Synthaxe json en python & JavaScript

Définition

API = Application Programming Interface
Permet donc de programmer une application s'appuyant sur le service Web (par opposition à une utilisation "manuelle" à travers un navigateur).

Comment tester une API ?

- Sous Firefox : ** RestClient **
- Sous Chrome : PostMan

1. Exemple la syntaxe JSON

`

```
{
  "id": 1,
  "label": "File",
  "tooltip": null,
  "items": [
    {"label": "New",   "visible": true},
    {"label": "Open",  "visible": true},
    {"label": "Close", "visible": false}
  ]
}
```

2. donnée a partir d'un fichier

```
import json
# avec des fichiers
data = json.load(open("file.json"))
json.dump(data, open("file2.json", "w"))
```

3. donnée avec des chaînes de caractère

```
import json
# avec des chaînes de caractères
data = """ {
  "id": 1,
  "label": "File",
  "tooltip": null,
  "items":
  }
  """

txt = json.dumps(data)
data2 = json.loads(txt)
```

4. avec Javascript on recupere data json avec fetch

> remarques : En terme de vocabulaire, on parle de tableau (array) plutôt que de liste,
> on parle d'objet plutôt que de dictionnaire.

exemple pour recuperez les enregistrements todo pour un utilisateur donnée

```
var url = `/pro/api/vtodo/${this.user_id}.json/`
console.log("url= " +  url )

fetch(url)
  .then(response => response.json())
  .then(data => this.todos = data )
  .catch(error => { console.log(error)})
  ///
  console.log("mounted! " +   this.todos.length)
},
```

5. Verbes HTTP

Introduction

Avec HTML, on utilise exclusivement les verbes GET et POST, mais HTTP définit d'autres verbes, qui sont particulièrement utiles lors de la définition d'APIs.

- **GET**

> n'attend pas contenu réclame une représentation de la ressource est sans effet de bord (i.e. ne modifie pas l'état de la ressource) rend possible l'utilisation des caches

- **HEAD**

> a la même sémantique que GET mais ne réclame que les en-têtes et pas la représentation de la ressource

- **PUT**

> attend un contenu demande la création ou la modification de la ressource de sorte que le nouvel état de la ressource corresponde au contenu fourni a donc un effet de bord idempotent

- **DELETE**

> n'attend pas de contenu demande la suppression de la ressource a donc un effet de bord idempotent

- **POST**

> attent un contenu peut avoir n'importe quel effet de bord (y compris non-idempotent) est souvent utiliser pour créer une ressource sans connaître a priori son URL

- API RESTful

> On affecte à chaque ressource une URL qui l'identifie, et chaque URL identifie une ressource. Les manipulations se font en utilisant les verbes

> HTTP appropriés.