Introduction to XML

Darren Beard

CICS Developer

darren_beard@uk.ibm.com

4756A

impact·venture*

This presentation contains an introduction to XML, SOAP, WSDL and related technologies.

What is XML?

- Extensible Markup Language
  - A W3C endorsed standard for document markup.
  - Defines a generic syntax used to mark up data.
  - Provides a standard format for computer documents and messages.

XML allows documents to be marked up in human readable form and in a standard manner such that computers can be made to 'understand' the documents. The format is very flexible and has been used in such diverse areas as web sites, vector graphics and voice mail systems.

XML - Extensible Markup Language

The Evolution of XML

- XML is a descendant of SGML, which was invented at IBM in the 1970's. Adopted as ISO 8879 in 1986.
- SGML's biggest success was HTML, which is an application of SGML.
- SGML very complex, so a 'lite' version was produced in February 1998. This was XML 1.0 and it was an immediate success.
- XML 1.1 is now a recommendation (dated September 2006) with the W3C.

The language which was to become SGML was invented at IBM in the 1970's and developed by several hundred people around the world. It was adopted as ISO 8879 in 1986. SGML was intended to solve many of the problems which XML solves. SGML is powerful and has been used in many areas such as the US government and the aerospace sector.

HTML is the main success story from SGML. HTML is, however, just one SGML application. Since HTML restricts users to a defined set of tags designed to describe web pages, HTML is not suitable for many other applications.

SGML was too complex for people to implement properly. Piecemeal implementations of it, some mutually exclusive, followed. A 'lite' version was needed which would be less complex, easier to implement and yet still offer a high level of function. The result was XML 1.0 which was produced in 1998. XML was immediately adopted and used in a wide range of applications.

XML 1.1 is now being actively developed.

http:/www.w3.org/XML/	This is a useful starting point for a lot of XML queries.
http://www.w3.org/TR/xml11/	This is a reference to the XML 1.1 recommendation document of the W3C.

SGML - Standard Generalized Markup Language
HTML - Hypertext Markup Language
XML - Extensible Markup Language

What XML is and is not

- XML is
  - a meta-markup language, therefore, it is flexible.
  - well defined by a strict grammar, allowing parsers to be written.
- XML is not
  - a programming language. XML cannot be executed.
  - a transport protocol. It will not send data over a network.
  - a database. It will not replace DB2 or Oracle.

XML is a meta-language. This means that it does not have a fixed set of tags and elements which everyone must use. XML developers can define elements which they need for a particular job. Physicists could use tags which describe atoms, forces, coordinates etc. Musicians may have tags to define note durations, pitch, key etc. Extensible means that the language can be extended and adapted to meet many needs.

Although the language is extensible, it has a grammar which is quite strict in other ways. The grammar regulates the placement of tags, which element names are legal etc. This allows XML parsers to be written that can read and understand XML.

XML is not a programming language. It is not executable code. An XML document simply *is*. It does not *do* anything.

XML will not send data across a network. Data which is sent via HTTP, FTP etc might well be in XML though.

XML is not a database. It is possible to store XML documents within a database for future reference, but the database itself cannot be XML.

IMPACT IBM

Some XML Constituents

- Documents are composed of named elements.
- Elements are
  - Delimited by tags eg. <name>content</name>
  - Can contain character data or other elements or both
  - Can be empty eg. <noContent/>
- Elements can have attributes
  - An attribute is a name-value pair attached to the element's start tag eg. <temperature **units="Celcius"**>18</temperature>
  - An attribute's value is simply a text string.

9

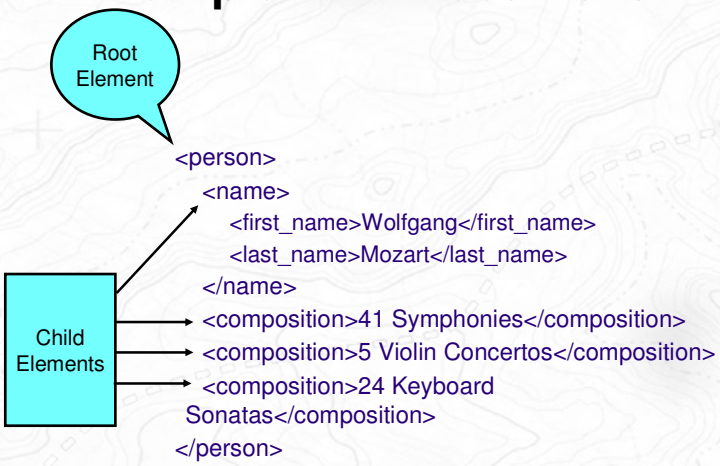
IMPACT IBM

XML documents are composed of elements. Elements can contain character data, other elements or both. If they contain both, the element is described as containing mixed content.

Element attributes are useful to help to make an XML document more readable.

10

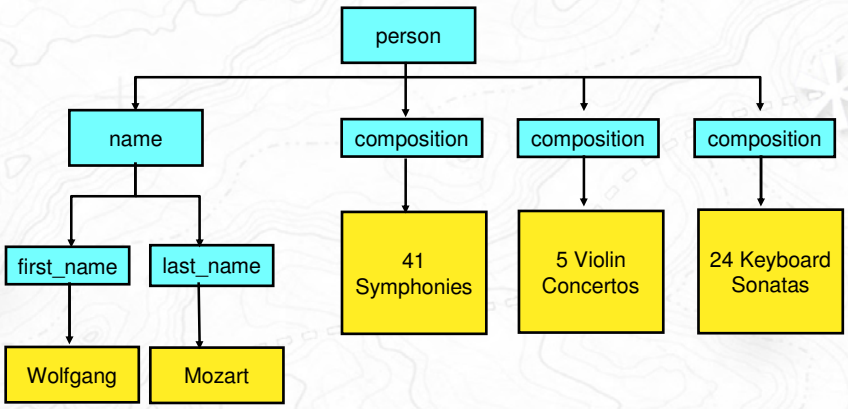
Example XML Document



The XML document contains one person element. This element is the first in the document and contains all the others. It is commonly referred to as the root element. It is also sometimes called the document element.

