

## Exercise 2: OAS Basics

---

For this exercise, you'll use the Swagger editor to document some simple requests. Let's say you were writing an OAS file to define an API for a music service. Let's define two requests: one to retrieve a list of playlists, and another to delete a playlist.

To start with, you can refer to the OAS file I created for the photo album requests. Note that the response sections are very basic, just saying that they return 200 and a description that indicates that it's a successful response. Also, remember that lines that start with # are just comments and are ignored, so you aren't required to have those, but they are recommended.

**Note:** REST resources are sometimes plural and sometimes singular. For example, the URL below could end with **albums** instead of **album**. For this course, I've chosen to use the singular, but it's actually more common to use the plural.

The URL for the endpoint is: **https://api.example.com/photo/album**

```
# Every Open API file needs this
swagger: '2.0'
```

```
# Document metadata
info:
  version: "0.0.1"
  title: Example Photo Service
```

```
# URL data
host: api.example.com
basePath: /photo
schemes:
  - https
```

```
# Endpoints
paths:
  # Photo albums
  /album:
    # Get one or more albums
    get:
      # Query parameters
      parameters:
        # Starting date
        - name: start
          in: query
          required: false
          type: string

        # Ending date
```

```

    - name: end
      in: query
      required: false
      type: string

# Incomplete response (to finish later)
responses:
  # Response code
  200:
    description: Successful response

# Photo album
/album/{id}:
  # Get an album
  get:
    # Query parameters
    parameters:
      # Album id
      - name: id
        in: path
        required: true
        type: integer

      # Customer level
      - name: Access-level
        in: header
        required: false
        type: string

    # Incomplete response (to finish later)
    responses:
      # Response code
      200:
        description: Successful response

```

## Playlist API

Now it's your turn. You'll create a definition for the API in general, and then two requests: one to retrieve information on one or more playlists, and one that deletes a playlist.

## General Information

Open the Swagger editor at: <http://editor2.swagger.io>

Select everything on the left side and delete it. You'll start from scratch.

The company whose API you are documenting has the domain muzicplayz.com. (It's not real.) This is version 3 of their API, so the base URL is:

```
https://api.muzicplayz.com/v3
```

Add these initial pieces into the Swagger file. It's good practice to add comments, but not required. Refer to the sample YAML above if you need help.

1. Always required: a **swagger** key with a value of **'2.0'**
2. An **info** key with two keys: **version** and **title**. Give it a version of 0.3.0 (needs to be in quotes) and a title of "Music API" (shouldn't be in quotes).
3. The host, basePath, and schemes.

If you look on the right side, you should see your API documented so far. You'll see the title and the version.



Note that the paths are missing, so we are seeing an error.

### Paths and GET Request

The first request to define will return one or more playlists and uses the GET method. Here's a sample request:

```
GET https://api.muzicplayz.com/v3/playlist?limit=10&offset=20&search=jazz
```

The path has this URL:

```
https://api.muzicplayz.com/v3/playlist
```

Add this to the YAML file:

1. Add a **paths** key.
2. Then add the path as the next key. It's the part of the URL after the base path. It should start with a /
3. Add the HTTP method as the next key. It should be lowercase.
4. Add the **parameters** key

This request will have three query parameters:

Query parameter	Required	Definition
<b>limit</b>	Optional	Number of playlists to return
<b>offset</b>	Optional	Index of the first playlist to return. (0=start at the beginning, 10 = skip the first 10, etc.)
<b>search</b>	Optional	Return playlists whose name contains this string

Add list items for each of the query parameters. You will need to have keys for **name**, **in**, **required**, and **type**. See if you can figure those out from the table above and the sample OAS document at the top of this page. Don't forget that you need a dash at the beginning of each list item.

Finally, add a basic response like this:

```
responses:  
  # Response code  
  200:  
    description: Successful response
```

The **responses** key should be at the same indentation as the **parameters** key. We will add more response information in the next exercise.

When done, on the right side, you should see the documentation on the right side:

## Paths

/playlist

GET /playlist

### Parameters

Name	Located in	Required	Schema
limit	query	No	⇔ integer
offset	query	No	⇔ integer
search	query	No	⇔ string

### Responses

Code	Description
200	Successful response

Try this operation

## DELETE Request

The second request to define will delete a playlist using the DELETE method. Here is a sample request:

```
DELETE https://api.muzicplayz.com/v3/playlist/playlist333
```

The path has this URL:

```
https://api.muzicplayz.com/v3/playlist/playlist333
```

where `playlist333` is the playlist ID. Note that this ID is a string because it contains both text and numbers.

