XML – Extensible Markup Language

Generic format for structured representation of data.
No predefined tags, but a syntax similar to HTML. Applications:

- Web services, business transactions
- XHTML – HTML on XML syntax
- The graphics format SVG
- Configuration files
- Much more . . .

XML – Strengths

- Open standard from W3C
- Simple text format, easy to parse
- Supported by numerous vendors and platforms
- Excellent for transactions between different systems
- Structure allows for search
- Facilitates separation between content and presentation

XML – Example

```
<?xml version="1.0"?>
<pricelist>
   <item>
      <name>Pears</name>
      <price>12.90</price>
   </item>
   <item>
      <name>Apples</name>
      <price>19.90</price>
   </item>
</pricelist>
```

XML – Form

- The XML declaration first, perhaps stating the file encoding.
  For example one of
  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <?xml version="1.0" encoding="ISO-8859-1"?>
  ```
- More declarations may follow.
- Thereafter exactly one XML element on the outermost level. (Pricelist in the example.)
- End tags required. (Compare with `<p>` in HTML.)
  Special case: An empty element may be abbreviated:
  `<a></a>` becomes `<a/>`. (`<a />` also allowed.)
- Correct nesting required. `<a><bbb></a></bbb>` never allowed.
- Attribute values must be between quote marks. Example (SVG):
  `<circle cx="10" cy="10" r="5" />`.
- As in HTML, ”entities” are used for some characters. Example: `&lt;` for `<` (Starts a tag otherwise).
- A *well-formed* document – follows the syntactic rules.

XML – Specifying valid content

Different applications expect different content in their XML files. Several techniques to specify valid content:

- DTD (document type definition). W3C's first standard.
- XML schemas. W3C's follow-up standard with data types and name spaces. Rich but complicated.
- Several private initiatives, including well-supported Relax NG.
- An instance document is valid if it satisfies a specification.

XML – Document Type Definition

DTD for the pricelist example

```
<!ELEMENT pricelist (item*)>
<!ELEMENT item (name, price)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

- A pricelist element contains any number of item elements.
- An item element contains one name and one price element.
- The name and price elements consist of *parsed character data*.

Reference to external DTD in instance document:

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

XML – Schemas

XML schemas offer more flexibility than DTDs.
Data types are supported, with several built-in types such as

- String types
- Numeric types
- Types for date and time

Minimum and maximum values may be specified, sets may be enumerated, etc. Unlike DTDs, schemas are themselves defined in XML.

XML – Name spaces

You may need to combine parts from different schemas.

Together with schemas, name spaces were introduced to avoid name conflicts.

A name space is identified with a URL, and used with an arbitrary prefix.

Note! The URL only serves as a name. There is no requirement on content.

XML – Example, name spaces

```
<ica:pricelist xmlns:ica="http://www.ica.se/">
  ...
</ica:pricelist>
```

Here, xmlns stands for *XML name space*.

Defining a default namespace (no prefix):

```
<pricelist xmlns="http://www.ica.se/">
  ...
</pricelist>
```

XML – Schema for the pricelist element (1/3)

The first part of the schema:

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ica="http://www.ica.se/"
        targetNamespace="http://www.ica.se/"
        elementFormDefault="unqualified">
...
```

XML – Schema for the pricelist element (2/3)

```
...
<element name="pricelist">
   <complexType>
      <sequence>
         <element name="item" type="ica:item"
             minOccurs="0" maxOccurs="unbounded">
         </element>
      </sequence>
   </complexType>
</element>
...
```

XML – Schema for the pricelist element (3/3)

```
    ...
    <complexType name="item">
       <sequence>
          <element name="name" type="string" />
          <element name="price" type="string" />
       </sequence>
    </complexType>
</schema>
```

XML – Comments to the schema

With `xmlns="http://www.w3.org/2001/XMLSchema"` we choose as default name space W3C's schema for schema definition. From there we use the elements `schema`, `element`, `complexType` and `sequence`, and the type `string`.

With `targetNamespace="http://www.ica.se/"` we define the name space of the new `pricelist` element, as well as the type `item`. To access this type ourselves, we also had to define the `ica` prefix.

Regarding `elementFormDefault="unqualified"`, see the next slide, and `http://www.xfront.com/HideVersusExpose.pdf`.

XML – Using the schema

Refer like this in the instance document:

