.NET 기술동향

차재호* · 권순각**

.NET은 Microsoft사의 Window와 Web 개발을 위한 차세대 Platform이라고 할 수 있다. Microsoft사는 개발자들을 위해 Runtime환경을 제공하는 플랫폼과 이러한 환경에 맞는 진보된 Language, 그 기반이 되는 확장 가능한 여러 Class Library를 새로운 .NET Framework로 응용프로그램의 개발과 배포를 단순화하고, 분산환경에서 응용프로그램의 설계 및 배포를 위한 가장 좋은 방법으로 XML Web Service를 추진하고 있다.

이러한 Microsoft사의 .NET 혁신은 Internet기반의 분산응용프로그램 기술로 Microsoft사의 역량을 재편성하고 모든 초점을 맞추는 것을 의미하며, 이러한 .NET의 목표로 다음과 같은 내용을 들 수 있다.

◦ Internet과 분산 응용프로그램을 위한 새로운 환경을 제공하는 기반 플랫폼의 제공

◦ 이를 이용한 응용프로그램 개발 및 배포의 단순화

◦ 이러한 분산환경에서 응용프로그램의 설계 및 배포를 위한 XML Web Service

◦ 시스템과 응용프로그램간의 상호운영성과 통합성의 향상

◦ "Any Device"가 의미하는 것과 같이 어떤 장치에서도 응용프로그램에 대한 일반적인 접근등을 들 수가 있다.

Microsoft .NET의 배경이 되는 원천은 .NET이 컴퓨팅 환경의 개개의 장치와 Web 사이트가 단순하게 Internet을 통해 연결되어 있는 환경으로부터 장치, 서비스, 컴퓨터들이 훨씬 더 효율적이고 향상된 기능을 가진 Solution을 Client에게 제공하기 위해 유기적으로 협력하는 환경을 만들겠다는 것이다.

Microsoft .NET은 다음과 같은 네 가지 핵심 구성요소를 포함하고 있다.

◦ .NET Building Block Service, Calendar, Passport.NET과 같은 서비스에 대한 프로그래밍적 접근

◦ 어떤 새로운 Internet 장비에서도 동작하는 .NET 장치 소프트웨어

◦ 보다 자연스러운 Interfacem 정보관리자, 스마트 태그 같은 사용자 정의 문서의 단어들과 단락들에 대한 정보를 자동적으로 하이퍼링크 시켜주는 .NET User Experience

◦ .NET Framework, VS.NET, .NET Enterprise Server 등의 .NET 하부구조

단순하게 .NET이라고 하면 .NET 하부구조를 의미 하는 것이지만 Microsoft사의 .NET은 자사

*㈜필라넷 수석컨설턴트, ㈜웹타임 전임강사
**동의대학교 소프트웨어공학과 교수

의 향후 Internet 환경을 중심으로 펼쳐지는 고도의 분산환경을 향한 정책을 말하며, 이 글에서는 그냥 .NET이라고 하면 .NET 하부구조를 의미 하는 것이다.

다음 장에서 알아보게 될 내용은 처음 Internet에서 시작하여 .NET에 이르기까지의 발전과정을 먼저 살펴보고, 새로운 컴퓨팅 환경으로서의 .NET을 조금 세부적으로 살펴본 다음 향후 발전과정을 추측해 보기로 한다.

1. Internet 소개

Internet의 발전을 간략하게 살펴보면 물론 Network의 발전은 핵심적인 역할을 수행했다고 볼 수 있지만, 그럼에도, 컴퓨터에 관련된 여러 가지 기술의 뒷받침이 없었다면 이루어 지지 않았을 것이다.

그런 면에서 몇 가지를 살펴 본다면 크게 3가지로 나누어 볼 수 있을 것이다.

첫째: 컴퓨터 자체 즉 하드웨어의 발전과 시스템적인 측면의 발전(OS)
둘째: 컴퓨터가 처리하는 Data의 처리 기술의 발전
셋째: Network의 발전

여기서 이야기 하고자 하는 것은 Microsoft의 .NET을 중심으로 기술되어 지는 것이기 때문에 하드웨어 적인 측면은 제외하고자 한다. 시스템의 발전과 Data의 처리 기술 중 소프트웨어적인 면만 살펴보고자 한다.

우선 최초의 컴퓨터인 에니악(ENIAC)이 발명된 후 20여 년 동안 컴퓨터간의 통신을 위해 여러 가지 방법들이 시도되어 왔지만 큰 진전은 없었다. 1960년대 들어서 컴퓨터분야의 과학자들은 컴퓨터들간 통신의 필요성을 절실히 깨닫기 시작했으며, 이 때부터 일부 대학의 컴퓨터들간의 Network과 군용 컴퓨터들을 연결한 Network이 만들어지기 시작했다.

또한, Network의 발전을 가져오기 시작하면서 거의 같은 시대에 컴퓨터에서의 Data처리 기술의 급속한 발전도 가져오게 되었다.

컴퓨터 자체는 1948년 벨 연구소에서 트랜지스터(transistor)가 발명되면서부터 컴퓨터는 급속도로 발전하기 시작했다. 이 시대를 제2세대 컴퓨터라고 하며, 아직은 사용자 프로그램에 있어서 문제가 되는 시기였다.

IBM사는 1964년 4월 집적회로(IC: Integrated Circuit)를 기억장치 구성소자로 사용한 'system 360'이라는 새로운 기종을 발표하였는데 이때부터를 제3세대라고 부른다. 제3세대 컴퓨터가 출현하면서 컴퓨터는 기계적인 측면, 즉 하드웨어(hardware) 면에서도 혁신적인 발전이 이루어졌고 또한 제3세대 컴퓨터에서는 이용하는 기술적인 측면, 즉 소프트웨어(software)면에서도 획기적인 발전을 가져왔다.

여기서, 우리가 주목할 것은 1960년대 중반에 제3세대 컴퓨터로 넘어 오면서 하드웨어적인 면이 뒷받침 되면서 소프트웨어 적인 측면에서도 발전을 가져오게 되었다는 것이다. 따라서 다양한 소프트웨어를 구사할 수 있는 기능이 크게 개선되었을 뿐만 아니라, 관리프로그램과 처리프로그램 및 사용자 프로그램 등의 소프트웨어 체계가 확립되었다. 즉 이 시대에 운영체제, 다중프로그램, 실시간 처리시스템, 시분할시스템 등이 실현되었다.

