Using and Overriding Conventions for OpenAPI Generation

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

The importance of ApiExplorer

Producing correct response types (status codes)

Improving our code with API analyzers

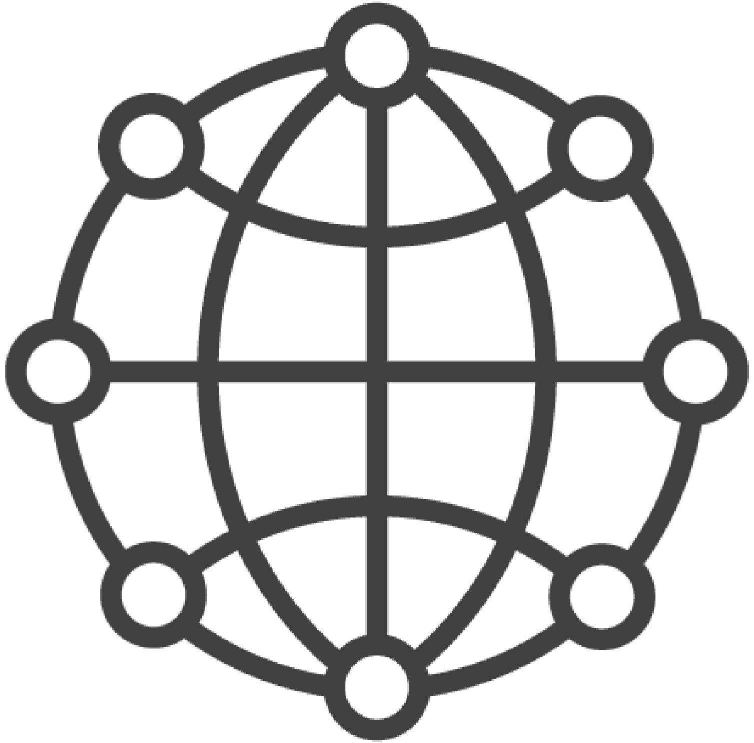
Convention-based OpenAPI generation

Content negotiation with the Produces and Consumes attributes

ApiExplorer is an abstraction on top of ASP.NET Core MVC that exposes metadata about that application

The importance of ApiExplorer

The Importance of ApiExplorer



Swashbuckle uses the metadata ApiExplorer exposes to generate an OpenAPI specification

ApiExplorer is enabled by default

- It’s registered when calling services.AddMvc()

Why It's Important to Produce Correct Response Types

An OpenAPI specification should include all possible response types (404, 422, …) for a method/resource URI

- Allows consumers to act accordingly
- Our specification must match the reality of our API

