API First Design

With OpenAPI

Craig Norman
cnorman@gasbuddy.com

PRAIRIE DEV CON
WEB | DEV | CLOUD | AI

Diamond Sponsors

GasBuddy

SASKATCHEWAN POLYTECHNIC

Gold Sponsors

SOLVERA
crafting results

iQmetrix

Outline

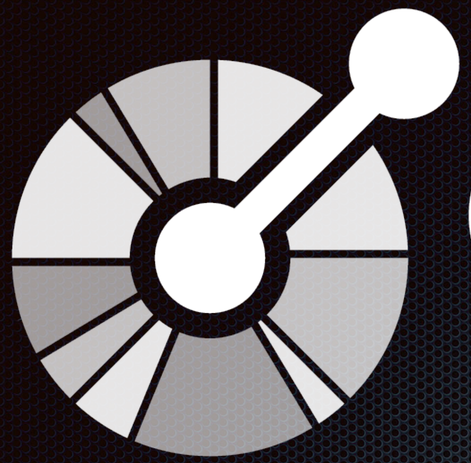
- What is OpenAPI (aka. Swagger)
- About REST
- Dive into the OpenAPI specification
- Examples writing simple calls

Resources

- https://github.com/normrider2001/OpenAPI3.0_prdc
- https://swagger.io/specification/
- https://editor.swagger.io

Why OpenAPI (OAS)?

- Standard way to document your api
- You can create clients and server stubs for many languages
- Rich set of UIs for people to use your API



