Getting Resources

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

Structuring our outer facing contract

Getting resources

- Single
- Collection
- Parent/Child

HTTP methods and routing

The importance of status codes

Faults and errors

Content negotiation

Structuring our Outer Facing Contract

Resource Identifier

http://host/api/authors

HTTP Method

http://bit.ly/2cNr8zu

Payload

(representation: media types)

Resource Naming Guidelines

Nouns: things, not actions

- ~~api/getauthors~~
- GET api/authors
- GET api/authors/{authorId}

Convey meaning when choosing nouns

Resource Naming Guidelines

Follow through on this principle for predictability

- ~~api/something/somethingelse/employees~~
- api/employees
- ~~api/id/employees~~
- api/employees/{employeeId}

Resource Naming Guidelines

Represent hierarchy when naming resources

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

Resource Naming Guidelines

Filters, sorting orders, ... aren't resources

- ~~api/authors/orderby/name~~
- api/authors?orderby=name

Resource Naming Guidelines

Sometimes, RPC-style calls don't easily map to pluralized resource names

- ~~api/authors/{authorId}/pagetotals~~
- ~~api/authorpagetotals/{id}~~
- api/authors/{authorId}/totalamountofpages

Resource Naming Guidelines

REST stops at the outer facing contract

- Underlying layers are of no importance
- The resource is conceptually different from what's in the backend data store

Resource Naming Guidelines

Resources are often named using an auto-numbered DB field as part of their URI

- Resource URIs should remain the same

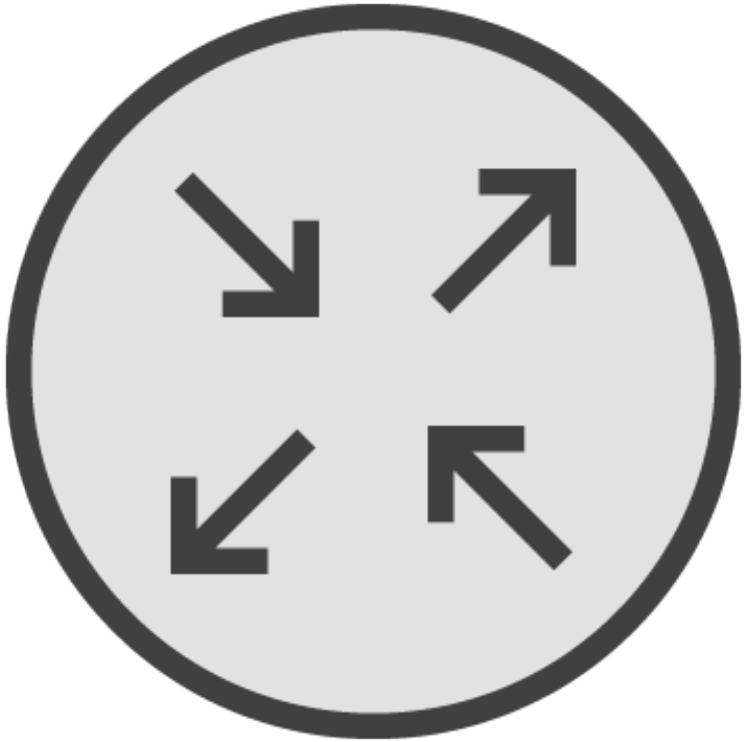
GUIDs can be used instead

- Allows switching out backend data stores
- Potentially hides implementation details

Demo

Implementing the Outer Facing Contract (Part 1)

