



2016

# LINQ

## STEP BY STEP

reference with C#

**TRƯỜNG MINH TUẤN**

a developer, a writer, a speaker

<http://truongminhtuan.info>

[ifsoft@live.com](mailto:ifsoft@live.com)

## MỤC LỤC

1. MỞ ĐẦU.....	3
2. LINQ LÀ GÌ? .....	4
3. KIẾN TRÚC VÀ CÁC THÀNH PHẦN CỦA LINQ .....	6
4. LINQ QUERIES (C#) .....	7
5. LINQ VÀ GENERIC TYPE.....	10
6. LINQ QUERY OPERATION.....	11
7. LINQ to SQL .....	29
8. LINQ to Entities .....	39
9. LINQ to Object .....	47
10. TÀI LIỆU THAM KHẢO .....	51

## 1. MỞ ĐẦU

*Chào bạn..!*

Lời đầu tiên, tôi gửi lời cảm ơn bạn đã ủng hộ quyển tài liệu LINQ STEP by STEP, cũng như gửi lời tri ân sâu sắc nhất đến với quý bạn đọc vì trong thời gian qua đã ủng hộ các bài viết trên Blog <http://truongminhtuan.info> của tôi.

Quyển tài liệu lần này, tôi gửi đến bạn đọc những bài viết cơ bản nhất về truy vấn LINQ, nó phù hợp cho những bạn sinh viên, học sinh đang theo học ngành CNTT, đặc biệt là những bạn đang có ý định đi theo nghiệp lập trình phần mềm, và những anh chị đang có ý định tự học lập trình theo công nghệ của Microsoft.

Có thể nói LINQ là nền tảng cơ bản nhất khi bạn có ý định học phát triển phần mềm trên nền tảng Internet (website) hoặc Application (Form) của Microsoft, nhưng trước tiên bạn hãy cho mình cơ hội học những ngôn ngữ lập trình như: C, C++, C#, VB.NET – hiện nay C# được sử dụng nhiều hơn VB.NET.

Trong năm 2016, Blog của tôi hướng đến các chủ đề rất nóng hổi hiện nay như:

- Web API, MVC
- Bootstrap, Json
- Node.JS, MongoDB

Tôi rất mong nhận được sự góp ý chân thành của quý bạn đọc để tôi từng bước hoàn thiện hơn về những bài viết sắp tới cũng như những quyển tài liệu dạng *step by step* được ra đời trong thời gian tới.

*Chân thành cảm ơn quý bạn đọc..!*

Trân trọng!

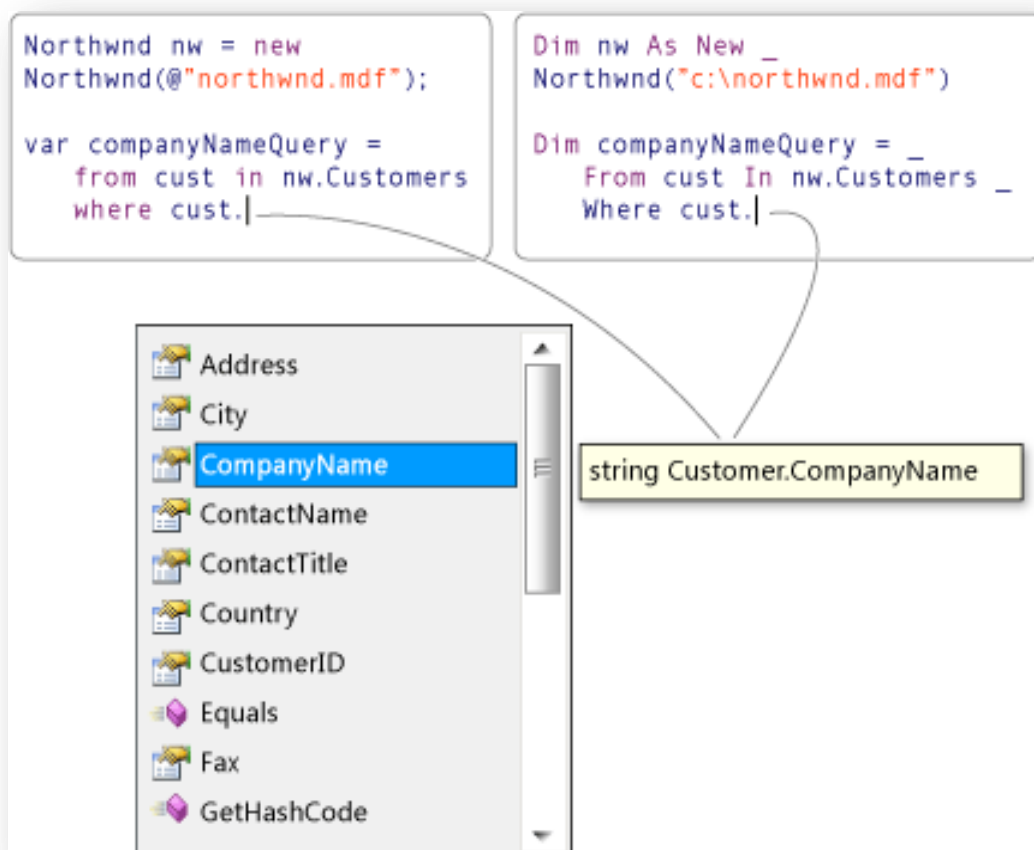
Trương Minh Tuấn

## 2. LINQ LÀ GÌ?

LINQ (Language Integrated Query) là một sự đổi mới và tập hợp các tính năng, được giới thiệu trong bộ Visual Studio 2008 và .NET Framework 3.5, LINQ giúp khả năng truy vấn trở nên mạnh mẽ hơn với cú pháp ngôn ngữ C# và Visual Basic.

Trước đây, lập trình viên muốn truy vấn đến cơ sở dữ liệu thì việc đầu tiên là phải có chuỗi kết nối (ConnectionString); sau đó, lập trình viên phải biết thêm một ngôn ngữ truy vấn cho mỗi loại cơ sở dữ liệu như: SQL Server, XML, Web Service,....

Và bây giờ, những việc đó, lập trình viên không cần phải tốn nhiều thời gian để tìm tòi và học hỏi. LINQ có đầy đủ tính năng mới, cho phép truy vấn dữ liệu trở nên dễ dàng hơn.



(Nguồn Microsoft)

LINQ thì mềm dẻo, dễ học, cho phép việc truy vấn, thao tác với bất kỳ loại dữ liệu nào.

Trong Visual Studio, lập trình viên sử dụng truy vấn LINQ trong C#, Visual Basic, với SQL Server databases, XML documents, ADO.NET datasets, và bất kỳ đối tượng nào có hỗ trợ IEnumerable hoặc IEnumerable<T> interface.

Việc sử dụng LINQ trong dự án hiện nay là rất cần thiết, nhưng phải dùng .NET Framework 3.5 trở lên. Tuy nhiên, bạn cũng có thể không dùng truy vấn LINQ trong dự án, việc này phụ thuộc vào từng công ty, doanh nghiệp mà bạn đang làm việc.

### **Namespace hỗ trợ của LINQ:**

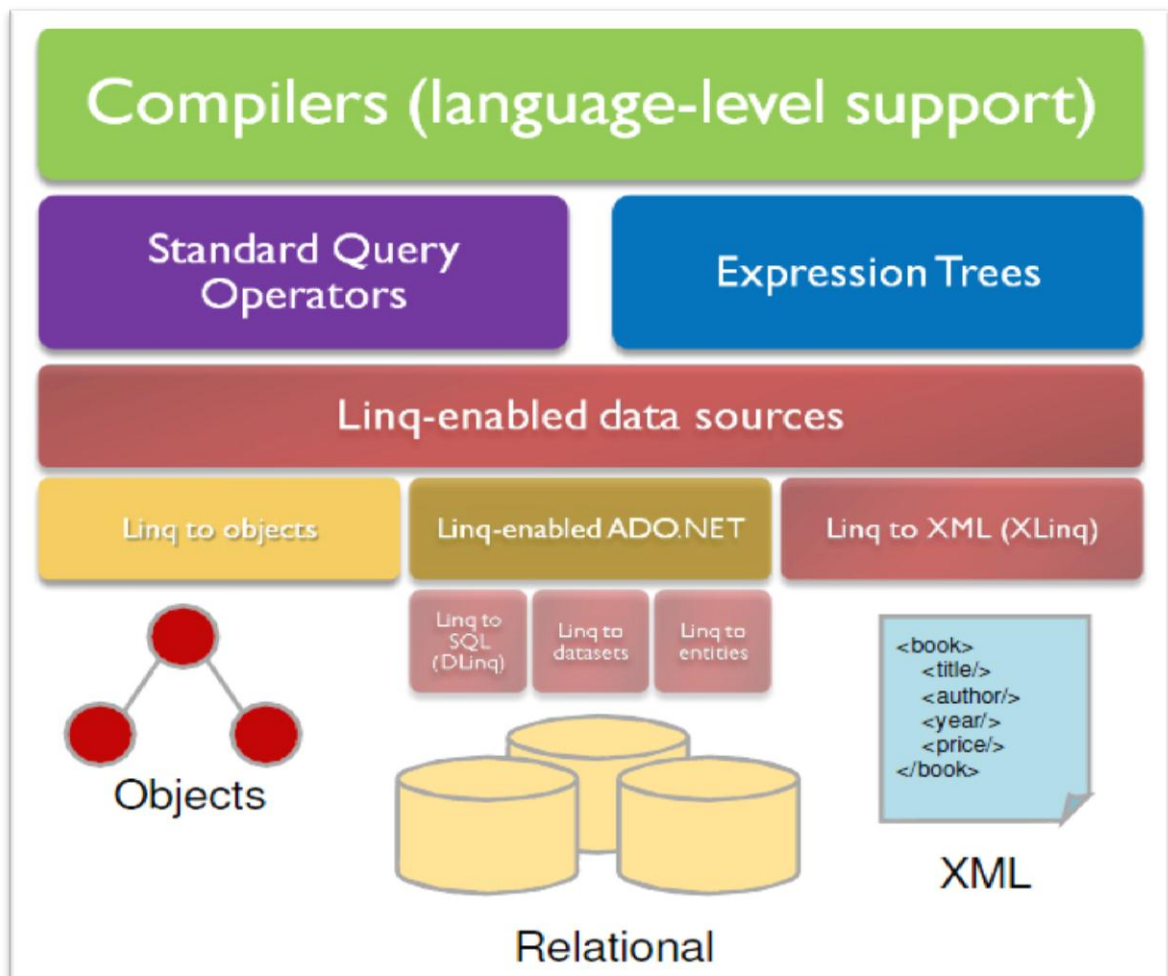
**System.Linq:** hỗ trợ sử dụng các Objects;

**System.Data.Linq:** hỗ trợ sử dụng các cơ sở dữ liệu quan hệ;

**System.Data.Objects:** hỗ trợ sử dụng các Entities;

**System.XML.Linq:** hỗ trợ sử dụng XML;

### 3. KIẾN TRÚC VÀ CÁC THÀNH PHẦN CỦA LINQ



**Level 1 (Compilers):** Hỗ trợ ngôn ngữ như C#, VB.NET,...

**Level 2 (Standard Query Operators):** Hỗ trợ các toán tử (from, where, order by, select) được dùng trong truy vấn LINQ

Expression Trees: Hỗ trợ việc nối các toán tử trước khi thực hiện truy vấn LINQ

**Level 3 (Linq-enabled data sources):** Tầng trung gian, hỗ trợ khi làm việc với các đối tượng như Linq to Objects (Objects), ADO.NET (Linq to SQL, Linq to Dataset, Linq to Entities), LINQ to XML (X.Linq)

Với kiến trúc trên thì việc LINQ hỗ trợ SQL Server là hiển nhiên, nhưng khi bạn làm việc với các hệ quản trị cơ sở dữ liệu khác như MySQL, Oracle thì để làm việc với LINQ phải thông qua công nghệ ADO.NET và dùng Linq to Dataset để thực hiện truy vấn.

**Cú pháp LINQ:**

**from** name **in** source  
**where** condition  
**orderby** ordering, ..

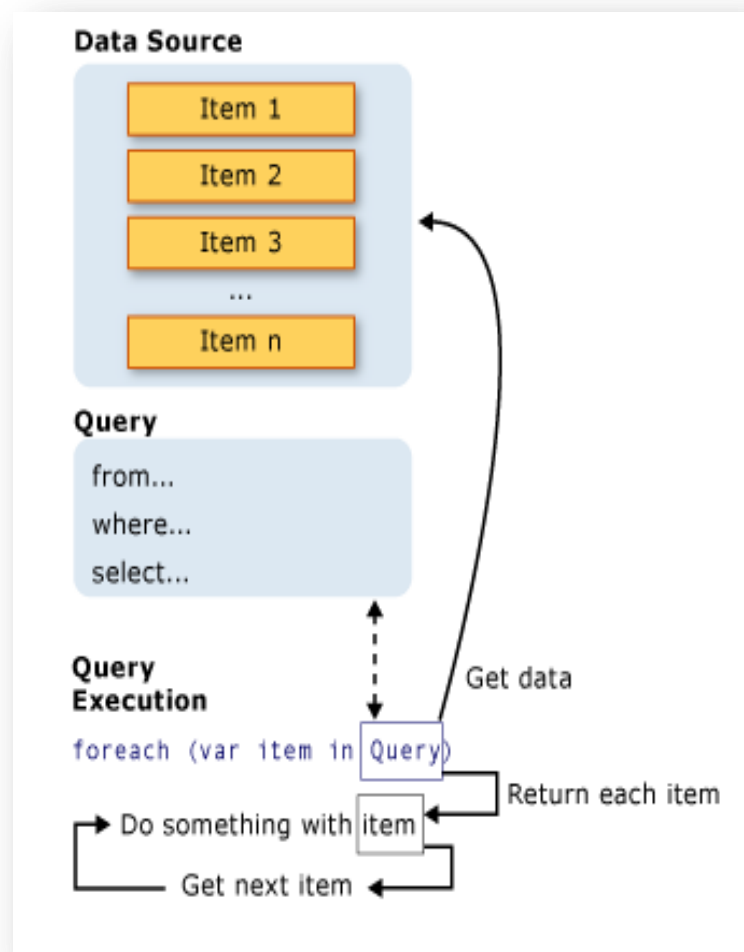
## 4. LINQ QUERIES (C#)

Thông thường, lập trình viên làm việc với cơ sở dữ liệu thông qua các truy vấn Query; vì thế, lập trình viên phải học cú pháp truy vấn cho từng loại cơ sở dữ liệu khác nhau, như: SQL Server, MySQL, Oracle, XML. Lý do, mỗi hệ quản trị cơ sở dữ liệu đều do các công ty khác phát triển nên trên cơ bản cú pháp đều giống nhau nhưng có vài điểm khác nhau để phù hợp.

LINQ đơn giản hoá các vấn đề trên bằng cách cung cấp mô hình phù hợp để làm việc trên tất cả nguồn dữ liệu khác nhau, vì thế trong LINQ, lập trình viên đều làm việc trên các đối tượng.

### Hoạt động LINQ: 3 phần

- Obtain the data source.
- Create the query.
- Execute the query.



Một ví dụ để tìm hiểu rõ hơn về truy vấn LINQ

```
class IntroToLINQ
{
    static void Main()
    {
        // The Three Parts of a LINQ Query:
        // 1. Data source.
        int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

        // 2. Query creation.
        // numQuery is an IEnumerable<int>
        var numQuery =
            from num in numbers
            where (num % 2) == 0
            select num;

        // 3. Query execution.
        foreach (int num in numQuery)
        {
            Console.WriteLine("{0,1} ", num);
        }
    }
}
```

### Nguồn dữ liệu:

Dữ liệu của tôi là một mảng, nó hoàn toàn hỗ trợ đặc điểm của IEnumerable<T> interface. Điều này có nghĩa là nó sử dụng LINQ để truy vấn. Và truy vấn được thực hiện với câu lệnh foreach. Loại có hỗ trợ IEnumerable<T> interface, IQueryable<T> interface được gọi là các loại Queryable.

```
foreach (int num in numQuery)
{
    Console.WriteLine("{0,1} ", num);
}
```

### Truy vấn:

Trong ví dụ trên, truy vấn LINQ trả về tất cả các số chia hết cho 2 từ mảng số nguyên. Các biểu thức truy vấn chứa ba mệnh đề: from, where, select.

- Mệnh đề from dùng để xác định nguồn dữ liệu.
- Mệnh đề where dùng để lọc dữ liệu.
- Mệnh đề select dùng để chọn ra những phần tử được trả về.



LINQ có 2 dạng cú pháp: **Query syntax** và **Method syntax**

```
namespace ProjectLINQ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] a = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

            //Query syntax
            IEnumerable<int> query = from _so in a
                                    where _so % 2 == 0
                                    select _so;
            foreach (var item in query)
            {
                Console.WriteLine(item);
            }

            //Method syntax
            IEnumerable<int> query2 = a.Where(w => w % 2 == 0).Select(s => s);
            foreach (var item in query2)
            {
                Console.WriteLine(item);
            }
            Console.ReadLine();
        }
    }
}
```

## 5. LINQ VÀ GENERIC TYPE

- LINQ query đều dựa trên Generic Type, được giới thiệu trong .NET Framework 2.0;
- Giá trị trả về của LINQ là **IEnumerable<T>** và **IQueryable<T>** và <T> là một đối tượng do lập trình viên đưa vào. Khi đó, lập trình viên dùng câu lệnh foreach để duyệt trong IEnumerable<T>
- Ngoài ra, bạn cũng có thể dùng từ khoá **var** để thay thế các Generic, và hiện nay lập trình viên đều sử dụng từ khoá này trong khi lập trình;
- Ví dụ bên dưới cho thấy hoạt động của Generic Type

```
class Program
{
    static void Main(string[] args)
    {
        Ví dụ 1: LINQ cơ bản

        #region LINQ and Generic type

        List<HocSinh> hs = new List<HocSinh>() {
            new HocSinh{IDHS=1,TenHS="Ty",TuoiHS=19},
            new HocSinh{IDHS=2,TenHS="Suu",TuoiHS=18},
            new HocSinh{IDHS=3,TenHS="Dan",TuoiHS=21},
            new HocSinh{IDHS=4,TenHS="Meo",TuoiHS=19},
            new HocSinh{IDHS=5,TenHS="Thin",TuoiHS=22},
            new HocSinh{IDHS=6,TenHS="Ty",TuoiHS=20},
            new HocSinh{IDHS=7,TenHS="Ngo",TuoiHS=21},
            new HocSinh{IDHS=8,TenHS="Mui",TuoiHS=22},
            new HocSinh{IDHS=9,TenHS="Than",TuoiHS=18},
            new HocSinh{IDHS=10,TenHS="Dau",TuoiHS=17},
            new HocSinh{IDHS=11,TenHS="Tuat",TuoiHS=23},
            new HocSinh{IDHS=12,TenHS="Hoi",TuoiHS=18},
        };

        IEnumerable<HocSinh> query = from a in hs
                                     select a;

        foreach (var item in query)
        {
            Console.WriteLine("IDHS: " + item.IDHS + " TenHS: " + item.TenHS + " TuoiHS: " + item.TuoiHS);
        }
        Console.ReadLine();
        #endregion
    }
}
```

## 6. LINQ QUERY OPERATION

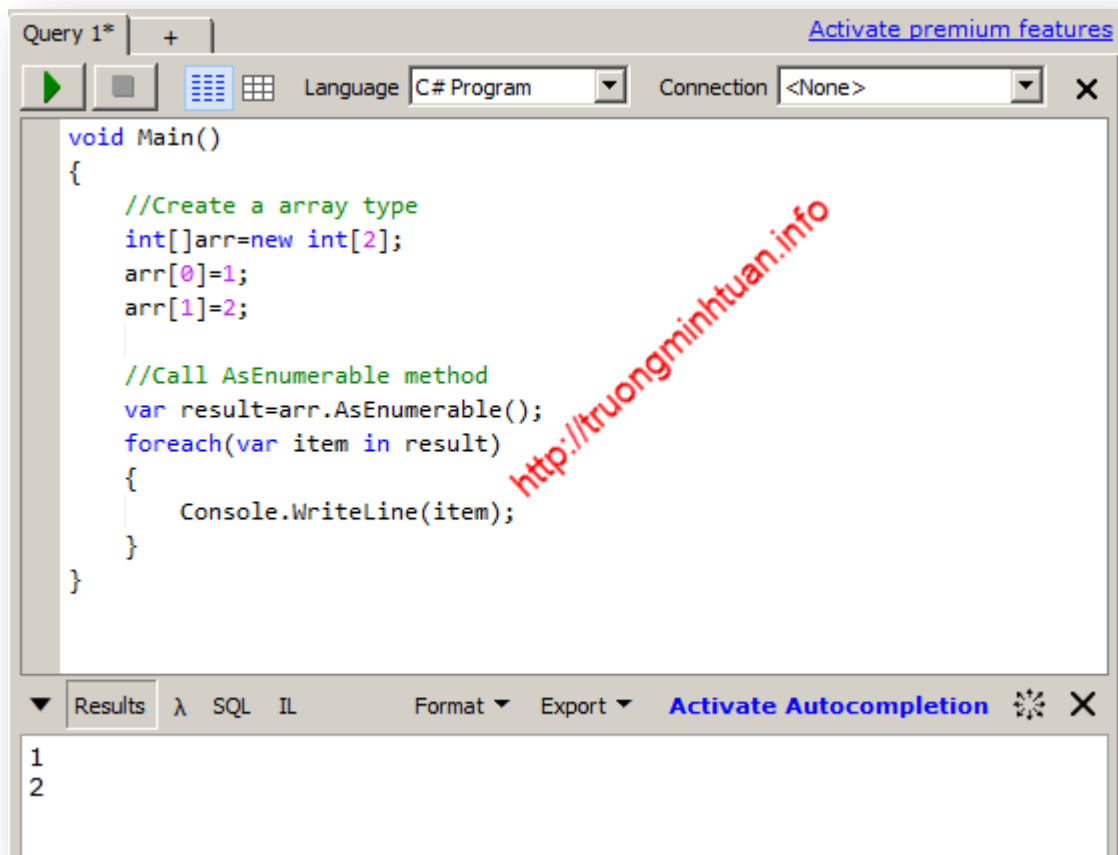
Dưới đây, tôi trình bày hàng loạt truy vấn LINQ với các từ khoá from, where, select, order by, distinct, join, group, min, max, count,... Như bạn đã biết, những từ khoá trên đều có trong truy vấn SQL của mỗi loại hệ quản trị cơ sở dữ liệu khác nhau.

