SOAP와 REST 기반 웹 서비스

황의철
광주여자대학교 교육미디어학과

SOAP and REST-Based Web Services

Eui-Chul Hwang
Dept. of Educational Media, Kwangju Women's University

요 약

REST(REpresentational State Transfer)는 분산 컴퓨팅 플랫폼 모델이며, 세계에서 가장 큰 분산 응용인 Web에서 사용하고 있는 웹 구조 스타일 모델이다. 현재 웹의 기본 요소는 URI, HTTP, XML(HTML) 이며, REST는 이러한 인터넷 표준만을 사용한다. REST 에서 리소스의 식별은 URI로, 상태는 상태가 표현된 문서(리소스)로써 HTTP를 통해 전달된다. 리소스의 내용은 XML로 기술하며, 리소스 탐색 및 참조에는 HTTP의 표준 메서드인 GET, PUT, POST, DELETE 등만을 이용하는 것으로 분산 컴퓨팅을 모델링하고 있다. 따라서 서비스마다 다양한 메서드를 기억하여야 하는 SOAP 기반 웹 서비스에 비해 REST 모델의 분산 컴퓨팅 응용은 확장성 및 웹 친화성 측면에 있어서 매우 유리함을 알 수 있다.

1. 서론

분산 컴퓨팅에서 원격 객체의 메서드 호출을 지원하는 미들웨어가 그동안 개발되어 왔다. DCOM, CORBA, RMI가 이들이다. 그런데, 이들 각각은 독자적인 바이너리 메시징 포맷을 사용하고 있어 상호 운용이 곤란하며, 파이어월 통과가 어렵다. 따라서 웹 환경에서 상호 운용성 및 파이어월 통과 등을 해결하고, 보다 보편적인 미들웨어 플랫폼을 제공하고자, 메세징 포맷을 XML로 구현하고자 하는 시도에서 나온 것이 SOAP 이다.

SOAP은 XML 기반으로 데이터, 데이터 타입 정보 및 데이터 구조 등을 정의하며[1], 웹 서비스는 메시징 프로토콜로 SOAP를 사용하며, HTTP와 XML의 결합으로서 분산 환경 하에서 정보의 상호 교환을 가능하게 하는 간단한 프로토콜이다[2].

그런데, 프로시저 지향형인 SOAP 기반 웹 서비스는 확장성에서 여러 문제점을 가지고 있으며, 현재 웹 구조 모델에도 잘 부합되지 않는 측면이 많다.

SOAP 기반 웹 서비스에서, 서버의 서비스는 보통 객체의 메서드 형태로 구현되고 있으며 이러한 메서드들의 모임이 하나의 웹 서비스로 제공되고 있는 데, 웹 서비스 종단점에 대해서만 하나의 URI(Uniform Resource Identifier)가 할당된다. 따라서 SOAP의 종단점과 매핑되는 응용의 내부 리소스는 직접 어드레싱이 불가능하고, 이들 리소스 조작을 위해 응용 각자의 독자적인 메서드들을 제공한다. 그러나 이러한 메서드들은 HTTP의 CRUD 같은 메서드(PUT, GET, POST, DELETE 등)들처럼 표준적이 아니기 때문에 클라이언트는 반드시 이러한 메서드들을 이해하고 있어야 한다. 추후 기능 확장 등을 위해서 메서드들 내용이 변경되어야 하는 경우, 클라이언트는 이 변경된 메서드를 이용하여야 하기 때문에 기존 클라이언트 응용 프로그램의 변경이 불가피하고, 이러한 점에서 확장성이 부족하다.

본 논문에서는 SOAP 메시지 프로토콜, SOAP의 장단점, SOAP 기반 웹서비스의 문제점을 살펴보고, SOAP의 단점 보완 및 확장성에 유리한 REST 모델과 비교 분석하여 추후 REST 기반의 웹 서비스 구축에 하나의 중요한 사례로써 제시한다.

2. SOAP과 웹 서비스

2.1 SOAP 메시지 프로토콜

SOAP이란, 분산 환경에서 소프트웨어 서비스들 간에 정보를 교환하기 위한 XML 기반의 간단한 프로토콜로 메시지 내용과 이를 처리하는 방법을 설명하

기 위해 프레임워크를 정의한 Envelope, 응용프로그램에서 정의한 데이터 타입의 인스턴스를 나타내는 일련의 인코딩 규칙, 원격 프로시저 호출 및 응답 등을 나타내는 규칙으로 구성되어 있다.

