

Protecting, Testing, and Documenting Our API



Kevin Dockx

ARCHITECT

@KevinDockx <https://www.kevindockx.com>



Coming Up



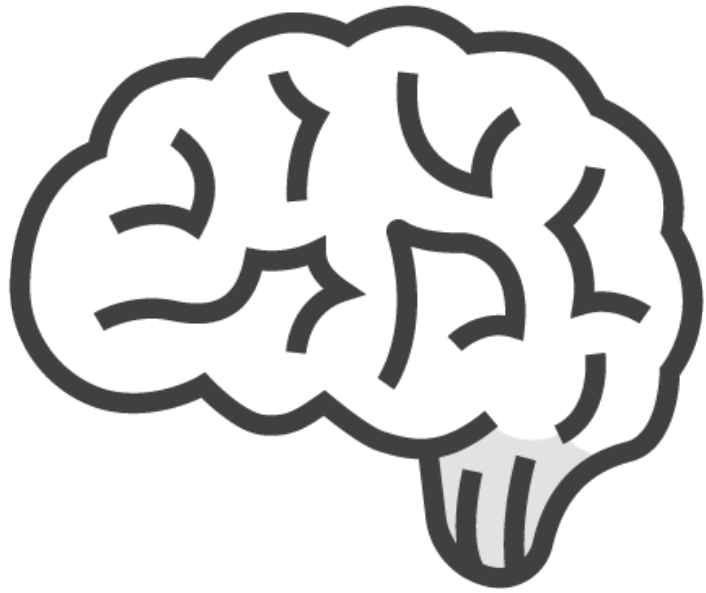
Rate limiting and throttling

Testing our API

Documenting our API

Consumer-level testing



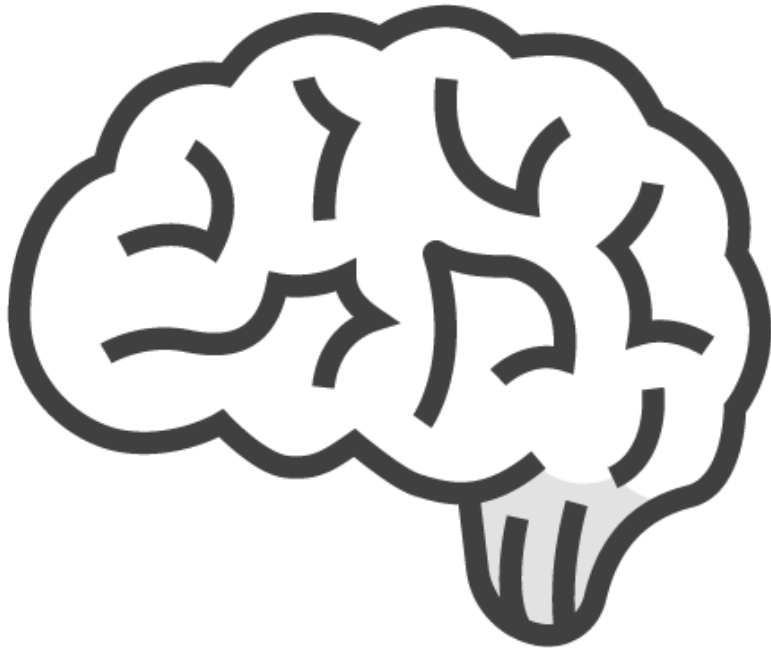


Rate Limiting

Limits the amount of allowed requests to an API (or a specific resource)



