Module 2

XML Basics
(XML, Namespaces,
Usage scenarios, DTDs)
History: SGML vs. HTML vs. XML

- SGML (1960)
- XML (1996)
- HTML (1990)
- XHTML (2000)

http://www.w3.org/TR/2006/REC-xml-20060816/
Why XML?

- HTML is to be interpreted by browsers
  - Shown on the screen to a human
- Desire to separate the “content” from “presentation”
  - Presentation has to please the human eye
  - Content can be interpreted by machines, for machines presentation is a handicap
- Semantic markup of the data
Information about a book in HTML

<td><h1 class="Books">Politics of experience by Ronald Laing, published in 1967</h1></td><td align="right" nowrap> Item number:320070381076</td><td align="right" valign="top"><img src="http://pics.booksstatic.com/aw/pics/globalAssets/rtCurve.gif" width="8" height="8"></td><td colspan="6" valign="middle" bgcolor="#5F66EE"><img src="http://pics.booksstatic.com/aw/pics/s.gif" width="1" height="4"></td></tr><tr><td bgcolor="#CCCCFF"><img src="http://pics.booksstatic.com/aw/pics/s.gif" width="1" height="1"></td><td bgcolor="#EEEEFF"><div id="FastVIPBIBO"><table border="0" cellpadding="0" cellspacing="0" width="100%">
The same information in XML

```xml
<book year="1967">
  <title>Politics of experience</title>
  <author>
    <firstname>Ronald</firstname>
    <lastname>Laing</lastname>
  </author>
</book>
```

- Information is (1) decoupled from presentation, then (2) chopped into smaller pieces, and then (3) marked with semantic meaning
- It can be processed by machines
- Like HTML, only syntax, not logical abstract data model
XML key concepts

- Documents
- Elements
- Attributes
- Namespace declarations
- Text
- Comments
- Processing Instructions
- All inherited from SGML, then HTML
The key concepts of XML

```xml
<book year="1967">
  <title>Politics of experience</title>
  <author>
    <firstname>Ronald</firstname>
    <lastname>Laing</lastname>
  </author>
</book>
```

- Documents
- Elements
- Attributes
- Text
- Nested structure
- Conceptual tree
- Order is important
- Only “characters”, not integers, etc.
Elements

- Enclosed in Tags
  - Begin Tag: e.g., `<bibliography>`
  - End Tag: e.g., `</bibliography>`
  - Element without content: e.g., `<bibliography />` is a shorthand for `<bibliography> </bibliography>`

- Elements can be nested
  `<bib> <book> Wilde Wutz </book> </bib>`

- Subelements can implement multisets
  `<bib> <book> ... </book> <book> ... </book> </bib>`

- Order is important!

- Documents must be well-formed
  `<a> <b> </a> </b>` is forbidden!
  `<a> <b> </b> </a>` is forbidden!
Attributes

- Attribute are associated to Elements
  
  ```xml
  <book price = "55" year = "1967">
    <title> ... </title>
    <author> ... </author>
  </book>
  ```

- Elements can have only attributes
  
  ```xml
  <person name = "Wutz" age = "33"/>
  ```

- Attribute names must be unique! (No Multisets)
  
  ```xml
  <person name = "Wilde" name = "Wutz"/>
  ``` is illegal!

- What is the difference between a nested element and an attribute? Are attributes useful?

- Modeling decision: should "name" be an attribute or a subelement of a person? What about "age"?
Text and Mixed Content

- Text appears in element content
  - `<title>The politics of experience</title>`
- Can be mixed with other subelements
  - `<title>The politics of <em>experience</em></title>`
- Mixed Content
  - For „documents“ data -- very useful
  - The need does not arise in „data“ processing, only entities and relationships
  - People speak in sentences, not entities and relationships. XML allows to preserve the structure of natural language, while adding semantic markup that can be interpreted by machines.
Continuous spectrum between natural language, semi-structured data, and structured data

1. Dana said that the book entitled “The politics of experience” is really excellent!

2. <citation author="Dana"> The book entitled “The politics of experience” is really excellent! </citation>

3. <citation author="Dana"> The book entitled <title>The politics of experience</title> is really excellent! </citation>

