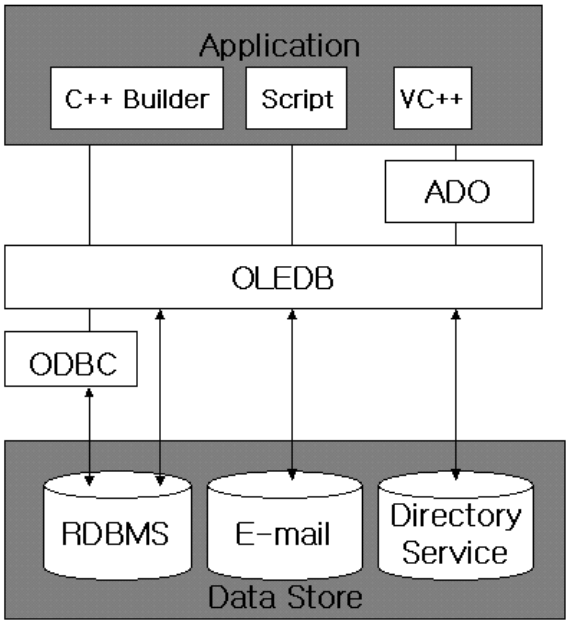
C++Builder ADO Programming (1) – ADO의 개념

ADO란?

1. ActiveX Data Object의 줄임말로 Microsoft사의 최신 데이터 접근기술을 말한다.
2. OLEDB에 기반한 고 수준 인터페이스이다. 고수준의 인터페이스이므로 OLEDB보다 훨씬 간단한 객체모델을 가지며 OLEDB의 복잡함을 감추고 기능들만을 노출시킨 일종의 래퍼(wrapper)라 할 수 있다.

OLEDB란?

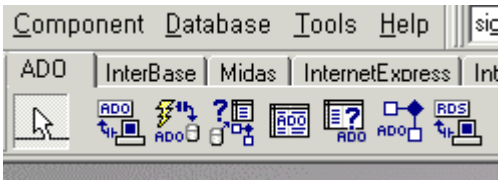
1. Microsoft사의 UDA(Universal Data Access) 전략에 핵심이 되는 아키텍쳐로 COM 기반의 저 수준 프로그래밍 인터페이스이다.
2. 조직의 모든 부분들로부터 얻을 수 있는 데이터에 대한 접근을 제공한다. 좀 더 자세하게는 데이터 소비자와 데이터 공급자라는 두 가지 구성요소를 이용해서 데이터 원본에 대한 연결성, 접근성을 제공한다.
3. 클라이언트/서버나 웹 기반 어플리케이션들은 데이터를 사용하는 입장에서 데이터 소비자가 된다. 데이터 공급자는 데이터 원본들로부터 데이터를 얻고 정보를 해석하고 그것을 공용 인터페이스를 통해 데이터 소비자에게 제공하는 역할을 한다. 여기서 데이터 소비자는 데이터가 어떻게 접근, 처리되는지를 알 필요가 없으며 그러한 기능은 OLEDB 자체에 숨겨져 있다.
4. OLEDB에 깔린 개념은 ODBC에 깔려있는 것과 비슷하다. 그러나 OLEDB는 ODBC보다 훨씬 더 넓은 범위의 데이터 원본에 접근할 수 있게 해준다. OLEDB는 ODBC를 통한 데이터 연결을 지원하므로 ODBC의 모든 능력들을 제공한다. 따라서 하나의 OLEDB 층을 만들어 두고 기존의 ODBC 연결들을 통해서 기존 Database들에 연결할 수도 있다. 이 말은 데이터 소비자인 어플리케이션은 OLEDB 공급자로 ODBC에 접근할 수 있다는 말이며 이 때 가능하면 단일한 OLEDB 층을 통하는 것이 더 효율적이라는 것을 말한다.
5. 여기서 중요한 것은 OLEDB는 관계형 데이터베이스 뿐만 아니라 비 관계형 데이터 원본들(전자우편이나 스프레드 시트, 디렉터리 서비스, 파일시스템등등)을 포함한 모든 종류에 대한 데이터에 대한 고성능 접근방식을 제공한다는 점이다. 이는 ODBC만으로는 얻을 수 없는 능력이다. OLEDB를 이용하면 간단하고 표준적인 수단을 이용해서 다종다양한 정보에 접근하고 정보를 조회, 조작할 수 있기 때문에 (이것이 Microsoft의 UDA전략이다), OLEDB는 Microsoft가 이전에 제시한 DAO나 RDO(이미 물 건너 간 기술이다) 같은 데이터 접근기술 및 전략보다 훨씬 나은 것이라고 할 수 있다.
6. 아래의 그림 1-1 은 위에서 설명한 내용을 요약하여 잘 보여주고 있다.



