使ってコマンド独自のオプションを駆使するため，初めて使う場合には使い方がまったく分からないこともありえる．

しかし，GUI のマウス中心の操作と比較すると，CUI のキーボード中心の操作には様々なメリットがあり，慣れれば GUI よりも生産性は高くなる．

まず，同じ処理を繰り返す事が容易である．失敗しても，シェルのヒストリ（実行履歴）を参照することで，前回と一部だけ変更して実行することが容易にできる．マウスでは，同じ動きを正確に繰り返したり，一部だけ変えて実行したりするのは難しい．

それから，動作確認したコマンドを記録しておき，それを連続して実行することで自動化が可能となる．GUI でもアプリケーションが対応していれば自動化できるが，CUI は異なるコマンドを組み合わせて自動化できる．

また，CUI は実行したコマンドや結果の出力がすべてテキストなので，記録に残し易く，メールなどで第三者に伝え易いという特徴もある．

2.3. XML データの操作

XML データを操作・編集には，以下のようなものが代表的な方法としてあげられる．

- DOM/SAX の API を使った操作
- 開発言語にバインドして操作し，XML に戻す
- XSLT による変換
- XML 対応ツールによる編集
- ETL（Extract Transform Load）による変換

WebShell では，ブラウザの JavaScript による DOM API による操作を採用する．

2.4. UNIX のファイル操作との比較

UNIX のコマンドラインでは，パイプやリダイレクトを使用し，複数の小さなコマンドを柔軟に組み合わせて，目的の結果を得る．特にテキストデータを操作することに特化したコマンドが豊富にあり，テキスト中の以下の要素に対して操作できる．

- データの先頭，終端
- 特定のパターンとマッチする行，もしくは部分
- csv 形式やログ形式であれば，特定のカラム

UNIX のコマンドの多くはテキストデータを対象としており，XML データを対象としたコマンドは非常に少なく，XML のメリットを生かしたまま複数のコマンドを組み合わせて使える場面が少ない．

そこで WebShell では XML を前提とし，全コマンドで XML に対応することで，XML であることのメリットを最大限に活かす環境となることを目指す．

2.5. データの保存先

データを保存には，ブラウザ側のメモリを使い，シェルからは変数として扱うこととする．ブラウザ側のリソースを使うので，サーバ側のリソースを一切使用しないで済む．変数名と通常の文字列を区別するために，UNIX と同様に変数名には先頭に$を付ける．

変数の格納には，リダイレクトを使っておこなう．UNIX であればリダイレクトするとファイルに出力されるが，WebShell では変数に出力する．

```
echo abcdef > $a
```

「変数の XPath 指定した特定の場所」を指定して入出力を行うことで，変数内の部分的な参照・書き換えを実現する．

XML データであれば，変数内に DOM 形式で格納しておく．DOM のまま扱うことで，DOM API の特徴を活用できる．例えば，変数に格納された DOM 内の一部を XPath で指定して，それを別の変数で参照した場合，それぞれ実体は 1 つの DOM のルートと子要素なので，どちらかの変数を使って内容を変更すると，両方に反映される．

テキストデータは，変数内にそのまま格納する．

XPath で指定した要素が複数あれば，配列として複数の子要素を 1 つの変数に格納する．

そのため，変数には，「DOM, テキストデータ，配列」のいずれかが格納されることになる．

2.6. XPath

XML データ内の特定の場所を指定するには，XPath(XML Path Language)が広く使われている．

XPath の一般的な書式では，XML データのある要素を基準として，ロケーションパスによって別の要素や属性を指定する．指定された要素や属性が存在しない場合もあるし，複数の要素が指定される場合もある．ロケーションパスとは，各要素や属性を「/a/b/c」の形式で指定する．この形式は，URL やファイルパスの指定と非常に似ている．「/a[@id='1']/b/c」のように条件を記述することもできる．

WebShell で変数内の XML データの特定の場所を指すには，変数名の後に「//」と書き，// 以降を XPath として記述する．

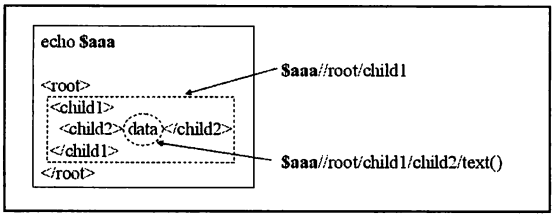

図 1　XPath の記述

2.7. HTML データの活用

Internet 上には，非常に多くの HTML データが存在している．

「Web スクレイピング」と呼ばれる HTML から必要な情報を抽出する技術があるが，WebShell では XML データのみを対象としているため，XHTML でなければ XPath を利用して操作できない．しかし，残念ながら多くの Web サイトの HTML は XHTML として妥当ではない．そこで WebShell では，HTML を XHTML に変換する機能を実装した．

