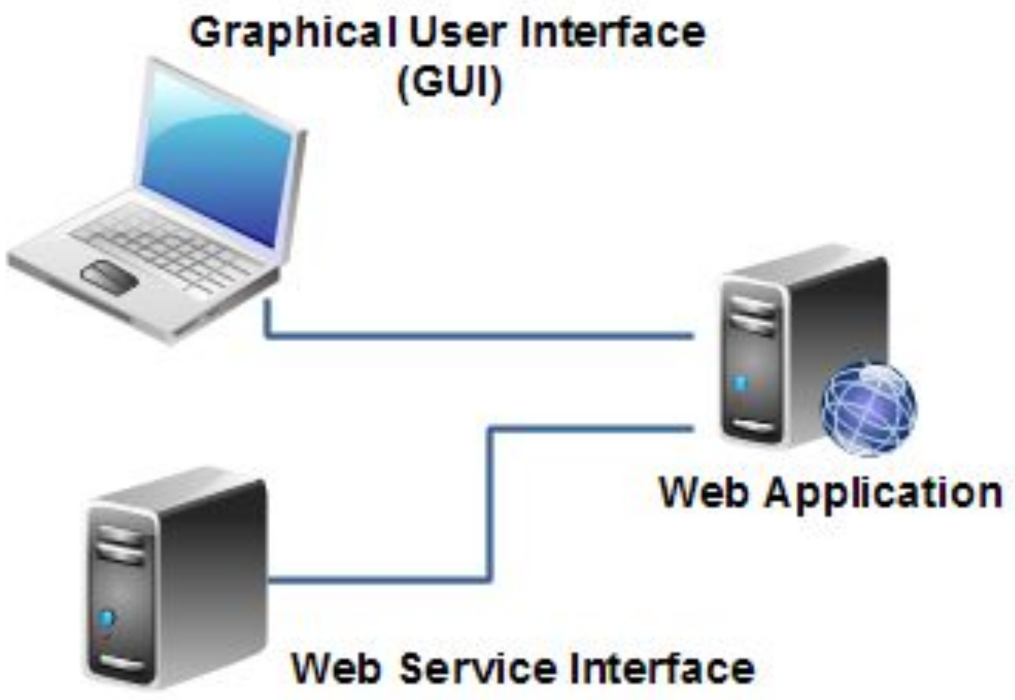
Web Services

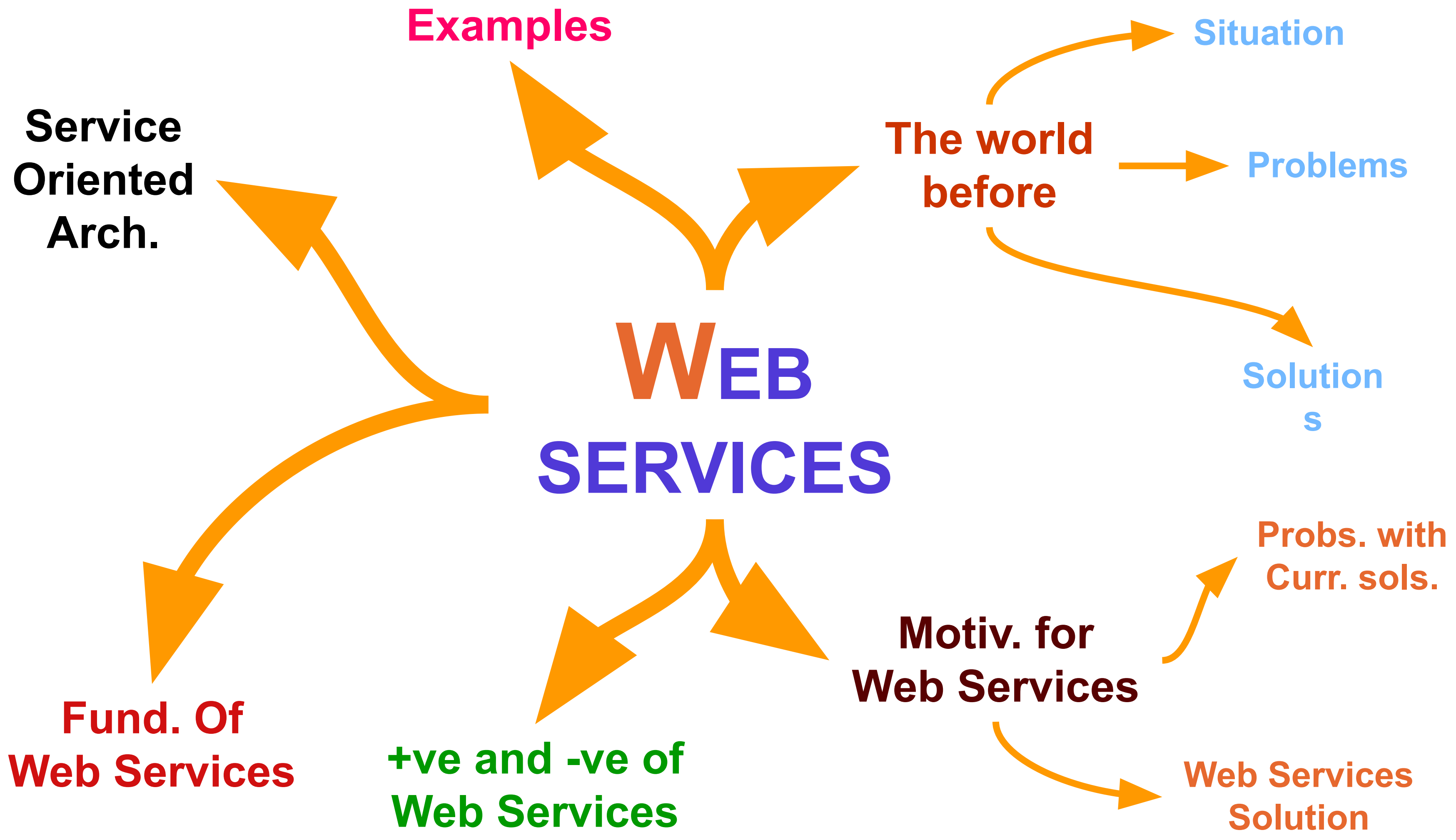
Why Webservice?

