Languages for Data Integration of Semi-Structured Data I – XML Basics

Recuperación de Información 2007
Lecture 2.
Overview

- Introduction and motivation
- Many different formats: documents, databases, semi-structured data. Ultimately, integration, mediation

- XML – definition
- XML features, i.e. what is special about XML?
- From SGML to simple HTML, from too simple HTML to XML
- XML syntactic structure
- Well-formed and valid documents: DTDs

- XML companion standards overview
- XML namespaces
Introduction and Motivation

Information exchange formats

Web as a large database:
- Extraction of Data from the web
- HTML/XML $\Leftrightarrow$ human/machine

Increasing quantity of Data with flexible irregular structure:
- Integration of heterogeneous data.

Document-view and Database-view become one:
- Databases: models of (semi-)structured data
- Documents: XML
Documents

- Intra- und inter-document structure, (e.g. tables, etc.)
- **Common** presentation formats such as HTML
  - reflect only rough structures but mainly restricted to layout information
  - based on SGML
- **Global** infrastructure for document exchange (e.g., Web)
  - But: Web is unstructured, in the best case it is a large graph

- Mix of documents and databases
- Requirements for management of large document collections

This lead to the need for developing of new formats for structured data exchange, in particular XML.
Databases: state of art

- Relational DB, EER for structure description
  - modelled as finite First Order Logic structure
  - In DB theory, Finite Model Theory (FMT) and descriptive complexity theory are important
  - Alternative: OODB-model

- Data models and query languages

- Break between logical view and physical implementation
  - Logical view: what are valid queries, ...
  - Physical implementation: how to store data, B-Trees, etc. ...

- External view: Views on how each user perceives data, representation

- Storage techniques and techniques for database consistency/integrity (key, integrity constraints, triggers, etc.)
Database architectures

Databases

- Traditional **Client/Server architecture**
- **Warehouses**: intermediate DB imported from other source-DBs; this will be queried upon (problems with updates etc.)
- **Mediator/Wrapper systems**: queries from clients will be splitted and upon query to the corresponding source-DB translated, the output will be produced on-the-fly by the mediator; especially for heterogeneous data sources

Web-usage

- **More layers**: DB-servers, legacy data, transformation components, application servers, etc. (to HTML, only layout, semantics/structure is somehow lost), **clients** (browser), ...
Semistructured data

The need for models for flexible and irregular structures has lead to models for semi-structured data:
- When there is no solid known explicit known schema
- When databases have many null-values (incomplete knowledge)
- When the database schema is large
- When data is not well-typed (i.e. Can be of different types, text including (marked-up) data, etc.)

Modelling as tree (XML) or more generally a graph (RDF... later on in this lecture)
Example: document as tree... somehow adhoc ;-)

This is a document with a table

<table>
<thead>
<tr>
<th>Robert</th>
<th>Baumgartner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axel</td>
<td>Polleres</td>
</tr>
</tbody>
</table>

and an email address
axel.polleres@uibk.ac.at

Blabla.doc

This is a document with a table:

<table>
<thead>
<tr>
<th>Robert</th>
<th>Baumgartner</th>
<th>Axel</th>
<th>Polleres</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:axel.polleres@uibk.ac.at">axel.polleres@uibk.ac.at</a></td>
<td>Blabla.doc</td>
<td>Text</td>
<td>Table</td>
</tr>
</tbody>
</table>

Example: document as tree... somehow adhoc ;-)
Mediators

- Mediator passes data between the user and data resource (Middleware)
- Mediators use wrappers which allow to have homogeneous access to heterogeneous content

We need a uniform data format for (semi-)structured Data → XML
XML
What is XML? XML in 10 points:

1. XML is a standard exchange format for semi-structured Data
2. XML looks a bit like HTML
3. XML is plain text, but not for the human reader
4. XML representation is not necessary concise
5. XML is a family of technologies (XML, XPath XSL, XLink, DOM, etc.)
6. XML is new but not so new (since 1998 a W3C standard, but SGML already existing since the early 80ies)
7. "HTML in XML" is called XHTML
8. XML is modular (by the use of namespaces)
9. XML is the basis for RDF and for the Semantic Web
10. XML is license free, platform independent and well-supported

In a nutshell:
- XML provides a standardized syntax for markup languages
- XML uses elements and attributes to define a tree structure
- An XML document can have a tree structure of arbitrary level of complexity
What is XML? More abstract:

- XML (eXtensible Markup Language) is a framework for defining markup languages.

