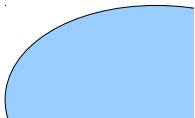
TC7

Technologie XML

Stéphane Bonhomme

stephane@exselt.com



Notion de document

- **« Chose écrite qui sert à renseigner, prouver »**
- **Vient de « docere », instruire.**
- **« ensemble d'informations porteur de sens conçu par un (des) auteur(s) pour un (des) lecteur(s) »**
- **Notion qui se diversifie par :**
  - les types de contenus : textes, images, vidéos, sons
  - les types de support de restitution : papier, terminaux, téléphones
  - les opérations que l'on peut effectuer : lecture/écoute, navigation, annotation, écriture
  - leur utilisation : information, commerce, loisirs, pédagogie

Rôle de la chaîne de publication documentaire

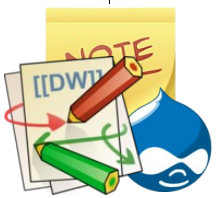
Données Sources Hétérogènes

Documents Rédactionnel

Données en base

Documents Techniques Illustrations

Données média



CMS, Wiki, Blogs...

Variété des publications

Sites Web
Web Services

Adaptation
Au matériel
Pc, tablette
Smarphone
Embarqué

Publication
Papier
Encore présente
Dans de nombreux domaines (normes)

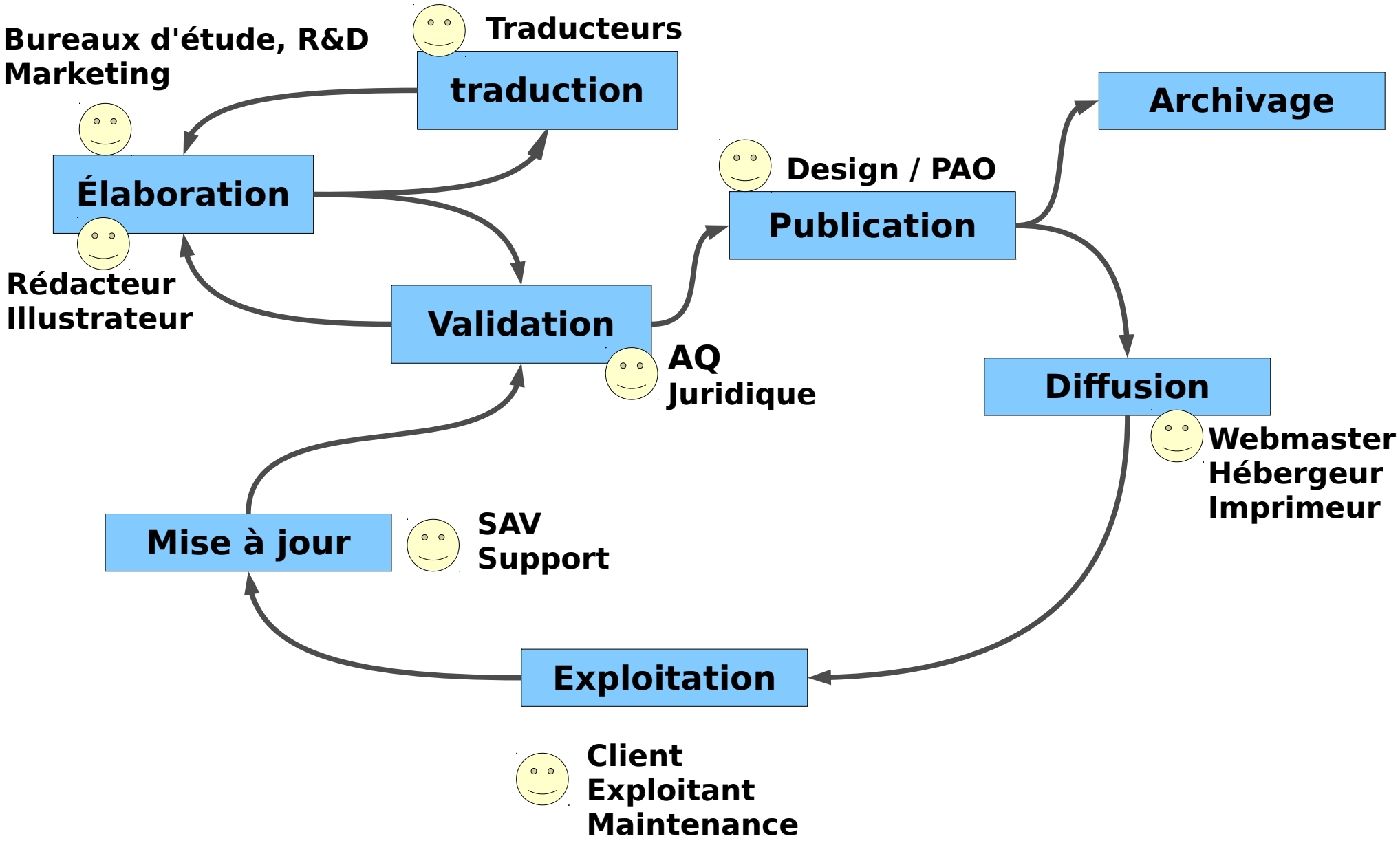
Quelques problématiques

- **Intégrer des données techniques aux documents**
- **Gérer des mises en forme différenciées pour un même contenu (web / papier ...)**
- **Gérer l'intégration de différents média et trouver les stratégies de restitution**
- **Gérer ma dimension temporelle des sources**
- **Volumes industriels, nécessité d'automatisation**
- **Traduction, même processus dans N langues**
- **Travail en équipe, intervenants divers**

Quelques éléments de réponse

- **Utilisation des formats standard et ouverts**
  - Accessibilité
  - Indépendance des plates-formes, interopérabilité
  - Pérennité du stockage
  - Composants logiciels préexistants
- **Briques logicielles de base**
  - Validation, contrôle des documents
  - Parseurs / Sérialiseurs
  - Transformation de formats
  - API d'accès et modification des documents
- **Dissociation du fond et de la forme**
  - Utilisation de documents structurés
  - Utilisation des feuilles de styles

Cycle de vie des documents électroniques



XML apporte

- **Modèle de données structurées**
  - Structuration logique des données
- **Méta-langage**
  - Possibilité de définir des formats métiers
  - Normalisation des formats les plus courants
- **Syntaxe de base universelle**
  - Analyseurs syntaxiques standards
    - Temps de développement réduit
    - Robustesse des applications
    - Indépendance % plateformes
- **Outils de traitement**
  - Mise en forme
  - Transformation, adaptation

Principes de base de XML

- **Séparation contenu, structure, présentation**
- **Syntaxe simple**
  - Adapté aux applications web, ou embarquées
- **Format textuel**
  - Lisible par les humains
- **Balises sans signification *a priori***
  - Ne véhiculent pas d'information de présentation
  - Vocabulaire fixé par la sémantique métier
- **... Autres langages dédiés à la présentation**
  - CSS, XSL
- **... Autres langages dédiés au traitements**
  - XSLT, Xquery
  - Langages « usuels » avec APIs standard DOM, SAX

Un peu d'histoire...

