Seminar Paper

Interactive Cross-Browser XSLT-Support for an Online Archive of Academic Papers

Christoph Martin

Vienna University of Economics and Business

January 2012

Contents

Abstract

This paper is about presenting data in an intuitive and simple fashion through a web-browser using an XML file and XSL transformation.

A brief overview of the heavily used Extensible Markup Language is given. The structure and syntax of an XML document are explained. In addition the concepts of XML file transformation are illustrated.

The theoretical concepts described in the first two chapters of this work are then applied to a real world scenario in chapter 3. An archive for academic papers, available in plain HTML at the time of writing, is provided with filter and sorting functions in order to simplify the usage of the archive. Dynamic lists of keywords and authors are generated from the underlying XML file and presented to access the available data in a simple way.

A concluding roundup of the work and an outlook is given at the paper's end.

List of figures

1 XML

This chapter gives a brief overview of the Extensible Markup Language or XML in short. It explains what XML is and what it can be used for. Furthermore, it illustrates the structure of an XML document and explains what a well-formed and a valid document is.

1.1 Versions/Short history

XML was derived from SGML, the ISO standard for the description of marked-up electronic text. In December 2011, there are two current versions of XML. The initial XML 1.0 dates back to 1998, development started in 1996. It was only altered slightly since then and still got its natal version number. XML 1.1, the second, was initially published in 2004. At the time of writing the second edition of XML 1.1 is the latest, released in August 2006. [WikiXv]

The changes between the versions mainly affected the philosophy of element names:

“Whereas XML 1.0 provided a rigid definition of names, wherein everything that was not permitted was forbidden, XML 1.1 names are designed so that everything that is not forbidden (for a specific reason) is permitted.” [W3CXv]

This approach is adapted in the fifth edition of XML 1.0, published in November 2008. XML supports all kinds of Unicode characters, such as Chinese or Cyrillic text, contributing to its international implementation and worldwide usage. [W3CX10]

All examples and files in this paper are using XML 1.0.

1.2 Uses and Goals

XML is mainly a markup language used for describing and structuring hierarchical data in a standardized way. It has come into common use for the interchange of data over the Internet.

The World Wide Web Consortium states the design goals of XML as follows:

- “XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.” [W3CXOr]

1.3 Structure and Syntax

In order to keep this paper to a reasonable volume, only the aspects relevant for the task will be presented.

In every XML document, there is a root element. It contains all the other elements of the document, that’s why it’s also referred to as the document element. Elements contained in the document element are called sub-elements, which may contain sub-elements on their own. Sub-elements not containing any further elements are also called leaves. The terminology used (e.g. root, leaf) indicates the tree structure of an XML-document. [W3CXSc]

An XML document starts with optional pieces of information, describing the document itself:

- XML declaration such as
  `<?xml version="1.0"?>`
  indicating which version of XML is used

- a document type definition (DTD) or reference to one such as
  `<!DOCTYPE theses SYSTEM "schema.dtd">`
  in which the word `theses` stands for the name of the root element of the XML document

1.3.1 Elements

“Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag.” [W3CXML]

In other words, each non-empty element is composed of a start-tag (e.g. <tagName>), followed by the actual content of the element and an end-tag containing the identical name as given in the start-tag (e.g. </tagName>).

As one can see in the example above, start-tags start with an opening angle bracket “<” and end with a closing angle bracket “>”. End-tags start with an opening angle bracket “<”, followed by a slash “/” and end with a closing angle bracket “>”.

These pairs of tags identify the content they surround. Elements that do not contain any content are called empty elements. There are two ways to describe these empty elements:

- by stating a start-tag, immediately followed by an end-tag
  (e.g. <empty></empty>)

- by stating the abbreviated form starting with an opening angle bracket “<” and ending with a slash “/” followed by a closing angle bracket “>”
  (e.g. <empty/>)

All elements have to be closed, including empty elements.

As the nesting of the elements of an XML document is essential, they have to be arranged correctly: the position of the end-tag is always after the start-tag and both start- and end-tag have to be within the corresponding parent element. [W3CXML]

For example, this is correct:

```
<parent>
 <child>Content</child>
</parent>
```

Whereas the following example is incorrect because the parent end-tag (`</parent>`) is positioned before the start-tag (`<parent>`) and the child end-tag (`</child>`) is positioned outside the parent element.

```
</parent>
 <child>Content<parent>
</child>
```

1.3.2 Attributes

“Attributes are used to associate name-value pairs with elements. Attribute specifications must not appear outside of start-tags and empty-element tags.” [W3CXML]

Their purpose is to describe the particular element to which it is attached, e.g. it could be used to identify a student by his matriculation number:

```
<student matrNr=”0521397”>John Doe</student>
```

The attribute’s name follows the name of the element after a white-space. It is compulsory that the value of the attribute is surrounded by quotes. Albeit each attribute’s name has to be unique, the number of attributes is not limited.

1.4 Well-formed XML documents

The term “well-formed” is used to describe an XML document, which satisfies a number of syntax rules stated in the XML specification. This list of rules is quite long, some essential points are:

- There is a single “root” element that contains all the other elements.

- All elements are correctly nested, with none start- or end-tags missing and none overlapping.
- The special syntax characters such as the angle brackets “<” and “>” may not appear in the content of the document, i.e. they are exclusively used for their markup purpose.
- It contains only properly encoded legal Unicode characters. [W3CXwf]

Note that this list is only an excerpt and by no means exhaustive.

An XML document that is not well-formed, i.e. violates one or more of the syntax rules, will not be processed by an XML processor. It will throw an exception. This strong error sensitivity stands in contrast to the processing of HTML, which is fairly error tolerant, as it’s designed to display an eligible result even if there are syntax errors. [WikiXw]

1.5 Valid XML documents

“An XML document is valid if it has an associated document type declaration and if the document complies with the constraints expressed in it. The document type declaration must appear before the first element in the document.” [W3CXDt]

The document type declaration associates a particular XML document with a document type definition (see 1.5.1).

“XML processors are classified as validating or non-validating depending on whether or not they check XML documents for validity. A processor that discovers a validity error must be able to report it, but may continue normal processing.” [WikiXS]

Whereas an XML processor ceases normal processing when encountering a syntax error (therefore a non-well-formed XML document), it may continue processing when encountering a validity error (therefore a non-valid XML document).

1.5.1 Document Type Definition

The XML Document Type Definition is one of numerous XML schemas. In a Document Type Definition (DTD) the logical structure of an XML document is described. A DTD defines

- elements and their attributes, like a vocabulary (this is a constraining definition, i.e. other elements or attributes than stated in the DTD may not be used)
- the number of occurrences of these elements and attributes (i.e. whether they are mandatory or optional and whether they may appear once or multiple times)
- the nesting of the elements (i.e. which elements may contain which other elements)
- default values or a list of possible values for attributes
- the order of the elements [WikiXD]

2 Introduction to XSLT

In this chapter a brief introduction into XSL and the concepts of XSLT is given. It is shown, how an XSL transformation works and what the features of the Extensible Stylesheet Language are. Please not that due to the limited space in this paper, not all functions can be described here.

2.1 XSL

The Extensible Stylesheet Language (XSL) is a language for expressing stylesheets. “An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary.”

The specification of XSL 1.1 states that “designers use an XSL stylesheet to express their intentions about how […] structured content should be presented; that is, how the source content should be styled, laid out, and paginated onto some presentation medium, such as a window in a Web browser or a hand-held device, or a set of physical pages in a catalog, report, pamphlet, or book.” [W3CXS]

2.2 XSLT

The Extensible Stylesheet Language Transformations (XSLT) is a language used for transforming XML-documents.

“XSLT is designed for use as part of XSL, which is a stylesheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary. XSLT is also designed to be used independently of XSL. However, XSLT is not intended as a completely general-purpose XML transformation language. Rather it is designed primarily for the kinds of transformations that are needed when XSLT is used as part of XSL.” [W3CXT]

2.3 XPath

In order to transform an existing XML document properly, it is obviously essential to be able to find specific pieces of information and to navigate through an XML tree. For this purpose, XSLT uses the XML Path Language (XPath) to address parts of an XML document. XPath is a component of W3C's XSLT standard. [W3CXP2]

“There are two kinds of location path: relative location paths and absolute location paths.” [W3CXP]

Relative location paths consist of one or more location steps, separated by a slash “/”, which are executed consecutively. With each step a new node or set of nodes is selected. The selection is then the starting point (“context node”) for the following instruction and so on.

An absolute location path is similarly composed but it starts with a slash “/”. A slash “/” by itself selects the root node. Therefore an absolute location path will – ceteris paribus - ultimately always select the same set of nodes whereas the result of a relative location path depends on the selected context node.

The mentioned addressing of parts of an XML document is only one feature of XPath: “XPath includes over 100 built-in functions. There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more.” [SCHXP]

For further information on the XML Path Language, please refer to the specification: http://www.w3.org/TR/xpath20/

2.4 XSL Transformation

In order to produce a presentation of XML source data the XML document itself and a XSL stylesheet is required. “There are two aspects of this presentation process: first, constructing a result tree from the XML source tree and second, interpreting the result tree to produce formatted results suitable for presentation

on a display, on paper, in speech, or onto other media. The first aspect is called tree transformation and the second is called formatting.” [W3CXSP]

It is important to understand that after the transformation there is a new result tree that can be significantly different to the original source tree. XSLT is not only used to present the data of an XML document, it is also able to execute complex transformations like filtering or sorting the original data, resulting in an altered XML tree.

The XSL stylesheet contains the rules used for the transformation of the original source tree into the result tree. These construction rules consist of two parts:

- a pattern, which is matched against the elements of the source tree and
- a so called template, which generates a part of the result tree.

In this way it is possible to use one stylesheet for several documents with the same source tree structure.

The transformation itself is performed by an XSLT processor. All modern web-browsers come with an integrated XSLT processor, so that no additional software is required. [SCHXbr]

2.5 Features and Functions

In this chapter, the elements of XSLT used for the transformations in Chapter 3 will be briefly described. Further information on all the available elements can be found at the W3C XSLT Elements Reference http://www.w3schools.com/xsl/xsl_w3celementref.asp. [SCHXSE]

2.5.1 Version and Namespace declaration

First of all, the versions of XML and XSLT used in the document and the XSLT namespace are declared. The XSL stylesheet starts with the following code:

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

“The root element that declares the document to be an XSL style sheet is `<xsl:stylesheet>` or `<xsl:transform>`.” [SCHXST]

This is achieved by the following code:

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

As one can see, the namespace for the XSLT instructions is the official W3C XSLT namespace at http://www.w3.org/1999/XSL/Transform. All XSLT-tags will start with the namespace “`<xsl:`”.

2.5.2 Templates

“An XSL style sheet consists of one or more set of rules that are called templates. A template contains rules to apply when a specified node is matched.” [SCHXt]

A template is defined by an `<xsl:template>` element. In order to specify on which nodes of the XML source file the template should be applied, the match-attribute is used.

For example, `<xsl:template match="/">` defines a template and the associates this template with the root of the XML source document.

2.5.3 xsl:value-of

“The `<xsl:value-of>` element can be used to extract the value of an XML element and add it to the output stream of the transformation.” [SCHXv]

To specify which element should be extracted, an XPath expression is stated in a select attribute.

2.5.4 xsl:for-each

“The XSL `<xsl:for-each>` element can be used to select every XML element of a specified node-set.” [SCHXf]

The following example should illustrate this concept:

Excerpt of students.xml:

```
<students>
  <student>
    <firstname>John</firstname>
    <lastname>Doe</lastname>
  </student>
  <student>
    <firstname>Max</firstname>
    <lastname>Mustermann</lastname>
  </student>
</students>
```

Excerpt of students.xsl:

```
<xsl:template match="/">
 <html>
  <body>
    <xsl:for-each select="students/student">
      <xsl:value-of select="firstname"/>
      <br />
    </xsl:for-each>
  </body>
 </html>
</xsl:template>
```

The `<xsl:for-each>` instruction loops through all `<student>` tags in the XML file and extracts the value of the field `<firstname>`. Note that the XPath expression in the select attribute of the `<xsl:value-of>` is a relative location path. The `<br />` stands for a line-break, which will be added after each student.

So the output of students.xml and students.xsl would be:

```
John
```

```
Max
```

“We can also filter the output from the XML file by adding a criterion to the select attribute in the <xsl:for-each> element.” [SCHXf]

For example, look at the following XSL stylesheet:

Excerpt of students_filter.xsl

```
<xsl:template match="/">
 <html>
  <body>
    <xsl:for-each
     select="students/student[firstname='John']">
      <xsl:value-of select="lastname"/>
    </xsl:for-each>
  </body>
 </html>
</xsl:template>
```

The content is now filtered for students with a <firstname> tag containing “John”. For the students matching this condition, the last name is extracted. Combined with the XML file “students.xml” from above, the result would be:

```
Doe
```

The equals sign “=” is only one possible operator. “Legal filter operators are:

- = (equal)
- != (not equal)
- < (less than <)
- > (greater than >)” [SCHXf]

2.5.5 xsl:sort

As the name suggests the <xsl:sort> element sorts the output.

The element is added inside an <xsl:for-each> or an <xsl:apply-templates> element. This element has a select attribute too, which contains

the XPath to the node used for the sortation. An optional `order` attribute can be stated to specify, whether the output should be ordered ascending or descending. The default value, which is used in case no `order` attribute is given, is ascending. Additional optional attributes are `data-type`, e.g. number or text and `case-order`, which specifies whether upper- or lowercase letters should be ordered first. [SCHXs]

2.5.6 xsl:if

“The `<xsl:if>` element is used to put a conditional test against the content of the XML file.” [SCHXi]

The condition is placed in a `test` attribute. The element itself is placed inside an `<xsl:for-each>` loop. The applicable operators are the same as stated in 2.5.4.

2.5.7 xsl:choose

“The `<xsl:choose>` element is used in conjunction with `<xsl:when>` and `<xsl:otherwise>` to express multiple conditional tests.” [SCHXc]

This structure is similar to the switch-case concept in many other programming languages where the `<xsl:choose>` in XSLT stands for the switch and the `<xsl:when>` element stands for the case.

The following example illustrates how this works:

Excerpt of students_choose.xls

```
<xsl:template match="/">
 <html>
  <body>
    <xsl:for-each select="students/student">
      <xsl:choose>
        <xsl:when test="firstname='John'">
          <xsl:value-of select="lastname"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="firstname"/>
        </xsl:otherwise>
    </xsl:for-each>
  </body>
 </html>
</xsl:template>
```

This example tests, whether the first name of the students is “John”. If this condition is satisfied, the last name of the student is extracted. Otherwise the first name is extracted.

A combination of the XML file “students.xml” above and “students_choose.xls” would produce the following output:

```
Doe
Max
```

2.5.8 xsl:param

„The `<xsl:param>` element is used to declare a local or global parameter.” [SCHXpar]

A parameter has a name and optionally a default value, which is stated as an XPath expression in a `select` attribute.

The value of XSL parameters can be obtained by writing a dollar sign “$” followed by the name of the parameter, e.g. `$parameter`.

In chapter 3 the values of the required parameters are assigned to them dynamically with Javascript and then given to the XSL stylesheet to specify the sorting order or the desired type of theses. [SCHXpar]

3 Applying XSLT to http://wi.wu.ac.at/rgf/diplomarbeiten/

The goal of this paper is to apply the above-mentioned concepts to a real-world scenario. This chapter describes a cross-browser XSLT solution for the website http://wi.wu.ac.at/rgf/diplomarbeiten/. [WUrgfd] An introduction into the problem is given and the desired goal is described. It is shown, how the current representation of the papers at the time of writing looks like and what the new concept for the archive is. The new features for filtering and sorting the items are presented and the result is described.

3.1 Introduction and Goal

Professor Rony G. Flatscher from the Institute for Management Information Systems at the Vienna University of Economics and Business publishes selected papers of his students online at http://wi.wu.ac.at/rgf/diplomarbeiten/.