Nếu như bạn đã làm quen với truy vấn SQL thì việc học qua LINQ query hoàn toàn dễ dàng, ngược lại, bạn nên nghiên cứu thêm về truy vấn SQL để có nền tảng trước khi tìm hiểu về LINQ query này.

Ngoài ra, bạn có thể tìm hiểu về SQL Server, cụ thể hơn là truy vấn SQL từ trang Blog cá nhân của tôi tại địa chỉ: <http://truongminhtuan.info/Books>

Các ví dụ bên dưới tôi dùng LINQPad để thực hiện, bạn cũng có thể dùng Visual Studio 2010 hoặc cao hơn để tìm hiểu từng từ khoá của LINQ.

**AsEnumerable:** IEnumerable, IList, IEnumerator là interface của LINQ, vì thế AsEnumerable method cho phép triển khai nhiều interface như thế.



The screenshot shows a C# IDE window titled "Query 1\*" with a toolbar and a dropdown menu for "Language" set to "C# Program". The code in the editor is as follows:

```
void Main()
{
    //Create a array type
    int[] arr=new int[2];
    arr[0]=1;
    arr[1]=2;

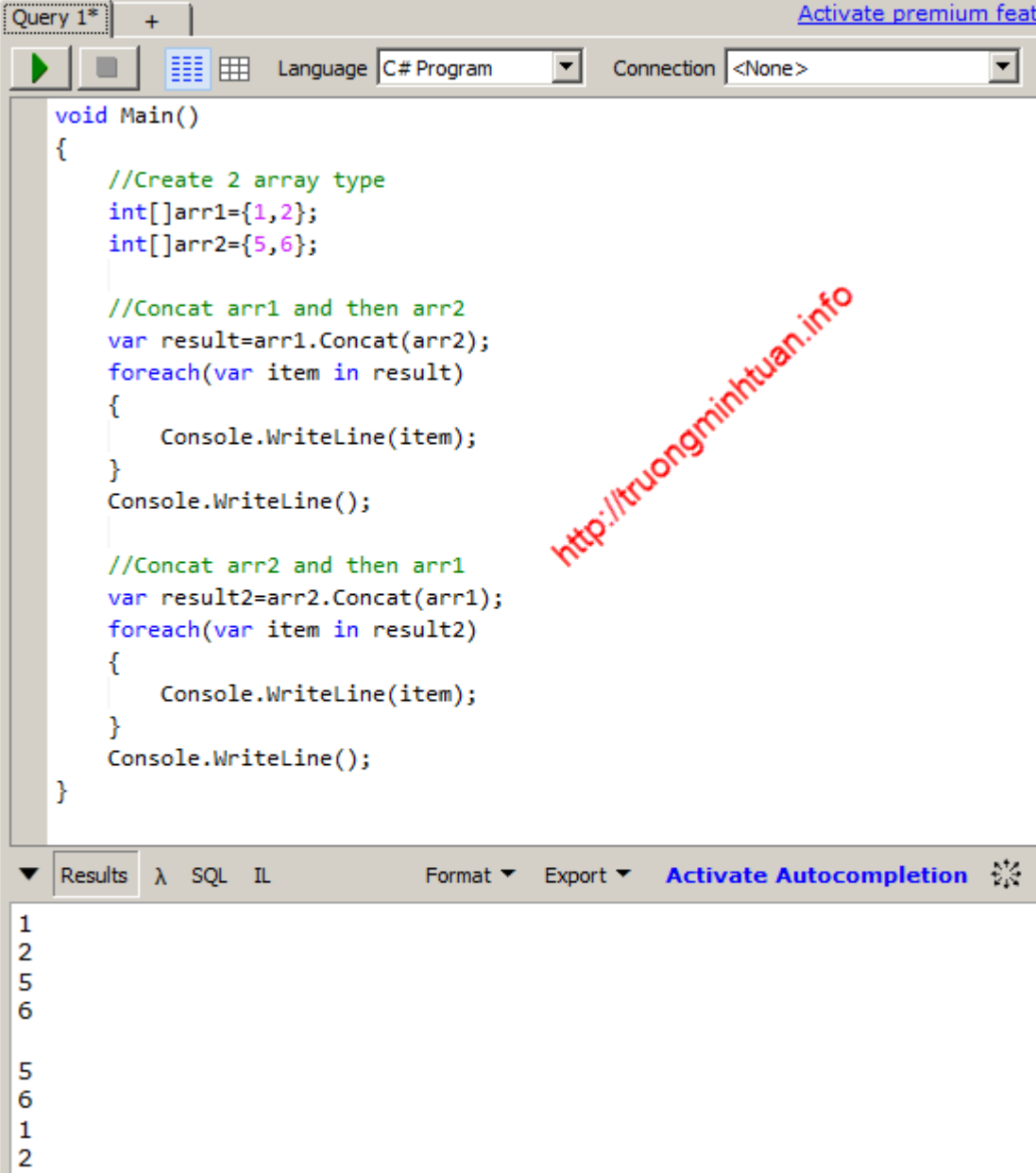
    //Call AsEnumerable method
    var result=arr.AsEnumerable();
    foreach(var item in result)
    {
        Console.WriteLine(item);
    }
}
```

The output window at the bottom shows the results of the program execution:

```
1
2
```

A red watermark "http://truongminhtuan.info" is visible diagonally across the code editor.

**Concat:** là một extension method, và nó làm việc trên cả 2 IEnumerable collections. Kết quả trả về là một collection với tất cả elements.



```
void Main()
{
    //Create 2 array type
    int[] arr1={1,2};
    int[] arr2={5,6};

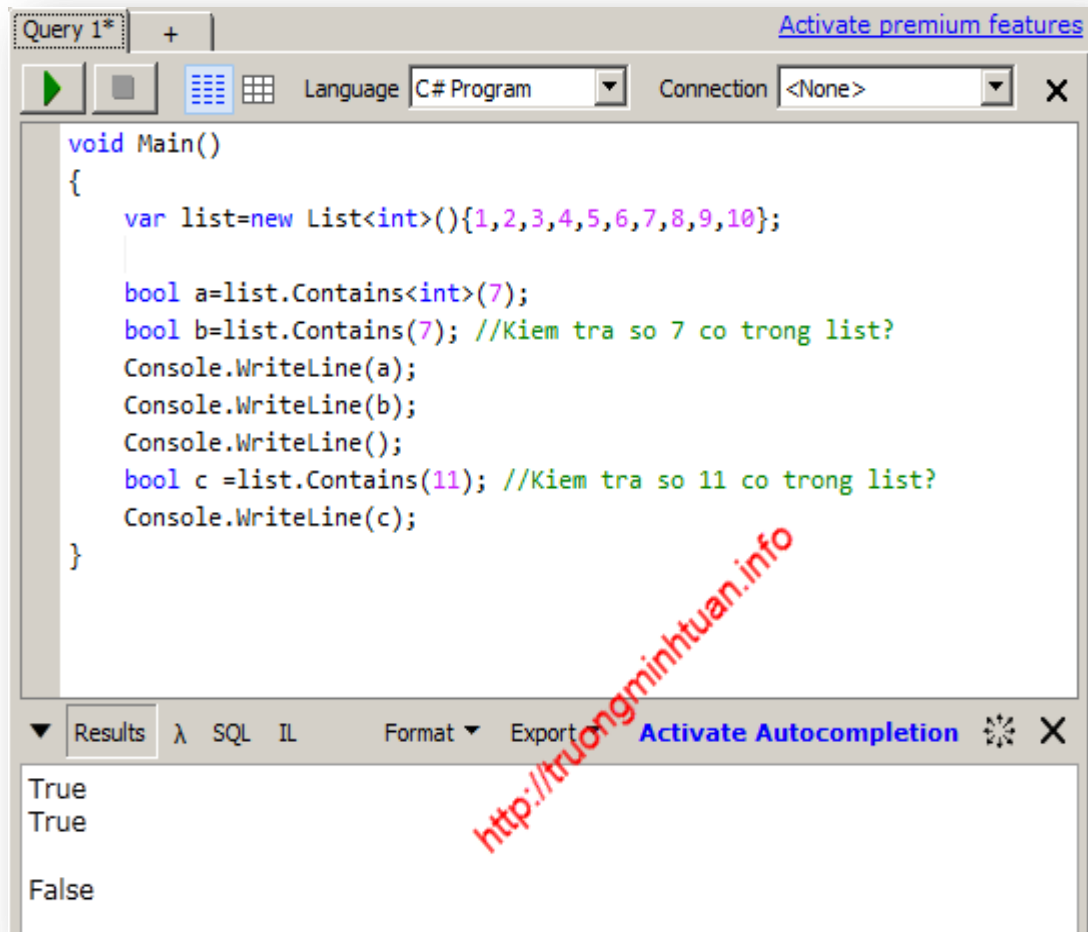
    //Concat arr1 and then arr2
    var result=arr1.Concat(arr2);
    foreach(var item in result)
    {
        Console.WriteLine(item);
    }
    Console.WriteLine();

    //Concat arr2 and then arr1
    var result2=arr2.Concat(arr1);
    foreach(var item in result2)
    {
        Console.WriteLine(item);
    }
    Console.WriteLine();
}
```

Results

1  
2  
5  
6  
  
5  
6  
  
1  
2

**Contains:** trả về TRUE hoặc FALSE. Nó kiểm tra một element có nằm trong IEnumerable collections không. Nếu có thì giá trị trả về là TRUE ngược lại là FALSE.



The screenshot shows a query editor window titled "Query 1\*" with a toolbar at the top containing a green play button, a grey square, a blue grid icon, and a grid icon. The "Language" dropdown is set to "C# Program" and the "Connection" dropdown is set to "<None>". The main text area contains the following C# code:

```
void Main()
{
    var list=new List<int>(){1,2,3,4,5,6,7,8,9,10};

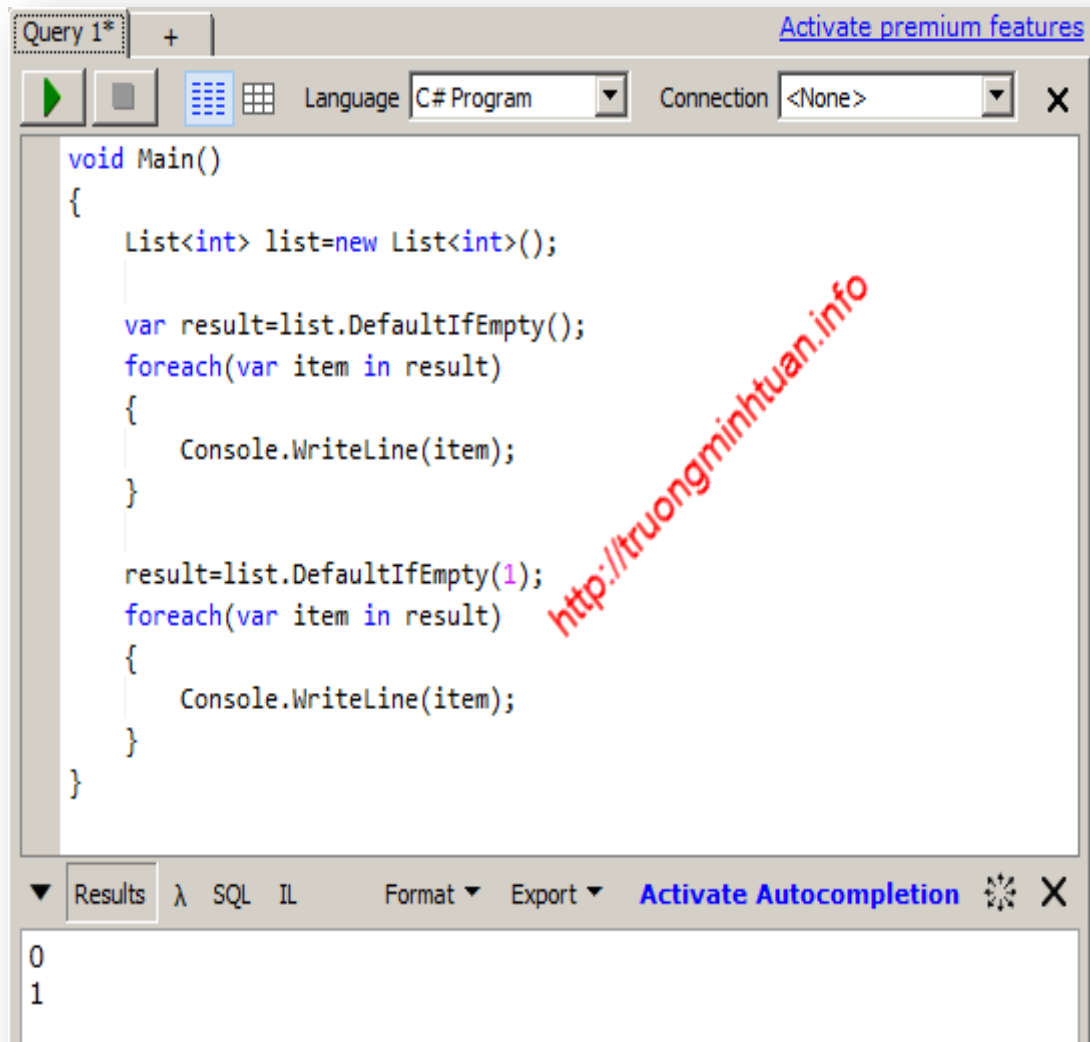
    bool a=list.Contains<int>(7);
    bool b=list.Contains(7); //Kiem tra so 7 co trong list?
    Console.WriteLine(a);
    Console.WriteLine(b);
    Console.WriteLine();
    bool c =list.Contains(11); //Kiem tra so 11 co trong list?
    Console.WriteLine(c);
}
```

The bottom of the window has a toolbar with a dropdown menu showing "Results", "SQL", and "IL". Other buttons include "Format", "Export", "Activate Autocompletion", and a close button. The "Results" pane at the bottom displays the output of the program:

```
True
True
False
```

A red diagonal watermark "http://truongminhtuan.info" is visible across the bottom half of the image.

**DefaultIfEmpty:** xử lý empty collections theo cách riêng. Nó trả về giá trị Default, không lỗi. Ví dụ, bạn khai báo một List<int> nhưng không khai báo các element cho nó thì việc sử dụng DefaultIfEmpty trở nên hữu dụng trong trường hợp;



The screenshot shows a query editor window titled "Query 1\*" with a toolbar at the top containing a green run button, a grey stop button, a grid icon, and a dropdown menu for "Language" set to "C# Program". To the right of the language dropdown is a "Connection" dropdown set to "<None>" and a close button "X". The main area contains C# code:

```
void Main()
{
    List<int> list=new List<int>();

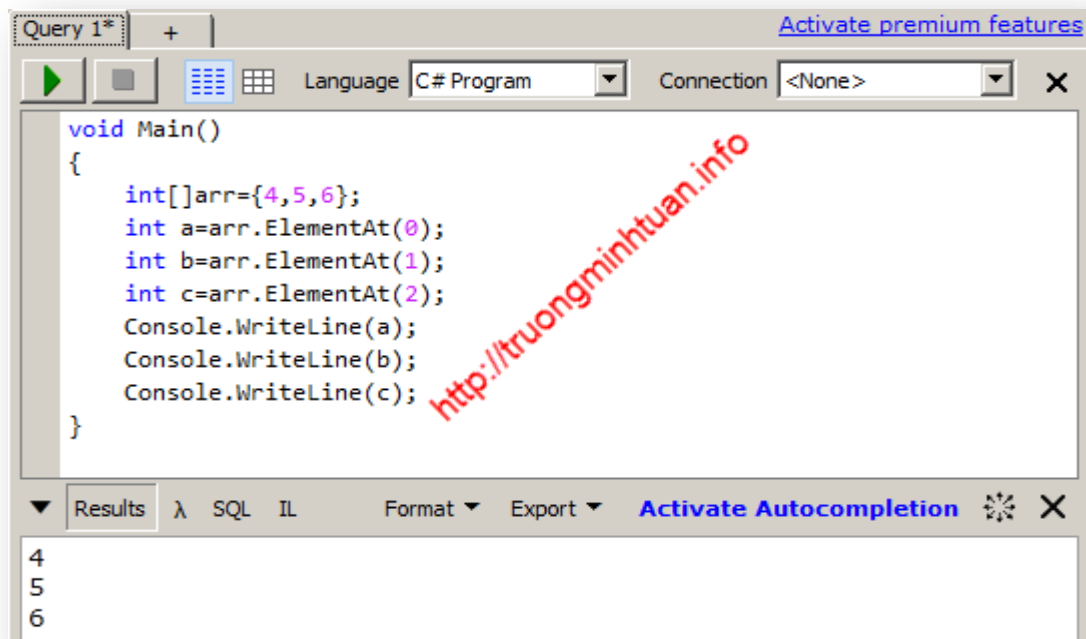
    var result=list.DefaultIfEmpty();
    foreach(var item in result)
    {
        Console.WriteLine(item);
    }

    result=list.DefaultIfEmpty(1);
    foreach(var item in result)
    {
        Console.WriteLine(item);
    }
}
```

A red diagonal watermark "http://truongminhtuan.info" is visible across the code. At the bottom of the editor is a toolbar with a dropdown menu currently showing "Results", followed by "SQL", "IL", "Format", "Export", and a link to "Activate Autocompletion" with a star icon and a close button "X". Below the code editor, a results pane displays the output of the program:

```
0
1
```

**ElementAt:** giúp cho việc truy cập đến một element và trả về giá trị ứng với vị trí đó. Các phần tử element phải tương ứng với vị trí của nó, ngược lại **Index was out of range**

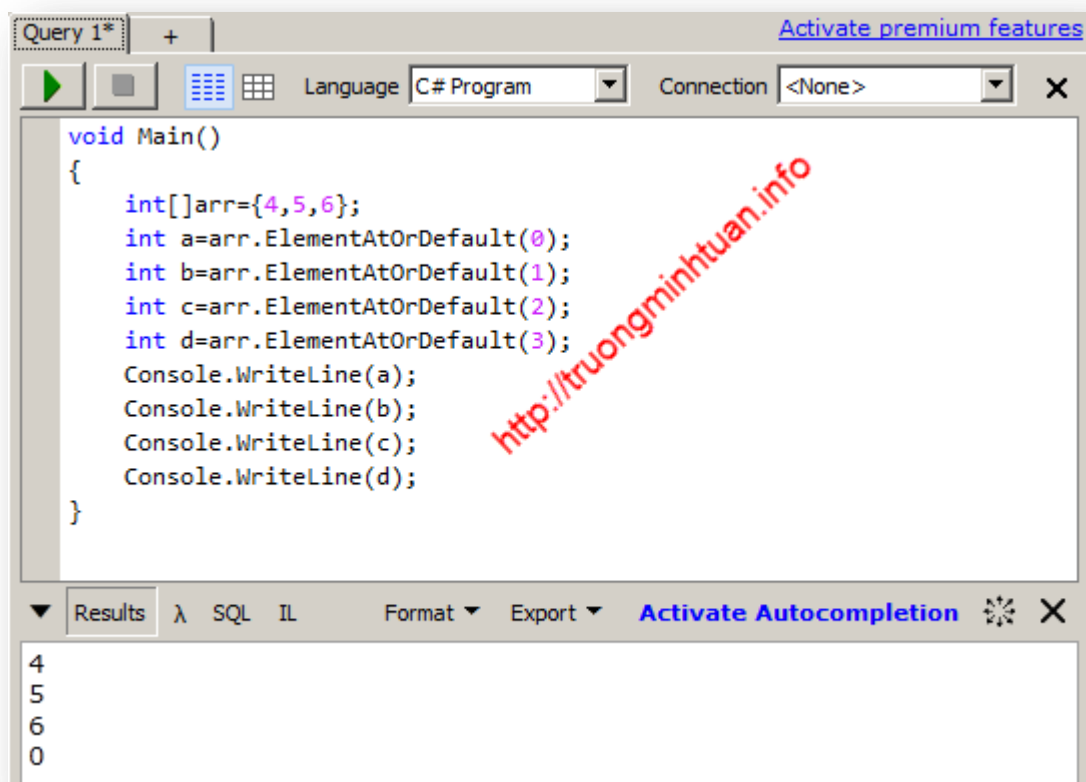


The screenshot shows a Visual Studio window with a C# program. The code defines an array `arr` with values {4, 5, 6} and uses `ElementAt` to retrieve elements at indices 0, 1, and 2. The output window shows the values 4, 5, and 6.

```
void Main()
{
    int[] arr={4,5,6};
    int a=arr.ElementAt(0);
    int b=arr.ElementAt(1);
    int c=arr.ElementAt(2);
    Console.WriteLine(a);
    Console.WriteLine(b);
    Console.WriteLine(c);
}
```

Results: 4, 5, 6

**ElementAtOrDefault:** tương tự như **ElementAt()** nhưng nó giải quyết việc vị trí vượt quá giới hạn cho phép, không như **ElementAt**.



