.NET 웹서비스 기술

정홍주* · 박화진**

웹 서비스는 인터넷 환경에서 분산 응용 프로그램을 개발하는 가장 좋은 대안이 될 것이다. 그 이유는 웹 서비스가 특정 회사의 독자적인 방법으로 제시된 개념이 아니라 개방형 표준을 따르는 프로토콜과 XML을 가지고 분산 응용 프로그램 환경을 제공하고 있기 때문이다.

.NET은 이러한 웹 서비스를 개발하기 위한 가장 강력한 기능을 제공하는데, 이는 Microsoft가 웹 서비스와 관련한 개방형 표준들을 정하는데 가장 주도적인 역할을 담당하고 있기 때문이다. 웹 서비스는 개발자가 다양한 응용 프로그램 로직을 단순히 웹 서버에 올리는 것만으로 쉽게 웹 서비스를 개발할 수 있다. 또한 웹 서비스를 제공받는 클라이언트들은 개방형 표준 프로토콜인 HTTP, SOAP, HTML, XML을 사용해서 어디서나 웹 서비스를 제공 받을 수 있다.

1. 문제의 배경

소프트웨어 개발의 역사에서 크게 두 가지의 전환점이 있었다. 하나는 컴포넌트 기술이고 다른 하나는 객체지향 프로그래밍이라고 할 수 있다. 객체 지향 프로그래밍은 80년대 초에 주류를 이루었다. 객체 지향 프로그래밍을 소프트웨어의 복잡성과 용량의 한계를 해결할 수 있는 새로운 해결책으로 보았다. 객체 지향은 코드의 구조를 객체로 이루어지도록 하여 재사용과 유지 보수를 가능하게 하였다. 하지만 아직도 많은 시간과 노력이 필요로 했었다.

1990년대에 컴포넌트 기술이 대두되었다. 이 기술은 여러 면에서 혁명적이었다고 할 수 있다. 1995년에 개발자들은 컴포넌트를 조립함으로써 응용 프로그램을 만들 수 있는 방법을 생각하게 되었다. 재사용이 가능한 비즈니스 컴포넌트에 대한 시장성도 예측했었다. 그렇지만 현실은 따라주지 못 했었다. 본질적으로 인터넷 Application은 컴포넌트 기술의 확장이라고 할 수 있다. 인터넷 Application은 앞서 이야기 하였듯이 이질적인 환경에서 Data를 주고 받으며, Application이 상호 연동할 수 있어야 한다.

그러나 컴포넌트 기술이 가지고 있었던 한계는 제조회사에 종속적인 면을 가지고 있었다.

2. IDL의 대안으로서 XML

Microsoft® Interface Definition Language (MIDL)은 client와 server프로그램 사이에서 interfaces를 정의한다. Microsoft사는 COM/DCOM의 interfaces와 remote procedure call

*㈜웹타임 교육센터, OpenSG 컨설턴트, Microsoft TechNet Advisor Group
**숙명여자대학교 멀티미디어과학과 교수

(RPC)개발자들이 활용 할 수 있게 Platform SDK에 MIDL 컴파일러를 포함한다. MIDL은 또한 OLE Automation을 위해 type library의 생성을 지원한다.

MIDL은 윈도우 OS를 기반으로 하는 모든 client/server application에 사용 될 수 있다. Unix 혹은 Apple과 같은 시스템을 포함하는 이질적인 network 환경에서 작동하는 client/server application을 생성하기 위해 사용 될 수 있다. Microsoft사는 1990년대 초에 OOP개념에서 코드의 재 사용성 등의 고려 사항들을 수용하기 위해 COM이라는 개발 모델을 만들었다.

COM 개체가 자신이 가지고 있는 기능을 client에게 노출시키고자 할 때 사용하는 기본적인 방법이 인터페이스(Interface)를 구현하는 것이다. 하나의 인터페이스는 어떤 서비스를 제공하는 하나의 메소드를 정의한다.

개념적으로 볼 때 인터페이스는 COM 개체와 client application간의 일종의 계약(Contract)이라고 할 수 있다. COM 개체는 이러한 계약(Contract)에 명시되어 있는 대로 서비스 즉 메소드를 제공하는 것이고 client application은 계약(Contract)에 명시되어 있는 대로 COM 개체가 제공하는 서비스 즉 메소드를 호출할 수 있다. 이러한 내용은 client application이나 COM 개체가 계약(Contract)이라는 표준에 따라서 작성되어 진다면 양자간 모두 어떤 언어로 작성되어 지든, 어떤 플랫폼이든 상관없이 계약(Contract)이라는 표준에 따라 서로 안전하게 서비스를 이용할 수 있다는 것을 의미한다.

따라서 COM 기술은 근본적으로 상호운용성에 관한 기술이라고 할 수 있다. 결국 각 COM 기술은 어느 정도까지 상호운용성을 가지느냐가 관건이라고 할 수 있을 것이다. COM 기술의 상호운용성의 정도를 살펴본다면 네 가지로 나누어 볼 수 있을 것이다.

