Représentation des données sur Internet

Olivier Coupelon

2019-2020

Besoin

- Systèmes d'informations hétérogènes
  - Windows / MAC / Unix ...
  - Endianess (ordre des mots en mémoire)
- Echange de données entre services de différentes compagnies, sur différents systèmes
- Exemples :
- Yahoo, Flickr, Youtube...

Encoding

- ASCII
  - 1 caractère = (7bit) → 128 caractères possibles
- ISO/CEI 8859-1
- Windows-1252
- Unicode, dont UTF-8

→ Codage binaire standardisé

XML

eXtensible Markup Language

- W3C recommendation http://www.w3.org/TR/2004/REC-xml11-20040204/
- Dérivé de SGML. v1.0 en 1998, v1.1 en 2004
- Un méta-langage pour créer des dialectes.
- Structure arborescente.

- Matérialisation commune sous une forme textuelle qui encourage l'emploi de UTF-8.

Use cases

- Echange de données entre applications
- Formats de documents (Docbook, OpenOffice, ...)
- Configuration d'applications
- Bases de données XML
- ... **web services**

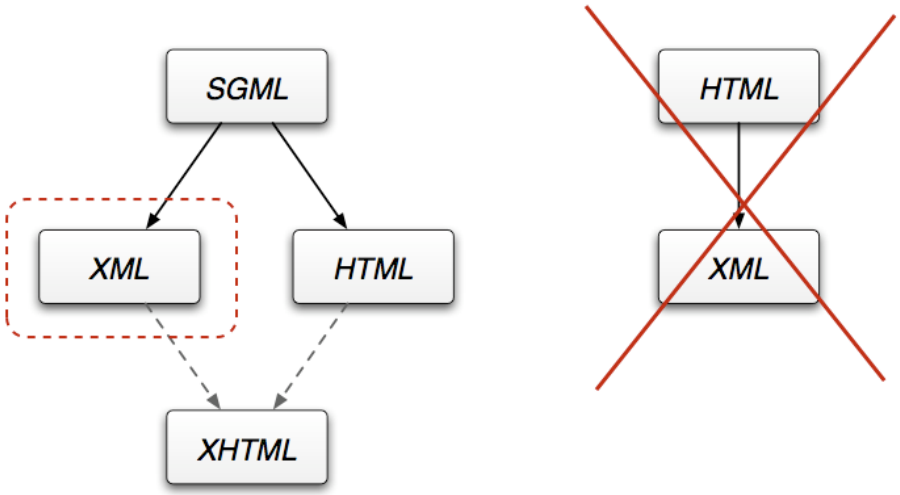

Figure 1: Origines de XML

Syntaxe

- Utilisation de *balises* ou *éléments* 'ouvrants', 'fermants', ou 'vides'
- À une balise ouvrante doit correspondre une balise fermante

```
<toto>BLABLA</toto>
<toto />
```

- Une balise peut contenir des attributs

```
<toto attribut="valeur" />
```

- Caractères clés du langage : <, >, &, ', "
- Ils doivent être remplacés pour toute utilisation hors langage par une syntaxe type :
  - < : <
  - > : >
  - & : &
  - ' : '
  - " : "

Le texte contenu par une balise CDATA peut être du texte brut, peu importe alors son formatage :

```
<![cdata[ Mon texte "brut".

Il peut contenir des caractères spéciaux &<>>, des sauts de lignes... ]]>
```

Well formness

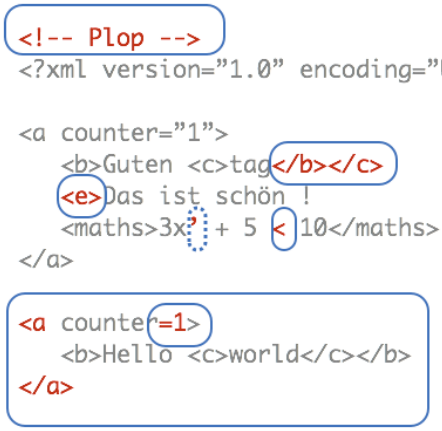

Figure 2: Un document mal formé

TD - exercice 1

- http://olivier.coupelon.net/ > WebServices ISIMA

Figure 3: QR code

Namespace

- Permet de créer des groupes de balises, identifiés par un nom unique, une URI.
- Cette URI peut être un lien vers le schéma, mais pas forcément.
- Création d'un nouveau schéma avec namespace :

  ```
  xmlns="http://www.w3.org/1999/xhtml"
  targetNamespace="http://www.w3.org/1999/xhtml"
  ```

