XML Consortium

～ 第7回 XMLコンソーシアムWeek ～

Webサービス実証部会

WebOSの最新動向と開発者向け新感覚WebOSの開発

WebShell（仮称）の開発

２００８年６月６日
アドソル日進株式会社
荒本道隆

はじめに

XML Consortium

- WebOSを使った実証実験を行うためには、WebOSを改造して機能拡張する必要があります。しかし、既存のWebOSは、例えソースが公開されていたとしても、ハデハデなGUIやチャットなどを実現するために、非常に複雑なソースコードになっていて、改造に着手するまでが大変です。
- そこで、「いっその事、WebOSを作ってみよう」ということになりました。WebOSをより深く理解するためにも、実際に作ってみるのは非常に有意義でした。

開発動機

- WebOSの研究を行う上で、自由に改造できるWebOSが必要
  - aPlatでWebOS間連携など、さまざまな実証実験を行いたい
  - 条件
    - ソースを修正できる
    - 内部構造が単純で、影響範囲を把握し易い
  - ソースを公開しているWebOSもあるが、構造が複雑

無いなら作ればいいじゃないか

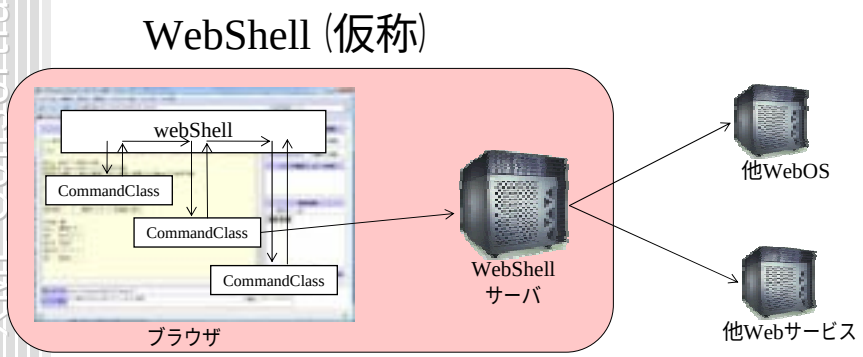
どんなものを作ろうか…

- XMLの操作に特化
  - Yahoo!PipesやPopfryで、本当にやりたいことができてますか？
  - UNIXでテキストファイルをgrepやawkを駆使して操作するように、XMLに対しても同じ事がスマートに行いたい
  - 操作→結果の確認→操作→…を効率的に行いたい
- 本当にGUIが必要なんだろうか？
  - 今までのWebOSは、ハデなGUIにばかり目が行ってしまう
  - GUIのデメリット
    - 操作が複雑、ツールごとに使い方を覚えなければならない
    - 見た目の部分に多くのコストがかかる
    - 同じことを繰り返すのが面倒、操作ミスする危険がある
- 趣味的な要素も多分にあります
  - キーボード中心の操作性
  - 実行結果の履歴参照
  - コマンドヒストリを呼び出して再実行

個人的見解ですが、
仕事で使うなら
CUIが一番

実行モデル

- ユーザーインターフェイスは CUI を採用
- ほとんどブラウザ側のJavaScriptで実現
- サーバ側は他サイトへのI/Oを中継するだけ



ブラウザ⇔サーバ間の通信内容

- html, JavaScriptのダウンロード
- 他サイトへの中継要求
  - POST /CUI_WebOS/ActionServlet?command=get&url=リクエスト先のURL
  - POST /CUI_WebOS/ActionServlet?command=post&url=リクエスト先のURL
  - POST /CUI_WebOS/ActionServlet?command=put&url=リクエスト先のURL
  - POST /CUI_WebOS/ActionServlet?command=delete&url=リクエスト先のURL
  - HTTPヘッダをそのまま中継
    - ただしWebShell自身の認証情報は削除
  - post と put は、HTTPボディも中継
  - RESTFul 対応
    - GET, POST, PUT, DELETE が発行できる
- ポーリングなどの複雑な仕組みは作らない
  - 構造が単純なので、機能追加する時に混乱しない
  - サーバ側はどんな言語にも移植可能
    - HTTPが使えて、GET, POST, PUT, DELETE が発行できる言語ならOK

