Using Additional Attributes to Describe Nullability

Jason Roberts
.NET DEVELOPER

@robertsjason dontcodetired.com

Overview

Overview of null-related attributes

Express nullability intent before an action

Express nullability intent after an action

Express nullability intent after an action takes place based on some condition

Express nullability intent if constructor helper methods are used

Help compiler understand that a method never returns

Considerations

Nullable-describing Attributes

Attributes that you can apply to your code when using non nullable reference types to further express your design intent. These attributes are understood by the compiler and enable improved compile-time checking of null-related intent violations.

Using these attributes only makes sense when you have the new nullable features enabled

Attribute Categories

Precondition

Postcondition

Conditional postcondition

Constructor helper methods

Unreachable code

Attribute Categories

AllowNull
DisallowNull

MaybeNull
NotNull

MaybeNullWhen
NotNullWhen
NotNullIfNotNull

MemberNotNull
MemberNotNullWhen

DoesNotReturn
DoesNotReturnIf

Attribute Categories

AllowNull
DisallowNull

MaybeNull
NotNull

MaybeNullWhen
NotNullWhen
NotNullIfNotNull

MemberNotNull
MemberNotNullWhen

DoesNotReturn
DoesNotReturnIf

Precondition

Express nullability intent before an action takes place (e.g. value of method parameter, property/field setting value)

[AllowNull]

- Field, parameter, property
- Only applies to property setter value
- Non-nullable type is normally not null, but may occasionally be null

[DisallowNull]

- Field, parameter, property
- Only applies to property setter value
- Nullable type may sometimes be null (e.g. on creation), but cannot explicitly be set to null

Postcondition

Express nullability intent after an action takes place (e.g. method parameter value was not null after call)

[NotNull]

- Field, parameter, property, return value
- Only applies to property get value
- Nullable type whose value will never actually be null

[MaybeNull]

- Field, parameter, property, return value
- Only checks getting property value
- Non-nullable type whose value may actually be set to null in some cases

Conditional Postcondition

Express nullability intent after an action takes place but based on some condition

[NotNullWhen]

- Parameter
- Helps compiler understand when a nullable parameter won't be null
- Conditional on the method's Boolean return value

[MaybeNullWhen]

- Parameter
- Helps compiler understand that a non-nullable type might be null
- Conditional on the method's Boolean return value

Conditional Postcondition

[NotNullIfNotNull]

- Parameter, property, return value
- Helps compiler understand when a nullable type will actually have a value
- Conditional on a method's argument not being null

Constructor Helper Method

Express nullability intent if constructor helper methods are used

[MemberNotNull]

- Method, property
- Helps compiler understand when non-nullable fields and properties have been initialized outside the constructor
- Constructor “helper methods”

[MaybeNullWhen]

- Method, property
- Conditional on the constructor helper method’s Boolean return value

```
class PlayerCharacterCtorExample
{
    private string _name;
    public string Bio { get; set; }
    public PlayerCharacterCtorExample() => Init();


    private void Init()
    {
        _name = "No name";
        Bio = "No Bio";
    }
}
```

WARNING: Non-nullable field '_name' must contain a non-null value when exiting constructor. Non-nullable property 'Bio' must contain a non-null value when exiting constructor.

```
class PlayerCharacterCtorExample
{
    private string _name;
    public string Bio { get; set; }
    public PlayerCharacterCtorExample() => Init();

    [MemberNotNull("_name", "Bio")]
    private void Init()
    {
        _name = "No name";
        Bio = "No Bio";
    }
}
```

```
class PlayerCharacterCtorExample
{
    private string _name;
    public string Bio { get; set; }
    public PlayerCharacterCtorExample() => Init();

    [MemberNotNull(nameof(_name), nameof(Bio))]
    private void Init()
    {
        _name = "No name";
        Bio = "No Bio";
    }
}
```

Unreachable Code

Help compiler understand that a method never returns

Method always throws an exception or exits the program

[DoesNotReturn]

- Method
- Not conditional on parameter values

[DoesNotReturnIf]

- Parameter
- Conditional on a passed-in Boolean parameter value
- [DoesNotReturnIf(false)] bool isValid

Considerations

Extra work/effort to add these attributes

Need to discuss risk/reward/benefits

- All developers in team will need to be trained on the use of these attributes
- All developers in team will need to commit to adding attributes

#nullable enable/disable

Remember some scenarios like reflection are not understood by compiler

Calling code may not have feature enabled

- No nullable related warnings

Adding Null Checking Code to Public API

E.g. Parameters of public methods, constructor parameters, property setters

ArgumentNullException

Consider adding if:

- API is going to be used by other developers in your org that may not have null features enabled in their projects
- You are creating a library for public consumption (e.g. NuGet)
- Your solution contains a mix of projects with nullable features enabled/disabled

Consider not adding if:

- Solution/application is self-contained and all projects have feature enabled

Summary

Overview of null-related attributes

Precondition: [AllowNull]

Postcondition: [NotNull]

Conditional postcondition: [NotNullWhen]

Constructor helper: [MemberNotNull]

Unreachable code: [DoesNotReturn]

Considerations

Course Summary

Reference types, value types, Nullable<T>

Null-coalescing, null-conditional operator

Null Object pattern

Non nullable reference types (C# 8)

- <Nullable>enable</Nullable>
- string?
- Additional null-related attributes

Allow nulls where no-value
is an appropriate state

Start to use non nullable reference types if available (min C# 8)

Consider using the Null Object pattern if it will add value and you cannot upgrade to C# 8+

Continue to add null argument checks where they make sense

Resources and Further Learning

Nullable reference types [MS]

- https://bit.ly/PSNullRefTypes
- #nullable restore, #nullable enable annotations, etc.

Migration [MS]

- https://bit.ly/PSNullMigration

Null attributes [MS]

- https://bit.ly/PSNullAttributes

EF Core [MS]

- https://bit.ly/PSNullEFCore

Resources and Further Learning

Nullable reference types [MS]

- https://bit.ly/PSNullRefTypes

Migration [MS]

- https://bit.ly/PSNullMigration

Null attributes [MS]

- https://bit.ly/PSNullAttributes

EF Core [MS]

- https://bit.ly/PSNullEFCore