**ADO.NET**

닷넷은 기본적으로 ADO.NET을 이용해 데이터베이스 기능을 처리함

- 기존 ADO 기술을 바탕으로 XML, 비연결 지향 데이터베이스 기법 등

한층 진보된 특징을 갖춘 새로운 데이터베이스 API

학습 내용 1) DB 기본 사용

연결(Connection), 조작(Command), 데이터 가져오기(DataReader)

2) SQL서버와 OLE DB연결 방법을 사용한 데이터 베이스 조작 방법

3) 메모리 데이터베이스 처리 기법

DataTable, DataView, DataSet

[ 그림 1 ADO.NET의 구성도 ]

**두 개의 중요 구성 요소**

* DataProvider : 데이터베이스 연결을 담당하는 역할(닷넷 데이터 공급자라고도 함)

- 닷넷 데이터 공급자

- 특정 데이터 저장소와 통신하는 방법을 알고 있는 형식들의 집합

- 형식들의 이름은 데이터 공급자에 따라 다르지만,

주요 인터페이스를 모두 구현하고 있다.

lDbConmmand, lDbDataAdapter, lDbConnection, lDataReader...

* DataSet : 비 연결 지향 도구

**[ ADO.NET 관련 네임스페이스]**

|  |  |
| --- | --- |
| **네임스페이스** | **포함하고 있는 클래스** |
| System.Data | 데이터 관련 클래스 |
| System.Data.Common | 데이터 공유 클래스 |
| System.Data.OleDb | OleDb 제공 클래스 |
| System.Data.SqlClient | SQL 서버 제공 클래스 |
| System.Data.SqlTypes | SQL Server 기본 데이터 형식 클래스 |

- System.Data : ADO.NET의 핵심 클래스를 다수 포함하고 있슴

테이블, 행, 열, 제약조건, 데이터셋

- System.Data.Common : DataProvider 사이에 공유되는 타입들을 정의함

- System.Data.OleDb : 데이터 셋, SQL쿼리 실행, OLE DB 데이터 연결하기 등의 기능 수행

-System.Data.SqlTypes : MS SQL 서버에서 사용하는 데이터 타입들의 정의

\*) ADO.NET은 System.Data.dll 어셈블리 파일에 정의되어 있으므로 ADO.NET 기반 프로그램을

작성할 경우 해당 파일을 반드시 참조해야 함

**[ Data Provider ]**

데이터베이스 연결을 담당하는 역할

OleDB와 SQL 제공자로 구분함

|  |  |  |
| --- | --- | --- |
| **역할** | **OleDB** | **SQL** |
| 데이터베이스 연결 | OleDbConnection | SqlConnection |
| SQL 명령 실행 | OleDbCommand | SqlCommand |
| 데이터 읽어오기 | OleDbDataReader | SqlDataReader |
| 데이터 전송 | OleDbDataAdapter | SqlDataAdapter |

**ADO.NET을 이용한 데이터베이스 조작 절차**

1. 데이터 소스 연결(Connection)
2. 데이터베이스 조작(Command)
3. 데이터베이스 불러오기(DataReader)
4. 데이터 전송하기

**[ 데이터베이스 연결(Connection) ]**

**Connection 클래스**

사용 절차

1. Connection 객체 생성
2. 데이터베이스 연결 문자열 지정(DNS)
3. Connection객체의 Open메서드를 호출해 데이터베이스 연결
4. Connection객체의 Close메서드를 호출해 데이터베이스 연결 해제

**Connection 클래스 생성자**

1. SQL 서버 Provider를 이용한 연결

방법1)

// 데이터베이스 연결

sqlConnection conn = sqlConnection();

// 데이터 소스 설정

conn.ConnectionString = @”Server=localhost\sqlexpress;database=ADO;uid=sa;pwd=;”;

conn.Open();

// 닫기

Conn.Close();

방법2)

String ConnectionString = @”Server=localhost\sqlexpress;database=ADO;uid=sa;pwd=;”;

sqlConnection conn = sqlConnection(ConnectionString);

conn.Open();

1. OLE DB Provider를 이용한 연결

방법1)

// 데이터베이스 연결

OleDbConnection conn = OleDbConnection();

// 데이터 소스 설정

conn.ConnectionString= “Provider=Microsoft.Jet.OLEDB.4.0;

Data Source = c:\CSharp\ado.mdb”;

conn.Open(); conn.Close();

방법2)

String ConnectionString = “Provider=Microsoft.Jet.OLEDB.4.0;

Data Source= c:\CSharp\ado.mdb”;

OleDBConnection conn = OleDbConnection(ConnectionString);

conn.Open(); conn.Close();

**Connection 클래스 속성**

|  |  |
| --- | --- |
| **속성** | **설명** |
| ConnectionString | 데이터베이스 연결 문자열 |
| ConnectionTimeout | 데이터베이스 연결하려는 동안 대기 시간 |
| Database | 현재 연결해 사용할 데이터베이스 이름 |
| DataSource | 연결한 SQL Server 인스턴스 이름 |
| ServerVersion | 클라이언트가 연결될 SQL Server인스턴스 버전을 포함한 문자열 |
| State | 현재 연결 상태 반환 |

