

15/04/2023

Version 3



XML

JÉRÉMY PERROUAULT

A decorative graphic on the left side of the slide consisting of two parallel, wavy vertical lines. The inner line is a light blue color, and the outer line is white. They start from the top left and extend towards the bottom left, creating a stylized, organic shape.

XML

SON HISTOIRE

L'HISTOIRE DE XML

- Les débuts d'Internet marquent la nécessité croissante d'échanger des données
 - Souvent dans des formats différents, non standardisés
- **SGML** (**S**tandard **G**eneralized **M**arkup **L**anguage) est né
 - Permet de décrire, à l'aide de **balises**, des informations
 - Langage de description

L'HISTOIRE DE XML

- Mais **SGML** est limité ...
 - Limité en nombre d'éléments (balises)
 - Manque de structure sémantique, il n'y a pas de notion de contenu
- L'idée est simple
 - On garde que les avantages de **SGML** et on améliore le reste
 - Tout en restant compatible avec **SGML**, on va pouvoir étendre un langage et le structurer
 - Il pourra être utilisé pour :
 - L'échange de données (son rôle premier)
 - L'évolution de **HTML** en **XHTML**
 - ...
 - Il ne suffit que d'un parseur **XML** !

L'HISTOIRE DE XML

- Dérivé du **SGML**, **XML** (e**X**tensible **M**arkup **L**anguage) est ... extensible !
 - Ajouter du vocabulaire et de la grammaire via des espaces de nom (comme **XHTML**, **SVG**, **RSS**, ...)

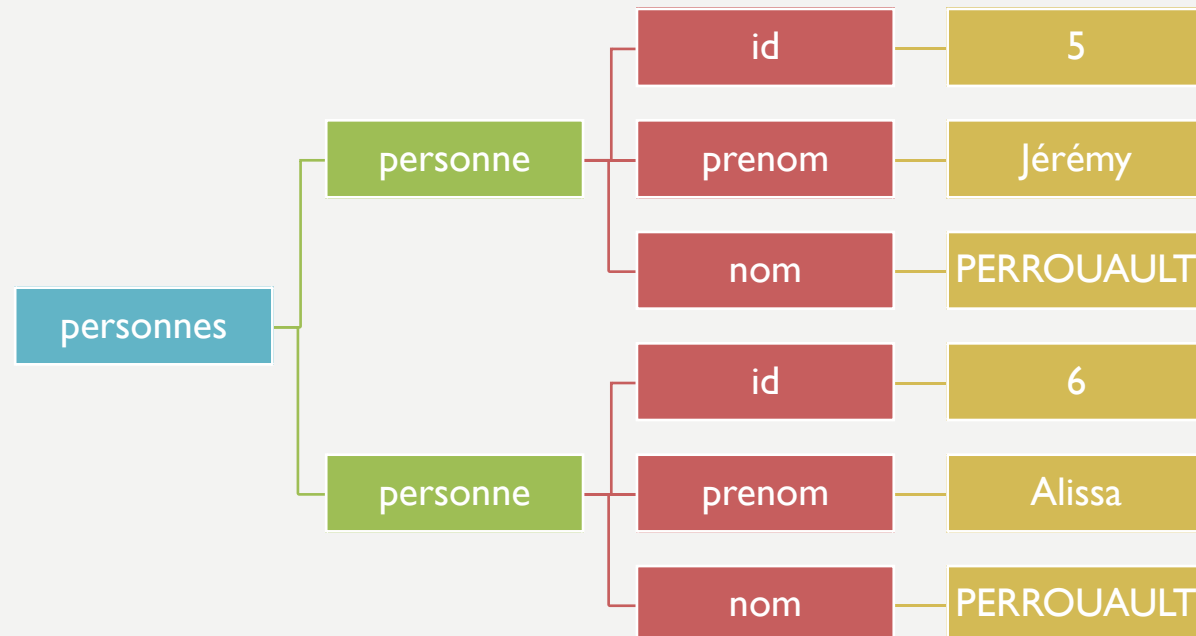
A decorative graphic on the left side of the slide consisting of two parallel, wavy vertical lines. The inner line is a light blue color, and the outer line is white. They start from the top left and extend towards the bottom left, creating a stylized, organic shape.