Demo

Describing response types (status codes) with ProducesReponseType

Demo

Using API analyzers to improve the OpenAPI specification

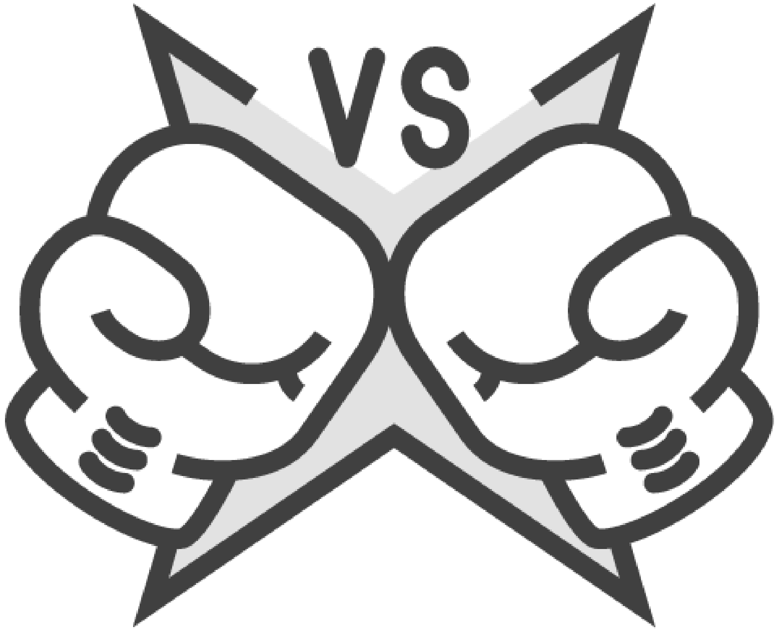
Demo

Working with API conventions

Demo

Creating custom conventions

Attributes Versus Conventions



Conventions

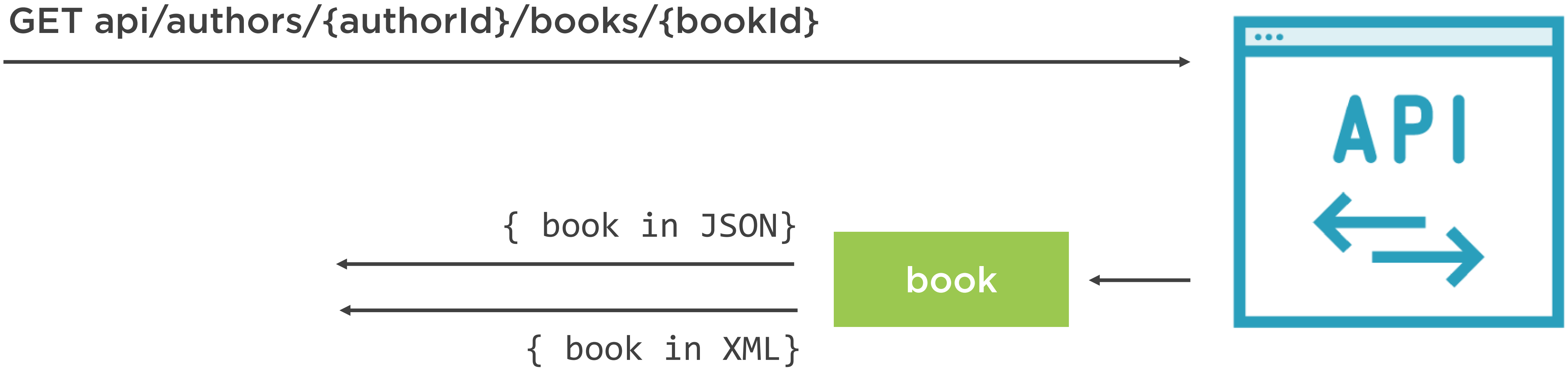
- Are overridden by attributes
- One mistake can have dire consequences
- Good for very simple APIs, hard for anything but the basics

Use attributes instead of conventions

Best Practices for Using Attributes

- ✓ Use API Analyzer, but don't rely on it to give you full coverage
- ✓ Use ProducesDefaultResponseType, but be specific where possible
- ✓ Apply attributes globally where possible

Content Negotiation



GET api/authors/{authorId}/books/{bookId}

{ book in JSON}

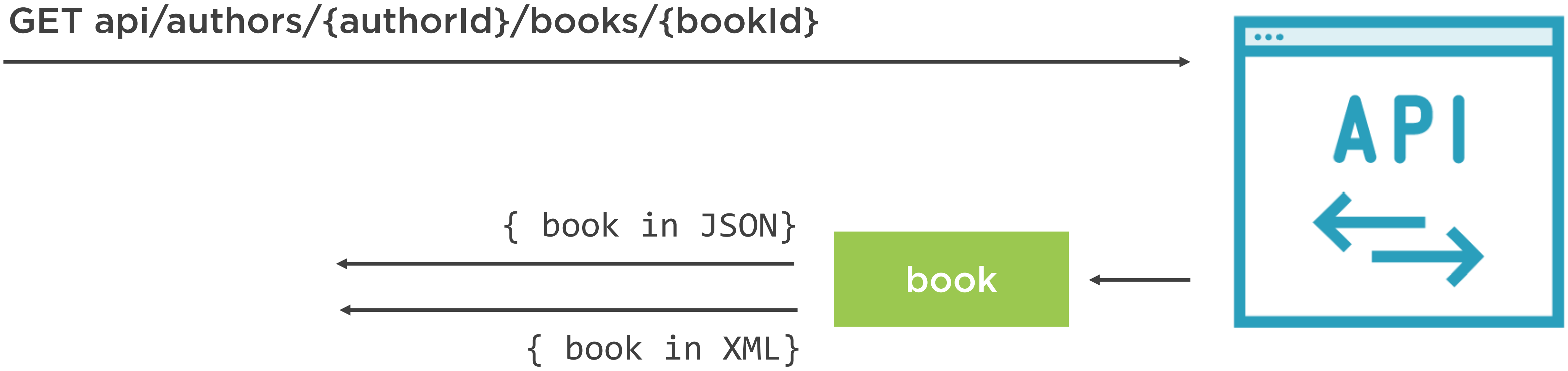
book

{ book in XML}

Content Negotiation

Content negotiation is the mechanism used for serving different representations of a resource at the same URI