2.2.1 SOAP의 장점

SOAP은 그 동안 진행된 XML에 대한 막대한 투자를 활용하는 방법을 제공한다는 것이다. 또한 SOAP는 대개 HTTP나 SMTP와 같이 "방화벽에 친숙한" 프로토콜에서 정의되기 때문에 방화벽 기술에 대한 대규모 투자도 적극 활용할 수 있다. SOAP는 최종적으로 Web Services의 프로덕션 구축에 존재했던 주요 장벽들을 제거한다. SOAP를 Web Services의 핵심 부분으로 정의하면 다른 전략을 도입했을 때보다 훨씬 빠르게 Web Services를 구축할 수 있을 것이다.

(1) 특정언어에 종속되어 있지 않다.
(2) 특정 전송 프로토콜에 종속되어 있지 않다.
(3) 그 어떤 분산 객체 환경에도 종속되지 않는다.
(4) 기존 산업 표준을 이용한다.
(5) 다중 환경에서 상호 운용성을 가능하게 한다.
(6) XML을 이용하므로 얻을 수 있는 확장성이다.

2.2.2 SOAP의 단점

SOAP을 이용할 때 배포가 쉽도록 하는데, SOAP 자체에 대한 보안 이슈는 가장 시급하게 보완되어야 할 부분이다. SOAP이 기존의 CORBA, IIOP, DCOM, RMI 와 같은 프로토콜을 대체할 수는 없다. 기존의 프로토콜이 가지고 있는 기능을 빼서 최대한 가볍게 만든 것이 목적이므로 CORBA나 COM등이 가지고 있는 객체생명주기 관리나 동적 호출 등 여러 가지 기능을 지원할 수 있는 방법을 따로 만들어야 한다. SOAP은 객체 모델도 가지고 있지 않다. 이외에도 다음의 단점이 있다.

(1) XML은 텍스트 기반이므로 복잡한 데이터 타입을 네트워크를 통하여 보낼 때 이러한 데이터 타입을 XML로 인코딩 하는데 데이터양도 많아지고, 인코딩/디코딩 하는데 꽤 많은 시간이 소모된다. 이런 경우 RMI나 IIOP가 SOAP 보다 더 적합하고 SOAP보다 더 오래 되었고 버그도 적으며, 신뢰도가 높다.
(2) SOAP Envelope를 SOAP 패킷에서 추출하는데 시간이 많이 걸린다.
(3) SOAP Envelope에 들어있는 XML 정보를 파싱하는 작업도 시간이 많이 걸린다. 현존하는 XML 파서는 코드 크기나 처리속도, 차지하는 메모리양의 면에서 많은 문제를 갖고 있다.
(4) XML 데이터의 경우에 최적화할 여지가 별로 없다.
(5) SOAP 인코딩 규칙은 모든 SOAP 메시지에 대한 데이터 타입 정보를 포함하도록 하고 있다.
(6) 메시지를 보내는 시스템이 다운될 경우 SOAP 시스템은 메시지를 어떻게 다시 보내야 할지 모르며, SOAP 클라이언트가 여러 서버에 요청을 하려면, 반드시 요청하려는 모든 서버에 직접 요청을 보내야 하는 단점 등이 있다[3].

2.2.3 SOAP의 단점 보완

현재 대부분의 SOAP 구현이 DOM(Document Object Model)에 기반을 두고 있는데 DOM 파서는 기본적으로 SAX(Simple API for XML)기반의 파서에 비해 처리속도가 느리다. 그러므로 SAX 기반으로 구현하는 것이 처리속도를 증가시키고, 메모리 요구사항도 줄일 수 있으며, 확장성 면에서도 훨씬 유리하다.

SOAP은 XML을 이용하기 때문에 만약 SOAP 메시지를 압축할 수 있다면 네트워크 대역폭을 차지하는 문제점을 상당히 개선할 수 있을 것이다. 일반적으로 XML 데이터가 바이너리 데이터에 비해 평균적으로 400% 정도 크다는 보고가 있다. 물론 압축을 이용하면 CPU에 오버헤드가 더 많이 가해지는 것은 피할 수 없을 것이다[4].