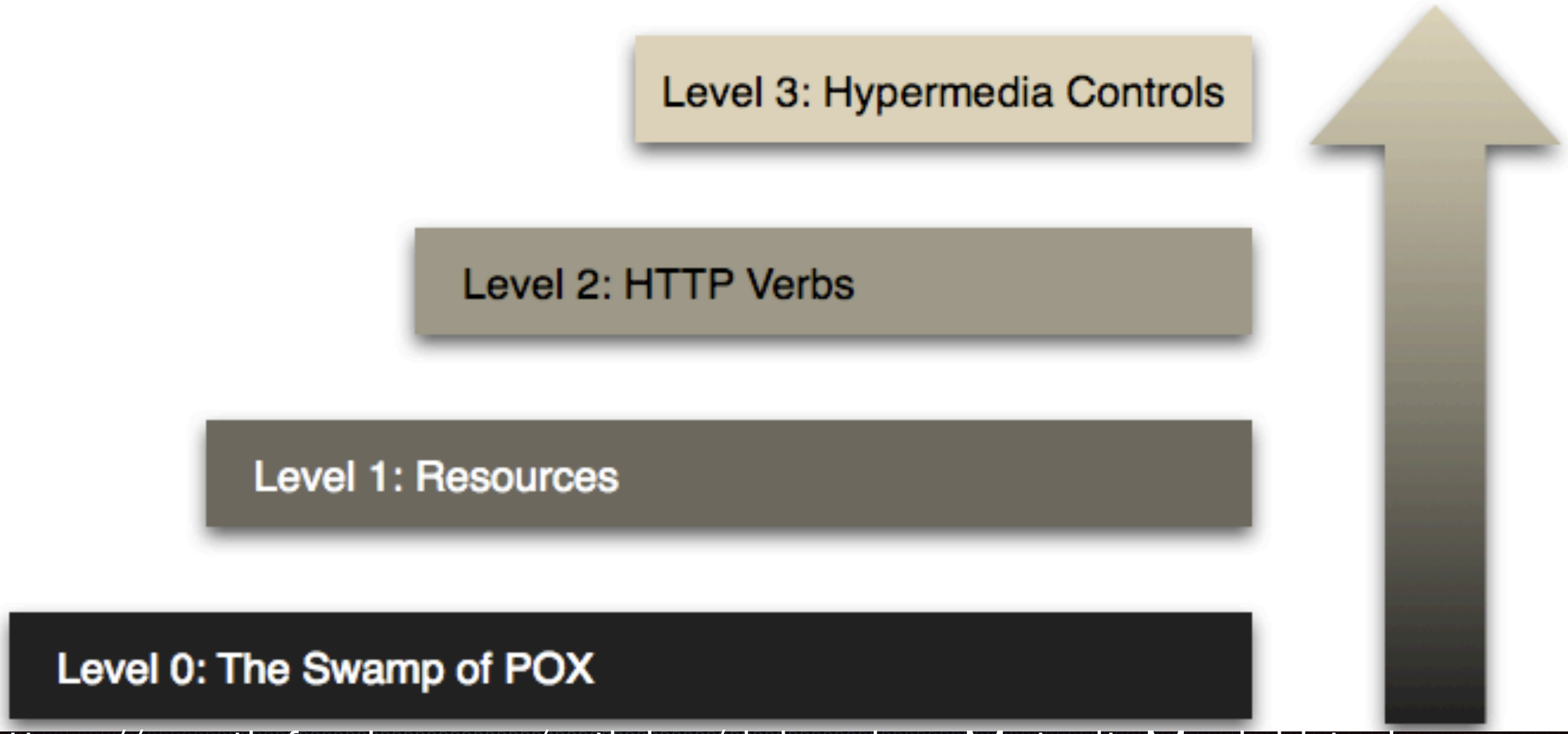
OPENAPI INITIATIVE

- Can be declared in JSON or YAML
- Fixed Fields or Patterned Fields

Why REST?

- Standard way of making an API
- Using HTTP Verbs makes your API cleaner
- Status Codes are good too

