CD-ROM 自分のサイトで使えるスクリプトをCD-ROMに収録！

今すぐ実践！

最新のウェブテクノロジーが日本で使える

Amazonウェブサービス

オンライン書店の機能を自分のサイトに組み込もう！

昨年より米国で提供されてきたAmazon Web Services（AWS）が、7月から日本でも利用できるようになった。個人にはまだなじみのないウェブサービスだが、個人が使えるウェブサービスによって、ウェブサイトはどう変わるのだろうか？実際にAWSを利用して、その力を体験してみよう。

amazon.co.jp™ and you're done.™

Text&Script：平田 大治

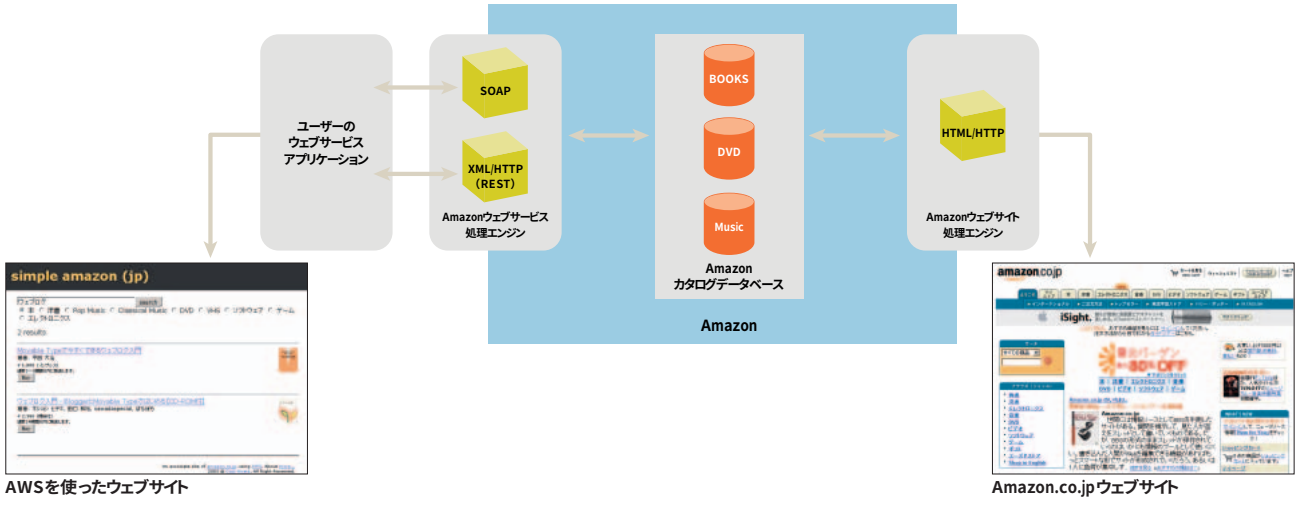
勉強編 期待のウェブサービスをすでに広く提供しているAWS

ウェブサービスって何だ？

ウェブサービスという言葉は、よく耳にするわりにはあまり理解されていない言葉の1つかもしれない。ウェブサービスとは、単なるウェブサイト上のサービスでもないし、ASP（アプリケーションサービスプロバイダー）でもない。ウェブサービスとは、ひらたく言えばコンピュータ同士が相互にやり取りをするための仕組みのことだ。

HTMLが中心のワールドワイドウェブと違って、ウェブサービスでは情報のやり取りにXMLを使う。通信もHTTPでなくても構わない。ウェブサービスは、要素となる技術の標準化がW3Cで進められており、将来は有望視されている。ウェブサービスが作られたのは、XMLの標準化が進み、ウェブの強力さが世の中を席捲していく中、ウェブの重要な要素である「疎結合」でのサーバー間通信を実現するためだった。「疎結合」とは、文字どおり「結合が密接でない」ことを言う。ウェブは、それぞれのサーバーは独立して動いているが、それぞれのページはハイパーリンクによって結合されている。そんな状態をイメージするといいだろう。疎結合であることのメリットは、相手のサーバーのことをあまり考えずに自分のサービスを構築できることだ。サーバーが密接に結合している状態では、1つのサーバーがダウンすると、他のサー

