XML Technologies and Applications

Rajshekhar Sunderraman

Department of Computer Science
Georgia State University
Atlanta, GA 30302
raj@cs.gsu.edu

VI. XML Applications

December 2005
Outline

- Introduction
- XML Basics
- XML Structural Constraint Specification
  - Document Type Definitions (DTDs)
  - XML Schema
- XML/Database Mappings
- XML Parsing APIs
  - Simple API for XML (SAX)
  - Document Object Model (DOM)
- XML Querying and Transformation
  - XPath
  - XSLT
  - XQuery
- XML Applications
Applications

• Countless!

• XML in Web Services (will cover this)
  – WSDL (Web Services Description Language)
  – SOAP (Simple Object Access Protocol) as the Message format

• XML in Bioinformatics
  – Markup Languages – BioML, NeuroML, ChemML
  – Canonical representations

• Configuration Files
  – Apache Tomcat
Web Services

- Web Services are “functions” that can be invoked over the Web via client programs.

- Powerful paradigm in B2B environment; Services are provided by programs from different businesses that send messages to each other with little or no human involvement.

- Businesses can
  - Provide their services to an expanding world market
  - Outsource non-essential functions to other businesses
  - Obtain services they need from other providers
  - Establish partnerships with other businesses to provide combined services that are better and less expensive than one business can provide
Technologies in Web Services

• **XML** (Extensible Markup Language)
  - Used to describe Web services
  - Messages exchanged between services are XML documents. For example:
    • When a C++ program invokes a Web service the arguments are converted into an XML document and sent
    • When the XML document reaches the services provided, it is converted into a call the a Java method call
  - Solves the interoperability problem

• **SOAP** (Simple Object Access Protocol)
  - Messages used by Web services are generally sent using the SOAP protocol
Technologies (Con’t)

• **WSDL** (Web Service Description Language)
  - Describes the operations required to obtain a Web service
    • Messages exchanged in each operation
    • Data contained in each message
  - Describes how the operation is bound to a specific message transport protocol

• **UDDI** (Universal Description, Discovery, and Integration)
  - Provides a registry (database) in which
    • Businesses than want to provide a service can publish information about it, including a description of how it can be invoked (perhaps described in WSDL)
    • Businesses that want to obtain a service can find information about servers that provide it and how they can be invoked
SOAP: Overview

• A lightweight protocol for exchanging information in a distributed, heterogeneous environment
  – It enables cross-platform interoperability

• Interoperable
  – OS, object model, programming language neutral
  – Hardware independent
  – Protocol independent

• Works over existing Internet infrastructure

• SOAP Messages – Requests and Responses
SOAP: The HTTP Aspect

SOAP Requests are HTTP POST requests

```
POST /WebCalculator/Calculator.asmx HTTP/1.1
Content-Type: text/xml
SOAPAction: "http://tempuri.org/Add"
Content-Length: 386

<?xml version="1.0"?>
<soap:Envelope ...>
  ...
</soap:Envelope>
```
SOAP: Message Structure

- **SOAP Message**
  - **Headers**
  - **SOAP Envelope**
    - **SOAP Header**
      - **Headers**
    - **SOAP Body**
      - **Message Name & Data**

- The complete SOAP message
- Protocol binding headers
- `<Envelope>` encloses payload
- `<Header>` encloses headers
- Individual headers
- `<Body>` contains SOAP message name
- XML-encoded SOAP message name & data
SOAP Message Format

An XML document using the SOAP schema:

```xml
<?xml version="1.0"?>
<soap:Envelope ...>
    <soap:Header ...>
        ...
    </soap:Header>
    <soap:Body>
        <Add xmlns="http://tempuri.org/">
            <n1>12</n1>
            <n2>10</n2>
        </Add>
    </soap:Body>
</soap:Envelope>
```
Server Responses

Server replies with a “result” message:

HTTP/1.1 200 OK
...
Content-Type: text/xml
Content-Length: 391

<?xml version="1.0"?><soap:Envelope ...>
  <soap:Body>
    <AddResult xmlns="http://tempuri.org/">
      <result>28.6</result>
    </AddResult>
  </soap:Body>
</soap:Envelope>
Encoding Complex Data

Data structures are serialized as XML:

```xml
<soap:Envelope ...>
  <soap:Body>
    <GetStockDataResult xmlns="http://tempuri.org/">
      <result>
        <Description>Plastic Novelties Ltd</Description>
        <Price>129</Price>
        <Ticker>PLAS</Ticker>
      </result>
    </GetStockDataResult>
  </soap:Body>
</soap:Envelope>
```
Example of a SOAP Request

```xml
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
SOAPAction: "Some-URI"

  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Example of a SOAP Response

HTTP/1.1 200 OK
Content-Type: text/xml;
charset=utf-8
Content-Length: nnnn

<SOAP-ENV:Envelope
    xmlns:SOAP-ENV= "http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:GetLastTradePriceResponse xmlns:m="Some-URI">
            <Price>34.5</Price>
        </m:GetLastTradePriceResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
Example of a SOAP Error

HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
   <SOAP-ENV:Fault>
     <faultcode> SOAP-ENV: MustUnderstand </faultcode>
     <faultstring>SOAP Must Understand Error </faultstring>
   </SOAP-ENV:Fault>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
WSDL: Web Services Description Language

- XML schema for describing Web Services
  1. Service interface definition
     - Abstract semantics for Web Service
  2. Service implementation definition
     - Concrete end points and network addresses where Web Service can be invoked
- Clear delineation between abstract and concrete messages
WSDL Schema

- `<definitions>` are root node of WSDL
- `<import>` allows other entities for inclusion
- `<types>` are data definitions - xsd
- `<message>` defines parameters of a Web Service function
- `<portType>` defines input and output operations
- `<binding>` specifies how each message is sent over the wire
- `<service>` specifies details about the implementation
- `<port>` contains the address itself
WSDL Abstract Level

- At the abstract level, obtaining a service is like executing a method of an object

- WSDL defines the following elements
  - An **portType** is like an object; it consists of a set of operations
  - An **operation** is like a method; it is invoked by messages
  - A **message** is composed of parts
  - A **part** is like a parameter and has an associated **type**
Example

<portType name = “GetQuotePT”>
   <operation name = “getQuoteOp”>
      <input message = “gs:getQuoteOpReq”/>
      <output message = “gs:getQuoteOpResp”/>
      <fault name = “invalidSymbolFault”
         message = “gs:invalidSymbolFaultMsg”/>
   </operation>
   <!-- other operations go here -->
</portType>

gs is the target namespace of the document containing this declaration and the message declarations
Patterns

• The messages exchanged when an operation is invoked conform to a pattern

• WSDL 1.1 has defined two patterns:
  – Request/response
    • Input sent by requestor, output produced by service
    • Requestor might wait for response (e.g., RPC) or might not
  – One-way
    • Input sent by requestor, no response expected
Faults

<output message = "gs:getQuoteOpResp"/>

<fault name = "invalidSymbolFault"
    message = "gs:invalidSymbolFaultMsg"/>

• Request/response pattern allows a fault message to replace the output message if server detects a fault

• One-way pattern does not allow fault message
<message name="getQuoteOpReq">
    <part name="stockSymbol" type="xsd:string"/>
</message>

<message name="getQuoteOpResp">
    <part name="stockSymbol" type="xsd:string"/>
    <part name="QuoteValue" type="xsd:float"/>
</message>

<message name="invalidSymbolFaultMsg">
    <part name="faultInfo" type="gs:faultType"/>
</message>
Parts of a Message

- A message can have many parts
  - Each part can be bound to a different position within the physical message sent by the transport
    - With SOAP parts can be distributed over body and header blocks
- Each part can have a simple or complex type defined in an XML schema