XML

SON FORMALISE

FORMALISME

- L'information étant structurée grâce au **XML**
 - On peut la représenter sous la forme d'un arbre
 - Avec **une** racine (élément principal) et des branches (sous-éléments) sur plusieurs niveaux



FORMALISME

- Utilisation de
 - Chevrons (< >) qui encadrent les **balises**
 - Une **balise** d'ouverture, une **balise** de fermeture

```
<balise></balise>
```

- **Balises** qui peuvent accueillir des **attributs**
 - Un peu comme des options
 - **Attributs** qui définissent une valeur

```
<balise attribut="valeur" autre="nouvelle valeur"></balise>
```

- Chaque **balise** peut avoir un contenu
 - Qui peut être du texte, une valeur numérique
 - ... ou d'autres balises !

```
<balise>Le contenu de la balise</balise>
```

```
<balise>  
  <autre-balise>Le contenu de la balise</autre-balise>  
</balise>
```


FORMALISME

- Il existe aussi des **balises** *auto-fermantes* (ou **balise** unique)
 - Elles peuvent contenir des **attributs**

```
<personne id="5" nom="PERROUAULT" prenom="Jérémy" />
```

- Il est possible de placer des commentaires

```
<!-- Mon commentaire -->  
<balise></balise>
```

FORMALISME

- **XML** fonctionne comme une poupée russe
 - Chaque poupée peut contenir une autre poupée
- Chaque **balise** ouverte doit être fermée
- Une **balise** qui englobe d'autres **balises** sera fermée **après** chaque **balise** qu'elle contient



```
<personne id="5">  
  <nom>PERROUAULT</nom>  
  <prenom>Jérémy</prenom>  
</personne>
```

FORMALISME

- Textes et autres valeurs : comment choisir entre un **attribut** et une **balise** ?
 - Généralement, on utilise un **attribut** dans les cas suivants
 - Le contenu n'est qu'une simple valeur
 - On cherche à réduire le fichier **XML**

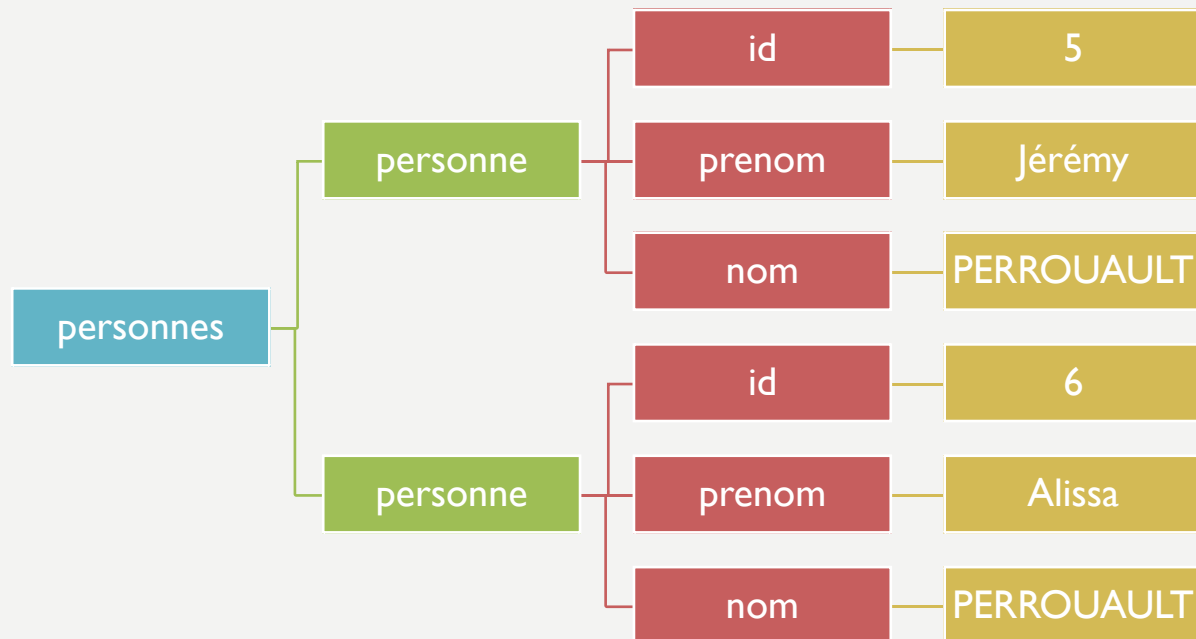
FORMALISME

- Les documents **XML** doivent préciser leur nature en en-tête, dans lequel on précise
 - La version (1.0 ou 1.1)
 - Le jeu de caractères (l'encodage)
 - Si le document est autonome ou s'il a un document qui lui est attaché

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

FORMALISME – EXERCICE

- Reprenons le schéma précédent
 - Le traduire en **XML**





DTD

LES DOCUMENTS DE DÉFINITION

DOCUMENT DE DÉFINITION

- Un document de définition, **DTD** (**D**ocument **T**ype **D**efinition)
 - Ensemble de règles qu'on impose au document **XML**
 - Si le document ne respecte pas ces règles, on dit qu'il est mal formé (*not well-formed*)
 - Un document est valide lorsqu'il est *well-formed* et qu'il est conforme à une définition
 - Cette **DTD** garantit que le document **XML** obéi bien au vocabulaire

