Extensible Markup Language, (XML), XML Schema and XSLT

BWA Slides #1

XML technologies

Huge family of technologies and languages

- XML, XML Schema, XPath, XLink, XPointer, XSLT, XHTML, XML Encryption, ...

We'll examine a few

- XML, XML Schema, XSLT (with a little XPath) and later XHTML
- Basic ideas usually no problem but details are creepy ...
- NetBeans has some support

We will not actively use XML but it "pops up" ...

The Problem

Passing data between heterogeneous systems (internet)

- Applications must understand the "format"
- Should be an open format (non proprietary)
- Should be possible to extend/customize format

Binary formats

- May change radically

Text formats

- May change, but the alphabet will not!
- Human readable, (possible) easier to find errors, many text tools
  - Old UNIX tradition
- Size often disadvantageous (verbose)

XML

XML is a widely accepted solution to previous problem

Basically

- Use structured data (markup) in text format (Unicode)
- Unicode encodings UTF-8 and UTF-16
- White space (\r, \n, \t and ' ') default preserved
- Normally *.xml file suffix

Note: XML has no look, it's raw (structured) data

XML Specification

XML is an open standard from W3C

Latest version: 1.0 (Fifth Edition) November 2008
(there's also XML 1.1 we don't use)
http://www.w3.org/TR/REC-xml/

Inspect Introduction of standard, and get an overview of this kind of documents!

XML Documents

"Definition: A data object is an **XML document** if it is **well-formed**, as defined in this specification. In addition, the XML document is valid if it meets certain further constraints."

```
document    ::=    prolog element Misc*
```

I.e. a document is composed of a prolog, then elements (tags) and possible some other things (Misc) and it must be well formed and possible valid

Well formedness

Well-formed

- A tree structure with a single root. Tags <...> are used to form the tree (**no fixed tag set**, extensible)
- Start and end tags must match; <a> ...
- Nesting must match; <a>...
- ... and some more
- Possible to check in NetBeans

If not well formed it's **not** an XML document. Software should reject it (**not** try to fix)

The XML declaration

The XML declaration (first in prolog)

```
XMLDecl::=

    '<?xml' VersionInfo EncodingDecl? SDDecl? S?'?>'
```

First in file, **really** absolutely first (no whitespace!)

Elements

Elements are the empty element or the matching tag-pairs

```
element ::=   EmptyElemTag | STag content ETag
```

Empty element example: <someEmptyElem/> (note slash)

Element example:

```
<movie>                  (STag)
  <title> Plan 9 from outer space</title> (content)
</movie>                 (ETag)
```

- Nested elements
- Element names (tags) are **case sensitive,** XML is case sensitive
- Use camel case: thisIsALongAndSillyTag

Attributes

Elements may have named attributes

```
<movie id="829" ...>
  <title> Plan 9 from outer space</title>
</movie>
```

- Attribute values always as strings "829" or '829'
- Unique attribute names in element
- Space separates attributes

Other content

```
content    ::=    CharData?

((element | Reference | CDSect
| PI | Comment) CharData?)*
```

Comment: <!-- --> (**not** nested)

Reference (entity references, problematic chars)

- Start with "&" end with ";"
- Examples: < (<), " ("), € (Euro symbol)

CDSect (escape markup): <![CDATA[....]] >

XML from Programmers View

An XML document in an in-memory tree,

- This is referred to as the XML DOM-tree (document object model)

There DOM also is a specification of a programmatic interface to handle trees (language, platform neutral)
http://www.w3.org/DOM/

Many versions (levels) of DOM
http://www.w3.org/DOM/DOMTR

XML Namespace

XML is extensible so anyone can create tags

Problems with recognition and collisions of tags

- Different "tags sets" (multiple markup vocabularies) to be handled by applications (possible in same document or merge of documents)

Example <table>...</table>

- Solution: qualified element names (add a prefix to name)

```
<thisIsALongUniqueString:table>...
                          </thisIsALongUniqueString:table>
<thisIsAnotherUniqueString:table>...
                          </thisIsAnotherUniqueString:table>
```

