CS 580 Client-Server Programming
Spring Semester, 2010
Doc 22 XML & SOAP
29 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><center>See the cat run</center></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>
```

XML Syntax

```
<!-- A simple XML document with comment -->
<greetings>
   Hello World!
</greetings>
```

Tags & Attributes

Tag

```
<GREETINGS>
<greetings>
<Greetings>
```

Case Sensitive

Tag with Attribute

```
<slide title="XML Slide" author="cat">

<slide title="Who's on First">

<name position='First'>
```

XML Terminology

Element

```
<greetings>Hello World!</greetings>
```

Nested Elements

```
<name>
    <firstName>John</firstName>
    <lastName>Fowler</lastName>
  </name>
```

Markup

Tags and comments

Content

Anything that is not markup

Document

XML structure in which one or more elements contains text intermixed with subelements

XML Document

```
<greetings>
  <from>
    <nnammee>
      <firstName>Roger</firstName>
      <lastName>Whitney</lastName>
    </nnammee>
  </from>
  <to>
    <name>
      <firstName>John</firstName>
      <lastName>Fowler</lastName>
    </name>
  </to>
  <message>
    How are you?
  </message>
</greetings>
```

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

Document Type Definition (DTD) is use to specify

- legal tags
- Nesting order of tags

Well-Formed XML

Basic Structure
Optional Prolog
Root Element

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

```
<?xml version="1.0" encoding="UTF-8"?>
<烏語>Китайська мова</烏語>
```

Prolog

Optional, but recommended

Optional attributes

version

    1.0 & 1.1

encoding

    UTF-8 (default), UTF-16, US-ASCII, etc.

standalone

Is the complete XML document in one file?

```
<?xml  version="1.0"  encoding='US-ASCII' standalone='yes' ?>

<?xml  version="1.0"  encoding='iso-8859-1' standalone="no" ?>
```

Comments

Comments can be placed nearly anywhere outside of tags

Comments can not come before <?xml version="1.0" ?>

```
<?xml version="1.0" ?>
<!-- Another comment -->
<greetings>
  <from>Roger<!-- Legal comment --></from>
  <to>John</to>
  <message>Hi</message>
</greetings>
<!-- Comments at the end -->
```

Root Element

Each XML Document has a single root element

Legal XML Document

```
<?xml version="1.0" ?>
<greetings>
  <from>Roger</from>
  <to>John</to>
  <message>Hi</message>
</greetings>
```

Illegal XML Document

```
<?xml version="1.0" ?>
<from>Roger</from>
<to>John</to>
<message>Hi</message>
```

Basic Rules for Well-Formed Documents

Non-empty elements must have start and end tags

Empty elements can use one tag

```
<greetings></greetings>

<greetings/>
```

All attribute values must be in quotes

Elements may not overlap

```
<b><center>Bad XML, but ok HTML</b></center>
```

White Space

Are the following the same?

```
<greetings>
    Hello World!
</greetings>

<greetings>Hello World!</greetings>
```

For some applications white space may be important
XML parsers are to pass white space to applications
The application decides if the white space in important

End of Line Characters

XML parsers are to convert all end of line characters to line-feed (ASCII 10)

Entities

Character	Entity
<	&lt;
>	&gt;
&	&amp;
'	&apos;
"	&quot;

Escaping

< > & have special meaning and must be escaped

```
<paragraph>
    Everyone knows that 5 < 10 & 1 > 0.
</paragraph>
```

illegal

```
<paragraph>
   Everyone knows that 5 &lt; 10 &amp; 1 > 0.
</paragraph>
```

OK

```
<paragraph><![CDATA[
    Everyone know that 5 < 10 & 1 > 0. ]]>
</paragraph>
```

OK

Defining Entities

```
<!DOCTYPE videocollection [
   <!ENTITY R "Romance">
   <!ENTITY WAR "War">
   <!ENTITY COM "Comedy">
   <!ENTITY SF "Science Fiction">
   <!ENTITY ACT "Action">
]>
```