DOCUMENT DE DÉFINITION

- Le **DTD** précise le vocabulaire disponible pour le document **XML**
 - Les **balises** disponibles

```
<!ELEMENT balise>
```

- Les **balises** pouvant être contenues dans d'autres **balises**

```
<!ELEMENT balise (autre-balise-1, autre-balise-2)>
```

- Le type de contenu possible dans une **balise**

```
<!ELEMENT balise (TypeContenu)>
```

- Les **attributs** des **balises**

```
<!ATTLIST balise attribut TypeAttr TypeDef ValeurDefaut>
```


DOCUMENT DE DÉFINITION

Contenu	Description
A?	0 ou 1 balise A
A+	1 ou n balises A
A*	0, 1 ou n balises A
A B	Balise A OU balise B
A,B	Balise A ET balise B
(A,B)	Les parenthèses permettent de regrouper
#PCDATA	Valeur alpha-numérique
EMPTY	Balise vide

```
<!ELEMENT personnes (personne)+>
<!ELEMENT personne (nom, prenom, age*)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT age (#PCDATA)>
```

DOCUMENT DE DÉFINITION

TypeAttr	Description
ID	La valeur doit être unique dans le XML
IDREF	Doit correspondre à 1 ID dans le XML
IDREFS	Doit correspondre à 1 ou n ID dans le XML
(A B C ...)	Liste de choix
CDATA	Texte 'Character Data'
NMTOKEN	Un seul mot

```
<!ATTLIST personne id ID #REQUIRED>
```

TypeDef	Description
#IMPLIED	Attribut en option
#REQUIRED	Attribut obligatoire
#FIXED Value	Attribut à valeur fixe

DOCUMENT DE DÉFINITION

- Dans un document **XML**, le **DTD** se déclare dans l'en-tête :

```
<!DOCTYPE RACINE TYPE "URL">
```

– RACINE

```
<!DOCTYPE personnes SYSTEM "les-personnes.dtd">
```

– TYPE

- [] Entre les crochets viennent la **DTD** « interne »
- PUBLIC **DTD** officiellement publiée
- SYSTEM **DTD** non officiel, fait « maison » qui peut être sur le système ou sur Internet

– URL

- URL relative ou absolue du **DTD**

NOTE : Dans le cas d'un **DTD** externe, le document **XML** n'est plus autonome

EXERCICE

- Créer le **DTD** associé au document **XML** des personnes
 - **Attributs** obligatoires
 - id
 - **Attributs** optionnels
 - nom, prenom, age
 - **Balises** optionnelles
 - nom, prenom, age, adresses email (plusieurs possibles)



SCHÉMA XML

LES SCHÉMAS DE DÉFINITION XML

SCHÉMA DE DÉFINITION

- Les **DTD** sont limités ...
 - Ils ne sont pas au format **XML**
 - Pas de typage de données (nombre entier, décimal, date, chaîne de caractères, ...)
- Introduction de **XSD** (e**X**tensible **S**chema **D**efinition)
 - Comme pour les documents **XML**, il faut l'en-tête et un élément racine
 - Il sera possible d'inclure plusieurs **XSD** pour un document **XML**

```
<?xml version="1.0" encoding="UTF-8" ?>  
  
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
  
</xsd:schema>
```

SCHÉMA DE DÉFINITION

- Pour identifier un **XSD**, on va utiliser un espace de nom, en plus de la localisation du schéma
 - SAUF pour les **XSD** « natifs »

```
<?xml version="1.0" encoding="UTF-8" ?>  
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
</xsd:schema>
```

Espace de nom

Préfix configuré dans le
document XML

SCHÉMA DE DÉFINITION

- Déclarer un **attribut**

```
<xsd:attribute name="id" type="xsd:int" use="required" />  
<xsd:attribute name="active" type="xsd:boolean" default="false" />
```

- Déclarer un élément « simple »

```
<xsd:element name="nom" type="xsd:string" />  
<xsd:element name="prenom" type="xsd:string" minOccurs="0" maxOccurs="5" />  
<xsd:element name="email" type="xsd:string" minOccurs="0" maxOccurs="unbounded" />
```

- unbounded pour illimité
- Quelques types
 - string, int, long, short, byte, double, float, decimal, boolean, nonPositiveInteger, negativeInteger, ...

