Server-Side APIs Reference

Coding Boot Camp

Module 06

HTTP - The Language of the Web!

- Workflow
  - Request →
  - Response ←
  - 1:1

- Methods
  - GET
    - Retrieve a **resource**
  - POST
    - Send/create a **resource**
  - PUT
    - Update a **resource**
  - DELETE
    - Delete a **resource**

HTTP - The Language of the Web!

- Where do I send the request?
  - API “Endpoint”
    - AKA “URI”
    - AKA “HATEOAS URL”
  - Example:
    - http://api.somewebsite.com/v1/libraries/book

- What is a resource?
  - The top-level of the path of the endpoint
  - Example:
    - http://api.somewebsite.com/v1/libraries/ **book**
      - “book” in the above example is the resource
      - Always a noun

HTTP - The Language of the Web!

- Specifying data
  - Path parameters
    - Variable parts of the path of an endpoint:
      - `http://api.somewebsite.com/v1/libraries/`**`horror`**`/books`
      - In the above example, “horror” can be replaced with any number of genre that the API supports
    - Never the top-part of the path (that’s the resource)

HTTP - The Language of the Web!

- Specifying data
  - Query parameters
    - Typically filter our responses somehow
      - Start after the resource with a ‘?’ symbol
      - Multiple query parameters are separated by ‘&’
      - Follow key/value pattern with an ‘=’ symbol between the two
    - Example:

`http://api.somewebsite.com/v1/libraries/horror/books` **`?year=2022&author=colbert`**

HTTP - The Language of the Web!

- Specifying data
  - Request body
    - Typical of POST requests
    - Aren’t part of the endpoint in any way
    - *How* you add the body to the request depends on the HTTP client/package you are using
    - JSON(typically)!:

```
{
    id: 123,
    username: erosas
}
```

HTTP - The Language of the Web!

- Responses
    - If there is any data coming from the server-side API, it comes in the form of the **response body**
    - Also JSON, typically

HTTP - The Language of the Web!

- Headers
  - Metadata about the request/response
  - Follow key/value as well
  - The only required header for a request is:
    - `Host: (domain name of API)`
    - Example:
      - `Host: http://api.github.com`
  - *Almost* without exception, these headers are also usually present in the request:
    - `Accept: application/json`
    - `Content-type: application/json`

HTTP - The Language of the Web!

- HTTP status codes:
  - 1XX
    - “Continue” codes
    - You will rarely see these or need to pay attention to them
  - 2XX level codes:
    - Successful request codes
  - 3XX
    - Codes that indicate some kind of redirect by the server
  - 4XX
    - Client error codes
  - 5XX
    - Server-side error codes

HTTP - The Language of the Web!

- Common HTTP status codes:
  - 200
    - Everything is just fine and dandy, your request received a response
  - 400
    - Bad request
    - Usually bad endpoint, path params, query params, or missing or bad headers
  - 401
    - Unauthorized
    - Usually when you forget an API key

HTTP - The Language of the Web!

