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.

This markup is HTML

```
<Memorandum>
From: Antonio Stradivarius
To: Domenico Scarlatti
Date: 13 April 1723
Message: Io bisogno una appartamento acoglienti a Cremona ...
</Memorandum>
```
Structure with XML

```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 accoglienti a Cremona ...
  </Body>
</Memo>
```

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

Structured Data storage

DB

SQL

Web Services
GET, POST, SOAP

XML

XSLT presentation

HTML

forms queries
Service Workflow

User's code → Catalog Service → Query Check Service → Query Estimator → Crossmatch Service → Storage Service → Archive Service

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

Start tag

Content

Antonio Stradivarius

End tag

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 `<br>` and `<hr>` are not proper: should be `<br/>` and `<hr/>*  

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

One element has no parent

*Root*  
*or Document element*
Attributes

An attribute is a name-value pair inside the start tag.

```xml
<From bom="1648" died="1737">
Antonio Stradivarius
</From>
```

Don’t forget the quotes!

Name must be unique in element

```xml
<From value="Antonio Stradivarius"/>
```

Can use an empty element with attributes
Element Names

Names can have a-zA-Z0-9_ - . :

Colon is reserved for namespaces

Names cannot have " ' ` $ ^ % ; <>


This is good XML

<tdéveloppe> 011 33 91 55 46 23 98 </tdéveloppe>
Text in XML

Must escape five symbols

<          &lt;
>          &gt;
&          &amp;
“          &quot;
’          &apos;

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

Symbol escapes
This is Greek theta &amp;#x03B8;
Fran&amp;#xE7;ois not Francois!
See http://www.unicode.org

Bulk escape through CDATA
<![CDATA[
H &lt; 3 &amp; K &gt; 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
  - \(<a><b>this is wrong</b></a></b>\)
- 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 attribute text
- Etc etc
Validation (DTD/Xschemra)

- **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*
Inherited from past, not XML

Example from VOTable.dtd

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

DTD
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|PARAM|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="COOYSYS" 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"/>
<?xml version="1.0"?>
<Date>
  <Day>13</Day>
  <Month>4</Month>
  <Year>1723</Year>
</Date>

<?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>
</xs:schema>
<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>

<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>
</xs:element>
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="http://server.com/mydata.dat" actuate="onLoad"/>
- `<STREAM file="file:///usr/home/me/mydata.dat"/>
## Data in VOTable

Table cell is array of *primitives*

<table>
<thead>
<tr>
<th>datatype</th>
<th>Meaning</th>
<th>FITS</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;boolean&quot;</td>
<td>Logical</td>
<td>&quot;L&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;bit&quot;</td>
<td>Bit</td>
<td>&quot;X&quot;</td>
<td>*</td>
</tr>
<tr>
<td>&quot;unsignedByte&quot;</td>
<td>Byte (0 to 255)</td>
<td>&quot;B&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;short&quot;</td>
<td>Short Integer</td>
<td>&quot;I&quot;</td>
<td>2</td>
</tr>
<tr>
<td>&quot;int&quot;</td>
<td>Integer</td>
<td>&quot;J&quot;</td>
<td>4</td>
</tr>
<tr>
<td>&quot;long&quot;</td>
<td>Long integer</td>
<td>&quot;K&quot;</td>
<td>8</td>
</tr>
<tr>
<td>&quot;char&quot;</td>
<td>ASCII Character</td>
<td>&quot;A&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;unicodeChar&quot;</td>
<td>Unicode Character</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>&quot;float&quot;</td>
<td>Floating point</td>
<td>&quot;E&quot;</td>
<td>4</td>
</tr>
<tr>
<td>&quot;double&quot;</td>
<td>Double</td>
<td>&quot;D&quot;</td>
<td>8</td>
</tr>
<tr>
<td>&quot;floatComplex&quot;</td>
<td>Float Complex</td>
<td>&quot;C&quot;</td>
<td>8</td>
</tr>
<tr>
<td>&quot;doubleComplex&quot;</td>
<td>Double Complex</td>
<td>&quot;M&quot;</td>
<td>16</td>
</tr>
</tbody>
</table>
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>
  </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*"/>
    </TABLE>
  </RESOURCE>
</VOTABLE>
VOTable Example

```xml
<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>
```
VOTable Example

