LINQ and Entity Framework

By Eman Fathi

Course Content

- **Day1 :** ***LINQ***
- **Day2 :** ***Entity Framework (Basics in Designer)***
- **Day3 :** ***Entity Framework (CRUD operation)***
- **Day4 :** ***Entity Framework (Enum,Fun.& SP)***
- **Day5 :** ***Entity Framework (Code First)***

ENTITY FRAMEWORK

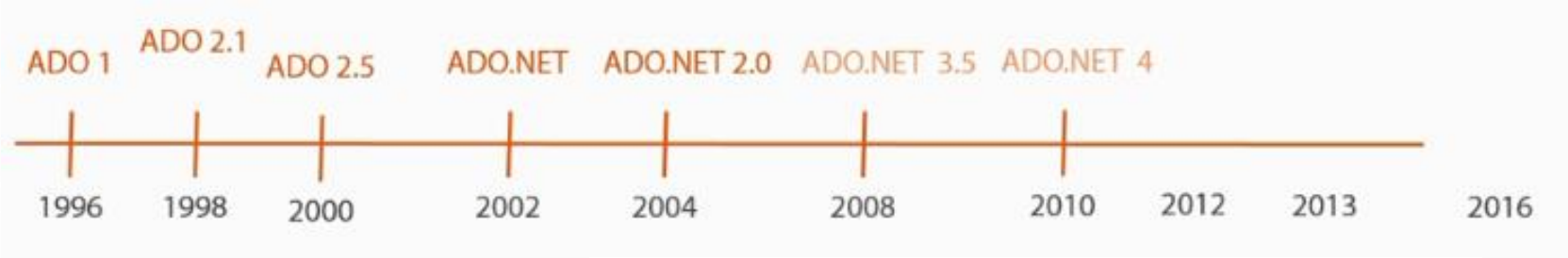
Focus on your domain and EF will take care of Database work for you

Why Entity Framework ?

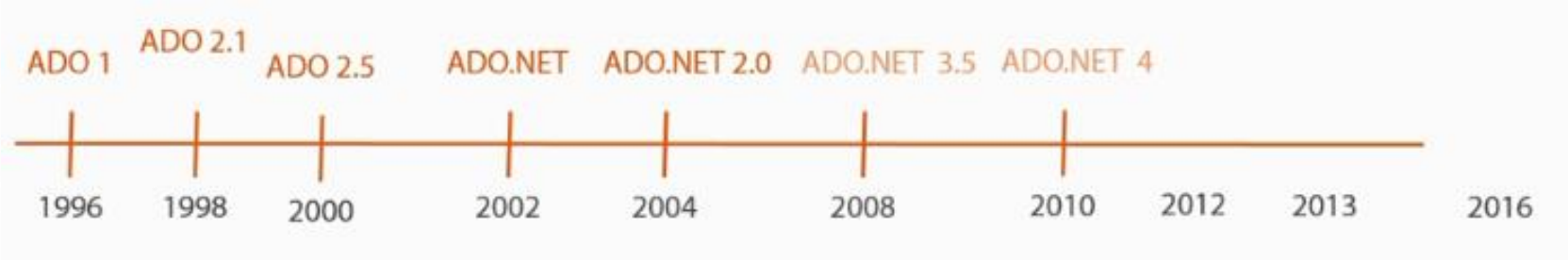
Data Base

Desktop Application

Web Application




```
string connectionString =
    "Data Source=(local);Initial Catalog=Northwind;"
    + "Integrated Security=true";

// Provide the query string with a parameter placeholder.
string queryString =
    "SELECT ProductID, UnitPrice, ProductName from dbo.products "
        + "WHERE UnitPrice > @pricePoint "
        + "ORDER BY UnitPrice DESC;";

// Specify the parameter value.
int paramValue = 5;

// Create and open the connection in a using block. This
// ensures that all resources will be closed and disposed
// when the code exits.
using (SqlConnection connection =
    new SqlConnection(connectionString))
{
    // Create the Command and Parameter objects.
    SqlCommand command = new SqlCommand(queryString, connection);
    command.Parameters.AddWithValue("@pricePoint", paramValue);

    // Open the connection in a try/catch block.
    // Create and execute the DataReader, writing the result
    // set to the console window.
    try
    {
        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("\t{0}\t{1}\t{2}",
                reader[0], reader[1], reader[2]);
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    Console.ReadLine();
}
```

