XML and Web Services for Astronomers

Roy Williams
California Institute of Technology
roy@caltech.edu

Robert Brunner
University of Illinois
rb@ncsa.uiuc.edu

ASASS 2002, 13 October 2002, Baltimore

- **XML and Structured Data**
- XML Syntax
- VOTable and other formats
- Transformation, Parsing, Binding

What is Markup?

Markup in a document means extra tags to define the meaning of the text.

```
<b>Memorandum</b>
<hr>
From: <i>Antonio Stradivarius</i><br>
To: <i>Domenico Scarlatti</i><br>
Date: 13 April 1723<br>
Message: Io bisogno una appartamento
acoglienti a Cremona …
<hr>
```

This markup is HTML

Rendering

Memorandum

From: *Antonio Stradivarius*
To: *Domenico Scarlatti*
Date: 13 April 1723
Message: Io bisogno una appartamento acoglienti a Cremona …

Structure with XML

```
<Memo>
<From>Antonio Stradivarius</From>
<To>Domenico Scarlatti</To>
<Date>
        <Day>13</Day>
        <Month>4</Month>
        <Year>1723</Year>
</Date>
<Body>
Io bisogno una appartamento
acoglienti a Cremona …
</Body>
</Memo>
```

Separation of **structure** from **presentation**

Rendering

4/13/23

April 13, 1723

17.iv.1723

Processing

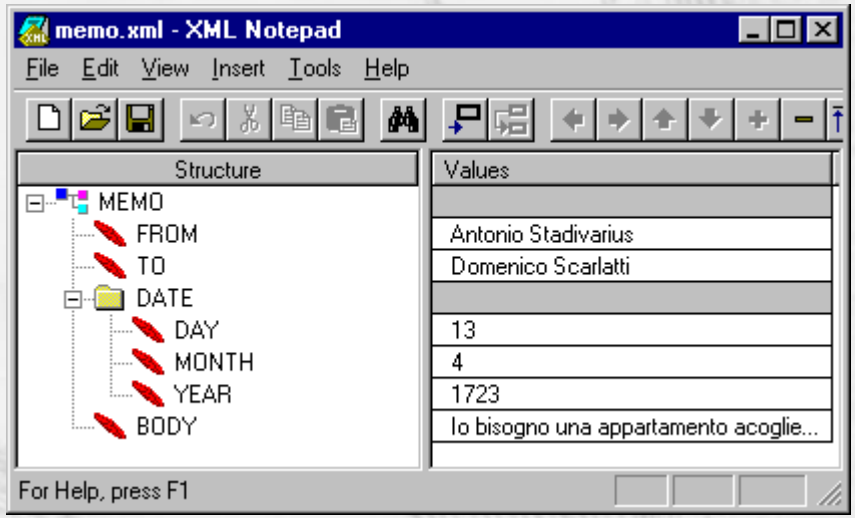
The computer can read the document:
"Find all memos from April 1723"

Why XML?

XML is a standard way to represent structured documents, including metadata and data

- **Platform neutral / Open**
- **Vendor supported / Vendor neutral**
- **Proven -- decades with SGML**
- **Extensible**
- **Syntax checking -- Explicit Schema**
- **Industry convergence**
- **Web friendly**

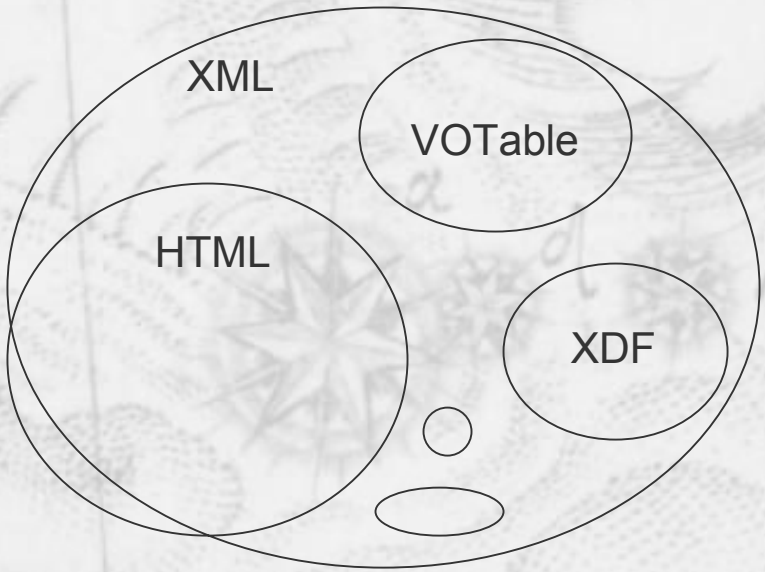
Why XML?