- **1986 : norme ISO SGML**
  - Standard Generalized Markup Language
  - utilisée pour les grands systèmes documentaires (ATA)
  - Et pour décrire la syntaxe des documents HTML

- **1991 : langage HTML**
  - Jeu de balises pour décrire des documents (simples) du web (hypertexte)
  - Balises de formatage dont le rendu est facile
  - Apparition des feuilles de style CSS
    - vers une séparation structure+contenu de la présentation
- **Besoin d'aller plus loin -> XML**

Un peu d'histoire, suite...

- **1998 : recommandation W3C XML 1.0**
  - eXtended Markup Language
  - version du 6 octobre 2000 (errata)
- **Sous-ensemble de SGML**
  - 80% des fonctionnalités de SGML,
  - ... Mais seulement 20% de sa complexité
- **Méta-langage**
  - balises personnalisées
- **Séparer contenu, structure et présentation**
- **Indépendant des plates-formes**

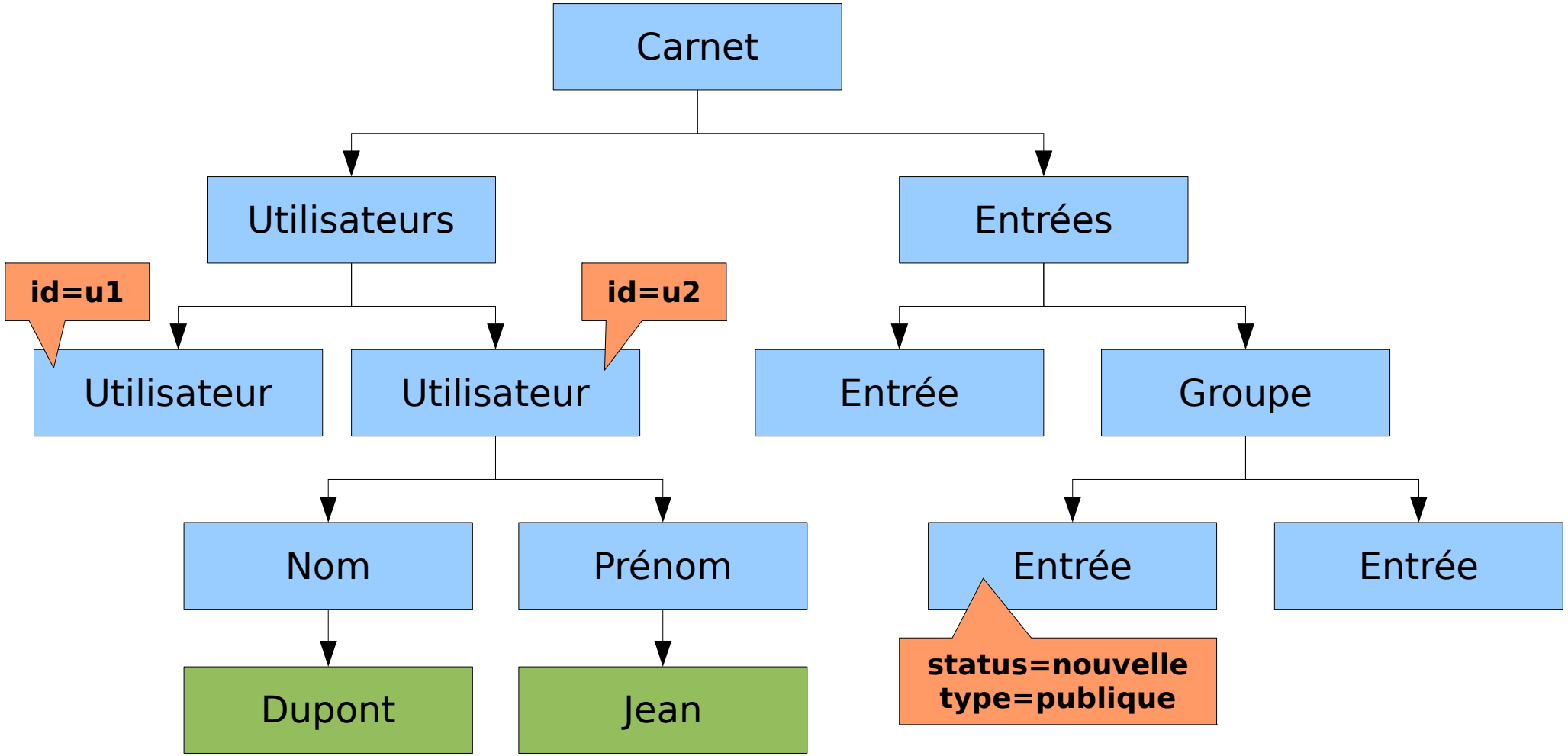
XML : Les concepts

- **Structure logique**
  - Décomposition abstraite de l'information structurée
  - Éléments, contenu, attributs
- **Représentation de la structure logique**
  - Représentation matérialisée de la structure logique
    - Flux de données (réseau)
    - Fichiers
    - Stockage en base de données
  - Balises, entités
- **Schémas**
  - Modélisation des structures logiques

Structures logiques

- **Objectif**
  - décrire l'organisation du contenu sous forme d'une structure hiérarchique selon une « logique métier »
  - … et indépendamment de tout format de restitution et de toute application de traitement.
- **Concepts**
  - Contenu
    - Objets de base non décomposables (texte)
  - Éléments
    - Objets composites hiérarchiques typés
    - Élément racine unique pour un document
  - Attributs
    - Qualification des éléments

Exemple de structure logique



Les éléments

- **Objet central de la structure logique**
- **Un document = une hiérarchie d'éléments**
  - Unicité de l'élément racine
- **Les éléments peuvent contenir**
  - Des éléments fils

    ET/OU
  - Du contenu texte
  - Il peuvent aussi ne pas avoir de contenu (vides)
- **Les éléments portent les attributs**
- **Les éléments sont typés**
  - Nom de l'élément est défini par son type

Les attributs

- **Précisent la sémantique des éléments**
  - Précision sémantique sur les éléments
  - Exemples : langue, statut, sous-type, identifiant, etc.
- **Associe d'un couple *nom=valeur* à l'élément**
- **Un élément peut avoir zéro, un ou plusieurs attributs**
- **Un élément ne peut pas porter plusieurs attributs de même nom**
- **Les attributs d'un même élément n'ont pas d'ordre**

Représentation de la structure logique

- **Langage de balisage**
  - Représentation de la structure logique
  - Forme textuelle
- **Utilisé pour**
  - Matérialisation de la structure logique
    - Fichiers
    - Transmission de données par le réseau
    - Communication inter-composants
- **Outils**
  - Parser
    - Structure physique --> applicatif
  - Serialiseur
    - applicatif --> structure physique