XML Namespace Declaration

Tedious to write long strings, use abbreviations

Declaration of namespace normally inside root element (xmlns = xml-namespace)

```
<c:catalog xmlns:c="http://www.chalmers.se/hajo/cdreg">
    <c:cd id="1">
        ...
    </c:cd>
</c:catalog>
```

c: is shorthand for the string (URL in this case)
URL has no meaning, it's just a hopefully unique string

XML Namespace Specification

XML namespace is another open standard from W3C
http://www.w3.org/TR/REC-xml-names/

XML Schemas

In Java (class based OO)

- An object is an instance of a class (type)
- Type system can check a lot

In XML

- A document is an instance of an schema
- The schema defines the constraints (~the type)
- Possible to verify (~ typecheck) documents

Recap: "Definition: A data object is an XML document if it is well-formed, as defined in this specification. In addition, the XML document is valid if it meets certain further constraints."

Valid XML document

Valid document = A well formed document adhering to a schema

- Need schema to validate document

Possible to validate in NetBeans (validate our document using our schema)

- Problematic but possible to validate the schema (it uses other schemas

What to constraint

A schema can constraint

Types

- Simple or complex

Elements (tags) and attributes and their types

Other constraints

- Order of elements
- Ranges for values
- ...much more...

XML Schema (no ending s)

"XML Schema" is a language to express schema's

- There are others; DTD (deprecated), Relax NG, Schematron, ...

XML Schema is written as an XML document!

Schema files uses .xsd suffix (XML Schema Definition)

Schema uses elements from the xsd namespace

```
<xsd:simpleType .... />
```

XML Schema Specification

Yet another specification from W3C

Known to be very complex (got a lot of criticism). Divided into

Part 1 Structures
http://www.w3.org/TR/xmlschema-1/

Part 2 Datatypes
http://www.w3.org/TR/xmlschema-2/

Also dependencies on other specifications: XML Namespaces, XPath,...

XML Schema Simple Types

Predefined: boolean, string, decimal, double, float, anyURI, QName, hexBinary, base64Binary, duration, date, time, dateTime, gYear, gYearMonth, gMonth, gMonthDay, gDay, and NOTATION

Create simple type String20 from predefined simple type xsd:string

```
<xsd:simpleType name="String20">
    <xsd:restriction base="xsd:string">
        <xsd:maxLength value="20" />
    </xsd:restriction>
</xsd:simpleType>
```

XML Schema Complex Types

Creating a complex type "Country" (note leading uppercase)

- Using our **String20** type

```
<xsd:complexType name="Country">
    <xsd:sequence>
       <xsd:element name="name" type="String20"/>
       <xsd:element name="population"type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:int" use="required" />
</xsd:complexType>
```

Simple types can't have attributes

Defining an Element

Define element country of type Country (in a schema) using our type County

```
<xsd:element name="country" type="Country"/>
```

In document

```
<country> ... </country>
```

Schema Constrained Document

In XML instance (document) specify the schema to use

```
 <root xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="...URI to schema..." >
    :
  </root>
```

- xsi prefix elements from XML Schema instance namespace
- xsi:schemaLocation, where to find...

XML Schema and Namespaces

The elements declared in the schema could (should!) belong to a namespace, the "targetNamespace"

Schema can force documents to use qualified elements and attributes

- elementFormDefault="qualified" (or "unqualified")
- attributeFormDefault="unqualified" (or "qualified")
- Default values for both: "unqualified"

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://www.chalmers.se/hajo/schema/country"
           targetNamespace="http://www.chalmers.
se/hajo/schema/country"
           elementFormDefault="qualified">
```

XML Schema and Name Spaces Examples

Specification section 3
http://web4.w3.org/TR/xmlschema-0/#NS

Modular XML Schema's

Possible to modularize schema's (reuse types)

- Use type Country to define type CountryList

