
Markup Languages
XML

Denis Helic

eXtensible Markup Language - XML: Introduction(1/7)

- ▶ New Markup Language developed by Web Consortium
- ▶ Why a new Markup Language?
 - ▶ Create richly structured documents
 - ▶ Use such documents over the Web
- ▶ Why don't we use Markup Languages that we already have?
 - ▶ HTML
 - ▶ SGML

eXtensible Markup Language - XML: Introduction(2/7)

Why not HTML?

By the time Mosaic took off worldwide in 1993, people were using HTML as a hammer and were seeing nails everywhere. Unfortunately, HTML is not a hammer; even HTML 4.0 [...] only furnished a limited tagset, and no single tagset will suffice for all of the kinds of information on the Web.

by Dan Connolly, “The Evolution of Web Documents: The Ascent of XML”

eXtensible Markup Language - XML: Introduction(3/7)

- ▶ Why not HTML?
 - ▶ mixture of content and presentation
 - ▶ presentation elements: (, <i>, etc.)
 - ▶ element set is fixed (we can not define new tags specific for our application)
 - ▶ Chemist: elements to describe chemical formulas
 - ▶ Mechanic: elements to describe machine parts
 - ▶ semantics of elements is fixed (e.g. <h1> is always a first level heading)
 - ▶ mixture of content and link structure (<a>)

eXtensible Markup Language - XML: Introduction(4/7)

- ▶ But then why not SGML (see SGML slides)?
 - ▶ SGML is too complex for people
 - ▶ SGML does not require elements to have end-tags (hard to write parsers)

eXtensible Markup Language - XML: Introduction(5/7)

- ▶ Web Consortium developed XML by overcoming problems of both SGML and HTML
- ▶ Overcome problems of SGML in XML
 - ▶ make it more simple for people (33 pages spec)
 - ▶ end-tags required (easy to write parsers)
 - ▶ SGML light (not HTML++)

eXtensible Markup Language - XML: Introduction(6/7)

- ▶ Overcome problems of HTML in XML
 - ▶ clear separation of content and presentation
 - ▶ clear separation of content and link structure
 - ▶ non-fixed set of elements
 - ▶ non-fixed semantics of tags

eXtensible Markup Language - XML: Introduction(7/7)

- ▶ To achieve these goals Web Consortium defines:
 - ▶ Standard for meta markup language (Core XML)
 - ▶ Standard for linking XML documents (XML Linking)
 - ▶ Standard for specifying presentation of XML documents

- ▶ XML is meta markup language
- ▶ allows the definition of markup languages (i.e. defining set of elements for a particular application)
 - ▶ easy to describe precisely the application data
 - ▶ easy to "understand" the data (for applications and for humans)

- ▶ Core XML standard defines
 - ▶ basic syntax (how markup is distinguished from normal text)
 - ▶ elements
 - ▶ attributes
 - ▶ entities
 - ▶ comments
 - ▶ declaration
 - ▶ processing instructions

- ▶ SELFHTML Tutorial:
<http://courses.iicm.edu/mmis/selfhtml80/xml/index.htm>

▶ start-tag `<date>`

▶ end-tag `</date>`

▶ case sensitive!

▶ `<date>27.03.2003</date>` is not the same element as

▶ `<DATE>27.03.2003</DATE>`

- ▶ Empty elements don't have:
 - ▶ sub-elements (nested elements)
 - ▶ content (e.g. text)
- ▶ empty element tag `<date />`

 Nesting of elements must be correct

```
<b><i>bold and italic, but still wrong!</b></i>
```

```
<b><i>bold and italic, but this time correct!</i></b>
```

```
<person>
  <name>Denis Helic</name>
  <email>dhelic@iicm.edu</email>
  <address>
    <street>Inffeldg. 16c</street>
    <city>8010 Graz</city>
    <phone>+43-316/873-5617</phone>
  </address>
</person>
```

- ▶ Example:
<http://coronet.iicm.edu/mmis/examples/xml/basic/person.xml>
- ▶ Tutorial:
<http://courses.iicm.edu/mmis/selfhtml80/xml/regeln/tagattrwerte.htm>

- ▶ contain additional information ()
- ▶ key-value pairs in start-tags (or empty element tag)
- ▶ value always within quotation marks
- ▶ key and value case sensitive

```
<person name = "Denis Helic">  
  <email>dhelic@iicm.edu</email>  
  <address street = "Inffeldg. 16c"  
    city = "8010 Graz" phone = "+43-316/873-5617" />  
</person>
```

 Example:

```
http://coronet.iicm.edu/mmis/examples/xml/basic/person\_  
attr.xml
```

▶ attributes vs. elements dilemma

▶ attributes

```
<person department="marketing">  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>
```

▶ Example:

```
http://coronet.iicm.edu/mmis/examples/xml/basic/dep\_attr.xml
```

▶ elements

```
<person>  
  <department>marketing</department>  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>
```

