



XML

をおいしく味わおう！



Episode 16 (最終回) : Java の復習

前回まで数回にわたって、.NET 環境での XML の扱い方に触れました。今回は、話題を Java に戻していきます。Java で一通り XML の扱い方を学べましたので、今回は Java の応用的な使用方法である、「Web サービス」での Java の使い方に関して学んでいきましょう。

あれよあれよというまに、この連載も 16 回目になってしまいました。残念なことに (?) 今回で、この連載には一度区切りをつけることになりました。いままで熱心に読んでくださった皆様、ほんとうにありがとうございました。

この雑誌がお手元に届くころには、すっかり夏になっているでしょうか。。。原稿を書き始めるちょっと前¹になりますが、今年は各地の桜を見て回ってきました。その中でも良かったのは、弘前でしょう。

図 1 が、弘前城の桜です。写っているのはやぐらではな



図 1: 弘前城の美しい桜

1. 6 ページの原稿書くのに何ヶ月掛かっているんだ、ってつつ込みは無しで、お願いします。

く、天守閣 (ち、小さい...) なのですが、その周りをぐるっと、桜が囲んでいてとても美風景でした。写真には写っていませんが、後ろを向くと、青森の富士山こと岩木山も見えたりして、なんとも癒される風景なのでした。

話は変わって、本題に入ります。今回は、再び Java の話題を扱ってみたいと思います。スターウォーズ最新作、Episode3 ~シスの復讐~ の人気にあやかって「Java の復習」なんてタイトルを考えてみたんですが (こういう解説を加えてしまうと単なるオヤジギャグになりますが、、、)、特に深い意味はありません (笑) .NET の話題に触れてきたんですが、この雑誌のほかのコーナーで .NET を扱う話題が出てきたので、私は Java に戻ってもいいかなあと考えてまして、再び Java の話題に戻ります。

Java で XML を扱う技術はどのように進歩してきたのでしょうか。最近、基礎技術 (DOM や SAX) はすっかり浸透してきたようで、あまり話題に上らなくなりました。その代わりに、応用技術が注目を集めています。と、いうことで、最後に基礎技術から離れて応用技術の 1 つである、Web サービスの使用方法に関して、解説を行っていきたいと思います。

Web サービス??

さて、Java での実装方法をお伝えする前に、Web サー

ビスってなんだろうってあたりと、それを支える技術に関して触れておきたいと思います。Web サービスとは、ネットワークを介して、様々なアプリケーションを相互接続できるようにするための技術、といえるでしょう。もちろん、同じような技術に MS の DCOM や Java の RMI などの技術も存在していますが、それらとの大きな違いは、実装言語や実行環境 (OS やハードウェアなど) にできるだけ依存しないように工夫されているという点です。前述の技術は OS であったり、特定の開発言語でなければ通信できない、というデメリットが存在しました。Web サービスであれば、環境に依存しない仕様になっていますので、ありとあらゆるアプリケーションに接続が可能となるのです。では、デメリットはないのでしょうか? 一般的に、「実行時の速度」がデメリット²になり得ます。環境に依存しないように作られているわけですので、開発言語に依存した手法等を使用できない場合もありますので、どうしても実行速度は犠牲にならざるを得ません。また、Web サービスで用いられるプロトコルも特殊なものではなく、汎用的に使用されるものが多いため、同様のことが言えます。

少し前の文献を見ますと 大抵 Web サービスというのは、3つの技術から構成されることになっています (図2)。それは、Web サービスを利用するためのプロトコル、その利用方法の内容を規定・公開するためのインターフェース記述言語、広く一般にインターフェースを公開するための、公開検索ディレクトリサービスです。ぶっちゃけていいますと、公開検索ディレクトリサービスはほとんど使われていません。詳しくは後述しますが、いろいろと問題も多く今では2つの技術でほとんど成り立っています。

まず、プロトコルに関してですが、SOAP³ というプロトコルが使用されます。このプロトコルは XML ベースのプロトコルで、メッセージの包み方と配達先の記述方法を規定しているプロトコルです。まあ、よく使われるたとえば、郵便にたとえると、封筒での手紙の包み方と、封筒

2. 最近では、ハードウェアや回線の速度も上がってきましたので、それほどデメリットとはならないかも知れませんが、... 1秒でも早く、という場面に Web サービスは向かないでしょう、
3. 一昔前まで、SOAP は、「Simple Object Access Protocol」の略称ってことになっていたんですが、最近では SOAP はあくまで SOAP ってことだそうです。呼び方が変わったからといって、その本質は変わらないですけどね、

