.NET Memory Expert

Course overview

Trainer

Konrad Kokosa
Dotnetos co-founder, Microsoft MVP, author, speaker

Author of the Pro .NET Memory Management book and independent consultant, blogger, speaker and a professional Tweeter. He shares his passion as a .NET trainer, especially in the performance and app diagnosis field.

Requirements

- Latest versions of the Chrome, Firefox, Safari or Microsoft Edge
- Basic knowledge of the .NET and C# language
- Visual Studio 2019 version 16.9.1 or higher. You may use any other IDE like JetBrains Rider or Visual Studio Code with Omnisharp for homework exercises, but please remember that some of the features shown in the course maybe not available.
- .NET 5 SDK (demos & exercises were tested on .NET 5.0.2) or higher
- nice to have Docker

Course agenda

The course contains 12 consecutive modules:

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose
6. **Roots, generations and memory leaks** - why do we have generations and what are the roots?!

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose
6. **Roots, generations and memory leaks** - why do we have generations and what are the roots?!
7. **The (necessary) GC internals** - some internals I believe it is good to know (not to much! 😇)

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose
6. **Roots, generations and memory leaks** - why do we have generations and what are the roots?!
7. **The (necessary) GC internals** - some internals I believe it is good to know (not to much! 😇)
8. **Allocations under control** - everything about allocations (and avoiding them)

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose
6. **Roots, generations and memory leaks** - why do we have generations and what are the roots?!
7. **The (necessary) GC internals** - some internals I believe it is good to know (not to much! 😇)
8. **Allocations under control** - everything about allocations (and avoiding them)
9. **Lifetime management** - why do we have **IDisposable** interface? whare are finalizers?

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose
6. **Roots, generations and memory leaks** - why do we have generations and what are the roots?!
7. **The (necessary) GC internals** - some internals I believe it is good to know (not to much! 😇)
8. **Allocations under control** - everything about allocations (and avoiding them)
9. **Lifetime management** - why do we have **IDisposable** interface? whare are finalizers?
10. **Advanced topics and APIs** - deep-dive into topics like **Span** and **Unsafe** class

Course agenda

The course contains 12 consecutive modules:

1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose
6. **Roots, generations and memory leaks** - why do we have generations and what are the roots?!
7. **The (necessary) GC internals** - some internals I believe it is good to know (not to much! 😇)
8. **Allocations under control** - everything about allocations (and avoiding them)
9. **Lifetime management** - why do we have **IDisposable** interface? whare are finalizers?
10. **Advanced topics and APIs** - deep-dive into topics like **Span** and **Unsafe** class
11. **Data Oriented Design** - way of programming with the memory access pattern in the centre

Course agenda

The course contains 12 consecutive modules:

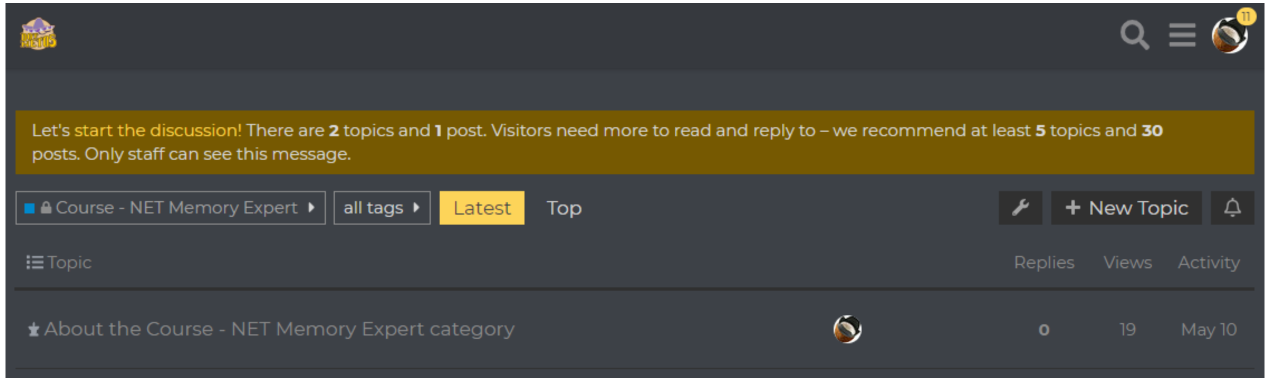
1. **.NET runtime** - introducing the whole .NET ecosystem a little, explaining things like IL and JIT
2. **Types basics** - the famous "value vs reference type"
3. **Memory management fundamentals** - what is automatic memory management
4. **.NET GC fundamentals** - high-level view of how .NET GC works
5. **Diagnostic tools** - comprehensive guide about available tools to monitor and diagnose
6. **Roots, generations and memory leaks** - why do we have generations and what are the roots?!
7. **The (necessary) GC internals** - some internals I believe it is good to know (not to much! 😇)
8. **Allocations under control** - everything about allocations (and avoiding them)
9. **Lifetime management** - why do we have `IDisposable` interface? whare are finalizers?
10. **Advanced topics and APIs** - deep-dive into topics like `Span` and `Unsafe` class
11. **Data Oriented Design** - way of programming with the memory access pattern in the centre
12. **Interoperability** - everyting about calling unmanaged code and how it cooperates with the GC

Certificate

We would like to reward you and motivate you to study by providing proof of course completion: personalized **.NET Memory Expert Certificate**! It will be automatically available after completing all lessons.

Discourse

- To increase your motivation and empower discussions we have invited you to the Dotnetos Community! Check your email and look for the invitation.
- Each participant has access to a private group that brings together all students of the course
- Feel free to ask questions and join discussion!



Let's start the discussion! There are **2** topics and **1** post. Visitors need more to read and reply to – we recommend at least **5** topics and **30** posts. Only staff can see this message.

Topic	Replies	Views	Activity
About the Course - NET Memory Expert category	0	19	May 10

Homework

- One, central repository for all homework in the course
- It's located on our Github account, https://github.com/dotnetos/memoryexpert-course
- Provided materials are licensed under license. Please read it before using
- Homework challenges are self-assignment tasks. Feel free to fork the repository and complete them on your own

Have a nice studying!

Some background...

Why GC in .NET topic?

Why GC in .NET topic?

- It is important and crucial because it is so complex

Why GC in .NET topic?

- It is important and crucial because it is so complex
- … greatly hidden because it is so good abstraction

Why GC in .NET topic?

- It is important and crucial because it is so complex
- … greatly hidden because it is so good abstraction
- … but

Why GC in .NET topic?

- It is important and crucial because it is so complex
- … greatly hidden because it is so good abstraction
- … but
- … eventually it makes som troubles

Why GC in .NET topic?

- It is important and crucial because it is so complex
- … greatly hidden because it is so good abstraction
- … but
- … eventually it makes som troubles
- As it is only a very well hidden giant **alien organism** in our program

High-performance in C#?!

High-performance in C#?!

- What's the point of using C# for high-performance, instead of using C/C++?!

High-performance in C#?!

- What's the point of using C# for high-performance, instead of using C/C++?!
  - only small percent of code requires such low-level - why use C/C++ for all the system then?

High-performance in C#?!

- What's the point of using C# for high-performance, instead of using C/C++?!
    - only small percent of code requires such low-level - why use C/C++ for all the system then?
    - (pro) C# job market is much bigger than (pro) C/C++ job market

High-performance in C#?!

- What's the point of using C# for high-performance, instead of using C/C++?!
  - only small percent of code requires such low-level - why use C/C++ for all the system then?
  - (pro) C# job market is much bigger than (pro) C/C++ job market
  - safety

High-performance in C#?!

- What's the point of using C# for high-performance, instead of using C/C++?!
    - only small percent of code requires such low-level - why use C/C++ for all the system then?
    - (pro) C# job market is much bigger than (pro) C/C++ job market
    - safety
- Example: Ixy network device drivers written in high-level languages