CS 580 Client-Server Programming
Spring Semester, 2009
Doc 17 XML & SOAP
21 Apr 2009

Computers changed how to produce

Books
Manuals
Forms
Presentations

How to represent a document?

Imbed commands or tags in the text

What should the commands do?

Format output

<bold> See the cat run </bold>

Structure the document

<ChapterHeader>See the cat run</ChapterHeader>

<header>Short History of Tags</header>

GenCode
- Late 1960s
- Used descriptive content tags (commands)

Generalized Markup Language (GML)
- Developed by IBM

Standard Generalized Markup Language (SGML)
- 1983 ANSI Standard
- System for developing tags for documents
- Used to create books, manuals, forms, etc
- Widely used by IBM, IRS, DOD etc.
- Not well known in computer industry

HTML

Markup language for WWW

1991 - first mention by Tim Berners-Lee

Wide spread use
Fixed set of tags

Some tags are presentational

```
<CENTER> <B>
```

Web Browsers permit poorly formed HTML

```
<A NAME="WhichOne"></a>
<b><center>Hello World</b></CENTER><Br>
<A NAME="WhichOne"></A>
```

These problems with HTML restrict Web functionality

XML

XML creators wanted
- Flexibility of SGML
- Simplicity of HTML

1998 - Version 1.0 of XML

Key differences from HTML
- Presentation is separate from document description
- Error Checking
- Unambiguous Structure

Uses
- Share Structured Data
- Encode documents
- Serialize data

XML is about

Document structure
Describing data

```
<?xml version="1.0" ?>
<CATALOG>
    <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    <CD>
        <TITLE>Hide your heart</TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>CBS Records</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1988</YEAR>
    </CD>
</CATALOG>
```

Parts of The XML Universe

Basic Syntax

XML 1.0 spec, XML 1.1

XLinks

Namespaces

Markup Languages

XHTML

MathML

SMIL

VoiceXML

Data Addressing & Query

XPath

XPointer

XML Query Language (XQL)

Document Modeling

Document Type Definitions (DTDs)

XML Scheme

Presentation and Transformation

XML Stylesheet Language (XSL)

XSL Transformation Language (XSLT)

Cascading Style Sheets (CSS)

Extensible Stylesheet Language for Formatting Objects (XSL-FO)

Levels of XML

Well-formed
XML document that satisfies basic XML structure

Valid
XML document that is well-formed and
Specifies which tags are legal

A Document Type Definition (DTD) is use to specify

- Legal tags
- Correct tag nesting

Well-Formed XML Documents

Optional Prolog
Root Element

```
<?xml version="1.0" ?>
<!-- A comment -->
<greetings>
   Hello World!
</greetings>
```

```
<?xml  version="1.0"  encoding='iso-8859-1' standalone=no ?>
<!-- A comment -->
<facts>
   <statement>We know 5 &lt; 10</statement>
   <statement><![CDATA[We know 5 <10]]></statement>
</facts>
```

Valid XML Documents

XML document that is well-formed and
Specifies which tags are legal

A Document Type Definition (DTD) is use to specify
- Legal tags
- Correct tag nesting

```
<?xml version="1.0" ?>
<!DOCTYPE greetings [
    <!ELEMENT   greetings    ( from, to, message, date?)>
    <!ELEMENT   from         ( name )>
    <!ELEMENT   to           ( name )>
    <!ELEMENT   message      ( #PCDATA )>
    <!ELEMENT   date         ( #PCDATA )>
    <!ELEMENT   name         ( #PCDATA )>
]>
<greetings>
    <from><name>Roger</name></from>
    <to><name>World</name></to>
    <message>Hi</message>
</greetings>
```

XML Namespaces

An XML namespace is a group of Elements & Attributes

XML namespaces allows mixing of elements from different DTDs

```
<?xml version="1.0" ?>
<whitney:greetings  xmlns:whitney=”http://www.eli.sdsu”
        xmlns:godot=”http://www.waiting.com”>
   <whitney:from>
        <godot:firstname>Roger</godot:firstName>
   </whitney:from>
   <whitney:to>
        <godot:firstname>John</godot:firstName>
   </whitney:to>
   <whitney:message>Hi</whitney:message>
</whitney:greetings>
```

XML Schema (XSD)

A way to define XML documents

- Spec published in 2001
- Allow element content to have a type
- Allow element content to be restricted

