Home
PUBLIC
Questions
Tags
Users
COLLECTIVES
Explore Collectives
FIND A JOB
Jobs
Companies
TEAMS
Create free Team

How IConfiguration.Bind() Works In .NET Core without 'out' or 'ref' in it's Arguments?

Ask Question

Asked yesterday Active today Viewed 36 times

0

In .NET Core we use `IConfiguration` to bind configuration to an instance.

```
var settings = new AppSettings();
Configuration.GetSection("AppSettings").Bind(settings);
```

2

To create a similar function like Bind() in my personal project (for a different requirement), I've created a method like this

```
public void BindOption(ref object instance)
{
  ...
}
```

And use it like this

```
var settings = new AppSettings();
MyCustomStaticClass.BindOption(ref settings);
```

Now my question is

How the `Bind()` function works without explicitly mentioning `out` or `ref` in it's arguments?

c# .net-core configuration appsettings

Share Edit Close Delete Flag

asked yesterday
Sangeeth Nandakumar
1,127 ●1 ●9 ●17

Ask yourself this: "Why do you have to do `var settings = new AppSettings();` before calling `Bind()`? The `new AppSettings()` part, in particular. – madreflection yesterday

My understanding is bind expects an instance of an object not an object itself. Means a pre-initialized object. Correct me if my understanding is wrong. – Sangeeth Nandakumar yesterday

The only way I could think other than using 'out' or 'ref' is relying on reflection to handle generically which is what I need to confirm here – Sangeeth Nandakumar yesterday

That's a distinction that doesn't exist. An instance *is* an "object itself". And yes, it uses reflection to populate the properties of the instance you give it. You would only need `out` if `Bind` were responsible for creating the instance and giving it back to you, but it's not; you created the instance so you don't need it to modify your variable. – madreflection yesterday

Add a comment

Question eligible for bounty in 21 hours

1 Answer

Active | Oldest | Votes

1

It sounds like you're misunderstanding `ref` / `out` - the only thing they do for you when you're working with a reference type (class) passed into a method, is allow the method to assign a new instance of an object to the argument variable, and the calling code sees the change. They are absolutely not needed to change some data content of the passed variable.

I personally think a lot of the confusion surrounding this comes from the phrase "passed by reference" having a word in it that starts with `ref`, coincidentally being the 3 letters in `ref`

When you make a reference type instance:

```
Person john = new Person(){ Name = "John", Age = 27 };
```

You can pass it to a method:

```
void ChangeName(Person p){
  p.Name = "Fred";
}
```

You don't need to specify `ref` to make this name change and have it stick

The Overflow Blog
Stack Gives Back 2021
Safety in numbers: crowdsourcing data on nefarious IP addresses
Featured on Meta
We've made changes to our Terms of Service & Privacy Policy - January 2022
New post summary designs on greatest hits now, everywhere else eventually
2021: a year in moderation
Sunsetting Jobs & Developer Story
Hot Meta Posts
28 Where are new users supposed to learn that they should show code and error...
7 In "Low Quality Answers", should we cross-check package references to...

Related

2056 How do I create an Excel (.XLS and .XLSX) file in C# without installing Microsoft Office?
963 What's the difference between the 'ref' and 'out' keywords?
2304 How do I get a consistent byte representation of strings in C# without manually specifying an encoding?
377 How to determine if .NET Core is installed
954 What is the difference between .NET Core and .NET Standard Class Library project types?
459 Build .NET Core console application to output an EXE
88 Get ConnectionString from appsettings.json instead of being hardcoded in .NET Core 2.0 App
274 Getting value from appsettings.json in .net core
2 .Net Core Worker Service appsettings.json not being read

Hot Network Questions

Declined PhD offer, but now I regret it
If one Starship can transport 100 people to Mars, how many could it safely land near Mercury's north pole after one Hohmann-like transfer?
A Python-program that makes successive API-calls
Why are there no volcanoes where continents collide with each other?
How to control color bar min and max in StreamPlot
How can we explain the "bad reputation" of higher-order polynomials?
Wrong Graphics coordinate system after Image to Graphics conversion
Is at-rest encryption worth it if the key has to be kept accessible for decryption?
What is this lego part number? Technic, right angle with 4 pins sticking out
What would happen to a star if most of its energy were reflected back at it?

```
ChangeName(john);
Console.Print(john.Name); //Fred
```