Working with Routing



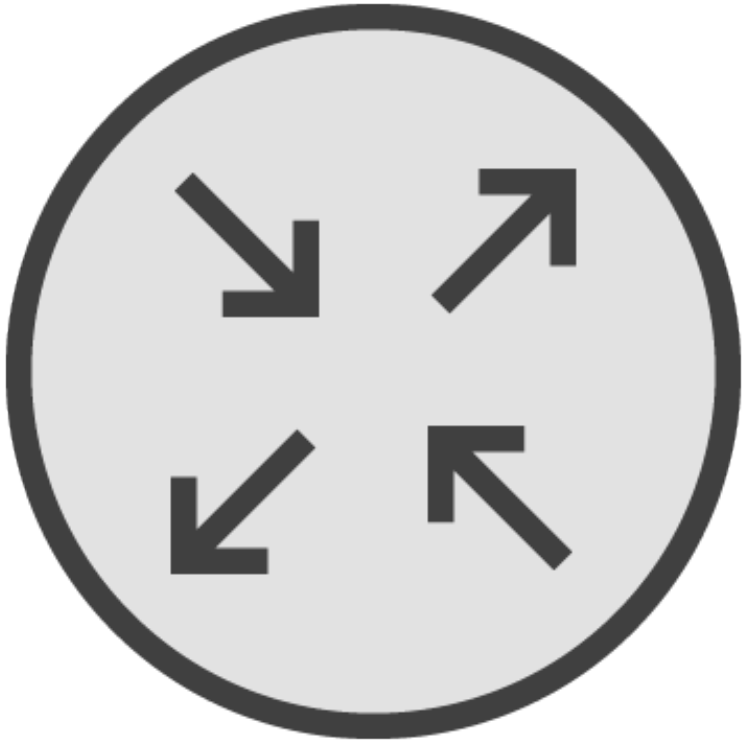
Routing matches a request URI to an action on a controller

```
app.UseMvc(routes =>{
    routes.MapRoute(
        name: "default",
        template: "{controller=Values}/{action=Index}/{id?}");
});
```

Convention-based Routing

The ASP.NET Core team advises to use attribute-based routing for APIs

Working with Routing



Use attributes at controller and action level: [Route], [HttpGet], ...

Combined with a URI template, requests are matched to controller actions

Outer Facing Contract

Resource Identifier

http://host/api/authors

HTTP Method

http://bit.ly/2cNr8zu

Payload

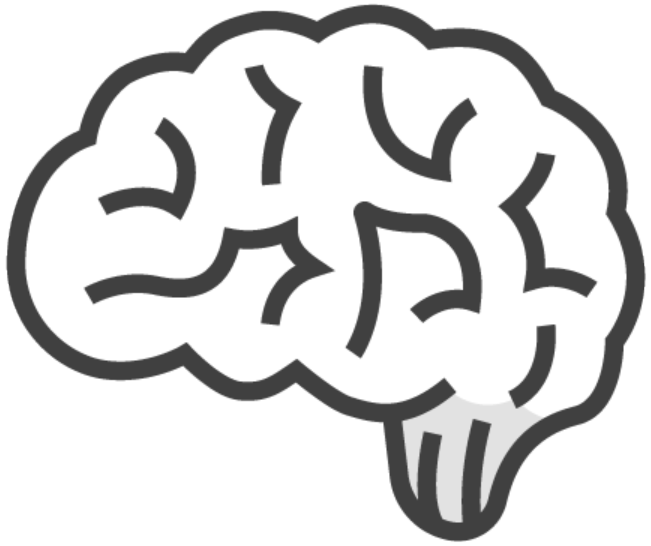
(representation: media types)

Interacting with Resources Through HTTP Methods

HTTP Method	Request Payload	Sample URI	Response Payload

Demo

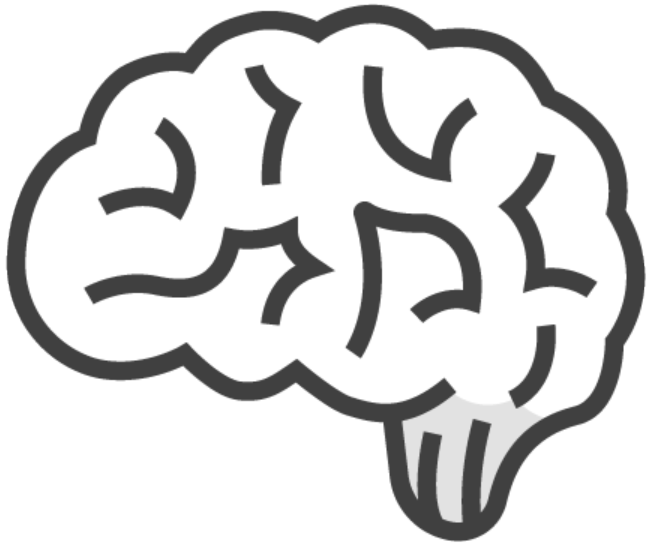
Implementing the Outer Facing Contract (Part 2)



