

TEI à la carte (2)

Lou Burnard, MEET

2012-01



Roma: un petit tour

- 1 Sur l'écran *New*, choisir *Open existing customization* et retrouver le fichier `tei_cartes.odd` dans le dossier *Travaux*. Cliquer le bouton rouge **Start**.
- 2 Sur l'écran *Customize*, sélectionner le bouton **Francais** et cliquer encore le bouton rouge **Start**.
- 3 Sur l'écran *Personnaliser* sélectionner l'onglet *Langage*
- 4 Sur l'écran *Choisissons...* sélectionner de nouveau le *Francais* et cliquer sur le bouton rouge **Start**
- 5 Cliquer l'onglet *Schema* pour ouvrir l'écran *Creation du schema*: prendre le format par défaut. Cliquer sur le bouton rouge **Generate**. Enregistrer le fichier `tei_cartes.rnc` dans votre dossier *Travaux*.
- 6 Cliquer l'onglet *Documentation* pour ouvrir l'écran *Documentation*: prendre le format par défaut. Cliquer sur le bouton rouge **Generate**. Ouvrir le fichier `tei_cartes_doc.html` avec votre browser



Qu'est-ce qu'on vient de faire?

On a traité un fichier ODD déjà existant pour en generer un schema RELAXNG et sa documentation HTML.



Testons le schema

- 1 Dans Oxygen, créez un nouveau fichier XML
- 2 En bas de l'écran il y a un bouton Personnaliser. Cliquez-le.
- 3 A droite du champs URL du Schéma, il y a un petit icone de dossier: cliquez-le. Selectionner Parcourir les fichiers locaux. Naviguer jusqu'au fichier tei_cartes.rnc que vous venez de créer avec Roma
- 4 Cliquer sur le bouton Créer en bas pour créer un nouveau document TEI conforme au schema tei_cartes.



En route!

Oxygen vous propose un gabarit comme ceci:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <?xml-model href="file:/home/lou/Public/MEET/Talks/Travaux/tei_cartes.rnc"
2 type="application/relax-ng-compact-syntax" ?>
3 <TEI xmlns="http://www.tei-c.org/ns/1.0">
4   <teiHeader>
5     <fileDesc>
6       <titleStmt>
7         <title></title>
8       </titleStmt>
9       <publicationStmt></publicationStmt>
10      <sourceDesc></sourceDesc>
11    </fileDesc>
12  </teiHeader>
13 </TEI>
```

- A vous de le compléter (et de corriger les erreurs)!
- Tapez un < n'importe ou dans la fenêtre d'édition d'Oxygen pour voir quelles balises sont disponibles a cet endroit. Laissez-vous être guidé(e) par le logiciel!
- En vous servant de cette schéma, essayez de compléter le balisage des fichiers cartes-1.xml, cartes-2.xml, et cartes-3.xml

Exploration de TEI par Roma

Roma est également un outil apte à l'exploration des possibilités du système TEI.

- Sélectionner l'onglet [Modules] pour voir les modules disponibles ou déjà pris
- Sélectionner le nom d'un module pour voir les éléments qu'il contient, et pour en faire le choix.
- (On peut par défaut supprimer tous les éléments d'un module, ou bien en sélectionner tous)
- Sélectionner le nom d'un élément pour voir sa documentation complète



Autre possibilités de personnalisations

Par exemple, dans notre schema...

- on veut contraindre les valeurs légaux de l'attribut *@type* sur `<div>`
- on veut ajouter un nouveau élément `< saintName >` pour encoder les noms des saints (pourquoi pas?)
- on veut insister que chaque `<text>` contienne un `<div type="recto">` suivi d'un `<div type="verso">`



Modification des valeurs legales pour un attribut

- 1 En Roma, selectionner l'onglet Modules pour voir les modules present dans votre schema. Cliquer sur textstructure
- 2 Une liste des elements disponible dans ce module s'affiche. Retrouver la ligne concernant l'element `<div>` et cliquer sur le lien `Changer les attributs` a droite
- 3 Une liste des attributs disponibles sur cet element s'affiche. Retrouver l'attribut `@type` et cliquer sur son nom, a gauche.
- 4 Dans la formulaire qui s'ouvre, faire les modifications suivante:
 - selectionner 'no' pour la mention **Facultatif**
 - selectionner 'yes' pour la mention **Liste fermee**
 - Dans le champs **Liste des valeurs** taper
recto, verso, obliteration, message, destinataire
 - Cliquer sur le bouton rouge Save
- 5 Selectionner l'onglet Schema et regenerez votre schema



Qu'est ce qu'on vient de faire?

Notre fichier ODD contient maintenant :

```
<elementSpec ident="div" module="textstructure" mode="change">
  <attList>
    <attDef ident="type" mode="change" usage="req">
      <valList type="closed" mode="replace">
        <valItem ident="recto"/>
        <valItem ident="verso"/>
        <valItem ident="obliteration"/>
        <valItem ident="message"/>
        <valItem ident="destinataire"/>
      <!-- ... -->
    </valList>
  </attDef>
</attList>
</elementSpec>
```

Nota: on peut ajouter de documentation directement dans l'ODD:

```
<valItem ident="obliteration">
  <desc>contient
    le texte d'une obliteration quelconque sur une carte postale</desc>
</valItem>
```



Ajout d'un élément nouveau < saintName >

Il ressemble a quels autres éléments? C'est une espece de nom, évidemment

Il peut contenir quoi? Il peut contenir que de texte.

Ou est-ce qu'il peut apparaitre? A l'interieur des titres, des phrases, des paragraphes, etc. mais pas entre les deux.

Conclusion:

- on le fait membre de la classe `model.nameLike`
- son contenu est definie par `macro.xText`



Roma: Definition d'un nouveau élément



Roma: Production des validateurs TEI

Vous travaillez actuellement sur **TEI Absolutely Bare**

Création d'un élément

[Nouvelle](#) [Personnaliser](#) [Langage](#) [Modules](#) [Ajouter des éléments](#) [Modifier les classes](#) [Schéma](#) [Documentation](#) [Enregistrer](#) [Contrôleur de validité](#)

[go back to list](#)

Création d'un élément en cours:

Nom

Namespace

Description

Classes
structurales

- | | |
|--|--|
| <input type="checkbox"/> model.addrPart | <input type="checkbox"/> model.addressLike |
| <input type="checkbox"/> model.applicationLike | <input type="checkbox"/> model.biblLike |
| <input type="checkbox"/> model.biblPart | <input type="checkbox"/> model.castItemPart |
| <input type="checkbox"/> model.catDescPart | <input type="checkbox"/> model.choicePart |
| <input type="checkbox"/> model.common | <input type="checkbox"/> model.dateLike |
| <input type="checkbox"/> model.dimLike | <input type="checkbox"/> model.div1Like |
| <input type="checkbox"/> model.div2Like | <input type="checkbox"/> model.div3Like |
| <input type="checkbox"/> model.div4Like | <input type="checkbox"/> model.div5Like |
| <input type="checkbox"/> model.div6Like | <input type="checkbox"/> model.div7Like |
| <input type="checkbox"/> model.divBottom | <input type="checkbox"/> model.divBottomPart |



Definition de modele de contenu

- Le contenu d'un élément TEI est défini en langage RELAXNG
- La plupart des éléments TEI définissent leur contenu par référence aux *classes* d'élément plutôt que par référence à des éléments spécifiques.
- Il existe des modèles prédéfinis très utiles, par exemple:
 - `macro.paraContent` le contenu des éléments qui ressemblent aux paragraphes
 - `macro.phraseSeq` séquence de caractères mélangés d'éléments de la classe `model.phraseLike`
 - `macro.xText` séquence de caractères mélangés avec l'élément `<g>` (qui sert à encoder les caractères non-Unicode)



Définition d'un nouveau élément 2

The screenshot shows a web browser window with the address bar containing `http://tei.oucs.ox.ac.uk/Roma/startroma.php?mode=listAddedElem`. The browser's address bar also shows the Google logo and the text "Google". Below the address bar, there are several tabs, including "Problem loading page" and "Roma: Production des valida...". The main content area displays a list of XML attributes, each with a checkbox. The attributes are organized into two columns. The first column includes attributes like `att.damaged`, `att.dateable.iso`, `att.declarable`, `att.dimensions`, `att.docStatus`, `att.duration.iso`, `att.editLike`, and `att.entndLike`. The second column includes attributes like `att.dateable`, `att.dateable.w3c`, `att.declaring`, `att.divLike`, `att.duration`, `att.duration.w3c`, `att.enjamb`, `att.handFeatures`, `att.internetMedia`, `att.lexicographic`, `att.metrical`, `att.naming`, `att.placement`, `att.pointing.group`, `att.ranging`, `att.readFrom`, `att.scoping`, `att.sourced`, `att.tableDecoration`, `att.timed`, and `att.translatable`. A search or filter dropdown menu is open over the list, showing a scrollable list of attribute names. The attribute `macro.phraseSeq` is highlighted in orange. Below the list, there is a section labeled "Contenu" with a text area containing the XML code: `<content xmlns:rng="http://relaxng.org/ns/structure/1.0"></content>`. The browser's status bar at the bottom shows the text "tge" and "DONIS".

Qu'est-ce qu'on vient de faire?

On a ajouté une spécification pour notre nouvel élément:

```
<elementSpec
  ident="saintName"
  ns="http://www.example.org/ns/nonTEI"
  mode="add">
  <desc>contains the name of a saint.</desc>
  <classes>
    <memberOf key="model.nameLike"/>
    <memberOf key="att.typed"/>
  </classes>
  <content>
    <rng:ref name="macro.xtext"/>
  </content>
</elementSpec>
```

Nota: ce nouveau élément n'est *pas* un élément TEI! Il appartient donc à une autre espace de noms.



Autre contraintes

- On peut contraindre le contenu d'un élément, ou la valeur d'un attribut en se servant des *datatype* (par exemple, pour insister que le contenu de l'élément `<date>` soit vraiment une date)
- La TEI prédéfinit plusieurs *macros* pour répondre à cette possibilité. Par exemple
 - `data.word` a single word or token
 - `data.name` an XML Name
 - `data.enumerated` a single XML name taken from a documented list
 - `data.temporal.w3c` a W3C date
 - `data.truthValue` a truth value (true/false)
 - `data.language` a human language
 - `data.sex` human or animal sex
- Ou on peut créer ses propres contraintes avec le langage Schematron



Contraintes Schematron

- Une spécification d'élément peut contenir un élément `<constraintSpec>` contenant des règles exprimés en langage ISO Schematron

```
<elementSpec ident="body" module="teisttructure" mode="change"
  xmlns:s="http://purl.oclc.org/dsdl/schematron">
  <constraintSpec mode="add" ident="bodyLim" scheme="isoschematron">
    <constraint>
      <s:assert
        test="tei:div/@type='recto' and tei:div/@type='verso' and
count(tei:div)=2">le body doit contenir un recto et un verso et rien de plus
      </s:assert>
    </constraint>
  </constraintSpec>
</elementSpec>
```

A noter...

- Pour ajouter de telles règles, il faut éditer le fichier ODD. Roma ne vous aide pas.
 - L'implantation de telles règles n'est pas forcément disponible
- avec tout processeur XML.