Chapter 12 Linq

01 Linq 소개

* Linq
* 컬렉션 형태의 데이터를 쉽게 다루고자, SQL 본따 만든 구문
* C# 객체의 집합을 쉽게 관리 가능
* SQL 서버와 함께 연동해서 데이터베이스 관리 용이
* 예
* //Linq 질의를 사용해 반환한다.

List<int> input = new List<int>() {1,2,3,4,5,6,7,8,9,10};

Return from item in input

Where item % 2 == 0

Select item;

* List<int> input = new List<int>() {1,2,3,4,5,6,7,8,9,10};

Var output = from item in input

Where item % 2 == 0

Orderby item

Select item;

* 데이터!데이터!
* C# 언어에 통합된 데이터 질의 기능
* 데이터 질의란 데이터네 대해 물어보는 것
* From : 어떤 데이터 집합에서 찾을 것인가
* Where : 어떤 값의 데이터를 찾을 것인가
* Select : 어떤 항목을 추출할 것인가
* 일반 코드 vs Linq 비교
* From profile in arrProfile

Where profile.Height <175

Orderby profile.Height

Select profile;

* From in select 구문
* From <변수 이름> in <컬렉션 이름>

Select <결과에 넣을 요소>

* Link<int> <output> = new List<int>( );

Foreach (var <변수 이름> in <컬렉션 이름>)

{

Output.Add(<결과에 넣을 요소>);

}

class Program

{

static void Main(string[] args)

{

List<int> input = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

var output = from item in input

select item \* item;

foreach (var item in output)

{

Console.WriteLine(item);

}

}

}

* Where 구문 : 조건 지정할 때 사용
* 기본 형태

From <변수 이름> in <컬렉션 이름>

Where <조건식>

Select <변수 이름>

* 예

List<int> <output> = new List<int>( );

Foreach (var <변수 이름> in <컬렉션 이름>)

{

If (조건식)

{

Output.Add(<결과에 넣을 요소>);

}

}

class Program

{

static void Main(string[] args)

{

List<int> input = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

var output = from item in input

where (item > 5) && (item % 2 == 0)

select item;

foreach (var item in output)

{

Console.WriteLine(item);

}

}

}

* Orderby 구문 : 정렬할 때 사용
* 기본 형태

From <변수 이름> in <컬렉션 이름>

Where <조건식>

Orderby <정렬 대상> <정렬 순서>

Select <변수 이름>

class Program

{

static void Main(string[] args)

{

List<int> input = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

var output = from item in input

where (item > 5) && (item % 2 == 0)

orderby item descending

select item;

foreach (var item in output)

{

Console.WriteLine(item);

}

}

* }
* Linq 결과의 자료형과 변환
* IEnumerable<int> output = from item in input

Where item % 2 == 0

Select item;

* Public int[ ] SelectEven(int[ ] input)

{

Return (from item in input

Where item % 2 == 0

Select item).ToArray<int>( );

}

* Public List<int> SelectEven(List<int> input)

{

Return (from item in input

Where item % 2 == 0

Select item).ToList<int>( );

}

class Program

{

static void Main(string[] args)

{

List<int> input = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

var output = from item in input

where item % 2 == 0

select new

{

A = item \* 2,

B = item \* item,

C = 100

};

foreach (var item in output)

{

Console.WriteLine(item.A);

Console.WriteLine(item.B);

Console.WriteLine(item.C);

Console.WriteLine();

}

}

}

class Profile

{

public string Name { get; set; }

public int Height { get; set; }

}

static void Main(string[] args)

{

Profile[] arrProfile =

{

new Profile() {Name = "정우성",Height = 186 },

new Profile() {Name = "김태희",Height = 158 },

new Profile() {Name = "고현정",Height = 172 },

new Profile() {Name = "이문세",Height = 178 },

new Profile() {Name = "하동훈",Height = 171 }

};

var profiles = from h in arrProfile

where h.Height > 175

orderby h.Height

select h.Height \* 0.393;

foreach (var profile in profiles)

{

Console.WriteLine(profile);