Figure 3.1 Screenshot of http://wi.wu.ac.at/rgf/diplomarbeiten/

3.1.1 Previous representation of papers (unstructured HTML)

Over the years the list of papers grew substantially and it contains approximately 70 papers from students of different universities at the time of writing. This data is presented as a simple HTML File containing the following pieces of information:

- Time of publication (year and month)
- Author(s)
    - First name
    - Last name
    - Mother tongue
- University, at which the paper was published
- Title of the work (with a link to download it)
- Language of the paper
- Description of the content of the paper (abstract)
- Links to further materials going with the work (with a link to download it)
- List of keywords describing the content of the paper

For the purpose of formatting the displayed information Cascading Stylesheets (CSS) are used. This will also be the case in the new concept with XML and XSLT.

Published seminar papers are separated from diploma, master and bachelor theses. The works in these two categories are listed chronologically. However, there is no way to easily find a certain paper as no means of filtering or sorting the content are given.

In addition the HTML code used is not consistent for all papers and is not valid (see http://www.w3.org/TR/html401/sgml/intro.html). [W3CSG]

```
<table width="100%" class="bakk" id="bakk_02">
<tr><td class="date" width="5%">
(2005)
<td class="author" width="70%">
<span class="author">Hahnekamp, Rainer</span>
<span class="text">(mother tongue: German),</span><br>
Wirtschaftsuniversit&auml;t Wien (Vienna University of Economics and Business Administration), Austria
<td class="language" width="25%">
<span class="text">Geschrieben in/language used:</span> English
<tr>
<td class="empty"> 
<td colspan="2" class="title">
<a href="BakkStuff/2005/200502_OOo-Hahnekamp/200502_Hahnekamp.pdf">
Extending The Scripting Abilities Of OpenOffice.org With BSF And JSR-223
</a>
<tr>
<td class="empty"> 
<td colspan="2" class="abstract" lang="en">
This work explores and implements a GUI for editing and deploying scripts to drive OpenOffice.org
in all programming languages that support BSF, i.e. the Java interface framework for deploying
scripts written in scripting languages. In addition it develops a Java class which allows for
formatting Java reflection results in HTML and thereby makes it possible to learn about interfaces
of Java classes at runtime.
<p>
<p class="material">
Further materials going with this thesis:
<a href="BakkStuff/2005/200502_OOo-Hahnekamp/200504_ljdEclipse.zip">
Life JavaDoc (ljd) for Eclipse
</a>.
<p>
Keywords: OpenOffice, OOo, StarOffice, Object REXX, BSF, BSF4Rexx, Bean Scripting Framework,
Nutshell Examples, UNO, Universal Network Objects, Java, Java Reflection, HTML
</table>
```

Figure 3.1.1 example of HTML code of previous representation of papers

3.1.2 New Concept (XML and XSLT)

In order to make the available information easily accessible a new XML structure will be defined and used to describe the papers. As the content itself will not be altered, the information available will be the same as before, but a new way to access it will be introduced.

As the transformation will take place at the client (“client side transformation”) and XSLT itself does not support any dynamic concepts, Javascript will be used to implement the desired features like sorting and filtering.

For excellent introductions into the concepts of Javascript please refer to

- http://de.selfhtml.org/javascript/intro.htm (german) or
- http://www.w3schools.com/js/ (english). [SELFja] [SCHja]

Figure 3.1.2 illustrates the new XML structure used to describe the papers.

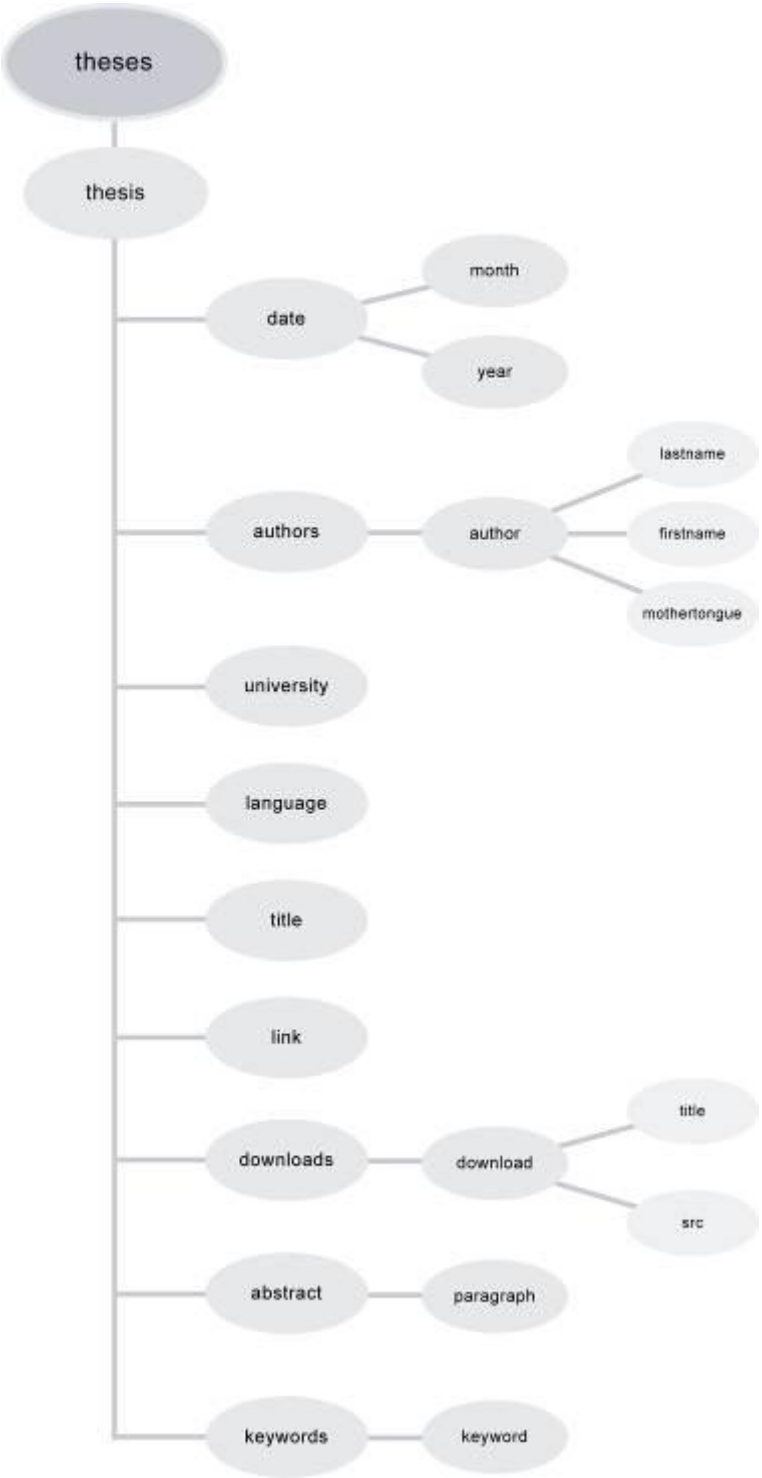

Figure 3.1.2 new XML structure

3.2 Filters and Orders

Several new ways to access the available information will be introduced.

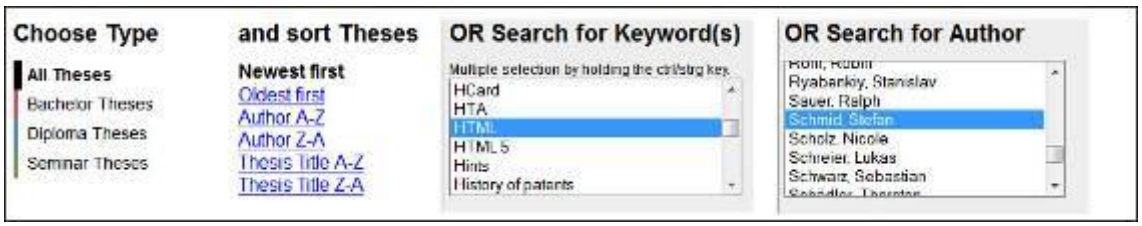

Figure 3.2 Screenshot of the new navigation on the website

3.2.1 Filtering the Type of Paper

As mentioned, there are three different kinds of theses available:

- Seminar theses
- Diploma theses
- Bachelor theses

The first possibility to filter the output is to choose which type of theses one would like to see. It will be possible to view all types or just one specific type of theses.

3.2.2 Sort Theses

The displayed items will be sorted by the time of publication in descending order by default. The following additional forms of sortation will be implemented:

- Oldest first (publication date)
- Author A-Z
- Author Z-A
- Thesis Title A-Z
- Thesis Title Z-A

In combination with the selection of the type of theses, these tools give the user powerful control over the displayed items.

3.2.3 Search for Keyword(s)

To many of the available theses keywords describing the content are assigned. This is helpful because the title of the thesis cannot always cover the entire scope of the actual work.

In order to take advantage of the availability of the keywords, a list of all occurring keywords will be dynamically extracted from the XML file and presented. By just clicking on them up to five keywords can be chosen and the related papers will be shown.

3.2.4 Search for Author

It may be the case that a user knows the name of the author, but not the title or type of the paper he/she is looking for.

For this case a list of authors, sorted by last name will be displayed. By just clicking on the name of the desired author, all papers written by him/her will be presented.

3.3 Result

In this chapter the solution found by the author is described and essential code excerpts are explained in greater detail.

The system was tested with the following browsers (on Windows 7):

- Internet Explorer 9.0.8112.16421
- Mozilla Firefox 7.0
- Google Chrome 16.0.912.63 m

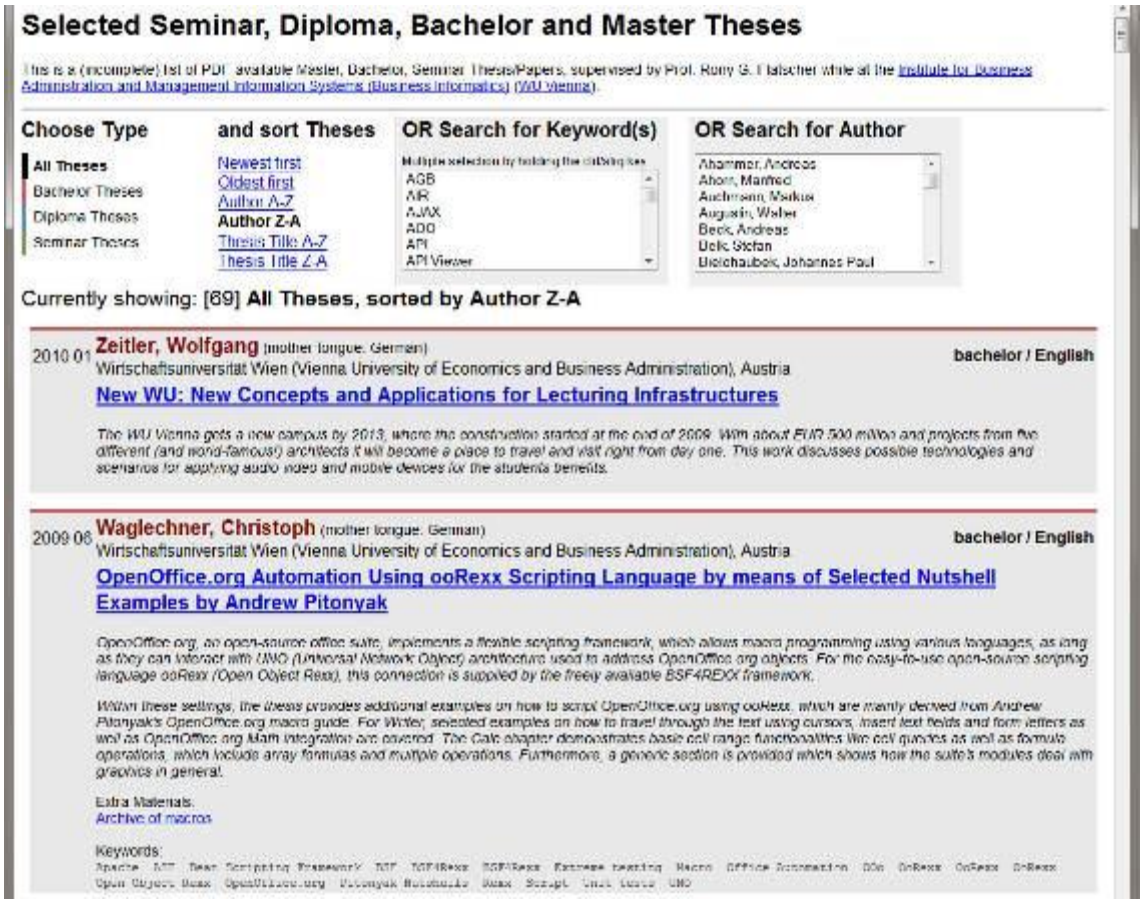

Figure 3.3 Screenshot of the new website

3.3.1 Saving the current state – `document.cookie`

The XSLT transformations are started by clicking on the navigation items at the top of the page. Since the selection of the type of the papers and the sortation can be combined, the current state has to be stored in order to display the correct “links”. For example: if the current type selection is “bachelor theses”, the links for the sortation have to be

- Bachelor theses – ordered by time (ascending & descending)
- Bachelor theses – ordered by title of thesis (ascending & descending)
- Bachelor theses – ordered by name of author (ascending & descending)

To achieve this, a string of two numbers is created and then stored locally at the clients' computer. The first number indicates the type of theses:

- 0 for all
- 1 for bachelor theses
- 2 for diploma theses
- 3 for seminar theses

The second digit indicates the sortation:

- 0 for newest first
- 1 for oldest first
- 2 for name of Author A-Z
- 3 for name of Author Z-A
- 4 for thesis title A-Z
- 5 for thesis title Z-A

A string of "00" stands for "All theses, sorted by publication date, newest first". A string of "24" stands for "Diploma theses, sorted by name of thesis title alphabetically".

In order to save this value at the clients' computer, a cookie is used.

The document object in Javascript refers to the content, which is displayed in a browser window, e.g. `document.title` refers to the title given in the HTML `<title>` tag.

The `document.cookie` is a string, which can be stored at the client. The Javascript code to set this cookie-String is simple:

```
<script type="text/javascript">
  document.cookie="10";
  document.write(document.cookie);
</script>
```

In order to insert Javascript in an HTML file the `<script>` tag is used. Every Javascript instruction ends with a semicolon ";". In this example, the String "10" is saved and then written in the document with the `write` method. [SELFdo]

3.3.2 Transformation with Javascript

The transformation is done with Javascript. The XML file and the XSL stylesheet are loaded by the browser and then the transformation is done at the client (client side transformation).

Based on the current content of the `document.cookie` the appropriate XSL file is chosen and the sorting order (ascending or descending) is added to the stylesheet dynamically as a parameter. Here's an excerpt of this code:

```
if(document.cookie.substr(1,1)=="0" || docu ↵
ment.cookie.substr(1,1)=="3" || docu ↵
ment.cookie.substr(1,1)=="5")
{
  sort_order="descending";
}
```

The `substr` method of Javascript allows extracting some of the characters of the whole string. The code `substr(1,1)` will return a part of the string, beginning with the second character (the first character is referenced with 0) and from there on one character will be extracted. As already mentioned, the second character of the `document.cookie` string contains the sorting information.

If the sorting information equals 0 (by time of publication descending), 3 (by name of author descending) or 5 (by title of thesis descending) the value "descending" is assigned to the variable `sort_order`. The two vertical strokes "||" stand for the logical operator "OR".