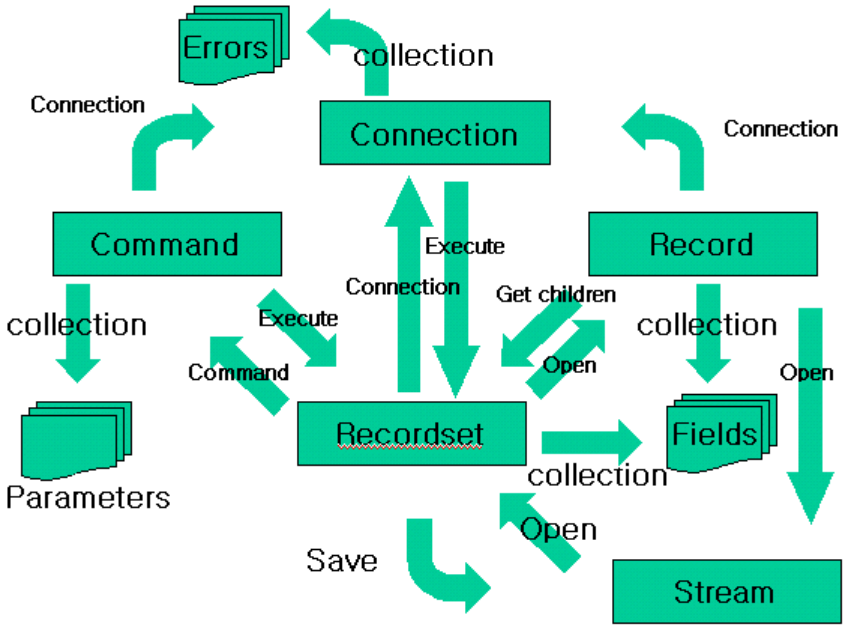
★ 여기서 RDBMS는 흔히 ORACLE, MS-SQL Server 정도가 될 수 있으나 별 말을 하지 않은 경우 앞으로 계속 MS-SQL Server를 대상으로 설명하는 것으로 한다. ORACLE은 MS-SQL Server와는 근본적으로 다른 RDBMS이고 그에 대한 내용 또한 방대하므로 상황에 맞게 주를 달거나 별도의 장에서 설명하기로 한다. 훌륭한 로컬 Database중의 하나인 Access의 경우도 그럴 것이다.

ADO 객체모델

일단 위에서 ADO가 개발자들에게 OLEDB를 사용하여 데이터베이스 프로그래밍을 하기 위한 효과적인 인터페이스를 제공하며 COM 기반의 ActiveX 데이터 객체들의 집합이라는 것을 알았을 것이다. 이 말은 즉 스크립트 언어를 사용하는 웹 환경에서의 개발뿐만 아니라 C++ Builder와 같은 4GL RAD 툴들을 사용하는 전통적인 개발환경에서도 ADO를 사용할 수 있다는 말이 된다. 실제로 C++ Builder에서 OLEDB 기술을 사용하기 위해 ADO탭에 총 7개의 VCL화 된 Component가 있다. (정식명칭은 ADO Express Component이다.)



여기서는 COM에 기반한 ADO 자체의 객체모델과 C++ Builder 내에서 VCL 화 된 클래스 구조를 살펴보기로 하자.

