SWAGGER AND YAML

Frank Walsh

BSc IT year 4

AGENDA

- Web APIs
  - “API First” Methodology
  - API Design
  - API Specifications
    - YAML
    - Swagger/YAML
  - Example

SWAGGER

- Swagger provides a specification for creating RESTful API documentation formatted in JSON or YAML.
  - Similar to Web Service Description Language (WSDL)
- YAML: YAML Ain't Markup Language
  - YAML is a human friendly data serialization standard for all programming languages.
- Generates API documentation automatically.

API DESIGN

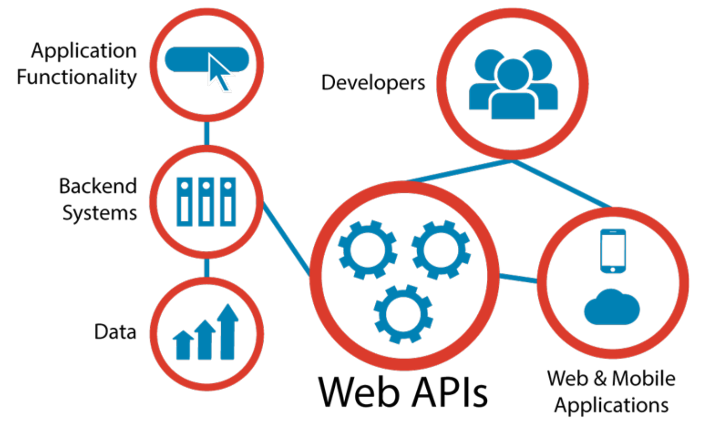
- To consume an API you must
  - Know the routes
  - Know where it is
  - Know the documentation (RTFM)
- To develop an API you must
  - Do the design

SWAGGER BENEFITS

- Interactive documentation.
- Discoverability.
- Client SDK generation.

WEB APIS

- Programmatic interface exposed via the web
- Uses open standards typically with request-response messaging.
  - E.g messages in JSON or XML
  - HTTP as transport
  - URIs
- Example would be Restful web service described in previous lectures.
- Typical use:
  - Expose application functionality via the web
  - Machine to machine communication
  - Distributed systems



WEB APIS

- There's APIs available to do pretty much anything:
  - XAAS, stripe, AWS services, Stripe, Facebook, Twitter
- Can be further abstracted using SDKs for developers
  - You've been using the AWS sdk in your labs, both on server and client side.
- This has led to a very broad and large set of capabilities.
- APIs have moved on from being an "add on" to the central product
- E.g. Stripe, Pubnub, Salesforce.

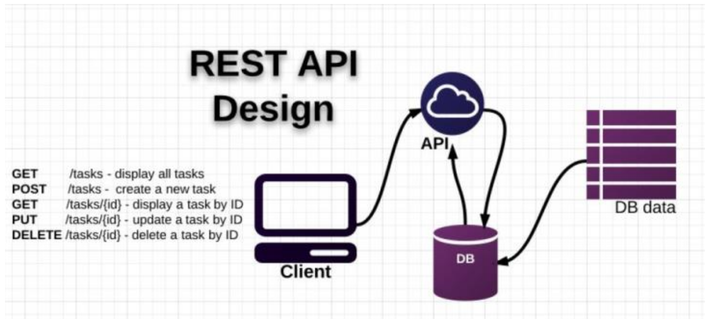
“API FIRST” APPROACH

- Collaboratively design, mockup, implement and document an API **before** the application or other channels that will use it even exist.
- Uses “clean-room” approach.
    - the API is designed with little consideration for the existing IT estate.
    - the API is designed as though there are no constraints.

source: http://www.programmableweb.com/news/introduction-to-api-first-design/analysis/2016/10/31

TRADITIONAL API DESIGN

- API design happens after the release of some a data-rich application
    - Existing application “wrapped” in API
- Created as an afterthought.
    - Tightly bound application needs data/function exposed as API.
    - Shoe-horned in as a separate entity.



```
GET       /tasks - display all tasks
POST      /tasks -  create a new task
GET       /tasks/{id} - display a task by ID
PUT       /tasks/{id} - update a task by ID
DELETE /tasks/{id} - delete a task by ID
```

