JSON

Antonio Espín Herranz

Contenidos

- Introducción y descripción del formato Json.
- Json en Python
- Módulo json / métodos

Introducción

- **JavaScript Object Notation** es un formato ligero de representación de datos.
- Idealmente pensado para el intercambio de datos, permite estructurar la información sin requerir XML.
- Normalmente, una representación JSON suele ser mas compacta que su equivalente XML.
- JSON puede ser analizado de forma más rápida que XML.

Introducción

- JSON está constituido por dos estructuras:
  - Una colección de **pares de nombre/valor**. Lo que como conocemos como un objeto.
  - Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como **arrays**.

- Un **objeto** es un conjunto ordenado desordenado de pares nombre/valor. Comienza con una **{** y termina con **}**

- Cada nombre es seguido de : (dos puntos) y los pares **nombre / valor** van separados por , (comas).

Introducción

- Un **array** es una colección de valores. Un array comienza con **[** (corchete izquierdo) y termina con **]** (corchete derecho). Los valores se separan con , (coma).

- Un **valor** puede ser:
  - Una **cadena de caracteres** con comillas dobles.
  - Un **número**,
  - o **true** o **false** o **null**,
  - o un **objeto**,
  - o un **array**.

- **Las estructuras pueden anidarse**. Podemos tener relaciones de **composición** entre **objetos**

Introducción

- Una **cadena de caracteres es una colección de 0 o más caracteres**, encerrados entre comillas dobles, usando barras divisorias invertidas como escape.

- Secuencias de escape: \n, \t, \r, etc.

Notación

- Es simple transformar un documento XML a la notación JSON:
  - El documento JSON comenzará y terminará en una llave.
  - Cada elemento XML se transforma en una cadena de caracteres encerrada entre comillas dobles, a la que sigue el carácter de dos puntos.
  - En XML: <titulo>
  - En JSON: “titulo”:

Notación

- El contenido textual de los elementos aparecerá posteriormente, encerrado entre comillas y seguido de una coma, si hay mas contenido:
  - XML: <titulo>Ejemplo de JSON</titulo>
  - JSON: “titulo”:”Ejemplo de JSON”,

Notación

- Si el elemento de XML contiene elementos hijos, estos aparecerán encerrados entre corchetes.
- No necesario con el elemento raíz.
- Si un elemento contiene atributos, cada uno se trata como un elemento y su valor como el contenido textual del elemento. Si tiene varios, se separan por comas.

Notación

- Si un elemento contiene atributos, cada uno se trata como un elemento y su valor como el contenido textual del elemento. Si tiene varios se separa por comas.

- Los atributos de los elementos se encerrarán entre llaves.

- <item valor=“demo” nota=“ejemplo” />
- Se transforma en:
  - “item” : [
    {“valor” : ”demo”, “nota” : “ejemplo” }
    ]

Ejemplo en XML

```
<?xml version="1.0"?>
<menu id="fichero" valor="Fichero">
   <popup>
      <menuitem id="nuevo" value="Nuevo Fichero"/>
      <menuitem id="abrir" value="Abrir Fichero" />
      <menuitem id="salir" value="Salir del menú/>
   </popup>
</menu>
```

Ejemplo en JSON

```
{ "menu" : {
    "id": "fichero",
    "valor": "Fichero",
    "popup": {
          "menuitem": [
                {"id": "nuevo", "value": "Nuevo Fichero"},
                {"id": "abrir", "value": "Abrir Fichero"},
                {"id": "salir", "value": "Salir del menu"}
          ]
    }
  }
}
```

Json en Python

- El formato Json es muy similar a la sintaxis de los diccionarios de Python.
- **Equivalencia** entre los tipos de Python y las estructuras de Json:

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

Módulo json / métodos

- import json

- Codificación en Json: **dumps**

  json.dumps(['foo', {'bar': ('baz', **None**, 1.0, 2)}])

  '["foo", {"bar": ["baz", null, 1.0, 2]}]‘

  – ***Le pasamos una estructura en Python***.

  None → null

  La tupla a […]

Módulo json / métodos

- Codificación en Json: **dumps**
  - **Ordenar por las claves:**

```
print(json.dumps({"c": 0, "b": 0, "a": 0}, sort_keys=True))
{"a": 0, "b": 0, "c": 0}
```

  - **Formatear la impresión:**

```
print(json.dumps({'4': 5, '6': 7}, sort_keys=True, indent=4))
{
    "4": 5,
    "6": 7
}
```

Módulo json / métodos

- Decodificación en Json: **loads**
  - json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')
  - ['foo', {'bar': ['baz', None, 1.0, 2]}]
  - ***Le pasamos una cadena en Json.***

Módulo json / métodos

- Para grabar en un fichero:
  - json.**dump**(obj, fichero)
  - Serializa el objeto en formato json a un fichero.

- Para cargar json de un fichero:
  - json.**load**(fp)
  - El archivo contiene un formato json válido y lo devuelve como un objeto Python.

Enlaces

- Visor online Json

  https://codebeautify.org/jsonviewer

- Documentación Python:

  https://docs.python.org/3/library/json.html