Chapter 14
ADO .NET

학습 목표

- ADO .NET은 데이터베이스 사용의 편의를 위해, MS사가 만든 표준 데이터베이스 인터페이스이다.
- 프로그램을 한다는 것에 있어서 빠질 수 없는 부분이 데이터베이스 부분이다.
- ADO .NET의 C#에서 활용을 학습하도록 한다.

ADO .NET의 개요

- ADO .NET
  - .NET에서 데이터베이스 조작에 관련된 .NET 클래스들의 집합
  - 다양한 방법으로 데이터베이스를 검색, 수정, 업데이트 등의 작업 가능
  - ADO의 발정 형태
    - Com InterOperation 서비스를 통해서 기존의 ADO를 사용
    - .NET의 부가적인 기능 포함
  - ADO .NET이 ADO와 비교해 개선된 사항
    - .NET 기반의 다양한 언어 지원
    - XML 지원
      - Framework 상에서 일관된 포맷으로 지원
    - 단절된 데이터 구조를 표준으로 사용
      - connection이 끊어진 상태에서 작업, 소형 메모리 데이터베이스 모델 지원

ADO .NET의 개요

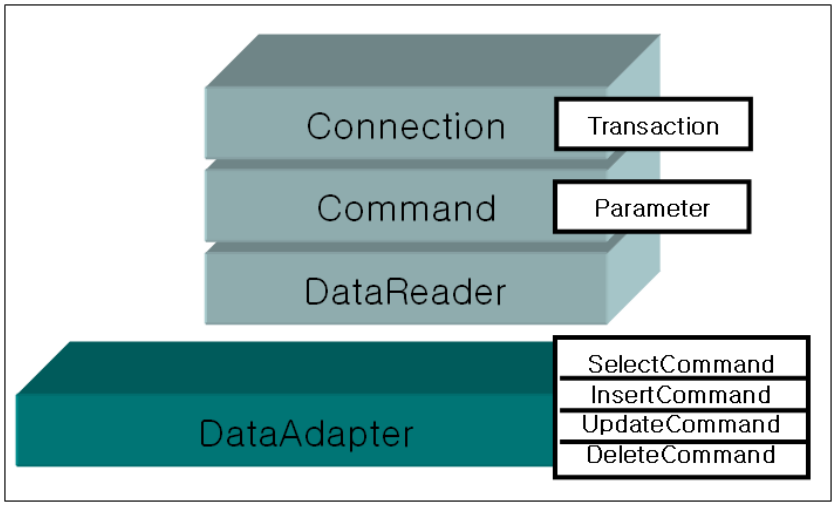
- ADO .NET과 관련된 네임스페이스

```
System.Data
System.Data.Common
System.Date.Oledb
System.Dara.SqlClient
```

- ADO .NET의 기본 구조
  - 구조적으로 .NET Data Provider와 DataSet의 두 부분으로 나뉘어 짐
  - .NET Data Provider
    - 데이터베이스에 연결하고, SQL문을 실행시키는 역할
  - DataSet
    - 데이터베이스의 데이터를 DataSet으로 쉽게 이식 가능
    - 데이터베이스와 분리되어 DataSet만으로도 대부분의 작업 가능

ADO .NET의 개요

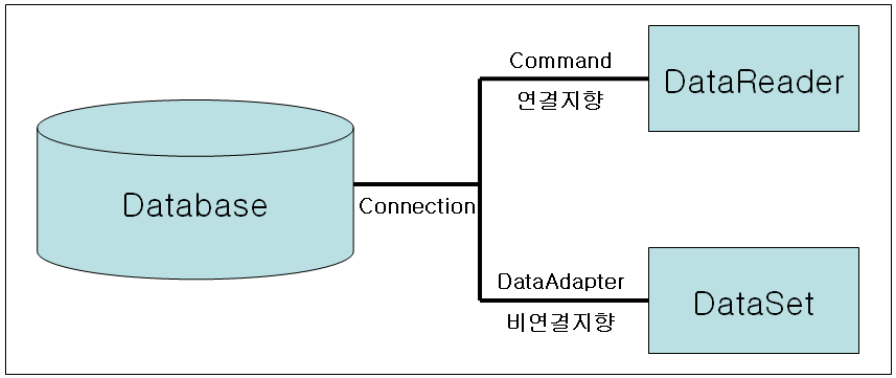
- .NET Data Provider의 구성 요소



- ADO .NET와 기존 ADO 연결 지향성
  - 데이터베이스에 연결된 후 데이터베이스와 동적으로 작업을 수행
  - Command 클래스와 관련이 깊음

