Sql server, .net and c# video tutorial

Free C#, .Net and Sql server video tutorial for beginners and intermediate programmers.

Part 5 - Aggregate function in LINQ

Suggested Videos
Part 2 - Writing LINQ Queries
Part 3 - Extension Methods
Part 4 - LINQ Aggregate Functions

In this video we will discuss the use of **Aggregate**() LINQ function. In Part 4 of LINQ Tutorial, we discussed the following functions.
Min
Max
Sum
Count
Average

Let us understand the use of **Aggregate**() function with examples.

Complete Tutorials
Angular tutorial for beginners
Angular 5 Tutorial for beginners

Important Videos
The Gift of Education
Web application for your business
How to become .NET developer
Resources available to help you

Example 1: Consider the following string array.

```
string[] countries = { "India", "US", "UK", "Canada", "Australia" };
```

We want to combine all these strings into a single comma separated string. The output of the program should be as shown below.
India, US, UK, Canada, Australia

Without LINQ, the program will be as shown below.

```
using System;
namespace Demo
{
   class Program
   {
      static void Main()
      {
         string[] countries = { "India", "US", "UK", "Canada", "Australia" };

         string result = string.Empty;
         for (int i = 0; i < countries.Length; i++)
         {
            result = result + countries[i] + ", ";
         }

         int lastIndex = result.LastIndexOf(",");
         result = result.Remove(lastIndex);

         Console.WriteLine(result);
      }
   }
}
```

With LINQ Aggregate function

```
using System;
using System.Linq;
namespace Demo
{
   class Program
   {
      static void Main()
      {
         string[] countries = { "India", "US", "UK", "Canada", "Australia" };

         string result = countries.Aggregate((a, b) => a + ", " + b);

         Console.WriteLine(result);
      }
   }
}
```

How Aggregate() function works?
Step 1. First **"India"** is concatenated with **"US"** to produce result **"India, US"**
Step 2. Result in **Step 1** is then concatenated with **"UK"** to produce result **"India, US, UK"**
Step 3: Result in **Step 2** is then concatenated with **"Canada"** to produce result **"India, US, UK, Canada"**

This goes on until the last element in the array to produce the final single string **"India, US, UK, Canada, Australia"**

Example 2: Consider the following integer array

```
int[] Numbers = { 2, 3, 4, 5 };
```

Dot Net Video Tutorials

ASP.NET Core Tutorial
Angular 6 Tutorial
Angular CRUD Tutorial
Angular CLI Tutorial
Angular 2 Tutorial
Design Patterns
SOLID Principles
ASP.NET Web API
Bootstrap
AngularJS Tutorial
jQuery Tutorial
JavaScript with ASP.NET Tutorial
JavaScript Tutorial
Charts Tutorial
LINQ
LINQ to SQL
LINQ to XML
Entity Framework
WCF
ASP.NET Web Services
Dot Net Basics
C#
SQL Server
ADO.NET
ASP.NET
GridView
ASP.NET MVC
Visual Studio Tips and Tricks
Dot Net Interview Questions

Slides

Entity Framework
WCF
ASP.NET Web Services
Dot Net Basics
C#

Compute the product of all numbers

Without LINQ

```
using System;
namespace Demo
{
   class Program
   {
      static void Main()
      {
         int[] Numbers = { 2, 3, 4, 5 };

         int result = 1;
         foreach (int i in Numbers)
         {
            result = result * i;
         }

         Console.WriteLine(result);
      }
   }
}
```

With LINQ:

```
using System;
using System.Linq;
namespace Demo
{
   class Program
   {
      static void Main()
      {
         int[] Numbers = { 2, 3, 4, 5 };

         int result = Numbers.Aggregate((a, b) => a * b);

         Console.WriteLine(result);
      }
   }
}
```

How Aggregate() function works?
Step 1: Multiply **(2X3)** to produce result **6**
Step 2: Result **(6)** in **Step 1** is then multiplied with **4 (6X4)** to produce result **24**
Step 3: Result **(24)** in **Step 2** is then multiplied with **5 (24X5)** to produce final result **120**

Example 3: Consider the following integer array

```
int[] Numbers = { 2, 3, 4, 5 };
```

One of the overloaded version of **Aggregate()** function has a **Seed** parameter. If we pass **10** as the value for Seed parameter

```
int result = Numbers.Aggregate(10, (a, b) => a * b);
```

1200 will be the result

Step 1: Multiply **(10X2)** to produce result **20**
Step 2: Result **(20)** in **Step 1** is then multiplied with **3 (20X3)** to produce result **60**
Step 3: Result **(60)** in **Step 2** is then multiplied with **4 (60X4)** to produce result **240**
Step 4: Result **(240)** in **Step 3** is then multiplied with **5 (240X5)** to produce final result **1200**

SQL Server

ADO.NET

ASP.NET

GridView

ASP.NET MVC

Visual Studio Tips and Tricks

Java Video Tutorials

Part 1 : Video | Text | Slides

Part 2 : Video | Text | Slides

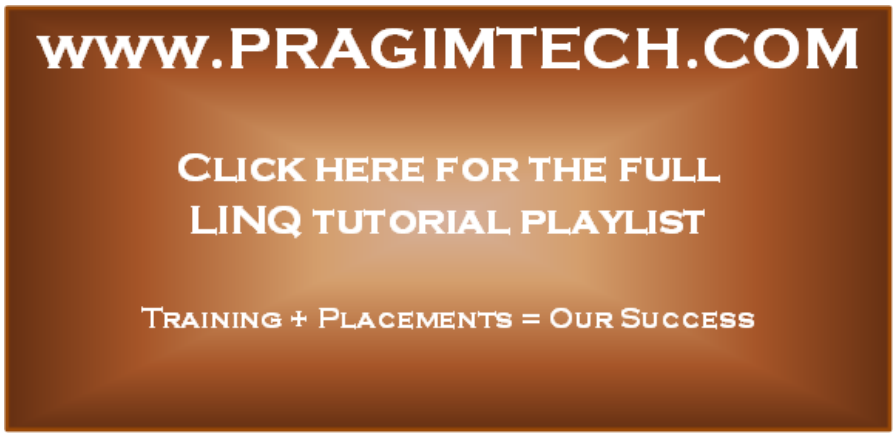
Part 3 : Video | Text | Slides

Interview Questions

C#

SQL Server

Written Test



1 comment:

Ahmet Toka July 22, 2014 at 5:50 PM

I watched all videos.All of them are perfect. Thank you for sharings.

Reply

Enter your comment...

Comment as: Hiện Doãn (C ▼ Sign out

Publish Preview Notify me

If you like this website, please share with your friends on facebook and Google+ and recommend us on google using the g+1 button on the top right hand corner.

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Powered by Blogger.