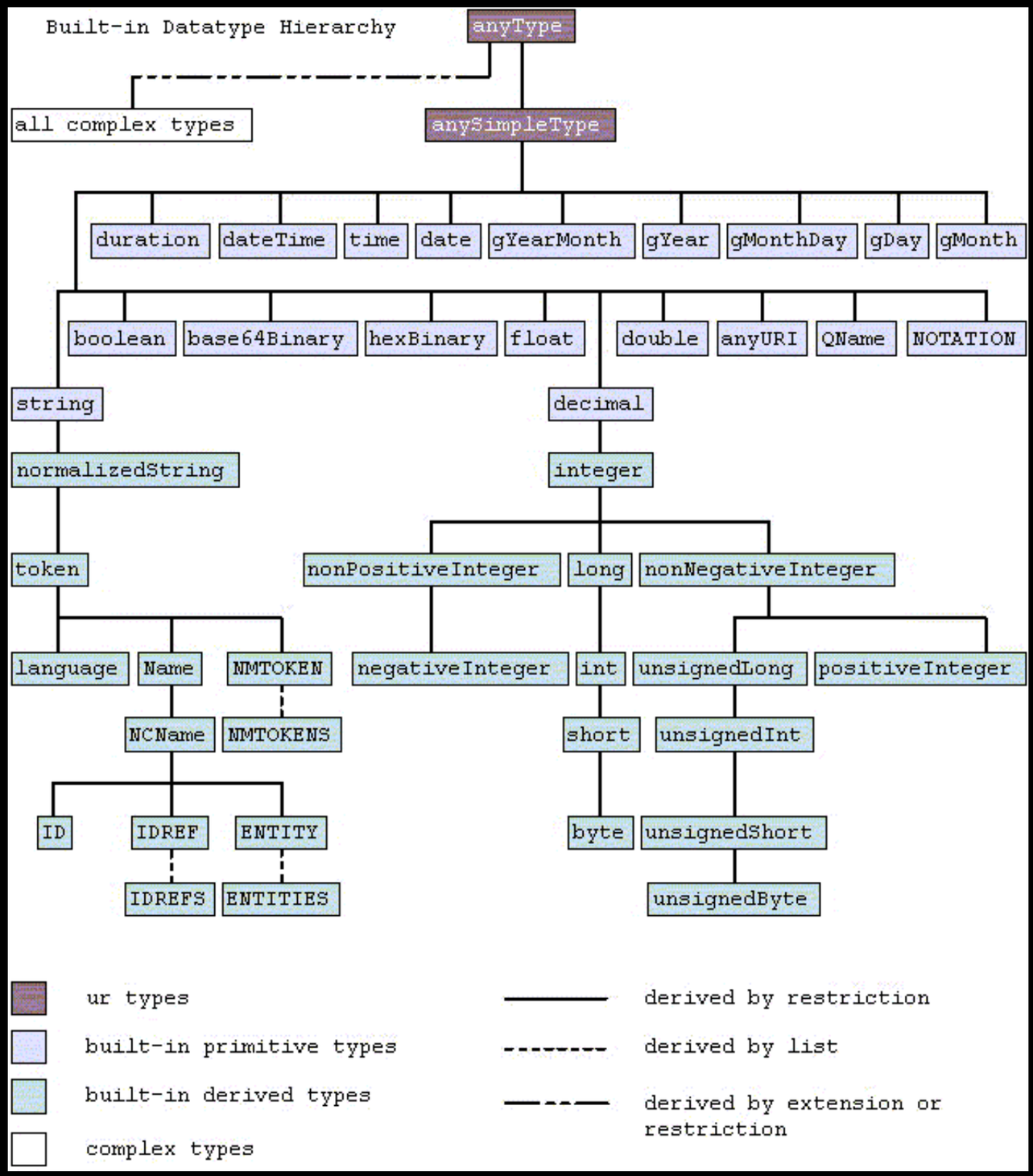
Sample Schema parts

```
<datatype name="Price">
   <scalar datatype="float" decimals="2"/>
</datatype>

<datatype name="MovieTicketPrice">
   <scalar datatype="Price" digits="1"
      maxvalue="9.50" minvalue="1.50"/>
</datatype>

<elementtype name="MovieTicket">
   <model>
      <element name="MovieTitle" type="string"/>
      <element name="TicketPrice" type="MovieTicketPrice">
   </model>
</elementtype>
```

XML Using Schema

```
<MovieTicket>
   <MovieTitle>Gone With the Wind</MovieTitle>
   <TicketPrice>6.50</TicketPrice>
</MovieTicket>
```

Basic Types in XML Schema



http://www.w3.org/TR/xmlschema-2/

RPC - Remote Procedure Call

Client "directly" calls a function on the server

Issues

Cross platform

Marshalling/unmarshalling of parameters and results
  How can one handle pointers as parameters?

