XML

eXtensible Markup Language

J. Saraydaryan

CPE - Lyon

XML

Introduction

- Objectif
- Origine
- Définition

- **Objectif**

Comment stocker et transférer des documents de façon universelle ?

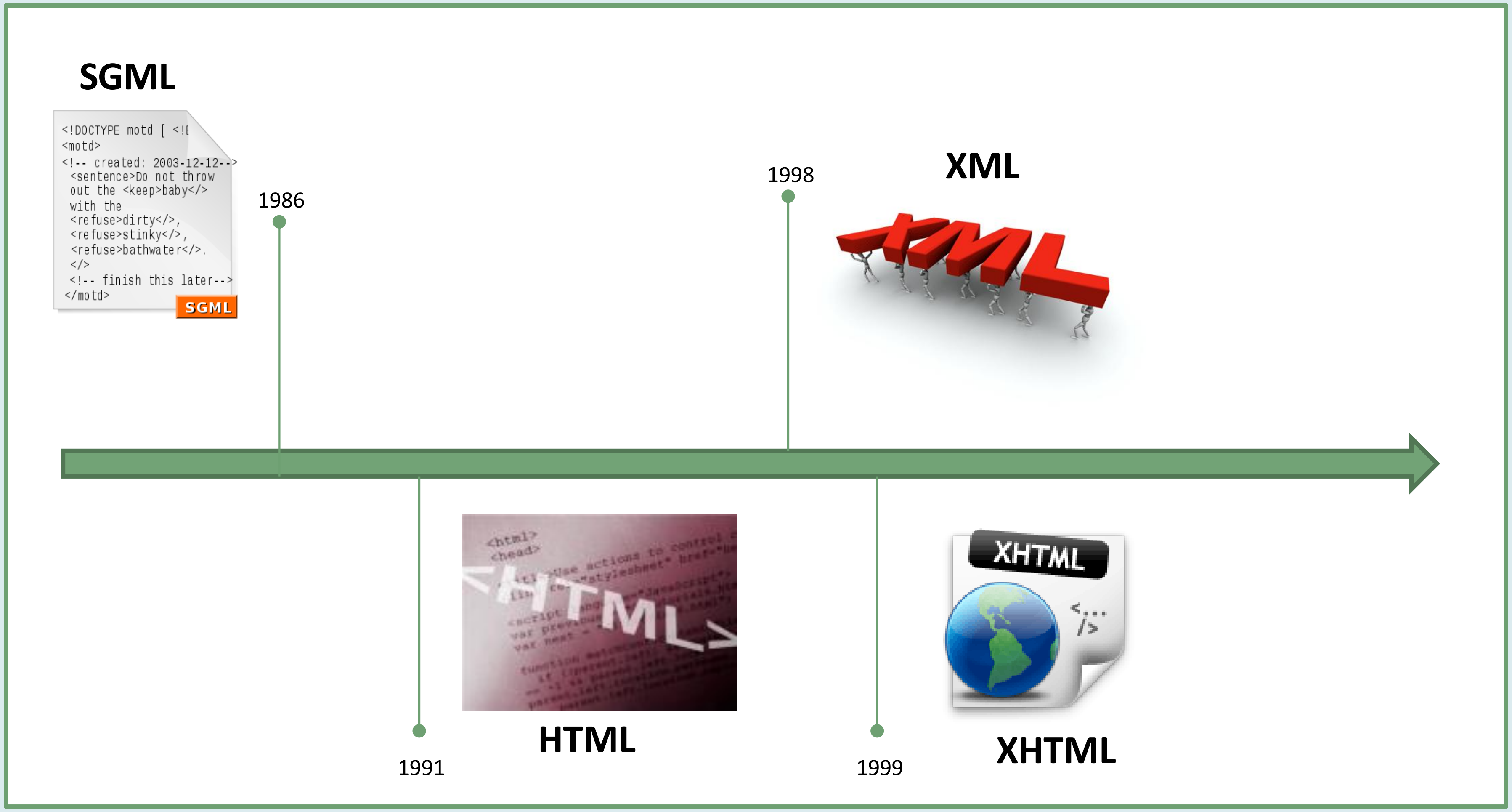
XML

eXtensible Markup Language

- **Objectifs**
  - ❑ Séparation stricte contenu / présentation
  - ❑ Structuration forte
  - ❑ Simplicité, universalité, extensibilité
  - ❑ Modèle de document
  - ❑ Format libre
  - ❑ Prise en charge native des caractères spéciaux
  - ❑ Simplification partage de données / transport d'information/ changement de plateforme

XML

- Origine

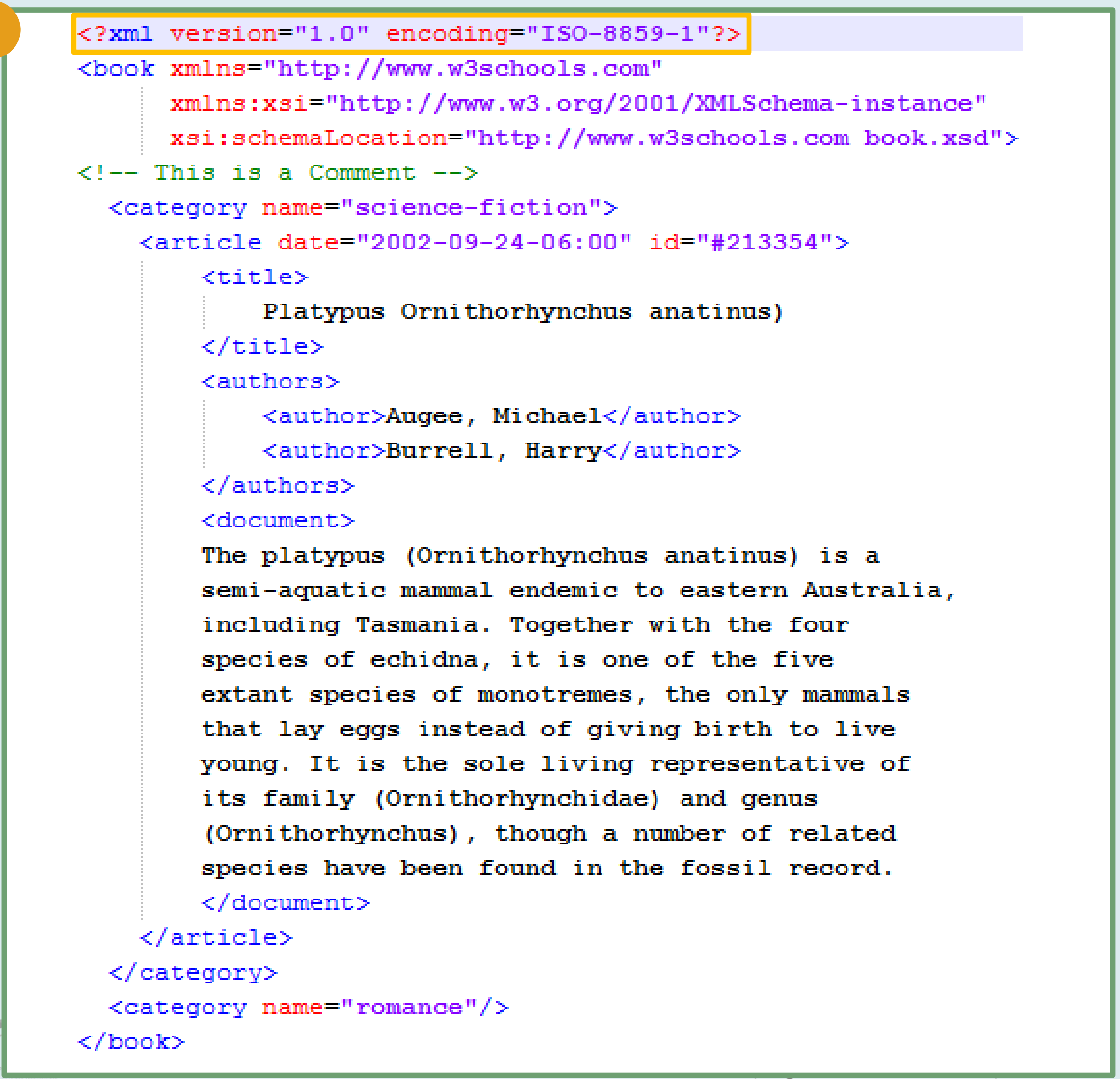


```
<!DOCTYPE motd [ <!E
<motd>
<!-- created: 2003-12-12-->
 <sentence>Do not throw
 out the <keep>baby</>
 with the
 <refuse>dirty</>,
 <refuse>stinky</>,
 <refuse>bathwater</>.
 </>
 <!-- finish this later-->
</motd>
```

XML

- **Définition**
  - ❑ Extensible Markup Language
  - ❑ " Markup Language " comme HTML
  - ❑ Conçu pour transporter de l'information pas pour l'afficher
  - ❑ Conçu pour être décrit par lui-même
  - ❑ Recommandation W3C

Définition : Syntaxe



1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

1

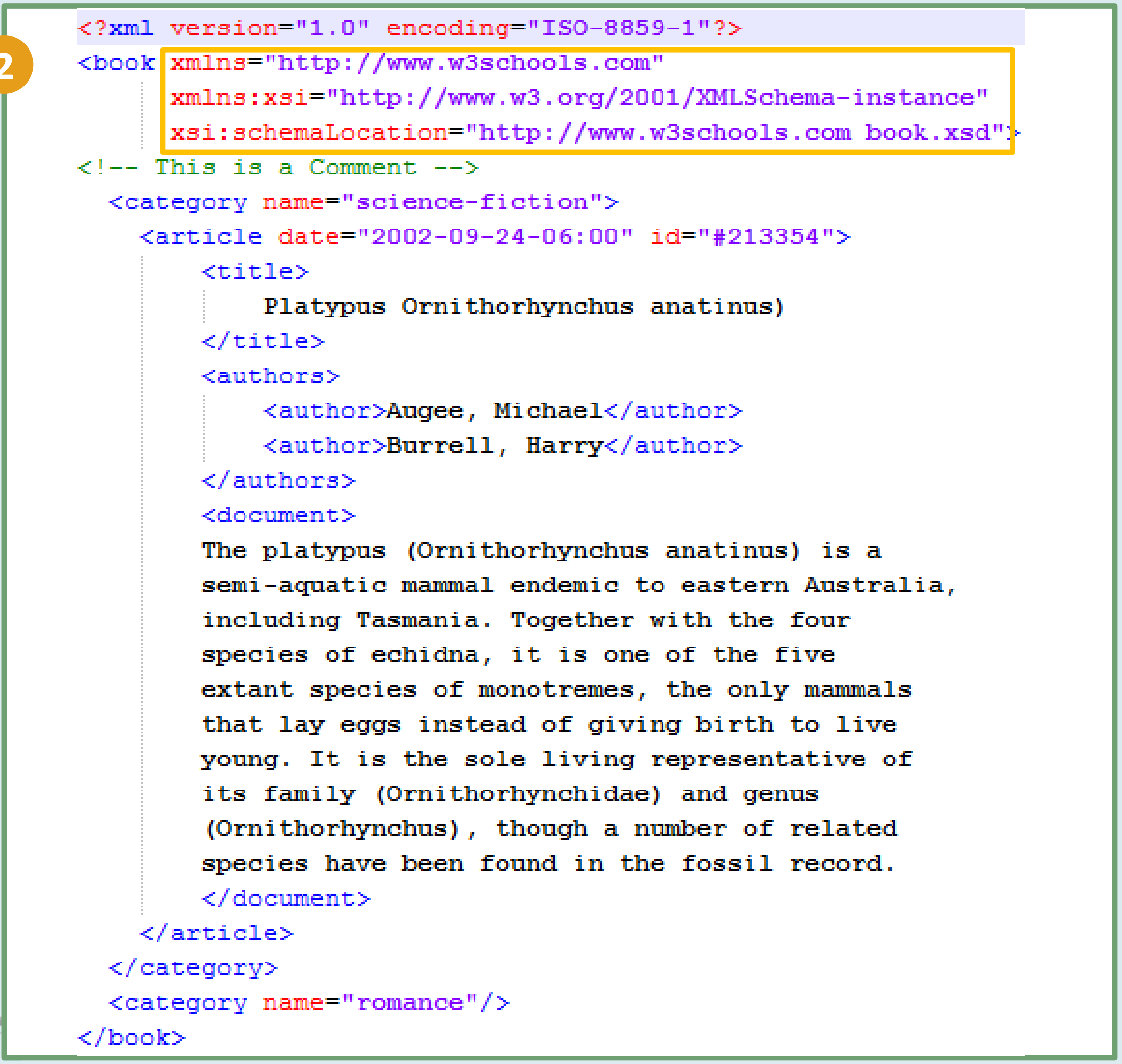
Entête XML

Version 1.0

Encodage ISO-8859-1

• Définition : Syntaxe

2



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

2

Name space et position du validateur

XML

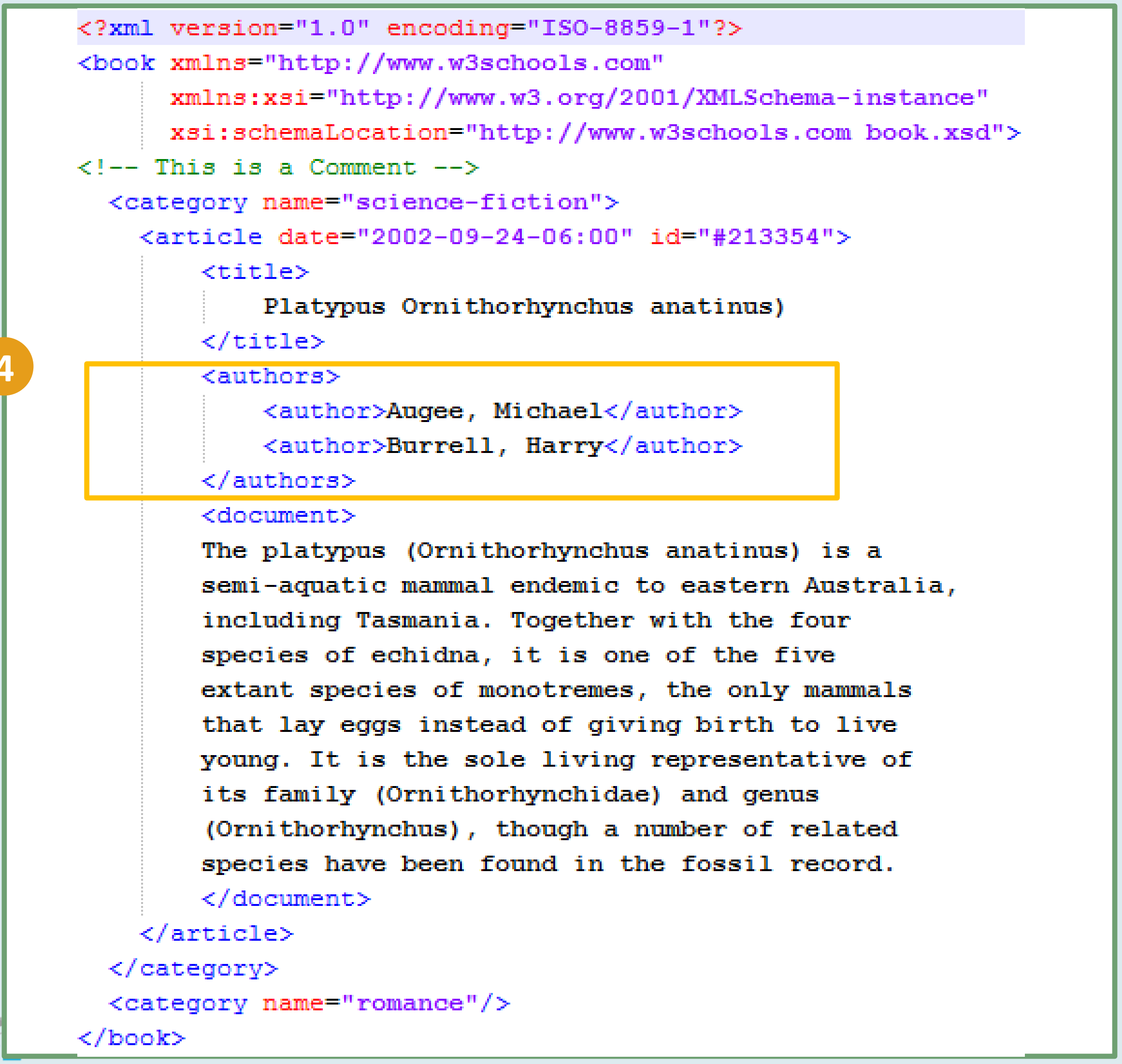
• Définition : Syntaxe

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

3

3 **Commentaire**

• Définition : Syntaxe



4

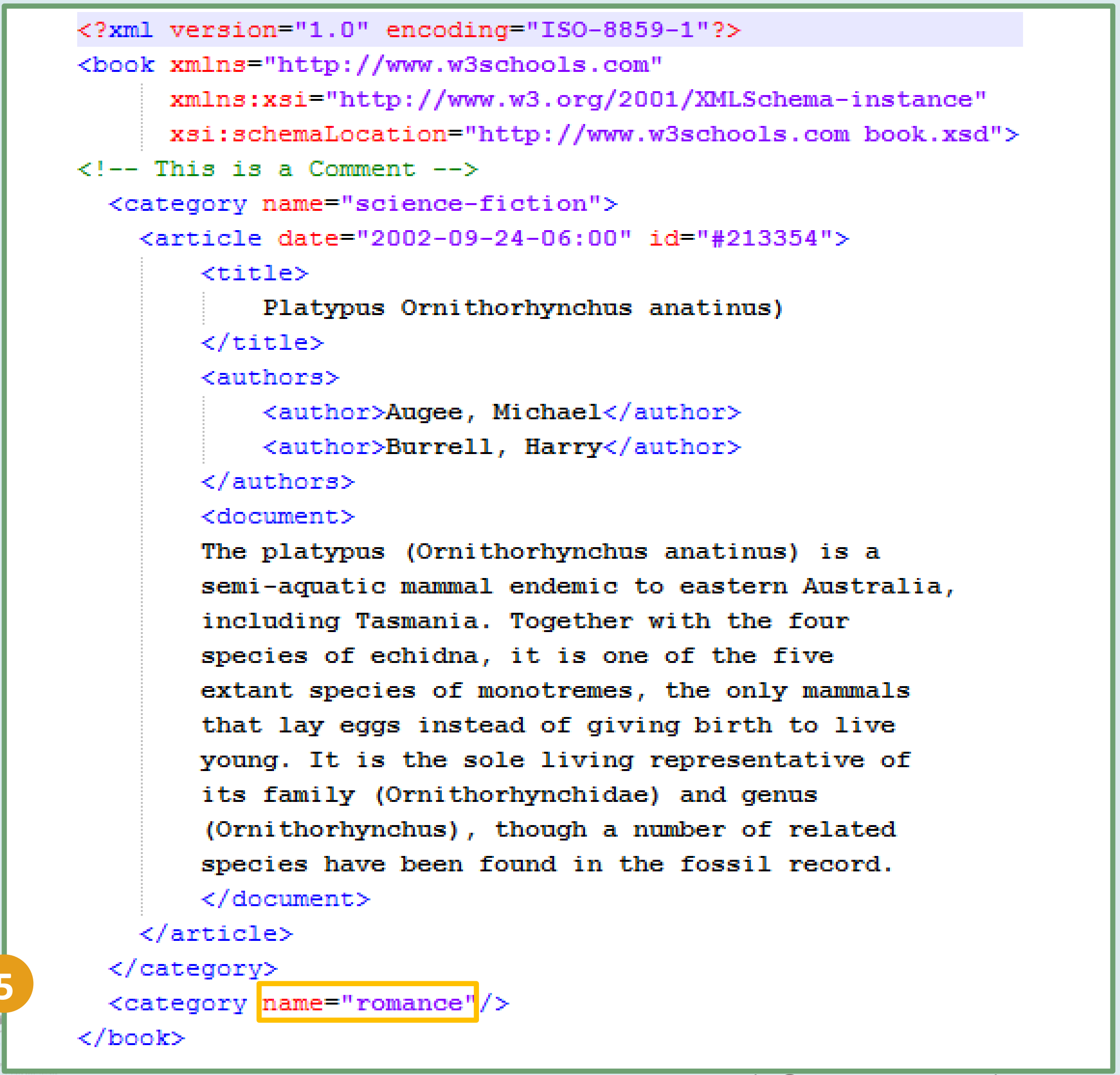
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

4

Balises d'éléments

Case sensitive

• Définition : Syntaxe



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

5

5

Attributs d'éléments

• Définition

❑ Caractères spéciaux

Remplacement dans le Texte

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Marker <![CDATA[...]]>

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        <![CDATA[
        <platypus> <<<<&Ornithorhynchus anatinus)>>>> is a
        "semi-aquatic mammal endemic to eastern 'Australia'"
        ]]>
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

XML

- **Définition : Structure**
  - Orthographe : règle de bonne écriture
  - Grammaire : agencement des mots dans une phrase
  - Validité document XML:
    - bien formé (syntaxique)
    - valide (structurelle)

• Définition : Structure

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book [
    ...
]>
<book>
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

Prologue

Entête XML

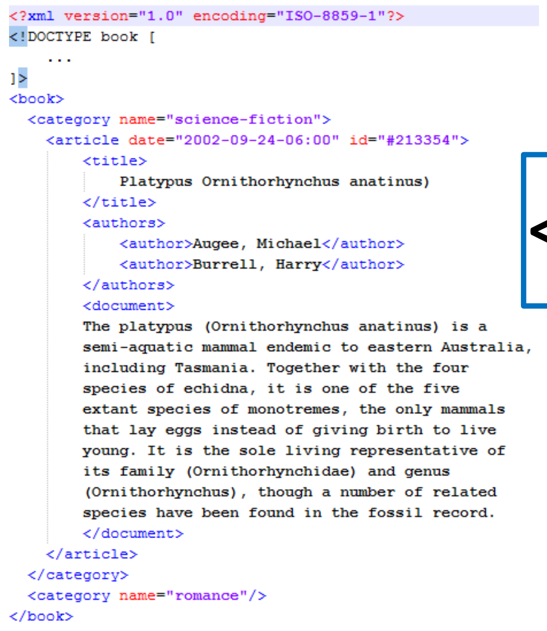
- version
- encoding
- standalone

Type de document

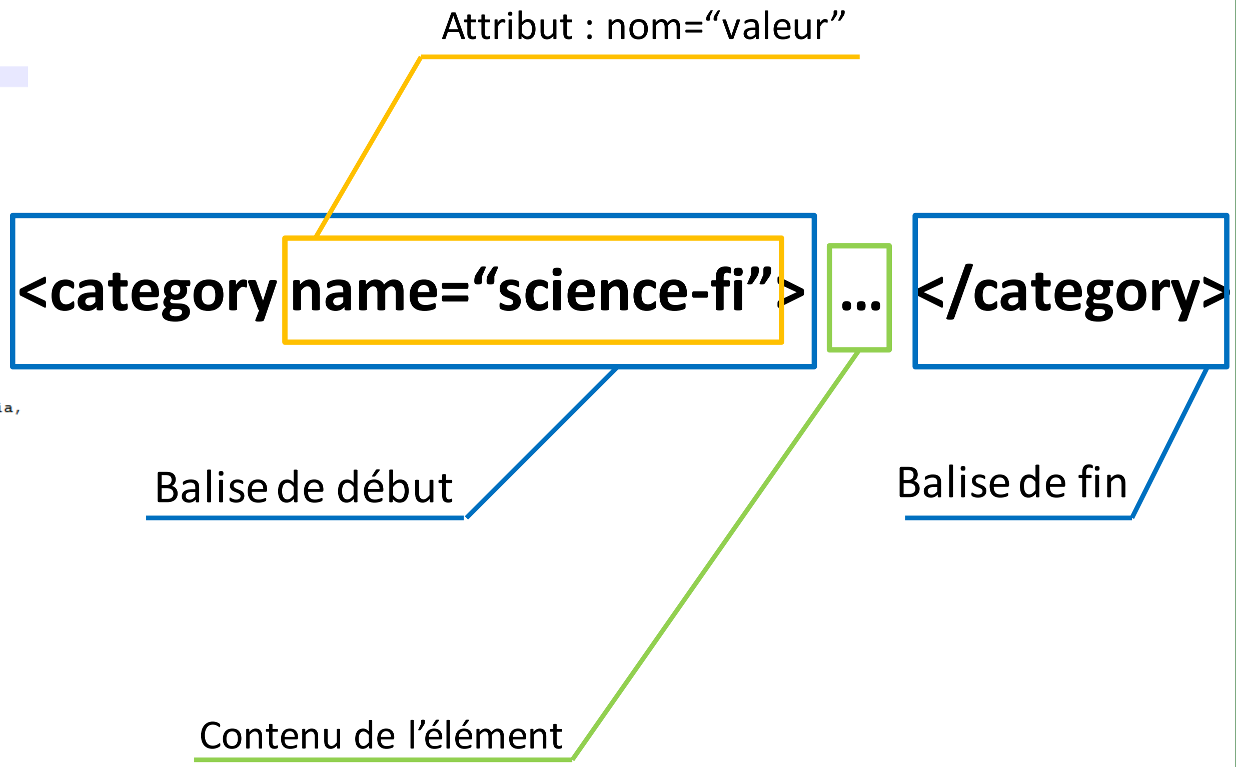
<! DOCTYPE root-element[…]>

Corps

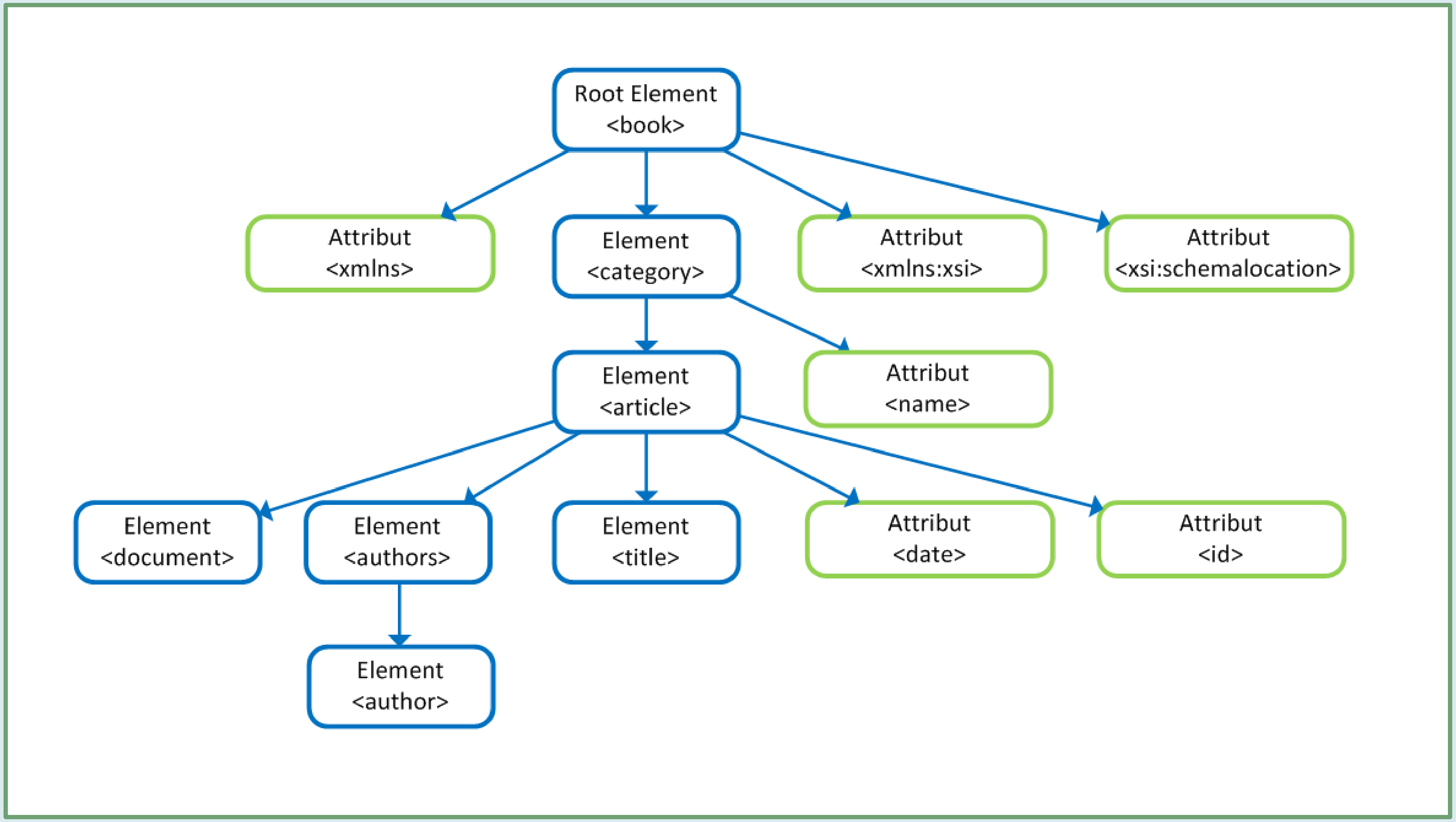
• Définition : Structure



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book [
    ...
]>
<book>
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```



• Définition : Structure

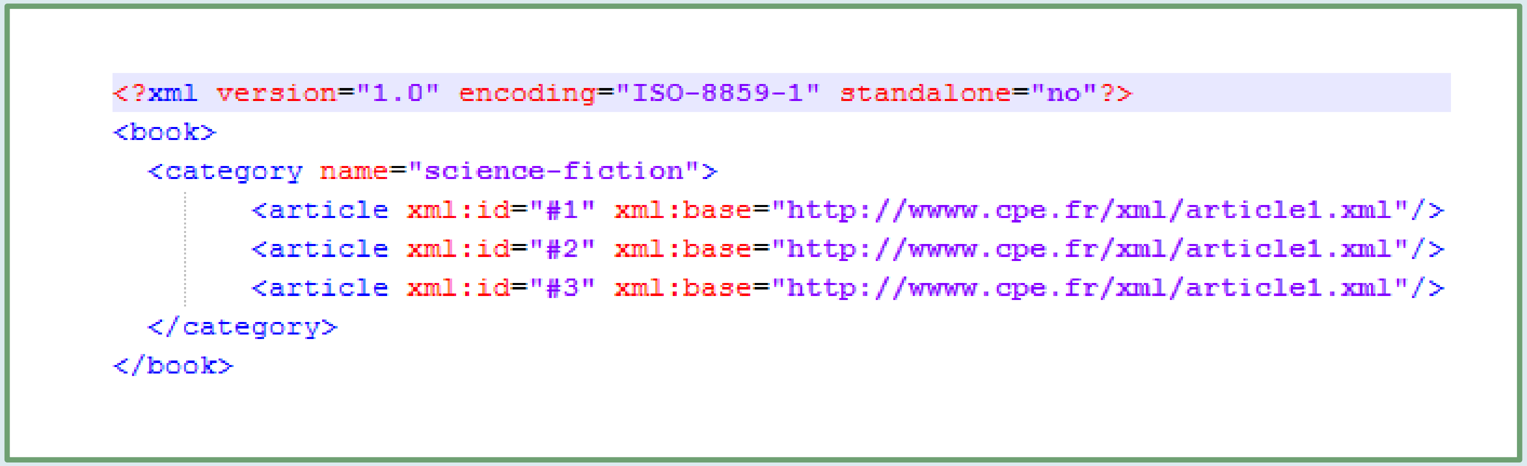


XML

- **Définition : Structure**
  - ❑ Attributs spéciaux
  - ❑ Elements/Attributs externes
  - ❑ Instruction de traitement
  - ❑ Xinclude

- **Définition : Structure**
  - ❑ Attributs spéciaux

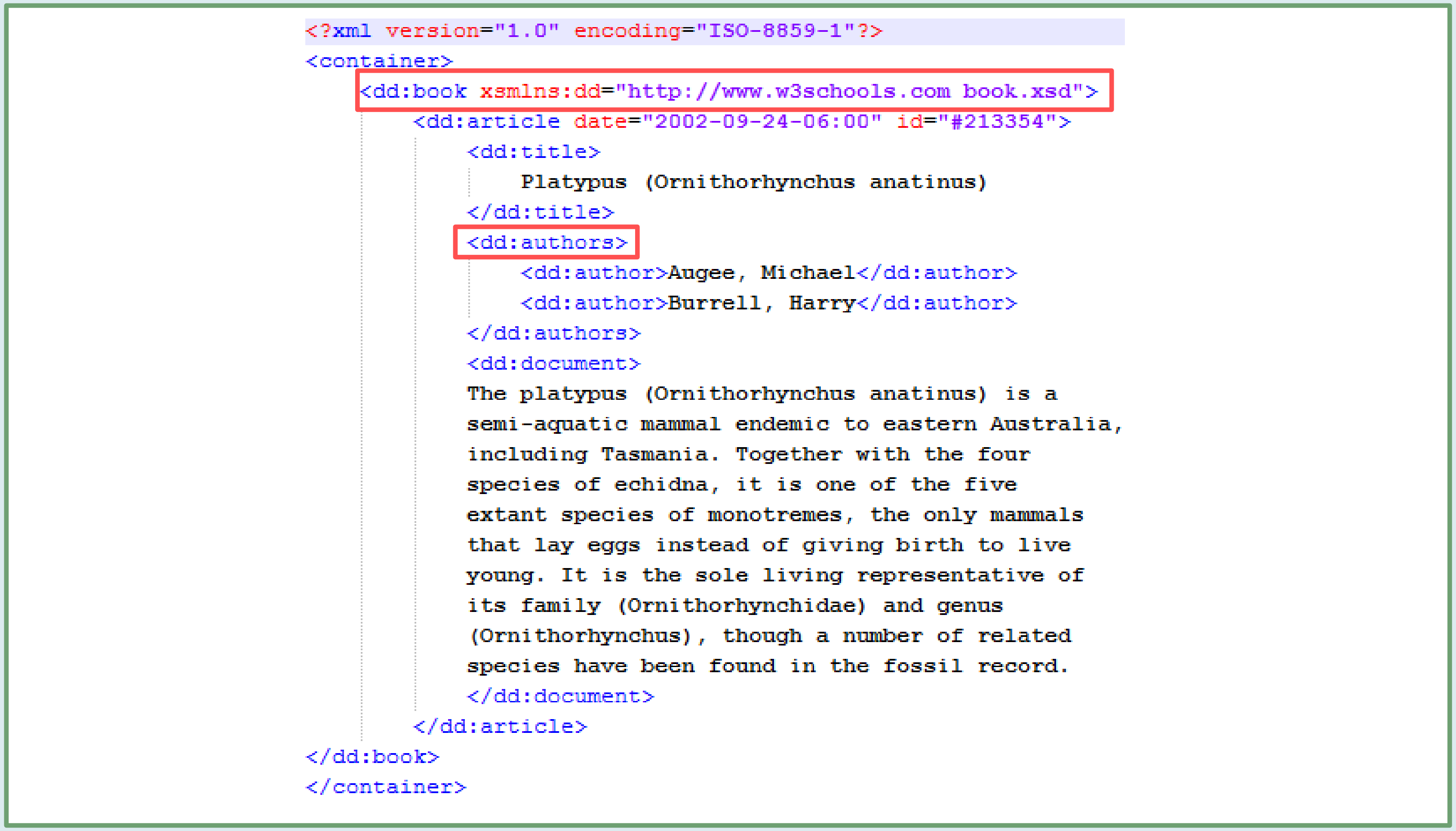
```
<book xml:id=“#dcsdcsd” >
<book xml:lang=“en-US” >
<article xml:base=“http://www.cpe.fr/xml/article1.xml”
```



```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<book>
  <category name="science-fiction">
        <article xml:id="#1" xml:base="http://www.cpe.fr/xml/article1.xml"/>
        <article xml:id="#2" xml:base="http://www.cpe.fr/xml/article1.xml"/>
        <article xml:id="#3" xml:base="http://www.cpe.fr/xml/article1.xml"/>
  </category>
</book>
```