```
<videocollection>
   <title id="1">Tootsie</title>
   <genre>&COM;</genre>
   <year>1982</year>
   <title id="2">Jurassic Park</title>
   <genre>&SF;</genre>
   <year>1993</year>

   <title id="3">Mission Impossible</title>
   <genre>&ACT;</genre>
   <year>1996</year>
</videocollection>
```

Example from http://www.javacommerce.com/displaypage.jsp?name=entities.sql&id=18238

Unicode Characters

中	decimal code point 20,013 hexadecimal code point 4E2D

In XML can represent 中 as either

```
&#20013;
&#x4e2d;
```

Processing Instruction

```
<?xml version="1.0" encoding="UTF-8"?>
<RootElement param="value">
    <FirstElement>
        Some Text
    </FirstElement>
    <?some_pi some_attr="some_value"?>
    <SecondElement param2="something">
        Pre-Text <Inline>Inlined text</Inline> Post-text.
    </SecondElement>
</RootElement>
```

Valid XML Documents

XML document that is well-formed and

Specifies which tags are legal

Document Type Definition (DTD) is use to specify

- legal tags
- Nesting order of tags

```
<?xml version="1.0" ?>
<!DOCTYPE greetings [
  <!ELEMENT greetings (#PCDATA)>]>
<greetings>
  Hello World!
</greetings>
```

Sample XML Document w DTD

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>Njandu Varuthude</name>
    <birthdate>04/02/1977</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

example.dtd

```
<!ELEMENT people_list (person*)>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)>
```

http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd

```
<!--================ Character mnemonic entities =========================-->

<!ENTITY % HTMLlat1 PUBLIC
   "-//W3C//ENTITIES Latin 1 for XHTML//EN"
   "xhtml-lat1.ent">
%HTMLlat1;

<!ENTITY % HTMLsymbol PUBLIC
   "-//W3C//ENTITIES Symbols for XHTML//EN"
   "xhtml-symbol.ent">
%HTMLsymbol;

<!ENTITY % HTMLspecial PUBLIC
   "-//W3C//ENTITIES Special for XHTML//EN"
   "xhtml-special.ent">
%HTMLspecial;

<!--================== Imported Names ====================================-->

<!ENTITY % ContentType "CDATA">
    <!-- media type, as per [RFC2045] -->

<!ENTITY % ContentTypes "CDATA">
    <!-- comma-separated list of media types, as per [RFC2045] -->