▶ Example:

```
http://coronet.iicm.edu/mmis/examples/xml/basic/dep\_el.xml
```

- ▶ From experience: better to use elements than attributes because:
 - ▶ attributes cannot have multiple values (e.g. person working for two departments)
 - ▶ attributes cannot be extended (e.g. subdepartment)
 - ▶ attributes do not imply structure (e.g. nested elements)
 - ▶ attributes are harder to check (against DTD)

- ▶ Exception - unique ID!

```
<messages>
  <note id="501">
    bla, bla, bla,...
  </note>
  <note id="502">
    and so on, and so on
  </note>
</messages>
```

- ▶ Example:

<http://coronet.iicm.edu/mmis/examples/xml/basic/id.xml>

- ▶ Tutorial:

<http://courses.iicm.edu/mmis/selfhtml80/xml/regeln/tagsattrwerte.htm>

- ▶ similar to SGML/HTML entities
- ▶ start with '&', ends with semicolon ';'
- ▶ Example: `"` for “
- ▶ define your own entities
- ▶ Tutorial:
`http://courses.iicm.edu/mmis/selfhtml80/xml/dtd/entities.htm`

- ▶ Names may include:
 - ▶ Letters (capital/small)
 - ▶ Digits (0 - 9)
 - ▶ Punctuation characters `_ - . :` (reserved for namespaces)

- ▶ Rules for names
 - ▶ No digit as the first character of a name (letter or punctuation character)
 - ▶ No spaces in names
 - ▶ Names are not allowed to start with xml or XML

▶ Examples for correct names

Address

weight

s2

domain

phone.mobile

▶ Tutorial:

[http://courses.iicm.edu/mmis/selfhtml80/xml/dtd/
bearbeitungsregeln.htm#namen](http://courses.iicm.edu/mmis/selfhtml80/xml/dtd/bearbeitungsregeln.htm#namen)

▶ Example:

```
<!-- this is a comment -->
```

▶ Tutorial:

```
http://courses.iicm.edu/mmis/selfhtml80/xml/regeln/  
tagsattrwerte.htm#kommentare
```

▶ Each XML document must start with XML declaration:

▶ Simple declaration:

```
<?xml version = "1.0"?>  
<!-- rest of the XML document -->
```

▶ Full declaration:

```
<?xml version = "1.0" encoding = "ISO-8859-1"  
  standalone = "yes"?>  
<!-- rest of the XML document -->
```

▶ Tutorial:

```
http://courses.iicm.edu/mmis/selfhtml80/xml/regeln/  
xmldeklaration.htm
```

- ▶ special instructions for the processing software
 - ▶ assumes that the processing application ("understanding" of data) knows what to do with PIs
 - ▶ if not simply ignore it

```
<?xml-stylesheet type = "text/css" href = "styles.css"?>
```

▶ Tutorial:

```
http://courses.iicm.edu/mmis/selfhtml80/xml/regeln/  
xmldeklaration.htm
```

XML Document Type Definition - DTD(1/5)

- ▶ XML provides just basic syntax
- ▶ DTD specifies element set for a particular application
 - ▶ DTD for chemical formulas
 - ▶ DTD for machine parts

XML Document Type Definition - DTD(2/5)

- ▶ Why use DTD?
 - ▶ XML provides an application independent way of sharing data
 - ▶ With a DTD people agree on a certain vocabulary for interchanging data
 - ▶ When you receive data you can verify it (parser)

XML Document Type Definition - DTD(3/5)

- ▶ DTD defines:
 - ▶ which elements may be used
 - ▶ which elements are required
 - ▶ what are relationships between elements (e.g. nested elements)
 - ▶ attributes of elements
 - ▶ required attributes of elements
 - ▶ entities

XML Document Type Definition - DTD(4/5)

- ▶ different document types
- ▶ syntactical check against the basic XML syntax
- ▶ syntactical check against the specific DTD syntax
 - ▶ *valid* document (DTD) vs. *wellformed* document (basic XML syntax only)

XML Document Type Definition - DTD(5/5)

- ▶ quite simple to define a new markup language
- ▶ Tutorial 1:
`http://courses.iicm.edu/mmis/selfhtml80/xml/dtd/index.htm`
- ▶ Tutorial 2:
`http://www.xml101.com/dtd/`

Defining Elements and Attributes with DTD(1/18)

- ▶ General element template

```
<!ELEMENT el_name (el_content)>
```

- ▶ Depending on the content, different alternatives for defining elements
- ▶ Example: Persons list, with e-mail, address, etc.