ADVANTAGES OF API FIRST

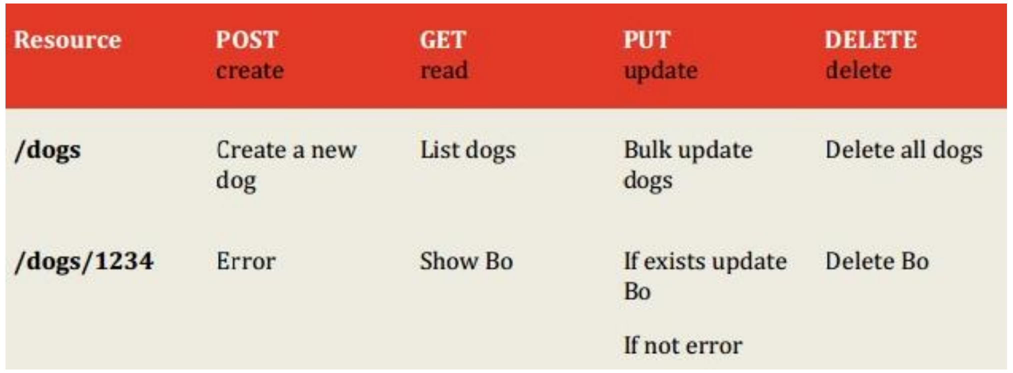
- Suits multi-device environment of today.
- An API layer can serve multiple channels/devices.
  - Mobile/tablet/IoT device
- Scalable, modular, cohesive and composeable
  - If designed properly(e.g. microservice architecture)
  - See later slides
- Concentrate on function first rather than data

API DESIGN

Use	Use principle of developer-first • put target developers' interests ahead of other considerations • Strive for a better developer experience
Commit	Commit to RESTful APIs
Use	Use a Interface Description Language like: • RESTful API Markup Language (RAML) • Swagger
Take	Take a grammatical approach to the functionality
Keep	Keep interface simple and intuitive

API DESIGN

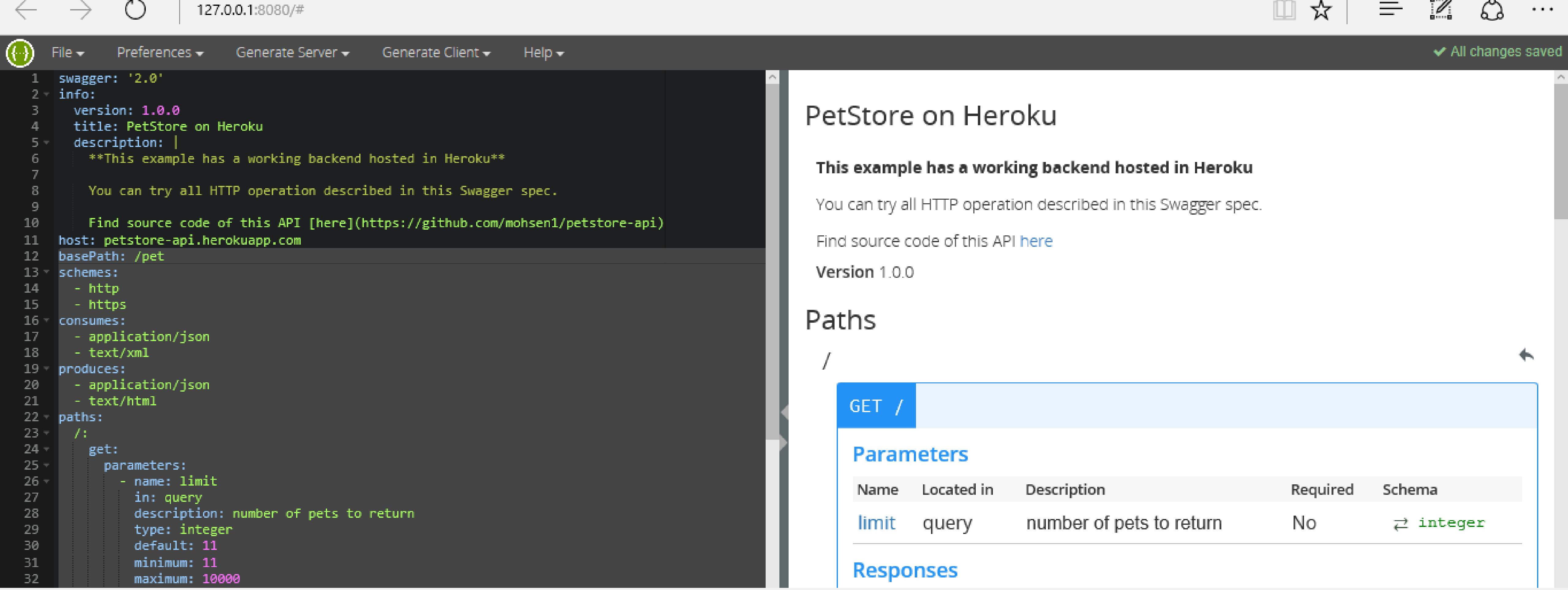
- In Rest, everything is based around resources
  - the "things" you're working with are modelled as resources described by URI paths--like /users, /groups, /dogs
  - Notice they are **nouns .**
  - **Verbs in URLs are BAD**
- The things that you do on these things (or nouns) are characterised by the fixed set of HTTP methods
  - What GET,POST,PUT does is something that the designer/developer gets to put into the model.
- The metadata (the adjectives) is usually encoded in HTTP headers, although sometimes in the payload.
- The responses are the pre-established HTTP status codes and body. (200, 404, 500 etc.)
- The representations of the resource are found inside the body of the request and response