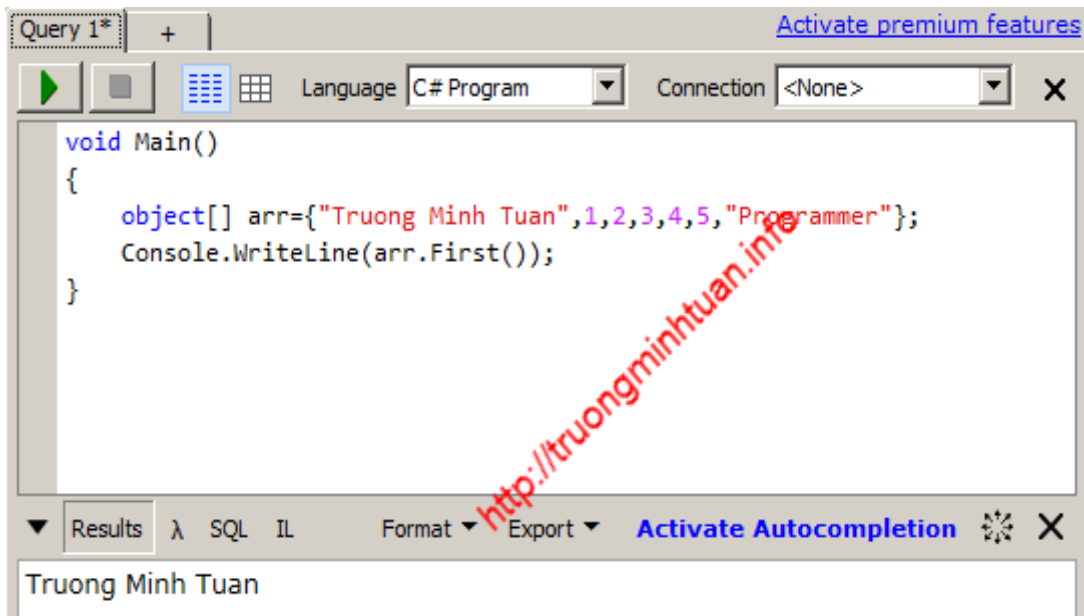
The screenshot shows a Visual Studio window with a C# program. The code defines an array `arr` with values {4, 5, 6} and uses `ElementAtOrDefault` to retrieve elements at indices 0, 1, 2, and 3. The output window shows the values 4, 5, 6, and 0.

```
void Main()
{
    int[] arr={4,5,6};
    int a=arr.ElementAtOrDefault(0);
    int b=arr.ElementAtOrDefault(1);
    int c=arr.ElementAtOrDefault(2);
    int d=arr.ElementAtOrDefault(3);
    Console.WriteLine(a);
    Console.WriteLine(b);
    Console.WriteLine(c);
    Console.WriteLine(d);
}
```

Results: 4, 5, 6, 0



**First:** trả về element đầu tiên trong object. Các object này không phân biệt kiểu. First chỉ dùng được khi các object đều có element, ngược lại Error.

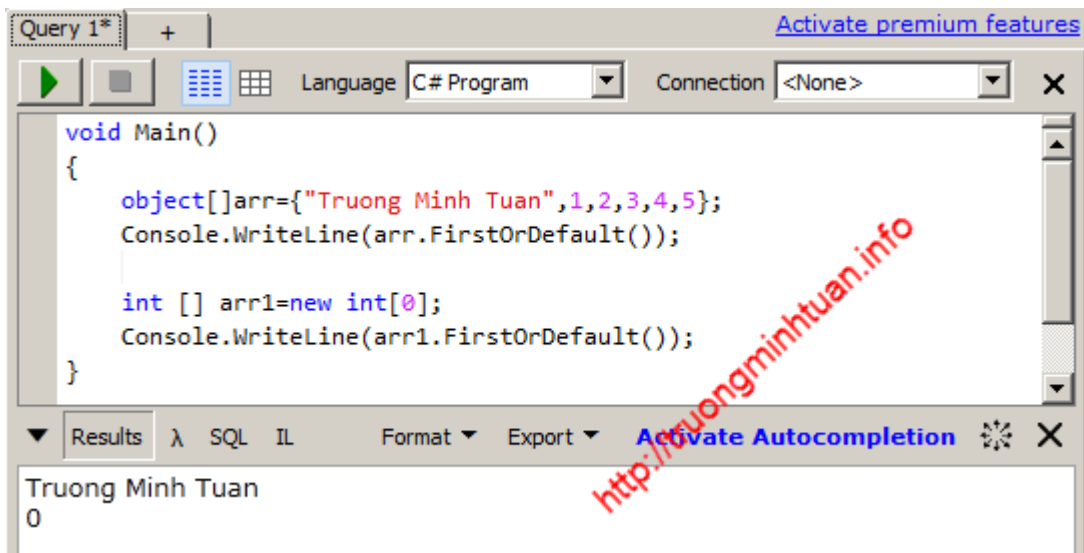


The screenshot shows a Visual Studio window titled "Query 1\*" with a toolbar and a dropdown menu for "Language" set to "C# Program". The code in the editor is as follows:

```
void Main()
{
    object[] arr={"Truong Minh Tuan",1,2,3,4,5,"Programmer"};
    Console.WriteLine(arr.First());
}
```

The output window at the bottom shows the result "Truong Minh Tuan". A red watermark "http://truongminhtuan.info" is visible across the code.

**FirstOrDefault:** tương tự như **First()** nhưng nó giải quyết việc object không có element nào, lúc đó mặc định sẽ là 0.



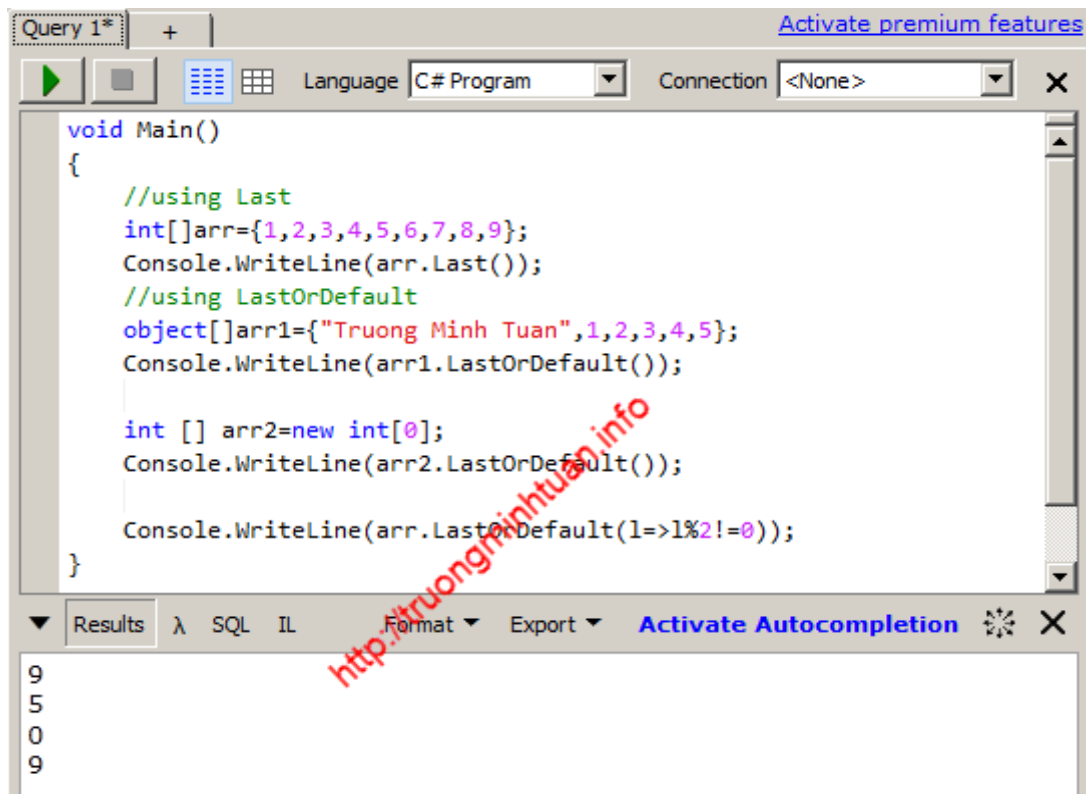
The screenshot shows a Visual Studio window titled "Query 1\*" with a toolbar and a dropdown menu for "Language" set to "C# Program". The code in the editor is as follows:

```
void Main()
{
    object[]arr={"Truong Minh Tuan",1,2,3,4,5};
    Console.WriteLine(arr.FirstOrDefault());

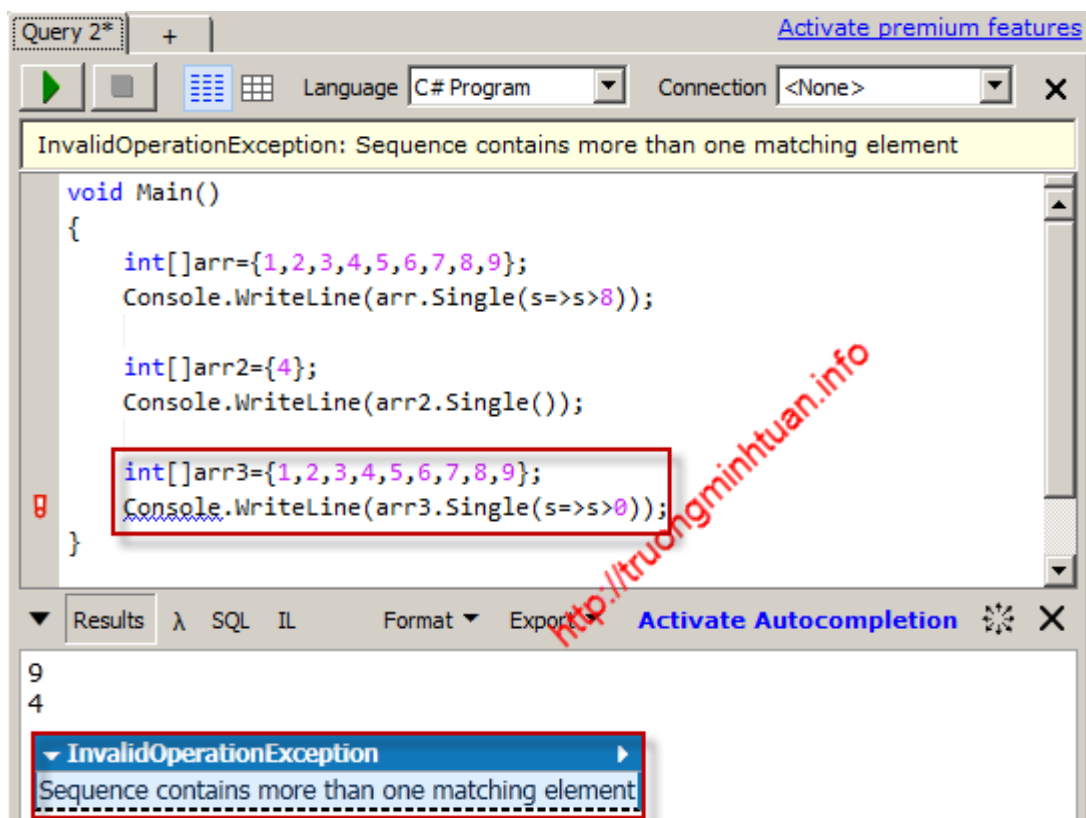
    int [] arr1=new int[0];
    Console.WriteLine(arr1.FirstOrDefault());
}
```

The output window at the bottom shows the results "Truong Minh Tuan" and "0". A red watermark "http://truongminhtuan.info" is visible across the code.

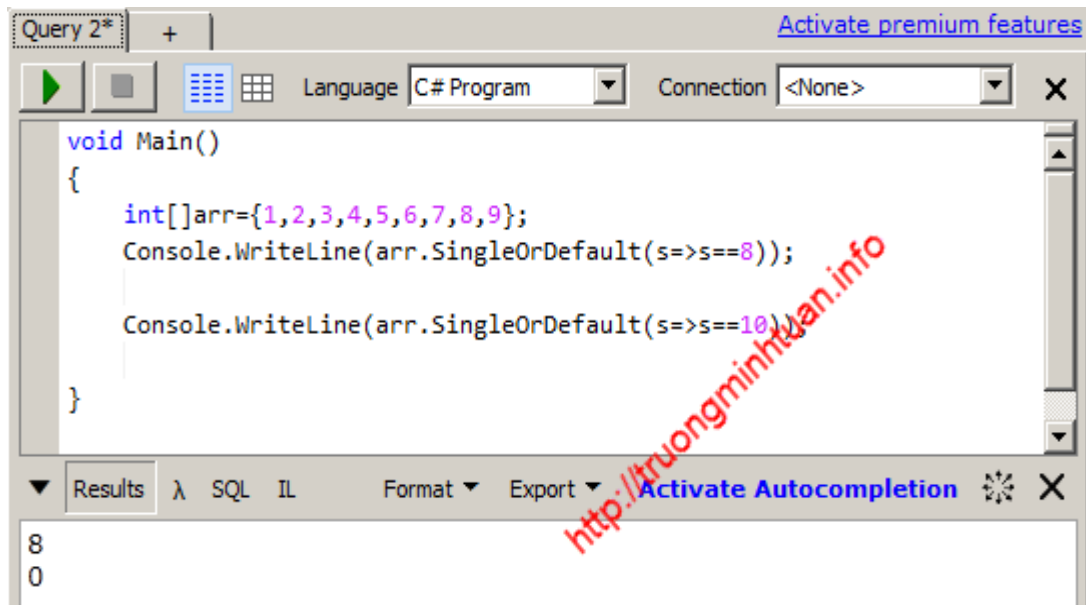
## Last, LastOrDefault: ngược lại với First(), FirstOrDefault()



**Single:** trả về một element duy nhất trong tập collection với điều kiện nào đó.



**SingleOrDefault:** tương tự như **Single()** nhưng trả về giá trị mặc định là 0 nếu không tìm thấy element nào trong tập collection



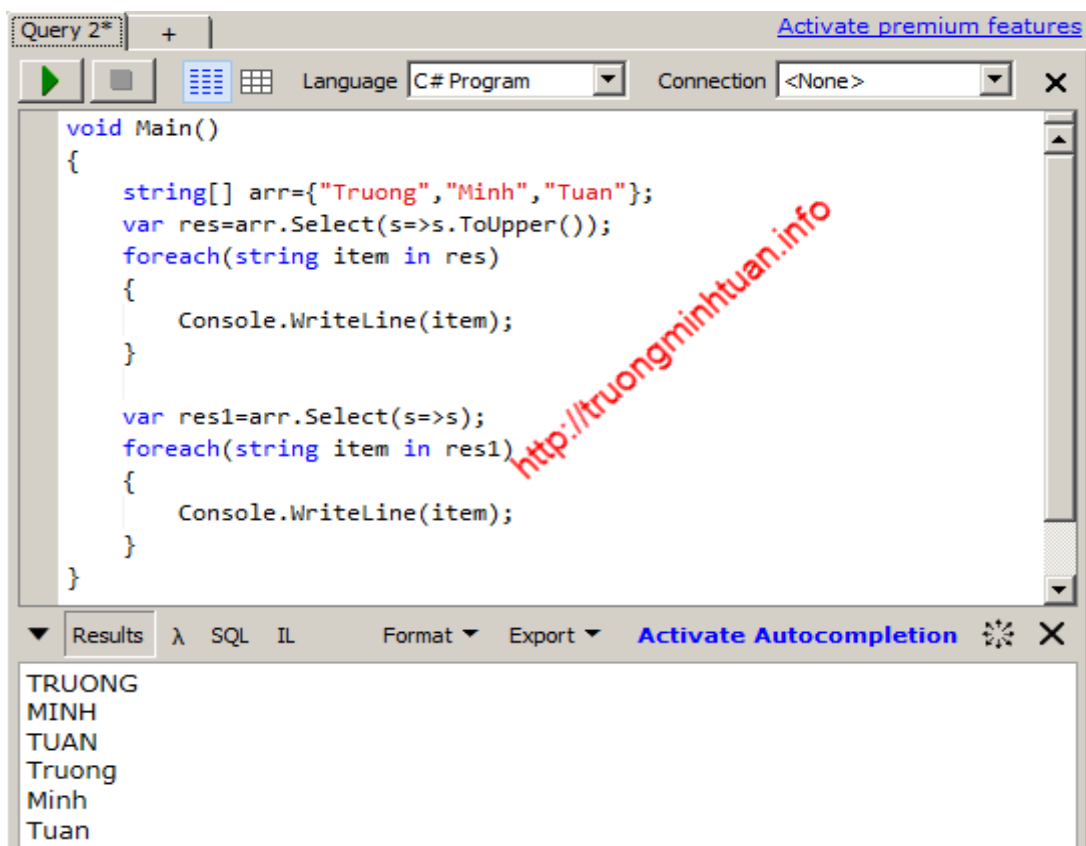
The screenshot shows a Visual Studio window titled 'Query 2\*' with a C# Program language. The code defines a `Main()` method that creates an integer array `arr` with values `{1, 2, 3, 4, 5, 6, 7, 8, 9}`. It then uses `SingleOrDefault` twice: first with a predicate `s => s == 8`, which returns `8`, and second with a predicate `s => s == 10`, which returns `0` because no element matches. The output window at the bottom shows the results `8` and `0` on separate lines.

```
void Main()
{
    int[] arr={1,2,3,4,5,6,7,8,9};
    Console.WriteLine(arr.SingleOrDefault(s=>s==8));

    Console.WriteLine(arr.SingleOrDefault(s=>s==10));
}
```

8  
0

**Select:** trả về các element trong collection. Nó gần tương tự như cú pháp truy vấn Query trong SQL



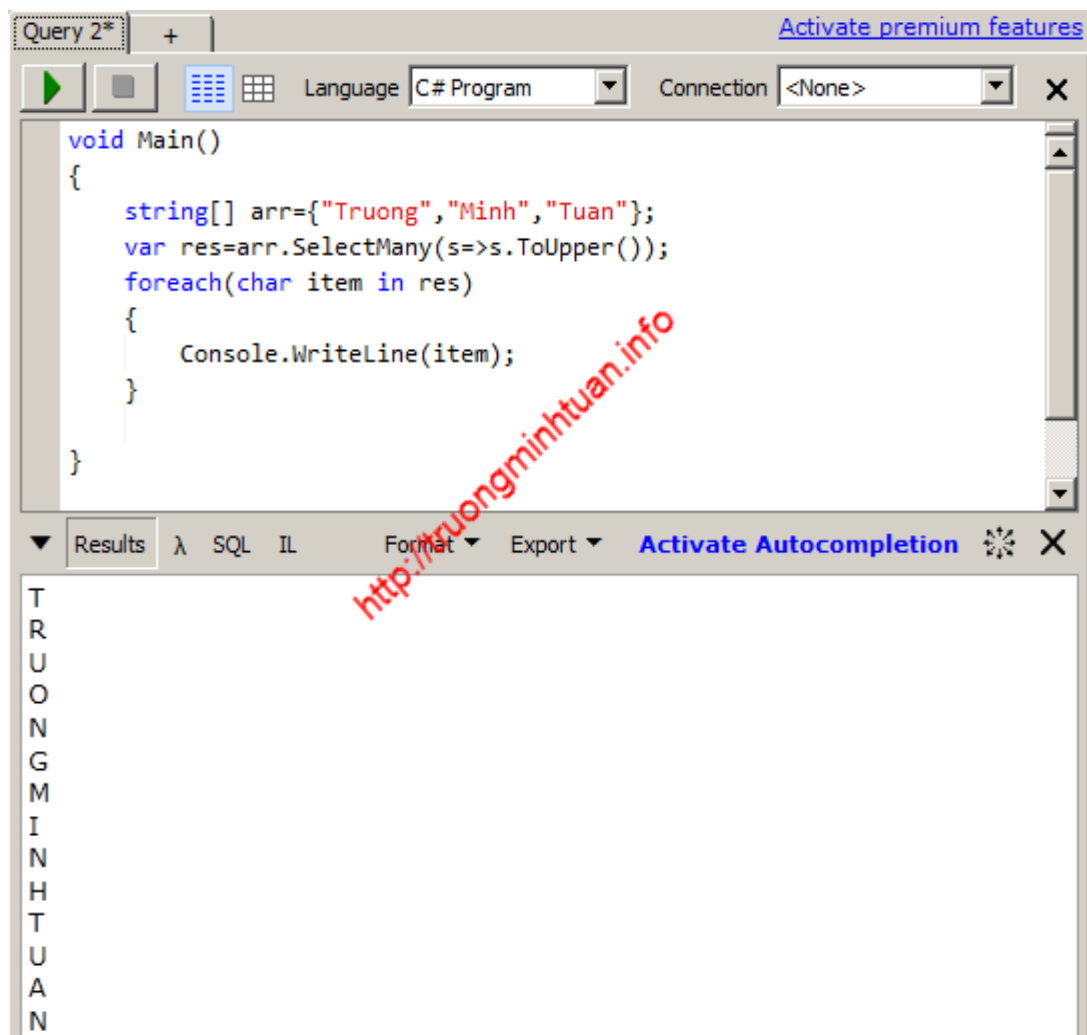
The screenshot shows a Visual Studio window titled 'Query 2\*' with a C# Program language. The code defines a `Main()` method that creates a string array `arr` with values `{"Truong", "Minh", "Tuan"}`. It then uses `Select` twice: first to transform the strings to uppercase (`s => s.ToUpper()`), resulting in `TRUONG`, `MINH`, and `TUAN`; and second to return the original strings (`s => s`), resulting in `Truong`, `Minh`, and `Tuan`. The output window at the bottom shows these six strings in sequence.

```
void Main()
{
    string[] arr={"Truong","Minh","Tuan"};
    var res=arr.Select(s=>s.ToUpper());
    foreach(string item in res)
    {
        Console.WriteLine(item);
    }

    var res1=arr.Select(s=>s);
    foreach(string item in res1)
    {
        Console.WriteLine(item);
    }
}
```