- XML derives from SGML (Standard Generalized Markup Language), and conforms to ISO 8879 (SGML).

- XML is a standardization effort by W3C.

- XML is more a syntax than a language i.e. there is no fixed set of markup tags (as opposed to e.g. HTML).
Preconditions for XML appearance

- **Limitations of HTML**
  - Separating layout from structure, no possibilities of reuse
  - Structure/Layout description only

- **Complexity of SGML**
  - Unsuitable for Web-applications, SGML more flexible, but more complex than XML

- **Other flexible markup languages needed**
  - Scientific Markup
  - Handies (WML), Palmtops
  - Document type/usage built-in core architecture

- **Information representation, e-commerce**
- **XML simplifies electronic data interchange**
What was XML designed for?

- To separate **syntax** from **semantics** to provide a common framework for structuring information syntactically
- To allow **tailor-made markup** for any possibly application domain
- To support **internationalization** and **platform independence**
- To be the future standard of structured information
- Easy of **transformation/exchange**, **ASCII/Unicode** based.
History

- Hypertext (1945)
  - Any navigation through documents
- GML (SGML predecessor) (1969)
- HTML Tim Berners-Lee, CERN (1989)
  - Goal: simple documents exchange
- W3C was founded (1994)
- SGML work group (1996)
- XML 1.0 (1998)
- DOM (1998)
- XSLT (1999)
- XHTML 1.0 (2000, revised 2002)
- XQuery und XPath 2.0 Working Drafts (2002)
- XML 1.1 Recommendation (Feb 2004)
World Wide Web Consortium
- Founded by CERN, MIT and others
- ca. 250 participants (Companies, Academic partners)
- Function: standardization of web formats
- Not normative: gives only recommendations, no ISO bearing standards

Six types of documents
- **Note**
  - Not a component in the standardization process
  - No declaration that W3C stands behind
- **Working Draft (WD)**
  - Documentation of a discussion condition
- **Last Call WD**
  - When the goals are reached
- **Candidate Recommendation (CR)**
  - Confirmation of success
- **Proposed Recommendation**
  - Extension; partial implementation
- **Recommendation**
  - official W3C standard
XML: Now how does XML differ from HTML?

- **New tags and attributes** can be defined
- Document structures can be nested to any level of complexity
- An XML document can contain or refer to optional descriptions of its grammar (DTDs, XSD) for use by applications that need to perform structure validation

- **Strict syntax**: all tags have to be closed! No overlapping tags, etc. (which is tolerated by many web browsers for HTML)
XML : How does XML differ from SGML?

- XML is a *restriction* of the Standard Generalized Markup Language, for instance also HTML is valid *SGML* but XML is more restrictive.

- simplicity, stricter syntax, made it much more successful than original SGML
XML: How does XML differ from HTML?

We can structure the same information in... HTML and XML

```xml
<?xml version="1.0"?>
<employees>
  <marketing>
    <employee id="1834">
      <name>Gustav Sielmann</name>
      <email>gsielmann@Dot.com</email>
      <tel>+43/0662/723942-124</tel>
      <fax>+43/0662/723942-800</fax>
    </employee>
  </marketing>
</employees>
```
Applications and instances

**Architecture**

- SGML
- XML

**Applications**

- HTML
- DocBook
- XHTML
- XSD
- CML

**Instances**

- NWG

**XHTML**: HTML as XML application

**CML**: Chemical Markup Language

**XSD**: XML Schema Definition
XML: Structure of an XML document

An XML document is composed of:

- Prolog
- Elements
- Attributes
- Entity References
- Comments
- Possibly a DTD (Document Type Definition)
How XML looks like

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- DERI -->
<teaching>
  <course jahr="2004">
    <title>Telecooperation</title>
    <keyword>XML</keyword>
    <keyword>extraction</keyword>
    <keyword>integration</keyword>
    <start>
      <date>Thu 3.3.</date>
      <time sine_tempore="no">11:00</time>
      <place>HS A</place>
    </start>
    </course>
  </teaching>