への宛名の書き方などが決まっています (図3)。メッセージの内容、つまり封筒に包まれる手紙の内容に関しては、XML 形式であれば、記述形式などを問わないことになっています。

SOAP は、HTTP や SMTP といったメッセージを配達するトランスポートプロトコルの上に規定されているプロトコルとなっています。一般的には HTTP 上に実装されることが多くなっています。

次は、やり取りの内容を規定する記述言語に関してです。これは WSDL という XML ベースの記述言語を用います。これは「Web Services Description Language」の略で、前述した SOAP でやり取りする内容や方法を記述する言語です。要するに、公開したいインターフェース (クラスやら

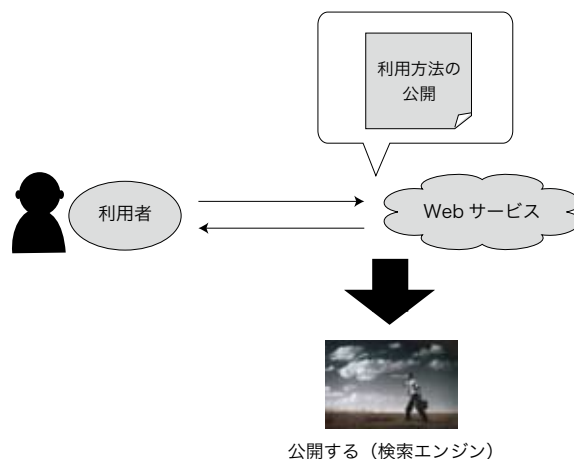


図2: Web サービスを支える3つの技術

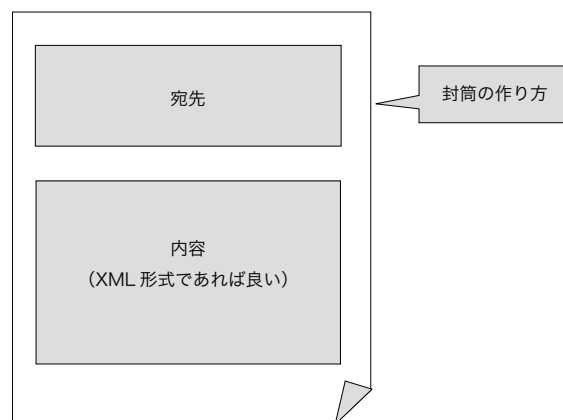


図3: SOAP で決められていること

メソッドやら)の内容を記述できるものです(図4)。Webサービスを提供する側はこれを用意しておくことによって、Webサービスを利用する側はこれを購入することによって、Webサービスの接続が可能になるわけです。

最後は公開検索ディレクトリサービスです。これは、どのようなものかという、提供しているWebサービスを登録しておき、このディレクトリサービスから検索を行って、接続したいWebサービスを容易に探せるように、というものです(図5)。インターネットで言えば、gooとかYahooといったディレクトリ型検索サービスといえるでしょう。UDDIは一見すると便利そうですが、実際のところ、あまり使われていません。なぜかといいますと、基本的にWebサービスは、ビジネス利用がその中心です。特にビジネス用途の場合、サービスの提供元がどのような会社で、信用に値する会社なのか、といったような信用情報なども必要です。そうした点をどのようにクリアするのか、といった規定がなく、今のところあまり使われていない技術となってしまっています。