ConnectionString

SqlConnection con = SqlConnection();

conn.ConnectionString = @”Server=localhost\sqlexpress;database=ADO;uid=sa;pwd=;”;

* Server : 연결하고자 하는 데이터베이스 서버의 네트웤 주소(Server는 Data Source, Address, Addr, Network Address로 바꿔 사용 가능)
* Database : 데이터베이스 이름(Database 대신 Initial Catalog로 바꿔 사용 가능)
* UID : 데이터베이스 접속 권한이 있는 사용자 아이디(User ID로 바꿔 사용 가능)
* PWD : 데이터베이스에 접속 권한이 있는 사용자 비밀번호(Password로 바꿔 사용 가능)

Ex) 동일한 개념인 연결 문자열

Conn.ConnectionString = “Server=localhost;database=ADO;uid=sa;pwd=;”;

Conn.ConnectionString = “Addr=127.0.0.1;Initial Catalog=ADO;uid=sa;pwd=;”;

Conn.ConnectionString = “Data Source=127.0.0.1;Database=ADO;uid=sa;pwd=;”;

ConnectionTimeout

연결을 기다리는 시간을 초단위로 지정 : 기본값은 15초임

**Connection 클래스 맴버 메서드**

|  |  |
| --- | --- |
| **메서드** | **설명** |
| ChangeDatabase | 열려있는 SqlConnection에 대해 현재 데이터베이스를 변경 |
| Close | 데이터베이스 연결 종료 |
| CreateCommand | SqlCommand 개체 생성 |
| Open | 데이터베이스 연결 |
| Dispose | SqlConnection 리소스 해제 |

**\*) 예제 참조 01\_데이터베이스 연결**

**[ 데이터베이스 조작(Command) ]**

**Command 클래스를 이용하여 연결된 데이터베이스 조작 가능(연결 이후 과정)**

* SQL 쿼리문을 전달해 데이터베이스를 조작하는 클래스임
* 다음과 같은 순서를 따라야 함

1. 데이터베이스 연결 : Connection 객체 생성
2. 명령 수행 준비 : Command 객체 생성
3. 속성 지정 : Command 객체의 CommandText 프로퍼티에 속성 지정(쿼리, 저장프로시저, 테이블)
4. SQL 쿼리문 실행 : Execute()메서드를 호출해 쿼리문 실행

**Command 클래스 생성자**

SQL 서버일 경우 Command 클래스 생성자

public SqlCommand();

public SqlCommand(string commandText);

public SqlCommand(string commandText, SqlConnection connection);

public SqlCommand(string commandText, SqlConnection connection, SqlTransaction transaction);

\*) OLEDB일 경우도 동일한 생성자 정의되어 있슴

\*) commandText : 실행할 쿼리문이나 저장 프로시저

\*) connection : Connection 개체 인스턴스

\*) transaction : 트랜잭션 처리 개체 인스턴스

- 트랜잭션 : 데이터의 무결성 처리 기능

데이터에 대한 100% 확신이 설때까지 변경된 정보를 데이터베이스에 반영하지 않고, 메

모리의 영역에 임시로 저장하는 기능임

Ex) commit : 임시로 저장된 데이터를 DB에 저장

rollback : 임시로 저장된 데이터를 버리고, 데이터 반영을 포기하는 명령어

**Command 클래스 속성**

|  |  |
| --- | --- |
| **속성** | **설명** |
| CommandText | 실행할 SQL 쿼리문이나 저장 프로시저 |
| CommandTimeout | 데이터베이스 명령 실행 대기 시간 |
| CommandType | CommandText 타입을 지정(쿼리, 저장 프로시저, 테이블) |
| Parameters | 쿼리문이나 저장 프로시저에 사용할 매개변수가 포함된  ParameterCollection을 가져옴 |
| Transaction | 트랜잭션 처리 속성 |
| Connection | 연결되어 있는 Connection 인스턴스 |

commandText : 데이터베이스에서 실행될 쿼리문, 저장 프로시저 이름 기술

ex1)

SqlCommand cmd = new SqlCommand();

cmd.commandText = “select \* from member where user=’magic’”;

cmd.CommandType = CommandType.Text;

ex2)

OleDbCommand cmd = OleDbCommand();

cmd.commandText = “update member set pass=’123’ where user=’magic’”;

cmd.CommandType = CommandType.Text;

