Ajay Patel

❤.NET

I don't know about [ApiController] in .NET

→

Ajay Patel

ApiController attribute

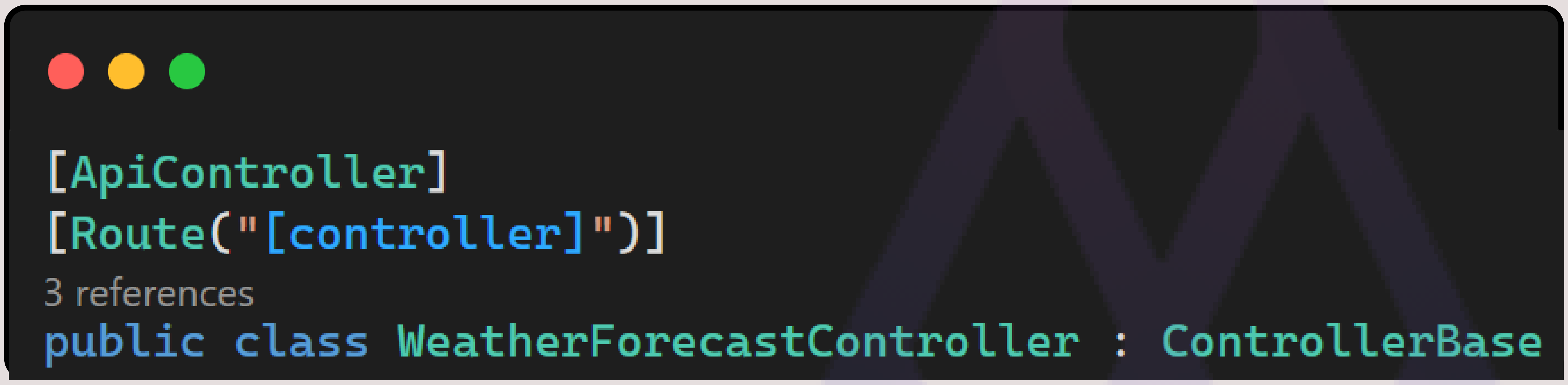
The [ApiController] attribute in ASP.NET Core is used to indicate that a controller is intended to serve as an API endpoint. It simplify the development of RESTful APIs.

It can be applied to a controller class to enable the following **opinionated,** API-specific behaviors:

- Attribute routing requirement
- Automatic HTTP 400 responses
- Binding source parameter inference
- Problem details for error status codes

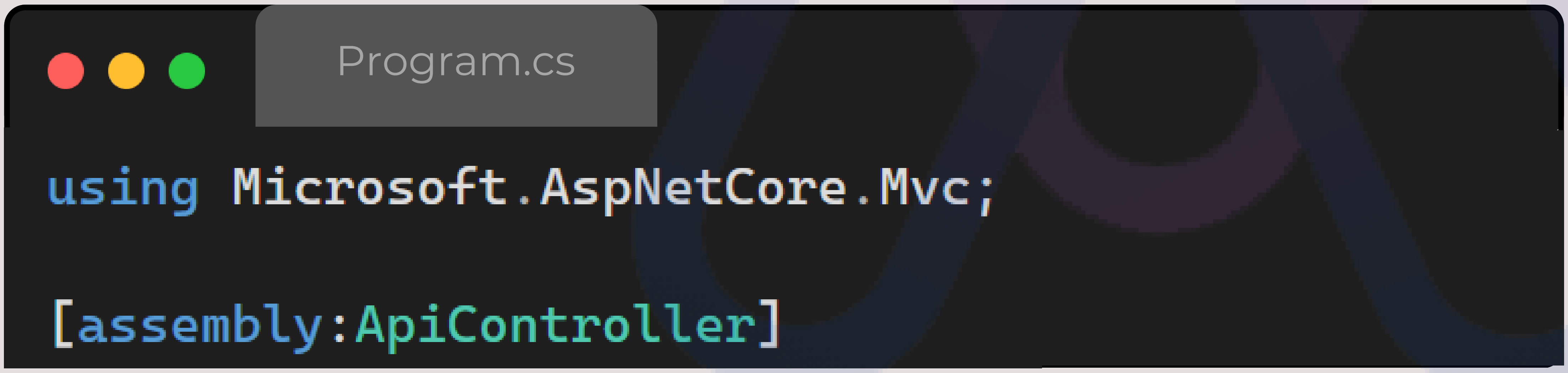
# How to Use ApiController Attribute?

