MINIMAL APIS

ENDPOINT FILTER

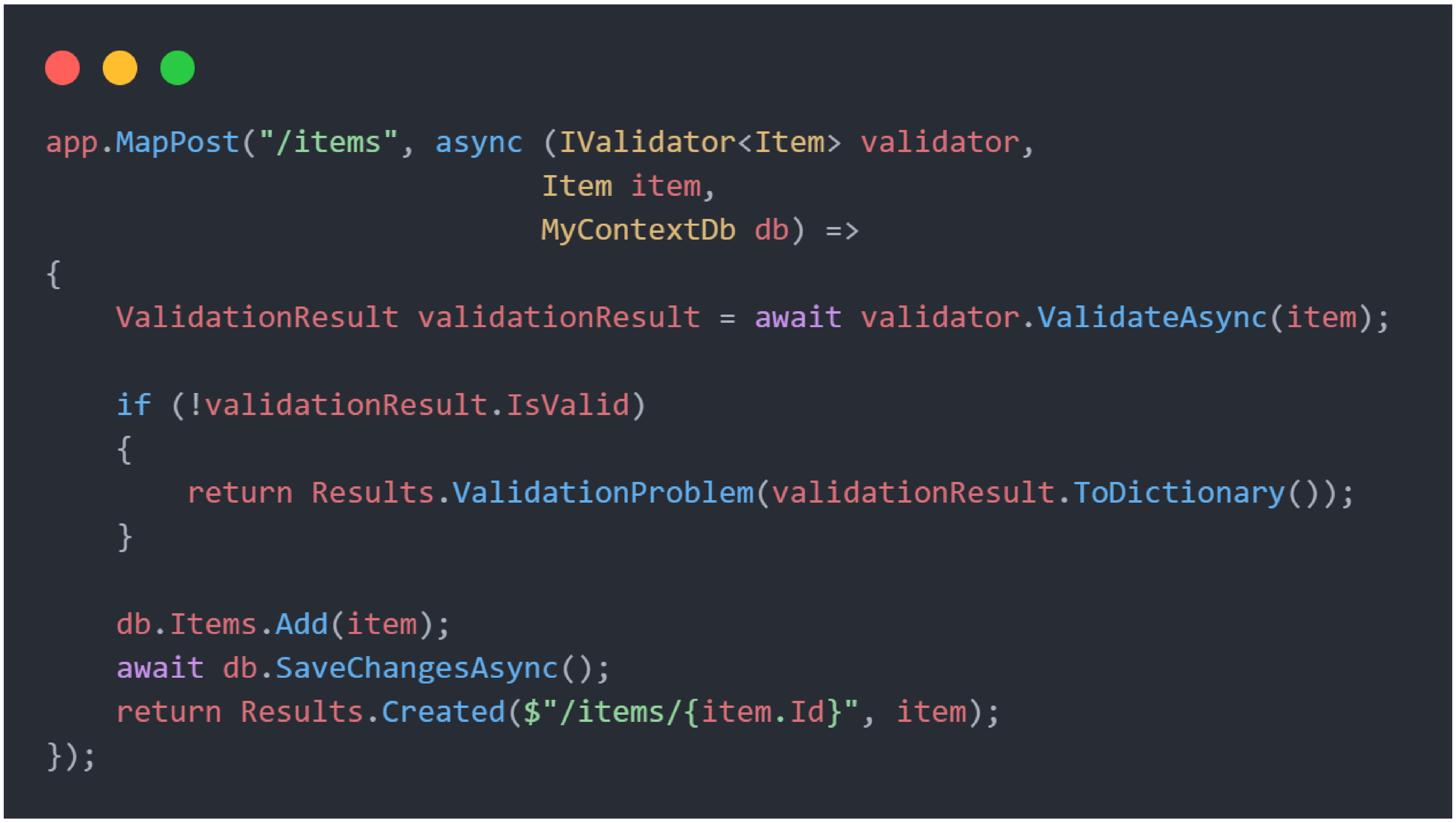
Minimal API filters allow developers to implement business logic that supports:

- Running code before and after the endpoint handler.
- Inspecting and modifying parameters provided during an endpoint handler invocation.
- Intercepting the response behavior of an endpoint handler.

@romain-od
www.code-review.tech

FLUENT WITHOUT FILTER

Here's a working example of using of fluentvalidation. Let's explore how we can use enpoint filter for this case.



```
app.MapPost("/items", async (IValidator<Item> validator,
                             Item item,
                             MyContextDb db) =>
{
    ValidationResult validationResult = await validator.ValidateAsync(item);

    if (!validationResult.IsValid)
    {
        return Results.ValidationProblem(validationResult.ToDictionary());
    }

    db.Items.Add(item);
    await db.SaveChangesAsync();
    return Results.Created($"/items/{item.Id}", item);
});
```

@romain-od

www.code-review.tech

CREATE CUSTOM ENDPOINT FILTERS

```csharp
public class ValidationFilter<T> : IEndpointFilter
{
    public async ValueTask<object?> InvokeAsync(EndpointFilterInvocationContext context,
                                                EndpointFilterDelegate next)
    {
        T? argToValidate = context.GetArgument<T>(0);
        IValidator<T>? validator = context.HttpContext
                                      .RequestServices
                                      .GetService<IValidator<T>>();

        if (validator is not null)
        {
            var validationResult = await validator.ValidateAsync(argToValidate!);
            if (!validationResult.IsValid)
            {
                return Results.ValidationProblem(validationResult.ToDictionary(),
                    statusCode: (int)HttpStatusCode.UnprocessableEntity);
            }
        }

        // Otherwise invoke the next filter in the pipeline
        return await next.Invoke(context);
    }
}
```

@romain-od

www.code-review.tech

ACTIVATING OUR **FILTER**

We can now simplify the above endpoint by removing the inline validation logic

```
app.MapPost("/items", async (Item item,
                              MyContextDb db) =>
{
    db.Items.Add(item);
    await db.SaveChangesAsync();
    return Results.Created($"/items/{item.Id}", item);
})
.AddEndpointFilter<ValidationFilter<Item>>();
```

THANKS FOR READY

SHARE YOUR THOUGHTS

FOLLOW ME FOR MORE SHARING

@romain-od

www.code-review.tech