<VOTABLE version="1.0">
  <RESOURCE ID="Stars">
    <PARAM ID="Mass" datatype="float" unit="solMass" value="1"/>
  </RESOURCE>
  <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>
  <RESOURCE ID="VerySmallStars">
    <PARAM ID="Mass-tiny" datatype="float" unit="solMass" value="0.05"/>
  </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>
  <parameter name="date" > <units><unitless/> <value>01-12-99</value> </units> </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" ...
    </array>
  </structure>
</XDF>
XDF Example

```xml
<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>
        </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>
  <AOBJ ECT>
    <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>
    </REFS>
  </AOBJ ECT>
</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

Diagram:
- XML document
- XSL stylesheet
- XSLT engine
- Output (HTML, Latex, Excel, ...
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
This document is a stylesheet

```xml
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/memo/date/day">
    <h1>The Memo Day is: <xsl:apply-templates/></h1>
  </xsl:template>
</xsl:stylesheet>
```

When you see this Xpath template

```xml
<xsl:template match="/memo/date/day">
  <h1>The Memo Day is: <xsl:apply-templates/></h1>
</xsl:template>
</xsl:stylesheet>
```

Copy this text

Then the text of the relevant element

Copy this text
XML Parsing with SAX

**SAX**: Event-Based
Handlers for StartElement, Text, EndElement, etc.

```xml
<startElement Memo From characters Antonio Stradivarius endElement From Date Day startElement characters 13 ...
```
XML Parsing with SAX

```java
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
}
```
public class myHandler implements ContentHandler {

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

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

    public void characters(char[] text, int start, int length)
    {
    
    + some other methods...
    
    }
XML Parsing with DOM

**DOM**: Document Object Model

Returns a tree-like Document object with data attached

---

**Memo**

**Body**

**From**

Antonio Stradivarius

**To**

Domenico Scarlatti

**Date**

Month 4

Day 13

Year 1723

Io bisogno una appartamento acoglienti a Cremona ...
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

 ```java
 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/
Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services
What are Web Services?

Web (From Dictionary.com)
1. A latticed or woven structure
2. Something intricately contrived, especially something that ensnares or entangles.
3. A complex, interconnected structure or arrangement

Shorthand for the World Wide Web
What are Web Services?

Service (From Dictionary.com)
1. The performance of work or duties for a superior or as a servant
2. An act or a variety of work done for others, especially for pay
3. Assistance; help

*Slang terms not suitable for print.*
What are Web Services?

- Web Service
  - Distributed Computing Model
  - Self-Contained Modular Applications
  - Platform Independent
  - Language Independent

Or

- An unpaid act of performing intricately contrived work for others that ensnares all?
public class HelloWorld {
    public java.lang.String getMessage() {
        return "Hello World!";
    }

    public static void main(String[] args) {
        HelloWorld hw = new HelloWorld();
        System.out.print(hw.getMessage());
    }
}
What are Web Services?

A Service that is accessed via the Web!
Who is in Control?

- W3C (www.w3c.org)
- WSDL
- SOAP/XML Protocol
- Web Service Activity
- Oasis (www.oasis-open.org)
- ebXML
- UDDI
- WS-I (www.ws-i.org)
W3C Web Services Activity

The World Wide Web is more and more used for application to application communication. The programmatic interfaces made available are referred to as Web services.

The goal of the Web Services Activity is to develop a set of technologies in order to bring Web services to their full potential. The Web Services Activity Statement explains the W3C's work on this topic in more detail.

Note: The XML Protocol Activity was incorporated in the Web Services Activity in January 2002.

Groups

The Activity is composed of one Coordination Group and currently three Working Groups:

- Web Services Architecture Working Group
- XML Protocol Working Group
- Web Services Description Working Group
How is this different?

- RPC Model Exists!
- CORBA
- COM/DCOM
- RMI
- ...

- Web Services use XML!!!!!
Practical Examples

🔹 Business to Business
  ✓ Inventory Records
  ✓ Bill of Laden
  ✓ Purchase Orders

🔹 Business to Consumer
  ✓ Financial Data
  ✓ Spelling/Search
  ✓ Product Listings
  ✓ Airline Reservations
Develop Your Own Applications Using Google

With the Google Web APIs service, software developers can query more than 2 billion web documents directly from their own computer programs. Google uses the SOAP and WSDL standards so a developer can program in his or her favorite environment - such as Java, Perl, or Visual Studio .NET.