さて、このような3つの要素がありますが、特にSOAPとWSDLを使ったWebサービスを、Javaでどのように実装するのかを見て行きましょう。

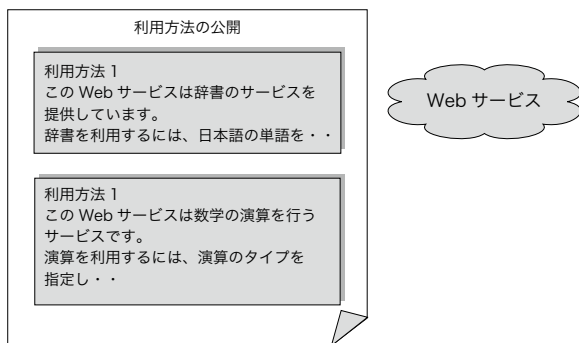


図4：WSDLで記述できること

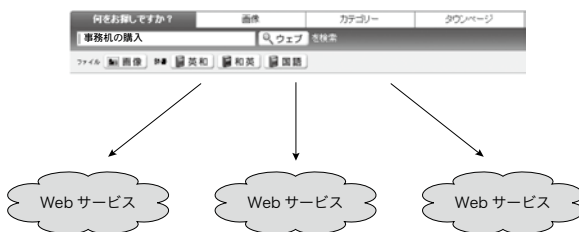


図5：Webサービスを探す

JavaでWebサービス

JavaでWebサービスを使うにはどうすればよいのでしょうか。Webサービスは簡単に言ってしまうと、(主に)HTTPプロトコル上でやり取りされる、XMLデータの交換です。Webサービスの使用に従ったXMLをやり取りできればそれでよいわけですが、、、同じようなXMLをいちいち書いたり、エラーハンドリングなど、わざわざ書いていくと結構面倒ですね(-_-;)。ということで、自分で1から書くのではなく、世の中に存在している便利なライブラリなどを使って、サクサクッと、Webサービスを作ってみましょう。

JavaでWebサービスを扱うためのライブラリはいくつか存在します。代表的なのは以下の2つといえるでしょう。

◇ Apache Webサービスプロジェクト

URL：<http://ws.apache.org/>

開発元：Apache Web Services Project

WebサーバApacheで有名な、Apache Projectのサブプロジェクトの1つにWebサービスのプロジェクトが存在しています。ここにWebサービスを利用するために必要な様々なライブラリが用意されています。

この中でも特に有名なのは、下記の2つでしょう。どちらもWebサービスを行う上での基本的なAPIや実行環境を持ったライブラリとなっています。

- Apache AXIS⁴
- Apache SOAP

◇ JWS DP

URL：<http://java.sun.com/webservices/jwsdp/index.jsp>

開発元：SUN

Javaの開発元であるSUNが提供するWebサービスのライブラリが、JWS DPです。JWS DPとは、「Java Web Services Developer Pack」の略称で、その名の通り、Webサービスを実装するうえで必要となる、SUNが提供する様々なライブラリをまとめたお得な(?)パッケージとなっています。

これさえあれば、Webサービスが手軽にできてしまうのです。ただし若干の注意点が、、、前述しましたように、こ

4. 「あきしす」って読みます。

これは、Web サービスのための専用の API というわけではなく、様々なライブラリの集合体となっています。この中には XML を扱うための汎用的なライブラリ (JAXB、JAXP) 等も含まれています。これが必ずしも最新のものの組み合わせになっているわけではなく、JWSDP として、動作する最新の組み合わせとなっています。そのため、インストールされている別個に導入した API とバッティングすることもあります。JWSDP を使用して、開発・実行する際には、明示的に JWSDP に含まれるライブラリ群を指定するようにしないと、正しく動作しないこともあるので、注意が必要です。

Web サービスを実装するときの異なる2つの実装方法

Web サービスを実装する、とひとくくりにしても、大きく分けて、2つのタイプが存在します⁵ (これは前述した2つのライブラリのいずれにも当てはまる事柄です)。

- RPC タイプ
- メッセージスタイル

RPC スタイルとは、いわゆる RPC (リモートプロシージャコール) の Web サービス版です。このスタイルをサポートするライブラリのメリットは、Web サービスであることをあまり意識せずに開発が可能である、という点です。極端に表現すると、前後のちょっとした手続きやちょっとしたルールを除き、通常の Java の開発をする感覚で、簡単に Web サービスが実装できてしまうことになります。Web サービスを実行する SOAP の細かなメッセージのやり取りは、「Web サービスランタイム」にお任せ、また、SOAP メッセージの作成も、「仲介プログラム」にお任せし、利用者が作成するのは、仲介プログラムに対する一般的な Java 呼び出しのみです。一方、デメリットとしては、簡単がゆえに、公開している (あるいはされている) Web サービスのインターフェースを変更したいという場合に、作り直しが面倒という点です。このあたりの詳細は後述します (図6)。

一方のメッセージスタイルとは何でしょうか？読んで字

5. 正確にいうと、ライブラリに依存したタイプではなく、Web サービスを実現する最もよく使用されるプロトコルである SOAP の仕様から来ているものなんです。SOAP の仕様書を見ると若干表現は異なりますが、RPC とメッセージスタイルの両方に関して触れられています。