• Définition : Structure

❑ Elements/Attributs externes



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<container>
    <dd:book xsmlns:dd="http://www.w3schools.com book.xsd">
        <dd:article date="2002-09-24-06:00" id="#213354">
            <dd:title>
                Platypus (Ornithorhynchus anatinus)
            </dd:title>
            <dd:authors>
                <dd:author>Augee, Michael</dd:author>
                <dd:author>Burrell, Harry</dd:author>
            </dd:authors>
            <dd:document>
            The platypus (Ornithorhynchus anatinus) is a
            semi-aquatic mammal endemic to eastern Australia,
            including Tasmania. Together with the four
            species of echidna, it is one of the five
            extant species of monotremes, the only mammals
            that lay eggs instead of giving birth to live
            young. It is the sole living representative of
            its family (Ornithorhynchidae) and genus
            (Ornithorhynchus), though a number of related
            species have been found in the fossil record.
            </dd:document>
        </dd:article>
</dd:book>
</container>
```

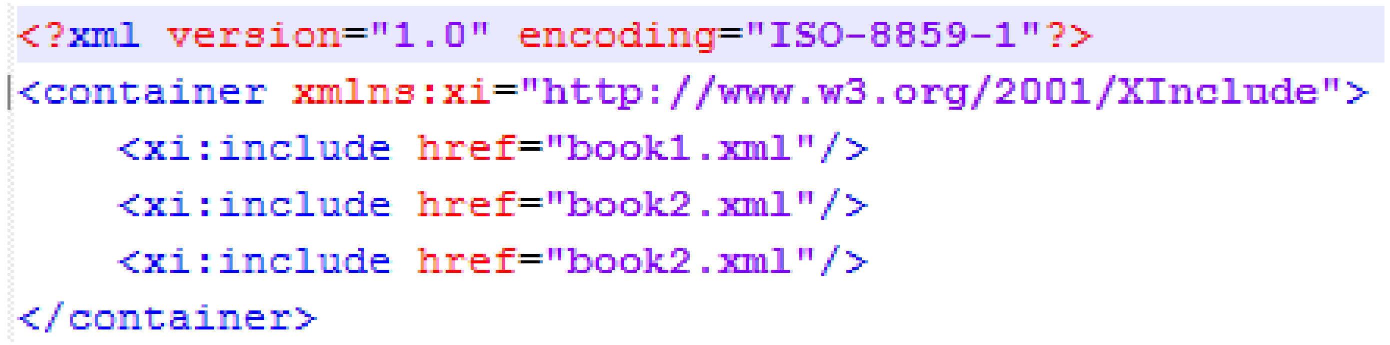
Définition : Structure

- Instruction de traitement
  - Information destinée aux applications qui traitent le XML
  - Délimiteur **<? …. ?>**

```
<? dbhtml filename="index.html" ?>
<?
    xml-stylesheet href="list.xsl"
    type="text/xsl" title="List title"
?>
```

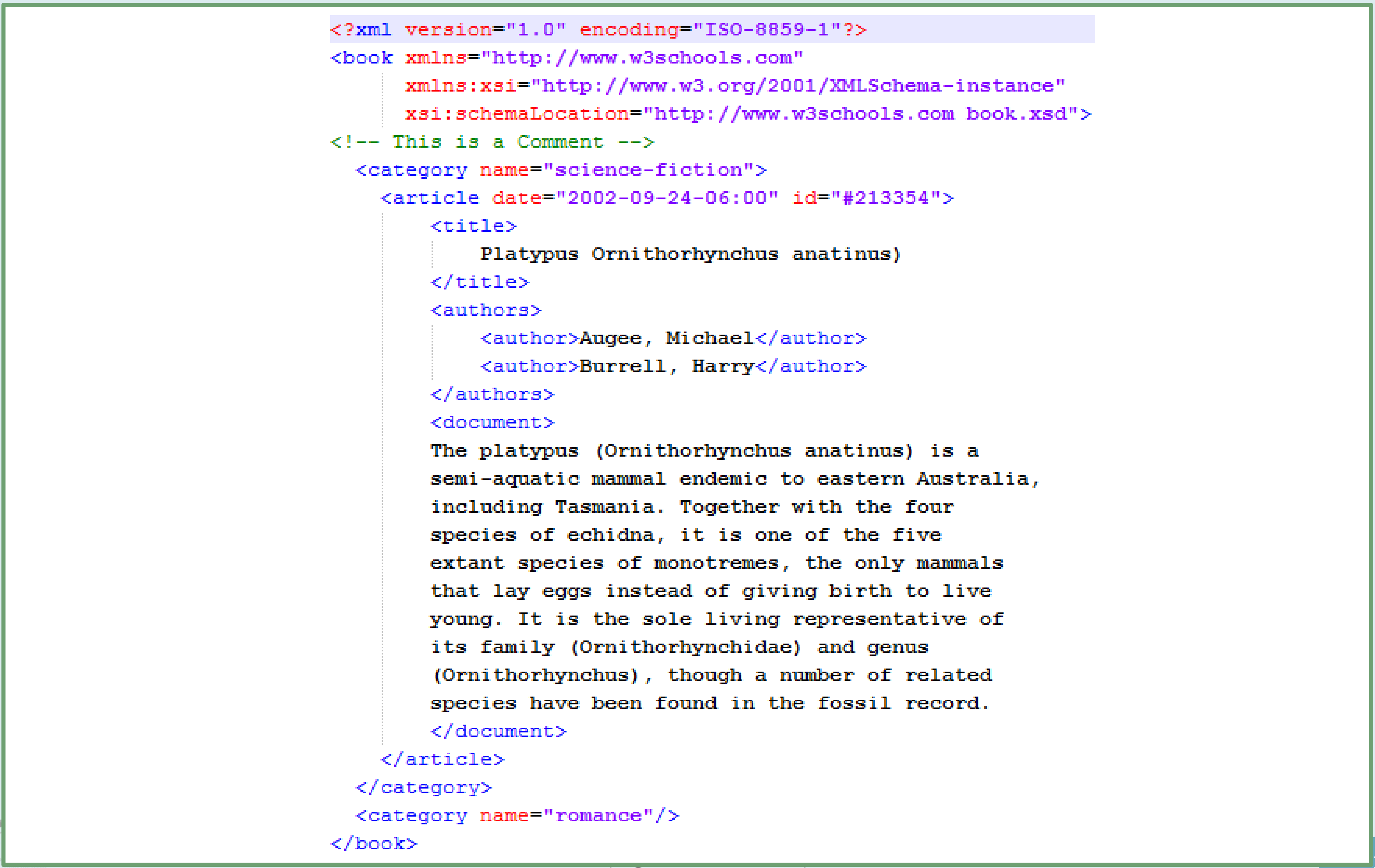
Définition : Structure

- Xinclude : Import de documents externes



```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<container xmlns:xi="http://www.w3.org/2001/XInclude">
    <xi:include href="book1.xml"/>
    <xi:include href="book2.xml"/>
    <xi:include href="book2.xml"/>
</container>
```

• Fichier XML



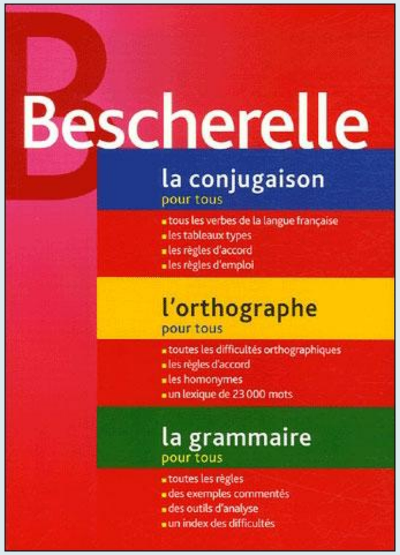
```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

Model et Validation

- DTD
- Schema XML

- **Objectif**

Comment définir des règles de construction et valider un document ?



XML | **DTD – XML Schema**

XML

- **Objectif**

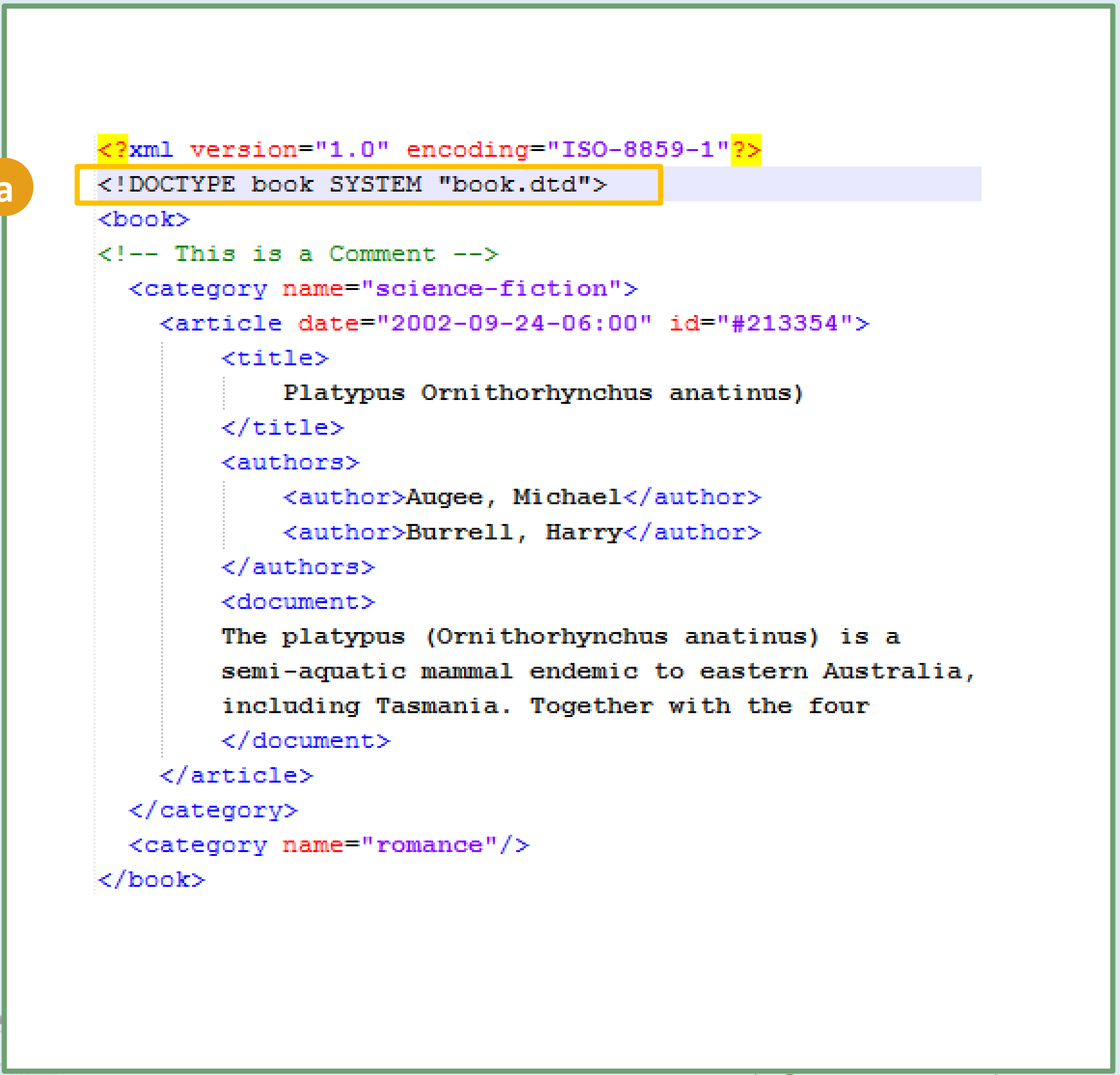
Comment définir des règles de construction et valider un document ?

DTD – XML Schema

XML

- **DTD**
  - ❑ Document Type Definition
  - ❑ Simple mais limité
  - ❑ Hérité du SGML
  - ❑ Contient des déclarations d'éléments et d'attributs

DTD: Comment l'utiliser

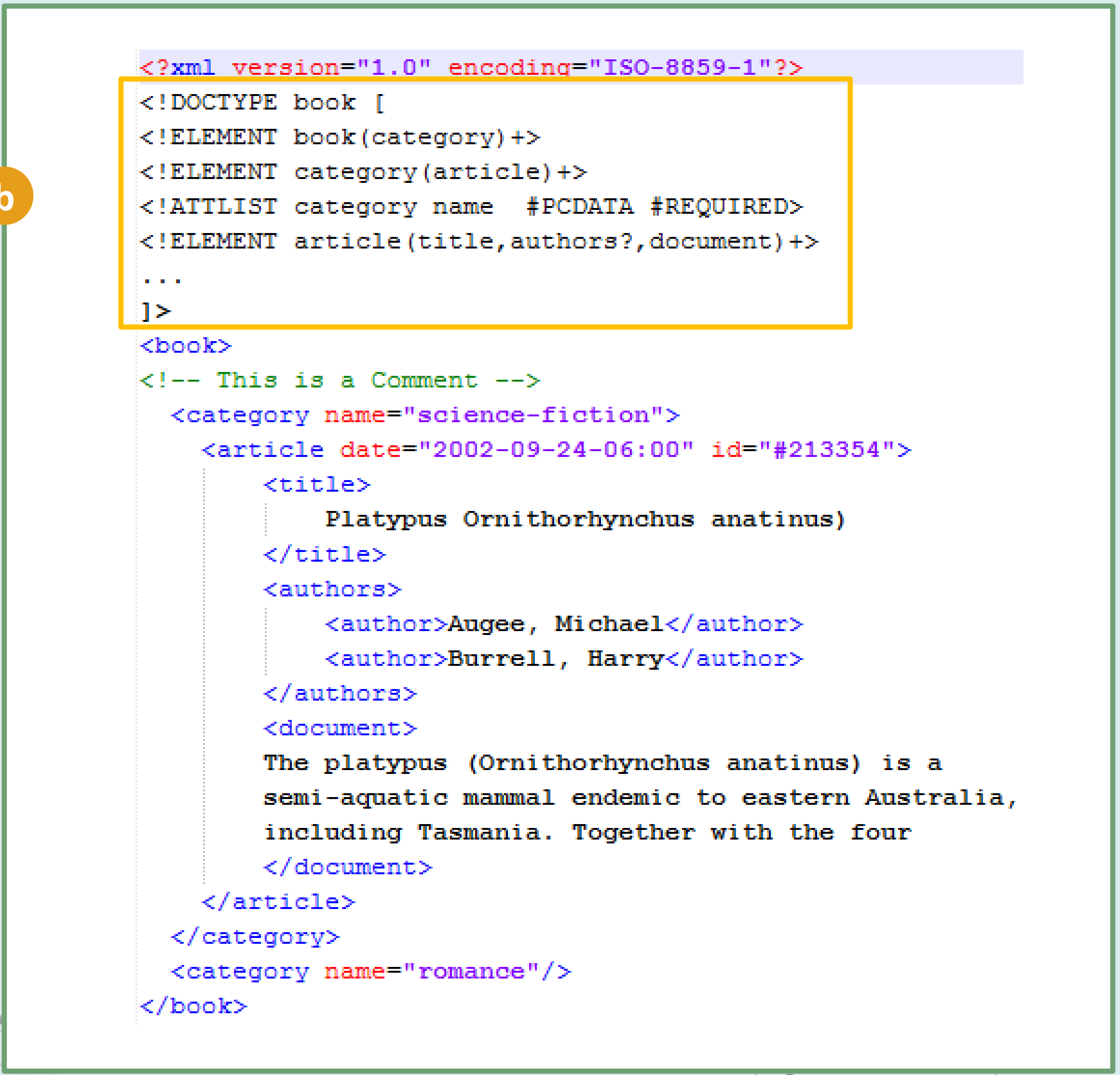

a

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book SYSTEM "book.dtd">
<book>
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

1 **Déclaration**

a Externe
Adressage URL ou FPI (Format Public Identifier)

DTD: Comment l'utiliser



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book [
<!ELEMENT book(category)+>
<!ELEMENT category(article)+>
<!ATTLIST category name  #PCDATA #REQUIRED>
<!ELEMENT article(title,authors?,document)+>
...
]>
<book>
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

1 **Déclaration**

a Externe

b Interne
<!DOCTYPE *root-element* **[***declaration***]>**

DTD : Les Entités (1/2)

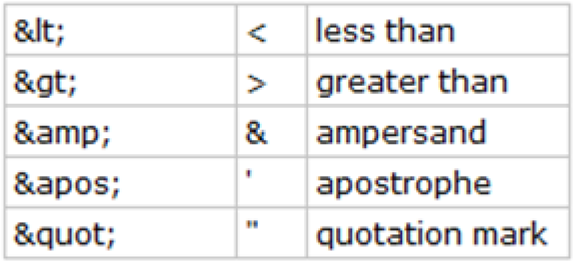
- Définition de constante dans le document ou dans la déclaration DTD

```
<!ENTITY dfn "definition">
<!ENTITY euro "#x20AC">
```

Définition Interne

```
…
<article id="#213354"  price="10 &euro;">
…
```

Utilisation

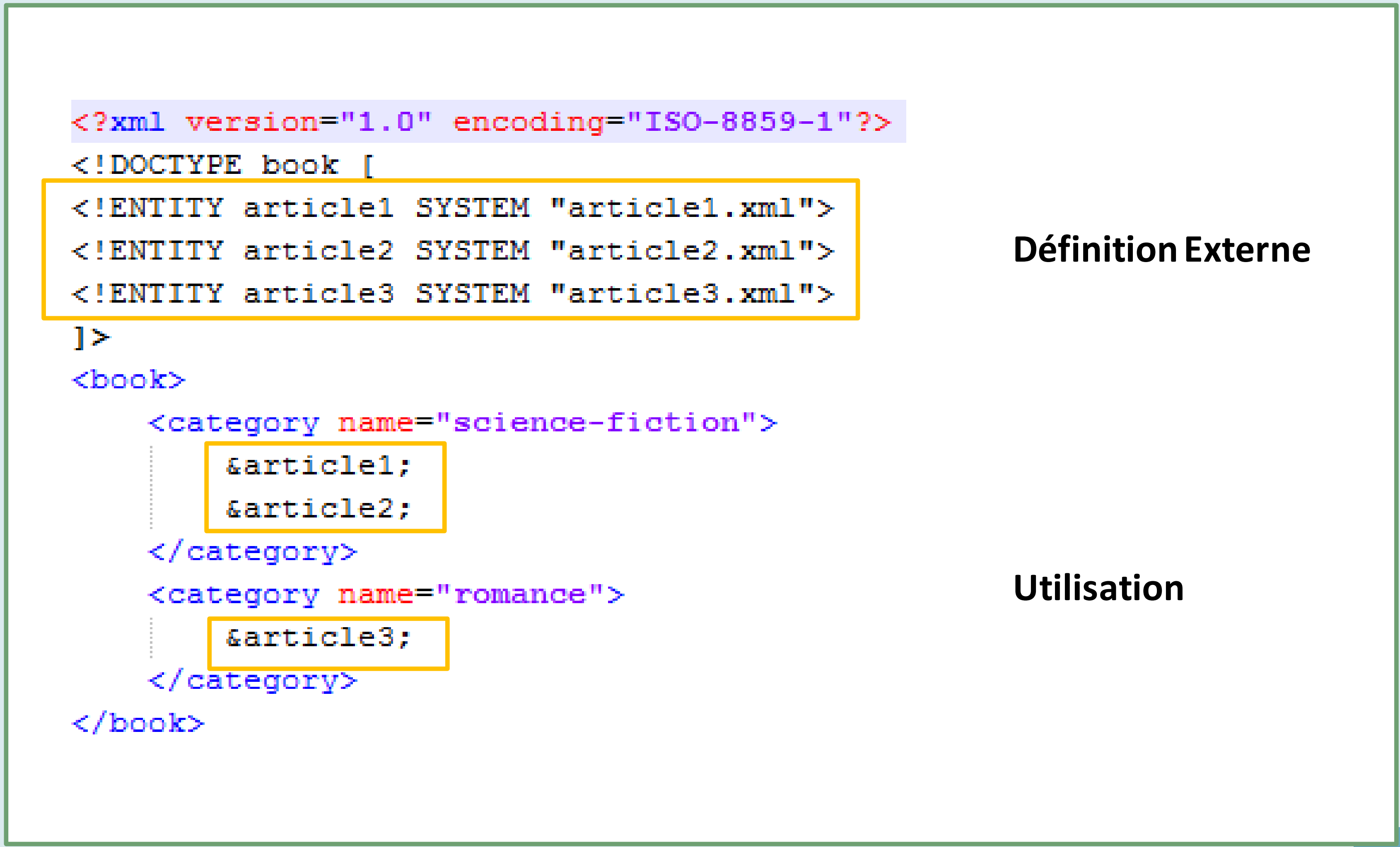


&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Entités prédéfinies

DTD : Les Entités (2/2)

- Définition de constante dans le document ou dans la déclaration DTD



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book [
<!ENTITY article1 SYSTEM "article1.xml">
<!ENTITY article2 SYSTEM "article2.xml">
<!ENTITY article3 SYSTEM "article3.xml">
]>
<book>
    <category name="science-fiction">
        &article1;
        &article2;
    </category>
    <category name="romance">
        &article3;
    </category>
</book>
```

DTD : Les Elements (1/3)

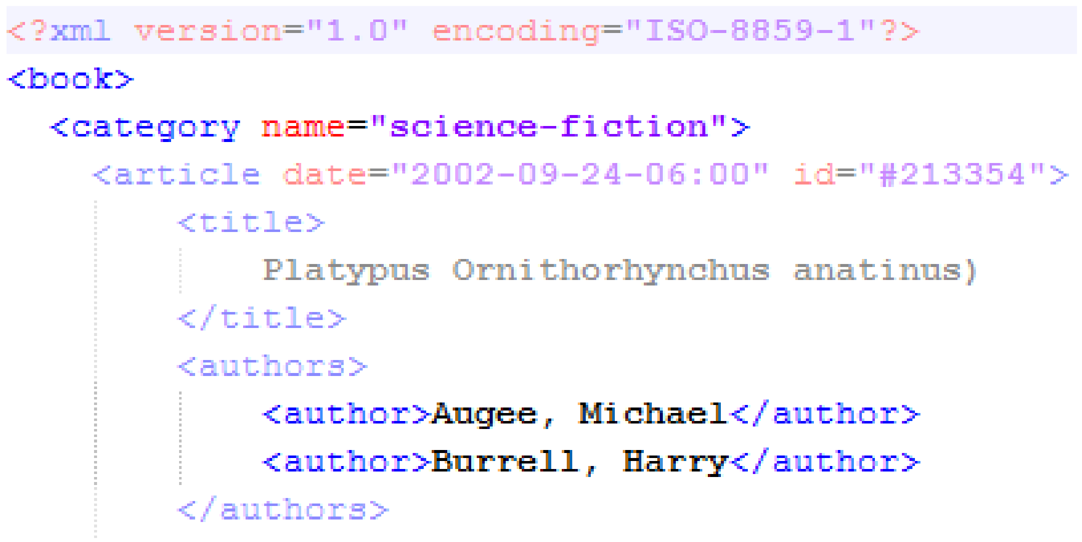
- Définition d’objet pouvant contenir des attributs et d’autres éléments

Définition

```
<!ELEMENT element regexType >
```

Exemple

```
<!ELEMENT book (category)+ >
<!ELEMENT author(#PCDATA) >
```



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
```

DTD : Les Elements (2/3)

❑ Définition d'objet pouvant contenir des attributs et d'autres éléments

RegexType

Opérateur	Définition
,	Mise en séquence
|	Choix
?	Occurrence 0,1
*	Répétition
+	Répétition non null

```
<!ELEMENT book (category)+ >
<!ELEMENT article(title, authors?, document)>
<!ELEMENT article(title, (authors?, document)+)>
<!ELEMENT article((title|annotation), authors, document)>
```

DTD : Les Elements (3/3)

- Définition d'objet pouvant contenir des attributs et d'autres éléments

Cas Particuliers

`<!ELEMENT author(#PCDATA) >`	**éléménț contenant du texte**
`<!ELEMENT info EMPTY >`	**élément vide**
`<!ELEMENT addInfo ANY>`	**élément quelconque**

DTD : Les Attributs (1/3)

- Définition des propriétés d'un élément

```
<!ATTLIST element attribut type default >

<!ATTLIST element attribut1 type1 default1
                  attribut2 type2 default2
                  attribut3 type3 default3>
```

Définition

```
<!ATTLIST article date CDATA “2000-01-01”
                  id     ID>
```

Exemple

```
<article date="2002-09-24-06:00" id="#213354">
```

DTD : Les Attributs (2/3)

❑ Définition des propriétés d'un élément

Type

Valeur de Type	Définition
CDATA	Aucune contrainte de valeurs
ID	Nom XML, un seul attribut de ce type
IDREF	Référence à un élément identifié par la valeur de l'attribut.
IDREFS	Liste de IDREF séparée par des espaces
NMTOKEN	Jeton
NMTOKENS	Liste de jeton séparée par des espaces
NOTATION	-
ENTITY	Entité externe non XML
ENTITIES	Liste d'entités externes séparée par des espaces

DTD : Les Attributs (3/3)

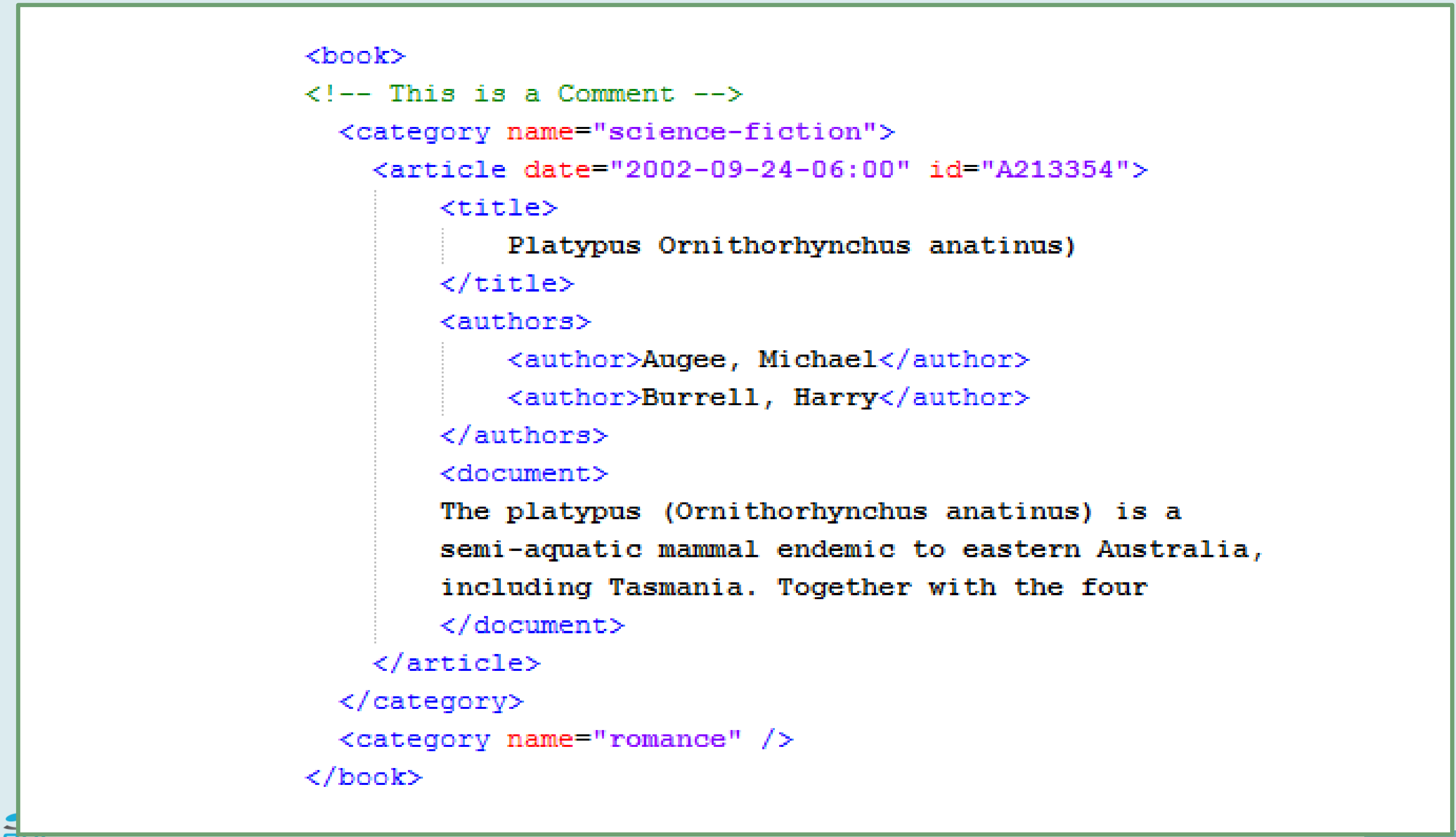
- Définition des propriétés d'un élément

Default

Valeur de Type	Définition
"valeur" ou 'valeur'	Valeur par défault
#IMPLIED	Attribut optionnel
#REQUIRED	Attribut obligatoire
#FIXED "valeur" ou 'valeur'	Valeur fixée par défault

```
<!ELEMENT article(title,authors?,document)+>
<!ATTLIST article date        CDATA "2000-01-01"
                  id          ID #REQUIRED
                  comment     CDATA #IMPLIED
                  version     CDATA #FIXED "v0.1">
```

• **DTD : A vous de jouer**



```
<book>
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="A213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        </document>
    </article>
  </category>
  <category name="romance" />
</book>
```

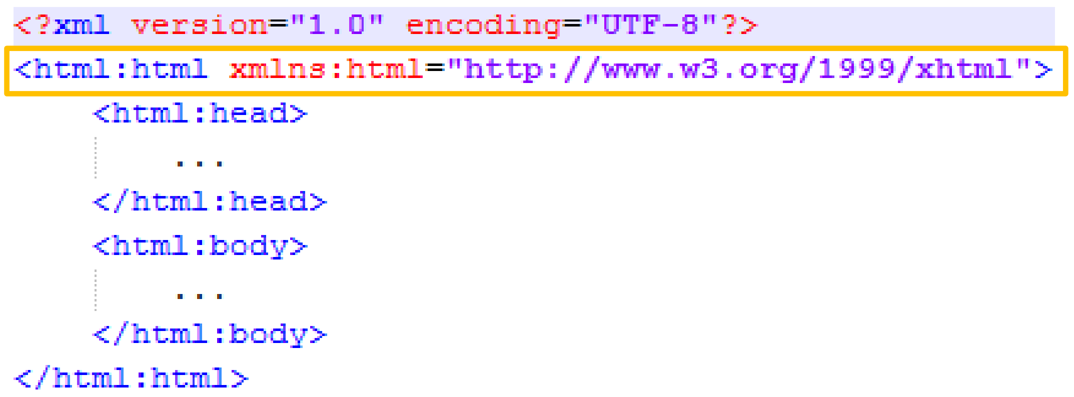
Model et Validation

- DTD
- Schema XML

XML

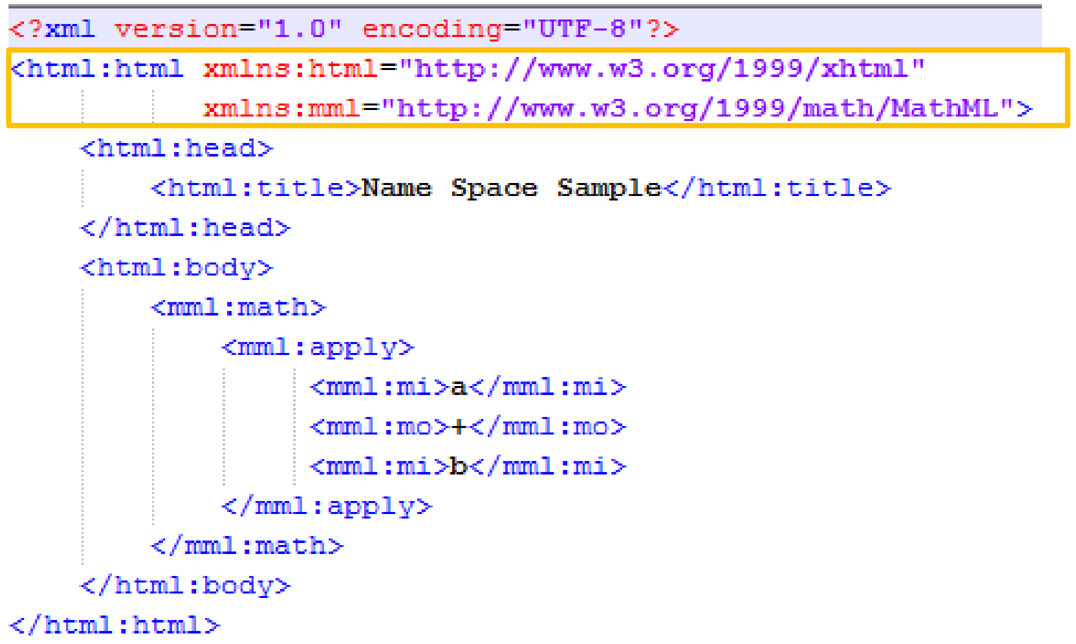
- **Schema XML**
  - ❑ Syntaxe purement XML
  - ❑ Plus de précision dans description
  - ❑ Plus de type de données
  - ❑ Définition de nouveaux type (customisation)
  - ❑ Modularité, réutilisation plus simple (notion de groupe…)
  - ❑ Utilisation d’espace de nommage
  - ❑ Extension des fichiers *.xsd

