

Semantic Web

XSLT, XSL-FO, and XQuery

F. Abel, D. Krause

IVS Semantic Web Group

06.11.2007

Exercise 1: XSLT

Apply the given XSLT document (*album.xsl*) to the given XML document (*tracks.xml*) and write down the resulting XML document.

XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl"
  href="album.xsl"?>
<album name="My Funny Valentine"
  artist="Chet Baker"
  website="http://chetbaker.com">
  <track number="2">
    <name>Someone To Watch Over Me</name>
    <duration>00:02:58</duration>
    <year>1974</year>
  </track>
  <track number="3">
    <name>Moonlight Becomes You</name>
    <duration>00:03:51</duration>
    <year>1974</year>
  </track>
  <track number="1">
    <name>My Funny Valentine</name>
    <duration>00:03:20</duration>
    <year>1974</year>
  </track>
</album>
```

XSLT file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:template match="album">
  <html>
  <h1> <xsl:value-of select="@name"/> </h1>
  Artist: <a href="{@website}">
    <xsl:value-of select="@artist"/>
  </a>
  <table>
    <xsl:for-each select="track">
      <xsl:sort data-type="number" select="@number"/>
      <tr>
        <xsl:for-each select="name">
          <td><xsl:apply-templates/></td>
        </xsl:for-each>
        <xsl:for-each select="duration">
          <td><xsl:apply-templates/></td>
        </xsl:for-each>
      </tr>
    </xsl:for-each>
  </table>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Resulting XML document:

```
<html>
<h1>My funny Valintine </h1>
  Artist: <a href="http://chetbaker.com">Chet Baker</a>
  <table>
    <tr>
      <td>My Funny Valentine</td> <td>(00:03:20)</td>
    </tr>
    <tr>
      <td>Someone To Watch Over Me</td> <td>(00:02:58)</td>
    </tr>
    <tr>
      <td>Moonlight Becomes You</td> <td>(00:03:51)</td>
    </tr>
  </table>
</html>
</html>
```

Exercise 2: XSL (XSLT + XSL-FO) [Homework]

The given XML document (*personen.xml*) should be transformed to a PDF document. The PDF document has to fulfill the following requirements:

- the *name* of a *group* should be centered and should have a larger text-size than other elements
- members of a group should be ordered according to their age, and should be presented as list
- for each *person* the PDF should show the name (*bolded*), the age, and a link that allows to send him an email (`mail`).
- each group should be visualized on a separate page
- the respective page number should be embedded in the footer of each page

Write an adequate XSL Stylesheet that transforms the given XML document into an XSL-FO document, which can be used to produce the required PDF document!

Exercise 2: XSL (XSLT + XSL-FO) - person.xml (1)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<groups>
  <group id="developerTeam">
    <name>Java Developer Team</name>
    <member>
      <person id="peter">
        <name>Peter Mueller</name>
        <age>24</age>
        <homepage>http://www.java.de/peter/</homepage>
        <workplaceHomepage>http://www.java.de/</workplaceHomepage>
        <workplaceHomepage>http://www.sun.com/</workplaceHomepage>
        <emailAddress>peter@sun.com</emailAddress>
        <knows friends="jan"/>
      </person>
      <person id="frank">
        <name>Frank Schmidt</name>
        <age>31</age>
        <workplaceHomepage>http://www.java.de/</workplaceHomepage>
        <emailAddress>frank@java.de</emailAddress>
        <knows friends="peter"/>
      </person>
    </member>
  </group>
  ...

```

Exercise 2: XSL (XSLT + XSL-FO) - person.xml (2)

```
...
<group id="managerTeam">
  <name>Project Manager Team</name>
  <member>
    <person id="jan">
      <name>Jan Hansen</name>
      <age>28</age>
      <workplaceHomepage>http://www.java.de/</workplaceHomepage>
      <emailAddress>jan@java.de</emailAddress>
      <emailAddress>jan@sun.com</emailAddress>
      <knows friends="frank"/>
    </person>
  </member>
</group>
</groups>
```

Exercise 2: XSL (XSLT + XSL-FO) [solution]

No standard solution. Try yourself and use the following Java code to test your solution:

<http://www.personal-reader.de/semweb08/slides/lecture05-examples.zip>

Construct XQuery expressions for the following requests based on the XML document from Exercise 2 (*personen.xml*):

- (a) Return all names of persons that are older than 25 years (age > 25). Order the result according to the age of the persons (*descending*).

Solution:

```
for $p in doc("personen.xml")//person
where $p/age > 25
order by $p/age descending
return $p/name
```

- (b) Return the average age of each group. The result should be formatted as follows:

```
<group>
  <name> ... </name>
  <alter> ... </alter>
</group>
```

Solution:

```
for $g in doc("personen.xml")//group
let $a := avg($g/member/person/age)
order by $a descending
return
<group>
  <name> {$g/name/text()} </name>
  <alter> {$a} </alter>
</group>
```

- (c) Return the *emailAddress* of those persons that have ‘‘http://www.java.de/’’ as *workplaceHomepage*.

Solution:

```
for $p in doc("personen.xml")//person
where some $wh in $p/workplaceHomepage satisfies
    contains($wh,"http://www.java.de/")
return $p/emailAddress
```

Exercise 3: XQuery [solution]

- (d) Return either the *emailAddress*, or the *name* of a person:
- if *age* < 30 then the *emailAddress* should be returned.
 - if the person is not younger than 30 then the *name* should be returned.

Format the output as list:

```
<ul><li> mail/name : ... </li> ... </ul>
```

Solution:

```
<ul>{  
for $p in doc("personen.xml")//person  
return  
  <li>  
    {  
      if ($p/age < 30) then  
        concat("mail: ", $p/emailAddress[1]/text())  
      else  
        concat("name: ", $p/name/text())  
    }  
  </li>  
}  
</ul>
```

Exercise 3: XQuery [solution]

- (e) The following XML document (*addressbook.xml*) is given in addition to *personen.xml*. Formulate a query that returns the phone number and the name (*not id!*) of persons whose phone number starts with '0511'.

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
  <contact id="frank" phone="040-111"/>
  <contact id="jan" phone="0511-222"/>
  <contact id="peter" phone="0511-333"/>
</contacts>
```

Solution:

```
<hannoveraner>
{
for $p in doc("personen.xml")//person
  for $c in doc("addressbook.xml")//contact[@id = $p/@id]
    where starts-with($c/@phone, "0511")
      order by $p/name
      return
        <contact name="{ $p/name/text() }"
                  phone="{ $c/@phone }"/>
}
</hannoveraner>
```

Exercise 3: XQuery *[solution]*

All XQuery expressions have been tested with the given XML documents (personen.xml and addressbook.xml) using Oxygen Editor:

<http://www.oxygenxml.com>

Solution can also be downloaded via:

<http://www.personal-reader.de/semweb08/slides/xquery.zip>