図1　Amazonウェブサービスを使えばAmazonの機能を自分のサイトで利用できる



AWSを使ったウェブサイト

Amazon.co.jpウェブサイト

バーも動作できなくなり、ひいては、サービス全体がダウンしてしまう。迷惑をかけないように作らないといけない。ウェブサービスであれば、落ちていればそのサービスは使えないが、他のサービスには、直接には影響しない。もちろん、リンク切れと同じように、寂しい思いをするのは間違いないし、下手な作り方をすれば、ほかにも影響は及んでくるだろうけど。

このような特徴を持っているウェブサービスは、企業間のシステムを接続するだろうとずっと期待されている。接続のための手順は標準化されているので、お互いがウェブサービスに対応していれば、今までのシステムでは必要だったような相互接続のための新規開発は不要だし、なによりも簡単に接続できるはず、そう言われてきた。しかしそれでもウェブサービスの話をあまり聞かないのは、実際に利用されている場面が意外と少なかったり、企業の中でしか利用されていなかったりするからだろう。

標準化されていると言っても、どのようにウェブサービスを使うか、お互いの企業が納得しなければ接続されることはない。また、既存のシステムをウェブサービスに対応させるのは、プログラムの作成といった面で見ればそんなに難しくないが、ウェブサービス化での負荷の増大など単純ではない。企業のシステムを他のシステムと接続するのであれば、セキュリティーも気になるだろう。いろいろな理由で、ウェブサービスは大きく普及しているとは言いがたい状態だ。

AWSを使ってアソシエイトの売り上げアップ

そんな中で、実際にウェブサービスを提供している最も有名な企業はAmazonだ。Amazonが「Amazonウェブサービス」(以下AWS)というウェブサービスを提供していることはニュースで知っている人は多くても、実際にAWSを利用したことがある人は日本ではまだ少ないだろう。

Amazonでは「アソシエイトプログラム」と呼ばれるアフィリエイトサービスが実施されているのはよく知られている。このアソシエイト、つまり自分のウェブサイトからAmazonに顧客を紹介することで紹介料を得る仕組みを利用している人は多い。AWSをうまく使いこなせば、自分のサイトをさらに情報が充実した魅力的なサイトにでき、アソシエイトの売り上げを伸ばすことも夢ではない。

実際に、AWSはAmazonそのものを提供するサービスで、Amazonが持つ機能、つまり商品検索やショッピングカートなど、さまざまな機能を提供してくれるため、米国では、すでにAWSを使って構築されているサイトが多数ある。サイトを作るためのプログラムまで市販されているほどだ。このようなサイトが多数作られることによって、Amazonは自分のサイトを越えて、さまざまな場面で利用してもらうことができる。これこそがウェブサービスの力だ。

図2　Amazonウェブサービスを利用してサービスを提供しているサイト

Amazon Light

とてもシンプルなアマゾンの検索画面を作って提供している。スタート直後は、Googleそっくりだったのだが、いつのまにか、現在のデザインに落ち着いている。

URL http://www.kokogiak.com/amazon/

Simplest-shop.com

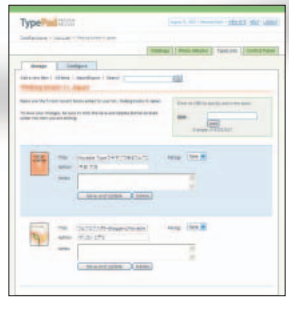
AWSだけで構成されたお店。カートなども装備されており、最後にチェックアウトするまでAmazonで買い物をすることになるとは気がつかないかもしれないくらい良くできている。

URL http://www.simplest-shop.com/

All consuming

ウェブログ上にある書評から本へのリンクを抽出し、ランキングを、その本の情報と一緒に表示している。それぞれの本を見るとアマゾンだけでなく、Googleのウェブサービスなどからも、情報を集めて表示している。

URL http://allconsuming.net/



TypePad

Movable Typeを作った Six Apartが運営するウェブログのホスティングサービス。TypeListと呼ばれるお気に入りリストを管理するための機能が搭載されており、本のリストは、AWSに連動している。すでに米国以外に、英国、ドイツ、日本のAWSにも対応している。

URL http://www.typepad.com/

基礎編

XML/HTTPでデータを要求してXMLデータを取得してみよう

ブラウザーだけでも試せるAWSでのデータ取得

AWSには2つのインターフェイスが用意されている。1つはWSDL/SOAP（Simple Object Access Protocol）を使った標準的な規格のウェブサービスだ。もう1つは、XML/HTTP、またはREST（REpresentational State Transfer）と呼ばれる簡易なインターフェイスだ。XML/HTTPを利用すればウェブサービスを簡単に使えるので、少し試してみよう。