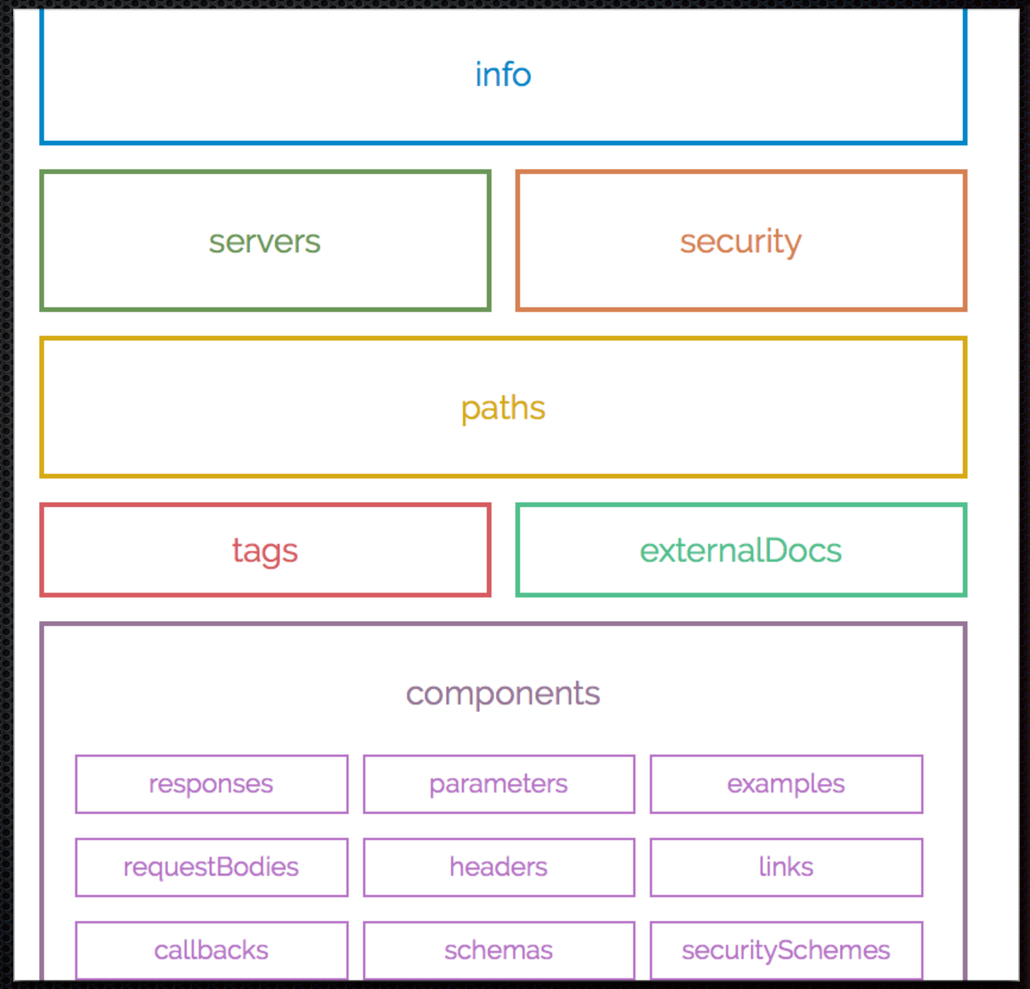
Glory of REST

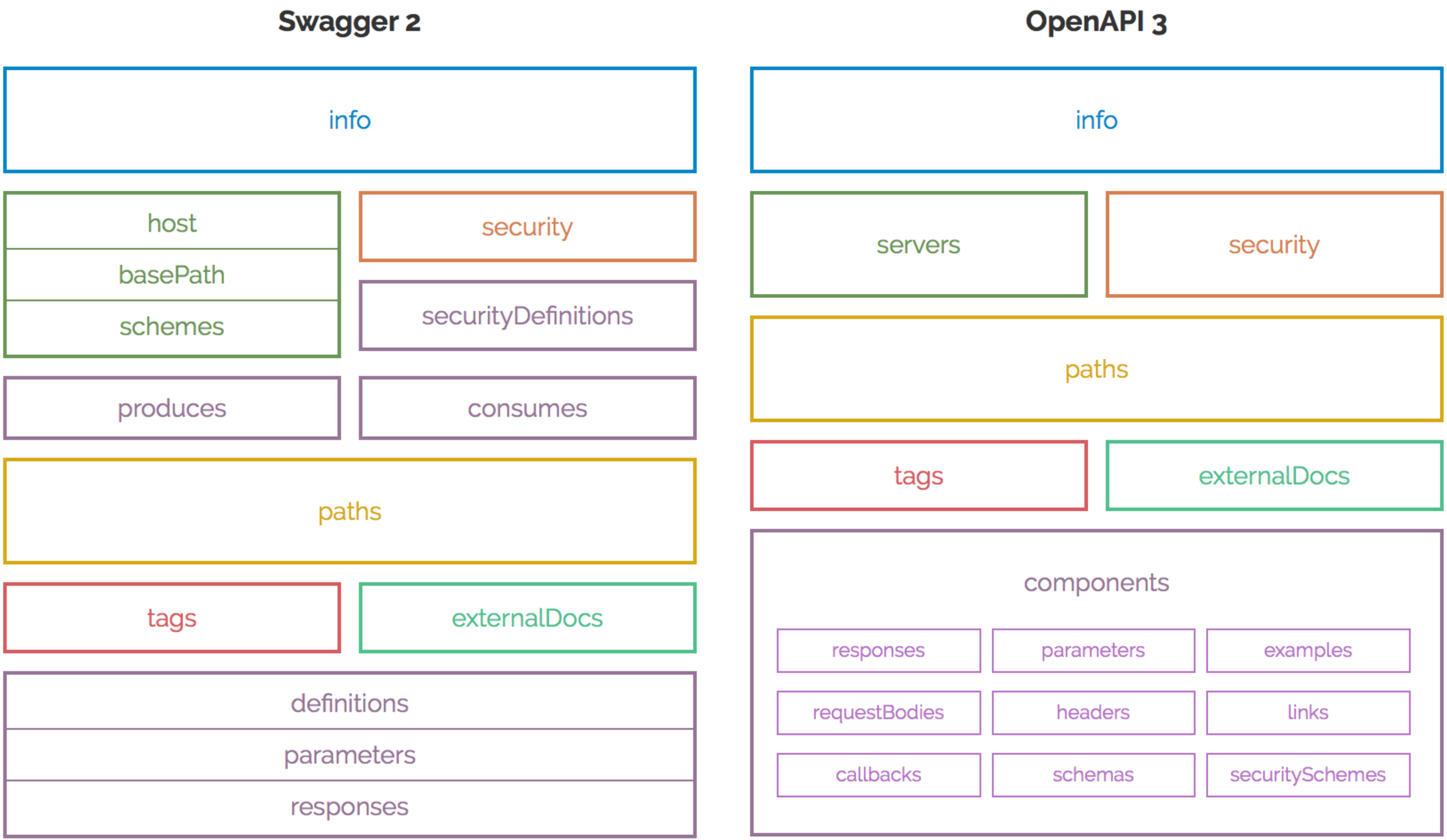


https://martinfowler.com/articles/richardsonMaturityModel.html

Open API 3.0

- info
- paths
  - components:
    - responses, parameters, requestBodies, schemas, links





https://blog.readme.io/an-example-filled-guide-to-swagger-3-2/

Basic Top Level Spec

```yaml
openapi: 3.0.0
info:
  title: Prairie Dev Con Example
  description: This is an example openapi spec to support Prairie Dev
Con talk.
  version: 1.0.0
paths:
  /pets:
    $ref: './paths/pets.yaml'
  /pets/{petId}:
    $ref: './paths/pet.yaml'
```