4. <citation>
   <author>Dana</author>
   <aboutTitle>The politics of experience</aboutTitle>
   <rating> excellent </rating>
</citation>
CDATA sections

- Sometimes we would like to preserve the original characters, and not interpret them as markup
- CDATA sections
  - Not parsed as XML
- `<message>
  <greeting>Hello, world!</greeting>
</message>`

- `<message> <![CDATA[<greeting>Hello, world!</greeting>]]> </message>`
Comments, PIs, Prolog

- Comment: Syntax as in HTML
  
  <!-- this is a comment -->

- Processing Instructions
  - Contain no data - interpretation by processor
  - Syntax: `<?pause 10 secs ?>`
  - Pause is „Target“; 10secs is „Content“
  - XML is a reserved target for prolog

- Prolog
  
  `<?xml version=“1.0“ encoding=“UTF-8“ standalone=“yes“ ?>`
  - Standalone defines whether there is a DTD
  - Encoding is usually Unicode.
Whitespaces declaration

- Whitespace = Continuous sequence of **Space**, **Tab** and **Return** character
- Special Attribute **xml:space** to control use
- Human-readable XML (with Whitespace)
  ```xml
  <book xml:space="preserve">  
  <title>The politics of experience</title>  
  <author>Ronald laing</author>  
  </book>
  ```
- (Efficient) machine-readable XML (no WS)
  ```xml
  <book xml:space="default">  
  <title>The politics of experience</title>  
  <author>Ronald Laing</author>  
  </book>
  ```
- Performance improvement: ca. Factor 2.
The quick brown fox jumps over the lazy dog.

What colour is it?

What color is it?
Universal Resource Identifiers on the Web

- **URLs, URIs, IRIs**
- **URL (Universal Resource Locators):** deferenceable identifier on the Web
  - The target of an URL pointer is an HTML file (virtual or materialized)
- **URIs (Unique Resource Identifier):** general purpose key to resources on the Web
  - Uniquely identifies a resource
  - Target is not an HTML file, can be anything (schema, table, file, entity, object, tuple, person, physical item, etc)
  - Lifetime and scope of this “key” is user dependent
- **IRI (Internationalized Resource Identifiers)**
  - Allow non Latin characters (Chinese, Arabic, Japanese, etc)
- **URL, URI, IRIs**
  - All strings
  - Very LONG strings
Namespaces

- Integration of Data from diverse data sources
- Integration of different XML Vocabularies (aka Namespaces)
- Each „vocabulary“ has a unique key, identified by a URI/IRI
- Same local name, from different vocabularies can have
  - Different meaning
  - Different structure associated with it

Qualified Names (Qname) to attach a „name“ to its „vocabulary“
- for all nodes in an XML document that has names (Attributes, Elements, PIs

\[
\text{QName ::= triple ( URI [ prefix: ] localname )}
\]
- Binding (prefix, URI) is introduced in elements start tag
- Later only the prefix is used, not the long URIs
- Prefix is optional, default namespaces
- Prefix and localname a separated by „:“

„http://w3.org/TR/1999/REC-xml-names“
Namespaces (cont)

- Namespace definitions look like Attributes
  - Identified by "xmlns:prefix" or "xmlns" (default)
  - Bind the Prefix to the URI
- Scope is the entire element where the namespace is declared
  - Includes the element itself, its attributes and its subtrees
- Example
  
  ```xml
  <ns:a xmlns:ns="someURI" ns:b="foo">
    <ns:b>content</ns:b>
  </ns:a>
  ```
Default namespaces

- Default namespaces, no prefix
  `<a xmlns="someURI" >
    <b/> <!-- a and b are in the someURI namespace! -->
  </a>`

- Only applies to subelements, not attributes
  `<a xmlns="someURI" c = "not in someURI namespace">
    <b/> <!-- a and b are in the someURI namespace! -->
  </a>`
Example: Namespaces

- DQ1 defines *dish* for *china*
  - Diameter, Volume, Decor, ...

- DQ2 defines *dish* for *satellites*
  - Diameter, Frequency

- How many „dishes“ are there?