複雑な処理は行わないので、サーバの負荷が少ない

機能紹介

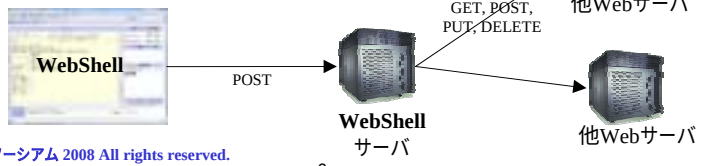
操作デモを交えて、
実装した機能について紹介します

基本機能

- 文字列の表示
  echo abcdefg
- 変数に、文字列, XML, 配列を入れることができる
  - 変数の初期化　set aaa abcdefg
  - 変数の削除　unset aaa
  - 参照時には先頭に$を　echo $aaa
  - XPath で要素指定　echo $xml//root/child1
  - 予約済み変数　$con, $CR, $PLUS, $SPACE, $$
- リダイレクト
  - 標準入力を置き換える　コマンド < abcdefg
  - 標準出力を変数に入れる　コマンド > $aaa ; コマンド >> $aaa
- 連続実行
  - 指定した順に実行　コマンド ; コマンド
  - パイプ　コマンド | コマンド

外部リソースへのアクセス

- HTTPで接続できれば何でもアクセス可能
  - cat http://www.google.co.jp/search?q=hogehoge
  - get http://www.google.co.jp/search?q=hogehoge
  - post http://localhost:8080/xmldb/archive/123/tags < name=value
- RESTFul 対応
  - get, post, put, delete
- カレントディレクトリの変更
  - cd http://www.google.co.jp/
    - 「get index.html」は「get http://www.google.co.jp/index.html」と同じ
- ファイル一覧の取得（WebOSのみ）
  - ls http://www.youos.com/aramoto/youos/



他WebOSへのアクセス－1

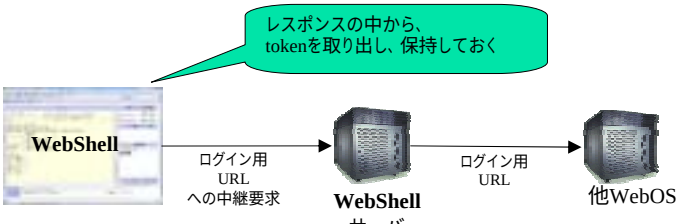
- 他WebOSへのログイン
  - login http://www.youos.com ユーザー名 パスワード
    - HTTPでは、以下のような通信をする

WebOSごとに、
ログイン方法が違う

```
GET http://www.youos.com/api?apiname=login&username=aramoto&password=xxxx

<apiresult name="login" status="OK" >
    <login auth="1" user="aramoto" token="TiZDPv...." cookie_domain=".youos.com" />
</apiresult>
```

- ログイン情報を保持
  - 関連するコマンド実行時に、ログイン情報を付加する



他WebOSへのアクセス－2

- 他WebOS上のファイルを取得
  - 通常のサイトへのアクセス

    cat http://hogehoge/**test.txt**

    ↓

    GET http://hogehoge/**test.txt**

    ↓

    ファイルの中身そのまま
  - WebOSへのアクセス（YouOSの場合）

    cat http://www.youos.com/**aramoto/youfs/test.txt**

    ↓

    GET http://www.youos.com/api?apiname=**fs_read**&path=**/aramoto/youfs/test.txt**&est=**TiZDPv....**

    WebOSごとに、書き方がまったく違う

    ↓

    ```
    <apiresult name="fs_read" status="OK">
        <read path="/aramoto/youfs/test.txt" filename="test.txt" mimetype="text/plain">
          <![CDATA[ファイルの中身]]></read>
      </apiresult>
    ```

