Web Services

Trainer :Sanjay Kumar Mobile:91-9818254421, 9910587200 Email id:javaconsultant.sanjaykumar@gmail.com ,Skyp id:sanjubaba1982

History

- Web services evolved from previous technologies that served the same purpose such as RPC, ORPC (DCOM, CORBA and JAVA RMI).
- Web Services were intended to solve three main problems:

1. Interoperability
2. Firewall traversal
3. Complexity

Firewall traversal

- Collaboration across corporations was an issue because distributed systems such as CORBA and DCOM used non-standard ports.

Firewall traversal

- Collaboration across corporations was an issue because distributed systems such as CORBA and DCOM used non-standard ports.
- Web Services use HTTP as a transport protocol and most of the firewalls allow access though port 80 (HTTP), leading to easier and dynamic collaboration.

Complexity

- Web Services is a developer-friendly service system.

Complexity

- Web Services is a developer-friendly service system.
- Most of the above-mentioned technologies such as RMI, COM, and CORBA involve a whole learning curve.
- New technologies and languages have to be learnt to implement these services.

Web Service definition revisited

- A more precise definition:
  - an application component that:
    - Communicates via open protocols (HTTP, SMTP, etc.)
    - Processes XML messages framed using SOAP
    - Describes its messages using XML Schema
    - Provides an endpoint description using WSDL
    - Can be discovered using UDDI

Web Services Components

- **XML** – eXtensible Markup Language – A uniform data representation and exchange mechanism.
- **SOAP** – Simple Object Access Protocol – A standard way for communication.
- **UDDI** – Universal Description, Discovery and Integration specification – A mechanism to register and locate WS based application.
- **WSDL** – Web Services Description Language – A standard meta language to described the services offered.

Example – A simple Web Service

Example – A simple Web Service

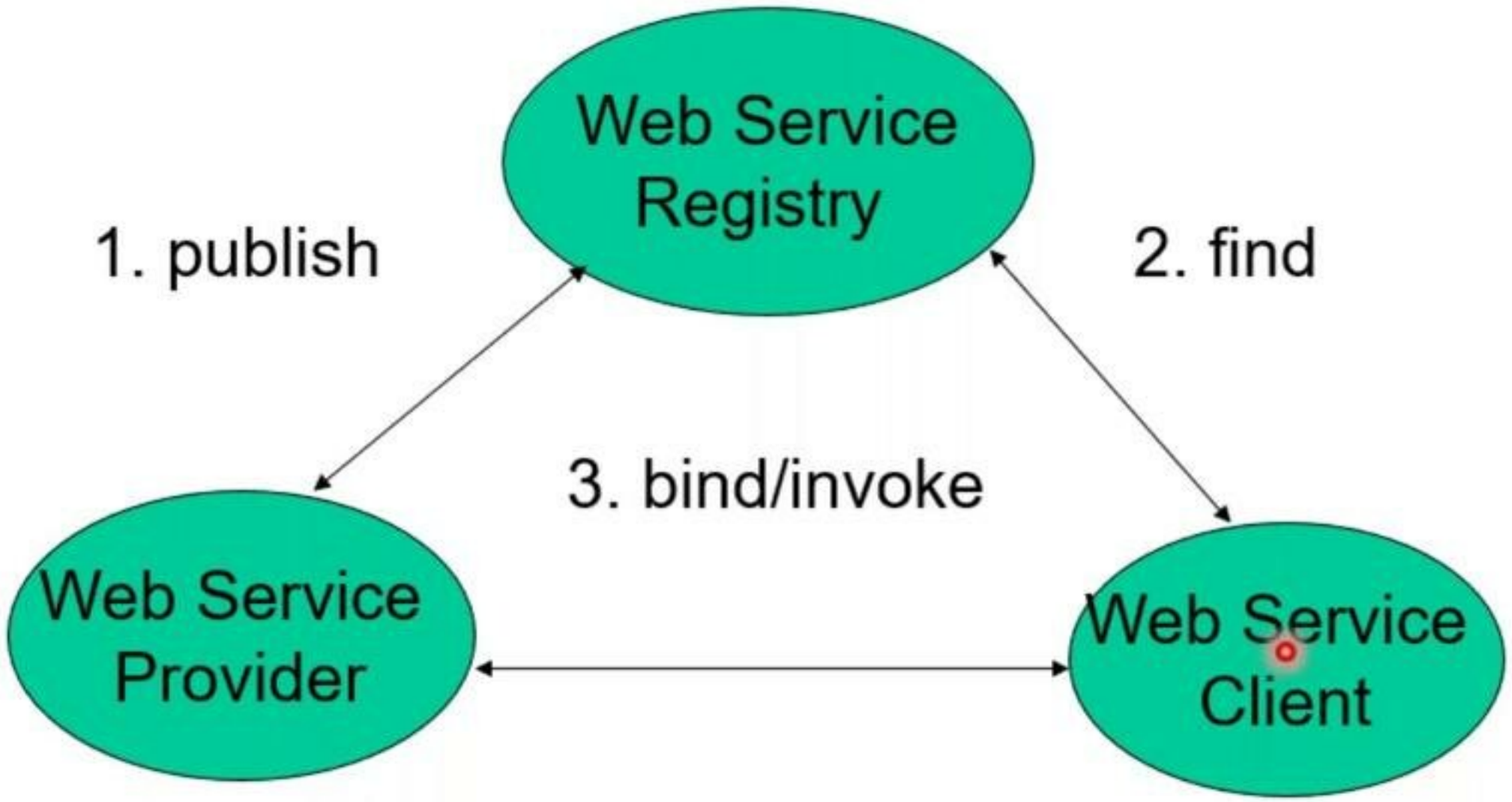
- A buyer (which might be a simple client) is ordering goods from a seller service.
- The buyer finds the seller service by searching the UDDI directory.