- Better ask for:
  - „How many *dishes* are there?“
  - or
  - „How many *dishes are there?“
Example: Namespaces

<gs:dish xmlns:gs = "http://china.com" >
  <gs:dm gs:unit = "cm">20</gs:dm>
  <gs:vol gs:unit = "l">5</gs:vol>
  <gs:decor>Meissner</gs:decor>
</gs:dish>

<sat:dish xmlns:sat = "http://satelite.com" >
  <sat:dm>200</sat:dm>
  <sat:freq>20-2000MHz</sat:freq>
</sat:dish>
Mixing Several Namespaces

<gs:dish xmlns:gs = "http://china.com"
         xmlns:uom = "http://units.com">
  <gs:dm uom:unit = "cm">20</gs:dm>
  <gs:vol uom:unit = "l">5</gs:vol>
  <gs:decor>Meissner</gs:decor>
  <comment>This is an unqualified element name</comment>
</gs:dish>
Example XML data

- XHTML (browser/presentation)
- RSS (blogs)
- UBL (Universal Business Language)
- HealthCare Level 7 (medical data)
- XBRL (financial data)
- Digital photography metadata (XMP)
- XMI (metadata)
- XQueryX (programs)
- XForms (forms)
- SOAP (message envelopes)
- Microsoft Office -- Powerpoint in XML (documents)
QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://purl.org/rss/1.0/">
  <channel
    rdf:about="http://www.xml.com/xml/news.rss">
    <title>XML.com</title>
    <link>http://xml.com/pub</link>
    <description>
      XML.com features a rich mix of information and services for the XML community.
    </description>
    <image
    <items>
      <rdf:Seq>
        <rdf:li
        <rdf:li
      </rdf:Seq>
    </items>
    <textInput
      rdf:resource="http://search.xml.com"/>
  </channel>
</rdf:RDF>
UBL (Universal Business Language)

- Vocabularies definitions for:
  - ApplicationResponse
  - AttachedDocument
  - BillOfLading
  - Catalogue
  - CatalogueDeletion
  - CatalogueItemSpecific
  - CatalogueUpdate
  - CataloguePricingUpdate
  - CatalogueRequest
  - CertificateOfOrigin
  - CreditNote
  - DebitNote
  - DespatchAdvice
  - ForwardingInstructions
  - FreightInvoice
  - Invoice
  - Order
  - OrderCancellation
  - OrderChange
  - OrderResponse
  - OrderResponseSimple
  - PackingList
  - Quotation
  - ReceiptAdvice
  - Reminder
  - RemittanceAdvice
  - RequestForQuotation
  - SelfBilledCreditNote
  - SelfBilledInvoice
  - Statement
  - TransportationStatus
  - Waybill
HealthCareLevel 7

- Medical information that is being exchanged between hospitals, patients, doctors, pharmacies and insurance companies
XBRL (Financial information)

- Goal: facilitate the exchange of business and financial performance information between companies, governments, insurance companies, banks, etc.
- Mandate by law in many countries
Extensible Metadata Platform (XMP)

- Used in PDF, photography and photo editing applications.
- Particular schemas for basic properties useful for recording the history of a resource as it passes through multiple processing steps, from being photographed, scanned, or authored as text, through photo editing steps (such as cropping or color adjustment), to assembly into a final image.
- XMP allows each software program or device along the way to add its own information to a digital resource, which can then be retained in the final digital file.
Microsoft Office in XML

- **Office 2003** was able to import/export all documents into XML
- **Office 2007** models the documents NATIVELY in XML
- **Examples of vocabularies and schemas:**
  - WordprocessingML (the XML file format for Word 2003), SpreadsheetML (Excel 2003), FormTemplate XML schemas (InfoPath 2003) and DataDiagramingML (Visio 2003)
Forms on the Web in XML

- XML Forms (Xforms)
  - http://www.w3.org/TR/xforms/

```xml
<xforms:model>   <xforms:instance>
    <ecommerce xmlns=""">   <method/>
        <number/>   <expiry/>
    </ecommerce>   </xforms:instance>
<xforms:submission
action="http://example.com/submit"
method="post" id="submit"
</xforms:model>
```
Programs and queries in XML