1960년대 말 미국 정부는 효과적인 대학교육과 복잡한 국방 Project를 추진하기 위해서는 컴퓨터간의 통신이 필요하게 되었으며 이로 인하여 작은 규모의 Network들을 서로 연결하기 위한 Network으로 인터넷 전신인 ARPANET(U.S Advanced Research Projects Agency Network) 개

발 Project를 수립하게 되었다. ARPANET Project의 목적을 살펴보면,

- 일부의 물리적 손상이 전체로 확대되는 것을 방지할 수 있는 Network설계
- 운용 중 새로운 Node가 추가되어도 영향을 적게 받는 Network설계
- 다양한 컴퓨터들간 원활한 통신이 가능한 Network개발

이라고 할 수 있는데,

예전의 (그리고 현재의 몇몇) Network들은 제3세대 컴퓨터가 등장하면서 소프트웨어의 발전이 이루어 지면서 몇몇 사용자들에 의해 사용되는 독립적인 형태로 나마 하드웨어 사양을 자신들이 속해있는 Network 사양에 맞추어 설정하게 되었다. 이 당시에는 각각 개별적으로 자신의 시스템에 맞는 Data Format과 Data를 전송하기 위한 프로토콜이 있었다.

여기서, 좀 더 유심히 살펴본다면 해당 시대에 각 Node들 즉, ARPANET에 연결되는 대학이나 군부대의 시스템이 이러한 이유로 서로 상이하다는 것이다. 더욱이 이 당시 모든 Network 사양을 지원하는 단일 Network가 존재하지 않기 때문에 하나의 하드웨어 사양을 이용한 광범위한 Network를 만드는 것이 불가능 했다. 따라서 서로 다른 상이한 시스템에서의 Data교환 Format에 관련된 문제와 Data를 전송하기 위한 기술이 필요로 하게 되었다. ARPANET Project에서 중점적으로 다루어 진 것은 서로 상이한 시스템에서 공통적으로 다룰 수 있는 Data Format과 Data를 전송하기 위한 공통적인 Protocol이었다.

이러한 노력의 여파로 해당 시기에 여러 가지 많은 표준들이 작성되기 시작했으며, 이들 표준들은 지금까지 많은 영향을 미치고 있다. 개별적으로 살펴보면 우선 가장 중요한 영향을 미쳤던 것으로 Data를 전송하는 측면에서 살펴보면 TCP/IP를 들 수 있다.

미국 정부에서 Internet의 중요성과 잠재력에 대해 인정하고 많은 기간 동안 Internet을 만들기 위한 기반 연구를 진행해 왔다. ARPANET Project에 의해 주도된 이들 연구는 Internet기술의 여러 가지 제반 기술과 개념들을 만들고 발전 시켜왔다. 이들 기술들 중에는 컴퓨터들이 서로 통신하는 것에 대한 세밀한 표준도 포함된다. 공식적으로 TCP/IP Internet Protocol Suite라고 불리고 일반적으로 TCP/IP라고 불리는 이 표준은 서로 연결된 어떠한 Network들의 집합들 사이에서라도 통신의 기본 프로토콜로써 사용된다. TCP/IP 기술은 가정과 대학 캠퍼스, 학교, 기업, 그리고 정부 연구기관 등이 서로 연결된 전세계적 규모의 인터넷을 구성하는 기반을 만들었다. NSF, DOE, DOD, HHS, NASA 등의 미국 연구 기관들이 인터넷을 창설하는데 참가하였고 자신들의 연구 사이트들을 연결하는데 TCP/IP를 사용하였다. ARPA/NSF 인터넷, TCP/IP 인터넷, 글로벌 인터넷으로도 불리기도 한다.

또 다른 측면으로 본다면, Network을 통해서 전달되는 Data를 이러한 서로 다른 환경의 시스템간에 전달하기 위하여 컴퓨터가 이해할 수 있는 Data File에 관련된 기술이 발전하게 된다.

우선, 컴퓨터가 이해할 수 있는 Data File의 종류는 두 가지가 있는데, 하나는 바이너리(Binary) 파일이고 다른 하나는 텍스트(Text) 파일이다.

바이너리 파일은 단순한 비트(1과 0)의 나열에 불과하다. 이 비트가 무엇을 의미하는지 알기 위해서는 해당 바이너리를 만든 Application을 사용해야만 알 수가 있다. 이유는 특정 바이너리 파일은 그 나름대로의 규칙을 가지고 있으며, 이런 규칙은 그 파일을 작성한 컴퓨터 프로그램에 의해서만 작성되고 읽혀 질 수 있기 때문이다.

바이너리 파일은 문서에 삽입된 기호들은 메타 데이터(Meta Data)라고 할 수 있는데, 이 메타 데이터가 서로 다른 종류의 파일들을 구별하는 기준이 된다. 즉 각각의 응용프로그램이 각기 다른 방법으로 제작되어 있기 때문에 다른 응용프로그램에서는 해당 파일을 사용할 수 가 없다.

바이너리 파일의 장점은 컴퓨터가 이해하기 쉽다는 데 있는데, 이는 컴퓨터가 훨씬 빨리 처리할 수 있고, 메타 데이터를 저장하기에 매우 유용하다는 것이다.

바이너리 파일의 단점은 "배타적"이라는 것이다.

즉 응용프로그램간의 상호 호환성이 없다는 것이다. 이는 서로 상이한 응용프로그램간의 문제뿐만 아니라, 같은 응용프로그램이라도 서로 상이한 플랫폼에서 운영될 경우 호환되지 않는다는 것이다.

또 다른 하나의 Data Format인 Text File도 사실 비트 정보의 나열에 불과하지만, 해당 비트 정보들은 표준화된 방법으로 문자표와 연결된 어떤 숫자로 구성할 수 있게 되어 있다. 여기엔 표준으로 작성되어 있다는 이유로, 여러 응용프로그램에서 읽혀 질 수 있으며, 여러 정보를 쉽게 공유할 수도 있게 된다. 초기 Internet은 거의 완벽하게 Text 기반으로 되어 있었다.

이는 Internet 초기에 커다란 장점으로 작용하였다. 결과적으로, Internet의 엄청난 성장을 가능하게 한 주 요인이라고 할 수 있다.