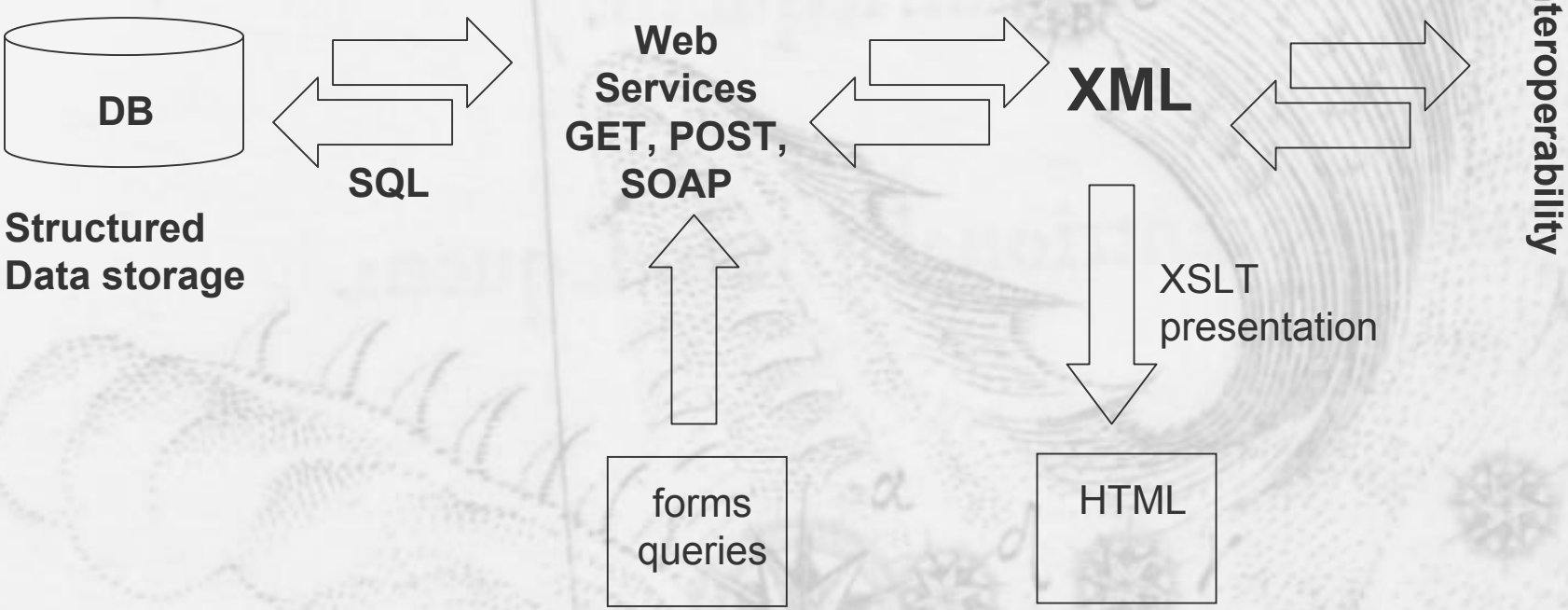
- Documents and data
- Human readable, editable, mailable
- Can encode many data models
- Can encode program too
- Many tools
  - Parsers in Java, C, C++, Perl, Python, ...
  - Browsers and editors
  - XML databases
  - Style sheets, formatting, transformation

What is Markup?

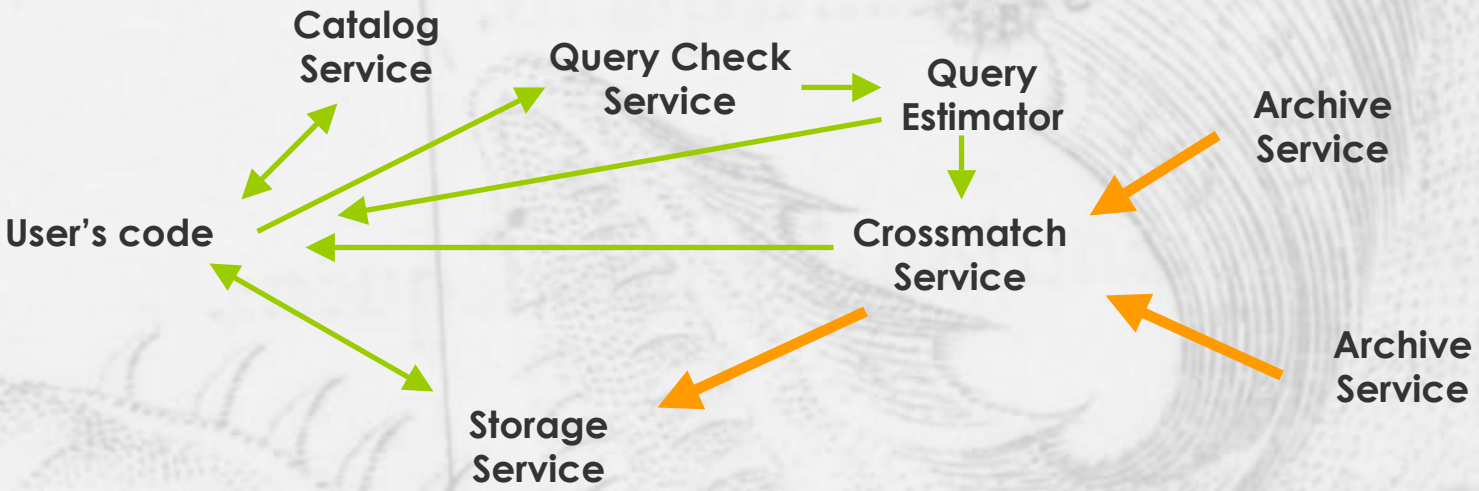
- Markup is everywhere
  - Latex, Postscript, FITS, ….
- From here we consider only XML dialects:



XML Usage Model



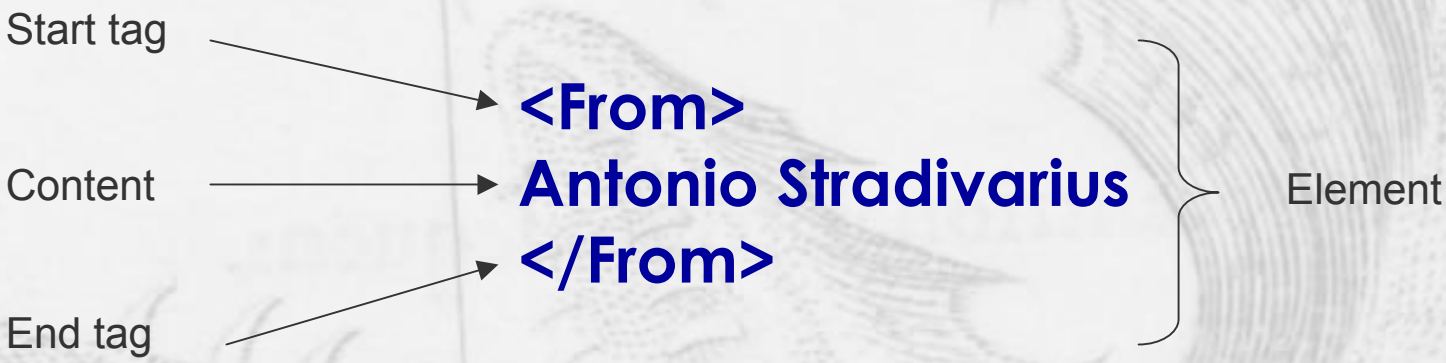
Service Workflow



SOAP envelopes of
XML: **VOTable and other VO dialects**
AND
broadband binary

- XML and Structured Data
- **XML Syntax**
- VOTable and other formats
- Transformation, Parsing, Binding

XML Syntax



White space is part of the content
-- Many applications ignore it

Element names are case-sensitive

<From> is not **<from>**

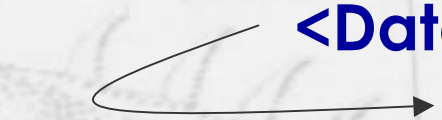
XML Syntax

Empty element:

<From/>

is equivalent to **<From><From/>**

*Note that the HTML constructions
 and <hr>*
*Are not proper: should be
 and <hr/>*

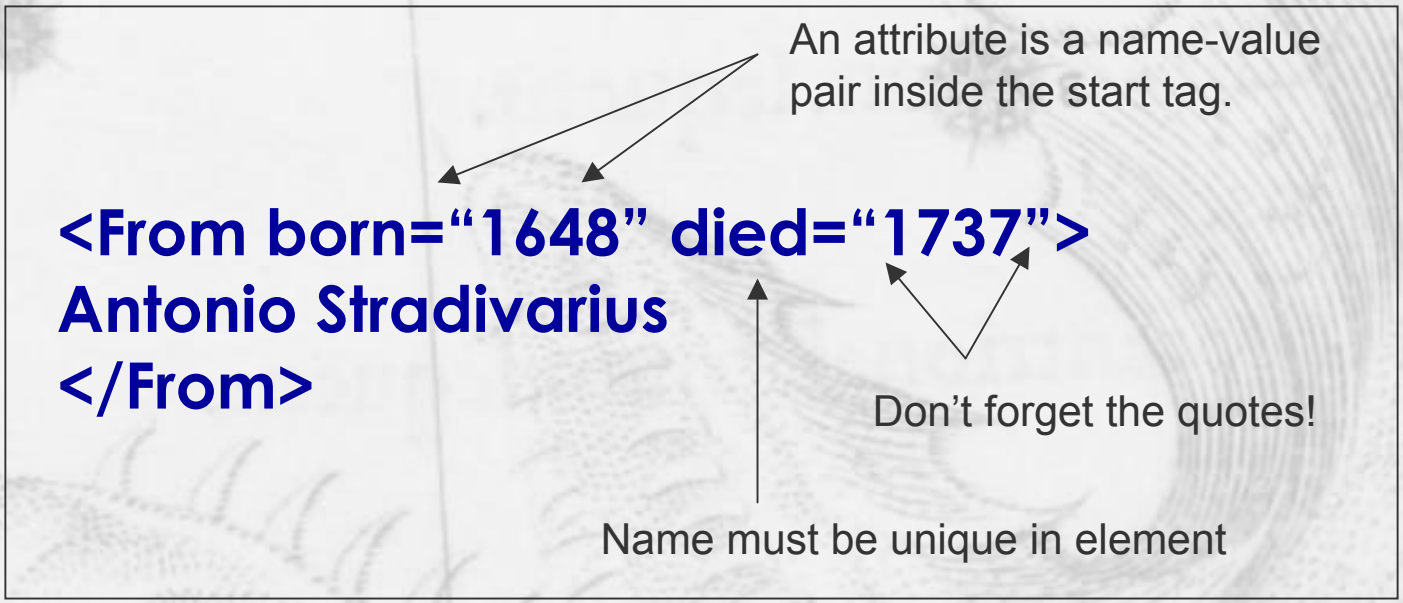



```
<Date>
    <Day>13</Day>
    <Month>4</Month>
    <Year>1723</Year>
</Date>
```

siblings

One element has no parent
Root
or *Document element*

Attributes



<From value="Antonio Stradivarius"/>

Can use an empty element with attributes

Element Names

Names can have **a-Z 0-9 _ - . :**

Colon is reserved for namespaces

Names cannot have **“ ‘ ` $ ^ % ; <>**

<Φρομ> Σοκρατεσ </Φρομ>

This is good XML

<téléphone> 011 33 91 55 46 23 98 </téléphone>

Text in XML

Must escape five symbols

<	<
>	>
&	&
“	"
‘	'

```
H &lt; 3 &amp; K &gt; 4
Patrick O&apos;Reilly
```

Symbol escapes

```
This is Greek theta &#x03B8;
Fran&#xE7;ois not Francois!
```

See http://www.unicode.org

Bulk escape through CDATA

```
<![CDATA[
        H < 3 & K > 4
        Patrick O'Reilly
]]>
```

Other stuff

Comments

<!-- This is a comment -->

Processing Instructions

<?myprinter color="purple" ?>
<?robots ignore="yes" ?>
<?xml-stylesheet type="text/xsl" href="http://us-vo.org/xml/VOTable-basic.xsl"?>

