ADO.NET 기술

류경석* · 김치용**

ADO.NET은 .NET Application에 저장된 Data를 사용할 수 있도록 도와주는 Microsoft .NET Framework에 포함된 라이브러리의 집합이다. ADO.NET 라이브러리는 데이터 소스에 연결하고, 쿼리를 전송하고, 결과를 처리하기 위한 클래스를 포함하고 있다. 또한 Disconnected 환경에서 견고하고, 계층적인 데이터 Cache를 이용할 수도 있다. Disconnected 상태의 객체에서 핵심이 되는 DataSet을 사용하여 계층적 데이터를 정렬, 검색, 필터, 지연변경 저장, 탐색 등을 할 수 있다. 또한 DataSet은 전통적인 데이터 액세스와 XML 개발간의 차이를 해결하는 다리 역할을 하는 다수의 기능을 포함하고 있다. 개발자는 XML 데이터를 전통적인 데이터 액세스 인터페이스를 통해서 작업할 수도 있고, 그 역 방향으로도 가능하다.

Microsoft사는 왜 이전의 ADO가 있는데 새로운 기술을 만들었는가? ADO는 지난 몇 년간 많은 개발자들에게 잘 사용되어 왔다. 그러나 좀 더 강력한 Application 개발을 필요로 하는 개발자들에게 핵심적인 기능이 부족하였었다. 예를 들어서 점점 더 XML 데이터로 작업하려는 개발자가 늘어 나면서 최근의 ADO 2.0 버전에서부터 XML 지원 기능이 포함되었으며, ADO 2.5 버전에서 XML 데이터를 스트림으로 처리할 수 있었지만 ADO는 XML 데이터 자체를 처리하기 위해 만들어진 것이 아니었기 때문에 ADO는 실제 데이터와 해당 스키마 정보를 구분할 수 없었다. ADO.NET은 XML 데이터를 염두에 두고 설계되었다. 따라서 ADO.NET은 실 데이터와 해당 Schema를 구분할 수 있다. ADO 커서엔진은 ADO Recordset을 Application의 다른 계층으로 넘겨 줄 수 있지만 여러 Recordset 객체의 내용을 결합할 수는 없다. 그러나, ADO.NET은 DataSet에서 여러 Table간의 내용을 결합하여 사용할 수 있다.

*다우 교육센터 .NET Framework 개발강사, 정보통신부 주관 .NET개발자 과정 강사
**동서대학교 디지털디자인학부 교수

1. 문제의 배경

단일 Desktop 프로그램은 사용자의 로컬 하드디스크에 있는 문서만을 다룬다. 그러나, 분산 프로그램은 모두 원격지에 저장되어 있는 데이터를 액세스 한다. Internet의 성장과 함께 Internet을 통해 데이터를 공유하려는 필요성이 대두 되었으며, 향후에는 모든 Internet Application이 원격지 데이터 저장소를 액세스하게 될 것이다.

일단 대부분의 Internet Application이 액세스하는 데이터는 실제 저장공간이 어디인지 모르는 경우도 있으며, Desktop Application이 액세스 하려고 하는 데이터의 특성과 근본적으로 다르다는