Resource	POST create	GET read	PUT update	DELETE delete
/dogs	Create a new dog	List dogs	Bulk update dogs	Delete all dogs
/dogs/1234	Error	Show Bo	If exists update Bo If not error	Delete Bo

API DESIGN USING SWAGGER

- Swagger provides a specification for creating RESTful API documentation formatted in JSON or YAML.
  - Similar to Web Service Description Language (WSDL)
- YAML: YAML Ain't Markup Language
  - YAML is a human friendly data serialization standard for all programming languages.
- Generates API documentation automatically.



```yaml
swagger: '2.0'
info:
  version: 1.0.0
  title: PetStore on Heroku
  description: |
    **This example has a working backend hosted in Heroku**

    You can try all HTTP operation described in this Swagger spec.

    Find source code of this API [here](https://github.com/mohsen1/petstore-api)
host: petstore-api.herokuapp.com
basePath: /pet
schemes:
  - http
  - https
consumes:
  - application/json
  - text/xml
produces:
  - application/json
  - text/html
paths:
  /:
    get:
      parameters:
        - name: limit
          in: query
          description: number of pets to return
          type: integer
          default: 11
          minimum: 11
          maximum: 10000
```

PetStore on Heroku

This example has a working backend hosted in Heroku

You can try all HTTP operation described in this Swagger spec.

Find source code of this API here

Version 1.0.0

Paths

/

GET /

Parameters

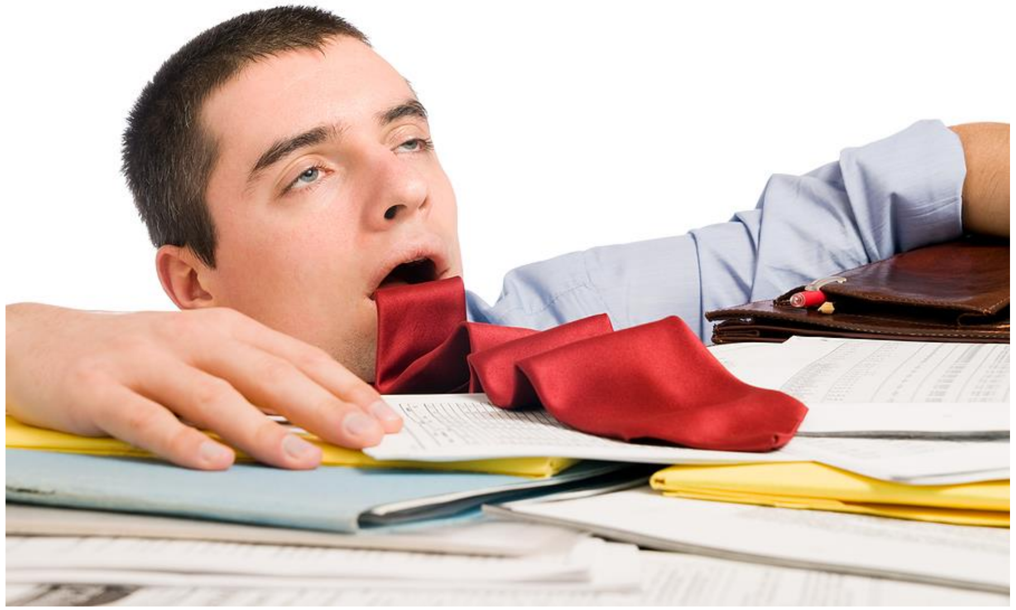
Name	Located in	Description	Required	Schema
limit	query	number of pets to return	No	⇄ integer