The code for the transformation itself looks as follows:

```
if (navigator.appName == "Microsoft Internet Explorer") {
 var xmlDoc = ↵
 new ActiveXObject("MSXML2.FreeThreadedDomDocument");
 xmlDoc.async = "false";
 xmlDoc.load("index.xml");
 var XML = xmlDoc;


 var xslDoc = ↵
 new ActiveXObject("MSXML2.FreeThreadedDomDocument");
 xslDoc.async = "false";
 xslDoc.load(xslUrl);
 var XSL = xslDoc;


 var XSLTCompiled = ↵
 new ActiveXObject("MSXML2.XSLTemplate");
 XSLTCompiled.stylesheet = XSL.documentElement;
 var XSLTProc = XSLTCompiled.createProcessor();
 XSLTProc.input = XML;


 XSLTProc.addParameter("type", "0");
 XSLTProc.addParameter("sort_order", "descending");


 XSLTProc.transform();
 document.getElementById("content").innerHTML ↵
 = XSLTProc.output;
} else {
 docRequest = new XMLHttpRequest();
 docRequest.open("GET", "index.xml", false);
 docRequest.send(null);
 xmlDoc = docRequest.responseXML;


 docRequest = new XMLHttpRequest();
 docRequest.open("GET", xslUrl, false);
 docRequest.send(null);
 xslDoc = docRequest.responseXML;
```

```
 docProcessor = new XSLTProcessor();
 docProcessor.importStylesheet(xslDoc);


 docProcessor.setParameter(null, "type", "0");
 docProcessor.setParameter(null, "sort_order", ↵
 "descending");


 docFragment = ↵
 docProcessor.transformToFragment(xmlDoc, document);
 document.getElementById("content").↵
 appendChild(docFragment);
}
```

The first part of the code is for users of the Internet Explorer. The second part, starting with the `else` is for all other browsers. Since the objective of this paper is a cross-browser solution and the Internet Explorer has a declining, yet still considerable market share at the time of writing, a separate solution for Microsoft's browser had to be found. The two parts are doing the exact same thing, only for different browsers. That's why in any case just one of the two parts is actually executed.

What happens in this code is that the XML and the XSL file are instantiated and loaded. Afterwards, an instance of an XSL processor (which performs the transformation of the input to the HTML output) is created and the XSL file is submitted to the processor.

We then tell the processor, which parameters we want to add to the XSL stylesheet. In this case, two parameters are submitted: `type` and `sort-order`. In this case, all types of papers are chosen and sorted in a reverse chronological order.

The HTML code resulting from the transformation is finally written into a `div` (i.e. an HTML element for an area, which can contain text, graphics, other areas etc.) with the id "content".

3.3.3 XSL Stylesheets

In this chapter, one of the used stylesheets is explained in more detail. It selects Bachelor theses and sorts them by time of publication.

As previously mentioned, the XSL stylesheet contains the instructions on how the XML source tree should be transformed and what output should be produced.

For our purpose, parameters are given to the XSL stylesheet. Therefore, they have to be defined in the stylesheet. This is achieved by the following code:

```
<xsl:param name="type" select="'null'" />
<xsl:param name="sort_order" select="'null'" />
```

The parameters are given the value “null” here because the actual values are added dynamically by the Javascript code. The value of XSL parameters can be obtained by writing a dollar sign “$” followed by the name of the parameter, e.g. `$type` or `$sort_order`.

The code for the sortation is

```
<xsl:sort select="date/year" data-type="number" or-
der="{$sort_order}" />
<xsl:sort select="date/month" data-type="number" or-
der="{$sort_order}" />
```

As one can see, two XML fields are used for the sortation: the year and the month. The publication date is saved numerically (i.e. “2” for February, not “February”), so the data-type is “number”. For the order attribute, which can be “ascending” or “descending”, the value of the parameter `$sort_order` is used.

In order to filter the output for Bachelor theses the parameter type has to be evaluated. This is done by the following code:

```
<xsl:choose>
  <xsl:when test="$type='1'">
    [… XSL instructions …]
  </xsl:when>
</xsl:choose>
```

The evaluation expression is placed in the test attribute of an `xsl:when` element. If the value of the parameter `$type` equals 1 (i.e. the code for bachelor theses, see 3.3.1) the XSL instructions within the `xsl:when` element are executed.

The XSL instructions create an HTML table, containing all the relevant information about the thesis.

The entire code can be found at the appendix.

4 Roundup and Outlook

In the first two chapters this paper aims at giving a brief introduction to XML and XSLT, explaining the relevant features and showing the possibilities of these powerful languages.

Chapter 3 illustrates how these technologies can be used to build dynamic filtering and sorting functions for an academic paper archive. It is needless to say that these concepts are applicable for many other scenarios, eventually with a much bigger underlying dataset.

XML is a highly used and powerful tool for structuring data in a standardized way and exchanging it. In combination with XSLT dynamic websites built on data available in an XML structure become possible. In a time of rapidly growing data amounts generated worldwide, it becomes crucial to deal with “big data” effectively and to take full advantage of their existence.

Appendix

Here you can see all the code used for chapter 3.

index.html

The index.html file is the most important file. It displays the navigation links as well as the selected theses. All Javascript methods are in this file. The transformations are called by “onClick”-events in this file.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<meta name="keywords" CONTENT="Diplomarbeiten, Masterarbeiten, Bakkalaureatarbeiten, Seminararbeiten, Rexx, Object Rexx, ooRexx, BSF4Rexx, Java, Wirtschaftsuniversit&auml;t Wien, Wirtschaftsinformatik, Rony G. Flatscher" LANG="de">
<meta name="keywords" CONTENT="Master Thesis, Vienna University of Economics and Business Administration, MIS Department, Business Informatics, Rony G. Flatscher Object Rexx, Rexx, ooRexx, BSF4Rexx, Java, Bachelor, seminar, Introduction, procedural Programming, Rony G. Flatscher" LANG="en">
<meta name="description" CONTENT="PDF-Diplomarbeiten, Masterarbeiten, Seminararbeiten, Bakkalaureatarbeiten" LANG="de">
<meta name="date" CONTENT="2011-12-23">
<meta name="copyright" CONTENT="&copy; 2002-2011 Rony G. Flatscher, Wirtschaftsuniversit&auml;t Wien" LANG="de">

<title>PDF-Diplom-/Master-/Bakk-/Seminar- [Paper | Thesis], Wirtschaftsuniversit&auml;t Wien, Inst. f. Wirtschaftsinformatik</title>
<link rel="StyleSheet" href="style.css" TYPE="text/css" />
</head>

<body>

<h1>Selected Seminar, Diploma, Bachelor and Master Theses</h1>

<p>This is a (incomplete) list of
PDF-available Master, Bachelor, Seminar Thesis/Papers,
supervised by Prof. Rony G. Flatscher
while at the <a href="http://wi.wu.ac.at/">Institute for Business Administration
and Management Information Systems (Business Informatics)</a>
(<a href="http://www.wu.ac.at/">WU Vienna</a>).
</p>

<hr />

<!-- ********** START NAVIGATION ********** -->
<div style="float:left;">
<!-- ********** START TYPE OF THESES ********** -->
<div class="navtitle" title="Choose which type of theses you would like to see">Choose Type</div>
<script type="text/javascript">

if(!document.cookie) { document.cookie="00"; }

var setcode0, setcode1, setcode2, setcode3;
setcode0 = "0".concat(document.cookie.substr(1,1));
setcode1 = "1".concat(document.cookie.substr(1,1));
setcode2 = "2".concat(document.cookie.substr(1,1));
setcode3 = "3".concat(document.cookie.substr(1,1));

if(document.cookie.substr(0,1)=="0") {
document.write("<div class='navlink' id='all2' onClick='javascript:set(setcode0);' title='Show all Theses'>All Theses</div>");
```

```
} else { document.write("<div class='navlink' id='all' onClick='javascript:set(setcode0);' ti-
tle='Show all Theses'>All Theses</div>"); }

if(document.cookie.substr(0,1)=="1") {
document.write("<div class='navlink' id='bac2' onClick='javascript:set(setcode1);' title='Show
Bachelor Theses'>Bachelor Theses</div>");
} else { document.write("<div class='navlink' id='bac' onClick='javascript:set(setcode1);' ti-
tle='Show Bachelor Theses'>Bachelor Theses</div>"); }

if(document.cookie.substr(0,1)=="2") {
document.write("<div class='navlink' id='dip2' onClick='javascript:set(setcode2);' title='Show
Diploma Theses'>Diploma Theses</div>");
} else { document.write("<div class='navlink' id='dip' onClick='javascript:set(setcode2);' ti-
tle='Show Diploma Theses'>Diploma Theses</div>"); }

if(document.cookie.substr(0,1)=="3") {
document.write("<div class='navlink' id='sem2' onClick='javascript:set(setcode3);' title='Show
Seminar Theses'>Seminar Theses</div>");
} else { document.write("<div class='navlink' id='sem' onClick='javascript:set(setcode3);' ti-
tle='Show Seminar Theses'>Seminar Theses</div>"); }
</script>
</div>
<!-- ********** END TYPE OF THESES ********** -->



<!-- ********** START SORTATION ********** -->
<div style="float:left; margin-left:20px;">
<div class="navtitle" title="Sort the displayed items by time of publication">and sort The-
ses</div>
<script type="text/javascript">
var setcode4, setcode5, setcode6, setcode7;
setcode4 = document.cookie.substr(0,1).concat("0");
setcode5 = document.cookie.substr(0,1).concat("1");
setcode6 = document.cookie.substr(0,1).concat("2");
setcode7 = document.cookie.substr(0,1).concat("3");
setcode8 = document.cookie.substr(0,1).concat("4");
setcode9 = document.cookie.substr(0,1).concat("5");

if(document.cookie.substr(1,1)=="0") {
document.write("<b class='navlink' onClick='javascript:set(setcode4);' title='Show newest Theses
first'>Newest first</b>");
} else { document.write("<span class='navlink' onClick='javascript:set(setcode4);' title='Show
newest Theses first'>Newest first</span>"); }
document.write("<br />");
if(document.cookie.substr(1,1)=="1") {
document.write("<b class='navlink' onClick='javascript:set(setcode5);' title='Show oldest Theses
first'>Oldest first</b>");
} else { document.write("<span class='navlink' onClick='javascript:set(setcode5);' title='Show
oldest Theses first'>Oldest first</span>"); }
document.write("<br />");
if(document.cookie.substr(1,1)=="2") {
document.write("<b class='navlink' onClick='javascript:set(setcode6);' title='Sort by Author A-
Z'>Author A-Z</b>");
} else { document.write("<span class='navlink' onClick='javascript:set(setcode6);' title='Sort
by Author A-Z'>Author A-Z</span>"); }
document.write("<br />");
if(document.cookie.substr(1,1)=="3") {
document.write("<b class='navlink' onClick='javascript:set(setcode7);' title='Sort by Author Z-
A'>Author Z-A</b>");
} else { document.write("<span class='navlink' onClick='javascript:set(setcode7);' title='Sort
by Author Z-A'>Author Z-A</span>"); }
document.write("<br />");
if(document.cookie.substr(1,1)=="4") {
document.write("<b class='navlink' onClick='javascript:set(setcode8);' title='Sort by Thesis
Title A-Z'>Thesis Title A-Z</b>");
} else { document.write("<span class='navlink' onClick='javascript:set(setcode8);' title='Sort
by Thesis Title A-Z'>Thesis Title A-Z</span>"); }
document.write("<br />");
if(document.cookie.substr(1,1)=="5") {
document.write("<b class='navlink' onClick='javascript:set(setcode9);' title='Sort by Thesis
Title A-Z'>Thesis Title Z-A</b>");
} else { document.write("<span class='navlink' onClick='javascript:set(setcode9);' title='Sort
by Thesis Title A-Z'>Thesis Title Z-A</span>"); }
</script>
```

```
</div>
<!-- ********** END SORTATION ********** -->



<!-- ********** START SEARCH FOR KEYWORDS ********** -->
<div style="float:left; margin-left:20px; width:260px; border-left:1px solid #666666; padding-
left:5px; background-color:#EEEEEE;">
<div class="navtitle" title="Choose a keyword you're interested in to see all related Papers">OR
Search for Keyword(s)</div>
<div style="font-size:12px;">Multiple selection by holding the ctrl/strg key.</div>
<div id="keys" style="display:none;"></div>
<form name="keyform" action="">
<script type="text/javascript">
function CheckSelection_key() {
var keysarray = new Array();
var zaehler = 0;

  for (i = 0; i < document.keyform.key.length; ++i) {
    if (document.keyform.key.options[i].selected == true) {
          keysarray[zaehler] = document.keyform.key.options[i].value;
          zaehler++;
        }
  }
  show_withkey(keysarray);
}

document.write("<select title='Select up to 5 keywords by holding the ctrl/strg key' name='key'
onchange='CheckSelection_key()' size='6' multiple='multiple' style='width:250px;'>");
var xmlDoc, xslDoc, docProcessor, docCache, DocRequest, docFragment, xslUrl;
var divID = "keys";
xslUrl = "keywords.xsl";

if (navigator.appName == "Microsoft Internet Explorer") {
var XML_IE; var XSL_IE;

var xmlDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
xmlDoc.async = "false";
xmlDoc.load("index.xml");
XML_IE = xmlDoc;

var xslDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
xslDoc.async = "false";
xslDoc.load(xslUrl);
XSL_IE = xslDoc;

if (window.ActiveXObject) {
        var XSLTCompiled = new ActiveXObject("MSXML2.XSLTemplate");
        XSLTCompiled.stylesheet = XSL_IE.documentElement;
        // create XSL-processor
        var XSLTProc = XSLTCompiled.createProcessor();
        XSLTProc.input = XML_IE;
        XSLTProc.transform();
        document.getElementById("keys").innerHTML = XSLTProc.output;
}

} else {
// instantiate, load the xml doc
docRequest = new XMLHttpRequest();
docRequest.open("GET", "index.xml", false);
docRequest.send(null);
xmlDoc = docRequest.responseXML;

// instantiate, load xsl doc
docRequest = new XMLHttpRequest();
docRequest.open("GET", "keywords.xsl", false);
docRequest.send(null);
xslDoc = docRequest.responseXML;

// instantiate processor, submit xsl
docProcessor = new XSLTProcessor();
docProcessor.importStylesheet(xslDoc);

// clear content-div
document.getElementById(divID).innerHTML = "";
```

```
// process into html, insert into content-div
docFragment = docProcessor.transformToFragment(xmlDoc, document);
document.getElementById(divID).appendChild(docFragment);
}

var key = document.all.keys.innerHTML;
var keysarr = key.split(":");
keysarr.sort();

function removeDuplicateElement(arrayName)
  {
  var newArray=new Array();
  label:for(var i=0; i<arrayName.length;i++ )
  {
  for(var j=0; j<newArray.length;j++ )
  {
  if(newArray[j]==arrayName[i])
  continue label;
  }
  newArray[newArray.length] = arrayName[i];
  }
  return newArray;
  }

keysarr = removeDuplicateElement(keysarr);

for (i=0; i<keysarr.length; i++) {
 if(!keysarr[i].substr(0,1)==" ") {
  document.write("<option value='" + keysarr[i] + "'>" + keysarr[i] + "</option>");
 }
}
document.write("</select>");

</script>
</form>
</div>
<!-- ********** END SEARCH FOR KEYWORDS ********** -->



<!-- ********** START SEARCH FOR AUTHOR ********** -->
<div style="float:left; margin-left:20px; width:260px; border-left:1px solid #666666; padding-
left:5px; background-color:#EEEEEE;">
<div class="navtitle" title="Choose the last name of an Author to see his/her Papers">OR Search
for Author</div>
<div id="authors" style="display:none;"></div>
<form name="authorform" action="">
<script type="text/javascript">
function CheckSelection_author() {
  for (i = 0; i < document.authorform.author.length; ++i) {
    if (document.authorform.author.options[i].selected == true) {
          show_withauthor(document.authorform.author.options[i].value);
        }
  }
}

document.write("<select name='author' title='Choose the last name of an Author to see his/her
Papers' onchange='CheckSelection_author()' size='7' style='width:240px;'>");
var xmlDoc, xslDoc, docProcessor, docCache, DocRequest, docFragment, xslUrl;
var divID = "authors";
xslUrl = "authors.xsl";

if (navigator.appName == "Microsoft Internet Explorer") {
var XML_IE; var XSL_IE;

var xmlDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
xmlDoc.async = "false";
xmlDoc.load("index.xml");
XML_IE = xmlDoc;

var xslDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
xslDoc.async = "false";
xslDoc.load(xslUrl);
XSL_IE = xslDoc;
```

```
if (window.ActiveXObject) {
        var XSLTCompiled = new ActiveXObject("MSXML2.XSLTemplate");
        XSLTCompiled.stylesheet = XSL_IE.documentElement;
        // create XSL-processor
        var XSLTProc = XSLTCompiled.createProcessor();
        XSLTProc.input = XML_IE;