Text File의 단점은 다른 메타 데이터를 추가하기 어렵고, 용량이 너무 크다는 것이다. 이러한 이유로 Text File은 그림이나 서식과 같은 대용량의 데이터를 처리하지 않기 때문에 Text File을 열어보면 단순히 단어들만 볼 수 있다.

지금까지는 바이너리 파일과 Text 파일에 대해 살펴 보았는데, 여기서 생각할 수 있는 것이 바이너리 파일의 고효율성과 풍부한 정보 저장 능력과 Text 파일의 보편성을 함께 가진 형태를 생각할 수 있다.

이러한 방향의 노력으로 SGML(Standard Generalized Markup Language)을 만들게 되었다. SGML은 보편적인 Data 형식을 가지며, 메타 데이터를 저장하기 위해 Mark up 데이터를 사용할 수 있는 Text기반 언어이다. SGML은 또한 자기 기술적(Self-Describing)인 방식이다. 이러한 Mark Up언어는 계층적인 구조를 가지고 있으며, Data가 메모리에 저장되는 구조와 유사한 방식으로 Data를 처리할 수 있다. SGML은 다양한 형태의 전자문서들을 서로 상이한 시스템들 사이에서 Data의 손실 없이 효율적으로 전송하고, 저장하며, 또한 처리 자동화를 위한 ISO(International Organization for Standardization:국제표준화기구)의 문서처리표준의 하나이다.

문서의 구조를 정의할 수 있는 메타언어로서의 국제표준으로 1986년에 최초로 공개되었는데, 기능이 복잡한 단점이 있어 널리 쓰이지 못한다.

이는 컴퓨팅 환경의 폭 넓은 발전으로 여러 가지 표준을 정립하려 했던 1960년대에서 1980년대 사이에 공통점으로 너무 많은 내용을 포함하려는 경향이 있어서 실제로 적용되어지기가 어려웠던 단점을 가지고 있었다.

Web 문서를 만드는 언어로 가장 보편적인 HTML과 차세대 인터넷표준언어로 채택된 XML은 모두 SGML에 근거하여 만들어진 것이다.

2. 공통 Infrastructure의 문제

1960년대의 컴퓨팅 환경의 변화는 또 다른 문제를 가지고 왔다.

컴퓨터 하드웨어적인 문제와 Network이 점점 발전 하면서, 하드웨어와 Network 대역폭은 비용

이 점점 저렴해지기 시작했으며, 이로 인해 그러한 컴퓨팅 환경을 지원하는 소프트웨어의 작성이 점점 더 어렵고 힘들어지기 시작하였다.

특히, Internet 환경에서의 소프트웨어의 작성은 상당히 고급 기술을 필요로 하게 되었으며, 이러한 Internet Application에서의 문제점은 Business Logic의 문제가 아니었다. 어차피 Business Logic은 Desktop Application과 같다고 할 수 있다. 그러나 Internet은 개방적이고, 자율적이며, 서로 상이한 시스템이기 때문에 서로 이질적인 면을 가지고 있었다.

이러한 환경에서 돌아가는 Application을 개발하는 데는 새로운 문제가 발생하게 된다. 특히 Desktop Application에선 일반적으로 보안에 관련된 사항에 대해 신경 쓰지 않았다. 그러나, Internet Application에서는 컴퓨터의 모든 연산에 대해 보안 문제에 대한 필요성이 제기되었다.

이러한 보안 Code의 작성은 결코 쉬운 것이 아니었으며, 그 Test나 Debug, 지원과 유지에 관한 문제 또한 쉽지 않았다.

Internet 컴퓨팅 환경은 새로운 종류의 문제점을 발생시키게 되는데, 그러한 문제점은 Client가 의뢰하는 작업인 Business Logic과는 아무런 관계가 없었으며, 이러한 제반 환경을 제공하는 하부구조(InfraStructure)와 관계된 문제였다.

즉, 이러한 보안의 문제와 분산 컴퓨팅 환경에서 발생하는 문제점은 하부구조의 일반적인 분류에 속한다고 볼 수 있다. 실제 Project에서 Business Logic이 문제가 아니라, 이러한 환경을 지원하는 하부구조를 개발한다는 것은 너무나 비효율적이며, 또한 이러한 하부구조는 Microsoft사와 같이 하부구조를 개발하지 않는 이상 일반 개발자들은 하부구조를 알기가 쉽지 않다.

이러한 Code를 일반 개발자들이 직접 작성한다면 자기가 무엇을 하고 있는지도 모르고 프로그램을 작성하게 되거나, 개발 기간 내에 프로그램 개발을 끝내지 못하는 경우가 발생하게 될 것이다. 즉, Business Logic의 문제보다 하부구조의 문제가 Project를 어렵게 만들게 된다고 볼 수 있다.

Application 개발자들은 Internet 하부구조에 대해서 상당히 취약하다고 볼 수 있다. 만약 개발자들이 Internet 하부구조까지 개발하려고 한다면 개발 Project를 제대로 관리하기도 힘들게 된다. 또한 비용적인 면이나, 개발하는데 걸리는 시간적인 면을 고려해 볼 때 개발자가 직접 하부구조를 개발하는 것은 사실상 불가능하다.

개발자들은 사실 누군가 하부구조를 만들어 주고 자신은 그냥 그 하부구조를 이용하길 바란다.

그렇다면 Internet 같은 고도의 분산환경을 만들기 위해 필요한 것은 과연 무엇일까? 이러한 질문에 간단하게 답할 수는 없을 것이다. 분명 분산환경은 수많은 문제를 만들어 낸다.

첫째: Distributed 환경은 해당 환경에 맞는 Distributed 하부구조를 필요로 한다. 이러한 하부구조는 Distributed 환경을 사용 가능하게 만들기 위한 여러 서비스와 맞아야 한다. 예를 들어 제공되는 서비스가 적절한 사용자에게만 허용되어야 한다.