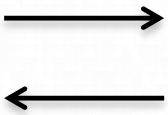


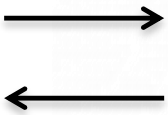
Microsoft .NET Entity Framework

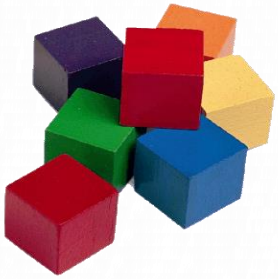
Focus on your domain and EF will take care of Database work for you

What Is Entity Framework

SQL







Classes

Object Relational Mapper

Categories	(Categories)
CategoriesID	int
CategoriesName	string
Description	string
Pictures	base64binary

Produc	
Produc	
ProductName	string
SupplierID	int
CategoryID	int
UnitPrice	decimal
UnitsInStock	short
UnitsOnOrder	short
Discontinued	boolean
EAN13	string

- Create connections
- Create commands for reading and writing data
- Execute queries for you on the database

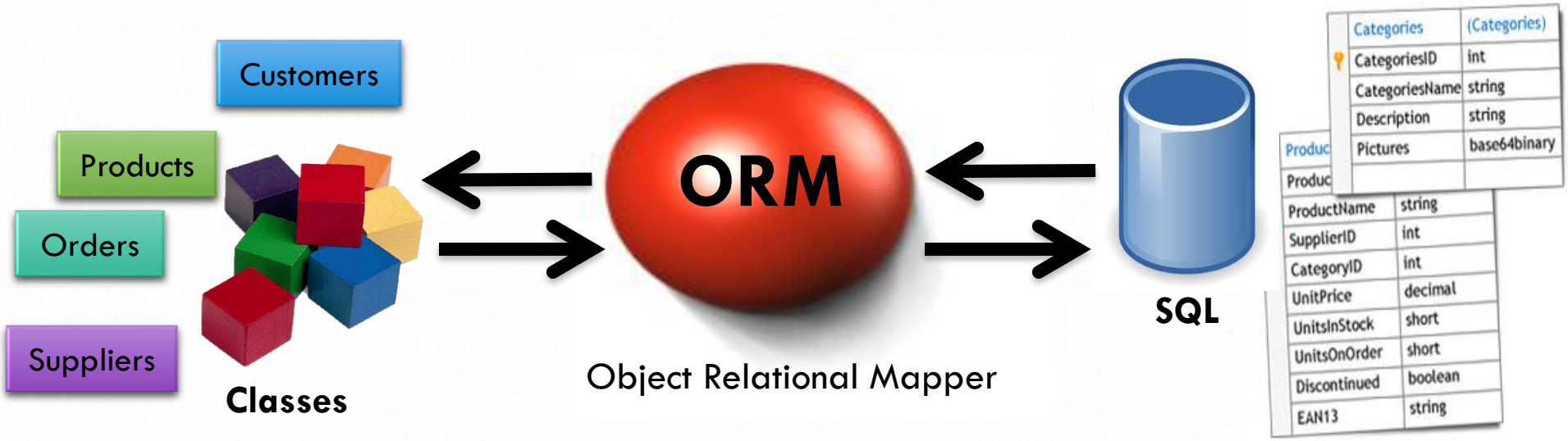
Products

Customers

Orders

Suppliers

What Is Entity Framework



Queries Entities (Objects) Using Linq to Entities

Translate and execute queries on DB

Return results as domain objects

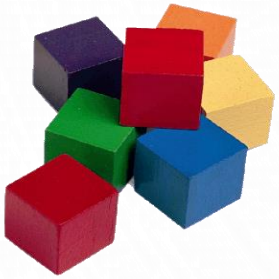
LINQ

Language Integrated Query

What Is LINQ

Microsoft introduces a query feature that offer a consistence API works on different data sources