Different contexts of client and server

Registering and finding servers

Sample Uses

Unix NFS (Network File System)
Unix license managers

RPC implementations
SUN RPC
Distributed Computing Environment (DCE)
XML-RPC

XML-RPC

RPC using
- HTTP as transport layer and
- XML to encode request/response
- Language and platform independent

Started by Userland (http://frontier.userland.com/) in 1998

Languages/Systems with XML-RPC implementations

Java, Perl, Python, Tcl, C, C++, Smalltalk
ASP, PHP, AppleScript, COM
Zope, WebCrossing

Led to the development of SOAP

Java Implementation

Apache maintains a Java version of XML-RPC

http://ws.apache.org/xmlrpc/xmlrpc2/index.html

Examples to follow use version 2.0.1

Handles for you

- Network connections
- Sending request to server
- Reading requests
- Responding to requests

XML-RPC Supported Datatypes

XML-RPC data type	Java
<i4> or <int>	java.lang.Integer
<boolean>	java.lang.Boolean
<string>	java.lang.String
<double>	java.lang.Double
<dateTime.iso8601>	java.util.Date
<struct>	java.util.Hashtable
<array>	java.util.Vector
<base64>	byte[ ]

http://www.xmlrpc.com/spec

Complex Datatype Examples

```
<struct>
  <member>
    <name>lowerBound</name>
    <value><i4>18</i4></value>
    </member>
  <member>
    <name>upperBound</name>
    <value><i4>139</i4></value>
    </member>
</struct>
```

```
<array>
  <data>
    <value><i4>12</i4></value>
    <value><string>Egypt</string></value>
    <value><boolean>0</boolean></value>
    <value><i4>-31</i4></value>
    </data>
  </array>
```

JSON Equivalents

```
{"lowerBound":18,"upperBound":139}
```

```
[12,"Egypt",false,-31]
```

Example - Add Server

```
import org.apache.xmlrpc.*;

public class AddServer {
    public Integer addtwo(int x, int y)  {
        return new Integer( x + y);
     }

    public static void main( String[] args)  {
        try {
            System.out.println("Starting server on port 8080");
            WebServer addTwoServer = new WebServer(8080);
            addTwoServer.addHandler("examples", new AddServer());
            addTwoServer.start();
            System.out.println("server running");
        }
         catch (Exception webServerStartError) {
            System.err.println( "JavaServer " + webServerStartError.toString());
         }
    }
}
```

Client can access all public instance methods in AddServer

Example - Client

```
import java.util.*;
import org.apache.xmlrpc.*;

public class XmlRpcExample {
    public static void main (String args[])  {
        try {
            XmlRpcClient xmlrpc =  new XmlRpcClientLite("http://127.0.0.1:8080/");
            Vector parameters = new Vector ();
            parameters.addElement (new Integer(5) );
            parameters.addElement (new Integer(3) );

            Integer sum = (Integer) xmlrpc.execute("examples.addtwo", parameters);

            System.out.println( sum.intValue() );
        } catch (java.net.MalformedURLException badAddress) {
            badAddress.printStackTrace( System.out);
        } catch (java.io.IOException connectionProblem) {
            connectionProblem.printStackTrace( System.out);
        } catch (Exception serverProblem) {
            serverProblem.printStackTrace( System.out);
        }
    }
}
```

Client Request - As sent on network

```
POST / HTTP/1.1
Host: 127.0.0.1:8080
Content-type: text/xml
Content-length: 190
User-Agent: Smalltalk XMLRPC version 0.5 (VisualWorks® NonCommercial, 7.4.1 of May 30, 2006)
Connection: Keep-Alive

<?xml version="1.0"?>
  <methodCall>
    <methodName>examples.addtwo</methodName>
    <params>
      <param><value><int>5</int></value></param>
      <param><value><int>3</int></value></param>
    </params>
  </methodCall>
```

Server Response

```
HTTP/1.1 200 OK
Server: Apache XML-RPC 1.0
Connection: close
Content-type: text/xml

<?xml version="1.0"?>
<methodResponse>
   <params>
      <param>
         <value><int>8</int></value>
      </param>
   </params>
</methodResponse>
```

Note

No explicit sockets
No parsing
Worker pool done for us

Protocol design - just public methods of Server object

Client program has to know

Server machine name or IP
Path to server program
Name of remote method
Number, Type and Order of arguments

