
Обобщенные атрибуты

Поехали!

Возможно [!/?]

```
1 reference
public class AwesomeAttribute : Attribute
{
    private readonly IAdorable adorable;

    0 references
    public AwesomeAttribute(IAdorable adorable) =>
        this.adorable = adorable;
}
```

```
2 references
public interface IAdorable
{
    0 references
    void ActCute();
}
```

Боль

```
[Awesome(new Adorable())]
0 references
public class Test { }
```

```
1 reference
public class Adorable : IAdorable
{
    1 reference
    public void ActCute() =>
        throw new NotImplementedException();
}
```

Боль

Error List

Entire Solution | 1 Error | 0 Warnings | 0 of 2 Messages | Build + IntelliSense

Search Error List

	Code	Description
⊗	CS0181	Attribute constructor parameter 'adorable' has type 'IAdorable', which is not a valid attribute parameter type

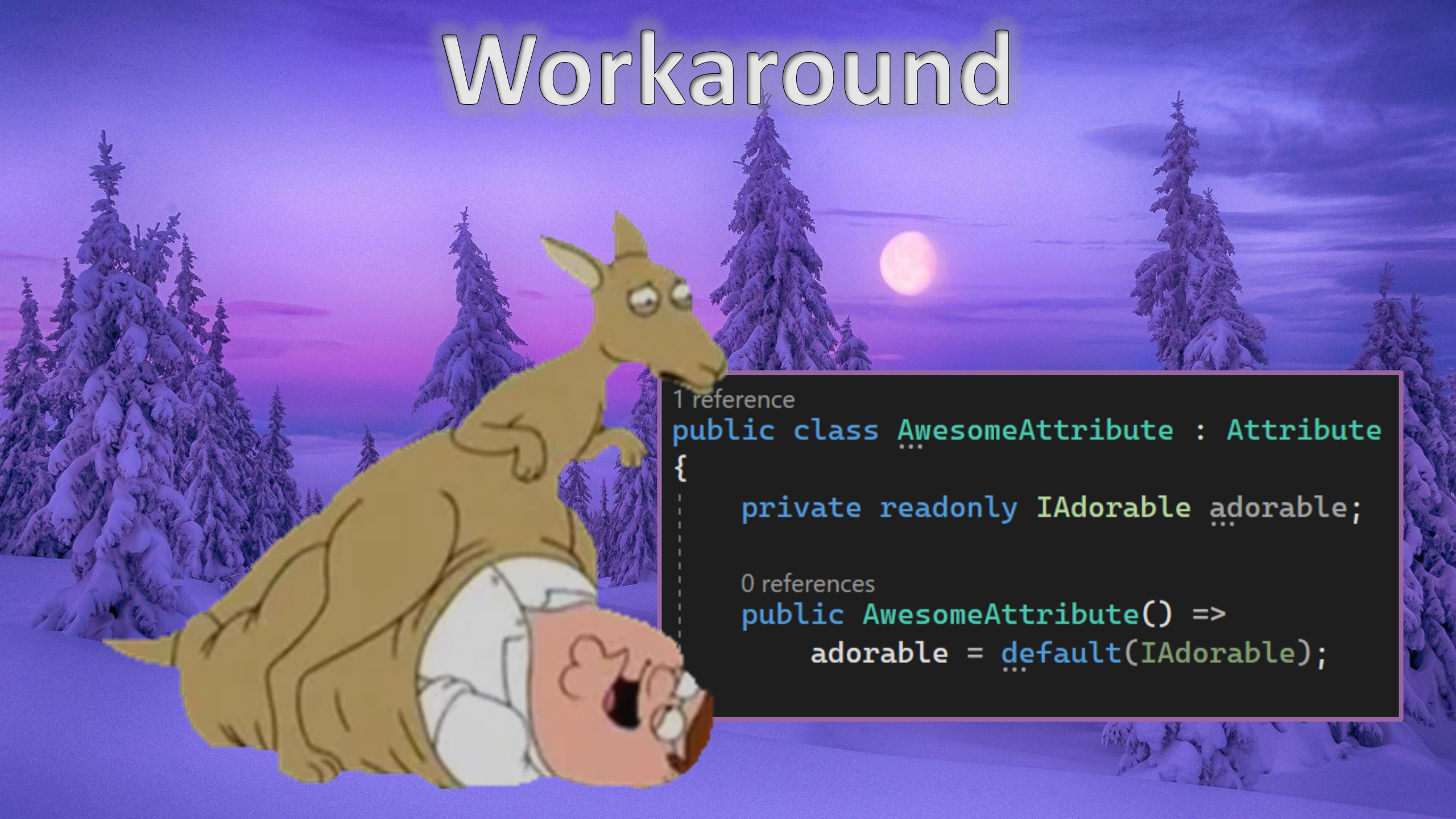
Матчасть

Конструктор атрибута может принимать:

1. **bool, byte, char, double, float, int, long, short, string и далее по примитивным, кроме decimal**
2. **object***
3. **System.Type**
4. **enum**
5. **одномерный массив любого из вышеперечисленных типов**

Атрибуты не могут обобщенными*

Workaround



```
1 reference
public class AwesomeAttribute : Attribute
{
    private readonly IAdorable adorable;

    0 references
    public AwesomeAttribute() =>
        adorable = default(IAdorable);
```

Немного рефлексии

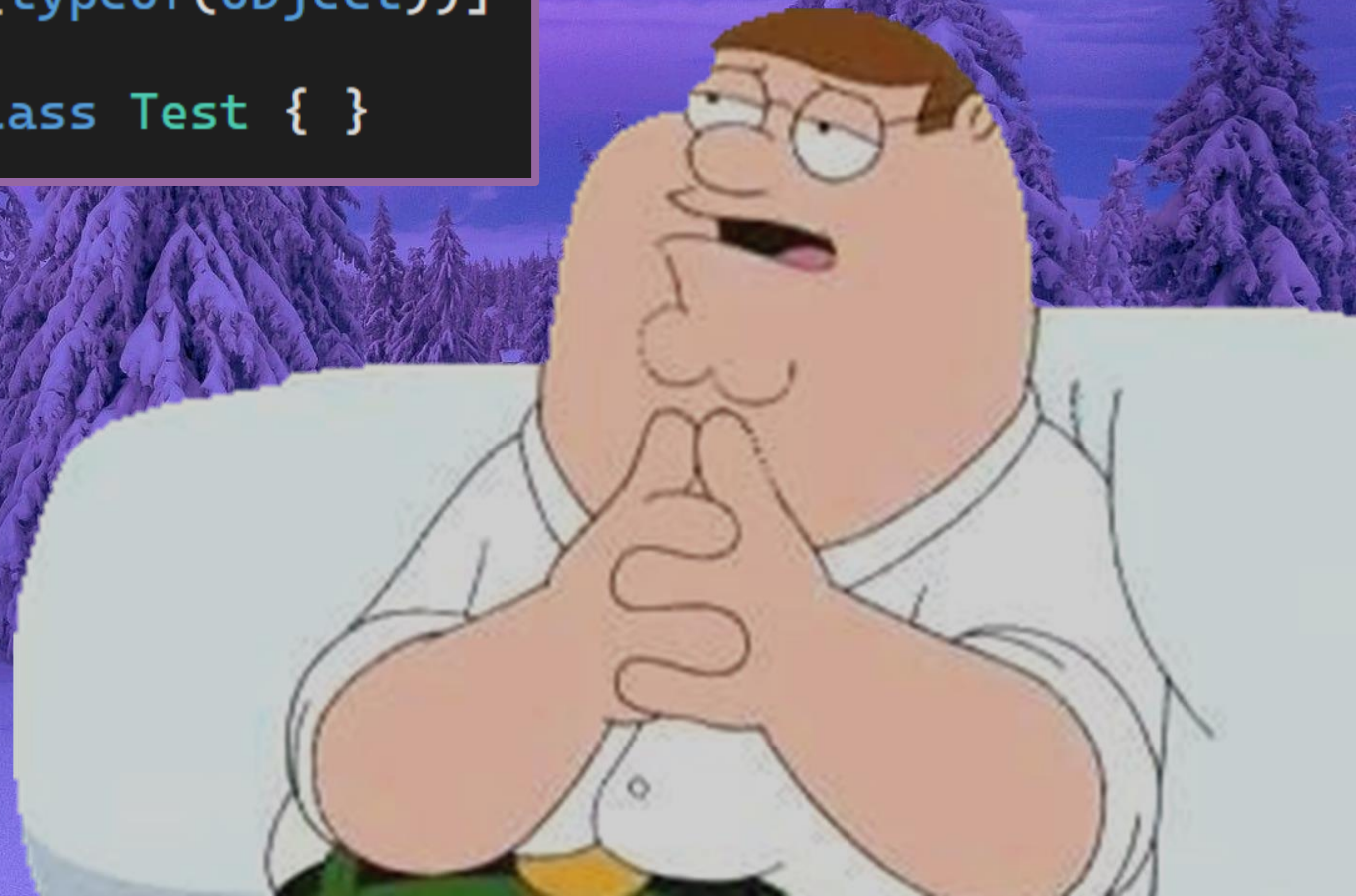
```
1 reference
public class AwesomeAttribute : Attribute
{
    private readonly IAdorable adorable;

    0 references
    public AwesomeAttribute(Type impl) =>
        adorable = (IAdorable)Activator.CreateInstance(impl);
```