TRUONG  
MINH  
TUAN  
Truong  
Minh  
Tuan

**SelectMany:** trả về một tập collection, là kết quả của mỗi element và được chuyển đổi thành một tập collection khác

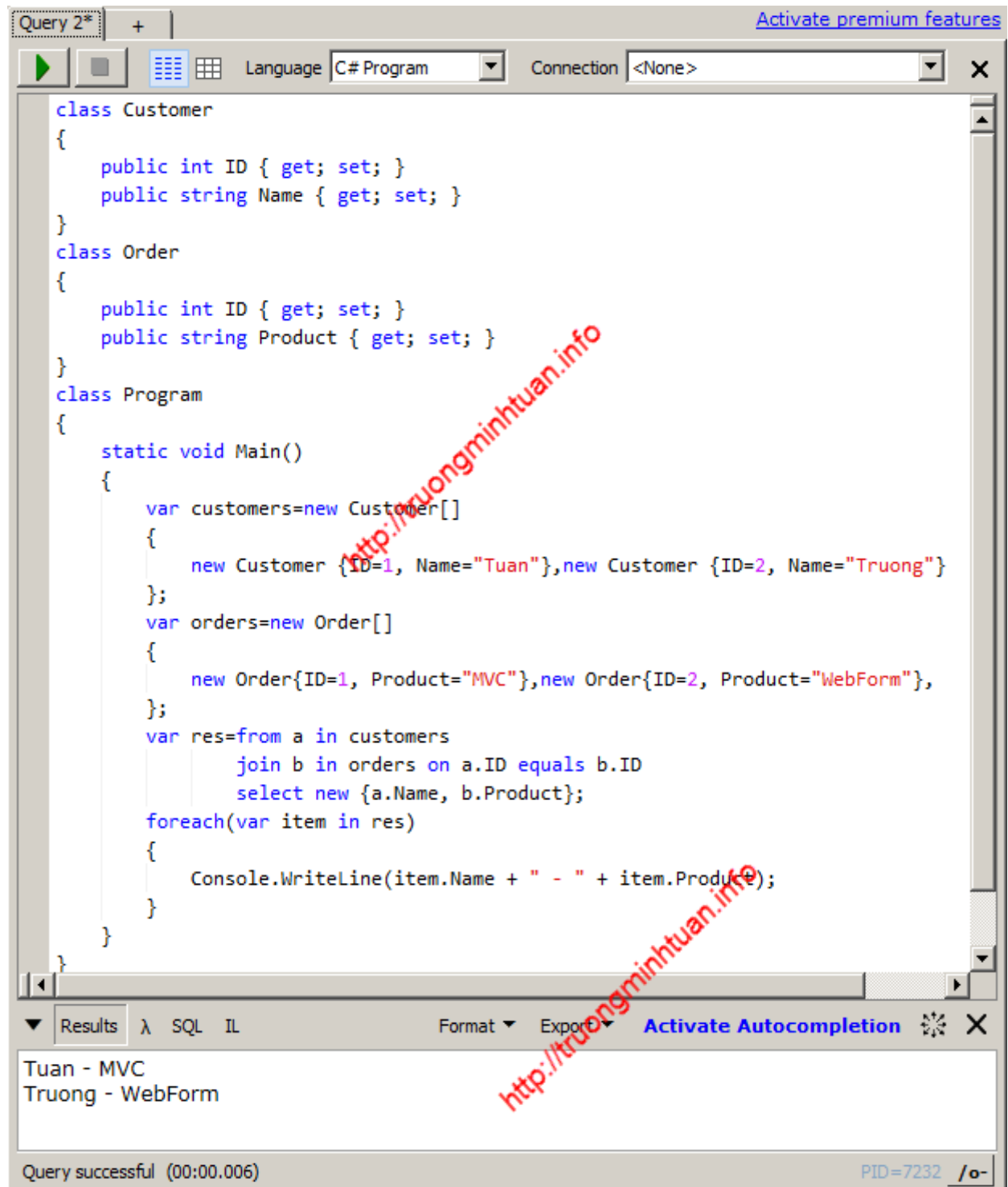


```
void Main()
{
    string[] arr={"Truong","Minh","Tuan"};
    var res=arr.SelectMany(s=>s.ToUpper());
    foreach(char item in res)
    {
        Console.WriteLine(item);
    }
}
```

Results

T  
R  
U  
O  
N  
G  
M  
I  
N  
H  
T  
U  
A  
N

**Join:** là từ khoá trong LINQ. Nó giống như cú pháp truy vấn Query in SQL. Ngoài ra, còn có Left join, Right join



The screenshot shows the Visual Studio IDE with a C# program named 'Query 2\*'. The program defines two classes, 'Customer' and 'Order', and a 'Program' class with a 'Main' method. The 'Main' method creates arrays of 'Customer' and 'Order' objects, performs a join query using 'join b in orders on a.ID equals b.ID', and prints the results to the console. The results pane at the bottom shows two lines of output: 'Tuan - MVC' and 'Truong - WebForm'. The status bar at the bottom indicates 'Query successful (00:00.006)' and 'PID=7232'.

```
class Customer
{
    public int ID { get; set; }
    public string Name { get; set; }
}
class Order
{
    public int ID { get; set; }
    public string Product { get; set; }
}
class Program
{
    static void Main()
    {
        var customers=new Customer[]
        {
            new Customer {ID=1, Name="Tuan"},new Customer {ID=2, Name="Truong"}
        };
        var orders=new Order[]
        {
            new Order{ID=1, Product="MVC"},new Order{ID=2, Product="WebForm"},
        };
        var res=from a in customers
                join b in orders on a.ID equals b.ID
                select new {a.Name, b.Product};
        foreach(var item in res)
        {
            Console.WriteLine(item.Name + " - " + item.Product);
        }
    }
}
```

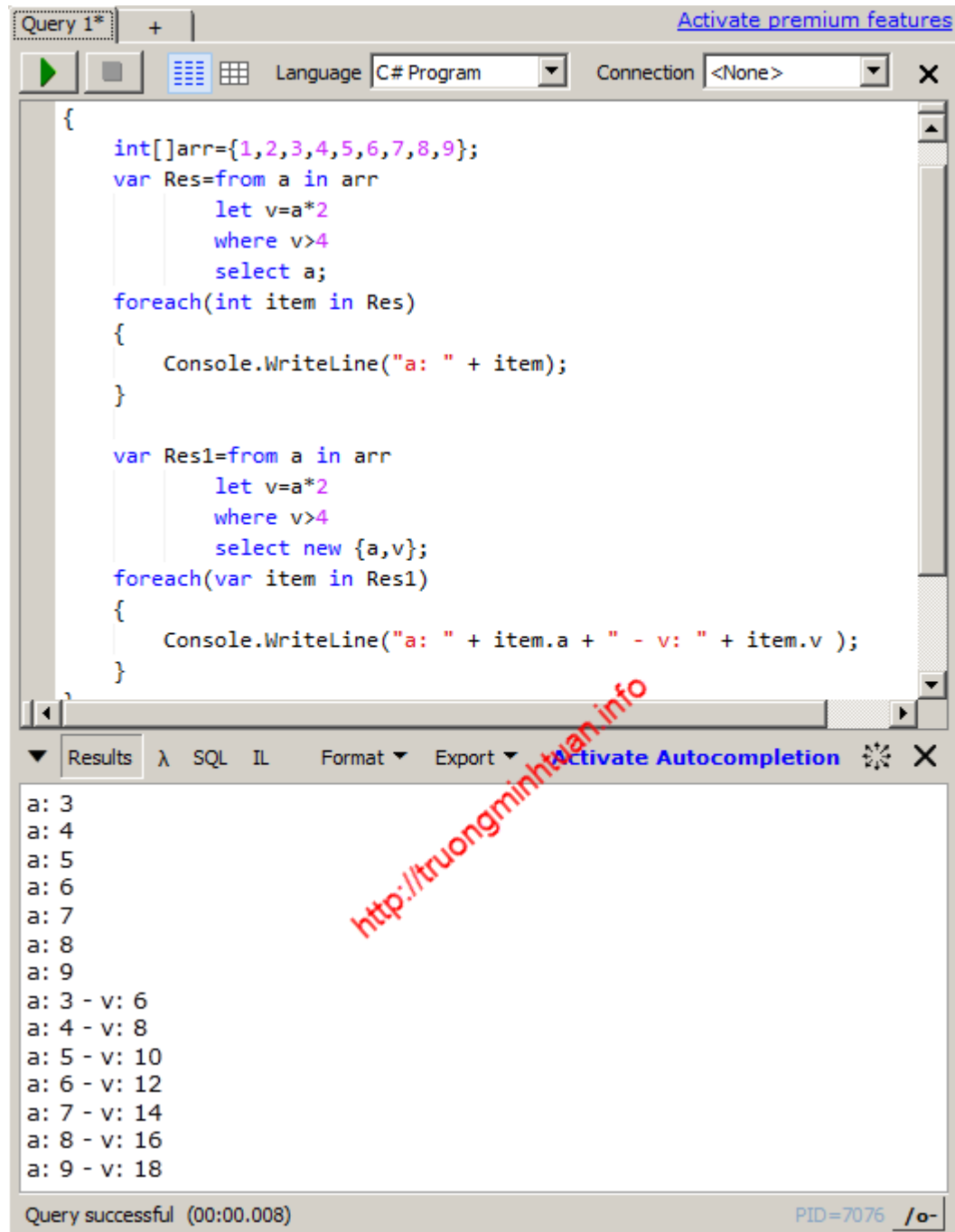
Results

Tuan - MVC  
Truong - WebForm

Query successful (00:00.006) PID=7232

Ngoài ra, bạn có thể tham khảo bài viết về Left Join, Right Join và các bài viết tương tự trên Blog cá nhân của tôi <http://truongminhtuan.info/articles>

**Let:** là một phần trong truy vấn của biểu thức LINQ, nó giúp bạn tạo ra một biến mới và tái sử dụng biến đó trong truy vấn. Điều này làm cho một số biểu thức truy vấn phức tạp trở nên đơn giản hơn.



The screenshot shows the Visual Studio IDE with a LINQ query in C# and its results. The query is as follows:

```
{
    int[] arr = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    var Res = from a in arr
              let v = a * 2
              where v > 4
              select a;
    foreach (int item in Res)
    {
        Console.WriteLine("a: " + item);
    }

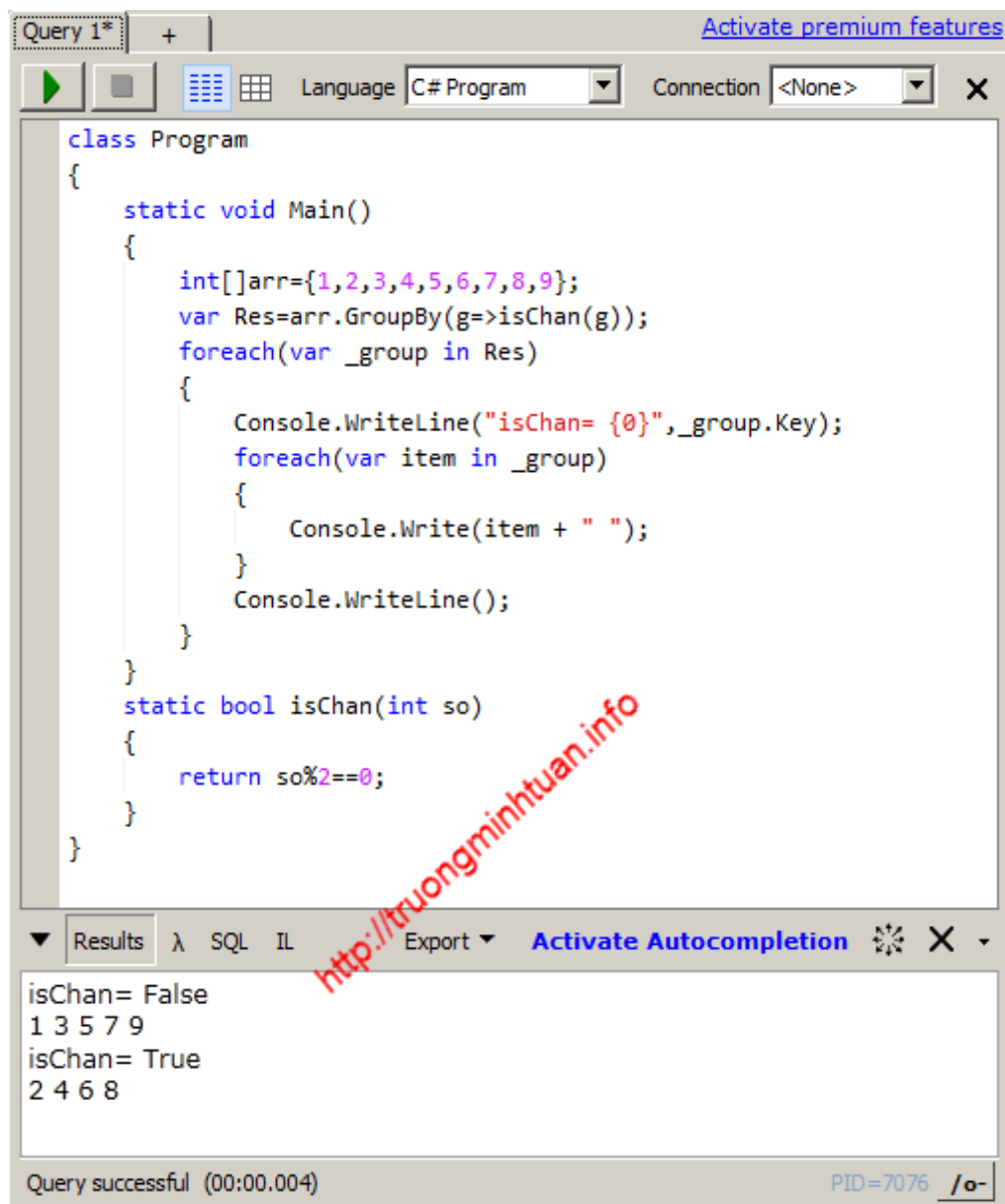
    var Res1 = from a in arr
               let v = a * 2
               where v > 4
               select new { a, v };
    foreach (var item in Res1)
    {
        Console.WriteLine("a: " + item.a + " - v: " + item.v);
    }
}
```

The results pane shows the output of the query:

```
a: 3
a: 4
a: 5
a: 6
a: 7
a: 8
a: 9
a: 3 - v: 6
a: 4 - v: 8
a: 5 - v: 10
a: 6 - v: 12
a: 7 - v: 14
a: 8 - v: 16
a: 9 - v: 18
```

The status bar at the bottom indicates "Query successful (00:00.008)" and "PID=7076".

**GroupBy:** biến một collection thành một group. Ứng với mỗi group có một key tương ứng. Nó gần giống như truy vấn SQL trong database;



The screenshot shows a C# program in a query editor. The program defines a `Program` class with a `Main` method. It creates an array `arr` with values `{1, 2, 3, 4, 5, 6, 7, 8, 9}` and groups them using `arr.GroupBy(g => isChan(g))`. The `isChan` method returns `so % 2 == 0`. The output shows two groups: 'isChan= False' with values '1 3 5 7 9' and 'isChan= True' with values '2 4 6 8'.

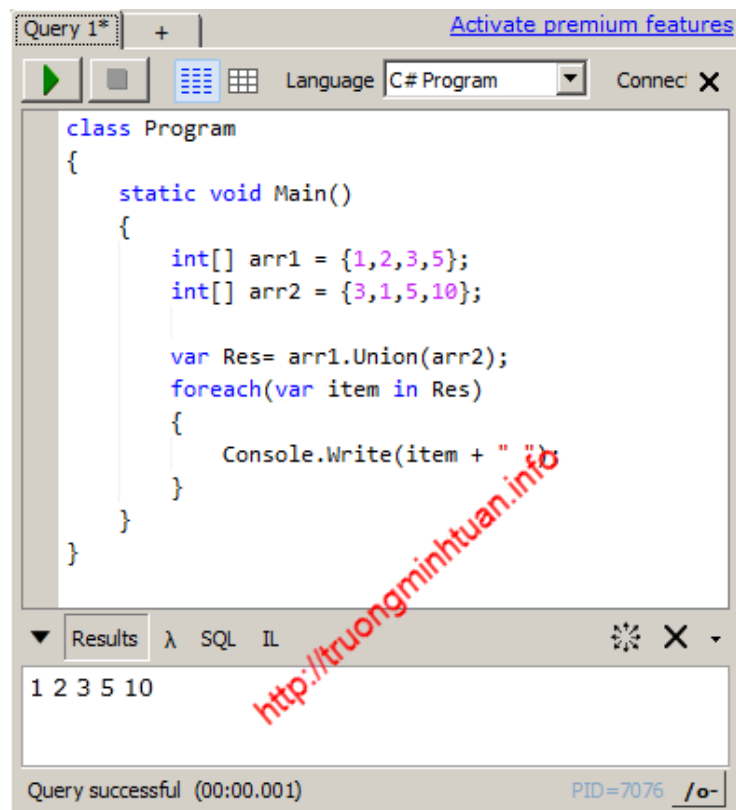
```
class Program
{
    static void Main()
    {
        int[] arr = {1, 2, 3, 4, 5, 6, 7, 8, 9};
        var Res = arr.GroupBy(g => isChan(g));
        foreach (var _group in Res)
        {
            Console.WriteLine("isChan= {0}", _group.Key);
            foreach (var item in _group)
            {
                Console.Write(item + " ");
            }
            Console.WriteLine();
        }
    }
    static bool isChan(int so)
    {
        return so % 2 == 0;
    }
}
```

Results

```
isChan= False
1 3 5 7 9
isChan= True
2 4 6 8
```

Query successful (00:00.004) PID=7076

**Union:** trả về một collection mới chứa tất cả element trong collection trước đó, nó loại bỏ các element trùng nhau.



The screenshot shows a Visual Studio window with a C# program. The code defines a class `Program` with a `Main` method. Inside `Main`, two integer arrays are created: `arr1 = {1, 2, 3, 5}` and `arr2 = {3, 1, 5, 10}`. The `Union` method is called on `arr1` with `arr2` as an argument, and the result is stored in `Res`. A `foreach` loop iterates over `Res`, printing each item followed by a space. The output window shows the result: `1 2 3 5 10`. The status bar indicates the query was successful.

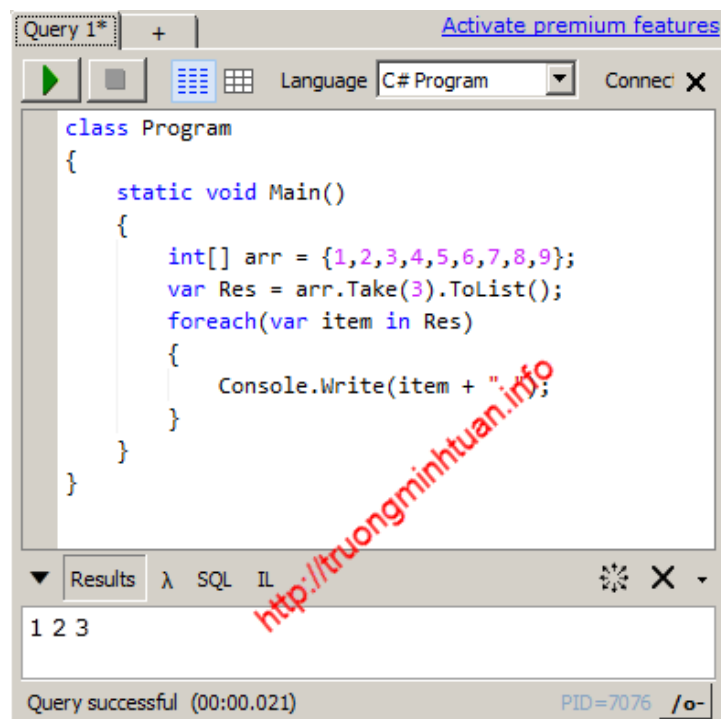
```
class Program
{
    static void Main()
    {
        int[] arr1 = {1,2,3,5};
        int[] arr2 = {3,1,5,10};

        var Res= arr1.Union(arr2);
        foreach(var item in Res)
        {
            Console.Write(item + " ");
        }
    }
}
```

Results: 1 2 3 5 10

Query successful (00:00.001) PID=7076

**Take:** trả về số lượng element tính từ vị trí đầu tiên. Take hoạt động theo một trình tự như `IEnumerable`, nó trả về một chuỗi `IEnumerable` chứa các element quy định.



The screenshot shows a Visual Studio window with a C# program. The code defines a class `Program` with a `Main` method. Inside `Main`, an integer array is created: `arr = {1, 2, 3, 4, 5, 6, 7, 8, 9}`. The `Take` method is called on `arr` with the argument `3`, and the result is converted to a list using `ToList()`. A `foreach` loop iterates over the list, printing each item followed by a space. The output window shows the result: `1 2 3`. The status bar indicates the query was successful.

