Examen Web/XML

Durée : 2 heures, Documents autorisés

Programmation Java avec le DOM (3 points)

Vous êtes le meilleur programmeur d'une entreprise qui vend des outils XML en Java.
Le projet actuel consiste à réaliser un moteur XPath qui utilise le DOM comme base de navigation.
Ecrivez un programme qui puisse parcourir l'axe `preceding-sibling`
Vous ne vous préoccuperez pas de l'analyse de l'expression XPath, mais seulement de son exécution.
Vous pourrez faire une classe abstraite qui serve de chapeau à tous les axes, et une autre qui se préoccupe de l'axe `preceding-sibling`
Faites un jeu de données XML, d'expression XPath à tester, et indiquez ce que vous obtenez avec votre programme.
Voir en annexes un extrait de l'API DOM.

Réponse :

On peut créer des classes basées sur les composantes d'une expression ; lorsqu'un axe retourne plusieurs items, on peut soit retourner un ensemble, soit un itérateur qui maintienne lui-même l'état de parcours de l'axe. Choisissons un itérateur :

```
package fr.unice.minfo.xpath;
public abstract class IterableAxis {
    protected NodeTest test;
    public IterableAxis( NodeTest test ) {
        this.test = test ;
    }
    public NodeIterator cross( Node context );
}
```

…qu'il faut aussi implémenter, sur le modèle de `java.util.Iterator` *(on pourrait d'ailleurs se demander s'il est utile d'implémenter cette interface…) :*

```
package fr.unice.minfo.xpath;
public abstract class NodeIterator {
    protected Node current;
    protected NodeTest test ;
    public NodeIterator ( Node context, NodeTest test ) {
        this.test = test ;
        this.current = getFirst( context );
    }
    public boolean hasNext() {
        return (current != null);
    }
    public abstract Node next();
    protected abstract Node getFirst( Node context );
}
```

La classe `NodeTest` *consite à réaliser un* ***test de nœud*** *sur le type de nœud à conserver (un élement quelconque ou particulier, un nœud de texte, de commentaire, etc). Notons que certains nœuds rencontrés (DocumentFragment, DocumentType, Entity, EntityReference, et Notation) devraient être systématiquement ignorés dans toutes les implémentations, puisque le modèle de données de XPath n'est pas totalement compatible avec celui du DOM.*
Le test de nœud fait partie du constructeur de l'itérateur puisqu'il est connu au moment de l'analyse de l'expression XPath (on ne demande donc pas le détail de son implémentation).

```
package fr.unice.minfo.xpath;
public abstract class NodeTest {
    public abstract boolean isKept( Node node );
}
```

Examen Web/XML

Durée : 2 heures, Documents autorisés

L'axe demandé se résume à ceci :

```
package fr.unice.minfo.xpath;
public class PrecedingSiblingAxis extends IterableAxis {
    public PrecedingSiblingAxis( NodeTest test ) {
        super( test );
    }
    public NodeIterator cross( Node context ) {
        return new NodeIterator( context, test ) {
            public Node next() {
                Node next = current;
                current = getNext( current ) ;
                return next;
            }
            protected Node getFirst( Node context ) {
                return getNext( context );
            }
            private getNext( Node node ) {
                do {
                    Node next = node.getPreviousSibling();
                } while ( next != null && ! test.isKept( next ) );
                return next ;
            }
        }
    }
}
```

On ne peut appeller `next()` *que si* `hasNext()` *a été appelé, sinon une exception peut être lancée.*

On dispose alors d'une bonne base pour implémenter un moteur XPath. Bien entendu, le contexte d'exécution est réduit ici à un simple `Node`*, alors qu'en réalité –selon la spécification– il devrait supporter la résolution des préfixes, des variables, des fonctions, etc…*

Test du programme :

Prenons en entrée le fichier XML de l'exercice suivant et prenons comme contexte l'élément data qui contient le texte « jkl », et noté ci-dessous `dataJkl`*.*

Avec une expression comme `preceding-sibling::data` *les nœuds autres que les nœuds d'éléments data seront ignorés par* `isKept()`*.*

L'objet `NodeIterator` *obtenu aurait comme variable interne*
`current=dataJkl.getPreviousSibling()=dataGhi`

Après appel de `next()` *, la valeur* `dataGhi` *est retournée, puis la variable interne devient :*
`current=dataGhi.getPreviousSibling()=dataDef`

Après appel de `next()` *, la valeur* `dataDef` *est retournée, puis la variable interne devient :*
`current=dataDef.getPreviousSibling()=dataAbc`

Après appel de `next()` *, la valeur* `dataAbc` *est retournée, puis la variable interne devient :*
`current=dataGhi.getPreviousSibling()=null`

Les 3 nœuds retournés sont bien ceux attendus.