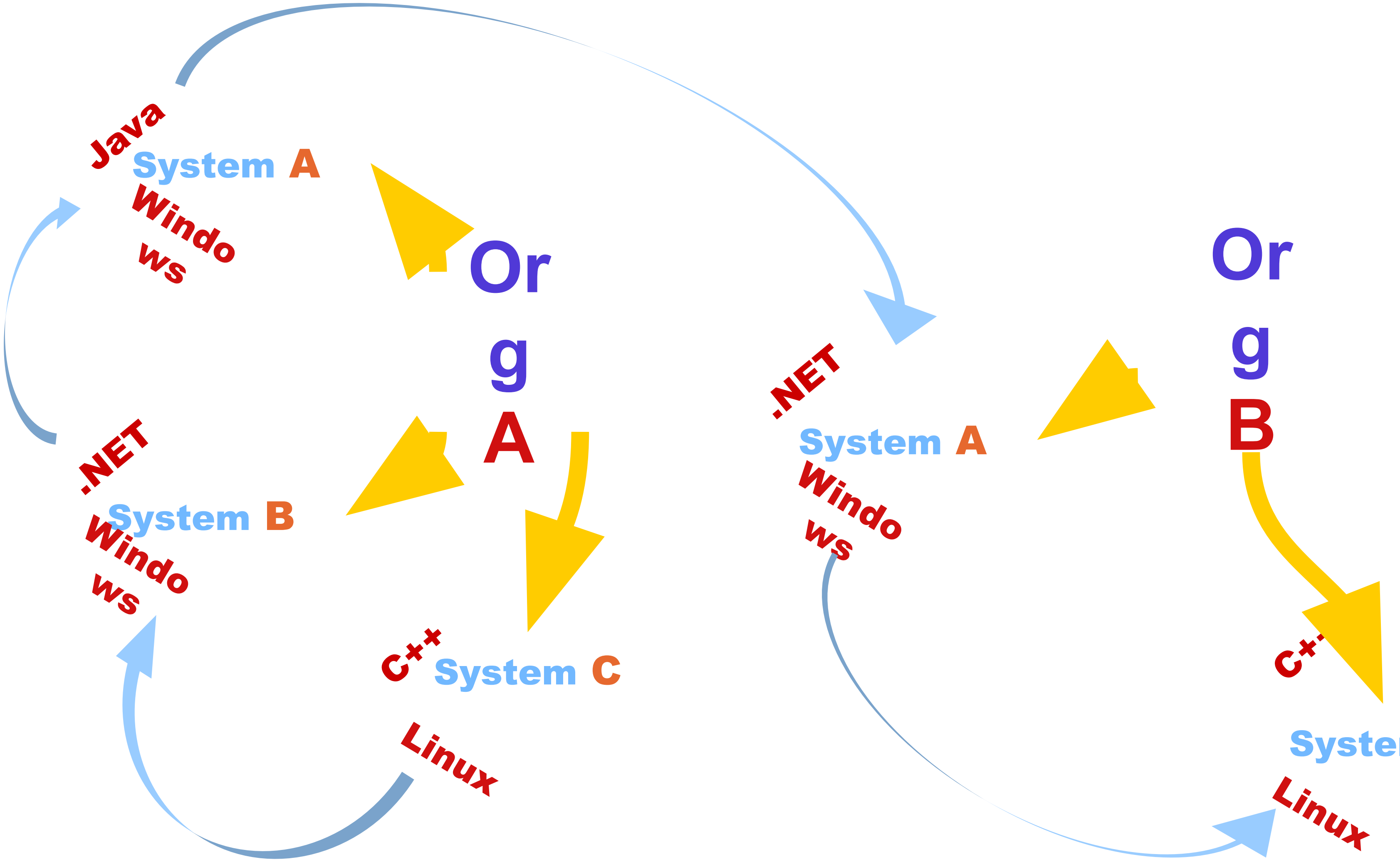
Web Services vs. Websites / Web Applications

- The main difference between a web service and a web site is, that a web site is typically intended for human consumption (human-to-computer interaction), whereas web services are typically intended for computer-to-computer interaction.

Web Service & Web App




WEB SERVICES
Examples
Service Oriented Arch.
The world before
Situation
Problems
Solutions
Fund. Of Web Services
+ve and -ve of Web Services
Motiv. for Web Services
Probs. with Curr. sols.
Web Services Solution