Langage de balisage

- **Chaque élément est représenté par une paire de balises (*tags*) encadrant son contenu**

```
<chapitre>...</chapitre>
```

- **Raccourci pour les éléments vides**

```
<interligne></interligne> <=> <interligne/>
```

- **Les balises ouvrantes portent les attributs**

```
<chapitre version="provisoire" date="16/06/03">...</chapitre>
```

- **L'imbrication et l'ordre des éléments reflètent la structure logique**

```
<ol xml:lang="fr">
   <li>Des balises décrivent la structure</li>
   <li>Structure arborescente</li>
</ol>
```

- **La première balise ouvrant correspond à la dernière balise fermante**

Document XML

Un document XML est composé :

- **D'un prologue** (optionnel)

```
<?xml version='1.0' encoding='UTF-8' standalone='no' ?>
```

  - version : "1.0"
  - encoding : ISO-8859-1 (Latin-1), UTF-8 , UTF-16
    - Valeur par défaut : Unicode UTF-8
  - standalone : le fichier fait appel à des ressources externes

- **D'un type de document** (optionnel)

```
<!DOCTYPE carnet
  PUBLIC "-//xxx/Yyy/en" "http://www..."
  [ .... ]>
```

- **Des balises, attributs et contenus**
  - Contenant toute la structure logique

Autres composants du document

- **Commentaires**

```
<!-- Ceci est un commentaire -->
```

  - Peut s'étendre sur plusieurs lignes
  - Ne doit pas contenir la séquence `--`

- **Des instructions de traitement (PI) :**
  - utilisées pour représenter une information additionnelle à la structure physique
    - Code de traitement embarqué

```
<?php for (i=...) { faire_traitement();} ?>
```

    - Emplacement de la sélection courante

```
<?FM select start ?>
```

Noms XML

- **Objets nommés**
  - Éléments, attributs, PI, ID…
  - Commencent par un caractère alphabétique (lettre) ou `_` `:`
  - Contiennent des caractères alphanumériques `. - _ : ou` caractères de liaison
- **Valides**
  - `gras`, `énorme`, `_note`, `:couleur`, `Titre1`, `m.gras-2`
  - `большой`, `きなカラ` ….
- **Invalides**
  - `2gras`, `.chapitre`, `texte gras`

Les caractères dans les documents XML

- **Encodage du contenu dans le prologue**
- **Certains caractères peuvent être ambigus**

  `< > " ' &`

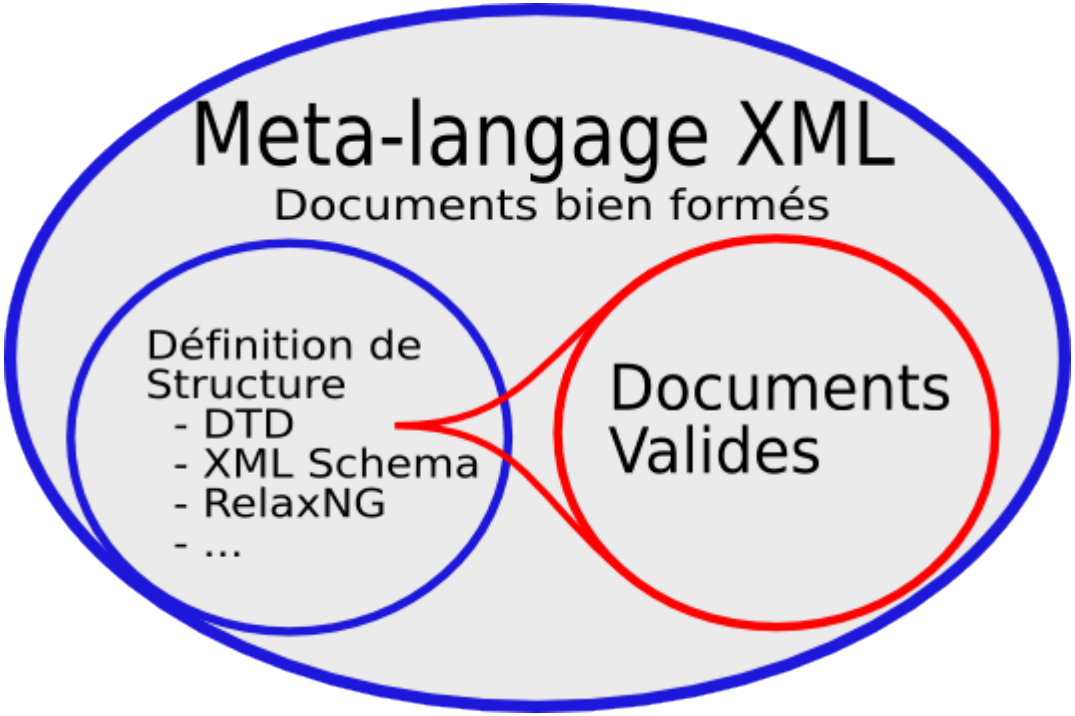
  - *Character entity,* un caractère désigné par son nom **`&alpha;`** Ce nom doit être défini dans le DOCTYPE.
    - *Entités prédéfinies :* **`<`**, **`>`**, **`&quot;`**, **`&apos;`**, **`&`**
  - *Character reference*, le code du caractère en décimal ou en hexa : **`&#945;`** ou **`&#x3b1;`**
- **Un fragment de texte entier peut être échappé dans une section CDATA**
  - Exemple : **`<![CDATA[<greeting>Hello!</greeting>]]>`**
    - **`<greeting>`** et **`</greeting>`** sont considérées comme du contenu, pas comme des balises
    - Utile pour inclure du javascript dans une page xml