        XSLTProc.transform();
        document.getElementById(divID).innerHTML = XSLTProc.output;
}

} else {
// instantiate, load the xml doc
docRequest = new XMLHttpRequest();
docRequest.open("GET", "index.xml", false);
docRequest.send(null);
xmlDoc = docRequest.responseXML;

// instantiate, load xsl doc
docRequest = new XMLHttpRequest();
docRequest.open("GET", "authors.xsl", false);
docRequest.send(null);
xslDoc = docRequest.responseXML;

// instantiate processor, submit xsl
docProcessor = new XSLTProcessor();
docProcessor.importStylesheet(xslDoc);

// clear content-div
document.getElementById(divID).innerHTML = "";

// process into html, insert into content-div
docFragment = docProcessor.transformToFragment(xmlDoc, document);
document.getElementById(divID).appendChild(docFragment);
}

var auth = document.all.authors.innerHTML;
var author = auth.split(":");
author.sort();

function removeDuplicateElement(arrayName)
  {
  var newArray=new Array();
  label:for(var i=0; i<arrayName.length;i++ )
  {
  for(var j=0; j<newArray.length;j++ )
  {
  if(newArray[j]==arrayName[i])
  continue label;
  }
  newArray[newArray.length] = arrayName[i];
  }
  return newArray;
  }

author = removeDuplicateElement(author);

var lastname;

for (i=0; i<author.length; i++) {
 if(!author[i].substr(0,1)==" ") {
  lastname = author[i].substr(0, author[i].indexOf(","));
  document.write("<option value='" + lastname + "'>" + author[i] + "</option>");
 }
}
document.write("</select>");

</script>
</form>
</div>
<!-- ********** END SEARCH FOR AUTHOR ********** -->
<!-- ********** END NAVIGATION ********** -->
```

```
<br />

<div style="clear:both;"></div>


<!-- ********** START SHOWING WHAT IS CURRENTLY DISPLAYED ********** -->
<div id="currently"></div>

<script type="text/javascript">
if(document.cookie=="00")
document.getElementById("currently").innerHTML = "<b>All Theses, Newest first</b>";
if(document.cookie=="10")
document.getElementById("currently").innerHTML = "<b>Bachelor Theses, Newest first</b>";
if(document.cookie=="20")
document.getElementById("currently").innerHTML = "<b>Diploma Theses, Newest first</b>";
if(document.cookie=="30")
document.getElementById("currently").innerHTML = "<b>Seminar Theses, Newest first</b>";

if(document.cookie=="01")
document.getElementById("currently").innerHTML = "<b>All Theses, Oldest first</b>";
if(document.cookie=="11")
document.getElementById("currently").innerHTML = "<b>Bachelor Theses, Oldest first</b>";
if(document.cookie=="21")
document.getElementById("currently").innerHTML = "<b>Diploma Theses, Oldest first</b>";
if(document.cookie=="31")
document.getElementById("currently").innerHTML = "<b>Seminar Theses, Oldest first</b>";

if(document.cookie=="02")
document.getElementById("currently").innerHTML = "<b>All Theses, sorted by Author A-Z</b>";
if(document.cookie=="12")
document.getElementById("currently").innerHTML = "<b>Bachelor Theses, sorted by Author A-Z</b>";
if(document.cookie=="22")
document.getElementById("currently").innerHTML = "<b>Diploma Theses, sorted by Author A-Z</b>";
if(document.cookie=="32")
document.getElementById("currently").innerHTML = "<b>Seminar Theses, sorted by Author A-Z</b>";

if(document.cookie=="03")
document.getElementById("currently").innerHTML = "<b>All Theses, sorted by Author Z-A</b>";
if(document.cookie=="13")
document.getElementById("currently").innerHTML = "<b>Bachelor Theses, sorted by Author Z-A</b>";
if(document.cookie=="23")
document.getElementById("currently").innerHTML = "<b>Diploma Theses, sorted by Author Z-A</b>";
if(document.cookie=="33")
document.getElementById("currently").innerHTML = "<b>Seminar Theses, sorted by Author Z-A</b>";

if(document.cookie=="04")
document.getElementById("currently").innerHTML = "<b>All Theses, sorted by Thesis Title Z-
A</b>";
if(document.cookie=="14")
document.getElementById("currently").innerHTML = "<b>Bachelor Theses, sorted by Thesis Title Z-
A</b>";
if(document.cookie=="24")
document.getElementById("currently").innerHTML = "<b>Diploma Theses, sorted by Thesis Title Z-
A</b>";
if(document.cookie=="34")
document.getElementById("currently").innerHTML = "<b>Seminar Theses, sorted by Thesis Title Z-
A</b>";

if(document.cookie=="05")
document.getElementById("currently").innerHTML = "<b>All Theses, sorted by Thesis Title Z-
A</b>";
if(document.cookie=="15")
document.getElementById("currently").innerHTML = "<b>Bachelor Theses, sorted by Thesis Title Z-
A</b>";
if(document.cookie=="25")
document.getElementById("currently").innerHTML = "<b>Diploma Theses, sorted by Thesis Title Z-
A</b>";
if(document.cookie=="35")
document.getElementById("currently").innerHTML = "<b>Seminar Theses, sorted by Thesis Title Z-
A</b>";
</script>
<!-- ********** END SHOWING WHAT IS CURRENTLY DISPLAYED ********** -->
```

```
<!-- ********** START SET document.cookie        ********** -->
<script type="text/javascript">
function set(num) {
if(num==10) { document.cookie="10"; }
else if(num==20) { document.cookie="20"; }
else if(num==30) { document.cookie="30"; }

else if(num==01) { document.cookie="01"; }
else if(num==11) { document.cookie="11"; }
else if(num==21) { document.cookie="21"; }
else if(num==31) { document.cookie="31"; }

else if(num==02) { document.cookie="02"; }
else if(num==12) { document.cookie="12"; }
else if(num==22) { document.cookie="22"; }
else if(num==32) { document.cookie="32"; }

else if(num==03) { document.cookie="03"; }
else if(num==13) { document.cookie="13"; }
else if(num==23) { document.cookie="23"; }
else if(num==33) { document.cookie="33"; }

else if(num==04) { document.cookie="04"; }
else if(num==14) { document.cookie="14"; }
else if(num==24) { document.cookie="24"; }
else if(num==34) { document.cookie="34"; }

else if(num==05) { document.cookie="05"; }
else if(num==15) { document.cookie="15"; }
else if(num==25) { document.cookie="25"; }
else if(num==35) { document.cookie="35"; }

else { document.cookie="00"; }
location.reload(false);
}
</script>
<!-- ********** END SET document.cookie  ********** -->



<!-- ********** CONTENT-DIV WHERE THESES-INFOS ARE INSERTED      ********** -->
<div id="content"></div>



<!-- ********** START TRANSFORMATION BASED ON document.cookie    ********** -->
<script type="text/javascript">
// declare local variables
var xmlDoc, xslDoc, docProcessor, docCache, DocRequest, docFragment, xslUrl, type, sort,
sort_order;
var divID = "content";

if(document.cookie.substr(1,1)=="2"||document.cookie.substr(1,1)=="3") {
xslUrl="sort_author.xsl"; }
else if(document.cookie.substr(1,1)=="4"||document.cookie.substr(1,1)=="5") {
xslUrl="sort_title.xsl"; }
else {
xslUrl="index.xsl"; }

if(document.cookie.substr(0,1)=="0") { type=0; }
else if(document.cookie.substr(0,1)=="1") { type=1; }
else if(document.cookie.substr(0,1)=="2") { type=2; }
else if(document.cookie.substr(0,1)=="3") { type=3; }

if(document.cookie.substr(1,1)=="0" || document.cookie.substr(1,1)=="3" || docu-
ment.cookie.substr(1,1)=="5") {
sort_order="descending";
}
else if(document.cookie.substr(1,1)=="1" || document.cookie.substr(1,1)=="2" || docu-
ment.cookie.substr(1,1)=="4") {
sort_order="ascending";
}

if (navigator.appName == "Microsoft Internet Explorer") {
```

```
        var XML;
        var XSL;

        var xmlDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
        xmlDoc.async = "false";
        xmlDoc.load("index.xml");
        XML = xmlDoc;

        var xslDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
        xslDoc.async = "false";
        xslDoc.load(xslUrl);
        XSL = xslDoc;

        var XSLTCompiled = new ActiveXObject("MSXML2.XSLTemplate");
        XSLTCompiled.stylesheet = XSL.documentElement;
        var XSLTProc = XSLTCompiled.createProcessor();
        XSLTProc.input = XML;

        XSLTProc.addParameter("type", type);
        XSLTProc.addParameter("sort_order", sort_order);

        XSLTProc.transform();
        document.getElementById("content").innerHTML = XSLTProc.output;

} else {
// instantiate, load the xml doc
docRequest = new XMLHttpRequest();
docRequest.open("GET", "index.xml", false);
docRequest.send(null);
xmlDoc = docRequest.responseXML;

// instantiate, load xsl doc
docRequest = new XMLHttpRequest();
docRequest.open("GET", xslUrl, false);
docRequest.send(null);
xslDoc = docRequest.responseXML;

// instantiate processor, submit xsl
docProcessor = new XSLTProcessor();
docProcessor.importStylesheet(xslDoc);

// PARAMETER
docProcessor.setParameter(null, "type", type);
docProcessor.setParameter(null, "sort_order", sort_order);

// clear content-div
document.getElementById(divID).innerHTML = "";

// process into html, insert into content-div
docFragment = docProcessor.transformToFragment(xmlDoc, document);
document.getElementById(divID).appendChild(docFragment);
}
</script>
<!-- ********** END TRANSFORMATION BASED ON document.cookie      ********** -->


<!-- ********** METHOD TRANSFORMATION BASED ON KEYWORDS  ********** -->
<script type="text/javascript">
function show_withkey(keyword) {
document.getElementById("currently").innerHTML = "<b>Theses with Keyword(s) \"<i>" + keyword +
"</i>\"</b>";

// declare the local variables
var xmlDoc, xslDoc, docProcessor, docCache, DocRequest, docFragment, xslUrl, type, sort;
var divID = "content";

// clear content from div
document.getElementById(divID).innerHTML = "";

if(keyword.length==1) { xslUrl="1key.xsl"; }
if(keyword.length==2) { xslUrl="2keys.xsl"; }
if(keyword.length==3) { xslUrl="3keys.xsl"; }
if(keyword.length==4) { xslUrl="4keys.xsl"; }
if(keyword.length==5) { xslUrl="5keys.xsl"; }
```

```
sort="descending";
type=0;

if (navigator.appName == "Microsoft Internet Explorer") {
        var XML;
        var XSL;

        var xmlDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
        xmlDoc.async = "false";
        xmlDoc.load("index.xml");
        XML = xmlDoc;

        var xslDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
        xslDoc.async = "false";
        xslDoc.load(xslUrl);
        XSL = xslDoc;

        var XSLTCompiled = new ActiveXObject("MSXML2.XSLTemplate");
        XSLTCompiled.stylesheet = XSL.documentElement;
        var XSLTProc = XSLTCompiled.createProcessor();
        XSLTProc.input = XML;

  // PARAMETERS
  if(keyword.length==1) { XSLTProc.addParameter("keyword", keyword[0]); }
  if(keyword.length==2) {
  XSLTProc.addParameter("keyword", keyword[0]);
  XSLTProc.addParameter("keyword2", keyword[1]);  }
  if(keyword.length==3) {
  XSLTProc.addParameter("keyword", keyword[0]);
  XSLTProc.addParameter("keyword2", keyword[1]);
  XSLTProc.addParameter("keyword3", keyword[2]);  }
  if(keyword.length==4) {
  XSLTProc.addParameter("keyword", keyword[0]);
  XSLTProc.addParameter("keyword2", keyword[1]);
  XSLTProc.addParameter("keyword3", keyword[2]);
  XSLTProc.addParameter("keyword4", keyword[3]);  }
  if(keyword.length==5) {
  XSLTProc.addParameter("keyword", keyword[0]);
  XSLTProc.addParameter("keyword2", keyword[1]);
  XSLTProc.addParameter("keyword3", keyword[2]);
  XSLTProc.addParameter("keyword4", keyword[3]);
  XSLTProc.addParameter("keyword5", keyword[4]);  }

        XSLTProc.transform();
        document.getElementById("content").innerHTML = XSLTProc.output;

} else {
try
{
  // instantiate, load the xml doc
  docRequest = new XMLHttpRequest();
  docRequest.open("GET", "index.xml", false);
  docRequest.send(null);
  xmlDoc = docRequest.responseXML;

  // instantiate, load xsl doc
  docRequest = new XMLHttpRequest();
  docRequest.open("GET", xslUrl, false);
  docRequest.send(null);
  xslDoc = docRequest.responseXML;

  // instantiate processor, submit xsl
  docProcessor = new XSLTProcessor();
  docProcessor.importStylesheet(xslDoc);

  // PARAMETER
  if(keyword.length==1) { docProcessor.setParameter(null, "keyword", keyword[0]); }
  if(keyword.length==2) {
  docProcessor.setParameter(null, "keyword", keyword[0]);
  docProcessor.setParameter(null, "keyword2", keyword[1]);  }
  if(keyword.length==3) {
  docProcessor.setParameter(null, "keyword", keyword[0]);
  docProcessor.setParameter(null, "keyword2", keyword[1]);
  docProcessor.setParameter(null, "keyword3", keyword[2]);  }
  if(keyword.length==4) {
```

```
  docProcessor.setParameter(null, "keyword", keyword[0]);
  docProcessor.setParameter(null, "keyword2", keyword[1]);
  docProcessor.setParameter(null, "keyword3", keyword[2]);
  docProcessor.setParameter(null, "keyword4", keyword[3]);  }
  if(keyword.length==5) {
  docProcessor.setParameter(null, "keyword", keyword[0]);
  docProcessor.setParameter(null, "keyword2", keyword[1]);
  docProcessor.setParameter(null, "keyword3", keyword[2]);
  docProcessor.setParameter(null, "keyword4", keyword[3]);
  docProcessor.setParameter(null, "keyword5", keyword[4]);  }

  // clear content-div
  document.getElementById(divID).innerHTML = "";

  // process into html, insert into content-div
  docFragment = docProcessor.transformToFragment(xmlDoc, document);
  document.getElementById(divID).appendChild(docFragment);
}
// catch any errors from the above code
catch(e)
{  } // end catch
} // end else (browser!=IE)
count_theses();
} // end go
</script>
<!-- ********** END METHOD TRANSFORMATION BASED ON KEYWORDS      ********** -->



<!-- ********** METHOD TRANSFORMATION BASED ON AUTHOR     ********** -->
<script type="text/javascript">
function show_withauthor(author) {
document.getElementById("currently").innerHTML = "<b>Theses from Author \"<i>" + author +
"</i>\"</b>";

// declare local variables
var xmlDoc, xslDoc, docProcessor, docCache, DocRequest, docFragment, xslUrl, type, sort;
var divID = "content";

// clear content from div
document.getElementById(divID).innerHTML = "";

xslUrl="author.xsl";

sort="descending";
type=0;

if (navigator.appName == "Microsoft Internet Explorer") {
        var XML;
        var XSL;

        var xmlDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
        xmlDoc.async = "false";
        xmlDoc.load("index.xml");
        XML = xmlDoc;

        var xslDoc= new ActiveXObject("MSXML2.FreeThreadedDomDocument");
        xslDoc.async = "false";
        xslDoc.load(xslUrl);
        XSL = xslDoc;

        var XSLTCompiled = new ActiveXObject("MSXML2.XSLTemplate");
        XSLTCompiled.stylesheet = XSL.documentElement;
        var XSLTProc = XSLTCompiled.createProcessor();
        XSLTProc.input = XML;

    // PARAMETER
    XSLTProc.addParameter("author", author);

        XSLTProc.transform();
        document.getElementById("content").innerHTML = XSLTProc.output;