To start writing programs using Google Web APIs:

Download the developer's kit
The Google Web APIs developer's kit provides documentation and example code for using the Google Web APIs service. The download includes Java and .NET programming examples and a WSDL.

Server and developer's kit updates
- Example Visual Basic .NET client
- Better support for multilingual queries
- Java client supports HTTP proxies
Amazon

Web Services How-To

In our Web services how-to section, you will find everything you need to learn more about our program, including:

- Instructions on how to download our developer's kit
- A Web services FAQ
- A selection of books about the burgeoning field of Web services
- A discussions board that features ideas, information and advice from other developers on how to use Amazon.com Web Services
- Technical documents such as:
  - A "heavy" and "lite" version of the DTD
  - A "heavy" and "lite" version of the XSD
  - A WSDL file for our SOAP interface

Also, be sure to check out our announcements board for information about our...
Multiple Invocations

Mockerybird: Conversations with Saleri - Microsoft Internet Explorer

Book Watch

This is a simple implementation of cnfocus.com's wonderful Book Watch web service. Basically, I've retrieved list via his xml feed, then found news information using the Google API, and found the item information via my Amazon API. The result, a constantly changing list of books that are generating buzz around the weblog.com with links to current news items via google. Read more. See also Media Watch.

Small Pieces Loosely Joined: A Unified Theory of the Web

Small Pieces Loosely Joined

1.

Google News Items:
- Small Pieces Loosely Joined
  http://www.smallpieces.com/
- Preface of Small Pieces - David Weinberger
  http://www.smallpieces.com/content/preface

Also:
- The F Ideas Page

Are you reading this book?
Practical Benefits

- Programmatic Access
- Platform/Language Independent
- Compose/Distribute
What about Astronomy

- Name Resolution
  - NED/SIMBAD Models
- Image Access
  - virtualsky
- Catalog Access
  - Intelligent Archive Queries
- Catalog Joins
  - Cross Identification Servers
Cone Search Profiles

![List of Profiles](http://skyserver.pha.jhu.edu/NVOconeprofile/register/showlist.asp)

<table>
<thead>
<tr>
<th>ServiceName</th>
<th>Waveband</th>
<th>Instrument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messier</td>
<td>optical</td>
<td>19-foot refractor</td>
</tr>
<tr>
<td>GSC2221</td>
<td>optical</td>
<td>Plate scans</td>
</tr>
<tr>
<td>HIP</td>
<td>optical</td>
<td>Hipparcos</td>
</tr>
<tr>
<td>GSC1</td>
<td>optical</td>
<td>Plate Scans</td>
</tr>
<tr>
<td>TYC</td>
<td>optical</td>
<td>TYCHO on Hipparcos</td>
</tr>
<tr>
<td>NCSA Astronomy Digital Image Library</td>
<td>radio</td>
<td>multiple instruments</td>
</tr>
<tr>
<td>Yale</td>
<td>optical</td>
<td>various</td>
</tr>
<tr>
<td>DPOSS Plates</td>
<td>optical</td>
<td>various</td>
</tr>
<tr>
<td>ASCA Master Observations</td>
<td>xray</td>
<td>ASCA</td>
</tr>
<tr>
<td>XTE Master Observations</td>
<td>xray</td>
<td>XTE</td>
</tr>
<tr>
<td>OSSE Observations</td>
<td>gammaray</td>
<td>CORO/OSSE</td>
</tr>
<tr>
<td>DPOSS</td>
<td>optical</td>
<td>Palomar Schmidt</td>
</tr>
<tr>
<td>SDSS EDR: PhotoObj</td>
<td>optical</td>
<td>SDSS, Mosaic Camera</td>
</tr>
<tr>
<td>USNO-A2 n</td>
<td>optical</td>
<td>PMN scans of POSS-1+R, UKSRC</td>
</tr>
</tbody>
</table>

*Local intranet*
SDSS EDR Cone Search

![Test Profile](http://skyserver.pha.jhu.edu/NVO/coneprofile/show/test.asp?id=10)
Web Service Paradigm

- Service Oriented Programming
  - Dynamically Locate Services
  - Services are “ON” the Network
  - Services can be coupled
- Multiple Transport Protocols
  - HTTP, SMTP, FTP, …
- Multiple Message Encodings
  - SOAP, XML-RPC, XP(?), …
Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services
Web Service Architecture

