Web et échange de données

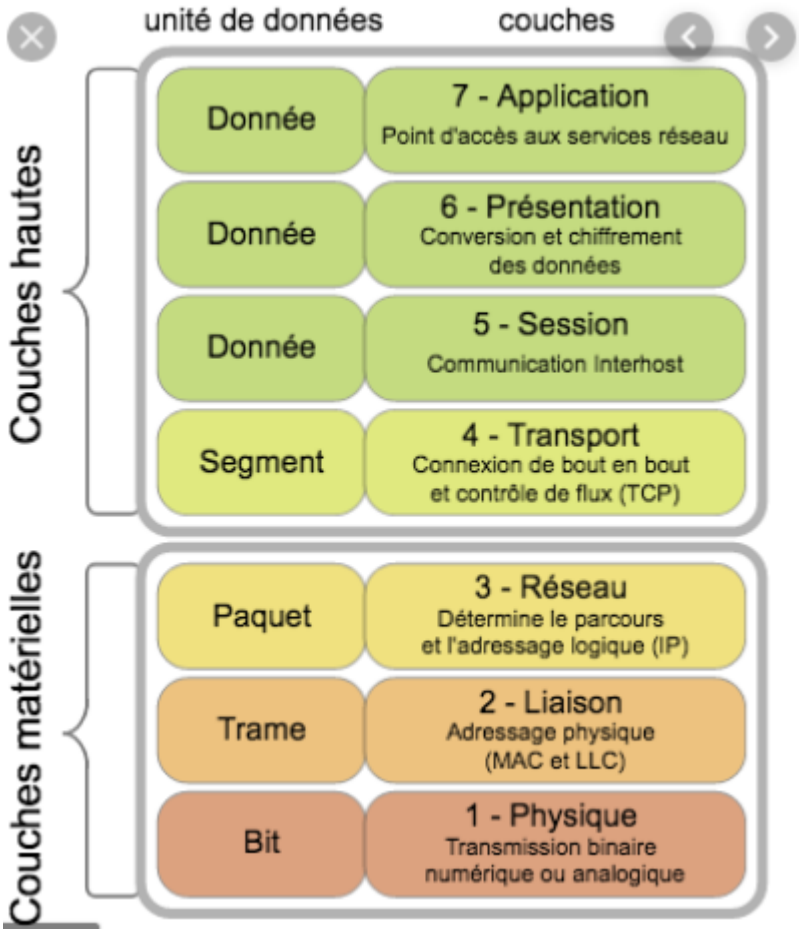
Avènement du web

les entreprises ont dut fournir à leur client et partenaire de plus en plus de services « à la demande » depuis les années 2000.

- Du fait de l'augmentation des capacité réseau et infrastructure.
- Du fait de la prolifération des « providers Internet » qui ont aussi contribué à la démocratisation des technologies dites « web ».

HTTP

Définition



Ainsi le protocole HTTP s'est imposé comme fondement « réseau » de la plupart des échanges entre application.

L'HyperText Transfer Protocol, plus connu sous l'abréviation HTTP — littéralement « protocole de transfert hypertexte » — est un protocole de communication client-serveur développé pour le World Wide Web.

HTTPS (avec S pour secured, soit « sécurisé ») est la variante du HTTP sécurisée par l'usage des protocoles SSL ou TLS.

HTTP est un protocole de la couche application. Il peut fonctionner sur n'importe quelle connexion fiable, dans les faits on utilise le protocole TCP comme couche de transport.

La version actuelle de http est la version 1.1. Elle a été publié par IETF en Février 2014 dans les RFC 7230 à 7237.

HTTP est un protocole client-serveur à commande : c’est-à-dire qu’il demande à un serveur de faire quelques choses et de renvoyer quelques choses à l’appelant.

La commande est encapsulée dans une requête les éventuels retour dans une réponse.

