XML Consortium

～ 第6回 XMLコンソーシアムWeek ～

Web2.0部会活動報告

RESTとSOAP

エンタープライズ2.0における、RESTとSOAPの使いこなしについて

2007年5月14日

XMLコンソーシアム Web2.0部会

荒本道隆 (アドソル日進株式会社)

はじめに

- かつてWebサービスと言えばSOAPでしたが、現状ではInternet上で公開されているWebサービスはRESTが多くなってきています。
- そのような現実を踏まえ、REST とSOAPについて「どう違うのか？」「どっちを使ったらいいんだ？」という疑問について、また、企業内のAjaxでのRESTやSOAPの利用について、部内で検討した内容を報告します。

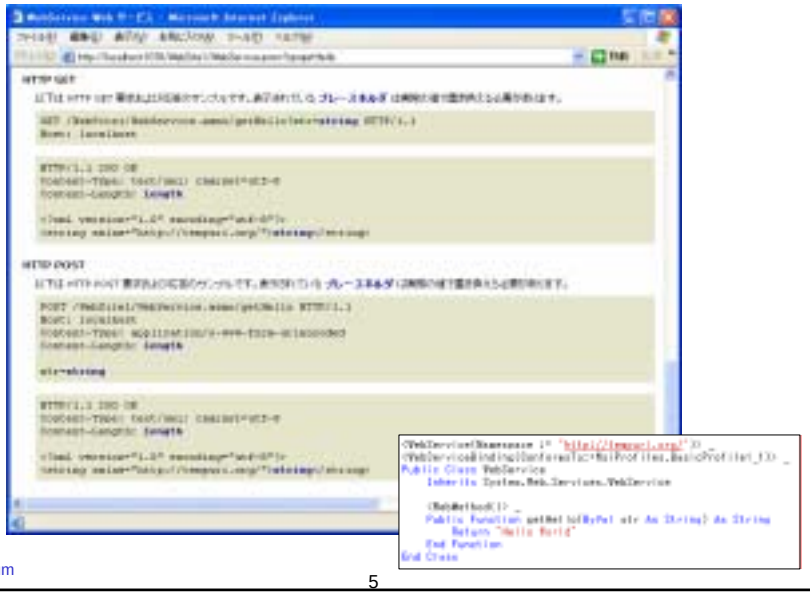
REST と SOAP

概要と特徴

RESTとは

- Representational State Transfer
  - REST準拠のWebサービス
- URLにアクセスすると、XMLを返す
  - プロトコルはHTTPのみ
  - パラメータはURLの後ろに追加する
  - URLの長さ制限
- 利用し易い
  - ブラウザを使って、動作テストができる
  - →使ってみようという気になる
- 開発し易い
  - Webサイト開発のノウハウがそのまま使える
  - 開発時の注意点も、Webサーバと同じ
    - キャッシュ、2重送信の防止、など

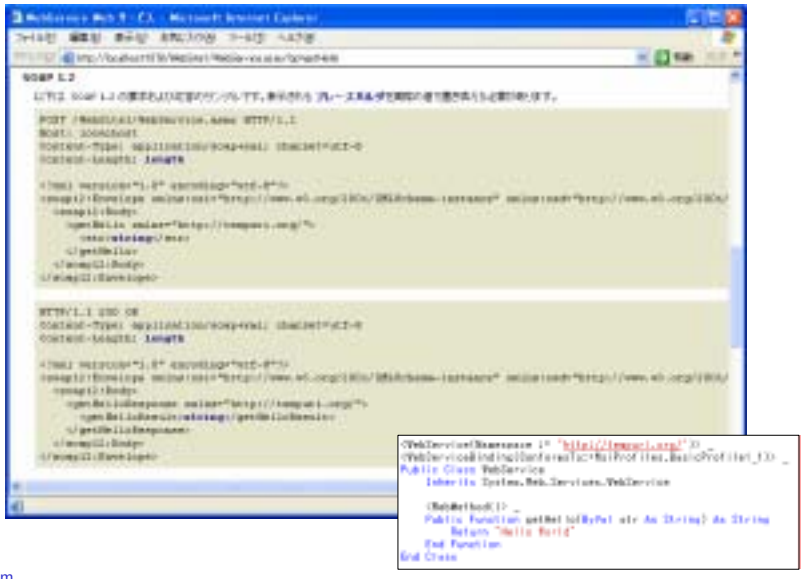
RESTのサンプル



SOAPとは

- **S**imple **O**bject **A**ccess **P**rotocol
  - SOAPを使ったWebサービス
- リクエストのXMLを送ると、レスポンスのXMLが返ってくる
- **WS-*による拡張性**
  - W3CやOASISなどの標準化団体により標準化
- リクエスト時に、複雑な情報を送れる
  - リクエストもXMLなので、構造化された情報を送れる
  - 複数のXMLを1つのXMLに入れる
- スキーマによる厳密なバリデーション
  - 入出力パラメータのフォーマットチェック
  - XML名前空間による厳密性
- SOAP対応のミドルウエアが多数ある
  - Java(axis, JAX-WS)、.NET Framework、PHP(SOAP拡張モジュール)、Perl（Perl/SOAP）、データベース（Oracle, DB2, SQLServer）、MS-Office（SOAP Toolkit）