- Déclaration de l'utilisation d'un namespace :

  ```
  xmlns:prefix="http://www.w3.org/1999/xhtml"
  ```

- Utilisation d'un élément d'espace de nommage prédéfini :

  ```
  <prefix:element />
  ```

Schéma XML

- XML est un méta-langage : *orthographe*
- Description du langage cible : *grammaire*
- Normes :
  - DOCTYPE
  - XmlSchema
  - RelaxNG

DOCTYPE

```
<!ELEMENT addressBook (card+)>
<!ELEMENT card (name, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

- Permet de déclarer éléments et attributs.
- Une DTD n'est pas un document XML.
- Pas de typage (texte, dates, entiers, . . .).
- https://www.w3schools.com/xml/xml_dtd_elements.asp
- Déclaration :

```
<!DOCTYPE html>
<html>
```

XmlSchema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="adressBook">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="card" minOccurs="1"
                         maxOccurs="unbounded">
                <xsd:element name="name" type="xsd:string" />
                <xsd:element name="email" type="xsd:string" />
            </xsd:element>
        </xsd:sequence>
```

```
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

- Documents XML : structure et types de données.
- https://www.w3schools.com/xml/schema_intro.asp

RelaxNG

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="addressBook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <oneOrMore>
    <element name="card">
      <element name="name">
        <text/>
      </element>
      <element name="email">
        <text/>
      </element>
    </element>
  </oneOrMore>
</element>
```

- Décrit des motifs valides de documents. Peu verbeux.
- http://www.relaxng.org/

TD - exercices 2 & 3

- http://olivier.coupelon.net/ > WebServices ISIMA

Parsers SAX (Simple API for XML)

1. Lecture du premier élément XML
2. Déclenchement de l'évènement correspondant au type d'élément lu
3. On recommence avec l'élément suivant

Figure 4: QR code

Parsers DOM (Document Object Model)

1. On charge le document en mémoire
2. On se place sur l'élément racine
3. On se déplace dans l'arbre XML pour atteindre l'élément recherché directement

---

XPath

- Langage d'extraction de contenu XML
- https://devhints.io/xpath
- Actuellement en version 2.0
- Langage basé sur l'arbre XML

Requêtes

```
<!xml version="1.0" ?>
<racine>
    <encyclopedie nom="Wikipedia" site="http://fr.wikipedia.org">
        <article nom="XPath">
            <auteurs>
                <auteur>
                    <nom>Dupont</nom>
                </auteur>
                <auteur>
                    <nom>Dubois</nom>
                </auteur>
            </auteurs>
        </article>
    </encyclopedie>
</racine>

/
/root
//article
/racine/encyclopedie
//article[@nom='XPath']
//auteurs/auteur[2]
//article[ count( article/auteurs/auteur ) >1 ]
```

XSLT

Extensible Stylesheet Language Transformation

Source XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Austin's library -->
<audiolibrary>
    <owner email="autin@powers.name">Austin Powers</owner>
    <record>
        <name>Soul Bossa Nova</name>
        <artist id="quincyjones">Quincy Jones</artist>
    </record>
```

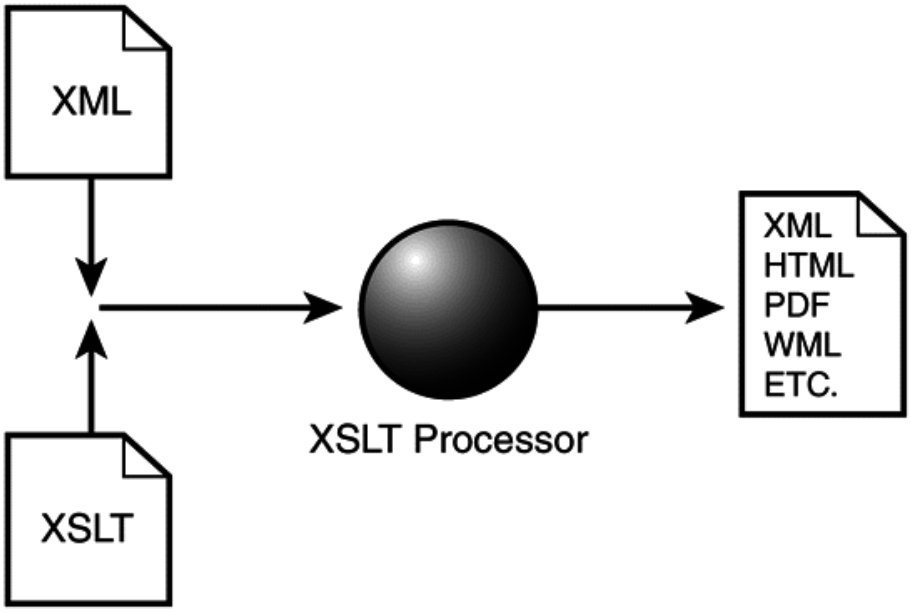

Figure 5: Moteur de transformation XSLT