• 인-메모리 상호운용(In-Memory Interoperation)

인-메모리 상호운용은 가장 높은 수준의 상호운용이라고 할 수 있다. 이것은 메모리 안에 다중의 컴포넌트를 혼합하는 것으로 모든 컴포넌트가 지켜야 하는 인-메모리 표현을 표준화해서 상호운용의 성능을 향상 시킨다. 또한 이러한 인-메모리 표준은 훨씬 더 광범위한 컴포넌트 관리를 적은 비용으로 할 수 있게 한다.

• 소스코드 상호운용(Source Code Interoperation)

컴포넌트 기술은 특정한 API(Application Programming Interface)에 대해서 개발자들이 명시적으로 프로그래밍 할 것을 요구하기도 한다. 즉 소스레벨에서 표준화하여 해당 컴포넌트 소스코드를 다른 업체에서도 다시 컴파일 할 수 있게 한다. COM은 COM 라이브러리와 Co api라는 것을 제공하여 플랫폼 간에 일관성을 유지하게 하며, COM 소스코드가 여러 플랫폼에서 재 컴파일 되는 것을 허용한다.

• Type 정보의 상호운용(Type Information Interoperation)

앞에서 언급한 컴포넌트 기술은 모두 개발자와 개발자에게 제공되어지는 지원 인프라가 사용하는 Type Information에 대한 표준화된 구현 방법을 제공하는 것이다. 텍스트 기반의 인터페이스 정의 언어(IDL)을 제공해서 모든 데이터의 Type을 공용으로 액세스할 수 있게 IDL로 정의한다면 어떤 프로그래밍 언어에서도 해당 개체를 액세스 할 수 있게 된다. IDL은 텍스트 기반이기 때문에 IDL 인식 Tool과 인프라는 고급언어뿐만 아니라 프로세서의 종속파일도 파싱이라는 절차를 수행해야 한다. 이 문제를 해결하기 위해 COM은

Type 라이브러리라고 하는 이진 형태의 Type 정보를 제공하게 된다. 이 Type 라이브러리에는 COM IDL에 있는 대부분의 정보가 들어 있게 된다.

• Wire 상호운용(Wire Interoperation)

컴포넌트 기술은 분산환경의 응용 프로그램을 쉽게 작성하는데 필요한 기술이다. 따라서 여러 컴포넌트가 분산된 여러 HOST 시스템을 통해서 서로 통신할 수 있도록 새로운 Network Protocol이 필요하다. Windows NT는 Open Software Foundation의 DCE (Distributed Computing Environment) RPC 메커니즘을 지향한다. Distributed COM (DCOM) 프로토콜 개체 활성화, 타입 표준화, 그리고 사이클 관리 등에 필요한 DCE RPC 인터페이스를 정의할 뿐이다. 따라서 DCOM은 또 다른 DCE RPC 응용 프로그램이라 할 수 있다.

지금까지 내용에서 볼 수 있듯이 상호운용성의 기술은 여러 가지가 있다. 특히 위에서 언급한 기술은 많은 사람들의 지지를 받아왔다. 그렇지만 위의 세 가지 기술 중 어느 하나도 Internet환경에 적용되지 않았다. Internet이 DCOM에서 실행될 것인가 또는 CORBA에서 실행될 것인가를 놓고 많은 논쟁이 있었지만 결국 Internet Protocol로 자리잡은 것은 HTTP (Hypertext Transfer Protocol)였다. 또한 기업의 많은 방화벽도 HTTP 패킷이 자사의 방화벽을 통과하는 것을 허용한다. HTTP는 아주 단순한 텍스트 기반의 프로토콜이며, 아주 적은 실행 시간만 지원된다고 하더라도 아무런 문제 없이 작동한다.

이러한 이유로, HTTP 서버 예를 들면 IIS와 Apache같은 웹서버의 확장성, 안정성 및 관리 용이성을 위한 노력들을 고려한다면, 오히려 HTTP 기술을 사용하여 소프트웨어 컴포넌트를 공개하는 것이 더 효율적일 수 있다.

이러한 과정 속에서 많은 사람들이 HTTP 프로토콜을 이용하는 XML에 대해 많은 연구를 하게 되었으며, 그 결과로 XML을 제 4의 컴포넌트 통합 기술로 간주하게 되었다. IDL과 XML이 가지는 가장 큰 공통점이라고 한다면 양자가 모두 하나의 계약(Contract) 또는 Specification을 작성하여 이를 근거로 데이터를 전송한다는 것이며, 다른 하나는 양자가 모두 데이터 자체에 대한 연산을 수행하는 것이 아니라 단지 데이터를 표현하고, 데이터 자체를 전달할 수 있다는 것이다. 즉 구현은 없다는 것이다.

그렇다면, XML이 컴포넌트 기반 시스템의 통합 기술 차원에서 사용되어 지는 이유는 무엇인가?

• XML은 최소의 컴포넌트 표준이다.