П-п-приведение!!!

```
1 refere
pu  wesomeAttribute : Attribute

    donly IAdorable adorable;


    Attribute(Type impl) =>
     (IAdorable)Activator.CreateInstance(impl);
```

Компилятор

```
[Awesome(typeof(object))]
0 references
public class Test { }
```

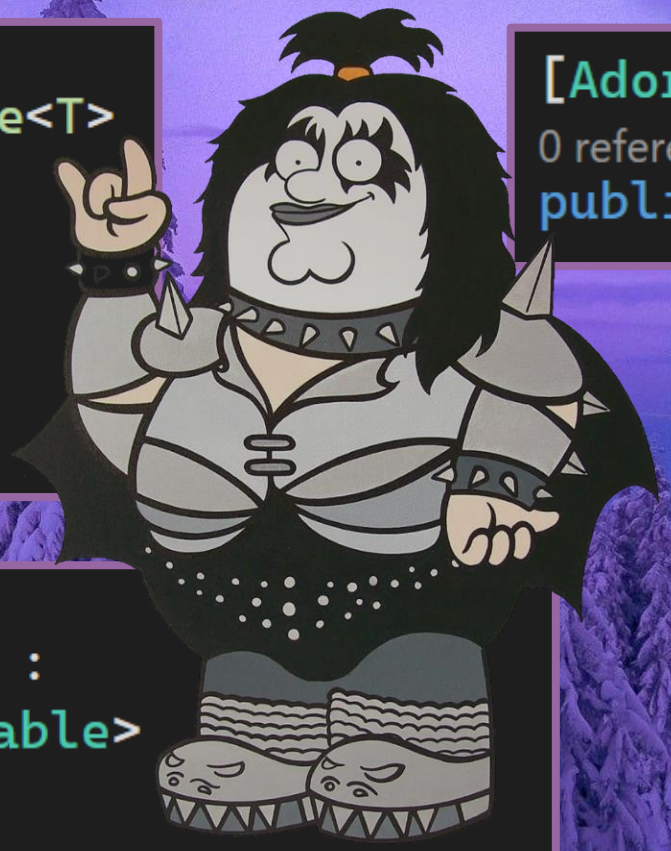


Runtime


Exception Unhandled
System.InvalidCastException: 'Unable to cast object of type 'System.Object' to type 'SpbDotNet.IAdorable'.'
Show Call Stack | View Details | Copy Details | Start Live Share session
Exception Settings

Хардкор!!!

```
1 reference
public interface IAwesomeAttribute<T>
    where T : IAdorable
{
    1 reference
    T Adorable { get; }
}
```



```
[AdorableAwesome]
0 references
public class Test { }
```

```
1 reference
public class AdorableAwesomeAttribute :
    Attribute, IAwesomeAttribute<Adorable>
{
    1 reference
    public Adorable Adorable { get; } = new Adorable();
}
```

Поддержка...

```
3 references
public class Admirable : IAdorable
{
    1 reference
    public void ActCute() =>
        throw new NotImplementedException();
}
```

```
0 references
public class AdmirableAwesomeAttribute :
    Attribute, IAwesomeAttribute<Admirable>
{
    1 reference
    public Admirable Adorable { get; } = new Admirable();
}
```