Documents Bien Formés

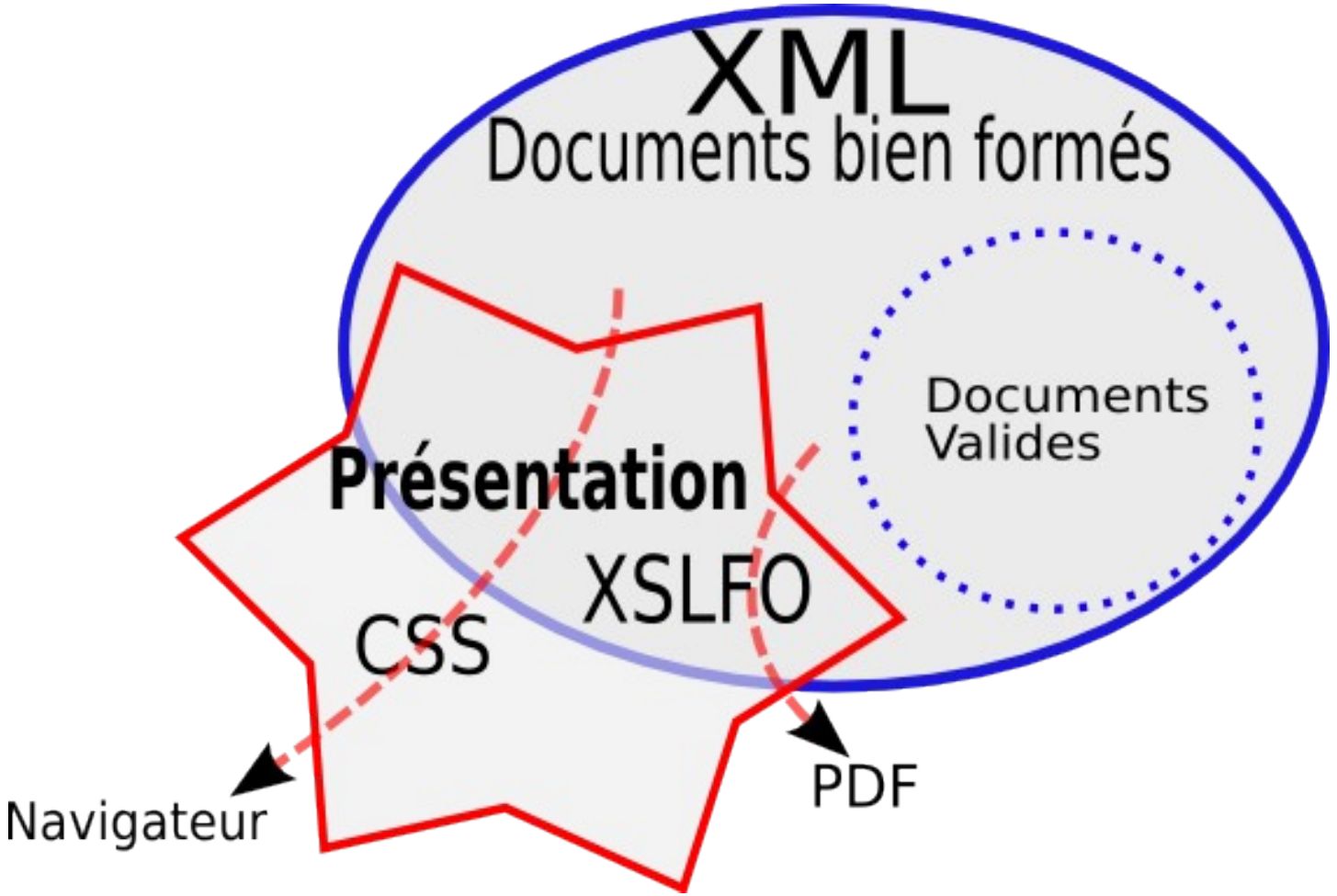
- **Respectent les règles syntaxiques de base de XML**
- **Tout document XML doit être au minimum bien formé**
- **Les parsers XML relèvent une erreur si les données ne sont pas bien formés**
- **Mais pas de contrôle possible sur :**
  - Quelles balises sont présentes dans un document
  - Quels attributs sont admis dans ces balises
  - Quels balises peuvent contenir du texte
  - Quelles sont les sous-éléments possibles dans un élément donné
- **Besoin de langages de modélisation des documents XML**