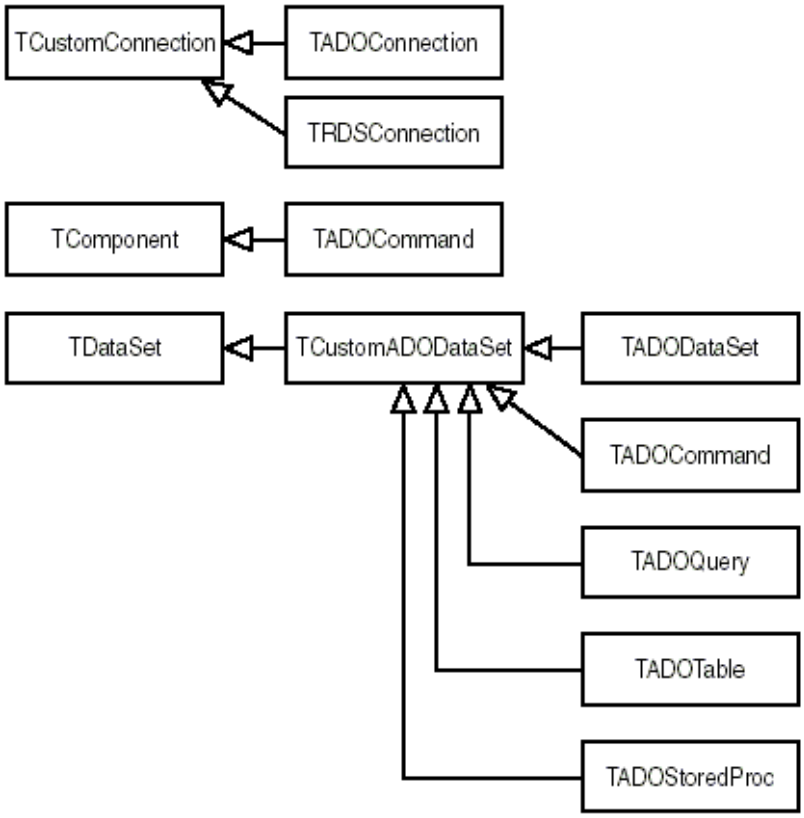


ADO 버전 2.5를 기준으로 했을 때 ADO는 위 그림과 같이 총 5개의 클래스(직사각형이 클래스이다)와 여러 컬렉션들과 Property, Method, 또 이 그림에서는 소개되지 않았지만 여러 이벤트들로 구성되어 있다. 이것들은 이후의 장에서 자세하게 설명될 것이며 어떻게 C++ Builder 내에서 VCL화 되어 있으며 서로 어떤 관련이 있는지 또 어떻게 사용해야 하는지를 필자의 코드와 개발경험을 통해 설명할 것이다. ADO 2.5는 windows 2000 Server를 설치할 때 자동적으로 함께 설치되고 MS-SQL Server 2000을 설치하게 되면 2.6 버전이 설치된다. 필자가 이 강의를 진행하는 시점에서 벌써 MDAC 2.7 버전이 발표되었고 ADO 또한 2.7 버전으로 업그레이드가 되었다. 하지만 여기서는 버전 2.5를 기본으로 강의가 진행될 것이다. 업그레이드 된 내용과 아울러 추가된 내용을 알고 싶은 분들은 Microsoft의 홈페이지를 참고하길 바란다.

관련링크는 http://www.microsoft.com/data/ 이며 여기에서 마이크로 소프트의 최신 데이터 접근 기술들과 각종 기술 지원 및 그들의 UDA 전략을 알아볼 수 있을 것이다. 위의 직사각형들은 본질적으로 COM의 클래스 모듈이므로 그것들의 CLSID(Class Identifier)또는 IID(Interface Identifier)와 VCL화 된 Component의 상호 관계를 알아보는 것도 좋을 것이다.

직사각형 이름	CLSID or IID	VCL Component
Connection	ADODB.Connection	TADOConnection
Command	ADODB.Command	TADOCommand
Stream	ADODB.Stream	ADO 탭에 존재하지 않는다
Record	ADODB.Record	ADO 탭에 존재하지 않는다.
Recordset	ADODB.Recordset	TADODataSet TADOTable TADOQuery TADOStoredProc

다음은 VCL 클래스 계층 구조이다. (Developer's Guide에서 참조)



위의 표와 그림에서 중요한 부분은 ADODB.Recordset 클래스가 VCL Component에서는 4개의 Component로 각각 사용될 수 있다는 것이다. 그림을 보면 이중에서 TADOTable, TADOQuery, TADOStoredProc Component들은 C++ Builder의 Data Access 탭에 있는 TTable, TQuery, TStoredProc 와 기본적으로 같은 성격을 가지는 Component들임을 쉽게 짐작할 수 있다. 이들은 모두 TDataSet으로부터 파생된 클래스이기에 당연한 이야기이며 (실제로 TTable, TQuery, TStoredProc 들은 각각 TDataSet --- TBDEDataSet --- TDBDataSet 의 계층 구조를 가진다) 따라서 그들은 TDataSet의 모든 특징들을 공유한다. 이 말은 어플리케이션 내에서 TTable, TQuery, TStoredProc를 사용해 프로그래밍을 한 것처럼 이들 3개의 Component들도 거의 비슷하게 별 무리 없이 사용할 수 있다. (Data Controls 탭의 Component들과 TDataSource Component를 이용하여 데이터를 표현, 조작할 수 있다는 말이다) 미리 말해 두지만 이 파트의 강의에서는 앞의 TTable, TQuery, TStoredProc 들과 TDataSource, Data Controls 탭들의 Component들을 사용하는 프로그래밍 방법

