Introduction to XML for ETDs

© Matthias Schulz, Sabine Henneberger, Susanne Dobratz
Humboldt University Berlin at ETD2003, 20. May 2003 Berlin, Germany

Table of Contents

© Matthias Schulz, Sabine Henneberger, Susanne Dobratz
Humboldt University Berlin at ETD2003, 20. May 2003 Berlin, Germany

Introduction to XML

What is the eXtensible Markup Language?

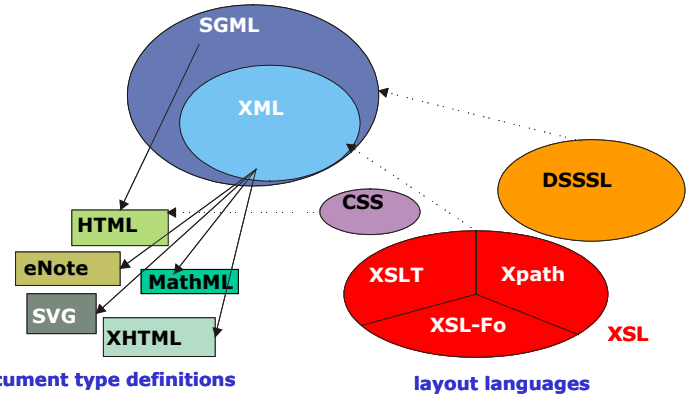
The eXtensible Markup Language (**XML**) is a markup or structuring language for documents, a so-called metalanguage that defines rules for the structural markup of documents independently from any output media. XML is a "reduced" version of the Structured Generalized Markup Language (**SGML**), which has been an ISO-certified standard since 1986. In the field of Internet publishing, SGML never achieved wide success due to the complexity of the standard and the high cost of the tools. It prevailed only in certain areas, such as technical documentation in large enterprises (Boeing, patent information). The main philosophy of SGML and XML is the strict separation of content, structure and layout of documents. The above-mentioned projects use either the SGML standard (ISO 8879 with Korregendum K vom 4.12.1997) or the definition of the World Wide Web Consortium (W3C) XML 1.0 (10.02.1998, revised 6.10.2000).

General Concept

XML is

- a meta language for describing data, regardless of what the data is.
- a simple, open, fully extensible standard for data.
- XML is a set of rules you can use to create your own markup language.
- a way to separate data from its display properties.
- a better, faster, cheaper and much more flexible way of dealing with data.
- a way to free applications from data storage architecture characteristics.
- a standard way to describe all possible data, serialized and optimized for use over the Web.
- a facility for universal data exchange using a vendor-independent data format understood by browsers.

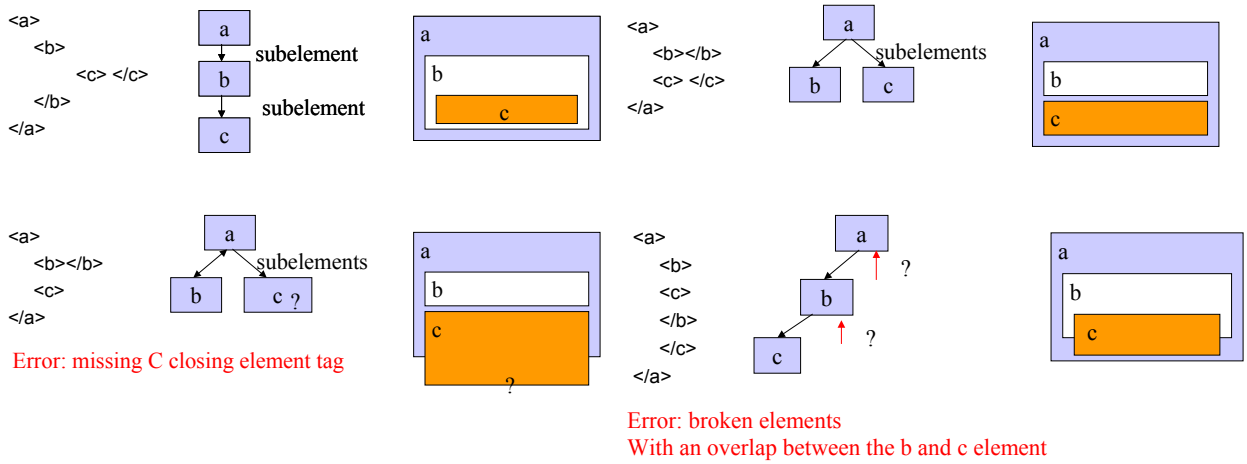
Elements can contain other elements, creating hierarchical structures. The hierarchical structures of XML were originally designed to support the structuring of traditional documents: sections, subsections, paragraphs, and so on. But the mechanism is general enough that it can be applied to data of any type. Given machine-readable document type rules, XML documents can be validated for conformance to these rules. This allows processors to ensure that the documents it is processing meet at least the syntactic requirements of the document type (there may be other requirements that cannot be specified in a DTD or schema).
XML consists of
(1) XML1.0 language definition
(2) Style Sheet Language XSL (eXtensible Style Language)