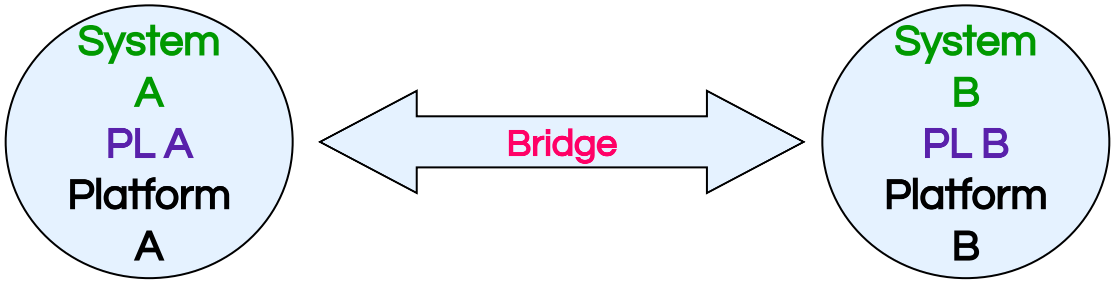
Java
System A
Windo
ws
Or
g
A
.NET
System B
Windo
ws
C++
System C
Linux
.NET
System A
Windo
ws
Or
g
B
C++
System B
Linux

Problems

- Different types of platforms
- Different types of programming languages

Solution

Create bridge



Another Better Solutions

RMI

EDI

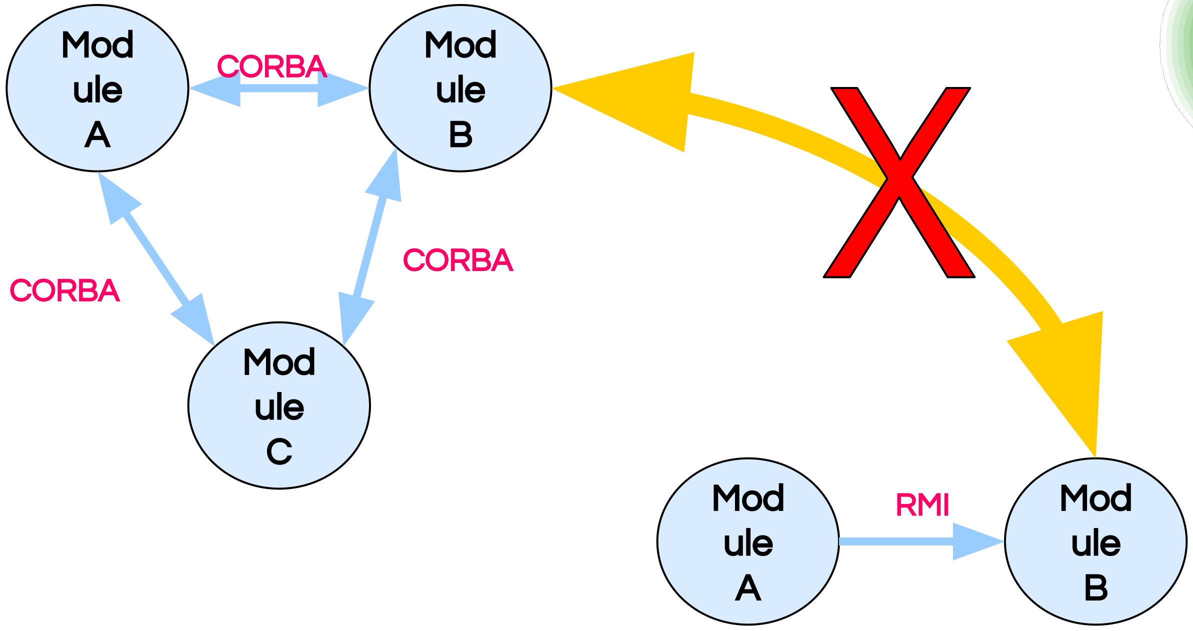
ebXML

CORBA

COM

Problems With Current Solutions

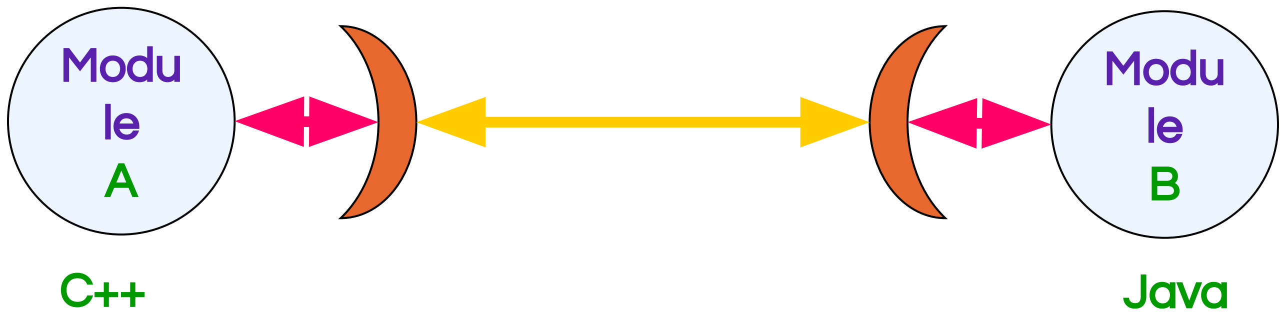
- Involve a whole learning curve
- Not based on standardized rules and specifications



PROVIDE

Standardized method of communication between software applications

Distributed components are interfaced via non-object-specific protocols



+VE AND –VE OF WEB SERVICES

Advantages of web services

Web services provide interoperability between various software applications running on disparate platforms/operating systems

Web services use open standards and protocols

By utilizing HTTP, web services can work through many common firewall security measures without requiring changes to the firewall filtering rules. Other forms of RPC may more often be blocked

+VE AND –VE OF WEB SERVICES

Advantages of web services

Web services allow software and services from different companies and locations to be combined easily to provide an integrated service.

Web services allow the reuse of services and components within an infrastructure.

Web services are loosely coupled thereby facilitating a distributed approach to application integration.

+VE AND –VE OF WEB SERVICES

Disadvantages of web services

Web services standards features such as transactions are currently nonexistent or still in their infancy compared to more mature distributed computing open standards such as CORBA.