SOAPのサンプル



```
<WebService(Namespace := "http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
Public Class WebService
    Inherits System.Web.Services.WebService

    <WebMethod()> _
    Public Function getHelloByPol str As String) As String
        Return "Hello World"
    End Function
End Class
```


比較

以前の方式(CORBA,RPCなど)との比較

- 高い相互接続性
  - RESTは、URLだけなので相互接続性は非常に高い
    - エンコードの問題で、日本語や機種依存文字には注意
  - SOAPは、WS-I による WS Basic Profile
  - 通信内容がすべてテキストなので、可読性がある
- HTTPを使うメリット
  - HTTPSによる暗号化(サーバ証明／クライアント証明)
  - HTTPプロキシを使ってFirewallを越える
  - リアルタイム応答
  - ライブラリやアプライアンスの流用
- XMLを使うメリット
  - XSLTを使ってのフォーマット変換
  - スキーマによる仕様定義
  - ライブラリやアプライアンスの流用

以前の方式(CORBA,RPCなど)との比較

- デメリット
  - パフォーマンスが悪い
    - XMLなので通信量が多くなりがち
    - XMLをパースするために、多くのメモリが必要
    - その都度HTTP接続するので、時間がかかる
    - 粒度に注意して、呼び出し回数が必要最低限＆汎用性を失わないように
  - 処理時間のかかるサービスは実装できない
    - サービス側で時間がかかると、HTTPのタイムアウトが発生する
    - 時間のかかる処理は、キューにためて、すぐに応答を返すように
  - コールバック処理が難しい
    - 途中にFirewallがあると、クライアント→サーバは許可されていても、サーバ→クライアントは許可されていない
    - クライアント側でHTTPサーバは起動していない
    - ポーリングやCometなどを使って、コールバックに相当する機能を実現
  - RDBの直接接続と比べて
    - 多くの機能を公開しようとすると、多くのサービスを開発する必要がある
    - 「利用者ごとにDBユーザーを変える」場合には注意

RESTとSOAPの比較

- REST
  - ブラウザで簡単に試せるし、初期の学習コストが低い
  - 入力パラメータが少なめ
    - 入力パラメータに日本語を含める場合は、エンコードや文字コードに注意
  - WADLにより、インターフェイスが明文化されるようになるのか？
- SOAP
  - 高機能ゆえに、初期の学習コストが高い
  - WS-* の機能が必要であればSOAP
    - ミドルウエアによって、様々な仕様に対応していける
    - そのWS-*に対応したミドルウエアが無いかも…
  - WSDL1.1は普及しているが、いつ2.0になるのか？
    - 1.1と2.0は、書き方がまったく違う

REST/SOAP以外のアプローチ

- HTML
  - HTMLの一部を返し、<frame> や を使って部分更新する
    - ずっと以前から使われていた手法
- JSON (JavaScript Object Notation)
  - XMLではなく、JavaScriptオブジェクトの表記法で返す
    - XMLよりも通信量が少なくて済む
    - ブラウザのJavaScriptでのDOM操作が不要
  - JavaScriptから利用する場合には、一番便利
- JSONP (JSON with Padding)
  - ドメイン境界制限を回避するための裏ワザ？
    - ブラウザのセキュリティ対策で使えなくなるかも

RESTに適しているサービス

- 入力パラメータが少ない
- 簡単に使える

↓

不特定多数を対象とした検索サービス等

- Google, Amazon, Yahoo などの検索サービス

- 使い方次第で、ブラウザのドメイン境界問題を回避できる

↓

HTMLの<xxx src="…"> を書き換える

- GoogleMapsなど

SOAPに適しているサービス

- 入力に対しても、スキーマを持っている
  - 入力が複雑な構造を持っている
  - 入力を厳密にチェックする必要がある

↓

基幹システム, B2Bなど

- WS-* を必要とするもの

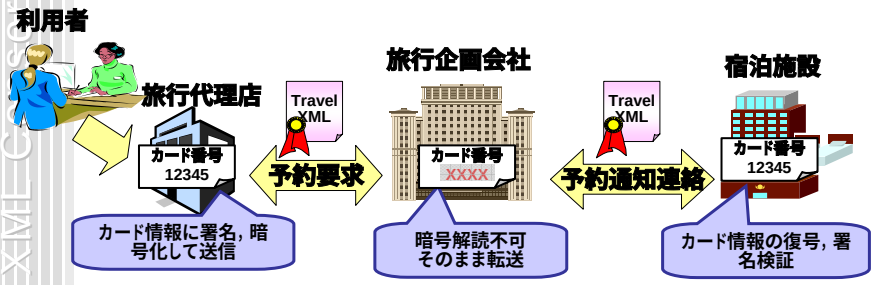
↓

WS-Securityを使った例→sPlat

参考資料：sPlatの概要

Webサービス実証部会とセキュリティ部会の共同プロジェクト

- 中継者を介したマルチホップのWebサービス(SOAP)
- WS-Securityによる End-to-Endのセキュリティを確保
  - httpsによるトランスポート層の暗号化では、中継者のところで一旦すべて復号されてしまう
  - WS-Securityによるドキュメント層の暗号化を使えば、中継者に見せたくない情報を見えないように出来る
- sPlatでは、中継者の具体的な課題の解決策を検討中