2.2 웹 서비스

웹 서비스는 웹에서 이미 널리 수용되어 사용되고 있는 표준인 URL, HTTP, XML 등에 기반하고 있으며, 대형 소프트웨어 개발 회사인 IBM, 마이크로소프트, SUN, HP 등이 적극적으로 지원하고 있기 때문에 현재 인터넷상에서 분산 응용 구축에 매우 유망한 웹 미들웨어로 각광을 받고 있다[5]. 웹 서비스 구조에서는 클라이언트는 메시지 형식으로 만든 XML 문서를 사용하여 서버에 요청을 보내고, XML 메시지 형식으로 응답을 받는다. 웹 서비스 표준은 메시지 형식을 규정하고, 전달되는 메시지에 대한 인터페이스를 기술하고 웹 서비스를 게시하고 발견하는 메카니즘을 정의한다. 웹 서비스를 구성하는 주요요소는 WSDL (Web Services Description Language), SOAP (Simple Object Access Protocol), UDDI(Universal Description, discovery, and Integration)의 3가지이다. WSDL은 웹 서비스 인터페이스를 기술한다. SOAP는 웹 서비스 사용을 위한 메시지 전송 프로토콜이다. UDDI는 웹 서비스를 게시하고, 찾는 절차를 규정한다. 현재, WSDL, SOAP 등은 W3C(World Wide Web Consosium)에서 표준 사양화 작업을 진행하고 있으며, UDDI는 OASIS UDDI member section에서 표준화 작업을 진행하고 있다. 웹 서비스는 데이터 표

현에 XML을 사용하고 있는 데, XML은 데이터 구조를 스스로 기술할 수 있기 때문에(즉, 데이터의 타입까지 표현), 데이터만 전송되는 CORBA, DCOM, RMI 등의 경우에 해당 데이터 타입을 이해하지 못하는 수신자는 수신된 데이터를 이해할 수 없어 동작할 수 없지만, 웹 서비스의 메시지는 데이터뿐만 아니라, 데이터 표현까지 포함되기 때문에 XML 파싱이 가능한 어느 누구도 메시지 내용을 이해할 수 있어서 인터넷상의 여러 응용 사이의 데이터 교환에 매우 유리하다. 또한, 특정 포트가 규정되어 있지 않아 파이어월 통과가 쉽지 않은 CORBA 객체 서비스에 비해 웹 서비스는 보통 SOAP 메시지의 HTTP 전송을 지원하는 데 HTTP는 보통 파이어 월을 통과하므로 서비스 사용이 용이하다.

2.3 SOAP 기반 웹 서비스의 한계

SOAP과 WSDL은 모든 객체를 관리하는 한 개의 종단점만을 참조하는 하나의 URI 만을 기술한다. 따라서 SOAP의 종단점과 매핑되는 응용의 내부 리소스는 직접 어드레싱이 불가능하고, 이들 리소스 조작을 위해, 응용에 따른 독자적인 메서드들을 제공한다. 그러나 이러한 메서드들은 HTTP의 CRUD 같은 메서드(PUT, GET, POST, DELETE 등)들처럼 표준적이 아니기 때문에 클라이언트는 반드시 이러한 메서드들을 이해하고 있어야 한다.

그림 1은 SpsWebService 웹서비스로 GetSumOfCr(대변 합계 처리), SelectPurSalesSlip(매입매출 현황 처리), GetSumOfTax(세금 합계 처리), GetSumOfDr(차변 합계 처리)의 4개의 메서드를 사용자의 요구에 따라 처리하고 그 값을 XML로 응답해주는 웹 서비스이다. 그런데, 추후 기능 확장 등을 위해서 메서드들 내용이 변경되어야 하는 경우, 클라이언트는 이 변경된 메서드를 이용하여야 하기 때문에 기존 클라이언트 응용 프로그램의 변경이 불가피하고, 이러한 점에서 확장성이 부족하다.

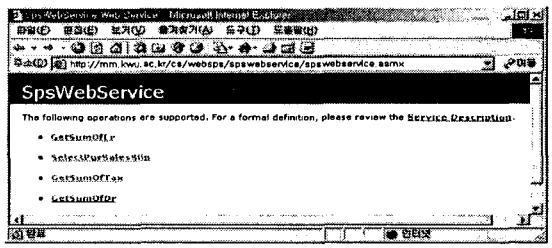

그림 1 SlipWebService 실행 페이지

따라서 SOAP 클라이언트는 웹 서비스의 인터페이스를 알아야 하며, 변경될 가능성이 있으므로 반드시 현재의 인터페이스를 알아야 한다. 물론 응용마다 제공되는 이러한 메서드 인터페이스에 대해서는 WSDL에 기술되며, 이는 개별적으로 또는 UDDI를 통해 인터페이스에 대한 정보 획득이 가능하다. 그러나 UDDI는 확장성이 부족하다. 이러한 점은 CORBA 및 DCOM 등이 광범위하게 사용되지 못한 이유와도 같다.

