Argonne NATIONAL LABORATORY

# XML At Work

*Debby Quock*

*PCaPAC 2006*

THE UNIVERSITY OF CHICAGO

Office of Science U.S. DEPARTMENT OF ENERGY

*A U.S. Department of Energy laboratory managed by The University of Chicago*

# Basics of XML

- Why XML?
- History and Development of Markup Languages
- XML Document
- XML Derivatives: Validation, HTML Style Sheets, DOM Parser
- Examples of XML at Work
- Summary of Key XML Concepts

# *Why XML?*

- Relational Databases:
  - Good for static-onsite data storage and retrieval
  - Are not designed for sending or receiving information, or exchanging data between applications
  - Example: APS IOC Viewer
    http://maia.aps.anl.gov/irmis2/top/irmis2.php

- XML (Extensible Markup Language)
  - Is a data modeling standard specifically designed to allow data flow between systems
  - In some situations, XML is better than using a relational database
  - Will find situations where it is beneficial to combine relational databases and XML documents

# *Markup Languages*

- Meta data is data that describes the real data
- Markup Language is a method of conveying meta data using string literals or tags to delimit and describe the data

- SGML (Standard Generalized Markup Language)
  - Adopted in 1986 by ISO as a standard, ISO8879
  - Used tags to describe meta data
  - Powerful and complex, too complex…

- XML
  - In 1996, World Wide Web Consortium (W3C) designed a simple markup language based on SGML
  - XML 1.0 became a W3C recommendation in 1998
  - XML 1.0 4th Edition standard is available at: http://www.w3.org/TR/REC-xml/

## *XML Document*

XML document has 3 main sections:

- XML Prolog
- XML Body
- XML Epilog

# XML Document Example – Input/Output Controller (IOC)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="aps_ioc_list.xsl"?>
<!-- This is a comment at the end of the XML Prolog -->
<aps_ioc_list xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="aps_ioc_list.xsd">
    <ioc name="iocs1vp">
        <card slot_number="1">
            <component_type>MVME 2100</component_type>
            <description>MVME 2100</description>
            <manufacturer>Motorola</manufacturer>
            <form_factor>B-size Eurocard</form_factor>
            <functions>CPU,UserPgmDevice,EPICS_CPU</functions>
            <cognizant_person>Ned Arnold</cognizant_person>
            <engineering_documentation>Documentation not available</engineering_documentation>

        <feature_sheet>http://www.aps.anl.gov/asd/controls/Motorola/MVME2100/MVME2100_Motorola_fs.pdf</feature_sheet>
        </card>
        <card slot_number="2">
            <component_type>VTM100</component_type>
            <description>Transition Module for PPC; 100Mbit redundant fiber</description>
            <manufacturer>ANL-APS-CTL</manufacturer>
            <form_factor>B-size Eurocard</form_factor>
            <functions>Network, Transition</functions>
            <cognizant_person>Bob Laird</cognizant_person>

        <engineering_documentation>http://www.aps.anl.gov/asd/controls/timing/vtm100/index.html</engineering_documentation>

        <feature_sheet>http://www.aps.anl.gov/asd/controls/documentation/components/custom/vtm100_fs.pdf</feature_sheet>
        </card>
    </ioc>
</aps_ioc_list>
<!--This is a comment after the root element closing tag -->
```

**Prolog**

**Body**

**Epilog**

# XML Document Components

- Prolog
  - XML declaration
  - Processing instructions
  - Comments <! -- Comments are not parsed -->
  - The Document Type Definition <!DOCType doc …>
  - Internal subset declarations
- Body
  - Elements
  - Character Data
  - Attributes (must have values)
  - Processing Instructions
  - Comments
  - CDATA Sections <! [CDATA [ …data here is not parsed…] ]>
- Epilog
  - Comments
  - Processing Instructions
  - Found after the document root element closing tag

# XML Markup Language Rules

- Every element has a starting tag, an optional content, and an ending tag
- Mixing text and sub-elements is allowed by the XML Standard
- Nesting elements:
  - When a closing tag is found, the name of the closing tag must match the name of the last opening tag found
- **Well-Formed:** Follows XML 1.0 specifications
- **Valid:**
  - Is a well-formed XML document that successfully instantiates an XML vocabulary.
  - An XML vocabulary defines the rules to describe information about a particular topic using XML. Example is a vocabulary to describe IOC crates.
- How does one validate an XML document?
  - Document Type Definition (DTDs)
  - Schemas (XSDs)

# XML DTDs (Document Type Definition)

- Internal DTD declaration
  - Inside the XML document have code:

```
<! DOCTYPE  ioc[
    <!ELEMENT ioc(card*)>
]>
```

- External DTD declaration
  - Are used for creating a common DTD that can be shared by multiple documents
  - Inside the XML document have code referencing another file:

```
<! DOCTYPE root_element SYSTEM http://my_uri_for_my/doc.dtd>
```

- Shortcomings of DTDs
  - Not XML language compliant
  - Cannot validate data types
  - Cannot validate element contents
  - An XML document can only refer to one external DTD
- So…  W3C introduced XML Schemas as a better mechanism for defining XML vocabularies ☺

# XML Schemas

Schemas can validate the same things as DTDs while offering several advantages:

- XML Schemas are XML documents (must be well-formed)
- XML Schemas support data types (date, string, integer, …)
- XML Schemas support constraints for element and attribute values
- XML Schemas allow the use of namespaces
- XML Schemas can be inherited and reused

# XML Schema Example – Input/Output Controller (IOC)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="aps_ioc_list">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ioc" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="card" maxOccurs="10">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="component_type" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="manufacturer" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="form_factor" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="functions" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="cognizant_person" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="engineering_documentation" type="xs:string" minOccurs="0"/>
                    <xs:element name="feature_sheet" type="xs:string" minOccurs="1" maxOccurs="1"/>
                  </xs:all>
                  <xs:attribute name="slot_number" use="optional"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

# What else can be done with XML documents?

*XML Processing!*

- *Display eloquently on the Web with XSL (Extensible Style Sheet Language)*
- *Parse with DOM, SAX, and other XML software tools available in Java, PHP…*
- *Transpose*
  - Non-XML to XML
  - XML to Non-XML
- *Computations*
- *Graphics, SVG (Scalable Vector Graphics)*
- *VoiceXML*
- *Distributed Applications*
  - XML-RPC (Remote Procedure Call)
  - SOAP (Simple Object Access Protocol)

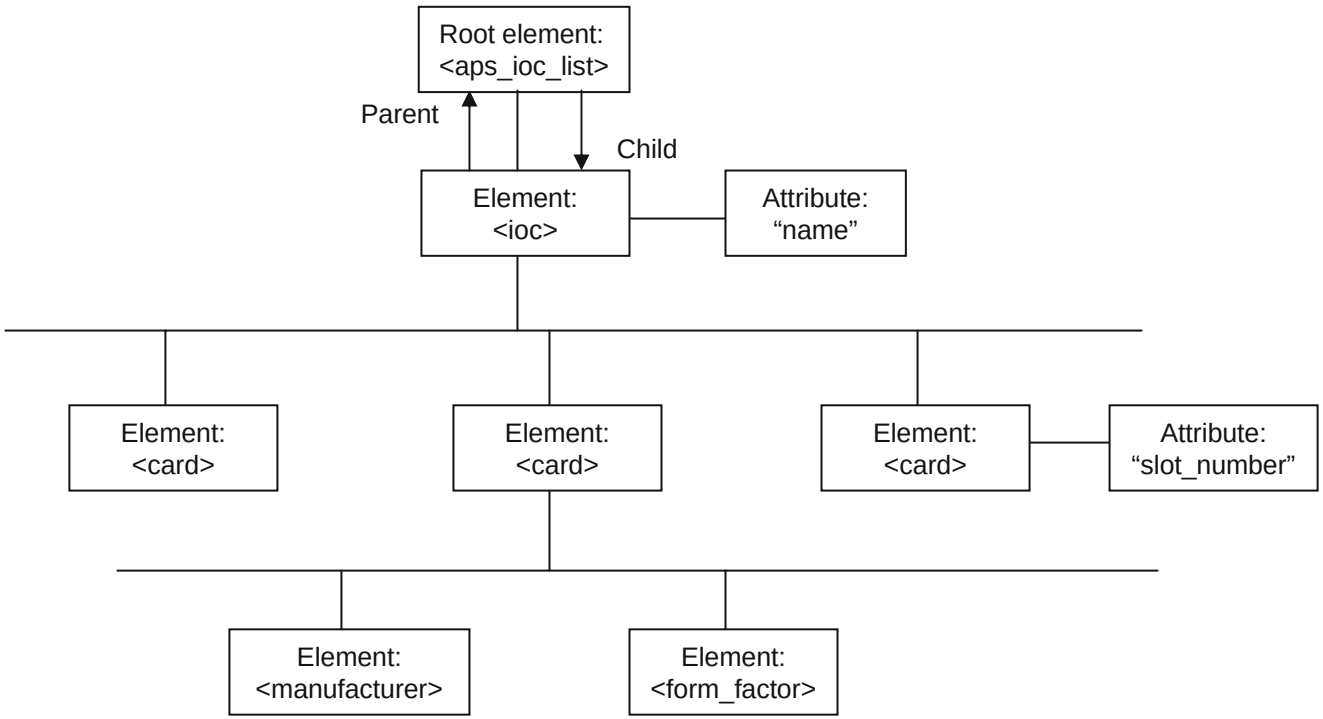
# XSL (Extensible Style Sheet)

- *Specification proposed by W3C to describe how XML documents can be recognized, styled, and transformed*
- *Similar to Cascading Style Sheets (CSS) used on HTML documents but much more powerful*
- *XSL is not as sophisticated as DOM or SAX parsers, but useful for small XML documents*
- *XSL can perform calculations and if, when, and choose conditional processing*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My IOCs</h2>
      <h3><xsl:value-of select="aps_ioc_list/ioc/@name"/></h3>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Crate Slot Number</th>
            <th>Component Type</th>
            <th>Description</th>
            <th>Manufacturer</th>
            <th>Form Factor</th>
            <th>Functions</th>
            <th>Cognizant Person</th>
            <th>Engineering Documentation</th>
            <th>Feature Sheet</th>
          </tr>
          <xsl:for-each select="aps_ioc_list/ioc/card">
            <tr>
              <td><xsl:value-of select="@slot_number"/></td>
              <td><xsl:value-of select="component_type"/></td>
              <td><xsl:value-of select="description"/></td>
              <td><xsl:value-of select="manufacturer"/></td>
              <td><xsl:value-of select="form_factor"/></td>
              <td><xsl:value-of select="functions"/></td>
              <td><xsl:value-of select="cognizant_person"/></td>
              <td><xsl:value-of select="engineering_documentation"/></td>
              <td><xsl:value-of select="feature_sheet"/></td>
            </tr>
          </xsl:for-each>
        </table>
    </body>
  </html>
</xsl:template></xsl:stylesheet>
```

