Is ApiController deprecated in .NET Core

37

4

Is it true, that " `ApiController` will get deprecated in .NET Core"? I really need professional insight on this. Since I'm planning to use it in new projects.

c# asp.net-mvc asp.net-web-api asp.net-core .net-core

edited Jan 23 '18 at 8:29
Henk Mollema
24.6k 9 64 84

asked Jul 29 '16 at 21:02
Viji
1,589 1 14 27

8 Short Answer: Yes. Not so short answer: Asp.Net MVC and Asp.Net Web API were merged into one code base in asp.net-core. So they all inherit from `Controller` and can all return implementations of `IActionResult` . Be it a `View` for MVC or `Json` for web api. You configure core based on what you need. docs.asp.net/en/latest/intro.html – Nkosi Jul 29 '16 at 21:14

1 `[ApiController]` was added back since 2.1. See below. – Riscie Jun 4 '18 at 8:26

4 Answers

49

✓

Update ASP.NET Core 2.1

Since ASP.NET Core 2.1 a new set of types is available to create Web API controllers. You can annotate your controllers with the `[ApiController]` attribute which enables a few new features such as automatic model state validation and binding source parameter inference. See the docs for more information: https://docs.microsoft.com/en-us/aspnet/core/web-api/index?view=aspnetcore-2.1#annotate-class-with-apicontrollerattribute.

There is indeed no particular `ApiController` class anymore since MVC and WebAPI have been merged in ASP.NET Core. However, the `Controller` class of MVC brings in a bunch of features you probably won't need when developing just a Web API, such as a views and model binding.

Or

Create your `ApiController` base class. The key here is to add the `[ActionContext]` attribute which injects the current `ActionContext` instance into the property:

```
[Controller]
public abstract class ApiController
{
    [ActionContext]
    public ActionContext ActionContext { get; set; }
}
```

Also, add the `[Controller]` attribute to the class to mark it as a controller for the MVC controller discovery.

See more details in my “Web API in MVC 6” blogpost.

edited Jun 14 '18 at 6:41

answered Jul 30 '16 at 10:06

Henk Mollema
24.6k 9 64 84

That's actually exactly what `ApiController` from the WebApi compatibility Shim Package does, it just binds more properties but they all base from ControlerContext that's injected via property github.com/aspnet/Mvc/blob/dev/src/… but comes with more compatibility stuff (that's completely optional), like route registration github.com/aspnet/Mvc/blob/dev/src/… or WebApi2 esque Attributes – Tseng Jul 31 '16 at 10:41

this comment helps with the choice of ControllerBase... // // Summary: // A base class for an MVC controller without view support. [Controller] public abstract class ControllerBase – granadaCoder Apr 11 '18 at 20:24

FYI, your answer is outdated for ASP.NET Core 2.1, see the answer by Riscie on the new `ApiControllerAttribute` . Would be great if you could update your answer, since it's the accepted one :) – Stijn Jun 13 '18 at 11:44

Thanks for the heads-up @Stijn. I've updated my answer. – Henk Mollema Jun 14 '18 at 6:41

1 @HenkMollema : I am still confused that (in ASP.Net core 2.1) why would we need to use either or both the *ApiController* and *ControllerBase* to denote a web API controller class. Why two things exist for the same task i.e. to make an API? – Koder101 Aug 28 '18 at 4:33

The `[ApiController]` attribute actually got added back in ASP.NET Core version 2.1.

Features coupled with the attribute are:

- No more need to define `[FromBody]` , `[FromRoute]` , ... attributes explicitly

Links to the docs:

- https://docs.microsoft.com/en-us/aspnet/core/aspnetcore-2.1?view=aspnetcore-2.1#apicontroller-actionresult
- https://docs.microsoft.com/en-us/aspnet/core/web-api/index?view=aspnetcore-2.1#annotate-class-with-apicontrollerattribute

Update

There is also the baseclass `ControllerBase` for controllers to inherit from which is suited for api-controllers because it ommits all view-related functionality.

- https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase?view=aspnetcore-2.1

edited Jun 4 '18 at 8:26 answered Jun 4 '18 at 6:31
Riscie
1,913 15 23

2 Is there any resource available which can show the comparative analysis between *ApiController* and the *ControllerBase*, i.e what one can do and the other cannot or when to use what? This is really confusing. – Koder101 Aug 28 '18 at 4:38

10

In ASP.NET core uses terms and concepts known from ASP.NET MVC and ASP.NET WepAPI. But basically it is a complete new framework. Therefore there are several concepts or base classes that we can simply forget.

ASP.NET MVC and ASP.NET WebApi are two coexisting but different frameworks and therefore a destinction has to be made to specify a controller as a WebApi Controller by using the `ApiController` as base class.

In ASP.NET Core this is simply not necessary anymore. The `Controller` base class can be used for actions that return HTML from Razor Views or JSON (with output formatters XML and other formats are possible as well). You don't even need the `Controller` base class. It is even possible to use a "Plain Old C# Object" as Controller without inheritence. This is an example of a Demo-Controller to outline, that even though the ApiController is not there, the structural approach to deliver data to the client is similar.

```
public class DemoController : Controller
{
     public async Task<IActionResult> Action()
```

```
        }
    }
```

edited Jul 29 '16 at 21:47 answered Jul 29 '16 at 21:39

Ralf Bönning
9,070 5 26 51

1 I chuckled when I read your Demo-Controller is a POC without Controller inheritance and then in your code you inherit from Controller - a typo I am sure :) – steve Feb 7 at 7:53

4

As others mentioned, ASP.NET Core is a complete new webstack that's not compatible with the old ASP.NET MVC webstack. This is explicitly reflected in it's name and versioning!

ASP.NET Core and ASP.NET Core MVC have the version 1.0.0 to make this incompatibility very clear.

ASP.NET Core merged the MVC and WebApi into one single Api just called.

And here's the thing you may have been looking for:

If you are migrating from a previous ASP.NET MVC or ASP.NET WebApi application, you may want to import the `Microsoft.AspNetCore.Mvc.WebApiCompatShim` package which provides some compatibility types which makes migrations easier from the previous versions. Among them is the `ApiController` class and certain attributes that were removed in the new webstack Api.

However, please note that this is only there to help you with migrating existing applications. When you create a new application you shouldn't use this compatibility shim and just use the new stuff.

answered Jul 30 '16 at 9:52

Tseng
37k 5 104 137