ADO .NET의 개요

- ADO .NET와 기존 ADO 비연결 지향성
  - 메모리 데이터베이스와 관련이 있음
  - 데이터베이스에 연결 설정 후 데이터베이스 형태로 보유하고 있으면서 메모리 데이터베이스마느올 작업을 수행
  - 데이터베이스와 동적으로 작업을 수행
- .NET에서 이용되는 데이터 구조



ADO .NET의 개요

- ADO .NET의 간략한 구조
  - DataSet
    - 데이터 원본과 단절
    - 데이터는 Memory In Cash 형태
    - XML과의 데이터 교환
  - .NET 데이터 공급자 컴포넌트 (.NET Data Provider)
    - Connection : 데이터 원본에 연결
    - Command : 검색, 삽입, 수정, 삭제 등의 명령 수행
    - DataReader : 순방향 읽기전용으로 연결된 결과 집합
    - DataAdapter : 데이터베이스로부터 DataSet에 값을 채우거나 필요로 할 때 데이터베이스로 업데이트 가능

데이터베이스 다루기

- 데이터베이스 공급자의 Connection 설정 – 1
  1. 네임스페이스 명시
     - Sql을 사용할 때

```
using System.Data.SqlClient
```

     - Oledb를 사용할 때

```
using System.Data.OleDb;
```

- 데이터베이스 공급자의 Connection 설정 – 2
  2. Connection 객체 생성 : 생성자에 연결 프로퍼티 설정
     - Sql을 사용할 때

```
SqlConnection conn;
conn = new SqlConnection(Server=localhost;user id=sa;password; database=northwind");
```

     - Oledb를 사용할 때

```
OleDbConnection conn;
conn = new
OleDbConnection(Provider = Microsoft.Jet.OLEDB.4.0; Data Source=test.mdb");
```

- 데이터베이스 공급자의 Connection 설정 – 3

3. Connecton 연결

```
conn.Open();
```

4. 작업수행(SQL 쿼리 작업)
5. 연결 닫기

```
conn.Close();
```

– 데이터베이스의 정보를 출력하는 예제
  - [코드 14-1] 참조

데이터베이스 다루기

- SqlCommand 클래스
  - Connection 객체를 이용해 데이터베이스와 연결한 후, DB에 필요한 명령을 전달하거나 결과를 받아올 때 사용
  - SqlCommand 클래스를 이용하여 수행할 수 있는 명령들
    - 단일 값 및 레코드 셋을 반환하는 SELECT, CREATE, ALERT, DROP 같은 DDL(Data Definition Language) 명령
    - GRANT, DENY, REVOKE 같은 DCL(Data Control Language) 명령
    - INSERT, UPDATE, DELETE 같은 DML(Data Modification Language) 명령
  - 명령(Command)에 관련된 여러 가지 정보를 관리 가능
  - 명령 수해을 위한 메서드들 포함

데이터베이스 다루기

- SqlCommand의 주요 속성들
  - CommandText
    - 데이터 소스에서 실행할 SQL 문이나 저장 프로시저를 가져오거나 설정
  - CommandType
    - CommandText 속성이 해석될 방법을 나타내는 값을 가져오거나 설정
  - Connection
    - SqlCommand의 인스턴스에서 사용하는 SqlConnection을 가져오거나 설정
  - Parameters
    - SqlParameterCollection을 가져옴
  - Transaction
    - SqlCommand가 실행하는 트랜잭션을 가져오거나 설정

데이터베이스 다루기

- SqlCommand의 주요 메서드
  - ExecuteNonQuery()
    - Connection에 대한 SQL 문을 실행하고 영향을 받는 행의 개수를 반환
  - ExecuteReader()
    - CommandText를 Connection에 보내고, SqlDataReader를 생성
  - ExecuteScalar()
    - 쿼리를 실행하고 쿼리에서 반환된 결과 집합의 첫 번째 행의 첫 번째 열 반환
  - ExecuteXmlReader()
    - CommandText를 Connection에 보내고, XmlReader 객체를 생성

