3E-1

Web サービスを利用した Java/.NET 連携フレームワーク

土屋隆† 原田雅史† 浅見可津志‡
三菱電機株式会社† 三菱電機インフォメーションシステムズ株式会社‡

1．はじめに

ブラウザで、Java サーバが生成した HTML を表示する Web アプリケーションが一般的であるが、一方、高度な GUI の必要性から、.NET クライアントと Java サーバを連携させる需要も存在する。この場合、双方の入出力パラメータの構造の違いを解決する手段が必要で、かつサービスの数だけ定義ファイルの設定が必要である。これらの作業負荷の軽減を目的に、Web サービスを利用した Java/.NET 連携フレームワークを開発した。

2．Web サービス利用における課題

.NET クライアントと Java サーバの連携に、Web サービスを利用することができる。標準技術の活用により相互運用性は高いものとなるが、一方で業務アプリケーション構築に Web サービスを利用するには、次のような課題があった。

- サービスの入出力パラメータの項目やデータ型を定義するための WSDL(Web Services Description Language)を、サービスの数だけ定義する必要がある。また、それに対応するクライアント側スタブを作成する必要がある（スタブレスで動的に Web サービスを実行する試み[1]もあるが、まだ標準ではない）。
- サービスの実装方法が、SOAP エンジン（Apache Axis など）によって異なり、ほかの環境に移行しにくい。

Java/.NET 連携フレームワークは、このような課題を解決するものである。

3．Java/.NET 連携フレームワーク

Java/.NET 連携フレームワークは、次のような特徴をもつ。

- Axis ベースの汎用メッセージサービスと、それを呼び出すサービスクライアントからなる。クライアントは.NET 用と Java 用があり、それぞれから同じメッセージサービスの呼び出しができる。
- １つの Web サービスで、異なる入出力パラメータをもつ複数の業務コンポーネントを呼び分ける。
- 入出力パラメータとして以下をサポートする。WSDL によるインターフェース[2]と遜色はなく、さらに WSDL にはないリストやマップの受け渡しが可能である。
  ① 数値、文字列などのプリミティブ型
  ② ①をプロパティにもつクラス。入れ子クラスをプロパティにもつクラスも可。
  ③ ①②を要素とする配列
  ④ ①②を要素とするリスト
  ⑤ ①②をキーおよび値とするマップ
  オブジェクトは XML にエンコードしたのち、SOAP エンベロープで包んで送受信する。
- Java クラスと.NET クラスのマッピングは、クラス名の対応表の作成のみで可能。
- 添付データをサポートし、１回の呼び出しで複数ファイルのアップロードとダウンロードが可能。
- SOAP エンジンの相違をフレームワークで隠蔽し、アプリケーション開発者が SOAP や WSDL などの Web サービス固有の仕様を意識せずに使える。

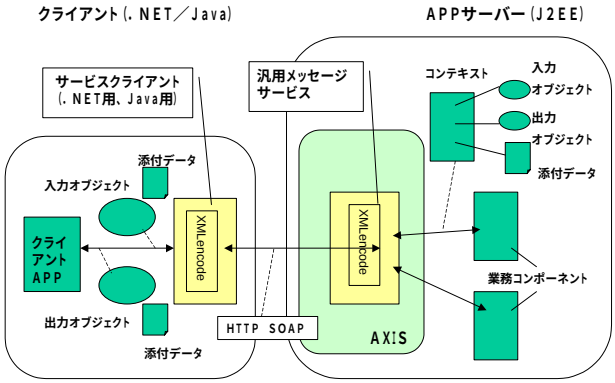

図 1.Java/.NET 連携フレームワークの構成

一方、通常の Web サービスと比較したときのデメリットとしては以下がある。

- データサイズ及び通信速度がそれぞれ約 1.2～1.5 倍になる。
- WSDL にサービス仕様が現れないため、一般に広く公開するサービスには適さない。

Java/.NET cooperation framework using Web Service
†Takashi Tsuchiya, Masafumi Harada
Mitsubishi Electric Corporation
‡Katsushi Asami
Mitsubishi Electric Information Systems Corporation

