MEEC - ADInt

Laboratory 9

In this laboratory students will learn how to call from JavaScript (running on the browser) REST services.

1 JQUERY Ajax

https://api.jquery.com/jquery.ajax/#jQuery-ajax-settings

jQuery Ajax is a method that allows a JavaScript program to make HTTP requests.

This function as a series of functionalities that also allow the simple calling of REST API.

Using this library, the programmer only need to define the endpoint URL, the method (GET, POST, …) and the arguments.

The library is responsible fro the creation of the well formatted message, sending it and process the response.

To use this function it is necessary to include jQuery library.

The creation of such call follows the next example:

```
1.  $.ajax({
2.     url: '/API/videos/',
3.     method: "GET",
4.     dataType: "json",
5.     success: function (data) {
6.            console.log(data);
7.     }
8. });
```

Line 1 starts the creation of the Ajax call :

- line 2 presents the URL/endpoint that is being called. The protocol, address and port is implicit from the page that is running the code
- line 3 defines the method that is being called
- line 4 defines that type of the data that is being sent in the response. If it is json the library will automatically convert the response to JavaScript values

- the function defined on line 5 will be executed when the successful response is received from the server
- The data argument contain a JavaScript object that represents the value returned by the server

it is also possible to define a function 9assigned to the **error** attribute that is called if the request fails.

If it is necessary to send data in a POST/PUT call, the programmer should follow this example:

```
1. $.ajax({
2.   url: '/API/videos/',
3.   type: "POST",
4.   dataType: "json",
5.   contentType: 'application/json',
6.   data: JSON.stringify(requestData),
7. });
```

The argument defined earlier remain, but two other should be used:

- on line 5 the programmer defines the encoding to be used to send the data in the request
- on line 6 the **data** attribute will contain the json that will be sent. The function JSON.stringify convert a JavaScript Object to json.

All other configurations (error, success) should be used

2 Video.js

https://docs.videojs.com/tutorial-setup.html

https://docs.videojs.com/player#play

The provided example allows playing videos (from youtube) on a page.

To play youtube videos on a web page it is necessary to use the **video.js** library by including the following lines in the <HEAD>:

```
<link href="https://vjs.zencdn.net/7.8.4/video-js.css" rel="stylesheet" />
<script src="http://vjs.zencdn.net/7.8.4/video.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/videojs-youtube/2.6.1/Youtube.min.js">
</script>
```

In the <BODY> of the page it is necessary to create am HTML element where the player will be placed:

```
<video id="videoPlayer"
  controls class="video-js vjs-default-skin"
  width="640"
 data-setup='{"techOrder": ["youtube"],
              "sources": [{ "type": "video/youtube" }'>
</video>
```

This **HTML** element will show any youtube video after the programmer supplies it the URL.

To dynamical supply an URL to be played the following JavaScript should be executed:

```
1. var vPlayer = videojs('videoPlayer');
2. vPlayer.src({ "type": "video/youtube",
3.       "src": "https://www.youtube.com/watch?v=xjS6SftYQaQ"});
4. vPlayer.play()
```

- Line 1 retrieves the video player object and stores it a in vPlayer variable.
- Lines 2 and 3 assign a new video to be played.

- Line 4 start playing the video.

This video player can be controlled (paused, restarted and even jump to a particular time:

Button PAUSE Video

```
vPlayer.pause()
var pauseTime =
        vPlayer.currentTime()
$("#resumetime").val(pauseTime)
```

- Pause Video
- Get current time
- Store time on input

Button RESUME Video

```
newTimeStr = $"#resumetime").val()
newTime = parseFloat(newTimeStr)
vPlayer.currentTime(newTime )
vPlayer.play()
```

- Get value from input field
- Convert time to Float
- Change video current time
- Resume video

3 Exercise 6

The provided code implements a REST service with the following API:

- GET /API/videos/
- GET /API/videos/<int:id>/
- POST /API/videos/
- PUT /API/videos/<int:id>/views

The index.html file contains a page with HTML elements and JavaScript that allows the interaction with the server.

This page will allow:

- the presentation of the list of registered videos
- The insertion of a new Video by providing the youtube URL and description of such video
- The selection of one of the available videos to be played. The user will insert the numeric id in the input filed and click Play
- The user will be able to control the video
    - pause
    - resume (at a specific time)

In order to conclude this exercise students should follow the next steps:

3.1 TODO 1

configure the $("#buttonUpdateVideotable") click handle in order to update the table when the user clicks this button

3.2 TODO 2

Update the table when the page is loaded

3.3 TODO 3

Get the values of the #newVideoURL and newVideoDescription and call the addNewVideo function with those values

3.4 TODO 4

Inside the addNewVideo create an JavaScript object that will contain the video URL and description. This object will be converted to JSON ans sent to the server in the AJAX call.

3.5 TODO 5

Update the table after sending the request to create a new video

3.6 TODO 6

get the value of #playVideoID

3.7 TODO 7

use the previous value to call the /videos/<id>/ endpoint to retrieve the video URL

3.8 TODO 8

update the <vPlayer with the received URL

3.9 TODO 9

increase the number of view in the server

3.10 TODO 10

Modify the function updateVideostable so that the **Views** column is filled with the values retrieved from the server.

3.11 TODO 11

update the table after submitting a new view