SCHÉMA DE DÉFINITION

- Déclarer un élément « complexe »

```
<xsd:element name="personne">
  <xsd:complexType>
    <!-- Eléments contenus -->

    <!-- Attributs -->
  </xsd:complexType>
</xsd:element>
```

SCHÉMA DE DÉFINITION

- Déclarer des éléments dans un élément complexe
 - sequence
 - Utilisation des éléments dans l'ordre défini dans la séquence
 - all
 - Utilisation des éléments dans n'importe quel ordre
 - choice
 - Utilisateur d'un des éléments, au choix

```
<xsd:element name="personne">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nom" type="xsd:string" />
      <xsd:element name="prenom" type="xsd:string" />
    </xsd:sequence>

    <!-- Attributs -->
  </xsd:complexType>
</xsd:element>
```

SCHÉMA DE DÉFINITION

- Rendre unique la valeur d'un **attribut** (pour l'identifiant par exemple)

```
<xsd:element name="personnes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="personne" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:unique name="uniqueId">
    <xsd:selector xpath="personne" />
    <xsd:field xpath="@id" />
  </xsd:unique>
</xsd:element>
```

SCHÉMA DE DÉFINITION

- Faire référence à un élément déclaré

```
<xsd:element name="personnes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="personne" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

SCHÉMA DE DÉFINITION

- Un schéma peut avoir un espace de nom (et c'est généralement le cas)
 - Identifié par `targetNamespace`
 - Permet de regrouper les éléments et attributs dans un « espace »
 - Permet d'éviter des conflits avec d'autres **XSD** qui pourraient utiliser les mêmes noms d'élément

```
<?xml version="1.0" encoding="UTF-8" ?>

<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ajc-ingenierie.fr/people"
  xmlns="http://www.ajc-ingenierie.fr/people"
  elementFormDefault="qualified">
  <!-- Définition -->
</xsd:schema>
```

SCHÉMA DE DÉFINITION

- Pour faire référence à un élément (**balise**), on peut utiliser un préfix

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

– Puisqu'il est possible d'inclure plusieurs **XSD** pour un seul document **XML**

- Il faut couvrir les cas où un nom d'élément est identique dans plusieurs **XSD** ... !

```
<xsd1:personne />  
<xsd2:personne />  
<xsd1:nom />
```

SCHÉMA DE DÉFINITION

- Dans un document **XML**, le ou les **XSD** se déclare(nt) dans l'élément racine :
 - Sans préfix (le **XSD** est seul, pas de situation de conflit entre noms identiques)

```
<?xml version="1.0" ?>
```

```
<personnes xmlns="http://www.ajc-ingenierie.fr/people"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.ajc-ingenierie.fr/people people.xsd">
```

```
<!-- Eléments -->  
</personnes>
```

Espace de nom (sans préfix)

Localisation du fichier

« Tu trouveras cet espace de nom dans ce fichier XSD »

Espace de nom

SCHÉMA DE DÉFINITION

- Dans un document **XML**, le ou les **XSD** se déclare(nt) dans l'élément racine :
 - Avec préfix (plusieurs **XSD**)

Espace de nom (avec préfix "people")

```
<?xml version="1.0" ?>
<people:personnes
  xmlns:people="http://www.ajc-ingenierie.fr/people"
  xmlns:inge="http://www.ajc-ingenierie.fr/inge"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.ajc-ingenierie.fr/inge inge.ingénieur.xsd
    http://www.ajc-ingenierie.fr/people people.xsd">
  <!-- Eléments -->
</people:personnes>
```

```
<inge:balise></inge:balise>
<people:personne>
  <people:nom>PERROUAULT</people:nom>
</people:personne>
```


SCHÉMA DE DÉFINITION

- Dans un document **XML**, le ou les **XSD** se déclare(nt) dans l'élément racine :
 - Avec et sans préfix (plusieurs **XSD**)

```
<?xml version="1.0" ?>

<personnes
  xmlns="http://www.ajc-ingenierie.fr/people"
  xmlns:inge="http://www.ajc-ingenierie.fr/inge"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.ajc-ingenierie.fr/inge ingénieur.xsd
    http://www.ajc-ingenierie.fr/people people.xsd">

  <!-- Eléments -->
</personnes>
```

```
<inge:balise></inge:balise>

<personne>
  <nom>PERROUAULT</nom>
</personne>
```

SCHÉMA DE DÉFINITION – EXERCICE

- Générer et adapter le **XSD** associé au document **XML** des personnes
 - Avec un espace de nom
 - **Attributs** obligatoires
 - id
 - **Attributs** optionnels
 - nom, prenom, age
 - **Balises** optionnelles
 - nom, prenom, age, adresses email (plusieurs possibles)

Le générer avec cet outil : <https://www.freeformatter.com/xsd-generator.html>