La gestion du format JSON

Le format JSON est une constituante indispensable des architectures REST. Bien que ces architectures puissent utiliser n'importe quel format d'échange, JSON n'en reste pas moins celui qui est largement plébiscité grâce à sa simplicité et à sa légèreté.

1. Présentation du format JSON

Le format JSON (*JavaScript Object Notation*) est un format de données textuel inspiré de la notation des objets JavaScript, créé à partir de 2002 par Douglas Crockford. L'objectif du format JSON est de faciliter les échanges de données applicatives sur le Web, échanges qui étaient auparavant systématiquement basés sur XML. Ce dernier étant jugé trop verbeux et donc consommateur de bande passante, JSON constitue une alternative plus légère pour ces échanges.

**a. Représentation des données en JSON**

JSON représente les données sous forme d'objets. Ils sont constitués d'attributs exprimés grâce à des paires clé/valeur, la clé correspondant au nom de l'attribut et la valeur à la donnée associée. Cette donnée peut être un numérique, une chaîne de caractères, un autre objet ou bien un tableau.

Les clés sont des chaînes de caractères et il est nécessaire de les exprimer entre guillemets.

Les valeurs sont séparées des clés par le caractère deux points (:) et chaque couple clé/valeur est séparé du suivant par une virgule (,).

*Exemple de structure*

...

"clé1": "Valeur de clé1",

"clé2": "Valeur de clé2"

...

**b. Types de données**

Les types de données définis par JSON sont :

- les chaînes de caractères : elles sont exprimées en Unicode et entre guillemets,
- les numériques : ce sont des nombres entiers ou décimaux,
- les booléens : ils sont exprimés avec les mots réservés **true** et **false**,
- les objets : ils représentent la base de la notation de JSON,

- les tableaux : ce sont des ensembles de valeurs des types précédents.

**c. Structures**

Les paires de clé/valeur de JSON sont exprimées entre accolades ({}), c'est la manière de représenter un objet. Ainsi, si l'on souhaite exprimer les données d'un objet **personne** qualifié par un **nom**, un **prénom** et un **âge**, on peut utiliser la structure suivante :

```
{
  "nom": "DUPONT",
  "prenom": "Robert",
  "age": 56
}
```

Dans cet exemple, les valeurs pour le nom et le prénom sont des chaînes de caractères, l'âge est un entier.

Les tableaux sont, quant à eux, exprimés entre crochets ([]), comme par exemple :

```
...
"clé1": [ "Première valeur de clé1", "Deuxième valeur de clé1" ]
...
```

Bien évidemment, les valeurs de tableaux peuvent elles-mêmes être des tableaux ou bien des objets. Prenons l'exemple d'une structure complexe qui représenterait un objet **client** possédant des attributs sur son identité, ainsi qu'un attribut **comptes** ayant comme valeur un tableau d'objets **compte**.

On pourrait représenter la structure de la manière suivante :

```
{
  "id": 1,
  "nom": "DUPONT",
  "prenom": "Robert",
  "adresse": "40 rue de la Paix",
  "codePostal": "75000",
  "ville": "Paris",
  "comptes": [
```

```
    {
      "numero": 245646786,
      "solde": 8300.0
    },
    {
      "numero": 263434345,
      "solde": 20100.0
    }
  ]
}
```


2. Le support de JSON en PHP

Dans le langage PHP, la manipulation de données en JSON s'articule essentiellement autour de deux fonctions : **json_encode()** et **json_decode()**.

La fonction **json_encode()** permet de transformer une structure de données PHP en flux JSON. Elle s'appuie sur des données simples (chaînes de caractères, entiers, réels) ou plus souvent sur des tableaux associatifs ou des objets.

La fonction **json_decode()** permet l'inverse : elle crée une structure de données PHP à partir d'un flux JSON. Selon la valeur de son deuxième paramètre, cette structure est un tableau associatif ou bien un objet créé à partir de la **stdClass** de PHP.

Bien que ces fonctions soient très utiles en PHP, elles n'en restent pas moins limitées. En effet, PHP ne propose pas nativement de gestion d'une réponse HTTP au format JSON ; ces fonctions ne font que gérer des flux JSON. Elles n'ont donc que peu d'intérêt dans une application Symfony.

3. Et dans Symfony ?

Dans Symfony le support de JSON est natif. Il permet aussi bien de manipuler des flux que de gérer des requêtes et des réponses dans ce format. Symfony utilise pour cela un sérialiseur.

Un **sérialiseur** (*serializer* en anglais) est un outil logiciel permettant de réaliser des opérations de **sérialisation** et **désérialisation**.

La **sérialisation** est le mécanisme permettant de transformer en flux de données la structure d'un objet logiciel résidant en mémoire afin de pouvoir stocker ce flux dans un fichier ou bien de le transporter sur le réseau. La **désérialisation** est l'opération inverse :

on construit l’état d’un objet en mémoire à partir d’un flux de données provenant d’un fichier ou bien du réseau.

Symfony vient avec un sérialiseur par défaut ; ce dernier est capable de gérer les formats XML, YAML, CSV et évidemment JSON.