Delegates Demo

Delegates

# Objectives

* 1. Demonstrate how to write a delegate.
  2. Demonstrate how to pass a delegate as a method parameter.
  3. Demonstrate how to define and use lambda expressions.

# CREATE NEW Project

* 1. Create a new C# console application and call it DelegatesDemo.

# Define Delegate

* 1. Inside of program.cs add the following delegate definition inside of the program class:

//Delegate signature is return string, no arguments

delegate void DispStrDelegate(string param);

* 1. Add the following method definitions:

// Method that capatilizes a string.

static void Capitalize(string text)

{

Console.WriteLine("Your input capatilized --> "+text.ToUpper());

}

// Method that lower cases a string.

static void LowerCased(string text)

{

Console.WriteLine("Your input lower cased --> "+text.ToLower());

}

* 1. Inside of the Main method add the following:

//Get a line of text to convert

Console.Write(“Please enter some text: “);

String text = Console.ReadLine();

//Instantiate three delegate objects

DispStrDelegate saying1 = new DispStrDelegate(Capitalize);

DispStrDelegate saying2 = new DispStrDelegate(LowerCased);

DispStrDelegate saying3 = new DispStrDelegate(Console.WriteLine);

//Call them one after the other

saying1(text);

saying2(text);

saying3(text);

* 1. Run the program. Notice that Capitalize, LowerCased and Writeline are called using the delegates.

# Multi-cast Delegate

* 1. Add the following code to Main():

//Get another text line

Console.Write("Please enter some text: ");

text = Console.ReadLine();

//Make a new delegate object and concatenate delegates

DispStrDelegate sayings = new DispStrDelegate(Capitalize);

sayings += new DispStrDelegate(LowerCased);

sayings += new DispStrDelegate(Console.WriteLine);

//Call the one delegate and run all three methods

Console.WriteLine("Running multi cast directly: ");

Sayings(text);

* 1. Run the program. Notice that Capitalize, LowerCased and Writeline are called using the just one delegate now!

# Passing a delegate as a method parameter

* 1. Add the following method to the program class:

// Method that takes a delegate as an argument

static void RunMyDelegate(DispStrDelegate del, string textParam)

{

del(textParam);

}

* 1. Add the following to Main():

//Pass delegate as a method argument

Console.WriteLine("Running by passing delegate to another method: ");

RunMyDelegate(sayings, text);

* 1. Run the program. Notice that the sayings delegate is passed to RunMyDelegate and is run by it.

# Delegates

* 1. Add the following to Main():

//Create and run a lambda expression

Console.WriteLine("Running by passing in a lambda expression: ");

RunMyDelegate((string t) => { Console.WriteLine("Lambda: " + t); }, text);

* 1. Run the program. Notice that the Lambda expression is passed in as a delegate and runs.
  2. Try the following variations on delegate syntax:

//Remove the type and let it be infered

Console.WriteLine("Lambda without type: ");

RunMyDelegate((t)=> { Console.WriteLine("Lambda: "+t); }, text);

//Remove the parenthesis

Console.WriteLine("Lambda without parenthesis: ");

RunMyDelegate(t => { Console.WriteLine("Lambda2: " + t); }, text);

* 1. Notice that they all work the same.
  2. Try the following:

//Add a lambda expression to our delegate

sayings += t => { Console.WriteLine("Lambda3: " + t); };

Console.WriteLine("Three Delegates and a lambda: ");

RunMyDelegate(sayings, text);

* 1. Notice that you can add a lambda to a multicast delegate and it also works.