Web Services

Moving towards Service Oriented Architectures
Agenda

• Service Oriented Architectures (SOA)
• Web Services
  – Simple Object Access Protocol (SOAP)
  – Web Services Description Language (WSDL)
  – Universal Description, Discovery, and Integration (UDDI)
  – Tools
  – Examples
• Grid Services
Service Oriented Architecture (SOA)

• SOAD vs. OOAD
  - New
    • Services
    • Flows
    • Components
• Services Layer
  - Placed on top of existing products
• Roles
  - Service provider
  - Service consumer
Service Oriented Architecture (SOA)

- Architectural style that defines an interaction model between 3 primary parties
  - Service Provider
    - Publishes a service description and provides the implementation for the service
  - Service Consumer
    - Uses either the uniform resource identifier (URI) for the service description directly or can find the service description in a service registry
  - Service Broker
    - Provides and maintains the service registry, although nowadays public registries are not in vogue.
Service Oriented Architecture (SOA)

- When To Use SOA
  - Heterogeneous Environment Present
  - Real-time Performance is not critical
  - Dynamic Information
  - Loose Coupling
Service Oriented Architecture (SOA)

• SOA Design Strategy
  – “Top Down”
  – Strategic and Business Aligned
  – Stems from Business Process Choreography
• Web Services
  – “Bottom Up”
  – Tactical Implementation of SOA
  – Stems from existing components
Services Layer

- **Service Types**
  - *Stateless Service*
    - Message exchanges, no information transferred from persistent resources
  - *Conversational Service*
    - Implements a series of operations
  - *Stateful Service*
    - Provides access to, or manipulates a set of logical stateful resources (documents or data sets)
Service Layer Implementation

- Web Services System
  - Identified by a URI
  - XML defined interfaces and bindings
  - Discoverable by other software system
  - Other systems interact with the Web service based on its definition
  - XML based messages between systems
Web Services

1. WS
2. RE
3. WS-Interface
4. Message Processing
5. Requests
6. Message Dispatch
7. Stateful Resource

Figure 1: Facets related to a Web service
Web Services

- Protocol Stack
  - Similar to TCP/IP

Diagram:
1. Service advertises itself in directory service.
2. Client looks up service details in directory service.
3. Client interacts with service.

Stack:
- DISCOVER (UDDI)
- DESCRIBE (WSDL)
- ACCESS (SOAP)
- STRUCTURE (XML)
- TRANSPORT (HTTP)
Simple Object Access Protocol (SOAP)

- XML message protocol to access web services

Diagram:
- HTTP Header
- SOAP Envelope
  - SOAP Header
    - Header
  - SOAP Body
    - Application Specific Message
    - Data
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation" env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff8fe8f7d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees" env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
      <n:name>Åke Jóguv Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2001-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2001-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference/>
      </p:return>
    </p:itinerary>
    <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>
Describing Web Services

• **WSDL**
  – Web Service Definition Language

• **Defines**
  – Web service can do
  – Where it resides
  – How to invoke it
WSDL Types

- **Service Interface**
  - Contains *Types*, *import*, *message*, *portType*, and *binding* elements.
  - A service interface contains the WSDL service definition that will be used to implement one or more services. It is an abstract definition of a web service, and is used to describe a specific type of service.

- **Service Implementation**
  - Contains *import* and *service* elements.
WSDL Skeleton

<definitions>
  <types> definition of data types........ </types>
  <message> definition of a message.... </message>
  <portType> definition of a port {1 way | req/resp | illicit resp | notification}....... </portType>
  <binding> definition of binding to SOAP.... </binding>
</definitions>

[5]
WSDL Example

```xml
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
(binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation soapAction="http://example.com/getTerm"/>
    <input> <soap:body use="literal"/> </input>
    <output> <soap:body use="literal"/> </output>
  </operation>
</binding>
```
UDDI

• **Universal Description, Discovery, and Integration**
  - Registry of web services

• **Phone book for web services**[6]
  - White pages
    - Basic contact information and identifiers.
  - Yellow pages
    - Information that describes a web service
  - Green pages
    - Information that describes behaviors and functions of a web service

• **WS-Inspection**
  - *WS-Inspection* is a new XML grammar that allows a Web server to be inspected for services that may be deployed.
UDDI Activities

• Using UDDI [7]
  – Establishing Connection
  – Publishing a Business Entity
  – Classifying a Business Entity
  – Publishing a Model
  – Publishing a Service
  – Deleting from a Registry
UDDI Data Types and Mapping

