Handling Faults and Errors

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

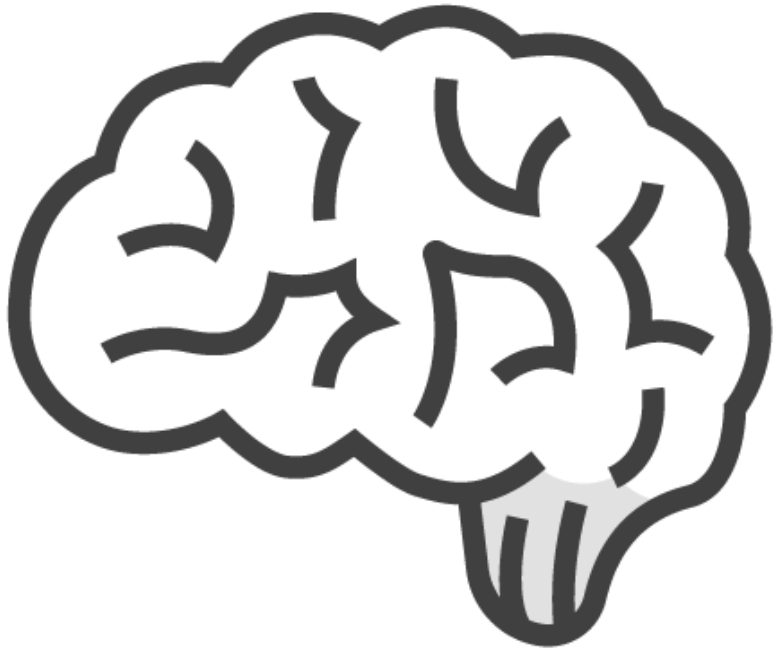
Coming Up

Inspecting Status Codes

Reading out Response Bodies on Error

Dealing with All-but-best-practice APIs

Inspecting Status Codes



EnsureSuccessStatusCode() throws an HttpRequestException on all but 2xx-level status codes