Well-formed XML

- Every start tag must have an end tag match
- Elements may nest, but not overlap
  (<a>this is wrong)
- There must be exactly one root element
- Attribute values must be quoted
- An element cannot have 2 attributes of the same name
- No comments inside tags
- No unescaped <, >, & in element text or attibute text
- Etc etc

Validation (DTD/Xschema)

- XML dialects
  - Applications accept particular types of data
    - Adobe Illustrator takes *Scalable Vector Graphics ML*
    - VO applications take *VOTable*
    - Browser takes *Platform for Privacy Preferences ML*
- Validation checks the XML file
  - Against DTD (Document Type Definition>
  - Against Xschema
- Validation is Optional
- Checks if ***Instance*** is member of ***Class***

DTD

- Inherited from past, not XML
- Example from VOTable.dtd

```
<!-- RESOURCEs can contain other RESOURCES,
     together with TABLEs and other stuff -->
<!ELEMENT RESOURCE (DESCRIPTION?, INFO*, COOSYS*, PARAM*, LINK*,
     TABLE*, RESOURCE*)>
<!ATTLIST RESOURCE
        name CDATA #IMPLIED
        ID ID #IMPLIED
        type (results | meta) "results"
>
```

XSchema

- XML-based document definition
  - Elements can be more complex
    - Type derivation and inheritance
  - Occurrence constraints
    - Eg a marriage has exactly two people
  - Simple data types
    - For Character data and attributes
    - `string`, `integer`, `dateTime`, etc
    - Patterns
      - Eg a US phone number is xxx-xxx-xxxx
  - Namespaces!

Xschema fragment

```
<!-- RESOURCES can contain DESCRIPTION, (INFO|PARM|LINK), (TABLE|RESOURCE) -->
  <xs:element name="RESOURCE">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="DESCRIPTION" minOccurs="0"/>
        <xs:element ref="INFO" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="COOSYS" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="PARAM" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="LINK" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="TABLE" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="RESOURCE" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:token"/>
      <xs:attribute name="ID" type="xs:ID"/>
      <xs:attribute name="type" default="results">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="results"/>
            <xs:enumeration value="meta"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
```

Namespaces

“We took the table and chair dimensions, and wrote them in a table.”

Namespace = mydomain.com/**furniture**

Namespace = mydomain.com/**word-processing**

This is a URI (NOT a URL).

A URI is a unique string.

A URL is an address on the Internet.

FITS keywords have no namespace!

Namespaces

- For reusing document definitions

```
<furniture:table material="oak"/>

<word-processing:table columns="5"/>
```

Xschema Example

```
<?xml version="1.0">
<Date>
        <Day>13</Day>
        <Month>4</Month>
        <Year>1723</Year>
</Date>
```

Instance

Class

```
<?xml version="1.0">
<xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Date">
    <xs:complexType>
      <xs:choice>
        <xs:element name="Day">
        <xs:element name="Month">
        <xs:element name="Year">
      </xs:choice>
    </xs:complexType>
  </xs:element>
```

Xschema Example

```
<xs:element name="Day" type="dayType">

  <xs:complexType name="dayType">
    <xs:simpleContent>
      <xs:restriction base="xs:positiveInteger">
        <xs:maxInclusive value="31"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
```

```
<xs:element name="Month" type="monthType">

<xs:complexType name="monthType">
  <xs:simpleContent>
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="January"/>
      <xs:enumeration value="February"/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

- XML and Structured Data
- XML Syntax
- **VOTable and other formats**
- Transformation, Parsing, Binding

VOTable

- **VOTable** = hierarchy of **Metadata** + **Tables**
- **Metadata** = **Parameters** + **Infos** + **Descriptions** + **Links** + **Fields**
- **Table** = list of **Fields + Data**
- **Data** = stream of **Rows**
- **Row** = list of **Cells**
- **Cell** = **Primitive**
  - or variable-length list of **Primitives**
  - or multidimensional array of **Primitives**
- **Primitive** = integer, character, float, floatComplex, etc

Data in VOTable

- Data expressed in XML
  - <TABLEDATA> <TR><TD>
- Or FITS binary table
  - <FITS><STREAM>
- Or BINARY format
  - simple format, can seek, parallelize
  - <BINARY><STREAM>

VOTable Stream

- STREAM can use different protocols:

- <STREAM href="ftp://server.com/mydata.dat"/>
- <STREAM href="ftp://server.com/mydata.dat" expires="2002-02-22"/>
- <STREAM href="httpg://server.com/mydata.dat" actuate="onLoad"/>
- <STREAM file="file:///usr/home/me/mydata.dat"/>

Data in VOTable

- Table cell is array of *primitives*

datatype	Meaning	FITS	Bytes
"boolean"	Logical	"L"	1
"bit"	Bit	"X"	*
"unsignedByte"	Byte (0 to 255)	"B"	1
"short"	Short Integer	"I"	2
"int"	Integer	"J"	4
"long"	Long integer	"K"	8
"char"	ASCII Character	"A"	1
"unicodeChar"	Unicode Character		2
"float"	Floating point	"E"	4
"double"	Double	"D"	8
"floatComplex"	Float Complex	"C"	8
"doubleComplex"	Double Complex	"M"	16

Metadata in VOTable

- Column header == FIELD
- Has name, ID, unit, accuracy, etc
- Has datatype, arraysize
- Has UCD
  - PHOT_INT-MAG_B — Integrated total blue magnitude
  - ORBIT_ECCENTRICITY — Orbital eccentricity
  - STAT_MEDIAN — Statistics Median Value
  - INST_QE — Detector's Quantum Efficiency

VOTable Example