것을 알았다. 그렇다면 Internet Application을 작성할 때 새로운 디자인 문제점이 발생할 것이라는 것을 쉽게 예상할 수 있다. 우리가 보고 있거나, 변경하고자 하는 데이터는 서로 다른 장소에 위치하고 있으며, 서로 다른 형태의 컨테이너에 들어있다. 이렇게 서로 다른 데이터의 소스가 많아질수록 서로 다른 소스에 액세스하는 Client Application을 작성하는 작업은 데이터 소스 자체가 매우 동적이기 때문에 Client Application에 이를 알려주는 일이 매우 어려울 것이다. 또한 이러한 다양한 데이터 저장소에 대한 서로 다른 프로그래밍 모델을 익히는데 많은 시간을 소모할 수 밖에 없다. 특히나 이러한 문제는 소규모의 데이터 서비스 공급 업체에게 매우 불리한 상황이 될 수도 있다. 이제는 Client의 입장에서 보면 각 데이터가 어디에 위치하고 있던 간에 모두 동일하게 보이도록 해야 한다. 다시 말해서 모든 데이터에 액세스하기 위한 하나의 기본적인 프로그래밍 모델이 필요하다는 것이다. 인터넷 환경은 비결정적이고, 이질적인 성격을 가지고 있기 때문에 인터넷 데이터 액세스 프로그램은 매우 어렵다. 예를 들어서 Desktop 프로그램은 데이터베이스에 접속해서 사용자의 작업 세션 동안 그 연결을 유지 시키는 코드를 작성하는 것은 매우 쉽다. 그러나 인터넷에서는 서버에 많은 접속자가 동시에 접속을 하며, 서버는 각 사용자의 연결을 모두 유지시켜야 하는 부담을 가지고 있으며 다양한 응답시간을 계산하여야 하며 서버의 리소스를 너무 오래 묶어두지 않도록 해야만 한다. 또한 XML과도 잘 작동하는 데이터 액세스 전략이 필요하다.

마지막으로, 새로운 전략은 기존의 코드나 데이터와의 호환성을 고려 해야 한다. 만약 새로운 전략이 무엇이든 기존의 호환성이 없다면 실패하게 될 것이다.

2. 해결방안으로의 Solution Architecture

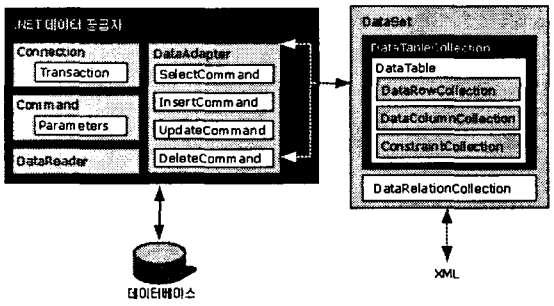
단일 프로그램에서 일반적인 데이터 액세스의 문제를 해결하기 위해 Microsoft에서 처음 시도한 것은 1995년에 출시된 OLE DB였다. OLE DB에서 모든 Data Provider는 Client가 Data Provider의 내부 구현을 몰라도 외부 액세스를 가능하게 해 주는 인터페이스 집합을 구현하였다. 그러나 OLE DB는 Client가 프로그램을 작성하기에 너무 어려웠다. Microsoft는 ADO라는 모델을 출시했다. ADO는 데이터 액세스 코드를 너무나 쉽게 작성할 수 있도록 해주었다. Client쪽 인터페이스는 프로그램 하기 매우 쉬웠고 서버 쪽에서는 OLE DB를 통해 프로바이더와 작업을 하였다. ADO는 3 계층 시스템의 중간 계층에서 매우 잘 작동 하였다.

그러나, 인터넷이 활성화 되면서 ADO는 인터넷으로 확장하는 데 문제가 있었다. 비 Microsoft 시스템과 쉽게 작동할 수 없었으며, 방화벽을 통과하는데도 문제가 있었다. ADO는 제한된 양의 비 연결 연산(Disconnected operation)을 지원하기도 하지만, 원래 연결 연산에 최적화되어 설계된 것이기 때문에 인터넷 세상에서 프로그래머의 요구사항을 모두 수용할 수가 없었다. Microsoft .NET은 ADO.NET을 포함하고 있다. ADO.NET은 일반적인 데이터 액세스와 프로그래밍을 쉽도록 하기 위해 COM 기반의 ADO를 .NET으로 바꿔주는 새로운 형태의 아키텍처이다. 자신의 데이터를 ADO.NET을 통해 .NET Client가 사용할 수 있도록 하고자 하는 Data Provider는 Data Provider와 .NET Client를 연결시켜주는 .NET 표준객체를 구현해야 한다. 이러한 객체로는 Connection, DataAdapter, Command, DataReader가 있다. 이들 객체들은 개발자가 데