}

}

* 여러 개의 데이터 원본에 접근하려면 우리는 이 from 문을 중첩해서 사용
* Foreach 문을 중첩해서 사용하는 것과 비슷하다고 할 수 있음

class Class {

public String Name { get;set;}

public int[] Score{get;set;}

}

Class[] arrClass = {

New Class(){Name = “연두반”,Score = new int[]{99,88,77,24}},

…….};

Var classes = from c in arrClass

From s in c.Score

Where s<60

Select new {c.Name, Lowest = s };

* Group by로 데이터 분류하기
* Group by는 특정 기준으로 데이터 분류
* Group by 절의 사용 형식

Group A by B into C

* Group by 절의 사용 예

Var listProfile = from profile in arrProfile

Group profile by profile.Height<`75 into g

Select new {GroupKey = g.Key, Profiles = g};

* Join
* 내부 조인
* 두 데이터 원본 사이에서 일치하는 데이터들만 연결한 후 반환
* 내부 조인은 교집합과 비슷
* 내부 조인 형식

From a in A

Join b in B on a.XX equals b.YYY

* 내부 조인 예제

Var listProfile = from profile in arrProfile

Join product in arrProduct on profile.Name

Equals product.Star

Select new

{

Name = profile.Name,

Work = product.Title,

Height = profile.Height

}

* 외부 조인
* 외부 조인 할 때 먼저 join 을 이용해서 조인을 수행한 후 그 결과를 임시 컬렉션에 저장하고, 이 임시 컬렉션에 대해 DefaultIfEmpty 연산을 수행해서 비어 있는 조인 결과에 빈 값을 채워 넣음
* DefaultIfEmpty 연산을 거친 임시 컬렉션에서 from 절을 통해 범위 변수를 뽑아내고, 이 범위 변수와 기준 데이터 원본에서 뽑아낸 범위 변수를 이용해서 결과를 추출해 내면 외부 조인!
* 외부 조인 예제

Var listProfile = from profile in arrProfile

Join product in arrProduct on profile.Name equals product.Star into ps

From product in ps.DefaultIfEmpty(new Product(){Title = “그런거 없음”})