Le monde xml : Données Structurées

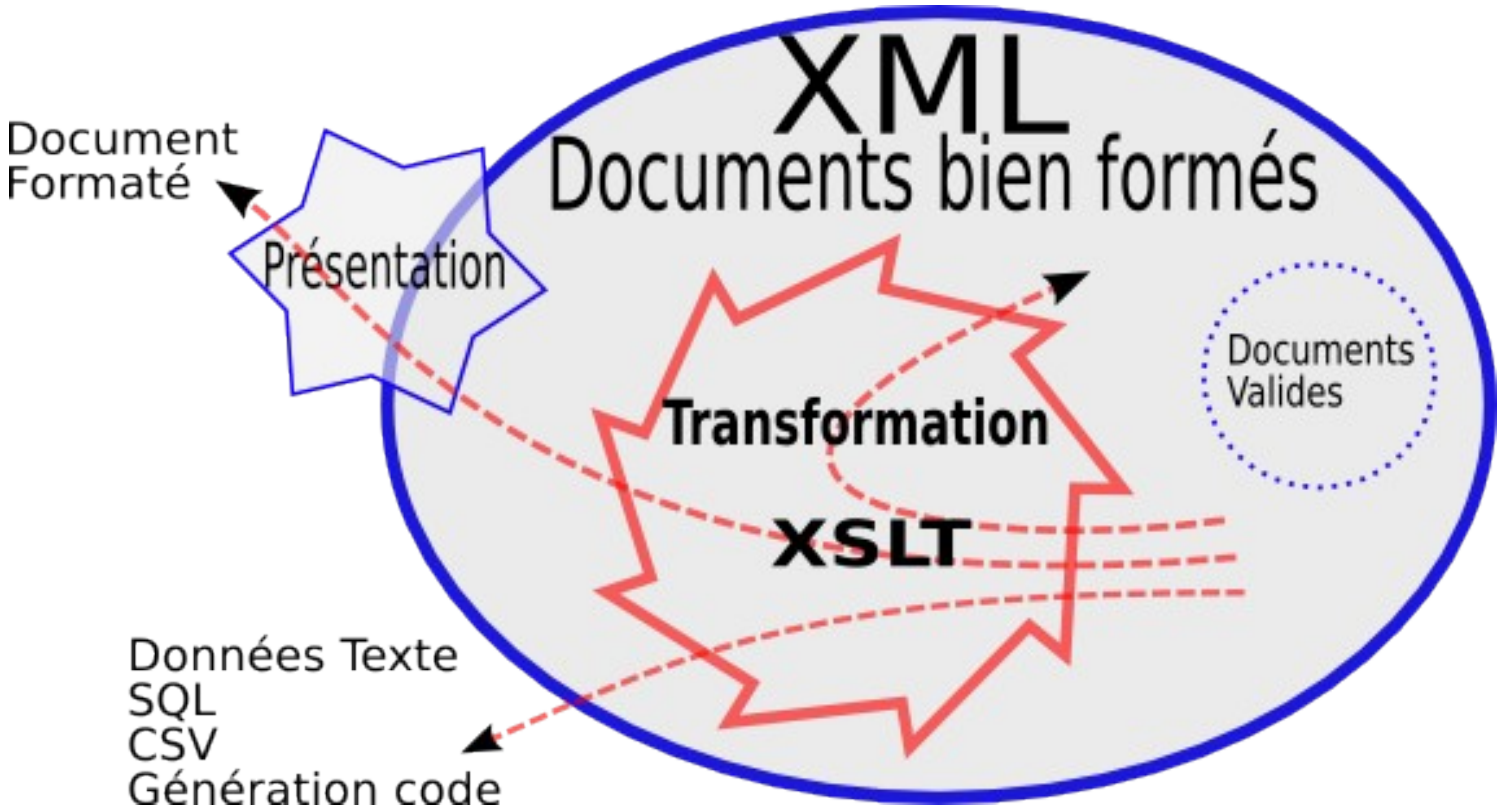
Le monde xml : Méta-langage



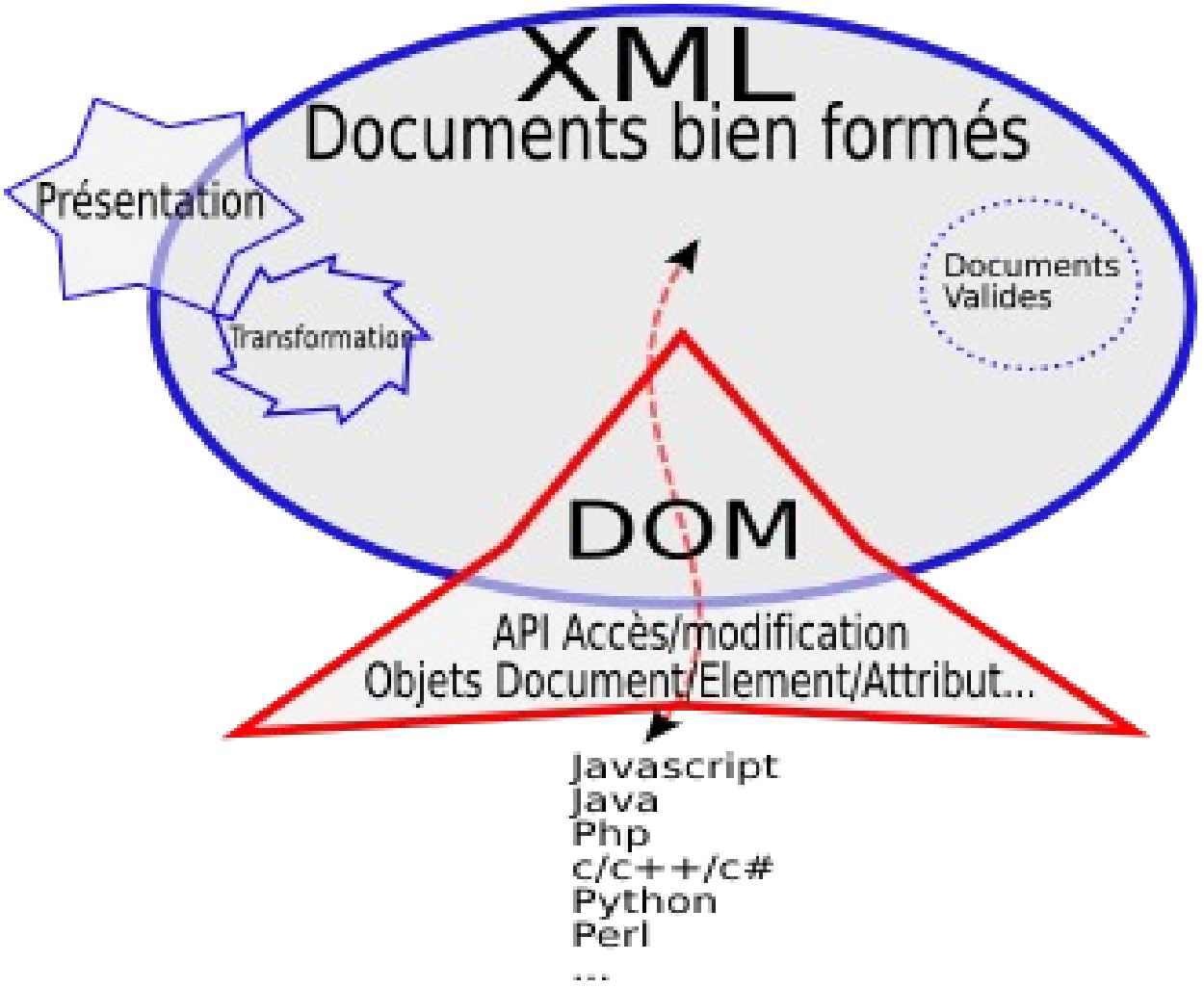
Le monde xml : Présentation



Le monde XML : transformations



Le monde xml : API programmation



Des instances aux modèles

- **Langages pour définir des structures communes à un ensemble d'instances**
  - Métalangage : DTD, XML Schema
  - Structuration selon des vocabulaires « métier »
  - Modélisation de l'organisation logique
- ► **intérêt :**
  - Validation des instances
    - Processus de développement en //
    - Contrat Fournisseur / consommateur d'information
  - Traitements homogènes, indexation, saisie adaptée
  - Réduit la quantité de code nécessaire pour accéder aux éléments

Langages de définitions de modèles

- **DTD, l'historique**
  - Hérité de SGML
  - Syntaxe spécifique
- **XML Schema, le plus répandu**
  - Plus de souplesse
    - types explicites, types de base plus fins
  - Syntaxe XML pour la description
- **RelaxNG, l'alternative**
  - Syntaxe XML
- **Quelques autres, plus confidentiels**
  - Schématron

Les DTD : Document Type Definition

- **Format texte**
- **Ensemble de règles**
  - sur une ou plusieurs lignes
  - débutent par **<!ELEMENT**, **<!ATTLIST** ou **<!ENTITY**
  - terminent par **>**
- **Incluses dans le document XML**
  - Rare ! Principalement déclaration d'entités
- **Dans un fichier externe**
  - Référencé depuis les documents
  - Ré-utilisation
  - Accessible au travers :
    - Du système de fichiers : catalogues de DTD
    - Du réseau : référence par URL

Référence à une DTD dans un document XML

- **Mot-clé : `<!DOCTYPE racine ... >`**
- **Inclusion dans le prologue**
  - le code de la dtd est directement écrit dans le prologue

```
<!DOCTYPE Carnet [
  Règles de déclarations des éléments,
                    attributs, entités
 ]>
```

  - ou bien, il est référencé dans le prologue

```
<!DOCTYPE Carnet
     PUBLIC "-//Exselt/DTD Carnet/en" "Carnet.dtd" >
```

Identifiant public de la DTD

Adresse physique de la DTD (url)

Document Type Definition

- **Ensemble de règles permettant de définir**
  - les balises autorisées dans le document, leur contenu

```
<!ELEMENT Carnet (Utilisateur|Entrées) >
```

  - les attributs associés à une balise, ainsi que leur type

```
<!ATTLIST Société idname CDATA #IMPLIED
                 siret  CDATA #REQUIRED >
```

