Less known facts about records

Fact #1

Records have been added as a C# 9 and .NET 5 feature. To use them, a project needs to target .NET 5 or later. However, you can use most C# 9 features that don’t need the new runtime changes in projects targeting earlier frameworks.

So let's try to use C# 9 Records in .NET Core 3.1.

Fact #1

The compiler does not recognize the type keyword and argues that body needs to be declared.

Because of that, the instance of the class cannot be created:

Fact #1

To do fix it, we need to manually enable C# 9 on the project level. We can do it by setting property in the csproj to the or .

It will still not compile, because of the lack reference to the unknown type

Fact #1

The easiest way to overcome this issue (except migrating to .NET 5) is to copy and paste this class definition from the .NET 5 source to our project :)

https://github.com/dotnet/runtime/blob/6072e4d3a7a2a1493f514cdf4be75a3d56580e84/src/libraries/System.Privat

Fact #1

... and run the project once again. This time compiler will be able to compile your code :)

If you are curious how does it work, the answer is quite simple. is a new class used by the , not by the record type itself. After adding it to our source code, the compiler has everything to compile record type.

Fact #2

You can use more C# 9 features without .NET 5. According to the discussion in the Roslyn repository, you can use more C# 9 features in the previous framework, for example:

- Top-level statements (module 3)
- Pattern matching improvements (module 4)
- target-typed new (module 5)
- Lambda discard parameters (module 6)
- Attributes on local functions (module 6)
- Skip locals init (module 8)

... and more. The full list of features and tricks is available at https://github.com/dotnet/roslyn/discussions/47701

Please remember that it is a community discovery and analysis, not an official Microsoft statement about backward compatibility. Some features don't require any hacks and tricks and it's safe to use them, some of them need extra code (like records).

Fact #3

Probably nobody will ever tell you this, but records can be generic! It's is not a common use case, but you can actually use create a generic record type:

you can also add constraints to generics

or combine positional and non-positional properties

Fact #4

A record type is compiled as a class, similar to the class it can implement interfaces

or with positional syntax

Fact #5

Since we know that a record is just a class underneath, it has more class-specific features, for example, it can be .

There is only one rule to remember - only one partial type definition can use positional syntax for defining the properties.

Fact #6

We mentioned generics, constraints, interfaces, and partial definitions. To finish with style, let's describe one more funny fact - attributes.

We can enrich record properties or fields with attributes, for example, to manipulate JSON serialization:

Declaring an attribute on a positional record is a little bit more tricky

Fact #6

With the same trick we can also set attributes on the backing fields, which creates very tricky syntax:

Summary

Records are just classes underneath and they inherit most of some behaviors and features.

Positional syntax sometimes brings more complexity, but it provides built-in construction and deconstruction methods.

It previously mentioned facts are just okay for you, and you need for something spicier, try to use features together 🌶️

Materials

- https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/record
- https://tooslowexception.com/6-less-popular-facts-about-c-9-records/ by Konrad Kokosa
- https://github.com/dotnet/roslyn/discussions/47701