Web services may suffer from poor performance compared to other distributed computing approaches such as RMI, CORBA, or DCOM.

Communication and standards

- Efficient (or indeed any) communication is dependent on a shared vocabulary and grammar.
- Because web services deals with inter-organisation communication these must be universal standards.

Underlying standards

The basic standards for web services are:

- XML (Extensible Markup Language)
- SOAP (simple object access protocol)
- WSDL (web services description language)
- UDDI (universal description, discovery and integration)

Web Services Architecture

- Web Services involve three major roles
  - Service Provider
  - Service Registry
  - Service Consumer

- Three major operations surround web services
  - Publishing – making a service available
  - Finding – locating web services
  - Binding – using web services

Making a service available

In order for someone to use your service they have to know about it.

- To allow users to discover a service it is published to a registry (UDDI).
- To allow users to interact with a service you must publish a description of it’s interface (methods & arguments).
- This is done using WSDL.

Making a service available

- Once you have published a description of your service you must have a host set up to serve it.
- A web server is often used to deliver services (although custom application – application communication is also possible).
- This is functionality which has to be added to the web server. In the case of the apache web server a ‘container’ application (Tomcat) can be used to make the application (servlet) available to apache (deploying).

The old transfer protocols are still there.

- Like the grid architecture web services is layered on top of existing, mature transfer protocols.
- HTTP, SMTP are still used over TCP/IP to pass the messages.
- Web services, like grids, can be seen as a functionality enhancement to the existing technologies.

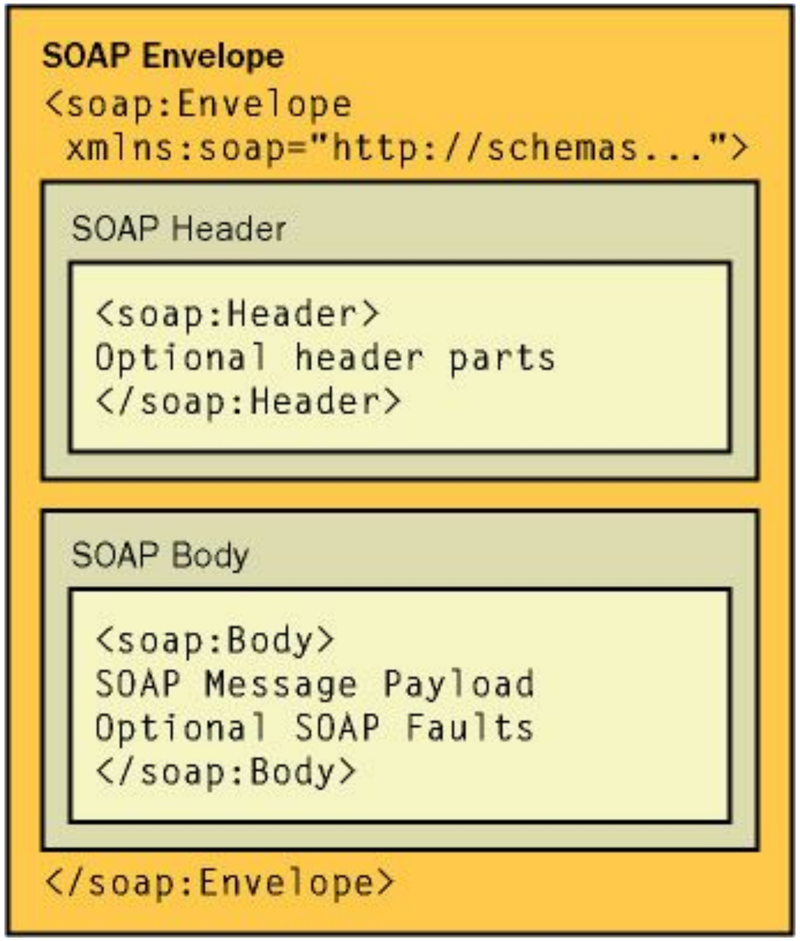
XML

- All Web Services documents are written in XML
- XML Schema are used to define the elements used in Web Services communication

SOAP

- Actually used to communicate with the Web Service
- Both the request and the response are SOAP messages
- The body of the message (whose grammar is defined by the WSDL) is contained within a SOAP “envelope”
- “Binds” the client to the web service

SOAP Format



```
<soap:Envelope
 xmlns:soap="http://schemas...">
<soap:Header>
Optional header parts
</soap:Header>
<soap:Body>
SOAP Message Payload
Optional SOAP Faults
</soap:Body>
</soap:Envelope>
```