Le protocole HTTP/2 diffère de HTTP/1.1 sur plusieurs aspects:

- Il est encodé en binaire plutôt qu'en texte. Il ne peut donc plus être lu ou écrit à la main. Malgré cette difficulté, il est désormais possible d'implémenter des techniques d'optimisation avancée.
- C'est un protocole multiplexé. Plusieurs requêtes en parallèle peuvent être gérées au sein de la même connexion, supprimant ainsi la limitation séquentielle de HTTP/1.x.

HTTP/2 compresse les en-têtes, étant donné que des en-têtes similaires sont échangés lors d'une suite de requêtes, on supprime ainsi la duplication et l'échange inutiles des données similaires.

Il permet au serveur de remplir le cache du client avant qu'il ne soit demandé par ce dernier, on parle alors d'évènements générés par le serveur.

Fonctionnement

La requête est composée d’une zone d’entête et d’un corps.

Les entêtes sont codifiées mais extensibles

Le corps référence une URI et une méthode à exécuter.

La réponse est composée d’une zone d’entête et d’un corps.

Les entêtes sont codifiées mais extensibles

Le corps comporte ou pas du contenu et peut être facultatif selon la méthode utilisée.

Méthodes

GET est la méthode la plus courante pour demander une ressource. Une requête GET est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.

HEAD ne fait que demander des informations sur la ressource, sans demander la ressource elle-même.

POST transmet des données en vue d'un traitement à une ressource. L'URI fourni est l'URI d'une ressource à laquelle s'appliqueront les données envoyées.

OPTIONS obtient les options de communication d'une ressource ou du serveur en général.

CONNECT pour utiliser un proxy comme un tunnel de communication.

TRACE demande au serveur de retourner ce qu'il a reçu, dans le but de tester et effectuer un diagnostic sur la connexion.

PUT remplace ou ajoute une ressource sur le serveur. L'URI fourni est celui de la ressource en question.

PATCH fait une modifie partiellement une ressource, contrairement à PUT.

DELETE supprime une ressource du serveur.

Données

Problématique

Informations et documents dans l'entreprise

Constat :

- Inflation du volume d'informations
- Coût et difficulté :
- Difficultés à retrouver un document
- Coût induit par la gestion de documents

Pour structurer cette « masse » d'information, il faut la transformer en donnée.

Une donnée est une information structurée.

Cela peut se faire soit en travaillant directement sur les informations ou en en ajoutant d'autres afin de pouvoir plus facilement retrouver ce que l'on recherche.

W3C

Le WORLD WIDE WEB Consortium crée des standards pour le Web.

Le W3C en 7 points…

- Accès universel (internationalisation, web mobile…)
- Web sémantique (RDF, XML…)
- Confiance (signatures électroniques, collaboratif…)
- Intéropérabilité (consensus industriel)
- Evolutivité (simplicité, modularité, compatibilité…)
- Décentralisation (systèmes répartis, tolérance aux erreurs)
- Multimédia (SVG, SMIL, HTML5)

Pour plus d'informations : www.w3c.org

Principales normalisations

	1986 : SGML	1991 : HTML	1998 : XML
Objectifs	adaptabilité	simple	puissance de SGML
	intelligence	portable	Simplicité de SGML
	gestion des liens	gestion de liens	
Inconvénients	complexee	non adaptable	
	difficilement portable	non intelligent	

XML

Définition

e**X**tensible **M**arkup **L**anguage

- Recommandation (norme) du W3C
- Spécifiant un langage
- Constitué d'un ensemble d'éléments appelés balises
- Utilisable pour créer d'autres langages

2 concepts fondamentaux

- Structure, contenu et présentation sont **séparés**
- Les balises ne sont pas **figées**

Conséquences :

- XML est un format de document
- XML est un format de données
- XML est un mode de structuration de l'information
- XML est un méta-langage

Caractéristiques

Richesse sémantique de SGML

- Compatible SGML
- Dédié au traitement des données
- Soutient une grande variété d'applications
- Assure un entretien aisé
- Intelligent

Facilité de mise en œuvre de HTML

- Simple et lisible
- Portable et facilement utilisable sur Internet
- Assure un développement aisé