(3) Linking Component XLL (eXtensible Linking Language) = Xlink + Xpointer

© Matthias Schulz, Sabine Henneberger, Susanne Dobratz
Humboldt University Berlin at ETD2003, 20. May 2003 Berlin, Germany

XML Basic philosophy is: strictly separate content, structure and layout
What do you need to "write" and "read" XML?Minimum: Text editor and XML parser (e.g. msxml-Parser by Microsoft) and XML-Browser (XML,XSLT) (e.g. Internet Explorer 6).
Extended: XML-Editor (e.g. XML Spy by Altova) (http://www.xmlspy.com) and XML-Browser with XSLT (IE 6) and a XSL:Fo Formatter (Antenna by Antennahouse) (http://www.antennahouse.com).
A Tag is a part of the document markup separated from the rest of the content by < and >. It contains the name of the element. An element is always enclosed within an Start-Tag *<elementname>* and an End-Tag *</elementname>*. It can contain subelements that form a hierarchy:
<author><first>Susanne</first><last>Dobratz</last></author>.
XML documents are hierarchical trees:



First Example XML document

```
<?xml version="1.0" encoding="UTF-16"?>
<!DOCTYPE article SYSTEM "myarticle.dtd">
<article status="revised">
        <title>First Test</title>
        <author>
                <first>Susanne</first>
                <last>Dobratz</last>
        </author>
        <para>This is the first XML document as test. It shows how <key>formulas</key>like <math>2+2=4</math>can be tagged with XML.
        Testing UNICODE characters: This is UNICODE U00E4 (hexadecimal number): &#x00E4;. and this is UNICODE U0061: &#x0061; and U00A8: &#x00A8;. Even mathematical characters can be encoded in UNICODE:
&#x2200; x &#x2208; &#x2203; i.
        </para>
        <date>28.08.2002</date>
</article>
```

Document Type Definitions

A **document type definition (DTD)**, in the sense of XML, defines rules or templates that are used to produce similarly structured documents. A DTD describes the content model of a class of documents. It consists of:

To define a DTD, a special syntax is needed, which does not conform to the usual XML syntax where a document contains elements which are enclosed in "tags:" a start tag (e.g. <author>) and an end tag (e.g. </author>), producing code like this:

<author> Susanne Dobratz </author>.

Example DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % auth "first|last">
<!ELEMENT article (title, author, para, date)>
<!ATTLIST article
        status (draft | revised | final) #IMPLIED >
<!ELEMENT author ((%auth;), (%auth;))>
<!ELEMENT date (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT math (#PCDATA)>
<!ELEMENT para (#PCDATA | key | math)*>
<!ELEMENT title (#PCDATA)>
```

XML Schemas

Schemas are a different version to define formal rules for a document structure. They define element types, attributes that can appear in a document. XML Schemas are ion XML encoded, so it is easy to validate them. Their advantage in comparison to DTDs is that they use complex data types, e.g. Natural Numbers, to define the occurance of single elements and attribute values.

Schemas, like DTDs, define *syntactic* constraints. That is, they define what the names of elements and attributes are, what order they can appear, where they can appear, and what the spelling rules for attribute values and element content are (data types). But schemas cannot define more complex business rules.

A DTD could express this as follws:

```
<!ELEMENT address
(company?, name, street+, city, state, zip)>
```

Wheras a schema would do:

```
<elementType name="address">
<sequence>
<elementTypeRef name="company" minOccur="0" maxOccur="1"/>
<elementTypeRef name="name" minOccur="1" maxOccur="1"/>
<elementTypeRef name="street" minOccur="1" maxOccur="2"/>
<elementTypeRef name="city" minOccur="1" maxOccur="1"/>
<elementTypeRef name="state" minOccur="1" maxOccur="1"/>
<elementTypeRef name="zip" minOccur="1" maxOccur="1"/>
</sequence>
</elementType>
<datatype name="zipCode">
<basetype name="string"/>
<lexicalRepresentation>
<lexical>99999</lexical>
<lexical>99999-9999</lexical>
</lexicalRepresentation>
</datatype>
<elementType name="zip">
<datatypeRef name="zipCode"/>
</elementType>
```

XML's Style Language: the eXtensible Stylesheet Language

Part	Date	Status	URL
Extensible Stylesheet Language (XSL)	**15.10.2001**	**Version 1.0 W3C Recommendation**	**http://www.w3.org/TR/xsl/**
XSL Transfor-mations (XSLT)	**16.08.2002**	**Version 2.0 W3C Working Draft**	**http://www.w3.org/TR/xslt20/**
XSLT	**24.08.2001**	**Version 1.0 W3C WD**	**http://www.w3.org/TR/xslt11**
XML Path Language (XPath)	**16.08.2002**	**Version 2.0 W3C Working Draft**	**http://www.w3.org/TR/xpath20/**
XPath	**16.11.1999**	**Version 1.0 W3C Recom.**	**http://www.w3.org/TR/xpath**

eXtensible Stylesheet Language Transformations (XSLT)

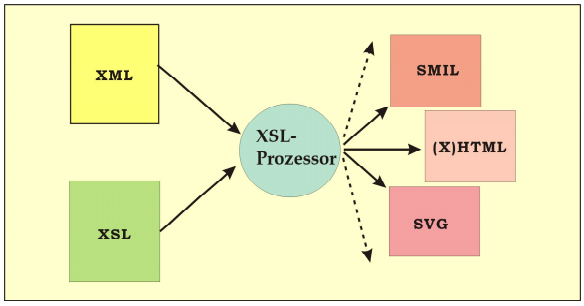
The eXtensible Stylesheet Language is a vocabulary for transforming XML documents into other formats like RTF, LaTeX, etc.. It contains:

- Formatting Vocabulary
- Query language for accessing XML documents

The advantage is that XSL statements are encoded into XML syntax!

Example XSLT document

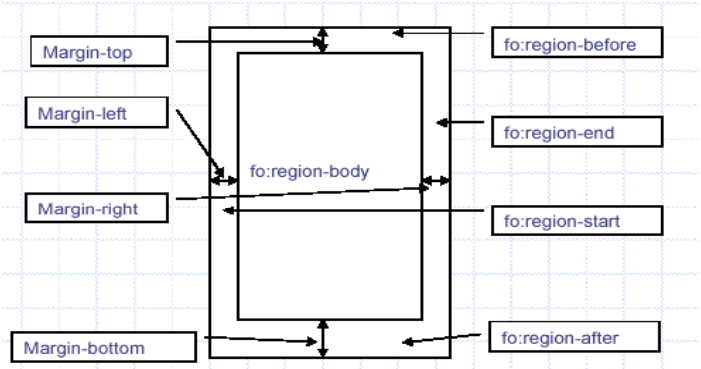
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
        <head>
        <link rel="stylesheet" type="text/css"
href="katalog1.css"/>
        </head>
        <body>
                <xsl:apply-templates select="article/*"/>
        </body>
   </html>
</xsl:template>

<xsl:template match="article">
        <xsl:apply-templates select="title"/>
        <xsl:apply-templates select="author"/>
        <xsl:apply-templates select="date"/>
        <xsl:apply-templates select="para"/>
</xsl:template>

<xsl:template match="title">
        <h1><xsl:value-of/></h1>
</xsl:template>

<xsl:template match="author">
        <h2><xsl:value-of/></h2>
</xsl:template>

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

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

<xsl:template match="key">
        <em>
                <xsl:value-of/>
        </em>
</xsl:template>

<xsl:template match="math">
        <b class="person">
                <xsl:value-of/>
        </b>
</xsl:template>

<xsl:template match="text()">
        <xsl:value-of/>
</xsl:template>

</xsl:stylesheet>
```



eXtensible Style sheet Language: Formatting Objects

The standard for creating high quality page-based and online renditions of data. An XML document type used to describe the contents of formatted pages or screens (Formatting Objects).

