Inject service into Action Filter

Asked 5 years, 6 months ago Active 2 years, 2 months ago Viewed 51k times

67

15

I am trying to inject a service into my action filter but I am not getting the required service injected in the constructor. Here is what I have:

```
public class EnsureUserLoggedIn : ActionFilterAttribute
{
    private readonly ISessionService _sessionService;

    public EnsureUserLoggedIn()
    {
        // I was unable able to remove the default ctor
        // because of compilation error while using the
        // attribute in my controller
    }

    public EnsureUserLoggedIn(ISessionService sessionService)
    {
        _sessionService = sessionService;
    }

    public override void OnActionExecuting(ActionExecutingContext context)
    {
        // Problem: _sessionService is null here
        if (_sessionService.LoggedInUser == null)
        {
            context.HttpContext.Response.StatusCode = (int)HttpStatusCode.Unauthorized;
            context.Result = new JsonResult("Unauthorized");
        }
    }
}
```

And I am decorating my controller like so:

```
[Route("api/issues"), EnsureUserLoggedIn]
public class IssueController : Controller
{
}
```

Startup.cs

```
services.AddScoped<ISessionService, SessionService>();
```

c# asp.net-core dependency-injection .net-core

Share Edit Follow

edited Jul 17 '19 at 15:50

Nkosi

201k 30 332 397

asked Mar 20 '16 at 0:47

hyde

2,455 3 20 46

Attributes decorations allows only constant values. I guess you should resolve your service in the default constructor. – T-moty Mar 20 '16 at 1:15

I am not sure how that is possible in this scenario. – hyde Mar 20 '16 at 1:16

1 Take a look here, i guess you only need a decorator – T-moty Mar 20 '16 at 1:34

3 Instead of attempting to inject services into attributes, write passive attributes. – Mark Seemann Mar 20 '16 at 9:37

3 You shouldn't implement your own filter or authorization attributes for authorization/policy. That's what the policy builder and Policy property of `AuthoirzeAttribute` . See this answer from an ASP.NET Developer responsible for the security parts of ASP.NET Core stackoverflow.com/a/31465227/455493 – Tseng Mar 20 '16 at 11:37

6 Answers

Active | Oldest | Votes

75

Using these articles as reference:

ASP.NET Core Action Filters

Action filters, service filters and type filters in ASP.NET 5 and MVC 6

Using the filter as a ServiceFilter

Because the filter will be used as a `ServiceType` , it needs to be registered with the framework IoC. If the action filters were used directly, this would not be required.

Startup.cs

```
public void ConfigureServices(IServiceCollection services) {
    services.AddMvc();

    services.AddScoped<ISessionService, SessionService>();
    services.AddScoped<EnsureUserLoggedIn>();

    ...
}
```

Custom filters are added to the MVC controller method and the controller class using the `ServiceFilter` attribute like so:

```
[ServiceFilter(typeof(EnsureUserLoggedIn))]
[Route("api/issues")]
public class IssueController : Controller {
    // GET: api/issues
    [HttpGet]
    [ServiceFilter(typeof(EnsureUserLoggedIn))]
    public IEnumerable<string> Get(){...}
}
```

There were other examples of

- Using the filter as a global filter
- Using the filter with base controllers
- Using the filter with an order

Take a look, give them a try and see if that resolves your issue.

Hope this helps.

Share Edit Follow

edited Jan 15 '18 at 2:17

answered Mar 20 '16 at 2:42

Nkosi
201k 30 332 397

1 Yup, this worked. I was unfamiliar with the ServiceFilter attribute before. That was the piece missing in my code. Thank you. – hyde Mar 20 '16 at 2:47

1 You shouldn't do authorization policy checks this way, it's not the way in was intended to. You are supposed to use i.e. Cookie Authorization or some other authorization type (jwt for example) and then use `AuthorizeAttribute` with policies you set up on application startup. Check my comment above – Tseng Mar 20 '16 at 11:39

Interestingly a global filter does not seem to a allow for DI, see: github.com/damienbod/AspNet5Filters/blob/... - the author of the blogs actually hard-wires dependencies himself there. – Stefan Hendriks May 18 '16 at 9:18

Found a blog that actually covers Global filters + dependency injection: weblogs.asp.net/ricardoperes/... – Stefan Hendriks May 18 '16 at 9:20

4 How does this work if you need to specify properties on the attribute? – Jeremy Holovacs Jan 12 '17 at 0:13

33

Global filters

You need to implement `IFilterFactory`:

```
public class AuthorizationFilterFactory : IFilterFactory
{
    public bool IsReusable => false;

    public IFilterMetadata CreateInstance(IServiceProvider serviceProvider)
    {
        // manually find and inject necessary dependencies.
        var context = (IMyContext)serviceProvider.GetService(typeof(IMyContext));
        return new AuthorizationFilter(context);
    }
}
```