• Schema XML : Espace de nommage



```
<?xml version="1.0" encoding="UTF-8"?>
<html:html xmlns:html="http://www.w3.org/1999/xhtml">
    <html:head>
        ...
    </html:head>
    <html:body>
        ...
    </html:body>
</html:html>
```

Définition simple



```
<?xml version="1.0" encoding="UTF-8"?>
<html:html xmlns:html="http://www.w3.org/1999/xhtml"
           xmlns:mml="http://www.w3.org/1999/math/MathML">
    <html:head>
        <html:title>Name Space Sample</html:title>
    </html:head>
    <html:body>
        <mml:math>
            <mml:apply>
                <mml:mi>a</mml:mi>
                <mml:mo>+</mml:mo>
                <mml:mi>b</mml:mi>
            </mml:apply>
        </mml:math>
    </html:body>
</html:html>
```

Définition Multiple

Schema XML : Espace de nommage

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns:html="http://www.w3.org/1999/xhtml">
    <head>
        <title>Name Space Sample</title>
    </head>
    <body>
        <math xmlns="http://www.w3.org/1999/math/MathML">
            <apply>
                <mi>a</mi>
                <mo>+</mo>
                <mi>b</mi>
            </apply>
        </math>
    </body>
</html>
```

Portée d'un espace de nommage

Mode simplifié

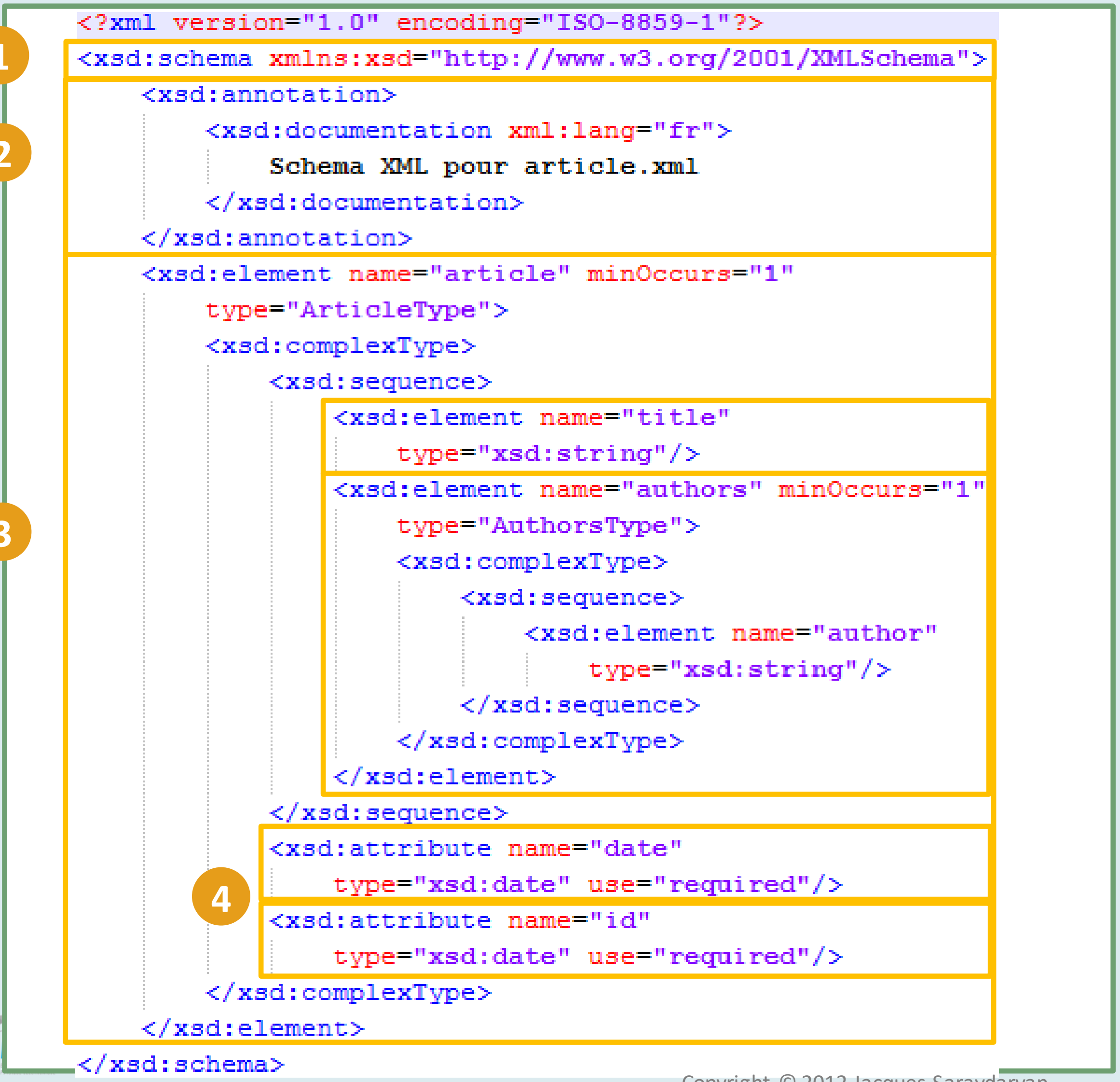
```
<math xmlns="http://www.w3.org/1999/math/MathML">
    <apply>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
    </apply>
</math>
```

```
<mml:math xmlns:mml="http://www.w3.org/1999/math/MathML">
    <mml:apply>
        <mml:mi>a</mml:mi>
        <mml:mo>+</mml:mo>
        <mml:mi>b</mml:mi>
    </mml:apply>
</mml:math>
```

• Schema XML : Import de schémas externes

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="xml.xsd"/>
    <xsd:element name="article" minOccurs="1"
        type="ArticleType"
        >
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="title"
                    type="xsd:string"/>
                <xsd:element name="authors" minOccurs="1"
                    type="AuthorsType">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="author"
                                type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="date"
                type="xsd:date" use="required"/>
            <xsd:attribute name="id"
                type="xsd:date" use="required"/>
            <xsd:attribute ref="xml:lang"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

• Schema XML : Syntaxe



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:annotation>
        <xsd:documentation xml:lang="fr">
            Schema XML pour article.xml
        </xsd:documentation>
    </xsd:annotation>
    <xsd:element name="article" minOccurs="1"
        type="ArticleType">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="title"
                    type="xsd:string"/>
                <xsd:element name="authors" minOccurs="1"
                    type="AuthorsType">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="author"
                                type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="date"
                type="xsd:date" use="required"/>
            <xsd:attribute name="id"
                type="xsd:date" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

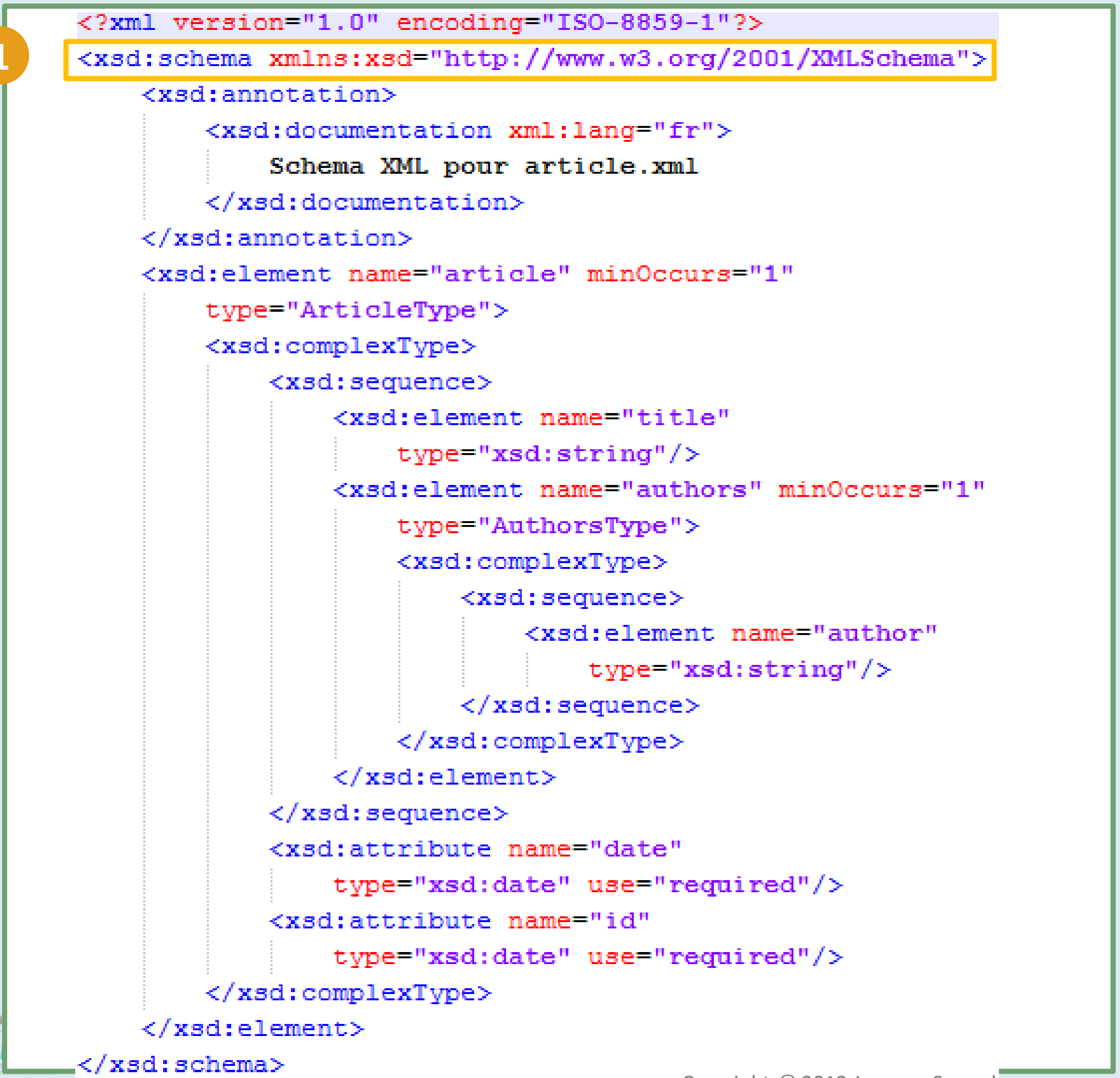
1 Espace de nommage utilisé pour les schema XML

2 Annotation du document

3 Description d'un élément

4 Description d'un attribut

Schema XML : Syntaxe



1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:annotation>
        <xsd:documentation xml:lang="fr">
            Schema XML pour article.xml
        </xsd:documentation>
    </xsd:annotation>
    <xsd:element name="article" minOccurs="1"
        type="ArticleType">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="title"
                    type="xsd:string"/>
                <xsd:element name="authors" minOccurs="1"
                    type="AuthorsType">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="author"
                                type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="date"
                type="xsd:date" use="required"/>
            <xsd:attribute name="id"
                type="xsd:date" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

1

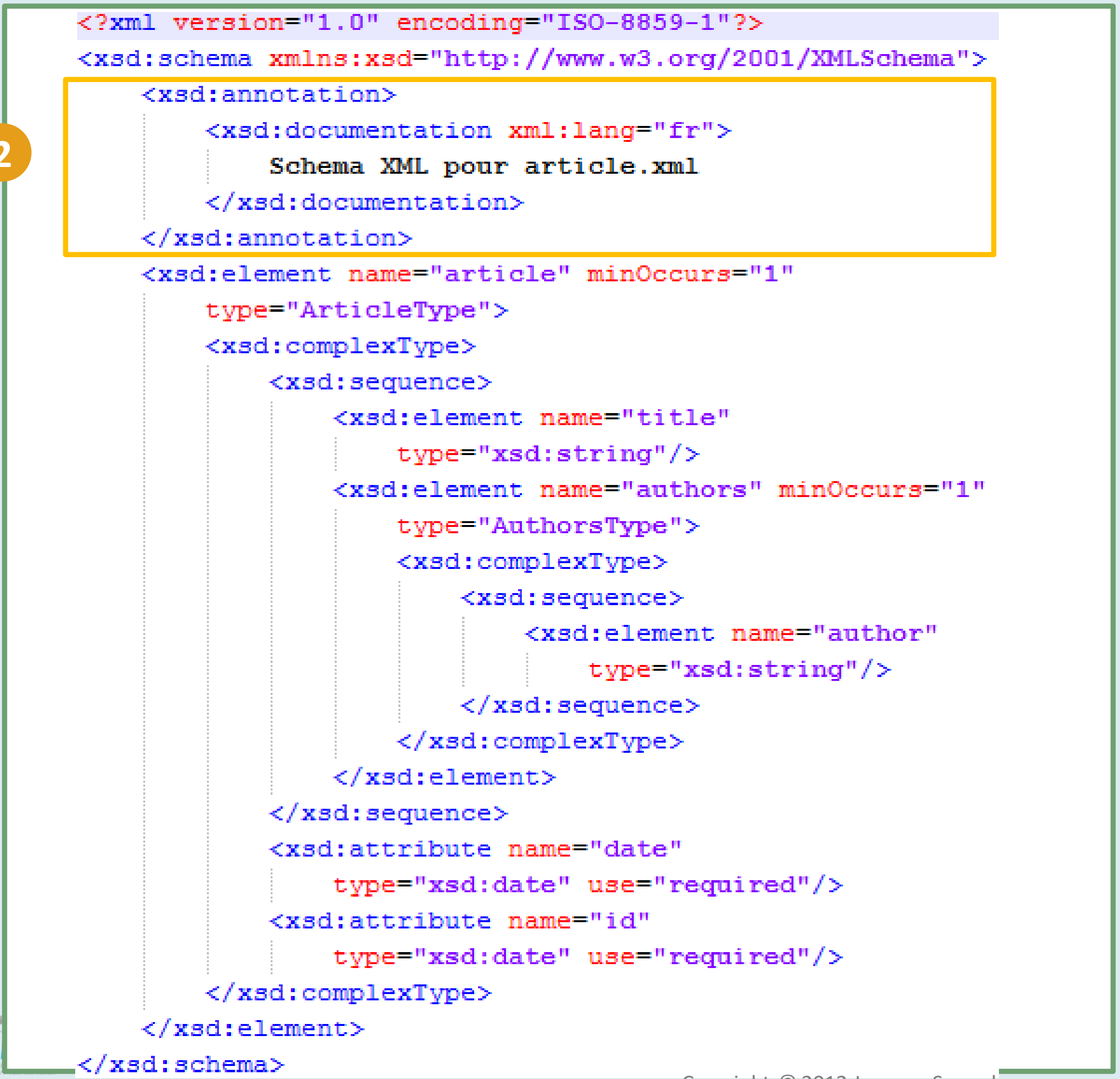
Espace de nommage utilisé pour les schema XML

Schema XML : Syntaxe

- Xsd: schema

Attribut	Définition
targetNamespace	Espace de nom cible
elementFromDefault	Détermine l'appartenance de l'élément à l'espace de nom cible [qualified,unqualified]
attributFromDefault	Détermine l'appartenance de l'attribut à l'espace de nom cible [qualified,unqualified]
blockDefault	Restriction sur la substitution des éléments/attributs définis [#all,extension,restriction, substitution]
finalDefault	Restriction sur la dérivation des éléments/attributs définis [extension, restriction,list,union]

Schema XML : Syntaxe



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:annotation>
        <xsd:documentation xml:lang="fr">
            Schema XML pour article.xml
        </xsd:documentation>
    </xsd:annotation>
    <xsd:element name="article" minOccurs="1"
        type="ArticleType">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="title"
                    type="xsd:string"/>
                <xsd:element name="authors" minOccurs="1"
                    type="AuthorsType">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="author"
                                type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="date"
                type="xsd:date" use="required"/>
            <xsd:attribute name="id"
                type="xsd:date" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

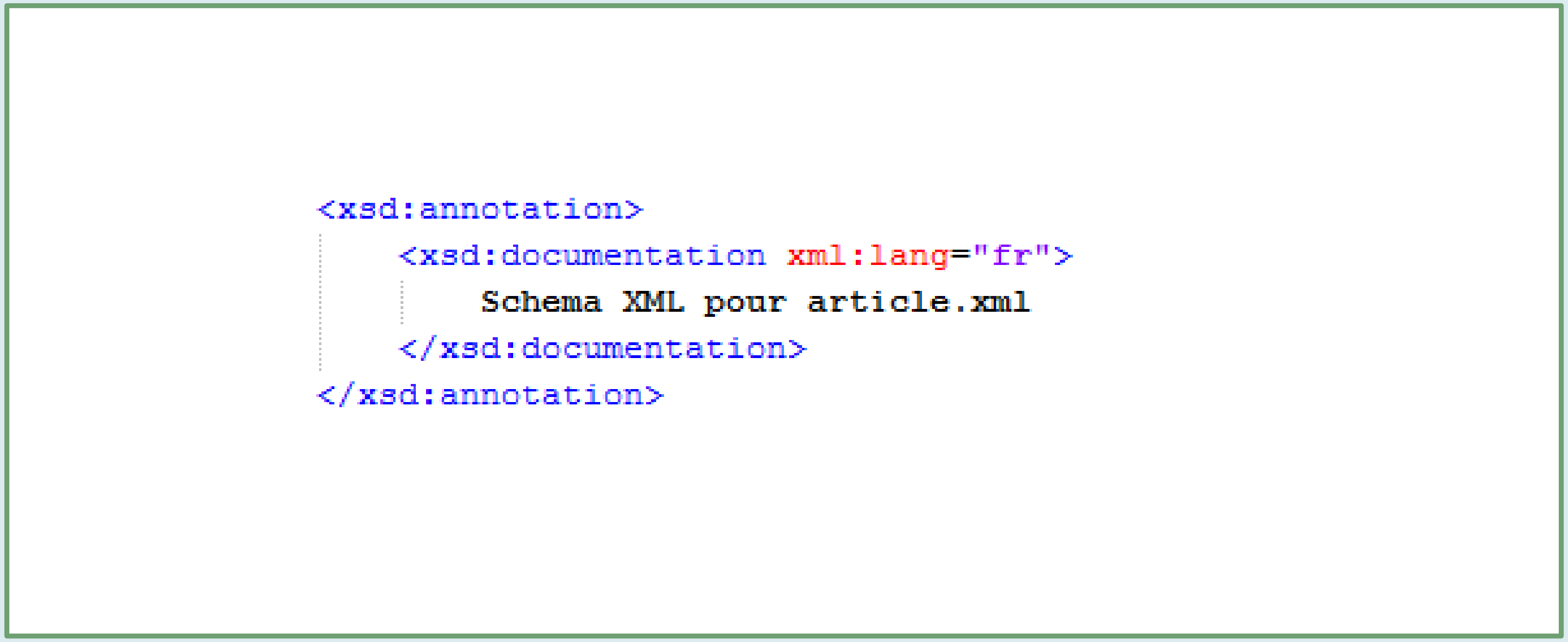
2

1 Espace de nommage utilisé pour les schema XML

2 **Annotation du document**

Schema XML : Syntaxe

- Annotation:
  - Commentaire décrivant le contenu du schema
  - Partie intégrante du schema (différent de <!-- -->)



```
<xsd:annotation>
    <xsd:documentation xml:lang="fr">
        Schema XML pour article.xml
    </xsd:documentation>
</xsd:annotation>
```

Schema XML : Syntaxe

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:annotation>
        <xsd:documentation xml:lang="fr">
            Schema XML pour article.xml
        </xsd:documentation>
    </xsd:annotation>
    <xsd:element name="article" minOccurs="1"
        type="ArticleType">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="title"
                    type="xsd:string"/>
                <xsd:element name="authors" minOccurs="1"
                    type="AuthorsType">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="author"
                                type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="date"
                type="xsd:date" use="required"/>
            <xsd:attribute name="id"
                type="xsd:date" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

3

1 Espace de nommage utilisé pour les schema XML

2 Annotation du document

3 Description d'un élément

- **Schema XML : Syntaxe**
  - Element:
    - Notation : **<xsd:element name=“**elementName**” type=“**elementType**”>**
    - Valeur fixe ou par default
    - Type anonyme possible

```
<xsd:element name="title" type="xsd:string"/>
```

Notation

```
<xsd:element name="title" type="xsd:string"
        default="Undefined Title"/>
<xsd:element name="title" type="xsd:string"
        fixed="Document Title"/>
```

Fixe ou default

```
<xsd:element name="title" >
    <xsd:simpleType>
        ...
    <xsd:simpleType>
</xsd:element>
```

Type anonyme

Schema XML : Syntaxe

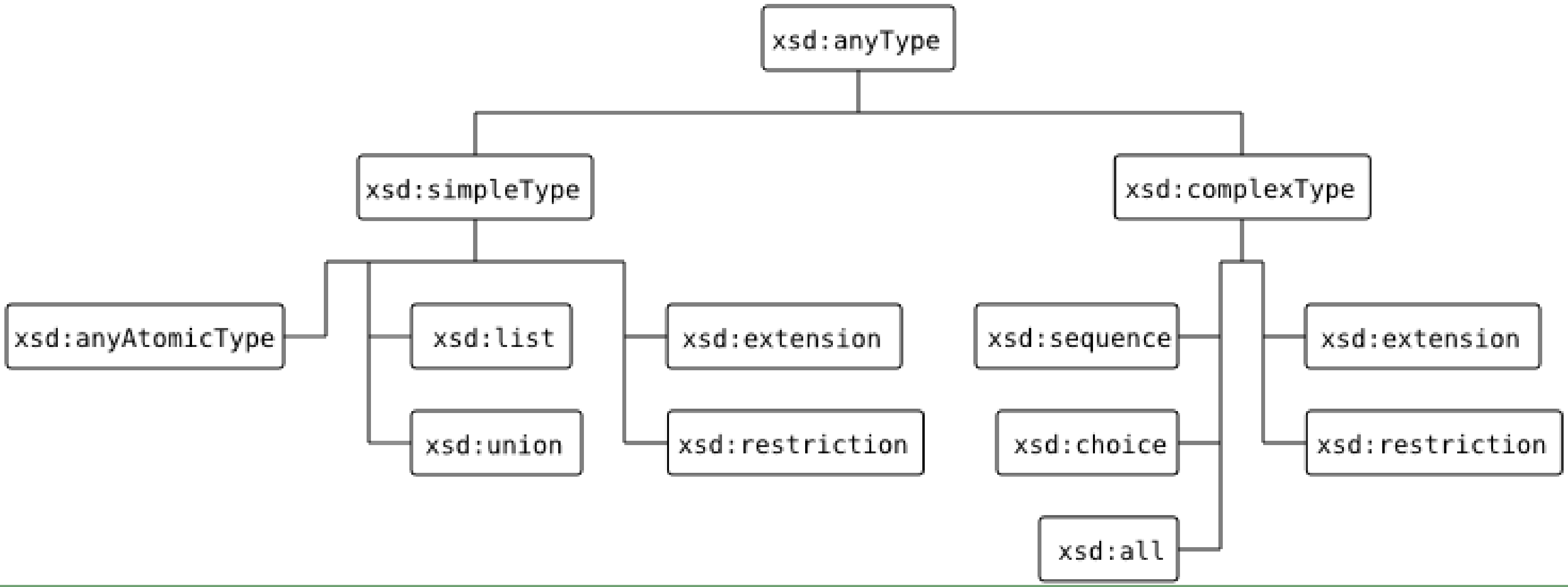
- Element référencement, utilisation:
  - Réutilisation d'un élément utilisation attribut **ref**

```
...
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="date" type="xsd:date"/>
...
<xsd:element name="article" >
    <xsd:complexeType>
        <sequence>
            <xsd:element ref="title"/>
            <xsd:element ref="date" />
        </sequence>
    <xsd:complexeType>
</xsd:element>
```

Schema XML : Syntaxe

- Element Définition des types
  - Types prédéfinis
  - xsd:simpleType
  - xsd:complexType

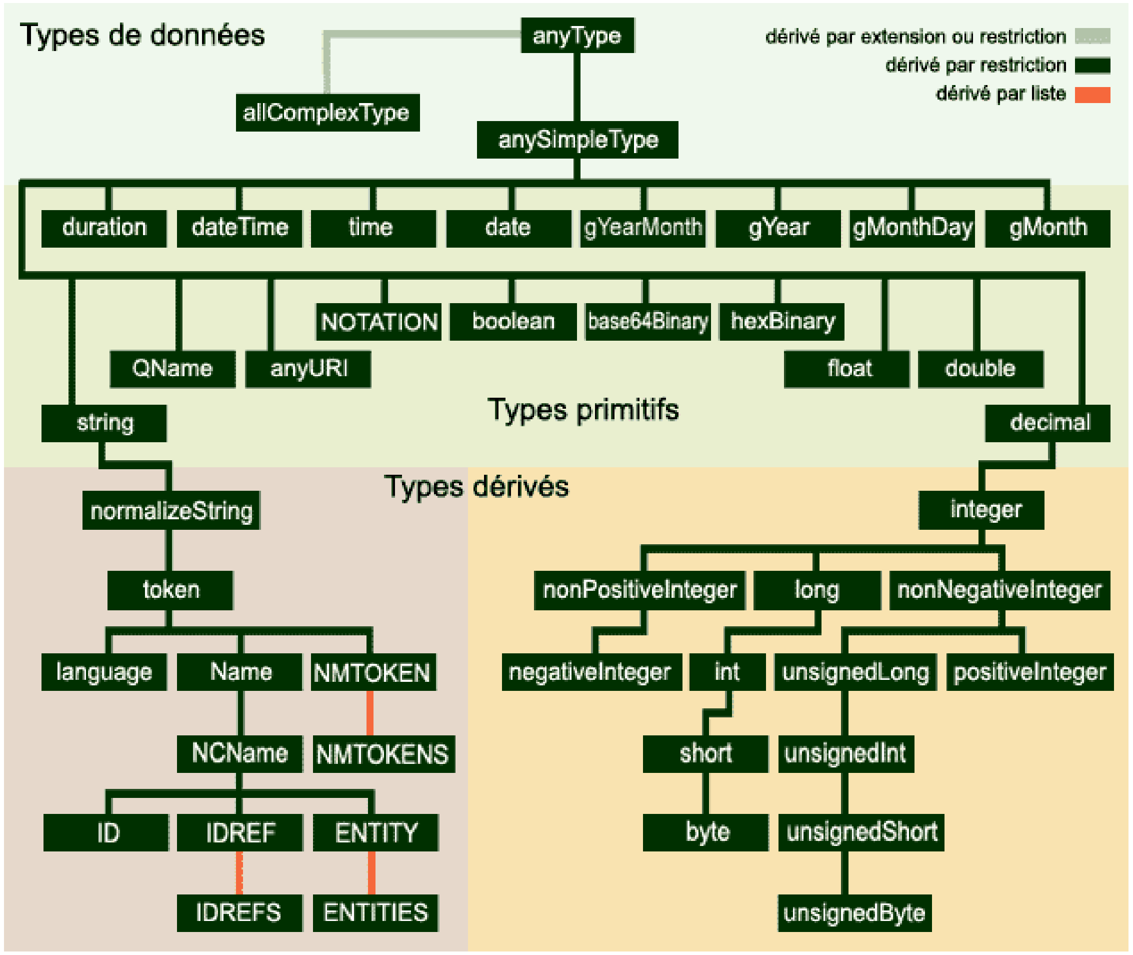
<xsd:element name=“elementName**” type=“**elementType**”>**



Source:http://www.liafa.univ-paris-diderot.fr/~carton/Enseignement/XML/Cours/Schemas/index.html

Schema XML : Syntaxe

- Element : Types prédéfinis



Source:http://www-igm.univ-mlv.fr/~dr/XPOSE2003/xml/contenu_schemas.htm

Schema XML : Syntaxe

- Element : Types prédéfinis

SCHEMA XML

```
...
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="date" type="xsd:date"/>
<xsd:element name="referenceNumber" type="xsd:integer"/>
...
```

XML

```
...
<title> The Risk Theory </title>
<date>2000-01-01<date>
<referenceNumber>452</referenceNumber>
...
```

Schema XML : Syntaxe

- Element: xsd:simpleType
  - Notation : **<xsd:**simpleType … **>** … **</xsd:**simpleType**>**
  - Souvent obtenu par restriction d'autre type
  - Définition globale possible

```
<xsd:simpleType>
    <xsd:restriction base="xsd:integer">
        <xsd:maxInclusive value="100"/>
    </xsd:restriction>
</xsd:simpleType>
```

Notation

```
<xsd:simpleType name="rating">
    <xsd:restriction base="xsd:integer">
        <xsd:maxInclusive value="100"/>
    </xsd:restriction>
</xsd:simpleType>
```

Definition globale

Schema XML : Syntaxe

- Element: xsd:simpleType
  - Opérateurs d'**Union**

```
<xsd:element name="start" type="IntegerOrDate">
<xsd:simpleType name="IntegerOrDate">
    <xsd:union menberType="xsd:interger xsd:date"/>
</xsd:simpleType>
```

```
<start>456312</start>
```
✓

```
<start>2013-01-01</start>
```
✓

```
<start>Second day of the year</start>
```
✗

Schema XML : Syntaxe

- Element: xsd:simpleType
  - Opérateurs de **Liste**

```
<xsd:attribut name="group" type="stringList">
<xsd:simpleType name="group">
    <xsd:list itemType="xsd:string"/>
</xsd:simpleType>


 <contact group="admin system user">
```

Schema XML : Syntaxe

- Element: xsd:complexType
  - conteneur d'autres éléments et d'attributs
  - contrainte sur les éléments contenus (all, sequence, choice)
  - contenu « pur » : uniquement composer d'autres éléments

```
<xsd:element name="contact">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="firstName" type="xsd:string"/>
            <xsd:element name="lastName" type="xsd:string"/>
            <xsd:element name="phoneNumber" type="xsd:unbounded"/>
        </xsd:sequence>
        <xsd:attribut name="id" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
```

Schema XML : Syntaxe

- Element: xsd:complexType

SCHEMA XML

```
<xsd:element name="contact">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="firstName" type="xsd:string"/>
            <xsd:element name="lastName" type="xsd:string"/>
            <xsd:element name="phoneNumber" type="xsd:unbounded"/>
        </xsd:sequence>
        <xsd:attribut name="id" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
```

XML

```
<contact id="ade145">
    First Name is : <firstName>John</firstName>
    Last Name is :  <lastName>Doe</lastName>
    The Phone number: <phoneNumber>666-555-341</phoneNumber>
</contact>
```

Schema XML : Syntaxe

- Element: xsd:complexType
  - Opérateurs de contrainte de contenu 1

Opérateurs	Définition
xsd:all	Ensemble d'élément dans un ordre quelconque
xsd:sequence	Ensemble d'élément dans l'ordre défini
xsd:choice	Ensemble des éléments possibles

• **Schema XML : Syntaxe**

❑ Element: xsd:complexType all

– Opérateurs de contrainte de contenu 1

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstName"
                type="xsd:string"/>
            <xsd:element name="lastName"
                type="xsd:string"/>
            <xsd:element name="phoneNumber"
                type="xsd:unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

SCHEMA XML

```
<contact id="ade145">
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
</contact>
```

✓

✗

```
<contact id="ade145">
    <phoneNumber>666-555-341</phoneNumber>
    <lastName>Doe</lastName>
    <firstName>John</firstName>
</contact>
```

XML

- **Schema XML : Syntaxe**
  - ❑ Element: xsd:complexType
    - – Opérateurs de contrainte de contenu 1

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:all>
            <xsd:element name="firstName"
                type="xsd:string"/>
            <xsd:element name="lastName"
                type="xsd:string"/>
            <xsd:element name="phoneNumber"
                type="xsd:unbounded"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>
```

```
<contact id="ade145">
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
</contact>
```

✓

✓

```
<contact id="ade145">
    <phoneNumber>666-555-341</phoneNumber>
    <lastName>Doe</lastName>
    <firstName>John</firstName>
</contact>
```

SCHE

"all" ne peut pas être imbriqué avec d'autres opérateurs.
"all" est toujours parent de "complexType" ou "complexContent"
"all" ne peut avoir que des "éléments" comme fils

Schema XML : Syntaxe

- Element: xsd:complexType
  - Opérateurs de contrainte de contenu 1

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element name="firstName"
                type="xsd:string"/>
            <xsd:element name="lastName"
                type="xsd:string"/>
            <xsd:element name="phoneNumber"
                type="xsd:unbounded"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

```
<contact id="ade145">
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
</contact>
```
✗

✓
```
<contact id="ade145">
    <firstName>John</firstName>
</contact>
```

SCHEMA XML | **XML**

- **Schema XML : Syntaxe**
  - Element: xsd:complexType
    - Opérateurs de contrainte de contenu 1 : combinaison d'opérateurs

```
<xsd:element name="person">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element name="legalPerson"
                type="xsd:unbounded"/>
            <xsd:sequence>
                <xsd:element name="firstName"
                    type="xsd:string"/>
                <xsd:element name="lastName"
                    type="xsd:string"/>
                <xsd:element name="phoneNumber"
                    type="xsd:unbounded"/>
            </xsd:sequence>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