    必要なデータはここだけ

WebOSの数だけ対応が必要

XMLの操作－1

- XMLデータを取得し、変数に入れる

  get http://hogehoge/aaa.xml > $xml

  ブラウザの evaluate() を使用
- XPathによる操作

  echo $xml//root/child1/child2[position()=1]

  echo $xml/count(//root/child1/child2)
- 一部分だけ取り出して別変数に入れる

  echo $xml//root/child1/child2[position()=1] > $xmlsub
- XMLの一部を書き換える

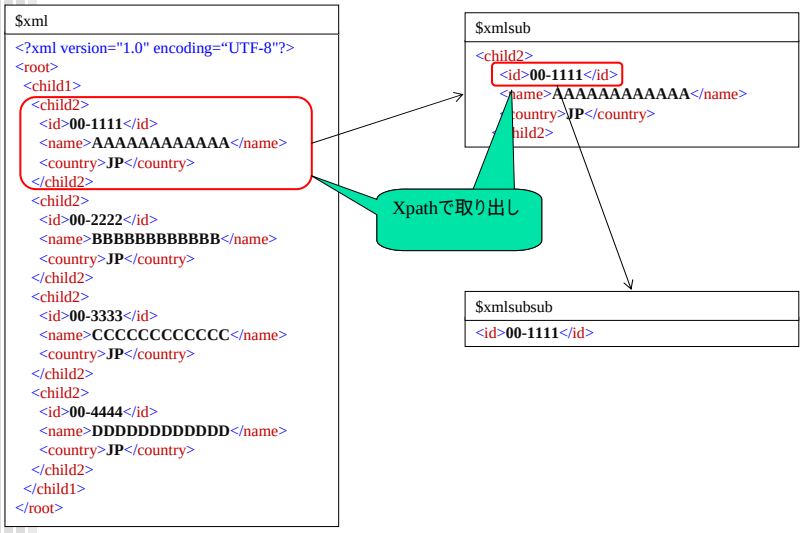
  echo 12345 > $xml//root/child1/child2[position()=1]/id/text()
  - echo 12345 > $xmlsub//child2/id/text() としても同じ

  echo $xml//root/child1/child2[position()=1] > $newxml//a/b

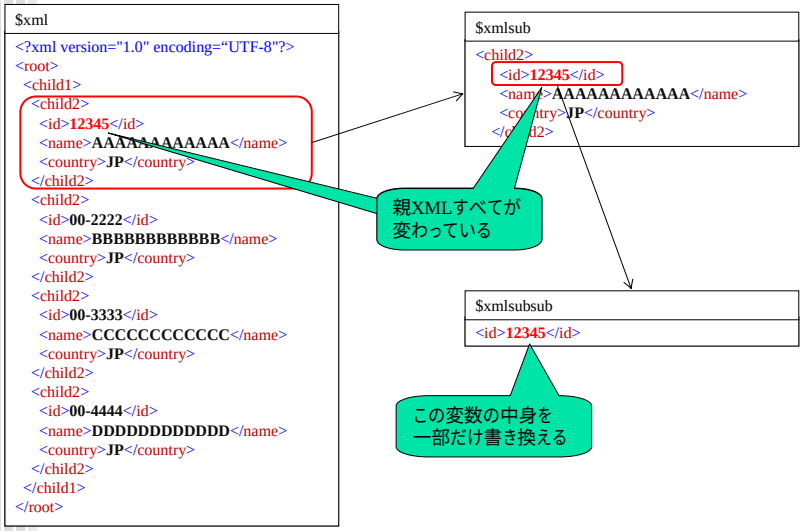
  echo $xml//root/child1/child2[position()=1] >> $newxml//a/b

  echo $xml//root/child1/child2[position()=1] >>> $newxml//a/b
  - >>> 最下位要素がすでにある場合は、もう1つ追加する。上記の例では要素 b を追加

XMLの操作－2



XMLの操作－2



ループ処理

- XMLの各要素に対して、コマンドを実行する

  for 配列 作業用変数 実行コマンド

  - 配列の要素1つづつを作業用変数に入れ、実行コマンドを呼び出す
  - 例. CSVを作成する

    for $xml//root/child1/child2 $work echo $work//child2/id/text()+,+$work//child2/name/text()+,,,+$CR >> $csv
  - 例. 別のサービスを呼び出し、結果を元XML内に埋め込む

    for $xml//root/child1/child2 $work get http://wikipedia.simpleapi.net/api?output=xml&keyword=+$work//child2/name/text() | echo $con//results > $work//child2/wikipedia