Acknowledgements: R. Baumgartner (thanks for some slides...)
Example of an XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- DBAI courses 2003 -->
<teaching>
  <course jahr="2003">
    <title>Web Daten Extraktion und Integration</title>
    <keyword>XML</keyword>
    <keyword>Extraktion</keyword>
    <keyword>Integration</keyword>
    <start>
      <datum>Mi 5.3.</datum>
      <time sine_tempore="yes">14:00</time>
      <place>Seminarraum 184/2</place>
    </start>
  </course>
</teaching>
```
XML : The Prolog

- The Prolog is the first structural element in the XML document.
- It is usually divided into an XML declaration and (optional) a DTD.

E.g.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```
XML: Elements

- All subsequent elements must be within one **document element**
- XML elements must contain a **start tag** and a matching **end tag** prefixed by a slash. E.g. `<YEAR>1976</YEAR>`
- **Empty elements** can be written `<CANCELLED/>` instead of using both tags without content
- XML is case-sensitive! (e.g. `<br/>` != `<BR/>`
- Element names must begin with an underscore or a letter. Subsequent letters in the element name may include: letters, digits, underscores, hyphens and periods, (Attention: `&`, `<`, `>`, ``, `,” are reserved!)
XML: Attributes

- XML attributes are attached to elements.
- They are a way of associating values to an element without making them part of the actual content.
- Attribute names must begin with a letter or an underscore and must not contain any white spaces.
- Attribute values must be quoted.
- An attribute name may occur **only once** per element.
- Attributes only in the start tag of an element.

```
<employee name="Axel Polleres">
  axel@polleres.net
</employee>
```
XML: Entity References

- Entity references are used to reference data that is not directly in the structure.
- Pre-built entity references are used to represent special characters, such as
  
  ```
  &amp; < &lt; > &gt; " &quot; ` &apos;
  ```

  or character-References: &#211; (dezimal), &#xF3; (hex)

- New entities can be declared in DTDs (see later slides)

  e.g. the string Peter&Tom("Don‘t cry for me") would be written:

  ```
  Peter&amp;Tom("Don&amp;apos;t cry for me")
  ```
XML : Why use Entity References?

We use Entity References ...
• ...to use symbols we could not use otherwise
• ...to make maintenance easy and scalable
• ...out of comfort

More about this when we speak about DTDs
XML: Comments

Comments are a special set of tags that start with `<!--` and end with `-->`

All data written between these two tags is ignored by the XML processor.

Comments are usually used to make small notes inside the XML document or to comment out entire sections of XML code.

```xml
<!-- I HAVE TO GET GUSTAVS EMAIL
<employee name="Gustav Sielmann">
  <email/>
</employee>
-->```
XML: Well-formedness and Validity - DTDs

With an XML parser an XML document can be checked for two things:

- **Well-formedness** i.e. if the document obeys the syntactical rules of XML (has a prolog, a document element, all elements closed, no overlaps...)

- **Validity** i.e. if the document obeys the rules in a DTD (or in an XML-Schema, next lecture) – Grammar!
Which elements should be used?
Which content models are allowed?
How elements are connected?
Which attributes are valid for an element?
Which data should an attribute contain?
XML : Document Type Definitions

- DTDs (Document Type Definitions) contain a list of elements (tags), attributes and entity references contained in an XML document and describes their relationships to each other.
- A DTD specifies a set of rules for the structure of a document therefore making it easy to share data with everyone that conforms to the same encoding standard.
- DTDs are part of the XML standard, but inherited from SGML.
XML: Why use DTDs?

We use DTDs ...

- ...to define a **grammar** for one of several XML documents
- ...to check XML documents against this grammar
XML: DTDs

Document Type Definitions can be:

- **Internal**, i.e. placed in the prolog of an XML document *or*

- **External**, being identified by an URL, thus making it easy to share the same encoding standard with other people
XML : Structure of a DTD

