



Armen Melkumyan • 1st

Technical / Solutions Architect

1yr • Edited •

...

From .Net C# technical interview: Unique Identifier

Challenge: Develop a system to generate unique identifiers across multiple application instances in a multi-threaded environment. Your mission, should you choose to accept it, is to ensure these identifiers are unique and generated efficiently without stepping on each other's toes.

Steps for Solution

- 1) Understand the Requirements: Realize the necessity for unique identifiers to be unique not only in a single instance but across multiple instances possibly running on different machines .
- 2) Research UUIDs: Look into universally unique identifier (UUID) standards and how they ensure uniqueness across distributed systems .
- 3) Thread Safety: Implement a method to generate these identifiers that is thread-safe, ensuring that concurrent access by multiple threads does not lead to duplicate identifier generation .
- 4) Optimization: Consider the performance implications of your solution. Ensure it is optimized for fast execution, given that it might be called frequently .

What Knowledge Will Be Validated:

- Understanding of thread safety and concurrency in .NET.
- Knowledge of distributed systems and how to generate unique identifiers across them.
- Ability to write optimized and efficient C# code.
- Proficiency in writing unit tests in C# to validate code functionality.

Github repo: <https://lnkd.in/dJjZCBWY>

#DotNet #CSharp #ProgrammingPuzzles #Concurrency

#ThreadSafety #SoftwareDevelopment #CodingChallenge

```
using System;
using System.Collections.Concurrent;
using System.Threading;

public class UniqueIdentifierGenerator
{
    // Method to generate unique identifier
    public static YourType GenerateUniqueIdentifier()
    {
    }
}

class Program
{
    // define here generatedIdentifiers

    static void GenerateAndStoreUniqueIdentifier()
    {
        // your logic here
        Console.WriteLine($"Generated Unique Identifier: {id}");
    }

    static void Main(string[] args)
    {
        const int numberOfThreads = 100;
        var threads = new Thread[numberOfThreads];

        for (int i = 0; i < numberOfThreads; i++)
        {
            threads[i] = new Thread(GenerateAndStoreUniqueIdentifier);
            threads[i].Start();
        }

        // Wait for all threads to complete
        foreach (var thread in threads)
        {
            thread.Join();
        }

        Console.WriteLine($"Total Unique Identifiers Generated: {generatedIdentifiers.Count}");
    }
}
```

Armen Melkumyan

   54

2 comments 6 reposts