IE 6などのXMLを扱えるブラウザーで図3のURLを入力してみてほしい（付属CD-ROMのHTMLからリンクをたどってもよい）。これは、AWSに「4844318128」というISBNコードを伝えて筆者の『Movable Typeで始めるウェブログ入門』（以下『ウェブログ入門』）の書籍情報を取得するURLだ。AWS経由でXMLが取得されて、解析された状態でブラウザーに表示されているだろう（図4）。

次はAWSで検索してみよう（図5）。「KeywordSearch=kitten」でキーワード「kitten（子猫）」を、「mode=books-jp」で検索対象に日本の書籍を指定している。すると、キーワードにkitten（子猫）が含まれる和書の情報が同様にXMLで取得できる（図6）。

図3　ISBNコードを指定してAWSからXMLデータを取得するURL

デベロッパートークン　アソシエイトID

http://xml.amazon.com/onca/xml3?dev-t=D1J99VU6RMQJZ6&t=dh0dc-22

日本語のheavy版XMLデータを指定　1ページ目の情報　ISBNの指定

&type=heavy&f=xml&locale=jp&page=1&AsinSearch=4844318128

図4　AWSから得た書籍情報のXMLデータと、その書籍のAmazon.co.jpでのHTMLページ

XML/HTTPインターフェイスを利用して取得したXMLファイルには、URLで送った情報に加えて書籍の情報が入っている。Amazonのウェブサイトでこの書籍の情報を表示すると書籍の情報以外にもウェブサイト内のナビゲーションやその他の情報が入っている。

SOAPよりも手軽なXML/HTTPウェブサービスを使ってみよう

AWSでは、SOAPとREST（XML/HTTP）の2つのインターフェイスが用意されているが、その違いは何だろうか。

ウェブサービスのプロトコルとしてよく名前を聞くのはSOAPだろう。SOAPの場合は、サーバーへの問い合わせのためのSOAPメッセージがXMLで作成される。受けとったサーバーは、XMLでできたメッセージを処理し、それに対応した結果をSOAPメッセージにして返信する。

SOAP自体はメッセージ交換のためのプロトコルで、HTTPだけでなく、FTPやSMTPを介してもやりとりできるが、AWSでは普通にHTTPを使っている。すでにいろいろな言語で、SOAPを利用するためのライブラリー群が提供されているので、WSDL（Web Services Description Language）なども合わせて利用すれば、ウェブサービスとして提供されている機能を簡単に使える。

一方REST（XML/HTTP）では、リクエストをURLとして構成し、結果をXMLで得る。本文の例で見たように、ブラウザーからでも簡単にアクセス可能であり、サーバーにとっても負荷が軽いため、大量のアクセスが想定される場合には有効なインターフェイスとなる。

SOAPとREST、どちらがいいと決めることは簡単ではない。しかし、個人で簡単にプログラムを作ったり、AWSのようなウェブサービスをちょっとだけ利用したりするだけであれば、RESTを使うほうがはるかに簡単なのは間違いないので、今回は、RESTを利用して、自分のサイトにAWSを組み込んでみる。