이터 소스에 최적화된 객체를 구현할 수 있게 해 준다. 이러한 클래스는 .NET CLR의 일부분으로 포함되어 있다. ADO.NET은 DataSet 객체의 형태로 실제 데이터를 제공한다. DataSet은 쿼리를 실행 결과로 생겨나는 데이터의 집합을 표현하는 .NET 클래스이다. 해당 클래스는 내부에 Table을 가지고 있으며 Table의 열과 행에 액세스 할 수 있도록 메소드를 제공한다. 또한 내부 구조를 설명하는 스키마도 갖고 있다. DataSet 객체는 UnTyped가 될 수도 있는데, 이 경우에는 요구하는 항목의 이름을 코드화된 문자열의 형태로 요청한다. 또한 Typed DataSet 객체를 생성할 수도 있는데 이런 경우 각 필드와 묶여 있는 멤버 변수를 갖고 있으며, 좋은 코드를 작성하기가 쉽다. 두 타입의 DataSet 객체는 자신을 XML로 직렬화하는 방법을 알고 있으므로 프로세스나 컴퓨터의 경계를 넘어 데이터를 전달할 수가 있다. 프로그래머가 DataSet을 가지고 하는 주된 작업은 사용자에게 그 데이터의 내용을 보여주는 것이다. 윈도우 폼과 웹 폼은 DataSet 객체를 가져와 사용자에게 그 내용을 보여주는 컨트롤을 포함하고 있다.

이러한 새로운 아키텍처를 그림으로 살펴보면

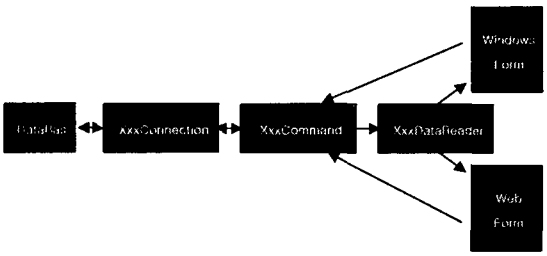
데이터의 처리는 보통 Connected 환경의 2계층 모델로 구현되지만, 데이터 처리 자체가 점차 여러 계층의 아키텍처를 사용하게 됨에 따라 프로그래머는 Disconnected환경의 데이터 연결 방법으로 전환하여 응용프로그램에 좀 더 뛰어난 확장성을 제공하게 된다.



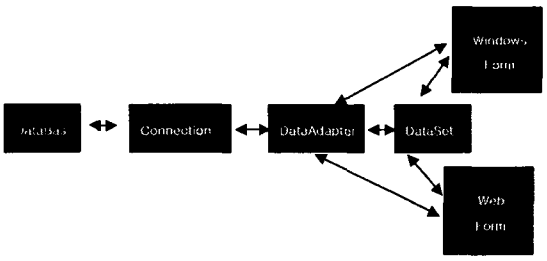
3. Disconnected에 관해

ADO.NET은 크게 두 가지의 형태로 나누어 볼 수 있다. 하나는 Connected환경을 말하고 다른 하나는 Disconnected 환경을 말한다.

Connected 환경일 경우 아키텍처는 아래의 그림과 같다.



Connected기반의 DataReader를 사용한 데이터 접근방법



Disconnected 기반의 DataAdapter와 DataSet을 사용한 데이터 접근방법

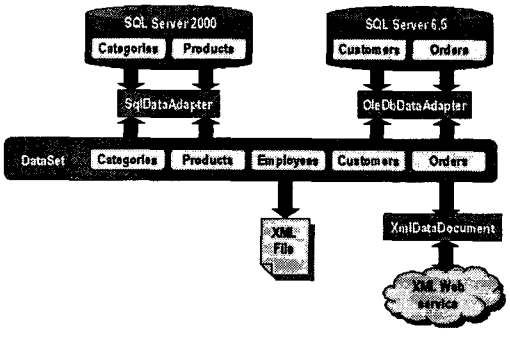
Connected 기반과 Disconnected 기반으로 나누어 보면 사용되는 객체가 다르다는 것을 알 수가 있다.