둘째: Distributed 환경은 Distributed Application의 생성을 받아들일 수 있어야 한다. 이 문제는 또 다른 문제를 가지게 되는데, 다양한 종류의 소프트웨어 간 통신을 어떻게 할 것이며, 결국 효과적인 분산환경은 수많은 Application, Computer, 사용자 등 전 세계적으로 보급되어진 것을 관리할 수 있는 능력을 말하는 것이다. 효율적인 분산환경을 만들기 위해서는 적절한 Distributed 서비스를 사용해야 하는데, Microsoft사는 자사의 OS인 Windows 2000에 많은 강력한 Dis-

tributed 서비스를 제공하고 있다.

Distributed 서비스란 앞서 언급했던 것처럼 Network에 관련된 여러 서비스와 Data Format에 관련된 여러 서비스로 나누어 볼 수가 있다.

우선 Network에 있어 여러 Distributed 서비스의 예를 든다면 TCP/IP를 들 수가 있다. TCP/IP는 종합적으로 모든 종류의 하부 Network와 연결된 컴퓨터 사이에서 Data를 전송하게 하는 신뢰할 수 있는 방법을 제공한다. 또한 TCP/IP는 이들을 이용해서 직접 서비스를 구축할 수도 있다.

지금까지 이야기 한대로 만약 이러한 프로토콜이 제공하는 서비스만 사용하여, 여러분들이 꼭 필요한 부분만으로 구성된 Distributed 환경을 만들 수도 있다. 하지만, 여러분들이 직접 전송 프로토콜에 대한 응용 프로그램을 만드는 것은 매우 힘든 일이다. 전송 프로토콜을 이용해서 직접 서비스를 구축하는 것 보다는 보편적인 Distributed 서비스를 이용하는 것이 바람직하다고 볼 수 있다.

보편적인 Distributed 서비스에 반드시 포함되어야 할 서비스로는

첫째: 하부구조 서비스 여기엔 Directory 서비스와 보안 서비스가 있다. Directory는 Distributed 환경에서 자원을 쉽게 찾을 수 있도록 하며, 보안 서비스는 정보와 기타 서비스에 대한 접근을 통제한다.

둘째: 응용프로그램 지원서비스 RPC를 사용하는 Distributed 응용 프로그램 구축 및 MSMQ (Message Queuing) 또는 HTTP 프로토콜을 이용한 Web 기반의 접근을 지원하고 보다 유연한 응용 프로그램을 구축하는데 보다 쉬운 방법을 제공한다.

이러한 필수적인 사항을 포함하고 있으며, 보편적인 Distributed 서비스를 제공하는 하부구조를 Microsoft사는 자사의 OS인 Windows 2000에 포함시켜 놓고 있다.

지금까지의 내용을 정리한다면 1960년대부터 컴퓨팅 환경은 하드웨어 적인 면과 소프트웨어 적인 면이 급속하게 발전하기 시작 하였으며, 또한 이들을 묶을 수 있는 Network에 관련된 부분도 발전하기 시작하였다.

Network의 발전은 서로 상이한 시스템간의 Data 교환에 관한 문제를 가져오게 되었고 하드웨어 적인 면과 소프트웨어 적인 면은 Application의 발전을 가져와 보다 많은 하부구조를 포함해야 하는 문제를 가져왔다. 또한 이러한 두 가지 내용으로 Desktop 내부에서만 움직이는 소프트웨어에서 이제는 Distributed Application을 구현해야 하는 어려움을 가져오게 되었다.

이러한 문제들의 해결책으로 Network에서는 TCP/IP라고 하는 표준을 정의하여 표준 Specification을 준수하여 Data를 전송하는 하부구조 서비스를 설계하고 개발하도록 하였으며, 서로 상이한 시스템간의 Data 교환에 관련하여서는 1980년대 후반부터 1990년대 중반까지 많은 발전을 하게 되었는데 이러한 내용은 별도로 이야기 하고자 한다.

Network에 관한 내용도 필수적인 Distributed 서비스를 포함하며 여러 가지 부가적인 서비스를 제공하는 보편적인 하부구조를 미리 제공하여 개발자들이 이를 이용하기만 하면 될 수 있도록 하듯이, Data 교환에 관한 문제도 하부구조 서비스로 제공 되게 된다.

여기서 이러한 하부구조 서비스들을 망라하는 것은 어려운 일이며, 여기서는 Microsoft사의 .NET을 중심으로 전개되어 지기 때문에 여기에 한정되어 진다.

Microsoft사는 개인용 컴퓨터의 발전에 소프트웨어 적인 면에서 막대한 영향을 남겼다. 처음

Windows Version 1, 2는 고전하였지만 개인용 컴퓨터가 AT 시대를 지나 386 컴퓨터가 개인의 책상에 올려지면서 Windows 3.x는 새로운 하드웨어 구조의 장점을 최대한 살린 Multi Tasking 시스템으로 자리를 굳히게 되었다.

이러한 컴퓨팅 업계 자체에도 1990년대에 들어와 Internet이라는 또 한번의 격변을 겪게 되는데,

현재 거의 모든 컴퓨터들이 어떠한 형태로든 Internet에 물려 있다고 해도 과언이 아니다. 그러나 이러한 변화의 초기에는 전자메일을 통해 편지를 주고받거나, 프린터나 하드디스크를 공유하는 정도의 수준이지 응용 프로그램 수준에서 서로 상호 작용하는 것은 아니라는 것이다. 예를 든다면, 메일을 보내는 하나의 Application은 메일이라는 형태의 Data 전송 Format으로 Data를 Format하여 시스템 하부구조를 이용해 Network에 Data를 주거나 받을 뿐이며, 실제로 해당 Data를 이용해 Application 자체가 연동되는 것은 아니라는 것이다.

그러나, Application 개발에 있어 바로 이어서 분산환경의 필요성이 대두되기 시작하였다, 분산환경은 흔히 Enterprise환경이다. 이를 지금까지 이야기 하는 맥락에서 구체적으로 다시 말하면 서로 다른 컴퓨터들에서 실행되는 서로 다른 프로그램들이 실제적인 상호 작용을 하는 환경이라고 할 수 있다. 개발자들은 Enterprise환경이라고 하는 새로운 길을 가게 되었다.

Enterprise Application이라는 말에는 3가지의 요점이 담겨 있다.

첫째: Enterprise Application은 덩치가 크다는 것이고

둘째: Enterprise Application은 분산되어 있다는 것이다.

따라서 한 개인이 Enterprise Application을 개발한다는 자체가 거의 불가능하게 되었다.