図5 キーワードを指定してAWSからXMLデータを取得するURL

デベロッパートークン　アソシエイトID

http://xml.amazon.com/onca/xml3?dev-t=D1J99VU6RMQJZ6&t=dh0dc-22

和書を指定　日本語のlite版XMLデータを指定　1ページ目の情報

&mode=books-jp&type=lite&f=xml&locale=jp&page=1

キーワード「Kitten」を指定して検索

&KeywordSearch=kitten

図6 AWSから得られた検索結果のXMLデータと、Amazon.co.jpで同じ検索をした場合のHTMLページ

「kitten」のキーワードでAmazonを検索した場合。AWSで取得したXMLと、Amazonのウェブサイトで検索したときの画面を比べてみてほしい。

XML/HTTPのウェブサービスでは、このようにCGIのクエリー文字列と同様な方法でデータを要求して、XMLで返されたデータをプログラムで利用するのだ。

取得したXMLデータをプログラムから利用するのだ

普通にAmazonを見ているときのようにHTMLで得られるページとどう違うのか、疑問を感じる人もいるかもしれない。ウェブサービスを利用して得られるXMLのデータはHTMLと違って、プログラムから簡単に利用できるところが違いだ。HTMLはブラウザーで美しく表示するための情報なので、プログラムがデータのやり取りを行うには不要な情報が多数含まれていて、自分の必要とする情報を抽出することは簡単ではない。一方、XMLで情報を取得できれば、その中から必要な情報を取り出すことは簡単だ。Perl、PHP、Python、Ruby、Java、その他多くのプログラミング言語では、XMLを扱うことは難しいことではなくなっている。これらの言語を使って、Amazonのサービスを自分の好きなように利用できるのだ。

AWSを利用するための準備

AWSを使うためには、まずAmazonからデベロッパートークンを入手する必要がある。これはAWSを使うための許可書のようなものだ。全世界共通なので、AWSを使ったいいサービスを開発すれば、Amazonがある地域すべてで同じものが利用できる。現在米国、日本以外にも、イギリス、ドイツで利用可能で、カナダ、フランスが今年中に対応する予定だ。

デベロッパートークンは、アソシエイトプログラムの参加者に配布されるアソシエイトIDとは異なる(アソシエイトIDは、国別に取得しなければならない)。

① Amazonウェブサービスのページをブラウザーで開く。
URL http://www.amazon.co.jp/webservices/

② ページの左側にあるリンクから「Licensing Agreement(日本語版)」をクリックして使用許諾条件の内容を確認しておく。

③ 元のページに戻り、「2.無料デベロッパー・トークンを申し込みます」のリンクをクリックする。

③ 英語のフォームが表示されるので、「Your e-mail」にメールアドレスを入力し、パスワードを決めて入力する。

④ 表示されている使用許諾条件はさきほど確認した日本語版と同じ内容なので、内容に問題がなければ下にある「I have read and accepted……」のチェックボックスをチェックして「Accept Terms & Conditions」のリンクをクリックする。

⑤ 「D3BCX432G4VZA5」のような文字列が表示される。これがデベロッパートークンだ。登録したメールアドレスにもメールでトークンが送られるのでなくさないように保存しておこう。

⑥ 必須ではないが、最初のページから参考のためにAWS SDKをダウンロードしておこう。ドキュメントとサンプルプログラムだ。ドキュメントは英語だが、わかりやすく書かれている。

⑦ 必須ではないが、まだアソシエイトIDを持っていなければ次のページから申し込んでおくといい。
URL http://www.amazon.co.jp/associates/

PHPで自分のサイトにAWSを組み込んでみよう

CD-ROMのサンプルスクリプトを導入してみよう

PHPを使って自分のページにAWSを組み込むためのスクリプトを用意した。付属CD-ROMに入っているAsinSearch.phpとKeywordSearch.phpを自分のサーバーにアップロードしてそれぞれ実行してみてほしい。実際にはamazon.phpにAWSの処理をする関数があり、それを呼び出しているので、amazon.phpも同じ場所にアップロードしておく必要がある。

ISBNを指定して書籍を検索してみよう

まずは指定したISBNで書籍の情報を取得して簡単な情報を表示するスクリプトから始めてみる(図7)。付属CD-ROMに入っているAsinSearch.php(図8)を実行してみよう。先ほども紹介した『ウェブログ入門』のISBNで取得した情報から書籍の表紙画像とAmazonページのURLを取り出して、表紙画像にリンクを付けて表示する(図9)。

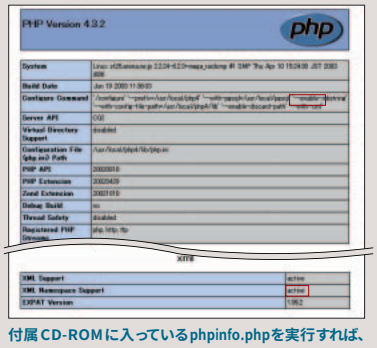
図8のAsinSearch.phpを詳しく解説しよう。中心となるのはamazon.phpスクリプトにあるAmazonSearchという関数だ。この関数は、引数を適切に与えるとAWSを呼び出し、得られた結果をXMLパーサーを通して$productsというグローバル変数に返す。

まず、amazon.phpを読み込み❶、必要な変数を設定する❷。変数はそれぞれデベロッパートークン(dev-t)、アソシエイ

スクリプトを試す前にココに注目!