Exemple

```
<song>
<title> Ma chanson </ title >
<composer> par l’auteur</ composer >
< producer > Dupond</producer>
<editor> Maison edition</ editor >
<duration > 6:20</duration>
<date> 1978</ date >
< artist >  Toto</artist>
</song>
```

- Balises “propriétaires” compréhensibles à la profession, dérivation des langages proprétaires.
- XML sépare le contenu de son aspect (à la différence de HTML ou tout peut être mélangé)

Structure

entete

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="presentation1.xsl"?>
<exposition>
```

Contenu

```
<tableau>
        <titre>Déjeuner sur l'herbe</titre>
        <artiste>Edouard Manet</artiste>
    </tableau>
    <tableau>
        <titre>Déjeuner des Canotiers</titre>
        <artiste>Auguste Renoir</artiste>
    </tableau>
    <tableau>
        <titre>Waterloo Bridge, le soleil dans le brouillard</titre>
        <artiste>Claude Monet</artiste>
    </tableau>
    <tableau>
        <titre>Mardi Gras, soleil couchant, bd Montmartre</titre>
        <artiste>Camille Pissaro</artiste>
    </tableau>
    <tableau>
        <titre>Les Joueurs de cartes</titre>
        <artiste>Paul Cézanne</artiste>
    </tableau>
</exposition>

</xml>
```

SPECIFICATIONS

Résumé des spécifications :

- Un document doit commencer par une déclaration XML
- Toutes les balises avec un contenu doivent être fermées
- Toutes les balises sans contenu doivent se terminer par les caractères />
- Le document doit contenir un et un seul élément racine
- Les balises ne doivent pas se chevaucher
- Les valeurs d'attributs doivent être entre guillemets
- La casse doit être respectée pour toutes les occurrences de noms de balise (MAJUSCULES ou minuscules).

Un document respectant ces critères est dit "bien formé"

Arbre XML

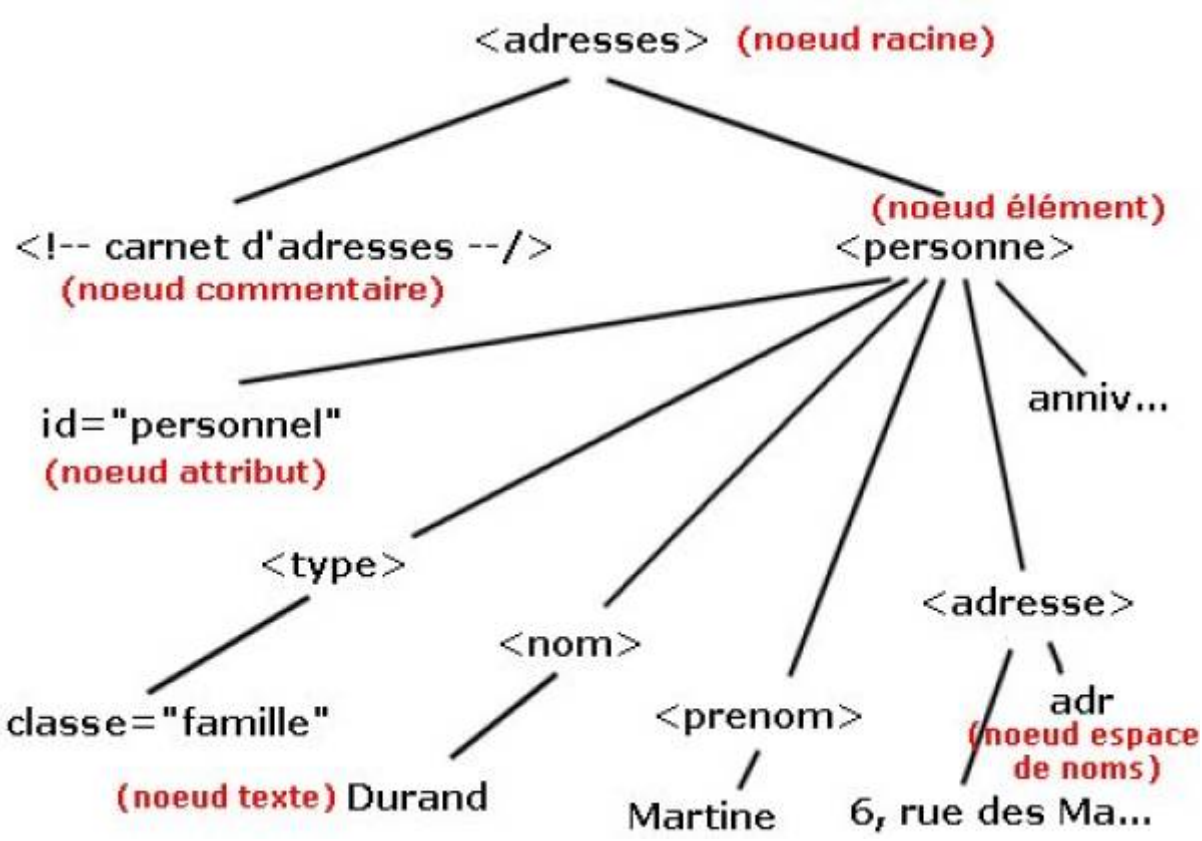
Exemple

Exemple de fichiers XML :