Defining Elements and Attributes with DTD(2/18)

▶ Elements with text content

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

▶ Valid XML

```
<email>dhelic@iicm.edu</email>
```

Defining Elements and Attributes with DTD(3/18)

Elements with sub-elements

```
<!ELEMENT name (fname, sname)>
```

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

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

Valid XML

```
<name>
```

```
  <fname>Nick</fname>
```

```
  <sname>Scerbakov</sname>
```

```
</name>
```

Defining Elements and Attributes with DTD(4/18)

▶ Repeating elements

```
<!ELEMENT plist (person)*>
```

▶ Valid XML

```
<plist>  
  <person>  
    . . . .  
  </person>  
  <person>  
    . . . .  
  </person>  
</plist>
```

Defining Elements and Attributes with DTD(5/18)

▶ Repeating elements

▶ Valid XML

```
<plist>  
</plist>
```

Defining Elements and Attributes with DTD(6/18)

 Repeating elements

```
<!ELEMENT plist (person)+>
```

 Valid XML

```
<plist>  
  <person>  
    . . . .  
  </person>  
  <person>  
    . . . .  
  </person>  
</plist>
```

Defining Elements and Attributes with DTD(7/18)

Alternative elements

```
<!ELEMENT address (postbox | haddress)>
<!ELEMENT postbox (#PCDATA)>
<!ELEMENT haddress (street, city)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
```

Valid XML

```
<address>
  <haddress>
    <street>Inffeldgasse 16c</street>
    <city>Graz</city>
  </haddress>
</address>
```

Defining Elements and Attributes with DTD(8/18)

 Alternative elements

 Valid XML

```
<address>
```

```
  <postbox>
```

```
    P.O.BOX 1155644
```

```
  </postbox>
```

```
</address>
```

Defining Elements and Attributes with DTD(9/18)

Optional elements

```
<!ELEMENT person (name, email+, phone+, address?,  
    comment?, image?)>
```

Valid XML

```
<person>  
    <name> .... </name>  
    <email> .... </email>  
    <phone> .... </phone>  
    <comment>  
        MMIS Lecturer  
    </comment>  
</person>
```

Defining Elements and Attributes with DTD(10/18)

▶ Optional elements

▶ Valid XML

```
<person>  
  <name> . . . . </name>  
  <email> . . . . </email>  
  <phone> . . . . </phone>  
  <address> . . . . </address>  
  <image> . . . . </image>  
</person>
```

Defining Elements and Attributes with DTD(11/18)

▶ Elements with mixed content

```
<!ELEMENT comment (#PCDATA | important)*>
```

▶ Valid XML

```
<comment>
```

```
  MMIS Lecturer.
```

```
  <important>Seminar/Project Lecturer</important>
```

```
</comment>
```

Defining Elements and Attributes with DTD(12/18)

▶ Empty elements

```
<!ELEMENT image EMPTY>
```

▶ Valid XML

```
<image/>
```

Defining Elements and Attributes with DTD(13/18)

General attribute template

```
<!ELEMENT el_name (el_content)>
<!ATTLIST el_name
    at_name_1 at_content [#REQUIRED|#IMPLIED|
        #FIXED "value"|"default"]
    at_name_2 at_content [#REQUIRED|#IMPLIED|
        #FIXED "value"|"default"]
>
```

Defining Elements and Attributes with DTD(14/18)

▶ Required and optional attributes

```
<!ATTLIST image
  src CDATA #REQUIRED
  alt CDATA #IMPLIED
>
```

▶ Valid XML

```
<image src="nsherbak.jpg" alt="Nick's Picture"/>
```

▶ Valid XML

```
<image src="nsherbak.jpg"/>
```

Defining Elements and Attributes with DTD(15/18)

▶ Attributes with predefined values

```
<!ATTLIST phone  
  site (home|office) #REQUIRED  
>
```

▶ Valid XML

```
<phone site="office">++43/316/873/5617</phone>  
<phone site="home">++43/650/2203019</phone>
```

Defining Elements and Attributes with DTD(16/18)

▶ Attributes with default values

```
<!ATTLIST important  
  category (normal|very) "normal"  
>
```

▶ Valid XML

```
<important>Seminar/Project Lecturer</important>
```

Defining Elements and Attributes with DTD(17/18)

Attributes as IDs

```
<!ATTLIST person
  pid ID #REQUIRED
>
```

Valid XML

```
<person pid="nr_1">
  . . . .
</person>
<person pid="nr_2">
  . . . .
</person>
```

Defining Elements and Attributes with DTD(18/18)

- ▶ Example DTD:
`http://coronet.iicm.edu/mmis/examples/xml/dtd/plist.dtd`
- ▶ Example Valid XML:
`http://coronet.iicm.edu/mmis/examples/xml/dtd/plist.xml`

Referencing DTD inside XML documents(1/5)



Internal DTD

```
<?xml version="1.0" encoding="ISO-8859-1"
  standalone = "yes"?>
<!DOCTYPE person [
  <!ELEMENT person (name, email, address)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
  <!ELEMENT address (street, city, phone)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
]>
```

Referencing DTD inside XML documents(2/5)

 rest of the document

```
<person>
```

```
  <name>Denis Helic</name>
```

```
  <email>dhelic@iicm.edu</email>
```

```
  <address>
```

```
    <street>Inffeldg. 16c</street>
```

```
    <city>8010 Graz</city>
```

```
    <phone>+43-316/873-5617</phone>
```

```
  </address>
```

```
</person>
```

 Example:

```
http://coronet.iicm.edu/mmis/examples/xml/dtd/person\_in.xml
```

Referencing DTD inside XML documents(3/5)

