Introduction to Web Services
[Part 2] Web Services Essentials

Instructor: Prof. Young-guk Ha
Dept. of Computer Science & Engineering
Overview

- **Part 1: Backgrounds**
  - World Wide Web
  - Introduction to Web Services
  - Web Services Technology Trend

- **Part 2: Web Services Essentials**
  - Web Services Architecture
  - Core Technologies (WSDL, SOAP, UDDI)

- **Part 3: RESTful Web Services**
  - Introduction to REST
  - RESTful Web Services Architecture
  - RESTful Web Services Implementation
Web Services Architecture

“An instance of SOA implemented with Web & XML technologies”
Core Technologies for Web Services

- XML (eXtensible Markup Language)
- WSDL (Web Services Description Language)
- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery, and Integration)
WSDL (Web Services Description Language)

- Provides the model and XML format to describe Web Services (how to invoke the Web Services)
  - Data types
  - Input and output messages
  - URL of the service implementation
  - Message exchange protocol
  - ...

- WSDL Documents
  - Published to the UDDI registry

- W3C recommendation
  - [http://www.w3.org/standards/techs/wSDL](http://www.w3.org/standards/techs/wSDL)
WSDL Documents Structural Overview

- **definitions**
  - `import (*)`
  - `include (*)`

- **types (?)**

- **message (*)**
  - `part (*)`

- **interface (*)**
  - `operation (*)`

- **service (*)**
  - `endpoint (*)`

- **binding (*)**
  - `soap:binding (*)`
  - `soap:operation (*)`

- : Abstract descriptions
- : Concrete descriptions (implementation dependent)

> 추상정보와 구체정보를 분리함으로써 서로 다른 서비스의 기술에 대해 추상적인 정의를 재사용할 수 있음
Terminology

- **Message**
  - Basic unit of communication between the service provider and requester (inputs and outputs for operations)

- **Operation**
  - Exchange of messages between the service provider and requester

- **Interface**
  - A collection of operations

- **Binding**
  - Mapping from an interface to a concrete message exchange protocol and message encoding

- **Endpoint**
  - Physical location (URL) of an operation

- **Service**
  - A collection of endpoints bound to the same interface
Terminology (cont'd)
WSDL Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="TicketAgent"
    targetNamespace="http://airline.wsdl/ticketagent/"
    xmlns="http://www.w3.org/ns/wsdl"
    xmlns:tns="http://airline.wsdl/ticketagent/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd="http://airline/">

<message name="listFlightsRequest">
    <part name="depart" type="xs:datetime"/>
    <part name="origin" type="xs:string"/>
    <part name="destination" type="xs:string"/>
</message>

<message name="listFlightsResponse">
    <part name="result" type="xsd:ArrayOfString"/>
</message>

<!- interfaces, services, and bindings definitions -->

</definitions>
```

* WSDL Namespace URI = http://www.w3.org/ns/wsdl
WSDL Example (cont'd)

```xml
<interface name="TicketAgent"
  <operation name="listFlights">
    <input name="listFlightsRequest" message="tns:listFlightsRequest"/>
    <output name="listFlightsResponse" message="tns:listFlightsResponse"/>
  </operation>
</interface>

<service name="TicketAgentService" interface="tns:TicketAgent">
  <endpoint name="EndPoint" binding="tns:TicketAgentSOAPBinding">
    <soap:address location="http://localhost:8080/soap/servlet/rpcrouter"/>
  </endpoint>
</service>

<binding name="TicketAgentSOAPBinding" interface="tns:TicketAgent">
  <soap:binding protocol="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="listFlights">
    <soap:operation style="rpc"/>
    <input name="listFlightsRequest" message="tns:listFlightsRequest">
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output name="listFlightsResponse" message="tns:listFlightsResponse">
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>
</binding>
```
SOAP (Simple Object Access Protocol)

- A lightweight platform-independent messaging protocol intended for exchange of structured information in a distributed and heterogeneous environment
  - Based on XML
  - Supporting interoperability between heterogeneous entities
  - Latest version: http://www.w3.org/standards/techs/soap

- The sender and receiver should have
  - Common model for the transferred data
  - Common mechanism for encoding and decoding data
Web Services Stack

- **Web Services Application**
- **SOAP Client** *(Framework)*
  - HTTP *(SMTP, FTP)*
  - TCP/IP
- **Web Services**
  - **SOAP Server** *(Framework, WAS)*
    - WAS: Web Application Server
    - HTTP *(SMTP, FTP)*
  - TCP/IP

- **Web Services Frameworks**
  - Java: Spring
  - Python: Django, Flask
  - Ruby: Ruby on Rails
  - etc …
SOAP Message Structure

- **SOAP message elements**
  - Envelope
    - Header (optional)
    - Body
  - Attachments (optional)
SOAP Message Structure (cont’d)

- **Envelope (필수)**
  - 바디 및 헤더를 위한 컨테이너
  - 메시지의 시작과 끝을 나타냄

- **Header (선택)**
  - SOAP 메시지 전송을 위한 옵션 및 확장 메커니즘 제공 (최종 수신자까지 가는 경로상의 모든 노드에 적용)
  - 예) 보안, 트랜잭션, 메시징 신뢰성 등과 같은 고급 기능을 정의

- **Body (필수)**
  - 실제로 전송될 메시지 정보를 담고 있음 (최종 수신자를 대상으로 함)
  - Conventional messaging 또는 RPC 형태로 사용 가능

- **Attachments (선택)**
  - XML, 아스키 혹은 바이너리 포맷과 같은 첨부용 데이터 지원
SOAP Message Structure (cont’d)

SOAP Envelope NS = http://www.w3.org/2003/05/soap-envelope

Envelope

- Header (Optional)
  - any element (*)

- Body
  - any element (*)
SOAP Message Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://example.org/reservation"
                   env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                   env:mustUnderstand="true">
      <m:dateAndTime>2003-11-11T13:00:00.000-05:00</m:dateAndTime>
    </m:reservation>
  </env:Header>

  <env:Body>
    <p:schedule xmlns:p="http://example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Log Angeles</p:arriving>
        <p:departureDate>2003-11-14T17:00:00</p:departureDate>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
    </p:schedule>
  </env:Body>
</env:Envelope>
```
SOAP RPC (Remote Procedure Call)