XML 에서는 기본적으로 모든 유용한 정보는 표준화된 주소에 XML 형태로 획득 가능한 것으로 기본적으로 가정하고 있다. URI가 없는 리소스는 RDF를 이용하는 것도 불가능하며, 리소스들 사이의 관계 정립을 위해 XLink를 사용하는 것도 불가능해진다. 또한, Xpoint, XInclude, XSLT 등을 이용하는 것도 불가능해진다. SOAP이 모든 리소스에 URI를 사용하면, SOAP의 Web에의 적용이 잘 들어맞을 수 있다. 그러나, SOAP 주창자들은 Web을 단지 SOAP의 전송 메카니즘의 하나로만 간주하며, 웹을 정보 시스템이 아니라, 미들웨어로 간주하기 때문에 모든 리소스에 URI를 적용하는 데에 다른 입장을 취하고 있다.

3. REST(REpresentational State Transfer)

REST는 "REpresentational State Transfer" 의 약자로써, 분산 컴퓨팅 플랫폼 모델이며, 세계에서 가장 큰 분산 응용인 Web에서 사용하고 있는 웹 구조 스타일 모델이다. REST는 분산 응용을 '상태 엔진'으로 본다. 클라이언트의 응용에 대한 요청은 응용의 현재 상태를 변화시키고, 이 변화된 상태는 클라이언트 요청에 대한 결과 정보를 포함하고 있다. 이 상태는 representation을 갖는데 보통 한 개 또는 여러 개의 문서로 표현되며(XML 문서), 이 문서는 하이퍼텍스트로 다른 상태와의 연결 고리(링크)를 갖는다. 즉, 응용의 상태들은 응용의 현재 정보를 나타내며 하이퍼텍스트 문서들로 표현된다. 이 하이퍼텍스트 문서들은 리소스로써 유일무이한 URI를 갖는다. 링크는 새로운 상태로의 이동을 지시한다. 다음 상태로의 전환은 다음 상태 정보를 표현하는 리소스로의 이동을 가르키는 링크를 선택함으로써 이루어진다. 또한, REST에서 리소스의 전달은 그 표현인 문서 형태로 HTTP를 통해 이루어진다. 응용의 클라이언트는 응용에 서비스를 요청하고, 이 서비스의 요청 결과는 응용의 요청 결과 상태를 표현하는 문서로써 클라이언트에 전달된다. REST에서 리소스에 대한 오퍼레이션은 보편적이며 보다 단순한 GET(문서 가져오기), PUT(문서 쓰기), POST(문서에 추가 정보 전달하기), DELETE(문서 삭제), HEAD(문서의 헤더만을 가져오기) 등의 메서드만을 지원하며, 이러한 메서드는 HTTP 에서 지원한다. 이상의 논의에서, REST 모델의 분산 컴퓨팅 응용은 확장성에 매우 유리함을 알 수 있다[6].

4. SOAP과 REST의 비교

REST는 "효율적으로 설계된 웹 애플리케이션이 행동하는 방법의 이미지를 불러일으키는 것으로, 사용자의 요청을 사용자가 옮기고, 그들의 사용에 대하여 관련된 다음 페이지로 링크(상태 전이)를 선택하는 것에 의해 진행되는 웹 페이지(가상 상태 머신)의 네트워크." 이라고 한다.

REST는 구조상의 형식으로 단지 그것을 이해하여, 그 형식에서 웹 서비스를 설계할 수 있다. REST는 표준이 아니며, HTTP, URL, XML/HTML/GIF/JPEG/etc(Resource Representations), text/xml, text/html, image/gif, image/jpeg, etc (MIME Types) 등을 표준으로 한다.

웹의 REST 형식은 표 1을 사용한다. 현재의 웹의 기본 요소는 URI, HTTP, XML(HTML) 이다. 이들 3가지 요소는 모두 확장성을 갖는다. XML은 자신의 문서 구조를 스스로 표현 할 수 있는 능력을 가지고 있어 그 확장성은 잘 알려져 있다.