External DTD (System)

```
<?xml version="1.0" encoding="ISO-8859-1" standalone = "no"?>
```

```
<!DOCTYPE person SYSTEM "person.dtd">
```

Example:

```
http://coronet.iicm.edu/mmis/examples/xml/dtd/person\_es.xml
```

Referencing DTD inside XML documents(4/5)

External DTD (Public)

```
<?xml version="1.0" encoding="ISO-8859-1" standalone = "no"?>
```

```
<!DOCTYPE person PUBLIC  
    "public identifier goes here" "person.dtd">
```

Example:

```
http://coronet.iicm.edu/mmis/examples/xml/dtd/person\_ep.xml
```

Referencing DTD inside XML documents(5/5)

 person.dtd

```
<!ELEMENT person (name, email, address)>
```

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

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

```
<!ELEMENT address (street, city, phone)>
```

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

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

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

 Example:

```
http://coronet.iicm.edu/mmis/examples/xml/dtd/person.dtd
```

 Declaration:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<!DOCTYPE book SYSTEM "book.dtd">
```



XML content:

```
<book>
  <info>this is some info about this book</info>
  <date>2000-10-31</date>
  <author>
    <name>Christof Dallermassl</name>
    <email>cdaller@iicm.edu</email>
  </author>
  <chapter name="the chapter heading">
    <section name="Section one">
      <image src="image1.jpg"/>
      <para>some text abtout Ted Nelson .....</para>
    </section>
  </chapter>
</book>
```

 Example:

`http://coronet.iicm.edu/mmis/examples/xml/dtd/book.xml`

DTD

```
<!ELEMENT book      (info, date?, author, chapter+)>
<!ELEMENT info      (#PCDATA)>
<!ELEMENT date      (#PCDATA)>
<!ELEMENT author    (name, email?)>
<!ELEMENT name      (#PCDATA)>
<!ELEMENT email     (#PCDATA)>
<!ELEMENT chapter   (section+, para?)>
<!ELEMENT section   (image*, para+)>
<!ELEMENT para      (#PCDATA)>
<!ELEMENT image     EMPTY>
<!ATTLIST chapter   name CDATA #IMPLIED>
<!ATTLIST section   name CDATA #IMPLIED>
<!ATTLIST image     src  CDATA #REQUIRED>
```

 Example:

`http://coronet.iicm.edu/mmis/examples/xml/dtd/book.dtd`

 Validator:

`http://www.stg.brown.edu/service/xmlvalid/`

- ▶ Can not define certain syntax restrictions
 - ▶ Exactly 10 elements of this type
- ▶ data types limited (only CDATA, IDs)
 - ▶ Integers, Strings, Dates
 - ▶ Especially bad for exchanging data from databases
- ▶ DTD syntax is not related to XML syntax
- ▶ no namespaces
 - ▶ workarounds

- ▶ Relax-NG
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=relax-ng
 - ▶ Similar possibilities as in DTD, but XML based
- ▶ XML Schema (W3C)
<http://www.w3.org/XML/Schema>
 - ▶ Different datatypes, syntactical possibilities, etc
 - ▶ Dumping data from databases
- ▶ More on these technologies: MMIS 2

- ▶ addition to XML-standard
- ▶ not necessary, but often useful
- ▶ provide unique element names (<address> vs. <my:address>)
- ▶ declaration:
 - ▶ with explicit namespace

```
<OReilly:Books
  xmlns:OReilly="http://www.oreilly.com/"
  <OReilly:Book>just one book .... </OReilly:Book>
</OReilly:Books>
```
 - ▶ using default-namespace

```
<Books xmlns="http://www.oreilly.com/"
  <Book>the same book as above ... </Book>
</Books>
```

- ▶ Scalable Vector Graphics (SVG)
 - ▶ More on SVG within Digital Images
- ▶ Synchronized Multimedia Language (SMIL)
 - ▶ More on SMIL within Digital Video

- ▶ eXtensible HyperText Markup Language (XHTML)
<http://www.w3.org/TR/xhtml1/>
 - ▶ Reformulation of HTML 4.0 as an XML DTD
- ▶ HTML 4.01 → 1.0
- ▶ XHTML 1.0 Strict DTD <http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>
- ▶ XHTML 1.0 Transitional DTD <http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
- ▶ XHTML 1.0 Strict DTD <http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd>

- ▶ XHTML 1.0 → Modularization of XHTML
<http://www.w3.org/TR/xhtml-modularization/>
- ▶ Decomposition of XHTML 1.0 into a number of modules
- ▶ Module is a set of elements used for the same purpose
 - ▶ Structure module: body, head, html, title
 - ▶ Text module: headings, div, span, etc,
 - ▶ Hypertext module: a
 - ▶ Table module: table, tr, td, etc.

- ▶ Each module defined with an XML DTD
- ▶ Purpose: create an XHTML dialect by combining only modules that are needed
- ▶ Example: XHTML Basic (for mobile browsers, PDA, mobile phones, etc.)
 - ▶ Only a subset of modules from XHTML 1.0
<http://www.w3.org/TR/xhtml1-basic/>
- ▶ XHTML 1.1 <http://www.w3.org/TR/xhtml111/>
 - ▶ Reformulation of XHTML 1.0 Strict as XHTML modules

Geographic Markup Language (GML)

<http://www.opengis.org/techno/specs/02-009/GML2-11.html>