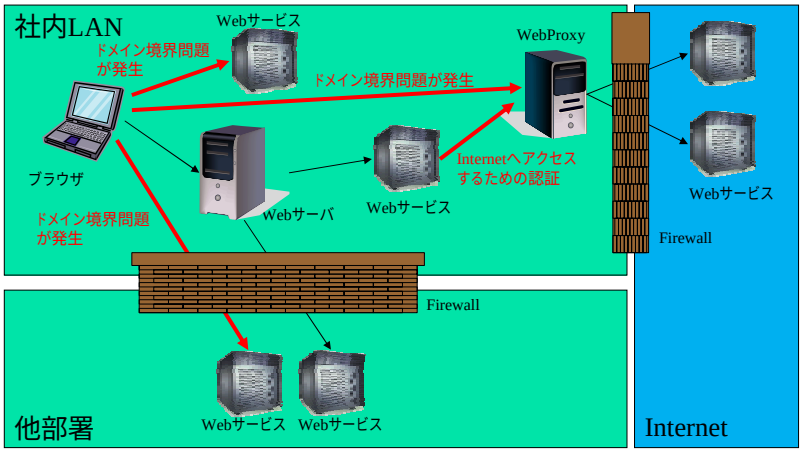
企業内での利用

企業内だからこそできること

- RESTやSOAPの企業内での利用
- スキーマの統一
  - 企業内でスキーマを統一しておくと、開発効率が非常に高くなる
    - RDBのテーブルの統一と同じで、なかなか実現は大変だけど…
- ドメイン境界問題の回避
  - Ajaxでマッシュアップする時の最大の課題
  - Webサイトから利用しているサービスへ中継
    - ブラウザから見ると、すべて同じサイト上にあるように見せる
  - 「信頼済みサイト」に登録
    - 別の問題が発生する危険もある
- シングル・サイン・オン
  - 全サービスは認証サーバを参照
  - BASIC認証であれば、Proxyをうまく使うだけで実現可能

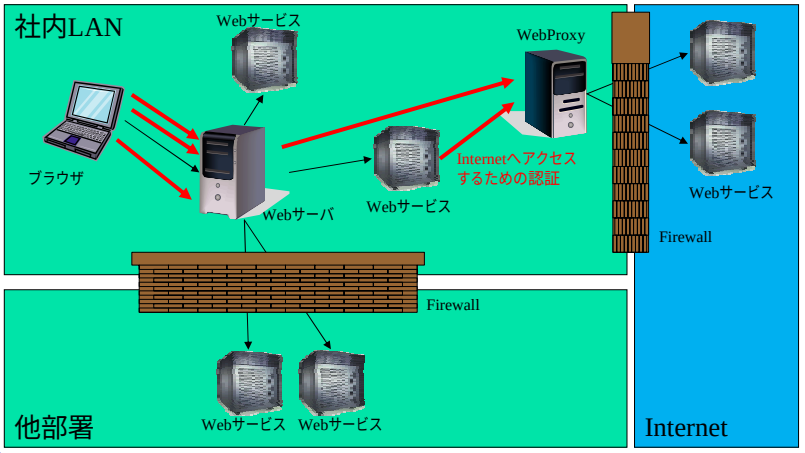
企業内でのHTTPの流れ－1

- 社内Webサイトから、様々なWebサービスを利用
  - ブラウザから直接呼び出そうとすると、ドメイン境界問題が発生



企業内でのHTTPの流れー2

- Webサーバがリクエストを中継
  - BASIC認証なら、HTTPヘッダ情報を削除しなければ転送可能
  - アプリで認証情報を転記してもOK



企業内でのREST/SOAPの利用

- 企業内の方が、利用し易いはず
  - スキーマの統一
  - サーバに対する自由度
  - サービスの利用状況の把握
    - あるサービスが停止する時には、伝達の徹底
  - ブラウザの統一
- 永遠のベータ
  - ベータのサービスを1つでも使うと、システム全体がベータに
    - 企業内システムでは敬遠されがち
    - ベータのサービスを1つも使わなければ問題無い
  - ベータ→無料→安価に構築できるというメリットも

まとめ

SOAPとREST、どっちを使う？

	SOAP	REST	JSON
学習容易性	×	○	○
コーディング量（自動生成分は除く）	○	×	○
テストのし易さ	×	◎	×
開発ツール	○	×	×
導入の容易性	×	○	○
開発にかかるトータル時間	×	○	◎
複雑なデータの入力	○	×	×
複雑なデータの出力	○	○	△
厳密な型チェック	○	×	×
汎用性（扱える開発言語の種類）	○	○	×
標準化団体	W3C	ー	ecma
拡張性（WS-*）	◎	×	×