} else {
try
{
```

```
  // instantiate, load the xml doc
  docRequest = new XMLHttpRequest();
  docRequest.open("GET", "index.xml", false);
  docRequest.send(null);
  xmlDoc = docRequest.responseXML;

  // instantiate, load xsl doc
  docRequest = new XMLHttpRequest();
  docRequest.open("GET", xslUrl, false);
  docRequest.send(null);
  xslDoc = docRequest.responseXML;

  // instantiate processor, submit xsl
  docProcessor = new XSLTProcessor();
  docProcessor.importStylesheet(xslDoc);

  // PARAMETER
  docProcessor.setParameter(null, "author", author);

  // clear content-div
  document.getElementById(divID).innerHTML = "";

  // process into html, insert into content-div
  docFragment = docProcessor.transformToFragment(xmlDoc, document);
  document.getElementById(divID).appendChild(docFragment);
}
// catch any errors from the above code
catch(e)
{  } // end catch
} // end else (browser!=IE)
count_theses();
} // end go
</script>
<!-- ********** END METHOD TRANSFORMATION BASED ON AUTHOR        ********** -->


<!-- ********** COUNT DISPLAYED ITEMS    ********** -->
<script type="text/javascript">
function count_theses()
{
if (navigator.appName == "Microsoft Internet Explorer") {
var string = document.getElementById("content").innerHTML;
} else { var string = document.all.content.innerHTML; }
 var c = 0;
 for (var i=0;i<string.length;i++)
 {
  if("<table" == string.substr(i,6) || "<TABLE" == string.substr(i,6))
  c++;
 }

 var curr = document.all.currently.innerHTML;
  document.getElementById("currently").innerHTML = "Currently showing: [" + c + "] " + curr;
}
count_theses();
</script>
<!-- ********** END COUNT DISPLAYED ITEMS        ********** -->


<hr />
<p>
<i><b>Note: </b>
If looking/reading English papers/theses
please bear in mind that the students usually have not had a lot of experience
in writing such kind of text in English!
They voluntarily agreed to write the text in English nevertheless,
in order for others to be able to take advantage of their hard work.</i>
</p>

