UNIVERSITÉ TOULOUSE III PAUL SABATIER — UPSSITECH ÉCOLE D'INGÉNIEURS

API RESTFul et JSON

Ph. Truillet

Octobre 2022– v. 1.7

0. déroulement du TP

Dans ce TP, vous aurez à :

- Manipuler des fichiers JSON (lecture et écriture)
- Utiliser et écrire des services RESTful

1. REST – Representational State Transfer

REST (**RE**presentational **S**tate **T**ransfer) n'est pas à proprement parlé un protocole mais un « *style d'architecture* » défendu par Roy Fielding en 2000.
Ce dernier a défini plusieurs contraintes afin d'être conforme à l'architecture REST (« *REST compliant* »)

- Le client (interface utilisateur) et le serveur sont indépendants (stockage, …)
- Aucune variable de session ou état volatile ne doit être enregistré côté serveur : chaque requête doit être indépendante.
- Le serveur indique au client s'il peut mettre en cache les données qu'il reçoit afin d'éviter les requêtes inutiles et préserver la bande passante.
- Une interface uniforme : chaque ressource est accessible de manière unique.
- Une hiérarchie par couche

A la différence des protocoles RPC (**R**emote **P**rocedure **C**all) et SOAP (**S**imple **O**bject **A**ccess **P**rotocol), **REST** n'impose que peu de contraintes. Les applications respectant cette architecture sont dites **RESTful**.
Les ressources peuvent subir quatre opérations de base : « CRUD » (**C**reate, **R**etrieve, **U**pdate et **D**elete). REST est souvent utilisé dans un contexte web avec le protocole HTTP en tirant parti du protocole lui-même (mots-clés GET, POST, PUT et DELETE) et de l'utilisation d'URI (Uniform Resource Identifier) comme représentant d'identification des ressources.
L'API peut utiliser n'importe quel moyen de communication pour initier l'interaction entre les applications. Les formats d'échanges entre les clients et le serveur sont la plupart du temps du plaintext, **xml** (e**X**tended **M**arkup **L**anguage) ou **JSON** (**J**ava**S**cript **O**bject **N**otation) définie par la RFC 4627 (`https://tools.ietf.org/html/rfc4627`).

REST a de nombreux avantages comme être **évolutif, simple à mettre en œuvre** avec des représentations multiples mais a l'inconvénient de ne garantir qu'une sécurité restreinte par l'emploi des méthodes HTTP.

2. JSON - JavaScript Object Natation

JSON est un format d'échange de données, facile à lire par un humain et interpréter par une machine. Basé sur JavaScript, il est complètement indépendant des langages de programmation mais utilise des conventions qui sont communes à tous les langages de programmation (C, C++, Perl, Python, Java, C#, VB, JavaScript, …).

Deux structures sont utilisées par JSON :

- Une collection de clefs/valeurs : **Object**. L'objet commence par un « { » et se termine par « } » et composé d'une liste non ordonnée de paires clefs/valeurs. Une clef est suivie de « : » et les paires clef/valeur sont séparés par « , »

- Une collection ordonnée d'objets : **Array**, liste ordonnée d'objets commençant par « [»et se terminant par «] », les objets sont séparés l'un de l'autre par « , »

JSON supporte plusieurs types de données :

- **une Valeur**
  - **Numérique** : nombre entier ou flottant
  - **Chaîne de caractères** : ensemble de caractères Unicode (sauf une double quote **"** et un antislash \) entouré par des doubles guillemets.
  - **Booléen** : true ou false
  - La valeur **null**

- **un Tableau** : un ensemble ordonné de valeurs entouré par des crochets [et]. Chaque valeur est séparée par un caractère virgule. Les types des valeurs d'un tableau peuvent être différents

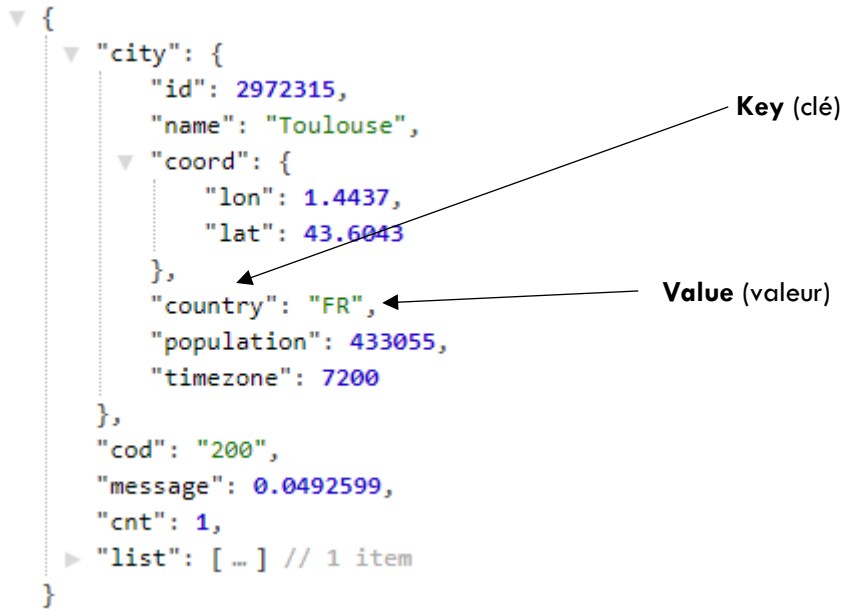
- **un Objet** : est composé de paires clé/valeur, chacune étant séparée par une virgule, entourées par des accolades **{** et **}**.
  Une clé est obligatoirement une chaîne de caractères. Une valeur peut être littérale (chaîne de caractères, un nombre, un booléen, la valeur null), un objet ou un tableau. Une clé est séparée de sa valeur par le caractère « deux points : »

Les valeurs d'un objet ou d'un tableau peuvent être d'un de ces types.