XML은 데이터와 메시지 교환을 위한 최소의 표준을 정의하고 있다. 이것은 컴포넌트가 서로 통신하는데 필요한 최소의 표준이라고 할 수 있다. 또한 이러한 표준을 구성하는 면에서 XML은 많은 융통성을 가지고 있으며, 그 계층적인 구조에도 불구하고 비 계층적인 구조의 데이터와도 잘 맞는다고 볼 수 있다. 따라서 기존의 컴포넌트 간 통신을 위한 계약(Contract) 대신에 XML의 Specification이 사용될 수 있으며, XML Specification에는 DTD가 있었으나 여러가지 한계로 인해서 현재는 Schema가 사용되고 있다.

• XML은 플랫폼이며, 언어인 동시에 벤더 독립적이다.

XML은 데이터를 전송하는 전송표현에 불과하기 때문에 어느 특정 운영체제나 프로그래밍 언어, 하드웨어의 아키텍처에 상관없이 메시지를 교환할 수 있으며, 따라서 두 시스템은 상호운용이 가능하다는 것이다. 또한 XML은 특정한 타입의

API나 인-메모리의 표현을 강요하지 않기 때문에 대부분의 프로그래밍 언어에서 적용할 수 있는 파서를 사용하면 다른 프로그래밍 언어로 개발되어진 각 Application에 대해서도 데이터를 교환할 수 있다. XML을 파싱하는데는 여러 표준화된 API가 있기는 하지만 다른 XML기반 시스템과 상호운용을 지원하기 위해 반드시 어떤 API를 사용해야 한다는 규정은 없다.

• 액세스가 가능한 XML

XML은 위에서 언급한 장점 이외에도 작성하기가 쉽고 읽기도 쉽다는 것이다. 이렇게 액세스가 쉬운 장점이 바로 XML에 관심을 기울이는 이유라고 할 수 있다. 이전의 컴포넌트 기술이 작성하거나 읽을 때 이진파일의 형태로 되어 상당한 어려움을 겪었지만 XML은 단순한 텍스트 편집기와 스크립팅 언어를 사용하여 쉽게 만들 수 있다.

• 확장성이 뛰어난 XML

XML은 주어진 데이터 스트림을 확장할 수 있게 한다, 즉 XML 네임스페이스는 URI(Uniform Resource Identifier) 네임스페이스를 통해 Attribute나 Element를 기존의 XML Document에 추가할 수 있도록 한다.

• XML - 적절한 수준의 Strong Type

XML은 개방형 어휘(vocabulary)와 구조(Structure)를 사용하기 때문에 weak 타입의 통신을 지원할 수 있다. 따라서, 범용의 응용프로그램, 데이터 중심의 응용프로그램 등에 XML을 사용할 수 있다. 즉 ADO Recordset을 사용하던 개발자들은 Recordset 대신에 Microsoft XML 파서(MSXML)를 데이터를 전송하는 기술로 사용하기 시작했으며, 이는 MSXML이 플랫폼 전환에 있어서 많은 장점을 가지고 있을 뿐만 아니라 비관계형 테이블 형식의 데이터를 보다 잘 지원하기 때문이다.

• 상호운용 관련 문제의 해결

이러한 내용을 바탕으로 해서 XML을 컴포넌트 통합 기술로 채택한다고 해서 상호 운용 문제가 완전히 해결될 수 있는 것은 아니다. 컴퓨터 산업의 상당 부분이 XML을 상호 운용 기술로 사용하고 있기는 하지만, 이것은 추상적인 차원에서 상호 운용 문제의 레벨을 한 단계 더 높은 수준으로 끌어올린다는 것을 의미할 뿐이다. 컴퓨터 업계 모두가 동시에 XML로 전환한다고 해도 여러 조직들이 서로 다른 XML vocabulary와 Structure를 사용할 것이 분명하다. 또한, 현재 도메인 전용 XML vocabulary와 Structure를 표준화려는 움직임이 활발하게 진행되고 있지만 이러한 현상도 또한 특정 응용 프로그램 도메인으로 집중되는 경향이 발생할 수도 있으며, 한다 하더라도 100% 특정 응용 프로그램 도메인으로 집중될지는 아직 알 수 없다. 그러나, 표준화된 vocabulary의 부재 현상을 XML 기술을 해결할 수 있다는 점은 그나마 다행이라고 할 수 있다. 특히, 두 개의 vocabulary가 존재하는 경우, 응용 프로그램 레벨의 게이트웨이에 의해 vocabulary "A"의 요청이 vocabulary "B"의 요청으로 변환 될 가능성이 크다. 이보다 더 유력한 솔루션은 XML 변환인데, XML 변환 기술은 변환 규칙을 지정하여(물론 XML로) 특정 XML vocabulary가 다른 vocabulary로 변환될 수 있게 한다. XML 변환은 원래 XML을 HTML로 맵핑하기 위해 고안된 것이지만, 현재는 다양한 시나리오에 응용되고 있다.