Elements within an element are called child elements. The person element in the foil has 4 child elements, one name element and three composition elements. The name element itself has two child elements, first_name and last_name. The name element and the composition elements are sometimes called siblings of one another.

XML tree



Elements and data here are shown as a tree structure with different colour boxes for elements (such as person, name etc) and character data (such as Wolfgang and Mozart).

The XML Declaration


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<person>
   <name>
      <first_name>Wolfgang</first_name>
      <last_name>Mozart</last_name>
   </name>
   <composition>41 Symphonies</composition>
   <composition>5 Violin Concertos</composition>
   <composition>24 Keyboard Sonatas</composition>
</person>
```


XML documents should, but do not have to, start with an XML declaration. It looks like a processing instruction, with <? and ?> tag, but it is technically not a processing instruction.

Version specifies that this document is written in XML version 1.0. Currently, this is the only version which will be seen, but this will change in the future as other versions of XML are developed. Version 1.1 is currently under consideration by the W3C.

Encoding specifies the character set for the document. The default is UTF-8 variable length encoding. If the encoding attribute is omitted, an XML parser might guess the encoding by checking the first few characters of the document. If you were producing XML on a mainframe system, you would not be using UTF-8.

Standalone is optional. The default is 'no'. If the value is 'no', then this means that the application may need to refer to a DTD (another file from the one it is reading now). Documents that do not have DTDs can use 'yes' for standalone. Documents which do have DTDs can still have 'yes' for this value as long as the DTD does not change the content of the document.

DTD - Document Type Definition. This is explained on a later foil.
UTF - Unicode Transformation Format.

Well-formed XML

- Every legal XML document must be well-formed. This means that it must obey a number of rules such as:-
  - Every start tag must have a matching end tag.
  - Elements must not overlap.
  - There must be one and only one root element.
  - Attribute values must be quoted.
  - An element must not have two attributes with the same name.
- A well-formedness error is fatal.

The list of conditions on the foil is not exhaustive, but gives some idea as to what is meant by well-formedness. An XML parser must report errors of this type. A parser is not allowed to attempt to fix a well-formedness error. It will not fill in missing quotes or insert an omitted end tag.

Document Type Definitions

- Document Type Definitions
  - written in formal syntax.
  - explains precisely which elements and entities are allowed in the document.
  - defines what the elements' contents and attributes are.
- A document which satisfies the required DTD is said to be valid.
- A validity error is not necessarily fatal.

A DTD allows constraints to be placed on the form which an XML document takes, but there can still be a lot of flexibility within those limits. A DTD never says anything about the length, structure, meaning or other aspects of an element.

A parser reading an XML document may or may not check for validity. A parser which does compare documents to their DTDs is called a validating parser.

Well-formedness is required of all XML documents. Validity is not. Programs may be able to use documents which fail validity checks due to minor errors. It is up to the program to decide whether this is possible and acceptable or not.

Document Type Declaration


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE person SYSTEM"http://people.org/xml/dtds/person.dtd">
<person>
   <name>
      <first_name>Wolfgang</first_name>
      <last_name>Mozart</last_name>
   </name>
   <composition>41 Symphonies</composition>
   <composition>5 Violin Concertos</composition>
   <composition>24 Keyboard Sonatas</composition>
</person>
```


The document type declaration statement specifies the DTD to which this XML must be compared. The declaration says that the document's root element is person and that the DTD for this document can be found at URI http://people.org/xml/dtds/person.dtd.

The specification of where the DTD is to be found does not have to be a URI. It could be specified as a file name if the DTD is 'local', that is, on the machine processing the XML. It is even possible to imbed the DTD within the XML document and refer to it simply by name. This is not of much use in practice, but is often useful for test purposes.

URI - Universal Resource Identifier. URIs are a superset of URLs. In practice, the only URIs in common use today are URLs.

XML Schemas

- A Well-formed document obeys XML syntax.
- A valid document also conforms to a DTD.
  - DTDs cannot enforce element datatypes, sizes or values.
- "Valid" against a DTD is not necessarily good enough!
- XML schemas can overcome the ambiguity of XML
  - replaces DTDs.
  - XML schemas are themselves written in XML syntax.
  - Extensive support for datatypes.

XML schemas are able to enforce rules more strictly than is possible with DTDs alone. These are now at the stage of being a W3C Recommendation. The documents referenced below are dated October 2004 and give much more detail concerning schemas than can be given in an introductory talk.

See W3C recommendations at **http://www.w3.org/TR/xmlschema-0** Primer
http://www.w3.org/TR/xmlschema-1 Structures
http://www.w3.org/TR/xmlschema-2 Datatypes

Consider the following XML

<!ELEMENT composition_date (#PCDATA)>

.....

<composition_date>Green</composition_date>

An XML parser will not know the characteristics of 'Green' and so would accept this as valid XML. The XML schema language allows information about the data to be included with the structural information.

<element name="composition_date" type="date">

If Green is now entered as a date, the XML parser will flag this as an error.