데이터베이스 다루기

- SqlCommand의 ExecuteNonQuery
  - ExecuteNonQuery() 메서드
    - SqlCommand를 이용하여 데이터베이스에 명령을 전달하고 이를 수행
    - Select 명령을 제외한 대부분의 명령을 ExecuteNonQuery()를 이용하여 처리 가능
    - ExecuteNonQuery() 예제
      - [코드 14-2] 참조

- SqlCommand의 Parameter(Insert 문장) – 1
  - 반복적인 데이터베이스의 작업이나 SQL 쿼리가 복잡할 경우, Parameter를 이용
    - 보다 효율적 작업 수행
    - 메모리 캐시 측면에서 성능 향상
  - Parameter 변수 setting 방법과 Parameter 타입의 지정 예

```
/* Parameter 변수를 셋팅하는 방법 */
string query ="Insert into Address values (@ID, @Name, @Address)";
SqlCommand comm = new SqlCommand(query, conn);

/* Parameter 타입을 지정하는 예 */
comm.Parameters.Add("@ID", SqlDbType.TinyInt);
comm.Parameters.Add("@Name", SqlDbType.Char);
comm.Parameters.Add("@Address", SqlDbType.Char);
```

  - 각각의 Parameter는 SQL문장에서 @표시를 앞에 붙여 표시
  - 각각의 Parameter에 타입을 지정할 때에는 SqlDbType 열거형의 멤버를 이용

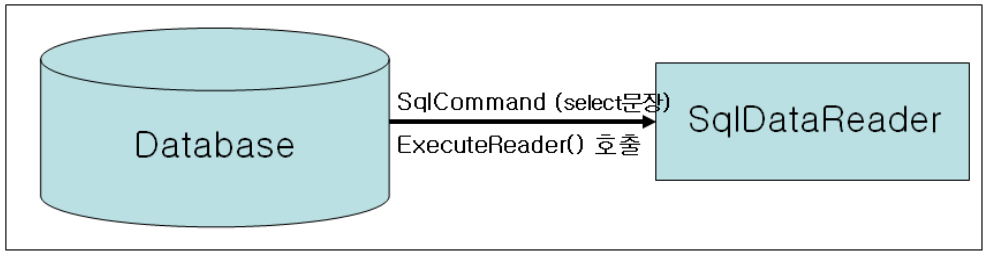
- SqlCommand의 Parameter(Insert 문장) – 2
  - Parameter에 값 setting 방법

```
comm.Parameters["@ID"].Value = 6;
comm.Parameters["@Name"].Value = "HongGilDDong";
comm.Parameters["@Address"].Value = "Seoul";
```

  - [코드 14-3] 참조
  - Parameter는 SQL 문장을 편리하게 사용하기 위해 모든 명령에 사용 가능

데이터베이스 다루기

- SqlCommand의ExecuteReader
  - Select 문장을 사용할 때 이용되는 메서드



- DB에 연결하고 데이터를 불러오는 순서 – 1
  1. 네임스페이스 명시

```
using System.Data.SqlClient;
```

  2. Connection 객체 생성 : 생성자에게 연결 정보 전달, 연결 프로퍼티 설정

```
SqlConnection conn;
conn = new SqlConnection(Server=localhost;user
id=sa;password=;database=northwind");
```

