XML vs JSON

Sommaire

XML – Extensible Markup Language

XML est un métalangage informatique de balisage. il permet de décrire et analyser toutes les sortes de documents (sauf binaire). C'est une norme de structure de données.

Il s'organise sous forme de balises qui ne sont pas figées. Laissant l'opportunité d'organiser le document comme on le souhaite et d'imposer sa présentation. Les documents XML ont donc une très forte structuration de document, au dépourvu de la lisibilité qui se voit un peu brouillon.

Structure au format XML :

```xml
<!-- 342 caractères-->
<?xml version='1.0' encoding="UTF-8" ?>
<employees>
 <employee>
   <firstName>John</firstName>
   <lastName>Doe</lastName>
 </employee>
 <employee>
   <firstName>Anna</firstName>
   <lastName>Smith</lastName>
 </employee>
```

```
  <employee>
    <firstName>Peter</firstName>
    <lastName>Jones</lastName>
  </employee>
</employees>
```

Description

Il y a tout d'abord une liste **d'employés**, qui contient des **”employé”**, qui contient les **”nom”** et **”prénom”** des employés.

Ce fichier décrit donc une liste d'employées avec leur nom et prénom et tout ça en 342 caractères. Nous verrons par la suite que JSON peu mieux faire, avec plus de visibilité.

Utilisation du XML

XML ne sert à rien. Du moins s'il est utilisé tout seul.

Ce qui est vraiment utile, ce sont tous les langages qui tournent autour. Certains sont utilisés pour des domaines très spécifique, comme le CML (**C**hemical **M**arkup **L**anguage) qui est utilisé pour décrire des composés chimiques.

On peut le mettre en forme avec du CSS et du XSL (e**X**tensible **S**tylesheet **L**anguage) et décrire le contenu avec du DTD (**D**ocument **T**ype **D**efinition) et encore plein d'autres utilités qui ne seront pas cités.

Ce sont ces langages qui complètent le XML et le rendent utile. Ainsi, si la solution choisie pour le traitement de données est le XML, il va falloir d’abord utiliser un ou plusieurs autres langages en rapport avec le XML.

XPATH

Le XPATH est un langage pour localiser une portion d’un document XML. Il est le langage de requête élémentaire de XSLT.

Voici un tableau des expression du XPATH

XPATH Expression	Description
/	l'objet ou l'élément racine
.	l'objet ou l'élément courant
//	descente récursive
*	wildcard
[]	expression de filtre

Quelques petits exemples.

1. exemple XPATH	Description
//employees	Récupère toutes le document XML

```
<employees>
 <employee>
   <firstName>John</firstName>
   <lastName>Doe</lastName>
 </employee>
 <employee>
   <firstName>Anna</firstName>
   <lastName>Smith</lastName>
 </employee>
 <employee>
   <firstName>Peter</firstName>
   <lastName>Jones</lastName>
 </employee>
</employees>
```

2. exemple XPATH	Description

//employees/employee/firstName	Retourne les ‘firstName’ des balises ‘employee’

```
<employees>
 <employee>
   <!-- <firstName>John</firstName> -->
   <lastName>Doe</lastName>
 </employee>
 <employee>
   <!-- <firstName>Anna</firstName> -->
   <lastName>Smith</lastName>
 </employee>
 <employee>
   <!-- <firstName>Peter</firstName> -->
   <lastName>Jones</lastName>
 </employee>
</employees>
```

3. exemple XPATH	Description
//employees/employee[firstName='John']	Retourne toutes les balises de la ‘employee’ qui comporte ‘John’ à l’intérieur d’elle.

```
<!-- Retourne -->
<employee>
   <firstName>John</firstName>
   <lastName>Doe</lastName>
</employee>
```

JSON

JSON est un format récursif compatible avec JavaScript. D'ailleurs, Il se construit comme un objet JavaScript.

Une valeur peut être: un objet, un tableau, booléenne, etc... Et il n'y a besoin de rien de plus à savoir pour créer un fichier JSON.

Exemple au format JSON :

```
/* 150 caractères */
{"employees":[
        { "firstName":"John", "lastName":"Doe" },
        { "firstName":"Anna", "lastName":"Smith" },
        { "firstName":"Peter", "lastName":"Jones" }
]}
```

Description

En comparant les deux Langages, il est possible de voir que le XML a pratiquement deux fois plus de caractères que son concurrent.

Utilisation du JSON

Bien qu'il utilise la notation JavaScript, il est indépendant de celui-ci. Il sert à communiquer avec des applications dans un environnement hétérogène. Il est notamment utilisé comme langage de transport de données par AJAX et les services Web.

Il peut aussi être utilisé pour :

- Les fichiers de configuration,
- La sérialisation et désérialisation d'objets,
- L'encodage de documents.