```
<River>
  <gml:description>The river that runs through Cambridge.</gml:description>
  <gml:name>Cam</gml:name>
  <gml:centerLineOf>
    <gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:coord><gml:X>0</gml:X><gml:Y>50</gml:Y></gml:coord>
      <gml:coord><gml:X>70</gml:X><gml:Y>60</gml:Y></gml:coord>
      <gml:coord><gml:X>100</gml:X><gml:Y>50</gml:Y></gml:coord>
    </gml:LineString>
  </gml:centerLineOf>
</River>
```


▶ Chemical Markup Language (CML):

▶ Demo: <http://www.adobe.com/svg/demos/main.html>

▶ Spec: <http://www.xml-cml.org/>

 Docbook: <http://docbook.sourceforge.net/>

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>
  <artheader>
    <title>My Article</title>
    <author><honorific>Dr</honorific><firstname>Emilio</firstname>
      <surname>Lizardo</surname></author>
  </artheader>
  <para> ... </para>
  <sect1><title>On the Possibility of Going Home</title>
  <para> ... </para>
</sect1>
  <bibliography> ... </bibliography>
</article>
```

 Website:
[http://docbook.sourceforge.net/release/website/
example/layout.html](http://docbook.sourceforge.net/release/website/example/layout.html)

Mathematical Markup Language (MathML)

$$\sum_{x=1}^b f(x)$$

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <sum/>
      <bvar><ci> x </ci></bvar>
      <lowlimit><ci> a </ci></lowlimit>
      <uplimit><ci> b </ci></uplimit>
      <apply>
        <fn><ci> f </ci></fn>
        <ci> x </ci>
      </apply>
    </apply>
  </math>
```

▶ MathML (continued)

▶ Spec: <http://www.w3.org/Math/>

▶ Demo: <http://www.mozilla.org/projects/mathml/demo/basics.xhtml>

▶ Election Markup Language (EML)

▶ <http://xml.coverpages.org/eml.html>

Displaying XML
CSS/XSLT/XSL-FO

Denis Helic

Displaying XML -Introduction(1/6)

- ▶ Document-oriented vs Data-oriented XML
- ▶ Data-oriented XML: Exchanging data from databases
 - ▶ Usually for automatic processing by software
 - ▶ Not intended for displaying
 - ▶ More on this in MMIS 2

Displaying XML -Introduction(2/6)

- ▶ Document-oriented XML for multimedia documents (e.g. XHTML, SVG, SMIL, etc.)
 - ▶ Intented for displaying on user screen
 - ▶ XML does not include information for presentation
 - ▶ No implied semantics for presentation like in HTML

Displaying XML -Introduction(3/6)

- ▶ Possible alternatives for displaying XML:
 - ▶ Write special viewers to display a particular XML document (e.g. SVG viewer from Adobe)
 - ▶ Browser
 - ▶ Existing viewers for PDF, PS, RTF, etc.

Displaying XML -Introduction(4/6)

- ▶ Displaying in browser:
 - ▶ Default presentation as tree of elements
 - ▶ Altering default presentation with CSS
 - ▶ Transforming XML into HTML/XHTML with XSLT - additional formatting with CSS possible

Displaying XML -Introduction(5/6)

- ▶ Existing viewers (PDF, PS, RTF):
 - ▶ Formatting XML as PDF/PS/RTF with XSL-FO
 - ▶ Advanced topic (MMIS 2)

Displaying XML -Introduction(6/6)

- ▶ Technologies for displaying XML:
 - ▶ Cascading Style Sheets (CSS)
 - ▶ eXtensible Style Sheets Transformations (XSLT)
 - ▶ eXtensible Style Sheets Formatting Objects (XSL-FO)

Displaying XML with CSS(1/2)

- ▶ To be able to display XML with CSS the browser must:
 - ▶ Parse XML
 - ▶ Support CSS (better if CSS 2.0)
 - ▶ Support CSS in connection with XML
- ▶ Mozilla, Opera, IE

Displaying XML with CSS(2/2)

- ▶ Embedding CSS in XML
 - ▶ Processing instruction!
 - ▶ Browsers understand this processing instruction

```
<?xml-stylesheet type = "text/css" href = "course.css"?>
```

Displaying XML with CSS: Example(1/17)

- ▶ Description of MMIS Course
 - ▶ DTD defining elements set
 - ▶ Valid XML document describing the course
 - ▶ CSS for defining presentation of elements

Displaying XML with CSS: Example(2/17)

▶ DTD for Course description

▶ Describing course properties

```
<!ELEMENT course      (title, description,  
                        homepage, lecturer,  
                        organization, language,  
                        content, goal,schedule)>  
  
<!ELEMENT title      (#PCDATA)>  
<!ELEMENT description (#PCDATA)>  
<!ELEMENT homepage   (#PCDATA)>  
<!ELEMENT lecturer   (#PCDATA)>  
<!ELEMENT organization (#PCDATA)>  
<!ELEMENT language   (#PCDATA)>
```

...

Displaying XML with CSS: Example(3/17)

 Describing course content and schedule (calender)

...

