Projet GL – Document de conception

# Modèles

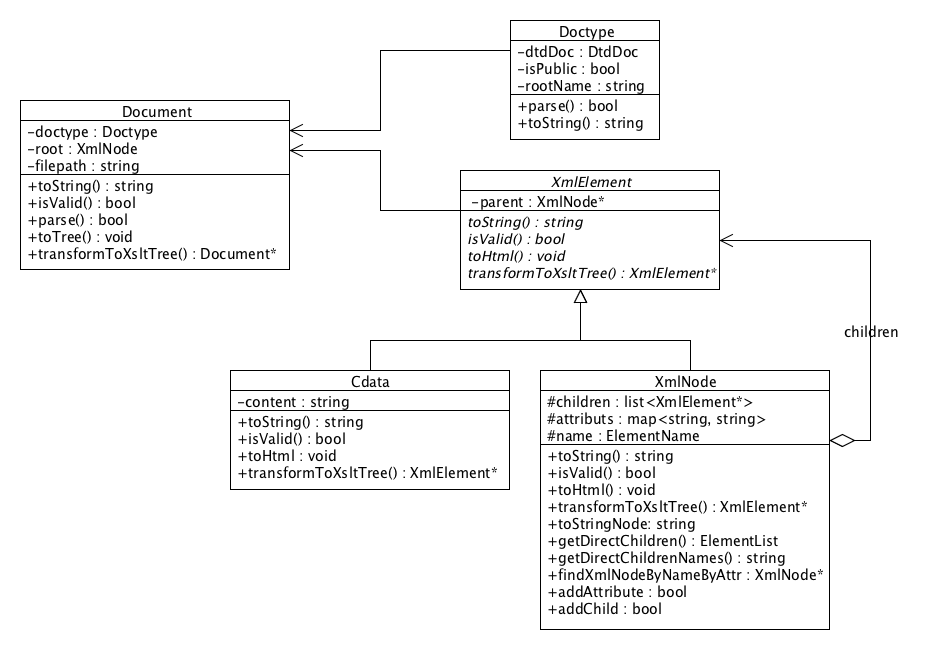
## Document XML

La représentation en mémoire d’un document XML fait appel à 5 classes :

* Document, 1er objet créé, représentant le fichier parsé
* Doctype, 1 doctype par document, qui va contenir la représentation mémoire de la DTD
* XmlElement, classe abstraite, représentant un élément XML, deux formes sont possibles :
  + XmlNode, nœud qui a éventuellement des attributs et/ou contient d’autres éléments
  + Cdata, character data, une chaîne de caractère non interprétés

Afin d’optimiser notre code, nous avons donc utilisé le design-pattern du composite.

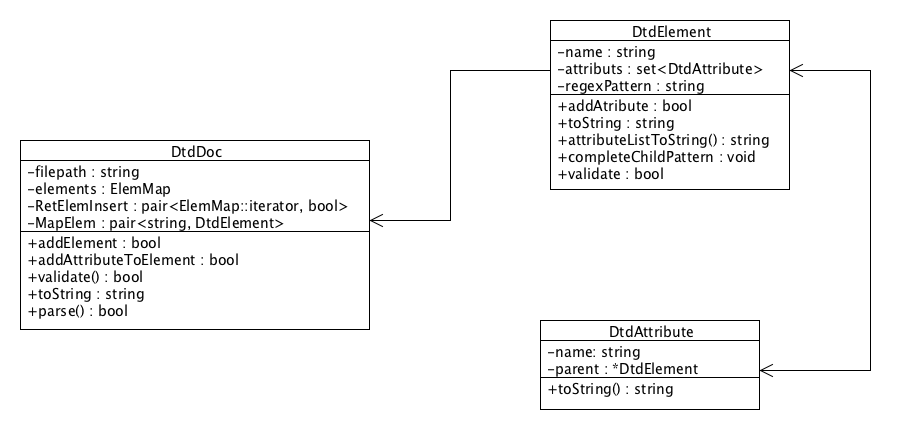
Il est possible de représenter nos classes ainsi :



## Document DTD

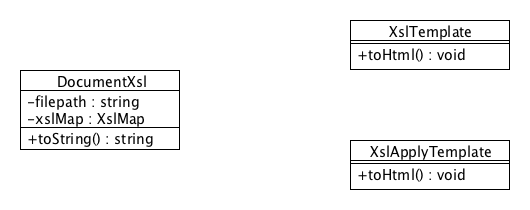
La représentation d’un document DTD est beaucoup plus simple. Nous avons ainsi :

* un objet DtdDoc représentant le document, racine, qui est pointé par l’attribut dtdDoc du Doctype du document XML. Il contient une liste de DtdElement.
* des éléments DtdElement, représentant des éléments ( !ELEMENT ), qui ont des liste d’attributs et ont leur contenu stocké sous forme d’une regex
* des éléments DtdAttribute, qui ont un nom, représentant les attributs ( !ATTLIST )



## Document XSLT

La représentation d’une telle document est plus simple : nous avons un élément « DocumentXsl », correspondant au fichier, et les différents templates, qui peuvent être appliqués par une méthode toHtml().



# Algorithme de validation

Cet algorithme est utilisé pour valider le fichier XML par rapport à sa DTD.

Le parse du fichier XML est lancé, si le fichier est bien formé (par d’erreur de caractère, tag mal fermé…), il est représenté en mémoire. Le doctype sera détecté, et la DTD correspondante parsée.

La fonction document->isValid() est ensuite lancée manuellement.

**Fonction Document->isValid() :**

// on vérifie que le 1er nœud est bien du type défini par le Doctype

Si document->root.getName() != doctype.getRootName()

retourne FAUX

FinSi

document->root->isValid() // on s’appelle sur le XmlElement racine

**Fonction XmlElement->isValid() :**

*Si s’agit d’un XmlNode :*

// fonction validate(attMap) : on vérifie si les attributs son autorisés

Pour (unAttribut dans attMap)

Si unAttribut introuvable dans DtdElement->attributs

retourne FAUX

FinSi

FinPour

// fonction validate(getDirectChildren()) : on vérifie si les éléments-fils sont autorisés

// on compare via Regex si le nom des éléments-fils est dans la liste (enum ou choice) du DTD

Pour (XmlElement élément dans children)

Si élément->isValid() == false // appel récursif

retourne FAUX

FinSi

FinPour

retourne VRAI

*S’il s’agit d’un Cdata :*

retourne VRAI // character data, si bien délimité forcément valide

# Algorithme de transformation

Manon et Nico !