GET /pets

- Get a list - good place for a search
- Parameters for querying, filtering, paging and sorting
- Expecting an array or an object that contains an array to be returned

GET /pets

```yaml
get:
  summary: List all pets
  parameters:
  - name: name
    in: query
    required: false
      schema:
        type: string
  responses:
    '200':
      description: An array of pets
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '../components/pets.yaml#/components/schemas/Pet'
```

Parameter objects

- in: path | query | header | cookie
- name
- schema: with type
- description (optional) - CommonMark markdown

Common Name	type	format	Comments
integer	integer	int32	signed 32 bits
long	integer	int64	signed 64 bits
float	number	float	
double	number	double	
string	string		
byte	string	byte	base64 encoded characters
binary	string	binary	any sequence of octets
boolean	boolean		
date	string	date	As defined by full-date - RFC3339
dateTime	string	date-time	As defined by date-time - RFC3339
password	string	password	A hint to UIs to obscure input.

https://swagger.io/specification/

Response Object

- Status Code - patterned field 1xx | 2xx | 3xx | 4xx | 5xx
- description
- content

Common Status Code Levels

- 200s - Normal intended behaviour
- 400s - Client needs to change something
- 500s - Server side errors

POST /pets

- Create this type of resource
- The resource to create will be in the body
- Where do find it after

POST /pets

```yaml
post:
  summary: Create a pet
  requestBody:
    description: The body of the pet to create
    content:
      application/json:
        schema:
          $ref: '../components/pets.yaml#/components/schemas/Pet'
  responses:
    '201':
      description: Pet was created
      content:
        applicaton/json:
         schema:
           type: object
           properties:
             id:
               type: integer
               format: int64
               description: Id of the created pet
```

requestBody Object

- New for version 3.0
- content
  - mime type: ie. (application/json | application/xml | text/plain)
    - schema
      - $ref

Links

```yaml
links:
  GetPetByPetId:
    operationId: getPetById
    parameters:
      petId: '$response.body#/id'
    description: >
     The `id` value returned in the response can be used as the
     `petId` parameter in `GET /pets/{petId}`.
```

Responses

Code	Description	Links

201	***Pet was created***
	
	applicaton/json
	Controls Accept header.
	
	Example Value | Model
	
	`{ "id": 0 }`
		GetPetByPetId
		
		The `id` value returned in the response can be used as the `petId` parameter in `GET /pets/{petId}`.
		
		`Operation `getPetById``
		
		`Parameters { "petId": "$response.body#/id" }`

GET pets/{petId}

- Get a resource by an identifier
- Typically return a 200 and the resource if found
- 404 if not found
- This should not change state

Path Templating

- Using {} curly braces to indicate a parameter in a path
- The name of the parameter must match the name in the path

/pets/{petId}

```yaml
parameters:
  - name: petId
    in: path
      schema:
        type: integer
        format: int64
    required: true
```

GET pets/{petId}

```yaml
get:
  operationId: getPetById
  summary: Get specific pet
   responses:
    '200':
      description: A single pet
      content:
        application/json:
          schema:
            $ref: '../components/pets.yaml#/components/schemas/Pet'
    '404':
       description: Not Found
```

OperationId

- VERB_PATH_PATH (ie. Get_Pet_PetId)
- Will be used as function names by code generators
- Must be unique in the spec

PUT pets/{petId}

- Update this specific resource
- Alternatively can be used to create a specific resource at this location
- Typically these are considered idempotent, but do change state

PUT pets/{petId}

```yaml
put:
  summary: Update or create a specific pet
  requestBody:
    description: The body of the pet to create
    content:
      application/json:
        schema:
          $ref: '#../components/pets.yaml#/components/schemas/PetType'
   responses:
    '200':
      description: Pet Updated
```

DELETE /pets/{petId}

- Delete this specific resource at this specific location
- Might be a good idea to return 204 - No Content
- 404 if the object can’t be found

DELETE /pets/{petId}

```yaml
delete:
  summary: Delete the pet
   responses:
    '204':
      description: Deleted entity
    '404':
      description: No pet found
```

HEAD /pets/{petId}

- Does this resource exist?
- HEAD will only send a header back not the response

PATCH

- Usually a more complicated update operation
- Best described by the JSON Patch
- /pets/{petId}
  - op: remove, path: /collar, value: red
  - op: add, path: /vaccination, value: rabies

Component Object

- Holds sets of resuable objects
  - Parameters
  - Responses, Request Bodies
  - Examples, schemas

