Introduction to Web Services

Concurrent & Distributed Software Systems

Motivation

- Today’s Web
  - Designed for human-application interactions
    - Browser front-end
  - Does not support application-application interaction on the web

- Web services
  - Enables applications to expose their services “programmatically”, i.e. the services can be invoked by programs
  - Enables software running on other computers (could be a desktop, mobile phone, PDA, etc.) to invoke operations exposed by Web applications
**Scenarios for Using Web Services**

- Allowing programmatic access to applications accessed over the Internet
- B2B integration – allowing applications from different organizations to communicate across the Internet
- A2A integration – allowing applications within an organization to communicate across an intranet

Why can’t we use CORBA/DCOM/Java RMI for this?
All important vendors – Microsoft, IBM, Sun, HP, etc. support web services

---

**Web Services Technology**

Four fundamental technologies
- **XML**: Describing information sent over the network
- **WSDL**: Defining Web service capability
- **SOAP**: Accessing Web services
- **UDDI**: Finding web services
XML

- Has emerged as the standard solution for describing information exchanged between heterogeneous systems
- A XML document contains one or more elements, each of which is demarcated using tags
- Example
  - `<Account> 729-1269-4785 </Account>`
  - `<Account type="checking">729-1269-4785</Account>`

XML cont’d

- How are elements defined?
  - Document Type Definitions (DTD)
  - XML Schema Definition (XSD) language
    Used to define rules on how the elements in a valid XML document can be used

- Namespaces
  - Allow the mixing of tags from different schemas in the same document
  - `<Account xmlns=http://www.quikbank.com/bank> 729-1269-4785 </Account>`
Describing Web Services: WSDL

- Provides functional description of network services:
  - IDL description
  - Protocol and deployment details
  - Platform independent description.
  - Extensible language.
- A short history:
  - WSDL v1.0, 9/2000
  - A de facto industry standard.

WSDL Structure

- portType
  - Abstract definition of a service (set of operations)
- Multiple bindings per portType:
  - How to access it
  - SOAP, JMS, direct call
- Ports
  - Where to access it
WSDL Example (simplified)

```xml
<definitions name="AccountAccess">
  <types>
    <element name="BalanceRequest">
      <!-- definition of input type, e.g. Account -->
    </element>
    <element name="BalanceResult">
      <!-- definition of output type, e.g. Balance -->
    </element>
  </types>
  <message name="GetBalanceInput">
    <part name="body" element="BalanceRequest"/>
  </message>
  <message name="GetBalanceOutput">
    <part name="body" element="BalanceResult"/>
  </message>
</definitions>
```

WSDL Example (cont'd)

```xml
<portType name="AccountAccessPortType">
  <operation name="GetBalance">
    <input message="GetBalanceInput"/>
    <output message="GetBalanceOutput"/>
  </operation>
</portType>

<binding name="AccountAccessSoapBinding" type="AccountAccessPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetBalance">
    <!-- definitions for input and output operations appear here -->
  </operation>
</binding>

<service name="AccountAccessService">
```

```xml
```
**WSDL Example (cont'd)**

```xml
<service name="AccountAccessService">
  <port name="AccountAccessPort"
       binding="AccountAccessSoapBinding">
    <soap:address
       location="http://www.quikbank.com/accounts.asmx"/>
  </port>
</service>
</definitions>
```

**Using WSDL**

1. **As extended IDL:** WSDL allows tools to generate compatible client and server stubs.
2. **Allows industries to define standardized service interfaces.**
3. **Allows advertisement of service descriptions,** enables dynamic discovery and binding of compatible services.
   - Used in conjunction with UDDI registry
4. **Provides a normalized description of heterogeneous applications.**
Accessing Web Services: SOAP

- **Why SOAP?**
  - A 'Wire Protocol' necessary for accessing distributed object services
  - Vendor and/or platform-specific nature of current wire protocols hinder interoperability

- **SOAP**
  - An Internet standard specification, the goal of which is to define a platform and vendor-neutral wire protocol based on Internet standard protocols [HTTP & XML] to access Web Services

- **Features:**
  - Uses XML to package requests for services exposed by Web Services, and responses generated by Web Services
  - Typically uses HTTP as a transport protocol

SOAP cont’d

- **What SOAP is NOT?**
  - SOAP is not a replacement for any of the current component technologies
  - It does not specify how key infrastructural elements of a Distributed Object Infrastructure need to be implemented [eg: Distributed garbage collection, Object activation etc.]
Accessing a Web Service Using SOAP

1 – Obtain/Examine SDL and Generate SOAP REQUEST

2 – Transmit REQUEST Payload [ HTTP POST ]

3 – Receive Request Payload on HTTP Port 80

4 – Parse Request, Process Request, and Generate SOAP RESPONSE

6 – Receive and Parse RESPONSE

5 – Transmit RESPONSE Payload

The SOAP Request Payload

- A standard XML-based format to describe a SOAP request for a Web Service
- Provides all the information required by the Web Service provider to process the request
- General format of a SOAP request:

HTTP Header
SOAP Action

<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <!-- Soap Header is optional -->
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <!-- Serialized method invocation data -->
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
The SOAP Response Payload

- A standard XML-based format to describe the Response generated by a Web Service
- Contains information that is to be passed back to the client
- General format of a SOAP response:
  
  HTTP Header
  
  `<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <!-- Serialized Response Data -->
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>`

Advantages of SOAP

- Lightweight. Requires two fundamental capabilities:
  - Capability to send and receive HTTP packets
  - Capability to process XML [Requires an XML parser ]
- Built on open technologies
- Facilitates true distributed interoperability
- No firewall restrictions
Disadvantages of SOAP

- Based on HTTP, and is therefore
  - Stateless
  - Based on a request/response architecture - implies no callback functionality
- Performance is slightly degraded for the following reasons:
  - An XML processor needs to be loaded each time to parse a SOAP request/response
  - The SOAP request/response has to be parsed to extract the required information
- Currently supports only parameter serialization by value

Finding Web Services: UDDI

- UDDI defines the operation of a service registry:
  - Data structures for registering
    - Businesses
    - Technical specifications: tModel is a keyed reference to a technical specification.
    - Service and service endpoints: referencing the supported tModels
  - Standard APIs for access (Inquiry API) and writing to the registry (Publishing API)
UDDI Relationships

Web Service

businessEntity

businessService

bindingTemplate

InstanceDetails

categoryBag

keyedReference

identifierBag

keyedReference

tModels