SQL



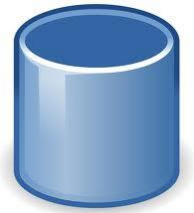
List of Objects

LINQ

XML

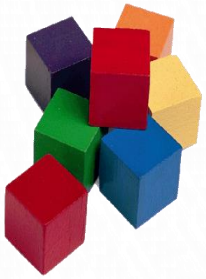
DataSet

What Is LINQ



SQL

```
var query = from c in customers
            where c.City == "Mansoura"
            select c.FirstName + " " + c.LastName;
```



List of Objects

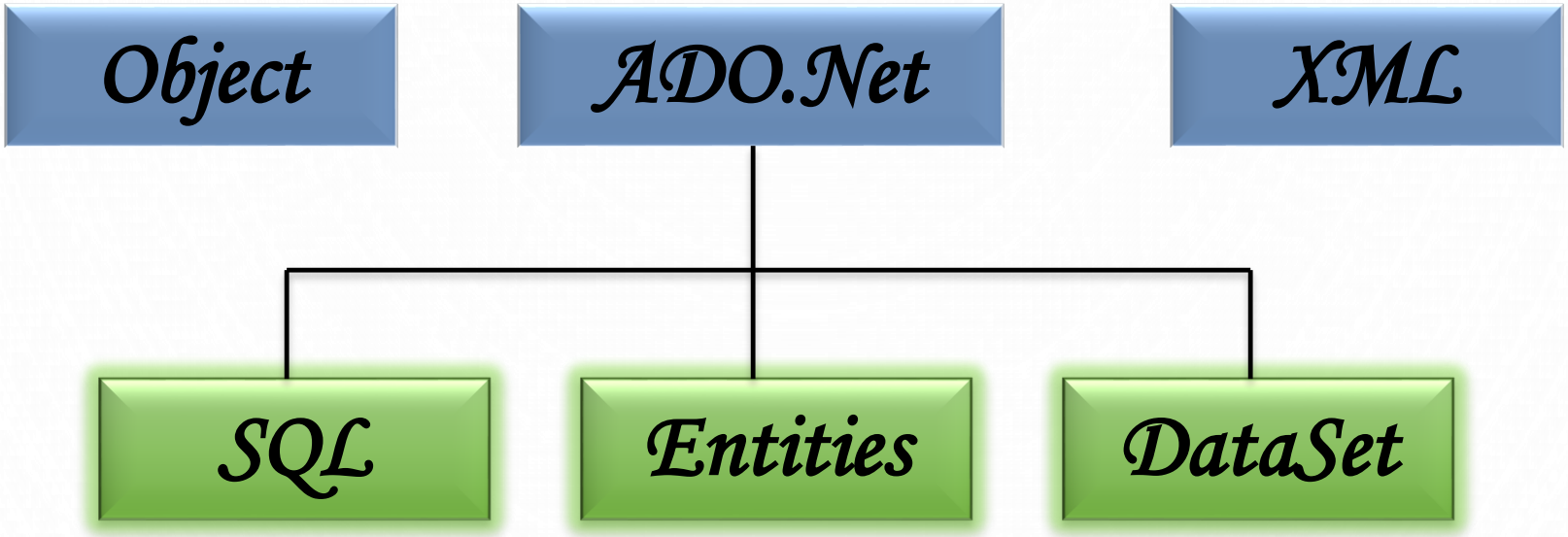
```
var query = customers.Where(c => c.City == "Mansoura")
                     .Select(c=>c.FirstName+ " "+c.LastName);
```

XML

DataSet

LINQ Areas

- MICROSOFT BASICALLY DIVIDES LINQ INTO THREE AREAS



How does LINQ Work?

Required Language Features

- *Implicitly Typed Local Variables*
- *Object Initializers*
- *Anonymous Type*
- *Anonymous fun. &Lambda Expression*
- *Extension Methods*

Implicitly Type Local Variable

var keyword

```
var number = 2;

number="any string";

number = null;

var number = null;
```

ERROR

Object Initializer

```
public class Customer
{
    4 references
    public string City { get; set; }
    6 references
    public string FirstName { get; set; }
    4 references
    public string LastName { get; set; }
    6 references
    public Order[] Orders { get; set; }
}
```

```
Customer c = new Customer() { FirstName = "Eman", LastName = "Fathi", City = "Mansoura" }
```