</body>
</html>
```

index.xml

The XML-file contains all information about the theses, e.g. type, author, university, title, link-addresses for downloads etc. Please note that this is only an excerpt of the entire file.

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

<!-- START ITEMS -->
<theses>


<thesis type="diploma">
 <date>
  <year>2002</year>
 </date>
 <authors>
  <author>
   <lastname>Jeschko</lastname>
   <firstname>Michael</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Administration), Austria</university>
 <language>German</language>
 <title>Java-basierte Application Server am Beispiel "IBM WebSphere Application Server"</title>
 <link>2002_Jeschko/DiplomarbeitApplicationServer.pdf</link>
 <abstract>
  <paragraph>Describes the architecture of the complex IBM WebSphere Java based application server and gives invaluable directions and examples for installing the entire system and testing it. (Installation times for students went down from 2,5 weeks to one day!)</paragraph>
 </abstract>
</thesis>


<!-- NEXT ITEM -->

<thesis type="diploma">
 <date>
  <year>2002</year>
 </date>
 <authors>
  <author>
   <lastname>Putscher</lastname>
   <firstname>Andreas</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Administration), Austria</university>
 <language>English</language>
 <title>Implementation of an XML Based Slide Toolkit - Devising and Applying XSLT-Transformations</title>
 <link>2002_Putscher/Putscher_Andreas_XSLT.pdf</link>
 <downloads>
  <download type="miscellaneous">
   <title>Slide Toolkit</title>
   <src>2002_Putscher/slide_toolkit.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>Describes the architecture and the principles of XSLT (good overview intro), uses a Sun DTD, extends it and adds additional transformations. This way one can create foils in English or any other language and render them to PDF, SVGML, HTML, WML, pure text. It should be easy to extend the applicability by adding one owns transformations.</paragraph>
 </abstract>
</thesis>
```

```
<!-- NEXT ITEM -->

<thesis type="diploma">
 <date>
  <year>2002</year>
  <month></month>
 </date>
 <authors>
  <author>
   <lastname>Strahlhofer</lastname>
   <firstname>Roland</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>German</language>
 <title>Die Erstellung von Businessplänen für den mobilen Markt</title>
 <link>2002_Strahlhofer/Die Erstellung von Businessplaenen für den mobilen Markt.pdf</link>
 <abstract>
  <paragraph>Discusses the principles and rules of creating business plans for mobile applica-
tions and applies them to a real world example (Siemens).</paragraph>
 </abstract>
</thesis>



<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2005</year>
 </date>
 <authors>
  <author>
   <lastname>Maier</lastname>
   <firstname>Jürgen</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>German</language>
 <title>Analyse und Bewertung von Fortune Global 500 Websites</title>
 <link>BakkStuff/2005/200506_Global500_Maier/200506_Fortune_Global_500.pdf</link>
 <abstract>
  <paragraph>Diese Arbeit versucht, die Webauftritte von Global 500 Unternehmen zu analysieren,
wobei ein Sample von 100 Auftritten dafür herangezogen wird.</paragraph>
 </abstract>
 <keywords>
  <keyword>Webauftritt</keyword>
  <keyword>Analyse</keyword>
  <keyword>Global 500</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2005</year>
  <month>07</month>
 </date>
 <authors>
  <author>
   <lastname>Hoisl</lastname>
   <firstname>Bernhard</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
```

```
 <title>Automating Subversion - An Open Object Rexx Approach</title>
 <link>BakkStuff/2005/200507_Subversion_Hoisl/200507_AutomatingSubversion.pdf</link>
 <downloads>
  <download type="code">
   <title>Source code of examples</title>
   <src>BakkStuff/2005/200507_Subversion_Hoisl/200507_examples.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This work explores and implements scripts which allow driving the source code ver-
sion control system "subversion" from ooRexx. As there are Java implementations for subversion
it is possible to employ BSF4Rexx to drive the application.</paragraph>
 </abstract>
 <keywords>
  <keyword>Subversion</keyword>
  <keyword>Object REXX</keyword>
  <keyword>OoRexx</keyword>
  <keyword>Open Object Rexx</keyword>
  <keyword>Automation</keyword>
  <keyword>Java</keyword>
  <keyword>BSF4Rexx</keyword>
  <keyword>Automation</keyword>
  <keyword>Scripts</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2005</year>
 </date>
 <authors>
  <author>
   <lastname>Ahorn</lastname>
   <firstname>Manfred</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>German</language>
 <title>Semantic Web - Grundlagen</title>
 <link>BakkStuff/2005//200510_SemanticWEB-Ahorn/200510_SemanticWeb_Ahorn_V20.pdf</link>
 <abstract>
  <paragraph>Diese Arbeit führt in das Semantic Web ein und vermittelt alle relevanten Grundla-
gen.</paragraph>
 </abstract>
 <keywords>
  <keyword>Semantic Web</keyword>
  <keyword>Einführung</keyword>
  <keyword>Grundlagen</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2005</year>
  <month>11</month>
 </date>
 <authors>
  <author>
   <lastname>Ahammer</lastname>
   <firstname>Andreas</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
```

```
 <title>OpenOffice.org Automation: Object Model, Scripting Languages, "Nutshell"-
Examples</title>
 <link>BakkStuff/2005/200511_OOo-Ahammer/200511_OOoAutomation.pdf</link>
 <downloads>
  <download type="code">
   <title>Source code of the examples</title>
   <src>BakkStuff/2005/200511_OOo-Ahammer/20051106_examples.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This work builds on the work of Mr.  Augustin: "Examples for Open Office Automation
with Scripting Languages". It explores and demonstrates how OpenOffice.org can be automated via
Object REXX by using the Java programming interfaces of OpenOffice.org and BSF4Rexx.</paragraph>
  <paragraph>This time specific ooRexx support for OpenOffice (module "UNO.CLS" by Prof.
Flatscher, derived from <a href="http://wi.wu-
wien.ac.at/rgf/rexx/orx16/2005_orx16_NutShell_OOo.pdf">OOO.CLS</a> which is <a
href="http://wi.wu-wien.ac.at/rgf/rexx/orx16/2005_orx16_Gluing2ooRexx_OOo.pdf">based on the Java
interface to the UNO component technology of OpenOffice</a>) is used, which cuts down the neces-
sary code dramatically and makes those programs easy ledgible and understandable (looks almost
like pseudo-code).</paragraph>
 </abstract>
 <keywords>
  <keyword>OpenOffice</keyword>
  <keyword>OOo</keyword>
  <keyword>UNO.CLS</keyword>
  <keyword>StarOffice</keyword>
  <keyword>Object REXX</keyword>
  <keyword>BSF</keyword>
  <keyword>BSF4Rexx</keyword>
  <keyword>Bean Scripting Framework</keyword>
  <keyword>Nutshell Examples</keyword>
  <keyword>UNO</keyword>
  <keyword>Universal Network Objects</keyword>
  <keyword>Java</keyword>
  <keyword>Java Reflection</keyword>
  <keyword>HTML</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2006</year>
  <month>05</month>
 </date>
 <authors>
  <author>
   <lastname>Burger</lastname>
   <firstname>Martin</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>OpenOffice.org Automatisation with Object Rexx</title>
 <link>BakkStuff/2006/200605_Burger/Bakk_Arbeit_Burger20060519.pdf</link>
 <downloads>
  <download type="code">
   <title>Source code of the examples</title>
   <src>BakkStuff/2006/200605_Burger/Bakk_Arbeit_BurgerExamples_All_20060519.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This work builds on the work of <a href="#bakk_07">Mr.  Ahammer (above)</a>. It
explores and demonstrates how OpenOffice.org can be automated via Object REXX by using the Java
programming interfaces of OpenOffice.org and BSF4Rexx.
  </paragraph>
  <paragraph>This time the OpenOffice.org 2.0 scripting framework (written in Java) is used,
which allows to deploy the scripts as OpenOffice.org/StarOffice macros, in a platform independ-
ent manner.</paragraph>
 </abstract>
 <keywords>
```

```
  <keyword>OpenOffice</keyword>
  <keyword>OOo</keyword>
  <keyword>UNO.CLS</keyword>
  <keyword>StarOffice</keyword>
  <keyword>Object REXX</keyword>
  <keyword>BSF</keyword>
  <keyword>BSF4Rexx</keyword>
  <keyword>Bean Scripting Framework</keyword>
  <keyword>Nutshell Examples</keyword>
  <keyword>UNO</keyword>
  <keyword>Universal Network Objects</keyword>
  <keyword>Java</keyword>
  <keyword>Java Reflection</keyword>
  <keyword>HTML</keyword>
  <keyword>Scripting Framework</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="diploma">
 <date>
  <year>2006</year>
  <month>05</month>
 </date>
 <authors>
  <author>
   <lastname>Heinisch</lastname>
   <firstname>Florian</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>XML, Servlets and JavaServer Pages - An Introduction</title>
 <link>2006_Heinisch_F/thesis_Florian_Heinisch.pdf</link>
 <downloads>
   <download type="code">
   <title>Source code of the examples</title>
   <src>http://www.heinisch.cc/thesis/</src>
  </download>
  <download type="archive">
   <title>Above WwW site zipped</title>
   <src>2006_Heinisch_F/200605_Heinisch_Thesis_WWWsite.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>The primary goal of this thesis is to introduce the reader to the Extensible Markup
Language (XML), Servlets and JavaServer Pages (JSP)  which are technologies that allow to en-
hance the development of dynamic  content.
  </paragraph>
  <paragraph>The first part introduces the Extensible Markup Language (XML). After  a short
introduction of the historical development of XML, XML basics and  the structure of XML docu-
ments are explained. Finally, it is shown how to  validate and display XML documents.
  </paragraph>
  <paragraph>The second part provides a basic introduction to the Hypertext Transfer  Protocol
(HTTP).
  </paragraph>
  <paragraph>The third part gives the reader a basic understanding of servlets.  Therefore, the
servlet's structure and life-cycle are described.  Additionally, the provided theory about
servlets is highlighted with the  use of examples. For that purpose, the deployment of servlets
is  described so the reader gains the crucial knowledge needed to build Web applications with
Java.
  </paragraph>
  <paragraph>The fourth part deals with the discussion of JavaServer Pages (JSP).  This chap-
ter's purpose is to expound the JSP's life-cycle and  the main components a JSP may consist of.
In order to illustrate that  JSPs can include scripts written in none-Java programming lan-
guages, a  short digression is made into the Bean Scripting Framework (BSF). The  development of
script-less JSPs is explained at the end.
  </paragraph>
  <paragraph>The discussion on XML, Servlets and JavaServer Pages concludes with a  roundup and
outlook.
  </paragraph>
```

```
  <paragraph>In order to demonstrate how to build powerful Web applications by  combining XML,
Servlets and JavaServer Pages, the author developed a  simple shopping cart Web application that
extensively uses these three  technologies. For that reason this Web application is explained in
depth in this paper's fifth part, the appendix.
  </paragraph>
 </abstract>
 <keywords>
  <keyword>XML</keyword>
  <keyword>JSP</keyword>
  <keyword>Java Server Pages</keyword>
  <keyword>Rexx</keyword>
  <keyword>OoRexx</keyword>
  <keyword>BSF4Rexx</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2006</year>
  <month>06</month>
 </date>
 <authors>
  <author>
   <lastname>Nikoll</lastname>
   <firstname>Daniel</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>Comparing and Surveying Open Source Licenses</title>
 <link>BakkStuff/2006/200606_Nikoll/20060630_OpenSourceLicenses.pdf</link>
 <abstract>
  <paragraph>The content of this Bachelor Thesis deals predominantly with Open Source Licenses.
At the beginning you will get an overview about the history and the development of free and Open
Source Licenses. Afterwards the most frequently used licenses are presented and described. The
next point is about the connexion of the GPL with the European Law as well as an outlook on
GPLv3. Finally, in the last section the economic effects like consideration of profitability,
economic motives for participation, and incentives why Open Source Software should be implement-
ed are specified.</paragraph>
 </abstract>
 <keywords>
  <keyword>OSS</keyword>
  <keyword>OSL</keyword>
  <keyword>Open Source</keyword>
  <keyword>Open Source License</keyword>
  <keyword>Free Software</keyword>
  <keyword>Open Source Software</keyword>
  <keyword>Public Goods</keyword>
  <keyword>Public Domain</keyword>
  <keyword>Copyleft</keyword>
  <keyword>Law</keyword>
  <keyword>Europe</keyword>
  <keyword>United States</keyword>
  <keyword>US</keyword>
  <keyword>GPL</keyword>
  <keyword>Gnu Public License</keyword>
  <keyword>LGPL</keyword>
  <keyword>Lesser GPL</keyword>
  <keyword>Library GPL</keyword>
  <keyword>BSD</keyword>
  <keyword>Berkeley Software Distribution</keyword>
  <keyword>MPL</keyword>
  <keyword>Mozilla Public License</keyword>
  <keyword>Apache License</keyword>
  <keyword>Artistic License</keyword>
  <keyword>CPL</keyword>
  <keyword>Common Public License</keyword>
  <keyword>GPLv3</keyword>
 </keywords>
</thesis>
```

```
<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2006</year>
  <month>07</month>
 </date>
 <authors>
  <author>
   <lastname>Hinz</lastname>
   <firstname>Michael</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>OpenOffice.org Automatisation with Object Rexx</title>
 <link>BakkStuff/2006/200607_Hinz/20060712_OOo_calc_automation.pdf</link>
 <downloads>
  <download type="code">
   <title>All source-code</title>
   <src>BakkStuff/2006/200607_Hinz/20060629_OOo_calc_examples.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This paper gives an introduction to the OpenOffice.org architecture and explains
how the OpenOffice.org Calc component can be automated by using the scripting language Open
Object Rexx (ooRexx). This components are open sourced and can be downloaded free of charge from
the internet.
  </paragraph>
  <paragraph>The paper is divided into a theoretical and a practical part. In the theoretical
part, the main components, ooRexx, OpenOffice.org and the Bean Scripting Framework for ooRexx,
will be described and it explains how the single components can work together. At the end of
this part you can find an short installation guide, which shows you how to retrieve and install
the single components. The practical part provides some nutshell examples, that should demon-
strate how the OpenOffice.org Calc component can be automated. The concluding part should give a
short summary of the paper.
  </paragraph>
 </abstract>
 <keywords>
  <keyword>OpenOffice</keyword>
  <keyword>OOo</keyword>
  <keyword>UNO.CLS</keyword>
  <keyword>StarOffice</keyword>
  <keyword>Object REXX</keyword>
  <keyword>BSF</keyword>
  <keyword>BSF4Rexx</keyword>
  <keyword>Bean Scripting Framework</keyword>
  <keyword>Nutshell Examples</keyword>
  <keyword>UNO</keyword>
  <keyword>Universal Network Objects</keyword>
  <keyword>Java</keyword>
  <keyword>Scripting Framework</keyword>
  <keyword>Scalc</keyword>
  <keyword>Spreadsheet</keyword>
  <keyword>OoRexx</keyword>
 </keywords>
</thesis>



<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2006</year>
  <month>07</month>
 </date>
 <authors>
  <author>
   <lastname>Prem</lastname>
   <firstname>Matthias</firstname>
```

```
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>ooRexx Snippets for OpenOffice.org Writer</title>
 <link>BakkStuff/2006/200607_Prem/20060724_ooRexxSnippetsOOoWriter_2.1_odt.pd</link>
 <downloads>
  <download type="code">
   <title>All source-code</title>
   <src>BakkStuff/2006/200607_Prem/ooRexxSnippetsOOoWriter_oorexx_snippets.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This paper deals with the use of ooRexx as a scripting language for automation  of
OpenOffice.org Writer.
  </paragraph>
  <paragraph>At first, there will be an introduction to the technical requirements, which in-
clude  the software that has to be installed. Concerning ooRexx there is also a  sub chapter
about its syntax and common instructions, to give a feeling for this  programming language.
  </paragraph>
  <paragraph>  The next chapter is about the architectural approach behind ooRexx and  OpenOf-
fice.org. It is described how OpenOffice.org can be accessed using  ooRexx.
  </paragraph>
  <paragraph>Chapter four is a small installation guide, which shows how to set up the different
software programmes and configure them correctly.  Chapter five and six show how the automation
of OpenOffice.org Writer can be  done. Small snippets, which are code examples, demonstrate
different tasks.  At last the conclusion gives a small summary and an outlook.
  </paragraph>
 </abstract>
 <keywords>
  <keyword>OpenOffice</keyword>
  <keyword>OOo</keyword>
  <keyword>UNO.CLS</keyword>
  <keyword>StarOffice</keyword>
  <keyword>Object REXX</keyword>
  <keyword>BSF</keyword>
  <keyword>BSF4Rexx</keyword>
  <keyword>Bean Scripting Framework</keyword>
  <keyword>Nutshell Examples</keyword>
  <keyword>UNO</keyword>
  <keyword>Universal Network Objects</keyword>
  <keyword>Java</keyword>
  <keyword>Scripting Framework</keyword>
  <keyword>Swriter</keyword>
  <keyword>Word processor</keyword>
  <keyword>OoRexx</keyword>
 </keywords>
</thesis>



<!-- NEXT ITEM -->

<thesis type="diploma">
 <date>
  <year>2006</year>
  <month>12</month>
 </date>
 <authors>
  <author>
   <lastname>Novak</lastname>
   <firstname>Daniela</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>An Introduction to the Enterprise JavaBeans technology and Integrated Development Envi-
ronments for implementing EJB applications</title>
 <link>2006_Novak_D/20061214_Diplomarbeit_EJB_DanielaNovak.pdf</link>
 <abstract>
```

```
  <paragraph>This Master Thesis introduces the J2EE Enterprise Java Beans version 2.1 and 3.0
together with self contained nutshell examples using different integrated development environ-
ments (IDE). The discussed IDEs are NetBeans 5.0, IBM Rational Application Developer for Web-
Sphere, Sun Studio Enterprise 8 and Eclipse with Web Tools Platform 1.5.1, JBoss Eclipse IDE,
and Oracle JDeveloper 10g Release 3.</paragraph>
 </abstract>
 <keywords>
  <keyword>EJB</keyword>
  <keyword>J2EE</keyword>
  <keyword>IBM</keyword>
  <keyword>Rational</keyword>
  <keyword>Sun</keyword>
  <keyword>Eclipse</keyword>
  <keyword>JBoss</keyword>
  <keyword>IDE</keyword>
  <keyword>Message Beans</keyword>
  <keyword>Entity Beans</keyword>
  <keyword>Session Beans</keyword>
 </keywords>
</thesis>



<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2007</year>
  <month>01</month>
 </date>
 <authors>
  <author>
   <lastname>Auchmann</lastname>
   <firstname>Markus</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>Apache Velocity</title>
 <link>BakkStuff/2007/200701_Auchmann/Auchmann-apache_velocity.pdf</link>
 <downloads>
  <download type="miscellaneous">
   <title>Presentation</title>
   <src>BakkStuff/2007/200701_Auchmann/Auchmann-apache_velocity_presentation.pdf</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This Bachelor Thesis discusses and introduces Apache's Velocity.</paragraph>
 </abstract>
 <keywords>
  <keyword>OoRexx</keyword>
  <keyword>Apache</keyword>
  <keyword>Velocity</keyword>
  <keyword>Jacl</keyword>
  <keyword>TcL</keyword>
  <keyword>Rhino</keyword>
  <keyword>JavaScript</keyword>
  <keyword>BSF4Rexx</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2007</year>
  <month>01</month>
 </date>
 <authors>
  <author>
   <lastname>Kauril</lastname>
```

```
   <firstname>Michael</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>Flexible Word Processing Automation with OpenOffice.org</title>
 <link>BakkStuff/2007/200701_Kauril/Kauril-MA14-
FlexibleWordProcessingAutomationWithOOo.pdf</link>
 <downloads>
  <download type="code">
   <title>Code</title>
   <src>BakkStuff/2007/200701_Kauril/th_final.html</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>The work gives a detailed description of the realisation of the automation project
of open source office software conducted in cooperation with the        city of Vienna's depart-
ments MA 14 and MA 22. It provides a theoretical background in order to get a better understand-
ing of the implementation part of the work. The chapters concerned with theoretic concepts de-
scribe the design of OpenOffice.org and gives information about the API of OOo and the used
programming language. It also gives information about the system requirements        concerning
installed software. The practical part describes in detail the realisation and gives explana-
tions to every part of the programming code.</paragraph>
 </abstract>
 <keywords>
  <keyword>OoRexx</keyword>
  <keyword>OpenOffice</keyword>
  <keyword>Word processor</keyword>
  <keyword>Automation</keyword>
  <keyword>Scripting</keyword>
 </keywords>
</thesis>



<!-- NEXT ITEM -->

<thesis type="diploma">
 <date>
  <year>2007</year>
  <month>02</month>
 </date>
 <authors>
  <author>
   <lastname>Kegel</lastname>
   <firstname>Rainer</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>TeRA - A TestRunner Application for ooRexxUnit</title>
 <link>2007_Kegel_R/TeRA.pdf</link>
 <downloads>
  <download type="code">
   <title>Content of the directory, including initial version of the code and a Windows install-
er to install the application.</title>
   <src>2007_Kegel_R/</src>
  </download>
   <download type="miscellaneous">
   <title>Presentation on TeRA for the 18th International Rexx Symposium, Tampa, 2007-04-30.
Includes updated version of the code.</title>
   <src>http://wi.wu-wien.ac.at/rgf/rexx/orx18/TeRA/</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>ooRexxUnit is a framework for creating and running test cases modelled after
Apache's junit framework.
  </paragraph>
  <paragraph>This thesis realizes the first test runner application for ooRexxUnit. It turns the
ooRexxUnit gathered information into XML encoded files and processes them to create different
reports and renderings using a mix of ooRexx code and XSLT transformations.
```

```
  </paragraph>
  <paragraph>The GUI interface in this version is realized with DHTML, which is available on the
Micrsoft Internet Explorer.
  </paragraph>
 </abstract>
 <keywords>
  <keyword>OoRexx</keyword>
  <keyword>OoRexxUnit</keyword>
  <keyword>XML</keyword>
  <keyword>XSLT</keyword>
  <keyword>CSS</keyword>
  <keyword>DHTML</keyword>
  <keyword>HTA</keyword>
 </keywords>
</thesis>



<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2007</year>
  <month>02</month>
 </date>
 <authors>
  <author>
   <lastname>Schmid</lastname>
   <firstname>Stefan</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>Facilitate Data Access in OpenOffice.org using ooRexx</title>
 <link>BakkStuff/2007/200702_Schmid/OODatabase_v4_1_final.pdf</link>
 <downloads>
  <download type="code">
   <title>Code snippets</title>
   <src>BakkStuff/2007/200702_Schmid/Ooo-Base-Snippets.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>
  The thesis deals with OpenOffice.org automation in view of data access using the scripting
programming language Open Object Rexx. This data access includes the communication of OpenOf-
fice.org with external databases, address books and even the creation of forms.
  </paragraph>
  <paragraph>Firstly some theoretically background is given about databases with an focus on
relational databases, since they are used in snippets of this thesis. Furthermore some installa-
tion instructions are given for the used databases. The next chapter deals witch all the tech-
nical prerequisites that are required to accomplish an OpenOffice.org automation.
  </paragraph>
  <paragraph>After the theoretically requirements have been presented, the following chapters
provide several snippets regarding to the access of data sources. They also include examples for
automatically creating a predefined query and a form.
  </paragraph>
 </abstract>
 <keywords>
  <keyword>OoRexx</keyword>
  <keyword>OpenOffice</keyword>
  <keyword>Base</keyword>
  <keyword>DataBase</keyword>
  <keyword>Automation</keyword>
  <keyword>Scripting</keyword>
 </keywords>
</thesis>


<!-- NEXT ITEM -->

<thesis type="bachelor">
 <date>
  <year>2011</year>
```

```
  <month>07</month>
 </date>
 <authors>
  <author>
   <lastname>Belk</lastname>
   <firstname>Stefan</firstname>
   <mothertongue>German</mothertongue>
  </author>
  <author>
   <lastname>Dennis</lastname>
   <firstname>Robert</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>Innovative Use of Mobile Applications Supporting Learning and Teaching Activities at the
New WU</title>
 <link>BakkStuff/2011/201107_Belk_Stoehr_Mobile_WU/Mobile_Apps_for_the_New_WU-
20110715.pdf</link>
 <downloads>
  <download type="code">
   <title>Addendum "LectureInfo", code (zip archive)</title>
   <src>BakkStuff/2011/201107_Belk_Stoehr_Mobile_WU/LectureInfo-code.zip</src>
  </download>
  <download type="code">
   <title>"QueryTool" code (zip archive)</title>
   <src>BakkStuff/2011/201107_Belk_Stoehr_Mobile_WU/QueryTool-code.zip</src>
  </download>
  <download type="code">
   <title>"ReservationSystem" code (zip archive)</title>
   <src>BakkStuff/2011/201107_Belk_Stoehr_Mobile_WU/ReservationSystem-code.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This bachelor thesis provides an overview about exemplary mobile applications for
the new WU. It  can be divided into three main parts.
  </paragraph>
  <paragraph>The first part covers the basics which are necessary for understanding the paper.
Relevant wireless  technologies and mobile operating systems are explained. Furthermore, the
planned IT deployments at  the university's new site are briefly presented.
  </paragraph>
  <paragraph>The second part gives details on the different application examples. These are
described as use  cases. Sixteen use cases are described in detail, and additional twelve use
cases are listed and  briefly described.
  </paragraph>
  <paragraph>In the third part, two use case examples are selected and implemented as applica-
tions. The first  shows how a reservation system for learning places at the library could look
like. It states that  while using HTML5 a tradeoff between new features, maintainability and
compatibility had to be made.  The second implementation allows students to ask questions digi-
tally in large lectures. It can  improve teaching quality, but in its form, it requires that
students and lecturers use Android  powered devices, putting those who haven't at a disad-
vantage.
  </paragraph>
 </abstract>
 <keywords>
  <keyword>Mobile</keyword>
  <keyword>Mobile Applications</keyword>
  <keyword>Android</keyword>
  <keyword>HTML 5</keyword>
  <keyword>Java</keyword>
 </keywords>
</thesis>



<!-- NEXT ITEM -->

<thesis type="diploma">
 <date>
  <year>2010</year>
  <month>10</month>
 </date>
 <authors>
```

```
  <author>
   <lastname>Hohenegger</lastname>
   <firstname>Franz</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>BNF4OOo - Managing Backus-Naur-Forms with OpenOffice</title>
 <link>2011_Hohenegger/Hohenegger_BNF4OOo.pdf</link>
 <downloads>
  <download type="archive">
   <title>Installation archive</title>
   <src>2011_Hohenegger/BNF4OOo_v100.zip</src>
  </download>
  <download type="miscellaneous">
   <title>Installation readme</title>
   <src>2011_Hohenegger/readme.txt</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>"This diploma thesis is about a program named    <a
href="http://sourceforge.net/projects/bnf4ooo/">BNF4OOo</a>    which enables the user to
manage different supported <a
href="http://en.wikipedia.org/wiki/Backus%E2%80%93Naur_Form">BNF</a>-dialects    and make own
customized <a href="http://en.wikipedia.org/wiki/Backus%E2%80%93Naur_Form">BNF</a>-dialects as
well. These dialects can be transferred into character-based syntax diagrams and a XML format
closely related to IBM's    <a
href="http://en.wikipedia.org/wiki/Darwin_Information_Typing_Architecture">DITA</a>.    <a
href="http://www.OpenOffice.org">OpenOffice</a> is used as a graphical inferface for the
transformations. This work describes the supported BNF-dialects with their used syntax    struc-
tures and their implementation in <a
href="http://sourceforge.net/projects/bnf4ooo/">BNF4OOo</a>. Furthermore it introduces the in-
ternal    logic and structure of <a
href="http://sourceforge.net/projects/bnf4ooo/">BNF4OOo</a>."
  </paragraph>
  <paragraph><b>Author's thanks:</b> "I want to thank Prof. Dr. Rony G. Flatscher for his great
help and support during the creation of this paper. Without his impulses <a
href="http://sourceforge.net/projects/bnf4ooo/">BNF4OOo</a> would not have this set of func-
tions. Furthermore <a href="http://sourceforge.net/projects/bnf4ooo/">BNF4OOo</a> would not run
without his work on <a href="http://sourceforge.net/projects/bsf4oorexx/">BSF4ooRexx</a> and <a
href="http://wi.wu-wien.ac.at/rgf/rexx/orx18/log4r/">log4rexx</a>. In addition I want to thank
Jean-Louis Faucher for showing me IBM's DITA and giving me important information for improving
the BNF-parser."
  </paragraph>
 </abstract>
 <keywords>
  <keyword>Open Object Rexx</keyword>
  <keyword>OoRexx</keyword>
  <keyword>BSF</keyword>
  <keyword>BSF4ooRexx</keyword>
  <keyword>Bean Scripting Framework</keyword>
  <keyword>Backus Naur Form</keyword>
  <keyword>BNF</keyword>
  <keyword>OpenOffice.org</keyword>
  <keyword>OOo</keyword>
  <keyword>LibreOffice</keyword>
  <keyword>LO</keyword>
  <keyword>Log4rexx</keyword>
 </keywords>
</thesis>



<!-- NEXT ITEM -->

<thesis type="seminar">
 <date>
  <year>2006</year>
  <month>07</month>
 </date>
 <authors>
  <author>
   <lastname>Görlich</lastname>
```

```
   <firstname>Gerhard</firstname>
   <mothertongue>German</mothertongue>
  </author>
  <author>
   <lastname>Realfsen</lastname>
   <firstname>Åsmund</firstname>
   <mothertongue>Norvegian</mothertongue>
  </author>
  <author>
   <lastname>Spanberger</lastname>
   <firstname>David</firstname>
   <mothertongue>German</mothertongue>
  </author>
 </authors>
 <university>Wirtschaftsuniversität Wien (Vienna University of Economics and Business Admin-
istration), Austria</university>
 <language>English</language>
 <title>BSF4Rexx and OpenOffice.org Nutshell-Examples</title>
 <link>Seminararbeiten/2006s_wu/20060628_BSF4RexxSnippets_version_4.pdf</link>
 <downloads>
  <download type="code">
   <title>Nutshell examples</title>
   <src>Seminararbeiten/2006s_wu/20060628_BSF4RexxSnippets_code.zip</src>
  </download>
 </downloads>
 <abstract>
  <paragraph>This seminar paper introduces the easy to learn syntax of Open Object
  Rexx (<a href="http://www.ooRexx.org">ooRexx</a>) and the
  <a href="http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/">BSF4Rexx</a>
  external Rexx function package, which allows the weakly typed language
  ooRexx to interface with (strictly typed) Java.
  </paragraph>
  <paragraph>
  It defines and explains numerous small (nutshell) examples where the
  functionality of Java class libraries is used for ooRexx. In addition,
  the students create examples for automating/scripting the opensource
  office package <a href="http://www.OpenOffice.org">OpenOffice.org (OOo)</a>
  in an openplatform way using the OOo Java interface for that purpose.</paragraph>
 </abstract>
 <keywords>
  <keyword>OpenOffice</keyword>
  <keyword>OOo</keyword>
  <keyword>StarOffice</keyword>
  <keyword>Object REXX</keyword>
  <keyword>Java</keyword>
  <keyword>BSF</keyword>
  <keyword>BSF4Rexx</keyword>
  <keyword>Bean Scripting Framework</keyword>
  <keyword>Nutshell Examples</keyword>
  <keyword>UNO</keyword>
  <keyword>Universal Network Objects</keyword>
  <keyword>OoRexx</keyword>
 </keywords>
</thesis>
```

index.xsl

The index.xsl file is used when the user selects sortation by time, regardless of the selected type/types of thesis. The parameters are type and sort_order.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />

<xsl:param name="type" select="'null'" />
<xsl:param name="sort_order" select="'null'" />

<xsl:template match="/">
<xsl:for-each select="theses/thesis">

<xsl:sort select="date/year" data-type="number" order="{$sort_order}" />
<xsl:sort select="date/month" data-type="number" order="{$sort_order}" />



<xsl:choose>
<xsl:when test="$type='0'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
```

```
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
</xsl:when>


<xsl:when test="$type='1'">
        <xsl:if test="@type='bachelor'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
```

```
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>


<xsl:when test="$type='2'">
        <xsl:if test="@type='diploma'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
```

```
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>


<xsl:when test="$type='3'">
        <xsl:if test="@type='seminar'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
```

```
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>
</xsl:choose>

</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

author.xsl

The author.xsl is used for displaying items written by a certain person. The used parameter “author” is checked against the “lastname”-tag in the XML-file.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:param name="author" select="'null'" />
<xsl:template match="/">
        <html>
                <head>
                </head>
                <body>
                <div>
                <xsl:for-each select="theses/thesis[authors/author/lastname=$author]">
                        <xsl:sort select="date/year" data-type="number" order="descending" />
                        <xsl:sort select="date/month" data-type="number" order="descending" />
                        <table width="100%">
                        <xsl:attribute name="class">
                        <xsl:value-of select="@type"/>
                        </xsl:attribute>
                        <tr>
                        <td class="date" width="5%">
                                <xsl:value-of select="date/year"/> <xsl:value-of se-
lect="date/month"/>
                        </td>
                        <td class="author" width="70%">
                                <span class="author">
```

```
                                <xsl:for-each select="authors/author">
                                <xsl:value-of select="lastname"/>, <xsl:value-of se-
lect="firstname"/> (mother tongue: <xsl:value-of select="mothertongue"/>)
                                <br />
                                </xsl:for-each>
                                </span>
                                <xsl:value-of select="university"/>
                        </td>
                        <td class="language" width="25%">
                                <xsl:value-of select="@type"/> / <xsl:value-of se-
lect="language"/>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="title">
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="link" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="abstract" lang="en">
                                <i>
                                <xsl:for-each select="abstract/paragraph">
                                <p><xsl:value-of select="." /></p>
                                </xsl:for-each>
                                </i>
                                <xsl:if test="downloads/download">
                                Extra Materials:
                                <xsl:for-each select="downloads/download">
                                <br />
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="src" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                                </xsl:for-each>
                                </xsl:if>
                                <xsl:if test="keywords/keyword">
                                <br /><br />
                                Keywords:
                                <br />
                                <code>
                                <xsl:for-each select="keywords/keyword">
                                <xsl:sort select="." data-type="text" order="ascending" />
                                <xsl:value-of select="." /> 
                                </xsl:for-each>
                                </code>
                                </xsl:if>
                        </td>
                        </tr>
                        </table>
                        <br />
                </xsl:for-each>
                </div>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

sort_author.xsl

This file is comparable to index.xsl. Whereas the index.xsl file is used for sortation by time, this file is used for sortation by name of author.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />

<xsl:param name="type" select="'null'" />
<xsl:param name="sort_order" select="'null'" />

<xsl:template match="/">

<html>
<head>
</head>
<body>
<div>

<xsl:for-each select="theses/thesis">
<xsl:sort select="authors/author/lastname" data-type="text" order="{$sort_order}" />
<xsl:sort select="authors/author/firstname" data-type="text" order="{$sort_order}" />


<xsl:choose>
<xsl:when test="$type='0'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
```

```
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
</xsl:when>



<xsl:when test="$type='1'">
        <xsl:if test="@type='bachelor'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
```

```
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>



<xsl:when test="$type='2'">
        <xsl:if test="@type='diploma'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
```

```
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>


<xsl:when test="$type='3'">
        <xsl:if test="@type='seminar'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
```

```
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>
</xsl:choose>

</xsl:for-each>
</div>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

sort_title.xsl

This file is comparable to index.xsl. Whereas the index.xsl file is used for sortation by time, this file is used for sortation by title of thesis.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />

<xsl:param name="type" select="'null'" />
<xsl:param name="sort_order" select="'null'" />

<xsl:template match="/">
<html>
<head>
</head>
<body>
<div>

<xsl:for-each select="theses/thesis">
<xsl:sort select="title" data-type="text" order="{$sort_order}" />

<xsl:choose>
<xsl:when test="$type='0'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
```

```
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
</xsl:when>


<xsl:when test="$type='1'">
        <xsl:if test="@type='bachelor'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
```

```
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>


<xsl:when test="$type='2'">
        <xsl:if test="@type='diploma'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
```

```
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>



<xsl:when test="$type='3'">
        <xsl:if test="@type='seminar'">
        <table width="100%">
        <xsl:attribute name="class">
        <xsl:value-of select="@type"/>
        </xsl:attribute>
        <tr>
        <td class="date" width="5%">
        <xsl:value-of select="date/year"/> <xsl:value-of select="date/month"/>
        </td>
        <td class="author" width="70%">
        <span class="author">
        <xsl:for-each select="authors/author">
        <xsl:value-of select="lastname"/>, <xsl:value-of select="firstname"/>
        <span class="mothertongue"> (mother tongue: <xsl:value-of se-
lect="mothertongue"/>)</span>
        <br />
        </xsl:for-each>
        </span>
        <xsl:value-of select="university"/>
        </td>
        <td class="language" width="25%">
        <xsl:value-of select="@type"/> / <xsl:value-of select="language"/>
        </td>
        </tr>
        <tr>
        <td class="empty">
        </td>
        <td colspan="2" class="title">
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="link" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </td>
        </tr>
        <tr>
        <td class="empty">
```

```
        </td>
        <td colspan="2" class="abstract" lang="en">
        <i>
        <xsl:for-each select="abstract/paragraph">
        <p><xsl:value-of select="." /></p>
        </xsl:for-each>
        </i>
        <xsl:if test="downloads/download">
        Extra Materials:
        <xsl:for-each select="downloads/download">
        <br />
        <a>
        <xsl:attribute name="href">
        <xsl:value-of select="src" />
        </xsl:attribute>
        <xsl:value-of select="title" />
        </a>
        </xsl:for-each>
        </xsl:if>
        <xsl:if test="keywords/keyword">
        <br /><br />
        Keywords:
        <br />
        <code>
        <xsl:for-each select="keywords/keyword">
        <xsl:sort select="." data-type="text" order="ascending" />
        <xsl:value-of select="." /> 
        </xsl:for-each>
        </code>
        </xsl:if>
        </td>
        </tr>
        </table>
        <br />
        </xsl:if>
</xsl:when>
</xsl:choose>

</xsl:for-each>
</div>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

1key.xsl

This XSL stylesheet and the following 4 (2keys.xsl, 3keys.xsl, 4keys.xsl and 5keys.xsl) are used for the search for keywords. If the user has chosen only one keyword, this file is used, if he/she has chosen two keys the file 2keys.xsl is used and so on.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:param name="keyword" select="'null'" />
<xsl:template match="/">
        <html>
                <head>
                </head>
                <body>
                        <div>
                        <xsl:for-each select="theses/thesis[keywords/keyword=$keyword]">
                                <xsl:sort select="date/year" data-type="number" or-
der="descending" />
                                <xsl:sort select="date/month" data-type="number" or-
der="descending" />
                                <table width="100%">
                                <xsl:attribute name="class">
                                <xsl:value-of select="@type"/>
                                </xsl:attribute>
                                <tr>
                                <td class="date" width="5%">
                                        <xsl:value-of select="date/year"/> <xsl:value-of
select="date/month"/>
                                </td>
                                <td class="author" width="70%">
                                        <span class="author">
                                        <xsl:for-each select="authors/author">
                                        <xsl:value-of select="lastname"/>, <xsl:value-of
select="firstname"/> (mother tongue: <xsl:value-of select="mothertongue"/>)
                                        <br />
                                        </xsl:for-each>
                                        </span>
                                        <xsl:value-of select="university"/>
                                </td>
                                <td class="language" width="25%">
                                        <xsl:value-of select="@type"/> / <xsl:value-of se-
lect="language"/>
                                </td>
                                </tr>
                                <tr>
                                <td class="empty">
                                </td>
                                <td colspan="2" class="title">
                                        <a>
                                        <xsl:attribute name="href">
                                        <xsl:value-of select="link" />
                                        </xsl:attribute>
                                        <xsl:value-of select="title" />
                                        </a>
                                </td>
                                </tr>
                                <tr>
                                <td class="empty">
                                </td>
                                <td colspan="2" class="abstract" lang="en">
                                        <i>
                                        <xsl:for-each select="abstract/paragraph">
                                        <p><xsl:value-of select="." /></p>
                                        </xsl:for-each>
                                        </i>
```

```
                                    <xsl:if test="downloads/download">
                                    Extra Materials:
                                    <xsl:for-each select="downloads/download">
                                    <br />
                                    <a>
                                    <xsl:attribute name="href">
                                    <xsl:value-of select="src" />
                                    </xsl:attribute>
                                    <xsl:value-of select="title" />
                                    </a>
                                    </xsl:for-each>
                                    </xsl:if>
                                    <xsl:if test="keywords/keyword">
                                    <br /><br />
                                    Keywords:
                                    <br />
                                    <code>
                                    <xsl:for-each select="keywords/keyword">
                                    <xsl:sort select="." data-type="text" order="ascending"
/>
                                    <xsl:value-of select="." /> 
                                    </xsl:for-each>
                                    </code>
                                    </xsl:if>
                                </td>
                                </tr>
                                </table>
                                <br />
                            </xsl:for-each>
                            </div>
                    </body>
            </html>
</xsl:template>
</xsl:stylesheet>
```

2keys.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:param name="keyword" select="'null'" />
<xsl:param name="keyword2" select="'null'" />
<xsl:template match="/">
        <html>
                <head>
                </head>
                <body>
                <div>
                <xsl:for-each select="theses/thesis[keywords/keyword=$keyword and key-
words/keyword=$keyword2]">
                        <xsl:sort select="date/year" data-type="number" order="descending" />
                        <xsl:sort select="date/month" data-type="number" order="descending" />
                        <table width="100%">
                        <xsl:attribute name="class">
                        <xsl:value-of select="@type"/>
                        </xsl:attribute>
                        <tr>
                        <td class="date" width="5%">
                                <xsl:value-of select="date/year"/> <xsl:value-of se-
lect="date/month"/>
                        </td>
                        <td class="author" width="70%">
                                <span class="author">
                                <xsl:for-each select="authors/author">
                                <xsl:value-of select="lastname"/>, <xsl:value-of se-
lect="firstname"/> (mother tongue: <xsl:value-of select="mothertongue"/>)
                                <br />
                                </xsl:for-each>
                                </span>
                                <xsl:value-of select="university"/>
                        </td>
                        <td class="language" width="25%">
                                <xsl:value-of select="@type"/> / <xsl:value-of se-
lect="language"/>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="title">
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="link" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                                </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="abstract" lang="en">
                                <i>
                                <xsl:for-each select="abstract/paragraph">
                                <p><xsl:value-of select="." /></p>
                                </xsl:for-each>
                                </i>
                                <xsl:if test="downloads/download">
                                Extra Materials:
                                <xsl:for-each select="downloads/download">
                                <br />
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="src" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
```

```
                                </xsl:for-each>
                                </xsl:if>
                                <xsl:if test="keywords/keyword">
                                <br /><br />
                                Keywords:
                                <br />
                                <code>
                                <xsl:for-each select="keywords/keyword">
                                <xsl:sort select="." data-type="text" order="ascending" />
                                <xsl:value-of select="." /> 
                                </xsl:for-each>
                                </code>
                                </xsl:if>
                        </td>
                        </tr>
                        </table>
                        <br />
                </xsl:for-each>
                </div>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

3keys.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:param name="keyword" select="'null'" />
<xsl:param name="keyword2" select="'null'" />
<xsl:param name="keyword3" select="'null'" />
<xsl:template match="/">
        <html>
                <head>
                </head>
                <body>
                <div>
                <xsl:for-each select="theses/thesis[keywords/keyword=$keyword and key-
words/keyword=$keyword2 and keywords/keyword=$keyword3]">
                        <xsl:sort select="date/year" data-type="number" order="descending" />
                        <xsl:sort select="date/month" data-type="number" order="descending" />
                        <table width="100%">
                        <xsl:attribute name="class">
                        <xsl:value-of select="@type"/>
                        </xsl:attribute>
                        <tr>
                        <td class="date" width="5%">
                                <xsl:value-of select="date/year"/> <xsl:value-of se-
lect="date/month"/>
                        </td>
                        <td class="author" width="70%">
                                <span class="author">
                                <xsl:for-each select="authors/author">
                                <xsl:value-of select="lastname"/>, <xsl:value-of se-
lect="firstname"/> (mother tongue: <xsl:value-of select="mothertongue"/>)
                                <br />
                                </xsl:for-each>
                                </span>
                                <xsl:value-of select="university"/>
                        </td>
                        <td class="language" width="25%">
                                <xsl:value-of select="@type"/> / <xsl:value-of se-
lect="language"/>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="title">
                                <a>
                                <xsl:attribute name="href">
```

```
                                <xsl:value-of select="link" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="abstract" lang="en">
                                <i>
                                <xsl:for-each select="abstract/paragraph">
                                <p><xsl:value-of select="." /></p>
                                </xsl:for-each>
                                </i>
                                <xsl:if test="downloads/download">
                                Extra Materials:
                                <xsl:for-each select="downloads/download">
                                <br />
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="src" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                                </xsl:for-each>
                                </xsl:if>
                                <xsl:if test="keywords/keyword">
                                <br /><br />
                                Keywords:
                                <br />
                                <code>
                                <xsl:for-each select="keywords/keyword">
                                <xsl:sort select="." data-type="text" order="ascending" />
                                <xsl:value-of select="." /> 
                                </xsl:for-each>
                                </code>
                                </xsl:if>
                        </td>
                        </tr>
                        </table>
                        <br />
                </xsl:for-each>
                </div>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

4keys.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:param name="keyword" select="'null'" />
<xsl:param name="keyword2" select="'null'" />
<xsl:param name="keyword3" select="'null'" />
<xsl:param name="keyword4" select="'null'" />
<xsl:template match="/">
        <html>
                <head>
                </head>
                <body>
                <div>
                <xsl:for-each select="theses/thesis[keywords/keyword=$keyword and key-
words/keyword=$keyword2 and keywords/keyword=$keyword3 and keywords/keyword=$keyword4]">
                        <xsl:sort select="date/year" data-type="number" order="descending" />
                        <xsl:sort select="date/month" data-type="number" order="descending" />
                        <table width="100%">
                        <xsl:attribute name="class">
                        <xsl:value-of select="@type"/>
                        </xsl:attribute>
                        <tr>
                        <td class="date" width="5%">
                                <xsl:value-of select="date/year"/> <xsl:value-of se-
lect="date/month"/>
                        </td>
                        <td class="author" width="70%">
                                <span class="author">
                                <xsl:for-each select="authors/author">
                                <xsl:value-of select="lastname"/>, <xsl:value-of se-
lect="firstname"/> (mother tongue: <xsl:value-of select="mothertongue"/>)
                                <br />
                                </xsl:for-each>
                                </span>
                                <xsl:value-of select="university"/>
                        </td>
                        <td class="language" width="25%">
                                <xsl:value-of select="@type"/> / <xsl:value-of se-
lect="language"/>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="title">
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="link" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="abstract" lang="en">
                                <i>
                                <xsl:for-each select="abstract/paragraph">
                                <p><xsl:value-of select="." /></p>
                                </xsl:for-each>
                                </i>
                                <xsl:if test="downloads/download">
                                Extra Materials:
                                <xsl:for-each select="downloads/download">
                                <br />
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="src" />
                                </xsl:attribute>
```

```
                                <xsl:value-of select="title" />
                                </a>
                                </xsl:for-each>
                                </xsl:if>
                                <xsl:if test="keywords/keyword">
                                <br /><br />
                                Keywords:
                                <br />
                                <code>
                                <xsl:for-each select="keywords/keyword">
                                <xsl:sort select="." data-type="text" order="ascending" />
                                <xsl:value-of select="." /> 
                                </xsl:for-each>
                                </code>
                                </xsl:if>
                        </td>
                        </tr>
                        </table>
                        <br />
                </xsl:for-each>
                </div>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

5keys.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:param name="keyword" select="'null'" />
<xsl:param name="keyword2" select="'null'" />
<xsl:param name="keyword3" select="'null'" />
<xsl:param name="keyword4" select="'null'" />
<xsl:param name="keyword5" select="'null'" />
<xsl:template match="/">
        <html>
                <head>
                </head>
                <body>
                <div>
                <xsl:for-each select="theses/thesis[keywords/keyword=$keyword and key-
words/keyword=$keyword2 and keywords/keyword=$keyword3 and keywords/keyword=$keyword4 and key-
words/keyword=$keyword5]">
                        <xsl:sort select="date/year" data-type="number" order="descending" />
                        <xsl:sort select="date/month" data-type="number" order="descending" />
                        <table width="100%">
                        <xsl:attribute name="class">
                        <xsl:value-of select="@type"/>
                        </xsl:attribute>
                        <tr>
                        <td class="date" width="5%">
                                <xsl:value-of select="date/year"/> <xsl:value-of se-
lect="date/month"/>
                        </td>
                        <td class="author" width="70%">
                                <span class="author">
                                <xsl:for-each select="authors/author">
                                <xsl:value-of select="lastname"/>, <xsl:value-of se-
lect="firstname"/> (mother tongue: <xsl:value-of select="mothertongue"/>)
                                <br />
                                </xsl:for-each>
                                </span>
                                <xsl:value-of select="university"/>
                        </td>
                        <td class="language" width="25%">
                                <xsl:value-of select="@type"/> / <xsl:value-of se-
lect="language"/>
                        </td>
                        </tr>
                        <tr>
```

```
                        <td class="empty">
                        </td>
                        <td colspan="2" class="title">
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="link" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                        </td>
                        </tr>
                        <tr>
                        <td class="empty">
                        </td>
                        <td colspan="2" class="abstract" lang="en">
                                <i>
                                <xsl:for-each select="abstract/paragraph">
                                <p><xsl:value-of select="." /></p>
                                </xsl:for-each>
                                </i>
                                <xsl:if test="downloads/download">
                                Extra Materials:
                                <xsl:for-each select="downloads/download">
                                <br />
                                <a>
                                <xsl:attribute name="href">
                                <xsl:value-of select="src" />
                                </xsl:attribute>
                                <xsl:value-of select="title" />
                                </a>
                                </xsl:for-each>
                                </xsl:if>
                                <xsl:if test="keywords/keyword">
                                <br /><br />
                                Keywords:
                                <br />
                                <code>
                                <xsl:for-each select="keywords/keyword">
                                <xsl:sort select="." data-type="text" order="ascending" />
                                <xsl:value-of select="." /> 
                                </xsl:for-each>
                                </code>
                                </xsl:if>
                        </td>
                        </tr>
                        </table>
                        <br />
                </xsl:for-each>
                </div>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

keywords.xsl

This file is used for displaying a list of all keywords, which can be found in the index.xml file. It separates all keywords by a colon.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="html" />
        <xsl:template match="/">
        <xsl:for-each select="theses/thesis/keywords/keyword">
                <xsl:sort select="keyword" data-type="text" order="ascending" /><xsl:value-of
select="." />:</xsl:for-each>
        </xsl:template>
</xsl:stylesheet>
```