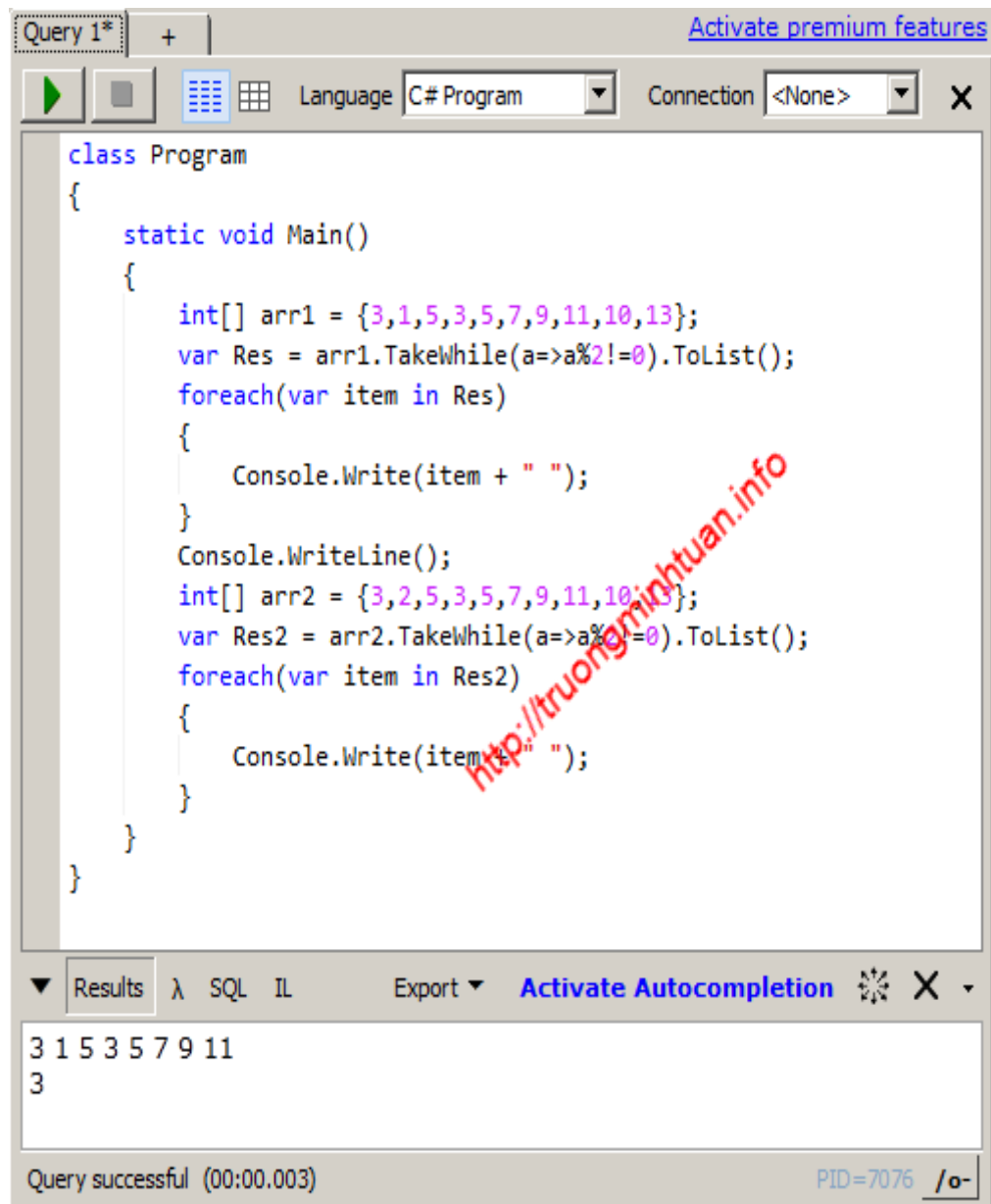
```
class Program
{
    static void Main()
    {
        int[] arr = {1,2,3,4,5,6,7,8,9};
        var Res = arr.Take(3).ToList();
        foreach(var item in Res)
        {
            Console.Write(item + " ");
        }
    }
}
```

Results: 1 2 3

Query successful (00:00.021) PID=7076



Ngoài ra, LINQ còn hỗ trợ **TakeWhile**: bạn nghiên cứu theo ví dụ bên dưới



The screenshot shows a C# program in a query editor. The program defines a class `Program` with a `Main` method. Inside `Main`, two integer arrays are defined: `arr1 = {3, 1, 5, 3, 5, 7, 9, 11, 10, 13}` and `arr2 = {3, 2, 5, 3, 5, 7, 9, 11, 10, 13}`. For each array, the `TakeWhile(a => a % 2 != 0)` method is used to filter out even numbers, and the results are printed to the console. The first array's results are `3 1 5 3 5 7 9 11`, and the second array's results are `3`. The query is successful, and the execution time is 00:00.003.

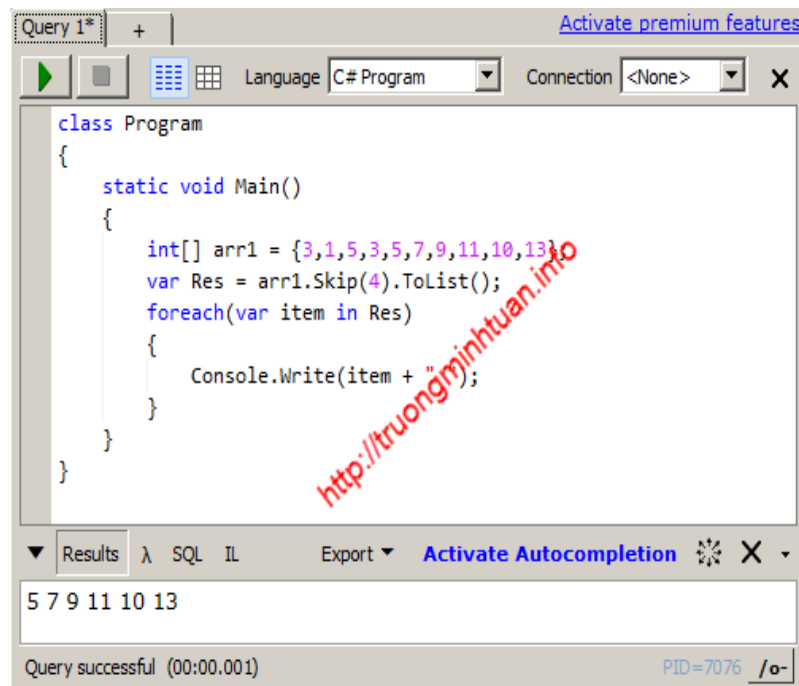
```
class Program
{
    static void Main()
    {
        int[] arr1 = {3, 1, 5, 3, 5, 7, 9, 11, 10, 13};
        var Res = arr1.TakeWhile(a => a % 2 != 0).ToList();
        foreach(var item in Res)
        {
            Console.Write(item + " ");
        }
        Console.WriteLine();
        int[] arr2 = {3, 2, 5, 3, 5, 7, 9, 11, 10, 13};
        var Res2 = arr2.TakeWhile(a => a % 2 != 0).ToList();
        foreach(var item in Res2)
        {
            Console.Write(item + " ");
        }
    }
}
```

Results

3 1 5 3 5 7 9 11  
3

Query successful (00:00.003) PID=7076 /o-

**Skip:** là một extension method. Nó chỉ trả về các element tương ứng, tính sau vị trí quy định của các phần tử trong collection. Skip vô cùng hữu ích khi bạn cần chọn lọc và xử lý vấn đề;

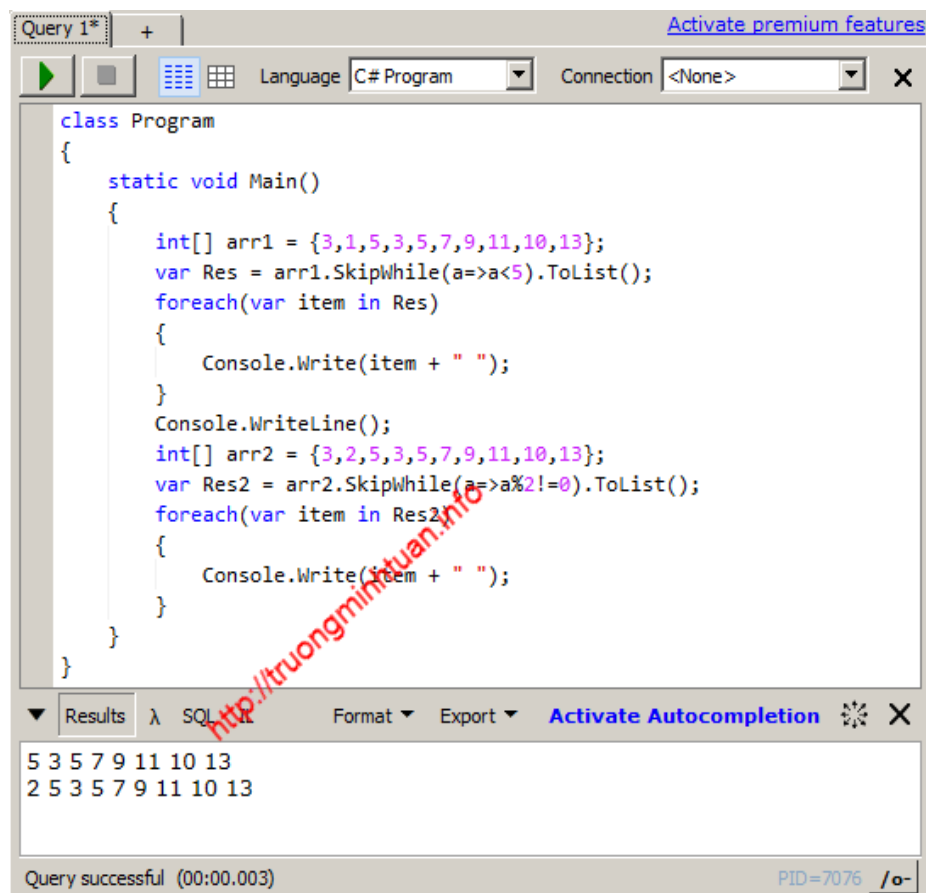


```
class Program
{
    static void Main()
    {
        int[] arr1 = {3, 1, 5, 3, 5, 7, 9, 11, 10, 13};
        var Res = arr1.Skip(4).ToList();
        foreach(var item in Res)
        {
            Console.Write(item + " ");
        }
    }
}
```

Results: 5 7 9 11 10 13

Query successful (00:00.001)

Ngoài ra, LINQ còn hỗ trợ **SkipWhile**: bạn nghiên cứu theo ví dụ bên dưới

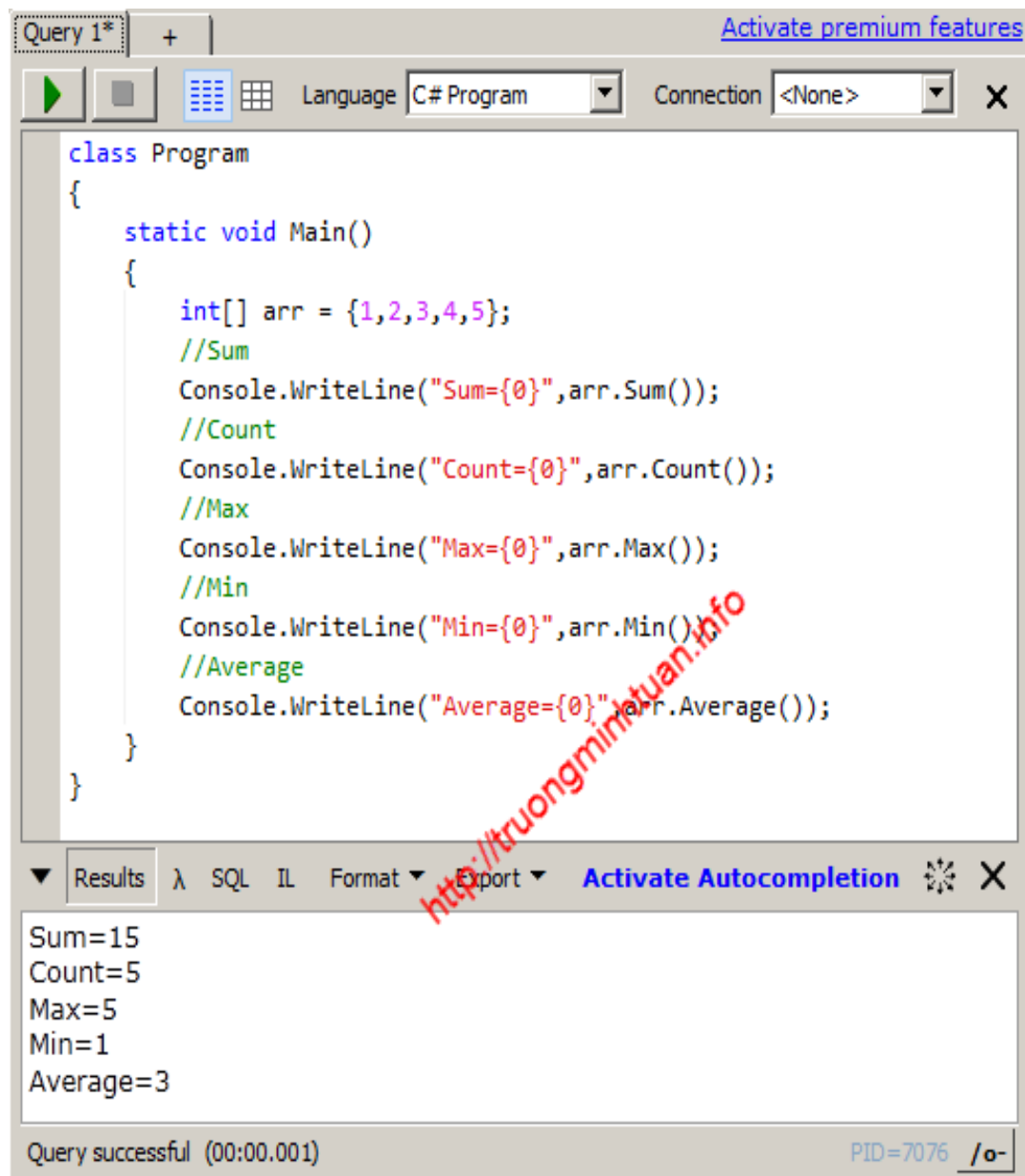


```
class Program
{
    static void Main()
    {
        int[] arr1 = {3, 1, 5, 3, 5, 7, 9, 11, 10, 13};
        var Res = arr1.SkipWhile(a => a < 5).ToList();
        foreach(var item in Res)
        {
            Console.Write(item + " ");
        }
        Console.WriteLine();
        int[] arr2 = {3, 2, 5, 3, 5, 7, 9, 11, 10, 13};
        var Res2 = arr2.SkipWhile(a => a % 2 != 0).ToList();
        foreach(var item in Res2)
        {
            Console.Write(item + " ");
        }
    }
}
```

Results: 5 3 5 7 9 11 10 13  
2 5 3 5 7 9 11 10 13

Query successful (00:00.003)

**Sum, Max, Min, Count, Average, OrderBy, Where, Descending, Ascending:** tôi không nói rõ ở đây, vì thông qua các ví dụ trên, bạn có thể tìm hiểu về các từ khoá này, nó hoàn toàn giống như truy vấn SQL; tức nhiên, tôi làm vài ví dụ bên dưới để bạn có thể tìm hiểu thêm;



The screenshot shows a window titled "Query 1\*" with a toolbar containing a green play button, a grey square, a blue grid icon, and a white grid icon. The "Language" dropdown is set to "C# Program" and the "Connection" dropdown is set to "<None>". The main text area contains the following C# code:

```
class Program
{
    static void Main()
    {
        int[] arr = {1,2,3,4,5};
        //Sum
        Console.WriteLine("Sum={0}",arr.Sum());
        //Count
        Console.WriteLine("Count={0}",arr.Count());
        //Max
        Console.WriteLine("Max={0}",arr.Max());
        //Min
        Console.WriteLine("Min={0}",arr.Min());
        //Average
        Console.WriteLine("Average={0}",arr.Average());
    }
}
```

Below the code editor, there is a "Results" tab and a toolbar with icons for SQL, IL, Format, and Export. The "Results" tab is active, displaying the output of the program:

```
Sum=15
Count=5
Max=5
Min=1
Average=3
```

At the bottom of the window, a status bar shows "Query successful (00:00.001)" and "PID=7076 /o-". A red watermark "http://truongminhtuan.info" is visible across the center of the image.

Query 1\* + [Activate premium features](#)

Language C# Program Connection <None>

```
class LopHoc
{
    public int IDLop {get;set;}
    public string TenLop {get;set;}
}
class Program
{
    static void Main()
    {
        List<LopHoc> lop = new List<LopHoc>(){
            new LopHoc{IDLop=1, TenLop="ADO.NET Entity Framework"},
            new LopHoc{IDLop=2, TenLop="C# step by step"},
            new LopHoc{IDLop=3, TenLop="LINQ step by step"},
            new LopHoc{IDLop=4, TenLop="ASP.NET MVC5"},
            new LopHoc{IDLop=5, TenLop="Web API"},
        };
        var Res=from a in lop
                where a.TenLop.Contains("A")
                orderby a.TenLop descending
                select a;
        foreach(var item in Res)
        {
            Console.WriteLine(item.TenLop);
        }
    }
}
```

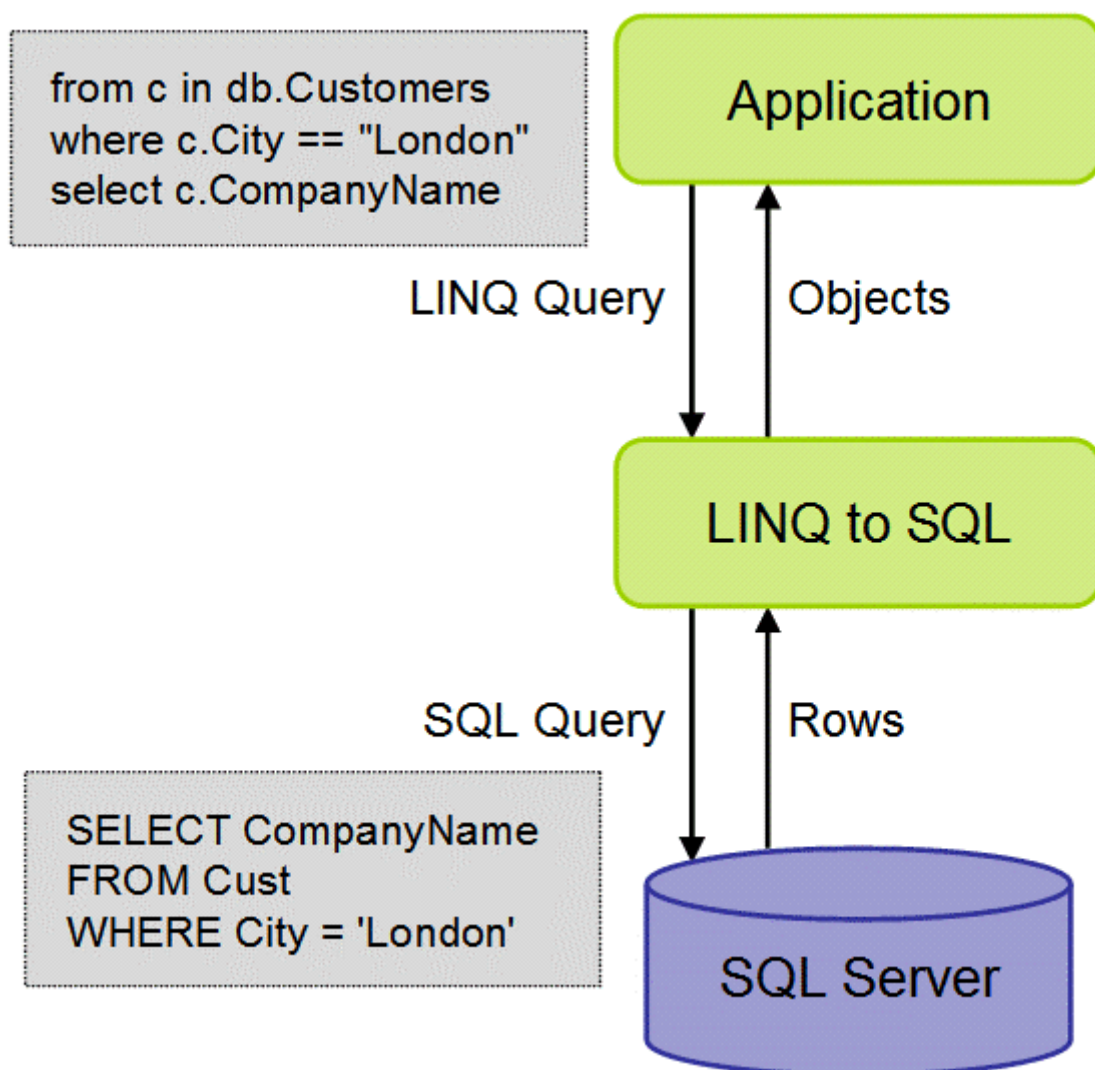
Results λ SQL IL Format Export [Activate Autocompletion](#)

Web API  
ASP.NET MVC5  
ADO.NET Entity Framework

Query successful (00:00.005) PID=7928 /o-

## 7. LINQ to SQL

LINQ to SQL cung cấp một cơ sở hạ tầng cho việc quản lý các dữ liệu quan hệ như các đối tượng. Nó là một thành phần của version 3.5 .NET Framework và LINQ to SQL là một phiên bản hiện thực hoá của O/RM (object relational mapping) có bên trong .NET Framework. Nó cho phép bạn mô hình hoá một cơ sở dữ liệu dùng các lớp .NET. Sau đó, bạn có thể truy vấn chúng bằng cách dùng LINQ – tức nhiên nó cho phép bạn các thao tác: Insert, Update, Delete, View, Store Procedure (SP), Transaction. LINQ to SQL chỉ support cho SQL Server, và bạn là người mới bắt đầu nghiên cứu thì LINQ to SQL là sự lựa chọn tuyệt vời nhất.