Internally in C#, imagine that you make `John` and C# allocates some memory, puts the name in it and gives you your variable reference:

```
john --> { Name: "John", Age: 27 }
```

You call the method and C# *temporarily* makes another reference called p (because the method takes a Person p)

```
john --> { Name: "John", Age: 27 } <-- p
```

New reference, same object. You change the name via p.Name, original object is affected

```
john --> { Name: "Fred", Age: 27 } <-- p
```

If we use `ref` C# doesn't make a new reference, imagine it just temporarily renames our existing one then names it back when the method is done

```
void ChangeName(ref Person p){
  p.Name = "Fred";
}


john --> { Name: "John", Age: 27 }      //start with
p --> { Name: "John", Age: 27 }          //when start calling the method
p --> { Name: "Fred", Age: 27 }          //method changes the person name
john --> { Name: "Fred", Age: 27 }      //method exits, name is changed
```

We never needed ref here

Different scenario. Person is immutable. The only way to change the name is to swap it for a whole new object

```
void ChangeName(Person p){
  p = new Person { Name = "Fred", Age: 0 };
}
```

Now, *this* needs ref. Look what happens when we don't have ref:

```
john --> { Name: "John", Age: 27 }
john --> { Name: "John", Age: 27 } <-- p
john --> { Name: "John", Age: 27 }     p --> { Name: "Fred", Age: 0 }
john --> { Name: "John", Age: 27 }
```

Calling the method makes a new reference to the old object, then saying `p = new...` reattaches that new reference to some new object, which is promptly destroyed at the end of the method leaving nothing changed

With ref:

```
john --> { Name: "John", Age: 27 }
p --> { Name: "John", Age: 27 }
p --> { Name: "Fred", Age 0 }
john --> { Name: "Fred", Age 0 }
```

Temporarily renaming that reference meant when we made a new object and assigned it, the original calling code experiences the `new` too

---

You need `ref` only if you will set a passed-in thing to a `new` and want the caller to see the new. Most of the time we don't need it

Footnote: `out` is like `ref` except the compiler will check that the method does certainly assign a value. You're required to use "out" or "ref" when passing the variable into the method, because it brings to your attention "this method might overwrite whatever you pass in" (for ref) or "this method will definitely overwrite whatever you pass in" (with out)

Modern .net also has the little-used `in` modifier, which makes the compiler check that you don't assign a `new` anything to a passed-in variable (running the risk of the non-ref scenario where Fred is lost at the end of the method)

edited 23 hours ago

answered 23 hours ago

Caius Jard
62.7k ● 5 ● 39 ● 69

C++ basic bank money class
Why is "ugly" in quotations?
Applications of complex exponential
Economics term for those who benefit even though they didn't contribute
Why to not create a warrior species if you can?
Prior to COVID-19, had any country fiscally targeted unvaccinated individuals?
Is it common and a good idea to give PhD candidates access to examiners' assessments prior to their defense?
Cultural context for "Abendland" and "Rettung" in this song lyric
How can ntfs.sys be loaded if the driver is located in an NTFS partition?
Creating a raster from scratch in QGIS
What is the highest possible number of pawn captures in a game?
Obfuscating HTTP Error Codes
Do you include coding assignments in an intro to complexity and computation course?
Does having no 'if' blocks in code mitigate side-channel attacks?
Question feed

Hey Caius, I got your point. But I'm confused on your answer's 3rd codeblock - "You don't need to specify ref to make this name change and have it stick". To my understanding this is correct for a global variable but not the other way. I was in an impression that pass by value will only create an manipulate a scoped copy within the function – Sangeeth Nandakumar 3 mins ago Edit

When we didn't specify ref explicitly it will accept the arg 'by value' and not 'by reference' correct? In that case if you change the var inside that fun scope 'ChangeName(Person p)', It will have no effect on original var outside unless it's a global var. This was my understanding. – Sangeeth Nandakumar 3 mins ago Edit

Reference: youtu.be/lYdcY5zuIXA?t=102 - Please correct me if I'm wrong – Sangeeth Nandakumar 1 min ago Edit

Add a comment

Answer Your Question