```
<person>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
</person>
```

✓

```
<person>
    <legalPerson>#1452-542-RC</legalPerson>
</person>
```

✓

SCHEMA XML **XML**

Schema XML : Syntaxe

- Element: xsd:complexType
  - Opérateurs de contrainte de contenu 2 : **minOccurs maxOccurs**

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstName"
                type="xsd:string"
                minOccurs="1"
                maxOccurs="unbounded"/>
            <xsd:element name="lastName"
                type="xsd:string"/>
            <xsd:element name="phoneNumber"
                type="xsd:unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
<contact id="ade145">
    <firstName>John</firstName>
    <firstName>Alfred</firstName>
    <firstName>Teddy</firstName>
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
</contact>
```

SCHEMA XML **XML**

Schema XML : Syntaxe

- Element: xsd:complexType
  - Opérateurs de contrainte de contenu 2 : **minOccurs maxOccurs**

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstName"
                type="xsd:string"
                minOccurs="0"/>
            <xsd:element name="lastName"
                type="xsd:string"/>
            <xsd:element name="phoneNumber"
                type="xsd:unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

SCHEMA XML

```
<contact id="ade145">
    <firstName>John</firstName>
    <firstName>Alfred</firstName>
    <firstName>Teddy</firstName>
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
</contact>
```

✓

```
<contact id="ade145">
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
</contact>
```

✓

XML

Schema XML : Syntaxe

- Element: xsd:complexType
  - Opérateurs de contrainte de contenu 2 : **any**

processContents	Définition
strict	Les éléments doivent être validés par un autre schéma
skip	Pas de validation des éléments
lax	Tentative de validation (echec authorisé)

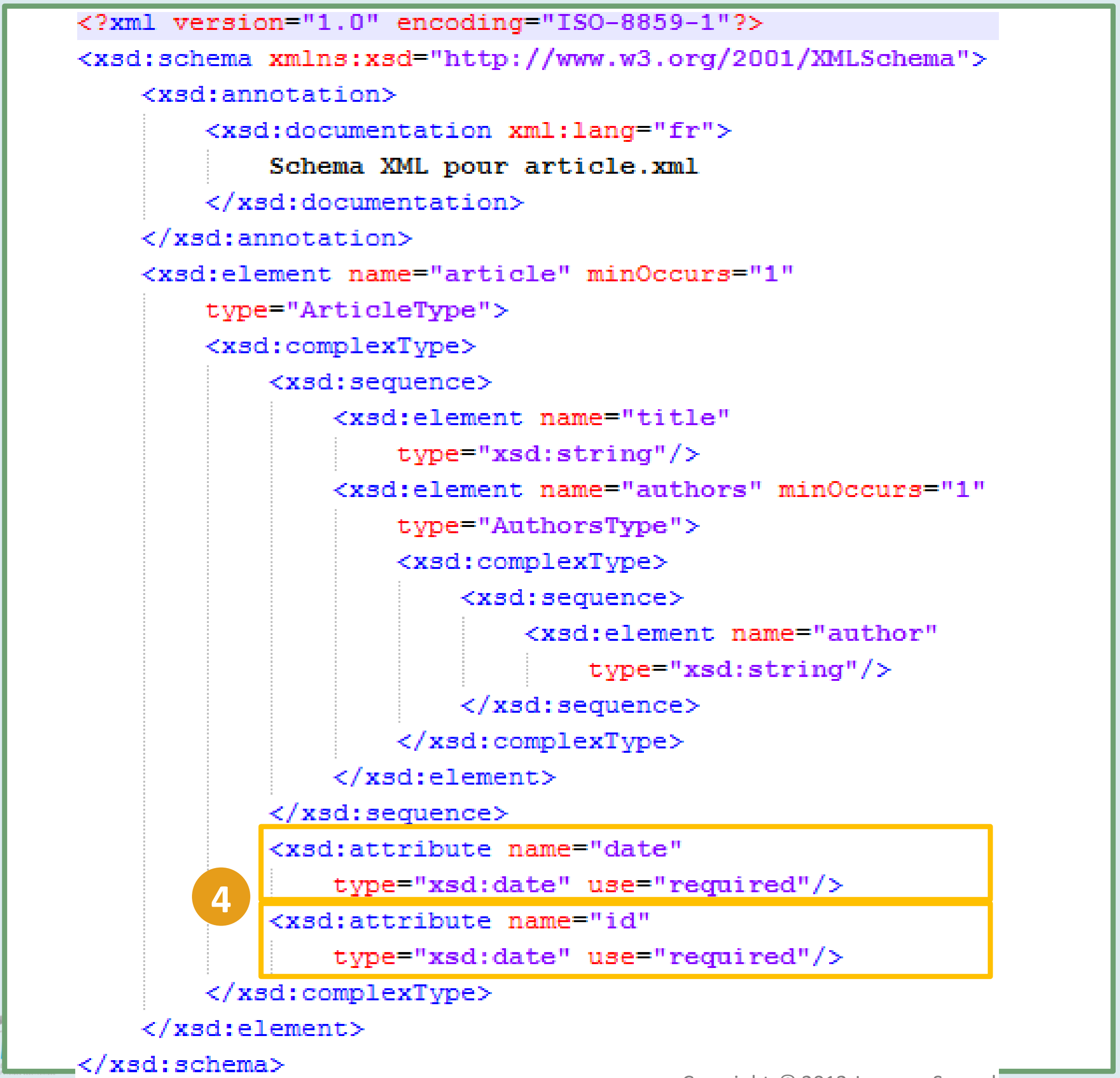
```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstName"
                type="xsd:string"/>
            <xsd:element name="lastName"
                type="xsd:string"/>
            <xsd:any processContents="lax" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

SCHEMA XML

```
<person>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <phoneNumber>666-555-341</phoneNumber>
    <mml:math> ... </mml:math>
</person>
```

XML

• Schema XML : Syntaxe



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:annotation>
        <xsd:documentation xml:lang="fr">
            Schema XML pour article.xml
        </xsd:documentation>
    </xsd:annotation>
    <xsd:element name="article" minOccurs="1"
        type="ArticleType">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="title"
                    type="xsd:string"/>
                <xsd:element name="authors" minOccurs="1"
                    type="AuthorsType">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="author"
                                type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="date"
                type="xsd:date" use="required"/>
            <xsd:attribute name="id"
                type="xsd:date" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

1 Espace de nommage utilisé pour les schema XML

2 Annotation du document

3 Description d’un élément

4 **Description d’un attribut**

Schema XML : Syntaxe

- Attribut:
  - Notation : **<xsd:attribut name="**elementName**" type="**elementType**">**
  - Nécessairement type simple
  - L'ordre des attributs n'a pas d'importance
  - Attribut global possible

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:attribut name="identifier" type="xsd:string"/>
        <xsd:attribut name="groupName" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
```

Notation

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:attribut name="identifier" type="xsd:string"/>
    <xsd:element name="contact">
        ...
```

Attribut global

Schema XML : Syntaxe

- Attribut:
  - usage : **optional**, **required** ou **prohibited**

SCHEMA XML

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:attribut name="name" type="xsd:string" use="required"/>
        <xsd:attribut name="groupName" type="xsd:string" use="optional"/>
        <xsd:attribut name="password" type="xsd:string" use="prohibited"/>
    </xsd:complexType>
</xsd:element>
```

XML

```
<contact name="jDoe" groupeName="system"/>
```
✓

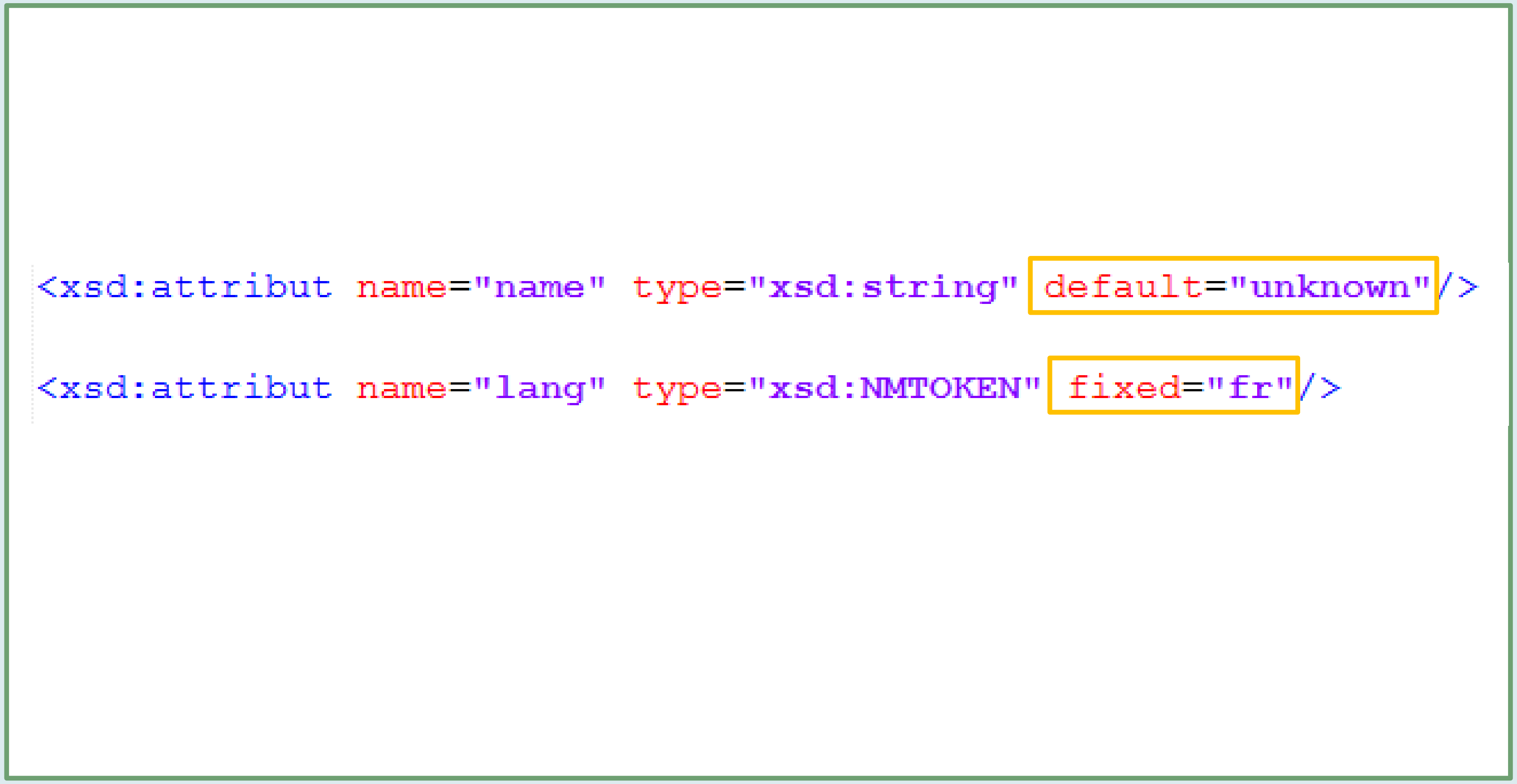
```
<contact name="jDoe" />
```
✓

```
<contact groupeName="system"/>
```
✗

```
<contact name="jDoe"  password="01234"/>
```
✗

Schema XML : Syntaxe

- Attribut:
  - valeur par default et valeur fixe



```
<xsd:attribut name="name" type="xsd:string" default="unknown"/>
<xsd:attribut name="lang" type="xsd:NMTOKEN" fixed="fr"/>
```

Schema XML : Syntaxe

- Attribut:
  - attribut: **any**

processContents	Définition
strict	Les attributs doivent être validés par un autre schéma
skip	Pas de validation des attributs
lax	Tentative de validation (echec authorisé)

SCHEMA XML

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:anyAttribut processContents="skip" />
    </xsd:complexType>
</xsd:element>
```

XML

```
<contact ref="j#41"  dd:dateCreation="2013-01-02"/>
```

• Schema XML : A vous de jouer!

```
<book>
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="A213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        </document>
        <copyright>2012</copyright>
    </article>
  </category>
  <category name="romance" />
</book>
```

Book=n category
Category = n articles
Authors= min 1 max 5
Article = min 1 document max 5

Schema XML : Opération sur les types

- ❑ Créer de nouveaux types d'éléments/attributs à partir de type existant.
- ❑ Extension de type
- ❑ Restriction de type
- ❑ Substitution de type

Schema XML : Opération sur les types

- Extension
  - Notation: **<xsd:extension base="** *reference* **">** new_content **</xsd:extension>**
  - Ajouter de nouvelles propriétés à un type existant

SCHEMA XML

```
<xsd:complexType name="rating">
    <xsd:simpleContent>
        <xsd:extension base="xsd:decimal">
            <xsd:attribut name="totalOfData" type="xsd:integer"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

XML

```
<rating totalOfData="100">4.3</rating>
```

• Schema XML : Opération sur les types

❑ Extension

SCHEMA XML

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstName'
                type="xsd:string"/>
            <xsd:element name="lastName"
                type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="detailedContact">
    <xsd:complexType>
        <xsd:extension base="contact">
            <xsd:sequence>
                <xsd:element name="title"
                    type="xsd:string"/>
                <xsd:element name="address"
                    type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexType>
</xsd:element>
```

XML

```
<contact>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
</contact>
```

```
<contact>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <title>Professor</title>
    <address>
        Central Intelligence Agency
        Office of Public Affairs
        Washington, D.C. 20505
    </address>
</contact>
```

- **Schema XML : Opération sur les types**
  - ❑ Restriction
    - –Notation: **<xsd:restriction base=" *reference* " >** rules **</xsd:restriction>**
    - – Restreindre les propriétés d'un type existant

SCHEMA XML

```
<xsd:complexType name="rating">
    <xsd:simpleContent>
        <xsd:restriction base="xsd:decimal">
            <xsd:minInclusive value="0.0"/>
            <xsd:maxInclusive value="5.0"/>
        </xsd:restriction>
    </xsd:simpleContent>
</xsd:complexType>
```

XML

```
<rating>4.3</rating>    ✓
<rating>8</rating>      ✗
<rating>-5</rating>     ✗
```

Schema XML : Opération sur les types

- Restriction: Enumeration

SCHEMA XML

```
<xsd:elementType name="MyLanguage">
    <xsd:restriction base="xsd:language">
        <xsd:enumeration value="de"/>
        <xsd:enumeration value="en"/>
        <xsd:enumeration value="fr"/>
    </xsd:restriction>
</xsd:elementType>
```

XML

```
<xsd:MyLanguage>fr</xsd:MyLanguage>
```
✓

```
<xsd:MyLanguage>uk</xsd:MyLanguage>
```
✗

Schema XML : Opération sur les types

- Restriction: Pattern

SCHEMA XML

```
<xsd:elementType name="login">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[a-zA-Z0-9]+"/>
    </xsd:restriction>
</xsd:elementType>
```

XML

```
<login>jDoe95</login>
```
✓

```
<login>jDoe_95</login>
```
✗

Schema XML : Opération sur les types

- Restriction: Sequence

SCHEMA XML

```
<xsd:element name="contact">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element name="personal"
                type="xsd:string"/>
            <xsd:element name="professional"
                type="xsd:string"/>
            <xsd:element name="undefined"
                type="xsd:string"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="simpleContact">
    <xsd:complexType>
        <xsd:restriction base="contact">
            <xsd:choice>
                <xsd:element name="undefined"
                    type="xsd:string"/>
            </xsd:choice>
        </xsd:restriction>
    </xsd:complexType>
</xsd:element>
```

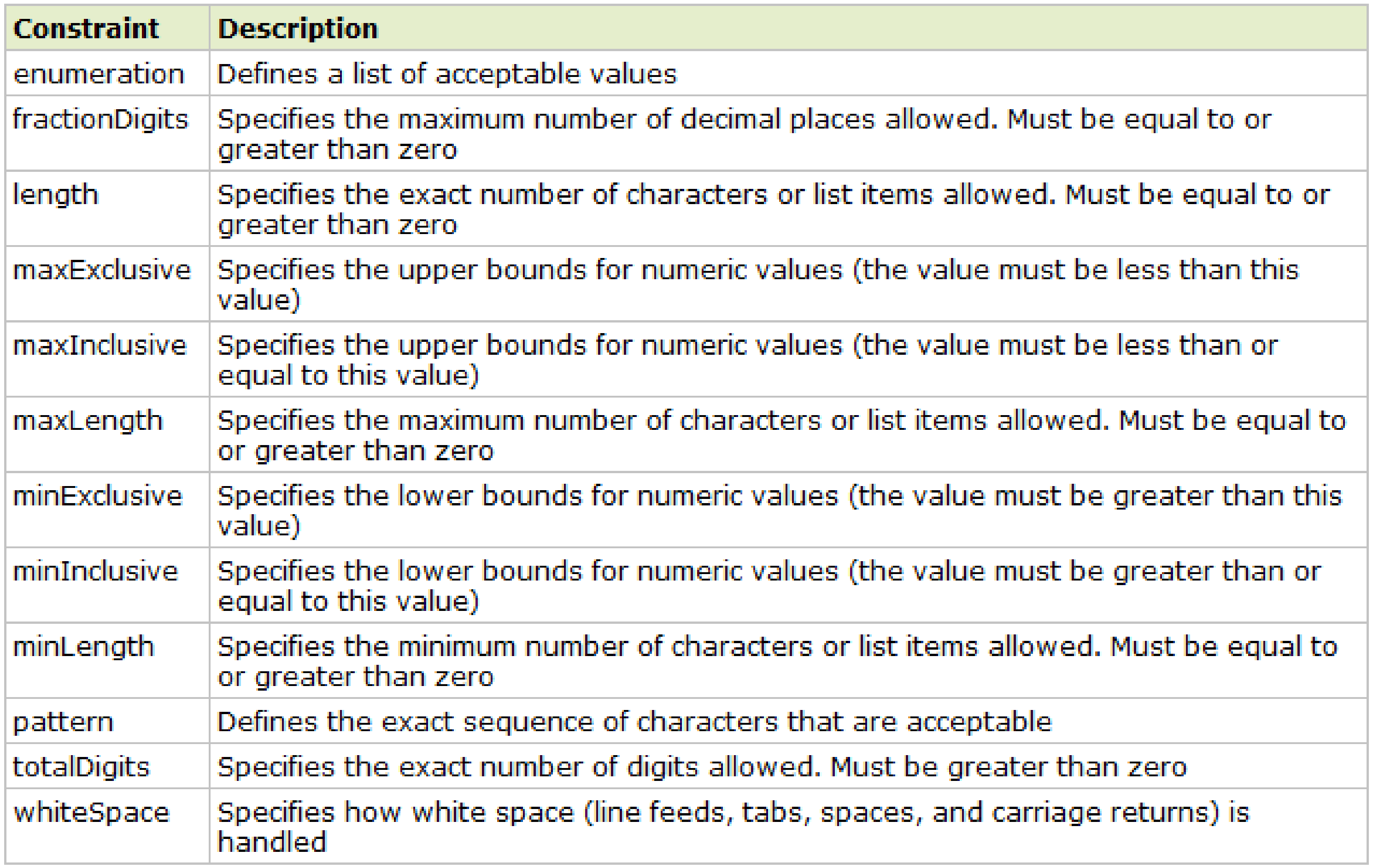
XML

```
<contact>
    <undefined>sandBox</undefined>
<contact>
```

```
<contact>
    <personal>MyFriend</personal>
</contact>
```

Schema XML : Opération sur les types

- Restriction: sum up



Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Source: http://www.w3schools.com/schema/schema_facets.asp

Schema XML : Opération sur les types

- Restriction: Substitution
  - Fournir une alternative au type de l'élément de base

SCHEMA XML

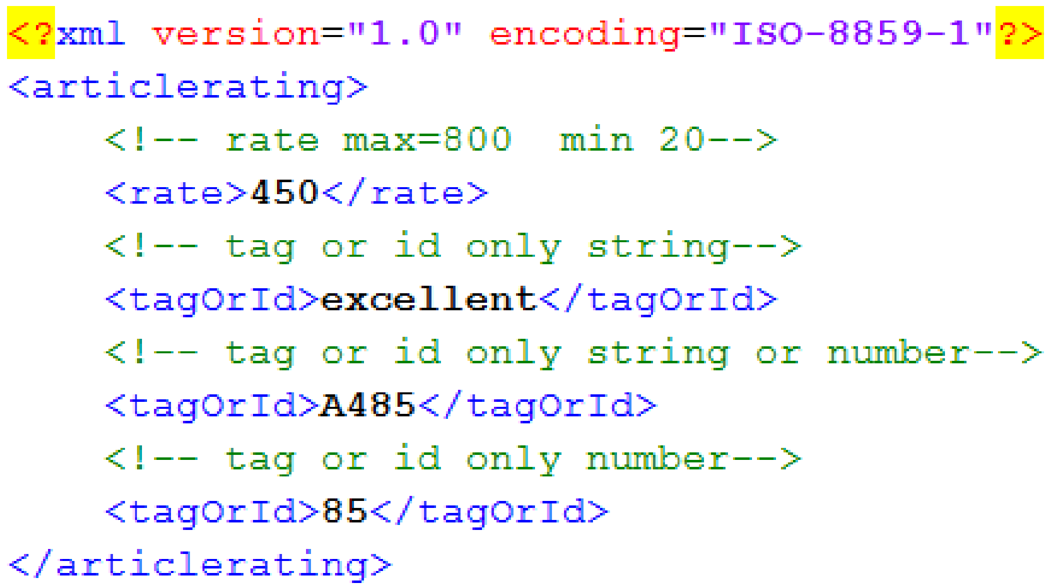
```
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="nom" substitutionGroup="name"/>
<xsd:element name="apellido" substitutionGroup="name"/>
```

XML

```
<name>jDoe</name>
<nom>aDuchant</nom>
<apellido>eRodriguez</apellido>
<cognome>eMatserati</cognome>
```

`<name>jDoe</name>`	✓
`<nom>aDuchant</nom>`	✓
`<apellido>eRodriguez</apellido>`	✓
`<cognome>eMatserati</cognome>`	✗

Schema XML : A vous de jouer!



```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<articlerating>
    <!-- rate max=800  min 20-->
    <rate>450</rate>
    <!-- tag or id only string-->
    <tagOrId>excellent</tagOrId>
    <!-- tag or id only string or number-->
    <tagOrId>A485</tagOrId>
    <!-- tag or id only number-->
    <tagOrId>85</tagOrId>
</articlerating>
```

Schema XML : Contrôle d'opération sur les types

- Restreindre les opérations sur les éléments/types existants
  - abstract : block utilisation d'un Type
  - block
  - final

Attribut	Définition
abstract	Bloque l'utilisation d'un type ou la présence d'un élément
block	Bloque la substitution d'un type ou d'un élément dans tout le document
final	Bloque la dérivation d'un type dans le schema ou l'ajout d'élément dans le groupe de substitution dans le schema

Schema XML : Contrôle d'opération sur les types

❑ Controles

```
<xs:element name="name" type="xs:string" block="substitution"/>

<xs:element name="name" type="xs:string" final="extension"/>

<xs:element name="name" type="xs:string" final="extension restriction"/>
```

```
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="nom" substitutionGroup="name"/>
```

❌

```
<xsd:complexType name="corporateName">
    <xsd:simpleContent>
        <xsd:extension base="name">
            <xsd:attribut name="loginId" type="xsd:string"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

❌

Schema XML : Contraintes d'unicité et d'existance

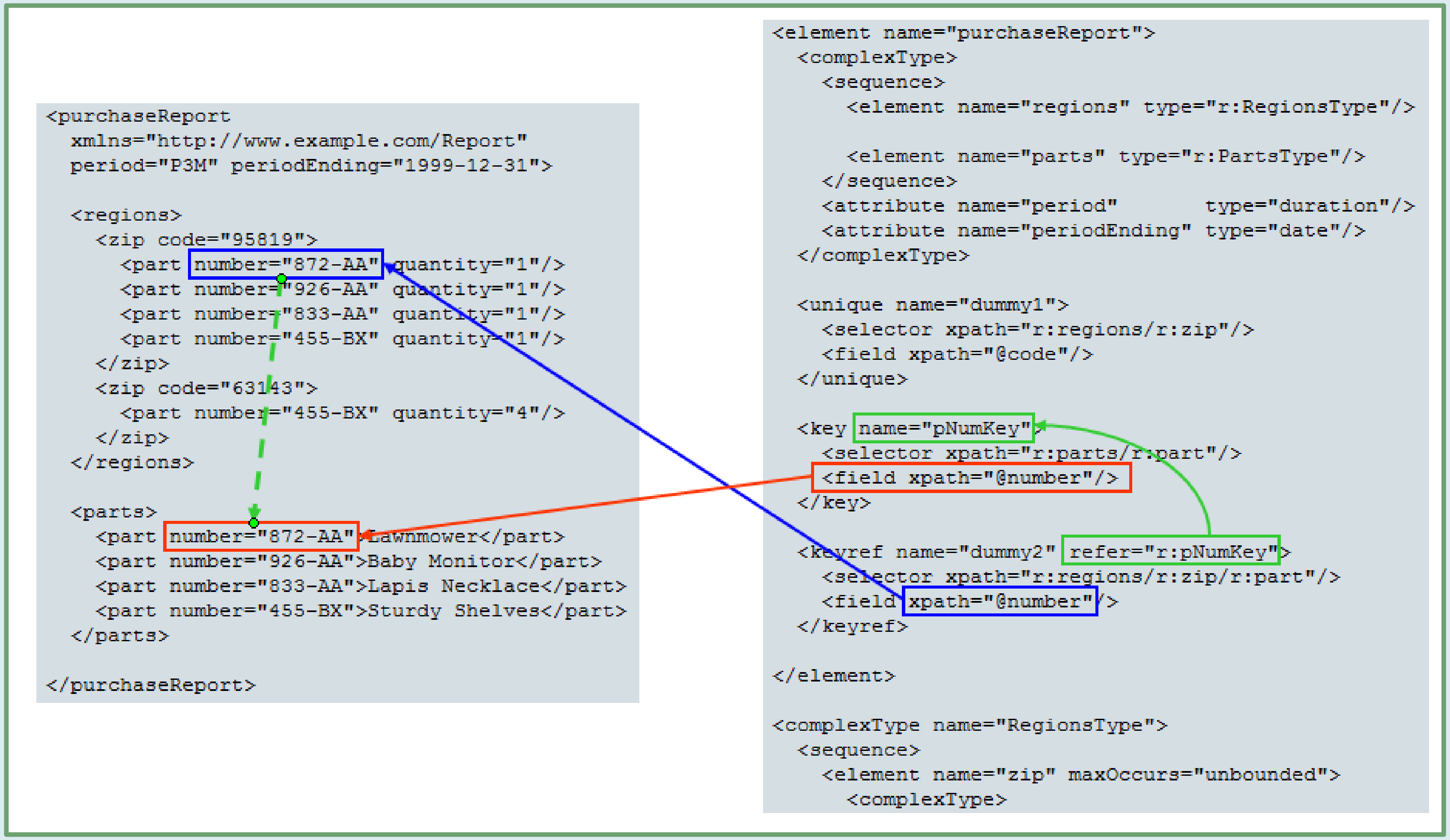
- key : spécifie un attribut ou un élément
- unique

Attribut	Définition
Key	Spécifie qu'un attribut ou qu'une valeur (ou ensemble de valeurs) d'élément doit être une clé dans la portée spécifiée. La portée d'une clé est l'élément **element** conteneur dans un document d'instance. Une clé signifie que les données doivent être uniques dans une portée spécifiée, qui ne peuvent prendre de valeur Null et toujours présentes.
unique	Définie qu'un élément ou la valeur d'un attribut doit être unique dans le périmètre spécifié

Source: http://msdn.microsoft.com/fr-fr/library/ms256101(v=vs.80).aspx

Schema XML : Contraintes d'unicité et d'existance

- Key / Unique

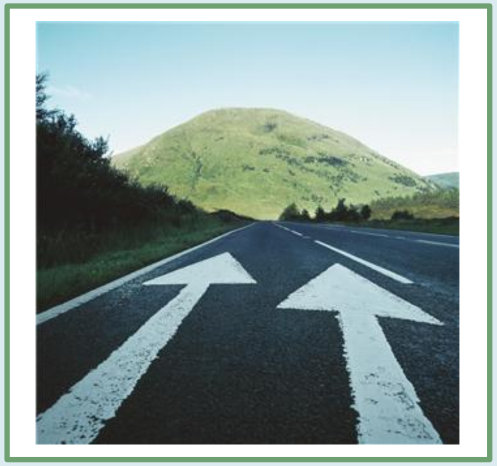


```
<purchaseReport
  xmlns="http://www.example.com/Report"
  period="P3M" periodEnding="1999-12-31">

  <regions>
    <zip code="95819">
      <part number="872-AA" quantity="1"/>
      <part number="926-AA" quantity="1"/>
      <part number="833-AA" quantity="1"/>
      <part number="455-BX" quantity="1"/>
    </zip>
    <zip code="63143">
      <part number="455-BX" quantity="4"/>
    </zip>
  </regions>

  <parts>
    <part number="872-AA">Lawnmower</part>
    <part number="926-AA">Baby Monitor</part>
    <part number="833-AA">Lapis Necklace</part>
    <part number="455-BX">Sturdy Shelves</part>
  </parts>

</purchaseReport>
```

```
<element name="purchaseReport">
  <complexType>
    <sequence>
      <element name="regions" type="r:RegionsType"/>

      <element name="parts" type="r:PartsType"/>
    </sequence>
    <attribute name="period"       type="duration"/>
    <attribute name="periodEnding" type="date"/>
  </complexType>

  <unique name="dummy1">
    <selector xpath="r:regions/r:zip"/>
    <field xpath="@code"/>
  </unique>

  <key name="pNumKey">
    <selector xpath="r:parts/r:part"/>
    <field xpath="@number"/>
  </key>

  <keyref name="dummy2" refer="r:pNumKey">
    <selector xpath="r:regions/r:zip/r:part"/>
    <field xpath="@number"/>
  </keyref>

</element>

<complexType name="RegionsType">
  <sequence>
    <element name="zip" maxOccurs="unbounded">
      <complexType>
```

XML | SCHEMA XML

- **Schema XML : Pour aller plus loin**
  - ❑ Contrainte d’existence
  - ❑ Groupe d’attribut
  - ❑ http://www.w3.org/XML/



Transformation

- XPath
- XSLT

XSL

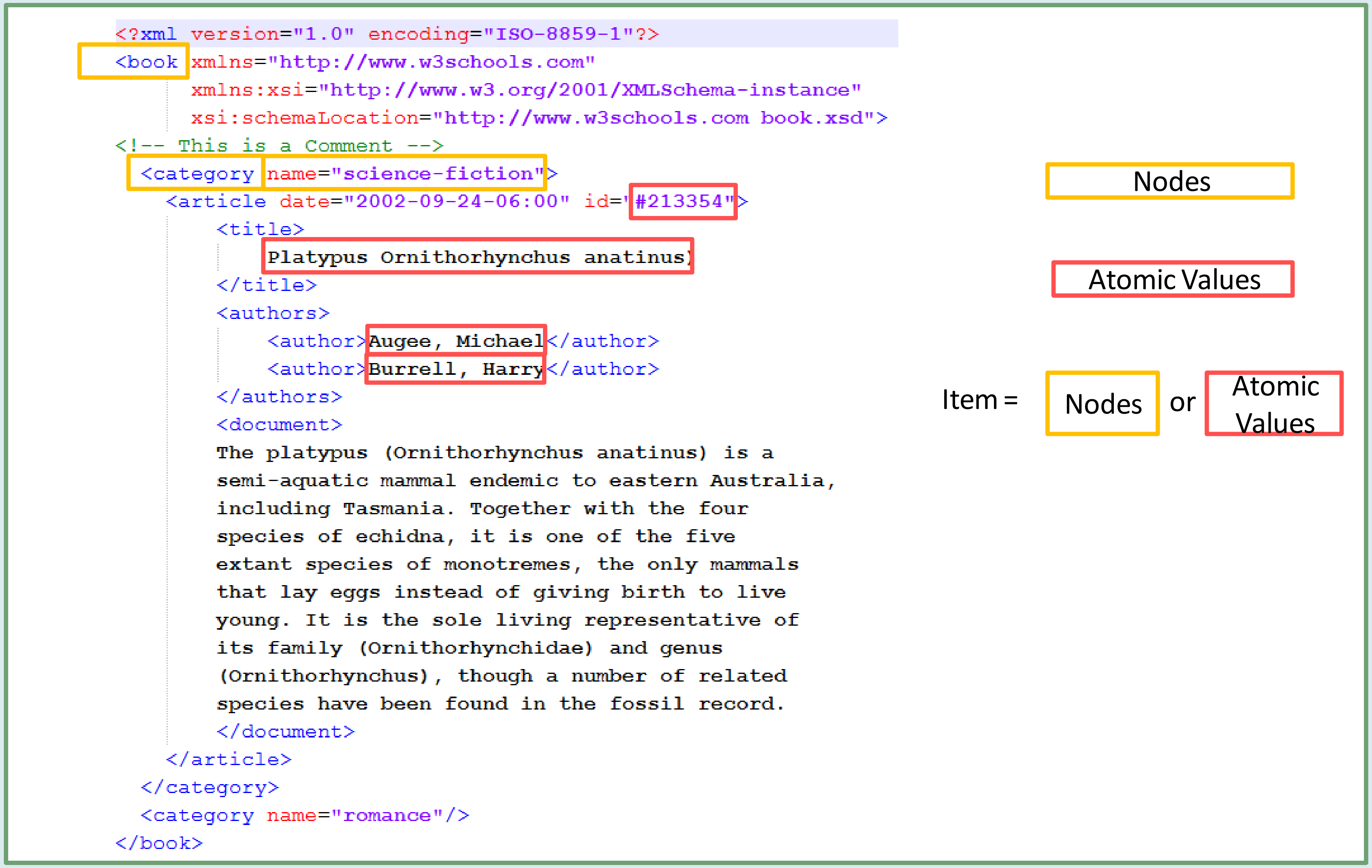
- EXtensible Stylesheet Language
  - → Besoin de formater les documents XML
- CSS permet de formater les documents HTML
- XSL pas de tags prédéfinis
- XSL permet de formater les documents XML
- Composé en 3 parties:
  - XPATH: navigation
  - XSLT: transformation
  - XSL-FO: formatage