Example Schema

Define the schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                                        xmlns="darren/schema"
                                        targetNamespace="darren/schema"
                                        attributeFormDefault="unqualified"
                                         elementFormDefault="qualified">
```

Define basic components

```
<xsd:element name="first_name"  type="xsd:string"></xsd:element>
<xsd:element name="last_name"   type="xsd:string"></xsd:element>
<xsd:element name="composition" type="xsd:string"></xsd:element>
```

Some definitions omitted

Define "complex" structures

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name" />
      <xsd:element ref="composition" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```


Here is the complete schema for the "person" XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                                        xmlns="darren/schema"
                                        targetNamespace="darren/schema"
                                        attributeFormDefault="unqualified"
                                         elementFormDefault="qualified">

<xsd:element name="first_name"  type="xsd:string"></xsd:element>
 <xsd:element name="last_name"   type="xsd:string"></xsd:element>
 <xsd:element name="composition" type="xsd:string"></xsd:element>

<xsd:element name="name">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element ref="first_name"/>
       <xsd:element ref="last_name"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

 <xsd:element name="person">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element ref="name" />
       <xsd:element ref="composition" minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

IMPACT IBM

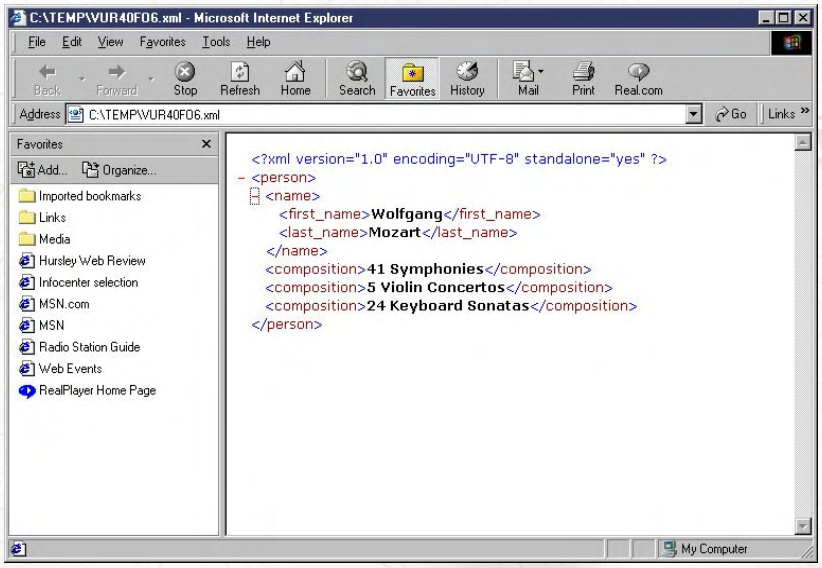
Browsers and XML

- Web browsers can be used to display and check XML.
  - Internet Explorer 4.0 can display XML from a file.
  - Internet Explorer 5 and 5.5 can both parse and display XML.
  - Internet Explorer 6 implements a number of W3 consortium XML specifications regarding XML.
  - Netscape 4.0 and earlier do not support XML.
  - Netscape 4.0.6 and later does use XML internally.
  - Netscape 6.0 and above does fully support display of XML.

IMPACT IBM

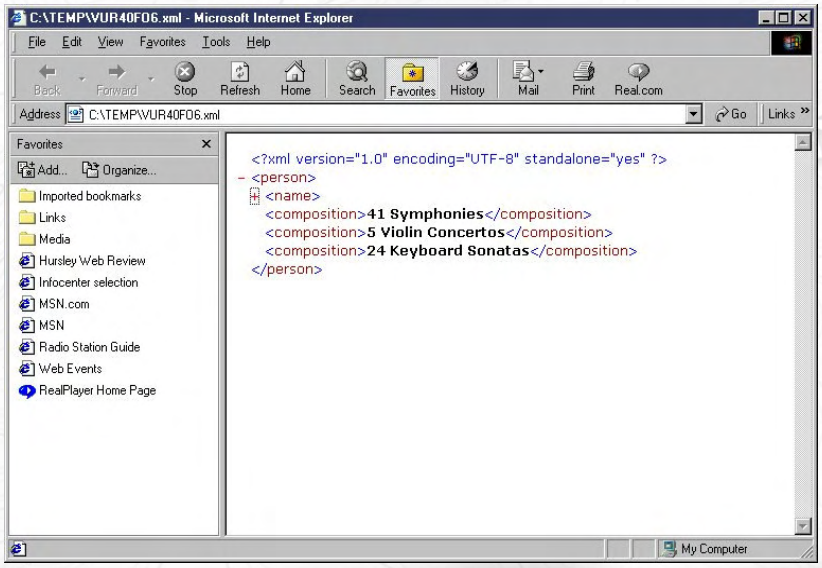
Web browsers are useful for checking and displaying XML documents. The foil lists the various releases of the two most commonly used browsers and details the XML support provided.

Browser Output



A browser is a good way of checking that XML is valid. Just about all browsers can accept XML as input. Giving the Mozart XML as input to IE5.5 gives the display shown in the foil.

Browser Output



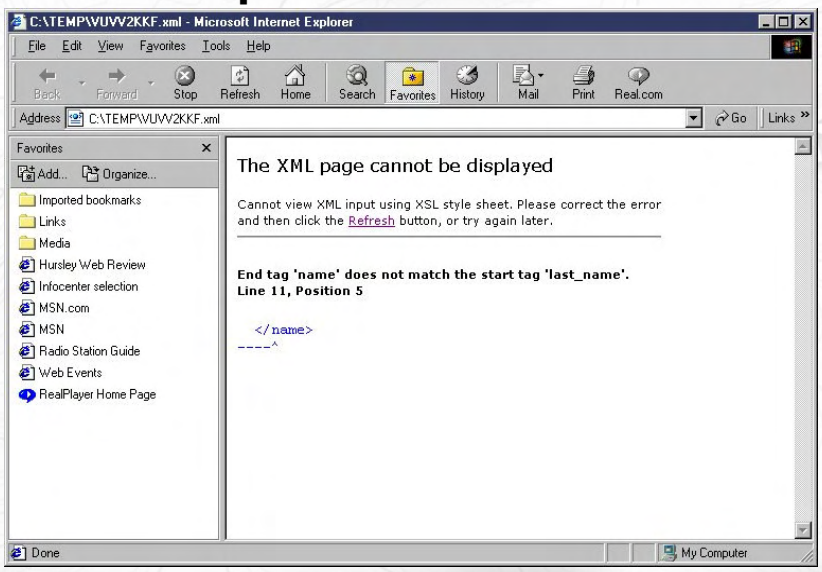
Clicking on the - next to name collapses the entry so that the child elements are not shown. For a complex piece of XML, this can be very useful as an aid to reading or checking.

Some not well-formed XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<person>
   <name>
      <first_name>Wolfgang</first_name>
      <last_name>Mozart<last_name>
   </name>
   <composition>41 Symphonies</composition>
   <composition>5 Violin Concertos</composition>
   <composition>24 Keyboard Sonatas</composition>
</person>
```