```
    <record>
        <name>Soul Bossa Nova - Lorie Punk Remix</name>
        <artist refid="quincyjones" />
        <artist>Lorie</artist> <!-- Arg!!! -->
    </record>
    <record ranking="top"> <!-- Classy tune! -->
        <name>You're Not From Brighton</name>
        <artist>Fatboy Slim</artist>
    </record>
</audiolibrary>
```

Code XSLT

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="/">
        <html>
            <head>
                <title><xsl:value-of select="/audiolibrary/owner/text()" />'s songs</title>
            </head>
```

```
            <body>
                <ul>
                    <xsl:for-each select="/audiolibrary/record">
                        <li><xsl:value-of select="name/text()" /></li>
                    </xsl:for-each>
                </ul>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Résultat après compilation

```
<html>
    <head>
        <title>Austin Powers's songs</title>
    </head>
    <body><ul>
    <li>Soul Bossa Nova</li>
    <li>Soul Bossa Nova - Lorie Punk Remix</li>
    <li>You're Not From Brighton</li>
    </ul></body>
</html>
```

XML et Java

JAXB (Java Architecture for XML Binding)

- Avoir une représentation de données Java sérialisables
- Création de classes Java depuis des schémas XML (XSD) et inversement
- Outils :
  - Java → XSD : schemagen
  - XSD → Java : xjc

Marshalling / Unmarshalling

```
JAXBContext jc = JAXBContext.newInstance("foo.bar");
Unmarshaller unmarshaller = jc.createUnmarshaller();
```

```
MonObject o = (MonObjet)
     unmarshaller.unmarshal(new File("test.xml"));
```

JSON

JavaScript Object Notation

- Utilisé pour les services exposé directement vers le navigateur (Mashup)
- Structure :
  - Paires clé / valeurs
  - Les valeurs peuvent être des listes de paires, des tableaux de paires, des objets, chaines de caractères...
- **Pas de standard** pour la validation des données

Exemple de JSON

```
{"menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                {"value": "New", "onclick": "CreateNewDoc()"},
                {"value": "Open", "onclick": "OpenDoc()"},
                {"value": "Close", "onclick": "CloseDoc()"}
            ]
        }
    }
}
```

Utilisation de JSON

- Depuis un navigateur
  - Pour l'interpréter, on utilisait la fonction JavaScript eval() :-/ :
    var donnees = eval('('+donnees_json+')');
  - Natif dans les navigateurs depuis 2009 (IE8, Firefox 3.5, Opera 10.5) :
    var donnees = JSON.parse(donnees_json);
- Dans les framework JavaScript (Node.js)

Note sur l'échange de données avec navigateur web

- JSON-P : JSON with Padding
  - Pour récupérer des données en outrepassant la "same-origin policy" (même port, même domaine).
  - On encapsule les données dans un fonction Javascript connue de la page.
  - Déprécié (ou alors pour de vieux navigateurs) au profit de CORS (Cross-Origin Resource Sharing), via l'ajout d'entêtes HTTP.

Protobuf

Protocol Buffers

- Créé par Google, disponible depuis 2008 en version publique
- Permet la sérialisation de données structurées, indépendamment du lanage et de la plateforme
- Autres alternatives : Apache Thrift, Microsoft Bond, Apache Avro
- https://developers.google.com/protocol-buffers/

Avantages de protobuf (vs XML)

- Plus simples : la validation du format de message est faite par le serialiseur, on dispose bien d'un schéma
- De 3 à 10 fois plus compact : le protocole est binaire, et non textuel
- De 20 à 100 fois plus rapide : le parsing du binaire est efficace
- Moins d'ambiguité, mais toujours pas de sémantique
- Les représentations dans les langages sont disponibles et doivent être utilisées

> Avec gRPC : https://grpc.io/docs/guides/

Comparaison

Représentation XML Schema :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="a" type="xs:byte"/>
</xs:schema>
```

Représentation Protobuf :

```
message Test1 {
required int32 a = 1;
}
```

Une instance de message Protobuf (binaire) :

```
08 96 01
```

Pour aller plus loin

Formats : https://en.wikipedia.org/wiki/Comparison_of_data_serialization_formats