URI는 리소스 이름이며 주소이다. URI는 리소스 위치를 표현하는 데 확장성을 갖는다. HTTP의 확장성은 어떠한 페이로드라도 헤더를 가지고 미리 정의된 또는 정의 가능한 새로운 메서드를 사용하여 전달할 수 있다는 데서 나온다. HTTP가 다른 프로토콜에 비해 특별한 것은 내장된 URI 지원이다[7-8].

RPC 중심의 SOAP 기반 웹 서비스와는 달리, REST에 기반한 웹 서비스는 문서 중심이다. SOAP 기반 웹 서비스에서는 웹 서비스 엔트리포인트에 대해서만 URI를 가지고 있고 내부의 다른 리소스에 대해서는 자기 자신의 어드레싱 방식을 사용하고 있고, 리소스를 발견하고 참조하는 데에 표준적인 방법이 아닌 자신의 메서드를 사용한다.

표 1 Web's REST style

Network Protocol	HTTP
Address Scheme	URIs-URLs and URNs
Resource Representations	XML, HTML, GIF etc.
Resource types	MIME Types-ext/xml, text/html,image/gif
Hyperlinks	XML's xlink attributes, HTML<a> tags

반면, REST 기반 웹 서비스는 기본적으로 웹의 주요 요소인 URI, HTTP 등을 그대로 이용하자는 것이다. REST 구조는 기존에 존재하는 웹 어드레싱 모델을 사용하여 모든 리소스에 대해 URI를 사용하고, 리소스 조작에 표준적인 방법을 사용한다. 즉, 기존의 웹 사용 모델을 그대로 사용하는 것이기 때문에, 기존 웹 사용자에게는 친숙한 사용 환경을 제공하므로 연속성이 높다. REST는 중개자에 의해 상호작용, 구성요소의 동적인 대용 가능성과 행동 처리의 캐싱과 재사용을 가능하게 하며, 인터넷의 확산으로 하이퍼미디어 시스템의 필요성을 갖는다. 현대의 웹은 REST 스타일 구조의 한 인스턴스 이다. 비록 웹에 기반을 둔 적용이 다른 스타일의 상호작용으로의 접근을 포함할 수 있지만, 그 프로토콜과 실행 중심의 초점은 분배되었던 하이퍼미디어이다[9].

끝으로 SOAP 기반 및 REST 기반의 웹 애플리케이션에서의 장단점 비교 연구가 요구된다.

5. 결론

본 논문에서는 메시지 프로토콜인 SOAP의 단점과 SOAP 기반 웹 서비스의 한계인 확장성을 극복하기 위하여 기존에 존재하는 웹 어드레싱 모델을 사용하여 모든 리소스에 대해 URI를 사용하고, 리소스 조작에 표준적인 방법을 사용하는 REST에 대하여 SOAP과 REST에 대하여 각각 살펴보고 비교하여 보았다.

최근 REST 웹 서비스는 HTML 보다는 XML을 사용한다. 따라서 이러한 서비스는 사용자에게 웹 서버에 의해 제공되는 디렉토리에 있는 자원의 목록화나 GET, PUT, DELETE 등과 같은 오퍼레이션을 제공할 수 있도록 고려해야 한다.

[참고문헌]

[1]"XML Web Service", http://msdn.microsoft.com/library/dnwebsrv/html.

[2] "Understanding SOAP", http"//msdn.microsoft.com/library/en-us/dnsoap/htm/.

[3]S.Jeelani Basha, Professional Java Web Services, WROX, pp.60-61, 2002.

[4]Scott Short, 'Building XML Web Services for the Microsoft .NET Platform, Microsoft, pp.48-49.

[5]J. Roy and A. Ramanujan, "Understanding Web Services, "IEEE IT Professional, vol. 3, issue 6, Nov.-Dec., pp.69-73, 2001.

[6]James Snell, Doug Tidwell, Pavel Kulcbenko, *Programming Web Services with* SOAP, pp.54-57, 2002.

[7]Amit Asaravala,"Giving SOAP a REST", http://www.devx.com/DevX/Article/8155, 2002, 10.

[8]Roger L. Costello, "Building Web Services the REST Way", http://www.xfront.com/REST Web Services.html.

[9]Roy T. Fielding and Richard N. Taylor, "Principled Design of the Modern Web Architecture", http://www.ics.uci.edu/~fielding/pubs/webarch_icse2000.pdf.