TD2 :Correction partie espaces des noms

Ex1.Espace des noms

Objectif: Cette section a pour objectif de vous familiariser avec la portée des espaces des noms dans un document XML (à commenter uniquement).

Q : Pour chaque portion de code, associer chaque élément et attribut à son espace des noms:

- ```
<lower:aaa xmlns:lower = "http://zvon.org/lowercase" >
  <lower:bbb xmlns:lower = "http://zvon.org/lowercase" >
   <lower:ccc xmlns:lower = "http://zvon.org/lowercase" />
  </lower:bbb>
  <upper:BBB xmlns:upper = "http://zvon.org/uppercase" >
   <upper:CCC xmlns:upper = "http://zvon.org/uppercase" />
  </upper:BBB>
  <xnumber:x111 xmlns:xnumber = "http://zvon.org/xnumber" >
   <xnumber:x222 xmlns:xnumber = "http://zvon.org/xnumber" />
  </xnumber:x111>
</lower:aaa>
```

  Correction: Pour les déclaration d'espace de noms, des attributs réservés commençant par "xmlns" sont utilisés. Vous pouvez déclarer un espace de nom pour chaque élément utilisé, mais cette approche est fastidieuse et le code résultant est difficile à lire.

- ```
<lower:aaa xmlns:lower = "http://zvon.org/lowercase" >
  <lower:bbb >
   <lower:ccc />
  </lower:bbb>
  <upper:BBB xmlns:upper = "http://zvon.org/uppercase" >
   <upper:CCC />
  </upper:BBB>
  <xnumber:x111 xmlns:xnumber = "http://zvon.org/xnumber" >
   <xnumber:x222 />
  </xnumber:x111>
</lower:aaa>
```

  Correction: Déclarer les espaces de noms comme dans l'exemple précédent serait peu pratique et source d'erreurs. Le standard prévoit plusieurs moyens d'accomplir cette tâche. La déclaration d'espace de noms pour un élément est aussi valide pour tous les éléments inclus dans celui-ci (enfants et descendants).

- ```
<lower:aaa xmlns:lower = "http://zvon.org/lowercase" xmlns:upper = "http://zvon.org/uppercase" xmlns:xnumber = "http://zvon.org/xnumber" >
  <lower:bbb >
   <lower:ccc />
  </lower:bbb>
  <upper:BBB >
   <upper:CCC />
  </upper:BBB>
  <xnumber:x111 >
   <xnumber:x222 />
  </xnumber:x111>
</lower:aaa>
```
```

Correction: Il est convenu de déclarer un espace de noms dans l'élément racine.

- ```
<lower:aaa xmlns:lower = "http://zvon.org/lowercase" xmlns:upper = "http://zvon.org/lowercase" xmlns:xnumber = "http://zvon.org/lowercase" >
  <lower:bbb >
   <lower:ccc />
  </lower:bbb>
  <upper:BBB >
   <upper:CCC />
  </upper:BBB>
  <xnumber:x111 >
   <xnumber:x222 />
  </xnumber:x111>
</lower:aaa>
```

Correction: Ce n'est pas le préfixe qui identifie l'espace de noms mais la valeur de l'attribut xmlns. Dans cet exemple, tous les éléments appatiennent au même espace de noms bien qu'ils aient des prefixes différents.

- ```
<aaa >
  <lower:bbb xmlns:lower = "http://zvon.org/lowercase" >
   <lower:ccc />
  </lower:bbb>
  <lower:BBB xmlns:lower = "http://zvon.org/uppercase" >
   <lower:CCC />
  </lower:BBB>
  <lower:x111 xmlns:lower = "http://zvon.org/xnumber" >
   <lower:x222 />
  </lower:x111>
</aaa>
```

Correction: Alors que dans l'exemple précédent Exemple 5 , les éléments appartenaient au même espace de noms avec des prefixes différents, dans le cas présent, ils ont chacun un espace de noms particulier bien qu'ils aient tous le même prefixe.

- ```
<aaa >
  <bbb xmlns = "http://zvon.org/lowercase" >
   <ccc />
  </bbb>
  <BBB xmlns = "http://zvon.org/uppercase" >
   <CCC />
  </BBB>
  <x111 xmlns = "http://zvon.org/xnumber" >
   <x222 />
  </x111>
</aaa>
```

Correction: Les espaces de noms n'ont pas besoin d'être déclarés explicitement avec des prefixes. L'attribut xmlns défini l'espace de noms par défaut d'un élément où il intervient et pour tous ses enfants et descendants.

- ```
<aaa xmlns:upper = "http://zvon.org/uppercase" xmlns:xnumber = "http://zvon.org/xnumber" >
  <bbb xmlns = "http://zvon.org/lowercase" >
   <ccc />
   <upper:WWW />
   <xnumber:x666 />
  </bbb>
  <BBB xmlns = "http://zvon.org/uppercase" >
   <upper:WWW />
```

```
  <xnumber:x666 />
  <CCC />
 </BBB>
 <x111 xmlns = "http://zvon.org/xnumber" >
  <x222 />
  <upper:WWW />
  <xnumber:x666 />
 </x111>
</aaa>
```

Correction: Même si un espace de noms par défaut est utilisé, des espace de noms pour certains éléments peuvent être explicitement déclarés.