- Depending on the actual status code we want to act differrently

The Importance of Status Codes

Level 200 Success

200 – OK

201 – Created

204 – No Content

Level 400 Client Error

400 – Bad Request

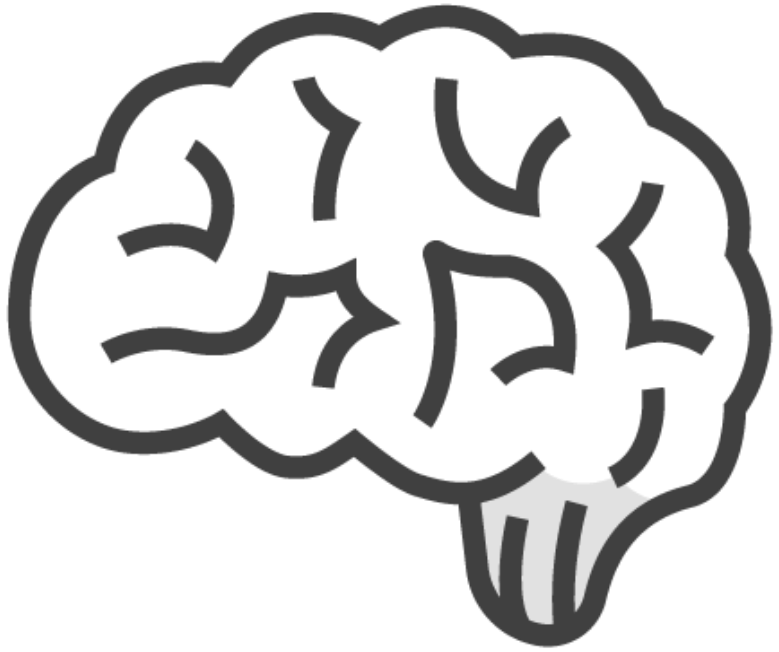
401 – Unauthorized

403 – Forbidden

404 – Not Found

422 – Unprocessable Entity

Inspecting Status Codes



Level 400 issues are errors: the API correctly rejects the request

The Importance of Status Codes

Level 200 Success

200 – OK

201 – Created

204 – No Content

Level 400 Client Error

400 – Bad Request

401 – Unauthorized

403 – Forbidden

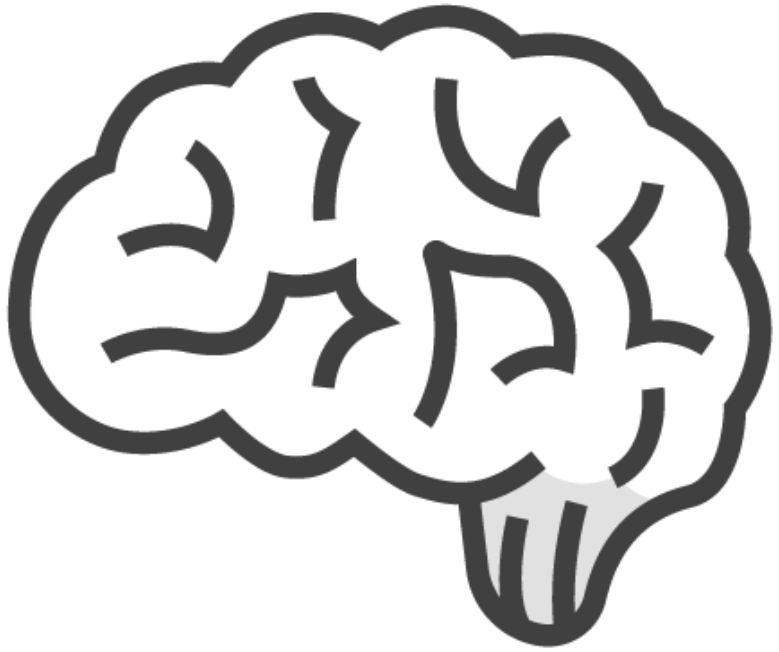
404 – Not Found

422 – Unprocessable Entity

Level 500 Server Error

500 – Internal Server Error

Inspecting Status Codes

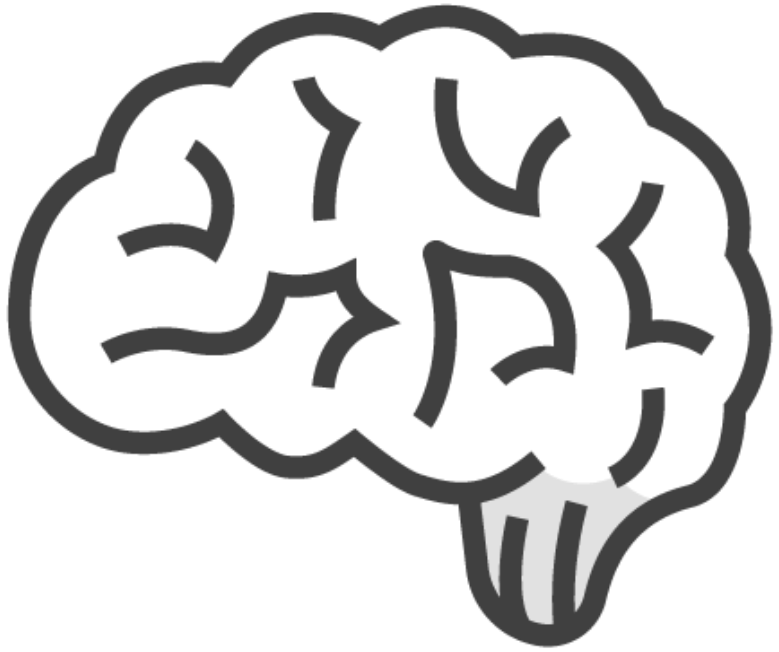


Level 500 issues are faults: the API fails to correctly return a response to a valid request

Demo

Inspecting Status Codes

Inspecting Response Messages



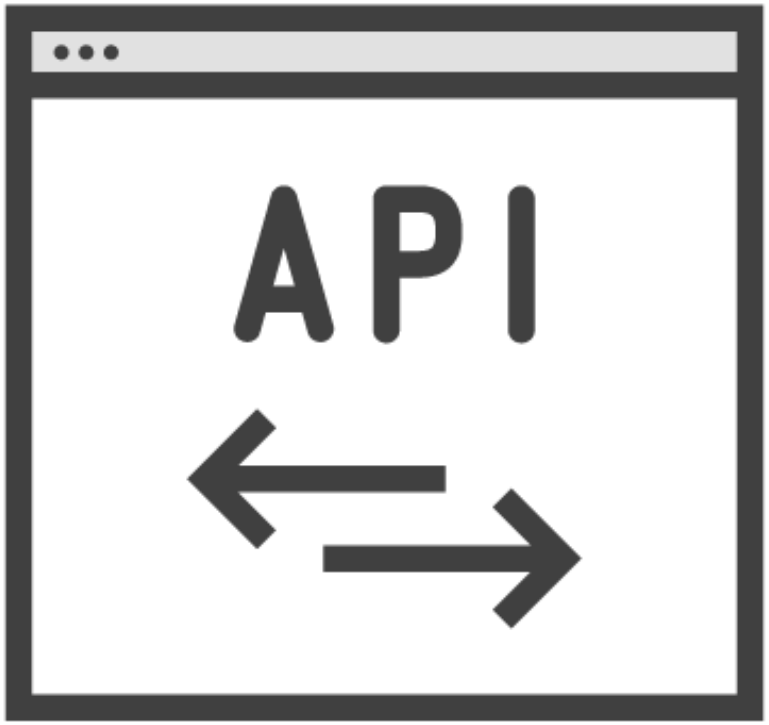
When an error happens, APIs can return additional information on that error in the response body

- Error messages
- Validation errors

Demo

Reading out the Response Body When Streaming

Dealing with All-but-best-practice APIs



Not all APIs correctly use status codes

- Some aren't specific enough
- Some just return 200 OK for everything...

Learn what the API supports and combine reading out status codes & inspecting response messages to deal with this

Summary

Status codes tell us

- Whether a request was successful
- If it wasn't, who made the mistake
- EnsureSuccessStatusCode() isn't fine-grained enough

A response body can contain additional information that can be useful for the client. Read it out using streams.