WSDL

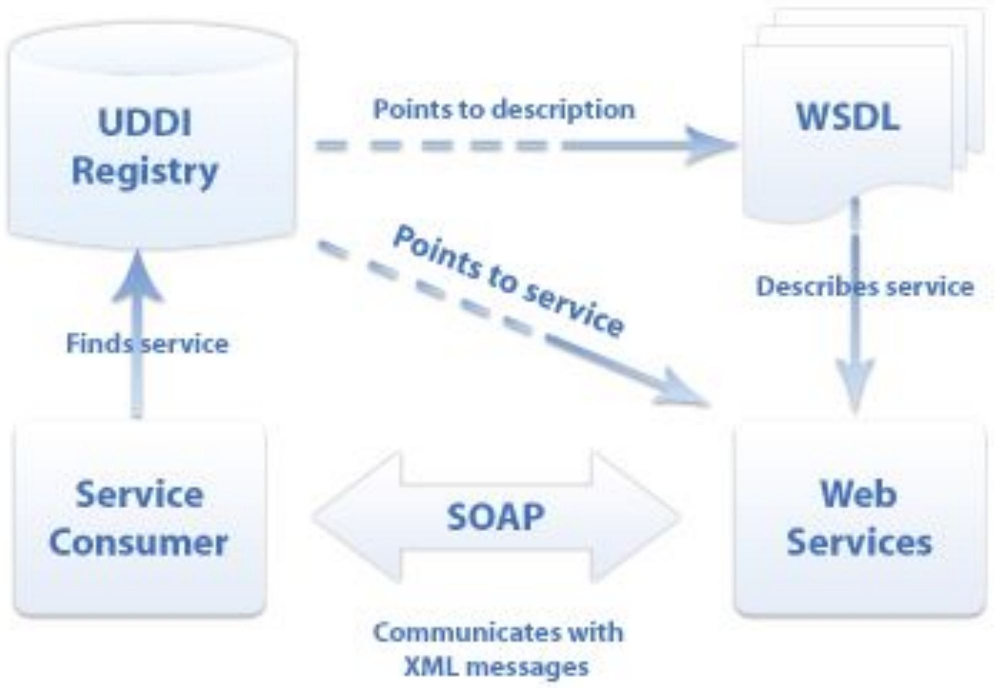
- Describes the Web Service and defines the functions that are exposed in the Web Service
- Defines the XML grammar to be used in the messages
  - Uses the W3C Schema language

UDDI

- UDDI is used to register and look up services with a central registry
- Service Providers can publish information about their business and the services that they offer
- Service consumers can look up services that are available by
  - Business
  - Service category
  - Specific service

Service Oriented Arch.

It's an architectural style of building software applications that promotes loose coupling between components so that you can reuse them



Java Architecture for XML Web Services (JAX-WS)

Web Service Ways

- It Could be Stand along
- It could e a web application

Lets Understand SOAP

- Simple Object Access Protocol (SOAP 1.1 and just SOAP in SOAP 1.2)
- W3C recommendation for message encoding.
- Provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized distributed environment.
- All SOAP messages are encoded in XML.
- The protocol is independent of runtime environments, so it works well in a heterogeneous environment for data exchange.
- Note: https://www.w3.org/2003/06/soap11-soap12.html

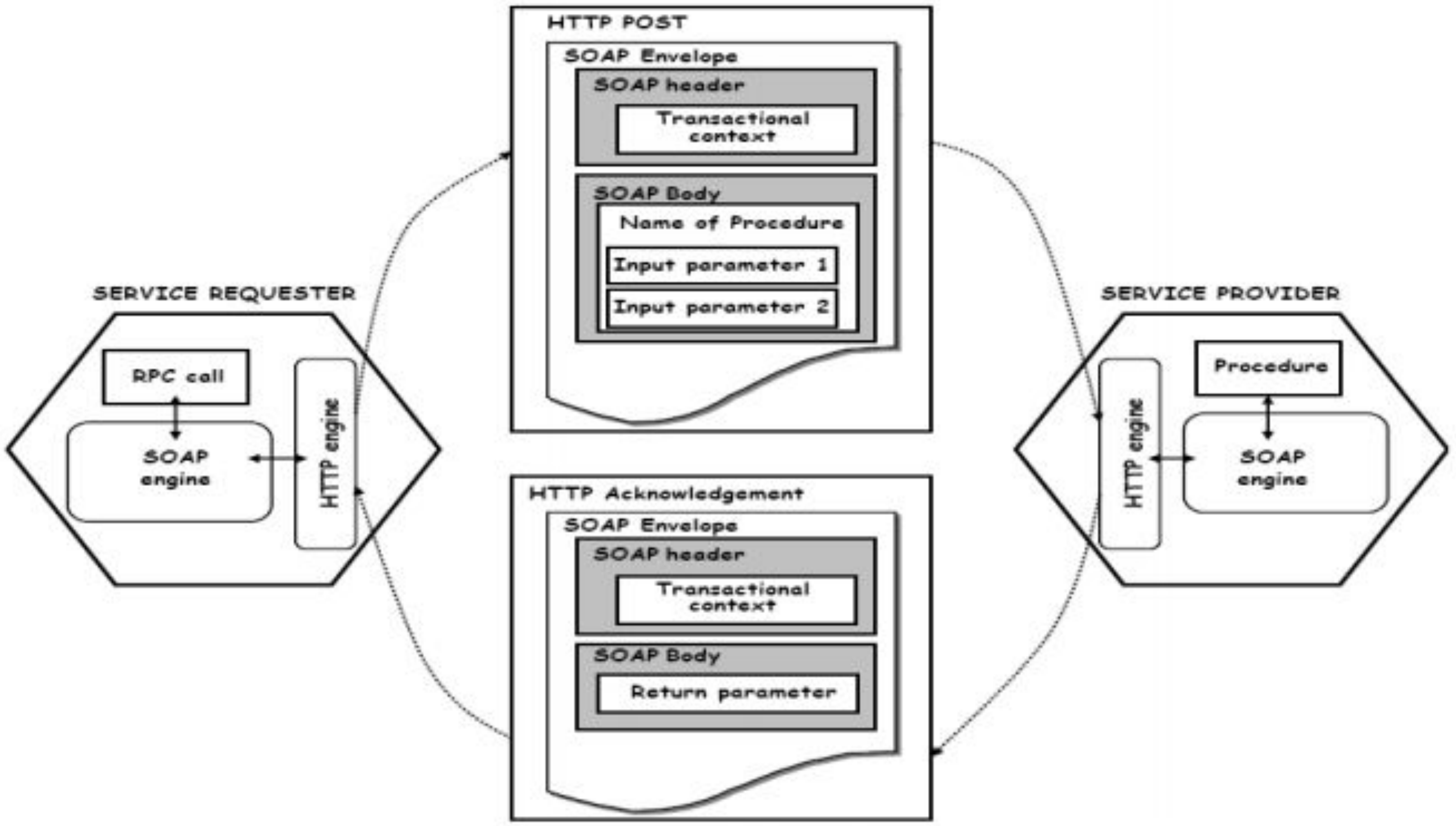
SOAP

- SOAP covers the following four main areas:
  - A message format for one-way communication describing how a message can be packed into an XML document
  - A description of how (the XML document that makes up) a SOAP message should be transported through the Web (using HTTP) or e-mail (using SMTP).