Connected 기반일 경우 SqlDataReader와 OleDbDataReader 객체는 각각 SqlConnection과 OleDbConnection 객체를 사용해서 데이터베이

스에 직접 접근한다. 즉 이전의 ADO에서 데이터를 얻는 방식과 유사하다. 이 객체들은 기본 커서 타입인 Read-Only, Forward-Only 이고, 데이터의 위치를 탐색할 수 없다. 그러나, 데이터를 가져오는 것은 매우 빠르다. 예를 들어 데이터를 한번 가지고 와서 읽기만 할 목적이라면 이 객체를 사용하는 것이 바람직하다.

Disconnected 기반의 데이터베이스 접근은 ADO.NET의 데이터 핸들링을 지원하는 핵심 객체가 바로 SqlDataAdpater, OleDbDataAdpater 객체이다. 이 객체들은 실질적으로 데이터베이스에 연결을 생성하고 SQL명령을 실행해서 Disconnected XML을 지원하는 DataSet 객체로 넘겨 주거나 DataSet 객체에 있는 변경된 데이터를 다시 데이터베이스에 넘겨 주는 역할을 한다. 특히 SqlConnection, OleDbConnection 객체를 SqlDataAdpater, OleDbDataAdpater 객체의 Command관련 프로퍼티에 지정만 하면 더 이상의 연결 관리가 필요 없어진다. 즉 SqlDataAdpater, OleDbDataAdpater 객체를 가지고 데이터베이스의 조회, 삽입, 갱신, 삭제 명령을 하면 자동으로 내부에서 데이터베이스와 연결을 열고 닫는 작업을 해준다. 이러한 Disconnected 기반의 데이터베이스 접근방법의 아키텍처를 살펴보면 아래의 그림과 같다.

Disconnected Architecture



Disconnected 기반의 데이터 연결방식일 때 DataSet은 응용 프로그램에서 사용할 데이터를 저장하는 컨테이너 혹은 캐시이다. 데이터 집합은 ADO.NET 아키텍처에서 가장 기본적인 요소이며 확장성과 더불어 뛰어난 데이터 액세스 기능을 제공한다. DataSet은 연결되지 않은 캐시에 데이터를 저장한다. DataSet의 구조는 관계형 데이터베이스의 구조와 비슷하며 Table, 행 및 열로 구성된 계층적 개체 모델을 취한다. 또한, DataSet의 구조에는 DataSet에 대해 정의된 제약 조건 및 관계도 포함되어 있다. 데이터 소스에 연결되지 않은 상태에서 Table 및 행 집합에 대해 작업하려면 DataSet을 사용한다. 그러나, 데이터 액세스에 관한 디자인을 할 때 DataSet을 사용하는 것이 항상 최선의 해결책이라고는 할 수는 없다.

DataSet의 주요 특징은 다음 두 가지 방법으로 로컬의 DataSet 내의 데이터에 액세스하여 해당 데이터를 조작할 수 있다.

• **관계형 데이터베이스의 Table로** DataSet에는 하나의 Table 또는 Table의 모음인 Table컬렉션이 포함될 수 있다. DataSet 의 중요한 특징은 마치 메모리 내의 관계형 데이터 저장소에 있는 것처럼 DataSet 에 포함된 Table 간의 관계를 추적하고 다룰 수 있다는 것이다.

• **XML(eXtended Markup Language) 구조로** DataSet 내의 데이터는 XML 데이터에 액세스할 수도 있다. 데이터를 XML로서 읽고 쓰며, DataSet의 구조를 XML 스키마로서 읽고 쓰기 위한 메서드가 제공된다. 또한, 데이터를 XML로서 보고 쿼리하고 수정하는 작업을 동시에 수행할 수 있도록 XmlDataDocument를 DataSet에 연결할 수 있다.