CD-ROMに収録したスクリプトは自由に改変して使ってもらって構わない。ただし、スクリプトを利用した結果については筆者も編集部も責任を負わないので自己責任で利用してほしい。

CD-ROMに収録したスクリプトを利用するには、XMLパーサーが使えるPHP4が自分のサイトで動いていることが前提となる。



付属CD-ROMに入っているphpinfo.phpを実行すれば、システムでXMLパーサーを使えるかどうか確認できる。冒頭のConfigure Commandに--with-xmlが含まれるか、ずっと下の方のxmlのブロックでXML Supportがactiveになっていれば問題ない。

スクリプトがうまく動作しない場合は次の点に注意してほしい。

- デベロッパートークン
  スクリプト中で「D1J99VU6RMQJZ6」となっている部分は自分の取得したデベロッパートークンに置き換える必要がある。
- アソシエイトID
  アソシエイトIDを持っているならば、スクリプト中で「dh0dc-22」となっている部分を自分のアソシエイトIDに置き換えるといいだろう。
- PHPのパス
  サーバー環境によっては、スクリプトの先頭で「#!/usr/bin/php4」のようにPHP4のパスを指定しなければいけない場合がある。

図7　1冊の書籍の情報を取得して表紙とリンクを表示する

図8　指定したISBNの情報を取得してページを作るAsinSearch.php

```
<?php
    include_once('amazon.php');                                    ❶
    $asin_element_list = array( 'dev-t', 't', 'type', 'f', 'locale', 'AsinSearch' );

    $q["dev-t"] = "D1J99VU6RMQJZ6";
    $q["t"] = "dh0dc-22";
    $q["type"]= "lite"; # "heavy" or "lite"
    $q["f"] = "xml";
    $q["locale"] = "jp";

    $q["AsinSearch"] = '4844318128';                               ❷

    AmazonSearch($q, $asin_element_list);                          ❸
    $p = $products[0];                                             ❹

    print "<a href= "$p[url] " alt= "$p[name] "><img src= "$p[imageurlsmall] "   ❺
alt= "$p[name] " /></a> n";
?>
```

図9　AsinSearch.phpを実行すると書籍の表紙がリンク付きで表示される

トID(t)、情報の種類(type)、フォーマット(f)、地域(locale)に、それぞれ適切な値を設定する。検索に使うのはAsinSearchという値だけなので、これに今回は、筆者の本のISBNコード「4844318128」を設定する。準備はこれだけだ。

そして、❷で設定した変数を与えてAmazonSearchを呼び出すと❸、グローバル変数$productsに配列として商品情報が格納される。今回は、商品は1つなので、$products配列の最初の要素を取り出し❹、目的の本のサムネールを出力する❺。

キーワードを指定して書籍を検索してみよう

次にKeywordSearch.phpを試してみよう(図10、図11)。今度はISBNを指定するのではなくキーワードとして「ネコ」を指定して検索して、表紙イメージと書名を表示してリンクを張っている(図12)。

図11のKeywordSearch.phpを詳しく解説しよう。最初の2行で漢字コードの設定を行っている❶。AWSでは、入出力にUTF-8を利用しているため、ここでは内部コードもUTF-8として扱い、ウェブサイトの出力もUTF-8にしている。

検索の種類ごとに必要となる引数が異なるため、KeywordSearch用の要素を設定している❷。また、今回は、検索結果が複数になる可能性がある。AWSは結果が多数になった場合、自動的にページ分けして結果を返してくるので、何ページ目を取得したいのかを設定する必要がある。最初は1を指定する❸。

次に、AWSへ与える他の値を設定する❹。「mode」には検索したい商品の種類を設定する。商品の種類には、和書(books-jp)、洋書(books-us)、CD(music-jp)、DVD(dvd-jp)、ソフトウェア(software-jp)などがある。KeywordSearchには、今回「ネコ」というキーワードを設定する。AWSでは日本語