세 번째는 실패해서는 안 되는 임무를 수행한다는 것이다.

즉, Enterprise Application은 실패해서는 안 되는 덩치가 크고 분산 되어 있는 응용 프로그램이라는 것이다. 이전의 Enterprise Application 개발은 기간도 길고, 어려웠으며, 비용도 많이 들었을 뿐만 아니라 성공을 확실하게 보장하지도 못했다. 이러한 환경에서 가장 어려웠던 것은 Business Logic을 Enterprise 수준으로 끌어 올리는데 필요한 기반구조를 매번 다시 만들었다는 것이다.

예로 들면, 보안적인 면을 들 수가 있는데 Desktop Application과는 달리 Enterprise Application은 대부분이 높은 수준의 보안을 요구하게 된다. 그렇다면 Enterprise Application 개발자라면 이러한 보안 내용을 적용 시켜야 할 것이고, 사용자를 인증하고, 사용자에게 특정 자원이나 기능에 대한 허용이나 금지를 설정 해야 하며, 사용자의 보안 권한을 설정하거나 삭제하는 시스템을 위한 Code를 직접 작성하고 이를 Test하며, Debugging하고 설치하고 유지/보수 해야 한다.

또한 이러한 보안 시스템뿐만 아니라, 자원에 대한 Pooling, 공유 자원에 대한 직렬화, Transaction에 대한 동기화 등등, Enterprise Application에 필요한 기반 서비스들은 엄청나게 많으며, 이러한 기반구조를 직접 작성한다는 것은 엄청난 개발기간과 비용을 투자해야 하는 것이다.

그러나, 사실 Enterprise Application의 기반구조는 서로가 비슷하다는 것을 알게 되었다. 예로 은행용 사용자 인증에 관한 시스템이나 병원용 사용자 인증 시스템이 달라야 할 이유는 없다. 따라서 한 번 만들어 놓고 계속 재사용할 수도 있다. Microsoft사는 이러한 공통적인 기반구조를 일단 만들어 놓고 필요할 때 그냥 상속해서 쓸 수 있도록 제공하고, 일반 개발자들은 남은 시간에 Busi-

ness Logic을 정교하게 만드는데 치중하게 하는 방법을 생각하게 되었다.

이러한 대안으로 1990년대 초에 Microsoft사는 COM과 MSMQ라는 해결책을 제시하게 되었다.

여기서는 COM과 MSMQ에 대한 세부적인 기술내용을 다룰 수 없으며, COM 이후의 COM+에 대해서 잠시 살펴볼 것이다.

Enterprise Application의 이러한 복잡한 요구조건을 만족하는 하부구조를 범용적인 기반구조 만들어 제공하게 되었다. 특히 3계층구조에서 Middle-tier, 즉 Business Layer의 기반 구조 문제들에 대한 해결책을 제공한다고 할 수 있다. 즉 다시 말해 COM+는 "Enterprise Application개발을 위한 Microsoft사의 조립식 Toolkits"이라고 할 수 있다.

COM+에는 다음과 같은 새롭고 다양한 Run-time 서비스들이 포함되어 있다.

- Transaction Service: 분산 시스템에서 통신 장애나 컴퓨터의 오류가 있더라도 Data의 무결성을 보장하기 위한 메커니즘이다.
- Security Service: 너무도 당연한 서비스이다.
- Synchronization Service: 분산 시스템 구조에서 여러 개의 Thread들이 동시에 어떤 컴포넌트에 접근할 때 생기는 위험을 방지하고자 하는 메커니즘이다.
- Queued Component: COM Client들이 서버에 연결되지 않은 상황에서도 서버에 있는 COM 객체에 대한 호출을 가능하게 하는 메커니즘이다.
- Event Service: Enterprise Application의 구성부분 중에서 갱신된 정보를 제공하는 프로그램을 주고 받는 것을 쉽게 한 메커니즘이다.
- In-Memory DataBase: Middle-tier에서 컴퓨터 상에 있는 Backend Table의 자동적인 메모리 Caching 기능을 제공하는 서비스이다.
- Load Balancing: Client의 객체 생성 요청을 자동적으로 서버들에게 균등하게 분담하는 서비스이다.

이들은 Enterprise Application 개발에 있어 고유한 여러 문제들을 쉽게 해결할 수 있도록 도와주는 역할을 한다.

Microsoft사는 이러한 여러 기반 서비스를 운영체제로부터 상속 받을 수 있게 하였다.

3. 패러다임 변화를 위한 기술적 회귀

그러나, 1990년대 후반으로 들어 오면서 한가지 문제점이 발생하게 되었다. 각각의 벤더들은 앞서 이야기한 문제들을 해결하기 위해 많은 노력을 하였지만, 해당 시스템들이 가지고 있는 한계 때문에 결국, 자신의 시스템 안에서 이러한 문제를 해결하려고 하였다.

따라서, 이러한 문제 해결 방법론은 결국 Internet의 광범위한 확산과 더불어 한계에 다다르기 시작 하였다. 지금부터는 시스템에 구애 받지 않고 Web을 통해서 Enterprise Application의 상호 연동을 위한 준비를 하게 되었다.

Microsoft사는 Windows 2000 DNA를 통해서 이러한 문제를 극복하려고 하였다. 그러나, 기존의 시스템 한계를 그대로 유지하면서 이러한 문제를 해결한다는 것은 어려운 문제였다.

Microsoft사는 COM의 Interface를 통해서 이러한 문제를 극복하려고 하였다. 그러나 시스템 하부구조에서 다루어지는 Data의 Type에 관한 문제로 많은 어려움을 겪게 되었다.

이러한 이유로 Microsoft사는 기존의 Enterprise Application을 충분히 수용할 수 있도록 하기 위해서 COM이 가지는 기능을 그대로 유지하

면서 다른 Enterprise Application이 컴포넌트를 활용해서 완전한 Enterprise Application의 상호 운영성을 추구하도록 하려 했다. 결국 Data Type의 표준화 작업과 기존의 COM이 가지고 있는 기능성을 유지하기 위해서 그 안의 구현 내용을 그대로 두고 어느 Enterprise Application에서나 접근 가능 하도록 하는 새로운 Interface를 필요로 하게 되었다.