```
<!ELEMENT content      (topic+)>
```

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

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

```
<!ELEMENT schedule     (event+)>
```

```
<!ELEMENT event        (date, title, slides)>
```

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

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

...

Displaying XML with CSS: Example(4/17)

 Referencing slides from schedule

...

```
<!ATTLIST slides
  xlink:type (simple) #FIXED "simple"
  xlink:href CDATA #IMPLIED
  xlink:role CDATA #FIXED
    "http://coronet.iicm.edu/mmis/example/xml/css"
  xlink:arcrole CDATA #IMPLIED
  xlink:title CDATA #IMPLIED
  xlink:show (new|replace|embed|other|none) #IMPLIED
  xlink:actuate (onLoad|onRequest|other|none) #IMPLIED>
```

...

Displaying XML with CSS: Example(5/17)

 Referencing homepage

...

```
<!ATTLIST homepage
```

```
  xlink:type (simple) #FIXED "simple"
```

```
  xlink:href CDATA #IMPLIED
```

```
  xlink:role CDATA #FIXED
```

```
    "http://coronet.iicm.edu/mmis/example/xml/css"
```

```
  xlink:arcrole CDATA #IMPLIED
```

```
  xlink:title CDATA #IMPLIED
```

```
  xlink:show (new|replace|embed|other|none) #IMPLIED
```

```
  xlink:actuate (onLoad|onRequest|other|none) #IMPLIED>
```

Displaying XML with CSS: Example(6/17)

- ▶ Example uses XLink: `http://www.w3.org/XML/Linking`
 - ▶ More on XLink in MMIS 2
- ▶ In the simplest case XLink is similar to linking in HTML
- ▶ Possible to have different link attributes, types, etc.
- ▶ Example shows how to use namespaces in DTD
 - ▶ Define attribute names as full names including namespace prefix

Displaying XML with CSS: Example(7/17)

- ▶ Valid XML document: XML declaration and root element

```
<?xml version = "1.0" encoding="ISO-8859-1" standalone = "no"?>  
<!DOCTYPE course SYSTEM "course.dtd">  
<?xml-stylesheet type = "text/css" href = "course.css"?>  
<course xmlns:xlink="http://www.w3.org/1999/xlink">
```

...

- ▶ Refer to namespace in the root element

Displaying XML with CSS: Example(8/17)

 Valid XML document (Course attributes)

```
<title>Multimedia Information Systems</title>
<description>...</description>
<homepage xlink:href = "http://courses.iicm.edu/mmis"
  xlink:type="simple">http://courses.iicm.edu/mmis
</homepage>
<lecturer>Denis Helic</lecturer>
<organization>...</organization>
<language>English/German</language>
```

Displaying XML with CSS: Example(9/17)

 Valid XML document (Course content)

```
<content>
  <topic>The Internet: Technological Background</topic>
  <topic>World Wide Web</topic>
  ...
</content>
<goal>...</goal>
```

Displaying XML with CSS: Example(10/17)

 Valid XML document (Course schedule)

```
<schedule>
  <event>
    <date>27.02.2003</date>
    <title>Introduction ...</title>
    <slides xlink:href = "slides/intro.pdf"
      xlink:type="simple">intro.pdf</slides>
  </event>
  ...
</schedule>
</course>
```

Displaying XML with CSS: Example(11/17)

- ▶ CSS for displaying course.xml
- ▶ Number of CSS statements:
 - ▶ Selector (which XML element this time!)
 - ▶ Declaration (how to display that element)

Displaying XML with CSS: Example(12/17)

 title element:

```
title{
  display: block;
  font-size: x-large;
  background: white;
  color: black;
  text-align: center;
  padding: 1em;
}
```

Displaying XML with CSS: Example(13/17)

 description element:

```
description{  
  display: block;  
  background: white;  
  color: black;  
  text-align: center;  
}
```

Displaying XML with CSS: Example(14/17)


 homepage:hover (pseudo selector):

```
homepage:hover{  
  background: white;  
  color: red;  
}
```

 event title: (nested type selector):

```
event title{  
  font-size: medium;  
  color: black;  
  display: table-cell;  
  padding: 0.5em;  
  background: lightblue;  
}
```

Displaying XML with CSS: Example(15/17)

 schedule element: (displaying as table!):

```
schedule{
  color: black;
  display: table;
  border-spacing: 3px;
  background: gray;
  text-align: center;
  margin-left: 2em;
  margin-top: 1em;
  margin-bottom: 1em;
}
```

Displaying XML with CSS: Example(16/17)

- ▶ How to display links with CSS?
 - ▶ Impossible! Just hope that the browser understands XLink (Mozilla only)

