

How do I customize ASP.Net Core model binding errors?



I would like to return only standardized error responses from my Web API (Asp.net Core 2.1), but I can't seem to figure out how to handle model binding errors.

9

The project is just created from the "ASP.NET Core Web Application" > "API" template. I've got a simple action defined as:



3

```
[Route("[controller]")]
[ApiController]
public class MyTestController : ControllerBase
{
    [HttpGet("{id}")]
    public ActionResult<TestModel> Get(Guid id)
    {
        return new TestModel() { Greeting = "Hello World!" };
    }
}

public class TestModel
{
    public string Greeting { get; set; }
}
```

If I make a request to this action with an invalid Guid (eg, `https://localhost:44303/MyTest/asdf`), I get back the following response:

```
{
  "id": [
    "The value 'asdf' is not valid."
  ]
}
```

I've got the following code in `Startup.Configure` :

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
```

Join Stack Overflow to learn, share knowledge, and build your career.

[Sign up with email](#)[Sign up with Google](#)[Sign up with Facebook](#)

```
}  
  
app  
    .UseHttpsRedirection()  
    .UseStatusCodePages(async ctx => { await  
JsonErrorMiddleware.Instance.Invoke(ctx.HttpContext); })  
    .UseExceptionHandler(new ExceptionHandlerOptions() { ExceptionHandler =  
JsonErrorMiddleware.Instance.Invoke })  
    .UseMvc()  
}
```

`JsonErrorMiddleware` is simply a class that converts errors to the correct shape I want to return and puts them into the response. It is not getting called at all for the model binding errors (no `Exception` is thrown and `UseStatusCodePages` is not called).

How do I hook into the model binding to provide a standardized error response across all actions in my project?

I've read a bunch of articles, but they all seem to either discuss global exception handling or validation errors.

[c#](#) [asp.net-core](#) [asp.net-core-webapi](#) [asp.net-core-middleware](#)

asked Jul 3 '18 at 0:39



Brian

27k 15 84 103

1 Answer



21



It's worth mentioning that ASP.NET Core 2.1 added the `[ApiController]` attribute, which among other things, automatically handles model validation errors by returning a `BadRequestObjectResult` with `ModelState` passed in. In other words, if you decorate your controllers with that attribute, you no longer need to do the `if (!ModelState.IsValid)` check.

Additionally, the functionality is also extensible. In

21 watchers 173 questions

Questions about Asp.Net Core's application pipeline that handles requests and responses.

```
services.Configure<ApiBehaviorOptions>(o =>
```

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



Sign up with Google

Sign up with Facebook



The above is just what already happens by default, but you can customize the lambda there that `InvalidModelStateResponseFactory` is set to in order to return whatever you like.

answered Jul 3 '18 at 17:05



Chris Pratt

168k

22

256

322

For completeness - the `ApiBehaviorOptions` will work on controllers only with `ApiController` attribute decorated! That also means that `Route` attribute on the controller itself has to be also provided and also the way how binding is done will change also. Details provided in the link here:

stackoverflow.com/questions/45941246/... – baHI Oct 16 '18 at 8:52

Heres the link to the actual ASP.NET Core 2.2 source code for the default `ApiBehaviourOptions` values:

github.com/aspnet/AspNetCore/blob/release/2.2/src/Mvc/Mvc.Core/... – KGC May 1 at 20:21

Got a question that you can't ask on public Stack Overflow? [Learn more](#) about sharing private information with Stack Overflow for Teams.



Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



Sign up with Google

Sign up with Facebook