Anonymous Type

- ✓ Used for projection
- ✓ Read Only

```
var Student=new { FirstName="Mona" , LastName="Mohamed" };
```

var Keyword + new + object initializer

```
student.ID = 5;
```

ERROR

```
var customer = new { FullName=c.FirstName+" " +c.LastName , c.City };
```

ERROR

```
var customer = new { c.FirstName+" " +c.LastName , c.City };
```

Anonymous Function and Lambda Expression

```
List<string> Names = new List<string>() { "mona","Ali" ,"Nada", "noha", "anas", "abdullah", "mai" };
```

```
List<string> result=Names.FindAll();
```

List<string> List<string>.FindAll(Predicate<string> match)
Retrieves all the elements that match the conditions defined by the specified predicate.
match: The Predicate<in T> delegate that defines the conditions of the elements to search for.

```
public static bool match(string str)
{
    return str.Length > 3;
}
```

```
List<string> result=Names.FindAll(match);
```

Anonymous Function and Lambda Expression

Anonymous Fun.

```
List<string> result=Names.FindAll(delegate(string str)
{
    return str.Length > 3;
});
```

```
List<string> result=Names.FindAll((string str)=>
                                  {
                                      return str.Length > 3;
                                  });
```

Lambda Exp.

```
List<string> result=Names.FindAll(str=>str.Length>3);
```

Extension Methods

```
public static class StringExtenstions
{
    0 references
    public static int wordCount(     string input)
    {
        string[] userString = input.Split(new char[] {' ','?' },
                                StringSplitOptions.RemoveEmptyEntries);
        int wordCount = userString.Length; ;
        return wordCount;
    }
}
```

```
StringExtenstions.wordCount(name);
```

Extension Methods

```
public static class StringExtenstions
{
    0 references
    public static int wordCount(this string input)
    {
        string[] userString = input.Split(new char[] {' ','?' },
                                StringSplitOptions.RemoveEmptyEntries);
        int wordCount = userString.Length; ;
        return wordCount;
    }
}
```

Extension Methods

Class and method must be static

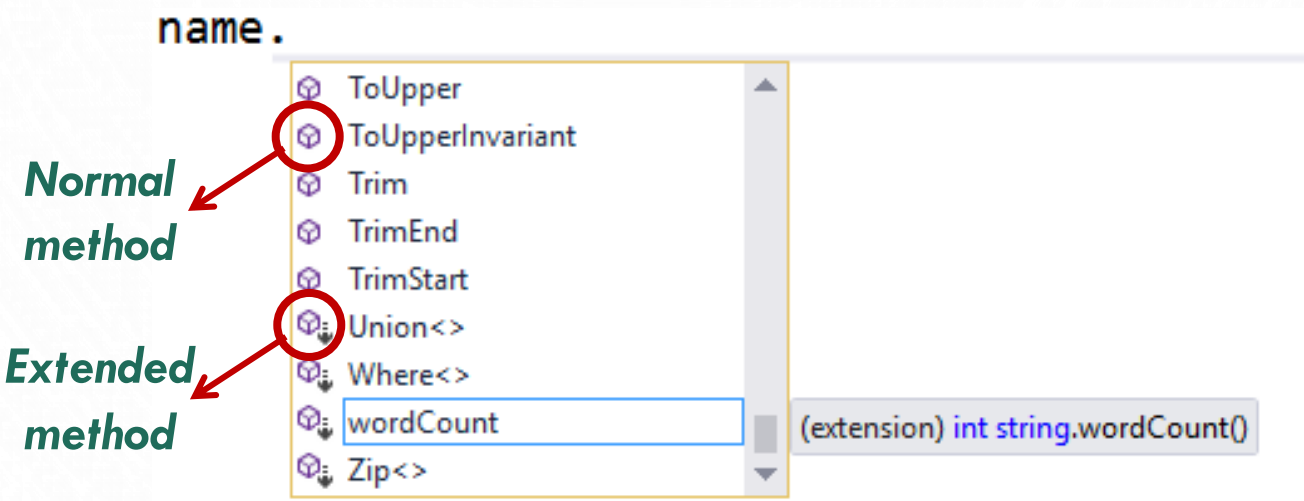
First parameter

Type to be Extended

```
public static class StringExtenstions
{
    0 references
    public static int wordCount(this string input)
    {
        string[] userString = input.Split(new char[] {' ','?' },
                                StringSplitOptions.RemoveEmptyEntries);
        int wordCount = userString.Length; ;
        return wordCount;
    }
}
```