source: xsl standard W3C



```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="/">
        <xsl:processing-instruction
                name="cocoon-format">type="text/xslfo"
        </xsl:processing-instruction>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
        <fo:simple-page-master page-height="297mm"
                        page-width="210mm"
                        margin-top="10mm"
                        margin-bottom="8mm"
                        margin-left="15mm"
                        margin-right="15mm">
        <fo:region-body margin-bottom="19pt"/>
        <fo:region-after extent="15pt"/>
        </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence>

<fo:static-content flow-name="xsl-region-after">
<fo:block text-align-last="center" font-size="6pt">
        <fo:leader leader-length="5mm"
                leader-pattern="rule"
                line-height="70%"
                vertical-align="super"/> 
        <fo:page-number/> 
        <fo:leader leader-length="5mm"
                leader-pattern="rule"
                line-height="70%"
                vertical-align="super"/>
</fo:block>
</fo:static-content>

<fo:flow flow-name="xsl-region-body">
      <xsl:apply-templates/>
</fo:flow>
</fo:page-sequence>
</fo:root>
</xsl:template>
<!-- F O R M A T I N G ++++++++++++++++++++-->
<xsl:template match="dehio">
        <fo:block font-family="Times">
        <fo:block font-weight="bold">Beispiel Dehio
        </fo:block>
<xsl:apply-templates/>
</fo:block>
</xsl:template>


<xsl:template match="object">
        <xsl:apply-templates/>
</xsl:template>
<xsl:template match="dehio/object/otitle">
        <fo:inline font-weight="bold" color="brown">
                <xsl:apply-templates/>
        </fo:inline>
</xsl:template>
<xsl:template match="dehio/object/location">
        <fo:inline font-style="oblique">
                (<xsl:apply-templates/>)
        </fo:inline>
</xsl:template>
<xsl:template match="dehio/object/description/artist">
        <fo:inline font-weight="bold" color="green" >
                <xsl:apply-templates/>
        </fo:inline>
</xsl:template>
<xsl:template match="dehio/object/description/person">
        <fo:inline font-weight="bold" font-style="italic">
                <xsl:apply-templates/>
        </fo:inline>
</xsl:template>
<xsl:template match="dehio/object/description/ort">
        <fo:inline font-weight="bold" color="navy">
                <xsl:apply-templates/>
        </fo:inline>
</xsl:template>
<xsl:template match="dehio/object/description/datum">
        <fo:inline font-weight="bold" color="blue">
                <xsl:apply-templates/>
        </fo:inline>
</xsl:template>
<xsl:template match="description">
        <fo:inline>
                <xsl:apply-templates/>
        </fo:inline>
</xsl:template>
<xsl:template match="image">
        <fo:block space-before="12pt" space-after="12pt">
         <fo:external-graphic text-align="center"
                        src="{@file}"/></fo:block>
</xsl:template>
</xsl:stylesheet>
```

© Matthias Schulz, Sabine Henneberger, Susanne Dobratz
Humboldt University Berlin at ETD2003, 20. May 2003 Berlin, Germany

Writing your ETD in Microsoft Word

Dissertation Template - Techniques•Templates contain paragraph- and charactersstyles
•Macros, text modules and toolbars
•Templates have to be installed on the drafts-index

Template dissertation.dot

•75 paragraph templates
–headlines, marking
•113 character templates
–bold, cursive, formating bibliographies (worktitle)
•Macros
–reference manager
•toolbar
–format, bibliography

What is XdiML.DTD•XdiML (DissertationMarkupLanguage in XML)
•First DTD, the DiML.dtd in SGML out of the ETD-ML.dtd of Virginia Tech in 1997
•ETD-ML was developed by Yuri Rubinsky
(SGML Pioneer) and Neil Kipp
•The DTD have a document structure like Books.
–Root Element: etd, Childs: front, body, back and then chapter...
•The structure of DTD is module-based like TEI-DTD
–Modules for MathML and other special DTDs
–And for the main parts of the DTD (chapter...)

Connecting between
Dissertation97 and XdiML •Überschrift 1 (Heading 1)
•Standard (Normal)
•Unterstrichen (Underline)
••Every Style corresponds to an element of the DTD.
•Hierarchies are built up by the converter•<!ELEMENT Chapter (head..
•<!ELEMENT P (#PCDATA
•<!ELEMENT U (#PCDATA

Practice Menu Dissertation (thesis)•All essential Styles are arranged in the menu Dissertati
•Advantage
–the styles are availble at a glance
–The order corresponds to the structure of a thesis. (titlepage = <front), headings... = <body>, appendices = <back>

Steps•Download dissertation-97.dot
–Attention: important are the wordversion and operating system
•Installing
–Attention: Where is the template folder?
•Writing my ETD
–Attention: Not open the Word and write but open the menu New, choose the Dissertation template and then write

Problems•Approx. 50% of the doctoral student formats not from beginning with the template
–formating afterwards is expensively.