- A DTD always starts with `<!DOCTYPE` and always ends with `>`.
- Directly after the `<!DOCTYPE` comes the name of the document (root) element followed by a bracket `[` or the `SYSTEM` or `PUBLIC` keyword.
- Then comes a list of all elements and attributes contained in the XML file, including the document element or a reference to an external DTD.
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE teaching SYSTEM "asignaturas.dtd">
<!-- URJC -->
<teaching>
  <course jahr="2007">
    <title>Recuperación de Información</title>
    <keyword>XML</keyword>
    <keyword>Web Services</keyword>
    <keyword>Integration</keyword>
    <keyword>Semantic Web</keyword>
    <start>
      <date>2007-02-22</date>
      <time timezone="CET">15:00:00</time>
      <place>Aulario I, Aula 002</place>
    </start>
  </course>
</teaching>
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE teaching [ 
<!ELEMENT teaching (course+)>
<!ELEMENT course ... 
<!ATTLIST course ... 
... }
]
!-- URJC -->
<teaching>
<course jahr="2007">
  <title>Recuperación de Información</title>
  <keyword>XML</keyword>
  <keyword>Web Services</keyword>
  <keyword>Integration</keyword>
  <keyword>Semantic Web</keyword>
  <start>
    <date>2007-02-22</date>
    <time timezone="CET">15:00:00</time>
    <place>Aulario I, Aula 002</place>
  </start>
</course>
</teaching>
Use of DTD

Embedding in XML

Document-Type-Declaration: `<!DOCTYPE ...>`

Name of the document elements must be "teaching"

`<!DOCTYPE teaching SYSTEM "asignaturas.dtd">`

Described by DTD data (System) or by public identifiers; when public identifier is given, there exists a SYSTEM identifier, which is used if the public link is can not be identified:

`<!DOCTYPE teaching PUBLIC "-/Teaching//URJC//ES" "http://www.urjc.es/asignaturas.dtd">` is not an attribute value!

In an DTD different potential document (root) elements are given, which can be chosen in a document instance.
Public vs. System Identifier

**System identifier is a "local" identifier**
- not necessarily local URLs, e.g.
  ```xml
  <!DOCTYPE seminar SYSTEM "http://www.seminar.se/se.dtd">
  ```
- for all non-standard documents

**Public identifier**
- must be known to the used XML Processor
- for known document types
- must follow a defined syntax
- can take another value, in case the SYSTEM identifies that the first value can not be applied:
  ```xml
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "xhtml11-flat.dtd">
  ```
XML: Example of a DTD

```xml
<employees>
  <marketing>
    <employee id="1834">
      <name>Gustav Sielmann</name>
      <email>gsielmann@Dot.com</email>
      <tel>+43/0662/723942-124</tel>
      <fax>+43/0662/723942-800</fax>
    </employee>
  </marketing>
</employees>
```

To validate: each used element and attribute have to be defined in the DTD!!!
Valid Documents: summary

- **Well-formed vs. valid**
  - Well-formed: following XML syntax
    - XML processor checked only with syntax rules
  - valid: valid in correspondence to DTD (or XML schema)
  - validating Parser
    - validates over DTD (or another language)

- **Elements, attributes, entities declaration**
  - DTD helps to distinguish desired from undesired elements/attributes/content

- **Document structure**
  - Needed for a validating XML Processor
  - Instances are filled in document structure
  - XML data can be shortened with different definitions

- **External/Internal declaration of DTD**
  - directly in document
  - in its own data
DTDs in detail...

- Element declarations
- Attribute declarations
- Internal/external Entity declarations
Element declaration

- An element can contain other elements or only text or a mixed content
- Shortening opportunities in DTDs
  - any element-content
  - empty element-content
  - appearance of inner elements
  - no multiple types, only strings
- Possible content models: only text, only elements, mixed, any, empty.
Element declaration: syntax

Key words
- #PCDATA characterizes leaf-elements (parseable character data)
- EMPTY: empty element
- ANY: any content (the elements can be defined in DTD, in XML schema or some extent of freedom possible)

Indicators of appearance
- Regular expressions +,*,?
- Without the expressions: precisely one time
- +: at least one time or possibly any times more
- *: 0-time or more
- ?: 0-time or one time

Grouping
- With parentheses (...).
<!ELEMENT teaching ANY>

*Instance:*

```xml
<teaching>irgend<was>steht</was></teaching>
```

<!ELEMENT teaching EMPTY>

*Instance:*

```xml
<teaching/>
```

<!ELEMENT course (name, jahr)>

*Instance:*

```xml
<course>
  <name>Seminar</name><jahr>2002</jahr>
</course>
```

*Not an instance:*

```xml
<course>
  <jahr>2002</jahr><name>Seminar</name><
</course>
```