Rate Limiting and Throttling



A throttle controls the rate of allowed requests

Rate limited to 100 requests/hour per client

Rate limited to 1000 request/day per IP, and 100 request/hour

Rate limited to 1000 request/day per IP, and 50 to api/authors

Rate Limiting and Throttling

Allowed requests

Should contain rate limiting information in the response headers

X-Rate-Limit-Limit
rate limit period

X-Rate-Limit-Remaining
number of remaining requests

X-Rate-Limit-Reset
date/time information on when the limits are reset

Disallowed requests

Warrant a 429 – Too many requests status code

May include a Retry-After response header

Response body should contain details explaining the condition



Demo



Rate Limiting and Throttling



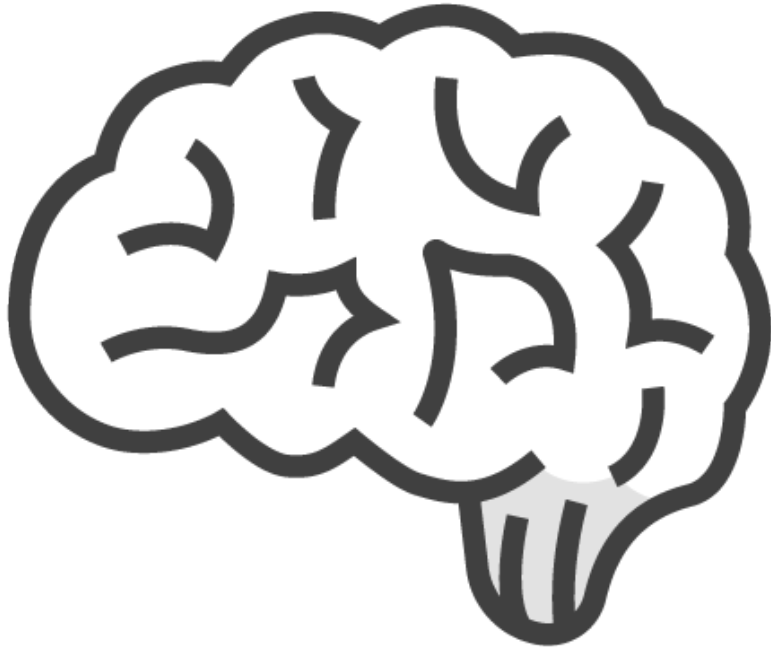
Demo



Testing our API with Postman



Automated Testing and Continuous Integration



When checking code in, the code should be verified by an automated build & tests should be run

Newman is a command-line collection runner for Postman

Allows easy integration with Jenkins, TFS, ... for automated testing



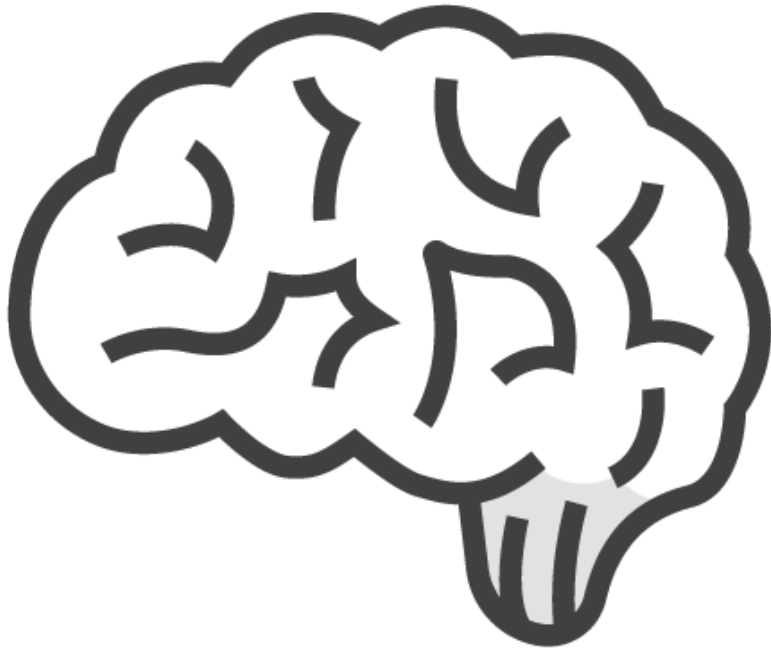

```
newman -c mycollection.postman_collection --exitCode 1
```

Automated Testing and Continuous Integration

Add a build step calling into Newman to run a collection of tests



Documenting Our API



The purpose is to provide documentation on how to consume the API

This type of documentation thus serves a different purpose than the technical design containing the outer-facing contract description!