- **XPATH**
  - ❑ Langage de découverte d’information dans un document XML
  - ❑ Syntaxe définissant les parties d’un document XML
  - ❑ Utilisé pour naviguer dans un document XML
  - ❑ Principalement utilisé dans XSLT
  - ❑ Recommandation W3C

XPATH : Terminology

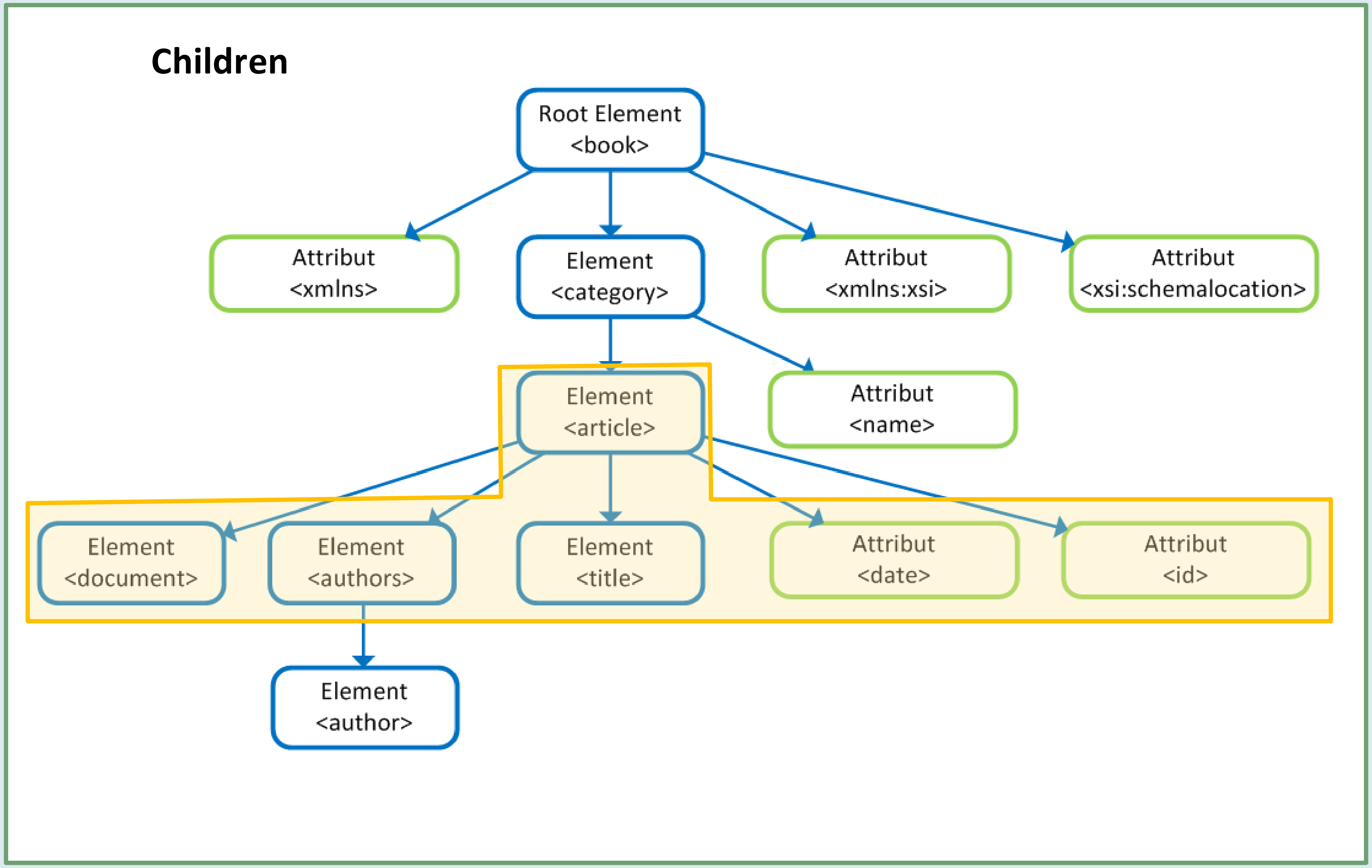
- Description d'un document xml au travers de 3 concepts:
  - Nodes
  - Atomic Values
  - Items
- Navigation au sein document xml au travers de liens entre les noeuds:
  - Parent
  - Children
  - Siblings
  - Ancestrors
  - Descendants

• XPATH : Terminology

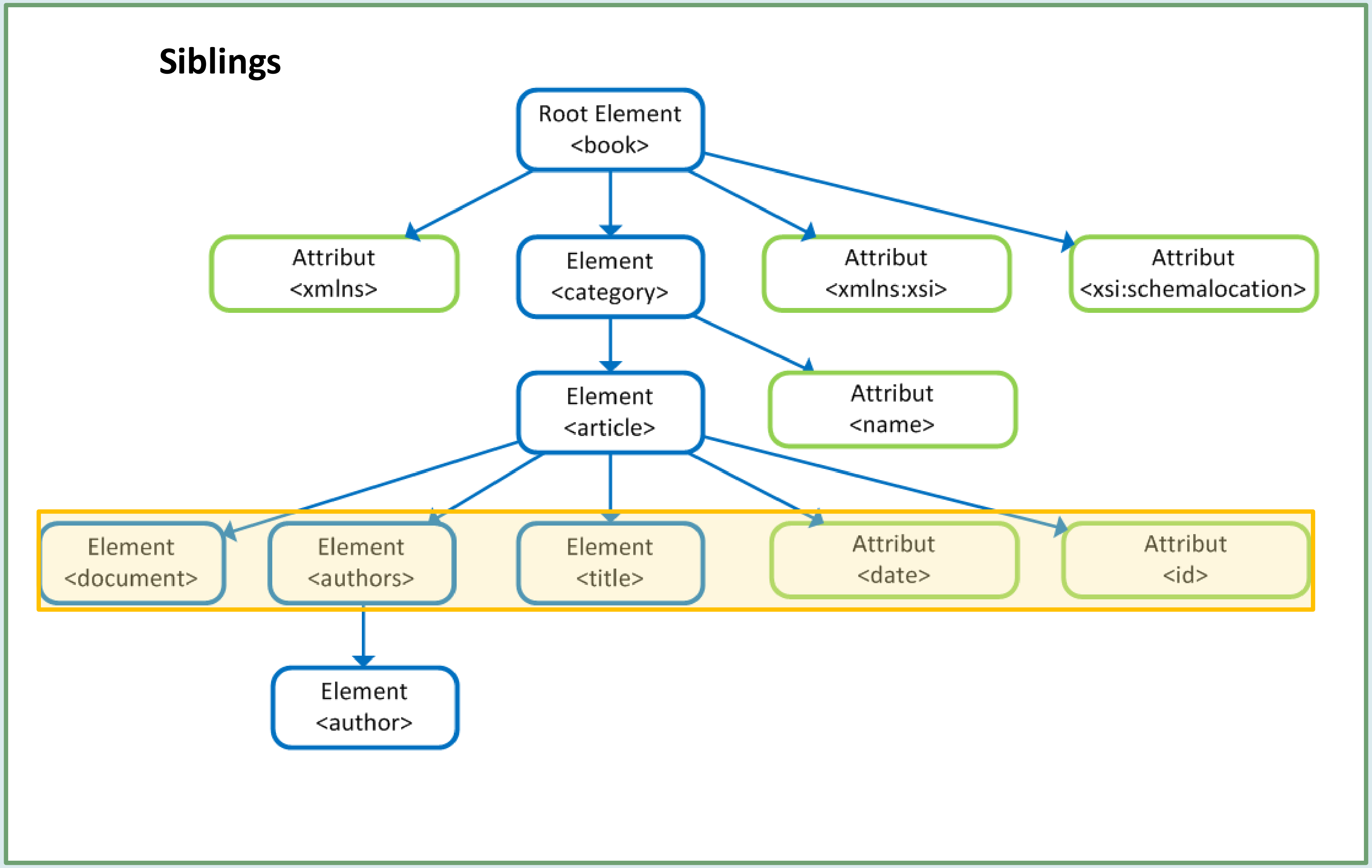


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

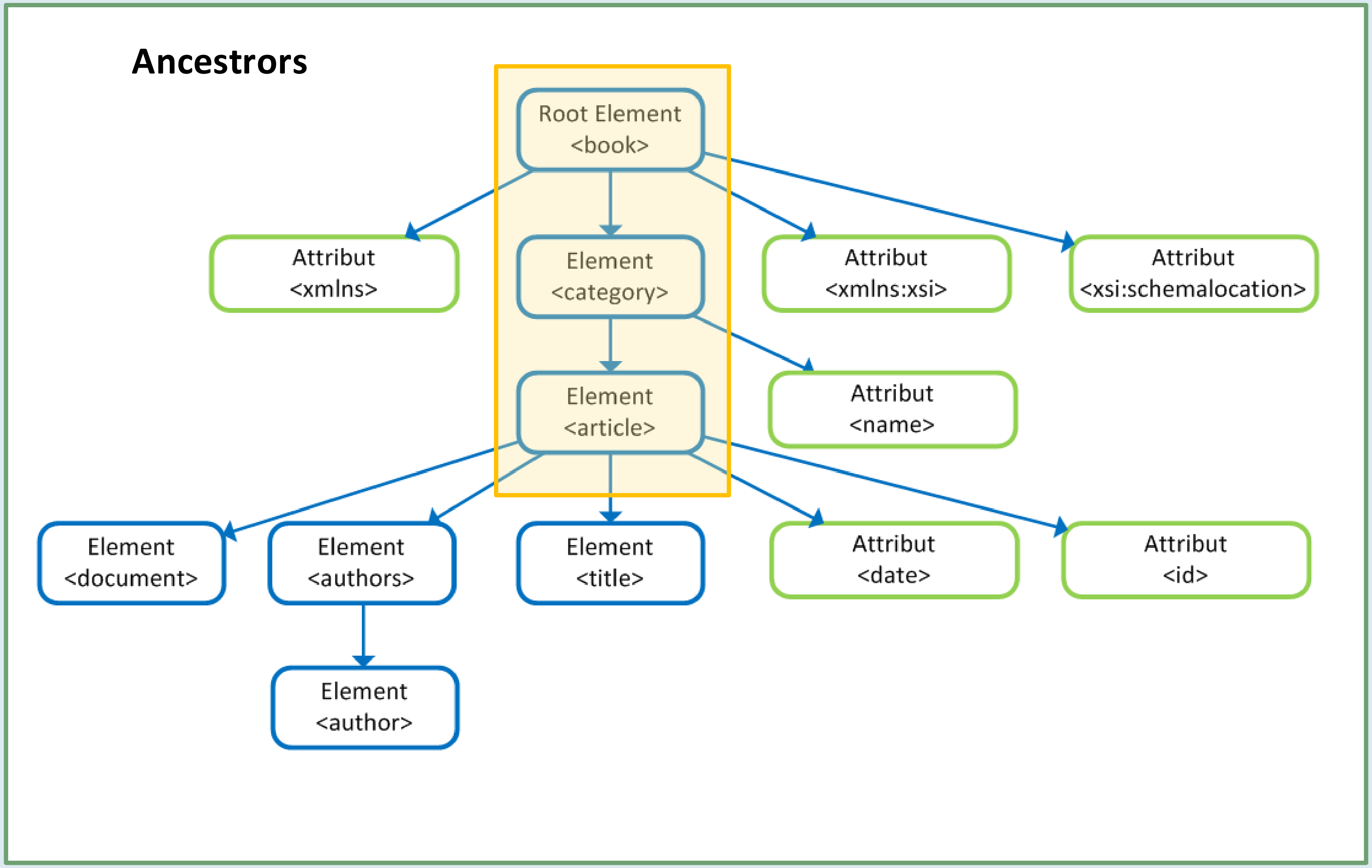
XPATH : Terminology



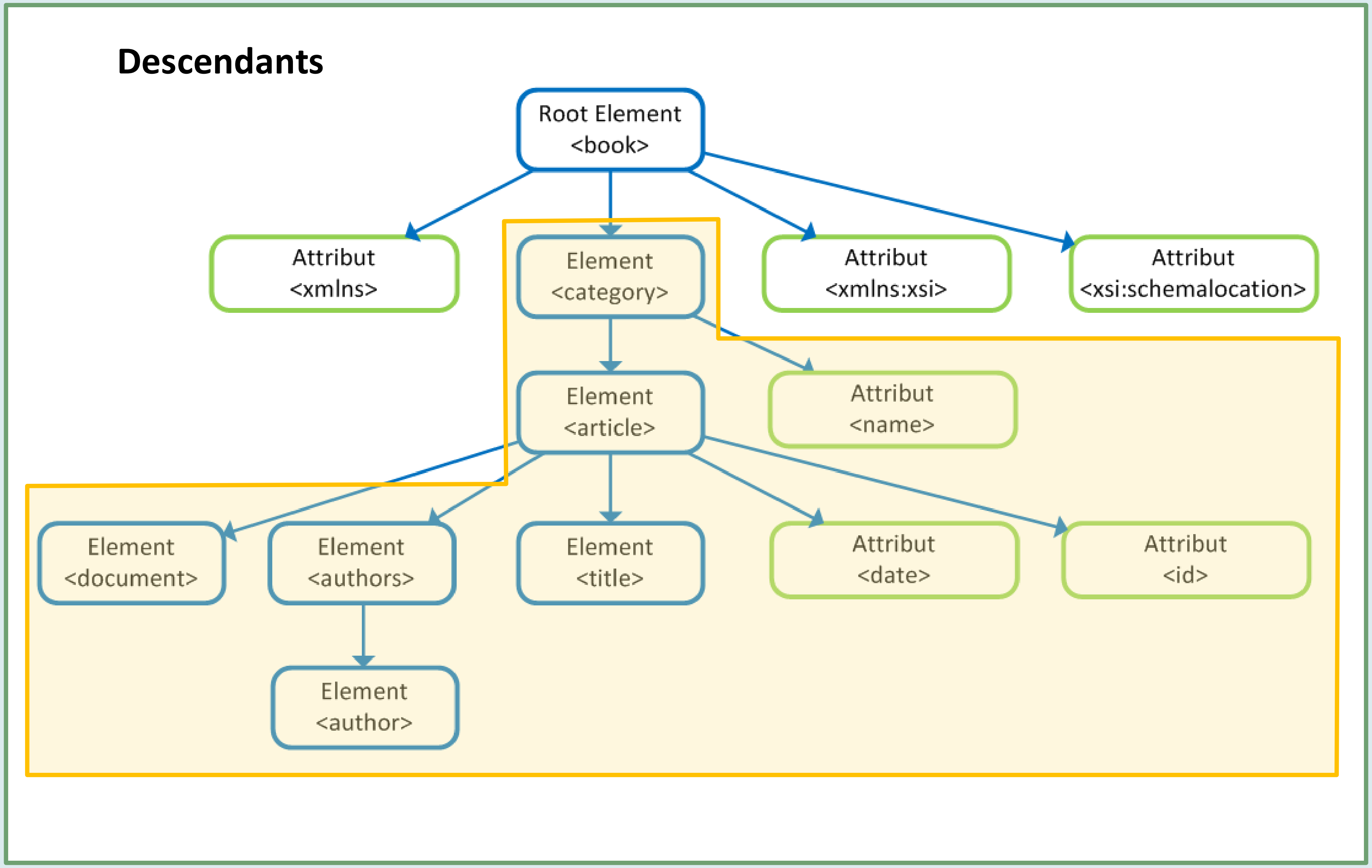
• XPATH : Terminology



• XPATH : Terminology

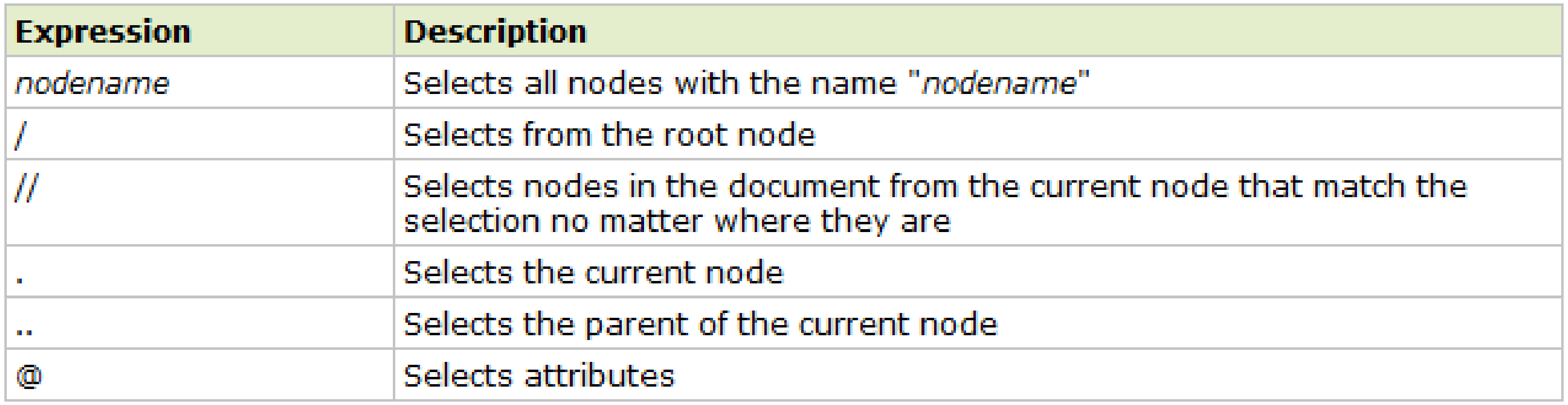


• XPATH : Terminology



XPATH : syntaxe

- Path expression

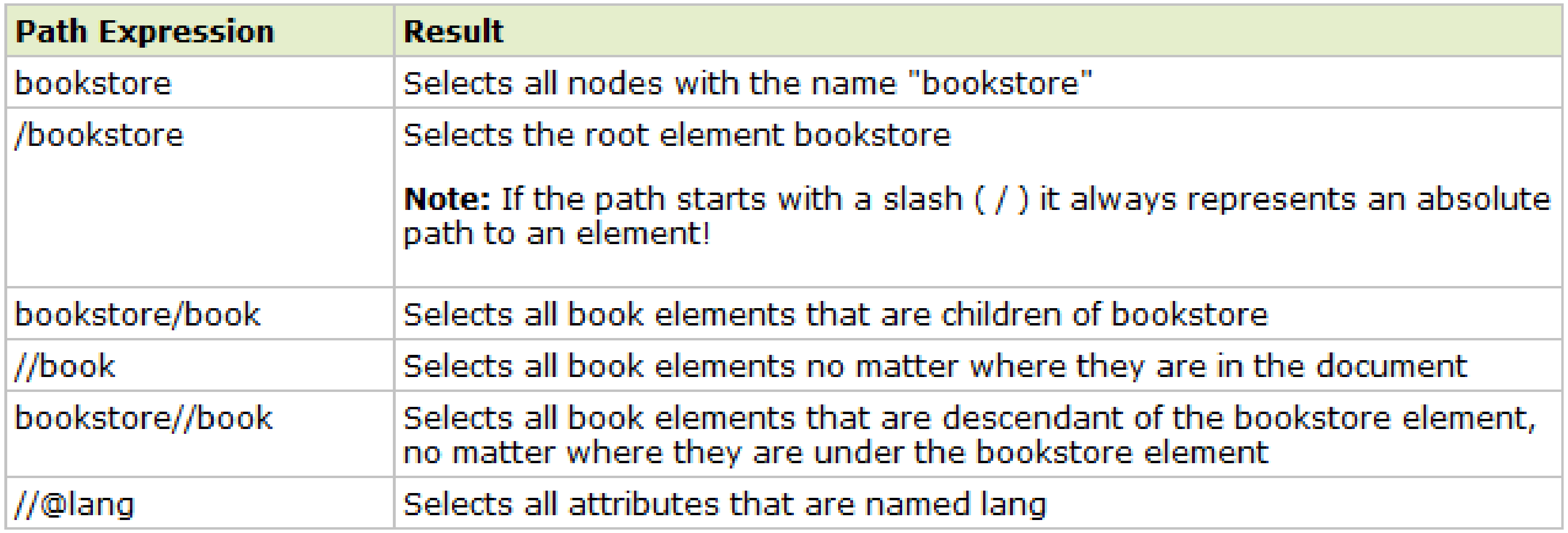


Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Source: http://www.w3schools.com/xpath/xpath_syntax.asp

XPATH : syntaxe

- Path expression

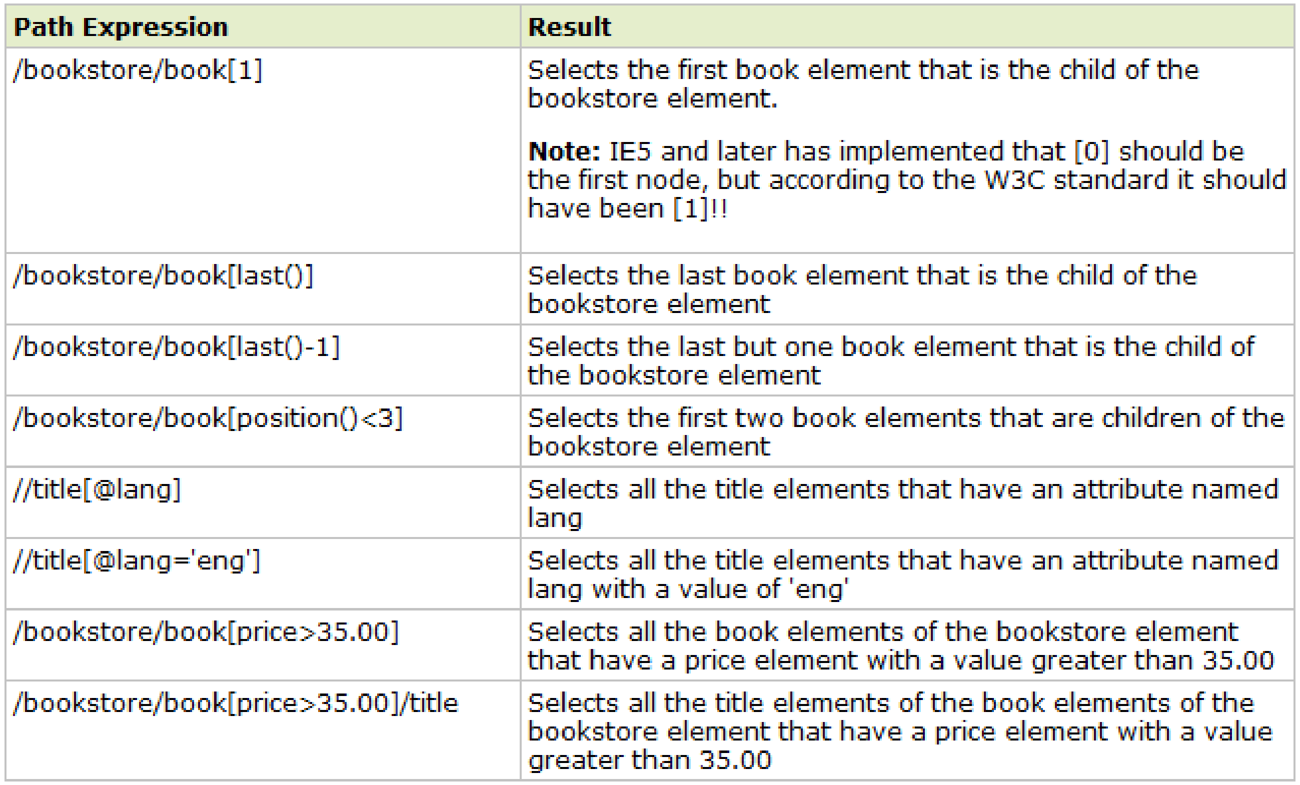


Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

Source: http://www.w3schools.com/xpath/xpath_syntax.asp

XPATH : syntaxe

- Condition de sélection

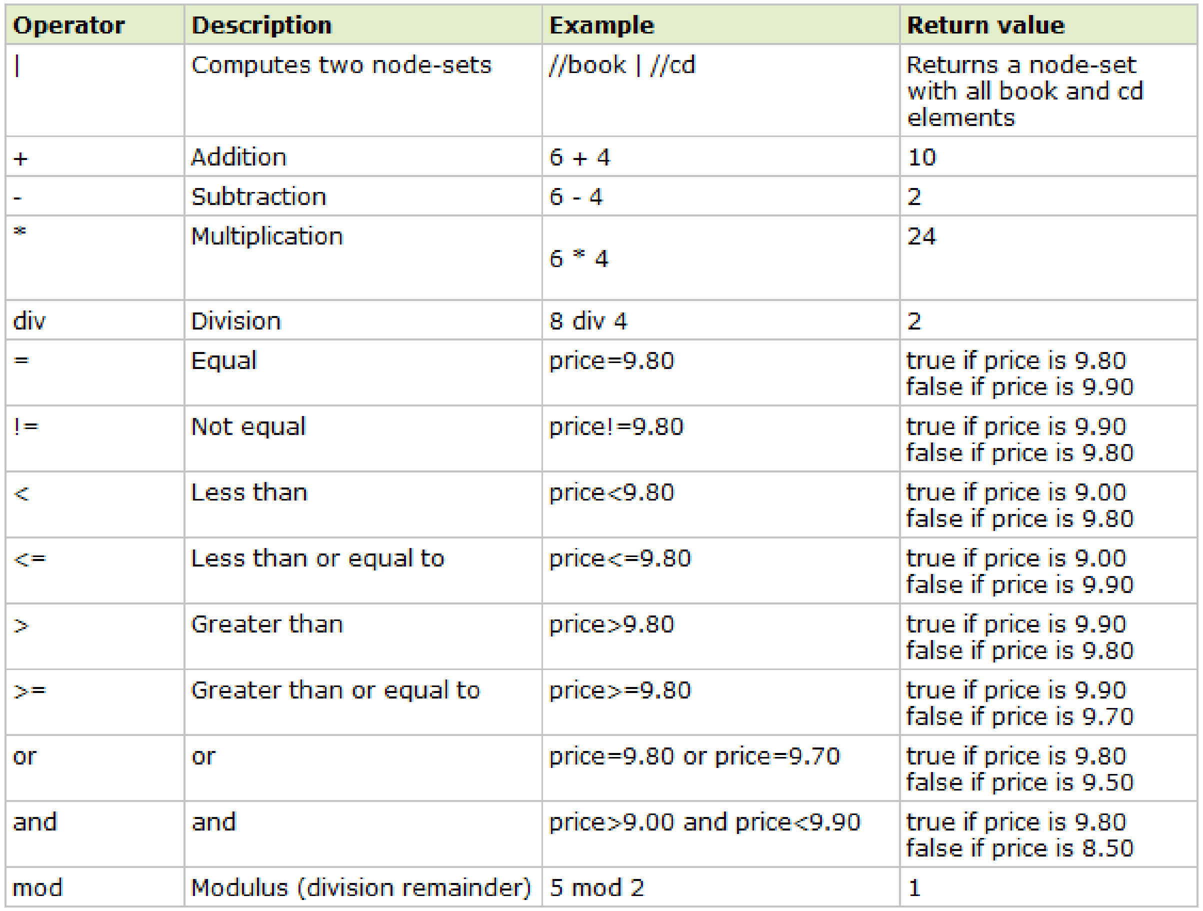


Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element. **Note:** IE5 and later has implemented that [0] should be the first node, but according to the W3C standard it should have been [1]!!
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

Source: http://www.w3schools.com/xpath/xpath_syntax.asp

• XPATH : syntaxe

❑ Condition de sélection

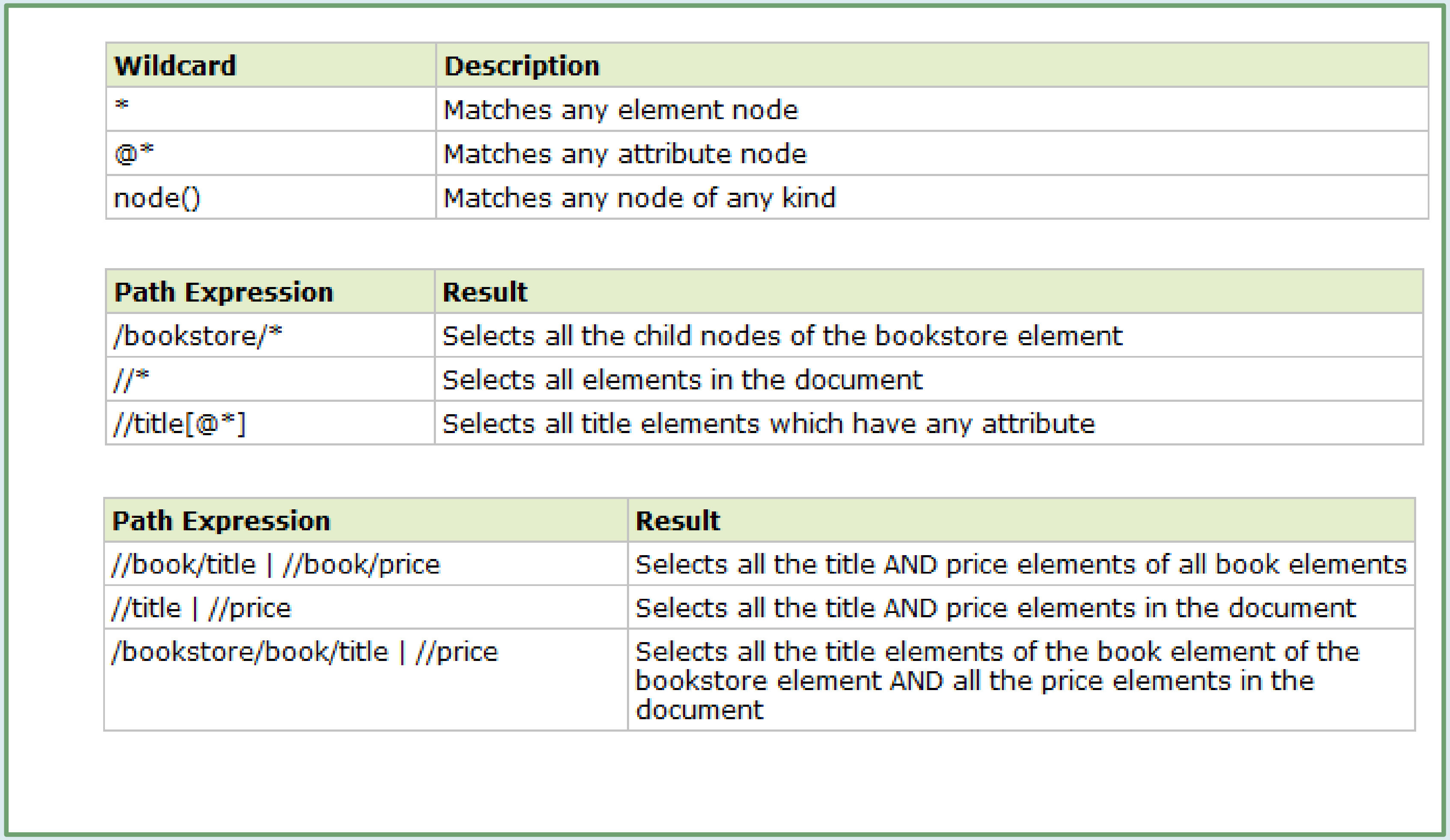


Operator	Description	Example	Return value
|	Computes two node-sets	//book | //cd	Returns a node-set with all book and cd elements
+	Addition	6 + 4	10
-	Subtraction	6 - 4	2
*	Multiplication	6 * 4	24
div	Division	8 div 4	2
=	Equal	price=9.80	true if price is 9.80 false if price is 9.90
!=	Not equal	price!=9.80	true if price is 9.90 false if price is 9.80
<	Less than	price<9.80	true if price is 9.00 false if price is 9.80
<=	Less than or equal to	price<=9.80	true if price is 9.00 false if price is 9.90
>	Greater than	price>9.80	true if price is 9.90 false if price is 9.80
>=	Greater than or equal to	price>=9.80	true if price is 9.90 false if price is 9.70
or	or	price=9.80 or price=9.70	true if price is 9.80 false if price is 9.50
and	and	price>9.00 and price<9.90	true if price is 9.80 false if price is 8.50
mod	Modulus (division remainder)	5 mod 2	1

