API and Python training

Session 9

This session’s agenda

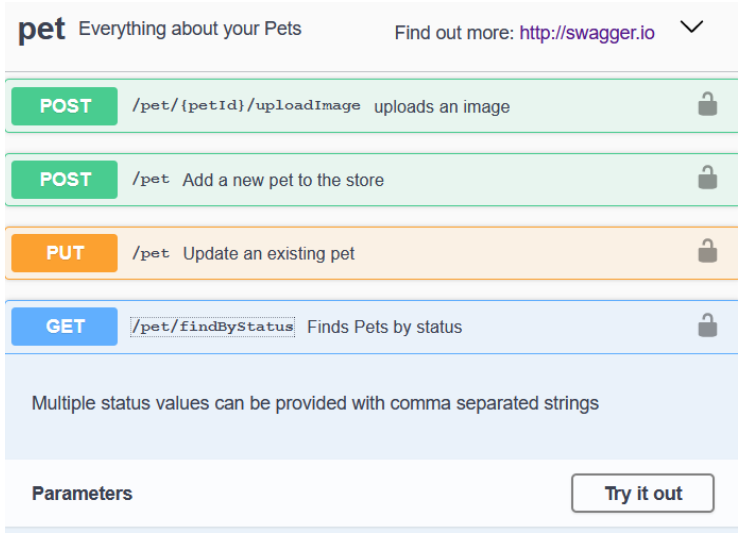
- Open API specification (Swagger)
- Connexion module
- Re-writing API server using Connexion
- Swagger UI console
- Demo
- Training summary

Open API specification (Swagger)

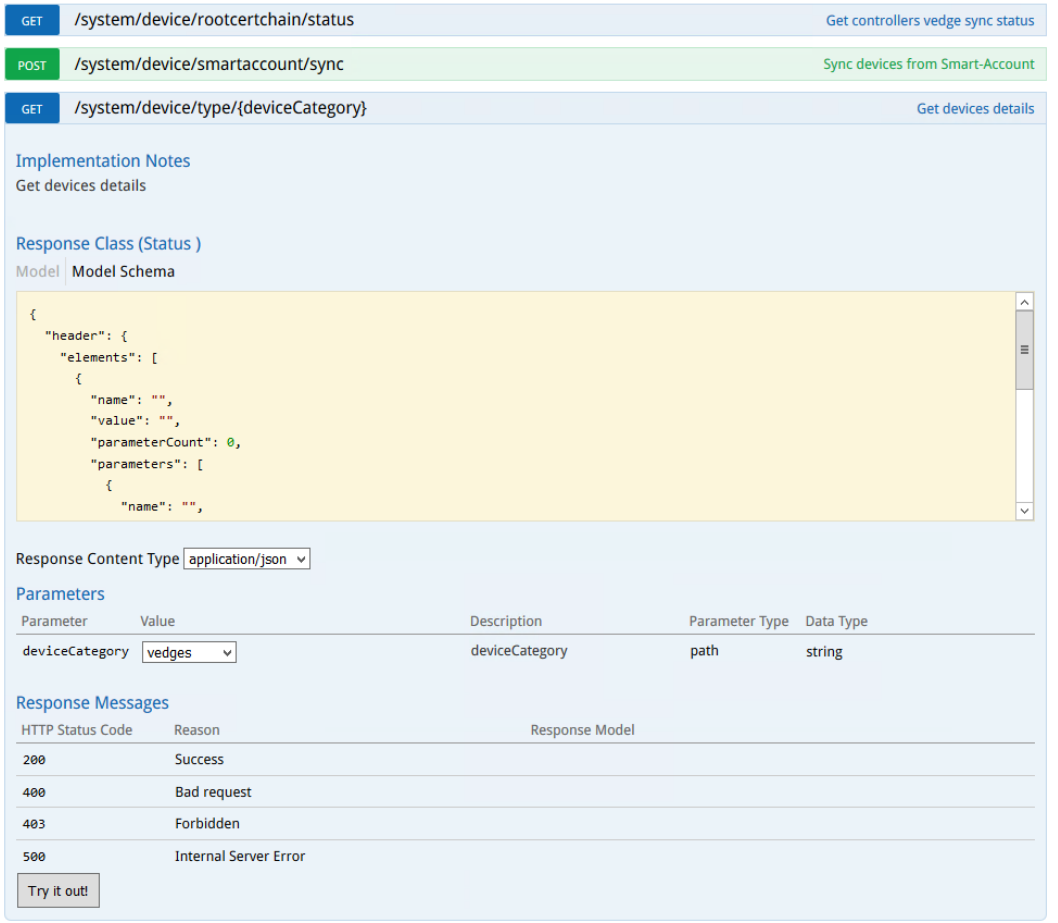
- The OpenAPI Specification (OAS) defines a standard **interface description** for REST API
- Describes API services and is represented in either YAML or JSON formats
- Removes guesswork in calling a service
- Human and machine readable
- Two main versions – 2.0 (also known as Swagger) and newer 3.0 and 3.1
- Can include interactive documentation (Swagger UI)
- Check the doc: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

