Enterprise Application Systems (EAS)

Why you should care about it

Daniel Fink

Institute for Computational Physics

June 15, 2022

What are Enterprise Application Systems?

Institute of Architecture of Application Systems

An (enterprise) application system is a software that directly supports one or more business activities of a company. Typical attributes for such a system are:

- It covers essential parts of the business activities of a company.
- It processes large amounts of persistent data.
- It allows simultaneous use by a large number of users.
- It exchanges information with similar systems.

Why should I bother you with that?

- You write software that other people/services use.
- Most likely, your software will be integrated into EAS.
- If you do software development after academia, it will be EAS.

I will use C# for all the following examples...

Why another programming language?

Everyone: I am happy with Python, why should I learn C#?

Why another programming language?

Everyone: I am happy with Python, why should I learn C#?

Me: Because C# is just awesome, trust me!

Why another programming language?

Everyone: I am happy with Python, why should I learn C#?

Me: Because C# is just awesome, trust me!

Everyone: But Python is awesome too!

Why another programming language?

Everyone: I am happy with Python, why should I learn C#?

Me: Because C# is just awesome, trust me!

Everyone: But Python is awesome too!

Me: Yes, but you don't have FeatureX!

Why another programming language?

Everyone: I am happy with Python, why should I learn C#?

Me: Because C# is just awesome, trust me!

Everyone: But Python is awesome too!

Me: Yes, but you don't have FeatureX!

Everyone: But I don't need FeatureX!

Why another programming language?

Everyone: I am happy with Python, why should I learn C#?

Me: Because C# is just awesome, trust me!

Everyone: But Python is awesome too!

Me: Yes, but you don't have FeatureX!

Everyone: But I don't need FeatureX!

Me: Yes, but ... !!!

Why another programming language?

My philosophy

Use the right tool for the job.

Why another programming language?

My philosophy

Use the right tool for the job.

- There are situations where Python is the perfect choice.

Why another programming language?

My philosophy

Use the right tool for the job.

- There are situations where Python is the perfect choice.
- There are situations where C++ is the perfect choice.

Why another programming language?

My philosophy

Use the right tool for the job.

- There are situations where Python is the perfect choice.
- There are situations where C++ is the perfect choice.
- There are situations where C# is the perfect choice.
  - Especially if you do EAS.

What I am NOT going to tell you

- C# is *better* than C++, Python,...
- C++, Python, ... are *bad*.
- You should *always* use C#.
- C# is *easy*.

What I am going to tell you

- Knowing C# is useful for your software stack.
- Industry experience is incorporated into the language.
- It has built-in design patterns and best practices.
- The language is not commonly seen in academia
  $\rightarrow$ Broaden your horizon

What we will do

- Discuss special aspects of the language.
- See patterns and best practices for EAS.
- Talk about paradigms and commonly used techniques.

What we will do

- Discuss special aspects of the language.
- See patterns and best practices for EAS.
- Talk about paradigms and commonly used techniques.
- Today: Introduction to C#

C# is...

- from the year 2000
- fully open source
- fully cross platform
- based on the .NET SDK
- multi-paradigm (object-oriented, functional, ...)
- strong statically typed
- popular (Tiobe 5th place, SO 4th place)
- still growing
- designed for readability
- influenced by: Java, C++, Haskell, ...

C# has...

- a large community
- a good documentation
- many packages (NuGet $>$ 280K, PyPi $>$ 350K)
- a large ecosystem
- just-in-time compilation
- garbage collection
- good IDEs
- influenced: D, Dart, Java, Kotlin, Swift, Rust, Go, ...

Hello World I

```
using System;

namespace HelloWorld
{
    public static class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
        }
    }
}
```

Hello World I

```
using System;

namespace HelloWorld
{
    public static class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
        }
    }
}
```

Looks like C++, right?

Hello World II

```
namespace HelloWorld;

class Program
{
    static void Main()
    {
        System.Console.WriteLine("Hello World");
    }
}
```

Hello World III

```
Console.WriteLine("Hello World");
```

Hello World III

```
Console.WriteLine("Hello World");
```

Looks like Python, right?

Conclusion

- C# is a mix of Python and C++.
- C# fits well into EAS.
- EAS play an important role in modern industry.
- It is good to have some knowledge about EAS when working in the industry.

Conclusion

- C# is a mix of Python and C++.
- C# fits well into EAS.
- EAS play an important role in modern industry.
- It is good to have some knowledge about EAS when working in the industry.
- If you are interested in EAS, I can make a series about, it's up to you :)

Possible Sub-Topics

Enterprise Application Systems (EAS) / csharp

- *Introduction*: Why is EAS/csharp relevant? Quick overview with motivation + elementary examples
- *Language I*: extension methods, fluent syntax, generics (vs. cpp templates), auto implemented properties, readonly expressions
- *Language II*: linq, pattern matching, special operators (?:, ??, ??=, ?. and ?[])
- *Language III*: generic variances, classes/structs/records(structs)/interfaces, destructors
- *Language IV*: special keywords (ref out in arguments, ref returns), async-await, asynchronous streams, default interface methods
- *Patterns*: dependency injection, mediator, repository, builder, abstract factory, adapter, singleton, proxy, observer, memento, publish/subscribe
- *Paradigms*: APIs, middlewares, Swagger/OpenAPI, gRPC, asynchronous messaging, microservices, SAGAs, CQRS/Event Sourcing, ef core
- *Misc*: reflection, clean code project, smart enums, RabbitMQ, Redis, kubernetes, GraphQL, Circuit Braker (Polly), mapper,
- A good overview of the history of language features can be found here.
- A good overview of patterns dan be found here.