**Evaluation Only. Created with Aspose.Words. Copyright 2003-2021 Aspose Pty Ltd.**

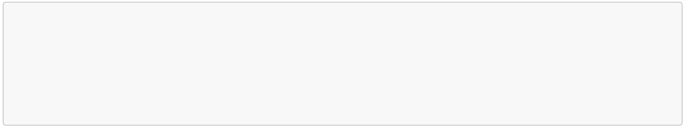
Linq



Linq es una API orientada al uso de consultas a diferentes tipos de contenido, como objetos, entidades, XML, etc. De esta manera se resume en una sintaxis sencilla y fácil de leer, tratar y mantener el tratamiento de diferentes tipos de datos.

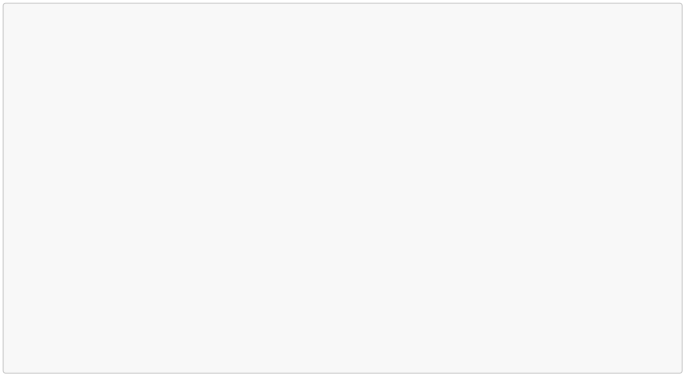
From

var cust = new List<Customer>(); //queryAllCustomers is an IEnumerable<Customer> var queryAllCustomers = (from cust in customers select cust);



Join

var innerJoinQuery =



from category in categories

join prod in products on category.ID equals prod.CategoryID

select new { ProductName = prod.Name, Category = category.Name };

List<Person> people = new List<Person> { magnus, terry, charlotte }; List<Pet> pets = new List<Pet> { barley, boots, whiskers, daisy };

// Create a list of Person-Pet pairs where

// each element is an anonymous type that contains a

// Pet's name and the name of the Person that owns the Pet.

var query =

people.Join(pets,

person => person,

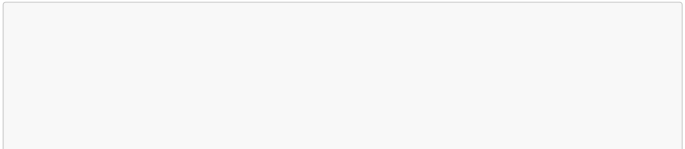
pet => pet.Owner,

(person, pet) =>

new { OwnerName = person.Name, Pet = pet.Name });

Let

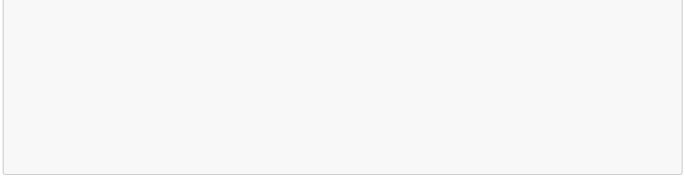
var earlyBirdQuery =



from sentence in strings

let words = sentence.Split(' ') from word in words

10\_ConsultasLINQ.md 1/21/2020



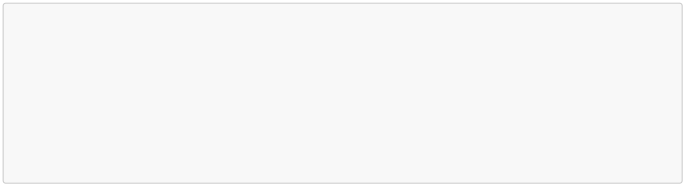
let w = word.ToLower()

where w[0] == 'a' || w[0] == 'e' || w[0] == 'i' || w[0] == 'o' || w[0] == 'u'

select word;

Where

var cust = new List<Customer>();



//queryAllCustomers is an IEnumerable<Customer>

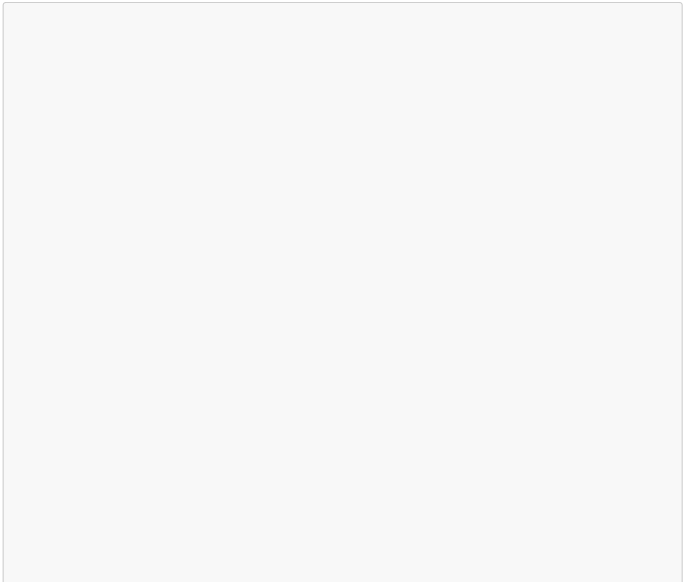
var queryLondonCustomers = from cust in customers

where cust.City == "London" select cust;

fruits.Where(fruit => fruit.Length < 6);

Group by

// queryCustomersByCity is an IEnumerable<IGrouping<string, Customer>> var queryCustomersByCity =



from cust in customers

group cust by cust.City;