```
<!DOCTYPE VOTABLE SYSTEM "http://us-vo.org/xml/VOTable.dtd">
<VOTABLE version="1.0">
 <DEFINITIONS>
 <COOSYS ID="myJ2000" equinox="2000." epoch="2000."
system="eq_FK5"/>
 </DEFINITIONS>
 <RESOURCE>
  <PARAM name="Observer" datatype="char" arraysize="*" value="William
Herschel">
   <DESCRIPTION>This parameter is designed to store the observer's name
   </DESCRIPTION>
  </PARAM>
  <TABLE name="Stars">
   <DESCRIPTION>Some bright stars</DESCRIPTION>
   <FIELD name="Star-Name" ucd="ID_MAIN" datatype="char"
arraysize="10"/>
   <FIELD name="RA" ucd="POS_EQ_RA" ref="myJ2000" unit="deg"
        datatype="float" precision="F3" width="7"/>
   <FIELD name="Dec" ucd="POS_EQ_DEC" ref="myJ2000" unit="deg"
        datatype="float" precision="F3" width="7"/>
   <FIELD name="Counts" ucd="NUMBER" datatype="int"
arraysize="2x3x*"/>
```

VOTable Example

```
<DATA>
      <TABLEDATA>
      <TR>
        <TD>Procyon</TD><TD>114.827</TD><TD> 5.227</TD>
        <TD>4 5 3 4 3 2 1 2 3 3 5 6</TD>
      </TR>
      <TR>
        <TD>Vega</TD><TD>279.234</TD>
        <TD>38.782</TD><TD>8 7 8 6 8 6</TD>
      </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
 </RESOURCE>
</VOTABLE>
```

Whitespace separated tokens
for array of primitives

VOTable Example

```
<VOTABLE version="1.0">
  <RESOURCE ID="Stars">
    <PARAM ID="Mass" datatype="float" unit="solMass" value="1"/>
    <RESOURCE ID="BigStars">
      <PARAM ID="Mass-big" datatype="float" unit="solMass" value="10"/>
    </RESOURCE>
    <RESOURCE ID="SmallStars">
      <PARAM ID="Mass-small" datatype="float" unit="solMass" value="0.2"/>
      <RESOURCE ID="VerySmallStars">
        <PARAM ID="Mass-tiny" datatype="float" unit="solMass" value="0.05"/>
      </RESOURCE>
    </RESOURCE>
  </RESOURCE>
</VOTABLE>
```

XDF (NASA Goddard)

- N-dimensional blocks
  - Spatial information
  - Scalar, vector fields on grid
  - Tables of multidimensional

Spectra with their wavelength scales,
images with coordinate axes,
vector fields with unitDirection,
data cubes in complicated spaces,
tables with column headers, and
series of tables with each table having a unique name

XDF Example

```
<XDF>
    <parameter name="date" > <units><unitless/></units>
        <value>01-12-99</value>
    </parameter>
    <structure name="2_vector_spaces">
        <array name="LoRes">
        <units><unit>m/s</unit></units>
        <axis name="vector components" axisId="comps-lo">
            <axisUnits><unitless/></axisUnits>
            <unitDirection axisIdRef="x-lo" name="x-hat" />
            <unitDirection axisIdRef="y-lo" name="y-hat" />
            <unitDirection axisIdRef="z-lo" name="z-hat" />
        </axis>
        <axis name="x" …
        <axis name="y" …
        <axis name="z" …
```

XDF Example

```
        <for axisIdRef="comps-lo">
          <for axisIdRef="x-lo">
            <for axisIdRef="y-lo">
              <for axisIdRef="z-lo">
                 <asciiFormat>
                      <repeat count="4">
                         <ascii type="fixed" width="8" precision="3"/>
                         <skipChar count="1"/>
                      </repeat>
                     <ascii type="fixed" width="8" precision="3"/>
                  </asciiFormat>
                </for>
              </for>
            </for>
          </for>
<data>
<![CDATA[
2432.234 2345.432 2333.553 5234.737 5234.220 5234.334 5234.220
2432.234 2345.432 2333.553 2345.432 2333.553 5234.334 5234.220
. . .
]]></data>
</array>
</XDF>
```

AML: Astronomical Markup Language'

- Standard exchange format for metadata in astronomy
  - astronomical object
  - article
  - table
  - set of tables
  - image
  - person
  - project

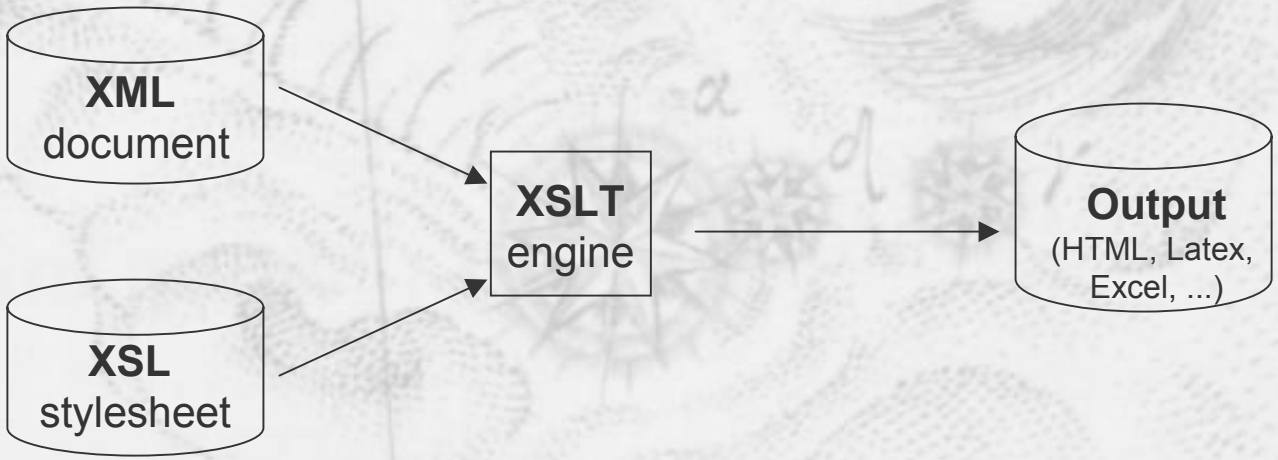
AML Example