```
<slides xlink:href = "slides/intro.pdf"  
  xlink:type="simple">intro.pdf</slides>
```

Displaying XML with CSS: Example(17/17)

- ▶ Example (XML):
`http://coronet.iicm.edu/mmis/examples/xml/css/course.xml`
- ▶ Example (DTD):
`http://coronet.iicm.edu/mmis/examples/xml/css/course.dtd`
- ▶ Example (CSS):
`http://coronet.iicm.edu/mmis/examples/xml/css/course.css`
- ▶ Tutorial:
`http://courses.iicm.edu/mmis/selfhtml80/xml/darstellung/css.htm`

Displaying XML with CSS: Analysis

▶ Advantages:

- ▶ Quick solution: no need to learn XSL-FO/XSLT, no need to process XML at the server side

▶ Disadvantages:

- ▶ Depends on CSS (2.0) support in browser (e.g. problems with IE)
- ▶ No possibility to process the content (e.g. sorting, filtering, etc.)
- ▶ No possibility to display links (depends on browser's support of XLink!)
- ▶ No possibility to display images

eXtensible Stylesheet Language: XSL(1/2)

- ▶ Standards family consisting of three parts:
- ▶ XSLT <http://www.w3.org/TR/xslt>
- ▶ XPath (how to find elements in XML) <http://www.w3.org/TR/xpath>
- ▶ XSL-FO <http://www.w3.org/TR/xsl/>
 - ▶ More on it in MMIS 2
- ▶ XSL: <http://www.w3.org/Style/XSL/>

eXtensible Stylesheet Language: XSL(2/2)

- ▶ eXtensible Stylesheet Language Transformation: XSLT
- ▶ Recommendation of october 15th, 2001 version 1.0
- ▶ Web Consortium works on version 2.0

- ▶ XSLT language allows you to:
 - ▶ transform an XML document (source) into another XML document (target)
 - ▶ target document might be an HTML document (XHTML!)
 - ▶ sort elements in a source document
 - ▶ filter elements from a source document

- ▶ XML documents are organized into a tree
- ▶ Tree consists of nodes:
 - ▶ root node
 - ▶ element nodes
 - ▶ text nodes
 - ▶ attribute nodes
 - ▶ namespace nodes
 - ▶ processing instruction nodes
 - ▶ comment nodes
- ▶ Example: `http://coronet.iicm.edu/mmis/examples/xml/xsl/course_raw.xml`

How does XSLT work?(2/3)

- XSLT transforms a source tree into a target tree by following *template rules*
- Each template rule consists of two parts:
 - Pattern for matching nodes from the source tree (XPath)
 - Output in the target tree when the pattern is matched

How does XSLT work?(3/3)

- ▶ XSLT document is a set of template rules
- ▶ XSLT processor traverses the source tree
- ▶ At each node it checks the rules to see if this node matches any of the rules patterns
- ▶ If the node matches the output is added to the target node

- ▶ Course XML document (modified DTD - no XLink needed)
- ▶ Declaration modified a bit!

```
<?xml version = "1.0" encoding="ISO-8859-1" standalone = "no"?>  
<!DOCTYPE course SYSTEM "course.dtd">  
<?xml-stylesheet type = "text/xsl" href = "course.xsl"?>  
<course>  
  ...
```


➤ XSLT document:

➤ header/namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output doctype-public="-//W3C//DTD HTML 4.01//EN"
  method="html" />
```

▶ template for root node

```
<xsl:template match="/">
  <html>
    <head>
      ...
    </head>
    <body>
      <xsl:apply-templates/>
      ...
    </body>
  </html>
</xsl:template>
```

 template for <title> element:

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

▶ template for <homepage>:

```
<xsl:template match="homepage">
  <div>
    <span class="key">Homepage of the course: </span>
    <a>
      <xsl:attribute name="href">
        <xsl:value-of select="."/>
      </xsl:attribute>
      <xsl:value-of select="."/>
    </a>
  </div>
</xsl:template>
```



template for <schedule>:

```
<xsl:template match="schedule">
  <p class="key">Course schedule:</p>
  <table>
    ...
    <xsl:for-each select="event">
      <xsl:sort select="date"/>
      <tr>
        <td><xsl:value-of select="date"/></td>
        ...
      </tr>
    </xsl:for-each>
  </table>
</xsl:template>
```

 end of `<xsl:stylesheet>`:
`</xsl:stylesheet>`

- ▶ Example (XML):
`http://coronet.iicm.edu/mmis/examples/xml/xsl/course.xml`
- ▶ Example (DTD):
`http://coronet.iicm.edu/mmis/examples/xml/xsl/course.dtd`
- ▶ Example (XSL):
`http://coronet.iicm.edu/mmis/examples/xml/xsl/course.xsl`

- ▶ XSLT has a lot of other features:
 - ▶ Filtering (with conditional statements)
 - ▶ Numbering
 - ▶ Variables, parameters, etc.
- ▶ Tutorial:
`http://courses.iicm.edu/mmis/selfhtml80/xml/darstellung/xslgrundlagen.htm`

XSLT Example: Processing(1/5)