Outer Facing Model

!=

Entity Model



Outer Facing Model

!=

Business Model

!=

Entity Model

Outer Facing Model Versus Entity Model

Outer facing model (Author)

```
Guid Id

string FirstName

string LastName

int Age
```

Entity model (Author)

```
Guid Id

string FirstName

string LastName

DateTimeOffset DateOfBirth
```

Outer Facing Model Versus Entity Model

Outer facing model (Author)

```
Guid Id

string Name

int Age
```

Entity model (Author)

```
Guid Id

string FirstName

string LastName

DateTimeOffset DateOfBirth
```

Outer Facing Model Versus Entity Model

Outer facing model (Author)

```
Guid Id

string Name

int Age

float Royalties
```

Entity model (Author)

```
Guid Id

string FirstName

string LastName

DateTimeOffset DateOfBirth
```

Demo

Getting a Resource Collection

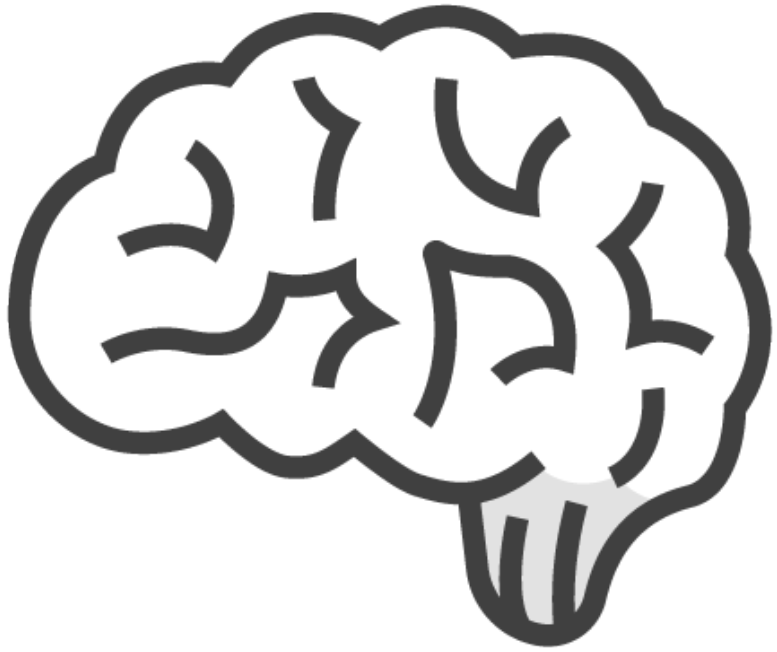
Demo

Introducing AutoMapper

Demo

Getting a Single Resource

The Importance of Status Codes



Status codes tell the consumer of the API

- Whether or not the request worked out as expected
- What is responsible for a failed request

The Importance of Status Codes

Level 200 - Success

200 – Ok

201 – Created

204 – No content

Level 400 – Client Mistakes

400 – Bad request

401 – Unauthorized

403 – Forbidden

404 – Not found

The Importance of Status Codes

Level 400 – Client Mistakes

405 – Method not allowed

406 – Not acceptable

409 - Conflict

415 – Unsupported media type

422 – Unprocessable entity

Level 500 Server Mistakes

500 – Internal server error

Errors Versus Faults

Errors	Faults
Consumer passes invalid data to the API, and the API correctly rejects this	API fails to return a response to a valid request
Level 400 status codes	Level 500 status codes
Do not contribute to API availability	Do contribute to API availability