Here is some XML which is not well-formed. The / has been removed before the last_name end of element tag.

Browser Output for not well-formed XML



The browser will not display XML which is not well formed. This XML is not well-formed, so it is a severe error. The error message points to roughly the correct place where the error exists.

Namespaces

- Namespaces have two uses in XML
  - To distinguish between elements and attributes from different XML applications which share the same name.
  - To group together all related elements and attributes from a single XML application so that software can recognize them.

Namespaces are needed for two reasons in XML. The first reason on the foil is perhaps the easier one to understand. The second reason is the more important in practice.

A good starting point for information on namespaces is :

http://www.w3.org/TR/REC-xml-names/

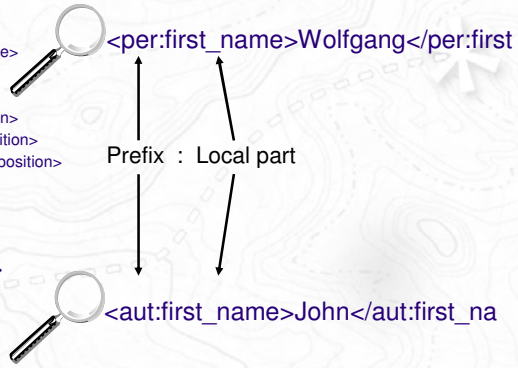
Another XML Document

```
<root>
 <person>
  <name>
   <first_name>Wolfgang</first_name>
   <last_name>Mozart</last_name>
  </name>
  <composition>41 Symphonies</composition>
  <composition>5 Violin Concertos</composition>
  <composition>24 Keyboard Sonatas</composition>
 </person>
 <author>
  <name>
   <first_name>John</first_name>
   <last_name>Smith</last_name>
  </name>
  <booktitle>The life of Mozart</booktitle>
  <booktitle>Mozart: The early years</booktitle>
 </author>
</root>
```


The XML shown in the foil, whilst it is well-formed, would certainly cause problems if it were to be used in an application. We have elements called name, first_name and last_name in two places within the document. This ambiguity of definition is clearly not acceptable. Namespaces can be used to overcome this problem.

XML Document using namespaces

```
<root>
 <person>
  <per:person
xmlns:per="http://www.myperson/syntax">
    <per:name>
     <per:first_name>Wolfgang</per:first_name>
     <per:last_name>Mozart</per:last_name>
    </per:name>
    <composition>41 Symphonies</composition>
    <composition>5 Violin Concertos</composition>
    <composition>24 Keyboard Sonatas</composition>
  </per:person>
 </person>
 <author>
  <aut:author
xmlns:aut="http://www.myauthors/syntax">
    <aut:name>
     <aut:first_name>John</aut:first_name>
     <aut:last_name>Smith</aut:last_name>
    </aut:name>
    <booktitle>The life of Mozart</booktitle>
    <booktitle>Mozart: The early years</booktitle>
  </aut:author>
 </author>
</root>
```



The XML in the foil used namespaces to define the elements unambiguously. Each prefix is associated with exactly one URI. Names with prefixes associated with different URIs are in different namespaces. Therefore, in the foil, per:name is different from aut:name. Elements and attributes in namespaces have names which contain exactly one colon.

per:name — Everything before the colon is called the prefix. Everything after the colon is called the local part. The whole name including the colon is called the qualified name, the Qname or the raw name. The prefix identifies the namespace to which the element or attribute belongs. The local part identifies the particular element or attribute within the namespace.

qualified name = prefix:local part

Document Object Model (DOM)

- DOM defines an object hierarchy, a tree model.
- Good points
  - Very useful for repeated random access to a document
    - document editing.
    - data retrieval.
  - Platform and language neutral.
- Bad points
  - Entire document must be parsed and stored in memory.
  - Unsuitable for document transformation applications.

DOM models XML data as a tree structure. In order for this to be possible, the entire document has to be in storage before it can be referenced. This clearly has implications for large documents. Also, the whole document needs to be read before any of the document can be used. DOM is excellent for applications which require repeated random access to a document. However, since it requires that the whole document is in storage, this is a potential problem for large documents. Also, some early implementations of DOM added up to 100% of the original document's size.

http://www.w3.org/TR/DOM-Level-3-Core

This is the latest DOM specification W3C recommendation, dated April 2004.

Simple API for XML (SAX)

- An event based API for parsing XML.
  - Good points
    - Document data is available as it is parsed.
    - Does not need large amounts of memory.
  - Bad point
    - No support for modifying or writing XML document data.

SAX2 is now the standard form of SAX. Since 2001, all major parsers which support SAX also support SAX2.

SAX is not a W3C publication. It is a standard from Megginson Technologies. It is event based. This means that the application is notified each time an XML feature is recognized. This enables the application to take action on parts of a document without having to wait for all of it to be loaded. The whole document does not have to be in storage, so this is not subject to the same memory considerations as DOM.