```
<AML>
  <AOBJECT>
    <IDENTS>
      <IDENT> UGC 6 </IDENT>
      <IDENT> MCG+04-01-013 </IDENT>
    </IDENTS>

    <COORD coosystem="equatorial">
      <RA>000309.55</RA> <DEC>+215736.4</DEC>
    </COORD>
    <OBJTYPE> Seyfert_2 </OBJTYPE>
    <MORPHO> Sc </MORPHO>
    <RADVELO unit="z"> 0.02226 </RADVELO>
    <DIM unit="arcmin"> 1.1 x 0.8 </DIM>
    <MAG filter="B"> 14.62 </MAG>
    <ORIANGL unit="deg"> 105 </ORIANGL>
    <REFS>
      <REF> 1997ApJS..108..155G </REF>
      <REF> 1997ApJS..108..229H </REF>
    </REFS>
  </AOBJECT>
</AML>
```

- XML and Structured Data
- XML Syntax
- VOTable and other formats
- **Transformation, Parsing, Binding**

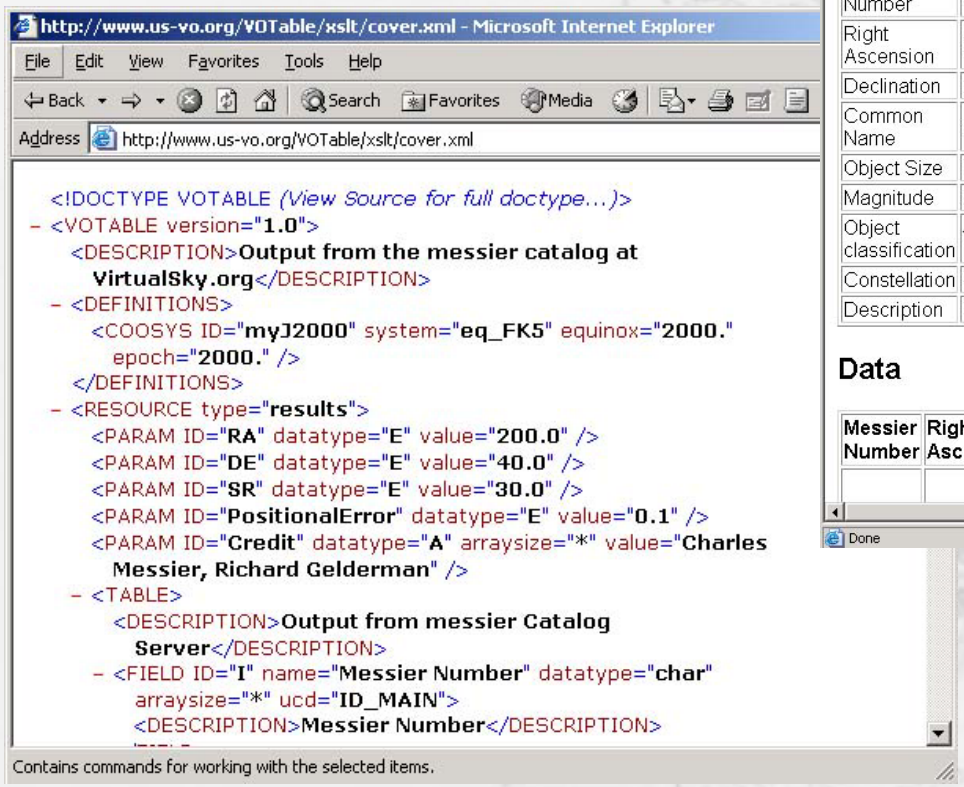
XPath and XSLT

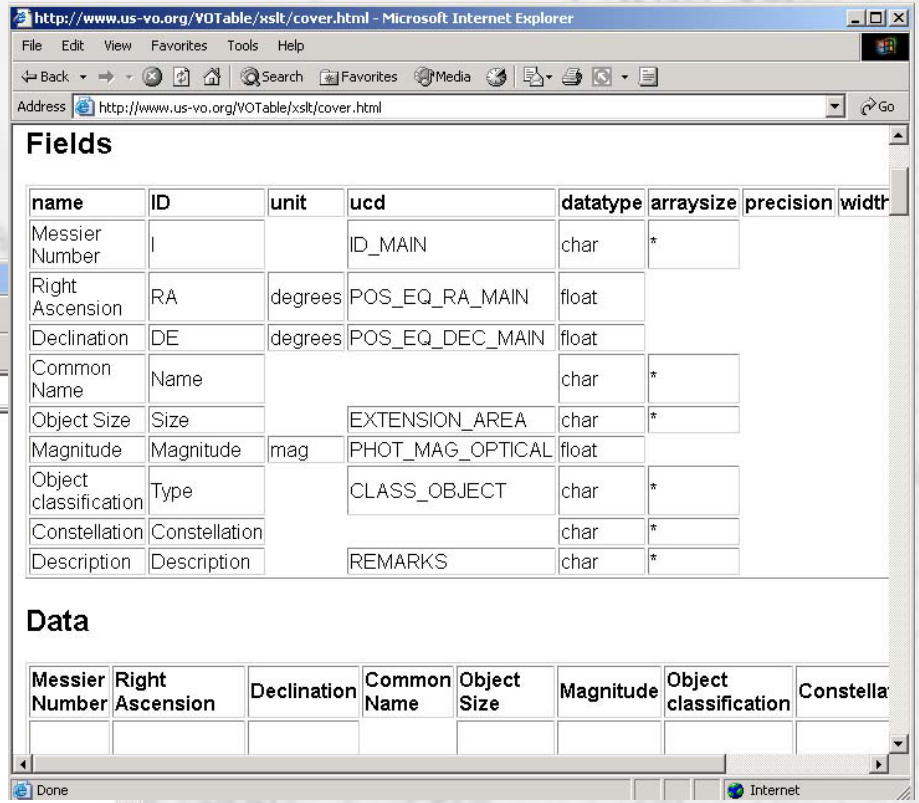
- XSL
  - Extensible Style Language
- XSLT
  - Extensible Style Language Transformation



XSLT example

see http://us-vo.org/VOTable
for details





XSLT in the browser

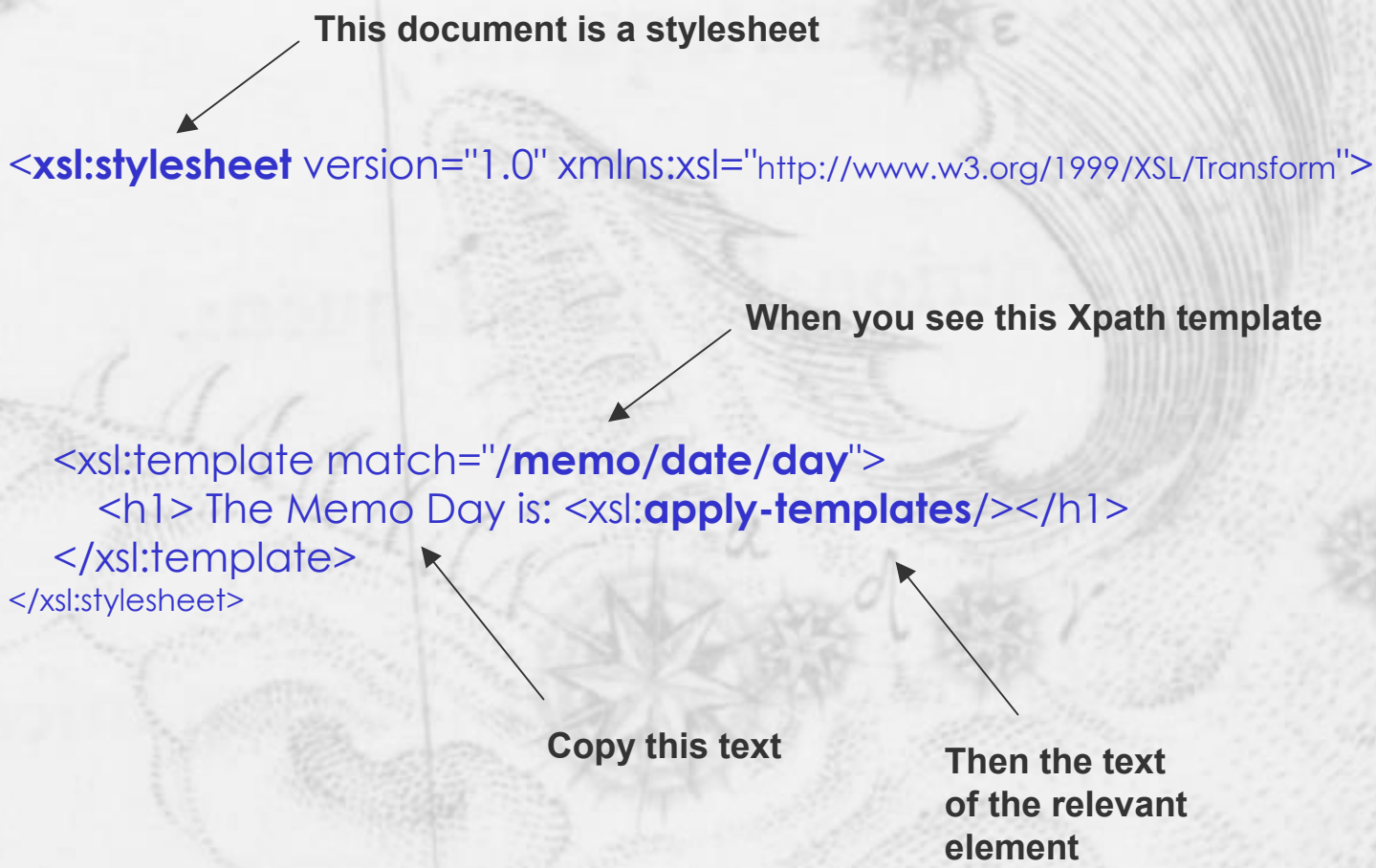
```
<?xml-stylesheet type="text/xsl" href="http://us-vo.org/xml/VOTable-basic.xsl"?>
```

First line of XML document

- ?xml-stylesheet is a *processing instruction*
- Works with Netscape 7
- And IE 6 -- set security to *medium-low*

see http://us-vo.org/VOTable for details

Building XSLT



XML Parsing with SAX

SAX: Event-Based

Handlers for StartElement, Text, EndElement, etc.