C#11: Generic attributes

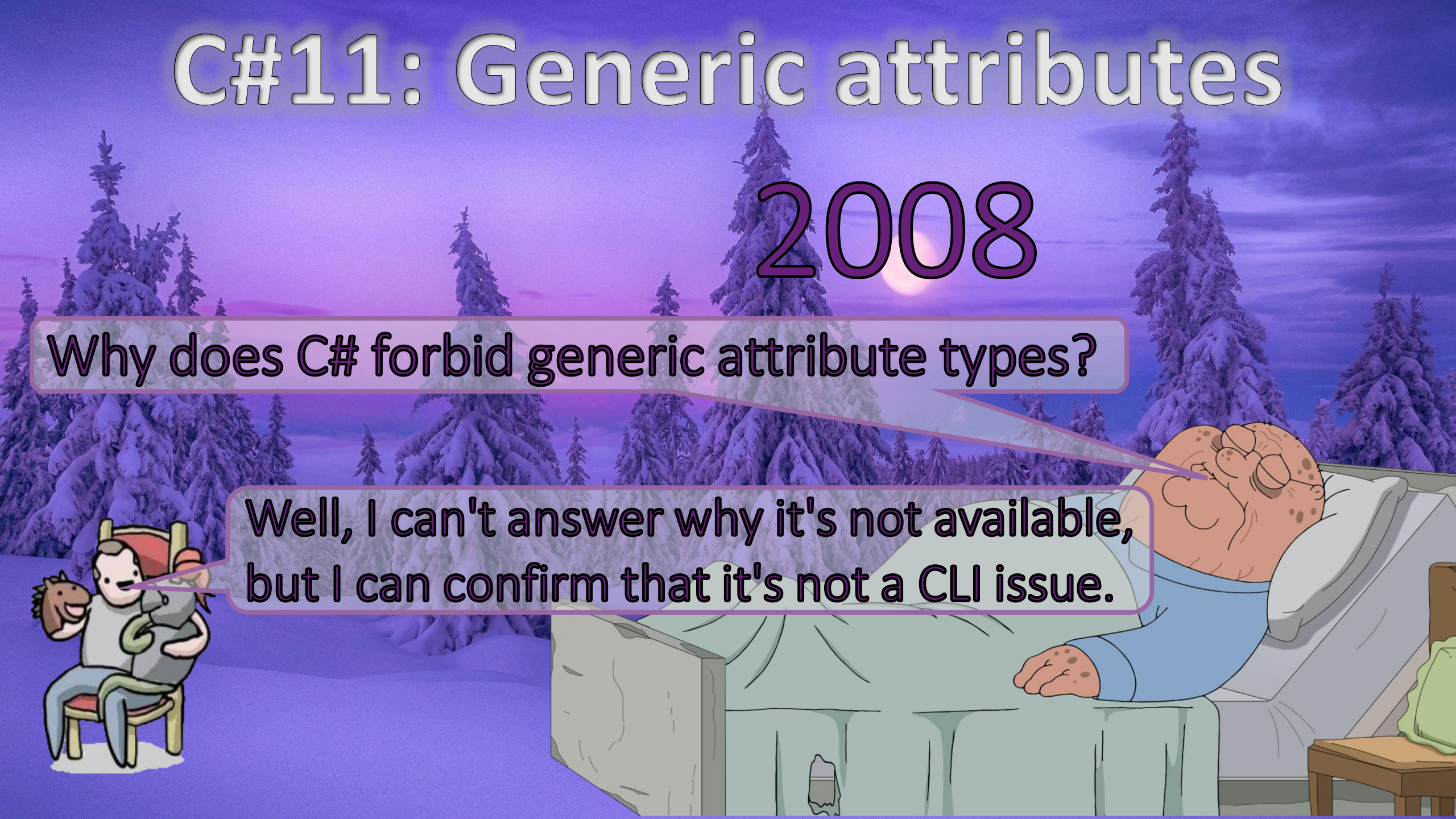
C#11: Generic attributes

1. История языка
2. Легаси

C#11: Generic attributes

2008

Why does C# forbid generic attribute types?



C#11: Generic attributes

2017

Champion "Allow Generic Attributes"

for C# 8

C#11: Generic attributes

```
2 references
public class AwesomeAttribute<TAdorable> : Attribute
    where TAdorable : IAdorable
{
    private readonly TAdorable adorable;

    1 reference
    public AwesomeAttribute() =>
        adorable = (TAdorable)Activator
        .CreateInstance(typeof(TAdorable));
}
```