Apply directly to individual controllers:



```
[ApiController]
[Route("[controller]")]
3 references
public class WeatherForecastController : ControllerBase
```

Apply attribute on an assembly



Program.cs

```
using Microsoft.AspNetCore.Mvc;

[assembly:ApiController]
```

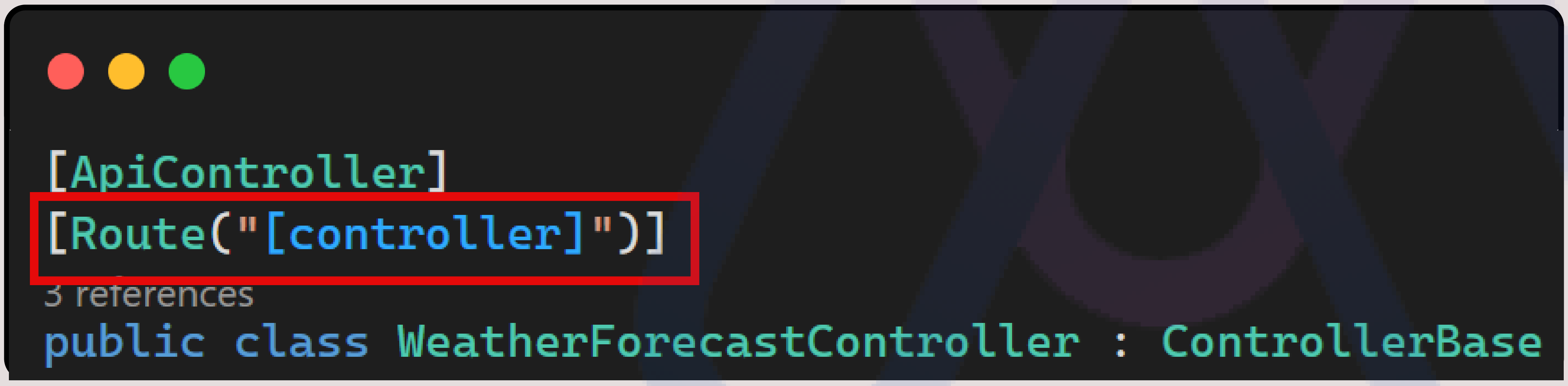
Lastly, We can also apply it to a custom controller base class. Then, all its subclasses will inherit ApiController behavior.

Ajay Patel

Key Features

1. Attribute routing requirement

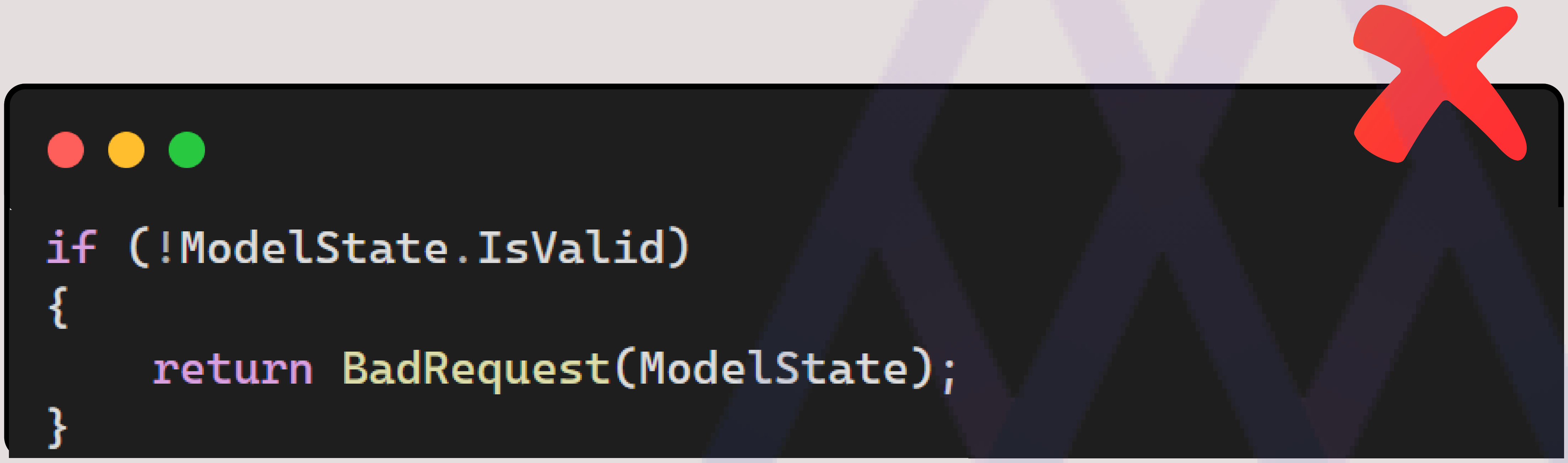
When you apply the [ApiController] attribute to a controller in ASP.NET Core, it enforces attribute routing, meaning actions in that controller will not be accessible via conventional routing (e.g., UseEndpoints, UseMvc, or UseMvcWithDefaultRoute)



```
[ApiController]
[Route("[controller]")]
3 references
public class WeatherForecastController : ControllerBase
```