  - les entités autorisées (macros)

```
<!ENTITY Email "moi@societe.fr" >
```

Définition d'éléments

- **Règle élément définit**
  - Le nom de l'élément
  - Son modèle de contenu
    - Ce que l'élément peut contenir
      - Éléments fils
      - Contenu texte seulement
      - Contenu mixte
- **Modèles de contenu**
  - éléments fils
  - élément vide
  - contenu libre
  - données (texte)
  - contenu mixte

```
<!ELEMENT Carnet      (personne|société)+>
<!ELEMENT Rôle         EMPTY>
<!ELEMENT Complément   ANY>
<!ELEMENT Nom         (#PCDATA)>
<!ELEMENT Paragraphe  (#PCDATA|gras|ital)*>
```

Modèle de contenu

- **Pour décrire les contenus mixtes/fils d'un élément, on définit une expression de composition des éléments fils**

- **Opérateurs de composition**
  - séquence (**,**)
  - alternative (**|**) :
  - occurence
    - ***** : 0 à n occurences
    - **+** : au moins 1 occurrence
    - **?** : 0 ou 1 occurence
  - regroupement de fragments d'expression **( )**

```
(Nom, Prénom, Adresse, Email)
```

```
(Personne | Société)
```

```
(Adresse, LigneAdresse*)+
```

```
(Nom, Prénom?, Email?)
```

Définition d'attributs

- **Règle de définition d'attributs**
  - nom de l'élément
  - Suite de tuples
    - nom attribut
    - type attribut ou valeurs énumérées
    - indicateur d'occurence ou valeur par défaut

```
<!ATTLIST Société xml:lang  NMTOKEN          #IMPLIED
                  idname    ID               #IMPLIED
                  type      (SA|SARL|EURL) 'SA'  >
```

Types d'attributs (1)

- **CDATA chaîne de caractère littérale**

```
<!ATTLIST soc name CDATA #IMPLIED>
```

- **ID, IDREF renvois à l'intérieur des documents**

```
<!ATTLIST soc idname ID #REQUIRED>
```

- **ENUMÉRÉ l'ensemble des valeurs possibles de l'attribut est défini**

```
<!ATTLIST personne fonction (ing|com|tech|admin) "ing">
```

Types d'attributs (2)

- **ENTITY , ENTITIES référence à une ou plusieurs entités externes non XML**

```
<!ATTLIST personne photo ENTITY #IMPLIED>
```

- **NMTOKEN , NMTOKENS 1 ou + noms symboliques.**

```
<!ATTLIST personne compet NMTOKEN default "xml">
```

- **NOTATION (obsolète) indique que le contenu de l'élément doit être passé à l'application identifiée par la notation**

```
<!NOTATION vcard system "/usr/bin/ns">
<!ELEMENT vcard-buffer any>
<!ATTLIST vcard-buffer vc NOTATION (vcard) #IMPLIED>
```

Indicateur d'occurence

- **#REQUIRED** **l'attribut est obligatoire**
- **#IMPLIED** **l'attribut est optionnel**
- **#FIXED "valeur"** **la valeur de l'attribut est fixée**
  - l'attribut est implicite pour toutes les balises pour lequel il est déclaré
- **"valeur"** **valeur par défaut**

```
<!ATTLIST personne fonction (ing|com|tech|admin) "ing">
<!ATTLIST soc idname ID #REQUIRED>
<!ATTLIST soc name CDATA #IMPLIED>
<!ATTLIST soc pays CDATA #FIXED "France">
```

Entités

- **Servent à référencer d'autres objets**
- **Raccourcis syntaxiques (remplacement)**
  - Dans la DTD
    - entités paramètre
  - Dans le corps du document
    - caractères
    - entités internes
- **Liaison d'objets externes**
  - Fragments XML
    - sous-documents
  - Objets non XML
    - références à des images, du contenu multimedia, etc

Entités internes

- **Entités de caractères**
  - but : représenter les caractères spéciaux

- **Entités générales internes (&)**
  - but : nommer des expressions pour les réutiliser
  - déclarée dans la DTD
  - utilisée dans le document

```
<!ENTITY mail "@domaine.com">
<bal>sbonhome&mail;</bal>
```

- **Entités paramètre (%)**
  - idem pour les DTDs
  - déclarée dans la DTD
  - utilisée dans la DTD

```
<!ENTITY % idt "(nom,prenom,age?)">
<!ELEMENT personne (%idt,bal,soc)>
```

Entités externes

- **Entités XML externes**
  - référence des documents XML externes

```
<!ENTITY moncv SYSTEM "/home/bonhomme/cv.xml">
```

- **Entités générale externes**
  - référence des documents non XML (binaires)

```
<!ENTITY maphoto SYSTEM "/home/bonhomme/photo.jpg" NDATA
jpeg>
```

Identification de ressource externes

- **Une entité externe peut être identifiée :**
  - par une URL
    - *ressources privés, mot clé* **SYSTEM**

```
<!ENTITY maphoto SYSTEM "/home/bonhomme/photo.jpg>
<!ENTITY mapage SYSTEM "http://www.serveur.com/sbe">
```

  - par une FPI (rare et obsolète)
    - *identifiant public, ressources partagées, mot clé* **PUBLIC**

```
<!ENTITY rec-XML PUBLIC "-//W3C//DOCUMENT Recommandation
open - The recommandation for XML 1.0//EN"
"http://www.w3.org/TR/1998/REC-xml-19980210.xml">
```

    - Toujours suivie d'un identifiant systeme (fallback)

Règles de conception d'une DTD

- **Déterminer les informations à modéliser**
- **Pour chacune, choisir sa forme : élément ou attribut**
  - pas de règle…mais
    - information complexe -> élément (attribut = chaîne simple)
    - plusieurs données -> élément (attribut = 1 valeur)
  - en général
    - données -> éléments
    - métadonnées -> attributs

XML Schema

- **Définition de vocabulaires de marquage partagés**
- **Définition de la structure, du contenu et de la sémantique des documents XML**
- **Meilleure intégration des XML Namespaces pour permettre des mécanismes d'héritage**
- **Application de schémas localement à des parties de document**
- **Homogénéité syntaxique : les XML Schemas sont des documents XML**
- **XML Schema 1.0 : W3C Recommendation 2 mai 2001**

Namespaces XML

- **Conflits de noms possibles lors de l'utilisation de plusieurs DTD/schemas dans un même document**
  - sémantique différente : comment interpréter la balise ?
  - construction différente : comment valider le document ?
  - Ex : l'élément `<input>` est défini par les schémas XUL et xhtml.
- **Principe**
  - déclarer un identifiant relatif à la DTD ou le schéma
  - identifiant = URI
  - définir un préfixe local au document associé à l'identifiant
  - préfixer les noms des balises appartenant au schéme

Namespaces XML : exemple

```
<?xml version="1.0">
<lettre xmlns="urn:letter"
   xmlns:addr="http://www.exemple.com/adresse">
  <expéditeur>
    <name>Stéphane Bonhomme</name>
    <addr:adresse siege-soc="yes">
      <addr:ligne>5, rue du moulin</addr:ligne>
      <addr:cp>38000</addr:cp>
      <addr:ville>Grenoble</addr:ville>
    </addr:adresse>
  </expéditeur>
  <!-- contenu supprimé-->
</lettre>
```

