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

From .Net C# interview questions

Question:

Explain please Memory<T>

Answer:

Memory<T> is similar to Span<T> but is not a stack-only type. It can be stored in fields, used in async methods, and is suitable for representing slices of arrays or memory that persist beyond a single method scope.

Key Methods and Properties:

Slice: Like Span<T>, it slices the memory block.

Span: Converts the Memory<T> instance into a Span<T> for stack-only, memory-safe operations.

Length, IsEmpty: Properties to get information about the memory block.

Under the Hood:

Managed Memory: Primarily used to represent managed memory (like arrays) but can also represent unmanaged memory.

Garbage Collection: The runtime tracks references to Memory<T> blocks to ensure that the underlying memory is not collected prematurely.

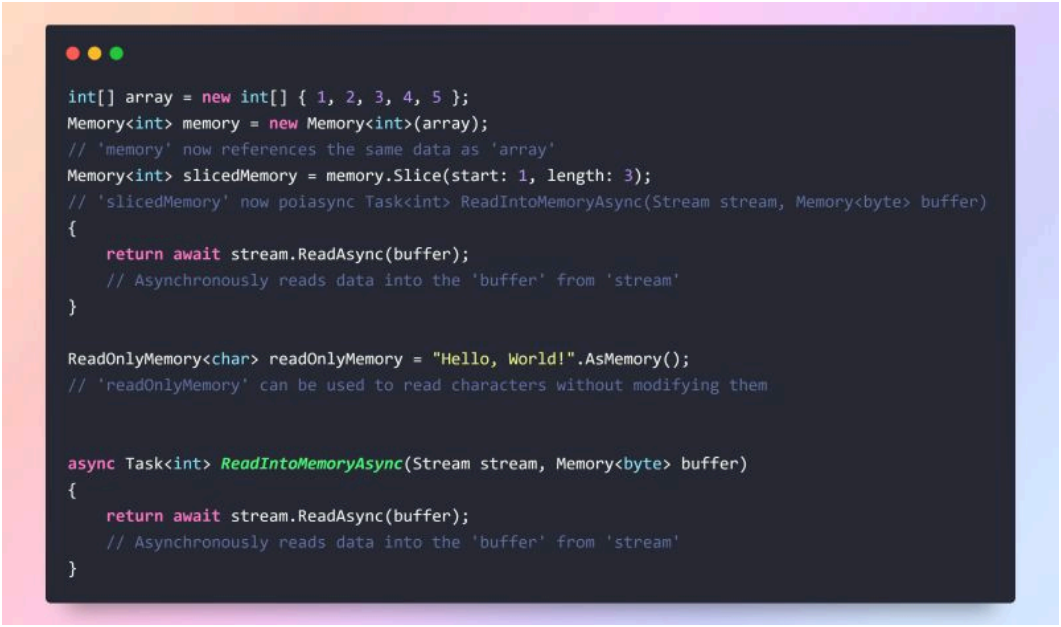
#CSharp #DotNet #SoftwareDevelopment

#MemoryManagement #CodingInterview #CSharpProgramming

#DotNetDevelopers #ProgrammingLanguages #TechInterviews

#DeveloperCommunity #CodingTips #ComputerScience

#AdvancedProgramming #TechCommunity #DataStructures



```
int[] array = new int[] { 1, 2, 3, 4, 5 };
Memory<int> memory = new Memory<int>(array);
// 'memory' now references the same data as 'array'
Memory<int> slicedMemory = memory.Slice(start: 1, length: 3);
// 'slicedMemory' now poiasync Task<int> ReadIntoMemoryAsync(Stream stream, Memory<byte> buffer)
{
    return await stream.ReadAsync(buffer);
    // Asynchronously reads data into the 'buffer' from 'stream'
}

ReadOnlyMemory<char> readOnlyMemory = "Hello, World!".AsMemory();
// 'readOnlyMemory' can be used to read characters without modifying them


async Task<int> ReadIntoMemoryAsync(Stream stream, Memory<byte> buffer)
{
    return await stream.ReadAsync(buffer);
    // Asynchronously reads data into the 'buffer' from 'stream'
}
```

5 1 repost