C#11: Generic attributes

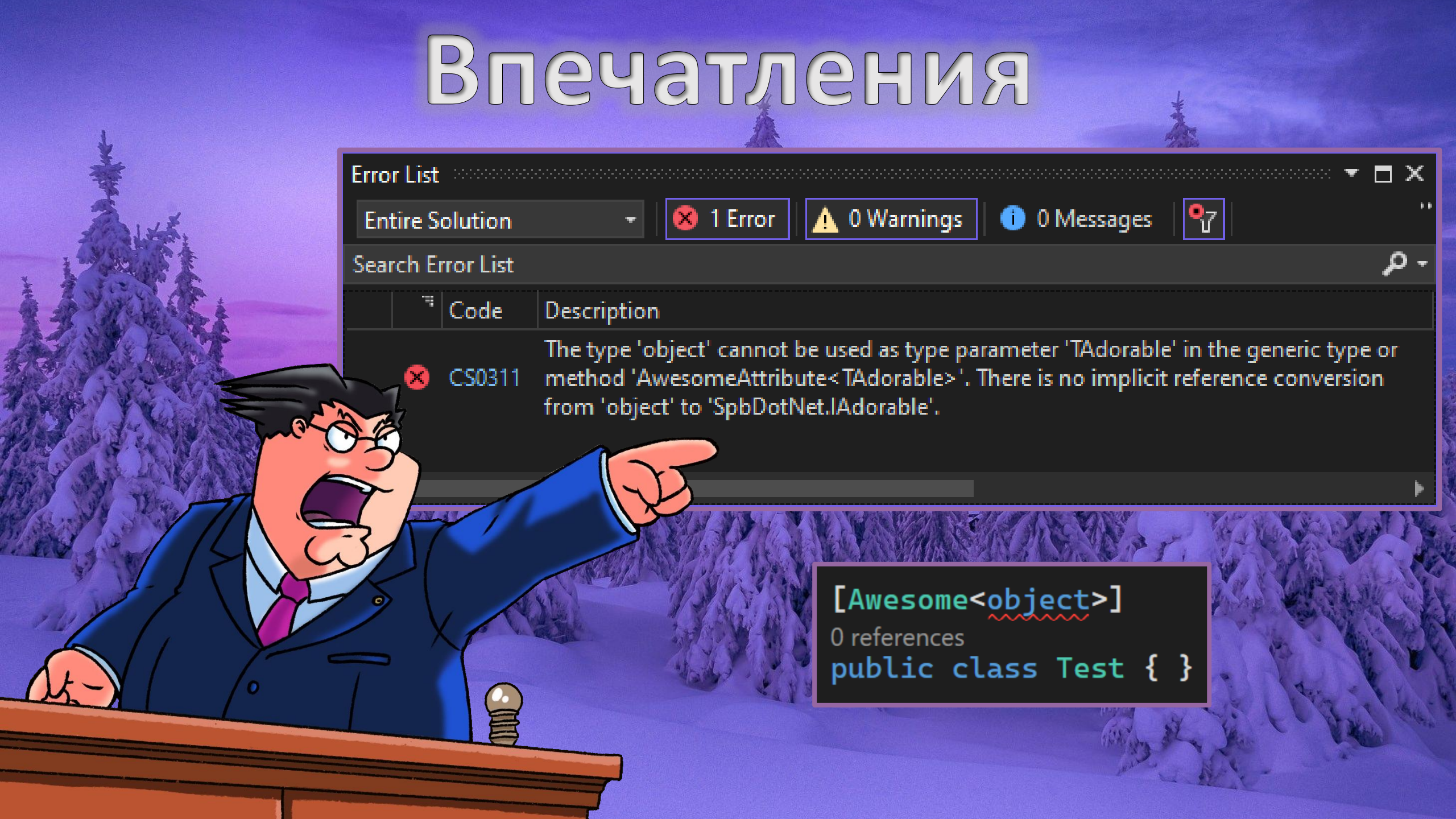
```
2 references
public class AwesomeAttribute<TAdorable> : Attribute
    where TAdorable : IAdorable, new()
{
    private readonly TAdorable adorable;

    1 reference
    public AwesomeAttribute() =>
        adorable = new TAdorable();
}
```

Паттерн ‘Команда’

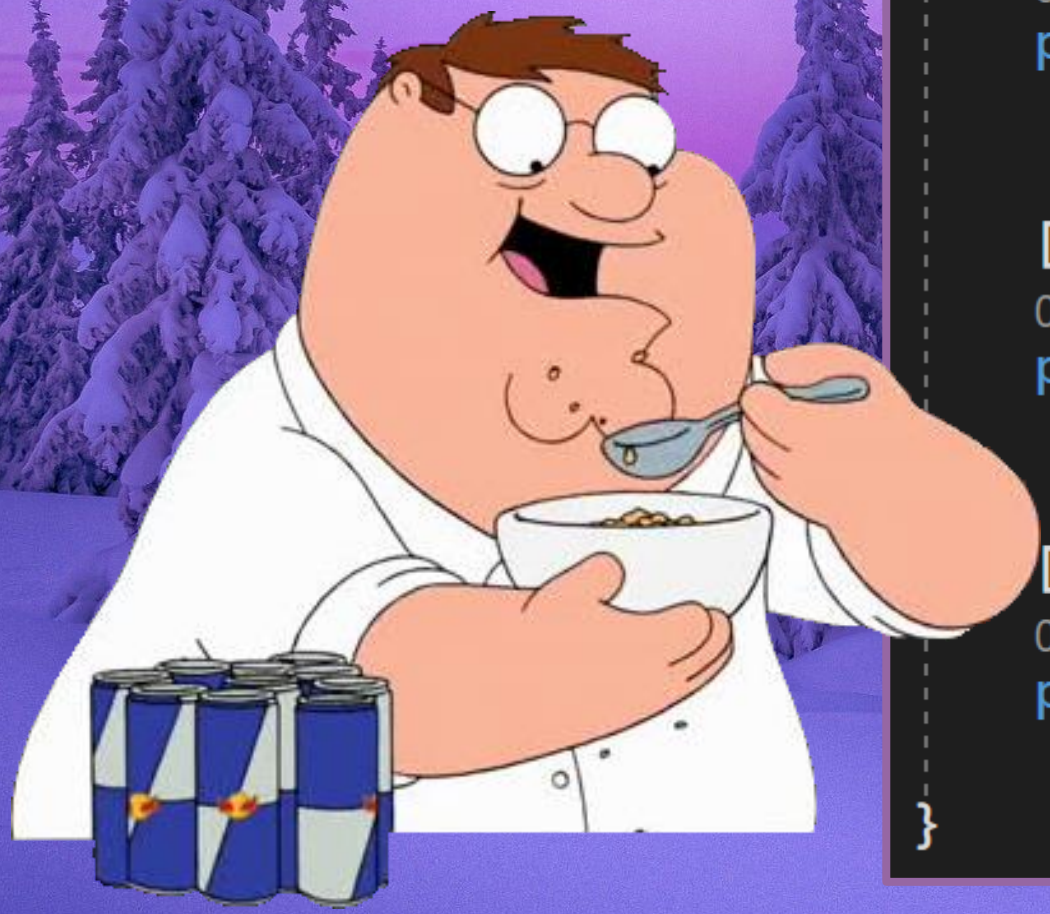
```
2 references
public interface IAdorable
{
    0 references
    void ActCute();
```

```
0 references
public interface ICommand
{
    0 references
    void Execute();
}
```