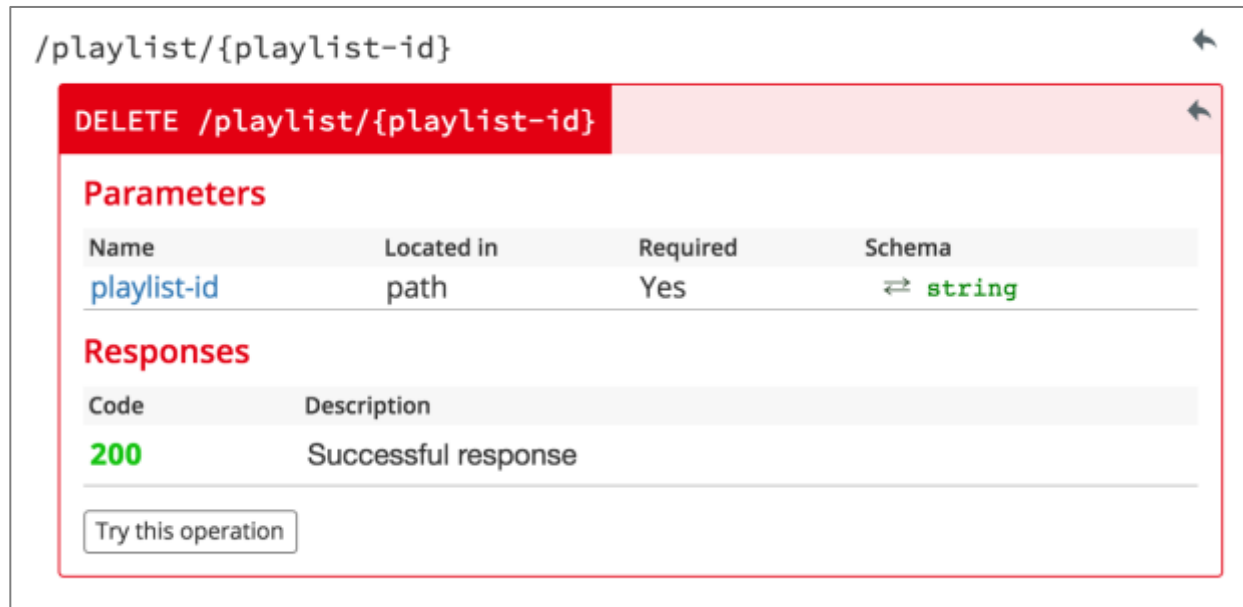
Continue the YAML file:

1. Add the path as the next key. It's the part of the URL after the base path. It should start with a / and contain `{playlist-id}` to indicate a path parameter.
2. Add the HTTP method as the next key.
3. Add the **parameters** key.
4. Add the path parameter as a list item. You will need to have keys for **name**, **in**, **required**, and **type**. See if you can figure those out from the sample OAS document at the top of this page. Don't forget that you need a dash at the beginning because even though there's only one parameter, it's still considered a list item.

Finally, add a basic response like this:

```
responses:  
  # Response code  
  200:  
    description: Successful response
```

Now your documentation on the right should show this information:



The image shows a snippet of a Swagger UI interface. At the top, the path `/playlist/{playlist-id}` is displayed. Below it, a red header bar indicates the HTTP method `DELETE` for the endpoint `/playlist/{playlist-id}`. Underneath, there are two sections: **Parameters** and **Responses**.

The **Parameters** section contains a table with the following data:

Name	Located in	Required	Schema
<code>playlist-id</code>	path	Yes	<code>string</code>

The **Responses** section contains a table with the following data:

Code	Description
<code>200</code>	Successful response

At the bottom of the snippet, there is a button labeled "Try this operation".

### Try this operation

Note that there's a **Try this operation** button on the right side. This will provide information to developers on what information they need to actually make a call to this request.

Click on the **Try this operation** button for the DELETE request. It will open up a box where you can input the playlist ID. You can then click the **Send Request** button. This won't actually work because the API is fictional and there is no server at muzicplayz.com. But if it were real, then clicking that button would actually make a call to the API and show you the results.

Try clicking it anyway. You will get a message: ERROR Server not found or an error occurred.

### Save

It's a good idea to save your YAML file after each exercise. From the **File** menu, choose **Download YAML**. Save this somewhere where you can easily get to it for the next exercise.

### Solution

If you get stuck, you can look at my version of the OAS file:

<http://sdkbridge.com/swagger/Exercise2Answer.yaml>.