Reporting Validation Errors

* The out-of-the-box format used for this is the problem details format based on the problem details for HTTP APIs RFC.
* The standard aims to provide a unified way for reporting API problems. HTTP status codes are sometimes not sufficient to convey enough information about an error to be helpful.
* As a human, we can be informed of an issue by having the API returned some HTML, for example. But nonhuman consumers are better off with a specific document format, and that's what the standard aims to provide, a document format that's designed to be reused by HTTP APIs, which can identify distinct problem types specific to their needs.
* Thus API clients can be informed of both the high-level error class by using the status code, and the finer-grain details of the problem by using the problem-details format. So this is a pretty good standard of which I'm quite happy the ASP.NET Core team adopted it. But the implementation isn't yet perfect.

Diagram

Description automatically generated

* Let's have a look at an example from the previous demo. The Content-Type is application/problem+json. That's a correct implementation of the standard. Then we see the validation errors. So far, so good, but we don't get much detail except for that. There's no type, no detail, and no instance information. Also, it's by default returned as a 400 Bad Request, and we learned that a 422 Unprocessable Entity Result is a better fit fit for validation errors.

Graphical user interface, text, application, email

Description automatically generated

* So it would be nice if we could end up with something like this.
* We provide a value for type, the idea is that this provides your reference to an HTML page consumers of the API can consult to get more information on errors like this, so it should be a URI. The status is 422. We also pass through a readable message that states where the details can be found, and we pass through the instance. That's the path of the resource. For creating, this might not look so useful, but once we start updating resources in the next module, this will refer to the actual resource we're updating, which gives the consumer the correct way to reference that resource. Luckily, we can customize the behavior of ApiController so we can actually get this type of result. Let's learn how.

Text

Description automatically generated