In `Startup` class instead of registering an actual filter you register your filter factory:

```
services.AddMvc(options =>
{
    options.Filters.Add(new AuthorizationFilterFactory());
});
```

Share Edit Follow

edited Sep 28 '16 at 12:47

answered Jul 27 '16 at 15:16

Andrei

38k 32 142 197

1 Thanks for that! This was not an obvious solution I'd have found easily on my own. – Stephen M. Redd Aug 11 '16 at 13:40

@StephenM.Redd make sure it is not Reusable. Otherwise it will be using disposed context. – Andrei Aug 14 '16 at 0:59

32

One more way for resolving this problem. You can get your service via Context as in the following code:

```
public override void OnActionExecuting(ActionExecutingContext context)
{
    _sessionService = context.HttpContext.RequestServices.GetService<ISessionService>
();
    if (_sessionService.LoggedInUser == null)
    {
        context.HttpContext.Response.StatusCode = (int)HttpStatusCode.Unauthorized;
        context.Result = new JsonResult("Unauthorized");
    }
}
```

Please note that you have to register this service in Startup.cs

```
services.AddTransient<ISessionService, SessionService>();
```

Share Edit Follow

answered Aug 19 '16 at 12:38

Igor Valikovsky

536 5 5

I can't use GetService<IMyService> . It tells me to use Microsoft.Extensions.DependencyInjection.ServerProviderServiceExtensions.GetService, but I can't seem to find that available. – SventoryMang Aug 24 '17 at 22:46

2 This looks like the best solution for me with this small adjustment: var service = context.HttpContext.RequestServices.GetService(typeof(IMyService)) as IMyService; – BrokeMyLegBiking Dec 21 '17 at 21:13

3 This may work but it is using Service Locator pattern which are sometimes considered anti pattern – Anjani Mar 28 '18 at 7:24

Example

12

```
private ILoginService _loginService;

public override void OnActionExecuting(ActionExecutingContext context)
        {
            _loginService =
(ILoginService)context.HttpContext.RequestServices.GetService(typeof(ILoginService));
        }
```

Hope it helps.

Share Edit Follow

answered Apr 8 '19 at 14:35

Bukunmi

2,028 13 11

2 Simple sample but straight forward. The right one that I was looking for. By using (T)context.HttpContext.RequestServices.GetService(typeof(T)) it resolves the dependency. Good job. – Jerameel Resco Sep 4 '20 at 14:59

9

After reading this article ASP.NET Core - Real-World ASP.NET Core MVC Filters (Aug 2016) I implemented it like this:

In Starup.cs / ConfigureServices:

```
services.AddScoped<MyService>();
```

In MyFilterAttribute.cs:

```
public class MyFilterAttribute : TypeFilterAttribute
{
    public MyFilterAttribute() : base(typeof (MyFilterAttributeImpl))
    {

    }

    private class MyFilterAttributeImpl : IActionFilter
    {
        private readonly MyService _sv;

        public MyFilterAttributeImpl(MyService sv)
        {
            _sv = sv;
        }

        public void OnActionExecuting(ActionExecutingContext context)
        {
            _sv.MyServiceMethod1();
        }

        public void OnActionExecuted(ActionExecutedContext context)
```

```
        {
            _sv.MyServiceMethod2();
        }
    }
}
```

In MyFooController.cs :

```
[MyFilter]
public IActionResult MyAction()
{
}
```

Edit: Passing arguments like `[MyFilter("Something")]` can be done using the Arguments property of the TypeFilterAttribute class: How do I add a parameter to an action filter in asp.net? (rboe's code also shows how to inject things (the same way))

Share Edit Follow

edited May 23 '17 at 11:47
Community ♦
1 1

answered May 4 '17 at 10:04
A.J.Bauer
2,458 22 30

1

While the question implicitly refers to "filters via attributes", it is still worth highlighting that adding filters "globally by type" supports DI out-of-the-box:

> [For global filters added by type] **any constructor dependencies will be populated by dependency injection (DI)**. Adding a filter by type is equivalent to filters.Add(new TypeFilterAttribute(typeof(MyFilter))). https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-2.2#dependency-injection

With regards to attribute-based filters:

> Filters that are implemented as attributes and added directly to controller classes or action methods cannot have constructor dependencies provided by dependency injection (DI). This is because attributes must have their constructor parameters supplied where they're applied. This is a limitation of how attributes work. https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-2.2#dependency-injection

However, as mentioned in the previous answers to the OP, there are ways of indirection that can be used to achieve DI. For the sake of completeness, here are the links to the official docs:

- ServiceFilterAttribute
- TypeFilterAttribute
- IFilterFactory implemented on your attribute

Share Edit Follow

answered Jan 12 '19 at 12:46

B12Toaster
9,797 6 51 54