Впечатления



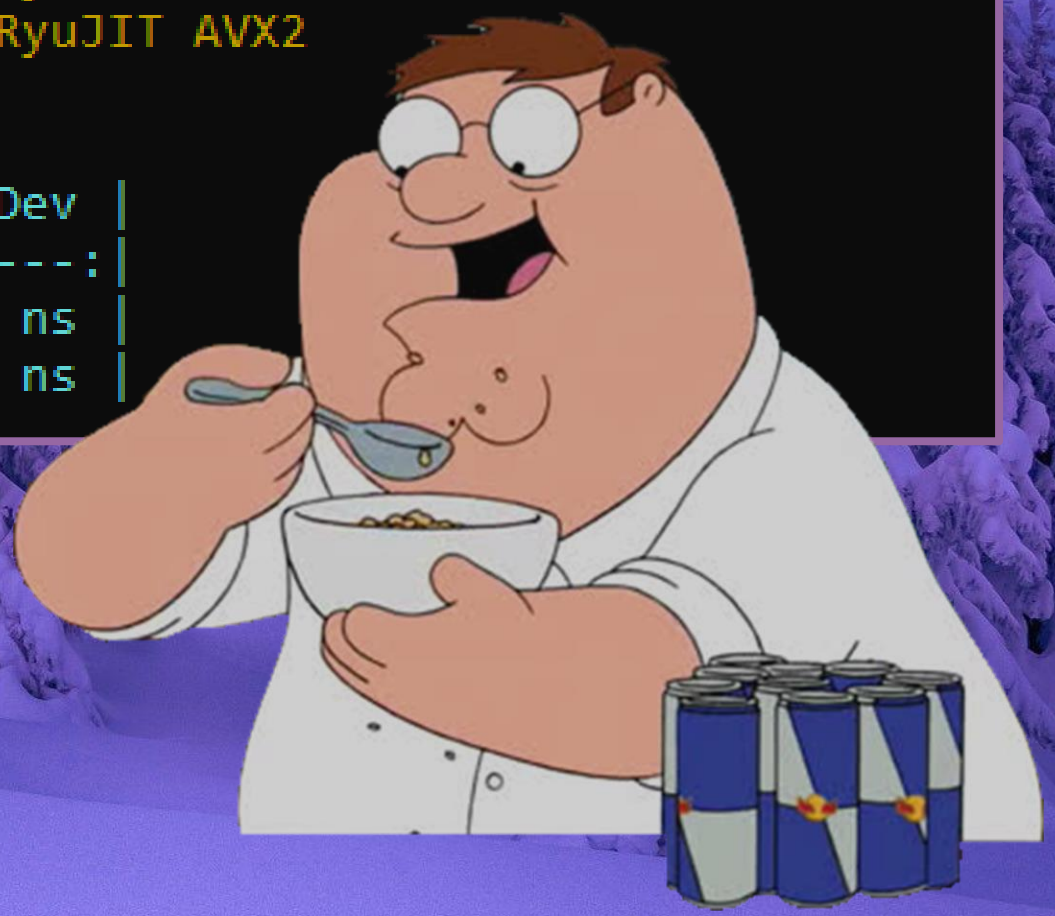
Впечатления



```
1 reference
public class Program
{
    0 references
    public static void Main() =>
        BenchmarkRunner.Run<Program>();

    [Benchmark]
    0 references
    public void Activator() =>
        new AwesomeAttribute(typeof(Adorable));

    [Benchmark]
    0 references
    public void Constructor() =>
        new AwesomeAttribute<Adorable>();
}
```