SOAP

- A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in that processing. It also specifies what parts of the messages should be read by whom and how to react in case the content is not understood.
- A set of conventions on how to turn a RPC call into a SOAP message and back as well as how to implement the RPC style of interaction (how the client side RPC call is translated into a SOAP message, forwarded, turned into a server side RPC call, the reply converted into a SOAP message and returned to the client)

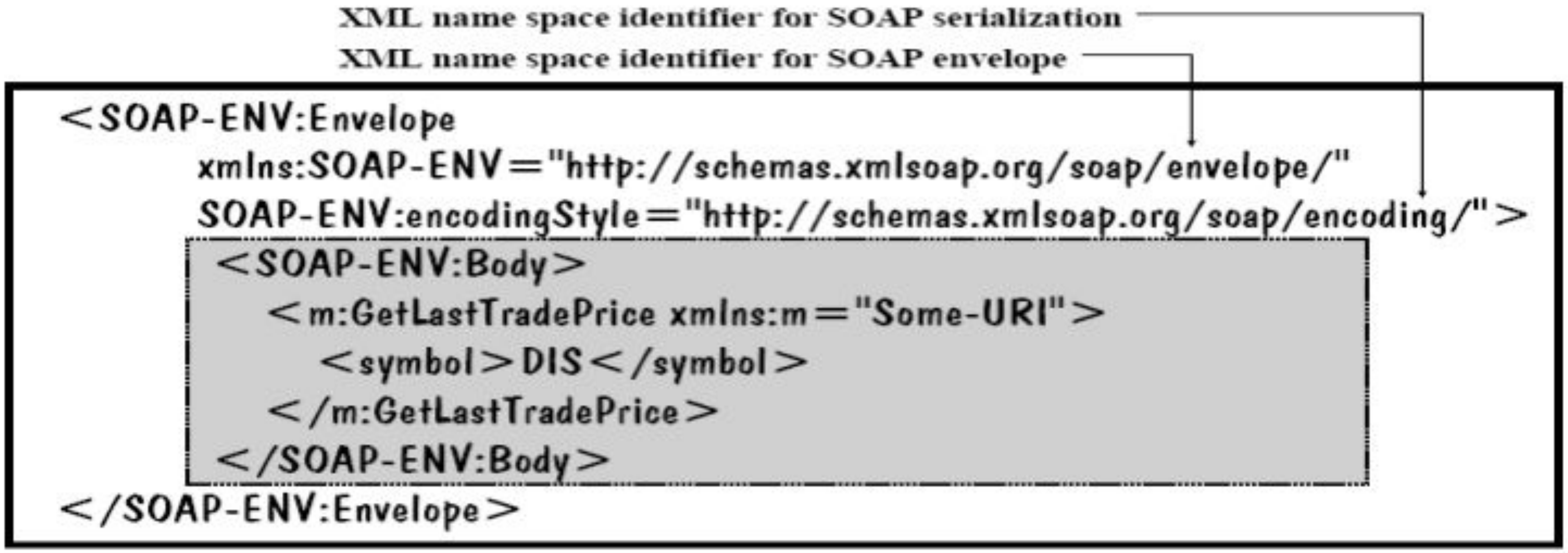
SOAP In Action



SOAP Messages

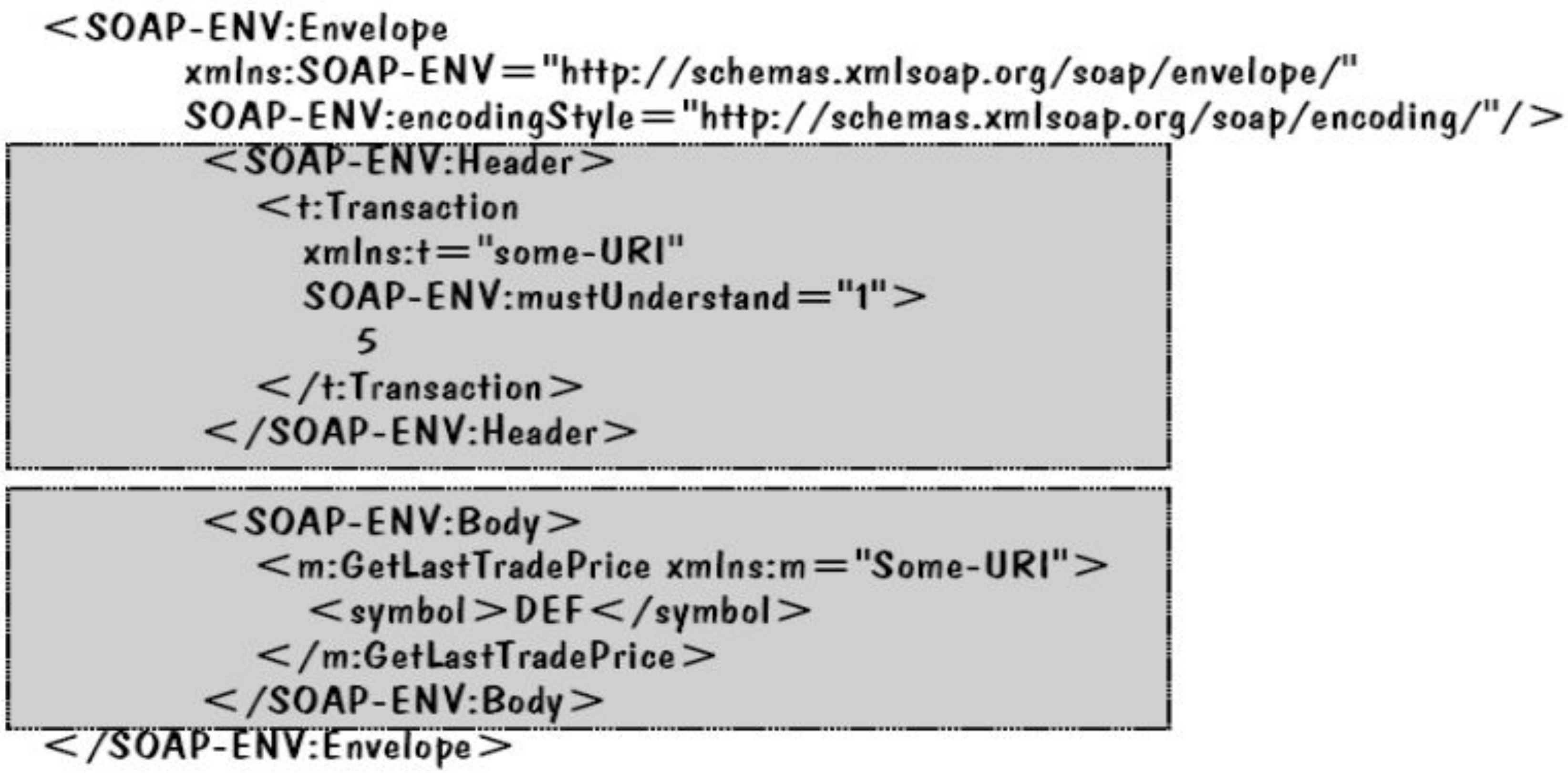
- SOAP is based on message exchanges
- Messages are seen as envelopes where the application encloses the data to be sent.
- A message has two main parts; header and body, which both can be divided into blocks
- SOAP does not specify what to do the body, it only states that the header is optional and the body is mandatory
- The use of header and body, however, is implicit.
  - The body is for application level data. The header is for infrastructure level data (security, transaction, etc)

For XML fans (SOAP, body only)


XML name space identifier for SOAP serialization
XML name space identifier for SOAP envelope

```
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <SOAP-ENV:Body>