•Why?
–inadequate information
–knowledge about electronic publishing

Solutions•roadshow about electronic publishing
•training
•contact persons via Phone, Mail and personal call

Writing your ETD in OpenOffice

What means OOo?

- OOo stands for OpenOffice.org. This is an open source organization.
- "To create, as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component based APIs (Application Program Interface) and an XML-based file format."
- "OpenOffice.org is both an Open Source product and a project. The product is a multi-platform office productivity suite. It includes the key desktop applications, such as a word processor, spreadsheet, presentation manager, and drawing program, with a user interface and feature set similar to other office suites'. Sophisticated and flexible, OpenOffice.org also works transparently with a variety of file formats, including those of Microsoft Office."*
- "OpenOffice.org is an Open Source project. This means, first, that they offer not only a product but a process, and second, that they depend upon the contributions of developers and endusers to make that process happen. "*

About the product:

It consists of

- Writer (word processing)
- Calc (spreadsheet program)
- Draw (graphics and diagrams)
- Impress (multimedia presentations)

It runs on almost every modern operating system, also on Mac OS X.

The text file format is XML.

OpenOffice is an open source project: Everyone is allowed to download and use it for free.

http://www.openoffice.org

StarOffice

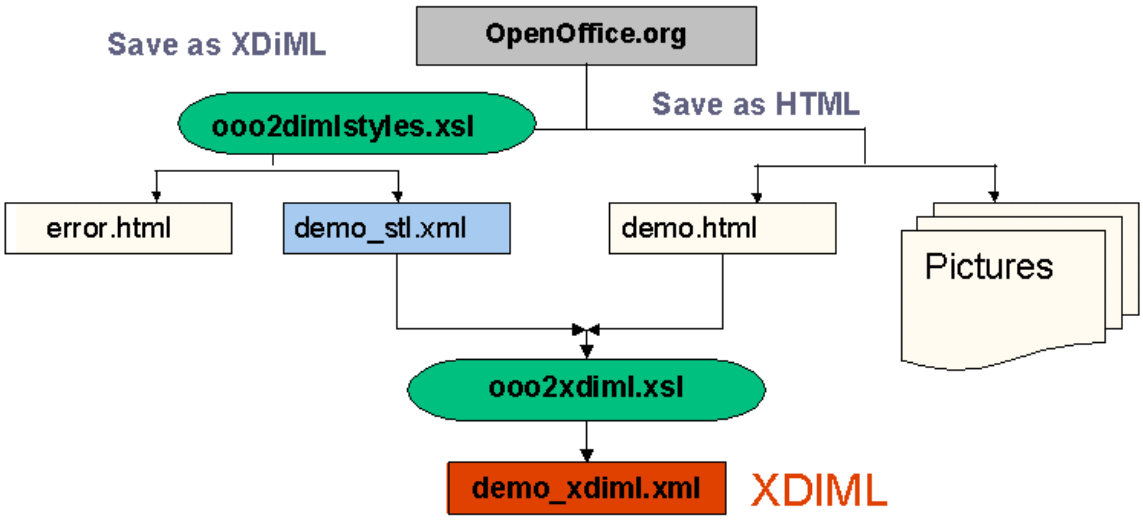
StarOffice is a product of Sun Microsystems. Newer versions of StarOffice (we have now StarOffice 6.0 Office Suite) are built using the OpenOffice.org source. That means the newer StarOffice versions have the same features like OOo.

Differences between StarOffice and OpenOffice
"StarOffice is not free, but academic and research institutions and Universities, are eligible for a no-cost site license of StarOffice 6.0."*
http://www.sun.com
OpenOffice is more sophisticated, more flexible and offers more possibilities for Extensions.
We use OpenOffice.org 1.1 Beta.
* citations from websites http://www.openoffice.org and http://www.sun.com

Tools for Writing Theses in OOo To adapt OOo for our needs, that means, for writing thesis or dissertation,we use 3 supplements:

- **Template** HU-Dissertationsvorlage (HU-Dissertationsvorlage.stw in \user\templates), required
- **Menu** "Dissertationen" (writermenubar.xml in \user\config\soffice.cfg), optional
- **Filter** xdiml (xdiml.jar in \program\classes, TypeDetection.xcu in \share\registry\data\org\openoffice\Office, Xalan-j_2_4_1 or newer), optional

How to get from OOo to XDiML?



The OpenOffice file format is xml.
sxw is an archive. Open it with zip!

- Take the xml.structure of the sxw-file (OOo file).
- Transform this with XSLT.

