Luigi De Russis, Alberto Monge Roffarello

LAB 7 – JQUERY, AJAX, REST: A FULL EXERCISE

GETTING STARTED

The goal of this set of exercises is to implement an Ajax front-end for the todo list manager.

Recap:

1. *Fork your own copy of the Git repository associated with this lab (https://github.com/AmI-2018/python-lab7) to your personal GitHub space*
2. *Open PyCharm Professional and select Checkout from Version Control > Git in the "Welcome to PyCharm" window, to clone your (forked) repository*
3. *Fill the requested fields (repository URL, location on disk, …) and press the "Clone" button*
4. *Commit and push the changes you made back to GitHub, from the VCS menu in PyCharm*

EXERCISE 1 – GET TASKS

Extend the client implemented in classroom[1], and, in particular, the "*tasks.js*" script, to visualize the urgent information associated with each task.

As an example, you can look at following repository: https://github.com/AmI-2018/rest-ajax

EXERCISE 2 – INSERT TASKS

Extend the client to provide basic support for **inserting** a task by JavaScript.

For this purpose:

a) Add a ***urgentTask*** checkbox inside the ***addForm*** (file "*tasks.html*") to specify whether the task is urgent or not.
b) Complete the ***submit*** handler (file "*tasks.js*") to extract the values from the form and post them to REST server.
c) Update the list of printed tasks as soon as the server responds with a success response.

As an example, you can look at the following repository: https://github.com/AmI-2018/rest-ajax

EXERCISE 3 – DELETE TASKS

Extend the client to provide basic support for **deleting** a task.

[1] You can find the basic client developed in classroom, along with a working REST server implementation, in the repository of this lab. Clone the repository to start working on it.

For this purpose:

1. Add a ***delete*** button near each task. This button should be added dynamically, from the "*tasks.js*" file.
2. Store the ID of each task in each ***delete*** button. For this purpose, you can use the custom data-* attributes available in HTML5 (https://www.w3schools.com/tags/att_global_data.asp).
3. Implement a ***deleteTask*** function to delete a task, and associate it to the ***click*** event of the ***delete*** button (file "*tasks.js*").
4. Update the list of printed tasks as soon as the server responds with a success response.

EXERCISE 4 – UPDATE TASKS

Extend the client to provide basic support for **updating** a task.

For this purpose:

1. Add an ***update*** button near the ***delete*** one for each printed task. This button should be added dynamically, from the "*tasks.js*" file.
2. Whenever the ***update*** button is pressed:
    a. Load the information related to the selected task in the ***taskDescription*** textbox and in the ***urgentTask*** checkbox of the ***addForm***.
    b. Change the text value of the ***addTask*** button from "Add" to "Update".
    c. Change the method of the ***addTask*** form from POST to PUT, and the ***submit*** handler to a new ***updateTask*** function.
3. Implement the ***updateTask*** function to perform the following operations:
    a. Send the data to the server through a PUT request.
    b. Update the list of printed tasks as soon as the server responds with a success response.
    c. Reset all the elements in the ***addForm*** to the original values/methods.