ADO.NET에서는 DataSet 뿐만 아니라 DataAdapter 객체도 지원하는데, DataAdapter

는 DataSet 및 데이터 소스를 연결시키는 역할을 하며 데이터를 검색하고 저장하는 데 사용된다. DataAdapter는 DataSet에 포함된 데이터를 변경하여 데이터 소스의 데이터와 일치하도록 하는 Fill()메소드와 데이터 소스에 포함된 데이터를 변경하여 DataSet의 데이터와 일치하도록 하는 Update()메소드를 매핑하여 이러한 연결을 제공한다.

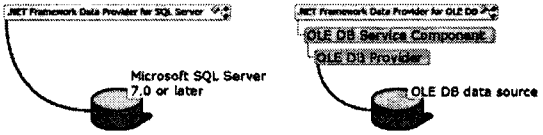
Microsoft SQL Server 데이터베이스에 연결하는 경우, SqlDataAdapter를 관련 SqlCommand 및 SqlConnection과 함께 사용하여 전체 성능을 향상시킬 수 있다. 다른 OLE DB 지원 데이터베이스의 경우, DataAdapter를 관련 OleDbCommand 및 OleDbConnection 개체와 함께 사용한다.

현재는 각종 데이터베이스에 연결하는 .NET Framework Data Provider는

- **Data Provider for SQL Server**
- **Data Provider for OLE DB**
- **Data Provider for ODBC**
- **Data Provider for Oracle** 의 네 가지가 제공된다.

Data Provider for SQL Server는 SQL Server와 통신하기 위해 TDS를 사용하여 통신한다. 추가적인 OLE DB나 혹은 ODBC(Open Database Connectivity) Layer없이 SQL Server에 접근하기 때문에 SQL Server에 액세스하는데 최적화 되어 있다.

다음의 그림은 Data Provider for SQL와 Data Provider for OLE DB를 비교하는 그림이다. Data Provider for OLE DB는 OLE DB Service Component를 통해서 OLE DB Data Source와 통신한다. OLE DB Service Component는 Data Provider for OLE DB에 Transaction Service와 Connection Pooling을 제공하게 된다.



Provider for SQL Server and the .NET Framework Data Provider for OLE DB의 비교

다음과 같이 Namespace를 참조한다.

```
[VisualBasic]
Imports System.Data.SqlClient
[C#]
using System.Data.SqlClient;
```

Data Provider for OLE DB는 Data에 액세스를 가능하게 하는 native OLE DB for COM interop을 사용한다. 로컬과 분산 Transaction을 지원하며, Windows 2000 Component Service로부터 Transaction Detail을 획득하며 Transaction에 자동적으로 enlist된다.

Data Provider for OLE DB는 OLE DB 버전 2.5 인터페이스를 지원하지 않는다. 또한 OLE DB Provider for ODBC (MSDASQL)와 작업할 수 없다.

Data Provider for OLE DB는 MDAC 2.6 혹은 그 이후버전이 설치되어 있어야 한다.

다음과 같은 Namespace를 참조한다.

```
[VisualBasic]
Imports System.Data.OleDb
[C#]
using System.Data.OleDb;
```

다음은 ADO.NET에서 테스트 되어진 Data Provider for OLE DB의 목록이다.

Driver	Provider
SQLOLEDB	Microsoft OLE DB Provider for SQL Server
MSDAORA	Microsoft OLE DB Provider for Oracle
Microsoft.Jet.OLEDB.4.0	OLE DB Provider for Microsoft Jet

Data Provider for ODBC는 데이터에 액세스하기 위해 native ODBC Driver Manager (DM) through COM interop을 사용한다. 이는 로컬과 분산 트랜잭션을 지원한다. Windows 2000 Component Service로부터 Transaction Detail을 획득하며 Transaction에 자동적으로 enlist된다. Data Provider for ODBC는 MDAC 2.6 혹은 그 이후버전이 설치되어 있어야 하며, MDAC 2.7이 추천된다.

다음은 ADO.NET에서 테스트 되어진 Data Provider for ODBC의 목록이다.