- Three Primary Roles
  1. Service provider
  2. Service requester
  3. Service broker

![Diagram showing the relationships between Provider, Requestor, and Broker.]
Web Service Architecture

Framework must support

1. Publishing Service
2. Finding a Service
3. Binding a Service
Web Service Lifecycle

1. Service Must be Created
2. Service Must Be Published
3. Service Must be Easily Located
4. Service Must be Invoked/Called
5. Service must be Unpublished
Service Provider

- Creates the Service
  - New Service
  - Wrap Legacy Service
  - Wrap “Other” Services

- Publishes the Service
  - Registries
  - Standard Hierarchies

- Supports the Web Service
- Unpublishes the Service
Service Broker

- Maintains Service Registry
- Simplifies Service Location
  - Categorization
  - Query Support
Service Requestor

- Locates Service
- Invokes Service
  - Direct Request
  - Indirect Request
The Big Three

- **Service Description – WSDL**
  - The most important, everything else derives from this

- **Service Invocation – SOAP**
  - Dominant Communication Protocol (XML Protocol)

- **Service Publication – UDDI**
  - Being Pushed Hard, but future not clear. (OGSA)
Describing a Service

☞ Web Services Description Language (WSDL)
  http://www.w3.org/2002/ws/desc/

☞ XML Document that provides the public interface to a Web Service

☞ Public Methods
☞ Data Type Information (IN/OUT)
☞ Transport Protocol Binding Information
☞ Service Location

☞ The What, Where, and How!
Invoking a Service

- Simple Object Access Protocol (SOAP)
  - Although as of V1.2 SOAP is no longer an acronym
    - http://www.w3.org/2000/xp/Group/
- XML protocol for exchanging messages
  - Platform/Language Independent
- Different Transport Protocols (General Case)
  - HTTP/HTTTR
  - SMTP
  - FTP
  - BEEP
  - …
Publishing a Service

- Universal Description, Discovery, and Integration (UDDI)
  - http://www.uddi.org (Now under OASIS)
- Technical specification for building WSDL document repositories
  - Documents can be published
  - Document can be searched
  - Formal Hierarchy
- UDDI Registry implements the specification
  - IBM, Microsoft, SAP, etc. have public Registries
  - astrouddi.org (?)
Hello World (WSDL Style)
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="http://localhost:8080/axis/HelloWorld.jws"
xmlns:impl="http://localhost:8080/axis/HelloWorld.jws"
xmlns:intf="http://localhost:8080/axis/HelloWorld.jws"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
...
</wsdl:definitions>
WSDL Document Elements

- `<wsdl:types>`
  The datatypes used by the Web Service

- `<wsdl:message>`
  The abstract definition of the data being transmitted

- `<wsdl:portType>`
  The abstract operations that constitute the Web service

- `<wsdl:binding>`
  The concrete protocol and data format used by the Web service

- `<wsdl:port>`
  The address for a single communication endpoint

- `<wsdl:service>`
  An aggregation of related ports
WSDL Types

- Define the datatypes used as arguments to the Web service as well as the return values from a Web service
- Preferably XML Schema
- XSD namespace
- Must Handle nillable (Java Wrapper Classes)
- SOAP
# WSDL Types

Map WSDL (XSD) to Language (e.g., Java)

<table>
<thead>
<tr>
<th>XSD Type</th>
<th>Java Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>xsd:boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>xsd:byte</td>
<td>byte</td>
</tr>
<tr>
<td>xsd:double</td>
<td>double</td>
</tr>
<tr>
<td>xsd:float</td>
<td>float</td>
</tr>
<tr>
<td>xsd:int</td>
<td>int</td>
</tr>
<tr>
<td>xsd:long</td>
<td>long</td>
</tr>
<tr>
<td>xsd:short</td>
<td>short</td>
</tr>
<tr>
<td>xsd:dateTime</td>
<td>java.util.Calendar</td>
</tr>
<tr>
<td>xsd:decimal</td>
<td>java.math.BigDecimal</td>
</tr>
<tr>
<td>xsd:hexBinary</td>
<td>byte[]</td>
</tr>
<tr>
<td>xsd:base64Binary</td>
<td>byte[]</td>
</tr>
<tr>
<td>xsd:QName</td>
<td>javax.xml.namespace.QName</td>
</tr>
<tr>
<td>xsd:integer</td>
<td>java.math.BigInteger</td>
</tr>
<tr>
<td>xsd:string</td>
<td>java.lang.String</td>
</tr>
</tbody>
</table>
WSDL Types