            <m:GetLastTradePrice xmlns:m="Some-URI">
              <symbol>DIS</symbol>
            </m:GetLastTradePrice>
          </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP example, HEADER and body



```
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         <SOAP-ENV:Header>
            <t:Transaction
               xmlns:t="some-URI"
               SOAP-ENV:mustUnderstand="1">
                  5
            </t:Transaction>
         </SOAP-ENV:Header>

         <SOAP-ENV:Body>
            <m:GetLastTradePrice xmlns:m="Some-URI">
               <symbol>DEF</symbol>
            </m:GetLastTradePrice>
         </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

THE SOAP HEADER

- The header is intended as a generic place holder for information that is not necessarily application dependent (the application may not even be aware that a header was attached to the message).
- Typical uses of the header are: coordination information, identifiers (for, e.g., transactions) and security information (e.g., certificates)

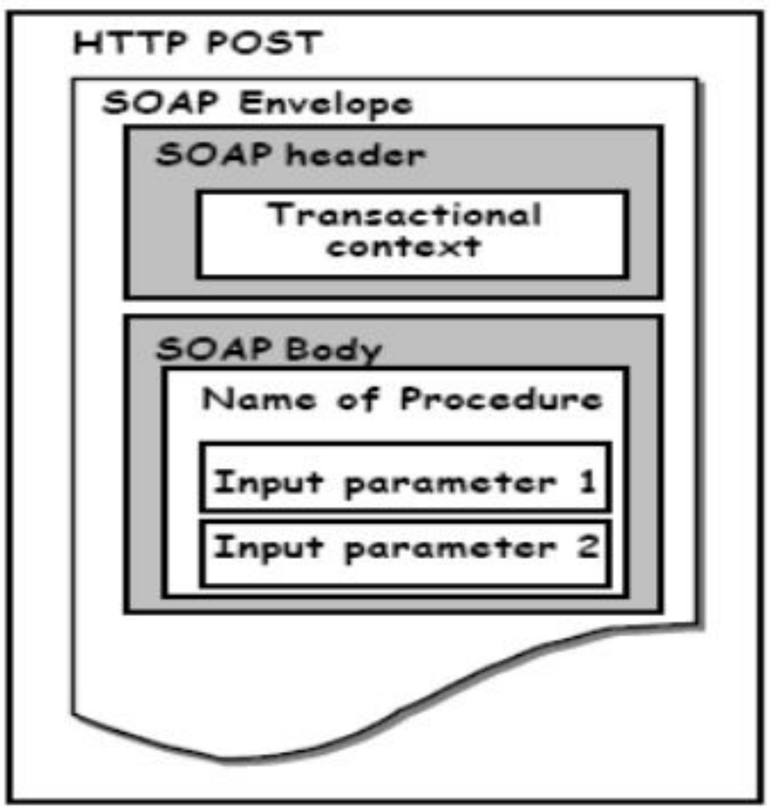
THE SOAP BODY

- The body is intended for the application specific data contained in the message
- Typically, it contains information about the name of the operation, parameter values