을 자세하게 설명하지는 않을 것이다. (여러 개의 폼과 데이터 모듈을 포함하는 일반적인 Win32 Database Application을 말하는 것이다. 하지만 중요한 코딩기법은 미리 말을 하고 소개가 될 것이다) 이것은 어떻게 보면 너무 많이 언급된 문제와 방법들로서 대부분의 책이나 매뉴얼에 아주 자세하게 설명되어 있다. 그리고 그것에 대한 사항은 다른 파트에서 자세하게 다룰 것이다. 아무튼 이 파트는 Microsoft사의 Windows DNA라는 하부구조와 Multi-tier 내에서 Enterprise급의 어플리케이션을 개발할 때 C++ Builder로 ADO Programming 방법론을 한 번 고찰해 보고 그것에 초점을 맞출 것이다.

그리고 또 한가지 중요한 부분은 ADO 탭에 Component가 존재하지 않는 ADODB.Record 와 ADODB.Stream 객체이다. 이 두 가지 객체는 ADO가 비 관계형이나 반 구조적 데이터에 접근, 조작, 관리하기 위해 버전 2.5부터 추가된 객체들이다. ADODB.Record 객체는 주로 파일 시스템이나 디렉터리 구조들에 대한 접근성을 제공하는 객체이고 ADODB.Stream 객체는 단순한 파일 I/O 기능 뿐만 아니라 말 그대로 데이터를 영속화 시키는 작업들(주로 VCL의 TPersistent나 TStream 객체들과 비슷한 일을 한다)을 한다. 이 객체들은 아주 막강해서 별도의 다른 Component 없이 ADO만으로도 Local 내지는 Remote 컴퓨터상의 파일 시스템에 접근(대체로 URL을 통해서)하고 그 파일 시스템을 관리(파일 시스템의 내용을 알아내거나 파일의 복사, 이동, 삭제)할 수 있게 해주며 파일을 읽고 쓰며(파일 I/O 기능) 데이터를 영속화 하거나 메모리상에서의 직접전송도 가능하게 하는 아주 막강한 객체이다. 필자가 생각하기에 이 Component가 ADO Express에 추가 되지 않은 것은 매우 안타까운 일이라고 생각한다. (ADO Express Component를 제작할 때 이전 ADO버전이 반영되지 않았나 어설프게 추측해보며 다음 버전의 ADO Express 와 C++ Builder 6 에는 반드시 두 객체가 추가되기를 바란다. 그러나 당장 아쉬운 대로 5에서 변통할 수 있는 방법은 설명도 할 것이다. 또한 이 기능들은 전통적인 개발툴엔 당연히 존재하는 장점이기도 하다.)
Database에 연결하여 데이터를 관리 조작하는 작업들은 나머지 Component들의 몫이다. TADOConnection 와 TRDSConnection 객체는 연결에 관계하고 TADOCommand는 명령을 데이터베이스에 전달하며, ADO 객체 모델의 핵심인 제일 중요한 TADODataSet은 말 그대로 데이터 레코드 셋이다. (미리 얘기한대로 앞에서 거론한 3개의 레코드 셋 Component들은 자주 사용하지 않을 것임을 알려둔다) 그러면 ADO를 C++ Builder Database Application 의 아키텍쳐 모델로 사용하게 될 경우 어떤 장점을 얻게 되며 또 불리한 점은 무엇이 있는지 알아보자.

ADO를 사용할 때 장점 (Developer Guide에서 참조)