<!ELEMENT teaching (#PCDATA)>

*Instance:*

```xml
<teaching>Seminar</teaching>
```

*Not an instance:*

```xml
<teaching>
  <jahr>2002</jahr><name>Seminar</name><
</teaching>
Element declaration: Syntax

- Connection operators: "," (=AND), "|" (=OR)
- With "," element order is important
- "|" means: only one of the following is applied (excluding or)
- "|" repeats with * or + allows thus arbitrary order (and multiple appearance)
- Enclosed bracketing
  - E.g. (lname, (fname | title))
Element declaration: mixed content

- Mixed content (elements + character data) vs. Element content
  - Element content: contents only elements
  - Example: `<!ELEMENT start (date, time, place)>`

- mixed content
  - always only separated with | in DTD and * at the end
  - Example: `<!ELEMENT name (#PCDATA | fname | lname)>`

- #PCDATA should be listed firstly
  - and only with | and * if mixed content is located inside!
    - `<content>This is <i>my</i> Content</content>`

- #PCDATA can not be used with ","
  - `<!ELEMENT teaching (#PCDATA, course)>`
    - Not valid
Element declaration: further examples

```xml
<!ELEMENT teaching (course+)>
```

teaching is a list of courses (at least one in this example).

```xml
<!ELEMENT booktitle (deutsch | english* | italiano)>
```

booktitle consists of a German or several English or an Italian titles.

```xml
<!ELEMENT name (#PCDATA | fname | lname)*)>
```

Name consists of first name, last name or a convenient mixture of text, order doesn’t matter.

```xml
<!ELEMENT course ((name, year?)*)>
```

List of names and years, setting „year“ is optional.
Attribute declaration

- An attribute has a name, a type and options
- Declared via attribute declaration lists
- All attributes can be in one or more declarations
- Relates to corresponding Element
- Its place does not make any difference

\([\text{Name}, \text{Type}, [\text{Options}]] \text{ List}\)
Attribute declaration: example

```xml
<!ELEMENT time (#PCDATA)>
<!ATTLIST time sine-tempore (yes|no) "no">

„time“ contains an Attribute. The Attribute sine-tempore can have values „yes“ or „no“, and „no“ is a default value.

<!ATTLIST test href CDATA #REQUIRED>

The href attribute must be defined in test and can contain any string value.

<!ATTLIST test lang NMTOKEN #IMPLIED>

If this special attribute is defined in a valid document, then it should be described in a defined way (here NMTOKEN (=name token) for attribute xml:lang)
Attribute declaration: syntax

```xml
<!ELEMENT time (#PCDATA)>
<!ATTLIST time sine-tempore (yes|no) "no"/>
```

- **Element name**
  - E.g. "time"
- **Attribute name**
  - E.g.: "sine-tempore"
- **Type (in principal, only string type)**
  - E.g. CDATA, NMTOKEN, ID, ...
- **Default value (optional)**
  - E.g. "no"
- **Options**

Attribute values can be more restrictive than element values
Attribute - Limitations

- **String-Attribute**
  - contains Text
  ```xml
  <!ATTLIST email href CDATA #REQUIRED>
  ```

- **Tokenized Attribute**
  - Content limitations
  ```xml
  <!ATTLIST entry id ID #IMPLIED>
  ```

- **Default value attributes**
  - Accept only one value from the list
  ```xml
  <!ATTLIST entry preferred (true|false) "false">
  ```
Values for attributes of various types:

- CDATA – Character Data, String
- ENTITY: has a general entity name as a value
- ENTITIES: List of entity names as a value
- NM_TOKEN: string – any allowed name in XML (starts with letter or underscore, no special characters, no spaces, etc.)
- NM_TOKENS: List of NM_TOKENs
  - Separation by spaces
- Lists of NM_TOKENS
  - Lists are represented as (RB|GG|CC), (yes|no), etc...
- ID – references (same syntactic restrictions as for NM_TOKEN, but unique in the document)
- IDREF, IDREFS – references to IDs
- Default value: optional; is applied when an attribute does not appear
Attribute options:

#REQUIRED: There must be a value for this attribute
#IMPLIED: optional, no default value
#FIXED “Literal”: the attribute value in the document is as specified in the DTD (no difference whether you define it in the document or not!)
“Literal”: this value will be taken by default

Default- and fixed-attributes add values to a document; make entities
Further Examples

<!ATTLIST time sine-tempore (yes|no) "no">
....
<!ATTLIST place building CDATA #IMPLIED>
....
<!ATTLIST course number ID #REQUIRED>
<!ATTLIST time belongsTo IDREFS #REQUIRED>
....
ID, IDREF