Consequences

Benefits

Protocol = public methods
Handles the network communications
Handles generation/parsing of messages
Multiple language support
Platform independent
Simple

Drawbacks

Long messages
Limited support for objects

Apache XML-RPC 3 Extensions

Adds more datatypes

Java Type	Description
None	null value
Byte	8-bit value
Float	32 bit
Long	64-bit integer
org.w3c.dom.Node	DOM Node
Short	16-bit integer
java.io.Serializable	any serializable object
BigDecimal	java.math.BigDecimal
BigInteger	Arbitrary sized integers
java.util.Calendar	

Other XML-RPC systems have similar extensions

Compatibility with these is unknown

JSON-RPC

http://json-rpc.org/

Uses JSON instead of XML for data transfer

Adds convention to support arbitrary objects

JavaScript implementation allows it to be used in Web pages

Web Services

SOAP – Simple Object Access Protocol

1998 Created by Winer, Box, Atkinson, Al-Ghosein

Version 1.2 dropped the acronym

WSDL – Web Services Description Language

UUDI – Universal Description, Discovery and Integration of Web Services

UDDI

Registry for businesses worldwide to list themselves on the Internet

UDDI business registration consists of:

- White Pages — address, contact, and known identifiers;
- Yellow Pages — industrial categorizations based on standard taxonomies;
- Green Pages — technical information about services exposed by the business.

2005 - 70% of Fortune 500 companies plan to use UDDI

2006, January

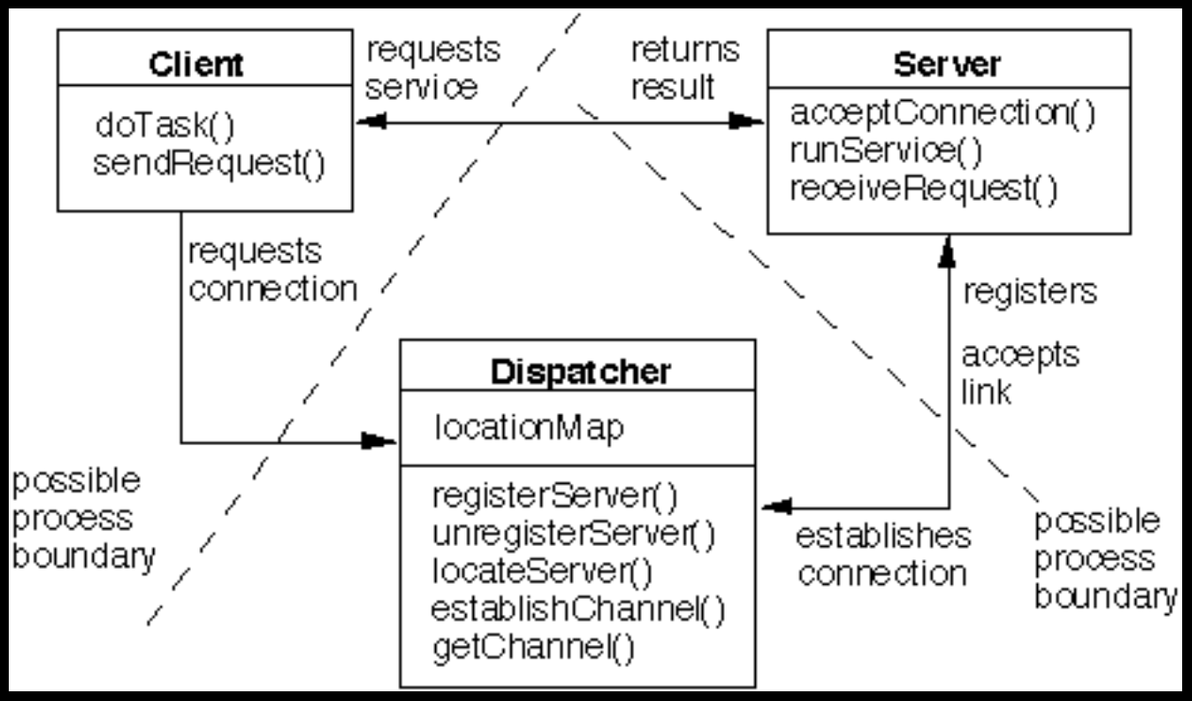
IBM, Microsoft shut down root UDDI servers

http://en.wikipedia.org/wiki/UDDI

UDDI Reborn

Plays the role of Dispatcher

Not clear how wide spread use is



WSDL

Description of how to interact with a Soap Server

Tools exist to

- Generate WSDL from a Server class
- Generate Server or client stub classes from WSDL