Dans une chaîne de caractères, le caractère d'échappement est le caractère antislash qui permet notamment de représenter dans la chaîne de caractères :

- **"** : une double quote
- \\ : un antislash
- \/ : un slash
- **b** : un caractère backspace
- **f** : un caractère formfeed
- **n** : une nouvelle ligne
- **r** : un retour chariot
- **t** : une tabulation
- **unnnn** : le caractère Unicode dont le numéro est nnnn

Un exemple de JSON



3. Exercices

3.1 Exercice JSON et Processing.org

Afin de manipuler des données JSON avec Processing.org, plusieurs opérations sont nécessaires.

1. Charger les données JSON en mémoire

   ```
   JSONObject json = loadJSONObject(chaine_JSON)
   ```
   Ici, on charge dans la structure JSON le contenu d'un fichier JSON (souvent récupéré après une requête sur le web)

Il va falloir ensuite accéder à chaque champ en chargeant d'éventuels tableaux ou objets pour y arriver

2. Accéder à un tableau
   Par exemple, nous pouvons extraire le tableau :
   `JSONArray values = json.getJSONArray("list");`

3. Accéder à un objet
   `JSONObject list = values.getJSONObject(0);`

4. Récupérer une valeur
   `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
  }
]
```


Développer une application Processing.org (cf. Figure 1) qui permet de gérer une collection de matériel physiques (exemple : des boites de capteurs, arduinos, …).
Cette application va permettre :

1. de flasher un QRCode codant un fichier JSON. Dans le fichier JSON, il sera stocké au moins un id, un nom et une url (vers la fiche produit)
2. d'afficher les données décodées sur le QRCode en réalité augmentée. L'url devra enfin être cliquable.

Figure 1 - Exemple de reconnaissance et d'affichage en réalité augmentée

Vous pouvez partir de l'exemple ci-dessous pour décoder des QRCodes :
`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/QRCode.zip`

3.2. Exercice avec un client http pour consommer les services

Vous pouvez effectuer cet exercice soit en java soit en python

3.2.1. Java

Nous utiliserons dans cet exercice le langage java et la librairie **Apache Components** (`https://hc.apache.org/downloads.cgi`) qui permet d'effectuer simplement des requêtes HTTP à partir d'une application Java.

Télécharger l'exemple à l'adresse suivante :
`https://github.com/truillet/upssitech/blob/master/SRI/3A/ID/TP/Code/JSON.zip`

Compiler et lancer le programme. On récupère le code retour de l'appel web puis le contenu du fichier (données JSON)

Décoder les données JSON dans une structure préalablement définie avec un parser JSON (vous pouvez télécharger un parser JSON simple ici : **`https://github.com/fangyidong/json-simple`** ou **`https://github.com/FasterXML/jackson-core/wiki`**)

3.2.2. Python

Nous utilisons ici le module python requests (`pip install requests`) qui permet d'effectuer des requêtes web à partir d'un fichier python.

Télécharger l'exemple à l'adresse suivante :
`https://github.com/truillet/upssitech/blob/master/SRI/3A/ID/TP/Code/myHttpClient.py`

Décoder les données JSON dans une structure préalablement définie avec un parser JSON (`pip install json`)

3.3 Lire et traiter du JSON

- Créer un compte (*gratuit*) sur *open exchange rates* (**`https://openexchangerates.org`**) et créer une application (dans le langage que vous souhaitez) qui utilise l'API REST proposée pour permettre d'afficher le taux de change entre différentes monnaies (exemple $US, € et £)

- Avec **`https://openweathermap.org`**, développer une application qui permet de demander à l'utilisateur **un nom de ville** (dans une interface graphique ou non), faire l'appel nécessaire, récupérer et afficher la météo du jour (icône en png) et la température de la ville concernée.

  Nota : *cet exercice a déjà été proposé dans le cadre de l'initiation à Processing.org en 1ère année* 😉

3.4 Produire du contenu JSON avec une API REST

En réutilisant une partie du serveur web créé dans le **TP « *sockets* »**, créer une petite application web « *Annuaire* » qui renvoie une structure JSON contenant les coordonnées complètes de la personne recherchée lorsque l'utilisateur tape une url depuis un navigateur web de type : `http://`***`@ip`***`/searchbyname?name=`*`nom`*

Créer enfin une application cliente (dans le langage de votre choix) qui fait les appels nécessaires au serveur et affiche les résultats dans un format « **lisible** ».

4. Liens

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