UNIVERSITÉ TOULOUSE III PAUL SABATIER — UPSSITECH ÉCOLE D'INGÉNIEURS

API RESTFul and JSON

Ph. Truillet

October 2021– v. 1.6

0. Preamble

In this Tutorial, you will have to:

- Handle JSON files (read and write)
- Use and write RESTful services

1. REST – Representational State Transfer

REST (**RE**presentational **S**tate **T**ransfer) is not stricto sensu a protocol but an "*architecture style*" defined by Roy Fielding in 2000.
He defined several constraints in order to be *REST* compliant

- The client (user interface) and the server are independent (storage, ...)
- No session variables or volatile states should be stored on the server side: each request should be independent.
- The server tells the client if it can cache the data it receives in order to avoid unnecessary requests and to conserve bandwidth.
- A uniform interface: each resource is accessible in a unique way.
- A hierarchy by layer

Unlike RPC (**R**emote **P**rocedure **C**all) and SOAP (**S**imple **O**bject **A**ccess **P**rotocol), **REST** imposes few constraints. Applications that respect this architecture are said to be **RESTful**.
Resources can undergo four basic operations: CRUD (**C**reate, **R**etrieve, **U**pdate and **D**elete). REST is often used in a web context with the HTTP protocol, taking advantage of the protocol itself (GET, POST, PUT and DELETE keywords) and the use of URIs (Uniform Resource Identifiers) as a resource identification representative.
The API can use any communication medium to initiate interaction between applications. The exchange formats between the clients and the server are mostly plaintext, **xml** (e**X**tended **M**arkup **L**anguage) or **JSON** (**J**ava**S**cript **O**bject **N**otation) defined by RFC 4627 (`https://tools.ietf.org/html/rfc4627`).
REST has many advantages such as being **scalable, simple to implement** with multiple representations, but has the disadvantage of only providing limited security through the use of HTTP methods.

2. JSON - JavaScript Object Natation

JSON is a data exchange format, easy to read by a human and interpret by a machine. Based on JavaScript, it is completely independent of programming languages but uses conventions that are common to all programming languages (C, C++, Perl, Python, Java, C#, VB, JavaScript, ...).
Two structures are used by JSON:

- A collection of keys/values: **Object**. The object starts with a "{" and ends with "}" and consists of an unordered list of key/value pairs. A key is followed by ":" and key/value pairs are separated by ",".

- An ordered collection of objects: **Array**, an ordered list of objects beginning with "["and ending with "]", the objects are separated from each other by ","

JSON supports several data types:

- a value
  - **Numeric**: integer or float
  - **String**: A set of Unicode characters (except for a double quote and backslash) enclosed in double inverted commas.
  - **Boolean**: true or false
  - The **null** value

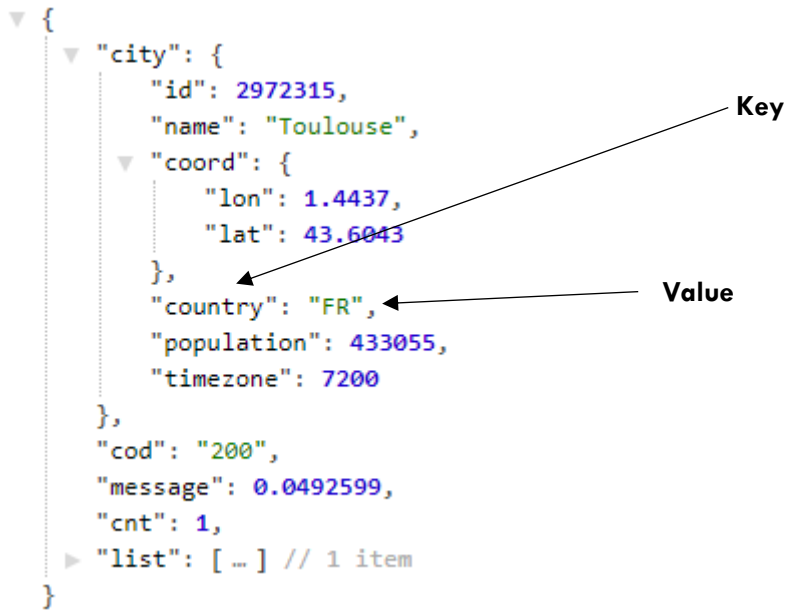
- **an array**: an ordered set of values surrounded by square brackets [and]. Each value is separated by a comma character. The types of values in an array can be different

- **an Object**: is composed of key/value pairs, each separated by a comma, surrounded by braces **{** and **}**. A key must be a string. A value can be literal (string, number, boolean, null), an object or an array. A key is separated from its value by a colon.