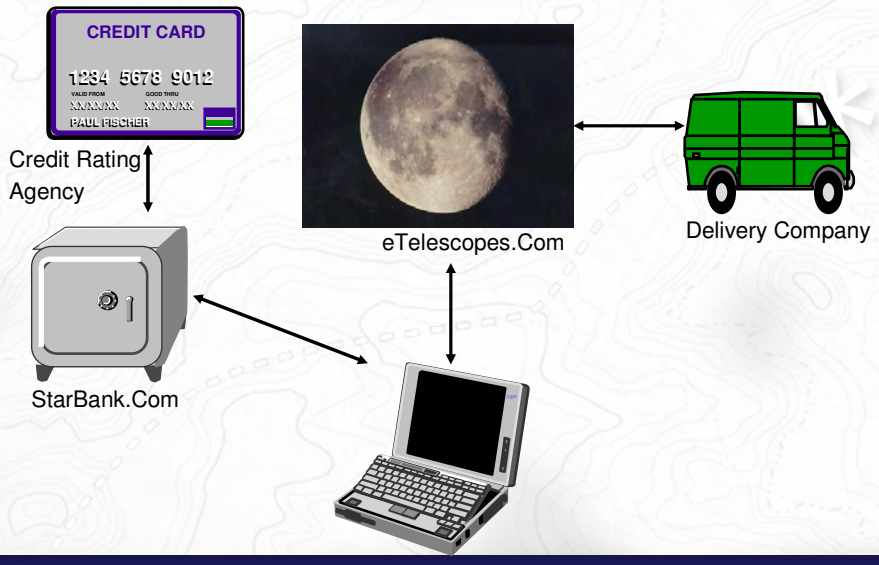
http://www.saxproject.org/ This web site gives information about SAX in general. It is the official SAX web site produced in February 2002.

Business Scenario

- My company (eTelescopes.com) sells astronomical telescopes and accessories. I want to be able to offer finance agreements on my web site as well as product information.

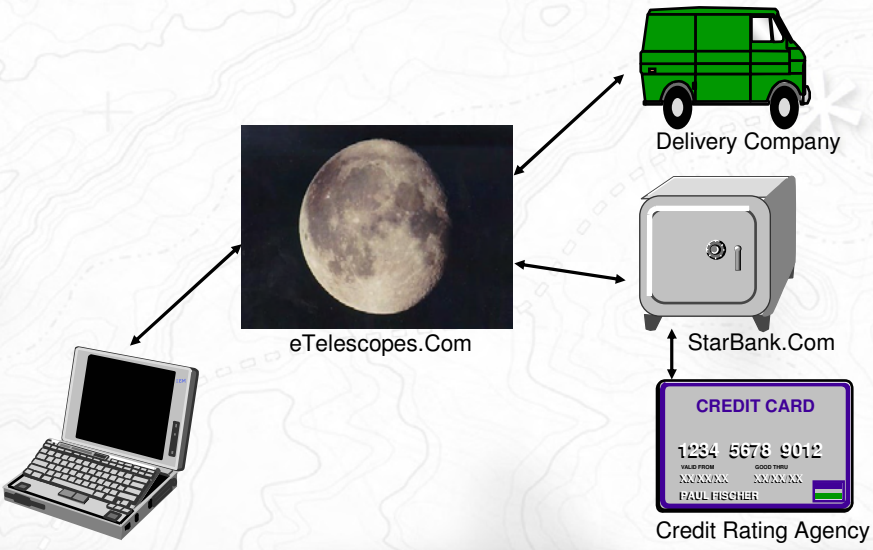
This example will introduce some further useful terms concerning XML. It is clearly only a simple example, but it should at least illustrate some of the basic points in a manner which moves from abstract concepts to a more definite business environment.

Current Customer View



Currently, my customers can only use eTelescopes.com to look up and order telescopes and accessories. If my customers need finance, they need to go to a separate loans company and arrange this. My company does arrange delivery, so it already deals with a delivery company.

Desired Customer View



I would like my customers to be able to organise everything from my web site, including any finance which might be required. This means that my eTelescopes company needs to send and receive messages to and from the StarBank company. A method of doing this could use XML.

How do we do this using XML?

- To use XML, the following items must be addressed
  - Select or write a DTD or schema.
  - Generate XML documents.
  - Transmit XML across a network.
  - Interpret XML documents.
  - Display XML documents.

In order for an application to use XML, a DTD might be needed. Once a DTD is available, the XML to be sent needs to be generated. Once it is generated, it must be sent to a target system or application. The target system needs to be able to interpret the XML coming in and perform whatever processing is required. When the target system returns XML, it needs to be interpreted and displayed in some way.

How to communicate

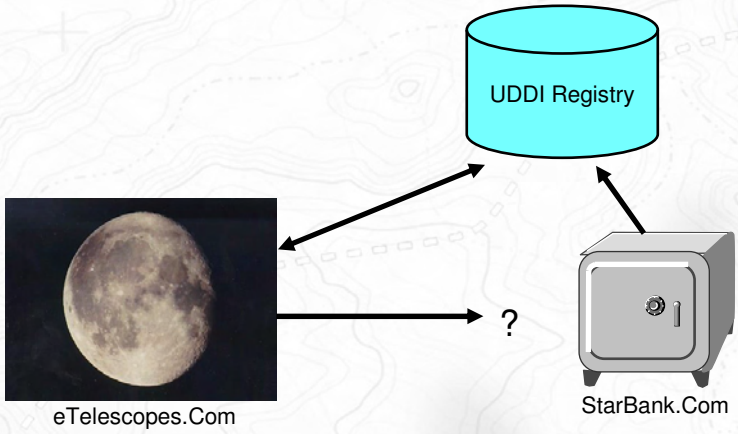
eTelescopes.Com

?