Examen Web/XML

Durée : 2 heures, Documents autorisés

Programmation XSLT (3 points)

Vous êtes le meilleur programmeur d'une entreprise qui vend des prestations XML.
Le client du moment vous demande de faire quelques feuilles de style XSLT.

1 – Mettre des données dans un tableau :

Fichier de test en entrée :

```
<?xml version="1.0"?>
<root>
  <data>abc</data>
  <data>def</data>
  <data>ghi</data>
  <data>jkl</data>
  <data>mno</data>
  <data>pqr</data>
  <data>stu</data>
  <data>vwx</data>
  <data>yz</data>
</root>
```

Sortie souhaitée en HTML :

abc	def	ghi
jkl	mno	pqr
stu	vwx	yz

Réponse :

On peut utiliser un paramètre pour le nombre de colonnes…

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="columnsCount" select="number(3)"/>
<xsl:template match="/">
  <html>
    <body>
      <table border="1">
        <xsl:for-each
select="root/data[(position() mod $columnsCount) = 1]"
        >
          <tr>
            <xsl:for-each
select=".|following-sibling::data[position() &lt; $columnsCount]"
            >
              <td>
                <xsl:value-of select="."/>
              </td>
            </xsl:for-each>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

2 – Changer d'espace de nommage :

Vous disposez de `copy.xslt` qui permet de faire une copie :

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="node(  ) | @*">
  <xsl:copy>
    <xsl:apply-templates select="@* | node(  )"/>
  </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

Fichier en entrée :

Examen Web/XML

Durée : 2 heures, Documents autorisés

```
<foo:someElement xmlns:foo="http://www.foo.com/namespaces/foo"
    xmlns:doc="http://www.doc.com/namespaces/doc">
    <foo:aChild>
        <foo:aGrandChild/>
        <foo:aGrandChild>
            <doc:doc>
                This documentation should not be removed
                or altered in any way.
            </doc:doc>
        </foo:aGrandChild>
    </foo:aChild>
</foo:someElement>
```

Sortie souhaitée en XML (les blancs peuvent être ignorés, seule la structure importe) :

```
<?xml version="1.0"?>
<bar:someElement
    xmlns:bar="http://www.bar.com/namespaces/bar">
    <bar:aChild>
        <bar:aGrandChild>
        </bar:aGrandChild>
        <bar:aGrandChild>
            <doc:doc xmlns:doc="http://www.doc.com/namespaces/doc">
                This documentation should not be removed
                or altered in any way.
            </doc:doc>
        </bar:aGrandChild>
    </bar:aChild>
</bar:someElement>
```

Réponse :

Il ne fallait pas utiliser `<xsl:namespace-alias>` *qui ne sert qu'à renommer des préfixes utilisés dans la feuille de style. Il suffit de laisser la feuille de style importée faire tout le travail de copie, et juste réagir sur les éléments préfixés.*

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:foo="http://www.foo.com/namespaces/foo"
    xmlns:bar="http://www.bar.com/namespaces/bar">

<xsl:import href="copy.xsl"/>
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
<xsl:strip-space elements="*"/>

<xsl:template match="foo:*">
    <xsl:element name="bar:{local-name(  )}">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

</xsl:stylesheet>
```

Examen Web/XML

Durée : 2 heures, Documents autorisés

XPath (0,5 point)

Vous êtes toujours le meilleur programmeur d'une entreprise qui vend des prestations XML.
Le client du moment vous demande de lui concocter une expression XPath qui sélectionne les catégories qui n'apparaissent qu'une fois :

```
<?xml version="1.0"?>
<dataset>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C2</category>
  </node>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C3</category>
  </node>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C2</category>
  </node>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C3</category>
  </node>
</dataset>
```

Réponse :

`//category[not(. = preceding::category)]` *retourne les 3 catégories demandées (en gras ci-dessus).*
A ne pas confondre avec `//category[. != preceding::category]` *qui retourne toutes les catégories sauf la première (voir le cours sur XPath, page 33).*

Problème (3,5 points)

Vous êtes le meilleur concepteur d'une entreprise qui vend des services et autres prestations du moment que le client paie.
Votre projet consiste à gérer le parc de machines de votre client qui exige que ses données soient en XML parce qu'il en a entendu parler mais ne sait pas trop ce que c'est.
1 – Identifiez ce que pourraient être les données à gérer, et faites une instance XML.
2 – Créez une DTD de sorte que votre instance soit valide vis à vis de la DTD.
3 – Faites une feuille de style XSLT pour visionner vos données en HTML.
4 – Dessinez le rendu HTML comme l'aurait fait un navigateur.

Réponse :

Trop facile pour mériter un corrigé…
☺