2. Automatic HTTP 400 Responses:

When [ApiController] is applied, it automatically validates the model and responds with an HTTP 400 status if there are validation errors. **This means we don't need to manually check ModelState.IsValid in our actions.**



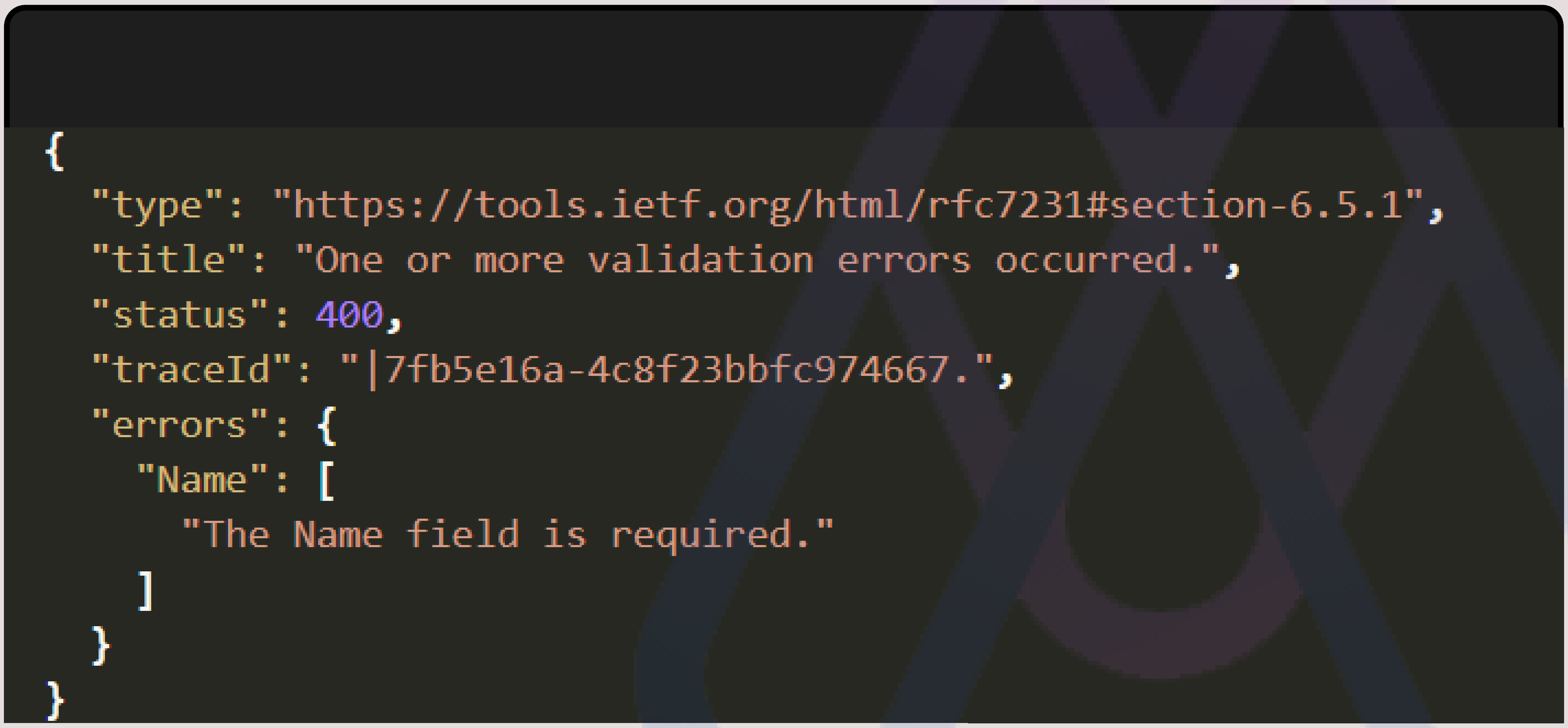
```
if (!ModelState.IsValid)
{
    return BadRequest(ModelState);
}
```

ASP.NET Core MVC uses the **ModelStateInvalidFilter** action filter to do the preceding check.