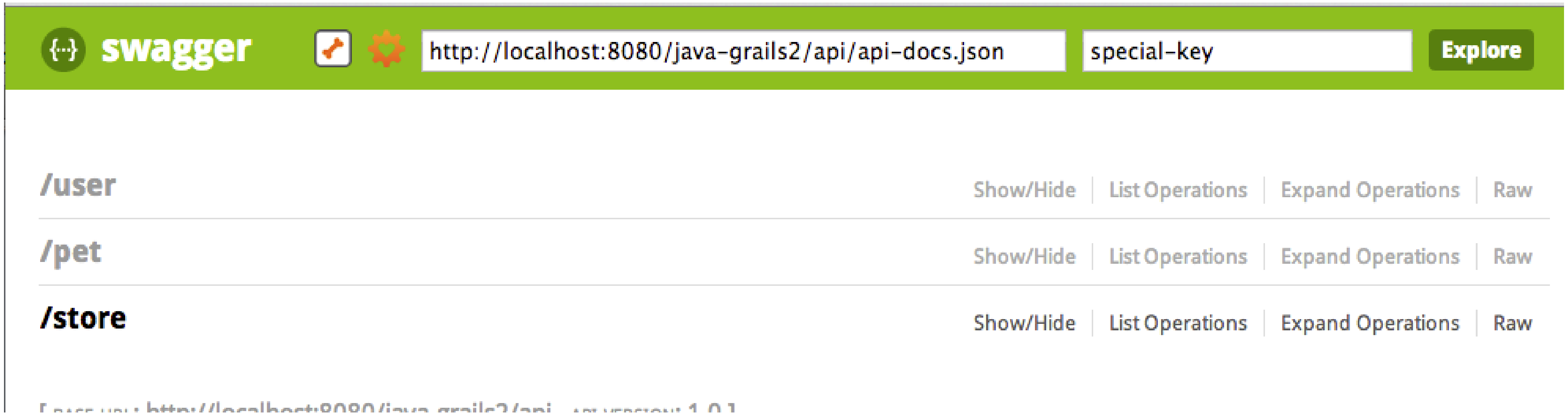
Responses

SWAGGER EXAMPLE IN CLASS

DOCUMENTATION

- Standards are not set
- Ad hoc approach to specification
- Manually manages sometimes
  - What if API changes?
- Sometimes not up to date