- Recommended approach
  - Use Elements not Attributes
  - Only define types that refer to abstract content of messages (not protocols)
  - Array types should extend the SOAP Array type
    - Name scheme: ArrayOfXXX
  - Xsd:anyType used to represent any type.
<wsdl:types>

<wsdl:types>
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="ArrayOf_xsd_string">
      <complexContent>
        <restriction base="soapenc:Array">
          <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
        </restriction>
      </complexContent>
    </complexType>
    <element name="ArrayOf_xsd_string" nillable="true" type="impl:ArrayOf_xsd_string" />
  </schema>
</wsdl:types>
Web service Messages

- Interactions between Web service client and server are called messages
- Message element describes the messages that can be exchanged
- Logical definition of a type of message that may be used by operations listed in portType element
  - Input
  - Output
  - Fault Message
- Components
  - Message must have a local name
Web service Messages

Components (wsdl:message element)
- Message must have a local name
- Use WSDL Namespace
- Zero or more Part descriptions
  - part name
  - part type
  - Arguments or return parameters.
  - Should follow XML Schema

Message element Future?
<wsdl:message>
  <wsdl:message name="mainRequest">
    <wsdl:part name="args" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
  <wsdl:message name="getMessageResponse">
    <wsdl:part name="getMessageReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getMessageRequest"/>
</wsdl:message>
<wsdl:message name="mainResponse"/>
WSDL Port Types

WSDL defines four transmission primitives (or operations) that an endpoint can support:

- **One-way (input element)**
  - The endpoint receives a request, but does not send a response.

- **Request-response (input then output element)**
  - The endpoint receives a request, and sends a **correlated** response.

- **Solicit-response (output then input element)**
  - The endpoint sends a response, and receives a **correlated** response.

- **Notification (output element)**
  - The endpoint sends a response, but does not receive a request.
WSDL portType

- A portType element defines the interfaces that a Web service exposes.
- Similar to a
  - Class
  - Module
  - or Function Library
- The operations are the class/module/library methods.
<wsdl:portType>
  <wsdl:portType name="HelloWorld">
    <wsdl:operation name="main" parameterOrder="args">
      <wsdl:input name="mainRequest"
        message="impl:mainRequest"/>
      <wsdl:output name="mainResponse"
        message="impl:mainResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getMessage">
      <wsdl:input name="getMessageRequest"
        message="impl:getMessageRequest"/>
      <wsdl:output name="getMessageResponse"
        message="impl:getMessageResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:portType>
WSDL Binding

- Defines message format
- For a given portType, defines protocol
  - for operations
  - for messages
- Requires unique name attribute
- Type attribute is portType Qname
<wsdl:binding>
  <wsdl:binding name="HelloWorldSoapBinding" type="impl:HelloWorld">
    <wsdlsoap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
    <wsdl:operation name="getMessage">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getMessageRequest">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
          namespace="http://localhost:8080/axis/HelloWorld.jws"/>
      </wsdl:input>
      <wsdl:output name="getMessageResponse">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
          namespace="http://localhost:8080/axis/HelloWorld.jws"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
WSDL Services

- A port defines a single endpoint
- The port can then be used for binding
- Multiple ports can reference the same address with different protocols
- A Service consists of one or more ports
- A service defines a single serviceType
<wsdl:service name="HelloWorldService">
  <wsdl:port name="HelloWorld" binding="impl:HelloWorldSoapBinding">
    <wsdlsoap:address location="http://localhost:8080/axis/HelloWorld.jws"/>
  </wsdl:port>
</wsdl:service>
Invoking a Service

- Use SOAP to communicate messages
  - SOAP Sender to SOAP Receiver
  - Potential SOAP Intermediaries

- Essentially a one-way communication between SOAP nodes.
  - RPC style
  - Document style
SOAP Basics

- Message is wrapped in the Envelope

- Envelope consists of:
  - Header (Optional) used by intermediaries
  - Body contains the actual message
    - Document
    - Service Call

- Fault Handling
  - Child element of body
  - Contains Reason and Code elements
SOAP Basics

✔ Fault Handling (V1.2)
  ✔ Fault Element is a child element of body
  ✔ No other elements in the body