```
<adresses>
  <!--Carnet d'adresses-->
  <personne id= "personnel">
    <type classe= "famille"/>
    <nom>Durand</nom>
    <prenom>Martine</prenom>
    <adr:adresse>6, rue des Magnolias</adr:adresse>
  </personne>
</adresses>
```

Arbre

Arbre correspondant :



JSON

JSON (Javascript Object Notation) est un format de donnée textuelle. Il est permet de représenter de l'information structurée et en particuliers des Objets Javascript.

Il est décrit dans la RFC 7159 de l'IETF.

En ce sens c'est un « concurrent » d'XML.

JSON n'utilise que deux types d'éléments structurels :

- Des paires nom/valeur
- Des listes ordonnées de valeurs

Ces éléments structurels peuvent représenter trois types de données :

- Des objets
- Des tableaux
- Des éléments génériques (tableau, objet, booléen, nombre, string, null)

JSON contrairement à XML ne se veut pas universel et ne peut être étendu.

Toutefois, il est moins verbeux qu'XML et aussi « lisible » que lui.

JSON ne peut être validé et il est donc à la charge du développeur de vérifier si les éléments récupérés sont bien ceux attendus.

Un document JSON représente un Objet Javascript.

Représentation

Le type MIME application/json est utilisé pour transporter un document JSON sur HTTP.

Historiquement, l'opérateur eval était utilisé pour « transformer » une chaine de caractère en Object javascript.

Pour des raisons de sécurité, les navigateurs « modernes » intègrent nativement JSON et fournissent l'objet Natif JSON pour évaluer les documents JSON

```
var data = JSON.parse(json_data);
```

Exploitation

La structuration de l'information est importante mais il est aussi nécessaire de pouvoir « comprendre » la structure d'un document.

Pour cela il est nécessaire de définir la grammaire de notre document.

Il existe deux modes de définition de celle-ci :

- Les DTD définition « historique »
- Les schémas XML basés sur une définition XML

DTD

Définition

DTD = Document Type Definition

La DTD fournit :

- la liste des éléments,
- la liste des attributs,
- des notations et
- des entités du document XML associé ainsi que
- les règles des relations qui les régissent.

Résumé des spécifications :

- Une DTD (grammaire) permet de déclarer :
    - un type d'élément,
    - une liste d'attributs d'un élément,
    - une entité
- Chaque balise du langage doit faire l'objet d'une et d'une seule déclaration

Un document XML est dit "valide" s'il possède une DTD et si sa syntaxe est conforme aux règles de la DTD

Un document “valide” est obligatoirement « bien formé »

La DTD est déclarée dans le document XML par la balise !DOCTYPE

Elle peut être :

- incluse dans le code source du fichier XML, ou DTD interne :

```
<!DOCTYPE élément-racine [déclaration des éléments]>
```

- décrite dans un fichier externe, ou DTD externe :