XML에는 지금까지 많은 수식어들이 따라 다니지만 XML이 모든 문제를 해결하는 것은 아니다. XML을 어떻게 활용하느냐에 따라서 개발의 수준을 높여 줄 수도 있으며, 그 반대일 수도 있다. XML 결코 C++이나 Java같은 프로그래밍 언어를

대신할 수는 없다. 그렇지만 분명한 것은 XML이 컴포넌트의 상호운용을 위한 수단으로 기존의 IDL을 대신하여 보다 유연하며, 효율적인 수단으로 사용될 것이라는 것과 개별적인 이기종 시스템 간의 게이트웨이 역할을 수행할 것이라는 것이다.

3. WebService의 구조

앞에서 언급했듯이 컴포넌트의 상호운용성을 위하여 IDL의 대안으로 XML이 등장하면서 컴포넌트간의 통신 프로토콜로 자리 잡게 되었다. 여기서 XML은 HTTP에 실어서 보낼 수 있는 프로토콜이라는 특징 때문에 Internet을 통하여 서비스의 호출 즉 메서드의 호출이 가능하게 되었다. 다시 말해서 웹 서비스는 웹을 통한 메서드의 호출이라는 말로 표현할 수 있다. 이로 인해 웹을 통해서 모든 이질적인 환경에서 응용 프로그램의 상호운용이 가능하게 되었다. 이러한 현상을 웹 서비스 측면에서 구체적으로 살펴본다면 XML 기반의 IDL이 여러 가지 다른 방법으로 출현하게 되었으며 여기서, 이러한 IDL을 표준화할 필요성을 인식하고 탄생한 것이 WSDL (Web Service Description Language) 이다. 현재 이 WSDL을 통해서 다른 사람에게 웹 서비스로 연락하고 사용하는 방법을 알려준다.

XML을 이용한 웹 서비스는 다음과 같은 구조로 이루어져 있다.

XML Web Services는 다른 시스템에서 인터넷을 통해 액세스할 수 있는 프로그래밍 가능한 하나의 단위이다. XML Web Services는 상호 운용성을 지원하는 XML, HTTP 및 기타 인터넷 표준을 광범위하게 수용한다.

XML Web Service는 단일 응용 프로그램에서 내부적으로 사용되거나 여러 응용 프로그램에서

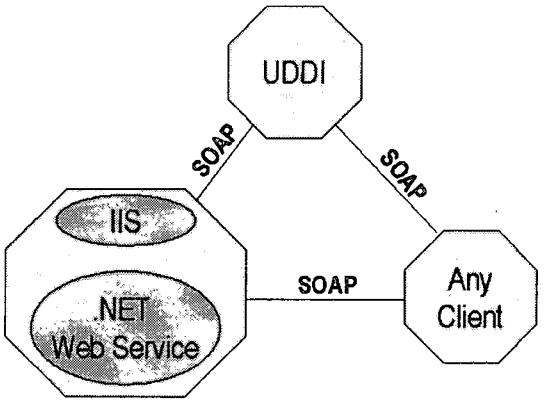

그림 1. .NET WebService 아키텍처

사용하기 위해 인터넷 상에서 외부적으로 제공될 수 있다. XML Web Service는 다양하고 이질적인 시스템이 단일 참조 방식으로 웹에서 함께 작동하는 것을 허용하는 표준 인터페이스를 통해 액세스할 수 있다.

XML Web Services는 코드 이식성에 대한 일반 기능을 사용하는 대신 데이터 및 시스템의 상호 운용성을 가능하게 하는 실행 가능한 솔루션을 제공한다. XML Web Services는 일치하지 않는 구성 요소 모델, 운영 체제 및 프로그래밍 언어를 사용하는 시스템 사이에서 데이터를 교환하는 XML 기반 메시징을 사용한다. 개발자는 분산 응용 프로그램에서 일반적으로 구성 요소를 사용하는 것과 상당히 유사한 방식으로 다양한 소스에서 XML Web Services를 함께 연결하는 응용 프로그램을 만들 수 있다.

XML Web Service의 중요한 특징 중 하나는 서비스 구현과 소비 사이에 기존에 Location 정보와 같은 강력한 유형의 매핑이 존재해야 하는 환경과는 다르게 서비스를 만들어 액세스하는 메커니즘으로 XML 기반 메시징을 사용하면 XML Web Service 클라이언트와 XML Web Service 공급자는 입력, 출력 및 위치 외에는 서로에 대해 알 필요가 없다.

XML Web Services는 분산 응용 프로그램 개발의 새 시대를 가능하게 한다. 응용 프로그램이 자신의 인프라를 사용하는 완전한 결합 시스템에서는 응용 프로그램의 상호 운용성을 잃게 된다. XML Web Services는 상호 운용성을 가능하게 하는 완전히 새로운 수준의 상호 운용성을 제공한다. 인터넷의 차세대 발전 형태인 XML Web Services는 모든 컴퓨터 장치를 함께 연결하는 기본 구조가 될 것이다.