XSLT in Practical Use

demo_ooo.xml

```
<office:document-content…>
…
<office:body>
        <text:sequence-decls>
            <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
            <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
            <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
            <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
            <text:sequence-decl text:display-outline-level="0" text:name="Abbildung"/>
            <text:sequence-decl text:display-outline-level="0" text:name="Tabelle"/>
        </text:sequence-decls>
        <text:p text:style-name="P1">Aus der Klinik und Poliklinik für Chirurgie<text:line-break/>der
Medizinischen Fakultät Charité<text:line-break/>der Humboldt-Universität zu Berlin</text:p>
        <text:p text:style-name="Dokumenttyp">Dissertation</text:p>
        <text:p text:style-name="P3">„Reflux im Kleinkindalter – Zeitpunkt der Operation“</text:p>
```

XSLT: ooo2dimlstyles.xsl

- Deletes unnecessary parts
- Assigns the xml-tags with the style
- Copies elements

```
….
<xsl:template match="office:body">
        <office:body>
            <xsl:apply-templates/>
        </office:body>
    </xsl:template>

<xsl:template match="text:sequence-decls"/>

<xsl:template match="text:line-break">
<xsl:copy select="."/>
</xsl:template>

<xsl:template match="*[@text:style-name]">
…
```

demo_stl.xml

```
<office:document>
…
<office:body>
        <text:p text:style-name="Einrichtung" tag="school" diml_parent="deckblatt">Aus der Klinik und
Poliklinik für Chirurgie<text:line-break/>der Medizinischen Fakultät Charité<text:line-break/>der
Humboldt-Universität zu Berlin</text:p>
        <text:p text:style-name="Dokumenttyp" tag="submission"
diml_parent="deckblatt">Dissertation</text:p>
        <text:p text:style-name="Thema" tag="title" diml_parent="deckblatt">„Reflux im Kleinkindalter –
Zeitpunkt der Operation“</text:p>
```

© Matthias Schulz, Sabine Henneberger, Susanne Dobratz
Humboldt University Berlin at ETD2003, 20. May 2003 Berlin, Germany

XSLT: ooo2xdiml.xsl

- Transforms into XDiML

```
…
<xsl:template match="/office:document-content/office:body">
        ...
        <etd>
            <front>
                <xsl:apply-templates select="*" mode="deckblatt"/>
                <xsl:apply-templates select="*[@diml_parent='front']" mode="front_back"/>
            </front>
            <body>
                <xsl:apply-templates select="text:h[1]" mode="body"/>
            </body>
            <back>
                <xsl:apply-templates select="text:p[@diml_parent='back']" mode="front_back">
                        </xsl:apply-templates>
            </back>
        </etd>
    </xsl:template>

<xsl:template match="text:p|text:span">
<xsl:element name="{@tag}">
                    <xsl:apply-templates/>
                </xsl:element>
</xsl:template>
<xsl:template match="text:line-break">
                    <br/>
            </xsl:template>
```

demo_xdiml.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE etd PUBLIC "-//HUB//DTD Electronic Theses and Dissertations Version DiML 2.0//EN"
"C:\DTD\xdiml.dtd" [
    <!ENTITY Graphic2 SYSTEM "Heinz_html_m23ca5230.gif" NDATA GIF>
]>
<etd>
    <front>
        <school>Aus der Klinik und Poliklinik für Chirurgie<br/>der Medizinischen Fakultät
Charité<br/>der Humboldt-Universität zu Berlin</school>
        <submission>Dissertation</submission>
        <title>„Reflux im Kleinkindalter – Zeitpunkt der Operation“</title>
```

Making XSLT work

To apply the XSLT we need a Parser.
The Parser reads the xml file, analyses it and transforms it with XSLT in the new structure.
We actually use the Xalan-Parser for the Filter.
Xalan is a prerequisite for establishing the filter.
We use the saxon parser for the second transformation and we can use it alternative to the filter for the first one to.
Windows:saxon demo_stl.xml ooo2xdiml.xsl >demo_xdiml.xml
For both parsers you need to install Java Runtime Environment. All of these programs you can download for free.

http://xml.apache.org/xalan-j/
http://saxon.sourceforge.net/
http://java.sun.com/

Contact:

Humboldt-Universität zu Berlin
Computing Centre / University Library
Electronic Publishing Group
Matthias Schulz, Sabine Henneberger, Susanne Dobratz
E-Mail: matthias.schulz.1 / shenneberger /dobratz@cms.hu-berlin.de
http://edoc.hu-berlin.de