The Web Service Model

- The Web Services architecture is based upon the interactions between three roles:
  - Service provider
  - Service registry
  - Service requestor
- The interactions involve the:
  - Publish operations
  - Find operation
  - Bind operations.

The Web Service Model (cont)

The Web Services model follows the *publish*, *find*, and *bind* paradigm.



XML vs HTML

An HTML example:

```
<html>
<body>
    <h2>John Doe</h2>
    <p>2 Backroads Lane<br>
        New York<br>
        045935435<br>
        john.doe@gmail.com<br>
        </p>
</body>
</html>
```

XML vs HTML

- This will be displayed as:

John Doe

2 Backroads Lane
New York
045935435
John.doe@gmail.com

- HTML specifies how the document is to be displayed, and not what information is contained in the document.
- Hard for machine to extract the embedded information. Relatively easy for human.

XML vs HTML

- Now look at the following:

```
<?xml version=1.0?>
<contact>
  <name>John Doe</name>
  <address>2 Backroads Lane</address>
  <country>New York</country>
  <phone>045935435</phone>
  <email>john.doe@gmail.com</email>
</contact>
```

- In this case:
  - The information contained is being marked, but not for displaying.
  - Readable by both human and machines.

SOAP

- SOAP originally stood for "Simple Object Access Protocol" .
- Web Services expose useful functionality to Web users through a standard Web protocol called SOAP.
- Soap is an XML vocabulary standard to enable programs on separate computers to interact across any network. SOAP is a simple markup language for describing messages between applications.
- Soap uses mainly HTTP as a transport protocol. That is, HTTP message contains a SOAP message as its payload section.

SOAP Characteristics

- SOAP has three major characteristics:
  - Extensibility – security and WS-routing are among the extensions under development.

Components of a SOAP configuration

A SOAP message is an ordinary XML document containing the following elements:

- A required Envelope element that identifies the XML document as a SOAP message.