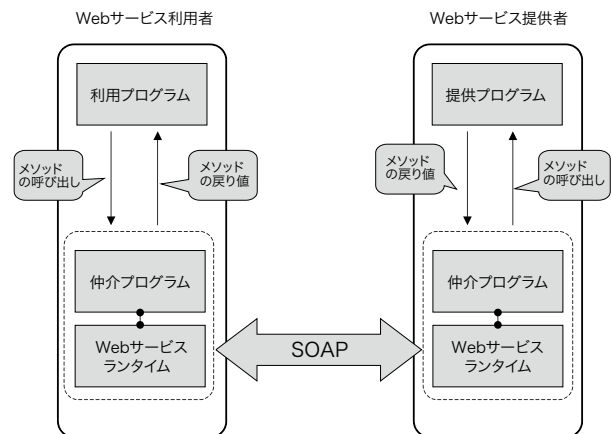


図6：RPC スタイルの Web サービス

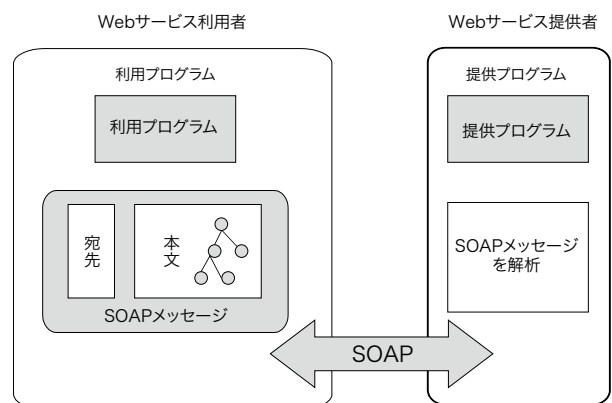


図7：メッセージスタイルの Web サービス

の如しですが (^_^)、メッセージ交換をするための Web サービスを実装するタイプです。データ交換など、XML データそのものを Web サービスで交換する場合に使用します。こちらはデメリットから入りますが、開発が面倒、というデメリットがあります。RPC スタイルとは異なり、そもそも SOAP に関してのある程度の知識が必要となりますし、SOAP 構造を意識しながら、プログラムを行っていくかなければなりません。そのため、通常の Java プログラムとはいろいろと手間ややり方が変わってきます。他方、それによるメリットもあります。最初の作成時は面倒なのですが、その後インターフェースが変わったような場合などに、変更点のみを修正すればよいというメリットがあります (図7)。

	RPCスタイル	メッセージスタイル
メリット	<ul style="list-style-type: none"> Webサービス(SOAP)をあまり意識しなくても良い 通常のJavaの開発に似ている 	<ul style="list-style-type: none"> インターフェースの変更が容易 複雑なメッセージや、自由なメッセージをやり取り可能
デメリット	<ul style="list-style-type: none"> インターフェースの変更がある場合、作り直す必要がある 	<ul style="list-style-type: none"> SOAP (Webサービス)に関する知識が必要 作成時の手続きが面倒

表 1 : RPC スタイルとメッセージスタイルの比較

まとめると表 1 のようになります。

早速実践に入っていきます。今回は、JESDP を取り上げて、Web サービスの使い方を学びましょう。

JWSDP を使う

早速 JWSDP をインストールしてみましょう。ではインストール方法を紹介します、、、ってやっていると紙面もなくなってきましたので、各自で Readme をみてインストールをがんばってみてください (^^)

JWSDP にも、先ほど取り上げたように Web サービスを実行するために 2 つの API が用意されています。RPC スタイルの API は、JAX-RPC、メッセージスタイルの API は SAAJ というものです。今回は特に JAX-RPC を取り上げたいと思います。

JAX-RPC では、Web サービスの内容を記述した WSDL という言語を中心に実装が回ります。WSDL は Web サービスのインターフェースを規定したものであると述べましたが、そこから、Java のクラスを自動的に生成してくれるツールを備えているのです。これには先ほどの説明の中で述べた、Web サービスランタイムと、仲介プログラムの両方を自動的に生成してくれることになります。これは実に楽ですね！ WSDL さえ記述できれば⁶、Web サービスの利用はとっても簡単になります。Web サービスを提供する側も、SOAP などの面倒なプログラムを行わずとも、通常の

6. まあ、WSDL を書くのは、それはそれで、ルールを知らないとかけないんですが、、、でも、一般的な Web サービスであれば、提供する、あるいは提供されるのが普通になっていますので、あまり心配は要らないでしょう。