図10　KeywordSearch.phpで得られる検索結果は複数の場合もある

図11　キーワード「ネコ」で検索するKeywordSearch.php

```
<?php
    mb_internal_encoding("UTF-8");                                    ❶
    mb_http_output('UTF-8');

    include_once('amazon.php');
    $keyword_element_list = array( 'dev-t', 't', 'type', 'f', 'locale', 'KeywordSearch', 'mode', 'page' );   ❷
    $script = "KeywordSearch.php";

    $q["page"] = ($_REQUEST["page"]) ? $_REQUEST["page"] : 1 ;         ❸

    $q["dev-t"] = "D1J99VU6RMQJZ6";
    $q["t"] = "dh0dc-22";
    $q["type"]= "lite"; # "heavy" or "lite"                           ❹
    $q["f"] = "xml";
    $q["mode"] = "books-jp";
    $q["locale"] = "jp";

    $q["KeywordSearch"] = utf2entity("ネコ", true);                   ❺

    AmazonSearch($q, $keyword_element_list);                          ❻

    $paging = '';
    if ($totalpages > 1) {
    if ($q[page]>1) { $paging = sprintf( "<a href= "%s?page=%s ">(prev)</a> ", $script, $q[page]-1);}
    for ($p=max(1,$q[page]-9); $p <= min($q[page]+9,$totalpages); $p++) {
        if ($p == $q[page]) {
        $paging .= $p;
        }
        else {
        $paging .= sprintf( "<a href= "%s?page=%s ">%s</a> ", $script, $p, $p);
        }
    }
    if ($q[page]<$totalpages) { $paging .= sprintf( "<a href= "%s?page=%s ">(next)</a> ", $script,
$q[page]+1);}                                                         ❼
    $paging .= "<br /> n";
    }

    if ($totalresults) {
        if ($totalresults > 1){ $s = 's'; }
    echo "$totalresults result$s<br /> n";
    foreach ($products as $p) {
            print "<hr /><a href= "$p[url] " alt= "$p[name] "><img src= "$p[imageurlsmall] "
alt= "$p[name] " border= "0 " />$p[name]</a><br />  n";               ❽
        }
    echo "<br />$paging <br /> n";
    } else {
    echo "No result<br /> n";
        $q[results] = 0;
    }
?>
```

図12　KeywordSearch.phpを実行すると該当する書籍の簡単な情報が一覧で表示される

を使う場合には、日本語をHTMLエンコードしたあとで、「%」を「%25」に変換しなければならない。そのためのutf2entityという関数をamazon.phpに用意して使っている❺。

❻で検索を行うと、今度は$productsに加えて、$totalresultsと$totalpagesの2つの変数にも値が設定される。$totalresultsには検索結果の総数が、$totalpagesには検索結果の総ページ数が設定されるので、この数字を利用してページの送りや戻しの表示を行うことができる❼。

実際の検索結果の表示は❽で行っている。このprint文を変更すれば、表示を自由にカスタマイズできる。

amazon.phpはこうやってAWSを処理している

実際にウェブサービスを呼び出してXMLデータを処理しているのはamazon.phpだ(図13)。誌面の都合ですべてのコードは掲載していない。省略されている部分のコードについてはCD-ROMに収録したスクリプトを参照してほしい。

AWSを使うにあたっての注意点

AWSの使用許諾条件の中のいくつかの点について簡単に説明しておく。ほかにもさまざまな条件があるので、必ず使用許諾条件の内容を確認してほしい。

- 1秒1コール:AWSは24時間365日利用可能なサービスだが、アクセスは1秒間に1回の制限がされている。
- キャッシング:AWSから得らえた情報は、キャッシュして再利用することが望ましいが、キャッシュの期限は24時間以内にしなければならない。
- テキストの編集:文章を短くするのはいいが、それ以外の編集は行ってはいけない。勝手に付け加えたりしてはいけない。
- リンク:AWSで取得した情報を表示するときには、すべてアマゾンへリンクさせなければいけない。
- 転用の禁止:AWSから取得した情報から、別のページへリンクさせてはいけない。

