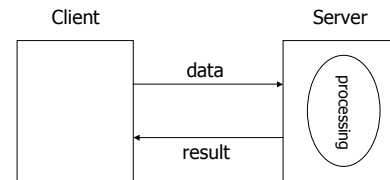


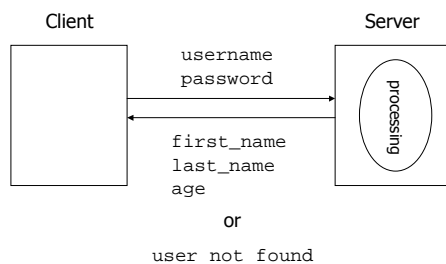
CS520 Web Programming Web Services

Chengyu Sun
California State University, Los Angeles

Client-Server Architecture



Client-Server Example



Socket Programming – Client

```
create socket
write string to socket
write string to socket
read string from socket
if( "user not found" ) return null;
else
    return new User(
        read string from socket
        read string from socket
        read integer from socket
    )
close socket
```

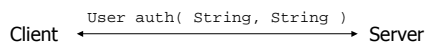
data -> packet

App-specific protocol

data <- packet

◆ Lots of networking code that has nothing to do with application logic and has to be repeated for each application

Client-Server Interaction as Function Calls



Client side:

```
User user = auth( username, password );
```

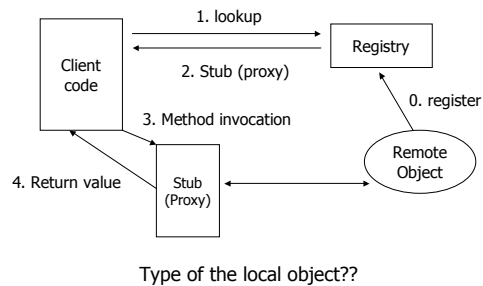
Server side:

```
User auth( String username, String password )
{
    ...
    if ( isValid ) return user;
    else return null;
}
```

Remote Procedure Call

- ◆ Remote Procedure Call (RPC)
 - C
- ◆ CORBA
 - Cross platform
 - Interface Definition Language (IDL)
- ◆ Remote Method Invocation (RMI)
 - Java
- ◆ Web services
 - XML as IDL

RMI



Interface

- ◆ Must extend `java.rmi.Remote`
- ◆ Shared by both client and server code
- ◆ E.g. `AuthInterface`

```
public interface AuthInterface extends java.rmi.Remote
{
    User auth( String username, String password )
        throws java.rmi.RemoteException;
}
```

Remote Object

```
public class AuthImpl implements AuthInterface
{
    public User auth( String username, String password )
        throws java.rmi.RemoteException
    {
        // user authentication
    }
}
```

Stub

- ◆ Created automatically

```
public class AuthStub implements AuthInterface
{
    public User auth( String username, String password )
        throws java.rmi.RemoteException
    {
        // connect to the server
        // send username and password to the server
        // return the result
    }
}
```

More About RMI

- ◆ SUN's RMI tutorial at <http://java.sun.com/docs/books/tutorial/rmi/>
 - Compilation and Execution

Spring RMI Support

- ◆ POJO interface and implementation
- ◆ `RmiServiceExporter` handles remote object registration, lookup, stub generation etc.
 - `service`, `serviceInterface`, and `serviceName`
 - `registryPort`
- ◆ Evelyn example

Alternatives to RMI

Name	Language	Message Type	Port
RMI	Java-to-Java	Binary	Default 1099
Hessian	Mostly Java-to-Java	Binary	HTTP
Burlap	Any	XML	HTTP
Spring HTTP Invoker	Java-to-Java	Binary	HTTP
Web services	Any	XML	HTTP

Web Services

- ◆ Roughly speaking, anything that encodes RPC calls in *XML messages* and transport them over *HTTP*
- ◆ Simple Object Access Protocol (SOAP)
- ◆ Web Service Description Language (WSDL)
- ◆ Universal Description, Discovery, and Integration (UDDI)

SOAP

- ◆ <http://www.w3.org/TR/soap/>
- ◆ Format conventions for message content and routing directions in the form of an *envelope*
- ◆ Rules for encoding custom data types
- ◆ Application of the envelop and the data encoding rules for representing RPC calls and responses
- ◆ Transport protocol binding (usually HTTP)

A Sample SOAP Message

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doSpellingSuggestion xmlns:ns1="urn:GoogleSearch"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string"> 00000000000000000000000000000000</key>
      <phrase xsi:type="xsd:string"> britney spears</phrase>
    </ns1:doSpellingSuggestion>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Things to Note

