Armen Melkumyan • 1st
Technical / Solutions Architect
1yr •

🌟 From .Net C# Interviews: Explanation of Delegates, Actions, Funcs, and Events 🌟

🔶 What's a Delegate?

Imagine having a type-safe function pointer that encapsulates a method. Delegates are your go-to for flexible method referencing, making them essential for sophisticated event handling and dynamic callback scenarios. They are the backbone of event-driven programming in .NET.

🔶 Action & Func – Your Code's Best Friends

Need to simplify your method references? Use Actions and Funcs!

Actions are perfect when you don't need any return values from your methods – just pure execution.

Funcs come into play when your methods must return a result. They help keep your code clean and your intentions clear.

🔶 Events – The Communication Hubs

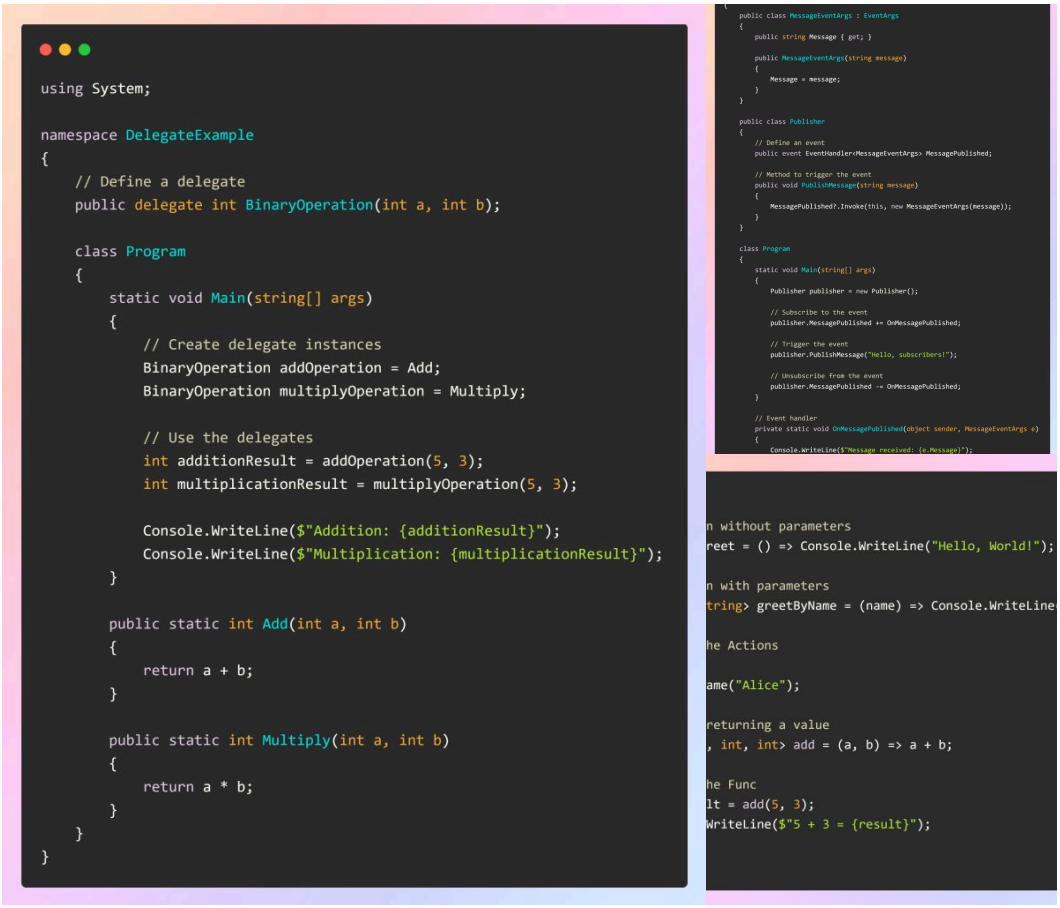
Events in C# are what keep different parts of your application in sync. They use delegates to multicast, allowing multiple event handlers to listen in on events and act accordingly. Perfect for creating highly responsive and interactive applications!

📘 Here are some practical examples to integrate these concepts into your next project and boost its efficiency and scalability:

1. Delegates for custom sorting algorithms.
2. Actions to handle simple notifications.
3. Funcs for data processing with return values.
4. Events to manage user interactions in GUI applications.

Bellow code snippets.

#Programming #SoftwareDevelopment #DotNet #EventDrivenArchitecture #Csharp



```
using System;

namespace DelegateExample
{
    // Define a delegate
    public delegate int BinaryOperation(int a, int b);

    class Program
    {
        static void Main(string[] args)
        {
            // Create delegate instances
            BinaryOperation addOperation = Add;
            BinaryOperation multiplyOperation = Multiply;

            // Use the delegates
            int additionResult = addOperation(5, 3);
            int multiplicationResult = multiplyOperation(5, 3);

            Console.WriteLine($"Addition: {additionResult}");
            Console.WriteLine($"Multiplication: {multiplicationResult}");
        }

        public static int Add(int a, int b)
        {
            return a + b;
        }

        public static int Multiply(int a, int b)
        {
            return a * b;
        }
    }
}
```

```
public class MessageEventArgs : EventArgs
{
    public string Message { get; }

    public MessageEventArgs(string message)
    {
        Message = message;
    }
}

public class Publisher
{
    // Define an event
    public event EventHandler<MessageEventArgs> MessagePublished;

    // Method to trigger the event
    public void PublishMessage(string message)
    {
        MessagePublished?.Invoke(this, new MessageEventArgs(message));
    }
}

class Program
{
    static void Main(string[] args)
    {
        Publisher publisher = new Publisher();

        // Subscribe to the event
        publisher.MessagePublished += OnMessagePublished;

        // Trigger the event
        publisher.PublishMessage("Hello, subscribers!");

        // Unsubscribe from the event
        publisher.MessagePublished -= OnMessagePublished;
    }

    // Event handler
    private static void OnMessagePublished(object sender, MessageEventArgs e)
    {
        Console.WriteLine($"Message received: {e.Message}");
```

```
n without parameters
reet = () => Console.WriteLine("Hello, World!");

n with parameters
tring> greetByName = (name) => Console.WriteLine

he Actions

ame("Alice");

returning a value
, int, int> add = (a, b) => a + b;

he Func
lt = add(5, 3);
WriteLine($"5 + 3 = {result}");
```

63 3 reposts