Examples:

https://petstore.swagger.io/



Cisco vManage controller --->



https://swagger.io/resources/articles/documenting-apis-with-swagger/

Connexion module

- **One of many tools** which can help you documenting REST API
- Connexion is a framework **on top of Flask** (but other servers can also be used)
- Doesn't generate specs based on the code, but you refer your existing Python functions in the spec
- Write an OpenAPI specification in YAML file and **map the endpoints to your Python functions**
- Validates requests and endpoint parameters automatically
- Can handle basic and OAuth 2 token-based authentication
- Provides **Swagger UI console**

install with pip:

pip install connexion

pip install connexion[swagger-ui]

https://connexion.readthedocs.io/en/latest/
https://github.com/zalando/connexion

Re-writing API server

- As it uses Flask, the syntax to run the server and its parameters is very similar (Line 7)
- Define YAML specification file (Line 4)
- You can put this file in a separate directory, add *specification_dir* parameter (Line 3)

```
1  import connexion
2
3  my_api_server = connexion.FlaskApp(__name__, specification_dir='openapi/')
4  my_api_server.add_api('my_api_spec.yaml')
5
6  if __name__ == '__main__':
7      my_api_server.run(port=5000, debug=True, host='127.0.0.1')
```

- In the file with Python functions handling API requests, you don't need to define Flask routes anymore

Note lines 21, 28, 35 – defining routes:

```
21  @my_api_server.route('/api/sensor/<name>')
22  def api_get_sensor_by_name(name):
23      with open('device_data.csv', 'r') as file:
24          data = list(csv.DictReader(file, delimiter=',', quotechar='"'))
25          filtered_data = [item for item in data if item['Device name'] == name]
26          return jsonify(filtered_data)
27
28  @my_api_server.route('/api/sensor/type/<type>')
29  def api_get_sensor_by_type(type):
30      with open('device_data.csv', 'r') as file:
31          data = list(csv.DictReader(file, delimiter=',', quotechar='"'))
32          filtered_data = [item for item in data if item['Device type'] == type]
33          return jsonify(filtered_data)
34
35  @my_api_server.route('/api/sensor/state/<state>')
36  def api_get_sensor_by_status(state):
37      with open('device_data.csv', 'r') as file:
```

No Flask routes, only function definitions:

```
9   def get_api_get_sensor_by_name(sensor_name):
10      with open('device_data.csv', 'r') as file:
11          data = list(csv.DictReader(file, delimiter=',', quotechar='"'))
12          filtered_data = [item for item in data if item['Device name'] == sensor_name]
13          return jsonify(filtered_data)
14
15  def get_api_get_sensor_by_type(type):
16      with open('device_data.csv', 'r') as file:
17          data = list(csv.DictReader(file, delimiter=',', quotechar='"'))
18          filtered_data = [item for item in data if item['Device type'] == type]
19          return jsonify(filtered_data)
20
21  def get_api_get_sensor_by_state(state):
22      with open('device_data.csv', 'r') as file:
```