✔ Contains
  ✔ Reason element (Mandatory)
  ✔ Code element (Mandatory)
    - Standard List
  ✔ Detail element (Optional)
  ✔ Node element (Optional)
  ✔ Role element (Optional)
SOAP Request (HelloWorld)

POST /axis/HelloWorld.jws HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.0
Host: localhost
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 407

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getMessage
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="http://localhost:8080/axis/HelloWorld.jws"/>
</soapenv:Body>
</soapenv:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Connection: close
Date: Wed, 09 Oct 2002 21:34:47 GMT
Server: Apache Tomcat/4.0.6 (HTTP/1.1 Connector)
Set-Cookie: JSESSIONID=8A6802F3136B882A53BC0E8E1E30F8CC;Path=/axis

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
 <ns1:getMessageResponse
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
  xmlns:ns1="http://localhost:8080/axis/HelloWorld.jws">
 <getMessageReturn xsi:type="xsd:string">Hello World!</getMessageReturn>
 </ns1:getMessageResponse>
 </soapenv:Body>
</soapenv:Envelope>
Web Service Registries

☞ UDDI Currently Dominant
  ☞ Public Registries
    ☞ IBM, MS, SAP, etc.
  ☞ Private Registries
  ☞ UDDI Functions
    ☞ Describe services
    ☞ Discover businesses
    ☞ Integrate business services

☞ The MetaData Problem
UDDI Registry

- **Business Entity**
  - Can have multiple Services

- **Business Service**
  - Has an associated specification

- **Specification Pointers**
  - Detailed information on service

- **Service Types**
  - Defined by a tModel
  - tModel and WSDL
UDDI Registry

UDDI Business Test Registry
Universal Description, Discovery, and Integration

The IBM Universal Description, Discovery and Integration (UDDI) Business Test Registry is a database that facilitates the rapid participation of business in the e-commerce and business-to-business marketplaces.

You can describe your business, discover business partners, and use the UDDI technical standards to communicate with the world.

The UDDI Business Test Registry makes your company more visible and accessible to potential buyers and marketplaces worldwide.

Not ready to use the IBM UDDI Business Registry? Try out the IBM UDDI Business Test Registry.

How does it work?
UDDI Private Registry

- Some development tools or products provide private UDDI registry server
  - Java WS Developer pack.
  - Oracle JDeveloper
  - IBM WS toolkit
  - MS VS .NET

- Greater control, no registration!
Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services
Building Web Services

☞ Simple Demonstration of Deploying a Web service

☞ Use Java (but other options exist: .NET, Perl, python, etc.)
Tomcat Server

The Jakarta Project
http://jakarta.apache.org

Tomcat
- Front Page
- News & Status
- Bug Database
- Resources
- IRC

Documentation
- Tomcat 3.2
- Tomcat 3.3
- Tomcat 4.0
- Tomcat 4.1

Apache Tomcat
Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process.

Tomcat is developed in an open and participatory environment and released under the Apache Software License. Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project. To learn more about getting involved, click here.

Tomcat Versions
For the impatient, current Tomcat production quality releases vs. Servlet/JSP specifications:
Introduction

NEWS (October 7, 2002): Axis 1.0 is now available!

Apache AXIS is an implementation of the SOAP ("Simple Object Access Protocol") submission to W3C.

From the draft W3C specification:

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.
Installation & Setup

- Install Tomcat
- Deploy Axis web apps into Tomcat webapps directory.
- Start Tomcat Server
- Validate AXIS Installation
Web Service Deployment

- Simple Technique (JWS)
  - Copy Java Source file containing the method(s) to be exposed to axis directory
    - HelloWorld.java -> HelloWorld.jws

- Complex Technique (WSDD)
  - AXIS solution
  - Web Service Deployment Descriptor
  - Annotations (.NET approach)
    - [WebMethod]
Hello World (Java)

```java
public class HelloWorld {
    public java.lang.String getMessage() {
        return "Hello World!" ;
    }
}
```
using System.Web.Services;
public class HelloWorld : WebService {
    [WebMethod]
    public string getMessage() {
        return "Hello World!";
    }
}