Dưới đây, tôi hướng dẫn cho bạn từng bước triển khai LINQ to SQL và các thao tác.

Ở đây, tôi dùng Visual Studio 2012, SQL Server 2008 để thực hiện, bạn cũng có thể dùng các phiên bản như Visual Studio 2010, 2013, 2015.

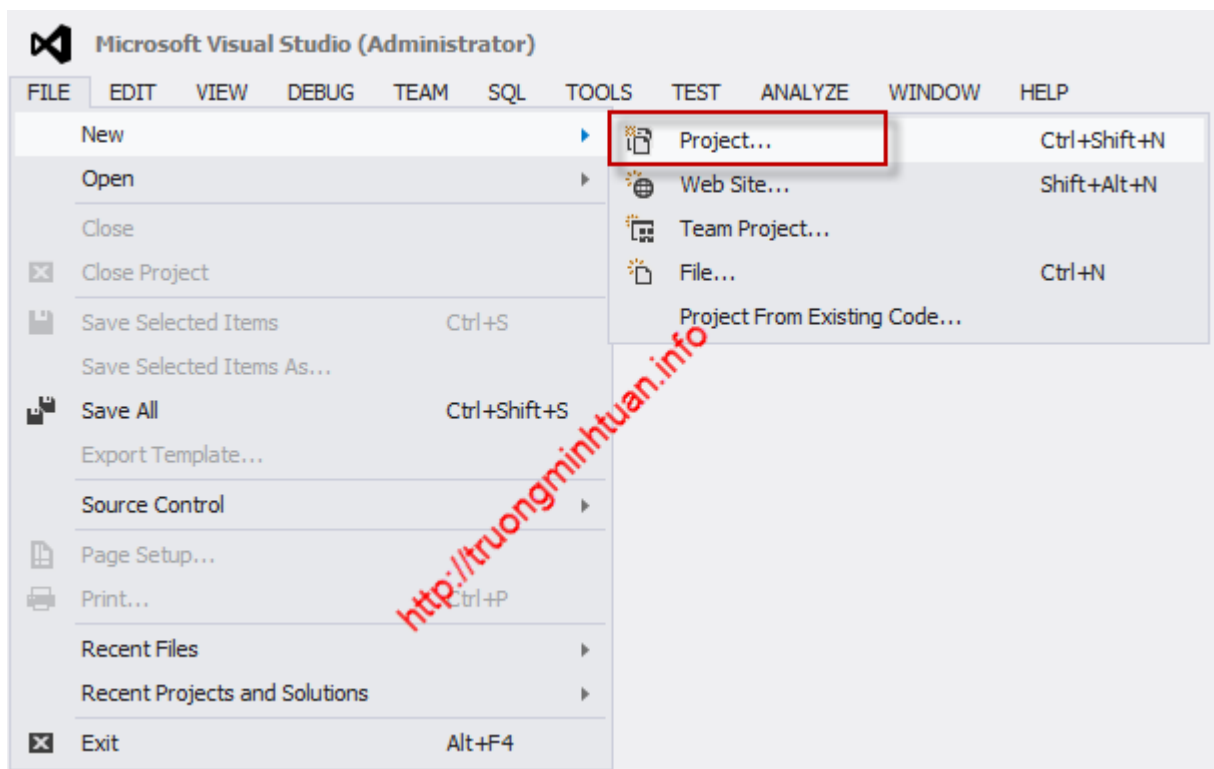
*Bước 01: Tôi tạo một Cơ sở dữ liệu tương ứng bên dưới có tên là:*

Database: LINQtoSQL

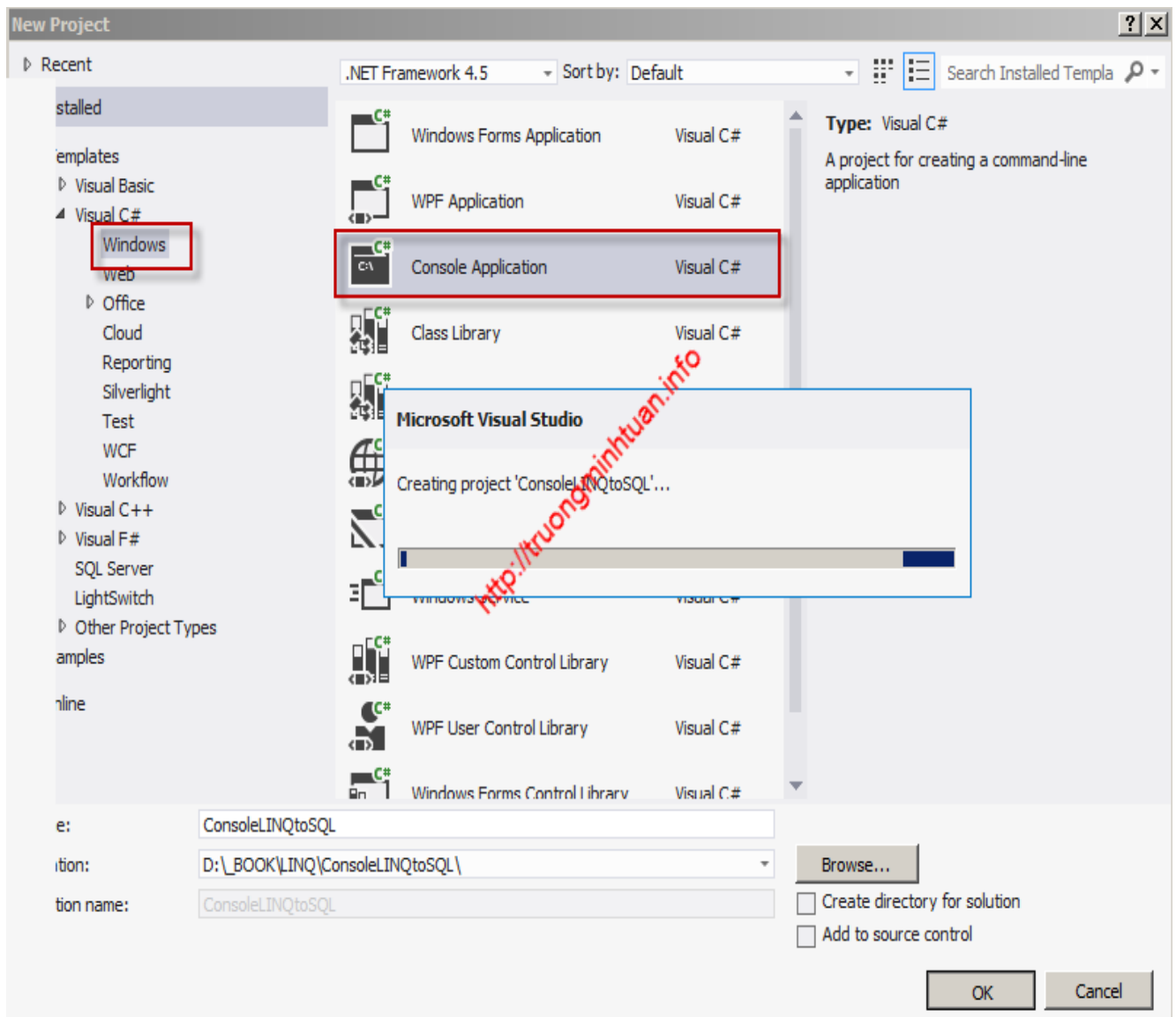
Table: LopHoc (IDLop, TenLop)

```
1  USE [LINQtoSQL]
2  GO
3
4  CREATE TABLE [dbo].[LopHoc] (
5      [IDLop] [int] IDENTITY(1,1) NOT NULL,
6      [TenLop] [nvarchar] (50) NULL,
7      CONSTRAINT [PK_LopHoc] PRIMARY KEY CLUSTERED
8      (
9          [IDLop] ASC
10     ) WITH (PAD_INDEX = OFF,
11     STATISTICS_NORECOMPUTE = OFF,
12     IGNORE_DUP_KEY = OFF,
13     ALLOW_ROW_LOCKS = ON,
14     ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
15 ) ON [PRIMARY]
16
17 GO
```

Bước 02: Tạo một Project mới **File / New / Project**

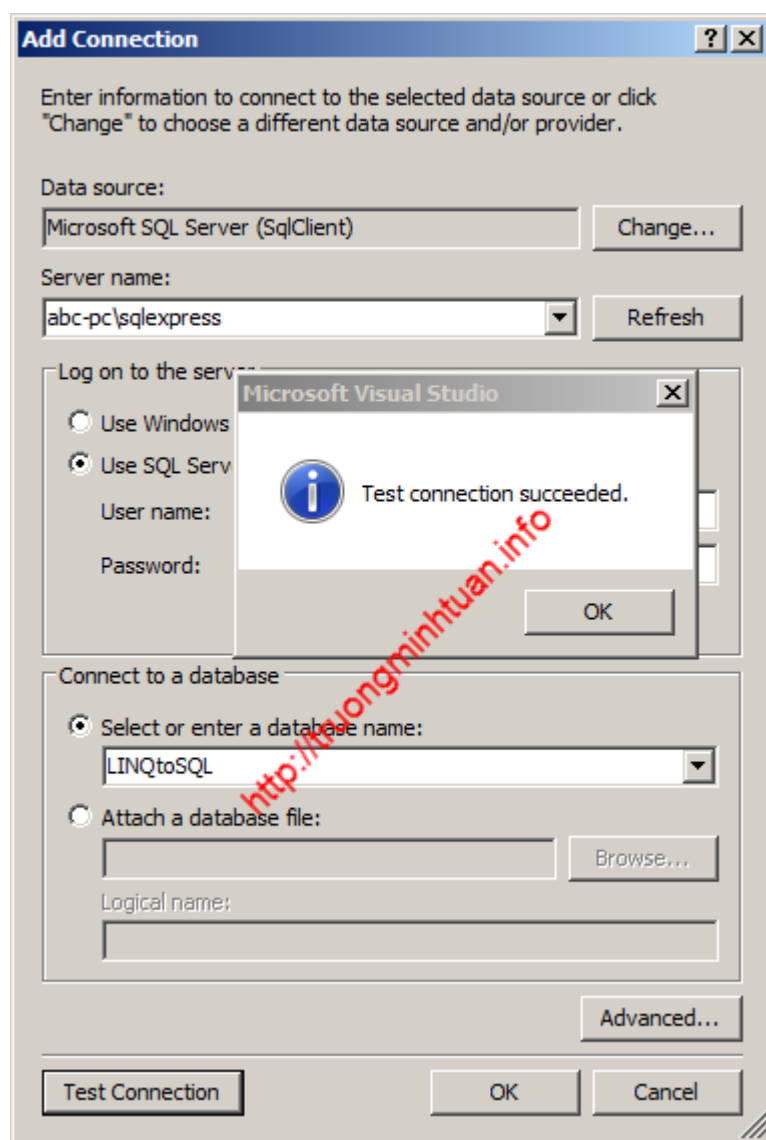


### Bước 03: Đặt tên Project và đường dẫn cho Project

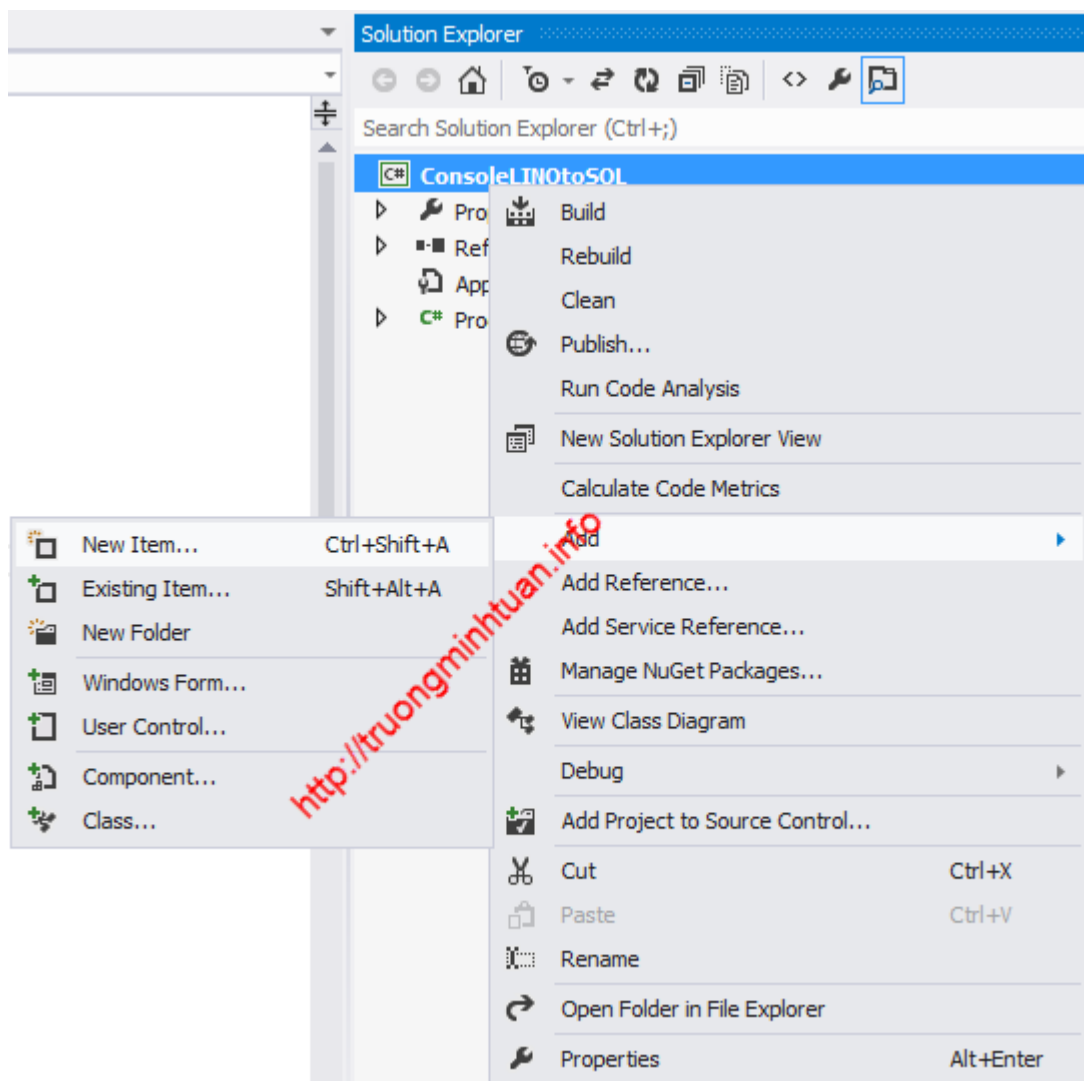


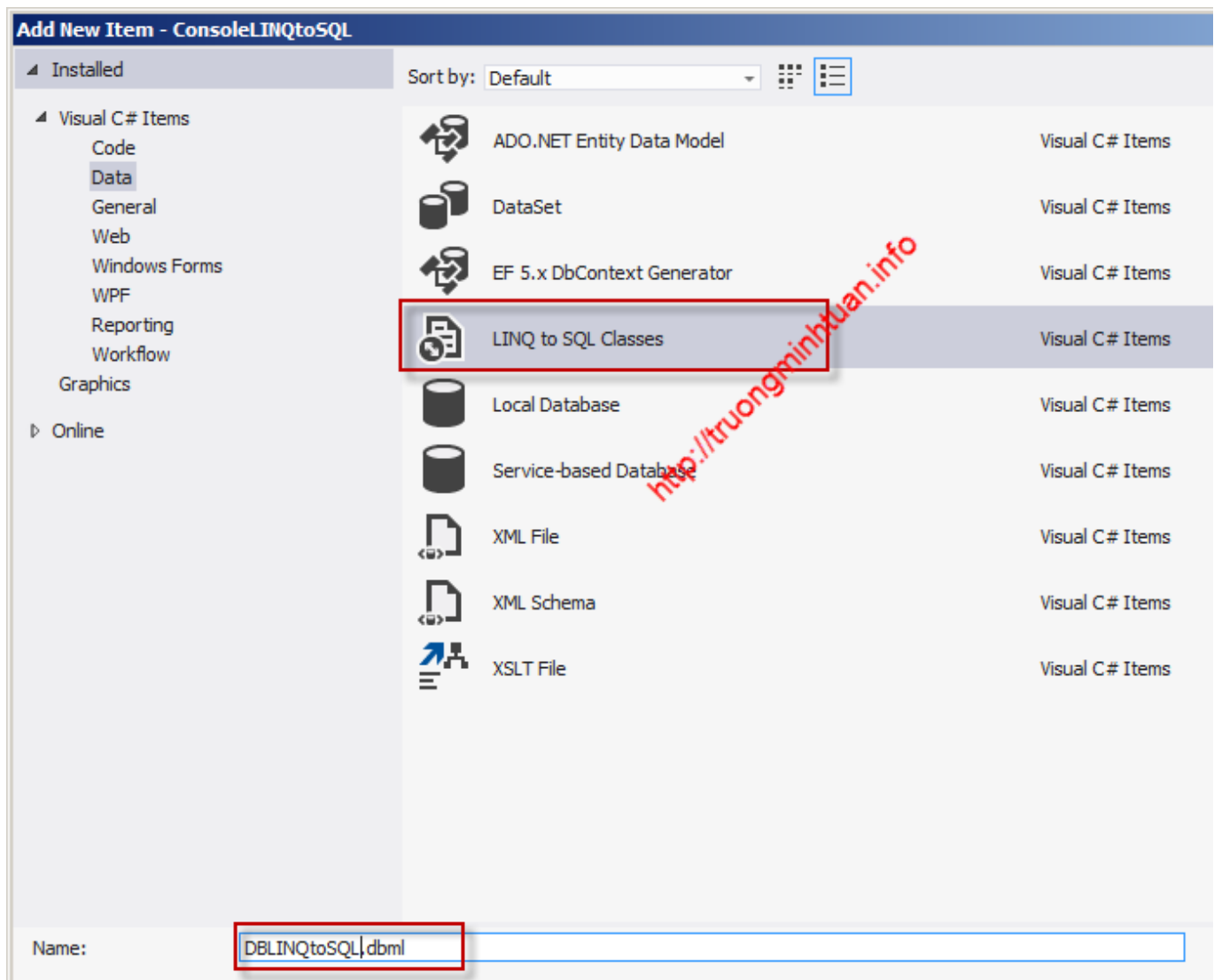


*Bước 04: Tạo một Data Connection để map dữ liệu từ SQL Server, bằng menu trong Visual Studio: View / Server Explorer / Data Connections / Add Connection*

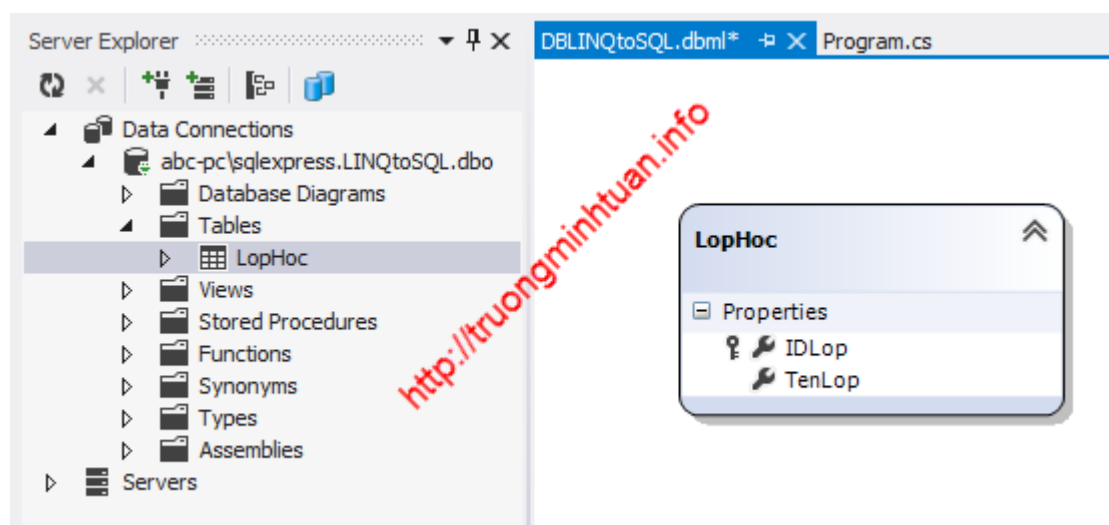


### Bước 05: Add LINQ to SQL





*Bước 06: Chọn Table LopHoc từ Server Explorer đưa vào LINQ to SQL class*



*Bước 07: Thực hiện thao tác Hiển thị dữ liệu dùng truy vấn LINQ*

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleLINQtoSQL
8 {
9     class Program
10     {
11         static DBLINQtoSQLDataContext db = new DBLINQtoSQLDataContext();
12         static void Main(string[] args)
13         {
14             LoadList();
15         }
16
17         static void LoadList()
18         {
19             var Res = db.LopHocs.ToList();
20             foreach (var item in Res)
21             {
22                 Console.WriteLine(item.TenLop);
23             }
24             Console.ReadLine();
25         }
26     }
27 }
```

*Bước 08: Thực hiện thao tác Thêm mới dữ liệu dùng truy vấn LINQ*

```
DBLINQtoSQL.dbml | Program.cs | ConsoleLINQtoSQL.Program | Main(string[] args)
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleLINQtoSQL
8 {
9     class Program
10    {
11        static DBLINQtoSQLDataContext db = new DBLINQtoSQLDataContext();
12        static void Main(string[] args)
13        {
14            LoadList();
15            //Goi Ham Insert
16            LopHoc lop = new LopHoc();
17            lop.TenLop = "PHP";
18            Insert(lop);
19        }
20
21        static void Insert(LopHoc lop)
22        {
23            db.LopHocs.InsertOnSubmit(lop);
24            db.SubmitChanges();
25        }
26
27        #region
28    }
29 }
```

### Bước 09: Thực hiện thao tác Cập nhật dữ liệu dùng truy vấn LINQ