ASP.NET 프레임워크는 관리되는 코드에서 XML Web Services의 프레임워크 역할도 한다. 그러한 XML Web Services에서는 인증, 캐싱 및 상태 관리 등 .NET Framework의 많은 기능에 액세스할 수 있다. 따라서 개발자는 인프라 코드를 작성하지 않아도 자유롭게 XML Web Services를 만들거나 액세스할 수 있다.

ASP.NET 응용 프로그램 모델에서 브라우저용 웹 페이지는 .aspx 확장명을 사용한다. 일반 ASP.NET 페이지와 XML Web Services를 구분하기 위해 XML Web Services에서는 .asmx 확장명을 사용한다.

XML Web Services는 XML Web Service 진입점과 XML Web Service 기능을 구현하는 코드로 구성된다. ASP.NET에서 .asmx 파일은 XML Web Service의 진입점을 알려주는 역할을 한다. 그리고 미리 컴파일된 어셈블리의 코드나 코드 숨김 파일 또는 .asmx 파일 자체에 포함된 코드를 참조한다.

XML Web Services 작업을 할 때에는 다음과 같은 두 가지 기본 역할이 있다.

• **XML Web Service 만들기** : XML Web Service를 만들 때 XML Web Service 클라이언트에 기능을 제공하는 응용 프로그램을 만든다.

• **XML Web Service에 액세스** : XML Web Service에 액세스할 때 클라이언트 응용 프로그램에서는 해당 XML Web Service 내에 포함된 기능을 찾아 참조하고 사용한다.

XML Web Services는 독립 실행형 응용 프로그램이 되거나 더 큰 웹 응용 프로그램의 하위 구성 요소가 될 수 있다. 최소한 클라이언트에서 XML Web Service에 메시지를 전달할 수 있어야 한다.

현재 응용 프로그램을 만드는 데 사용할 수 있는 플랫폼은 많이 있다. 각 플랫폼은 전통적으로 시스템 간 통합을 위해 이진 특성을 갖는 자체 프로토콜을 사용했기 때문에 플랫폼 내의 응용 프로그램들만 데이터를 공유할 수 있다는 제한이 있었다. 이러한 제한을 인식하게 되면서 데이터 형식과 데이터 교환의 표준화 작업이 엄청난 속도로 진행되고 있다. 이러한 추이는 전통적인 하드웨어와 소프트웨어 장벽이 자연스럽고 웹 사용 가능한 서비스 통합의 새로운 컴퓨팅 패러다임으로 빠르게 진화할 것이라는 시각에서 유래한다.

이러한 시각의 중심에는 상호 운용성(간단히 "interop")의 개념 또는 종류가 다른 시스템 간에 데이터를 자연스럽게 공유하고 통신할 수 있는 기능이 자리하고 있다. 이것이 웹 서비스의 목표이다. 웹 서비스는 표준 인터넷 프로토콜을 사용하여 액세스할 수 있는 프로그램 가능 응용 프로그램 논리 또는 시스템 간 통신과 응용 프로그램 간 통신을 투명하게 하기 위한 웹 지원 표준의 구현이다.

SOAP(Simple Object Access Protocol), WSDL(Web Service Description Language), HTTP(HyperText Transfer Protocol) 등과 같은 많은 웹 서비스 기술이 시스템 간에 메시지를 주고 받는 데 사용되고 있다. 이러한 메시지는 메서드 호출에서 구매 주문서 제출에 이르기까지 복잡

하고 다양하다. 웹 서비스의 일반적인 고급 기능 중 하나는 RPC 스타일 통신(한 시스템의 프로그램에서 다른 시스템의 프로그램을 실행하는 원격 프로시저 호출)을 구현하는 것이다..

RPC 스타일 SOAP 메시징을 수행하는 경우 여러 가지 이유로 상호 운용성 문제가 발생할 수 있다. 흥미롭게도 많은 상호 운용성 문제는 본질적으로 SOAP 문제가 아니라 원본으로 사용하는 전송 또는 XML 엔진과의 상호 운용성 문제이다. 다시 말해서 상호 운용성 문제는 다음과 같다.

- HTTP 문제
- XML 문제
- SOAP 불연속성

또한 이 사양을 만든 이가 확실하지 않은 경우와 모호성으로 인해 올바른 단일 동작을 결정하기가 매우 어려운 경우에도 발생한다.

XML Web services 아키텍처의 주요 장점 중 하나는 별도의 플랫폼에 별도의 언어로 작성된 프로그램들이 표준 기반으로 서로 통신할 수 있다는 점이다. 이 점에 대해 산업 분야에 종사했던 사람들은 DEC 이전과 CORBA에서도 같은 내용이 있었기 때문에 차이점이 없을 것 같다는 의문이 생길 수 있다. 첫 번째 차이점은 SOAP가 이전의 접근 방법보다 훨씬 간단하여 표준 규격의 SOAP 구현을 입력하는 작업이 더욱 용이하다는 것이다. SOAP 구현은 규모가 큰 소프트웨어 회사에서 찾을 수 있지만 개인 개발자가 작성하고 유지 관리하는 구현도 많이 찾을 수 있다. 이전에 비해 XML Web services가 갖는 또 다른 큰 장점은 표준 웹 프로토콜인 XML, HTTP 및 TCP/IP와 작동한다는 것이다. 수 많은 회사들이 웹 인프라와 이를 유지 관리하는 지식과 경험이 있는 직원들을 갖추었기 때문에 XML Web services에 대한 항목에 소모되는 비용이 이전 기술에 비해 현저히 감소된다.