- ```
  <aaa xmlns = "http://zvon.org/lowercase" >
   <bbb >
    <ccc xmlns = "" >
     <ddd />
    </ccc>
   </bbb>
  </aaa>
  ```

  Correction: Si la valeur de l'espace de noms est une chaîne vide la déclaration d'espace de noms par défaut est annulée.

-
  ```

TD2: Correction partie schéma

EX2: Déclarations simples

Objectif: Mise en jambe

Q : Qu'est ce? (à commenter uniquement).

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="root" type="xsd:integer"/>
</xsd:schema>
```

Q : Qu'est ce? (idém).

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="root">
   <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
     <xsd:maxExclusive value="25"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
</xsd:schema>
```

Q : Qu'est ce? (idém).

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="root">
   <xsd:simpleType>
    <xsd:union>
     <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
       <xsd:minInclusive value="0"/>
       <xsd:maxInclusive value="100"/>
      </xsd:restriction>
     </xsd:simpleType>
     <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
       <xsd:minInclusive value="300"/>
       <xsd:maxInclusive value="400"/>
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:union>
   </xsd:simpleType>
  </xsd:element>
</xsd:schema>
```

Q : Écrivez un XML Schéma qui donne un modèle de document XML ne contenant qu'un élément "AAA". Cet élément ne contient que du texte.

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="AAA" type="xsd:string"/>
</xsd:schema>
```

Q : Nous voulons un élément racine "AAA" ne contenant qu'un élément "BBB" et qu'un élément

"CCC" (ces deux éléments sont de type string, par exemple). L'ordre d'apparition n'est pas important. On utilisera le pattern (patron?) "all" avec ses attributs "minOccurs" et "maxOccurs"

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="AAA">
   <xsd:complexType >
    <xsd:all minOccurs="1" >
     <xsd:element name="BBB" type="xsd:string"/>
     <xsd:element name="CCC" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Q : Nous voulons un élément racine "AAA" ne contenant qu'un élément "BBB" suivis d'un élément "CCC"(ces deux éléments sont de type string, par exemple). L'ordre d'apparition est pas important. Utilisez le pattern (patron?) "sequence"; ses attributs ne sont pas nécessaires, leur valeurs par défaut étant 1.

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="AAA">
   <xsd:complexType >
    <xsd:sequence >
     <xsd:element name="BBB" type="xsd:string"/>
     <xsd:element name="CCC" type="xsd:string"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Q : Ici, l'élément "AAA" contient un nombre quelconque d'éléments "BBB" et "CCC" (peut être zéro). On utilisera le pattern "sequence" avec les bonnes valeurs d'attributs. Le but ici est de voir la fonction des attributs "minOccurs" du pattern "sequence" et ceux des éléments fils (inclus dans la séquence). Le document suivant est valide:

```
<AAA xsi:noNamespaceSchemaLocation="correct_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <BBB>111</BBB>
  <CCC>YYY</CCC>
  <BBB>222</BBB>
  <BBB>333</BBB>
  <CCC>ZZZ</CCC>
</AAA>
```

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="AAA">
   <xsd:complexType >
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
     <xsd:element name="BBB" type="xsd:string"minOccurs="0" />
     <xsd:element name="CCC" type="xsd:string"minOccurs="0" />
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Q : Désormais, l'élément "AAA" contient soit "BBB", soit "CCC". On utilisera le pattern "choice". 5

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="AAA">
   <xsd:complexType >
    <xsd:choice >
     <xsd:element name="BBB" type="xsd:string"/>
     <xsd:element name="CCC" type="xsd:string"/>
    </xsd:choice>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Q : Nous voulons un élément racine "root" qui ne contient que du texte et un seul attribut ("xx")

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="root">
   <xsd:complexType>
    <xsd:simpleContent>
     <xsd:extension base="xsd:string">
      <xsd:attribute name="xx" type="xsd:string" use="required"/>
     </xsd:extension>
    </xsd:simpleContent>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Q : Écrivez un XML Schéma qui donne un modèle de document XML ne contenant qu'un élément "AAA". Cet élément racine ne contient que du texte...qui est l'élément par défaut...

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="AAA" type="xsd:string"/>
</xsd:schema>
```

Q : Nous voulons maintenant que cet élément racine "AAA" puisse contenir à la fois du texte et un élément "BBB". Pour cela, il faut fixer l'attribut "mixed" de "complexType" à "vrai".

Correction:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="AAA">
   <xsd:complexType mixed="true">
    <xsd:sequence >
     <xsd:element name="BBB" type="xsd:string"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```