```
DBLINQtoSQL.dbml Program.cs
ConsoleLINQtoSQL.Program Update(LopHoc lop)

4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleLINQtoSQL
8 {
9     class Program
10    {
11        static DBLINQtoSQLDataContext db = new DBLINQtoSQLDataContext();
12        static void Main(string[] args)
13        {
14            //Goi ham Update
15            LopHoc lop = new LopHoc();
16            lop.IDLop = 8;
17            lop.TenLop = "Ngon ngu lap trinh PHP";
18            Update(lop);
19        }
20
21        static void Update(LopHoc lop)
22        {
23            var Res = db.LopHocs.Where(w => w.IDLop.Equals(lop.IDLop)).SingleOrDefault();
24            if (Res != null)
25            {
26                Res.TenLop = lop.TenLop;
27            }
28            db.SubmitChanges();
29        }
30    }
}
```

### Bước 10: Thực hiện thao tác Xóa dữ liệu dùng truy vấn LINQ

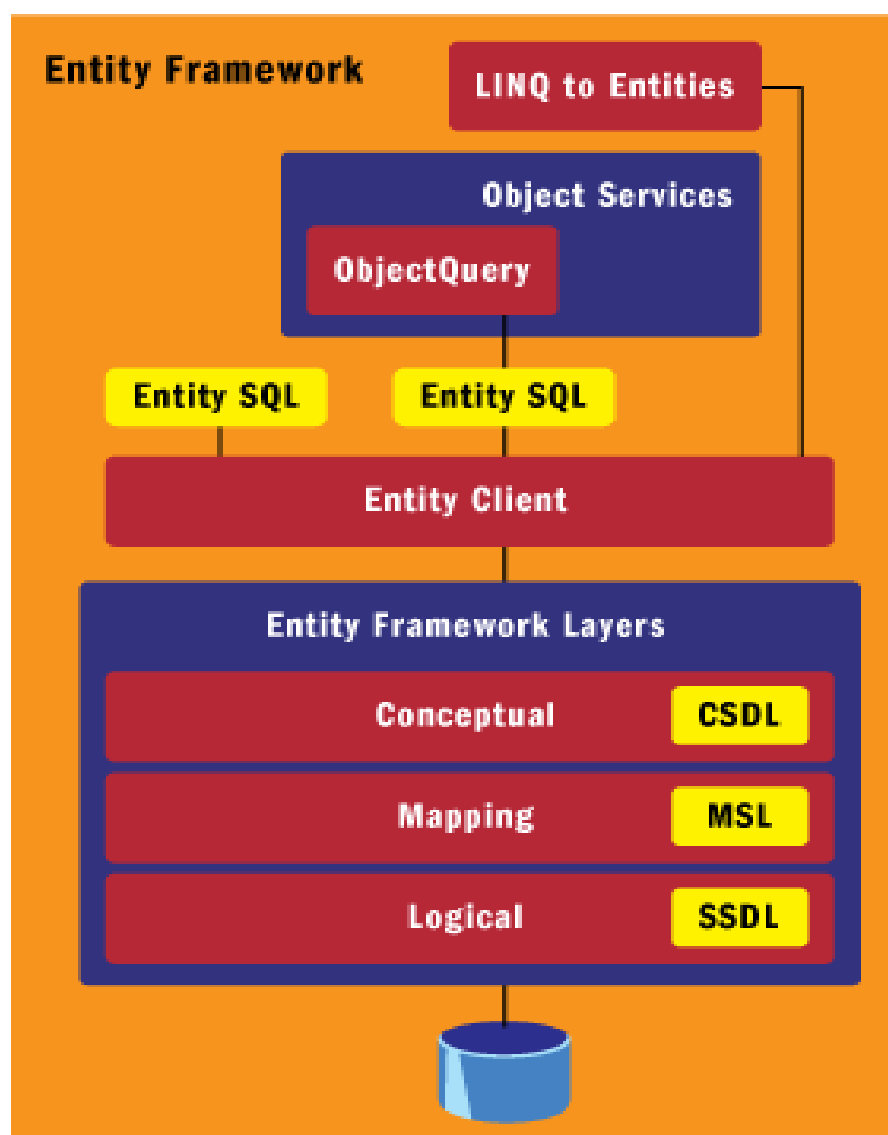
```
DBLINQtoSQL.dbml Program.cs
ConsoleLINQtoSQL.Program Insert(LopHoc lop)

7 namespace ConsoleLINQtoSQL
8 {
9     class Program
10    {
11        static DBLINQtoSQLDataContext db = new DBLINQtoSQLDataContext();
12        static void Main(string[] args)
13        {
14            //Goi ham Delete
15            LopHoc lop = new LopHoc();
16            Delete(8);
17        }
18
19        static void Delete(int ID)
20        {
21            var Res = db.LopHocs.Where(w => w.IDLop.Equals(ID));
22            if (Res != null)
23            {
24                db.LopHocs.DeleteAllOnSubmit(Res);
25                db.SubmitChanges();
26            }
27        }
28    }
}
```

## 8. LINQ to Entities

LINQ to Entities là một phần của ADO.NET Entity Framework. Nó thì linh hoạt hơn LINQ to SQL, nhưng rất ít lập trình viên sử dụng vì độ phức tạp của nó. Tuy nhiên, LINQ to Entities thì lại hỗ trợ tốt các hệ quản trị cơ sở dữ liệu như Oracle, MySQL không như LINQ to SQL chỉ hỗ trợ duy nhất SQL Server.

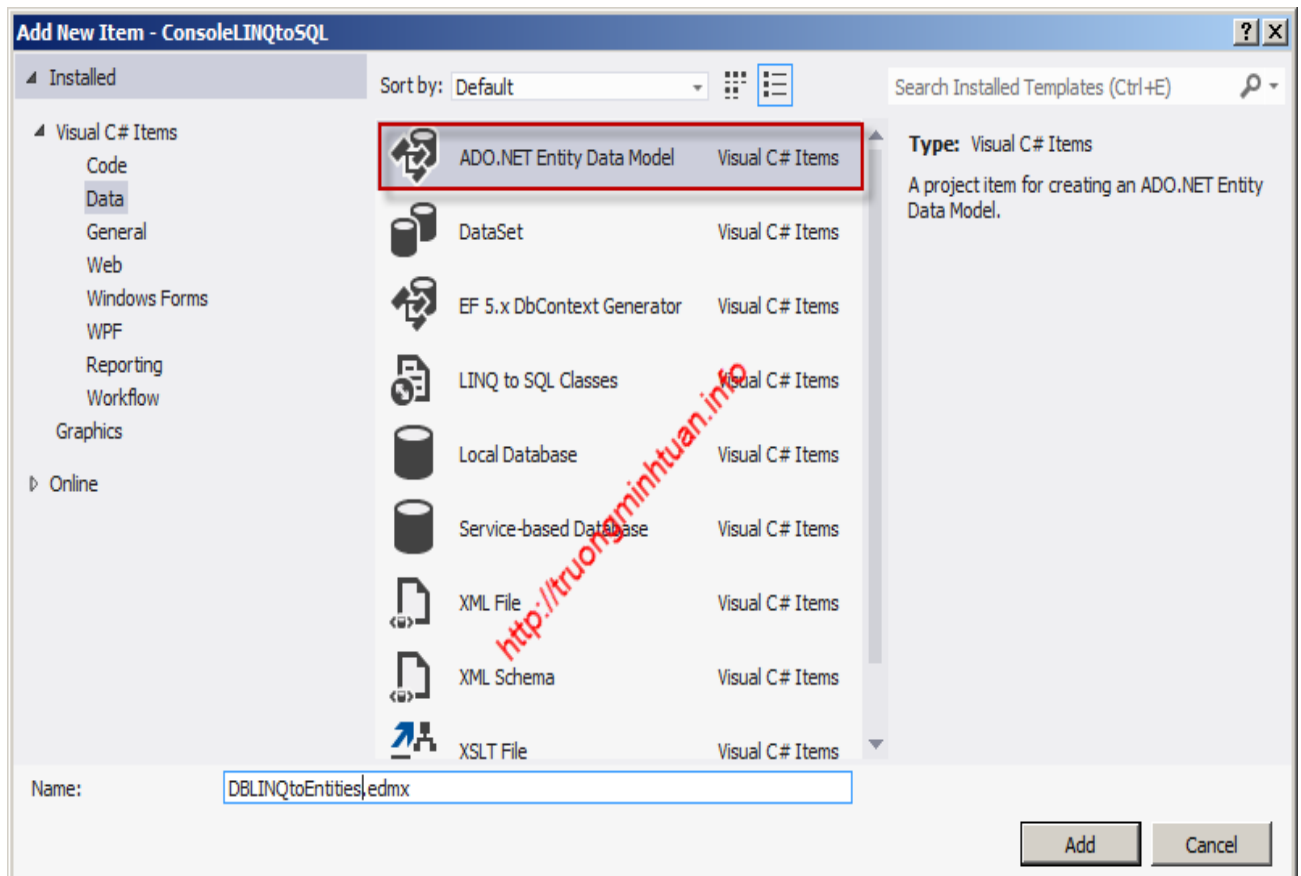
Ngày nay, LINQ to Entities lại được nhiều người lựa chọn. Vì tính chuyên nghiệp của nó. Kể cả Blog và các dự án của tôi hiện nay đều sử dụng LINQ to Entities.



Tôi sẽ hướng dẫn cho bạn vài bước thao tác Insert, Update, Delete dùng truy vấn LINQ với LINQ to Entities dưới đây.

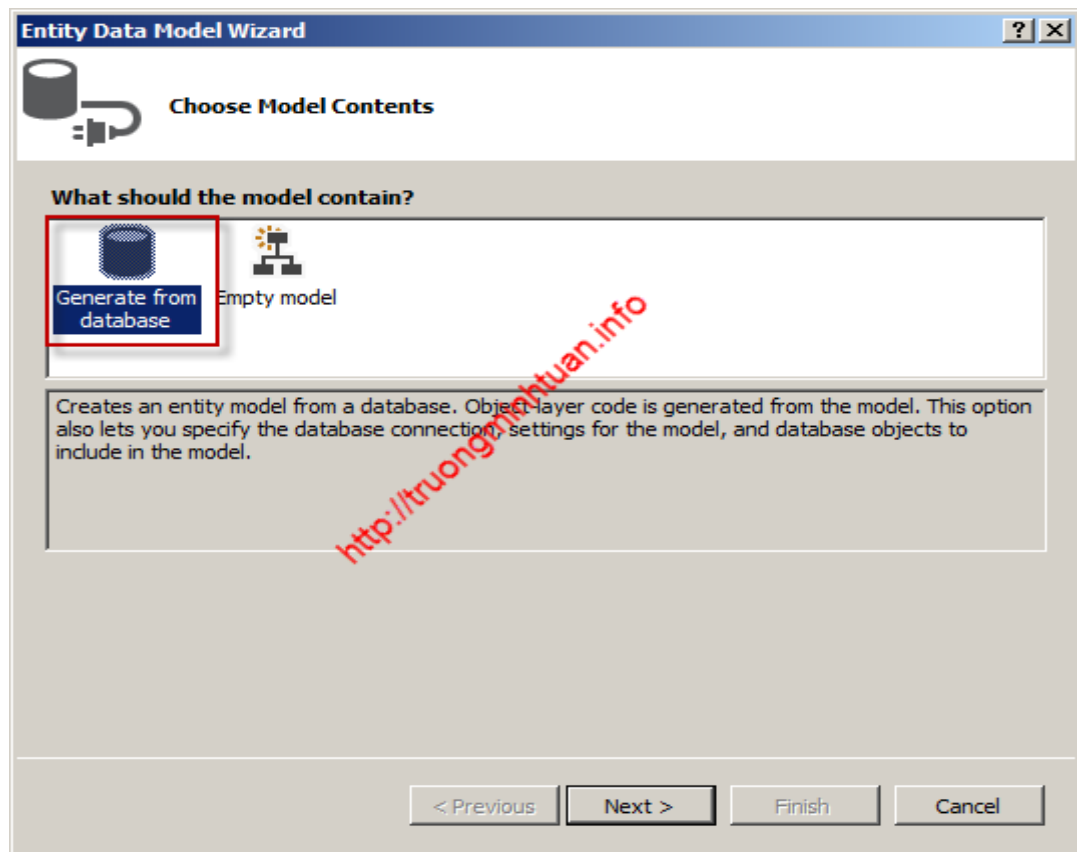
*Bước 01: Tôi dùng Database và Project đã được tạo ở trên để biểu diễn các bước trong LINQ to Entities.*

*Đầu tiên tôi chọn ADO.NET Entity Data Model thay vì chọn LINQ to SQL như loạt bài ở trên.*

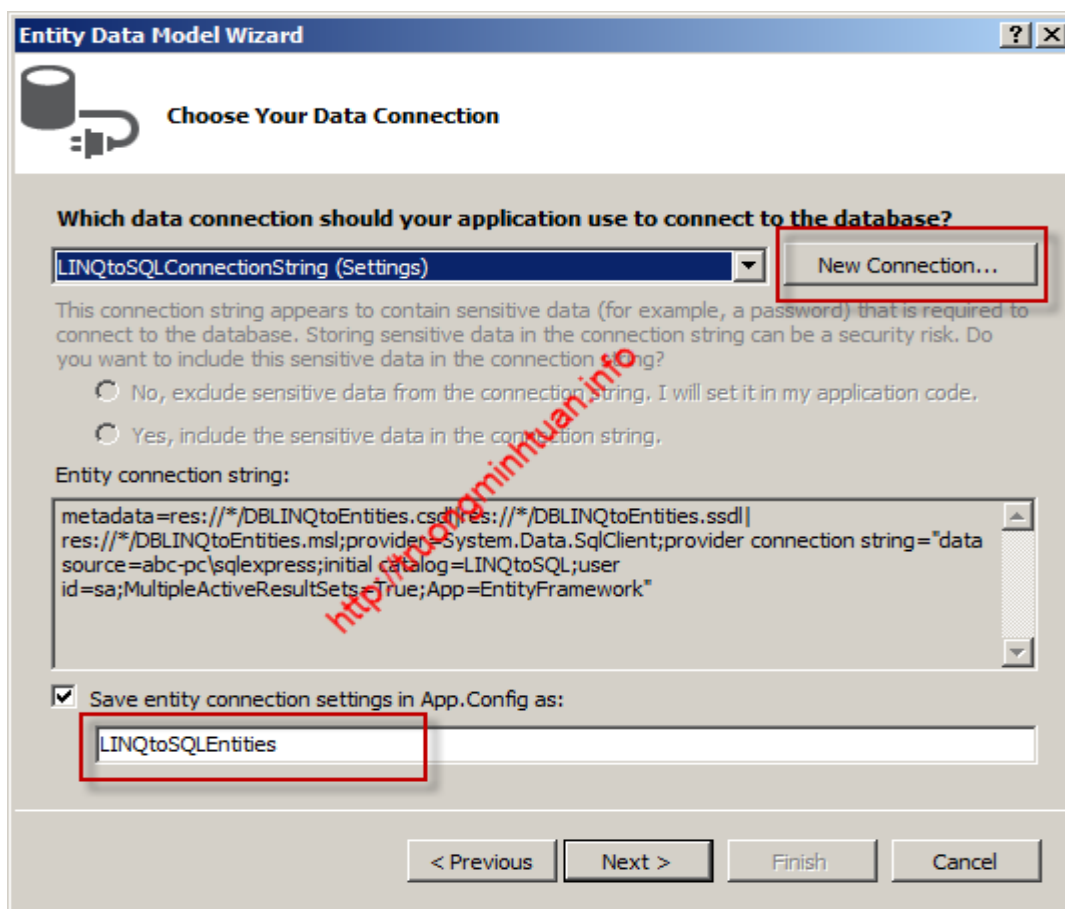




Bước 02: Chọn Generate from database, vì tôi đã có sẵn database ở SQL Server



Bước 03: Tạo một connectionString để kết nối với SQL Server (tương tự như phần LINQ to SQL bên trên)



The image shows the 'Entity Data Model Wizard' dialog box, specifically the 'Choose Your Data Connection' step. The title bar reads 'Entity Data Model Wizard'. Below the title bar is a database icon and the text 'Choose Your Data Connection'. The main question is 'Which data connection should your application use to connect to the database?'. A dropdown menu shows 'LINQtoSQLConnectionString (Settings)' selected. To the right of the dropdown is a 'New Connection...' button. Below this, there is a warning message: 'This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?'. There are two radio buttons: 'No, exclude sensitive data from the connection string, I will set it in my application code.' and 'Yes, include the sensitive data in the connection string.'. Below the radio buttons is a text box labeled 'Entity connection string:' containing the following text: 'metadata=res://\*/DBLINQtoEntities.csdl;res://\*/DBLINQtoEntities.ssdl|res://\*/DBLINQtoEntities.msl;provider=System.Data.SqlClient;provider connection string="data source=abc-pc\\sqlexpress;initial catalog=LINQtoSQL;user id=sa;MultipleActiveResultSets=True;App=EntityFramework"'. At the bottom, there is a checkbox labeled 'Save entity connection settings in App.Config as:' which is checked. Below the checkbox is a text box containing 'LINQtoSQLEntities'. At the very bottom are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'. A red watermark 'http://trungminhtuan.info' is visible across the center of the dialog.

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

LINQtoSQLConnectionString (Settings) New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string, I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Entity connection string:

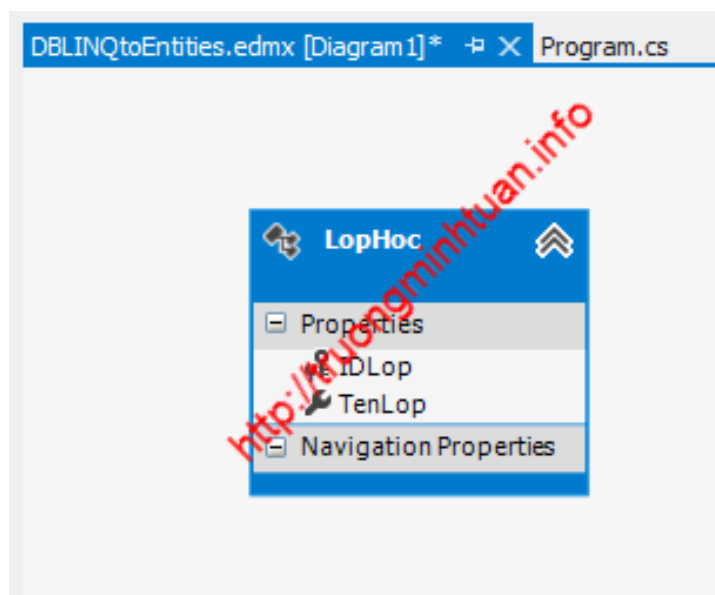
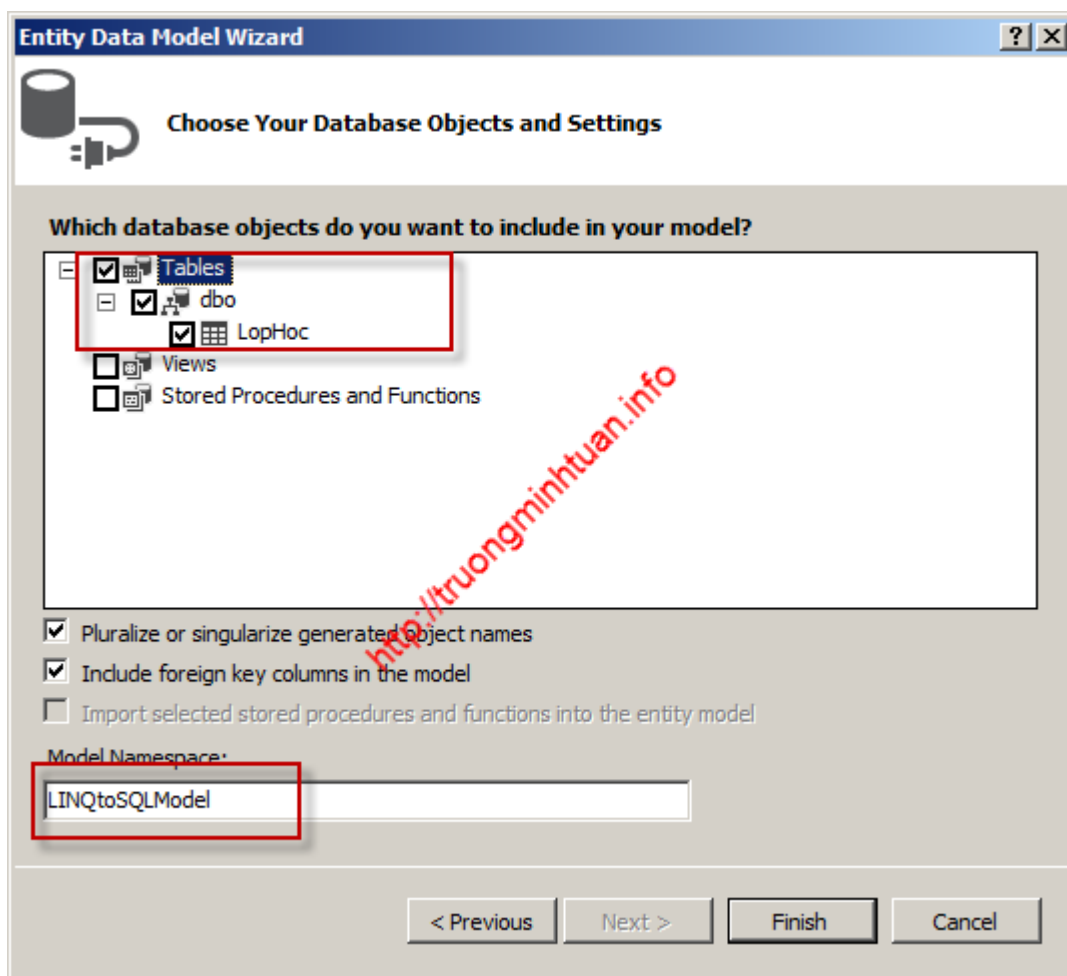
metadata=res://\*/DBLINQtoEntities.csdl;res://\*/DBLINQtoEntities.ssdl|res://\*/DBLINQtoEntities.msl;provider=System.Data.SqlClient;provider connection string="data source=abc-pc\\sqlexpress;initial catalog=LINQtoSQL;user id=sa;MultipleActiveResultSets=True;App=EntityFramework"

☒ Save entity connection settings in App.Config as:

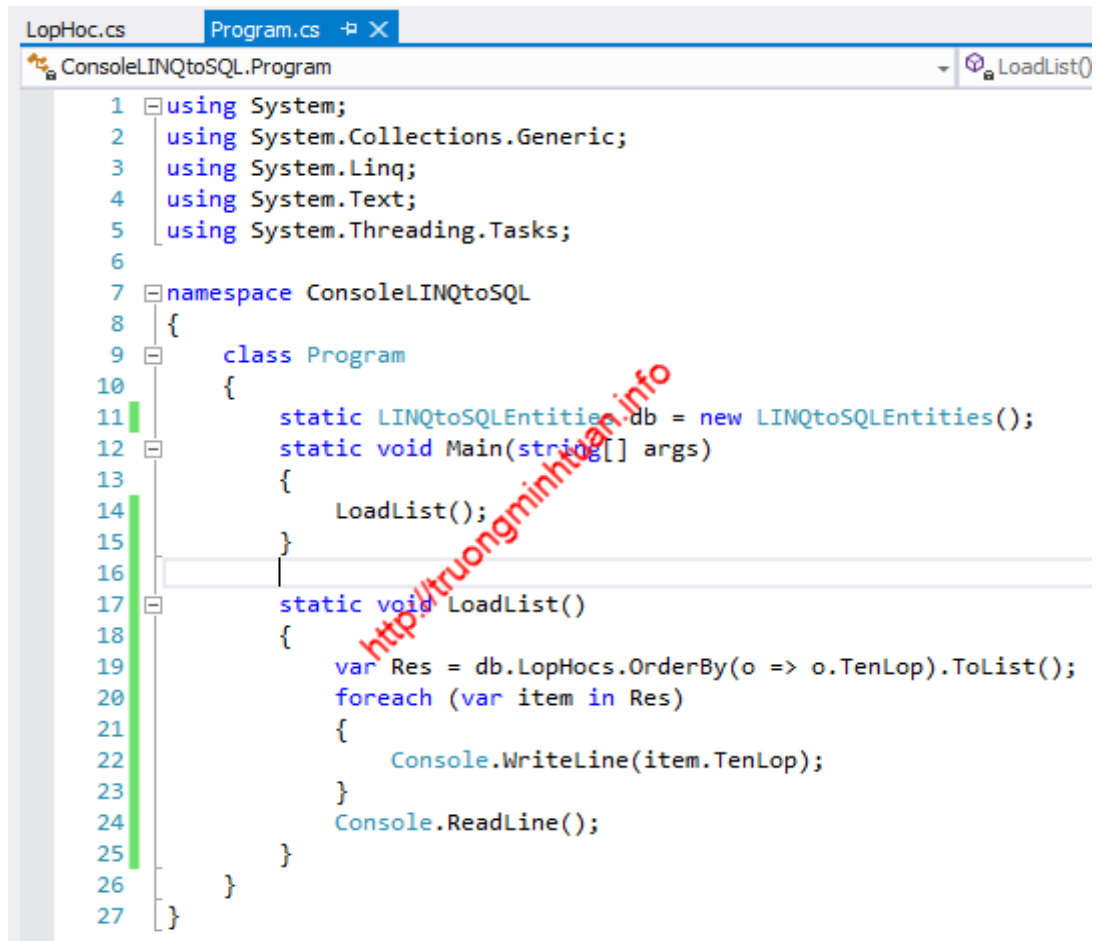
LINQtoSQLEntities

< Previous Next > Finish Cancel

Bước 04: Tại đây, tất cả Table, View, Store Procedure đều load lên để giúp bạn trong việc chọn lựa dễ dàng. Ở đây, tôi chỉ chọn Table, vì tôi không có View, SP.

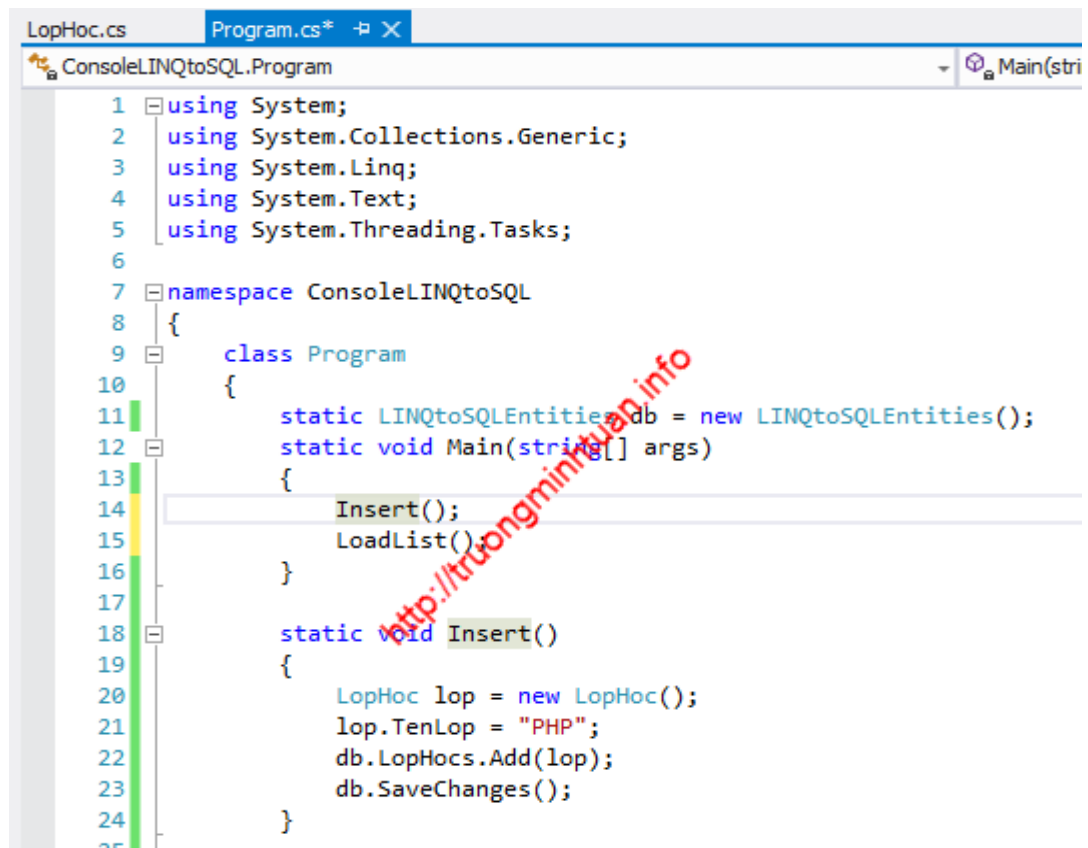


*Bước 05: Thực hiện thao tác Hiển thị dữ liệu dùng truy vấn LINQ*



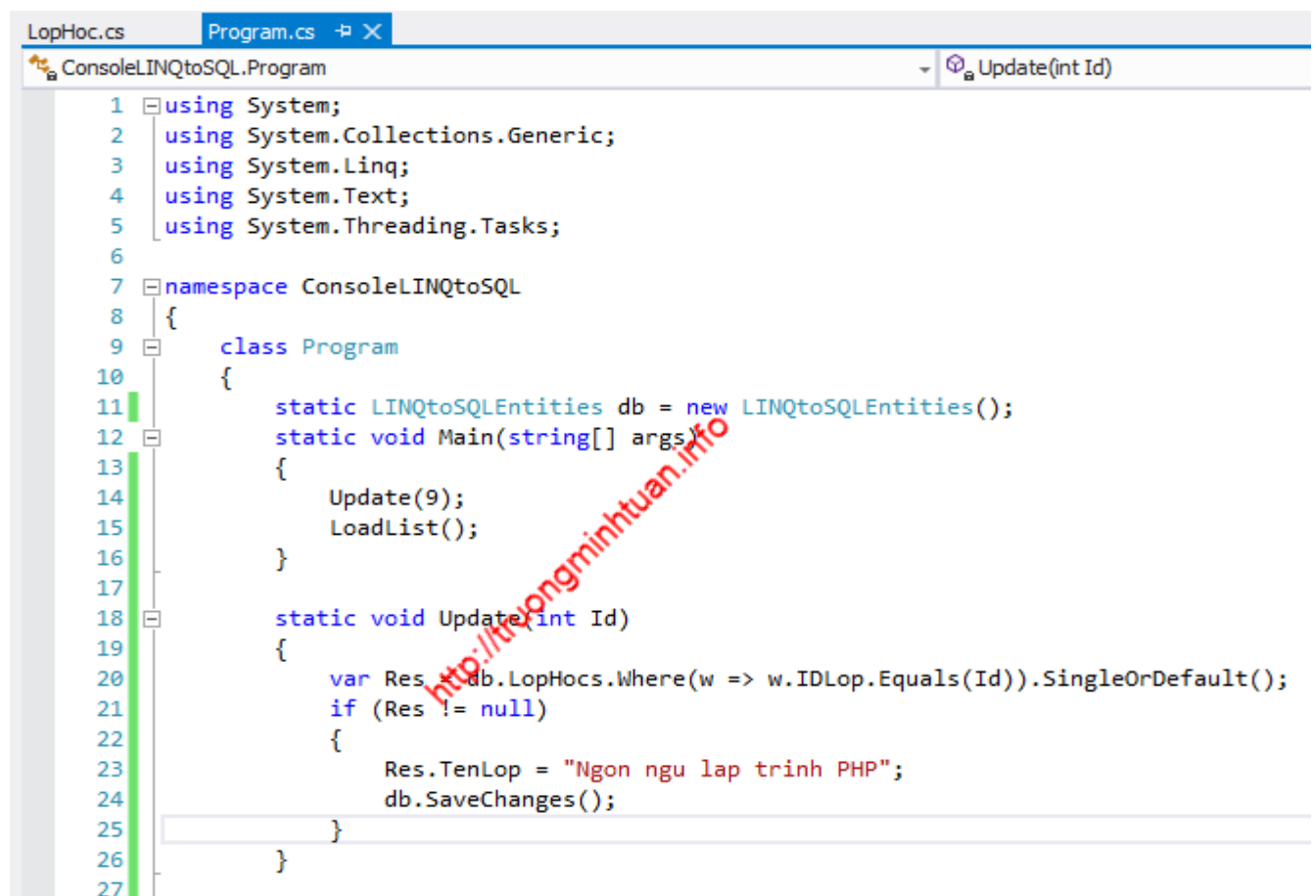
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleLINQtoSQL
8 {
9     class Program
10    {
11        static LINQtoSQLEntities db = new LINQtoSQLEntities();
12        static void Main(string[] args)
13        {
14            LoadList();
15        }
16
17        static void LoadList()
18        {
19            var Res = db.LopHocs.OrderBy(o => o.TenLop).ToList();
20            foreach (var item in Res)
21            {
22                Console.WriteLine(item.TenLop);
23            }
24            Console.ReadLine();
25        }
26    }
27 }
```