Documenting Our API



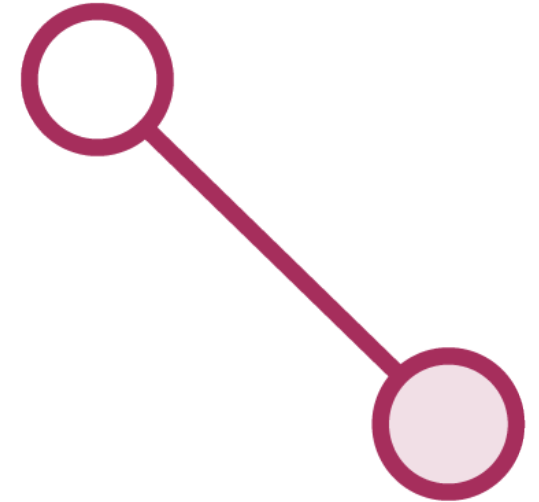
Resource
identifier



HTTP method



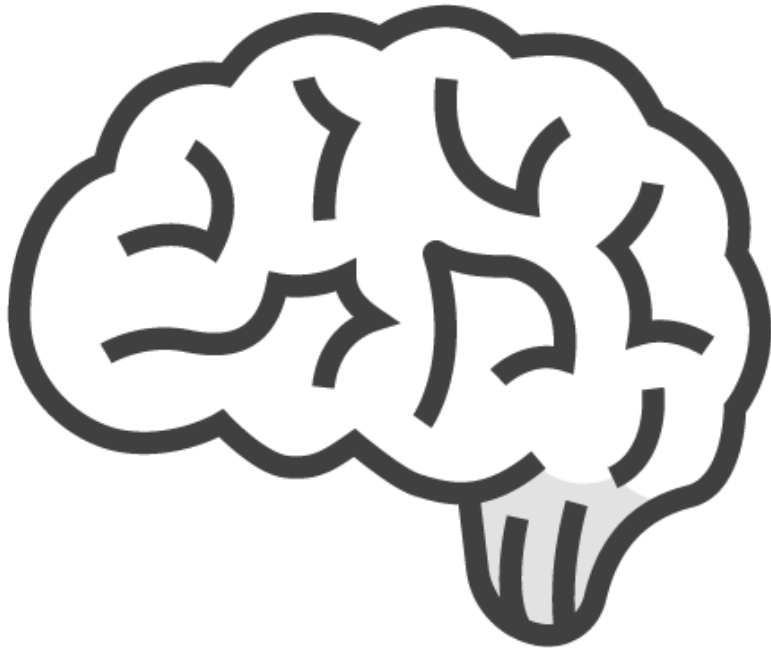
Payload
(+ media types)



Interactive UI



Documenting Our API



Swagger OpenAPI (<http://bit.ly/2joQ3y1>)

Definition format to describe RESTful APIs (currently at v2)

Swashbuckle (<http://bit.ly/1R9flui>)

Currently in prerelease (1.0.0-rc1)

No support for action overloading

Planned for v3



In ConfigureServices (Startup class)

```
services.AddSwaggerGen(setupAction =>
{
    setupAction.SwaggerDoc("libraryapi",
        new Info { Title = "Library API Documentation" });
});
```

Using Swashbuckle

Register services



In Configure (Startup class)

```
app.UseMvc();
```

```
app.UseSwagger();
```

Using Swashbuckle

Add Swagger middleware to the request pipeline

swagger.json available at `swagger/libraryapi/swagger.json`



In Configure (Startup class)