Впечатления

```
BenchmarkDotNet=v0.13.4, OS=Windows 10 (10.0.19041.1415/2004/May2020Update/20H1)
Intel Core i5-4570S CPU 2.90GHz (Haswell), 1 CPU, 4 logical and 4 physical cores
.NET SDK=7.0.100
  [Host]     : .NET 7.0.0 (7.0.22.51805), X64 RyuJIT AVX2
  DefaultJob : .NET 7.0.0 (7.0.22.51805), X64 RyuJIT AVX2


|      Method |       Mean |     Error |    StdDev |
|------------ |-----------:|----------:|----------:|
|   Activator | 16.987 ns | 0.1964 ns | 0.1741 ns |
| Constructor |  8.485 ns | 0.0863 ns | 0.0807 ns |
```



‘new()’ – не вариант?

```
9 references
public class Adorable : IAdorable
{
    2 references
    public Adorable(string name) { }

    1 reference
    public void ActCute() =>
        throw new NotImplementedException();
}
```

‘new()’ – не вариант?

```
2 references
public interface IAdorableFactory
{
    2 references
    IAdorable CreateAdorable(string str);
}
```

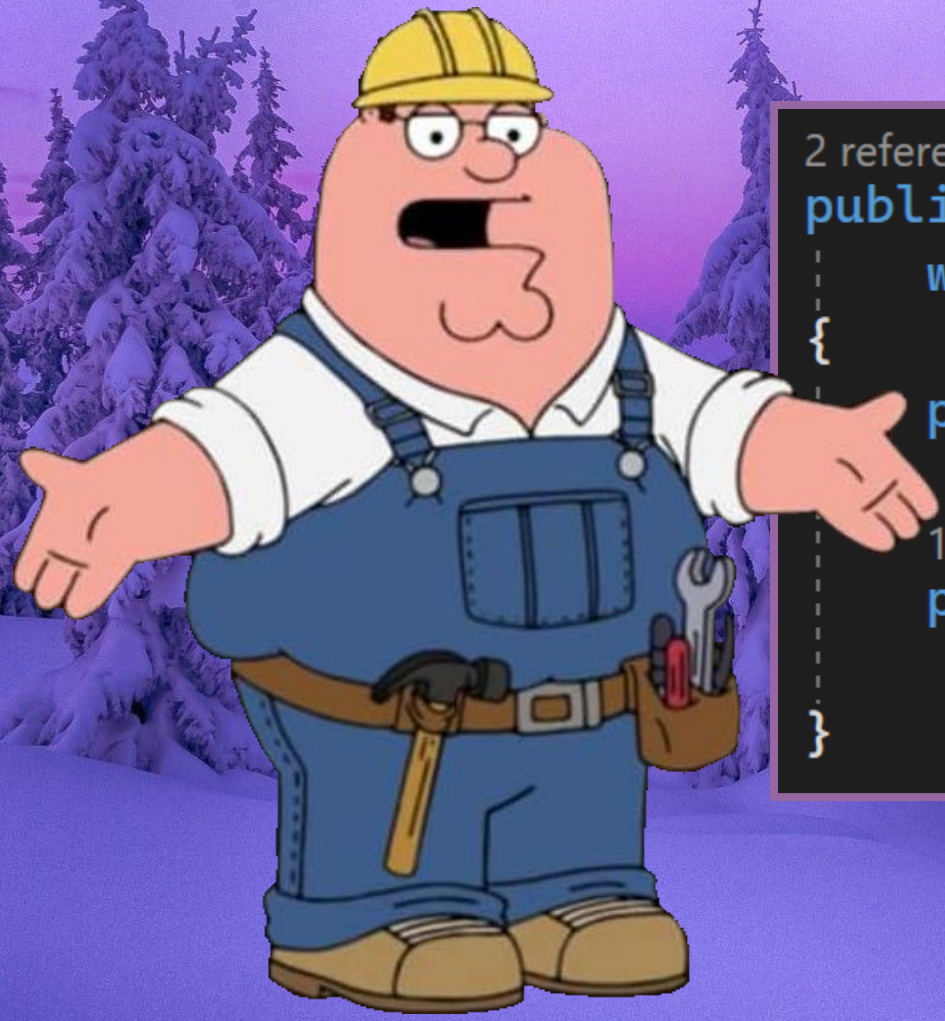
```
0 references
public class AdorableFactory : IAdorableFactory
{
    2 references
    public IAdorable CreateAdorable(string str) =>
        new Adorable(str);
}
```