데이터베이스 다루기

- DB에 연결하고 데이터를 불러오는 순서 – 2

3. Connecton 연결

```
conn.Open();
```

4. Command 객체 생성 : 실행할 쿼리와 Connctotion을 인수로 함

```
SqlCommand comm = new SqlCommand("select * from 테이블", conn);
```

5. 데이터 읽어오기

```
SqlDataReader reader = comm.ExecuteReader();
```

6. 작업수행
7. 연결 닫기

```
conn.Close();
```

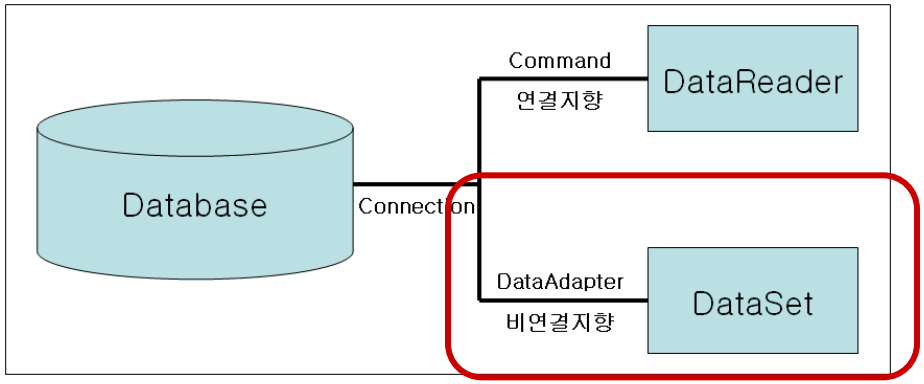
데이터베이스 다루기

- DB에 연결하고 데이터를 불러오기
  - [코드 14-4] 참조

- SqlCommand의 Parameter(Select문장)
  - Select 문장에서 Parameter를 이용
  - [코드 14-5] 참조

DataSet과 DataAdapter

- DataSet과 DataAdapter의 관계
  - 웹 프로그래밍의 작업에 가장 큰 문제가 되는 데이터베이스의 과부하 문제의 해결책으로 제시되는 .NET의 새로운 architecture



- DataAdapter 속성

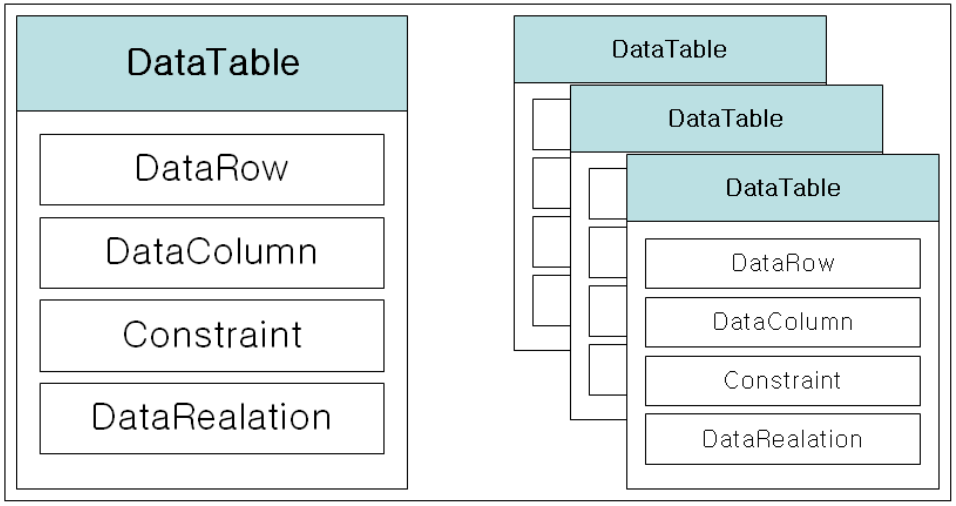
필 드	설 명
DeleteCommand	삭제 명령을 사용하는 Command 필드
InsertCommand	입력 명령을 사용하는 Command 필드
SelectCommand	조회 명령을 사용하는 Command 필드
UpdateCommand	갱신 명령을 사용하는 Command 필드