Components of a SOAP configuration

A SOAP message is an ordinary XML document containing the following elements:

- A required Envelope element that identifies the XML document as a SOAP message.
- An optional Header element that contains header information.
- A required Body element that contains call and response information.
- An optional Fault element that provides information about errors that occurred while processing the message.

SOAP Request

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8 Content-Length: 150

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle=http://www.w3.org/2001/12/soap-encoding”>

    <soap:Body xmlns:m="http://www.stock.org/stock">
        <m:GetStockPrice>
            <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
    </soap:Body>
</soap:Envelope>
```

SOAP Security

- SOAP uses HTTP as a transport protocol and hence can use HTTP security mainly HTTP over SSL.

SOAP Security

- SOAP uses HTTP as a transport protocol and hence can use HTTP security mainly HTTP over SSL.
- But, since SOAP can run over a number of application protocols (such as SMTP) security had to be considered.
- The WS-Security specification defines a complete encryption system.

The WSDL Document Structure

- A WSDL document is just a simple XML document.
- It defines a web service using these major elements:

WSDL Document

```
<message name="GetStockPriceRequest">
    <part name="stock" type="xs:string"/>
 </message>
<message name="GetStockPriceResponse">
    <part name="value" type="xs:string"/>
 </message>

<portType name="StocksRates">
      <operation name="GetStockPrice">
          <input message="GetStockPriceRequest"/>
          <output message="GetStockPriceResponse"/>
      </operation>
</portType>
```

UDDI

- UDDI stands for Universal Description, Discovery and Integration.

UDDI

- UDDI stands for Universal Description, Discovery and Integration.
- UDDI is a directory for storing information about web services , like yellow pages.

JAX-RPC vs. JAX-WS

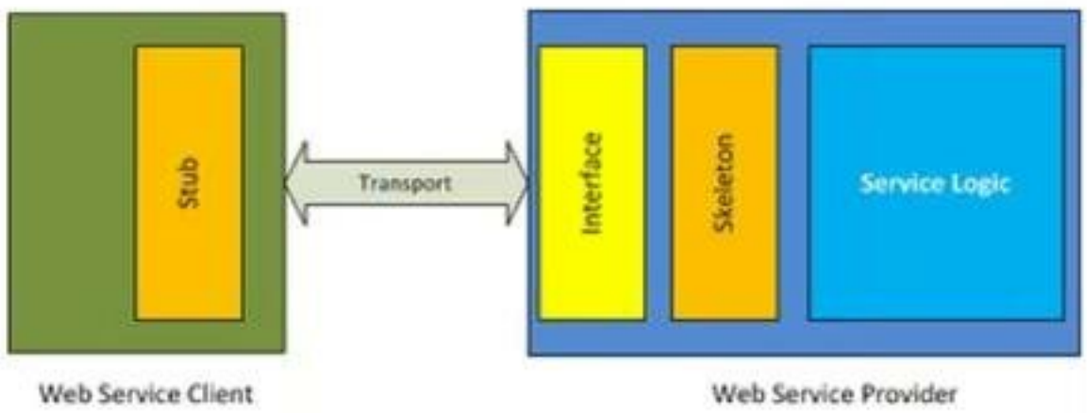
- Web services have been around a while now. First there was SOAP. But SOAP only described what the messages looked like. Then there was WSDL. But WSDL didn't tell you how to write web services in Java™.

JAX-RPC vs. JAX-WS

- Web services have been around a while now. First there was SOAP. But SOAP only described what the messages looked like. Then there was WSDL. But WSDL didn't tell you how to write web services in Java™.