**Command 클래스 주요 메서드**

|  |  |
| --- | --- |
| **메서드** | **설명** |
| CreateParameter | Parameter 생성 |
| Prepare | DB 캐시에 쿼리문 저장 |
| Cancel | DB 연결 종료 |
| ExecuteNonQuery | DB 명령 수행(Insert, Update, Delete)  데이터베이스 내용을 변경할 때 사용 |
| ExecuteReader | DB레코드를 Reader 개체로 읽어옴(Select쿼리문 수행시)  DB내용은 변경하지 않고, 결과만 반환  즉 select 쿼리문 전용 메서드임  해당 메서드는 SqlDateReader(OleDbDataReader)를 반환한다.  반환개체에는 Select 쿼리문의 결과가 포함되어 있다. |
| ExecuteScalar | 반환되는 결과 값이 하나일 때 사용하는 메서드  쿼리문의 첫 번째 열을 반환   * 반환 값이 아무리 많더라도 첫번째 데이터만 가져오며, 값이 object형으로 반환되기 때문에 적당히 캐스팅해서 사용해야 함 |
| ReSetCommandTimeout | CommandTimeOut 프로퍼티를 30초 내로 재설정 |
| Cancel | 실행중인 명령을 취소 |

\*) 예제 프로그램 확인 : 윈폼 기반 SQL 쿼리 프로그램

\*) 디자이너를 이용한 Connection 실습

**[ 데이터베이스 가져오기(DataReader) ]**

**DataReader 클래스는 데이터베이스에 있는 레코드를 Select 쿼리문을 이용해서 가져올 수 있슴**

**SQL 서버용과 OLE DB 용으로 구분됨**

* **SqlDataReader**
* **OleDataReader**

**DataReader 클래스 생성자**

Command 개체의 ExecuteReader() 메서드에 의해 생성됨

1. SQL 서버일 경우

SqlConnection conn = new SqlConnection();

SqlCommand cmd = new SqlCommand(“select \* from member”, conn);

SqlDataReader read = cmd.ExecuteReader(); // SqlDataReader 생성

1. OLE DB일 경우

OleDbConnection = new OleDbConnection ();

OleDbCommand cmd = new OleDbCommand(“select \* from member”, conn);

OleDbDataReader read = cmd.ExecuteReader(); // OleDbDataReader개체 생성

**DataReader 클래스 속성**

|  |  |
| --- | --- |
| **속성** | **설명** |
| FieldCount | 현재 레코드의 컬럼(필드) 수를 반환 |
| IsClosed | DataReader 개체가 닫혀 있는지 확인 |
| RecordsAffected | 데이터를 조작할 경우 데이터베이스에서 바뀐 레코드수 반환 |
| ItemDataReader | 개체의 반환된 필드를 지정하는 배열 |

\*)FieldCount 속성

SqlCommand cmd = new SqlCommand(“select id, pwd from member”, conn);

SqlDataReader read = cmd.ExcuteReader();

Console.WriteLine(“반환된 컬럼 개수 : {0}”, read.FiledCount); // 2

\*) RecordsAffected

Insert, Update, Delete 쿼리문을 Command개체로 실행하면 내부 변화가 생김

이때 변화된 레코드의 수를 얻을 수 있슴

SqlCommand cmd = new SqlCommand(“Update member set id = ‘magic’”, conn);

SqlDataReader read = cmd.ExcuteReader();

Console.WriteLine(“반환된 레코드 수 : {0}”, read. RecordsAffected t); // XXX

\*) Item 속성

ADO.NET에서는 컬럼명, 서수 두가지 형식으로 사용됨

Public object this[ string name] { get; } // 컬럼(필드)명 사용

Public object this[int i] { get; } // 서수 사용 : 0부터 진행

SqlCommand cmd = new SqlCommand(“Select \* from member”, conn);

SqlDataReader read = cmd.ExcuteReader();

While( read.Read() )

{

Console.WriteLine(read[“id”] + “\t” + read[“pwd”] + “\t” + read[“name”]);

Console.WriteLine(read[0] + “\t” + read[1] + “\t” + read[2]);

}

**DataReader 클래스 주요 메서드**

|  |  |
| --- | --- |
| **메서드** | **설명** |
| Read | 레코드 커서를 다음 레코드로 이동 |
| NextResult | 여러 개의 레코드셋에서 다음 레코드셋으로 이동 |
| GetName | 레코드셋의 필드 이름을 반환 |
| GetOrdinal | 레코드셋의 필드 서수값 반환 |
| GetValue | 특정 필드의 값을 출력 |
| GetValues | 현재 레코드의 모든 필드값을 배열 형태로 반환 |
| GetXXX | 레코드셋의 데이터를 XXX함수를 이용해서 가져옴 |
| Close | DataReader개체를 닫음 |

\*)Read 메서드

레코드의 시작점(BOF : Begin Of File)에서 마지막 레코드(EOF : End Of file)까지 하나씩 레코드 커서

를 이동시킴

While( read.Read() ) // 존재하지 않을떄까지 레코드 이동

{ }

\*)GetXXX메서드

GetString, GetInt32…

관련 메서드는 값을 읽어올 때, DB의 정의된 데이터 타입에 따라 처리 메서드가 달라짐

**예제 확인) 테이블 분석 프로그램 : member테이블 구성 및 사용**