- ◆ Namespaces
- ◆ <Envelop>
 - Optional <Header> - for information related to processing of the message
 - <Body>
- ◆ encodingStyle
- ◆ <Fault>
 - Only sub-element of <Body> defined by SOAP

SOAP Encoding

- ◆ <http://schemas.xmlsoap.org/encoding>
- ◆ Include all built-in data types of *XML Schema Part 2: Datatypes*
 - xsi and xsd name spaces

SOAP Encoding Examples

```
int a = 10;      <a xsi:type="xsd:int">10</a>
float x = 3.14159; <x xsi:type="xsd:float">3.14159</x>
String s = "SOAP"; <s xsi:type="xsd:string">SOAP</s>
```

Compound Values and Other Rules

```
<iArray xsi:type=SOAP-ENC:Array SOAP-ENC:arrayType="xsd:int[3]">
  <val>10</val>
  <val>20</val>
  <val>30</val>
</iArray>
<Sample>
  <iVal xsi:type="xsd:int">10</iVal>
  <sVal xsi:type="xsd:string">Ten</sVal>
</Sample>
```

- ◆ References, default values, custom types, root attribute, complex types, custom serialization ...

SOAP RPC Elements

- ◆ Target object URI in HTTP header
- ◆ Namespace qualified method name and method parameters
- ◆ Optional SOAP header for additional data that's not part of the parameter list

A Sample SOAP RPC Response

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doSpellingSuggestionResponse xmlns:ns1="urn:GoogleSearch"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xsi:type="xsd:string">britney spears</return>
    </ns1:doSpellingSuggestionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A Sample Fault Response

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>Client Error</faultstring>
      <detail>
        <m:dowJonesfaultdetails xmlns:m="DowJones">
          <message>Invalid Currency</message>
          <errorCode>1234</errorCode>
        </m:dowJonesfaultdetails>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

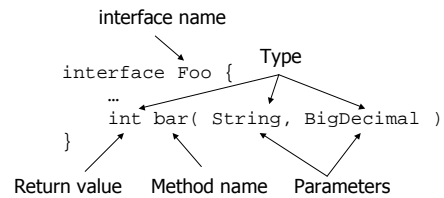
WSDL

- ◆ A language for describing web services
 - Where the service
 - What the service does
 - How to invoke the operations of the service
- ◆ *Why do we need WSDL when we have API documentation??*

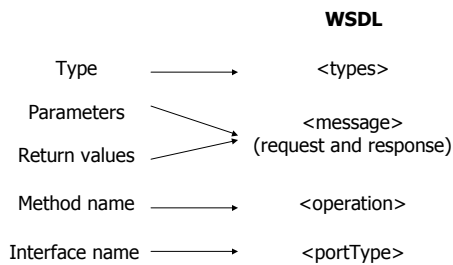
Sample WSDL Documents

- ◆ Amazon ECS - <http://webservices.amazon.com/AWSECCommerceService/AWSECommerceService.wsdl>
- ◆ Google Web APIs - <http://api.google.com/GoogleSearch.wsdl>

How Do We Describe an API?



How Do We Describe an Web Service API?



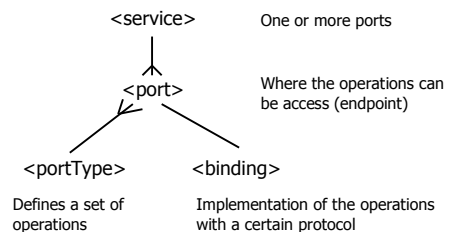
A Little More Details

- ◆ The name attribute uniquely identifies a <message>, an <operation>, or a <portType>
- ◆ Operation behavior patterns
 - Client initiated
 - Request-response
 - One-way
 - Server initiated
 - Solicit-response
 - Notification
- ◆ <fault>

Other WSDL Elements

- ◆ <definitions>
 - targetNamespace
- ◆ <import>
- ◆ <binding> - concrete protocol and format specification for a <portType>
 - E.g. <input> should be in SOAP header or body, what encoding rules should be used etc.
- ◆ <service>

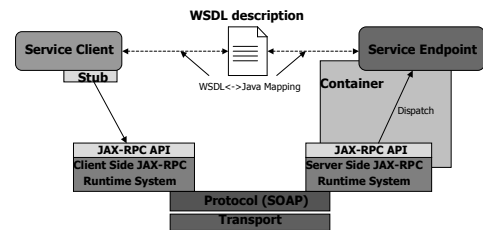
Service



JAX-RPC

- ◆ <http://java.sun.com/webservices/jaxrpc/index.jsp>
- ◆ A specification for building XML-based web services and clients using Java