Sample Server

```
public class HelloServer
{
   public String hello(String aName)
      {
      return  "Hello to: " + aName;
      }
}
```

WSDL - Namespaces & Service

```
<definitions
   targetNamespace="urn:http://www.eli.sdsu.edu/cs580"
   xmlns:tns="urn:http://www.eli.sdsu.edu/cs580"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <service name="SoapServer">
     <port name="HelloWorld" binding="tns:HelloWorld">
        <documentation>A simple Soap Example</documentation>
        <soap:address location="http://localhost:4920/HelloWorld"/>
     </port>
   </service>
```

WSDL - binding

```
<binding name="HelloWorld" type="tns:HelloWorld">
   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="Hello">
      <soap:operation soapAction="urn:http://www.eli.sdsu.edu/cs580#Hello" style="rpc"/>
      <input>
         <soap:body use="literal" namespace="urn:http://www.eli.sdsu.edu/cs580"/>
      </input>
      <output>
         <soap:body use="literal" namespace="urn:http://www.eli.sdsu.edu/cs580"/>
      </output>
   </operation>
</binding>
```

WSDL - PortType

```
<portType name="HelloWorld">
  <operation name="Hello" parameterOrder="yourName">
    <documentation>Returns a greeting Input: your name</documentation>
    <input message="tns:HelloSoapIn"/>
    <output message="tns:HelloSoapOut"/>
  </operation>
</portType>
```

WSDL - Message signatures

```
<message name="HelloSoapIn">
   <part name="yourName" type="xsd:string"/>
</message>
<message name="HelloSoapOut">
   <part name="return" type="xsd:string"/>
</message>
```

SOAP

Exchanging XML messages over computer network

Message Exchange Patterns
- RPC - Remote procedure call
- Document - one way message

Transport
- HTTP, HTTPS, SMTP

Data types Supported
- All base Schema types
- Struct
- Array

Struct Example

Book as Struct

```
<e:Book>
  <author>Henry Ford</author>
  <preface>Prefatory text</preface>
  <intro>This is a book.</intro>
</e:Book>
```

Book Class

```
public class Book {
  String author;
  String preface;
  String intro;
```

Schema Defining Book

```
<element name="Book">
<complexType>
  <element name="author" type="xsd:string"/>
  <element name="preface" type="xsd:string"/>
   <element name="intro" type="xsd:string"/>
</complexType>
</e:Book>
```

Soap structs are used to send objects in messages

http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Sample Java Client

```
import org.apache.soap.rpc.*;
import org.apache.soap.Constants;

import java.util.Vector;
import java.net.URL;

class HelloSoapClient {
    public static void main(String[] args) throws Exception {
        Call call = new Call();
        call.setTargetObjectURI("urn:http://www.sdsu.edu/cs580");
        call.setMethodName("Hello");
        call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);

        //creating a parameter list
        Vector params = new Vector();
        params.addElement(new Parameter("yourName", String.class, "Roger", null));
        //adding the parameter(s) to the Call object
        call.setParams(params);

        //invoke the soap method
        Response res = call.invoke(new URL("http://localhost:4920/HelloWorld"), "");
        System.out.println(res);
    }
}
```

Sample Ruby Client

```
require 'soap/wsdlDriver'

proxy = SOAP::WSDLDriverFactory.new("http://www.eli.sdsu.edu/courses/spring07/cs580/Hello.wsdl").createDriver
puts proxy.Hello('Roger')
```

The Examples don't work

Soap is complex

Soap has problems working between venders

Some Performance

Time in seconds

	Connect time	Send String 21,000 Chars	Send 5,000 integers	Server LOC	Message size sending 100 integers
socket	0.002242	0.001377	6.71	25	85,863
Corba	0.000734	0.004601	1.52	18	27,181
XML-RPC	0.007040	0.082755	100.34	17	324,989
SOAP	0.000610	0.294198	1,324.30	10	380,288

Factor slower/larger than using Socket

	Connect time	Send String 21,000 Chars	Send 5,000 integers	Server LOC	Message size sending 100 integers
Corba	0.3	3.3	0.2	0.7	0.3
XML-RPC	3.1	60.1	15.0	0.7	3.8
SOAP	0.3	213.7	197.4	0.4	4.4

Code written in Python

http://www-128.ibm.com/developerworks/webservices/library/ws-pyth9/