- ID is an attribute that gives an element a label guaranteed to be unique in the document
- ID must be an NMTOKEN, i.e. a valid XML name
- IDREF is similar to ID, but refers to the ID of another element
- IDREF is used for linking within a document
- IDREFS is a list of IDREFs: separated with empty lines
- Consider: XML tree vs. XML graph

```
<course number="_1">
    <place building="Favoritenstrasse">Seminarr.184/2</place>
</course>

....
<time belongsTo="_1" sine-tempore="yes">14:00</time>
```
Entity declarations –
General Entities:

The GENERAL ENTITY Declaration:

- The types of general entities include:
  - INTERNAL (PARSED)
  - EXTERNAL (PARSED)
  - EXTERNAL (UNPARSED)
INTERNAL (PARSED) GENERAL ENTITY:

```xml
<!ENTITY name "entity_value">
```

Entity_value: quoted string not containing special characters
`&`, `%`, `"`, `;`

**Example:**
```xml
<?xml version="1.0" ?>
<!DOCTYPE author [ 
  <!ELEMENT author (#PCDATA)> 
  <!ENTITY js "Jo Smith"> ]> 
<author>&js;</author>
```
EXTERNAL (PARSED) GENERAL ENTITY:

```xml
<!ENTITY name SYSTEM "URI">
<!ENTITY name PUBLIC "public_ID" "URI">
```

- URI, public_ID, similar as in external DTD definition.
- Reference text which is shared among different documents

**Example:**
```xml
<?xml version="1.0" ?>
<!DOCTYPE copyright [
  <!ELEMENT copyright (#PCDATA)>
  <!ENTITY c SYSTEM "http://www.xmlwriter.net/copyright.xml">
]>
<copyright>&c;</copyright>
```

**or:**
```xml
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE copyright [
  <!ELEMENT copyright (#PCDATA)>
  <!ENTITY c PUBLIC "-//W3C//TEXT copyright//EN"
  "http://www.w3.org/xmlspec/copyright.xml">
]>
<copyright>&c;</copyright>
```
EXTERNAL (UNPARSED) GENERAL ENTITY:

<!ENTITY name SYSTEM "URI" NDATA name>
<!ENTITY name PUBLIC "public_ID" "URI" NDATA name>

Example:
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE img [ 
  <!ELEMENT img EMPTY> 
  <!ATTLIST img src ENTITY #REQUIRED> 
  <!ENTITY logo PUBLIC "-//W3C//GIF logo//EN" "http://www.w3.org/logo.gif" NDATA gif> 
  <!NOTATION gif PUBLIC "gif viewer"> ]>
Entity declarations – Parameter Entities:

The PARAMETER ENTITY Declaration:

The types of general entities include:

- INTERNAL (PARSED)
- EXTERNAL (PARSED)

These define entities only to be used within the DTD.
PARAMETER ENTITIES: %

**Internal** parameter entities define macros WITHIN the DTD, not to be used in the XML markup, e.g.:

```xml
<!ENTITY % p "(#PCDATA)">
<!ELEMENT student (id,surname,firstname)>
<!ELEMENT id %p;>
<!ELEMENT surname %p;>
<!ELEMENT firstname %p;>
```

**External** parameter entity references are used to link external DTDs, e.g.:

```xml
<?xml version="1.0" ?>
<!DOCTYPE student [
  <!ENTITY % student SYSTEM "http://www.university.com/student.dtd">
%student; ]>
```
So, we have...

XML Document

Document Type Declaration

Document Type-Definition (DTD)
- element, attribute,
- general entity,
- parameter entity,
- parameter entity reference

Process Instructions (wie Stylesheetverwendung)

Document Element  with/without namespace declaration

Document
- Start-Tag, End-Tag, empty-Element-Tag, PCDATA,
- general entity references, CDATA blocks*

*e.g.:  <![CDATA tterter& http://xyz.com dasdw]>  not “touched” by the parser
XML: Companion Standards

- **XML Namespaces** allow for modular document definition, multiple inheritance and collision avoidance
- **XPath** or the XML Path Language allows navigation of the document tree
- **XPointer** allows tree components as targets
- The **XML Linking Language** defines linking capability
XML: Companion Standards

