XML Avancée 2015/2016

~~difference~~ - Avantage de xml sur HTML:

- Lisibilité
- aucune connaissance ne doit théoriquement être necessaire pour comprendre un contenu du document XML.
- autodescriptif et extensible (on peut etendre ses règle et definir son propre type)
- Structure arborescente qui permet de modeliser la majorité des problèmes informatique.
- universalité et portabilité
- les differences de caractère sont prise en compte.
- deployable, il peut être facilement distribué par n'importe quelle **protocol.**

EX01 : ~~Le~~

1 - Ecrire la DTD de recette.

```
<!DOCTYPE restaurant [
   <!ELEMENT restaurant (recette+)>
   <!ELEMENT recette (composition, Procedure)>
      <!ATTLIST recette
                nom CDATA #REQUIRED
                Type (dessert | ~~plat~~ resistance | entree) "entree" >
   <!ELEMENT composition ( ingredient+ )>
      <!ATTLIST composition nombrePlat CDATA #Required>
   <!ELEMENT ingredient (#PCDATA)>
      <!ATTLIST CDATA #Required>
   <!ELEMENT Procedure (#PCDATA)
]>
```

EXO2 : recette v1.0

2- document xml.

```
<?xml version="1.0" encoding="UTF-8"?>
  <!Doctype restaurant SYSTEM recette.dtd>
  <restaurant>
    <recette nom="mousse au chocolat" type="dessert">
      <composition nombre_Plats="4">
        <ingredient quantite="500g"> chocolat </ingredient>
        <ingredient quantite="4"> oeufs frais </ingredient>
      </composition>
      <Proprecedure>
        com
      </Procedure>
    </recette>
  </restaurant>.
```

3- XMLschema:

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLschema">
    <xs:element name="composition">
      <xs:complexType>
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="ingredient"/>
        </xs:sequence>
        <xs:attribute name="nombre_plats" type="xs:
                    string" use="required">
      </xs:complextype>
    </xs:element>
```

EXO2 : requête Xpath

on donne le document XML maraicher.xml suivant:

1 - l'élément racine : / recupere tout le document

/* recupere à partir de l'élement racine. (/produits)

2 - tout les descendants de la racine: //Produit

pour avoir les fruits //fruit.

// veut dire le niveau à partir duquel on veut lister les elements.

3 - Lister tout les attributs: //@*

4 - les differents producteurs de fruits: //fruit/producteur/text()

→ le deuxieme producteur de fruits: //fruit[2]/producteur/text()

5 - les legumes de calibre 2: //legume[@calibre='2']

6 - deuxième producteur des legumes: ~~//Producteur~~

//legume[2]/Producteur/text()

7 - nombre de fruits presentés count(//fruit)

8 - les legumes et les fruits dont le producteur est marius production: //Producteur[text()="marius Produ"]/~~legume~~..

//Producteur[text()="marius Produ"]/..

9 - la quantité totale des fruits

~~Count~~ sum(//fruits/qtity)

10 - le prix global des fruits presentés:

sum((//fruits[qtity)*(//fruits/@prix))

XML Avancé 2014-2015

Exo1 Créons un document xml qui soit valide à la dtd

```
<? xml version ="1.0" encoding ="utf-8" >
<!Doctype Carnet System "Carnet.dtd" >
<Carnet>
    <Personne Nom ="GUEYAF" prenom ="NEGANE"
              tel ="695272525" />
    <Personne nom ="voufo" tel ="698315182" />
</Carnet>
```

. Empty veut dire que l'élément n'a pas de fils.