その他の機能

- 文字列
  - 一致する文字列を検索する — match regexp
  - 一致する文字列を含んだ行を検索する — grep regexp
  - 一致する文字列を置換する — replace regexp newString
- 整形
  - HTMLをXHTMLに変換する — xhtml
  - XMLを整形する — xmls
- 処理結果の利用と保存
  - 変数の中身を編集する — edit 変数名
  - ローカルにファイル保存する — save $xml
  - ブラウザの別窓に表示する — view $html
  - コンソールのクリア — clear

JavaScriptによる拡張

- JavaScriptのコードをそのまま実行

  exec function(){ alert ("test"); }

  exec $js
- WebShell内のコマンド呼び出し

  exec function(){ webShell.execute ("echo $xml//root/child1/child2"); }
  - 使えるコマンドに制限アリ
- WebShell内の変数参照

  exec function(){ var x=webShell.getenv("xml"); alert (x); }
- コマンド拡張
  - exec CommandClass.prototype.hello = function(args){ alert ("hello"); }
  - コマンドの追加、既存コマンドの拡張

コマンド一覧

- echo 値
- set
- unset 変数名

- cat url
- **get url**
- **post url**
- **put url**
- **delete url**
- cd url
- ls [url]

- **login 他WebOS [ユーザー名] [パスワード]**
- **logout 他WebOS**

- **リダイレクト** >, >>, >>>
- 順次実行 ;
- パイプ |

- match regexp
- grep regexp [regexp]...
- replace regexp newString
- xslt xml xsl
- xhtml [html]
- xmls [xml]

- **for 配列 一時変数名 コマンド...**

- edit [変数名]
- save [テキスト]
- view [テキスト]
- clear

- exec JavaScript

- 変数
  - $変数名
  - $変数名//XPath表記

制限事項

- 対応ブラウザは Firefox2 オンリー
  - ブラウザに依存している部分
    - XPathの使用
      - evaluate( )メソッド
    - ファイルに保存
      - location.href = "data:application/octet-stream,….";
  - ブラウザを限定することで、開発がスピードアップ
- 一部コマンドは、バッチ実行に未対応
  - cat, get, post, put, delete (別サーバにリクエストするもの)
  - for
  - exec

開発ボリューム

- 総ステップ数
  - HTML+JavaScript **2.2KStep**
    - prototype.js 使用
  - Servlet **0.2KStep**
    - ServletAPI 使用
- ほとんど JavaScript で実装
  - replace など、JavaScript の機能をそのまま使った物も
  - マルチブラウザ対応は大変そう
- 最初のバージョンは3日で完成
  - まずはサーバ無しで動作
    - IEのメリットを最大限に利用 → IEでしか動作しない
  - 「拡張性がない」ので全て作り直し → Firefox2オンリーに
  - あいている時間を見つけて作業、2ヶ月間で現状のレベルに
    - コードを書いている時間より、アイデアを考えている時間の方が圧倒的に長い

今後の予定

- 他WebOSへのアクセス機能を強化
  - 現状はYouOSの一部のコマンドのみ対応
- aPlatで検討している認証方式の実証実験
  - SAML, OpenID, OAuth, …
- サーバ側でのバッチ実行の実現
  - バッチファイルのURLをリクエストすると、実行結果を返す
    - サーバ側でJavaScript実行 or 他の開発言語への移植
    - サーバに、バッチファイルをPOST → マッシュアップアプリ
- 操作機能の強化
  - XMLの属性、名前空間の取り扱い、HTTPヘッダ、変数のCookie保存
- あと2回ほど作り直しをしたい
  - 機能拡張方法の整理、内部構造の整理、「>>>」の拡張、など
- 「永遠のアルファ版」
  - 実験的なトライを最優先

XML Consortium

応用アプリケーション編 に続きます

Shellとしての機能は少ないですが、
CUIによるインターフェイス＆XML操作に特化することで生まれる可能性について、デモを交えてご覧ください。