Source: http://www.w3schools.com/xpath/xpath_syntax.asp

• XPATH : syntaxe

❑ Opérateurs de sélection



Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

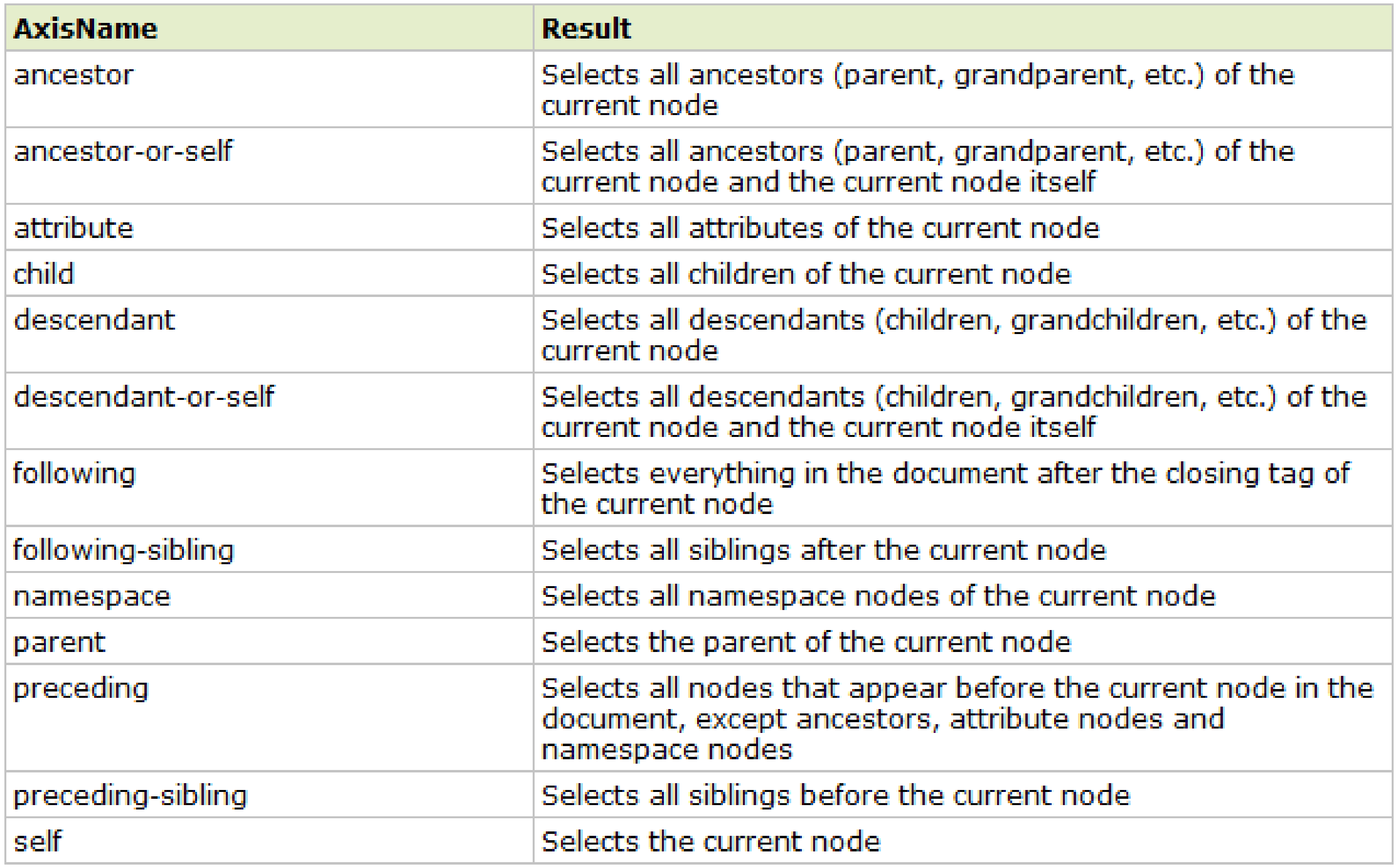
Path Expression	Result
/bookstore/*	Selects all the child nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have any attribute

Path Expression	Result
//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all the title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

Source: http://www.w3schools.com/xpath/xpath_syntax.asp

• XPATH : syntaxe

❑ Opérateurs de sélection par référence

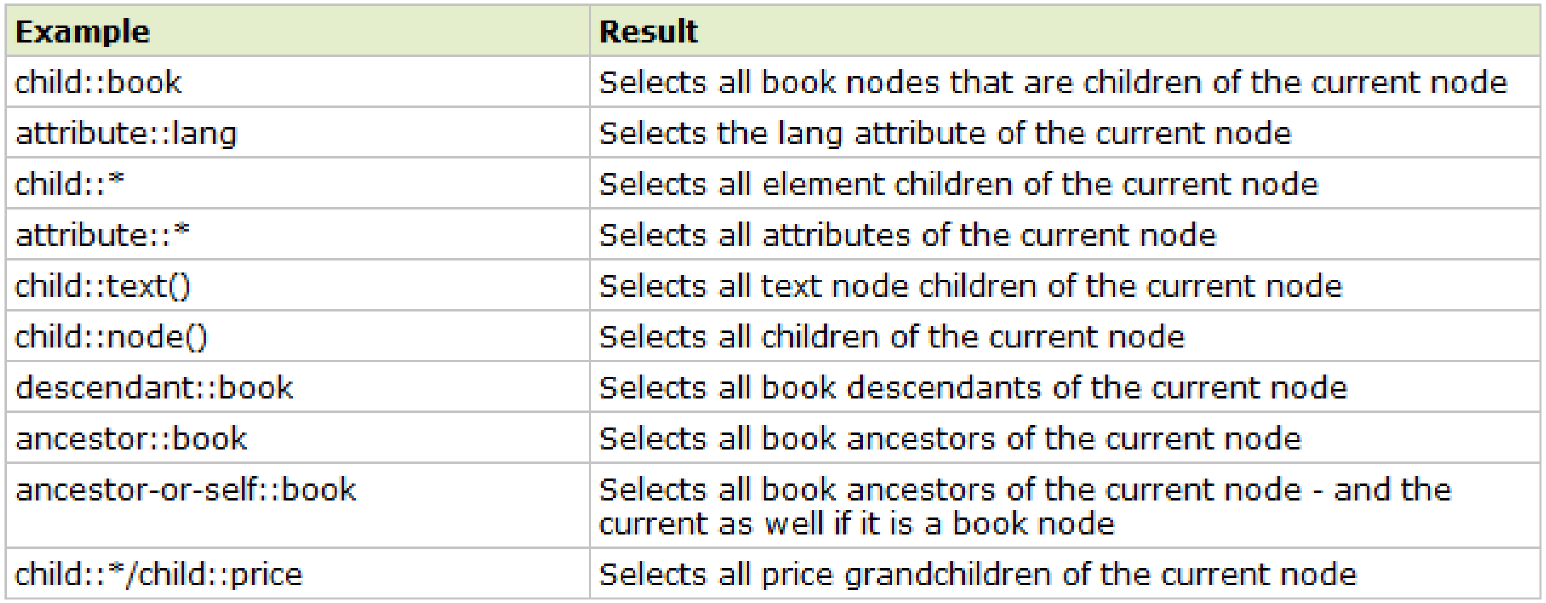


AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

Source: http://www.w3schools.com/xpath/xpath_syntax.asp

• XPATH : syntaxe

❑ Opérateurs de sélection par référence



Example	Result
child::book	Selects all book nodes that are children of the current node
attribute::lang	Selects the lang attribute of the current node
child::*	Selects all element children of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects all text node children of the current node
child::node()	Selects all children of the current node
descendant::book	Selects all book descendants of the current node
ancestor::book	Selects all book ancestors of the current node
ancestor-or-self::book	Selects all book ancestors of the current node - and the current as well if it is a book node
child::*/child::price	Selects all price grandchildren of the current node

Source: http://www.w3schools.com/xpath/xpath_syntax.asp

XPATH : A vous de jouer

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com book.xsd">
<!-- This is a Comment -->
  <category name="science-fiction">
    <article date="2002-09-24-06:00" id="#213354">
        <title>
            Platypus Ornithorhynchus anatinus)
        </title>
        <authors>
            <author>Augee, Michael</author>
            <author>Burrell, Harry</author>
        </authors>
        <document>
        The platypus (Ornithorhynchus anatinus) is a
        semi-aquatic mammal endemic to eastern Australia,
        including Tasmania. Together with the four
        species of echidna, it is one of the five
        extant species of monotremes, the only mammals
        that lay eggs instead of giving birth to live
        young. It is the sole living representative of
        its family (Ornithorhynchidae) and genus
        (Ornithorhynchus), though a number of related
        species have been found in the fossil record.
        </document>
    </article>
  </category>
  <category name="romance"/>
</book>
```

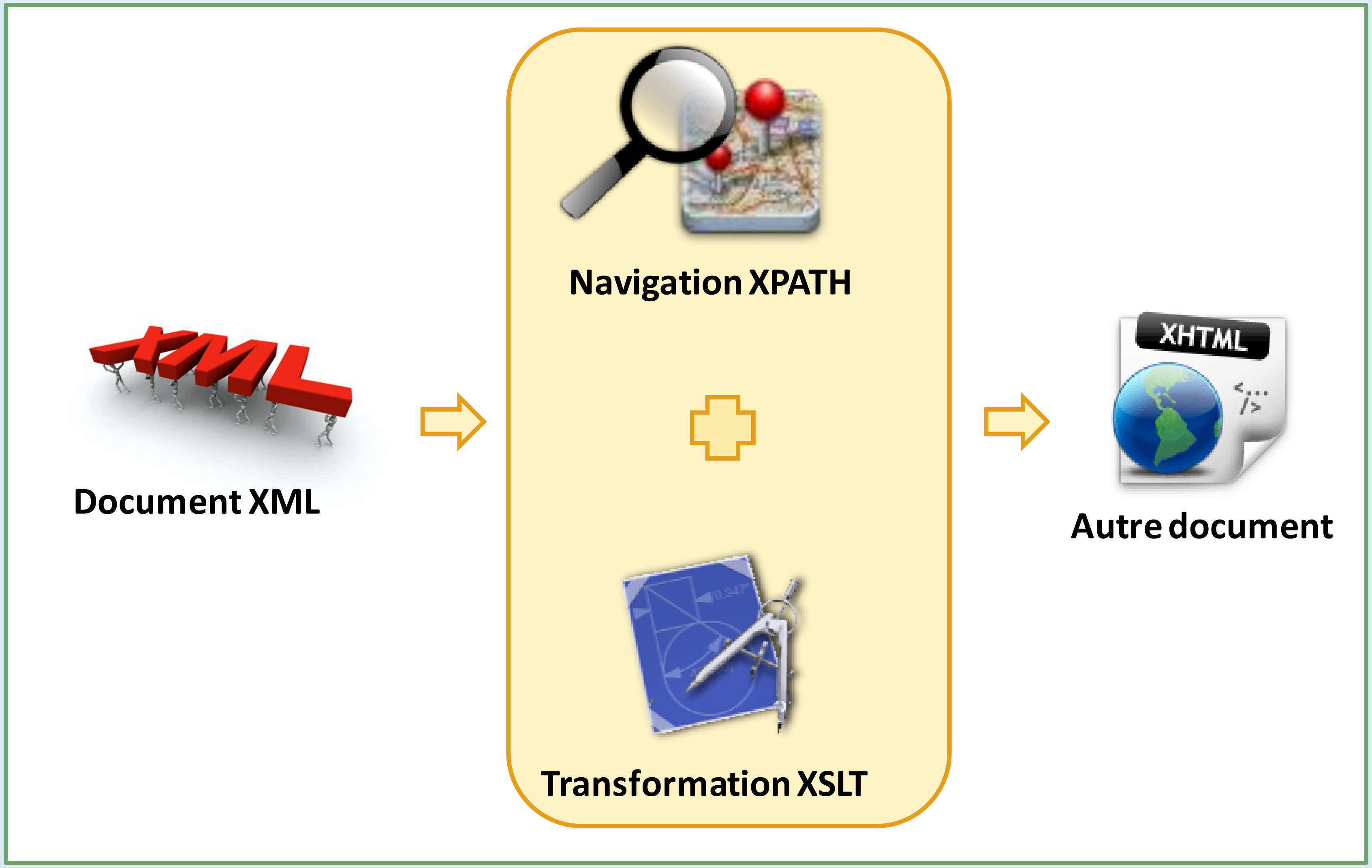
1. Tous les éléments “author”
2. L’élément “category” de nom “science-fiction”
3. Tous les ancetres de “author”
4. Tous les “document” des “article” **OU** les “title” des “article”
5. Listes des attributs des “category” **ET** de leurs descendants

Transformation

- XPath
- XSLT

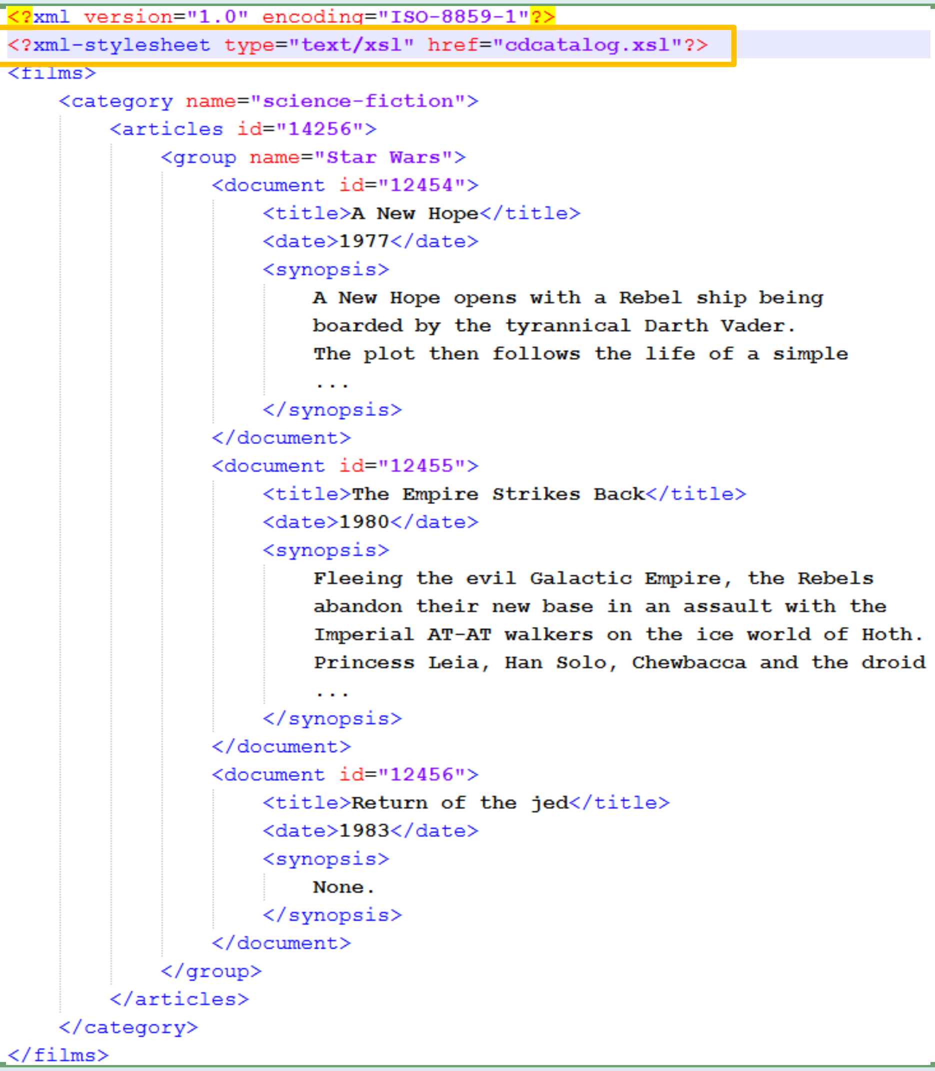
- **XSLT**
  - ❑ eXtensible Stylesheet Language Transformations
    → Transforme un document XML en un autre document XML (ou autres types)
  - ❑ La plus importante partie de XSL
  - ❑ Utilisation de XPATH pour naviguer dans le document XML
  - ❑ Possibilité d'ajouter, supprimer des éléments/attributs d'un document XML

XSLT : Comment ça marche ?



• XSLT : Les Bases

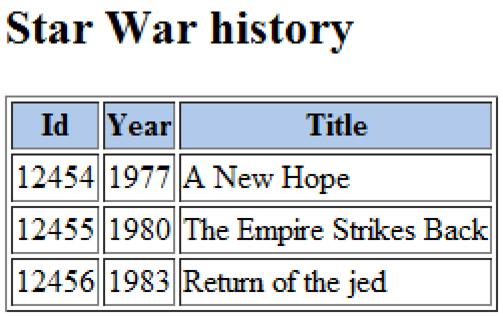
Référence au xsl utilisé



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<films>
    <category name="science-fiction">
        <articles id="14256">
            <group name="Star Wars">
                <document id="12454">
                    <title>A New Hope</title>
                    <date>1977</date>
                    <synopsis>
                        A New Hope opens with a Rebel ship being
                        boarded by the tyrannical Darth Vader.
                        The plot then follows the life of a simple
                        ...
                    </synopsis>
                </document>
                <document id="12455">
                    <title>The Empire Strikes Back</title>
                    <date>1980</date>
                    <synopsis>
                        Fleeing the evil Galactic Empire, the Rebels
                        abandon their new base in an assault with the
                        Imperial AT-AT walkers on the ice world of Hoth.
                        Princess Leia, Han Solo, Chewbacca and the droid
                        ...
                    </synopsis>
                </document>
                <document id="12456">
                    <title>Return of the jed</title>
                    <date>1983</date>
                    <synopsis>
                        None.
                    </synopsis>
                </document>
            </group>
        </articles>
    </category>
</films>
```

• XSLT : Les Bases

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
      <body>
      <h2>Star War history</h2>
      <table border="1">
        <tr bgcolor="#B1CAEB">
          <th>Id</th>
          <th>Year</th>
          <th>Title</th>
        </tr>
        <xsl:for-each
            select='//group[@name="Star Wars"]/document'>
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
        </xsl:for-each>
      </table>
      </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```



Star War history

Id	Year	Title
12454	1977	A New Hope
12455	1980	The Empire Strikes Back
12456	1983	Return of the jed

XSLT : Syntaxe

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
      <body>
      <h2>Star War history</h2>
      <table border="1">
        <tr bgcolor="#B1CAEB">
          <th>Id</th>
          <th>Year</th>
          <th>Title</th>
        </tr>
        <xsl:for-each
            select='//group[@name="Star Wars"]/document'>
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
        </xsl:for-each>
      </table>
      </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

1 Espace de nommage utilisé pour la transformation XSLT

2 Définition d'une règle

3 Construction du contenu

4 Opérateurs de construction

XSLT : Syntaxe

1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <body>
    <h2>Star War history</h2>
    <table border="1">
      <tr bgcolor="#B1CAEB">
        <th>Id</th>
        <th>Year</th>
        <th>Title</th>
      </tr>
      <xsl:for-each
          select='//group[@name="Star Wars"]/document'>
      <tr>
        <td><xsl:value-of select="attribute::id"/></td>
        <td><xsl:value-of select="date"/></td>
        <td><xsl:value-of select="title"/></td>
      </tr>
      </xsl:for-each>
    </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

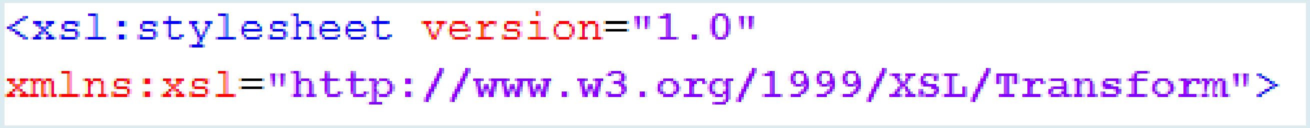
1

Espace de nommage utilisé pour la transformation XSLT

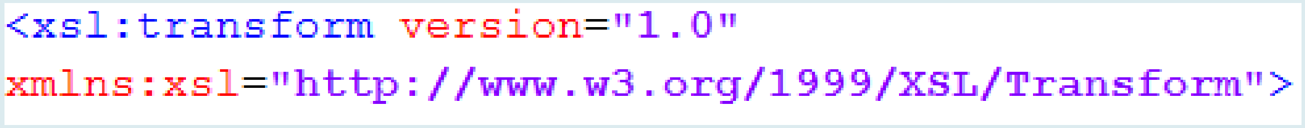
- **XSLT : Syntaxe**
  - ❑ Espace de nommage

    → import les éléments de traitement

  - ❑ Entête possible



```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```



```
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

XSLT : Syntaxe

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
      <body>
      <h2>Star War history</h2>
      <table border="1">
        <tr bgcolor="#B1CAEB">
          <th>Id</th>
          <th>Year</th>
          <th>Title</th>
        </tr>
        <xsl:for-each
            select='//group[@name="Star Wars"]/document'>
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
        </xsl:for-each>
      </table>
      </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

2

1 Espace de nommage utilisé pour la transformation XSLT

2 **Définition d'une règle**

XSLT : Syntaxe

- ❑ Règle de transformation

  → Ensemble définissant le comportant/contenu du document de sortie

  - ❑ 1 règle porte sur une partie (ou totalité) du document XML source
  - ❑ 1 règle peut être appelé au sein d'une autre règle

• XSLT : Syntaxe

❑ Règle de transformation

```
<xsl:template
    match="documentXpath">
```

Définition Simple

```
<xsl:template
    match="documentXpath"
    name="ruleName"
    priority="priorityNumber"
    mode="templateTag">
```

Définition Complete

Exemple simple

```
<xsl:template
    match="//document">
    ...
</xsl:template>
```

Exemple complet

```
<xsl:template match="//document"
    name="docRule" priority="5"
    mode="docFormat">
    ...
</xsl:template>
```

• XSLT : Syntaxe

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
      <body>
      <h2>Star War history</h2>
      <table border="1">
        <tr bgcolor="#B1CAEB">
          <th>Id</th>
          <th>Year</th>
          <th>Title</th>
        </tr>
        <xsl:for-each
            select='//group[@name="Star Wars"]/document'>
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
        </xsl:for-each>
      </table>
      </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

3

3

1 Espace de nommage utilisé pour la transformation XSLT

2 Définition d'une règle

3 **Construction du contenu**

XSLT : Syntaxe

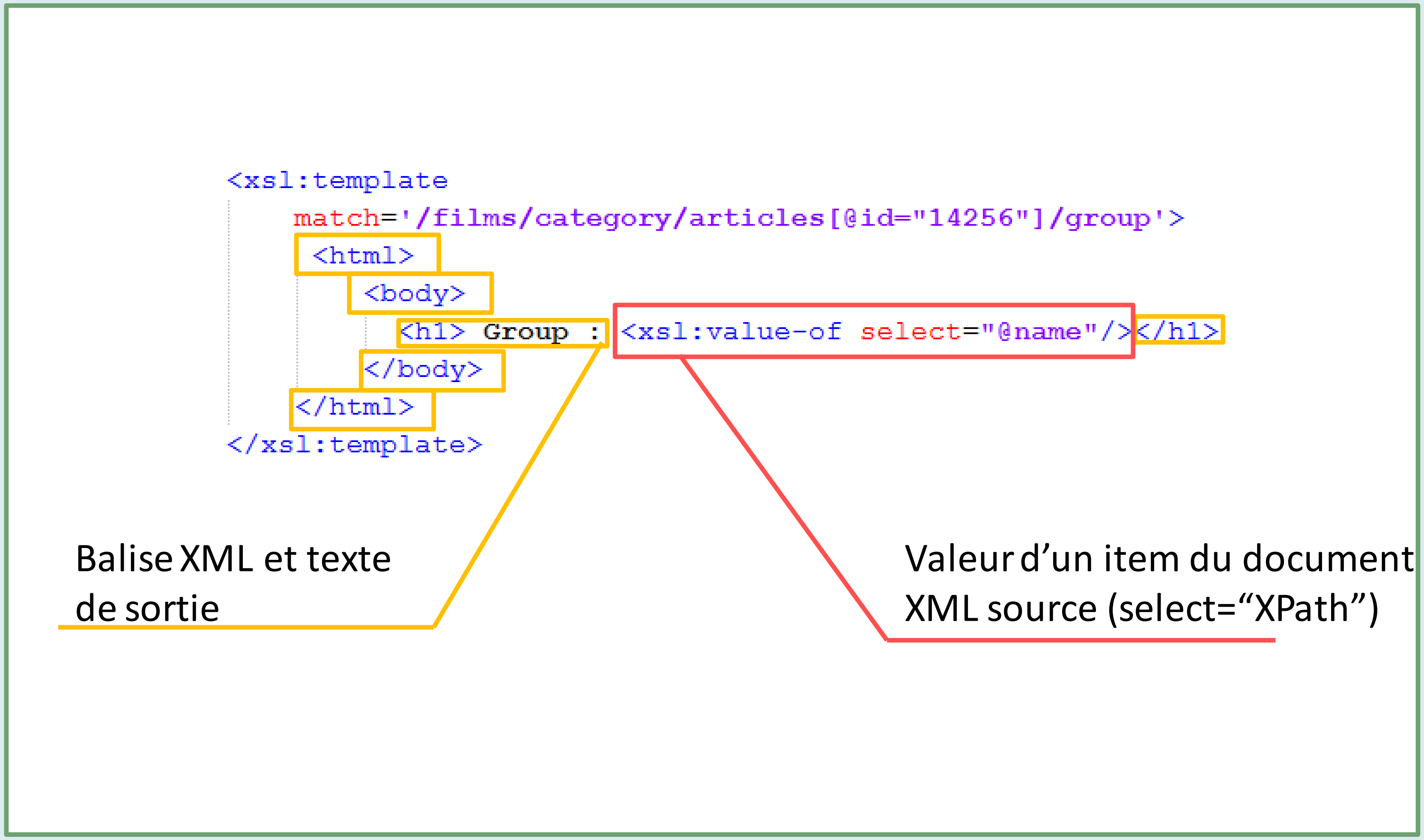
- Construction du contenu de sortie

  → Ensemble d'éléments XML modifiant/présentant les informations du document source

  - Ajout du contenu de sortie souhaité (format XML)
  - Récupération des valeurs du document source par parcours de son arborescence

• XSLT : Syntaxe

- ❑ Construction du contenu de sortie

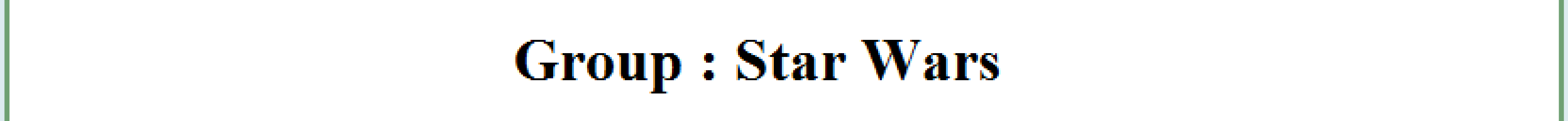

Balise XML et texte de sortie
Valeur d'un item du document XML source (select="XPath")

```
<xsl:template
    match='/films/category/articles[@id="14256"]/group'>
  <html>
    <body>
      <h1> Group : <xsl:value-of select="@name"/></h1>
    </body>
  </html>
</xsl:template>
```

• XSLT : Syntaxe

❑ Construction du contenu de sortie

```
<xsl:template
    match='/films/category/articles[@id="14256"]/group'>
     <html>
        <body>
           <h1> Group : <xsl:value-of select="@name"/></h1>
        </body>
    </html>
</xsl:template>
```



Group : Star Wars

• XSLT : Syntaxe

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
      <body>
      <h2>Star War history</h2>
      <table border="1">
        <tr bgcolor="#B1CAEB">
          <th>Id</th>
          <th>Year</th>
          <th>Title</th>
        </tr>
        <xsl:for-each
            select='//group[@name="Star Wars"]/document'>
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
        </xsl:for-each>
      </table>
      </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

4

1 Espace de nommage utilisé pour la transformation XSLT

2 Définition d'une règle

3 Construction du contenu

4 **Opérateurs de construction**

XSLT : Syntaxe

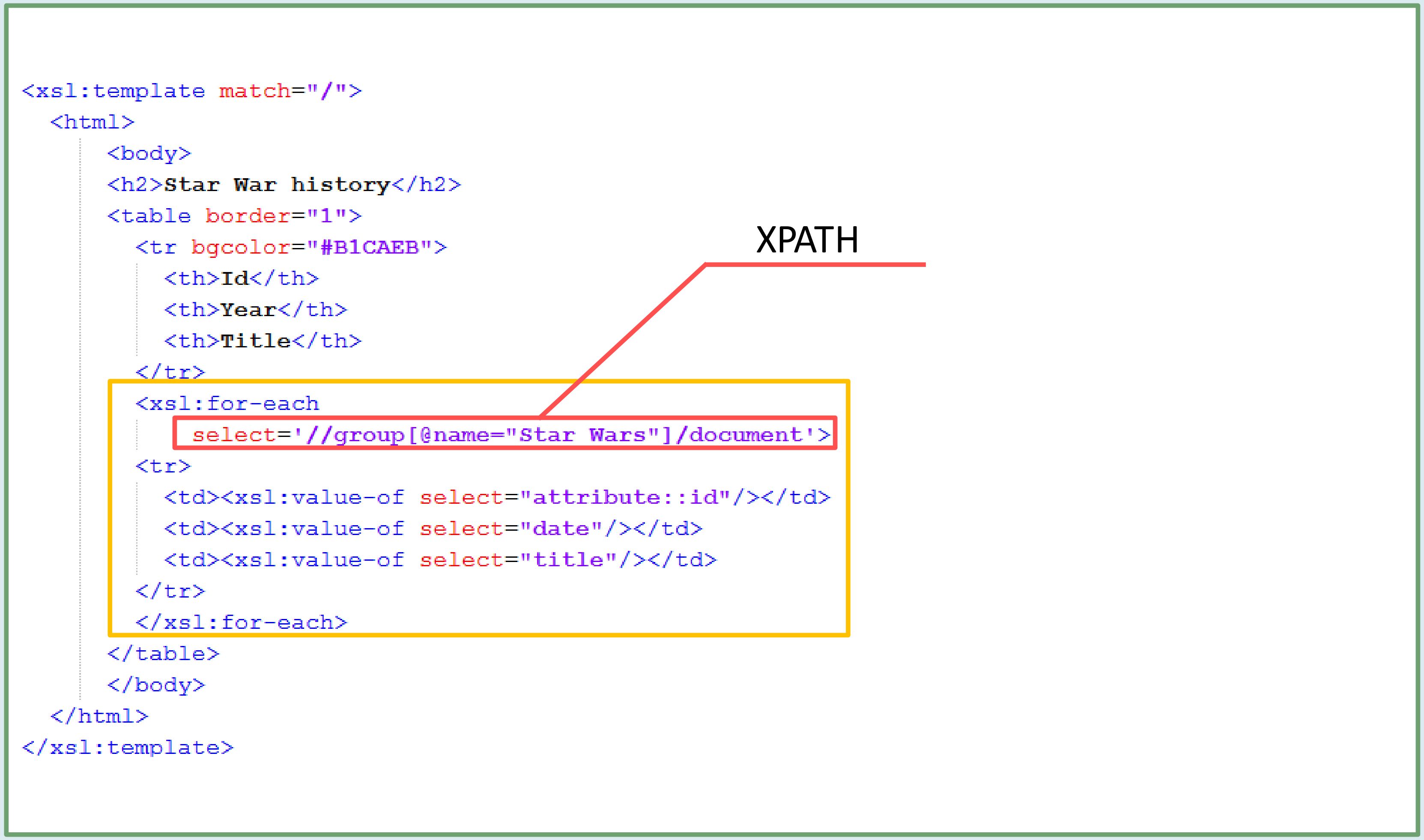
- Opérateur de construction

  → Opérateurs permettant de manipuler des objets XML du document source

  - for each
  - sort
  - if
  - choose

• XSLT : Syntaxe

❑ Opérateur de construction : for each



```
<xsl:template match="/">
  <html>
      <body>
      <h2>Star War history</h2>
      <table border="1">
        <tr bgcolor="#B1CAEB">
          <th>Id</th>
          <th>Year</th>
          <th>Title</th>
        </tr>
        <xsl:for-each
          select='//group[@name="Star Wars"]/document'>
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
        </xsl:for-each>
      </table>
      </body>
  </html>
</xsl:template>
```

- **XSLT : Syntaxe**
  - ❑ Opérateur de construction : for each

XSLT

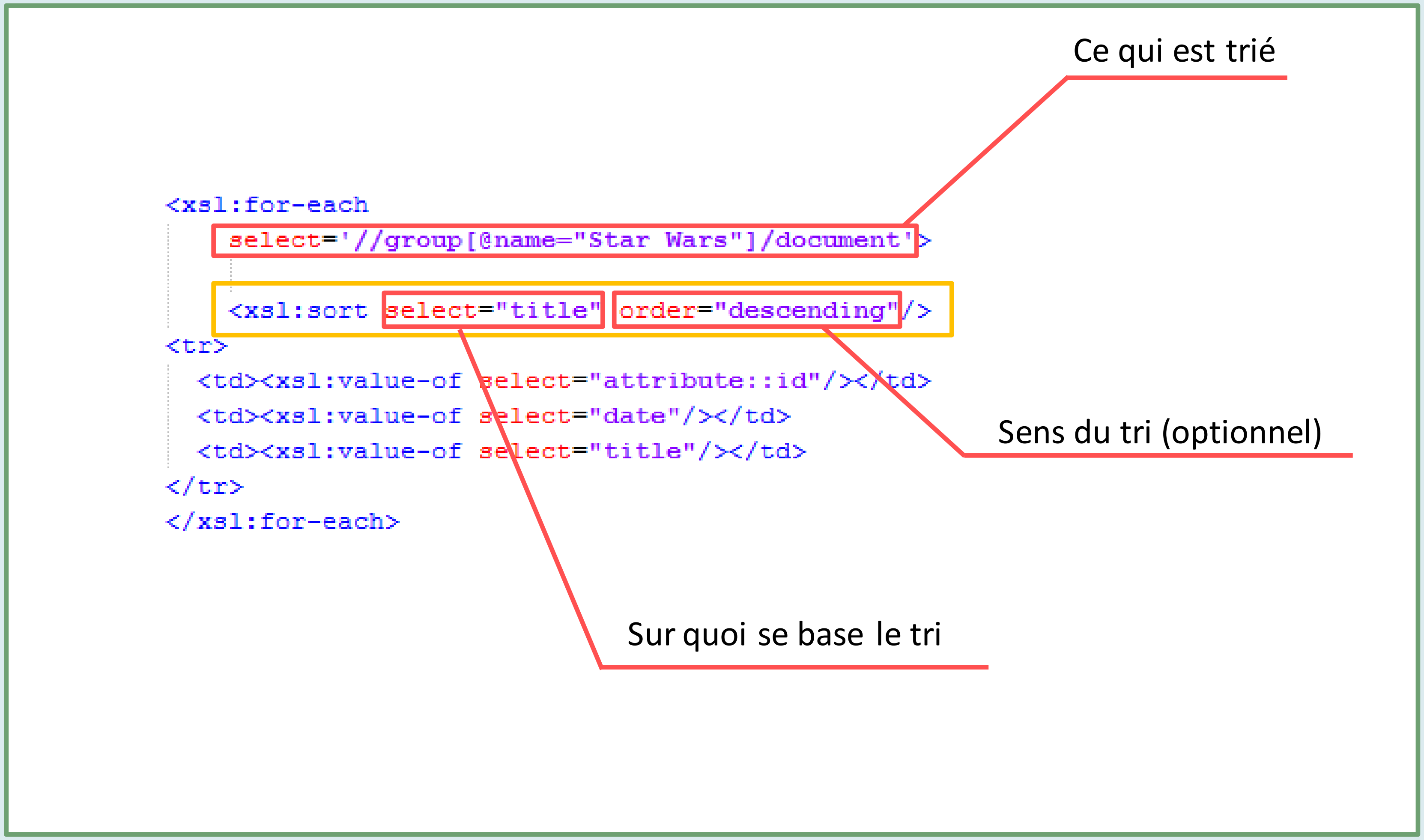
```
<xsl:template match="/">
  <html>
      <body>
      <h2>Star War history</h2>
      <table border="1">
        <tr bgcolor="#B1CAEB">
          <th>Id</th>
          <th>Year</th>
          <th>Title</th>
        </tr>
        <xsl:for-each
            select='//group[@name="Star Wars"]/document'>
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
        </xsl:for-each>
      </table>
      </body>
  </html>
</xsl:template>
```