// customerGroup is an IGrouping<string, Customer> foreach (var customerGroup in queryCustomersByCity) {

Console.WriteLine(customerGroup.Key);

foreach (Customer customer in customerGroup)

{

Console.WriteLine(" {0}", customer.Name); }

}

// Group the pets using Age as the key value

// and selecting only the pet's Name for each value. IEnumerable<IGrouping<int, string>> query =

pets.GroupBy(pet => pet.Age, pet => pet.Name);

// Iterate over each IGrouping in the collection. foreach (IGrouping<int, string> petGroup in query) {

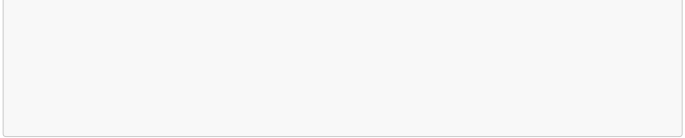
// Print the key value of the IGrouping.

Console.WriteLine(petGroup.Key);

// Iterate over each value in the

// IGrouping and print the value.

10\_ConsultasLINQ.md 1/21/2020



foreach (string name in petGroup)

Console.WriteLine(" {0}", name); }

Order by

var cust = new List<Customer>(); //queryAllCustomers is an IEnumerable<Customer> var queryLondonCustomers3 =



from cust in customers

where cust.City == "London"

orderby cust.Name ascending // descending select cust;

// ascending

IEnumerable<Pet> query = pets.OrderBy(pet => pet.Age);

// descending

IEnumerable<Pet> query = pets.OrderByDescending(pet => pet.Age);

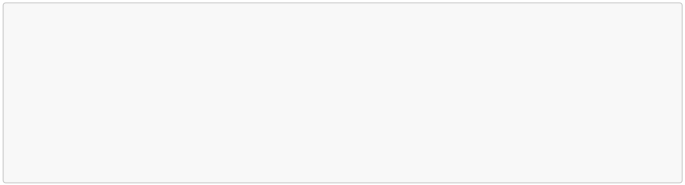
Las consultas de arriba tratan la variable con linq para realizar las consultas y dependiendo del tipo de consulta que sea, estos te devolveran objetos interfaz del tipo IEnumerable<T>, IQueryable<T>, etc.

Esto son objetos configurables para poder realizar consultas particionadas en memoria y en diferentes puntos, un ejemplo de uso, por ejemplo que no se muestren ciertos datos si no se cumple una condición especifica que hay que comprobar, o quieres reutilizar la misma consulta.

Para poder convertir definitivamente estos objetos en objetos en memoria definitivos y poder tratarlos, se deberá de usar los siguientes métodos después de la consulta.

ToList()

var cust = new List<Customer>(); //queryAllCustomers is an IEnumerable<Customer> var queryLondonCustomers3 =



(from cust in customers

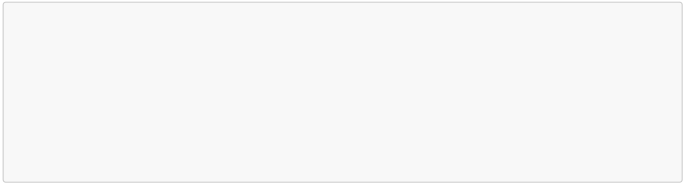
where cust.City == "London"

orderby cust.Name ascending

select cust).ToList();

ToArray()

var cust = new List<Customer>(); //queryAllCustomers is an IEnumerable<Customer> var queryLondonCustomers3 =



(from cust in customers

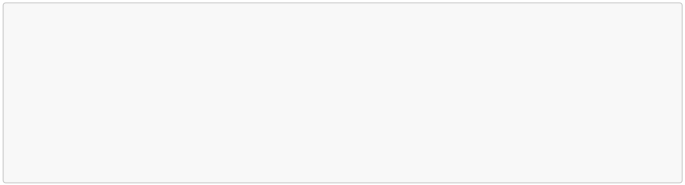
where cust.City == "London"

orderby cust.Name ascending

select cust).ToArray();

ToDictionary()

var cust = new List<Customer>(); //queryAllCustomers is an IEnumerable<Customer> var queryLondonCustomers3 =



(from cust in customers

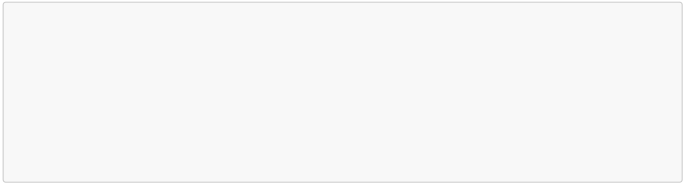
where cust.City == "London"

orderby cust.Name ascending

select cust).ToDictionary();

ToLookup()

var cust = new List<Customer>(); //queryAllCustomers is an IEnumerable<Customer> var queryLondonCustomers3 =



(from cust in customers

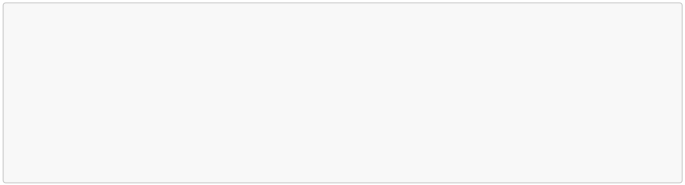
where cust.City == "London"

orderby cust.Name ascending

select cust).ToLookup();

Count()

var cust = new List<Customer>(); //queryAllCustomers is an IEnumerable<Customer> var queryLondonCustomers3 =



(from cust in customers

where cust.City == "London"

orderby cust.Name ascending

select cust).Count();

10\_ConsultasLINQ.md 1/21/2020