지금까지 WSDL 파일에 설명되고 UDDI에 등록된 XML Web service를 SOAP를 통해 웹에 나타나는 소프트웨어 서비스로 정의했다. 그 다음 논리적인 질문은 XML Web services로 무엇을 할 수 있는가 하는 것이다. 처음에 XML Web services는 주식 시세, 일기 예보, 스포츠 등 응용 프로그램에 쉽게 통합될 수 있는 정보원이었다. 원하는 정보를 분석하고 집계하여 다양한 방법으로 표시하도록 작성할 수 있는 응용 프로그램이 있다. 예를 들어, Microsoft® Excel 스프레드시트를 사용하여 주식, 401K, 은행 계좌, 대출 등의 전체 금융 상태를 요약할 수 있다. 이 정보를 XML Web services에서 사용할 수 있는 경우 Excel은 이 Web services를 계속 업데이트한다. 이 정보 중 일부는 무료이며 어떤 정보는 서비스 구독이 필요할 수도 있다. 대부분의 정보는 웹에서 현재 사용할 수 있지만 XML Web services는 더욱 쉽고 안정적인 프로그래밍 방식으로 액세스 된다.

기존 응용 프로그램을 XML Web services로 나타내면 XML Web services를 빌딩 블록으로 사용하는 더욱 강력한 새 응용 프로그램을 작성할 수 있다. 예를 들어, 여러 공급업체의 가격 정보를 자동으로 얻을 수 있고 공급업체를 선택할 수 있으며 주문한 다음 제품이 도착할 때까지 배달 과정을 확인할 수 있다. 공급업체 응용 프로그램은 웹 상에 서비스를 표시한 다음 XML Web services를 사용하여 고객의 신용 확인, 요금 부과 및 배달 회사에 배달을 의뢰할 수도 있다.

앞으로 몇몇 중요한 XML 웹 서비스를 사용하면 웹을 사용하는 응용 프로그램을 지원하여 오늘날에는 실행하지 못하는 사항들을 실행할 수 있게 될 것이다. 예를 들어, 일정 관리 서비스는 Microsoft .NET My Services 프로젝트가 지원하는 서비스 중 하나이다. 치과 의사나 정비사가

XML Web service를 사용하여 일정을 관리하는 경우 고객이 온라인으로 시간 약속을 예약하거나, 고객이 원할 경우 치과 의사나 정비사가 직접 고객과 청소 및 정기 점검 약속을 할 수 있다. 조금만 생각해 보면, 웹을 프로그램 할 수 있는 능력이 있기만 하면 작성할 수 있는 응용 프로그램이 많이 있다.

SOAP는 XML Web services용 통신 프로토콜이다. SOAP를 통신 프로토콜로 설명하면 대부분의 사람들은 DCOM 또는 CORBA를 생각하고 "SOAP가 객체를 활성화하는 방법은?" 또는 "SOAP가 사용하는 이름 서비스는?" 등과 같은 질문을 한다. SOAP 구현에 이러한 내용이 포함되기는 하지만 SOAP 표준은 이 내용을 지정하지 않습니다. SOAP는 메시지 즉, 요청된 사양의 일부에 대한 XML 형식을 정의하는 사양이다. 몇몇 SOAP 요소에 포함된 올바른 형식의 XML 조각이 있는 경우 SOAP 메시지가 나타납니다. 간단하다.

SOAP 사양의 다른 부분은 프로그램 데이터를 XML로 표시하고 SOAP를 사용하여 원격 프로시저 호출을 수행하는 방법을 설명한다. 이러한 사양의 선택적 부품은 클라이언트에서 전송되는 SOAP 메시지에 호출 가능한 함수와 해당 함수에 전달되는 매개 변수가 포함되어 있는 RPC 스타일 응용 프로그램을 구현하는 데 사용되며 서버는 실행된 함수의 결과와 함께 메시지를 반환한다. COM 또는 CORBA 응용 프로그램을 실행하는 프로그래머들은 RPC 스타일을 이해하기 때문에 현재 대부분의 SOAP 구현은 RPC 응용 프로그램을 지원한다. 또한 SOAP는 SOAP 메시지가 XML 문서에서 래퍼 기능만 수행하는 문서 스타일 응용 프로그램도 지원한다. 문서 스타일 SOAP 응용 프로그램은 매우 융통성이 있으며 여러 새 XML Web services는 이 장점을 활용하여 RPC를 사용하여 구현하기 어려운 서비스를 만든다.