```
<xsd:complexType name="CountryList">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <!-- Reuse country element from other schema -->
      <xsd:element ref="c:country"/>
   </xsd:sequence>
</xsd:complexType>
```

Advanced XML Schema Design

Visibility

- Single global root element or all elements global
- Types local (inside element) or global

Will affect

- Understandability
- Reuse
- Coupling

If you're serious about this you have to study and think it over

Extensible Stylesheet Language Transformation, XSLT

Language for transforming XML documents

- XML is raw data often used as an intermediary format (from database to XML to...)

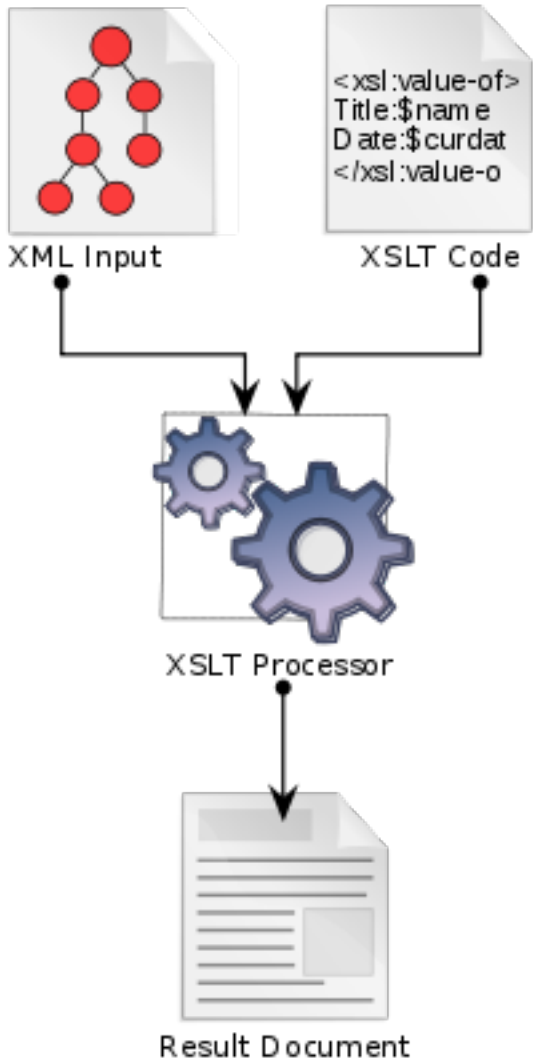
Possible need to

- Change format (i.e. build another tree)
- Display it nicely (XHTML, PDF, Spreadsheet,...)

XSLT is a declarative language written in XML

- XSLT if of course another W3C specification http://www.w3.org/TR/xslt.html

Participants in Transformation



This can be done inside NetBeans!

XPath

XPath is a small expression language describing which nodes in XML-DOM-tree to select

- Expressions are written as strings
- Similar to a file path (simplified)

XSLT uses a subset of XPath (embedded) to select nodes to be transformed

- W3C specification http://www.w3.org/TR/xpath20/

Hava a quick look at ZVON.org
http://zvon.org/xxl/XPathTutorial/Output/example1.html

Basic XSLT Program Structure

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

    <xsl:template match = "XPathExpression">
      <!-- Transform selected nodes
           (possible call other template)-->
    </xsl:tempate>

    <xsl:template match = "otherXPathExpression">
        <!-- Transform selected nodes
           (possible call other template)-->
    </xsl:tempate>

    ...

</xsl:stylesheet>
```

XSLT Examples

Check ZVON again, how to transform to HTML
http://www.zvon.org/xxl/XSLTutorial/Output/contents.html, page 5, 13, 26

Programmatic Handling of XML

Many (most) languages have API's for XML handling

XML as documents

- SAX, Handled as a stream of chars. Only small parts of document in memory. Can't manipulate document
- DOM, Converting between text and internal (in memory) representation. Complete document in memory!

XML from/to classes/objects.

- Converting between documents and classes/objects in OO languages. Classes as generated files, objects in memory

Java API's for XML

JAXP (javax.xml.*, org.w3c.dom.* org.xml.sax.*)

- SAX, DOM, XPath, XSLT and more ...

JAXB (javax.xml.bind.*)

- Map between XML Document/Schema and object/class
- Used in background with Web Services, more to come

Non standard

- JDOM, XStream, ...