XML

```
<group name="Star Wars">
<document id="12454">
    <title>A New Hope</title>
    <date>1977</date>
    <synopsis>
        A New Hope opens with a Rebel ship being
        boarded by the tyrannical Darth Vader.
        The plot then follows the life of a simple
        ...
    </synopsis>
</document>
<document id="12455">
    <title>The Empire Strikes Back</title>
    <date>1980</date>
    <synopsis>
        Fleeing the evil Galactic Empire, the Rebels
        abandon their new base in an assault with the
        Imperial AT-AT walkers on the ice world of Hoth.
        Princess Leia, Han Solo, Chewbacca and the droid
        ...
    </synopsis>
</document>
<document id="12456">
    <title>Return of the jed</title>
    <date>1983</date>
    <synopsis>
        None.
    </synopsis>
</document>
```

XSLT : Syntaxe

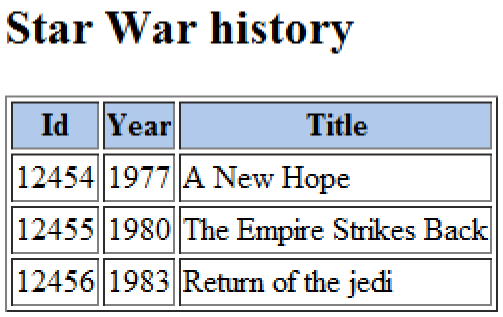
- Opérateur de construction : sort



```
<xsl:for-each
      select='//group[@name="Star Wars"]/document'>

      <xsl:sort select="title" order="descending"/>
<tr>
  <td><xsl:value-of select="attribute::id"/></td>
  <td><xsl:value-of select="date"/></td>
  <td><xsl:value-of select="title"/></td>
</tr>
</xsl:for-each>
```

- **XSLT : Syntaxe**
  - ❑ Opérateur de construction : sort

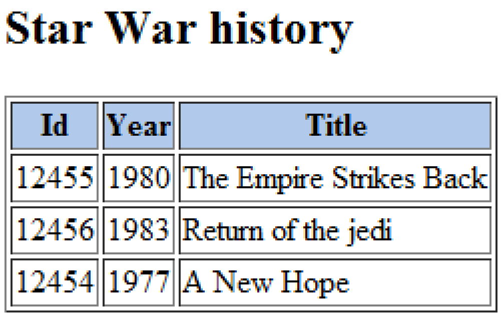
```
<xsl:for-each
    select='//group[@name="Star Wars"]/document'>

    <xsl:sort select="title" order="descending"/>
<tr>
  <td><xsl:value-of select="attribute::id"/></td>
  <td><xsl:value-of select="date"/></td>
  <td><xsl:value-of select="title"/></td>
</tr>
</xsl:for-each>
```



Star War history

Id	Year	Title
12454	1977	A New Hope
12455	1980	The Empire Strikes Back
12456	1983	Return of the jedi

Sans le tri

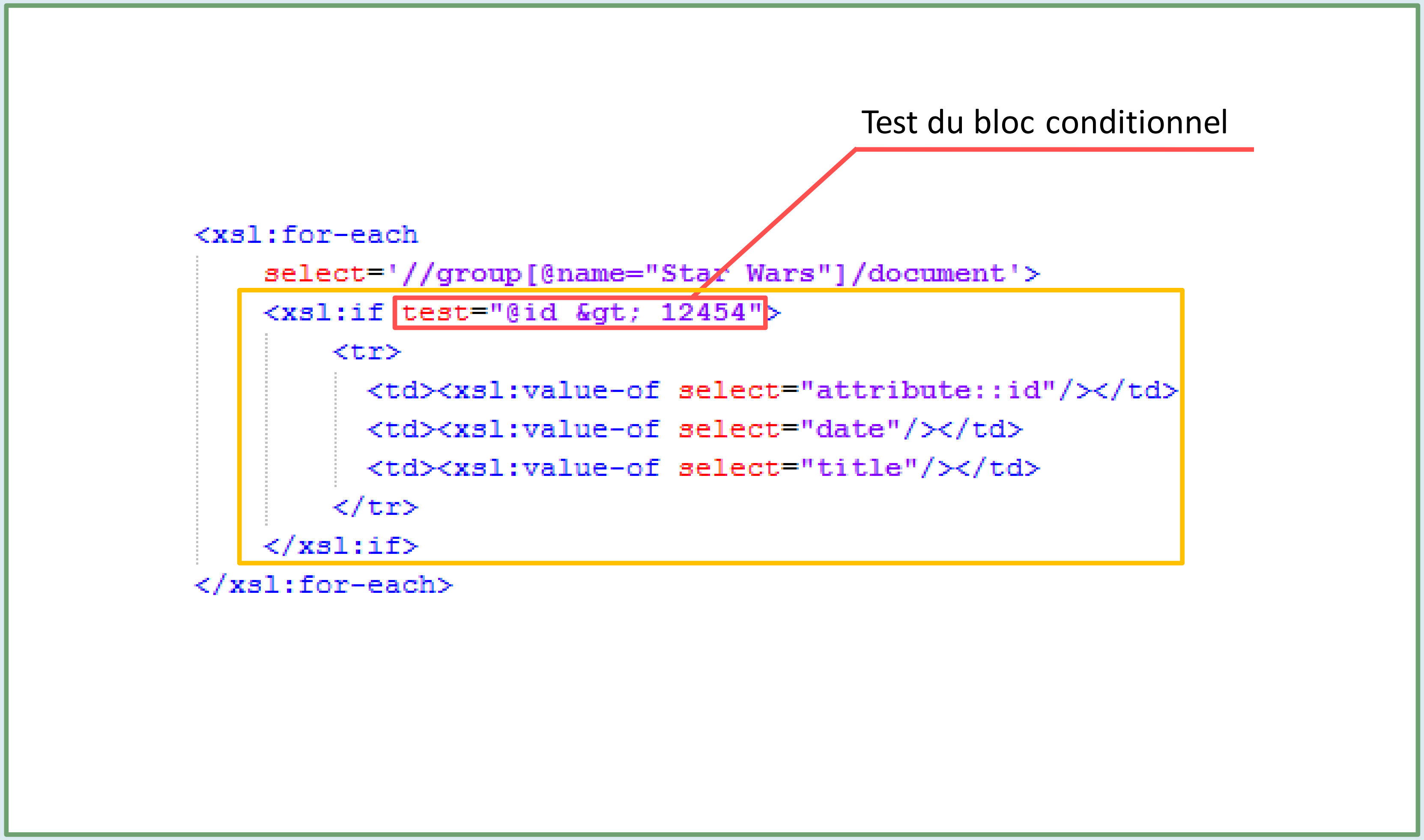


Star War history

Id	Year	Title
12455	1980	The Empire Strikes Back
12456	1983	Return of the jedi
12454	1977	A New Hope

Avec le tri

- **XSLT : Syntaxe**
  - Opérateur de construction : if



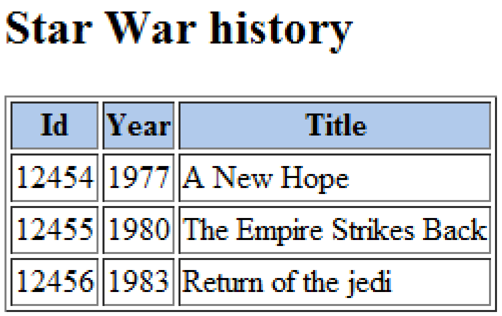
Test du bloc conditionnel

```
<xsl:for-each
    select='//group[@name="Star Wars"]/document'>
    <xsl:if test="@id &gt; 12454">
        <tr>
            <td><xsl:value-of select="attribute::id"/></td>
            <td><xsl:value-of select="date"/></td>
            <td><xsl:value-of select="title"/></td>
        </tr>
    </xsl:if>
</xsl:for-each>
```

XSLT : Syntaxe

- Opérateur de construction : if

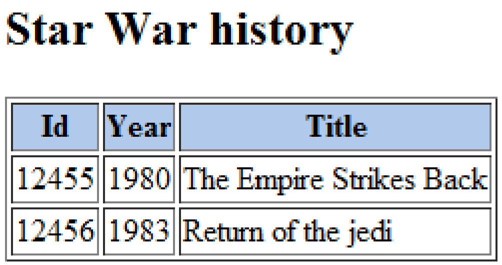
```
<xsl:for-each
    select='//group[@name="Star Wars"]/document'>
    <xsl:if test="@id &gt; 12454">
        <tr>
          <td><xsl:value-of select="attribute::id"/></td>
          <td><xsl:value-of select="date"/></td>
          <td><xsl:value-of select="title"/></td>
        </tr>
    </xsl:if>
</xsl:for-each>
```



Star War history

Id	Year	Title
12454	1977	A New Hope
12455	1980	The Empire Strikes Back
12456	1983	Return of the jedi

Sans condition



Star War history

Id	Year	Title
12455	1980	The Empire Strikes Back
12456	1983	Return of the jedi

Avec condition

• XSLT : Syntaxe

❑ Opérateur de construction : choose

```
<xsl:choose>
  <xsl:when test="expression">
    ... content if condition rises ...
  </xsl:when>
  <xsl:when test="expression">
    ... content if condition rises ...
  </xsl:when>
  <xsl:when test="expression">
    ... content if condition rises ...
  </xsl:when>
  ...
  <xsl:otherwise>
    ... content any condition rose ....
  </xsl:otherwise>
</xsl:choose>
```

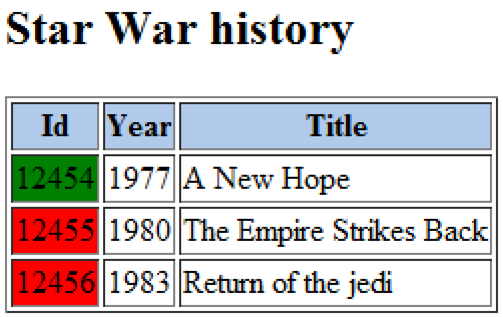
N blocs conditionnels

Contenu si aucune condition est validée

• XSLT : Syntaxe

❑ Opérateur de construction : choose

```
<xsl:for-each
    select='//group[@name="Star Wars"]/document'>
    <tr>
        <xsl:choose>
            <xsl:when test="@id &gt;= 12455">
                <td bgcolor="red"><xsl:value-of select="attribute::id"/></td>
            </xsl:when>
            <xsl:when test="@id = 12454">
                <td bgcolor="green"><xsl:value-of select="attribute::id"/></td>
            </xsl:when>
            <xsl:otherwise>
                <td><xsl:value-of select="attribute::id"/></td>
            </xsl:otherwise>
        </xsl:choose>
      <td><xsl:value-of select="date"/></td>
      <td><xsl:value-of select="title"/></td>
    </tr>
</xsl:for-each>
```



Star War history

Id	Year	Title
🟩 12454	1977	A New Hope
🟥 12455	1980	The Empire Strikes Back
🟥 12456	1983	Return of the jedi

XSLT : Syntaxe Avancée

- Appel de règle
  - Apply
  - Call

XSLT : Syntaxe avancée

❑ Appel d'une règle de transformation

```
<xsl:call-template name = "targetedRuleName">
```

Call

- Appel explicitement une règle définie
- Ne change pas le noeud courant

```
<xsl:apply-templates select="XpathTarget"/>
```

Apply

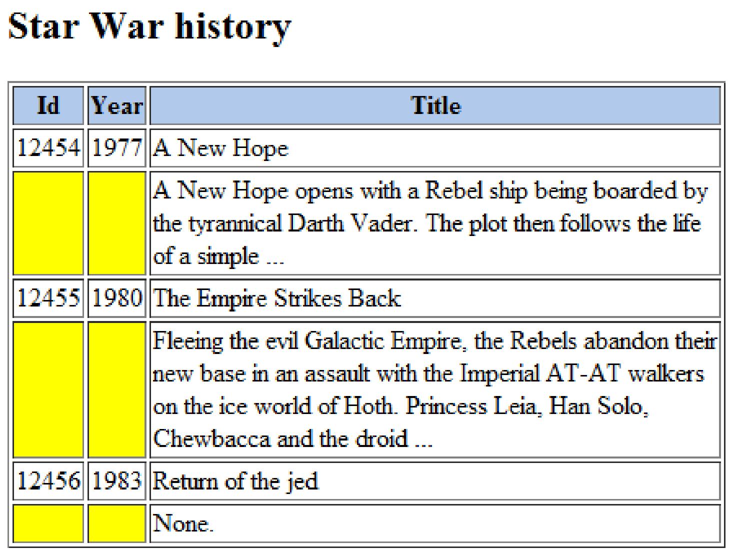
- Traite tous les enfants du Xpath donné
- **Change** le noeud courant
- si select non spécifié, traite tous les fils du noeud courant

```
<xsl:apply-templates
     select="XpathTarget"
     mode="templateTag"/>
```

- Traite tous les enfants du Xpath donné declachant **uniquement** les règles possédants le tag donné

Call

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
      <html>
          <body>
          <h2>Star War history</h2>
          <table border="1">
            <tr bgcolor="#B1CAEB">
              <th>Id</th>
              <th>Year</th>
              <th>Title</th>
            </tr>
            <xsl:for-each
                select='//group[@name="Star Wars"]/document'>
            <tr>
              <td><xsl:value-of select="attribute::id"/></td>
              <td><xsl:value-of select="date"/></td>
              <td><xsl:value-of select="title"/></td>
            </tr>
            <xsl:call-template name="yellowBgLine"/>
            </xsl:for-each>
          </table>
          </body>
      </html>
    </xsl:template>

    <xsl:template name="yellowBgLine">
        <tr>
            <td bgcolor="yellow"/>
            <td bgcolor="yellow"/>
            <td><xsl:value-of select="synopsis"/></td>
        </tr>
    </xsl:template>
</xsl:stylesheet>
```

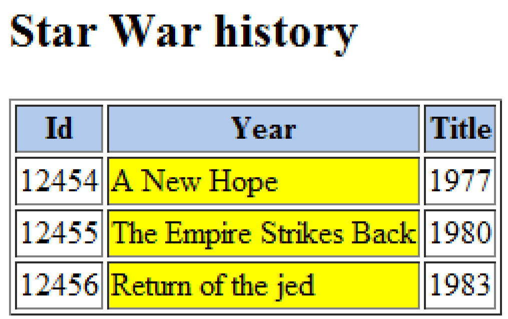


Star War history

Id	Year	Title
12454	1977	A New Hope
🟨	🟨	A New Hope opens with a Rebel ship being boarded by the tyrannical Darth Vader. The plot then follows the life of a simple ...
12455	1980	The Empire Strikes Back
🟨	🟨	Fleeing the evil Galactic Empire, the Rebels abandon their new base in an assault with the Imperial AT-AT walkers on the ice world of Hoth. Princess Leia, Han Solo, Chewbacca and the droid ...
12456	1983	Return of the jed
🟨	🟨	None.

Apply

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
      <html>
          <body>
          <h2>Star War history</h2>
          <table border="1">
            <tr bgcolor="#B1CAEB">
              <th>Id</th>
              <th>Year</th>
              <th>Title</th>
            </tr>
            <xsl:for-each
                select='//group[@name="Star Wars"]/document'>
            <tr>
                <td><xsl:value-of select="@id"/></td>
                <xsl:apply-templates select="."/>
            </tr>
            </xsl:for-each>
          </table>
          </body>
      </html>
    </xsl:template>

    <xsl:template match="date">
              <td><xsl:value-of select="."/></td>
    </xsl:template>

    <xsl:template match="title">
        <td bgcolor="yellow"><xsl:value-of select="."/></td>
    </xsl:template>

    <xsl:template match="synopsis">
    </xsl:template>

</xsl:stylesheet>
```



Star War history

Id	Year	Title
12454	A New Hope	1977
12455	The Empire Strikes Back	1980
12456	Return of the jed	1983

Utilisation XML

- Programmer avec XML
- DOM
- SAX
- JAXP
- JAXB

XML

- **Programmer avec XML**
  - ❑ Utiliser les documents XML
    - → Lecture du contenu en continu (SAX) en intégralité (DOM)
    - → Appliquer des transformations (JAXP)
    - → Convertir en class objets (JAXB)

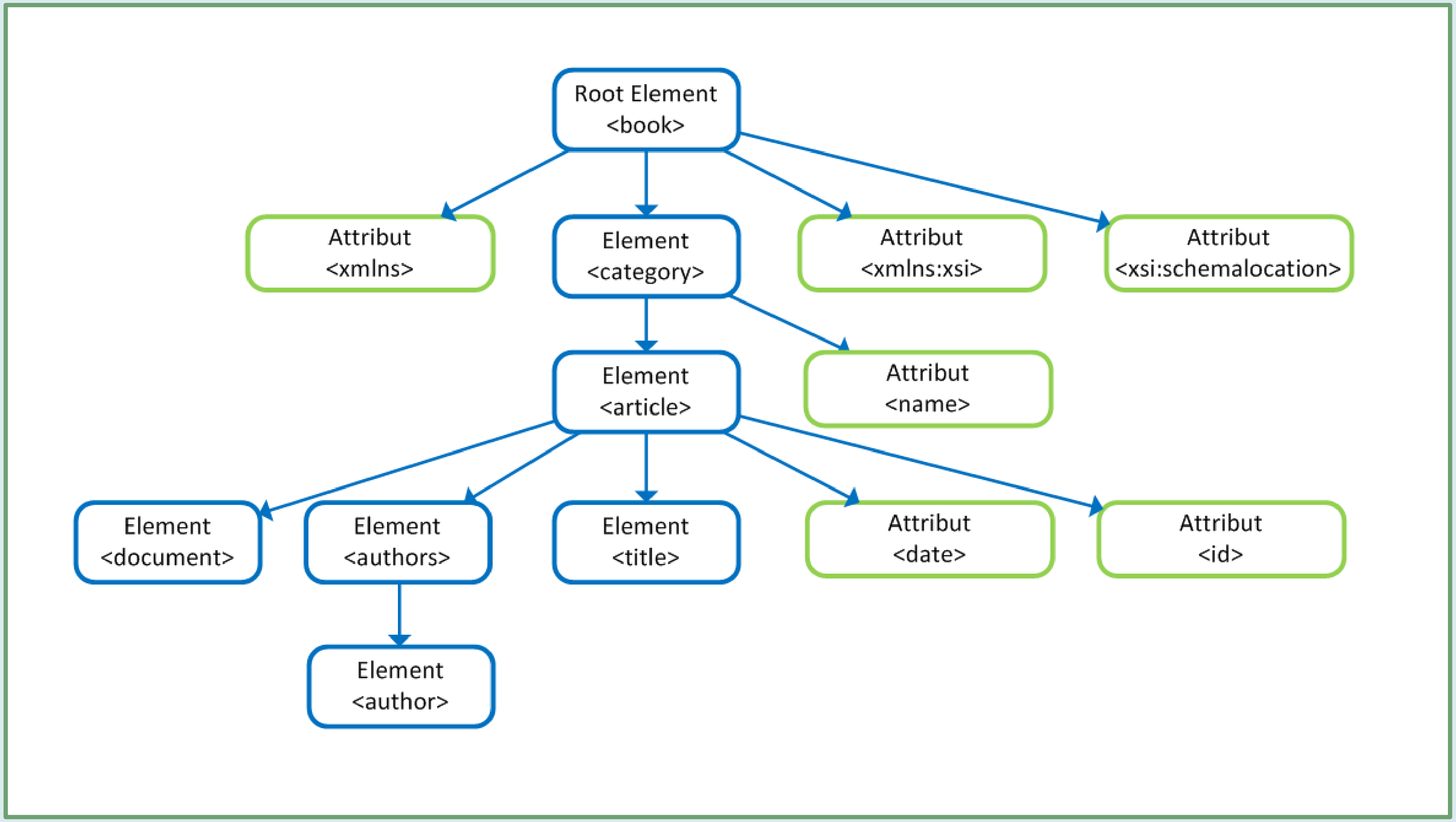
Utilisation XML

- Programmer avec XML
- DOM
- SAX
- JAXP
- JAXB

XML

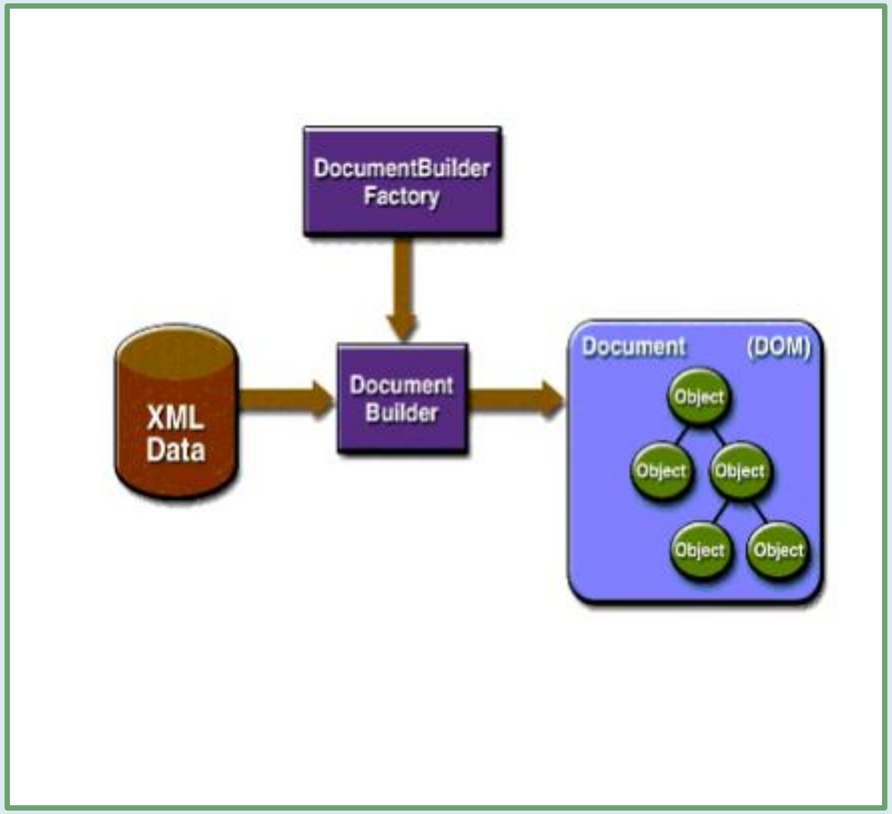
- **Programmer avec XML : DOM**
  - ❑ Document Object Model
    - → Accéder et manipuler les documents XML
  - ❑ Standard W3C
  - ❑ Interprétation des documents XML en vue d'arbre
  - ❑ Charge le document XML en entier pour le traiter

- **Programmer avec XML : DOM**



XML

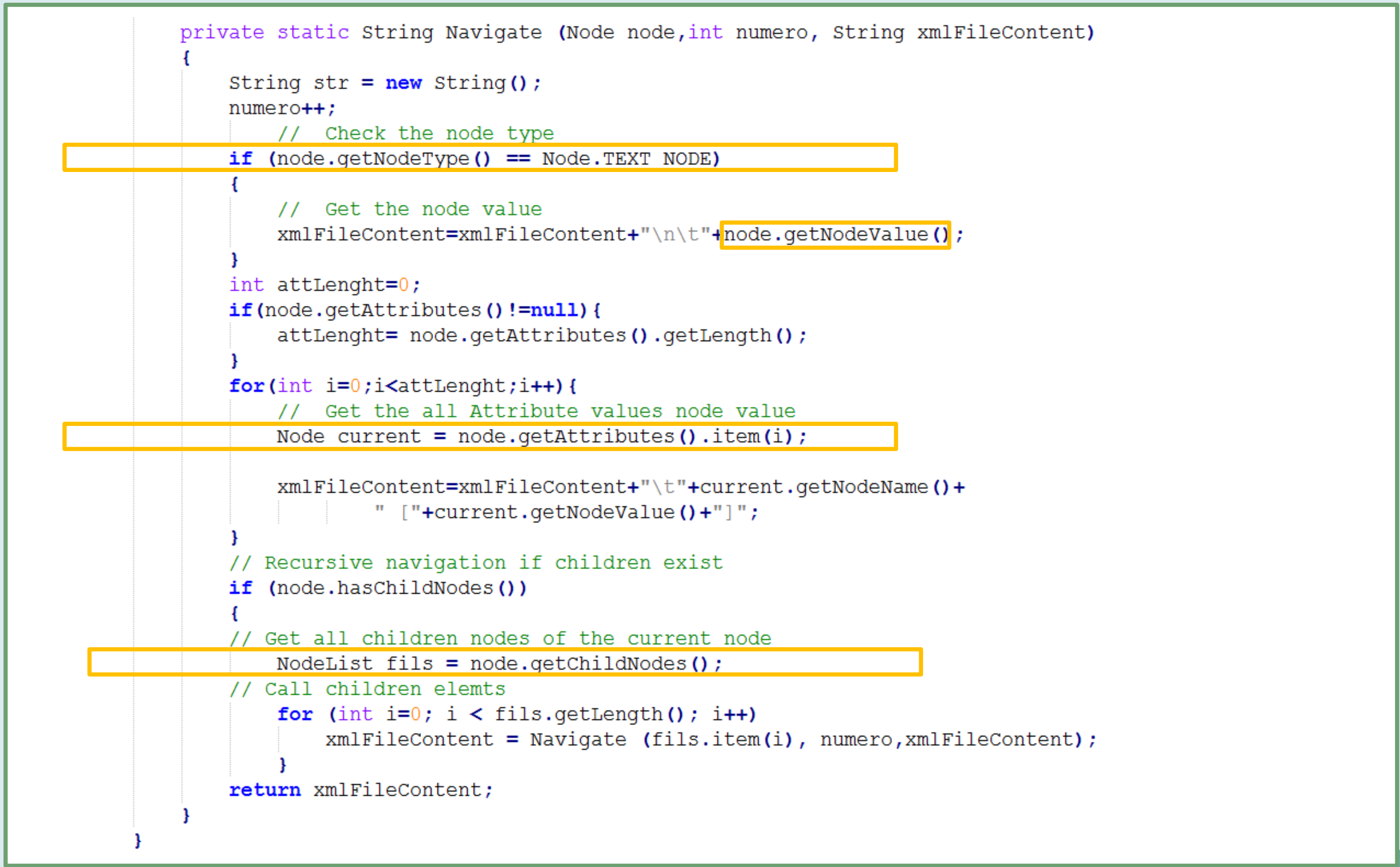
- **Programmer avec XML : DOM**
  - Document Object Model
  - Parser java
  - Parser javascript



• Programmer avec XML : DOM Parser Java

```
// Needed classes
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
class Preordre
{
    public static void main (String args [])
    {
        try {
        // load the XML document
        File fdom = new File ("XSLTFirstSample.xml");
        // Instanciate a factor of parser
        DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();
        DocumentBuilder builder;
        //Creat a new document builder
            builder = factory.newDocumentBuilder();
        // Parse the document
            Document dom = builder.parse(fdom);
        // Start to parse the document with the first elt
        Node elementRacine = dom.getDocumentElement();
        String result="";
        result=Navigate (elementRacine, 1,result);
        System.out.println(result);

        } catch (ParserConfigurationException | SAXException | IOException e) {
            e.printStackTrace();
        }
    }
```

Programmer avec XML : DOM Parser Java



```java
    private static String Navigate (Node node,int numero, String xmlFileContent)
    {
        String str = new String();
        numero++;
            //  Check the node type
        if (node.getNodeType() == Node.TEXT_NODE)
        {
            //  Get the node value
            xmlFileContent=xmlFileContent+"\n\t"+node.getNodeValue();
        }
        int attLenght=0;
        if(node.getAttributes()!=null){
            attLenght= node.getAttributes().getLength();
        }
        for(int i=0;i<attLenght;i++){
            //  Get the all Attribute values node value
            Node current = node.getAttributes().item(i);

            xmlFileContent=xmlFileContent+"\t"+current.getNodeName()+
                    "  ["+current.getNodeValue()+"]";
        }
        // Recursive navigation if children exist
        if (node.hasChildNodes())
        {
        // Get all children nodes of the current node
            NodeList fils = node.getChildNodes();
        // Call children elemts
            for (int i=0; i < fils.getLength(); i++)
                xmlFileContent = Navigate (fils.item(i), numero,xmlFileContent);
            }
        return xmlFileContent;
    }
}
```

• Programmer avec XML : DOM Parser Java script

```
if (window.XMLHttpRequest)
 {
      xhttp=new XMLHttpRequest();
 }
      else // IE 5/6
 {
      xhttp=new ActiveXObject("Microsoft.XMLHTTP");
 }
xhttp.open("GET","books.xml",false);
xhttp.send();
xmlDoc=xhttp.responseXML;
```

Chargement d'un document
(Fichier distance)

```
xmlDoc=loadXMLDoc("books.xml");
```

Chargement d'un document
(Fichier local)

- **Programmer avec XML : DOM Parser Java script**

```
if (window.DOMParser)
 {
     parser=new DOMParser();
     xmlDoc=parser.parseFromString(text,"text/xml");
 }
else // Internet Explorer
 {
    xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
    xmlDoc.async=false;
    xmlDoc.loadXML(text);
 }
```

Chargement d'un document (String)

- **Programmer avec XML : DOM Parser Java script**

Propriétés	Définition
x.nodeName	Nom de l'élément x
x.nodeValue	Valeur de l'élément x
x.nodeType	Retour le code de l'élément x 1 – element 2 – attribute 3 – texte 4 – commentaire 5 – document
x.parentNode	Parent de l'élément x
x.childNodes	Enfants de l'élément x
x.attributes	Attributs de l'élément x

• Programmer avec XML : DOM Parser Java script

Méthodes	Définition
x.getElementsByTagName(*nom*)	Récupère tous les éléments marqué par le *nom* spécifié
x.appendChild(node)	Ajoute un enfant au noeud x
x.removeChild(node)	Supprime un enfant du noeud x

• Programmer avec XML : DOM Parser Java script

```
<script language="Javascript">
    xmlDoc=loadXMLDoc("XSLTFirstSample.xml");
    x=xmlDoc.getElementsByTagName("films").childNodes;
    for (i=0;i<x.length;i++)
    {
        firstGroupElt=x[i].getElementsByTagName("group")[0];
        groupEltAttribute=  firstGroupElt.getAttribute("name");
        if(groupEltAttribute== "Star Wars"){
            ...
        }
    }
</script>
```

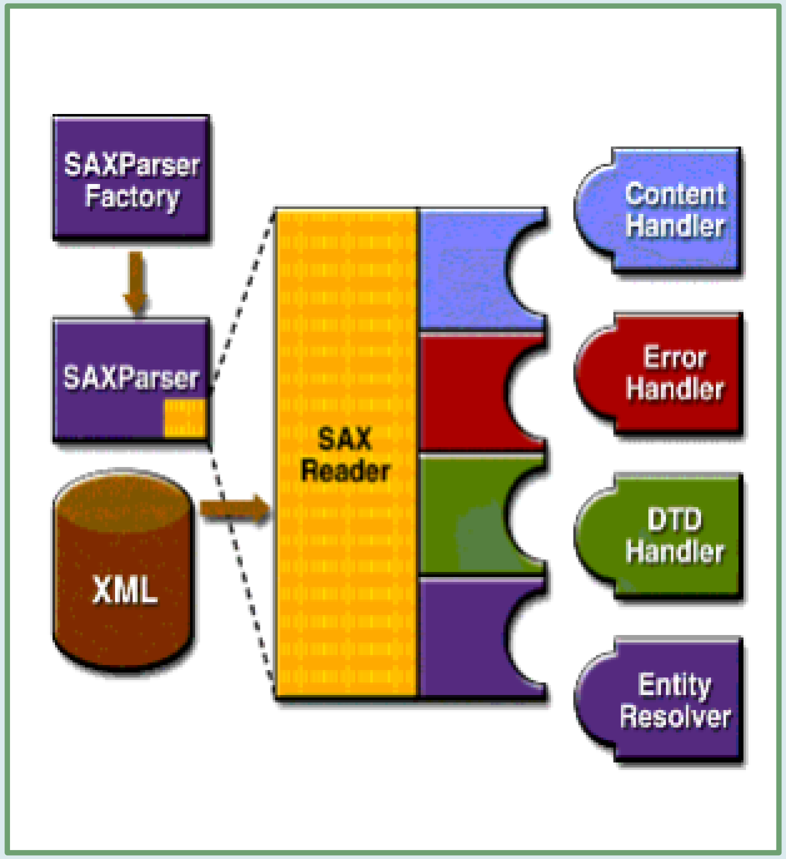
Utilisation XML

- Programmer avec XML
- DOM
- SAX
- JAXP
- JAXB

- **Programmer avec XML : SAX**
  - ❑ Simple API for XML
    - → Accèder et manipuler les documents XML
  - ❑ Evénementielle
  - ❑ Lecture de flux de données XML
  - ❑ Consommateur de données, rien n’est conservé en mémoire

Programmer avec XML : SAX

- **SAXParserFactory :** créateur de parser
- **SAXParser:** parcours le document XML
- **SAXReader:** gestion communication SAX event <-> handler
- **DefaultHandler:** implement ContentHandler, ErrorHandler, DTDHandler, EntityResolver.
- **ContentHandler:** startDocument, endDocument, startElement, endElement ...
- **ErrorHandler:** error, fatalError pendant la lecture
- **DTDHandler:** peu utilisé, utilisé pour les unparsed entity
- **EntityResolver:** utilise les URI/URN pour trouver les documents/information