그 대안으로 Microsoft사는 기능의 구현은 없고 Data를 표현하는 면에서 Data의 구조를 규격화 할 수 있고, Data Type의 표준을 위해 이미 서로 다른 Data Type을 표현 할 수 있는 Data의 어휘에 관련된 내용을 수용하는 자기 기술적인 면을 필요로 하게 되었다. 양쪽에서 Data Type의 표준을 마련해서 그 마련한 Specification에 준해서 Data의 구조와 어휘를 해당 Specification에 근거해서 바꿀 수 있다면 하고자 하는 목적을 달성할 수 있을 것이다.

이러한 대안으로 선택할 수 있었던 것이 Mark Up 언어에서 찾게 되었다.

그러나, SGML은 그 당시 표준화 물결과 마찬가지로 너무나 방대하고, 그 Specification을 준수하기가 너무 어려웠기 때문에 어느 정도의 필요한 부분만 기술하는 축소화 작업이 필요하게 되었다. 그러한 맥락에서 Microsoft사 뿐만 아니라 다른 벤더에서도 같은 필요성을 느끼고 있었기 때문에 이러한 공동의 목표를 중심으로 SGML에서 XML이 나오게 된다.

이러한 현상은 또 다른 분산환경인 Directory Service에서도 마찬가지로 나타난다. 이전의 Network Directory Service에 대한 표준인 X.400 계열이 너무 많은 내용을 담고 있어서 서드파티에서 이를 구현한다는 것이 매우 어려운 문제였다.

LDAP(Lightweight Directory Access Protocol)이 이러한 맥락에서 나왔다고 볼 수 있다. Microsoft사는 이러한 표준을 수용하면서 부가적인 내용을 담고 있는 Directory Service로 Active Directory Service를 운영체제에서 분산 서비스로 제공한다.

결론적으로, Microsoft사는 향후의 Internet 분산환경을 목표로 기존의 COM에서 Interface를 대체하는 언어로 XML을 선택하게 된 것이다. XML은 Web환경에서 작동하는 Mark Up언어이며, HTML과는 다르게 Data의 구조와 어휘를 자기 기술적으로 나타낼 수 있으며, XML로 어떠한 Business Logic의 구현이 없다는 면에서 목적과 일치하게 된 것이다.

이런 면에서 일반적인 기술적 회귀라기 보다는 좀 더 향상된 방향으로의 수용이라 할 수 있을 것이다.

4. 그렇다면, .NET이란 무엇인가?

Microsoft .NET은 Internet Application을 개발할 때 지금까지 이야기했던 Internet 분산환경의 하부구조를 지원하기 위한 Microsoft사의 Internet Infrastructure(인터넷 인프라)를 말한다. 개발자들은 이를 이용해서 앞에서 대두 되었던 여러 가지 공통의 문제를 해결할 수가 있다.

.NET은 .NET Framework이라는 새로운 Runtime환경을 제공한다. .NET Framework은 개발자가 양질의 견고한 Code를 작성할 수 있도록 만들어 줄 뿐만 아니라, 개발자가 해당 Code를 보다 효율적으로 관리하고, 배치하며, 수정할 수 있도록 만들어 준다.

개발자가 작성하는 프로그램과 컴포넌트는 이러한 Runtime 환경에서 수행 된다. .NET Framework은 자동 메모리 관리와 같은 기능을 제공하기도 하며, 모든 시스템에 쉽게 접근할 수 있도록 해주며, Web을 통해서 Database에 접속할 수 있

게 해주기도 하며, 한번 작성 해 놓은 Code를 재사용할 수 있도록 하는 새로운 메커니즘을 제공해 준다.

이런 방법은 이전의 COM보다 훨씬 더 강력하고 유연성이 좋아지며, 사용하기도 훨씬 쉬워 진다고 볼 수 있다. .NET Framework은 레지스트리(Registry) 설정이 필요 없기 때문에 프로그램의 배치도 훨씬 쉬워지며, 시스템 Level에서의 표준화된 버전관리를 해 주기도 한다.

.NET은 또한 ASP.NET이라는 새로운 Web Application개발 프로그래밍 모델을 제공한다. 이 모델은 HTML기반의 Web Page를 만들어 주는 Code를 쉽게 작성할 수 있게 해주며, ASP.NET은 Code를 작성하고 해당 Code를 해당 Web Page의 요청에 결합시키는 방법을 제공해 준다. 이러한 방법은 프로그램 언어와는 독립적으로 구성되며, Web Form은 Web Control과 상호 작용하는 이벤트 기반의 프로그래밍 모델이다. ASP.NET은 기존의 ASP보다 훨씬 향상된 성능과 기능성을 가지고 있으며, 더 견고하게 작성된다.

.NET은 또한 .NET Web Service라는 방법을 통해 Internet을 통해 서버가 자신의 Method를 노출시켜 어떠한 장치의 어떠한 플랫폼에서도 호출할 수 있도록 하여, 자신의 기능을 직접적으로 상호 연동하면서 사용할 수 있도록 한다.

.NET은 또한 Application 개발에 있어서 거의 대부분이라고 할 수 있는 Database에 대한 접속을 .NET Framework환경에서 제공하여 훨씬 효율적으로 활용할 수 있게 한다.

현재 Microsoft사 에서는 이러한 .NET Framework의 기본적인 하부구조 서비스뿐만 아니라 **"Microsoft Application Blocks for .NET"**이라는 이름으로 개발자들이 보다 효율적으로 프로그램을 작성할 수 있도록 Building Block Service를 또한 제공해 준다.

해당 Block Service에서는

• **Data Access Application Block:** Data Access Application Block은 저장 프로시저를 호출하고 SQL Server 데이터베이스에 대한 SQL 텍스트 명령을 발급하는 것을 도와주는 최적화된 데이터 액세스 코드가 포함된 .NET 구성 요소이다. SqlDataReader, DataSet 및 XmlReader 개체를 반환한다. .NET 응용 프로그램에서 Data Access Application Block을 빌딩 블록으로 사용하면 사용자 지정 코드를 만들고 테스트하고 유지 관리하는 양을 줄일 수 있다.

특히, Data Access Application Block을 사용하면 다음을 실행하는 데 도움이 된다.

