Injecter des données provenant d'un JSON dans un XML avec XSLT 3.0

Jules Nuguet

Injecter des données provenant d’un JSON dans un XML

Jules Nuguet

21 juin 2024

Université
Jean Monnet
Saint-Étienne

Inconvénients de JSON par rapport à XML

Inconvénient 1 : Absence de schéma

- JSON n'a pas de schéma intégré pour valider la structure des données.
- XML utilise des schémas (XSD, DTD) pour définir et valider la structure des documents.
- L'absence de schéma rend plus difficile la validation des données JSON.

Inconvénient 2 : Standardisation et outils

- XML bénéficie d'une standardisation et d'un écosystème d'outils riche (XPath, XSLT, etc.).
- JSON manque de certains outils de manipulation et de transformation de données disponibles pour XML.

Avantages 1 : Simplicité

- Moins verbeux que XML
- trés facile à utiliser pour le web

Comment fonctionne un JSON

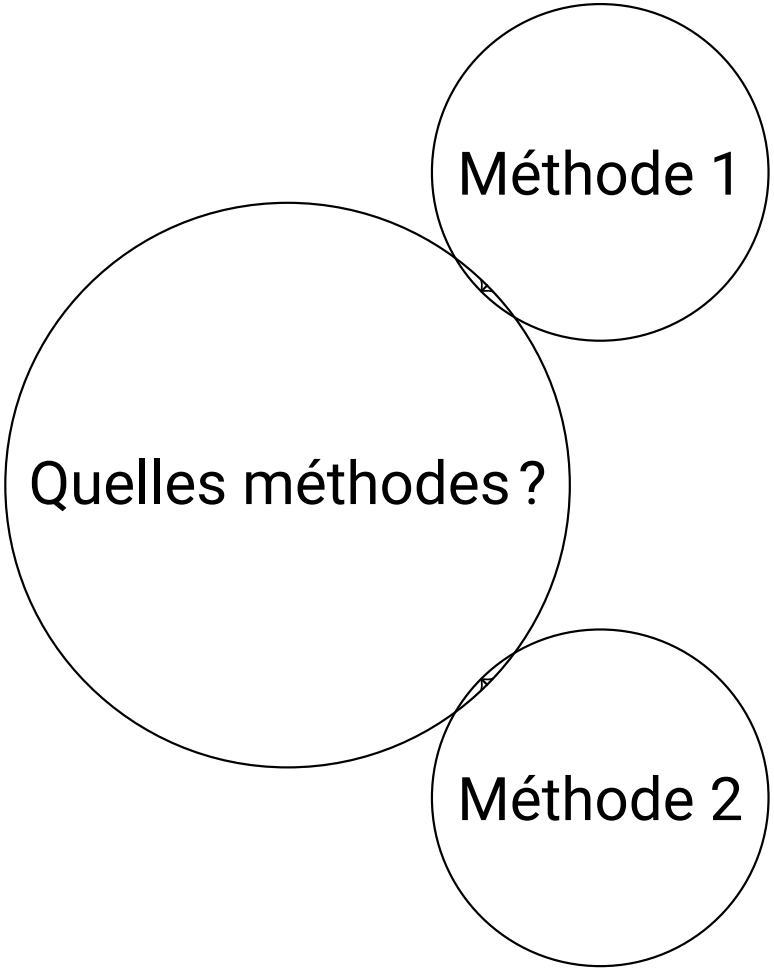
Vocabulaire

- { } objet
- "🔑" : clé
- [ ] une liste

Exemple

```
{ "livre": "Vingt mille lieues sous
↪  les mers",
  "personnages": [
    { "nom": "Capitaine Nemo",
      "age": "Inconnu",
      "nationalite": "Inconnue",
      "attributs": ["Charismatique
      ↪  ","Érudit","Inflexible
      ↪  ","Secret"]
    },
    { "nom": "Professeur Pierre
    ↪  Aronnax",
      "age": 50,
      "nationalite": "Française",
      "attributs": ["Curieux","Sava
      ↪  nt","Observateur","Human
      ↪  iste"
      ]
    }
  ]
}
```

Quelles méthodes ?



- **Méthode 1 : Transformer le JSON en XML**
  - Permet des traitements plus rapides en évitant de créer trop de boucles.
- **Méthode 2 : Naviguer dans le JSON et récupérer les valeurs**
  - Permet de conserver la structure logique du JSON.

Exercice Guidé - intrp

json-to-xml(unparsed-text('input.json'))

```
<map xmlns="http://www.w3.org/2005/xpath-functions">
   <string key="livre">Vingt mille lieues sous les mers</string>
   <array key="personnages">
      <map>
         <string key="nom">Capitaine Nemo</string>
         <string key="age">Inconnu</string>
         <string key="nationalite">Inconnue</string>
         <array key="attributs">
            <string>Charismatique</string>
            <string>Érudit</string>
            <string>Inflexible</string>
            <string>Secret</string>
         </array>
      </map>
      <map>
         <string key="nom">Professeur Pierre Aronnax</string>
         <number key="age">50</number>
         <string key="nationalite">Française</string>
         <array key="attributs">
            <string>Curieux</string>
            <string>Savant</string>
            <string>Observateur</string>
            <string>Humaniste</string>
         </array>
      </map>
   </array>
</map>
```

Exercice Guidé - 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tei="http://www.tei-c.org/ns/1.0"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    exclude-result-prefixes="xs fn tei"
    version="3.1">
    ...
</xsl:stylesheet>
```

Exercice Guidé - 1