残念ながら，現在の HTML のほとんどは，ブラウザで表示されることのみを前提としており，目的のデータではないデザイン部分が特に煩雑になっているので，XHTML に変換できても構造がパターン化されておらず，必要な部分のみを取り出すことが難しい．しかし，IE8 から追加される WebSlice では，HTML 上に部品として切り取るための目印を付けることを要求しているため，かなりの数のサイトが対応すると予想される．

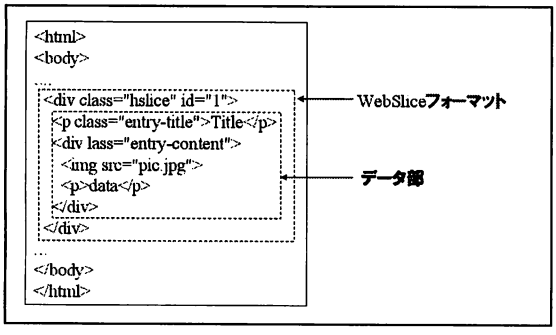

図 2　WebSlice データ例

この目印を利用すれば，必要な部分を容易に取り出すことができるため，HTML をより有効活用できる．

2.8. XML データの整形

WebShell では，XML データの仕様やスキーマを参照するのではなく，取得した XML データの構造を見てみて，「この辺が必要だ」と判断することを前提にしている．そこで，人が読み易いように XML データに改行やインデントを付ける機能を実装した．階層構造が読みやすければ，XPath の記述が容易になる．

```
<root xmlns="http://hogehoge"><child1><child2>data</child2></child1></root>
```
→
```
<root xmlns="http://hogehoge">
  <child1>
    <child2>data</child2>
  </child1>
</root>
```

データ部分に改行やスペースによるインデントを足すということは，厳密に言うと XML データとしての意味が変わってしまい，例えば XML 署名では検証が通らなくなってしまうので，整形したデータを扱う場合には注意する必要がある．

2.9. 名前空間

様々な XML データをマッシュアップすると，それぞれが別の名前空間を持っているので，操作が非常に煩雑になってしまう．名前空間は XML を扱う上で必要不可欠の技術ではあるが，XPath では prefix で記述できないために，指定が長くなってしまう．

今回の実装では名前空間に関する特別な実装は行わず，XPath での記述で対応してもらうこととした．

名前空間を指定する記述：

```
echo $aaa//*[local-name()='child1' and namespace-uri()='http://hogehoge']/text()
```

名前空間を指定しない記述：

```
echo $aaa//*[local-name()='child1']/text()
```

2.10. 他サイトの指定方法

コマンドラインで URL を指定するには，毎回フルパスで指定するだけでなく，『相対パス』でも指定できる．

現在のカレント URL を変更するのは，UNIX のシェルと同様に『cd 移動先の URL』を利用する．以降，URL を「相対パス」で指定した場合には，カレント URL と合成することで，入力する URL を省力化する．WebShell から見た場合，あらゆる URL に移動可能となる．

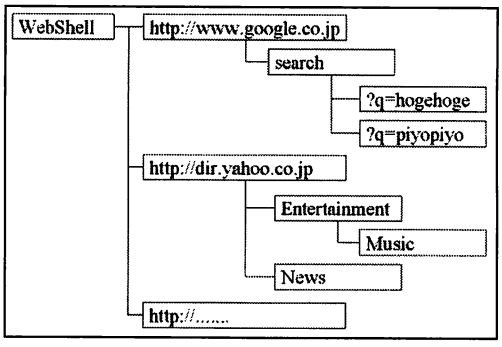

図 3　他サイトの参照イメージ

また，パイプなどを使って複数のサイトを使う場合にも，URLをフルパスで指定することで，様々なサイトを連続的に利用できる．

```
get http://www.google.co.jp/search?q=xxx | xhtml |
  get http://dir.yahoo.co.jp/+$con//xxx/text()
```

2.11. その他の機能

これら以外にも，以下の機能を実装した．

- コマンドヒストリ
- XSLT
- grep, replace などのコマンド
- JavaScript による機能拡張
- バッチ実行
- save, edit