Driver
SQL Server
Microsoft ODBC for Oracle
Microsoft Access Driver (*.mdb)

아래와 같이 네임스페이스를 참조한다.

```
[VisualBasic]
Imports System.Data.Odbc
[C#]
using System.Data.Odbc;
```

Data Provider for Oracle는 Oracle client connectivity software를 통해서 Oracle 데이터 소스에 액세스한다. 이 프로바이더는 Oracle Client software 버전 8.1.7과 그 이후버전을 지원한다. 로컬과 분산 트랜잭션을 지원한다. 그러나, **EnlistDistributedTransaction** method를 지원하지 않으므로 자동으로 enlist되지 않는다.

다음과 같이 **System.Data.OracleClient** namespace를 사용한다.

적절한 Data Provider의 선택

Provider	Notes
Data Provider for SQL Server	Microsoft SQL Server 7.0 혹은 그 이후 버전을 사용하는 middle-tier application에 추천된다. Microsoft Data Engine (MSDE) 혹은 그 이후 버전을 사용하는 middle-tier application에 추천된다. .NET Framework Data Provider for OLE DB와 함께 사용하는 OLE DB Provider for SQL Server (SQLOLEDB)에 추천되어 진다. Microsoft SQL Server version 6.5 와 그 이후버전에 대하여,.NET Framework Data Provider for OLE DB와 함께 사용되는 OLE DB Provider for SQL Server에 사용되어야 한다.
Data Provider for OLE DB	Microsoft SQL Server 6.5 혹은 그 이전 버전, 또는 the .NET Framework Data Provider for OLE DB (OLE DB 2.5 interfaces are not required)와 함께 사용되는 OLE DB Interface에 리스트되어 있는 OLE DB Interface를 지원하는 OLE DB 프로바이더. Microsoft Access database를 사용하는 Single-tier, middle-tier application에서는 추천되지 않는다.
Data Provider for ODBC	ODBC data source를 사용하는 middle-tier 혹은 single-tier application에 추천된다. **Note** Data Provider for ODBC는.NET Framework version 1.0에는 포함되어 있지 않다. 만약 사용하려고 한다면 http://msdn.microsoft.com/downloads 사이트에서 다운로드 할 수 있다. 해당 네임 스페이스는 **Microsoft.Data.Odbc**.이다.
Data Provider for Oracle	Oracle data source를 사용하는 middle-tier 혹은 single-tier application에 추천된다. Oracle client software version 8.1.7와 그 이 후 버전을 지원한다. **System.Data.OracleClient** 네임 스페이스를 사용한다. **Note** Data Provider for Oracle는.NET Framework version 1.0에는 포함되어 있지 않다. 만약 사용하려고 한다면 http://msdn.microsoft.com/downloads 사이트에서 다운로드 할 수 있다.

[Visual Basic]

Imports System.Data.OracleClient

[C#]

using System.Data.OracleClient;

류 경 석

- 2000년 전남대학교 경영학과 학사 졸업
- 현재 : 다우 교육센터 .NET Framework 개발강사, 정보통신부 주관 .NET개발자 과정 강사
- 자격사항
  Microsoft Certified Solution Developer(MCSD)
  Microsoft Certified Application Developer(MCAD)
  Microsoft Certified Trainer(MCT)

김 치 용

- 1991년 인제대학교 물리학과 (이학사)
- 1994년 인제대학교 대학원 전산물리학과 (이학석사)
  - 카오스이론 및 프랙탈디자인 전공
- 2000년 인제대학교 대학원 전산물리학과 (이학박사)
  - 멀티미디어그래픽 및 3D 애니메이션 전공
- 1991년~2000년 인제대학교 컴퓨터디자인교육원 선임연구원
- 2002년 AWIC 마야(Maya) 인증 강사
- 2000년~2003년 부산정보대학 정보통신계열 교수
- 2003년~동서대학교 디지털디자인학부 교수
- 관심분야 : 3D 애니메이션, 영상애니메이션, 영상편집, 캐릭터디자인, 카오스&프랙탈디자인, 게임디자인
- 저서 : 멀티미디어그래픽, 3D애니메이션과멀티미디어, 정보출판, 컴퓨터그래픽시험예상문제 등