JAX-RPC Architecture



Source: <http://www.pankaj-k.net/axis4tag/>

Service Invocation Patterns

- ◆ Static binding
 - statically generated stub
- ◆ Dynamic binding
 - statically generated interface
 - `javax.xml.rpc.Service.getPort()`
- ◆ Dynamic Invocation Interface (DII)
 - `javax.xml.rpc.Call`

Apache Axis

- ◆ <http://ws.apache.org/axis/>
- ◆ An implementation of SOAP for Java
- ◆ Simplifies producing and consuming web services
 - Create WSDL document from Java source code
 - Create Java classes from WSDL document
 - Encode and decode XML requests and responses
 - ...

Consume a Web Service with Axis and Spring

- ◆ Using static binding
- ◆ Examples:
 - Google Search Service

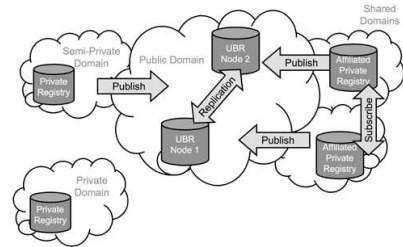
Provide a Web Service with Axis and Spring

- ◆ Axis configuration
 - Axis servlet in `web.xml`
 - `server-config.wsdd` under `/WEB-INF`
- ◆ Spring-related code
 - JaxRpc wrapper around POJO service object
- ◆ Examples:
 - Evelyn

UDDI

- ◆ A registry for web services
 - Information about the service providers
 - Classifications of services
 - Technical information about the service interfaces
- ◆ A web API for publishing, retrieving, and managing information in the registry

Registries



Core Data Types

- ◆ <businessEntity>
- ◆ <businessService>
- ◆ <bindingTemplate>
- ◆ <tModel>

http://www.uddi.org/schema/uddi_v1.xsd
http://www.uddi.org/schema/uddi_v2.xsd
http://www.uddi.org/schema/uddi_v3.xsd

<businessEntity>

- ◆ Information about the service provider

```
<businessEntity businessKey="uuid:xxxxxxxxxxxxxxxxxxxxx"
  operator="http://some.com"
  authorizedName="John Doe">
  <name>Some Company</name>
  <description>We provide web services</description>
  <contacts>...</contacts>
  <businessServices>
  ...
  </businessServices>
  ...
</businessEntity>
```

<businessService>

- ◆ Descriptive information about services

```
<businessService serviceKey="uuid:xxxxxxxxxxxxxxxxxxxxx"
  businessKey="uuid:xxxxxxxxxxxxxxxxxxxxx">
  <name>Hello World</name>
  <description>A great web service</description>
  <bindingTemplates>
  ...
  </bindingTemplates>
</businessService>
```

<bindingTemplate>

- ◆ Technical information about services

```
<bindingTemplate bindingKey="xxxxxxxxxxxxxxxxxxxxx"
  serviceKey="xxxxxxxxxxxxxxxxxxxxx">
  <description xml:lang="en">
  SOAP binding for Hello World
  </description>
  <accessPoint URLType="http">
  http://localhost:8080/soap
  </accessPoint>
  <tModelInstanceDetails>
  <tModelInstanceInfo tModelKey="xxxxxxxxxxxxx" />
  </tModelInstanceDetails>
</bindingTemplate>
```

<tModel>

◆ Interface specification about services

```
<tModel TModelKey="uuid:xxxxxxxxxxxxxxxxxxxxxxx"
  operator="http://some.com"
  authorizedName="John Doe">
  <name>Hello World Port Type</name>
  <description>
    Interface for a great web service
  </description>
  <overviewDoc>
    <overviewURL>
      http://localhost:8080/soap/helloworld.wsdl
    </overviewURL>
  </overviewDoc>
</tModel>
```

UDDI APIs

◆ Node API Sets

- Interaction among registry nodes

◆ Client API Sets

- Publish services to a registry
- Search a registry for services

WSDL for UDDI Client API

- ◆ http://www.uddi.org/wsd/publish_v2.wsdl
- ◆ http://www.uddi.org/wsd/inquire_v2.wsdl

Tools and Libraries

- ◆ <http://uddi.org/solutions.html>
- ◆ Ruddi - <http://www.ruddi.biz/>