```
<?xml version="1.0"?>
<ica:pricelist xmlns:ica="http://www.ica.se/">
   <item>
      <name>Pears</name>
      <price>12.90</price>
   </item>
</ica:pricelist>
```

Note. Only `pricelist` is name space qualified.

With `elementFormDefault="qualified"` all elements would have needed qualification.

XML – Best Practices

A schema for an organization should perhaps

- work smoothly with other schemas
- allow updating without making old instance document invalid
- allow instance documents to contain extra information

This is not easy to attain.

See advice at
`http://www.xfront.com/BestPracticesHomepage.html`

XML – Relax NG

- A simpler schema definition language than that from W3C.
- Has become an ISO standard (ISO/IEC 19757-2) in sept 2009.
- Two syntaxes: Compact Syntax and an XML syntax.
- See links at the end of
  `http://www.xmlhack.com/read.php?item=2061`

XML – Schema with Relax NG Compact Syntax

```
namespace ica = "http://www.ica.se/"
element ica:pricelist {
   element item {
      element name {text},
      element price {text}
   }*
}
```

The compact form may be translated to the XML form with the java program trang.
See `http://www.abbeyworkshop.com/howto/xml/xml_relax_overview/`

XML – Schema with Relax NG XML Syntax

```
<?xml version="1.0"?>
<element name="ica:pricelist" xmlns:ica="http://www.ica.se/"
         xmlns="http://relaxng.org/ns/structure/1.0">
   <zeroOrMore>
      <element name="item">
         <element name="name">
            <text />
         </element>
         <element name="price">
            <text />
         </element>
      </element>
   </zeroOrMore>
</element>
```

XML – Validation

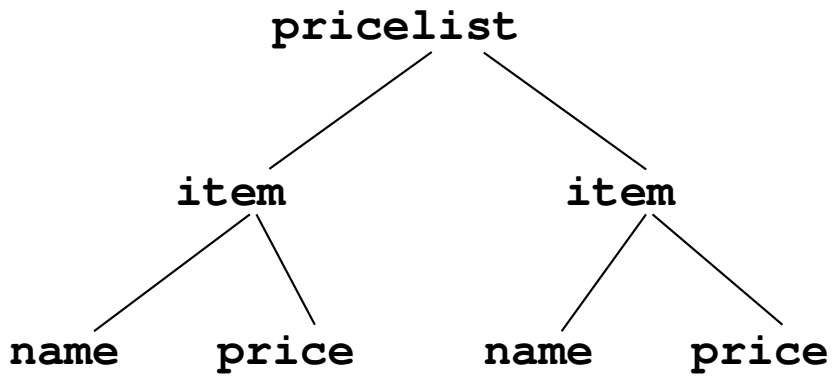
On Unix/Linux `xmllint --noout` file
check for validity, only show errors

`--dtdvalid`	validate against external DTD
`--schema`	validate against W3C-schema
`--relaxng`	validate against Relax NG schema

Web pages such as
`http://tools.decisionsoft.com/schemaValidate.html`

XML – The parse tree

```
<pricelist>
    <item>
        <name>Pear</name>
        <price>12.90</price>
    </item>
    <item>
        <name>Apple</name>
        <price>19.90</price>
    </item>
<pricelist>
```



XSL – Extensible Stylesheet Language

For presentation of XML documents.

Compare with HTML, which conveys presentational structure by itself. Additional style information may be put in a stylesheet.

XML says nothing about presentation. So XSL has three different components:

- XSLT (XSL Transformation) – selects elements in the XML file. Can sort, perform tests, etc.
- XPath – syntax for positioning in the XML tree. Similar to path notation in a file system.
- XSL-FO (XSL Formatting Objects) – Page formatting.

XSL – Example on XSLT and XPath

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:ica="http://www.ica.se/">
   <xsl:template match="/">
      <html>
         <body>
            <table border="1" cellpadding="5" cellspacing="0">
               <tr><th>Item</th><th>Price</th></tr>
               <xsl:for-each select="ica:pricelist/item">
                  <tr><td><xsl:value-of select="name"/></td>
                      <td><xsl:value-of select="price"/></td>
                  </tr>
               </xsl:for-each>
            </table>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

XSL – Comments on the XSLT example

`<xsl:template match="/">` says that the template should start matching from the root of the XML tree.

For each item in pricelist we then create a row in an HTML table.

The row will contain the name and the price of the item.

Referring to the stylesheet(s) in the XML file

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="pricelist.xsl" ?>


<ica:pricelist xmlns:ica="http://www.ica.se/">
   <item>
      <name>Pears</name>
      <price>12.90</price>
   </item>
   <item>
      <name>Apples</name>
      <price>19.90</price>
   </item>
</ica:pricelist>
```