UDDI DataTypes

WSDL to UDDI Mapping
WSDL to UDDI Mapping

WSDL Service Implementation

```xml
<definitions name="StockQuoteService"
targetNamespace="http://...">
  <import namespace="http://..."
    location="http://..."/>
  <service name="StockQuoteService">
    <port name="SingleSymbolService"
      binding="iface:SingleSymbolBinding">
      ...
    </port>
  </service>
</definitions>
```

WSDL Service Interface

```xml
<definitions
  name="StockQuoteService-interface"
targetNamespace="http://...">
  <message name="SingleQuoteRequest">
    ...
  </message>
  <portType name="SingleSymbolService">
    ...
  </portType>
  <binding name="SingleSymbolBinding"
    type="tns:SingleSymbolService">
    ...
  </binding>
</definitions>
```

UDDI Registry

```xml
<businessEntity businessKey="...">
  <name>Stock Quote Service, Inc.</name>
  ...
  <businessService serviceKey="..."
    name="StockQuoteService">
    <bindingTemplates>
      <bindingTemplate bindingKey="...">
        <tModelInstanceInfo tModelKey="...">
          ...
          <overviewDoc>
            <overviewURL>
              http://...#SingleSymbolService
            </overviewURL>
          </overviewDoc>
          ...
        </tModelInstanceInfo>
      </bindingTemplate>
    </bindingTemplates>
    <businessService>
      <businessEntity>
      </businessEntity>
    </businessService>
  </businessService>
</businessEntity>
```

```xml
<tModel tModelKey="...">
  <name>http://.../</name>
  <overviewDoc>
    <overviewURL>
      http://...#SingleSymbolBinding
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="..."
      keyName="udi-org:types"
      keyValue="wsdlSpec"/>
  </categoryBag>
</tModel>
```
Tools Used in Examples

• Apache Software Foundation
  – Xerces
    • XML Schema processor
  – SOAP Implementation (Axis Project is related)
    • Library to enable invocation of SOAP services
  – Tomcat
    • Servlet Container which is the official Reference Implementation for the Java Servlet and JavaServer Pages technologies
Servlets and Containers

- **Servlet**
  - A Java program that runs within a JVM on a web server and sends HTML data to a web browser
  - Implements `javax.servlet.Servlet`

- **Servlet Container**
  - Web Server Component that interacts with Servlets
  - Responsible for forwarding the requests and responses to the appropriate Servlet
  - Maps URL’s to Servlets
Servlet Containers

• **How Java Web Servers Work**
  - HTTPServer Class

• **How Servlet Containers Work**
  - Servlet is called for the first time
    - Load the servlet class
    - Call its init method (once only).
  - For each request
    - Construct an instance of `javax.servlet.ServletRequest`
    - Construct an instance of `javax.servlet.ServletResponse`.
  - Invoke the servlet's service method
    - Pass the ServletRequest and ServletResponse objects to the servlet
  - When the servlet class is shut down
    - Call the servlet's destroy method and unload the servlet class.

• “How Tomcat Works”
  - Entire book on the internal workings of Tomcat
Web Service Examples

- Automobile Finder Web Service
- WSDL Generation and Stub Generation
- Semantic Web Service Finder
- UDDI Publishing
Grid Services

• All Grid Services are Web Services
  – Vice versus not true
• “A web service that conforms to a set of conventions (interfaces and behaviors) that define how a client interacts with it”
Grid Services

- **OGSI Describe Grid Services**
  - Open Grid Services Infrastructure
  - Extension upon WSDL
  - Base for Open Grid Services Architecture (OGSA)
OGSI

• Defines Grid Services (since 2002)
  – WSDL 1.1 extensions (to be in WSDL 2.0)
  – Discovery (Grid Handles and References)
  – Dynamic Service Creation
  – Lifetime Management
  – Notification
  – Manageability
  – Fault Conventions
Existing Grid Services (1/2)

- **GILDA**

<table>
<thead>
<tr>
<th>SERVICE</th>
<th>HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Broker (RB)</td>
<td>grid004.ct.infn.it</td>
</tr>
<tr>
<td>Resource Broker for DAG (RB)</td>
<td>grid007.ct.infn.it</td>
</tr>
<tr>
<td>Information Index (BDII)</td>
<td>grid017.ct.infn.it</td>
</tr>
<tr>
<td>Backup BDII</td>
<td>grid018.ct.infn.it</td>
</tr>
<tr>
<td>LDAP (for GILDA VOs)</td>
<td>grid-vo.cnaf.infn.it:10389</td>
</tr>
<tr>
<td>GridICE</td>
<td>alifarm7.ct.infn.it:50080</td>
</tr>
<tr>
<td>Replica Location Service (RLS)</td>
<td>grid008.ct.infn.it</td>
</tr>
</tbody>
</table>
• GILDA raster image
Reference

• [1] – Semantic Web Book
• [2] – IBM PPT
• [3] - monty