- ▶ Few alternatives to process the transformation:
 - ▶ Run a stand-alone processor for pre-generation of output (e.g. XML-Spy, Xalan, etc.)
 - ▶ Browser transformation
 - ▶ Server-side transformation

XSLT Example: Processing(2/5)

▶ processing with Xalan (<http://xml.apache.org/xalan-j>)

```
export CLASSPATH="path_to_xalan/xercesImpl.jar;  
                 path_to_xalan/xalan.jar;  
                 path_to_xalan/xml-apis.jar"
```

```
java org.apache.xalan.xslt.Process -in course.xml  
    -xsl course.xsl -out course.html
```

▶ Example:

```
http://coronet.iicm.edu/mmis/examples/xml/xsl/course.  
html
```

XSLT Example: Processing(3/5)

- ▶ processing by browser
- ▶ browser needs to know about XSLT
 - ▶ IE implements very old XSLT working draft
 - ▶ Mozilla implements XSLT 1.0
- ▶ Example:
`http://coronet.iicm.edu/mmis/examples/xml/xsl/course.xml`
- ▶ Example:
`http://coronet.iicm.edu/mmis/examples/xml/xsl/course.xsl`

XSLT Example: Processing(4/5)

- ▶ processing by server
- ▶ Extend the server with XSLT processor (e.g. PHP XSLT processor)
- ▶ Cocoon Apache XML publishing framework
 - ▶ Written in Java, runs as a Java servlet
 - ▶ Cocoon: <http://cocoon.apache.org/>

XSLT Example: Processing(5/5)

▶ Java Servlet that processes XML/XSLT

▶ Example:

```
http://coronet.iicm.edu/xsltprc/XSLTProcessor?xml_url=  
/mmis/examples/xml/xsl/course_ser.xml
```

- ▶ Stand-alone processing
- ▶ Advantages:
 - ▶ Pre-generated HTML served as normal HTML: fast
- ▶ Disadvantages:
 - ▶ Any modification in source requires re-generating of HTML

XSLT Processing: Analysis(2/3)

- ▶ Browser processing
- ▶ Advantages:
 - ▶ No need to consume server processing power: fast if client is fast
 - ▶ Since dynamic no additional work if source changes
- ▶ Disadvantages:
 - ▶ Poor support in browsers for XSTL processing

XSLT Processing: Analysis(3/3)

- ▶ Server processing
- ▶ Advantages:
 - ▶ Since dynamic no additional work if source changes
 - ▶ HTML served - all browsers can display it
- ▶ Disadvantages:
 - ▶ Consumes server processing power (can be improved with caching on the server side)

XSLT Pattern Matching: XPath(1/3)

- ▶ pattern matching language XPath

<http://www.w3.org/TR/xpath.html>

- ▶ direct path to child-element from current element:

```
<xsl:value-of select="/course/title" />
```

- ▶ children of children:

```
<xsl:template match="course//event" >
```

- ▶ wildcards:

```
<xsl:template match="course/schedule/*" >
```

- ▶ boolean expressions:

```
<xsl:template match="title | description" >
```

XSLT Pattern Matching: XPath(2/3)

- ▶ comments:
`<xsl:template match="course/comment()" >`
- ▶ processing instructions:
`<xsl:template match="course/pi()" >`
- ▶ attributes:
`<xsl:template match="course[@info]" >`
- ▶ attributes with specific values:
`<xsl:template match="course[@info='blablabla']" >`

XSLT Pattern Matching: XPath(3/3)

- ▶ use attribute values:

```
<xsl:template match="course">  
  <xsl:value-of select="@info"/>  
</xsl:template>
```

- ▶ XML Publishing Framework (inkl. Transformer and Formatting Objects) Cocoon:
<http://xml.apache.org/cocoon>
- ▶ XSL transformer Xalan:
<http://xml.apache.org/xalan-j>
- ▶ XML-Spy: Austrian Company! <http://www.xmlspy.com>
- ▶ IBM-XSL-Editor:
<http://www.alphaworks.ibm.com/tech/xsleditor>

- ▶ primary access point is the XML-website of the W3C:
<http://www.w3.org/XML>
- ▶ quite good online-intro XML101:
<http://www.xml101.com>
- ▶ W3C Website for Schemas:
<http://www.w3.org/TR/xmlschema-0/>

- ▶ O'Reilly XML Pocket Reference by Robert Eckstein (in english at amazon.com: http://www.amazon.com/exec/obidos/ASIN/1565927095/ref=sim_books/102-4236263-7569756) also available in german (“kurz und gut”)
- ▶ XML Bible by Elliotte Rusty Harold (<http://ibiblio.org/xml/books/bible2/>)
 - ▶ chapter 17 (XSLT) of the XML-bible available online at <http://ibiblio.org/xml/books/bible2/chapters/ch17.html>
 - ▶ chapter 18 (XSL-FO) of the XML-bible available online at <http://www.ibiblio.org/xml/books/bible2/chapters/ch18.html>
 - ▶ “O'Reilly: Mastering XML Transformations”