YAML Specification file – common settings

- The file you defined as API specification file when started API server
- Two main parts – common settings and paths
- Common settings example:

Note basePath is added to all API endpoints,

<server><basePath><endpointPath>

---------->

Next, you define individual paths -------->

```yaml
---
swagger: '2.0'
info:
  description: Swagger configuration file for test REST API server
  version: '1.0'
  title: My REST API server

basePath: '/api/v1'

consumes:
  - 'application/json'
produces:
  - 'application/json'

paths:
```

YAML Specification file - paths

- Line 30 – **path**, variables are in {}
- Line 31 – **operation** (GET, POST, etc)
- Line 32 – operationID – refers to **python function** for the path

  format: <file>.<function_name>

- Line 33 – Tags allow grouping paths in Swagger Console (next slide)
- Lines 37-42 - request parameters

  Note name: (Line 39) - this should match Python function's parameter

- Lines 46-60 – describes response body content

  types:

  array = list

  object = dictionary

  int, string

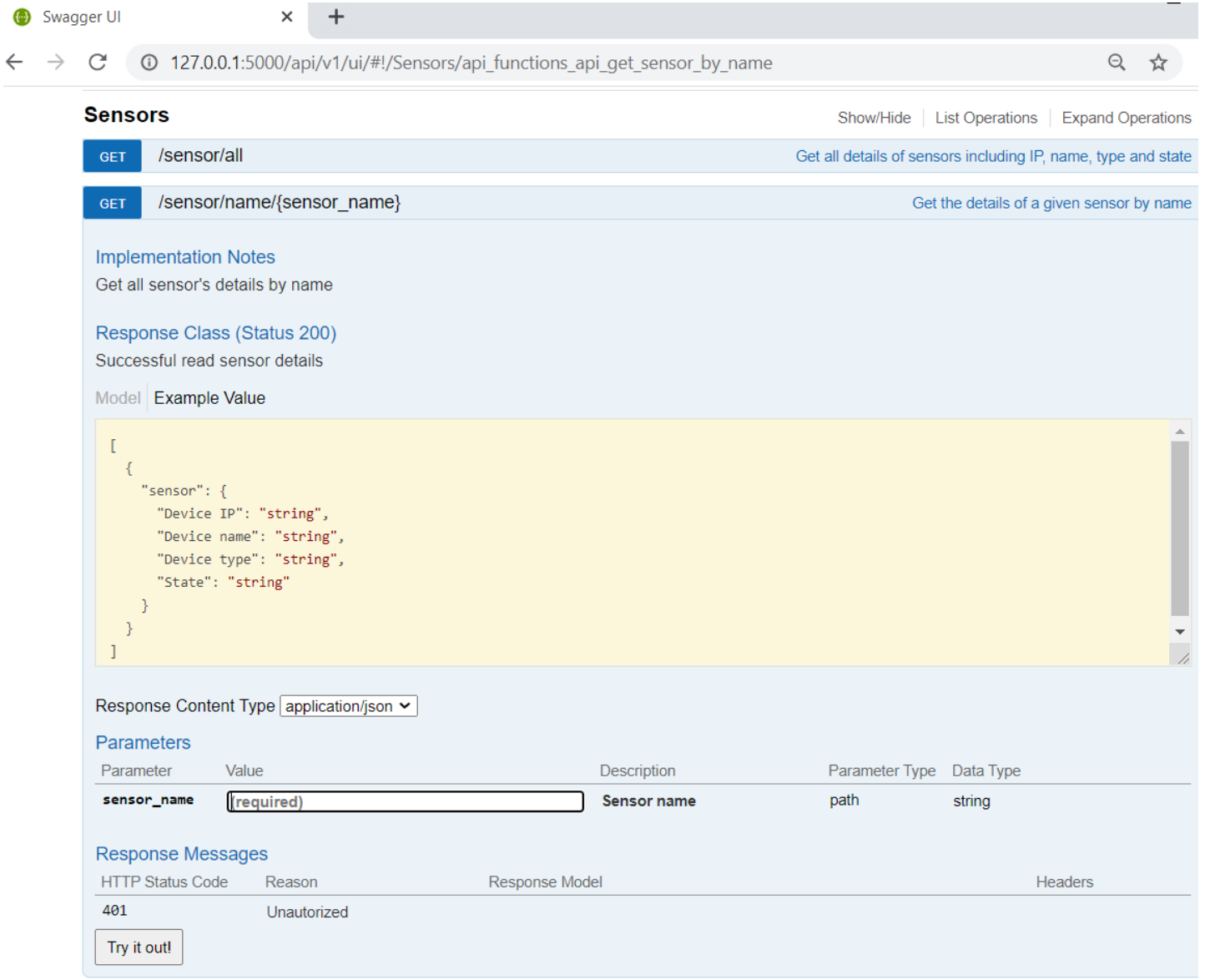
```
30  /sensor/name/{sensor_name}:
31    get:
32      operationId: 'api_functions.api_get_sensor_by_name'
33      tags:
34        - 'Sensors'
35      summary: 'Get the details of a given sensor by name'
36      description: "Get all sensor's details by name"
37      parameters:
38        - in: path
39          name: sensor_name
40          type: string
41          required: true
42          description: Sensor name
43      responses:
44        200:
45          description: 'Successful read sensor details'
46          schema:
47              type: array
48              items:
49                properties:
50                  sensor:
51                    type: object
52                    properties:
53                      Device name:
54                        type: string
55                      Device type:
56                        type: string
57                      Device IP:
58                        type: string
59                      State:
60                        type: string
61        401:
62          description: 'Unautorized'
```