JSONPATH

Comme pour le XPath de XML, JSON possède également un outil de traitement; le JSONPath. Un de ses avantages majeur est la façon d'écrire une requête. En effet, puisque le JSON utilise une structure de données semblable à celle d'un langage de programmation tel que le C. Voici donc un exemple avec une requête XPath :

```
/employees/employee[1]/firstName
```

Et ressemble à en JSONPath :

```
employees[1].firstName
```

ou

```
['employees'][1]['firstName']
```

Liste des opérateurs XPath et leur équivalent en JSONPath

Xpath	JSONPath	Description
/	$	l'objet ou l'élément racine (il n'y en a pas forcément en JSON)
.	@	l'objet ou l'élément courant
//	..	descente récursive
*	*	wildcard
n/a	[start,end]	Liste d'indexes
		Opérateur de découpage d'array, step

n/a	[start:end:step]	représente le nombre d'index "sauté" à chaque step
[]	?()	expression de filtre
n/a	()	expression faisant appel au langage de script natif

Exemple

En reprenant l'exemple de fichier JSON ci-dessus, nous allons écrire une requête dans les deux façons possibles qui va afficher toutes les personnes ayant pour prénom "John" :

1. On veut rechercher une information dans les enfants de l'objet "**employees**", on va donc le sélectionner et appliquer dessus un filtre

```
employees[?()]
```

2. Ce qui nous intéresse, c'est le prénom de la personne. On utilise **@.firstName** pour indiquer qu'on va appliquer ce filtre sur les enfants de l'éléments courant

```
employees[?(@.firstName)]
```

3. On définit la condition qui fait que le filtre renvoie **true**, dans notre cas le prénom "John"

```
employees[?(@.firstName == John)]
```

Notez qu'il n'est pas nécessaire d'utiliser des quotes pour comparer un string, contrairement à la structure du fichier JSON qui l'impose.

Dans l'autre façon d'écrire des requêtes, cette dernière ressemblerait à :

```
['employees'][?(@.firstName == John)]
```

Avantages et désavantages

JSON

- Il est plus simple de lire du JSON.
- Le traitement est plus rapide.
- Avec sa structure en arborescence et sa syntaxe, il reste très "léger" (moins de caractères) et efficace.
- Simple à mettre en oeuvre.
- Pas besoin de parser un fichier XML. Ce format est très ouvert donc il est pris en charge par de nombreux langages, comme le JavaScript.
- Pour les applications AJAX, le JSON est plus rapide que le XML
- Il permet de stocker différents types:
- Chaine de caractères (string);
- Tableaux (array);
- Nombre (int, float);

XML

- Il est extensible quand au langage, on peut créer des formats comme le SVG ou même le RSS.
- Il pourrait mieux convenir aux utilisateurs (non programmeurs).

Exercices

1. XML vs JSON

transformer la structure XML en JSON

```
<employees>
  <employee>
          <firstName>John</firstName>
          <lastName>Doe</lastName>
  </employee>
  <employee>
          <firstName>Anna</firstName>
          <lastName>Smith</lastName>
  </employee>
  <employee>
          <firstName>Peter</firstName>
          <lastName>Jones</lastName>
  </employee>
</employees>
```

2. JSONPATH

En JSONPATH, l'équivalent de XPATH, récupérer le numéro de téléphone fixe (home)

```
{
"firstName": "John",
"lastName" : "doe",
"age"      : 26,
"address"  : {
  "streetAddress": "naist street",
  "city"         : "Nara",
  "postalCode"   : "630-0192"
},
"phoneNumbers": [
  {
    "type"  : "iPhone",
    "number": "0123-4567-8888"
  },
```

```
    {
      "type"   : "home",
      "number": "0123-4567-8910"
    }
  ]
  }
```

3. XPATH

Pareil avec le XML

```
<persons>
        <person>
                <firstName>John</firstName>
                <lastName>doe</lastName>
                <age>26</age>
                <address>
                        <streetAddress>naist
street</streetAddress>
                        <city>Nara</city>
                        <postalCode>630-
0192</postalCode>
                </address>
                <phoneNumbers>
                        <type>iPhone</type>
                        <number>0123-4567-
8888</number>
                </phoneNumbers>
                <phoneNumbers>
                        <type>home</type>
                        <number>0123-4567-
8910</number>
                </phoneNumbers>
        </person>
</persons>
```

3. Convertir le JSON en objet JavaScript

Convertissez le JSON, ...

```
{"mycars" :[
        {"name":"Audi"},
        {"name":"BMW"},
        {"name":"Mercedes"},
        {"name":"Volvo"}
]}
```

en objet Javascript pour qu'il s'affiche de la manière suivante :

```
Audi
BMW
Mercedes
Volvo
```