Structuring and Implementing the Outer Facing Contract

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

Structuring our outer facing contract

- Resource URI
- HTTP Method
- Payload

Endpoint routing

Status codes

Content negotiation

Structuring Our Outer Facing Contract

Resource Identifier

http://host/api/authors

HTTP Method

http://bit.ly/2cNr8zu

Payload

(representation: media types)

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}/courses`
- `api/authors/{authorId}/courses/{courseId}`

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`

Demo

Implementing the outer facing contract (Part 1)

Routing

Routing matches a request URI to an action on a controller

Working with Endpoint Routing

app.UseRouting()

- Marks the position in the middleware pipeline where a routing decision is made

app.UseEndpoints()

- Marks the position in the middleware pipeline where the selected endpoint is executed

```
app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints => { // map endpoints });
```

Working with Endpoint Routing

Middleware that runs in between selecting the endpoint and executing the selected endpoint can be injected

```
app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints => {
        endpoints.MapControllerRoute(
             name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
        endpoints.MapRazorPages();});
```

Convention-based Routing

Endpoints are added to actions on a controller following a convention

```
app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints => {
        endpoints.MapControllers();});
```

Attribute-based Routing

No conventions are applied

This is the preferred approach for APIs

Working with Endpoint 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
GET	-	/api/authors /api/authors/{authorId}	author collection single author
POST	single author	/api/authors	single author
PUT	single author	/api/authors/{authorId}	single author or empty
PATCH	JsonPatchDocument on author	/api/authors/{authorId}	single author or empty
DELETE	-	/api/authors/{authorId}	-
HEAD	-	/api/authors /api/authors/{authorId}	-
OPTIONS	-	/api/...	-

Interacting with Resources through HTTP Methods

HTTP Method	Request Payload	Sample URI	Response Payload
GET	-	/api/authors /api/authors/{authorId}	author collection single author
POST	single author	/api/authors	single author
PUT	single author	/api/authors/{authorId}	single author or empty
PATCH	JsonPatchDocument on author	/api/authors/{authorId}	single author or empty
DELETE	-	/api/authors/{authorId}	-
HEAD	-	/api/authors /api/authors/{authorId}	-
OPTIONS	-	/api/...	-

Interacting with Resources through HTTP Methods

HTTP Method	Request Payload	Sample URI	Response Payload
GET	-	/api/authors /api/authors/{authorId}	author collection single author
POST	single author	/api/authors	single author
PUT	single author	/api/authors/{authorId}	single author or empty
PATCH	JsonPatchDocument on author	/api/authors/{authorId}	single author or empty
DELETE	-	/api/authors/{authorId}	-
HEAD	-	/api/authors /api/authors/{authorId}	-
OPTIONS	-	/api/...	-

Demo

Implementing the outer facing contract (Part 2)

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 vs. 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

```
{   "type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",
    "title": "Not Found",
    "status": 404,
    "detail": null,
    "instance": null,
    "extensions": {
        "traceId": "|cc717b15-4278504187d8c78c." }}
```

Enhancing Responses with Problem Details

Problem details for HTTP APIs RFC (https://tools.ietf.org/html/rfc7807)

- Defines common error formats for those applications that need one
- Sometimes status codes don't convey enough information
- Content-type: application/problem+json

Content Negotiation

The process of selecting the best representation for a given response when there are multiple representations available

Formatters and Content Negotiation

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

Formatters and Content Negotiation

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 (media type)

Summary

Resource identifiers

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

Summary

HTTP methods

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

Routing matches a request URI to an action on a controller

Summary

Payload

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

Content negotiation is the process of selecting the best representation for a given response when there are multiple representations available

Summary

Status codes

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