- Restrictions on data models and encodings to represent SOAP Remote Procedure Call
  - “Invocation & Response” message patterns
  - `xmlns:rpc = http://www.w3.org/2003/05/soap-rpc`

- RPC types
  - Resource retrieval type (returns data)
    - `GetPrice(PartNumber="123")` // returns price
    - `Add(i1=10, i2=12)` // returns i1+i2
  - General type (changes state)
    - `UpdatePrice(PartNumber="123", NewPrice="200")` // updates DB
SOAP RPC Example

RPC Invocation:
(Envelop Body)

```xml
<!-- getPrice(PartNumber="123") -->
<GetPrice>
  <PartNumber>123</PartNumber> <!-- in parameter -->
</GetPrice>
```

RPC Response:
(Envelop Body)

```xml
<!-- getPrice(PartNumber="123") returns 100 -->
<GetPriceResponse>
  <rpc:result>100</rpc:result> <!-- return value -->
</GetPriceResponse>
```
Implementation of Web Services

- **SOAP Binding Protocols**
  - HTTP (using GET/POST methods)
    - mainly used
  - SMTP
  - FTP
  - ...

- **Web Services Development Environment**
  - OS: Linux, Windows, ...
  - Language: Java, JavaScript, Python, Ruby, PHP, ...
  - WAS: Apache Tomcat, WebSphere, WebLogic, ...
  - Web Services Frameworks: Spring, Django, Flask, Rails, ...
  - Web Services Engine (WAS + Framework): Apache Axis2
UDDI (Universal Description, Discovery, and Integration)

- **Requirements for UDDI**
  - Originally developed for ebXML registry/repository
  - For electronic business and commerce, it must be able to find partners, agree and access same XML vocabulary

- **Brief history of UDDI**
  - Microsoft, IBM, and Ariba founded UDDI.org (Sep, 2000)
    - More than 300 companies joined UDDI.org
  - UDDI.org was absorbed by OASIS (Jul, 2002)
  - UDDI version 3 is ratified as OASIS standard (Feb, 2005)
UDDI XML Registry

- UDDI registry & repository allows registration and discovery of XML-related documents
  - Defines required data models in XML
  - Provides SOAP interfaces for registering and discovering business and service information

- Registered objects
  - XML Documents (including WSDL documents)
  - XML DTD/Schemas
  - Trading Partner Agreements (for ebXML)
  - Business Process Description (for ebXML)
  - Software Components (binary data)
  - ...

Web Services Usage Scenario

- **Travel agent service example**
  - **Travel agents**
    - Provide customers with complete vacation packages and reservation services
    - Airline ticketing, hotel reservation, etc.
  - **Airlines and hotel chains**
    - Provide query and reservation services for their offerings (airlines and hotels) to customers
  - **Credit card companies**
    - Provide services to guarantee payments made by their customers
Service Entities for Travel Agent Scenario

- **Service providers (SP)**
  - 1 Travel Agent Service
  - 3 Airline Services
  - 3 Hotel Services
  - 1 Credit Card Service

- **Service requesters (SR)**
  - Customers
  - 1 Travel Agent Service
  - 3 Airline Services
  - 3 Hotel Services
Travel Agent Service: Step 1

- SPs register service descriptions to UDDI registry
  - Service descriptions in WSDL
  - UDDI registration via SOAP interfaces
Travel Agent Service: Step 2

- SRs search UDDI registry for required services
  - Discovery of WSDL documents
  - UDDI retrieval via SOAP interfaces
Service Invocations from each SR to SP

- Implementation of SOAP client based on WSDL
- Invocation of services thru SOAP messaging
Web Services share many characteristics with other distributed computing technologies

- Such as DCE, CORBA, Java RMI, Microsoft COM, ...
- Interface definition (WSDL vs. IDL)
- Platform independency (like CORBA or Java RMI)
- Location transparency (RPC)
- ...
Advantages of Web Services over other distributed computing technologies

- **Loosely coupled (robust)**
  - Stateless connection
  - Supports asynchronous messaging

- **Lightweight communication**
  - Both protocol headers and payloads are text (e.g., HTTP, XML, SOAP)
  - Less bandwidth issue than DCE, CORBA, COM, ...

- **Service description and discovery**
  - WSDL and UDDI
  - Communication with each other without any prior knowledge

- **Towards Ubiquitous Web Services**
  - XML and HTTP are de-facto standards for universal communication (on any device, any OS, any SW platform)
  - Firewall friendly (HTTP port #80)
References

- W3C, http://www.w3.org
- Web Services Architecture, http://www.w3.org/standards/techs/wsarch
- Web Services Description, 
  http://www.w3.org/standards/webofservices/description
- Web Services Protocols, 
  http://www.w3.org/standards/webofservices/protocols
- WSDL, http://www.w3.org/standards/techs/wsdld
- SOAP, http://www.w3.org/standards/techs/soap
- UDDI, http://www.uddi.org