- Then along came JAX-RPC 1.0. After a few months of use, the Java Community Process (JCP) folks who wrote that specification realized that it needed a few tweaks, so out came JAX-RPC 1.1. After a year or so of using that specification, the JCP folks wanted to build a better version: JAX-RPC 2.0. A primary goal was to align with industry direction, but the industry was not merely doing RPC web services, they were also doing message-oriented web services.
- So "RPC" was removed from the name and replaced with "WS" (which stands for web Services, of course).
- Thus the successor to JAX-RPC 1.1 is JAX-WS 2.0 - the Java API for XML-based web services.

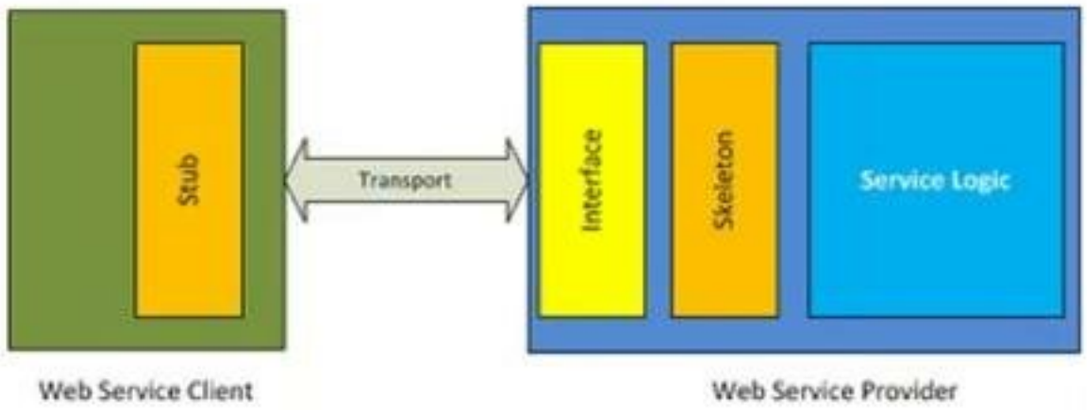
Stubs and Skeletons

- Stub and skeleton are counterparts in a web service setup.
- Skeleton belongs to service provider side and stub belongs to receiver side.



Stubs and Skeletons

- Stub and skeleton are counterparts in a web service setup.
- Skeleton belongs to service provider side and stub belongs to receiver side.
- At lower level stub and skeleton communicate with each other. From client side the business objects communicates with stub objects and stub takes the responsibility to form the message and invoke the web service.



Step by Step – Deployment

Web Service Client Creation

Besides configuring the service implementation, Eclipse generates a web service client.

This will create a dynamic web java project and web service client.

We could also create a java project and write a client to access the web service.

Deploy Web Service and Client

Click Next .

This step will start the associated runtime Tomcat.

3
4
5
6
7

Click to add notes

WEB_INTELLECT7 - Eclipse IDE

File Edit Navigate Search Project Run Window Help

Project Explorer

- 001.Greetings
- 002.CarryingDataFromFormToServlet
- 003.CarringNDataToServlet
- 004.CarringMultiValuedData1
- 005.CarringMultiValuedData2
- 006.UsingServletConfigAndServletContext
- 007.DbAccessFromServlet
- 008.SessionManagementByURLReWriting
- 009.SessionManagemtByHiddenFields
- 010.SessionManagementByCookies
- 011.SessionManagementByHttpSession
- 012.HttpSessionAndAttributListener
- 013.JspBasicTagDemo
- 014.JspForwardAndInclude
- 015.JspBeanHandling
- 016.JspResponseAndExceptionHandling
- 017.JspBeansAndExceptionHandling
- 018.JspDatabaseHandling
- 019.JunitTestingPurseManger
- 020.JunitTestingCalculator
- Calculator
  - Deployment Descriptor: Calculator
  - JAX-WS Web Services
  - JRE System Library [JavaSE-1.8]
  - src/main/java
    - com.dxc.webservice
      - Calculator.java
  - Server Runtime [Apache Tomcat v9.0]
  - build
  - src
- Complaint-management-system
- EmployeeProject [EmployeeProject main ↑2]
- Servers

Markers Properties Servers Data Source Explorer Snippets Console Progress

<terminated> Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk1.8.0_311\bin\javaw.exe (20-Apr-2022, 2:44:45 pm – 2:46:13 pm)

src/main/java - Calculator