swagger

http://localhost:8080/java-grails2/api/api-docs.json

special-key

Explore

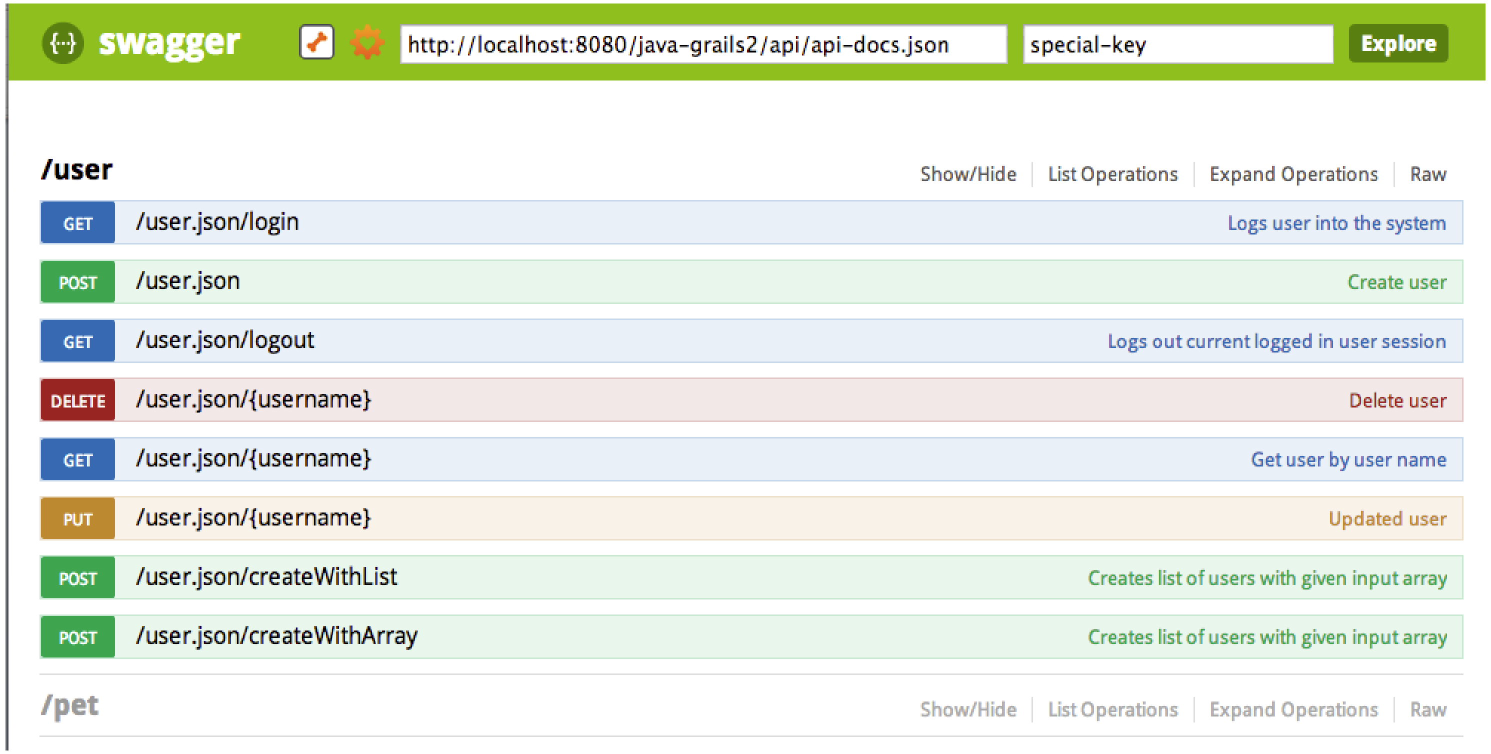
/user Show/Hide | List Operations | Expand Operations | Raw

/pet Show/Hide | List Operations | Expand Operations | Raw

/store Show/Hide | List Operations | Expand Operations | Raw

[BASE URL: http://localhost:8080/java-grails2/api , API VERSION: 1.0]