```
<!DOCTYPE élément-racine SYSTEM "nom_fichier.dtd">
```

Exemple

DTD interne

```
<?xml version="1.0" standalone="yes"?>
<!--Comme vous définissez une DTD interne, votre fichier est indépendant
(standalone).-->

<!DOCTYPE parent [

<!--Début de la DTD interne avec parent comme élément de racine.-->

<!ELEMENT parent (garcon,fille)>
<!--L’élément racine parent contiendra les sous-éléments garcon et fille.-->

<!ELEMENT garcon (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
<!--#PCDATA indique au Parser XML que l'élément garcon contient des données
exprimées en chiffres ou en lettres. Idem pour l'élément fille.-->

]>
<!--Fin de la DTD-->

<parent> <!--Racine du document XML.-->
<garcon>François</garcon>
<fille>Elisabeth</fille>
</parent>  <!--Fin du document XML.-->
```

DTD externe

fichier XML

```
<?xml version="1.0" standalone="no"?>
<!--Comme vous définissez une DTD externe, votre fichier n’est plus
```

```
indépendant (standalone).-->

<!DOCTYPE parent SYSTEM "parent.dtd">
<!--Déclaration de la DTD externe dans le Fichier parent.dtd.-->

<parent>
<!--Racine du document XML.-->

<garcon>François</garcon>
<fille>Elisabeth</fille>
</parent>
<!--Fin du document XML-->
```

Fichier parent.dtd

```
<!ELEMENT parent (garcon,fille)>
<!--L'élément racine parent contiendra les sous-éléments garcon et fille.-->
<!ELEMENT garcon (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
<!--
#PCDATA indique au Parser XML que
l'élément garcon contient des données
exprimées en chiffres ou en lettres.
Idem pour l'élément fille.
-->
```

La DTD contient :

Une ou plusieurs définitions d'éléments introduites par la balise !ELEMENT :

```
<!ELEMENT nom-élément valeur>
```

Une ou plusieurs listes d'attributs introduites par la balise !ATTLIST :

```
<!ATTLIST nom-élément attribut type défaut>
```

Une ou plusieurs définitions d'entité introduites par la balise !ENTITY :

```
<!ENTITY nom-entité "valeur">
```

ou

```
<!ENTITY nom-entité SYSTEM "nom_fichier">
```

Espaces de nom

- XML Namespaces est une recommandation permettant d'utiliser le vocabulaire (les balises) de 2 DTD distinctes sans risque d'ambigüité.

Inconvénients des DTD :

- Une DTD est difficile à lire
- Une DTD est non extensible (ce n'est pas un document XML).
- Une DTD ne permet pas de typer les données
- Une DTD ne peut prendre en compte qu'un seul espace de nom (Namespace).

XML schema

En réponse aux lacunes des DTD, une alternative a été proposée comme recommandation : il s'agit de XML-Data dont XML-Schema est un sous-ensemble.

Cette nouvelle norme achève de faire d'XML un format pivot…

La version 1.1 de XML Schema (datée de mai 2001) se compose de 3 normes :

- XML Schema tome 0 : Introduction
- XML Schema tome 1 : Structures
- XML Schema tome 2 : Types de données

Les documents XML-Schema sont des documents qui :

- espectent la syntaxe XML,
- peuvent décrire la structure d'un document XML d'une façon beaucoup plus complète que les DTD.

XML-Schema permet en effet de :

- spécifier la typologie des données que va contenir le document XML décrit par le XML-Schema,
- gérer une quarantaine de types de données simples,
- gérer des types complexes,
- gérer les occurrences des données.

Exemple

Document XML

```
<entree>
  <nom>Harry Potter</nom>
  <telephone>0102030405</telephone>
</entree>
```

Document XML-schema correspondant

```
<wsd:schema xmlns:xsd="http://www.w3org/2000/10/XMLSchema"> <!--entete-->
<xsd:element name="entree"> <!--Définition de la balise complexe "entree"-->
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nom" type="xsd:string"
                   minoccurs="1" maxoccurs="1"/> <!-- Définition de la
balise string "nom"-->
      <xsd/ement name="telephone" type="xsd:decimal"/> <!--Définition de
```

```
la balise de type Décimal telephone-->
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Référence à un XML-Schema dans un document XML

```
<entree xmlns="http://www.annuaire.org"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="http://www.annuaire.org/entree.xsd">
  <nom>Harry Potter</nom>
  <telephone>012030405</telephone>
</entree>
```

L'espace de nommage xsi correspond aux instances de documents XML respectant les contraintes définies dans un document XML-Schema. Le W3C a défini une librairie de balises et attributs pouvant être utilisés par ces documents.