Schema Object

- Definition of input and output data types
- Version 3.0 makes some interesting features work including anyOf, oneOf

Schema

```yaml
components:
  schemas:
    Pet:
      type: object
      discriminator:
        propertyName: petType
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
        petType:
          type: string
      required:
      - name
      - petType
```

OneOf

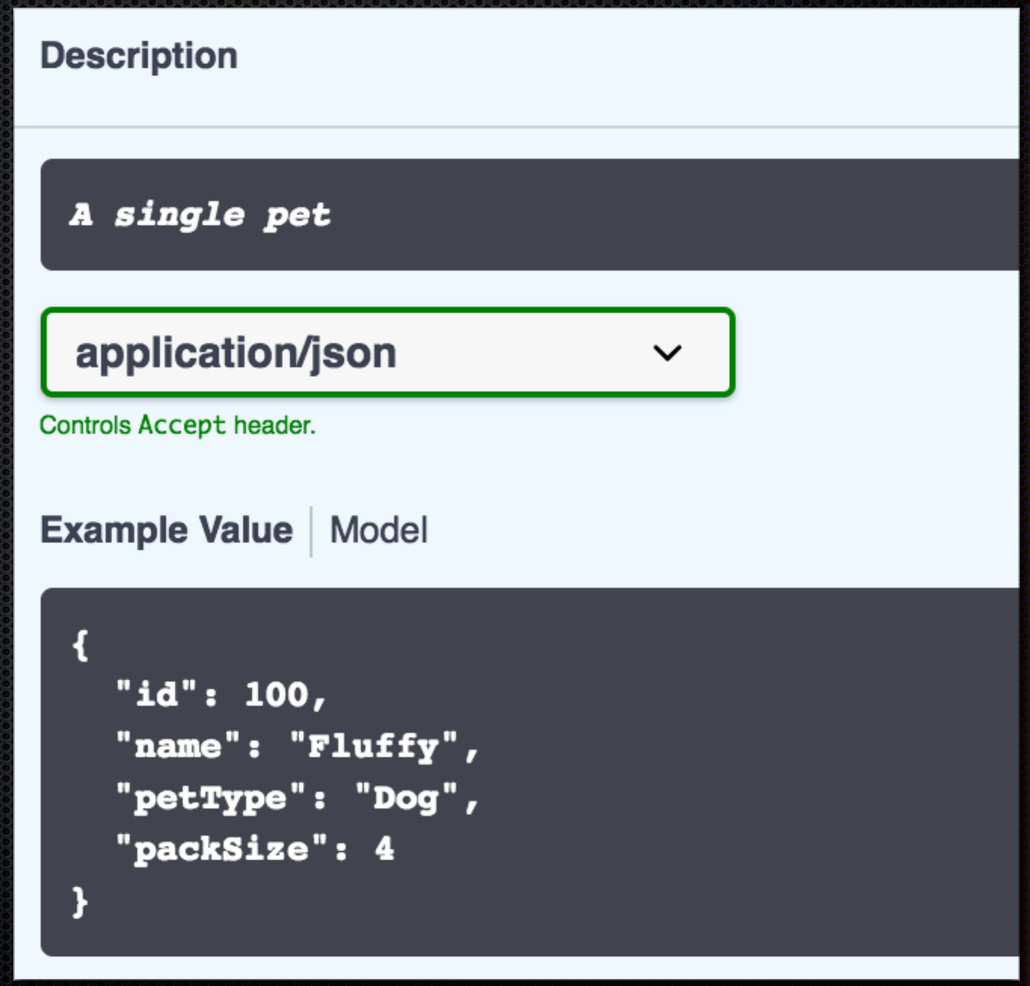
```yaml
    PetType:
      oneOf:
      - $ref: '#/components/schemas/Cat'
      - $ref: '#/components/schemas/Dog'
      discriminator:
        propertyName: petType
```

AllOf

```yaml
Dog:  ## "Dog" will be used as the discriminator value
    description: A representation of a dog
    allOf:
      - $ref: '#/components/schemas/Pet'
      - type: object
        properties:
          packSize:
            type: integer
            format: int32
            description: the size of the pack the dog is from
            default: 0
            minimum: 0
        required:
        - packSize
```

Examples

- Examples simple JSON or YAML can be provided in a value field



Supporting Tools

- Swagger Editor - https://editor.swagger.io
- https://github.com/OpenAPITools/openapi-generator
- https://github.com/Rebilly/generator-openapi-repo Open API docs with Github Pages