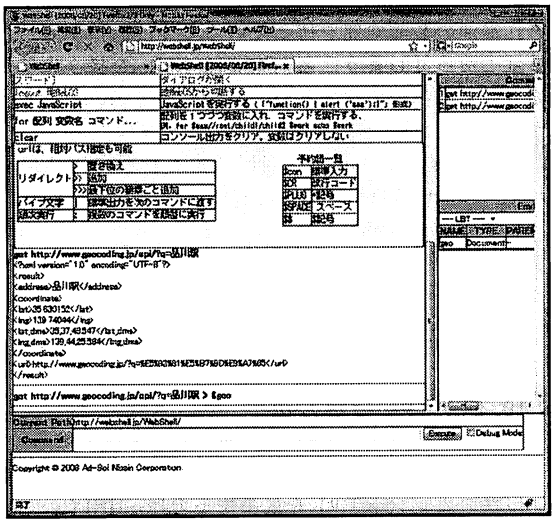

図4　WebShellのスクリーンショット

3. WebShellの動作例

WebShellを使ったマッシュアップの流れを『指定した場所から，近くて安いガソリンスタンドをWebサービスを使って検索』という例を使って解説する．

1．HTTPにより，指定したURLに対してGETコマンドを発行し，XMLを取得
2．変数に取得したXMLデータを保存
3．XPathによってXMLデータの部分を指定
4．HTTPリクエストに変数の内容を使う
5．リダイレクトによる変数の書き換え，整形
6．結果の表示

まず，「指定した場所（住所）から緯度・経度に変換」をおこなう．「Geocoding API」では，住所やランドマークから緯度・経度が取得できる．

```
get http://www.geocoding.jp/api/?q=品川駅 > $geo
```

HTTPによってInternet上の他サイトからXML, HTML, テキストデータを取得する．しかしJavaScriptには，ブラウザのドメイン境界制限があるため，自由に他サイトにアクセスすることができない．そこでJavaScriptからは自身のサーバにアクセスし，サーバから他サイトに対してHTTPでアクセスをおこなう．サーバ経由であれば，ブラウザと違って一切の制限がかからない．また，サーバからであればHTTPのGET, POSTメソッド以外にもPUT, DELETEのメソッドも制限なしに使える．

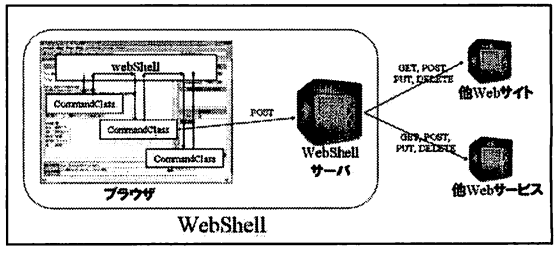

図5　他サイトへのアクセス

Geocoding APIは，以下のようなXMLを返してくれるので，それが変数geoに格納されている．

```
echo $geo

<?xml version="1.0" encoding="UTF-8"?>
<result>
  <address>品川駅</address>
  <coordinate>
    <lat>35.630152</lat>
    <lng>139.74044</lng>
    <lat_dms>35,37,48.547</lat_dms>
    <lng_dms>139,44,25.584</lng_dms>
  </coordinate>
  <url>http://www.geocoding.jp/?q=%E5%93%81%E5
%B7%9D%E9%A7%85</url>
</result>
```

変数geoから，緯度・経度だけをXPathで取り出す．

```
echo 緯度：+$geo//lat/text() 経度：+$geo//lng/text()

    緯度：35.630152
    経度：139.74044
```

その緯度・経度を使って，「ガソリン価格情報WEBサービスAPI」のURLを生成し，アクセスする．

「ガソリン価格情報WEBサービスAPI」では，指定した緯度・経度を中心に，半径10km内のガソリンスタンドの価格情報を取得できる．

```
get http://api.gogo.gs/v1.1/?apid=guest&lat=+$geo//l
at/text()+&lon=+$geo//lng/text() > $gas
```

取得した XML データが読みづらい場合は整形し，使いたい要素を探す．

```
xmls $gas

<?xml version="1.0" encoding="UTF-8"?>
<PriceInfo>
  <Version>1</Version>
  <ResultCountTotal>79</ResultCountTotal>
  <ResultCount>50</ResultCount>
  <Item>
    <ShopCode>0000000000</ShopCode>
    <Brand>OOOOOOOO</Brand>
    <ShopName>店名</ShopName>
    <Latitude>35.0000000000000</Latitude>
    <Longitude>139.000000000000</Longitude>
    <Address>東京都大田区○○x-x-x</Address>
    <Price>xxx</Price>
    <Date>2008/11/01</Date>
    <Photo>1111111111</Photo>
    <Rtc/>
    <Self/>
  </Item>
  <Item>
    <ShopCode>0000000000</ShopCode>
    <Brand>OOOOOOOO</Brand>
    <ShopName>店名</ShopName>
    <Latitude>35.0000000000000</Latitude>
    <Longitude>139.000000000000</Longitude>
    <Address>東京都大田区○○x-x-x</Address>
    <Price>xxx</Price>
    <Date>2008/11/01</Date>
    <Photo>1111111111</Photo>
    <Rtc/>
    <Self/>
  </Item>
  <Item>
  .....
```