La DTD permet de définir facilement et rapidement des grammaires simples.

XML-Schema permet de définir de manière plus formelle et complète une grammaire mais c'est au prix d'une complexité accrue.

Un document XML-Schema respecte la syntaxe XML.

Un document XML-Schema est généralement plus volumineux et plus difficile à lire qu'une DTD (pour un opérateur humain).

Exemple de XML-XSD

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<cave xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation='td2.xsd'>
<appellation id="01" nom="Saint Estèphe">
    <chateau id="0101">
        <nom> Chateau Bellevue</nom>
        <adresse> 12 Rue du Pont</adresse>
        <téléphone> 0556124321</téléphone>
    </chateau>
    <chateau id="0102">
        <nom> Chateau Le Bernadot</nom>
        <adresse> 21 Avenue du Cygne</adresse>
        <téléphone> 0556324231</téléphone>
    </chateau>
</appellation>
</cave>
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
 <xsd:element name="cave">
  <xsd:complexType>
```

```
   <xsd:sequence>
     <xsd:element ref="appellation" minOccurs='1' maxOccurs='unbounded'/>
   </xsd:sequence>
  </xsd:complexType>
  </xsd:element>

  <xsd:element name="appellation">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element ref="chateau" minOccurs='0' maxOccurs='unbounded'/>
   </xsd:sequence>
   <xsd:attribute name="id"  type="xsd:integer" use='required'/>
   <xsd:attribute name="nom" type="xsd:string"/>
  </xsd:complexType>
 </xsd:element>
<xsd:element name="chateau">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element ref="nom" minOccurs='1' maxOccurs='1'/>
     <xsd:element ref="adresse"/>
     <xsd:element ref="téléphone"/>
   </xsd:sequence>
   <xsd:attribute name="id" type="xsd:integer" use='required'/>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="nom" type="xsd:string"/>
 <xsd:element name="adresse" type="xsd:string"/>
 <xsd:element name="téléphone" type="xsd:decimal"/>
 </xsd:schema>
```

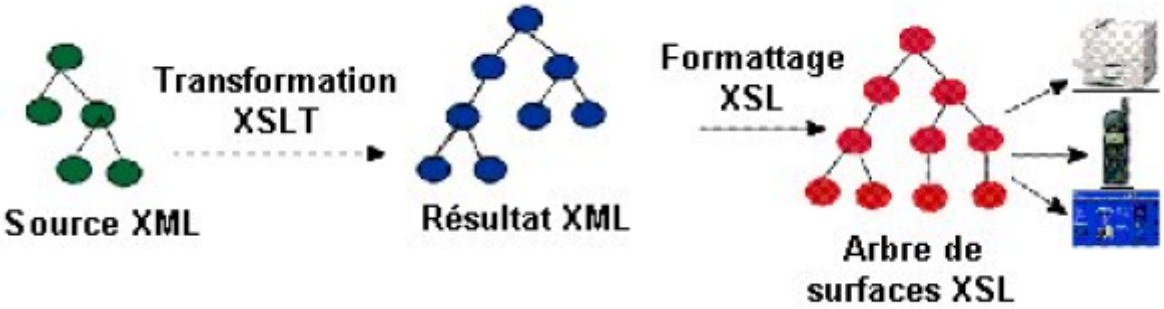
TRANSFORMATION D'UN DOCUMENT XML

Les outils destinés à transformer les documents XML représentent ceux-ci comme un arbre de nœuds XML.

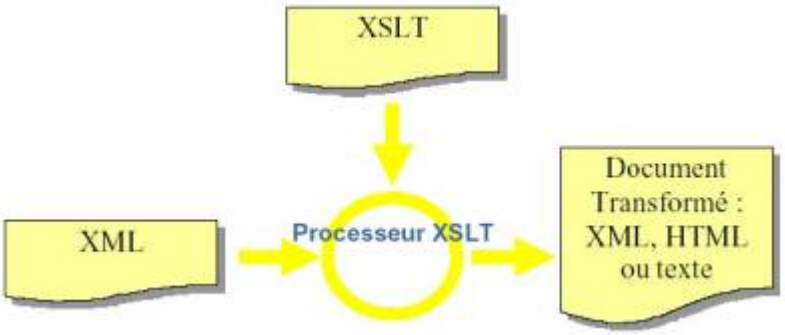
XSL est le langage qui permet d'écrire des feuilles de style. Une feuille de style est constituée d'un ensemble de règles de transformations, s'appliquant chacune à un ou plusieurs nœuds de l'arbre et permettant de transformer ce nœud en un nouveau nœud de l'arbre résultat.

XPath est le langage qui permet d'adresser une partie ou plusieurs parties d'un document, c'est à dire un ou plusieurs nœuds du document XML.

Transformation effectuée par un processeur XSL Selon http://www.w3.org/Style/XSL/