SOAP 사양의 마지막 선택 부분은 SOAP 메시지를 포함한 HTTP 메시지의 형식을 정의한다. HTTP 바인딩은 HTTP가 거의 모든 현재 OS 및 현재가 아닌 여러 OS에 의해 지원되기 때문에 중요하다. HTTP 바인딩은 선택 사항이지만 SOAP에 대해 유일하게 표준화된 프로토콜이기 때문에 거의 모든 SOAP 구현은 HTTP 바인딩을 지원한다. 이러한 이유로 인해 SOAP에 HTTP가 필요한 것으로 잘못 생각하는 경우가 있다. 몇몇 구현은 MSMQ, MQ 시리즈, SMTP 또는 TCP/IP 전송을 지원하지만 거의 모든 현재 XML Web services는 흔히 사용되는 HTTP를 사용한다. HTTP는 핵심 웹 프로토콜이므로 대부분의 회사는 HTTP를 지원하는 네트워크 인프라와 이를 관리하는 인력을 갖추고 있다. 보안, 모니터링 및 HTTP에 대한 로드 균형 인프라는 오늘날 쉽게 사용할 수 있다.

SOAP를 시작할 때 혼동되는 주요 원인은 SOAP 사양과 여러 SOAP 사양 구현간의 차이점이다. SOAP를 사용하는 대부분의 사람들은 SOAP 메시지를 직접 작성하지는 않지만 SOAP 도구 키트를 사용하여 SOAP 메시지를 만들고 구문 분석한다. 일반적으로 이러한 도구 키트는 함수 호출을 몇몇 종류의 언어에서 SOAP 메시지로 변환한다. 예를 들어, Microsoft SOAP Toolkit 2.0은 COM 함수 호출을 SOAP로 전환하고 Apache Toolkit는 JAVA 함수 호출을 SOAP로 변환한다. 함수 호출의 유형과 지원되는 매개 변수의 데이터 유형은 각 SOAP 구현에 따라 다양하기 때문에 한 개의 도구 키트와 함께 작동하는 함수는 다른 도구 키트와 작동하지 않을 수도 있다. 이것은 SOAP의 제한 사항이 아니라 사용자가 사용하는 특정 구현이다.

훨씬 강제적인 SOAP의 기능은 많은 별도의 하

드웨어 및 소프트웨어 플랫폼에 구현되었다는 점이다. 이것은 사용자의 회사 내부, 외부의 서로 다른 시스템을 연결하는 데 SOAP를 사용할 수 있다는 의미이다. 과거에 시스템을 통합하는 일반 통신 프로토콜을 개발하려는 노력은 많았지만 아무도 SOAP를 일반적으로 채택하지는 않았습니다. 왜냐하면, SOAP는 이전의 많은 프로토콜에 비해 구현하는 데 훨씬 작고 단순했기 때문이다. 예를 들어, DCE 및 CORBA는 구현하는 데 수 년이 걸렸기 때문에 소수의 구현만이 출시되었다. 그러나 SOAP는 기존의 XML 파서 및 HTTP 라이브러리를 사용하여 대부분의 힘든 작업을 수행하므로 수 개월 내에 SOAP 구현을 완료할 수 있다. 따라서 70종류의 SOAP 구현을 사용할 수 있다. SOAP는 DCE 또는 CORBA가 수행하는 작업을 모두 수행하지는 않지만 기능 교환이 복잡하지 않기 때문에 SOAP를 쉽게 사용할 수 있다.

HTTP의 보편성와 SOAP의 간소성이 동시에 존재하면 거의 모든 환경에서 호출할 수 있는 XML Web services를 구현하는 데 있어 이상적인 기반이 된다.

whiz-dull이라고도 하는 WSDL는 Web Services Description Language의 약어이다. 편의상, WSDL 파일을 SOAP 메시지 집합 및 해당 메시지가 교환되는 방법을 설명하는 XML 문서라고 할 수 있다. 다시 말해서 WSDL은 IDL이 CORBA 또는 COM인 SOAP이다. WSDL은 XML이기 때문에 읽을 수 있고 편집할 수 있지만 대부분의 경우에는 소프트웨어에 의해 작성되고 사용된다.

WSDL의 값을 보려면 비즈니스 파트너가 제공하는 SOAP 메서드를 호출해야 한다. 비즈니스 파트너에게 몇몇 예제 SOAP 메시지를 요청하고 해당 예제와 같은 메시지를 작성하고 사용하는 응용 프로그램을 작성할 수 있으나, 오류가 발생하기 쉽다. 예를 들어, 고객 ID가 2387인 경우 메시지가 문자열이면 ID는 정수이다. WSDL은 요청 메시지에 포함되는 사항과 응답 메시지의 형식을 명백한 노테이션으로 지정한다.