DataSet과 DataAdapter

- DataAdapter 주요 메서드

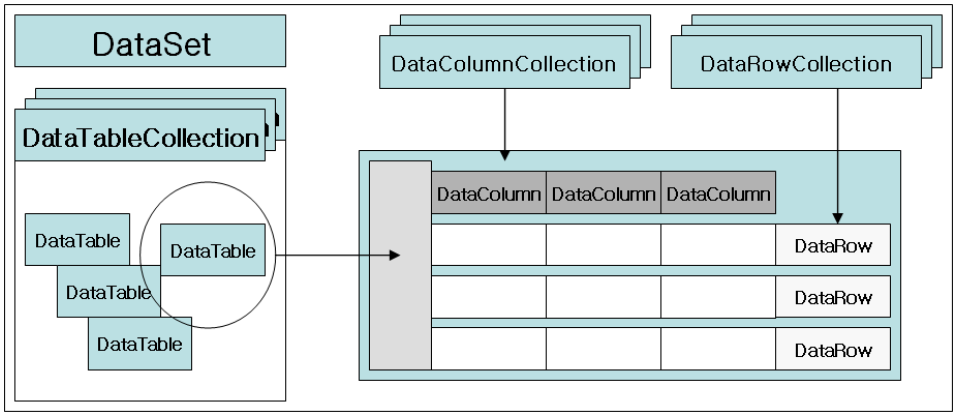
메서드	설 명
Fill()	데이터 원본에 있는 데이터를 DataSet의 DataTable에 넣어 줌
FillSchema()	데이터 원본의 스키마에 맞게 DataSet에 DataTable을 생성
Update()	DataSet의 DataTable에서 변경된 데이터를 원본에 반영시킴

- DataSet의 구성요소

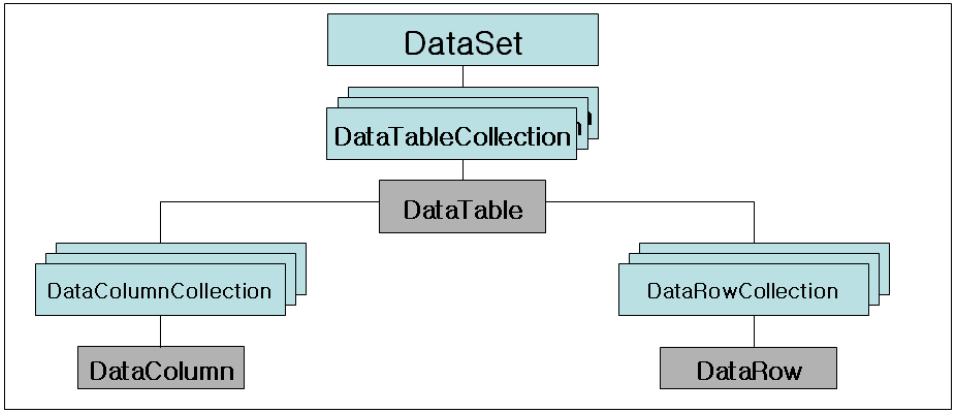


DataSet과 DataAdapter

- DataSet의 개념상 구성요소



- DataSet의 계층 구조



DataSet과 DataAdapter

- DataSet 구성 요소의 정리
  - DataSet
    - 전체 DataSet을 담을 수 있으며, 하나 이상의 테이블 또는 전체 데이터베이스를 담을 수 있음
  - DataTable
    - 하나의 테이블 정보를 담을 수 있음
  - DataRow
    - 테이블에서 하나의 행에 해당하는 값을 담고 있음
  - DataColumn
    - 테이블에서 하나의 열에 대한 정보를 담고 있음
  - DataRelation
    - 두 테이블 사이에 관계 설정을 담고 있음
  - Constraint
    - 테이블에서 값을 제어하는 규약을 담고 있음

DataSet과 DataAdapter

- DataAdapter와 DataSet을 만드는 방법 - 1

1. Connection과 Query 생성

```
SqlConnection conn;
conn = new SqlConnection(Server=localhost; user id=sa; password=;
database=northwind");
String sql = "select * from Address";
```

2. DataAdapter 생성

```
SqlDataAdapter adapter = new SqlDataAdapter();
```

3. SqlCommand 할당

```
adapter.SelectCommand= new SqlCommand(sql, conn);
```

4. DataSet 생성

```
DataSet ds = new DataSet();
```

DataSet과 DataAdapter

- DataAdapter와 DataSet을 만드는 방법 - 2
  4. DataAdapter를 이용하여 DataSet 채우기

```
adapter.Fill(ds);
```

