Université Pierre et Marie Curie

Modelisation Objet

TME 6 – Parser xml

Auteur:
William Fabre

Professeur:
Madame Marie-Minerve Louerat

Année 2018-2019

Contents

1 Introduction **2**

2 Question 1 Implanter Term* Term::fromXml(Cell*, xmlTextReaderPtr) **3**

3 Question 2 Implanter Instance* Instance::fromXml(Cell*, xmlTextReaderPtr) **4**

4 Question 3 Implanter bool Node::fromXml(Net*,xmlTextReaderPtr) **5**

5 Question 4 Net* Instance::fromXml(Cell*, xmlTextReaderPtr) **6**

6 Question 5 Test en boucle **7**

Introduction

Tout d'abord je vais decrire la metodologie que j'ai utilisee pour

toutes les fonctions du parser de Term, instance, Node, Net. Puis nous verrons les details de l'implementation de chacun des parsers, j'aurais pu faire une fonction qui est appellee par tous les parser avec un variadics car la seule chose qui change est le nombre de variable que l'on veut transformer depuis des attributs xml vers des strings. Un gros changement s'opere pour la fonction Net qui va aller tous les nodes qui lui sont lies mais le parsing d'argument reste le meme. Pour parser il faut faire les etapes suivantes:

1. caster la string qui designe l'attribut que l'on cherche a read en un const xmlChar*.
2. Utiliser XmlTextReaderGetAttribute avec le reader et le nom de l'attribut.
3. Transformer ce que l'on a lu en string.
4. Verifier que notre string n'est pas vide.
5. Nous pouvons maintenant utiliser nos attributs.

Question 1 Implanter Term* Term::fromXml(Cell*, xmlTextReaderPtr)

On essaye de lire ici : Un name, une direction, un x et un y. Si tout est lu correctement, il faut maitnenant creer le nouveau term on possede la celle comme arguemnt de fonction. On utilise toDirection pour creer une direction a partir d'une string et on peut aussi set la position avec le x et y en utilisant atoi.

Question 2 Implanter Instance* Instance::fromXml(Cell*, xmlTextReaderPtr)

On essaye de lire ici : Un name, une mastercell, un x et un y. Si tout est lu correctement, il faut maitnenant creer la nouvelle instance. On va quand meme verifier que la mastercell existe avec Cell::find(string) qui nous permet de lui passer la string. On peut desormais creer une instance a partir de la cell en arguemnt, la mastercell trouvee et le nom de cette instance. On finit par set la position.

Question 3 Implanter bool Node::fromXml(Net*,xmlTextReaderPtr)

On essaye de lire ici : Un term, une isntance si elle existe et un id. Si le term est vide ou si l'id est vide il y a un probleme alors on sort immediatement. On fait le traitement si on est une instance ou si on est une celle en verifiant tout simplement si le champ instance a ete remplie.

On peut dans le cas d'une instance recuperer a partir du net la cell correspondante et recuperer l'instance correspondante. On va ensuite connecter pour cette isntance le nom du term avec le net en arguement.

On peut dans le cas d'une cell recuperer la cell avec getcell sur le net en argument puis connecter pour la celle le term au net.

On finit bien sur par set l'id dans les deux cas avec un setId sur un getnode sur un getterm.

Question 4 Net* Instance::fromXml(Cell*, xmlTextReaderPtr)

La question n'etait pas precisee mais on peut voir

if (Net::fromXml(cell,reader)) continue;

dans le code fournis. Le probleme ici est non seulement de lire les attributs mais aussi de parcourir la sous liste de node, j'ai donc pris exemple sur le parser dans Cell et refait un parcours.

les elements du Net sont un name et un type. On les check et s'ils sont vides on sort directement. Sinon on commence a parser les nodes.

On parse des node en fonction d'un nodeTag "node", on recupere le localName et on sortira si on est a la fin du reader ou si si le nodetag n'est plus le nodename. On verifie le status du reader au cas ou il y a une erreur. On ne prend pas compte des commentaires, espaces ou tabulations. Ensuite on lance la lecture de Node::fromXml.

lorsque tout est fini on retourne le net.

Question 5 Test en boucle

Pour le teste en boucle, l'indentation fournie n'est pas exactement respectee. C'est a dire qu'il existe des espaces non previsible ou difficilement previsibles. Par exemple prendre en compte que si on a

```
<instance name="xor2_1 mastercell="xor2" x="150" y="200"/>
<instance name="and2_1 mastercell="and2" x="150" y="50"/>
```

Et bien il existe en fait un espace apres la guillement du y="50" pour aligner avec le y="200". Ceci n'a pas ete pris en compte.

Pour le parser il a donc fallut sauvegarder via la fonction save ce que j'avais lu et lorsque j'ai compris que seuls ces espaces n'etaient pas bon alors j'ai fait 3 scripts qui permettent de sauvegarder en boucle et de comparer.

Il faut un petit peu de configuration : Tout d'abord mettre le dossier cells dans build, puis mettre le script make.sh dans build et mettre le script compare.sh et check.sh dans le dossier cells.

compare.sh fait la comparaison entre les deux fichiers en faisant un tail -n +1 car la premiere ligne differe forcement, les deux fichiers n'ont pas le meme nom. ce script prend en argument le nom des deux fichiers a comparer et va nous ecrire si les fichiers sont egaux ou pas.

check.sh fait l'appelle a compare.sh pour les 8 fichiers a comparer, ecrit le nouveau fichier dans l'ancien fichier et detruit les fichiers _test qui etaient generer par la sauvegarde.

make.sh Va faire l'appelle au tme6 et lancer le script check.sh. Tout ca dans une boucle de 10 iterations.

```
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
```

```
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
```

```
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
Saving <Cell and2> in <./cells/and2_test.xml>
Saving <Cell or2> in <./cells/or2_test.xml>
Saving <Cell xor2> in <./cells/xor2_test.xml>
Saving <Cell halfadder> in <./cells/halfadder_test.xml>
comparing xxx_test.xml and xxx.xml
halfadder.xml == halfadder_test.xml files are the same
or2.xml == or2_test.xml files are the same
xor2.xml == xor2_test.xml files are the same
and2.xml == and2_test.xml files are the same
rewriting xxx_test.xml into the xxx.xml
supressing files xxx_test.xml
```