Instances, DTD et Schémas

- **Un exemple dans le domaine des ressources biblio**
  - 1. Balisage XML d'un catalogue de livres : BookCatalogue.xml
  - 2. DTD correspondante : Bookcatalogue.dtd
  - 3. Schémas XML
    - du catalogue
    - et des numéros ISBN
- **Validation de l'instance avec une DTD:**
  - xmllint --dtdvalid Bookcatalogue.dtd BookCatalogue.xml
- **Validation de l'instance avec un schema XML:**
  - xmllint --schema Bookcatalogue.dtd BookCatalogue.xml

Instance de ressources bibliographiques

```
<?xml version="1.0" encoding="UTF-8"?>
<BookCatalogue>
        <Book>
           <Title>The First and Last Freedom</Title>
           <Author>J. Krishnamurti</Author>
           <Date>1954</Date>
           <ISBN>0-06-064831-7</ISBN>
           <Publisher>Harper &amp;amp; Row</Publisher>
        </Book>
        <Book>
           <Title>Illusions The Adventures of a Reluctant Messiah</Title>
           <Author>Richard Bach</Author>
           <Date>1977</Date>
           <ISBN>0-440-34319-4</ISBN>
           <Publisher>Dell Publishing Co.</Publisher>
        </Book>
        <Book>
           <Title>Document numérique 3-4 (2002) Unicode, écriture du
                   monde ?</Title>
           <Author>J. André & H Hudrisier</Author>
           <Date>2002</Date>
           <ISBN>2-7462-0594-7</ISBN>
           <Publisher>Hermès Science Publications</Publisher>
        </Book>
</BookCatalogue>
```

DTD des ressources bibliographiques

```
<!ELEMENT BookCatalogue  (Book*)>
<!ELEMENT Book
            (Title, Author, Date, ISBN,
  Publisher)>
<!ELEMENT Title      (#PCDATA)>
<!ELEMENT Author     (#PCDATA)>
<!ELEMENT ISBN       (#PCDATA)>
<!ELEMENT Date       (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

Schéma XML des ressources biblio

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.publishing.org"
   xmlns="http://www.publishing.org" elementFormDefault="qualified">
    <xsd:include schemaLocation="isbn.xsd"/>
    <xsd:element name="BookCatalogue">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Book" minOccurs="1" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Title" type="xsd:string"/>
                            <xsd:element name="Author" type="xsd:string"/>
                            <xsd:element name="Date" type="xsd:string"/>
                            <xsd:element name="ISBN" type="ISBN-type"/>
                            <xsd:element name="Publisher" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Schéma XML des numéros isbn

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:simpleType name="ISBN-type">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="0-[0-1][0-9]-\d{6}-[0-9x]">
                <xsd:annotation>
                    <!-- ISBN definition for English-speaking Countries -->
                    <xsd:documentation>
                     group/country ID = 0 (hyphen after the 1st digit)
                     Publisher ID = 00...19 (hyphen after the 3rd digit)
                     Block size = 1,000,000 (requires 6 digits)
                     check digit is 0-9 or 'x'
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:pattern>

            <xsd:pattern value="2-\d([0-9]|-){7}\d-[0-9x]">
                <xsd:annotation>
                    <!-- ISBN definition for French-speaking Countries -->
                    <xsd:documentation>
                     group/country ID = 2 (hyphen after the 1st digit)
                     Country = France, Belgium, Canada, Switzerland
                     check digit is 0-9 or 'x'
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:pattern>
           ...
```

Syntaxe Xml Schema

- **Document XML**
  - Namespace **`http://www.w3.org/1999/XMLSchema`**
- **Élément racine `schema`**
- **Définitions de types :**
  - types complexes : pour les modèles de contenus
  - types simples : pour le contenu texte des élements et attributs
- **Définition d'éléments et d'attributs**
  - Par référence à leur type
- **Extensions de type**
- **Modularité**

Exemple

```
<schema xmlns="http://www.w3.org/1999/XMLSchema"
        targetnamespace="urn:carnet"
        xmlns:ca="urn:carnet">

 <simpleType name="fiche" type="string"/>

 <element name="carnet">
  <complexType>
   <sequence maxOccurs="unbounded">
    <element type="ca:fiche"/>
   </sequence>
  </complexType>
 </element>
</schema>
```

Définition de types simple

- **Balise simpleType**
- **Types simples prédéfinis (namespace de XML Schema)**
- **Types simples construits**
  - Par référence à un type simple existant
  - Par restriction en utilisant les facettes
    - Chaque type simple possède des facettes permettant de contraindre les valeurs :
      - string : pattern, enumeration, length, maxlength,...
      - integer : pattern, enumeration, precision, minInclusive, maxInclusive...
    - On peut contraindre les type en combinant plusieurs facettes
      - Union des valeurs pour les facettes **`pattern`** et **`enumeration`**
      - Intersection des valeurs pour les autres facettes

Exemple

```
<simpleType name="année">
 <restriction base="integer">
     <minInclusive value="1900"/>
     <maxInclusive value="2100"/>
  </restriction>
</simpleType>

<simpleType name="numTel">
  <restriction base="string">
    <pattern value="([0-9]{2} ){4}[0-9]{2}"/>
    <pattern value="+[0-9]{2} [0-9]+"/>
  </restiction>
</simpleType>
```

Exemple

```
<simpleType name="couleur">
  <restriction base="string">
    <enumeration value="bleu"/>
    <enumeration value="rouge"/>
    <enumeration value="vert"/>
    <enumeration value="jaune"/>
  </restiction>
</simpleType>
```

Types simples prédéfinis

string — Type chaine

- Types dérivés prédéfinis : normalizedString, token, language, Name, NCName, ID, IDREF, ENTITY...

decimal — Type nombres décimaux

- Types dérivés prédéfinis : integer, nonPositiveInteger, negativeInteger, long, int, short, byte, nonNegativeInteger, positiveInteger, unsignedLong, unsignedInt, unsignedShort, unsignedByte

boolean, float, double — Autres types numériques

duration — Type durée

- Types dérivés prédéfinis : dateTime, date, time, gMonth, gYear, gYearMonth, gDay, gMonthDay

base64Binary, hexBinary — Types binaires

qname, anyURI, NOTATION — Types autres

Définition de types complexes

- **Définissent un modèle de contenu**
- **Élément complexType, contient :**
  - les définitions d'attributs
  - le constructeur de type
    - **`sequence`**, **`all`** ou **`choice`**
- **Les constructeurs peuvent porter les attributs**
  - **`minOccurs`**, **`maxOccurs`** : permettent de fixer la cardinalité
    - Les domaines de valeurs et les valeurs par défault dépendent du constructeur
- **Les constructeurs contiennent :**
  - Des définitions de sous éléments
  - Des constructeurs imbriqués