StarBank.Com

eTelescopes.com (my company) needs to communicate with StarBank.Com, but how? What services do they offer, what format must the XML be in, what communications transport do they use?

Universal Description, Discovery and Integration (UDDI)



If StarBank.Com has published an entry into the UDDI, then eTelescopes.Com can find the entry and discover how to communicate.

UDDI

- UDDI is an industry initiative.
- UDDI enables businesses to
  - Describe business and services.
  - Discover other businesses which offer required services.
  - Discover how to integrate web services with business processes.
  - Publish how they want to do business electronically and give notice of application changes.

An entry in the UDDI can specify a businesses' services and how it wants to carry out electronic business.

http://www.uddi.org This web site contains a lot of information about UDDI including the latest specifications.

UDDI results

- The WSDL part of the UDDI registry entry told me that StarBank.Com uses SOAP over HTTP for its electronic business.

The UDDI registry contains technical information concerning the ebusiness requirements of a company. This is specified using Web Services Description Language (WSDL). The sorts of things specified using WSDL are indicated here.

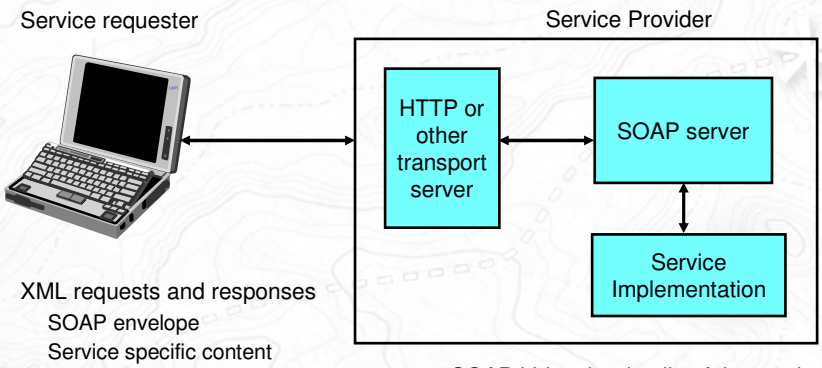
Types	– a container for data type definitions using some type system, such as XSD (XML schema definition).
Message	– an abstract, typed definition of the data being communicated.
Operation	– an abstract description of an action supported by the service.
Port Type	– an abstract set of operations supported by one or more endpoints.
Binding	– a concrete protocol and data format specification for a particular port type.
Port	– a single endpoint defined as a combination of a binding and a network address.
Service	– a collection of related endpoints.

For much more information concerning WSDL, see the web reference below.

http://www.w3.org/TR/wsdl	This is the web site for information on WSDL 1.1
http://www.w3.org/TR/wsdl20	This web site gives information about WSDL 2.0. It is a W3C candidate recommendation dated March 2006.

So now I know that I need to use SOAP over HTTP in order to do business with StarBank.Com.

SOAP Model



XML requests and responses
- SOAP envelope
- Service specific content

SOAP hides the details of the service implementation from the requester.

The requester sends and receives XML using SOAP envelopes and service specific content. The use of a SOAP server by the service provider means that the details of the implementation of the requested service are hidden. This is useful because requesters do not then become dependent upon any particular implementation, thus allowing the provider to upgrade the service in a compatible manner without requiring changes to all the requesters.

Simple Object Access Protocol (SOAP)

- SOAP defines an XML "envelope"
  - Applications define what goes into the envelope.
  - Namespaces are used to distinguish message data from the envelope.
- The main goal of SOAP is to facilitate interoperability.

SOAP defines the use of XML and the transport layer to access services, objects and servers in a platform independent manner. SOAP is a protocol which acts as a glue between heterogeneous software components. If developers can agree on the application level XML and transport to be used, SOAP offers a way of bridging competing technologies in a standard way.

A brief history of SOAP

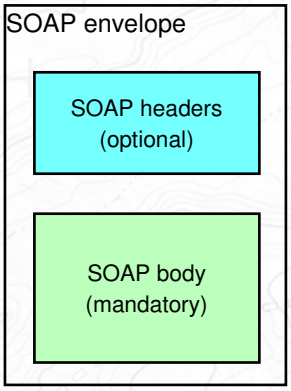
SOAP 1.0
- internet enabled.
- messages can be flowed over HTTP.

SOAP 1.1
- messages can be flowed over any protocol.
- Can use languages other than Java.

SOAP 1.2
- Currently a recommendation under development by the W3C. It extends the SOAP architecture.

http://www.w3.org/TR/soap This web site provides access to the SOAP 1.1 and SOAP 1.2 specification documents.

SOAP fundamentals



The SOAP envelope typically contains headers concerning such things as
- routing information
- date and time stamp
- authentication

The SOAP body contains the data which are understood by the application program.

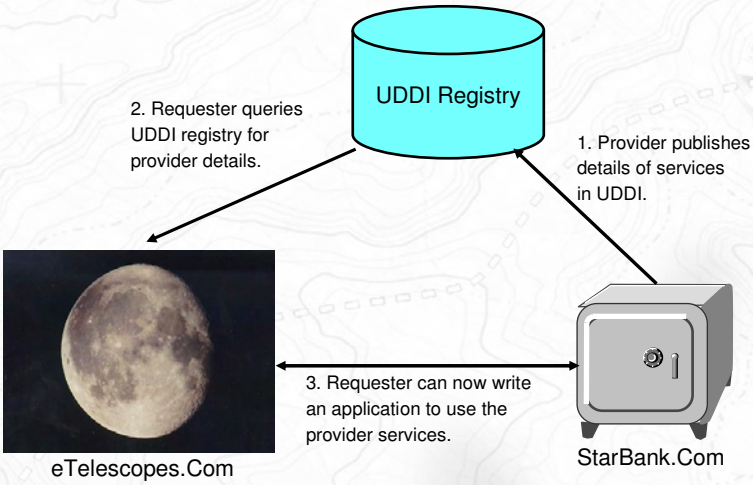
SOAP allows standardization of things like routing information, security protocols etc. The XML within the SOAP body is purely for use by the application program. Use of SOAP allows servers to hide the implementation details of an application from a requester.

