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

From C# .Net interview questions

Question:

Explain please Span<T> and how it work.

Answer:

Span<T> represents a contiguous region of arbitrary memory. It is a stack-only type that grants memory-safe access to array segments, strings, or unmanaged memory without allocations.

Key Methods and Properties:

1) Slice: Creates a new span that represents a sub-section of the original span.

2) Length: Gets the number of items in the span.

3) this[int index]: Indexer to access elements at specific positions.

4) ToArray: Converts the span to an array. This does cause an allocation.

Under the Hood:

ByRef-like Type: Span<T> is a ref struct, making it a stack-only type that can't be boxed or assigned to object or dynamic types.

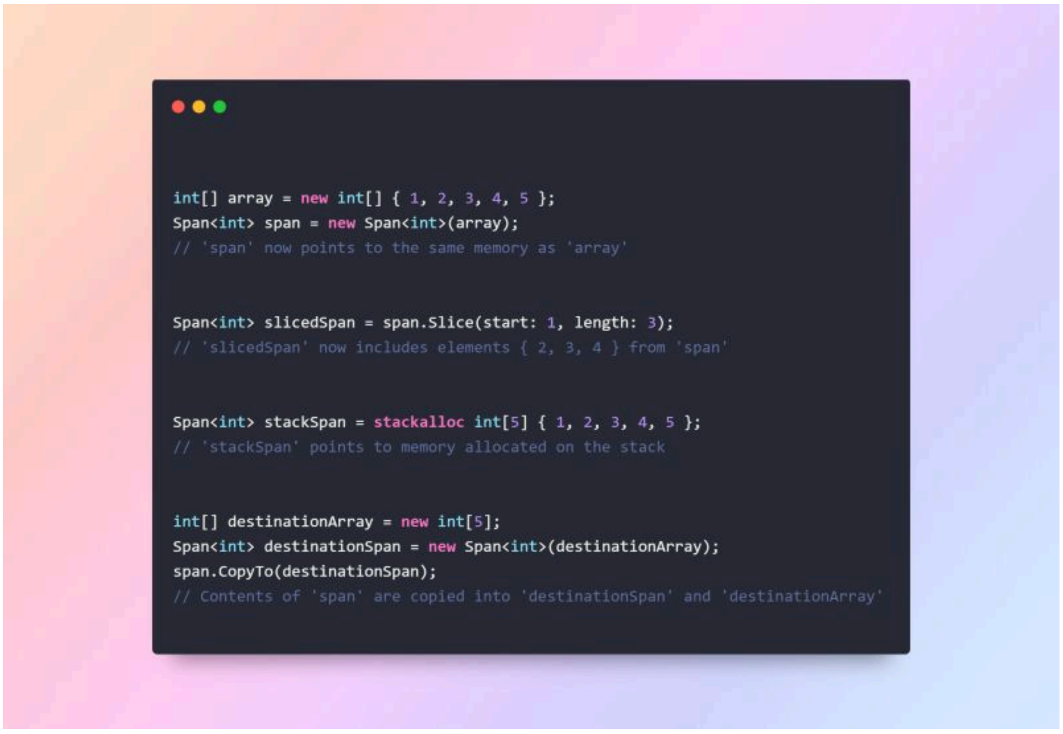
Memory Range: Internally, it holds a reference to the start of the memory block and the length of the span.

Safety: The runtime ensures that the memory referenced by a span is not subject to garbage collection and remains valid while the span is in use.

#CSharp #DotNet #SoftwareEngineering #DotNetCore

#MemoryManagement #CodingInterview #TechInterviews

#AdvancedProgramming #TechTips #DataStructures



```
int[] array = new int[] { 1, 2, 3, 4, 5 };
Span<int> span = new Span<int>(array);
// 'span' now points to the same memory as 'array'


Span<int> slicedSpan = span.Slice(start: 1, length: 3);
// 'slicedSpan' now includes elements { 2, 3, 4 } from 'span'


Span<int> stackSpan = stackalloc int[5] { 1, 2, 3, 4, 5 };
// 'stackSpan' points to memory allocated on the stack


int[] destinationArray = new int[5];
Span<int> destinationSpan = new Span<int>(destinationArray);
span.CopyTo(destinationSpan);
// Contents of 'span' are copied into 'destinationSpan' and 'destinationArray'
```

6