*Bước 06: Thực hiện thao tác Thêm mới dữ liệu dùng truy vấn LINQ*



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleLINQtoSQL
8 {
9     class Program
10    {
11        static LINQtoSQLEntities db = new LINQtoSQLEntities();
12        static void Main(string[] args)
13        {
14            Insert();
15            LoadList();
16        }
17
18        static void Insert()
19        {
20            LopHoc lop = new LopHoc();
21            lop.TenLop = "PHP";
22            db.LopHocs.Add(lop);
23            db.SaveChanges();
24        }
25    }
26 }
```

*Bước 07: Thực hiện thao tác Cập nhật dữ liệu dùng truy vấn LINQ*



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleLINQtoSQL
8 {
9     class Program
10    {
11        static LINQtoSQLEntities db = new LINQtoSQLEntities();
12        static void Main(string[] args)
13        {
14            Update(9);
15            LoadList();
16        }
17
18        static void Update(int Id)
19        {
20            var Res = db.LopHocs.Where(w => w.IDLop.Equals(Id)).SingleOrDefault();
21            if (Res != null)
22            {
23                Res.TenLop = "Ngôn ngữ lập trình PHP";
24                db.SaveChanges();
25            }
26        }
27    }
28 }
```

*Bước 08: Thực hiện thao tác Xóa dữ liệu dùng truy vấn LINQ*

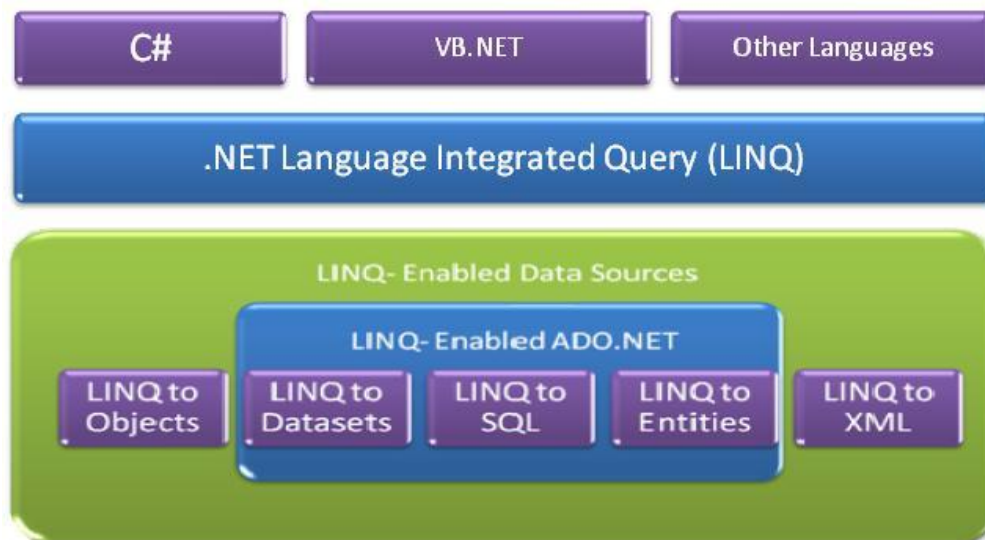
```
LopHoc.cs  Program.cs  Insert()
ConsoleLINQtoSQL.Program
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleLINQtoSQL
8  {
9      class Program
10     {
11         static LINQtoSQLEntities db = new LINQtoSQLEntities();
12         static void Main(string[] args)
13         {
14             Delete(9);
15             LoadList();
16         }
17
18         static void Delete(int Id)
19         {
20             var Res = db.LopHocs.Where(w => w.IDLop.Equals(Id)).SingleOrDefault();
21             if (Res != null)
22             {
23                 db.LopHocs.Remove(Res);
24                 db.SaveChanges();
25             }
26         }
27     }
28 }
```

## 9. LINQ to Object

LINQ to Object cung cấp sử dụng bất kỳ cú pháp truy vấn LINQ có hỗ trợ `IEnumerable<T>`, nhằm truy cập vào bộ nhớ của collection mà không cần có LINQ provider (API) như các trường hợp bạn đã biết: LINQ to SQL, LINQ to XML.

Truy vấn trong LINQ to Objects trả về duy nhất kiểu `IEnumerable<T>`. Cách truy vấn gần tương tự như các truy vấn khác. Khi sử dụng LINQ to Object trong vòng lặp `foreach` thì trở nên mạnh mẽ hơn: đọc, lọc, nhóm,...

Hiện nay, tôi cũng thường sử dụng LINQ to Object vào các dự án MVC nhằm nâng cao khả năng tiện dụng trong quá trình triển khai chương trình.



### Bước 01: Tạo 2 Class tương ứng

```
Program.cs  Program.cs
ConsoleLINQtoSQL.DataSource

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleLINQtoSQL
8 {
9     class LopHoc
10    {
11        public int IDLop { get; set; }
12        public string TenLop { get; set; }
13    }
14
15    class Sinhvien
16    {
17        public int SinhvienID { get; set; }
18        public string MaSV { get; set; }
19        public string Hoten { get; set; }
20        public bool Phai { get; set; }
21        public int IDLop { get; set; }
22    }
23 }
```

### Bước 02: Tạo dữ liệu cho 2 class trên

```
class DataSource
{
    public static List<LopHoc> lop = new List<LopHoc>(){
        new LopHoc {IDLop=1,TenLop="ASP.NET MVC"},
        new LopHoc {IDLop=2,TenLop="ASP.NET WebForm"},
        new LopHoc {IDLop=3,TenLop="Node.JS"}
    };

    public static List<Sinhvien> sv = new List<Sinhvien>() {
        new Sinhvien {SinhvienID=1,MaSV="001",Hoten="Ty",Phai=true,IDLop=1},
        new Sinhvien {SinhvienID=2,MaSV="002",Hoten="Suu",Phai=true,IDLop=3},
        new Sinhvien {SinhvienID=3,MaSV="003",Hoten="Dan",Phai=false,IDLop=1},
        new Sinhvien {SinhvienID=4,MaSV="004",Hoten="Meo",Phai=true,IDLop=2},
        new Sinhvien {SinhvienID=5,MaSV="005",Hoten="Thin",Phai=false,IDLop=2},
        new Sinhvien {SinhvienID=6,MaSV="006",Hoten="Ngo",Phai=false,IDLop=3},
        new Sinhvien {SinhvienID=7,MaSV="007",Hoten="Mui",Phai=true,IDLop=3},
        new Sinhvien {SinhvienID=8,MaSV="008",Hoten="Than",Phai=false,IDLop=1},
        new Sinhvien {SinhvienID=9,MaSV="009",Hoten="Dau",Phai=true,IDLop=2},
        new Sinhvien {SinhvienID=10,MaSV="010",Hoten="Tuat",Phai=true,IDLop=1}
    };
}
```



### Bước 03: Hiển thị dữ liệu thông qua truy vấn LINQ

```
46 class Program
47 {
48
49     static void Main(string[] args)
50     {
51         //Hiển thị danh sách Sinh viên
52         var Res = DataSource.sv.OrderBy(o => o.Hoten).ToList();
53         foreach (var item in Res)
54         {
55             Console.WriteLine(item.Hoten);
56         }
57
58         var Res1 = from a in DataSource.sv
59                     orderby a.Hoten ascending
60                     select a;
61         foreach (var item in Res1)
62         {
63             Console.WriteLine(item.Hoten);
64         }
65         Console.ReadLine();
66     }
67 }
68 }
```

### Hiển thị dữ liệu dùng từ khoá join

```
46 class Program
47 {
48
49     static void Main(string[] args)
50     {
51         //Join 2 table Lophoc and Sinh vien
52         var Res = from a in DataSource.sv
53                     join b in DataSource.lop
54                     on a.IDLop equals b.IDLop
55                     select new {
56                         TenLop=b.TenLop,
57                         Hoten=a.Hoten,
58                         Phai=a.Phai
59                     };
60
61         foreach (var item in Res)
62         {
63             Console.WriteLine("Lop:" + item.TenLop
64                               + "\tHoTen: " + item.Hoten
65                               + "\tPhai: " + item.Phai);
66         }
67         Console.ReadLine();
68     }
69 }
70 }
```

## Hiển thị dữ liệu theo dạng Join, GroupBy

```
toSQL..Program Main(string[] args)
class Program
{
    static void Main(string[] args)
    {
        //Join 2 table Lophoc and Sinh vien
        var sql = from l in DataSource.lop
                  join s in DataSource.sv
                  on l.IDLop equals s.IDLop
                  where s.Hoten.StartsWith("T")
                  select s;
        foreach (var item in sql)
        {
            Console.WriteLine(item.Hoten);
        }
        Console.WriteLine();
        //Group 2 table Lophoc and sinh vien
        IEnumerable<IGrouping<int, Sinhvien>> Res = from a in DataSource.sv
                                                    group a by a.IDLop into g
                                                    select g;

        foreach (var group in Res)
        {
            Console.WriteLine("IDLop:" + group.Key);
            foreach (var value in group)
            {
                Console.WriteLine("Ho Ten: " + value.Hoten);
            }
        }
        Console.ReadLine();
    }
}
```

## 10. TÀI LIỆU THAM KHẢO

- [1]. [https://msdn.microsoft.com/en-us/library/bb397926\(v=vs.90\).aspx](https://msdn.microsoft.com/en-us/library/bb397926(v=vs.90).aspx)
- [2]. <http://Asp.net>
- [3]. <http://Asp.net.vn>
- [4]. <http://stackoverflow.com>
- [5]. <http://dotnet-tricks.com>

---

Để hoàn thành tốt quyển tài liệu **LINQ step by step**, tôi có sử dụng một số nguồn tư liệu từ Internet và Blog của những anh, chị trong lẫn ngoài nước. Nếu có gì không hài lòng mong quý anh, chị thông cảm.

---

Mọi thắc mắc xin vui lòng liên hệ:

Trương Minh Tuấn  
eMail: ifsoft@live.com  
Skype: tieudinhtuan  
Website: <http://truongminhtuan.info>