API LISTING



http://localhost:8080/java-grails2/api/api-docs.json

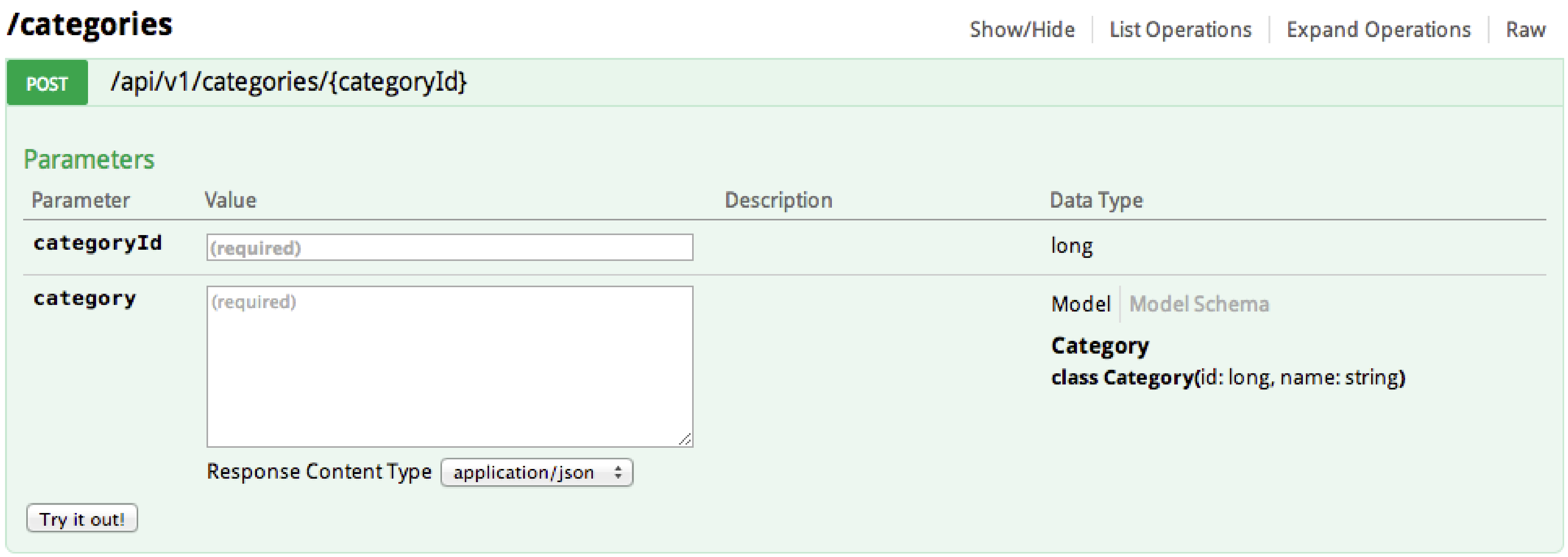
special-key

/user

GET	/user.json/login	Logs user into the system
POST	/user.json	Create user
GET	/user.json/logout	Logs out current logged in user session
DELETE	/user.json/{username}	Delete user
GET	/user.json/{username}	Get user by user name
PUT	/user.json/{username}	Updated user
POST	/user.json/createWithList	Creates list of users with given input array
POST	/user.json/createWithArray	Creates list of users with given input array

/pet

API OPERATION


/categories

POST /api/v1/categories/{categoryId}

Parameters

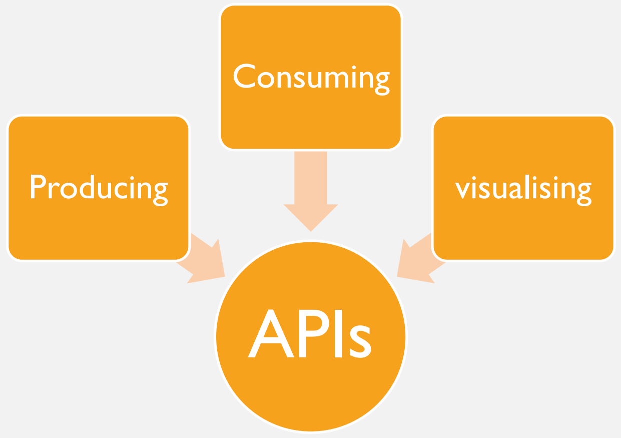
Parameter	Value	Description	Data Type
categoryId	(required)		long
category	(required)		Model Model Schema **Category** **class Category(**id: long, name: string**)**

Response Content Type application/json

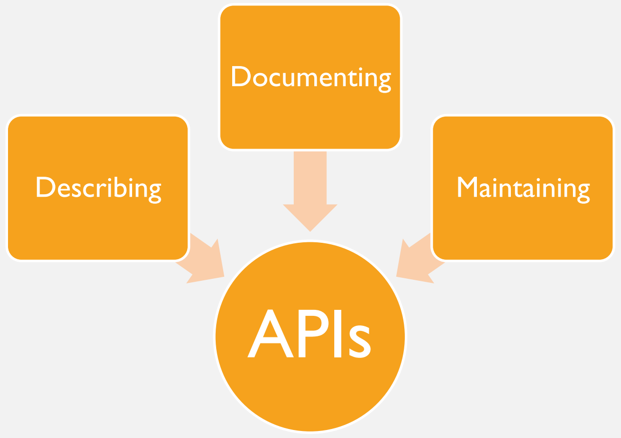
API PARAMETERS AND RETURN TYPES

SWAGGER PROVIDES…

- Technology for…



- Methodology for…


Documenting
Describing
Maintaining
APIs