Ajay Patel

3. Default BadRequest response:

The default response type for an HTTP 400 response is **ValidationProblemDetails**. The following response body is an example of the serialized type:



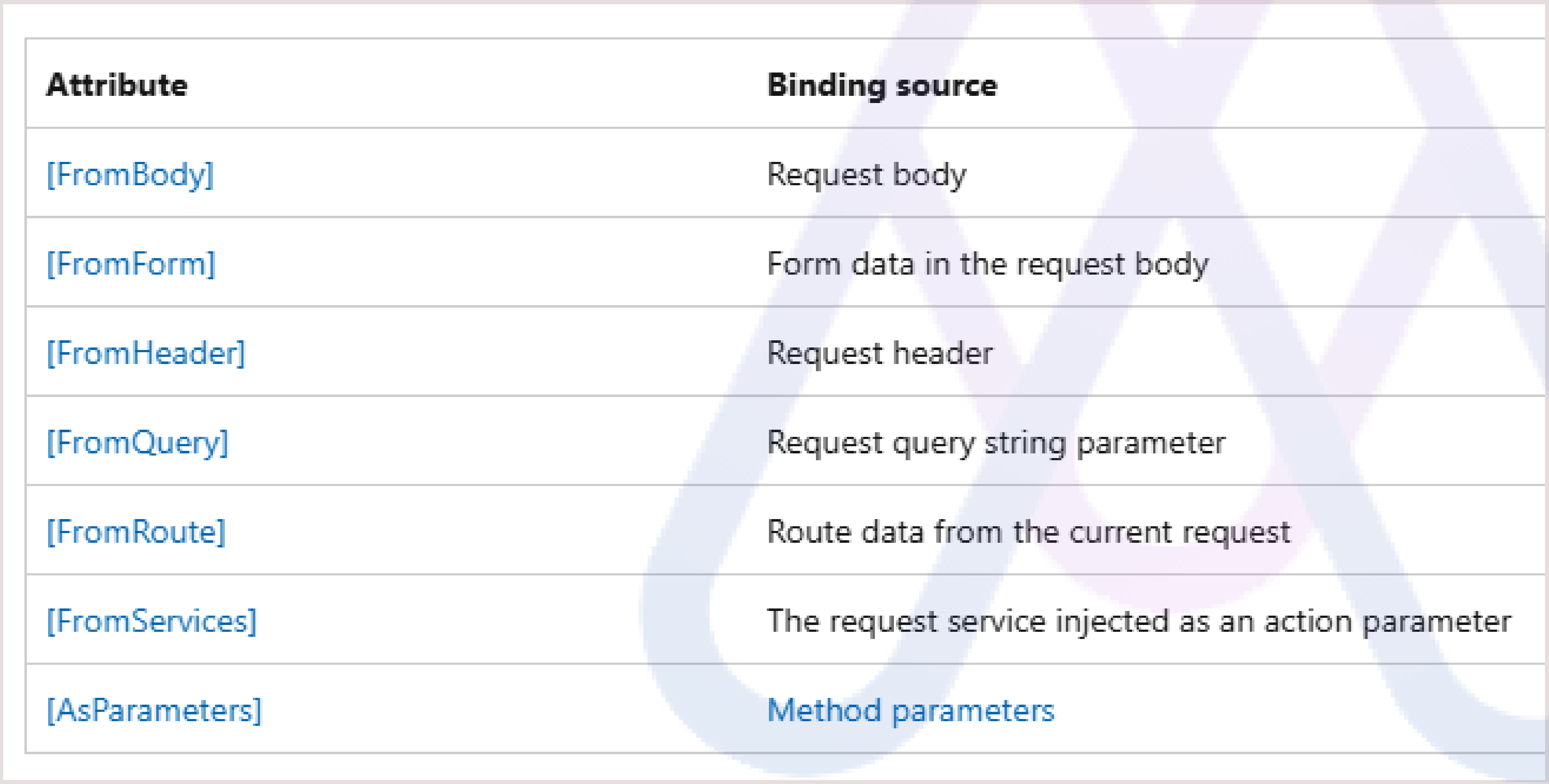
```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "traceId": "|7fb5e16a-4c8f23bbfc974667.",
  "errors": {
    "Name": [
      "The Name field is required."
    ]
  }
}
```

We can fully customize our validation problem responses as well. To do that, we provide our own implementation of the InvalidModelStateResponseFactory delegate in the **ConfigureApiBehaviorOptions()** extension method in Program.cs file.

Ajay Patel

4. Binding source parameter inference:

[ApiController] automatically determines the source of certain action parameters, so we don't always need to use binding attributes like [FromBody] or [FromQuery]. It follows specific inference rules to make this decision.



Attribute	Binding source
[FromBody]	Request body
[FromForm]	Form data in the request body
[FromHeader]	Request header
[FromQuery]	Request query string parameter
[FromRoute]	Route data from the current request
[FromServices]	The request service injected as an action parameter
[AsParameters]	Method parameters

Ajay Patel

❤.NET

Knowledge is contagious, let's spread it!

DO YOU LIKE THIS POST?

REPOST IT!

THANKS FOR READING