- XQuery, the XML query language, has an XML representation
- Programs and queries are also DATA
- Blurring the distinction between data, metadata, code

```xml
<qxml:functionName>distinct</qxml:functionName>
  <qxml:parameters>
    <qxml:expr xsi:type="qxml:pathExpr">
      <qxml:expr xsi:type="qxml:functionCallExpr">
        <qxml:functionName>document</qxml:functionName>
        <qxml:parameters>
          <qxml:expr xsi:type="qxml:stringConstantExpr">
            <qxml:value>http://www.bn.com</qxml:value>
          </qxml:expr>
        </qxml:parameters>
      </qxml:expr>
    </qxml:expr>
  </qxml:parameters>
  <qxml:expr>
    <qxml:stepExpr>
      <qxml:xpathAxis>descendant-or-self</qxml:xpathAxis>
      <qxml:elementTest>
        <qxml:nodeName>
          <qxml:QName>author</qxml:QName>
        </qxml:nodeName>
      </qxml:elementTest>
    </qxml:stepExpr>
  </qxml:expr>
</qxml:expr>
```
SOAP and Web Services

- Web Services is the favorite way of exchanging information between applications
- XML exchange over HTTP, with a specific protocol (SOAP)

```xml
<?xml version='1.0' ?>
<env:Envelope
xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header>
<m:reservation
xmlns:m="http://travelcompany.example.org/reservation"
    xmlns:env="http://www.w3.org/2003/05/soap-envelope" env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
    env:mustUnderstand="true">
    <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
    <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
</m:reservation>
<n:passenger
xmlns:n="http://mycompany.example.com/employees"
    xmlns:env="http://www.w3.org/2003/05/soap-envelope/role/next"
    env:mustUnderstand="true">
<n:name>Åke Jógvan Øyvind</n:name>
</n:passenger>
</env:Header>
<env:Body/>
</env:Envelope>
```
The need for XML “schemas”

- Unlike any other data format, XML is totally flexible; elements can be nested in arbitrary ways.
- We can start by writing the XML data -- no need for a priori design of a schema:
  - Think relational databases, or Java classes.
- However, schemas are necessary:
  - Facilitate the writing of applications that process data.
  - Constraint the data that is correct for a certain application.
  - Have a priori agreements between parties with respect to the data being exchanged.
- Schema: a model of the data:
  - Structural definitions.
  - Type definitions.
  - Defaults.
History and role of XML Schema Languages

- Several standard Schema Languages
  - DTDs, XML Schema, RelaxNG
- Schema languages have been designed after, and in an orthogonal fashion, to XML itself
- Schemas and data are completely decoupled in XML
  - Data can exist with or without schemas
  - Or with multiple schemas
  - Schema evolutions rarely impose evolving the data
  - Schemas can be designed before the data, or extracted from the data (DataGuide -- Stanford)
- Makes XML the right choice for manipulating semi-structured data, or rapidly evolving data, or highly customizable data
DTDs

- Inherited from SGML
- Part of the original XML 1.0 specification
- Describe the “grammar” of the XML file
  - **Element declarations:** how elements are allowed to nest within each other by rules and constraints
  - **Attributes lists:** describe what attributes are allowed on which element
  - Some constraints on the value of elements and attributes
  - Which is the root element of the XML file
- Checking the structural constraints: **DTD validation** (valid vs. invalid documents)
- DTD very useful for a while, not used anymore, several major limitations
Declaring the structure of elements

- Grammar that describes the structure of the element
  - Subelements, identified by Name or
  - #PCDATA
- Combinators:
  - "+" for at least 1
  - "*" for 0 or more
  - "?" for 0 or 1
  - " , " for concatenation
  - " | " for choice
- `<!ELEMENT a ( (b | c)* , d ? , e ) >`
- PCDATA: only textual content allowed
  - `<!ELEMENT a #PCDATA>`
- EMPTY : the element must be empty
  - `<!ELEMENT a EMPTY>`
- ANY: allows any content
  - `<!ELEMENT a ANY>`
Example DTD for recipes

```xml
<!ELEMENT collection (description,recipe*)>
<!ELEMENT description ANY>
<!ELEMENT recipe (title,ingredient*,preparation,comment?,nutrition)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT ingredient (ingredient*,preparation)?>
<!ELEMENT preparation (step*)>
<!ELEMENT step (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT nutrition EMPTY>
```
Defining the attribute lists