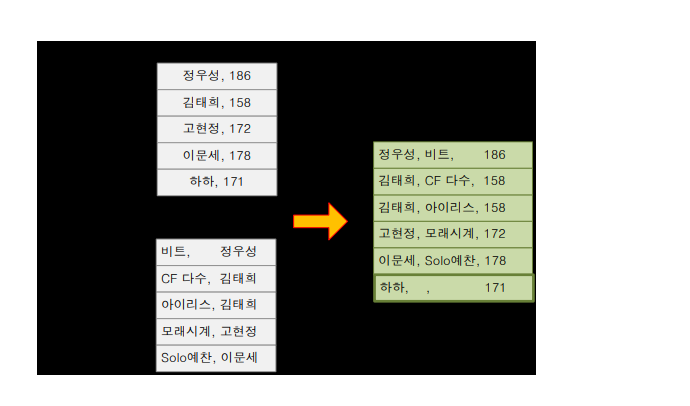
Select new{

Name = profile.Name,

Work = product.Title,

Height = profile.Height

}

* 외부 조인
* 기본적으로 내부조인과 비슷하지만, 조인 결과에 기준이 되는 데이터 원본은 모두 포함된다는 점이 다름

04 함께하는 응용예제

* Linq to Xml
* Xml 계층 구조

<people>

<person birthday = “12/09”><name>윤인성</name></person>

</people>

* Linq to Xml 코드
* using System;
* using System.Collections.Generic;
* using System.Linq;
* using System.Text;
* using System.Threading.Tasks;
* using System.Xml.Linq;
* namespace LinqToXML
* {
* class Program
* {
* class Weather
* {
* public string Hour { get; set; }
* public string Day { get; set; }
* public string Wf { get; set; }
* public string Temp { get; set; }
* public string WdKor { get; set; }
* public string WfKor { get; set; }
* public string Tmn { get; set; }
* public string Tmx { get; set; }
* }
* static void Main(string[] args)
* {
* string url = "http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1150061500";
* XElement xElement = XElement.Load(url);
* var xmlQuery = from item in xElement.Descendants("data")
* select new Weather()
* {
* Hour = item.Element("hour").Value,
* Day = item.Element("day").Value,
* Temp = item.Element("temp").Value,
* WdKor = item.Element("wdKor").Value,
* WfKor = item.Element("wfKor").Value,
* Tmn = item.Element("tmn").Value,
* Tmx = item.Element("tmx").Value
* };
* foreach (var item in xmlQuery)
* {
* Console.Write(item.Hour + "\t");
* Console.Write(item.Day + "\t");
* Console.Write(item.Temp + "\t");
* Console.Write(item.WdKor + "\t");
* Console.Write(item.WfKor + "\t");
* Console.Write(item.Tmn + "\t");
* Console.Write(item.Tmx + "\t");
* Console.WriteLine();
* }
* }
* }
* }

06 이벤트 : 객체에 일어난 사건 알리기

* 이벤트 : 사건을 나타내는 객체
* 동작 원리는 델리게이트와 유사
* 이벤트는 델리게이트를 event 한정자로 수식하여 선언
* 다음은 이벤트를 선언하고 사용하는 절차

1. 델리게이트 선언
2. 클래스 내에 (1) 번에서 선언한 델리게이트의 인스턴스를 event 한정자로 수식해서 선언
3. 이벤트 핸들러를 작성, 이벤트 핸들러는 (1) 번에서 선언한 델리게이트와 일치하는 메소드
4. 클래스의 인스턴스를 생성하고 이 객체의 이벤트에 (3) 번에서 작성한 이벤트 핸들러 등록
5. 이벤트가 발생하면 이벤트 핸들러가 호출됨

* (1) 델리게이트 선언

Delegate void EventHandler(string message);

* (2) 클래스 내에 (1) 번에서 선언한 델리게이트의 인스턴스를 event 한정자로 수식해서 서언

Class MyNotifier{

Public event EventHandler somethingHappened;

Public void DoSomething(int number)

{

Int temp = number % 10;

If (temp!=0&&temp %3==0){

SomethingHappened(String.Format(“{0}”,number));

}

}

* (4) 클래스의 인스턴스를 생성하고 이 객체의 이벤트에 (3)번에서 작성한 이벤트 핸들러 등록
* class MainApp
* {
* static public void MyHandler(string message)
* {
* Console.WriteLine(message);
* }
* static void Main(string[] args)
* {
* MyNotifier notifier = new MyNotifier();
* notifier.SomethingHappened += new EventHandler(MyHandler);
* for (int i = 1; i < 30; i++)
* {
* notifier.DoSomething(i);
* }
* }
* }
* (5) 이벤트가 발생하면 이벤트 핸들러가 호출됨

class MainApp

{

static public void MyHandler(string message)

{

Console.WriteLine(message);

}

static void Main(string[] args)

{

MyNotifier notifier = new MyNotifier();

notifier.SomethingHappened += MyHandler;

for (int i = 1; i < 30; i++)

{

notifier.DoSomething(i);

}

}

* }

07 델리게이트와 이벤트

* 질문 : 이벤트는 델리게이트에 event 키워드로 수식해서 선언한 것에 불과 합니다. 언뜻 봤을 때는 둘의 차이가 별로 없는 것 같은데, 마이크로소프트의 C# 팀은 왜 이벤트를 언어에 추가했을까?
* 답 : 이벤트는 외부에서 직접 사용할 수 없기 때문에 클래스 외부에서 이벤트를 임의로 일으킬 수 없지만, 델리게이트는 그렇지 않음.   
    
  예를 들어 이메일 서버에 접근하여 이메일이 도착하면 “이메일 도착 사건”을 표현한다고 해보자. 이것을 델리게이트로 표현하면 프로그래머가 임의로 “이메일 도착 사건”을 일으킬 수 있지만, 이벤트로 표현하면 임의로 조작하는 것이 불가능해짐. 이벤트를 사용하는 것이 더 안전한 코드를 만드는 길이며, 델리게이트는 콜백 목적으로만 사용하는 것이 맞음.