Swagger UI console

- Allows you to validate your API doc
- Allows users to navigate though API
- Allows users to try API requests
- In Connextion use path:

basePath+/ui

Where basePath is defined in YAML file



Demo

Congratulations ! ☺

We've finished the Python and API training !

More than 9 hours of lessons

Session 1

- What is API
- A bit of history
- API examples
- Types of API
- REST API
- How to consume API
- Controllers

Session 2

- API authentication
- Postman

Session 3

- Python IDE
- Python basics
- Simple data types
- Complex data types

Session 4

- Python libraries, import statements
- Requests module
- Accessing JSON data

Session 5

- Python control structures
- Handling exceptions

Session 6

- Python functions

Session 7

- Classes and objects

Session 8

- What is SDK
- Building API server
- Flask module

Session 9

- Swagger
- Connexion module

What we learned

What is API
A bit of history
API use cases and examples
Types of API
REST API
How to consume API
Webhooks
Controllers
REST API Authentication
Basic authentication
Bearer authentication
API Keys
OAuth
Cookies
Custom headers
Combination of methods
Practice – API authentication with cURL
Postman interface
Main features
Postman practice
Why Python
IDE - Integrated Development Environment
Python virtual environments
Variables
Simple data types – integer, float, string, bool
Data types conversion, output, f-strings
Complex data types – dictionaries and lists
Import packages
Python requests library
Making API requests
JSON
Control structures
If-else, if – elif -else
While
For
Using control structures for handling API responses
Handling exceptions
Recap from last session – nested data structures
Accessing JSON data
Python functions
Functions in programming languages
Functions in Python
Function arguments
Position vs named arguments
Variable scope
Return values
Classes
Class instances – Objects
Class properties/attributes
Class methods
OOP – Object-oriented programming
SDKs – Software Development Kits
Using SDKs in Python
SDK examples
Building simple API server
Recap from Session 1: Controllers
Northbound and Southbound APIs
Intro to Flask
Defining routes
Returning JSON data
Reading data from files
Filtering data - List comprehension
Open API specification
Connexion module

Please provide your feedback

- Was the training useful for you?
- What you liked and (most importantly) what you didn't?
- What content would you add or remove?
- Any specific topics you found boring and interesting?
- Were there enough details and examples?
- Would you prefer more practical exercises?
- Would you prefer shorter session, or more often (2 times a week) ?
- Would you be interested in further trainings, what topics?

- Let's connect in LinkedIn:

https://www.linkedin.com/in/alexander-zyuzin/

Thank you for participating !

Stay curious and keep learning ☺