- 저장 프로시저 또는 SQL 텍스트 명령 호출
- 매개 변수 정보 지정
- SqlDataReader, DataSet 또는 XmlReader 개체 반환

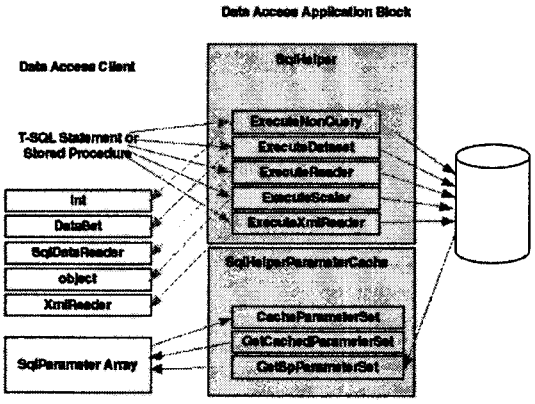


그림 1. Data Access Application Block

• **Exception Management Application Block:**

Exception Management Application Block은 .NET Application에서 Building Block으로 쉽게 사용할 수가 있다. Error Handling Code를 작성하

는데 많은 시간을 소비하지 않도록 해주며, Debugging을 훨씬 더 강력하게 해줄 수 있다.

특히 Exception Management Application Block을 사용하면 다음을 실행하는데 도움을 줄 수 있다.

- Exception 관리의 효율적이고 일관성 있는 방법
- Business logic과 관련된 Code로부터 분리된 exception관리 Code작성
- Exception을 Log하고 Handle하는 사용자 작성 Code의 최소화

이러한 내용들을 담고 있는 **"Microsoft. ApplicationBlocks"** Service들이 점점 더 많은 지원이 이루어 질 예정이다.

5. 서비스로서의 소프트웨어

.NET의 궁극적인 목적은 소프트웨어를 하나의 서비스로 보는 것이다. 예전의 소프트웨어는 단순히 디스켓이나 CD-ROM에 담아서 적당한 가격에 파는 것이라고 생각했지만, Internet이 발전하면서 소프트웨어는 단순하게 독립적인 컴퓨터 안에서 실행 되는 것이 아니라 항상 Internet과 연결되어 있어서 소프트웨어를 통해 서비스를 받는 것이 되었다. 이러한 예는 지금도 많이 볼 수 있다. Internet을 통해 소프트웨어가 설치되고, 웹 기반에서 편집한 파일이 물리적으로 어디에 저장되어 있는지 알지 못해도 Internet에 연결되어 있다면 언제든지 이용할 수가 있다. 이외에도 많은 예를 볼 수가 있다. 심지어 앞으로 Internet이 연결되어 있지 않다면 소프트웨어를 이용하지 못할 수도 있을 것이다. 물론 지금 현재 Internet을 항상 이용할 수는 없겠지만 향후 Network과 하드웨어의 발전을 통해 저렴한 비용으로 어떠한 장비를 통해서도 항상 Internet과 연결되어 있게 될 것이다. .NET의 목표는 그러한 환경을 염두에 두고 있다. 서비스로서의 소프트웨어에 대한 예로 그런 환경 속에서 자그마한 계산기 프로그램도 Internet을 통해 연산에 관한 서비스를 제공 받는 경량의 프로그램만 설치가 되고 실제 연산은 어디선가 웹을 통해 서비스로 제공 될 것이다.

Microsoft .NET은 이러한 변화를 인식하고 모든 역량을 Internet이라는 환경에 맞추어 서비스로서의 소프트웨어라는 비전을 제시하는 것이다.

그리고 소프트웨어 개발자들에게 서비스로서의 소프트웨어를 보다 쉽게 만들 수 있는 기반 환경을 제공하게 된다.

개발자의 입장에서 응용프로그램을 개발하는 환경은 여전히 폐쇄적이며, 특히 웹 응용프로그램을 개발하는 경우에는 기존의 윈도우 응용 프로그램을 개발하는 것보다 언어적인 면과 개발환경적인 면에서 매우 열악하다고 볼 수 있다.

그러나, .NET은 그 하부구조를 Internet을 염두에 두고 설계되었기 때문에 서비스로서의 소프트웨어를 개발하고자 하는 면에서 쉽고 강력한 접근성을 제공해주게 된다.

.NET은 Internet이라는 하부구조를 XML을 통해서 쉽게 서비스할 수 있고, 특정언어나 특정 시스템에 독립적이고 응용 프로그램의 특성에 관계없이 공통된 개발방법으로 소프트웨어를 개발할 수 있게 해준다.

6. .NET에서의 XML

Web을 통한 소프트웨어의 서비스, 즉 WebService라는 강력하고 개방적인 분산 응용프로그램을 지원하기 위해 .NET은 XML을 선택했다. XML WebService는 운영체제나 프로그래밍 언어에 상관없이 모든 응용 프로그램들이 Internet상에서 서로 통신하며 데이터를 공유하도

록 한다. 이러한 많은 일들을 가능하게 해주는 것은 XML이 표준이기 때문이다.

.NET 설계자는 XML의 통합이 성공의 관건이 된다는 것을 알았고, 그 결과 XML은 .NET Platform에 직접적으로 통합이 되었다. XML의 통합은 매우 광범위해서 Data처리에서 원격 시스템의 사용을 위한 객체의 나열까지 모든 것을 포함하고 있다.

.NET이 XML과 결합되면서 더 유연하고, 더 효율적이며, 더 다루기 쉬운 프로그램을 만들 수 있게 해준다. 특히 ASP.NET에서 표현 코드와 Business Logic이 따로 분리되어 더 많은 장점을 가져다 준다는 것을 예로 들 수 있다.

이러한 유연함은 프로그램에서 가장 핵심적인 부분이라고 할 수 있는 Data를 다루는 부분을 보면 잘 알 수 있다.

.NET은 어떠한 Data Source에서도 Data를 받을 수 있게 해주는 유연함을 가지고 있다. XML은 DataSet이라고 하는 새로운 Data 객체 타입의 기초가 되며, .NET 플랫폼 곳곳에서 사용된다. .NET 개발자에게 Data 교환에 있어서 상당한 이점을 제공해 준다. Application간의 Data 교환뿐만 아니라 방화벽이나 거리상의 문제로 인해 분리된 다른 객체들 사이에서의 Data 교환은 모두 포함한다.