Some simple SOAP

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  SOAPENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    <SOAP-ENV:Body>
      <m:GetCurrentInterestRate
        xmlns:m="http://www.starbank.loanrates.com">
          <Symbol>StBk</Symbol>
          <Company>StarBank</Company>
          <Rate units="percent">5.1</Rate>
      </m:GetCurrentInterestRate>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


The foil shows some simple SOAP. The 'envelope' and the 'body' have been coloured differently for ease of differentiation.

How far have we got?



With the use of the UDDI, we are able to query which services are available and how they are to be used. This then enables eTelescopes.Com to communicate with StarBank.Com as required to enhance the service being offered.

This scenario of a service provider publishing details of the service and the service requester querying these details is known as Service Oriented Architecture.

Using XML, what we can do so far

- To use XML, the following items must be addressed
  - Select or write a DTD or schema.
  - Generate XML documents.
  - Transmit XML across a network.
  - Interpret XML documents.
  - Display XML documents.

So far, we have covered most of the points required in the use of XML. We can write or select a DTD or schema based on information in the UDDI. We can generate XML to send to the provider, using SOAP if required. We can then transmit the XML message using HTTP. When the result comes back from the service provider, we can use either a DOM or SAX parser to interpret the XML. What we cannot yet do is to display the results on a web page. XML is not in a form which will display nicely on a web browser. Ideally we would like to transform it in some way to make the web page more user friendly.

Extensible Stylesheet Language Transformations (XSLT)

- XSLT is an XML application that specifies rules by which one XML document is transformed into another.
  - Can change XML into HTML (this is an example of transcoding)
  - Can work with namespaces

Although, as we have seen, it is possible to display XML on browsers, it is usually necessary to transform the XML document into HTML before sending it to a browser. XSLT has been designed to do this and other transformations of XML.

XML can be transformed into XML, for example to invert an XML tree structure. It can also be used to transform XML into other formats, such as HTML or WML. The transformation of XML into other formats is known technically as transcoding.

http://www.w3.org/TR/xslt This URL references a web site giving the W3C recommendation document for XSLT. It is dated November 1999.

http://www.w3.org/TR/xslt20 This is the latest specification document for XSLT 2.0. It is a W3C recommendation dated January 2007.

Implementations of XSLT:

Apache Xalan see **http://xml.apache.org**

LotusXSL

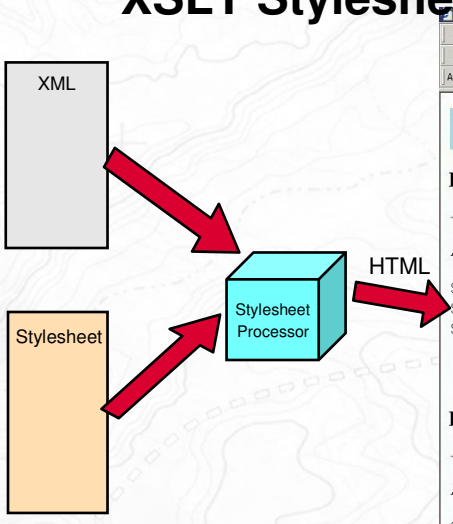
WML - Website Meta Language. WML is a free HTML generation toolkit for Unix.

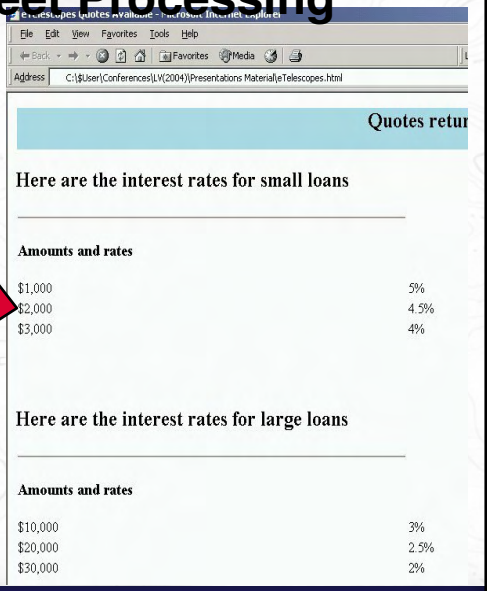
XML Stylesheet Processing Instruction

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"
                 href="http://www/eTelescopes.com/styles/loans.xsl"?>
<loans>
  .....
```


If an XML document is to be served directly to a web browser, then it can have an xml-stylesheet processing instruction in the prolog to tell the browser where to find the stylesheet for the document. If the stylesheet is XSLT, then the type attribute should have the value text/xml.

XSLT Stylesheet Processing





XML as received