- Structure: `<!ATTLIST ElementName definition>`

- `<!ATTLIST ingredient name CDATA #REQUIRED
amount CDATA #IMPLIED
unit CDATA #FIXED „cup“ >`

- CDATA means normal content

- `#REQUIRED, or #IMPLIED` refer to the fact that the attribute is optional or not

- Default value possible
Attributes (cont.)

- **#REQUIRED**
  - Document must specify a value for attribute

- **#IMPLIED**
  - Attribute is optional, there is no default

- **value**
  - Default value, if no other value specified

- **#FIXED value**
  - Default value, if no other value specified
  - If value specified, it must be the fixed value
Major attribute types

- PCDATA: normal Text content
- ID
  - Value is unique within document
  - Element has at most one attribute of this type
  - No default values allowed
- IDREF, IDREFS
  - References to other elements within the document
  - IDREFS: Enumeration, " " as separator
### ID and IDREF attributes

- `<!ATTLIST book
  isbn   ID    #REQUIRED
  price  CDATA  #IMPLIED
  index  IDREFS "" >`

- `<book id="1" index="2 3" />
  <book id="2" index="3"/>
  <book id ="3"/>`
Attributes list example

```xml
<!ELEMENT ingredient (ingredient*,preparation)?>
<!ATTTLIST ingredient name CDATA #REQUIRED
amount CDATA #IMPLIED
unit CDATA #IMPLIED >

<!ELEMENT nutrition EMPTY >
<!ATTTLIST nutrition protein CDATA #REQUIRED
carbohydrates CDATA #REQUIRED
fat CDATA #REQUIRED
```
Mixed content in DTDs

- Mixing PCDATA declarations with other subelements means that the content can be “mixed”

<!ELEMENT p (#PCDATA | a | ul | b | i | em) >

<p>some text <em>some emphasized text</em> blah <b>some bold text</b> </p>
Declarations of DTDs

- No DTD (well-formed Documents)
- DTD inside the Document:
  `<!DOCTYPE name [definition] >`
- DTD external, specified by URI:
  `<!DOCTYPE name SYSTEM „demo.dtd“>`
- DTD external, Name and optional URI:
  `<!DOCTYPE name PUBLIC „Demo“>`
  `<!DOCTYPE name PUBLIC „Demo“ „demo.dtd“>`
- DTD inside the document + external:
  `<!DOCTYPE name1 SYSTEM „demo.dtd >`
Correctness of XML documents

- **Well formed** documents
  - Verify the basic XML constraints, e.g. `<a></b>`

- **Valid documents**
  - Verify the additional DTD structural constraints

- Non well formed XML documents cannot be processed
- Non-valid documents can still be processed (queried, transformed, etc)
Limitations of DTDs

- DTDs describe only the “grammar” of the XML file, not the detailed structure and/or types.

- This grammatical description has some obvious shortcomings:
  - we cannot express that a “length” element must contain a non-negative number (constraints on the type of the value of an element or attribute).
  - The “unit” element should only be allowed when “amount” is present (co-occurrence constraints).
  - the “comment” element should be allowed to appear anywhere (schema flexibility).
Good Schema design principles

- The XML schema language shall be:
  1. more expressive than XML DTDs
  2. expressed in XML
  3. self-describing
  4. usable by a wide variety of applications that employ XML
  5. straightforwardly usable on the Internet
  6. optimized for interoperability
  7. **simple** enough to be implemented with modest design and runtime resources
  8. coordinated with relevant W3C specs
Recapitulation

- XML as inheriting from the Web history
  - SGML, HTML, XHTML, XML
- XML key concepts
  - Documents, elements, attributes, text
  - Order, nested structure, textual information
- Namespaces
- XML usage scenarios
  - Financial, medical, metadata, blogs, etc
- DTDs and the need for describing the “structure” of an XML file
- Next: XML Schemas