Procesamiento de Formatos en Aplicaciones Telemáticas - Examen convocatoria ordinaria, curso 2015/16
Grado en Ingeniería Telemática
Depto. de Ingeniería Telemática
Universidad Carlos III de Madrid

Duración: 1 hora 15 minutos
Puntuación: 30 puntos
Nota: Se pueden usar libros y apuntes

En los dos ejercicios se utilizará el siguiente DTD:

```
<!ELEMENT matrixE (filaE+)>
<!ATTLIST matrixE
totalCeldas CDATA #IMPLIED>


<!ELEMENT filaE (celdaE+)>
<!ATTLIST filaE
celdas CDATA #IMPLIED>

<!ELEMENT celdaE (#PCDATA)>
```

Un ejemplo de documento XML válido de acuerdo con este DTD sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE matrixE SYSTEM "matrix.dtd">

<matrixE>
<filaE><celdaE>1</celdaE><celdaE>0</celdaE><celdaE>2</celdaE></filaE>
<filaE><celdaE>3</celdaE><celdaE>1</celdaE><celdaE>5</celdaE></filaE>
<filaE><celdaE>0</celdaE><celdaE>1</celdaE><celdaE>2</celdaE></filaE>
</matrixE>
```

EJERCICIO 1 (15 puntos)

Se pide implementar un método en Java cuyo prototipo es:

```
String ordinaria(Document doc)
```

que dado un objeto `Document doc` que representa un documento XML válido de acuerdo con el DTD definido anteriormente devuelva el contenido del primer hijo `celdaE` del primer hijo `filaE` del elemento raiz del documento XML dado.

Por ejemplo, suponiendo que `doc` representase el documento XML dado al inicio del enunciado, `ordinaria(doc)` devolvería `"1"`.

Se puede suponer que los elementos `celdaE` contienen un único nodo hijo de tipo `Text`.

Solución:

```
String ordinaria(Document doc) {
    Element docEl;
    NodeList nl1, nl2, nl3;
    int i, j, k, len1, len2, len3;
    Node n1, n2, n3;
    docEl= doc.getDocumentElement();
```

```
        nl1= docEl.getChildNodes();
        len1=nl1.getLength();
        for (i=0; i<len1; i++) {
            n1= nl1.item(i);
            if (n1.getNodeType()==Node.ELEMENT_NODE) {
                nl2=n1.getChildNodes();
                len2=nl2.getLength();
                for (j=0; j<len2; j++) {
                    n2=nl2.item(j);
                    if (n2.getNodeType()==Node.ELEMENT_NODE) {
                        nl3=n2.getChildNodes();
                        len3=nl3.getLength();
                        for (k=0; k<len3; k++) {
                            n3=nl3.item(j);
                            if (n3.getNodeType()==Node.TEXT_NODE ||
                                n3.getNodeType()==Node.CDATA_SECTION_NODE)
                                return n3.getNodeValue();
                        }
                        return "";
                    }
                }
                return "";
            }
        }
        return "";
    }
```

EJERCICIO 2 (15 puntos)

Escriba una hoja de estilo XSLT que dado un documento XML válido de acuerdo con el DTD dado al inicio del enunciado devuelva un documento XML válido para el mismo DTD igual al documento original salvo por lo siguiente:

- Se añade al elemento `matrixE` un atributo `totalCeldas` cuyo valor es el número total de elementos `celdaE` que contiene **el documento original**.
- Se añade a cada elemento `filaE` un atributo `celdas` cuyo valor es el número total de elementos `celdaE` que contiene dicho elemento `filaE` en **el documento original**.
- Se copian al documento destino solamente los siguientes elementos `celdaE`: el primer elemento `celdaE` de cada elemento `filaE` así como aquellos otros elementos `celdaE` tales que su valor sea estrictamente mayor que el de al menos uno de los elementos `celdaE` que le preceden en la mismna fila.

Por ejemplo, para el documento XML proporcionado al principio del enunciado, el resultado de ejecutar la hoja de estilo pedida sería:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE matrixE SYSTEM "matrix.dtd">
<matrixE totalCeldas="9">
<filaE celdas="3">
  <celdaE>1</celdaE><celdaE>2</celdaE>
</filaE>
<filaE celdas="3">
  <celdaE>3</celdaE><celdaE>5</celdaE>
</filaE>
<filaE celdas="3">
  <celdaE>0</celdaE><celdaE>1</celdaE><celdaE>2</celdaE>
</filaE>
</matrixE>
```

No se preocupe por como queda indentado el documento XML resultante.

Solución:

```
<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

<xsl:output method="xml" encoding="ISO-8859-1"
  doctype-system="matrix.dtd" indent="yes"/>

<xsl:strip-space elements="*" />

<xsl:template match="*">
<xsl:copy>
<xsl:apply-templates/>
</xsl:copy>
</xsl:template>

<xsl:template match="matrixE">
<matrixE totalCeldas="{count(descendant::celdaE)}">
<xsl:apply-templates/>
</matrixE>
</xsl:template>


<xsl:template match="filaE">
<filaE celdas="{count(celdaE)}">
  <xsl:apply-templates
    select="celdaE[position()=1]|celdaE[.>preceding-sibling::*]"/>
</filaE>
</xsl:template>

</xsl:stylesheet>
```

Solución mejorada

```
public static String ordinaria(Document doc) {
    try {
        return ((Element) doc.getElementsByTagName("filaE").item(0)).getElementsByTagName("celdaE").item(0).getFirstChild().getNodeValue();
    } catch (Exception e) {
        return "";
    }
}
```