Records

Agenda

Introduction to records

Records and equality

Records and immutability

Destructuring

More about records

Record inheritance

Introduction to records

Records

It’s not really a new C#9 type

It’s a reference type

Generates a class

Provides immutability, value type equality and destructuring

Declaration

Declared with the `record` keyword

Declaration

```
public record Movie(int Id, string Title, string Description);
```

Generated members

The compiler generates many members behind the scenes

Formatting

The `ToString()` method returns the names and values of public properties and fields

Demo

Create a record

Decompile the assembly to see generated members

Demo

Records and equality

Records immutability and with expressions

Positional parameters

Positional parameters allow to declare init-only properties in a concise way

It can be customized by defining a property with the same name

Records immutability

Init-only properties have shallow immutability

Records can be mutable, but it’s not intended to

with expressions

Creates a copy of a record instance with modified members

These members must have an init or set accessor

with expressions

Calls the generated Clone method

with expression

```
var newMovie = movie with
{
    Description = "Another description"
};
```

Demo

Copy immutable objects using with expressions

Deconstructing

Deconstructing

Provides a way to retrieve multiple values from an object

For non record types

Classes, structures and interfaces, can be deconstructed by implementing one or more Deconstruct methods

For records

Records offer built-in support for deconstructing

Demo

Deconstructing record and class movie instances

Demo

More on records

Records inheritance

Inheritance support

A record can inherit from records only

Equality

To be equal, two records must have the same runtime type and all the properties must be equal

Demo

Inherit from a record

Summary

With records you can create immutable objects

Records provide value type equality

The `ToString` method of a record instance returns the names and values of public properties and fields

You can define properties with positional parameters or by adding “regular” properties

With expressions allow to create a copy of a record or a class

Destructuring provides a way to retrieve multiple values from an object

A record can inherit from a record only