```
<?xml version="1.0" encoding="UTF-8"?>
<quotes>
  <description>Quotes returned by Star Bank</description>
  <quotations>
    <quote id="Q1">
      <quoteText>
        Here are the interest rates for small loans
      </quoteText>
      <amount>
        <amountText>$1,000</amountText>
        <amountRate>5%</amountRate>
      </amount>
      <amount>
        <amountText>$2,000</amountText>
        <amountRate>4.5%</amountRate>
      </amount>
      <amount>
        <amountText>$3,000</amountText>
        <amountRate>4%</amountRate>
      </amount>
    </quote>
    <quote id="Q2">
      <quoteText>
        Here are the interest rates for large loans
      </quoteText>
      <amount>
        <amountText>$10,000</amountText>
        <amountRate>3%</amountRate>
      </amount>
      <amount>
        <amountText>$20,000</amountText>
        <amountRate>2.5%</amountRate>
      </amount>
      <amount>
        <amountText>$30,000</amountText>
        <amountRate>2%</amountRate>
      </amount>
    </quote>
  </quotations>
</quotes>
```

XSLT stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
  xmlns:xalan="http://xml.apache.org/xslt">
  <xsl:template match="/">
  <html>
    <head>
      <title>eTelescopes Quotes Available</title>
    </head>
    <body>
      <form action="/ILSC01/quote" method="post">
        <table width="100%"><tbody>
          <tr>
            <td align="center" bgcolor="lightblue">
              <h2>
                <xsl:value-of select="/quotes/description"/>
              </h2>
            </td>
          </tr>
          </tbody>
          </table>
          <xsl:apply-templates select="//quote"/>
      </form>
    </body>
  </html>
  </xsl:template>
  <xsl:template match="quote">
  <h2>
    <xsl:value-of select="quoteText"/>
  </h2>
    <table width="50%">
    <tbody>
    <tr>
      <td>
        <hr/>
        <h3>Amounts and rates</h3>
        <xsl:apply-templates select='amount'/>
      </td>
    </tr>
    </tbody>
    </table>
    <br/>
    <br/>
  </xsl:template>
  <xsl:template match="amount">
    <tr>
      <td align="left">
        <xsl:value-of select="amountText"/>
      </td>
      <td align="left">
        <xsl:value-of select="amountRate"/>
      </td>
    </tr>
  </xsl:template>
</xsl:transform>
```

XSLT Support in Browsers

- Internet Explorer 5 and 5.5 support their own version of XSLT which is not the same as XSLT 1.0.
- Internet Explorer 6 does support XSLT.
- Netscape does not support XSLT at all.

The browser output shown earlier in the presentation in fact showed Internet Explorer 5.5 applying a default stylesheet to an XML document. The little + and - signs to expand and collapse child elements in the displayed XML are a result of the application of an XSLT stylesheet. The indentation and colouring were also the result of the stylesheet processing.

XML Benefits

- Flexible method of representing data.
- Can control the content of data with DTDs and XML Schemas.
- Important step towards true any-any interoperability.
  - Platform independent.
  - Language independent.
- Useful in many standardisation activities.
  - Business to Business structured data interchange.

XML is important because it is a flexible markup language which is quite easy for humans to read and yet with a grammar rigid enough to allow computers to handle it. It is platform and language neutral, so it does not tie an implementation to a particular computer language or operating system.

Because of the platform independence, it should facilitate business to business communication. This should help large organisations to grow and integrate with partners more easily. Small companies should be able to enhance their business opportunities by being able more easily to communicate with larger suppliers etc.

XML Problems

- Most applications can not handle XML.
  - A system which does not handle XML needs some way of transforming XML into something it does understand.
- Amount of data to be transmitted increases.
  - XML tagging is an overhead on messages.
- No common XML dialect across industries.
  - Many industry standards are developing eg.
    - IFX (Interactive Financial eXchange)
    - OBI (Open Buying on the Internet)
    - IOTP (Internet Open Trading Protocol)

Using XML is not a panacea for all ills. It does have some problems, as indicated in the foil. However, these are being addressed.

As the use of XML becomes more widespread, more applications will be written either to handle XML directly or they will be front-ended in such a way that they can accept XML.

It might be possible to overcome some of the problems of increased data in the messages due to XML tagging by compressing the XML prior to transmission.

Many dialects of XML are developing. It might be possible to standardise on something such as SOAP as a standard dialect which could be applicable across industry sectors.

IFX is mainly used by financial institutions currently.
OBI is used by many companies in the areas of finance, media and passport control.

Where to find out more

- Check the web sites referenced in the notes to several of the foils in this presentation.
- http://www-128.ibm.com/developerworks/xml/
  - A resource for customers and developers using XML.
- www.w3.org
  - The World Wide Web Consortium
- Any of the good introductory books on XML.

There are lots of places to find out more about XML. The foil lists some ideas for further reading. This is by no means an exhaustive list.

IBM 2007 IMPACT

IBM

Questions
and
Answers

impact·venture*

IBM 2007 IMPACT

IBM

The end.

Glossary of Abbreviations

DOM	Document Object Model. A model used by XML parsers. It treats XML as a tree structure.
DTD	Document Type Definition. Specifies XML document rules.
HTML	Hypertext Markup Language. Used extensively for web pages.
HTTP	Hypertext Transfer Protocol. A protocol widely used on the web for sending and receiving data.
PCDATA	Parsed Character Data. Used in DTDs, usually as #PCDATA, to define the contents of elements.
SAX	Simple API for XML. A model used by XML parsers. It is event driven.
SGML	Standard Generalised Markup Language. A markup language from which XML was derived.
SOAP	Simple Object Access Protocol. An XML protocol to aid interoperability.
UDDI	Universal Description, Discovery and Integration. A global directory of businesses and their eBusiness setup.
URI	Universal Resource Identifier. A superset of the more common URL.
URL	Universal Resource Locator. The thing which is entered into a web browser to go to a web site.
UTF	Unicode Transformation Format. One of the unicode formats for encoding characters.
WML	Website Meta Language. It is a free HTML generation toolkit for Unix.
WSDL	Web Services Description Language. An XML grammar for describing network services.
XML	Extensible Markup Language. The subject of this presentation.
XSD	Extensible Markup Language Schema Definition. It is used to define XML contents unambiguously.
XSLT	Extensible Stylesheet Language Transformations. An XML application to transform XML into other formats.