# XML Parsing – DOM

- The XML Document Object Model (XML DOM) defines a standard way for accessing and manipulating XML documents
- The DOM presents an XML document as a tree-structure (a node tree), with the elements, attributes, and text defined as nodes



# DOM Example -- Microsoft DOM Parser with JavaScript

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>
<body>

<script type="text/javascript">

xmlDoc=loadXMLDoc("aps_ioc_list.xml");
var x=xmlDoc.getElementsByTagName('description');

for (i=0;i<x.length;i++)
  {
    document.write("Card Description: ")
    document.write(x[i].childNodes[0].nodeValue)
    document.write("<br />")
  }

</script>
</body>
</html>
```

```
function loadXMLDoc(dname)
{
  var xmlDoc;
  // code for IE
  if (window.ActiveXObject)
  {
    xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
  // code for Mozilla, Firefox, Opera, etc.
  else if (document.implementation &&
                    document.implementation.createDocument)
  {

xmlDoc=document.implementation.createDocument("","",null);
  }
  else
  {
    alert('Your browser cannot handle this script');
  }
  xmlDoc.async=false;
  xmlDoc.load(dname);
  return(xmlDoc);
}
```

*C:\quock\PCaPAC 2006\xml_tutorial\aps_ioc_list_parser.html*

# XML References

**Educational**

- XML Spy editing software http://www.xmlspy.com/
- Online XML tutorial http://www.w3schools.com/xml/

**FUN!**

- The Google Maps JavaScript API lets you embed Google Maps in your own web page. Uses XML to describe latitude and longitude data. http://www.google.com/apis/maps/documentation/
- RSS (Rich Site Summary) is a simple XML-based system that allows users to subscribe to their favorite websites. http://en.wikipedia.org/wiki/RSS_(file_format)

# *Summary - XML Key Concepts*

- XML Document is useful for data that is organized in a tree-like structure
- DTDs and Schemas are used to validate XML documents
  - Enforce data composition and data relationship rules
- XSL is a language used to process XML documents
  - Can perform if, when, choose, calculations, and other actions on data
- Applications built in Java, PHP, Python, JavaScript, etc. can process XML documents