```
<xsl:output indent="yes"/>
<xsl:variable name="json">
    <xsl:copy-of select="json-to-xml(unparsed-text('input.json'))"/>
</xsl:variable>

<xsl:template match="@*|node()">
    <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
</xsl:template>
...
```

Exercice Guidé - 1

```
<xsl:template match="tei:text">
    <text xmlns="http://www.tei-c.org/ns/1.0" >
    <xsl:element name="listPerson" namespace="http://www.tei-c.org/ns/1.0">
        <xsl:message><xsl:value-of
        ↪  select="$json//fn:array[@key='personnages']"/></xsl:message>
        <xsl:for-each select="$json//fn:array[@key='personnages']//fn:map">
            <person>
                <persName><xsl:value-of
                ↪  select="./fn:string[@key='nom']//text()"/></persName>
                <age><xsl:value-of select="./*[@key='age']//text()"/></age>
                <nationality><xsl:value-of
                ↪  select="./fn:string[@key='nationalite']//text()"/></nationality>
                <traitDesc>
                    <xsl:for-each select="./fn:array[@key='attributs']//fn:string">
                        <trait><xsl:value-of select=".//text()"/></trait>
                    </xsl:for-each>
                 </traitDesc>
            </person>
        </xsl:for-each>
    </xsl:element>
    </text>
</xsl:template>
```

Fonctions XPath 3.1 pour les Maps et les Arrays

Opérations sur les Maps

- `map:size` : Retourne la taille d'une map.
- `map:keys` : Retourne toutes les clés d'une map.
- `map:get` : Récupère la valeur d'une clé spécifiée dans une map.
- `map:merge` : Fusionne deux maps en une seule.
- `map:for-each` : Itère sur chaque entrée d'une map.

Opérations sur les Arrays

- `array:size` : Retourne la taille d'un array.
- `array:get` : Récupère un élément à partir d'un index spécifié dans un array.
- `array:append` : Ajoute un élément à la fin d'un array.
- `array:for-each` : Itère sur chaque élément d'un array.

Conversion vers et depuis JSON

- `fn:parse-json` : Convertit une chaîne JSON en structure de données XPath.
- `fn:json-doc` : Charge un document JSON à partir d'une source.
- `fn:json-to-xml` : Convertit une structure JSON en XML.
- `fn:xml-to-json` : Convertit une structure XML en JSON.

Pour plus de détails, consultez la documentation complète des fonctions XPath 3.1 sur
https://www.w3.org/TR/xpath-functions-31/#

Exercice Guidé - 2

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tei="http://www.tei-c.org/ns/1.0"
    xmlns:array="http://www.w3.org/2005/xpath-functions/array"
    xmlns:map="http://www.w3.org/2005/xpath-functions/map"
    exclude-result-prefixes="xs fn tei map array"
    version="3.1">
    ...
</xsl:stylesheet>
```

Exercice Guidé - 2

```
<xsl:output indent="yes"/>

<xsl:template match="@*|node()">
    <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
</xsl:template>
...
```

Exercice Guidé - 2

```
<xsl:template match="tei:text">
    <text xmlns="http://www.tei-c.org/ns/1.0" >
    <xsl:element name="listPerson" namespace="http://www.tei-c.org/ns/1.0">
       <xsl:message ></xsl:message>
        <xsl:for-each select="array:flatten(map:get(json-doc('input.json'),
        ↪  'personnages'))">
            <xsl:variable name="character" select="." as="map(*)"/>
            <person>
                <persName><xsl:value-of select="map:get($character,'nom')"/></persName>
                <age><xsl:value-of select="map:get($character,'age')"/></age>
                <nationality><xsl:value-of
                ↪  select="map:get($character,'nationalite')"/></nationality>
                <traitDesc>
                    <xsl:for-each select="array:flatten(map:get($character,'attributs'))">
                        <trait><xsl:value-of select="."/></trait>
                    </xsl:for-each>
                 </traitDesc>
            </person>
         </xsl:for-each>
    </xsl:element>
    </text>
</xsl:template>
```

Exercice Guidé - 2 V2

```
<xsl:template match="tei:text">
    <text xmlns="http://www.tei-c.org/ns/1.0" >
    <xsl:element name="listPerson" namespace="http://www.tei-c.org/ns/1.0">
       <xsl:message ></xsl:message>
        <xsl:for-each select="json-doc('input.json')!map:get(., 'personnages')?*">
            <xsl:variable name="character" select="." as="map(*)"/>
            <person>
                <persName><xsl:value-of select="$character?nom"/></persName>
                <age><xsl:value-of select="$character?age"/></age>
                <nationality><xsl:value-of select="$character?nationalite"/></nationality>
                <traitDesc>
                    <xsl:for-each select="$character?attributs?*">
                        <trait><xsl:value-of select="."/></trait>
                    </xsl:for-each>
                 </traitDesc>
            </person>
         </xsl:for-each>
    </xsl:element>
    </text>
</xsl:template>
```

Exercices pratiques

Transformer un IIIF en TEI

L' objectif :

> À partir du manifeste ci-dessous :
> https://iiif.io/api/cookbook/recipe/0068-newspaper/newspaper_issue_1-manifest.json,
> récupérer le texte contenu dans les annotations.

Récupérer des informations supplémentaires via une API de réconciliation

L'objectif :

> À partir de la liste de personnes du fichier `exercice-reconcile.xml`, récupérer des identifiants dans la base de Biblissima via l'adresse API suivante :
> https://data.biblissima.fr/reconcile/fr/api?query=\{valeur\}
> Puis, dans un second temps, ajouter des informations supplémentaires comme les dates de naissance et de mort.