How to customize error response in web api with .net core?

1

I am using .net core 2.2 with web api. I have created one class i.e. as below :

```
public class NotificationRequestModel
{
    [Required]
    public string DeviceId { get; set; }
    [Required]
    public string FirebaseToken { get; set; }
    [Required]
    public string OS { get; set; }

    public int StoreId { get; set; }
}
```

Using above class I have created one method. Now I want to return custom object but it's return it's own object. API method is :

```
public ActionResult<bool> UpdateFirebaseToken(NotificationRequestModel model)
   {
       if (!ModelState.IsValid)
       {
          return BadRequest(FormatOutput(ModelState.Values));
       }
       var result = _notificationService.InsertOrUpdateFirebaseToken(model);
       return Ok(result);
   }
```

Here `FormatOutput` method is format the output.

```
protected Base FormatOutput(object input, int code = 0, string message = "", string[]
details = null)
    {
        Base baseResult = new Base();
        baseResult.Status = code;
```

```
            return baseResult;
        }
```

But the issue is it returns :

> { "errors": { "DeviceId": ["The DeviceId field is required."] }, "title": "One or more validation errors occurred.", "status": 400, "traceId": "80000049-0001-fc00-b63f-84710c7967bb" }

I want to customize this error with my model. I need error message and details from return output and passed it to my model. How can I do that? I had try to debug my code and found that breakpoint on API method is not calling. So I can't handle my custom method. Is there any solution? What am I doing wrong? Any help will be much appreciated. Thank you.

asp.net-core model asp.net-core-webapi

asked Mar 1 at 10:00
Srusti Thakkar
732 8 31

2 Answers

1 ✓

As Chris analized, your issue is caused by Automatic HTTP 400 responses .

For the quick solution, you could supress this feature by

```
services.AddMvc()
        .ConfigureApiBehaviorOptions(options => {
            options.SuppressModelStateInvalidFilter = true;
        }).SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
```

For a efficient, you could follow suggestion from Chris, like below:

```
                return new BadRequestObjectResult(FormatOutput(modelState));
            };
        }).SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
```

And, there is no need to define the code below any more in your action.

```
if (!ModelState.IsValid)
    {
        return BadRequest(FormatOutput(ModelState.Values));
    }
```

answered Mar 4 at 6:13

Tao Zhou

9,668 3 16 34

2

When utilizing a controller with the `ApiController` attribute applied, ASP.NET Core *automatically* handles model validation errors by returning a 400 Bad Request with `ModelState` as the response body. As such, your conditional testing `ModelState.IsValid` is essentially always false (and therefore not entered) because the only requests that will ever get this far are valid ones.

You could simply remove the `ApiController` attribute, but that removes a bunch of other beneficial stuff the attributes adds as well. The better option is to use a custom response factory:

```
services.Configure<ApiBehaviorOptions>(o =>
{
    o.InvalidModelStateResponseFactory = actionContext =>
        new BadRequestObjectResult(actionContext.ModelState);
});
```

That's essentially what's happening by default, so you'd simply need to change the action provided there accordingly to customize it to your whims.

answered Mar 1 at 15:36

Chris Pratt