See the result on

http://www.csc.kth.se/utbildning/kth/kurser/DD1335/gruint10/test/pricelist.xml

XSL – Tip

Put the files in your `public_html` directory and view them in a web browser the normal way (`http://...`).

The browser depends on the MIME-type that the server sends in the HTTP heading.

With “Open File” this information is not obtained.

DOM – Document Object Model

- W3C object oriented APIs for XML documents.
- Access and change a document via the DOM parse tree.
- Methods such as
  - `documentElement` – returns the root node
  - `childNodes` – returns all children of a node
  - `attributes` – returns all attributes of a node
  - `nodeType`, `nodeValue`, etc.
  - `removeChild`, `appendChild`, etc.

XML in Java

JAXP – Java API for XML Processing: a common interface to DOM, SAX, and XSLT.
SAX:

- ”Simple” API for XML
- Processes an XML file while reading through it
- Fast, memory efficient
- More complicated than DOM

Many other APIs:

- JDOM, DOM4J – other DOM implementations
- JAXB – converts XML into classes, and vice versa
- JAXM, JAX-RPC for asynchronous and synchronous messaging

XML DOM in JAXP

```
import javax.xml.parsers.*;
import org.w3c.dom.Document;
// Reads an XML file into a DOM structure.
// Usage: java DomExample filename
public class DomExample {
   public static void main(String argv[])
   throws Exception {
      DocumentBuilderFactory factory =
         DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      Document document = builder.parse(argv[0]);
      // Explore the document here.
   }
}
```

Web services

“Applications available over a network via standard protocols.”

Client-server, as well as peer-to-peer.

Typical traditional web service: HTML content, down-loadable with the HTTP protocol.

New web services: composed of distributed parts that are linked dynamically to run seamlessly.

XML plays a key role.

Protocols for new web services

- SOAP – encodes and transmits messages between programs on the web.
- WSDL (Web Services Description Language) XML format to describe web services (operations, messages, types etc).
- UDDI (Universal Description Discovery and Integration) Protocol for distributed registries over web services. Company information like name, category and services. Uses WSDL and SOAP. A program should automatically be able to find and use the services it needs.

SOAP

Simple Object Access Protocol (a misnomer, not really object oriented).

Smaller and simpler to implement than earlier distributed protocols, which did not become so wide spread.

Allows programs written in different languages and running on different platforms to communicate.

The SOAP format

```
<?xml version="1.0"?>
<soap:Envelope
      xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
      soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
   <soap:Header>
      ...
   </soap:Header>
   <soap:Body>
      ...
   </soap:Body>
</soap:Envelope>
```

Use of SOAP

The Body element contains the message in XML. Two common cases are ”document-style” for arbitrary documents, and ”RPC-Style” for function calls. The Body element may also contain a Fault element to describe that a fault has occurred.

The Header element is optional, and intended for instructions to intermediaries on the way to the message destination. Intermediaries may add information, verify payment, etc.

SOAP is normally sent over HTTP, but other transport protocols (even e-mail) can be used.

XML – Summary

We have looked at

- the XML syntax
- Three ways to specify allowed elements: DTDs, XML schemas and Relax NG
- Name Spaces
- XSL for presentation
- XML support in Java
- SOAP, WSDL and UDDI for flexible web services