authors.xsl

This file is used for displaying a list of all authors, which can be found in the index.xml file. It separates all authors by a colon.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:template match="/">
        <xsl:for-each select="theses/thesis/authors/author">
        <xsl:sort select="lastname" data-type="text" order="ascending" /><xsl:value-of se-
lect="lastname" />, <xsl:value-of select="firstname" />:</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

style.css

This is the Cascading Style Sheet file. It is called from the index.html file and responsible for the formatting of the displayed items and the navigation.

```
body {
font-family:Arial, sans-serif;
font-size:14px;
}

.navtitle {
font-size:140%;
font-weight:bold;
padding-bottom:10px;
}

div.navlink {
cursor:pointer;
}

span.navlink {
color:blue;
cursor:pointer;
text-decoration:underline;
font-size:16px;
}

span.navlink:hover {
```

```
text-decoration:none;
font-size:16px;
cursor:pointer;
}

b.navlink {
cursor:pointer;
font-size:16px;
}

#all {
width:160px;
height:25px;
line-height:25px;
padding-left:8px;
border-left:4px solid #000;
}

#all2 {
width:160px;
height:25px;
line-height:25px;
padding-left:5px;
border-left:7px solid #000;
font-weight:bold;
}

#all:hover, #all2:hover  {
background-color:#000;
color:#FFF;
cursor:pointer;
}

#dip {
width:160px;
height:25px;
line-height:25px;
padding-left:8px;
border-left:4px solid #528CAE;
}

#dip2 {
width:160px;
height:25px;
line-height:25px;
padding-left:5px;
border-left:7px solid #528CAE;
font-weight:bold;
}

#dip:hover, #dip2:hover {
background-color:#528CAE;
color:#FFF;
cursor:pointer;
}

#sem {
width:160px;
height:25px;
line-height:25px;
padding-left:8px;
border-left:4px solid #71945B;
}

#sem2 {
width:160px;
height:25px;
line-height:25px;
padding-left:5px;
border-left:7px solid #71945B;
font-weight:bold;
}

#sem:hover, #sem2:hover {
background-color:#71945B;
```

```
color:#FFF;
cursor:pointer;
}

#bac {
width:160px;
height:25px;
line-height:25px;
padding-left:8px;
border-left:4px solid #C65A5A;
}

#bac2 {
width:160px;
height:25px;
line-height:25px;
padding-left:5px;
border-left:7px solid #C65A5A;
font-weight:bold;
}

#bac:hover, #bac2:hover  {
background-color:#C65A5A;
color:#FFF;
cursor:pointer;
}

div#currently {
font-size:150%;
margin-bottom:15px;
}

table {
color                : black ;
background-color     : #e8e8e8 ;
font-family          : sans-serif ;
}

table.diploma {
border-top:3px solid #528CAE;
}

table.bachelor {
border-top:3px solid #C65A5A;
}

table.seminar {
border-top:3px solid #71945B;
}

h1.diploma {
color:#528CAE;
}

h1.bachelor {
color:#C65A5A;
}

h1.seminar {
color:#71945B;
}

div.diploma {
border-left:3px solid #528CAE;
padding-left:10px;
}

div.bachelor {
border-left:3px solid #C65A5A;
padding-left:10px;
}

div.seminar {
border-left:3px solid #71945B;
padding-left:10px;
```

```
}

td.empty {
background-color:#e8e8e8;
}

td.language {
font-weight:bolder;
text-align:right;
}

td.text {
font-weight:normal;
}

td.title {
font-weight:bold;
font-size:125%;
color:maroon;
}

td.title a:hover {
text-decoration:none;
}

td.abstract {
font-size:14px;
}

span.text {
font-weight:normal;
font-size:90%;
}

span.author {
font-weight:bold;
font-size:125%;
color:maroon;
}

span.mothertongue {
font-weight:normal;
font-size:14px;
color:#000;
}

code {
font-size:12px;
color:#444444;
}

.current_keyword {
font-weight:bold;
color:red;
}
```

schema.dtd

This is the Document Type Definition for the index.xml file. It contains constraints concerning the names of elements and attributes and the number of occurences. The index.xml file is well-formed and valid.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT theses (thesis+)>
<!ELEMENT thesis ((date, authors, university, language, title, link, abstract) | (date, authors,
university, language, title, link, downloads, abstract) | (date, authors, university, language,
title, link, keywords) | (date, authors, university, language, title, link, abstract, keywords)
| (date, authors, university, language, title, link, downloads, abstract, keywords))>
<!ATTLIST thesis
          type CDATA #REQUIRED>
<!ELEMENT date ((year) | (year, month))>
<!ELEMENT year (#PCDATA)>
<!ELEMENT authors (author+)>
<!ELEMENT author (lastname, firstname, mothertongue)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT mothertongue (#PCDATA)>
<!ELEMENT university (#PCDATA)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT link (#PCDATA)>
<!ELEMENT abstract (paragraph+)>
<!ELEMENT paragraph (#PCDATA | a | b | br | ul)*>
<!ELEMENT downloads (download+)>
<!ELEMENT download (title, src)>
<!ATTLIST download
          type CDATA #REQUIRED>
<!ELEMENT src (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT keywords (keyword+)>
<!ELEMENT keyword (#PCDATA)>
<!ELEMENT a (#PCDATA)>
<!ATTLIST a
          href CDATA #REQUIRED>
<!ELEMENT b (#PCDATA)>
<!ELEMENT br EMPTY>
<!ELEMENT ul (li+)>
<!ELEMENT li (#PCDATA)>
```

References

[SCHja]	w3schools: JavaScript Tutorial http://www.w3schools.com/js/ equested on 18.12.2011
[SCHXbr]	w3schools: XSLT Browsers http://www.w3schools.com/xsl/xsl_browsers.asp requested on 18.12.2011
[SCHXc]	w3schools: XSLT <xsl:choose> Element http://www.w3schools.com/xsl/xsl_choose.asp requested on 18.12.2011
[SCHXf]	w3schools: XSLT <xsl:for-each> Element http://www.w3schools.com/xsl/xsl_for_each.asp requested on 18.12.2011
[SCHXi]	w3schools: XSLT <xsl:if> Element http://www.w3schools.com/xsl/xsl_if.asp requested on 18.12.2011
[SCHXP]	w3schools: XPath Tutorial http://www.w3schools.com/xpath/ requested on 18.12.2011
[SCHXpar]	w3schools: XSLT <xsl:param> Element http://www.w3schools.com/xsl/el_param.asp requested on 18.12.2011
[SCHXs]	w3schools: XSLT <xsl:sort> Element http://www.w3schools.com/xsl/el_sort.asp requested on 18.12.2011
[SCHXSE]	w3schools: XSLT Elements Reference http://www.w3schools.com/xsl/xsl_w3celementref.asp requested on 18.12.2011
[SCHXST]	w3schools: XSLT - Transformation http://www.w3schools.com/xsl/xsl_transformation.asp requested on 18.12.2011

[SCHXt]	w3schools: XSLT <xsl:template> Element http://www.w3schools.com/xsl/xsl_templates.asp requested on 18.12.2011
[SCHXv]	w3schools: XSLT <xsl:value-of> Element http://www.w3schools.com/xsl/xsl_value_of.asp requested on 18.12.2011
[SELFdo]	Verein SELFHTML e.V.: Objektreferenz document http://de.selfhtml.org/javascript/objekte/document.htm#allgemeines requested on 18.12.2011
[SELFja]	Verein SELFHTML e.V.: Einführung in JavaScript und DOM http://de.selfhtml.org/javascript/intro.htm requested on 18.12.2011
[W3CSG]	World Wide Web Consortium: SGML reference information for HTML http://www.w3.org/TR/html401/sgml/intro.html requested on 18.12.2011
[W3CX10]	World Wide Web Consortium: XML in 10 points http://www.w3.org/XML/1999/XML-in-10-points.html.en requested on 18.12.2011
[W3CXDt]	World Wide Web Consortium: Extensible Markup Language 1.0 specification - Document Type Declaration http://www.w3.org/TR/REC-xml/#dt-valid requested on 18.12.2011
[W3CXML]	World Wide Web Consortium: Extensible Markup Language 1.0 specification http://www.w3.org/TR/REC-xml/ requested on 18.12.2011
[W3CXOr]	World Wide Web Consortium: XML 1.1 specification - Origin and Goals of XML http://www.w3.org/TR/xml11/#sec-origin-goals requested on 18.12.2011

[W3CXP]	World Wide Web Consortium: XML Path Language (XPath) Version 1.0 specification http://www.w3.org/TR/xpath/ requested on 18.12.2011
[W3CXP2]	World Wide Web Consortium: XML Path Language (XPath) 2.0 (Second Edition) http://www.w3.org/TR/xpath20/ requested on 18.12.2011
[W3CXS]	World Wide Web Consortium: Extensible Stylesheet Language (XSL) Version 1.1 specification http://www.w3.org/TR/xsl/ requested on 18.12.2011
[W3CXSc]	World Wide Web Consortium: XML Schema Part 1: Structures Second Edition - Overview of XML Schema http://www.w3.org/TR/xmlschema-1/#d0e504 requested on 18.12.2011
[W3CXSP]	World Wide Web Consortium: Extensible Stylesheet Language (XSL) Version 1.1 specification - Processing a Stylesheet http://www.w3.org/TR/xsl/#d0e147 requested on 18.12.2011
[W3CXT]	World Wide Web Consortium: XSL Transformations (XSLT) Version 1.0 specification http://www.w3.org/TR/xslt requested on 18.12.2011
[W3CXv]	World Wide Web Consortium: Rationale and list of changes for XML 1.1 http://www.w3.org/TR/xml11/#sec-xml11 requested on 18.12.2011
[W3CXwf]	World Wide Web Consortium: Extensible Markup Language 1.0 specification - Well-Formed XML Documents http://www.w3.org/TR/REC-xml/#sec-well-formed requested on 18.12.2011

[WikiXD]	Wikipedia: Document Type Definition, Section 3 XML DTDs and schema validation http://en.wikipedia.org/w/index.php?title=Document_Type_Definition&oldid=456264051 requested on 18.12.2011
[WikiXS]	Wikipedia: XML, Section 4 Schemas and validation http://en.wikipedia.org/w/index.php?title=XML&oldid=466311081 requested on 18.12.2011
[WikiXv]	Wikipedia: XML, Section 8.2 Versions http://en.wikipedia.org/w/index.php?title=XML&oldid=465988993 requested on 18.12.2011
[WikiXw]	Wikipedia: XML, Section 3 Well-formedness and error-handling http://en.wikipedia.org/w/index.php?title=XML&oldid=466311081 requested on 18.12.2011
[WUrgfd]	Institute for Management Information Systems at the Vienna University of Economics and Business: Ausgewählte Diplom-, Master-, Bakkarbeiten http://wi.wu.ac.at/rgf/diplomarbeiten/ requested on 18.12.2011