Constructeur sequence

- **Liste ordonnée d'éléments**
- **Attributs**
  - **`minOccurs`** : entier non négatif (par défaut 1)
  - **`maxOccurs`** : entier non négatif ou **unbounded** (par défaut 1)
- **Exemple**

```
<complexType name="groupe">
  <sequence minOccurs="2" maxOccurs="10">
    <element name="nom" type="text"/>
    <element name="prenom" type="text"/>
  </sequence>
</complexType>
```

Définit une liste de 2 à 10 tuples (nom, prénom)

Constructeur choice

- **Alternative entre plusieurs éléments**
- **Attributs**
  - **`minOccurs`** : entier non négatif (par défaut 1)
  - **`maxOccurs`** : entier non négatif ou **unbounded** (par défaut 1)
- **Exemple**

```
<complexType name="type-société">
  <choice>
    <element name="eurl" type="text"/>
    <element name="sarl" type="text"/>
    <element name="sa" type="text"/>
  </choice>
</complexType>
```

Constructeur all

- **Défini un groupe non ordonné**
- **Attributs**
  - **`minOccurs`** : 0 ou 1 (par défaut 1)
  - **`maxOccurs`** : toujours 1
- **Ne peut contenir que des définitions de sous éléments**
- **Exemple**

```
<complexType name="caracteristiques">
  <all>
    <element name="longueur" type="integer"/>
    <element name="largeur" type="integer"/>
    <element name="hauteur" type="integer"/>
  </all>
</complexType>
```

Définitions d'éléments

- **Par référence à un type nommé**

```
<element name="adresse" type="ns:type-adresse"/>
```

  - Type simple => element contient du texte
  - Type complexe => element composite

- **Par utilisation d'un type anonyme**

```
<element name="adresse">
  <complexType>
    <sequence>
        <element name="nom" type="text"/>
        <element name="voie" type="text"/>
        <element name="codepostal" type="text"/>
        <element name="ville" type="text"/>
   </sequence>
  </complexType>
</element>
```

Définitions imbriquées

- **En utilisant les types anonymes, on peut imbriquer les définitions**

```
<element name="carnet">
  <complexType>
    <sequence>
      <element name="adresse">
        <complexType>
          <all>
            <element name="nom" type="text"/>
          </all>
         </complex-type>
       </element>
    </sequence>
  </complexType>
</element>
```

Déclaration d'attributs

- **Élément `attribute`, fils de `complexType`**
- **Attributs :**
  - **`type`** : référence un simpleType
  - **`use`** : valeurs **`required`, `optional`** ou **`prohibited`**
  - **`default`** : valeur par défaut, peut être surchargée
  - **`fixed`** : valeur implicite (attribut prohibited)

- Les attributs peuvent être construits :
  - par référence à un type simple, avec l'attribut type
  - avec un type simple anonyme, contenu dans la balise **`attribute`**

Définition d'attributs : exemple

```
<complexType name="type-adresse">
   <attribute name="numindex" type="integer" use="required"/>
   <attribute name="typeaddr" use="optional"
  default="personnelle">
      <simpleType>
        <restriction base="string">
           <enumeration value="personnelle"/>
           <enumeration value="professionnelle"/>
        </restriction>
      </simpleType>
    </attribute>
    <!-- modele de contenu -->
</complexType>
```

Schemas et namespaces

- **Dans le schema**
  - Attribut **`targetNamespace`**
    - Définit le namespace des objets décrits par le schema
  - Déclaration de namespace :
    - NS de XML schema
    - NS des objets décrits par le schéma si on veut les réutiliser
- **Dans l'instance XML**
  - Déclaration du namespace des objets
    - tel que le définit le **`targetNamespace`** du schéma
  - Liaison avec le schéma
    - Namespace **`http://www.w3.org/2001/XMLSchema-instance`**
    - Attribut **`schemaLocation`** ou **`noNamespaceSchemaLocation`**
      - associe un namespace (uri) avec la ressource schema (url)

Namespaces du schema (1)

```
<schema xmlns="http://www.w3.org/1999/XMLSchema"
        targetnamespace="urn:carnet"
        xmlns:ca="urn:carnet">
 <simpleType name="fiche" type="string"/>
 <element name="carnet">
  <complexType>
   <sequence maxOccurs="unbounded">
    <element type="ca:fiche"/>
   </sequence>
  </complexType>
 </element>
</schema>
```

Namespaces du schema (2)

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema"
        targetnamespace="urn:carnet"
        xmlns="urn:carnet">
 <xsd:simpleType name="fiche" type="xsd:string"/>
 <xsd:element name="carnet">
  <xsd:complexType>
   <xsd:sequence maxOccurs="unbounded">
    <xsd:element type="fiche"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Namespaces de l'instance

```
<carnet xmlns="urn:carnet"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:carnet  carnet.xsd">

   <fiche>
     ....
   </fiche>
</carnet>
```

Modularisation des schemas

- **Élément include**
  - Permet d'importer les composantes d'un schéma dans un autre
  - Tous les objets apprtiennent au namespace du schema incluant

```
<include schemaLocation="carnet.xsd"/>
```

- **Élément import**
  - Permet d'importer un schéma en conservant son namespace

Modularisation : Exemple xsd:import

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema"
            targetnamespace="urn:annuaire"
            xmlns:ca="urn:carnet"
            xmlns="urn:annuaire">
 <xsd:import schemaLocation="carnet.xsd"
              namespace="urn:carnet"/>
 <xsd:element name="annuaire">
  <xsd:complexType>
   <xsd:sequence maxOccurs="unbounded">
    <xsd:element type="ca:carnet"/>
```

Instance avec schemas modulaires

```
<annuaire xmlns="urn:annuaire"
          xmlns:ca="urn:carnet"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:annuaire  annuaire.xsd">
   <ca:carnet>
     ....
   </ca:carnet>
</annuaire>
```

Ressources en ligne

- **Transparents : http://www.brics.dk/~amoeller/XML/**
- **XML syntaxe annotée :**
  - Standard annoté : http://www.xml.com/axml/axml.html
  - Standard en français : http://babel.alis.com/web_ml/xml
  - Ressources : http://www.oasis-open.org/cover/xml.html
- **Exemples de DTD**
  - XHTML: http://www.w3.org/TR/xhtml1/
  - docbook: http://www.oasis-open.org/docbook/

Conclusion

- **Avantages des schémas XML sur les DTD**
  - Grand nombre de types offerts, possibilité d'ajouter des contraintes sur les types fournis
  - Possibilité de créer de nouveaux types
  - Héritage de types
  - Support des espaces de noms
  - Indicateurs d'occurences plus riches
  - Meilleur contrôle des contenus mixtes
  - Modularité
  - Documentation