

NPRG036, A7B36XML: **XML Technologies**

Practical Class 5:

XSLT: Exercises

28. 4. 2016

Martin Svoboda

Jiří Helmich

<http://www.ksi.mff.cuni.cz/~svoboda/courses/2015-2-NPRG036/>

<http://www.ksi.mff.cuni.cz/~svoboda/courses/2015-2-A7B36XML/>

Exercise 1

- Create an empty XSLT script for the provided real estate XML document
 - Use HTML as the output format
 - Check the behavior of implicit templates

Exercise 2

- Extend the previous XSLT script
 - Generate basic structure of the output web page

```
<html>
  <head>
    <title>Real Estate Report</title>
  </head>
  <body>
    ...
  </body>
</html>
```

- Suppress any other output (if necessary)

Exercise 3



- Extend the previous XSLT script
 - Create a table of properties and add it into the output

```
<h2>Table of Properties</h2>
<table>
  <tr>
    <th>Property Id</th>
    <th>Property Name</th>
    <th>Owner Id</th>
  </tr>
  ...
  <tr><td>ID</td><td>NAME</td><td>OWNER</td></tr>
  ...
</table>
```

- Generate rows of this table (one for each property)
- Create only one new unnamed template,
 - use *xsl:apply-templates*, do not use *xsl:value-of*

Exercise 4



- Extend the previous XSLT script

- Create an ordered list of flats

```
<h2>List of Flats</h2>
<ol>
  <li id="flat-ID">Flat ID: NAME</li>
  ...
</ol>
```

- Replace ID and NAME with corresponding values

- Create an unnamed template for *flat* elements

- Do not create any other templates

- Use computed constructors only

- I.e. *xsl:element*, *xsl:attribute*, *xsl:text*

Exercise 5



- Extend the previous XSLT script
 - Add a new column into the table of properties
 - For each property, generate an unordered list of references to its flats

```
...
<td>
  <ul>
    <li><a href="#flat-ID">NAME</a></li>
    ...
  </ul>
</td>
...
```

- Use unnamed templates together with modes

Exercise 6

- Extend the previous XSLT script
 - Integrate the following CSS into the *head* element

```
<style>
  th { background-color: #AAAAAA; }
  tr.even td { background-color: #DDDDDD; }
</style>
```

- Modify the table of properties
 - Add a new column with document order numbers of properties

```
<tr><td>1</td>...</tr>
<tr><td>2</td>...</tr>
...
```

- Set the background color of all even rows

```
<tr class="even">...</tr>
```

- Use *xsl:if* conditional instructions

Exercise 7



- Extend the previous XSLT script
 - Modify the table of properties
 - Instead of printing an optional owner identifier, fetch the corresponding owner name
 - When this owner reference or owner or their name is missing, print `<i>Unknown</i>` instead
 - Create and call a named template for this purpose
 - Use a parameter to specify the input owner identifier
 - When called without this parameter, fetch the owner identifier nevertheless from the given property data

Exercise 8



- Extend the previous XSLT script
 - Add a new column with a comma separated list of property features into the table of properties

```
<tr>
  ...
  <td>alarm, doorkeeper, street view, subway</td>
</tr>
```

- Order this list alphabetically
- Do not create new templates, i.e. use *xsl:for-each*

Exercise 9



- Extend the previous XSLT script
 - Modify the way how property features are printed
 - Print all features belonging to and only to category *location* in blue,

```
<span style="color: blue;">...</span>
```
 - print all other features from category *security* in red, and
 - preserve the default color otherwise
 - Use *xsl:choose* instruction for this purpose
 - Do not forget that each feature may contain more particular categories listed in its *categories* attribute
 - I.e. use *contains(haystack, needle)* function

Exercise 10



- Extend the previous XSLT script
 - Switch XSLT version from 1.0 to 2.0
 - Add a new unordered list of flat features

```
<h2>List of Features of Flats</h2>
<ul>
  <li><b>FEATURE</b>: FLATNAME@PROPERTYID, ...</li>
  ...
</ul>
```

- Use *xsl:for-each-group* for the aggregation
- Exploit *string-join(sequence, separator)* function and XPath 2.0 extended expressivity of the / operator to compose individual lists of *flat@property* items

Exercise 11



- Extend the previous XSLT script
 - Export XML data of each property to a separate output file with name *11-property-ID.xml*
 - I.e. always copy the entire content of a given *property* element