Extension Methods

```
string name = "My Name is Eman ";

StringExtenstions.wordCount(name);

name.wordCount();
```



How Does C# Add Linq Functionality To C# Existing Classes

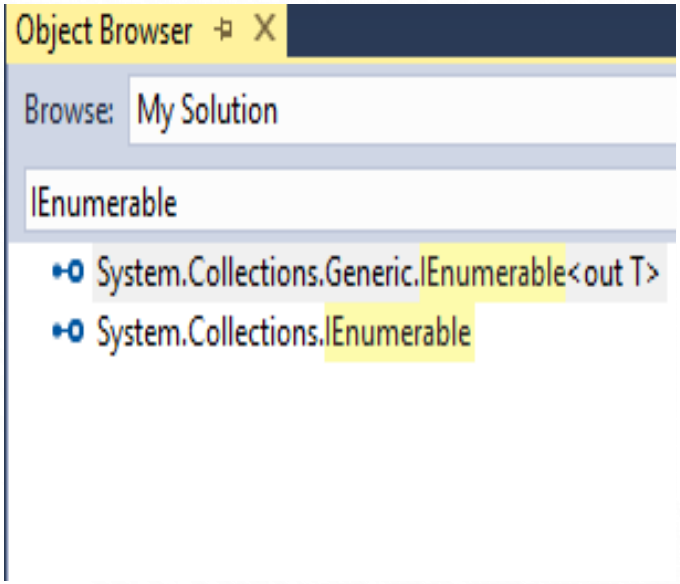
Enumerable [from metadata] | Program.cs* | app.config | DB.designer.cs | DB.dbml