View WSDL

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions
  targetNamespace="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:impl="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:int="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:apacheSoap="http://xml.apache.org/xml-soap"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsd1/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsd1/"
  xmlns="http://schemas.xmlsoap.org/wsd1/">
  <wsdl:types>
    <schema
      xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://localhost:8080/axis/HelloWorld.jws">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <complexType name="ArrayOf_xsd_string">
        <complexContent>
          <restriction base="soapenc:Array">
            <attributeRef ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]"/>
          </restriction>
        </complexContent>
      </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="mainRequest">
    <wsdl:part name="args" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
  <wsdl:message name="getMessageResponse">
    <wsdl:part name="getMessageReturn" type="xsd:string"/>
  </wsdl:message>
</wsdl:definitions>
```
Web Service Client

- Generate Client Stub from WSDL
  - wsdll2java tool included with AXIS

  `>java org.apache.axis.wsdll.WSDLL2Java

  - Generates
    - `localhost\HelloWorld.java`
    - `localhost\HelloWorldService.java`
    - `localhost\HelloWorldServiceLocator.java`
    - `localhost\HelloWorldSoapBindingStub.java`
Utilizing the Stub Classes

 HelloWorldClient.java

```java
package localhost;

public class HelloWorldClient {
    public static void main(String[] args) throws Exception {
        // Make a service
        HelloWorldService service = new HelloWorldServiceLocator();

        // Now use the service to get a stub
        HelloWorld port = service.getHelloWorld();

        System.out.println(port.getMessage());
    }
}
```
AXIS Extras

Generate Server Skeleton Stub from WSDL


Generates

localhost\HelloWorldSoapBindingImpl.java

More arguments for additional functionality
AXIS TCP Monitor

POST /axis/HelloWorld.js HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.0
Host: localhost
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""

HTTP/1.1 200 OK
Connection: close
Date: Fri, 11 Oct 2002 15:30:09 GMT
Server: Apache Tomcat/4.0.5 (HTTP/1.1 Connector)
Set-Cookie: JSESSIONID=61487605F7E2D6C6781454A58F1DCFB7; Path=/axis

<?xml version="1.0" encoding="UTF-8"?>

  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:axis="http://ws.apache.org/axis"
  xmlns:soap="http://schemas.xmlsoap.org/soap/"
  xmlns:a="http://schemas.xmlsoap.org/soap/actor/uas"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapheaders="http://schemas.xmlsoap.org/soap/headers">
  <SOAP-ENV:Body>
    <soap:Envelope>
      <soap:Body>
        <axis:HelloWorld xmlns:axis="http://ws.apache.org/axis">
          <actor:Content xsi:type="axis:HelloWorld">Hello World</actor:Content>
        </axis:HelloWorld>
      </soap:Body>
    </soap:Envelope>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services
Roadblocks or Speedbumps?

- Reliable Protocol Needed (HTTPR, BEEP)
- Lack of State
- Implementation Inconsistencies
  - unsigned
  - multipart/structures
- Security!
Reliable Protocols

HTTP – Reliable HTTP

IBM Initiative


Adds Persistence to HTTP

BEEP (Blocks Extensible Exchange Protocol)

http://www.ietf.org/rfc/rfc3080.txt

Connection-oriented

Asynchronous interactions
DIME (Direct Internet Message Encapsulation)

- General purpose binary message format
- Enable Web services to efficiently handle multiple attachments
  - Encrypted messages
  - Graphics
  - Multimedia content
  - General Documents
- DIME Message (application/dime)
  - 1+ records to deliver payload
  - Chunking

BPEL4WS

- Business Process Execution Language for Web Services
  - Implementing executable business processes.
  - Describing non-executable abstract processes.

- Merging of WSFL and Xlang
  - Ugliest WS Acronym award

- Define new Web service as a composition of existing Web services
Simplify Development/Deployment

Launches the Web Service Publishing Wizard, which allows you to publish methods available in a Java class as web services.

To enable this option, you must select a project or a file within a project in the Navigator.
J2EE Web Services

- Java APIs for XML
  - JAX-RPC
  - JAXM (SAAJ)
  - JAXR

- JSR 109 - Implementing Enterprise Web Services

- JSR 110 - Java APIs for WSDL
Security

- **Issues include**
  - Message Integrity
  - Message Confidentiality
  - Authentication

- **Technologies include**
  - Secure Sockets Layer (SSL)
  - Transport Layer Security (TLS)
  - Message Encryption
  - Digital Signatures

- But Standards !!!!!
Summary

- Web services provide a powerful programming paradigm
- Mucho Hype
- Looking for Real Applications (NVO)
- Open Grid Services Architecture (OGSA)