<!ENTITY % Charset "CDATA">
    <!-- a character encoding, as per [RFC2045] -->

<!ENTITY % Charsets "CDATA">
    <!-- a space separated list of character encodings, as per [RFC2045] -->
```

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

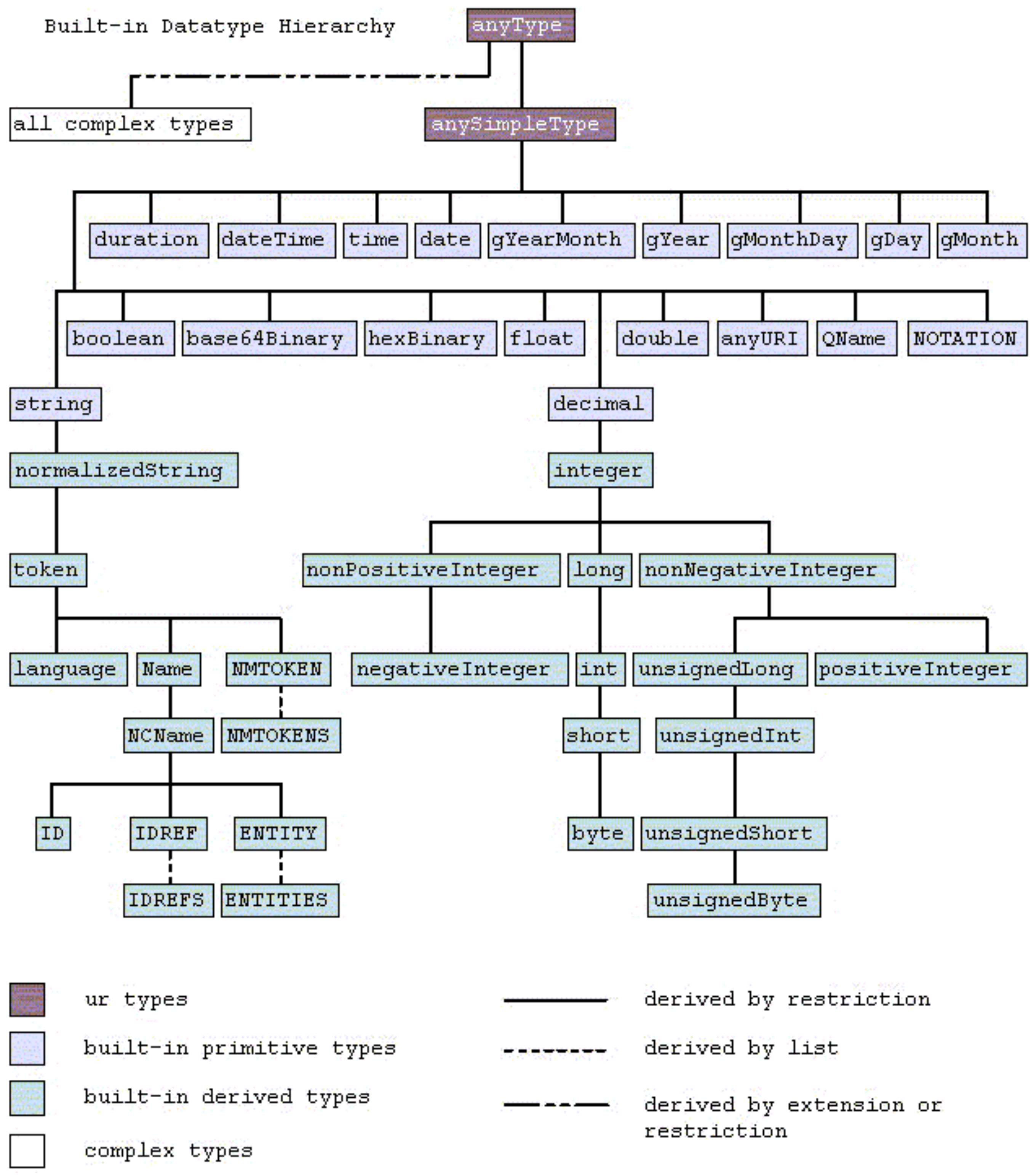
Sample Schema parts

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

<datatype name="MovieTicketPrice">
	<scalar datatype="Price" digits="1"
		maxvalue="19.50" 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/

XML Schema Languages

Document Type Definitions
W3C XML Schema
RELAX NG (REgular LAnguage for XML Next Generation)
Schematron

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

RELAX NG

Presentation and Transformation

XML Stylesheet Language (XSL)

XSL Transformation Language (XSLT)

Cascading Style Sheets (CSS)

Extensible Stylesheet Language for Formatting Objects (XSL-FO)

Parsing XML

Apache Xerces project http://xerces.apache.org/xerces2-j/
Java JDK reference parser

Java for XML Processing (JAXP)

- Simple API for XML (SAX)
- Document Object Model (DOM)
- XML Stylesheet Language for Transformations (XSLT)

Simple API for XML (SAX)

Implement a Handler
Handler knows about parts of XML
Give XMLParser your handler
Parser tells handler when it finds each part of XML document

```
characters(char[] ch, int start, int length)
endDocument()
endElement(String uri, String localName, String qName)
ignorableWhitespace(char[] ch, int start, int length)
skippedEntity(String name)
startDocument()
startElement(String uri, String localName, String qName, Attributes attributes)
```

SAXExample