Demo

Returning Correct Status Codes

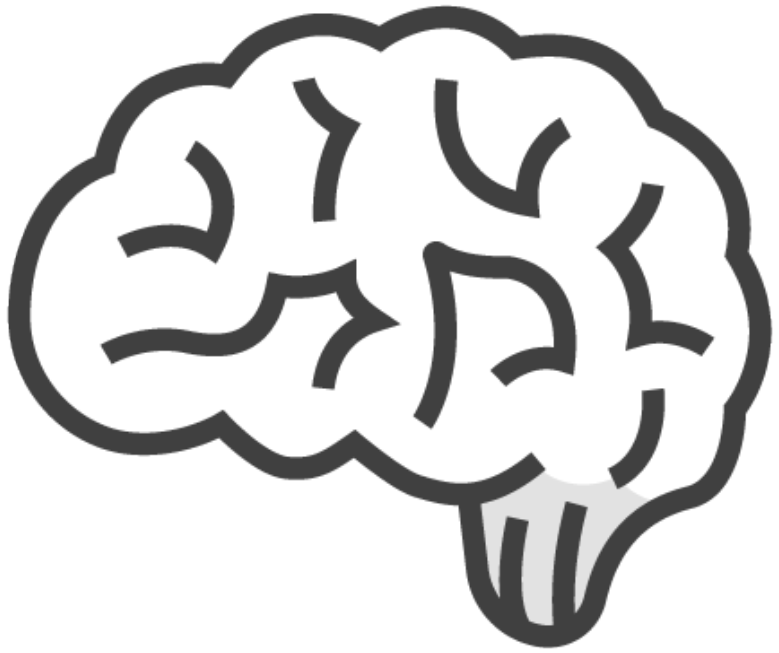
Demo

Handling Faults

Demo

Working with Parent/Child Relationships

Formatters and Content Negotiation

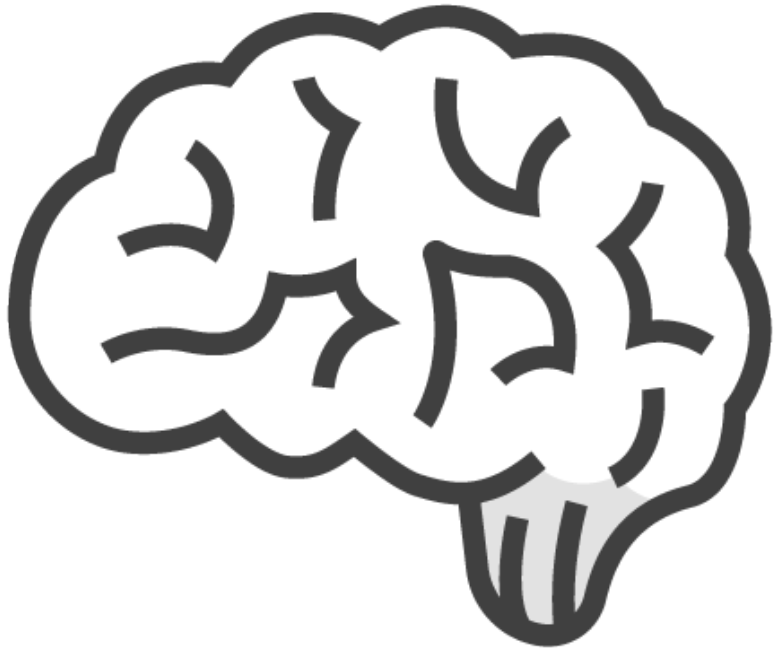


Selecting the best representation for a given response when there are multiple representations available

Media type is passed via the Accept header of the request

- application/json
- application/xml
- ...

Formatters and Content Negotiation



Returning a representation in a default format when no Accept header is included is acceptable

Returning a representation in a default format when the requested media type isn't available isn't acceptable

- Return 406 – Not acceptable

Formatters and Content Negotiation

Output formatter

Deals with output

Media type: accept header

Input formatter

Deals with input

Media type: content-type header

Demo

Working with Content Negotiation and Output Formatters

Summary

Outer facing contract

- Resource identifiers
- HTTP methods
- Optional payload

Resource identifiers

- Use pluralized nouns that convey meaning
- Represent model hierarchy
- Be consistent

Summary

HTTP methods

- GET
- POST
- PUT/PATCH
- DELETE
- HEAD
- OPTIONS

Payload

- Media type: application/json, application/xml, ...

Summary

Status codes

- Level 200: success
- Level 400: errors (client)
- Level 500: faults (server)

The outer facing model is conceptually different from the business model and/or entity model