

Programming for IoT applications

Lab 2

SUGGESTION: Use **Postman**, a chrome plugin for testing REST web services by managing HTTP requests. In case you use **VSCo**de, you can use **ThunderClient** for testing REST web services

Exercise 1. Extend *Exercise_1* proposed during the *Lab 1*, for designing a RESTful-style calculator.

Develop the **HTTP GET** method to manage the following commands:

- **add**: to add two operands and send in the *HTTP body* the JSON;
- **sub**: to subtract two operands and send in the *HTTP body* the JSON;
- **mul**: to multiply two operands and send in the *HTTP body* the JSON;
- **div**: to divide two operands and send in the *HTTP body* the JSON. CHECK that the operation is possible, if not an exception must be raised with the suitable HTTP code;

Manage possible errors in invoking the web services (e.g. wrong command or wrong number of parameters).

The output should be a JSON reporting both input operands, the executed command and the result (validate with <http://jsonlint.com/>)

Example:

- `http://localhost:8080/add?op1=10&op2=12`
where **add** is the command and the parameters **op1** and **op2** provide the input operands
- `http://localhost:8080/sub?op1=10&op2=9`
where **sub** is the command and the parameters **op1** and **op2** provide the input operands

Exercise 2. *Exercise_1 follow-up*: redesign RESTful-style calculator for exposing full URL fashion web services where parameters must be provided slash-separated.

Example:

- `http://localhost:8080/add/10/12/`
- `http://localhost:8080/sub/10/9/`

Exercise 3. Extend *Exercise_2* proposed during the *Lab 1*, for designing a RESTful-style calculator.

Develop the **HTTP PUT** method for receiving in the body-message the following JSON:

```
{
  "command": "add",
  "operands": [10, 9, 8, 7, 6, 5, 3, 2, 1]
}
```

where "command" indicates the operation to be performed among: add, sub, div, mul.
"operands" is an array with the inputs for the operation.

Finally, the **PUT** method should return a JSON reporting the input operands, the executed command and the result (validate with <http://jsonlint.com/>)

Exercise 4. Develop REST web services for deploying a device registering platform (use the version provided as additional material) with CherryPy.

Develop the **HTTP GET** method for providing *index.html*

Develop the **HTTP POST** to save the in the file *devices.json* information provided by the form

The POST method must be able to receive a json like the following one:

```
{
  "deviceId": "1",
  "deviceName": "DHT11",
  "measureType": "temp",
  "unit": "C"
}
```

It's important to keep the name of the field in the example in order to make everything work correctly in the web page

The **POST** method should also return a dictionary like the following one:

```
{"devicesList": [...]}
```