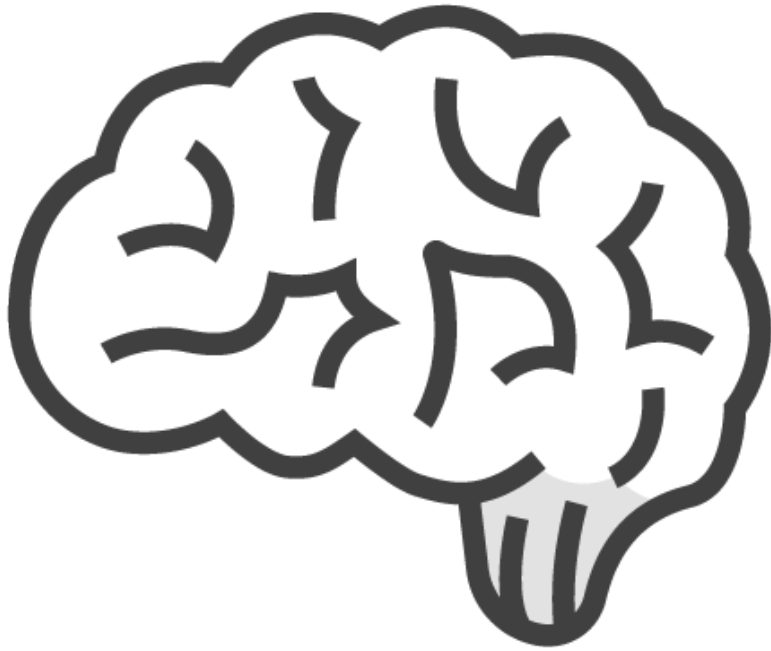
```
app.UseSwaggerUi(c =>
{
    c.SwaggerEndpoint("/swagger/lib/swagger.json", "Library API");
})
```

Using Swashbuckle

Add SwaggerUi middleware to the request pipeline



Working with OPTIONS



Can be used by consumers of the API to
Determine the options and/or
requirements associated with a resource
Determine the capabilities of a server,
without implying a resource action or
initiating a resource retrieval

OPTIONS api/authors

200 OK

Allow: GET, OPTIONS, POST

Working with OPTIONS

Works on resource level

Should return an Allow header



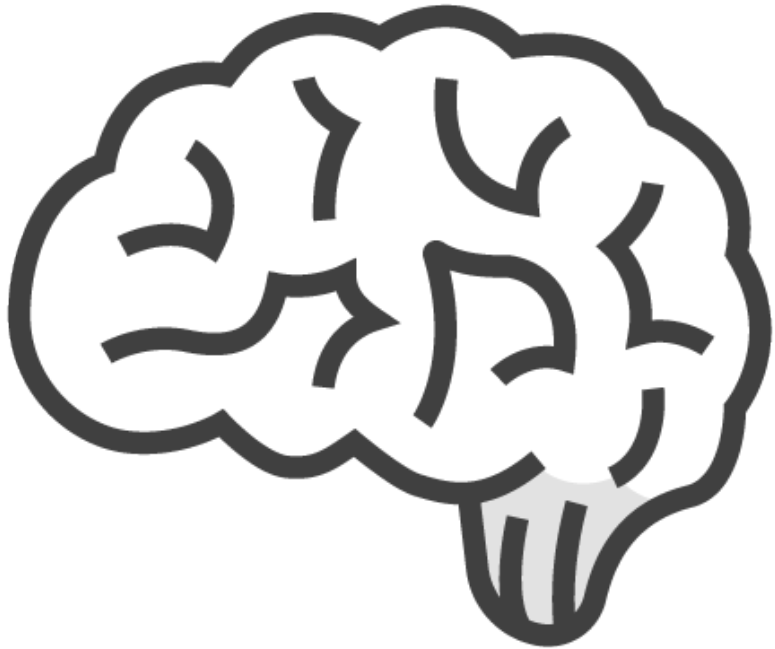
Demo



Supporting the OPTIONS Method



Working with HEAD



HEAD is identical to GET, but without a response body

Can be used by consumers of an API to test for validity, accessibility, and recent modification

Demo



Supporting the HEAD Method



Summary



Rate limiting limits the amount of allowed requests to an API (or a specific resource)

Postman can be used to write tests against an API

Use Newman for CI

Good documentation includes

Resource identifiers

HTTP methods

Payload information (media types)



Summary



Use **OPTIONS** to determine the options and/or requirements associated with a resource, or the capabilities of a server

Use **HEAD** to test for validity, accessibility, and recent modification

You're ready to be AWESOME



@KevinDockx