The values of an object or array can be of one of these types.
In a character string, the escape character is the backslash character which allows you to represent in the character string:

- \": a double quote
- a backslash
- \/ : a slash
- **b**: a backspace character
- **f** : a formfeed character
- \n : a new line
- \r : a return to the cart
- \t: a tab
- **unnnn**: the Unicode character whose number is nnnn

JSON example



3. Exercises

3.1 JSON and Processing.org

In order to manipulate JSON data with Processing.org, several operations are required.

1. Load JSON data into memory

```
JSONObject json = loadJSONObject(string_JSON)
```

Here, the content of a JSON file (often retrieved after a web request) is loaded into the JSON structure
You will then have to access each field by loading any tables or objects to get there

2. Accessing a table

For example, we can extract the table:

```
JSONArray values = json.getJSONArray("list");
```

3. Accessing an object

```
JSONObject list = values.getJSONObject(0);
```

4. Retrieving a value

```
float pressure = list.getFloat("pressure");
```


```
"cnt": 1,
"list": [
  {
    "dt": 1602154800,
    "sunrise": 1602136805,
    "sunset": 1602177778,
    "temp": { … }, // 6 items
    "feels_like": { … }, // 4 items
    "pressure": 1024,
    "humidity": 57,
    "weather": [
      {
        "id": 803,
        "main": "Clouds",
        "description": "broken clouds",
        "icon": "04d"
      }
    ],
    "speed": 1.96,
    "deg": 40,
    "clouds": 58,
    "pop": 0
  }
]
```


Develop a Processing.org application (see Figure 1) that allows you to manage a collection of physical hardware (e.g. sensor boxes, arduinos, ...).
This application will allow:

1. to flash a QRCode that encode a JSON file. In the JSON file, at least an id, a name and a url (to the product sheet) will be stored
2. display the data decoded on the QRCode in augmented reality. The url should also be clickable.

Figure 1 - Example of recognition and display in augmented reality

You can use the example below to decode QRCodes:
https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/QRCode.zip

3.2 Exercise with an http client to consume services

In this exercise we will use the Java language and the **Apache Components** library (http://hc.apache.org/downloads.cgi) which allows to simply make HTTP requests from a Java application.
Download the example at the following address:
https://github.com/truillet/upssitech/blob/master/SRI/3A/ID/TP/Code/JSON.zip

Compile and run the program. We get the return code of the HTTP call and the content of the file (JSON data)
Decode the JSON data into a previously defined structure with a JSON parser (you can download a simple JSON parser here: **https://github.com/fangyidong/json-simple** or **https://github.com/FasterXML/jackson-core/wiki**)

3.3 Read and manage JSON

- Create a (*free*) account on *open exchange rates* (`https://openexchangerates.org`) and create an application (in the language you want) that uses the proposed REST API to display the exchange rate between different currencies (e.g. $US, € and £)
- • With `http://openweathermap.org`, develop an application which allows the user to ask for **a city name** (in a graphical interface or not), make the necessary calls, retrieve and display the weather forecast of the day (icon in png) and the temperature of the concerned city

Note: this exercise was already proposed within the framework of the initiation to Processing.org 2 years ago 😉

3.4 Producing JSON with an API REST

Reusing part of the web server created in the ***"sockets" tutorial***, create a small "Yellow Pages" application that returns a JSON structure containing the full contact details of the person searched for when the user types in a url from a web browser of the type: `http://@ip/searchbyname?name=nom`

Finally, create a client application (in the language of your choice) that makes the necessary calls to the server and displays the results in a "readable" format.

4. Links

- Best practices for developing REST APIs, `https://www.gekko.fr/les-bonnes-pratiques-a-suivre-pour-developper-des-apis-rest`
- Introducing JSON, `https://www.json.org/json-en.html`