図13 AWSを処理する心臓のamazon.php(一部略)

```
<?php
mb_internal_encoding("UTF-8");
mb_http_output('UTF-8');

function AmazonSearch ($q, $elements) {                    実際にデータを取得するAmazonSearch関数
    global $DEBUG, $products;

    $products = array();                                   ❶
    $xmlurl = "http://xml.amazon.com/onca/xml3";

    $query_cgi = array();
    foreach ($elements as $key) {
        array_push($query_cgi, "$key=$q[$key]") ;          ❷
    }
    $url = $xmlurl . "?" . join("&", $query_cgi);

    $result = @file($url);                                 ❸
    $data = join( "", $result );

    if ($DEBUG) {echo "<!-- ", $data, " --> n"; }

    $parser = xml_parser_create();
    xml_set_element_handler($parser, "startElementHandler", "endElementHandler");   ❹
    xml_set_character_data_handler($parser, "cdataHandler");

    if(!xml_parse($parser, $data)){
        die(sprintf("XML error %d %d",
        xml_get_current_line_number($parser),              ❺
        xml_get_current_column_number($parser)));
    }
}

function startElementHandler($parser, $name, $attrib){……省略……}   ❻   XMLパーサーのコールバック関数
function endElementHandler($parser, $name){……省略……}              ❼
function cdatahandler($paser, $data){……省略……}                     ❽

function display_result ($products) {……省略……}            ❾   HTML出力を生成する関数
function add_cart ($asin) {……省略……}                      ❿

function utf2entity($str, $amazon = false) {               日本語のエンコードをするutf2entity関数
    # converting $str to HTML numeric entities
    # if $amazon = true, additionally converting from '%' to '%25'.
    if (mb_detect_encoding($str) == 'UTF-8') {
        for( $i=0; $i < mb_strlen($str); $i++ ) {
        $q = mb_substr($str, $i, 1);
        if ( strlen($q) == 1 && ord($q) < 0x80 ) {
            $qq .= $q;
        } else {
            for ( $j=0; $j < strlen($q); $j++ ) {
            $format = ($amazon == true) ? "%%25%02X" : "%%%02X";
            $qq .= sprintf( $format, ord(substr($q, $j, 1)));
            }
        }
        }
    } else { $qq = $str; }
    return $qq;
}                                                          ⓫

function sanitize ($str) {……省略……}                       ⓬   セキュリティー確保のためのsanitize関数
?>
```

AmazonSearch関数は、実際にREST（XML/HTTP）でAWSを使って情報を取得する重要な関数だ。引数としてCGIに渡すパラメーターの連想配列$qと、どの要素を実際にAWSに送信すればよいかを指定する配列$elementsを取る。

まず、商品データを格納する配列$productsを初期化する❶。次に$elementsに指定された要素を、$qから取りだし、AWSへ問い合わせができるように整形する❷。そして、実際にURLから情報を取得し❸、XMLパーサーを初期化したあと❹、結果を解析する❺。解析された結果は、XMLパーサーが直接グローバル変数$productsに格納する。

続く3つの関数startElementHandler❻、endElementHandler❼、cdataHandler❽は、XMLパーサーのハンドラーで、AWSから得られるXMLの要素に合わせて記述している。

関数display_result❾は、$productsに格納された情報にHTMLタグを付けて整えた形で出力するための関数だ。XMLパーサーで解析された商品情報は連想配列に格納されているので、それぞれの要素を使ってHTMLを出力している。XSLTに慣れていれば、AWSから得られるXMLを使って変換してもよいだろう。AWSから得られる商品情報では、本であれば「author」、CDなどでは「artist」と商品群ごとに微妙に異なる要素が用いられる。それぞれの場合に合わせた表示にする必要もあるだろう。また、商品によって、特に絶版の商品などは、価格情報だけを信じるのは危険である。このスクリプトでは、価格のないもの、もしくは1円に設定されているものは表示しないようにしている。

関数add_cart❿は同じくHTML出力のためのもので、関数display_resultで使っている。引数で与えられたASINに対応する商品をアマゾンのカートに追加するためのフォームの文字列を生成するものだ。

関数utf2entity⓫は、utf8の文字列から、数値エンティティーを作成するために利用する。またAWSでは数値エンティティーに変換した後、URLに含まれる「%」をさらに「%25」に変換しなければならないことを思い出してほしい。2つ目の引数にtrueを設定すると、AWSに合わせた変換が行われる。

関数sanitize⓬はフォームに入力された特殊記号を除去するための関数だ。これらの記号はスクリプトの誤動作やセキュリティー上の問題を引き起こすことがある。実際の設置にあたっては、特にフォームを利用して検索窓を作成する場合など、十分に注意してほしい。

考えるのはあなた

XML/HTTPを利用すると、とても簡単にアマゾンウェブサービスを利用できることがわかってもらえたと思う。今回はAWSを体験することを目的としたため、AWSへの問い合わせや得られるデータのフォーマットを詳細には解説していない。これらの情報についてはAWS SDKのAPIガイドを参照してほしい。