• Programmer avec XML : DOM Parser Java script

Propriétés	Définition
ContentHandler	Interface permettant de recevoir les notifications des contenus logiques d'un document
setDocumentLocator	Défini un objet permettant de localiser l'origine des evenements SAX
start/endDocument	Notification de début/fin de document
start/endPrefixMapping	Notification pour chaque instruction de fonctionnement rencontré (<?xml ….?>)
start/endElement	Notification de début/fin d'un élément
characters	Notification de données de charactères
ignorableWhiteSpace	Notification de non prise en compte des espaces dans le contenu d'un élément
skippedEntity	Notification de non traitement d'une entité

XML

• Programmer avec XML : SAX

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<films>
    <category name="science-fiction">
        <articles id="14256">
            <group name="Star Wars">
                <document id="12454">
                    <title>A New Hope</title>
                    <date>1977</date>
                    <synopsis>
                        A New Hope opens with a Rebel ship being
                        boarded by the tyrannical Darth Vader.
                        The plot then follows the life of a simple
                        ...
                    </synopsis>
                </document>
                <document id="12455">
                    <title>The Empire Strikes Back</title>
                    <date>1980</date>
                    <synopsis>
                        Fleeing the evil Galactic Empire, the Rebels
                        abandon their new base in an assault with the
                        Imperial AT-AT walkers on the ice world of Hoth.
                        Princess Leia, Han Solo, Chewbacca and the droid
                        ...
                    </synopsis>
                </document>
                <document id="12456">
                    <title>Return of the jedi</title>
                    <date>1983</date>
                    <synopsis>
                        None.
                    </synopsis>
                </document>
            </group>
        </articles>
    </category>
</films>
```

Only Start Element event

```
Visited element : films
Visited element : category
Visited element : articles
Visited element : group
Visited element : document
Visited element : title
Visited element : date
Visited element : synopsis
Visited element : document
Visited element : title
Visited element : date
Visited element : synopsis
Visited element : document
Visited element : title
Visited element : date
Visited element : synopsis
```

• Programmer avec XML : SAX

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<films>
    <category name="science-fiction">
        <articles id="14256">
            <group name="Star Wars">
                <document id="12454">
                    <title>A New Hope</title>
                    <date>1977</date>
                    <synopsis>
                        A New Hope opens with a Rebel ship being
                        boarded by the tyrannical Darth Vader.
                        The plot then follows the life of a simple
                        ...
                    </synopsis>
                </document>
                <document id="12455">
                    <title>The Empire Strikes Back</title>
                    <date>1980</date>
                    <synopsis>
                        Fleeing the evil Galactic Empire, the Rebels
                        abandon their new base in an assault with the
                        Imperial AT-AT walkers on the ice world of Hoth.
                        Princess Leia, Han Solo, Chewbacca and the droid
                        ...
                    </synopsis>
                </document>
                <document id="12456">
                    <title>Return of the jedi</title>
                    <date>1983</date>
                    <synopsis>
                        None.
                    </synopsis>
                </document>
            </group>
        </articles>
    </category>
</films>
```

All events

```
Start document
start element : films
start element : category
start element : articles
start element : group
start element : document
start element : title
end element : title
start element : date
end element : date
start element : synopsis
end element : synopsis
end element : document
start element : document
start element : title
end element : title
start element : date
end element : date
start element : synopsis
end element : synopsis
end element : document
start element : document
start element : title
end element : title
start element : date
end element : date
start element : synopsis
end element : synopsis
end element : document
end element : group
end element : articles
end element : category
end element : films
End document
```

• Programmer avec XML : SAX

Enregistrement d'un handler

Content Handler

```
String fileName = "…";
ContentHandler myContentHandler = new …;
…/…
XMLReader parser = XMLReaderFactory.createXMLReader();
parser.setContentHandler(myContentHandler);
parser.parse(fileName);
```

DTDHandler

```
String fileName = "…";
ContentHandler myContentHandler = new …;
DTDHandler myDTDHandler = new …;
ErrorHandler myErrorHandler = new …;
EntityResolver myEntityResolver = new …;
…/…
XMLReader parser = XMLReaderFactory.createXMLReader();
parser.setContentHandler(myContentHandler);
parser.setDTDHandler(myDTDHandler);
parser.setErrorHandler(myErrorHandler);
parser.setEntityResolver(myEntityResolver);
parser.parse(fileName);
```

Default

```
String fileName = "…";
DefaultHandler handler = new …;
…/…
XMLReader parser = XMLReaderFactory.createXMLReader();
parser.setContentHandler(handler);
parser.setDTDHandler(handler);
parser.setErrorHandler(handler);
parser.setEntityResolver(handler);
parser.parse(fileName);
```

• Programmer avec XML : SAX

```
import java.io.IOException;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;

public class XMLSaxSample extends DefaultHandler {
  public XMLSaxSample() {}
  static public void main(String[] args) {
    try {
        DefaultHandler handler = new XMLSaxSample();
        XMLReader parser = XMLReaderFactory.createXMLReader();
        parser.setContentHandler(handler);
        parser.parse("XSLTFirstSample.xml");
    } catch (SAXException e) {e.printStackTrace();
    } catch (IOException e) {e.printStackTrace();
    }
  }

  @Override
  public void startElement(String uri, String localName, String qName, Attributes atts) {
    System.out.println("Visited element : " + qName);
  }
```

Programmer avec XML : DOM vs SAX

- DOM
  - + Accès aléatoire au document
  - + Vue hiérarchique du document
  - + Possibilité de modifier le document
  - – Nécessite de charger la totalité en mémoire
- SAX
  - + Rapide
  - + gestion en flux (peu de ressource mémoire)
  - – Accès séquentiel
  - – Pas de modification du document

Utilisation XML

- Programmer avec XML
- DOM
- SAX
- JAXP
- JAXB

Programmer avec XML : JAXP

- Java API for XML Processing
  - → parse XML document
  - → transform XML document
  - → validate XML document
- API Independante (DOM/SAX…)
- Support XSLT
- Support des espaces de noms

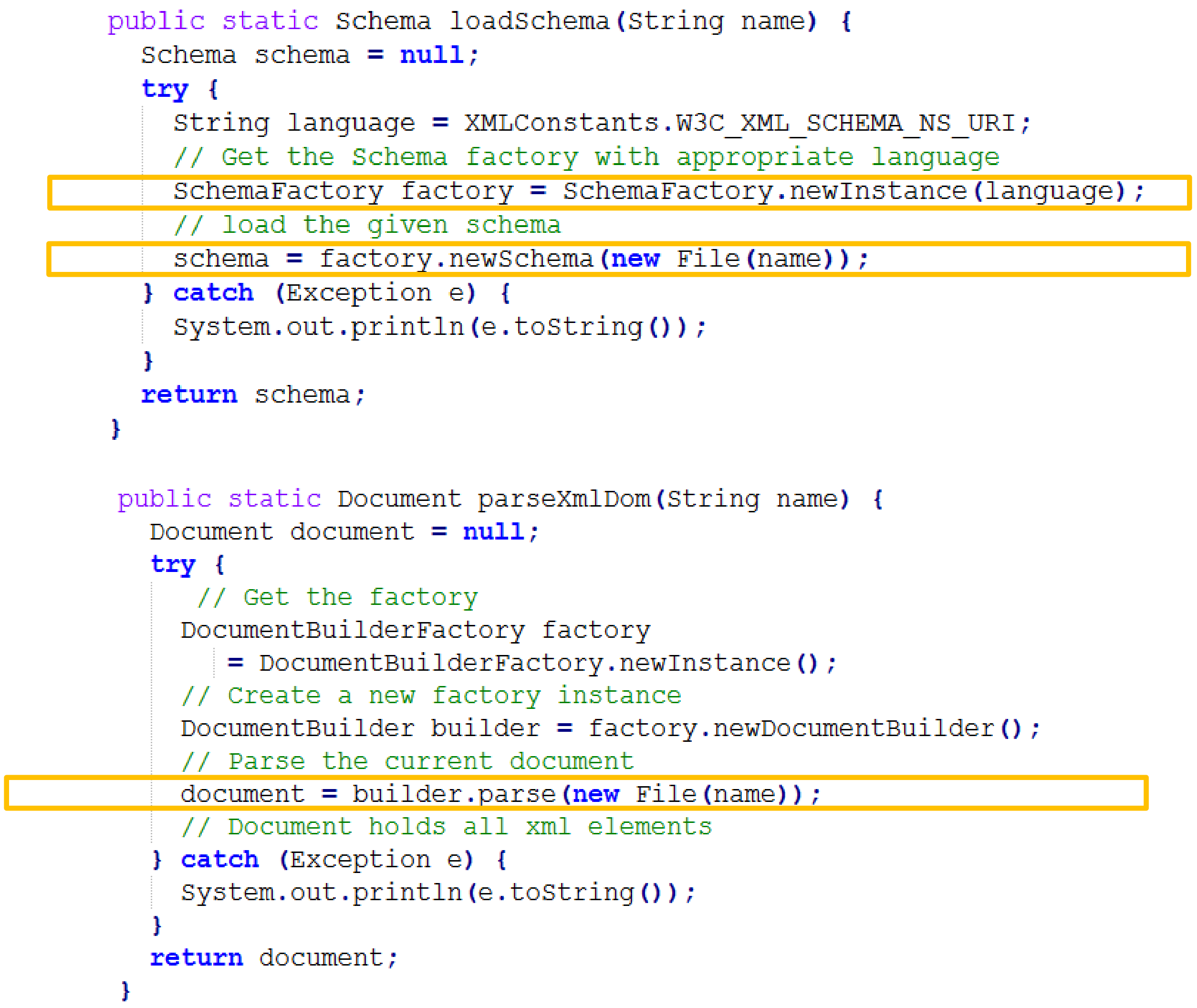
- **Programmer avec XML : JAXP**
  - ❑ API (javax.xml.parsers)
    - javax.xml.parsers.SAXParserFactory
    - javax.xml.parsers.DocumentBuilderFactory
    - javax.xml.validation.SchemaFactory
    - javax.xml.transform.TransformerFactory

• Programmer avec XML : JAXP : Validation avec un Schema XML

```java
import java.io.File;

import javax.xml.XMLConstants;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;

import org.w3c.dom.Document;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class JAXPValidation {

  private static int errorCount = 0;
  public static void main(String[] a) {
      String schemaFileName = "XMLSchemaSampleCorr.xsd";
      String xmlFileName = "XMLSchemaSample2.xml";
      Schema schema = loadSchema(schemaFileName);
      Document document = parseXmlDom(xmlFileName);
      validateXml(schema, document);
  }
```

```java
public static void validateXml(Schema schema,
                                Document document) {
  try {
    // Creating a Validator instance
    Validator validator = schema.newValidator();

    // Redirect error to my own error handler
    validator.setErrorHandler(new MyErrorHandler());

    // XML Validation
    validator.validate(new DOMSource(document));

    if (errorCount>0) {
      System.out.println("Document Validation encounter" +
                " error: "+errorCount);
    } else {
      System.out.println("Passed.");
    }

  } catch (Exception e) {
    // Catching all validation exceptions
    System.out.println(e.toString());
  }
}
```

• Programmer avec XML : JAXP : Validation avec un Schema XML



```
public static Schema loadSchema(String name) {
  Schema schema = null;
  try {
    String language = XMLConstants.W3C_XML_SCHEMA_NS_URI;
    // Get the Schema factory with appropriate language
    SchemaFactory factory = SchemaFactory.newInstance(language);
    // load the given schema
    schema = factory.newSchema(new File(name));
  } catch (Exception e) {
    System.out.println(e.toString());
  }
  return schema;
}

public static Document parseXmlDom(String name) {
  Document document = null;
  try {
     // Get the factory
    DocumentBuilderFactory factory
       = DocumentBuilderFactory.newInstance();
    // Create a new factory instance
    DocumentBuilder builder = factory.newDocumentBuilder();
    // Parse the current document
    document = builder.parse(new File(name));
    // Document holds all xml elements
  } catch (Exception e) {
    System.out.println(e.toString());
  }
  return document;
}
```

• Programmer avec XML : JAXP : Validation avec un Schema XML

```
static class MyErrorHandler implements ErrorHandler {
      public void fatalError( SAXParseException e )
         throws SAXException {
        System.out.println(e.toString());
        throw e;
      }
      public void error( SAXParseException e ) throws SAXException {
        System.out.println(e.toString());
        errorCount++;
        // continue with validatin process
        // throw e;
      }
      public void warning( SAXParseException e ) throws SAXException {
          //display warning during the validation process
        System.out.println(e.toString());
      }
}
```

Programmer avec XML : JAXP : Transformation à partir d'un fichier XSLT

```java
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

public class JAXPTransformation {

  public static void main(String[] a) {
      String xsltFileName = "XSLTFirstTransformationSample.xsl";
      String xmlFileName = "XSLTFirstSample.xml";
      String outputFileName="result.html";
      Transformer trans = loadXslt(xsltFileName);
      transformXml(trans, xmlFileName,outputFileName);
  }
```

• Programmer avec XML : JAXP : Transformation à partir d'un fichier XSLT

```java
public static Transformer loadXslt(String name) {
  Transformer trans=null;
  try {
      // Get the tranformer factory
      TransformerFactory transFact = TransformerFactory.newInstance( );
      // Create a new tranformer with the given XSLT file name
      trans = transFact.newTransformer(new StreamSource(name));
  } catch (Exception e) {
    System.out.println(e.toString());
  }
  return trans;
}

public static void transformXml( Transformer trans, String xmlFileName, String outputFileName) {
  try {
      FileOutputStream os =new FileOutputStream(outputFileName);
      // Apply the transformation to the given source (DOM is automatically load)
      // The steam result contain the result file
      trans.transform(
              new StreamSource(xmlFileName), new StreamResult(os));

  } catch (Exception e) {
    // Catching all transformation exceptions
    System.out.println(e.toString());
  }
}
```

Utilisation XML

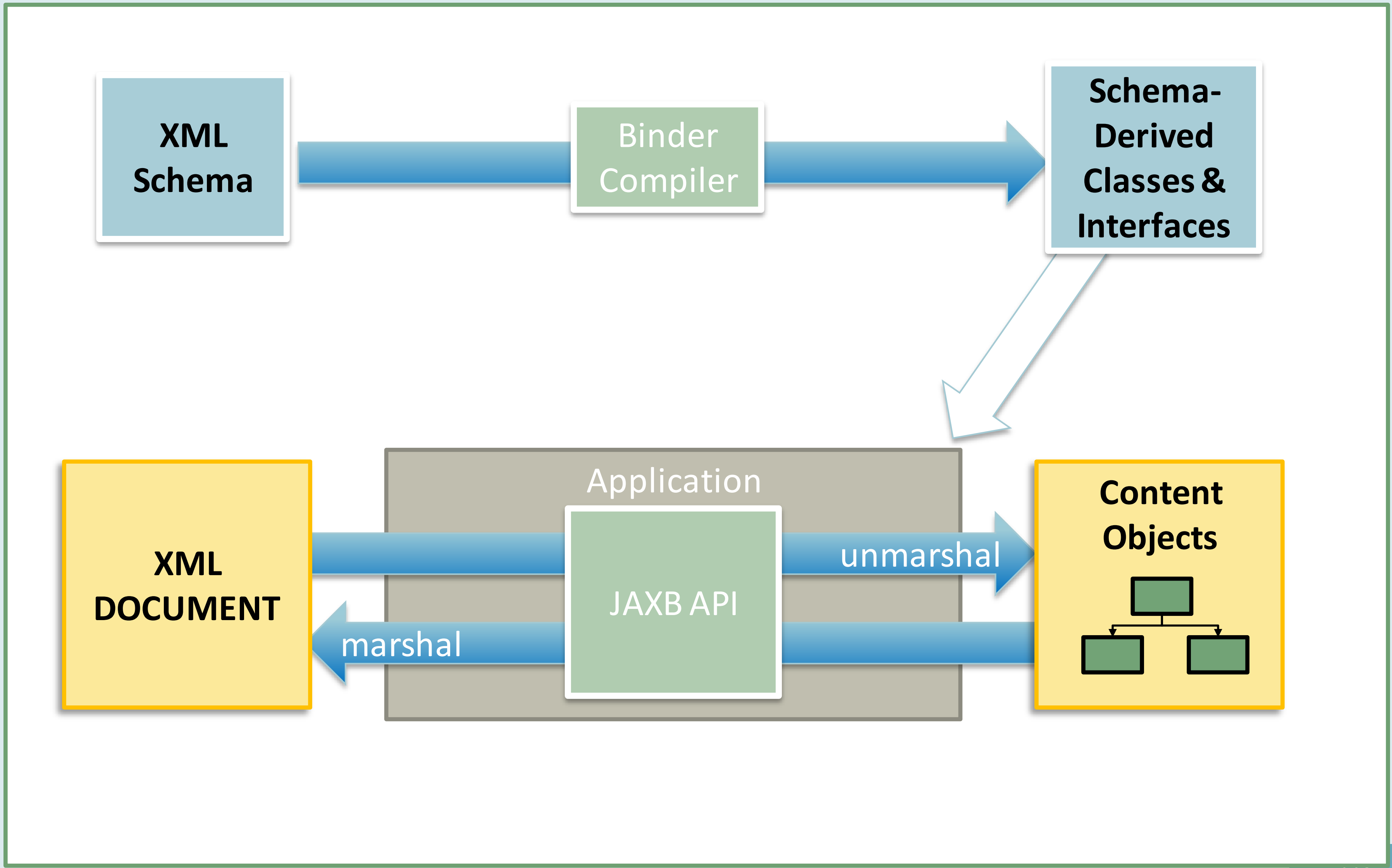
- Programmer avec XML
- DOM
- SAX
- JAXP
- JAXB

- **Programmer avec XML : JAXB**
  - ❑ Java **A**rchitecture for **X**ml **B**inding
    - → parse XML document
    - → Validate XML document
    - → transform XML document

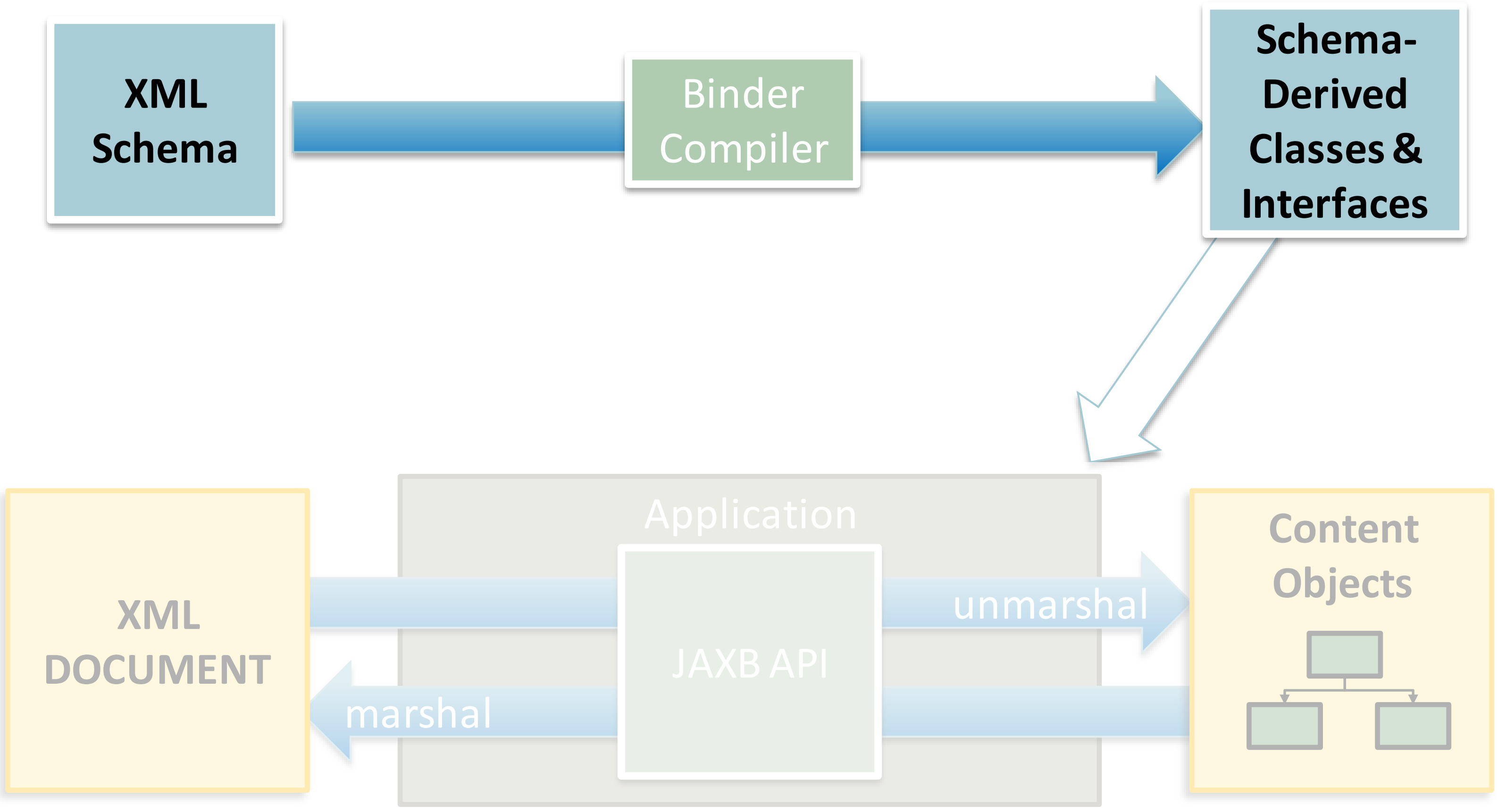
→ Tranformation (UnMarshal) du document XML en objets java

→ Tranformation (Marshal) d'objet java en document XML

• **Programmer avec XML : JAXB**



• Programmer avec XML : JAXB



xjc -p SampleJaxb book.xsd

• Programmer avec XML : JAXB Unmarshall

```
<xsd:element name="article" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="title"
                type="xsd:string" minOccurs="1"/>
            <xsd:element name="authors" >
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="author"
                            type="xsd:string"
                            minOccurs="1" maxOccurs="5"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="document" type="xsd:string"
                minOccurs="1" maxOccurs="5" />
            <xsd:element name="copyright" type="xsd:integer"
                minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="date"
            type="xsd:date" use="required"/>
        <xsd:attribute name="id"
            type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
```

Extrais de book.xsd

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "title",
    "authors",
    "document",
    "copyright"
})
@XmlRootElement(name = "article")
public class Article {

    @XmlElement(required = true)
    protected String title;
    @XmlElement(required = true)
    protected Article.Authors authors;
    @XmlElement(required = true)
    protected List<String> document;
    protected BigInteger copyright;
    @XmlAttribute(name = "date", required = true)
    @XmlSchemaType(name = "date")
    protected XMLGregorianCalendar date;
    @XmlAttribute(name = "id", required = true)
    protected String id;

    public String getTitle() {
        return title;
    }

    ...
```

Extrais Article.java

Programmer avec XML : JAXB Marshall

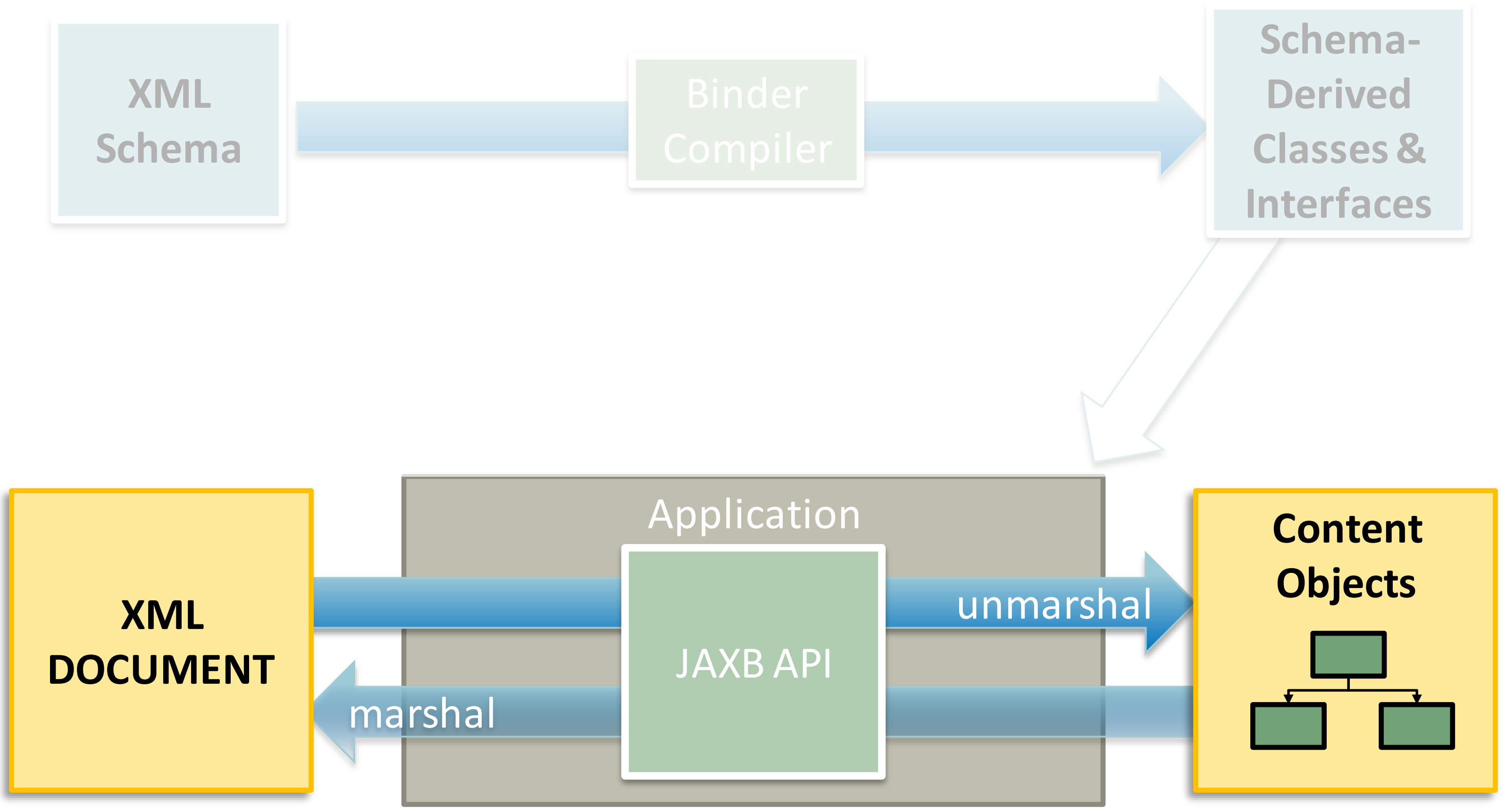
```xml
<xsd:element name="article" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="title"
                type="xsd:string" minOccurs="1"/>
            <xsd:element name="authors" >
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="author"
                            type="xsd:string"
                            minOccurs="1" maxOccurs="5"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="document" type="xsd:string"
                minOccurs="1" maxOccurs="5" />
            <xsd:element name="copyright" type="xsd:integer"
                minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="date"
            type="xsd:date" use="required"/>
        <xsd:attribute name="id"
            type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
```

Extrais de book.xsd

```java
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "author"
})
public static class Authors {

    @XmlElement(required = true)
    protected List<String> author;

    public List<String> getAuthor() {
        if (author == null) {
            author = new ArrayList<String>();
        }
        return this.author;
    }
```

Extrais Article.java

Programmer avec XML : JAXB



```
xjc -p SampleJaxb book.xsd
```

• Programmer avec XML : JAXB

```
public static void main(String[] args) {
    try{

  1      JAXBContext jc = JAXBContext.newInstance( "SampleJAXB" );

  2      Unmarshaller u = jc.createUnmarshaller();

  3      u.setSchema(loadSchema("XMLSchemaSampleCorr.xsd"));

         Book book =
  4        (Book)u.unmarshal(
             new FileInputStream("book.xml"));

         System.out.println(book);

      } catch( UnmarshalException ue ) {
        System.out.println( "Caught UnmarshalException" );
      } catch( JAXBException je ) {
        je.printStackTrace();
      } catch( IOException ioe ) {
        ioe.printStackTrace();
    }
}
```

1 Localisation des classes java correspondantes au xsd

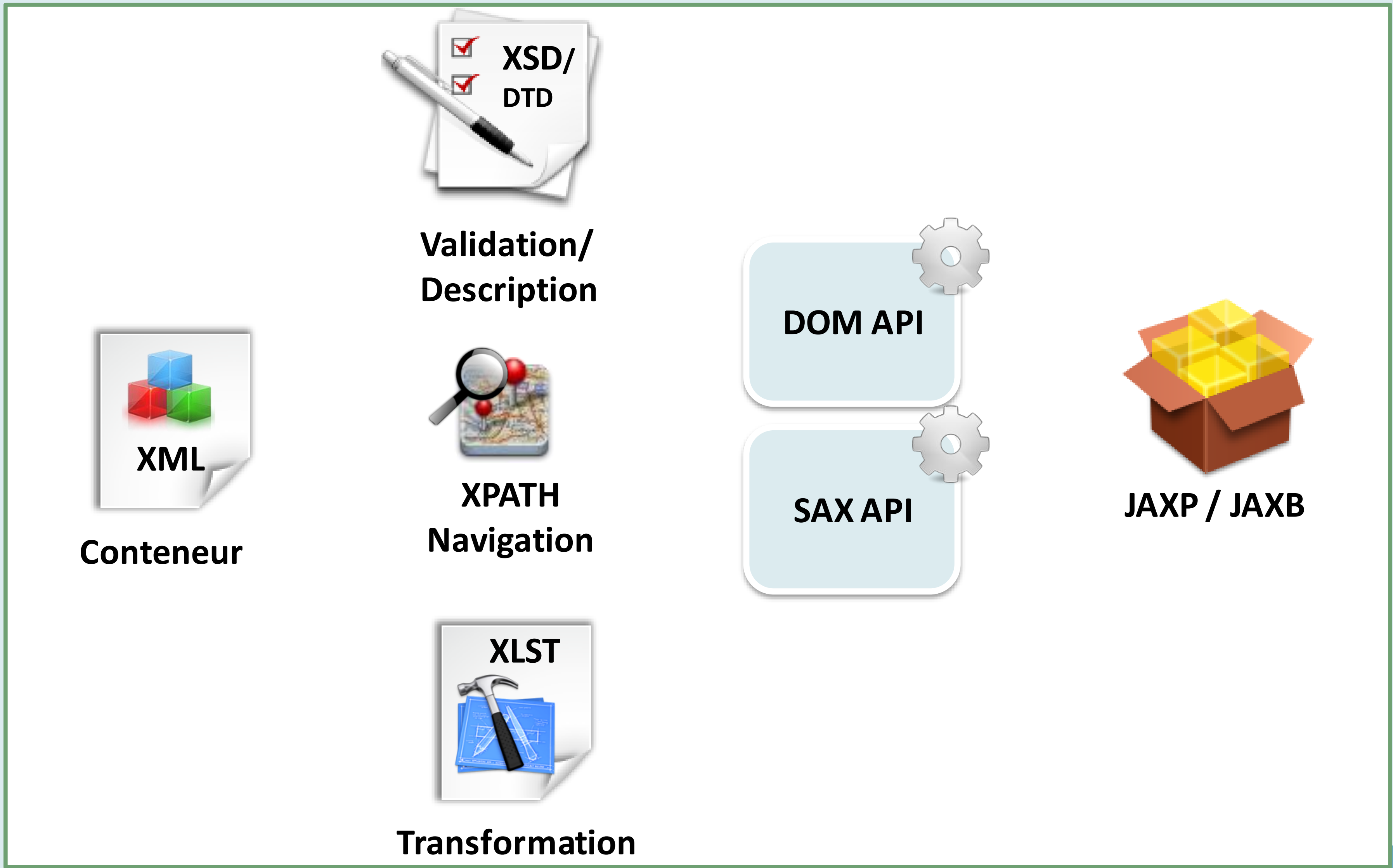
2 Creation d’un unmarshaller pour le contexte donné

3 Ajout du schema XML pour la validation

4 Importation du contenu de book.xml dans l’objet java Book

Conclusion

- Conclusion


XSD/
DTD
Validation/
Description
XML
Conteneur
XPATH
Navigation
XLST
Transformation
DOM API
SAX API
JAXP / JAXB

Sources

- Olivier Carton, L'essentiel de XML,Université Paris Diderot, 2012
- W3C School, XML, http://www.w3schools.com/xml/default.asp
- W3C School, XML XSLT, http://www.w3schools.com/xml/xml_xsl.asp
- W3C , XML SCHEMA Part 2, http://www.w3.org/TR/xmlschema-2/
- Technologies Internet et Education, Eléments de programmation XSLT, http://tecfa.unige.ch/guides/tie/html/xml-xslt2/xml-xslt2-6.html
- Jean-luc Massat, Java & XML, http://jean-luc.massat.perso.luminy.univmed.fr/ens/xml/6-java.html
- Ioan Calapodescu, java.developpez.com, FAQ Java XML, http://java.developpez.com/faq/xml/?page=dom
- Sebastien Meric, Lecture d'un flux XML via SAX, http://smeric.developpez.com/java/cours/xml/sax, 2003
- Philippe Poulard, INRIA, DOM & SAX,http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/06-xml-dom-sax.pdf
- Mohamed Sallami, La technologie XML,2012
- Oracle,The J2EE 1.4 Tutorial, http://docs.oracle.com/javaee/1.4/tutorial/doc/, 2005
- Herong Yang, JAXP - XML Schema (XSD) Validation, http://www.herongyang.com/XML-Schema/JAXP-XSD-XML-Schema-Validation.html, 2012
- Oracle, Java Architecture for XML Binding (JAXB), http://www.oracle.com/technetwork/articles/javase/index-140168.html, 2003

XML

- Liens utiles
  - D Vint, XML SCHEMA – Data Types Quick Reference, 2003
  - W3C validators,XML validator, http://validator.w3.org
  - FreeFormatter, XML Validator –XSD (XML Schema), http://www.freeformatter.com/xml-validator-xsd.html
  - Whitebeam, Xpath Expression Testbed, http://www.whitebeam.org/library/guide/TechNotes/xpathtestbed.rhtm, 2008
  - W3C School, test XSLT, http://www.w3schools.com/xsl/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog

Questions ?