```
import java.io.FileReader;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;
import com.sun.xml.internal.ws.streaming.Attributes;

public class SAXExample extends DefaultHandler {

	public void characters(char[] ch, int start, int length) {
		System.out.println("char: " + new String(ch, start, length));
	}

	public void startDocument() { System.out.println("Started document"); }

	public void startElement(String uri, String localName, String qName,
			Attributes attributes) {
		System.out.println("uri: " + uri + " local name: " + localName
				+ " qName: " + qName + " attributes " + attributes);
	}

	public void endDocument() { System.out.println("End document"); }
```

SAXExample

```
public static void main(String args[]) throws Exception {
		XMLReader xmlParser = XMLReaderFactory.createXMLReader();
		SAXExample handler = new SAXExample();
		xmlParser.setContentHandler(handler);
		xmlParser.setErrorHandler(handler);

		FileReader sampleXML = new FileReader("example.xml");
		xmlParser.parse(new InputSource(sampleXML));

		System.out.println("done");
	}
```

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

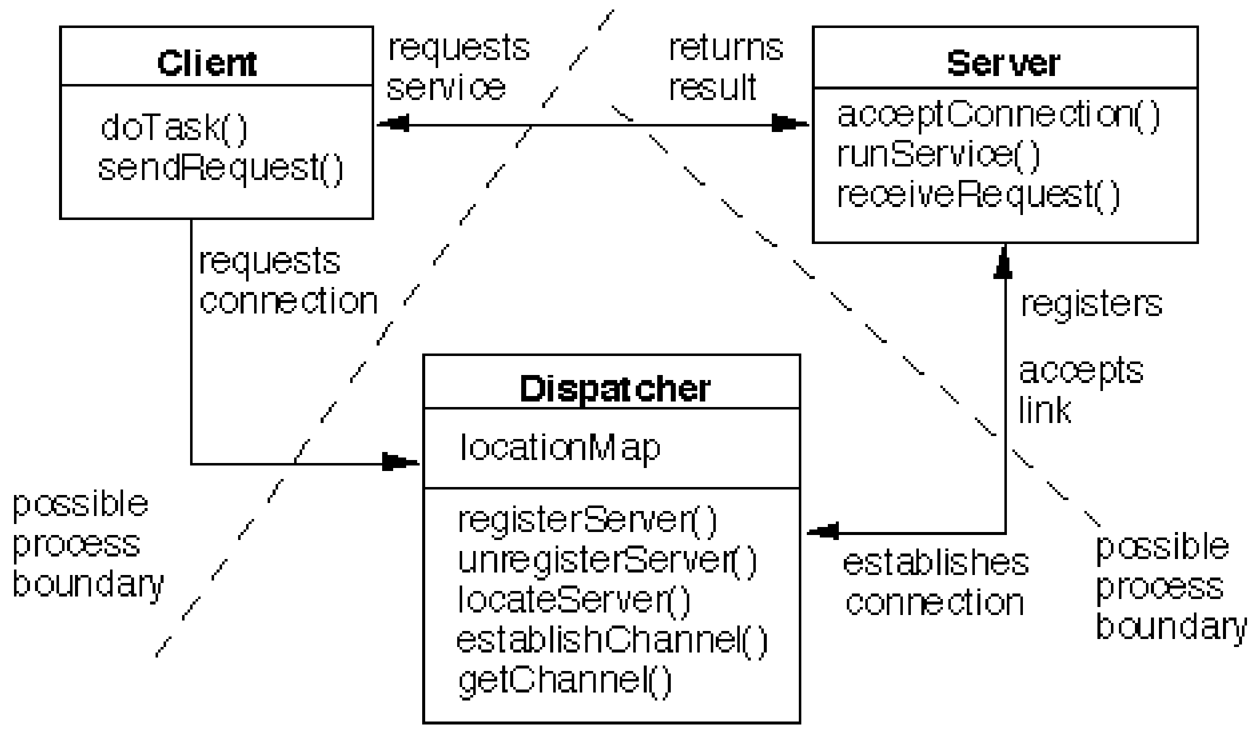
IBM, Microsoft shut down root UDDI servers

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

```java
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/