- The **XML Style Language** defines presentation capability.
- **XSLT** provides for the transformation of documents.
- **XML Schema** to allow DTDs to be defined as XML documents and to define custom data types in order for content value control.
XML : XML Namespaces

- The XML namespaces recommendation defines a way to **distinguish between duplicate element type and attribute names**.
- An XML namespace is a collection of element type and attribute names. The namespace is identified by a unique name, which is a URI.
- XML namespaces are declared with an `xmlns` attribute, which can associate a **prefix** with the namespace.
Why namespaces?

```xml
<?xml version="1.0"?>
<clases xmlns="http://www.alumnos.info/clases/ns/">
...
<asignatura>
  <titulo>Telecooperation</titulo>
  <tipo>interesante! :-)</tipo>
</asignatura>
...
</clases>
```

Same element, different meaning!

```xml
<?xml version="1.0"?>
<clases xmlns="http://www.urjc.es/horario/ns/">
...
<asignatura>
  <titulo>Telecooperation</titulo>
  <tipo>lecture</tipo>
</asignatura>
...
</clases>
```
Namespaces, scope:

```xml
<?xml version="1.0"?>
<teaching xmlns:stud="http://www.alumnos.info/clases/ns/
   xmlns:urjc="http://www.urjc.es/horario/ns/
   xmlns="http://www.polleres.net/ri2007-namespace#">
...

<course>
   <title>Recuperación de Información</title>
   <beschreibung xmlns="http://www.elsewhere.org/test">
      <title>Informationsverarbeitung</title>
      <themen>XML, Semantic Web, Web Services</themen>
   </beschreibung>
   <stud:tipo>interesante! :-) </stud:tipo>
   <urjc:tipo>lecture</urjc:tipo>
</course>
...
</teaching>
```

Elements and attributes in the subelements lie in the newly defined default namespace.
For XML, this is just the same:

```xml
<?xml version="1.0"?>
<teaching xmlns:stud="http://www.alumnos.info/clases/ns/"
          xmlns:urjc="http://www.urjc.es/horario/ns/"
          xmlns="http://www.polleres.net/ri2007-namespace#"
          xmlns:xyz="http://www.elsewhere.org/test">

  ...

  <course>
    <title>Recuperación de Información</title>
    <xyz:beschreibung>
      <xyz:title>Informationsverarbeitung</xyz:title>
      <xyz:themen>XML, Semantic Web, Web Services</xyz:themen>
    </xyz:beschreibung>
    <stud:tipo>interesante! :-)</stud:tipo>
    <urjc:tipo>lecture</urjc:tipo>
  </course>

  ...

</teaching>
```
Namespaces

Don’t be confused by the use of URI! This is not necessarily a real resource but just a unique identifier!

- Important: A URI is not necessarily a document!!!

ns can be overridden

Prefixes: A namespace-aware XML application shall only use URI, not the prefix!

i.e. a:element = b:element if a and b refer to the same URI
Some XML Tools
(many others available of course...)

XML browsers
- IE 5 supports XSL incompletely
- Mozilla (Open Source), Netscape
- InDelv XML Browser
- Opera 6.0
- Amaya (W3C Browser/Editor) (also MathML support!, no XSLT)

XML editor types
- Text-based with syntax highlight
- Tree-based
- pseudo-wysiwyg (for example, XMetaL uses Stylesheets for displaying)

XML editors
- XMLSpy (www.xmlspy.com)
  - Visual DTD/Schema development
  - very extensive tool; but only for Windows! 😊
  - 30 days version available
- XMetaL and XML Authority
- Adobe Framemaker (SGML)
- XML Notepad (simple, MS)
- Microstar (XML Modelling Tools)

XML Parser
- Apache Xerces

XSLT
- E.g., Apache Xalan

Databases
- Tamino, Poet, Cache, ...
- native vs. xml-enabled
References

- **XML**: [http://www.w3.org/XML/](http://www.w3.org/XML/)


- **Lectures by Robert Baumgartner (TU Vienna)**: [http://www.dbai.tuwien.ac.at/staff/baumgart/](http://www.dbai.tuwien.ac.at/staff/baumgart/)

- **Good Editor with validation**: **XMLSpy** (unfortunately commercial) [http://www.xmlspy.com](http://www.xmlspy.com)

- **Quick tutorials on most important W3C standards** (including XML, and companion standards): [http://www.w3schools.com/](http://www.w3schools.com/)