특히, XML이 브라우저와 무선 장비들의 지원이 늘어남에 따라 XML과 ASP.NET을 사용한 웹 Application 개발은 최종 사용자에게 좀 더 쉽고 더 적은 코드로 Data를 표현 하게 해준다.

또한 이종의 장비들을 겨냥한 페이지들은 XML 기반의 저장소를 사용하여 더 효율적으로 이용될 수 있을 것이고, 이 저장소는 ASP.NET과 XSLT를 이용해서 Client의 장비에 따라 변환될 수 있을 것이다. 이렇게 하면 Client는 더 빠른 속도로 서비스를 제공 받을 수 있는데 이는 필터링과 분류 또는 다른 것들이 서버로 다시 갔다 올 필요가 없기 때문이다.

WebService의 증가는 세계 곳곳의 벤더들이 발표하고 있는 XML기반의 Data들을 사용할 수 있게 해줄 것이고, 또한 개발자들이 자신의 ASP.NET 프로그램에 삽입할 수 있도록 해줄 것이다. ASP.NET 개발자들은 필요한 서비스를 제공하기 위해 XML을 폭 넓게 사용할 수 있을 것이다.

따라서, .NET 개발자로서 효율적인 .NET/XML 프로그래밍을 위해서 반드시 XML의 기본을 이해하여야 할 것이다.

7. 향후 .NET의 방향

지금까지 언급한대로 .NET은 그 하부구조 자체에 대한 설계조차 이전과는 다르게 설계되어졌다. 그 하부구조의 Data의 흐름은 XML이 담당할 것이며, 이로 인해 지금까지의 많은 제약 조건들을 극복하고 진정한 소프트웨어의 통합을 이루며, 나아가 각종 디바이스의 통합까지 이루어 질 게 명백하다.

Microsoft .NET은 Microsoft XML 웹 서비스 플랫폼이다. XML 웹 서비스는 운영 체제, 장치 또는 프로그래밍 언어에 관계 없이 인터넷을 통해 응용 프로그램에서 데이터를 통신하고 공유할 수 있게 해준다. Microsoft .NET 플랫폼은 개발자에게 XML 웹 서비스를 만들고 이들을 서로 통합하는 데 필요한 기술을 제공하고, 일반 사용자에게는 완벽하고 안전한 환경을 제공하며, XML 웹 서비스의 개발, 관리, 사용, 경험 등 모든 측면을 제공한다. XML 웹 서비스는 현재 이미 사용 중인 Microsoft 응용 프로그램, 도구 및 서버의 일부분이 될 것이며 모든 비즈니스 요구를 만족할 수 있는 새로운 제품에 포함될 것이다. 현재 Microsoft가 .NET 플랫폼에 구축하는 영역을 보면 다

섯 가지 영역으로 나누어 볼 수 있다. 다섯 가지 영역은 도구, 서버, XML 웹 서비스, Client 및 .NET 환경이다.

예로 Client 영역을 보면 Client로는 PC, 랩톱, 워크스테이션, 전화, 휴대용 컴퓨터, 태블릿 PC, 게임 콘솔 및 기타 스마트 장치를 들 수 있다. 이들 장치가 "스마트" 할 수 있는 이유는 XML 웹 서비스에 액세스할 수 있다는 점이다. 스마트 클라이언트는 XML 웹 서비스를 지원하는 소프트웨어를 사용하며 사용자는 위치, 사용하는 Client 의 종류와 수에 상관 없이 데이터에 액세스할 수 있게 된다. Microsoft가 제공할 .NET 클라이언트 소프트웨어에는 Windows CE, Windows Embedded, Windows 2000 및 Windows XP 등이 있다. 이 소프트웨어를 사용하여 PC, 랩톱, 워크스테이션, 전화, 휴대용 컴퓨터, 태블릿 PC, 및 XBox™ 게임 콘솔 등을 사용할 수 있다.

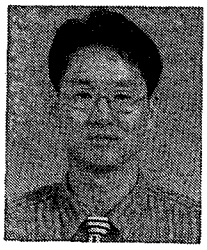
Microsoft .NET은 컴퓨팅 장치에 대한 사고 방식과 컴퓨팅 장치의 사용 방법을 근본적으로 변화시킬 것이다. 현재 컴퓨팅에 대한 일반적인 개념은 서버와 클라이언트 데스크톱 두 가지이다. Microsoft .NET은 현재의 서버/클라이언트 컴퓨팅 모델을 다양한 분산 컴퓨팅 패러다임인 느슨한 결합(loosely-coupled) 서비스로 확장시킨다. 기존의 클라이언트와 서버의 구분은 사라지고 서버, PC, 핸드헤드 장치, 스마트 장치 등 여러 장치 중 가장 적합한 곳에서 프로세스가 이루어지게 된다. 이것이 차세대 스마트 장치에 대한 새로운 스마트 컴퓨팅 개념이다.

향후 XML Web Service는 이러한 기반 구조를 통해서 우리가 SF영화에서 꿈꾸어 오던 모든 분야에서의 통합을 이루는 그런 세상을 여는 새로운 시작이 되는 기술이 될 것이다.

또한 .NET은 이러한 기술 동향을 뒷받침하는 인프라를 제공하게 될 것이다.

차 재 호

- 1993년 울산대학교 행정학과 학사 졸업
- 전 MBC, 경실련, 매경디지탈캠퍼스 전임강사
- 현재 : ㈜필라넷 수석컨설턴트, ㈜웹타임 전임강사
- 자격사항
  Microsoft Certified Trainer (MCT)
  Microsoft Certified System Engineer (MCSE)
  Microsoft Certified Solution Developer (MCSD)
  Microsoft Certified Database Administrator (MCDBA)
  Microsoft Certified Application Developer (MCAD)



권 순 각

- 1990년 2월 경북대학교 전자공학과 졸업(공학사)
- 1992년 2월 한국과학기술원 전기 및 전자공학과 졸업 (공학석사)
- 1998년 2월 한국과학기술원 전기 및 전자공학과 졸업 (공학박사)
- 1998년 3월~1998년 8월 전자통신연구원 선임연구원
- 1998년 9월~2001년 2월 기술신용보증기금 기술평가센터 차장
- 2001년 3월~현재 동의대학교 소프트웨어공학과 교수
- 관심분야 : 영상부호화 및 전송기법, 영상신호처리