‘new()’ – не вариант?

```
3 references
public class AwesomeAttribute<TAdorableFactory> :
    Attribute where TAdorableFactory :
    IAdorableFactory, new()
{
    private readonly IAdorable adorable;

    2 references
    public AwesomeAttribute(string str) =>
        adorable = new TAdorableFactory()
        .CreateAdorable(str);
}
```

```
[Awesome<AdorableFactory>("Hello world")]
0 references
public class Test { }
```

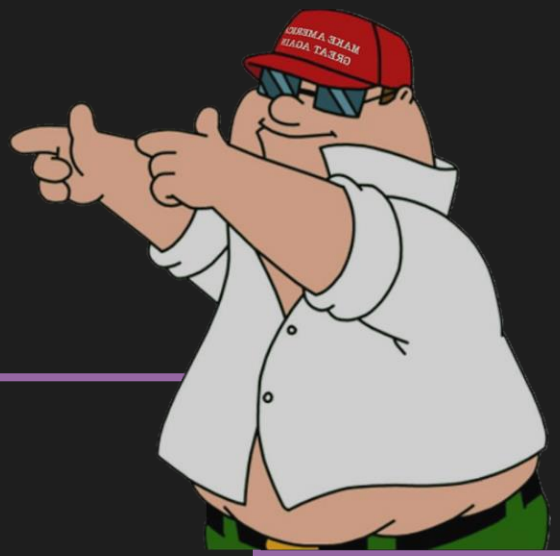
‘new()’ – не вариант?



```
2 references
public class AwesomeAttribute<TAdorable> : Attribute
    where TAdorable : IAdorable
{
    private readonly TAdorable adorable;

    1 reference
    public AwesomeAttribute() =>
        adorable = DI.Resolve<TAdorable>();
}
```

Применение

```
Assembly Microsoft.AspNetCore.Mvc.Core, Version=2.2.5.0
namespace Microsoft.AspNetCore.Mvc
{
    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = true, Inherited = true)]
    [DebuggerDisplay("ServiceFilter: Type={ServiceType} Order={Order}")]
    public class ServiceFilterAttribute : Attribute, IFilterFactory, IFilterMetadata, IOrderedFilter
    {
        public Type ServiceType { get; }

        public ServiceFilterAttribute(Type type)
        {
            if (type == null)
            {
                throw new ArgumentNullException("type");
            }

            ServiceType = type;
        }
    }
}
```



```
1 reference
public class ServiceFilterAttribute<TActionFilter> :
    ServiceFilterAttribute where TActionFilter :
    IActionFilter
{
    0 references
    public ServiceFilterAttribute() : base(typeof(TActionFilter))
    {
    }
}
```

Применение

Применение

Ограничения

```
1 reference
public class GenericAttribute<T> : Attribute { }

1 reference
public class GenericClass<T>
{
    // CS8968
    // 'T': an attribute type argument
    // cannot use type parameters
    [GenericAttribute<T>]
    0 references
    public GenericClass() { }
}
```

Ограничения

```
1 reference
public class GenericAttribute<T> : Attribute { }

1 reference
public class GenericClass<T>
{
    // CS8970
    // Type 'dynamic' cannot be used in this context
    // because it cannot be represented in metadata.
    [GenericAttribute<dynamic>]
    0 references
    public GenericClass() { }
}
```

Ограничения

The type arguments must satisfy the same restrictions as the typeof operator. Types that require metadata annotations aren't allowed. For example, the following types aren't allowed as the type parameter:

- dynamic
- string? (or any nullable reference type)
- (int X, int Y) (or any other tuple types using C# tuple syntax).

These types aren't directly represented in metadata. They include annotations that describe the type. In all cases, you can use the underlying type instead:

- object for dynamic.
- string instead of string?.
- ValueTuple<int, int> instead of (int X, int Y).

Источники

1. https://stackoverflow.com/questions/294216/why-does-c-sharp-forbid-generic-attribute-types
2. https://stackoverflow.com/questions/1852837/is-there-a-generic-constructor-with-parameter-constraint-in-c
3. https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/proposals/csharp-11.0/generic-attributes
4. https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-11#generic-attributes
5. https://github.com/dotnet/roslyn/issues/953
6. https://github.com/dotnet/csharplang/issues/124
7. https://habr.com/ru/post/468287/
8. https://habr.com/ru/post/681886/
9. https://www.infoq.com/news/2019/01/Generic-Attributes/
10. https://gsferreira.com/archive/2022/csharp-11-generic-attributes-more-than-syntax-sugar/
11. https://thecodeblogger.com/2022/09/18/c-11-introducing-support-for-generic-attributes/
12. https://prographers.com/blog/first_class_support_for_attributes_in_cs11

Обсуждение