Séance 11 et 12 : ODD (2 et 3) – Personnalisations avancées

Pour personnaliser son schéma, il existe 4 manipulations principales :
- Ajouter des éléments ;
- Supprimer des éléments ;
- Changer des éléments ;
- Personnaliser les attributs et les valeurs d'attribut d'un élément.

I-Suppression d'un élément

1- Suppression simple d'un élément avec @mode="delete"
- <elementSpec ident="head" mode="delete"/>

2- Suppression d'un élément ou une classe dans un module avec @exclude
- <moduleRef key="core" except="head"/>

3- Non insertion dans un module ou une classe avec @include
- <moduleRef key="core" include="p head author title l lg"/>

II-Modification d'un élément

Modification d'un élément avec @mode="change"

```
<elementSpec ident="lg" mode="change">
            <attList>
              <attDef ident="part" mode="delete"/>
              <attDef ident="type" mode="change">
              <valList mode="add" type="closed">
                  <valItem ident="quatrain"/>
                  <valItem ident="sizain"/>
                  <valItem ident="sonnet"/>
                  <valItem ident="tercet"/>
                </valList></attDef>
            </attList>
</elementSpec>
```

III-Typer ses données

Utilisation dans l'élément de définition d'une balise ou d'un attribut <dataType> et de son enfant `<dataRef>`avec l'attribut @key pour pointer vers un type de données défini par la TEI

```
<attDef ident="n" mode="change">
          <datatype>
              <dataRef key="teidata.count"/>
         </datatype>
</attDef>
```

teidata.count =

```
<content>
 <dataRef name="nonNegativeInteger"/>
</content>
```

IV-Addition d'un élément

- Déclaration d'un élément dans le `<moduleRef>` avec @include
- `<moduleRef key="textstructure" include="TEI text body front"/>`

- Création d'un nouvel élément qui n'appartient pas au domaine TEI :

```
<elementSpec ident="alexandrin" mode="add" ns="http://www.example.com/ns/nonTEI">
   <classes>
      <memberOf key="model.lLike"/>
      <memberOf key="macro.paraContent"/>
   </classes>
   <content>
     <textNode/>
   </content>
</elementSpec>
```

IV-Les éléments structurants du schemaSpec

Les déclarations sont organisées grâce aux éléments suivants (liste non exhaustive) :

- moduleSpec (spécification de module) documente la structure, le contenu et les fonctions d'un module.
- moduleRef (référence de module) référence un module qui doit être incorporé dans un schéma.
- elementSpec (spécification d'élément) documente la structure, le contenu et l'emploi d'un élément.
- classSpec (spécification de classe) contient des informations de référence pour une classe d'éléments TEI, c'est-à-dire un groupe d'éléments qui figurent ensemble dans des modèles de contenu ou qui partagent un attribut commun, ou qui ont l'un et l'autre.

V-Les éléments structurants de la documentation

1-Dans la documentation rédigée de l'ODD

- specList (liste de spécification) marque l'endroit où insérer une liste de descriptions dans le texte documentaire ;
- specDesc (specification description) indique qu'une description de l'élément particulier ou de la classe particulière doit être incluse à ce point dans un document ;
- egXML et @xmlns="http://www.tei-c.org/ns/Examples permettent d'insérer des exemples en XML dans sa documentation ;

att (attribut) contient le nom d'un attribut apparaissant dans le courant du texte.
gi (identifiant générique) contient le nom d'un élément.
tag (balise) le contenu d'une balise ouvrante ou fermante, avec éventuellement des spécifications d'attributs, mais à l'exclusion des caractères marquant l'ouverture et la fermeture de la balise.
val (valeur) contient une seule valeur d'attribut.

2-Dans le schemaSpec dans les déclarations d'éléments

- Dans elementSpec ou attDef.

gloss (glose) identifie une expression ou un mot utilisé pour fournir une glose ou une définition.

desc (description) contient une courte description de l'objet documenté par son élément parent, qui comprend son utilisation prévue, son but, ou son application là où c'est approprié.

Exemple

```
<elementSpec ident="lem" mode="change">
 <gloss>Lemme</gloss>
 <desc>Permet de signaler la leçon choisie dans le texte édité.</desc>
 [...]
 </elementSpec>
```

VI-Personnaliser son ODD en pure ODD et schematron

1-Définir une séquence d'éléments

La règle définissant une séquence apparaît directement en dessus de la balise ouvrante de l'elementSpec dans un élément content.

La séquence est contenue dans un élément sequence avec un attribut preserveOrder qui permet de spécifier si l'ordre de déclaration des éléments de la séquence est signifiant.

Chaque élément est appelé à l'aide d'un elementRef et d'un attribut key qui permet de donner le nom de l'élément. On peut également définir les modalités d'apparition des éléments de la séquence à l'aide des attributs minOccurs et maxOccurs.

Exemple

```
<elementSpec ident="div1" mode="change">
  <content>
    <sequence preserveOrder="true">
      <elementRef key="head" minOccurs="1" maxOccurs="1"/>
        <elementRef key="p" minOccurs="1" maxOccurs="unbounded"/>
     </sequence>
  </content>
</elementSpec>
```

NB : Pour autoriser du texte comme contenu, on peut ajouter dans la séquence : <textNode/>

2-Contraindre un élément en fonction du contexte, schematron

*- La contrainte est introduite par un élément <constraintSpec>. Le langage utilisé est déclaré dans l'attribut scheme="schematron", la règle est nommée à l'aide de l'attribut *ident*.*
- La règle est contenue dans une balise <contraint>.

*NB : Attention à bien déclarer le nom de domaine *schematron* dans le préambule : xmlns:s="http://purl.oclc.org/dsdl/schematron"*

Exemple :

```
<constraintSpec ident="fromTo" scheme="schematron">
```

```
  <constraint>
    <s:rule context="tei:app[@type='structure']">
     <s:assert test="@from and @to">
     The beginning and the endpoint of the
       lemma have to be identifie/
      </s:assert>
    </s:rule>
  </constraint>
</constraintSpec>
```