솔직히 이 ADO라는 데이터베이스 접근 아키텍쳐는 인터넷과 어플리케이션 서버들, 그리고 COM을 사용할 때 적합하다. 열거해보면,

1. BDE를 대체할 수 있다. 즉 어플리케이션 작성시, Borland Database Engine을 사용할 필요가 없다는 것이다. 특히 어플리케이션을 배포할 때 상대적으로 무거운 부분을 차지하는 BDE를 추가하지 않아도 되는 사실에 공감하는 개발자들도 많이 있으리라 본다.
2. 인터넷에 적합하고 배포가 쉽다. 단적으로 Microsoft의 웹 서버인 IIS 상에서 돌아가는 스크립트 언어인 ASP에서 같은 회사의 Database(Access, MS-SQL)를 접근, 조작하는데 많이 사용되고 있고 설정도 스크립트 내에서 직접 사용되거나 include 되는 방법으로 또는 간단한 ODBC DSN Setting으로 이루어 진다. 그 사항은 C++ Builder Database Application들(일반 Win32 Application, ISAPI, ActiveX Server, Asp Object등등)에도 같이 적용된다.
3. 서류가방 모델과 XML을 지원한다. 이것들은 이후의 장에서 아주 비중 있는 이슈로서 아주 자세하고 밀도 있게 구체적인 코드와 함께 설명될 것이다. XML의 잠재력과 가능성에 대해서 언급하는 것은 바보스러울 정도이다. 이미 이 기술은 거의 모든 벤더들에 의해서 채용되고 있으며 국제적인 표준으로서 광범위한 지원을 받고 있다. 이후의 장에서 ADO 레코드 셋을 XML로 영속화 시키고 XML을 레코드 셋으로 불러들여 작업하는 방법이나 클라이언트에 전송하는 방법, 서류가방 모델의 핵심 구조인 단절된 레코드 셋에서 XML을 사용하는 방법이나 서버 객체 내에서 사용하는 문제 등, Multi-tier 환경에서 Enteprise급 어플리케이션을 작성할 때 유용한 방법들을 살펴볼 것이다.

4. OLE Database(예를 들면 Access나 MS-SQL)에 연결, 접근하는 경우 BDE보다 나은 성능을 제공한다. 아마도 같은 회사에서 만든 것이니 그럴 것이라고 생각할 뿐이다. 흥미 있는 사람은 테스트를 해보아도 좋을 것이다. 적어도 Microsoft는 그렇게 떠들고 있다.

ADO를 사용할 때 단점 (Developer Guide에서 참조)

1. 단지 Windows 플랫폼에서만 사용가능하다. 당연한 얘기이다. 그리고 ADO는 지나치게 Microsoft 제품 지향적이다.
2. BDE 보다 낮은 제어성을 제공한다. BDE Administrator의 각 Database들의 Alias 설정들의 세부 사항들을 한 번 본다면 이 말이 쉽게 이해될 것이다.
3. MIDAS와 같이 작동하지 않는다. 대신 ADO 내에는 RDS 기술이라는 것이 있다. RDS는 MIDAS와 비슷한 분산 환경에서의 개발 기술로 이후의 장에서 설명하기로 한다.
4. OLE Database Provider들이 항상 신뢰성 있고 만능은 아니라는 점이다. 이것은 ADO를 개발과정에 아키텍쳐로 도입하기 전에 항상 그 적합성을 체크 해 보아야 한다는 이야기이다. ORACLE과 SYBASE등 다른 비 Microsoft 계의 RDBMS와 ADO와의 궁합이 대표적인 사항이 될 것이다. 이것 역시 이후의 장에서 알아보기로 하자.

이상 ADO의 장점이나 단점에 대해서 몇 가지 정도를 알아보았다. 어느 기술이 더 좋다 또는 어떤 접근 방법을 채택하는 것이 더 좋다 라는 사실에 관한 논의는 이 장의 목적이 아니다. 하지만 그것은 프로젝트나 개발에 들어가기 앞서 신중하게 고려해야 되며 실제 개발에 들어갔을 때의 추가적인 부담에 관여한 문제이다. 굳이 자신의 효과적인 방법이 있는데도 불구하고 무리하게 새로운 기술을 배우고 적용하는 문제에 욕심을 낼 필요는 없으며 또한 반대로 더 나은 방법이 있는 데에도 불구하고 그것에 대해 관심을 가지지 않는 것도 개발자의 올바른 태도가 아니라고 생각한다. 여기서 중요한 것은 단지 ADO라는 기술이 있으며 여러분은 '아 이런 접근 방식도 있구나' 하고 한 번 보는 것, 경험을 늘려가는 것이 중요한 것이라고 생각한다.

최근에 현실적으로 업계의 주도자인 Microsoft는 ADO의 .Net 버전인 ADO.Net을 소개했다. Microsoft는 누군가는 세상을 이끌어야 한다며 개발자들을 레밍처럼 절벽으로 막 몰고 있는데 그런 점에서 본다면 이 강의는 약간 우울한 결말이 기대되는 변주곡의 시작일 수 있다.
하지만 같이 하다 보면 혹시 알겠는가? 절벽에서 점프해 날아서 다른 땅에 멋지게 도달 할 수 있을지! 시작은 짧을수록 좋다. 왜냐하면 일의 반을 빨리 해치울 수 있기 때문이다. 자, 그럼 어서 빨리 우리의 레밍에게 Borland의 딱지가 붙은 아주 멋진 날개 달린 신발을 주도록 하자~ =ㅅ=;;

Mortalpain