４．アプリケーション・アーキテクチャ

当社では、Web アプリケーション構築の生産性向上のため、標準アーキテクチャを導入し、システム設計・製作方法の共通化を図っている。Java/.NET 連携フレームワークも、標準アーキテクチャに基づいている。

標準アーキテクチャでは、Web アプリケーションを、プレゼンテーション層、ビジネスロジック層、データアクセス層の３層に分離し、DI コンテナでこれらを疎結合している。

Java/.NET 連携フレームワークを利用する場合は、プレゼンテーション層に当たるところをクライアント層とサービスインターフェース層に置き換え、ビジネスロジック層とデータアクセス層は、全く同じ作り方とする。メッセージサービスは、送られた SOAP メッセージに含まれる業務名パラメータによって複数の業務コマンドを呼び分ける。

このような構成をとることにより、既存の Web アプリケーションを容易に Web サービス化することができる。

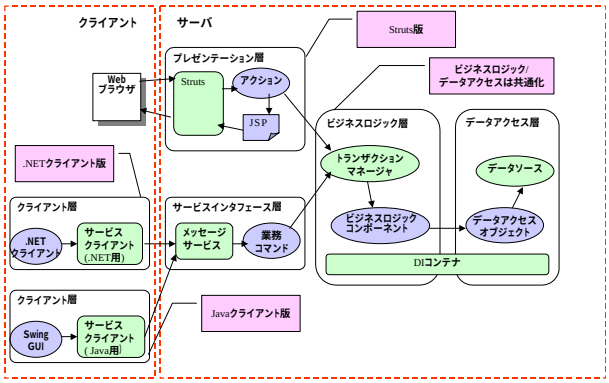

図 2.アプリケーション・アーキテクチャ

５．実システムへの適用と評価

Java/.NET 連携フレームワークは、ある実システムの開発に適用されている。

このシステムでは、クローズドな環境では既存のソフトウェア資産を生かして VB.NET クライアントを使用し、インターネット向けに新たに Struts を使用した HTML 画面を開発するが、ビジネスロジックは共通化したいという要件があった。そこで上記のアプリケーション・アーキテクチャと Java/.NET 連携フレームワークが採用された。

業務コマンド数は約 30 個ある。通常ならこれら１つ１つが Web サービスの API にあたるものだが、フレームワーク適用により、業務コマンドを増やすたびに Web サービスの設定・登録、クライアント側スタブの作成を行う必要はない。また、ビジネスロジックの機能追加によって、入出力パラメータに項目が追加されることがあっても、Web サービスの設定を変更することはない。これによりクライアント側/サーバ側双方の製作・試験の作業時間を削減できた。

このように、Web サービスの作成・設定・運用に関する知識をもつ開発者が多くない中で、開発効率を上げることができた。

一方で、冗長な XML を送受信するため通信速度が遅いという問題も顕在化した。これに対しては、次のような対策を行った。

- HTTP の gzip エンコーディングを利用してデータを圧縮する。
- クライアント側のネットワーク帯域が低い場所については、中継サーバを設け、クライアントと中継サーバ間を専用のバイナリ通信、中継サーバと業務サーバ間を本フレームワークで結ぶ。

システムは.NET 1.1 ベースで開発したが、.NET 2.0 への移行も視野に入れており、それに向けて.NET2.0 対応のサービスクライアントと、SOAP エンジンを Axis から Axis2 に変更したメッセージサービスも作成した。

一部を切り出した移行プロトタイプでは、フレームワークの参照変更と再コンパイルのみでアプリケーションの移行ができている。

６．まとめ

本フレームワークは Web サービスの一般的な使用方法とは異なるが、通信 API の数が多く、項目追加などの API の変更もよくある業務システムでは、生産性向上の効果が期待できる。

今後の改善点として以下を検討している。

- 一つのデータ設計情報から、対応する Java クラスと.NET クラスを生成するような開発ツールの提供。
- オブジェクト/XML エンコード（デコード）の高速化
- 通信データサイズのさらなる縮小

参考文献

[1] 越田高志、「入出力データ型に透過な Web サービス動的実行システム」、情報処理学会第 68 回全国大会

[2] W3C 、「 Web Services Description Language(WSDL) Version 2.0 」