SOAP and HTTP

- A binding of SOAP to a transport protocol is a description of how a SOAP message is to be sent using that transport protocol
- The typical binding for SOAP is HTTP
- SOAP uses the same error and status codes as those used in HTTP so that HTTP responses can be directly interpreted by a SOAP module

SOAP AND HTTP

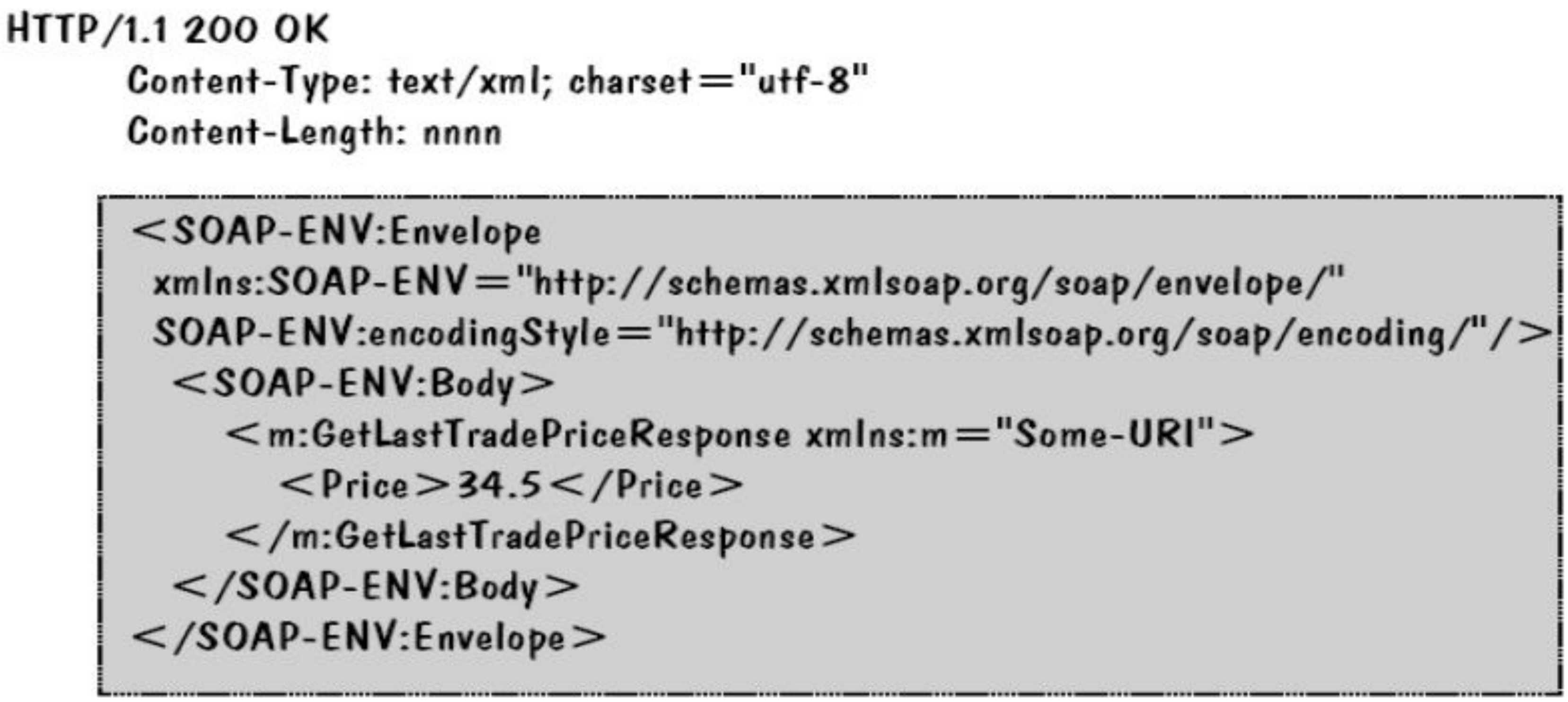


HTTP REQUEST (IN XML)

```
POST /StockQuote HTTP/1.1
        Host: www.stockquoteserver.com
        Content-Type: text/xml; charset="utf-8"
        Content-Length: nnnn
        SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI">
        <symbol>DIS</symbol>
     </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HTTP RESPONSE (IN XML)



```
HTTP/1.1 200 OK
        Content-Type: text/xml; charset="utf-8"
        Content-Length: nnnn

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <SOAP-ENV:Body>
     <m:GetLastTradePriceResponse xmlns:m="Some-URI">
        <Price>34.5</Price>
     </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Deploy JAX-WS web services on Tomcat

1. Create a web service (of course).
2. Create a sun-jaxws.xml, defines web service implementation class.
3. Create a standard web.xml, defines WSServletContextListener, WSServlet and structure of a web project.
4. Build tool to generate WAR file.
5. Copy JAX-WS dependencies to “${Tomcat}/lib” folder.
6. Copy WAR to “${Tomcat}/webapp” folder.
7. Start It.

Understanding WSDL

Element	Description
Definition	It is the root element of all WSDL documents.
Data types	The data types to be used in the messages are in the form of XML schemas.
Message	t is an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.
Operation	It is the abstract definition of the operation for a message, such as naming a method,
Port type	It is an abstract set of operations mapped to one or more end-points,

Element	Description
Binding	It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
Port	It is a combination of a binding and a network address, providing the target address of the service communication
Service	It is a collection of related end-points encompassing the service definitions in the file