```
startElement Memo
startElement From
characters Antonio Stradivarius
endElement From
startElement Date
startElement Day
characters 13
….
```

XML Parsing with SAX

```
try {
        XMLReader parser = XMLReaderFactory.createXMLReader();

        parser.setContentHandler(new myHandler());

        parser.parse("http://musicalmemos.org/strad.xml");
}
catch(SAXParseException e) {
        // Well-formed error
}
catch(SAXException e) {
        // Could not find XMLReader
}
catch(IOException e) {
        // could not read file from net
}
```

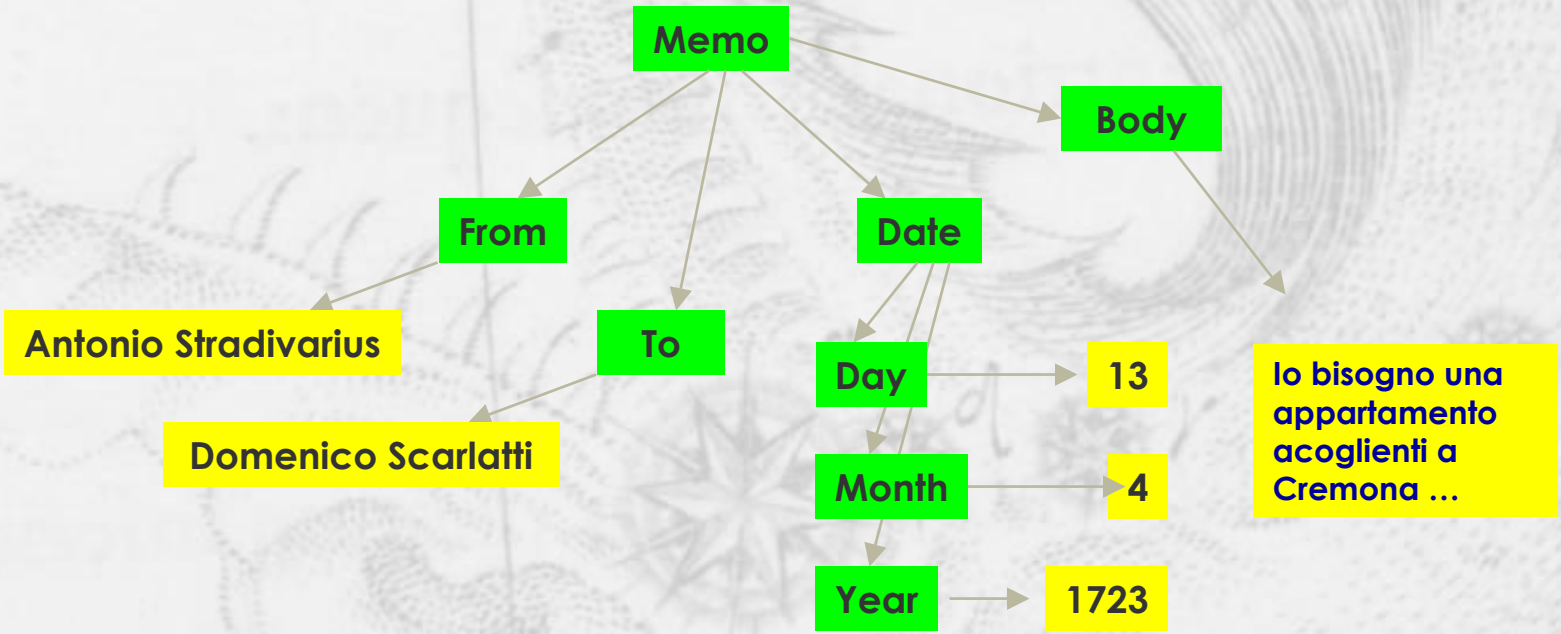
XML Parsing with SAX

```
public class myHandler implements ContentHandler {

        public void startElement(…, String elementName,…, Attributes atts){
        }

        public void endElement(…, String elementName, …){
        }

        public void characters(char[] test, int start, int length){
        }

        + some other methods…
}
```

XML Parsing with DOM

DOM: Document Object Model

Returns a tree-like Document object with data attached



Parsing XML with DOM

```
DOMParser dp = new DOMParser();

dp.parse(("http://musicalmemos.org/strad.xml");

Node nd = dp.getDocument().getDocumentElement();

int count = numberOfNodes(nd);


public int numberOfNodes(Node nd){
        int number = 1;
        Nodelist nl = nd.getChildNodes();
        for(int i=0; i<nl.getLength(); i++)
                if(nl.item(i).getNodeType() == Node.ELEMENT_NODE)
                        number += numberOfNodes(nl.item(i));
}
```

XML Binding

- Automatically makes code from DTD/XSchema
  - eg. Element <Date> generates
    - getDay(), setDay()
    - getMonth(), setMonth()
    - getYear(), setYear()
  - Much easier than building it with DOM

XML Binding

```
Votable v = votw.getVotable();

// just get the first resource -- there may be more that we ignore
Resource r = null;
if(v.getResourceCount() > 0)
        r = (Resource)v.getResourceAt(0);
else …

// just get the first table -- there may be more that we ignore
Table table = null;
if(r.getTableCount() > 0)
        table = (Table)r.getTableAt(0);
else …
```

XML Binding

- Parsing VOTable
  - Finding the RA, dec columns by UCD

```
for(int i=0; i<table.getFieldCount(); i++){

        f = (Field)table.getFieldAt(i);

        String s = f.getUcd(); …

        if(s.equals("POS_RA_EQ_MAIN")
                // this field contains right ascension

        if(s.equals("POS_DEC_EQ_MAIN")
                // this field contains declination
```

XML Binding Tools

Can use binder from
`breezefactory.com`

Also soon
JAXB
`java.sun.com/xml/jaxb/`