WSDL 파일이 메시지 형식을 설명하기 위해 사용하는 노테이션은 XML 스키마 표준을 기반으로 하며, 이것은 프로그래밍 언어 중립적이며 또한 표준 기반이어서 다양한 플랫폼과 프로그래밍 언어에서 액세스할 수 있는 XML Web services 인터페이스를 설명하기에 적합하다는 것을 의미한다. WSDL은 메시지 컨텐트를 설명할 뿐만 아니라 서비스를 사용할 수 있는 위치 및 서비스와 대화하는 데 사용되는 통신 프로토콜을 정의한다. 즉, WSDL 파일은 XML Web service와 함께 작동하는 프로그램을 쓰는 데 필요한 모든 사항을 정의한다. WSDL 파일을 읽고 XML Web service와 통신하는 데 필요한 코드를 생성할 수 있는 몇 가지 도구가 있다. 그 중에서 성능이 뛰어난 몇몇 도구들이 Microsoft Visual Studio® .NET에 있다.

많은 현재 SOAP 도구 키트에는 기존 프로그램 인터페이스에서 WSDL 파일을 생성하는 도구가 포함되지만 WSDL을 직접 작성하는 도구는 거의 없으며 WSDL에 대해 필요한 만큼 도구가 지원되지 않는다. WSDL 파일 작성자에게 도구가 지원되어 COM IDL과 같은 Proxy 및 Stub을 작성하는 것은 대부분의 SOAP 구현의 일부분이 된다. 이런 점에서 WSDL은 XML Web services에 대한 SOAP 인터페이스를 만드는 데 좋은 방법이다.

UDDI(Universal Discovery Description 및 Integration)는 Web services의 Yellow 페이지이다. 전통적인 Yellow 페이지에서는 필요한 서비스를 제공하는 회사를 찾아 제공된 서비스를 검토한 후에 담당자와 연락하여 자세한 정보를 구할

수 있다. 지하실에서 비즈니스를 시작하고 말로 광고를 할 수 있는 것처럼 UDDI에 등록하지 않고도 Web service를 제공할 수는 있다. 그러나 시장을 넓히기 위해서는 고객이 Web services를 찾을 수 있도록 UDDI에 등록해야 한다.

UDDI 디렉터리 항목은 비즈니스 및 제공되는 서비스를 설명하는 XML 파일이다. UDDI 디렉터리의 항목은 세 부분으로 되어 있다. "화이트 페이지"는 서비스를 제공하는 회사의 이름, 주소, 연락처 등을 설명한다. "Yellow 페이지"에는 북미 산업 분류 시스템 및 표준 산업 분류 등과 같은 표준 분류법을 기반으로 하는 산업 범주가 나와 있다. "그린 페이지"는 Web service를 사용하는 응용 프로그램을 작성할 수 있도록 서비스에 인터페이스를 세부적으로 설명한다. 서비스 방법은 유형 모델 또는 tModel이라고도 하는 UDDI 문서를 통해 정의된다. 많은 경우에, tModel에는 XML Web service에 SOAP 인터페이스를 설명하는 WSDL 파일이 포함되어 있지만 tModel은 융통성이 있어 거의 모든 종류의 서비스를 설명할 수 있다.

또한 UDDI 디렉터리에는 사용자의 응용 프로그램을 작성하는 데 필요한 서비스를 검색하는 몇 가지 방법이 있다. 예를 들어, 지정된 지역 및 지정된 유형의 비즈니스로 서비스 공급자를 검색할 수 있다. 그런 다음 UDDI 디렉터리는 정보, 연락처, 링크 및 기술적인 데이터를 공급하여 사용자의 요구 사항에 맞는 서비스를 평가할 수 있도록 한다.

UDDI를 사용하여 Web services를 얻을 수 있는 비즈니스를 찾을 수 있다. 비즈니스 파트너는 알고 있지만 제공되는 서비스가 무엇인지 모를 경우에는 WS-Inspection specification 을 사용하여 특정 서버에 제공된 XML Web services의 컬렉션을 검색하여 필요한 서비스를 찾을 수 있다.

정 홍 주

- 2003년 숭실대 정보과학대학원 정보통신학과 석사
- 현재 : (주)웹타임 교육센터, OpenSG 컨설턴트, Microsoft TechNet Advisor Group
- 자격사항
  MCAD (Microsoft Certified Application Developer)
  MCSD (Microsoft Certified Solution Developer)
  MCDBA (Microsoft Certified Database Administrator)
  MCSE (Microsoft Certified System Engineer)
  MCT (Microsoft Certified Trainer)

박 화 진

- 1987년 숙명여자대학교 전산학과 졸업 (학사)
- 1989년 숙명여자대학교 대학원 전산학과 졸업 (석사)
- 1997년 Arizona State University Computer Science 컴퓨터 그래픽 전공(공학박사)
- 1997년~1998년 삼성 SDS 연구소 선임연구원
- 1998년~2000년 평택대학교 전임강사
- 2000년~현재 숙명여자대학교 멀티미디어과학과 교수
- 관심분야 : 컴퓨터 그래픽, 3D 모델링, 가상현실, 멀티미디어
- e-mail: hwajinpk@sookmyung.ac.kr