XSL



XSL : eXtensible Stylesheet Language.

Objectifs : Ce langage déclaratif permet de transformer un document XML en :

- un autre document XML, HTML, WML, SMIL
- un document papier: PDF, LaTeX
- du texte

Principes

Le langage XSLT décrit des règles pour transformer un document XML.

Ces règles de transformations s'appliquent chacune à un ou plusieurs nœuds de l'arbre et spécifient la transformation à effectuer sur un nœud pour le transformer en un nouveau nœud de l'arbre résultat.

Un processeur XSLT applique à un document XML les transformations décrites dans un document XSLT et génère un nouveau document (XML, HTML, texte)

Fonctions de base (transformations) offertes par une feuille de style XSLT :

- extraction de données
- génération de texte
- suppression de contenu (noeuds)
- déplacement de contenu (noeuds)
- duplication de contenu (noeuds)
- tri de données

Site de référence :

- http://openclassrooms.com/fr/courses/1766341-structurez-vos-donnees-avec-xml
- http://www.w3.org/

Exemple

Exemple Processeur côté serveur : un processeur installé sur le serveur envoie, après traitement le document résultant au client déploiement des servlets de transformation sous tomcat).

Dans l'exemple qui suit, on souhaite transformer un document XML en document HTML :

XML source

```
<personne id="0001">
    <nom>Durand</nom>
    <prenom>Martine</prenom>
</personne>
```

HTML cible

```
<html>
    <body>
        <p><b>Durand</b> Martine</p>
    </body>
</html>
```

XSLT correspondant

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="yes" encoding="ISO-8859-1"/>
<xsl:template match="personne">
    <html>
       <body>
    <p>
       <b><xsl:value-of-select="nom"/></b>
        <xsl:value-of select="prenom"/>
   </p>
       </body>
    </html>
     </xsl:template>
  </xsl:stylesheet>
```

Résumé

XSL est un langage XML permettant de définir des feuilles de style

Les composants d'un document XSL sont :

- XSLT : pour manipuler les documents
- XPath : pour naviguer dans la structure hiérarchique des documents
- XSL Formatting Objects : pour définir la mise en forme des documents

Conclusion

Avantages d'XSLT :

- A partir d'un même document XML, on peut générer plusieurs formats en sortie (HTML, WML...)

Inconvénients d'XSLT :

- Bien que la version 1.0 ait été stabilisée, les outils implémentant XSLT ne respectent pas tous la recommandation (fonctionnalités manquantes, ajout de fonctionnalités et d'éléments propriétaires).

XSLT versus CSS :

- XSLT peut tout à fait être utilisé de manière combinée avec les CSS, en particulier quand la cible est du HTML4 ou du XHTML1.