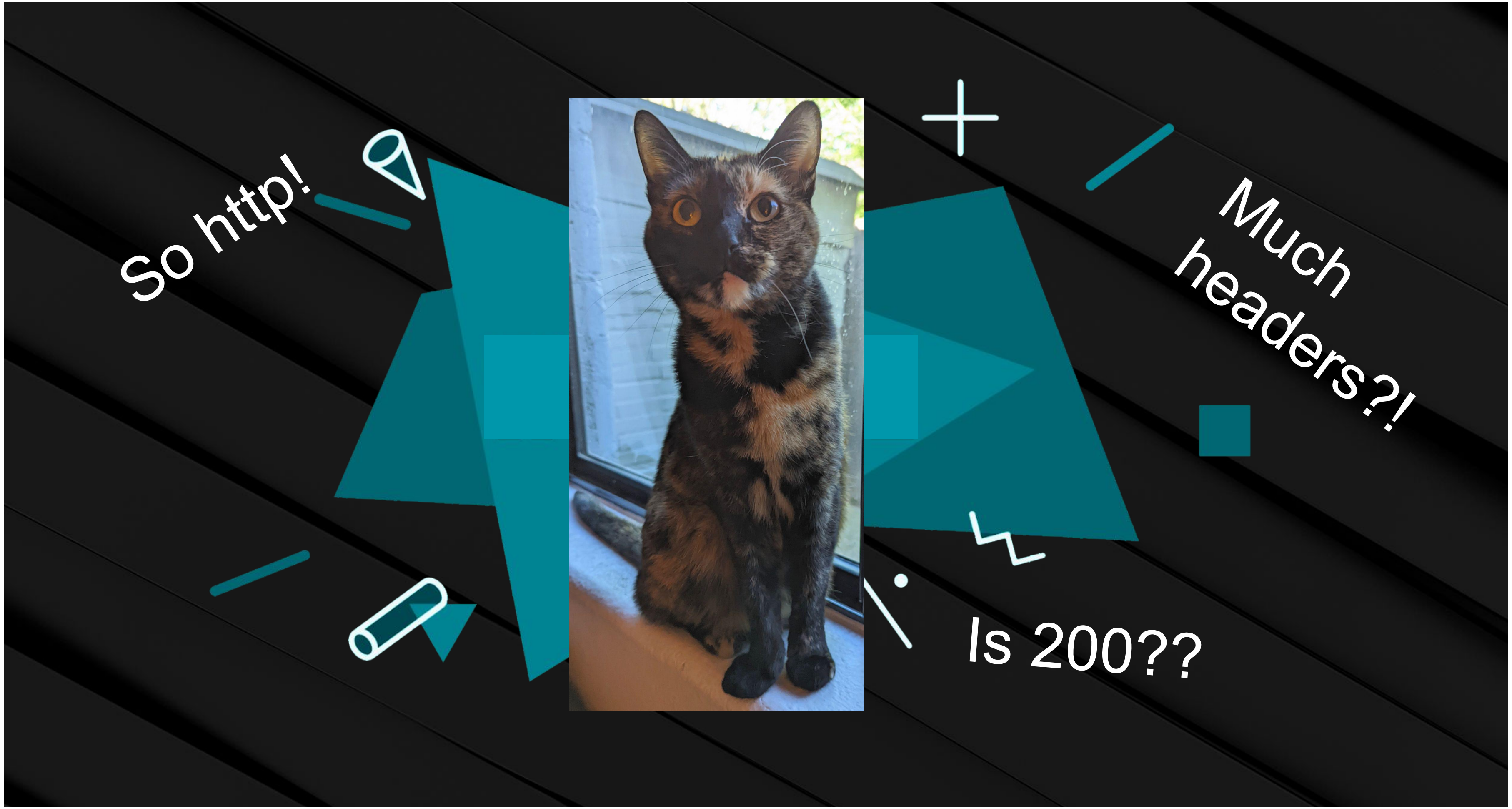
- Common HTTP status codes:
  - 403
    - Forbidden
    - Usually when you already are authenticated, this means you do not have access to the resource
  - 404
    - Most common, not found!
    - Endpoint is usually wrong
  - 405
    - Method not allowed
    - Review the documentation to ensure you are using the correct method

HTTP - The Language of the Web!

- Common HTTP status codes:
  - 403
    - Forbidden
    - Usually when you already are authenticated, this means you do not have access to the resource
  - 404
    - Most common, not found!
    - Endpoint is usually wrong
  - 405
    - Method not allowed
    - Review the documentation to ensure you are using the correct method

HTTP - The Language of the Web!

- Common HTTP status codes:
  - 500
    - Something happened on the server-side that caused the request or response to fail
    - Unless you wrote the server-side code then there’s nothing you can do, but report it to the organization that owns the API
  - 503
    - The service is down/unavailable
    - Proceed to cry in the corner


So http!
Much headers?!
Is 200??