Intro to Web Services

CS 475

Motivation

- World wide web
  - Designed for human-application interactions
  - No support for application-application interactions

- 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
  - Built on top of underlying protocols and mechanisms for web (e.g., HTTP)
Scenarios for 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/.NET/Java RMI for this?
  - Interoperability: All important vendors - Microsoft, IBM, Sun/Oracle, HP, etc. support web services
  - Firewall traversal

Example

The ‘travel agent service’ combines other web services
Web Services Technology

Two competing approaches
REST-style vs SOAP-based

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

RESTful approach focuses on using HTTP operations (GET, PUT, POST, DELETE) to manipulate data resources represented in XML
- No WSDL + SOAP
- More about this later

Web Services Infrastructure and Components

<table>
<thead>
<tr>
<th>Applications</th>
<th>Security</th>
<th>Directory service</th>
<th>Choreography</th>
<th>Service descriptions (in WSDL)</th>
<th>SOAP</th>
<th>URIs (URLs or URNs)</th>
<th>XML</th>
<th>HTTP, SMTP or other transport</th>
</tr>
</thead>
</table>

Web Services Overview
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>`
- Markup language like HTML but not simply for displaying pages.
  - Can be read by programs and interpreted in an application-specific way.

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>`
**WSDL: Describing web services**

- Provides functional description of network services
  - IDL description
  - Protocol and deployment details
  - Platform independent description.
  - Extensible language.

**The main elements in a WSDL description**

- definitions
  - types
  - message
  - interface
  - bindings
  - services

- target namespace
- document style
- abstract
- request-reply style
- how
- concrete
- where
**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>
```
WSDL Example cont’d

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

Using WSDL

- As extended IDL: WSDL allows tools to generate compatible client and server stubs.
- Allows industries to define standardized service interfaces
- Allows advertisement of service descriptions, enables dynamic discovery and binding of compatible services
  - Used in conjunction with UDDI registry
Accessing web services: SOAP

- Why SOAP?
  - A 'Wire Protocol' necessary for accessing distributed object services
  - Vendor and/or platform-specific 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 messages

- SOAP message carried in an "envelope"
  - Optional header and a body inside envelope

- SOAP messages can be used to
  - Convey documents (placed inside the body element)
  - Support client-server communication (Request/Reply messages)

- SOAP messages typically use HTTP as the transport protocol
SOAP message in an envelope

![SOAP message diagram](image)

**Web Services Overview**

### 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:**
  
  ```xml
  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>
  ```

**Web Services Overview**
Example of a simple request without headers

```
<env:envelope xmlns:env="namespace URI for SOAP envelopes">
  <env:body>
    <m:exchange xmlns:m="namespace URI of the service description">
      <m:arg1>Hello</m:arg1>
      <m:arg2>World</m:arg2>
    </m:exchange>
  </env:body>
</env:envelope>
```

Example of reply

```
<env:envelope xmlns:env="namespace URI for SOAP envelope">
  <env:body>
    <m:exchangeResponse xmlns:m="namespace URI for the service description">
      <m:res1>World</m:res1>
      <m:res2>Hello</m:res2>
    </m:exchangeResponse>
  </env:body>
</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:
  
  ```html
  HTTP Header
  
  <SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <!--Serialized Response Data -->
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
  ```

Use of HTTP POST Request in SOAP client-server communication

POST `/examples/stringer` endpoint address

- `Host: www.cdk4.net`
- `Content-Type: application/soap+xml`
- `Action: http://www.cdk4.net/examples/stringer#exchange` action

```xml
<env:envelope xmlns:env=namespace URI for SOAP envelope>
  <env:header></env:header>
  <env:body></env:body>
</env:Envelope>
```
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)

The main UDDI data structures
REST

- REST - Representational State Transfer protocol (Fielding)
- Design philosophy
  - Everything on the web is a resource with a URI
  - HTTP is not just a transport protocol
    - It provides an API (POST, GET, PUT, DELETE) for Create, Read, Update, and Delete operations on a resource
  - Approach isolates application complexity at the end points (client and server) and keeps it out of the transport
- Web services developers divided into REST and SOAP camps
  - Vendors (e.g. Microsoft) seem to prefer SOAP but REST seems to be gaining among developers

Remaining classes

- Next class
  - Implementing Simple Web services in Java
- Next week
  - Peer-to-peer computing
  - Cloud computing