System.Linq.Enumerable | Average(IEnumerable<decima

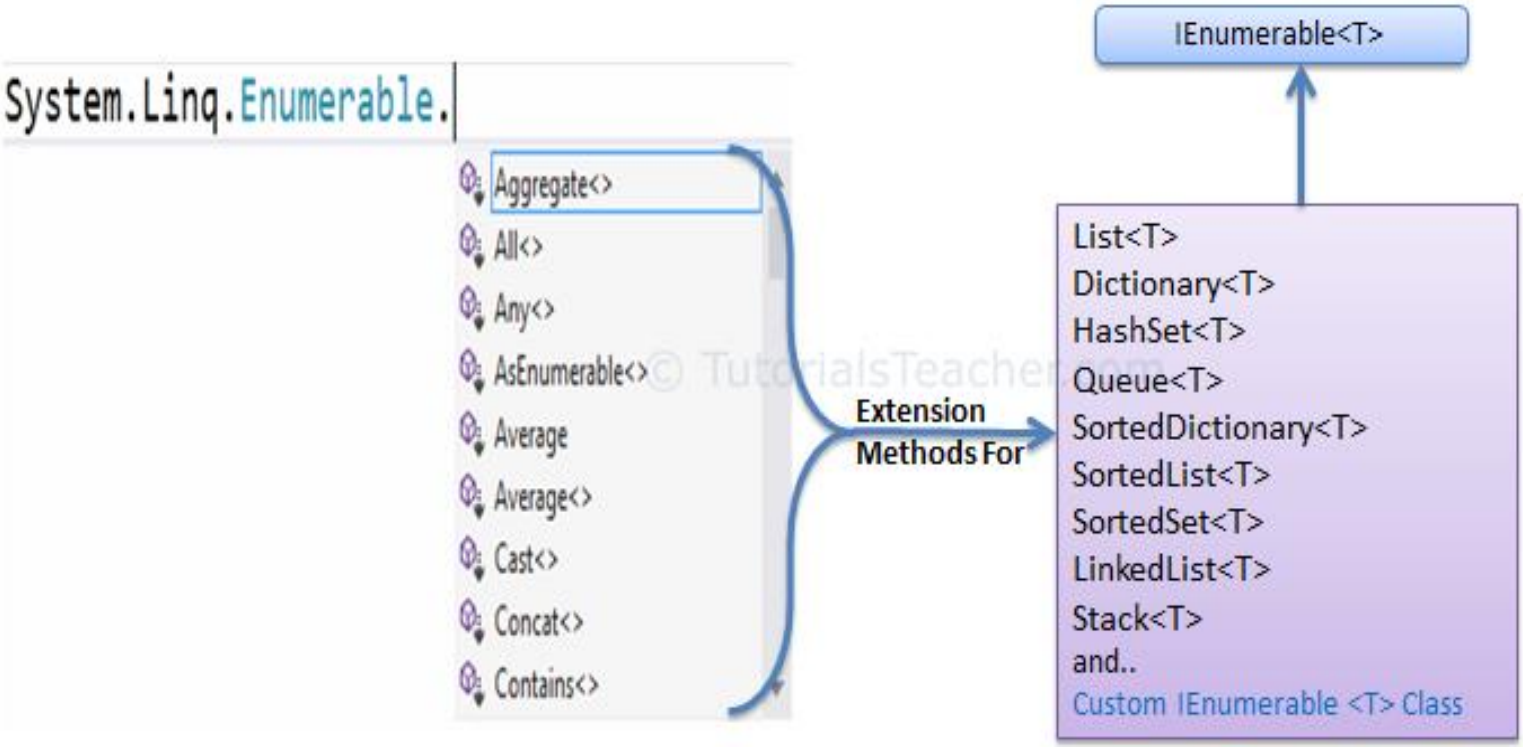
```
Assembly System.Core.dll, v2.0.50727

using System;
using System.Collections;
using System.Collections.Generic;
using System.Runtime.CompilerServices;

namespace System.Linq
{
    public static class Enumerable
    {
        public static TSource Aggregate<TSource>(this IEnumerable<TSource> source, Func<TSource, TSource, TSource>
        public static TAccumulate Aggregate<TSource, TAccumulate>(this IEnumerable<TSource> source, TAccumulate se
        public static TResult Aggregate<TSource, TAccumulate, TResult>(this IEnumerable<TSource> source, TAccumula
        public static bool All<TSource>(this IEnumerable<TSource> source, Func<TSource, bool> predicate);
        public static bool Any<TSource>(this IEnumerable<TSource> source);
        public static bool Any<TSource>(this IEnumerable<TSource> source, Func<TSource, bool> predicate);
        public static IEnumerable<TSource> AsEnumerable<TSource>(this IEnumerable<TSource> source);
        public static decimal? Average(this IEnumerable<decimal?> source);
        public static decimal Average(this IEnumerable<decimal> source);
        public static double? Average(this IEnumerable<double?> source);
        public static double Average(this IEnumerable<double> source);
        public static float? Average(this IEnumerable<float?> source);
        public static float Average(this IEnumerable<float> source);
        public static double? Average(this IEnumerable<int?> source);
        public static double Average(this IEnumerable<int> source);
        public static double? Average(this IEnumerable<long?> source);
        public static double Average(this IEnumerable<long> source);
        public static decimal? Average<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal?> selector
        public static decimal Average<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal> selector);
        public static double? Average<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector);
        public static double Average<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector);
        public static float? Average<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector);
```

- ✓ applied on classes that's implement **IENUMERABLE<T>** interface
- ✓ applied on any collection that contains the same datatype

Like T [ ] and Generic Collections




```
public interface IEnumerable<out T> : IEnumerable
{
    new IEnumerator<T> GetEnumerator();
}
public interface IEnumerator<out T> : IDisposable, IEnumerator
{
    new T Current { get; }
    bool MoveNext();
}
```


System.Linq.Enumerable.
Aggregate<>
All<>
Any<>
AsEnumerable<>
Average
Average<>
Cast<>
Concat<>
Contains<>
Extension Methods For
IEnumerable<T>
List<T>
Dictionary<T>
HashSet<T>
Queue<T>
SortedDictionary<T>
SortedList<T>
SortedSet<T>
LinkedList<T>
Stack<T>
and..
Custom IEnumerable <T> Class

```
var result =
    Customers
    .Where( c => c.City == "Boston" )
    .Select( c => new { c.Name, c.Address } );
```

Local variable type inference

Lambda expressions

Extension methods

Anonymous types

Object initializers