データを要求する仕組みと取得したデータの処理の方法がわかれば、さらに検索のキーワードをフォームで入力できるようにしたり、商品ジャンルを選べる機能を追加したりすると、シンプルながらもかなり使えるAWSサイトになるはずだ（図14）。

しかし、単純にAWSを呼び出して、その結果を表示するだけであれば、スクリプトを数十行書くだけでよい。しかし、それだけではおもしろいサービスにはならない。自分で工夫して、他のサービスや情報とうまく組み合わせることで、AllConsuming.netなどのように、新しいサービスとして認められるようになる。逆に新しいサービスを考えついたとき、それをすべて、自分で一から作る必要もない。AWSを利用すればアソシエイトの売り上げをアップする仕組みを簡単に作ることができるかもしれないのだ。

Movable TypeからAWSを利用するプラグイン

ウェブログでも自分で読んだ本やお気に入りの音楽をネタにするときに、AWSを使えると内容が充実する、と考えるのは自然なことだろう。すでにMTAmazonというプラグインが存在しているのだが、残念なことの、AWS 2.0に対応したバージョンしか発表されていない（日本に対応しているのはAWS 3.0）。そこで簡単ではあるが、筆者が同様のプラグインを作成した。よかったら使ってほしい。

URL http://amazon.uva.ne.jp/log/

HTMLページのJavaScriptからPHPを呼び出すには

PHPを利用することはできるが、PHPのコードを直接組み込みにくいページの場合は、JavaScriptを利用するといいだろう。asin.phpは、出力結果を一旦JavaScriptにして出力するスクリプトの例だ。ウェブページへの組み込みは、asin.htmlでしているように、

```
<script language="javascript" type="text/ javascript" src="asin.php?asin=4844318128"></script>
(asin.html)
```

のようにする。「asin.php」を自分が設置したスクリプトの名前に、「asin=……」のコードを自分の表示したいASINに設定すれば、他の商品を呼びだすこともできるし、asin.phpの出力部分を変更すれば、自分のページに合わせた、別のフォーマットを出力することもできる。埋めこみ先のhtmlファイルの漢字コードがUTF-8以外の場合は、「?asin=xxx」の後ろに「&c=euc」のようにコード指定するとコードが変換される。SHIFT JISの場合は「sjis」を設定すればよい。

図14　キーワード入力機能を追加してデザインを加えれば立派なAWSサイトの完成だ

［インターネットマガジン　バックナンバーアーカイブ］ ご利用上の注意

このPDFファイルは、株式会社インプレスR&D（株式会社インプレスから分割）が1994年～2006年まで発行した月刊誌『インターネットマガジン』の誌面をPDF化し、「インターネットマガジン　バックナンバーアーカイブ」として以下のウェブサイト「All-in-One INTERNET magazine 2.0」で公開しているものです。

http://i.impressRD.jp/bn

このファイルをご利用いただくにあたり、下記の注意事項を必ずお読みください。

- 記載されている内容（技術解説、URL、団体・企業名、商品名、価格、プレゼント募集、アンケートなど）は発行当時のものです。
- 収録されている内容は著作権法上の保護を受けています。著作権はそれぞれの記事の著作者（執筆者、写真の撮影者、イラストの作成者、編集部など）が保持しています。
- 著作者から許諾が得られなかった著作物は収録されていない場合があります。
- このファイルやその内容を改変したり、商用を目的として再利用することはできません。あくまで個人や企業の非商用利用での閲覧、複製、送信に限られます。
- 収録されている内容を何らかの媒体に引用としてご利用する際は、出典として媒体名および月号、該当ページ番号、発行元（株式会社インプレスR&D）、コピーライトなどの情報をご明記ください。
- オリジナルの雑誌の発行時点では、株式会社インプレス R&D（当時は株式会社インプレス）と著作権者は内容が正確なものであるように最大限に努めましたが、すべての情報が完全に正確であることは保証できません。このファイルの内容に起因する直接的および間接的な損害に対して、一切の責任を負いません。お客様個人の責任においてご利用ください。

このファイルに関するお問い合わせ先
株式会社インプレスR&D
All-in-One INTERNET magazine 編集部
im-info@impress.co.jp