変数 gas から店名（<ShopName>）だけを取り出すには，以下のコマンドを実行する．

```
echo $gas//PriceInfo/Item/ShopName

<ShopName>店名</ShopName><ShopName>店名</ShopName>....
```

XPath ではなく，テキストとして grep によって目的の要素を取り出すこともできる．

```
xmls $gas | grep <ShopName>

<ShopName>店名</ShopName>
<ShopName>店名</ShopName>
....
```

変数 gas から，XPath で○円以下の店数を取り出す．

```
echo $gas/count(//PriceInfo/Item[Price<○])
```

表示用の XHTML を作成する．

```
echo <html><body><table></table></body></html> | xhtml  > $html
```

xhtml コマンドは，HTML を XHTML に変換して DOM として返すコマンドなので，変数 html の中身は DOM となり，XPath による操作が可能になる．

条件に合致する子要素それぞれに対して，HTML の部品を作成し，変数 html の table タグの中に追加していく．

```
for $gas//PriceInfo/Item[Price<○] $work  echo <tr><td>+$work//ShopName/text()+</td><td>+$work//Price/text()+円</td></tr> | xmls >> $html//body/table
```

これで，２つの Web サービスを使って，『指定した住所から 10km 以内で○円以下のガソリンスタンドを一覧表示する XHTML』が作成できた．

```
xmls $html

<html>
  <body>
    <table>
      <tr>
        <td>店名</td>
        <td>○円</td>
      </tr>
      <tr>
        <td>店名</td>
        <td>○円</td>
      </tr>
    </table>
  </body>
</html>
```

XHTML 形式なので，ブラウザでそのまま表示できる．

```
view $html
```

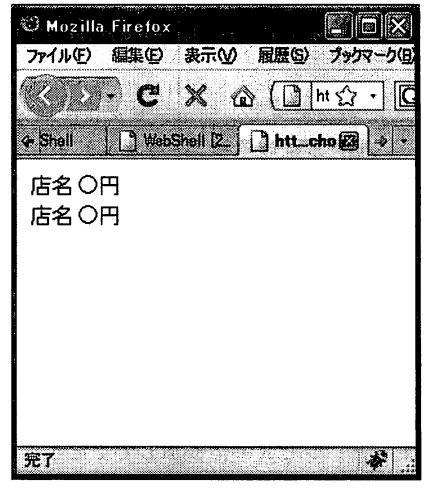

図５　ブラウザによる表示

これらの作業が，実際の XML データを確認しつつ Try and Error でコマンドを実行していくことができる．

4. 今後の拡張性

今後，UNIX と同様のコマンドや，JavaScript ライブラリを利用したコマンドを順次実装していくことで，WebShell の利便性を上げていくことができるだろう．

また，今回はテキストファイルに保存した内容をバッチ実行する機能を簡易的に実現したが，ブラウザ側で実行するので，本当の意味での自動化にはならない．サーバ側で実行できるようになれば，マッシュアップツールではなく，マッシュアップサイトを構築するインフラとしても WebShell を活用できるようになる．

5. まとめと考察

今回，WebShell では，変数内の XML データ（DOM）に対して XPath で特定の要素を指定し，リダイレクトによる参照・更新を実現した．UNIX のシェルでも使われている変数と，XPath という広く使われている記法を組み合わせることで，これらがコマンドラインインターフェイスに適した XML データの操作方法であることが確認できた．また，DOM の特徴を生かし，変数内の XML データの特定要素を別変数でも参照し，別変数に対する操作が元変数から見ても反映されることで，WebShell では UNIX のシェルにはない利便性を確認できた．

その直感的な XML データの操作性と，コマンドラインインターフェイスによる利便性と，シェルのヒストリによる実行履歴の参照を使うことで，Try and Error でマッシュアップを試行錯誤する環境が構築できた．

実際に WebShell を使ってみると，「HTML を XHTML に変換する機能」「XML データを整形する機能」によって，Web サイトの HTML を XHTML として見ることで，様々なことに利用できる可能性を感じた．

また，「XML を整形して，文字列として grep し，その結果を文字列として加工した上で，XML に戻す」という，新しい利用方法も見つかった．XML を DOM と文字列という 2 種類のデータ形式としてシームレスに扱えることで，より柔軟なデータ操作が実現できた．

6. 謝辞

WebShell 開発では，XML コンソーシアム・Web サービス実証部会にてレビューを行い，いろいろな知見を頂きました．レビューに参加してくださった皆さまには心から感謝いたします．

開発した WebShell は http://webshell.jp にて公開中です．

文　献

[1] WebShell 公開サイト
http://webshell.jp/
[2] Google
http://www.google.co.jp/
[3] Yahoo カテゴリ
http://dir.yahoo.co.jp/
[4] Geocoding API
http://www.geocoding.jp/api/
[5] ガソリン価格情報 WEB サービス API
http://api.gogo.gs/