  5. 작업수행
  6. 연결 닫기

```
conn.Close();
```

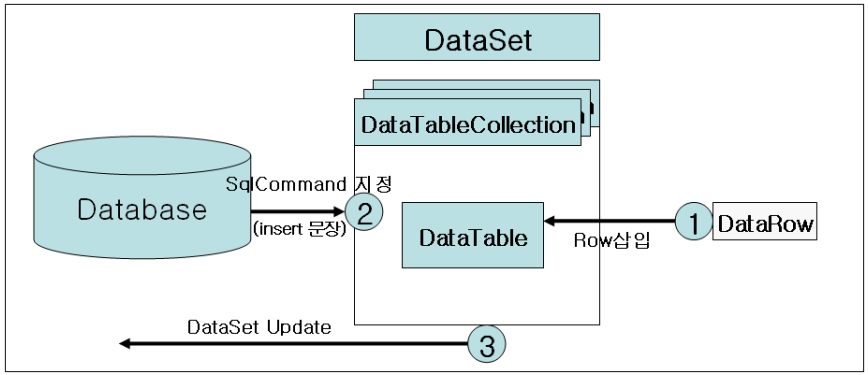
  - DataAdapter를 만드는 다양한 방법

```
public SqlDataAdapter();
public SqlDataAdapter(SqlCommand selectCommand);
public SqlDataAdapter(string selectCommandText, SqlConnection selectConnection);
public SqlDataAdapter(string selectCommandText, string selectConnectionString);
```

  - [코드 14-6], [코드 14-8] 참조

DataSet과 DataAdapter

- DataSet으로 작업하기
  - 데이터베이스와 connection이 끊어졌더라도 DataSet을 계속적으로 이용할 수 있음
  - [코드 14-7] 참조
- DataSet에서 Command 사용하기
  - 연결이 끊어진 DataSet에서 다시 데이터베이스의 연결을 재개하고, InsertCommand 명령을 이용하는 방법
    1. DataSet 자체에 레코드를 삽입
    2. qlCommand를 지정
    3. ataSet을 데이터베이스로 업데이트

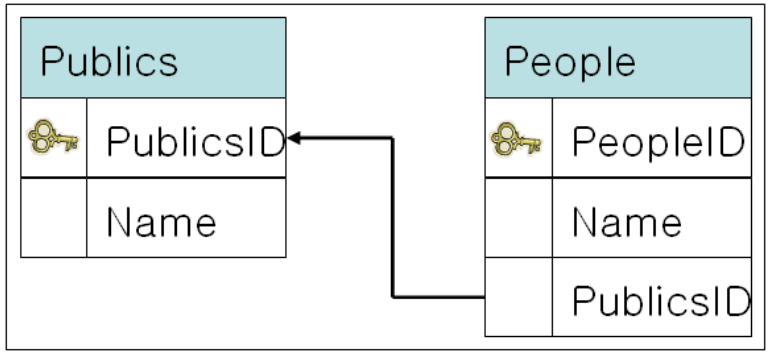


DataSet과 DataAdapter

- Connection 설정 문제
  - Update() 메서드를 호출했을 때 Connection 객체의 상태를 확인
    - 닫혀 있을 경우에는 데이터베이스와의 연결을 열어 줌
    - 데이터를 업데이트한 후 데이터베이스와의 연결을 자동으로 닫아 줌
    - 데이터베이스와 연결이 열렸을 경우는 그 연결을 유지한 채로 데이터를 채움, 이 경우 명시적으로 연결을 닫아 주어야 함
  - [코드 14-9] 참조

DataSet과 DataAdapter

- 두 개의 DataSet 병합하여 사용하기
  - DataSet이 여러 개 있을 때 두 개의 DataSet을 병합 사용 가능
  - Merge() 메서드 이용
  - [코드 14-10] 참조
- DataSet에서의 관계 설정
  - RelationShip
    - 데이터베이스 내에 테이블이 여러 개 존재하고 이 테이블들 간의 관계를 의미



  - [코드 14-11] 참조

학습정리

- ADO .NET은 데이터베이스 사용의 편의를 위해, MS사가 만든 표준 데이터베이스 인터페이스이다.
- MS-SQL 전용으로 사용하기 위한 SqlClient 공급자와 각각의 데이터베이스 회사들에 맞게 구현해 놓은 OleDb 공급자를 이용하여 데이터베이스 관련 프로그램을 할 수 있다.
- ADO .NET은 기존의 ADO에 DataSet의 개념과 XML 기능이 추가되었다.
- 기존의 데이터베이스의 기능을 그대로 수용하면서 보다 발전적인 모델이 바로 ADO .NET이다.