Content Negotiation



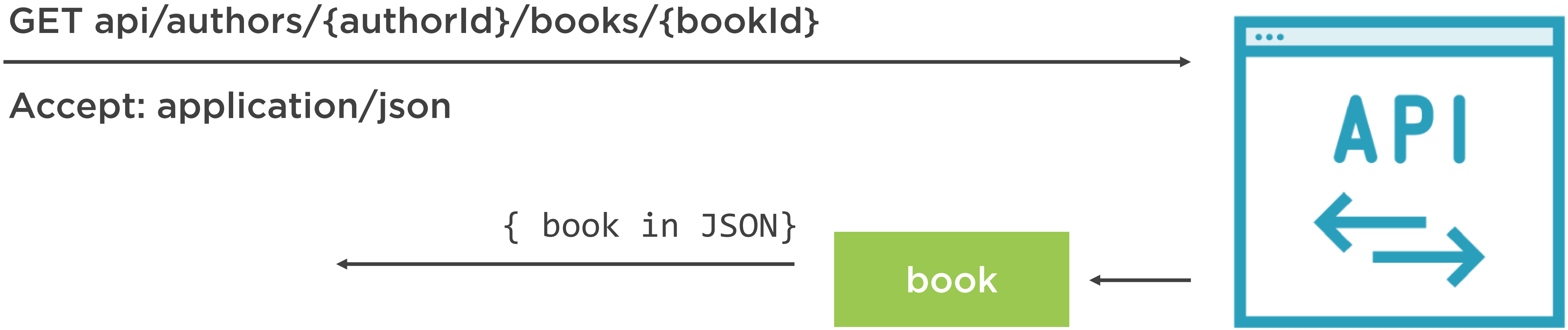
GET api/authors/{authorId}/books/{bookId}

{ book in JSON}

book

{ book in XML}

Content Negotiation



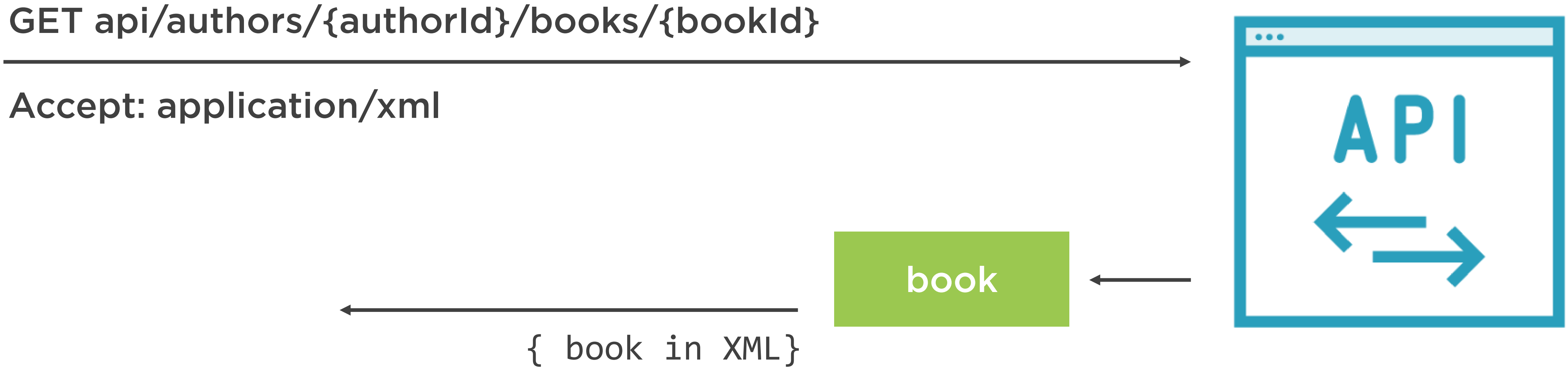
GET api/authors/{authorId}/books/{bookId}

Accept: application/json

`{ book in JSON}`

book

Content Negotiation



GET api/authors/{authorId}/books/{bookId}

Accept: application/xml

book

{ book in XML}

Demo

Specifying the response body type with the Produces attribute

Demo

Specifying the request body type with the Consumes attribute

Summary

Use the ProducesResponseType attribute to specify the possible response types (status codes)

Use API Analyzers to help with discovering these (but don’t rely on them alone)

Prefer attributes over conventions

Summary

Use the Produces attribute to specify the response body (media) type

Use the Consumes attribute to specify the request body (media) type