Java 開発と同じような手順で行えるので、開発が非常に楽になります。この Java のツールは「wscompile」というツールです。

では、最後に、実際に Web サービスのプログラムを作ってみましょう。紙面の都合もありまして、今回は、Web サービス利用プログラムを作成してみます。

ん?? 利用ってことは、なにか提供されているサービスを使うってことですね。。。じゃあ、準備として Web サービスを提供するプログラムを作って、、、って、作っちゃうし！（三村風）というのは冗談で、世の中で公開されている Web サービスを利用するプログラムを作りましょう。

その Web サービスとは、、、Google の Web サービスです。皆さんは、Google の検索が Web サービスでも使用できるをご存知でしたか？ Google の Web サービスは以下の URL で公開されています。

<http://www.google.com/apis/index.html>

このサービスを利用するためには、無料のアカウントを取得する必要があります。左のメニューに用意されている「Create Account」から、アカウントを各自取得してください。次に、Web サービスのインターフェースが記述された、WSDL 文書を入手します。以下の URL に存在します⁷。XML ファイルですので、そのまま、ローカルに保存しましょう。

<http://api.google.com/GoogleSearch.wsdl>

さて、これで準備は整いました。まず、WSDL ファイルか Web サービスの仲介プログラムを作成します。これには wscompile のための設定用 XML ファイルを記述する必要があります。以下にそのファイルの内容を示します。

リスト 1 : Config.xml の例

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <wsdl location="GoogleSearch.wsdl"
    packageName="sample.googlesearch" />
</configuration>
```

7. wscompile では、URL から取得できるので、この URL を直接指定しても OK です。

ここまで準備ができれば、

```
wscompile -gen:client -keep config.xml
```

と、コマンドを打ちます。これで、沢山(^_^;)Java のファイルができれば出来上がりです。なんで、こんなに Java のファイルができるのでしょうか。。。そんなに複雑な API なのでしょう。いえいえ、そんなことはありません。これ、実は、仲介プログラムの下の Web サービスランタイムの部分も一緒に作成されているからなんです。Google を利用する場合は、このうちのわずかな Java ファイルを利用するだけです。

では、最後にプログラムを作ってみましょう。必要な手順は以下の通りです。

- 1) 仲介プログラムのインスタンスの作成
- 2) Web サービス URL の設定
- 3) Web サービスメソッドの実行

え、たったこれだけ？って思われる人もいるでしょう。そうなんです。たったこれだけです。ね、Web サービスって簡単でしょ。では、非常に短い実際のコードをご覧ください。

リスト 2 : GoogleSearchSample.java

```
import java.io.*;
import javax.xml.rpc.*;
import sample.googlesearch.*;

public class GoogleSearchSample {
    public static void main(String[] args) {
        try {

            // 1) 仲介プログラムのインスタンスの作成
            GoogleSearchPort_Stub stub = (GoogleSearchPort_Stub)
            (new GoogleSearchService_Impl().getGoogleSearchPort());

            // 2) Web サービス URL の設定
            stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY,
            "http://api.google.com/search/beta2");

            // 3) Web サービスメソッドの実行 (検索を実行)
            GoogleSearchResult SearchResult = stub.
            doGoogleSearch("License-Key","Kensaku-shitai-
            kotoba",1,10,false,"",false,"", "", "");
```

```
// 検索結果件数の表示
System.out.println(SearchResult.getEstimatedTotalResult
sCount());

// 検索結果 Element の取得
ResultElement[] SearchResultE = SearchResult.
getResultElements();

// 検索結果 Element の内容を 1 件ずつ表示 (検索結果を
1 件ずつ表示)
for (int i = 0; i < 10; i++){
    System.out.println(i + " 件目 ¥n");
    System.out.println(SearchResultE[i].getTitle());
    System.out.println(SearchResultE[i].getSummary());
    System.out.println(SearchResultE[i].getURL());
    System.out.println(" ~~~~~ ");
}

} catch (Exception e) {
    e.printStackTrace();
    System.exit(1);
}
}
```

これを実際にコンパイルして、実行してみると！！！！興味のある方は、実際の Google の検索結果と見比べてみてください。

と、まあ、こんな感じで RPC スタイルを使うと非常に簡単に Web サービスの利用ができます。

まとめ

最後の方は駆け足になりましたが(^_^;)ざっと Web サービスに関してご紹介しました。Web サービスの提供を考えられた方にはちょっと物足りなかったでしょうか。。

さて、これで、この連載は一回終了となります。ここまで辛抱強く (?) 読んでくださってありがとうございます。少しでも XMLF をプログラムするためにはどうすればよいのだろうかってことの理解の助けになれば、幸いです。

(吉田 晃伸)