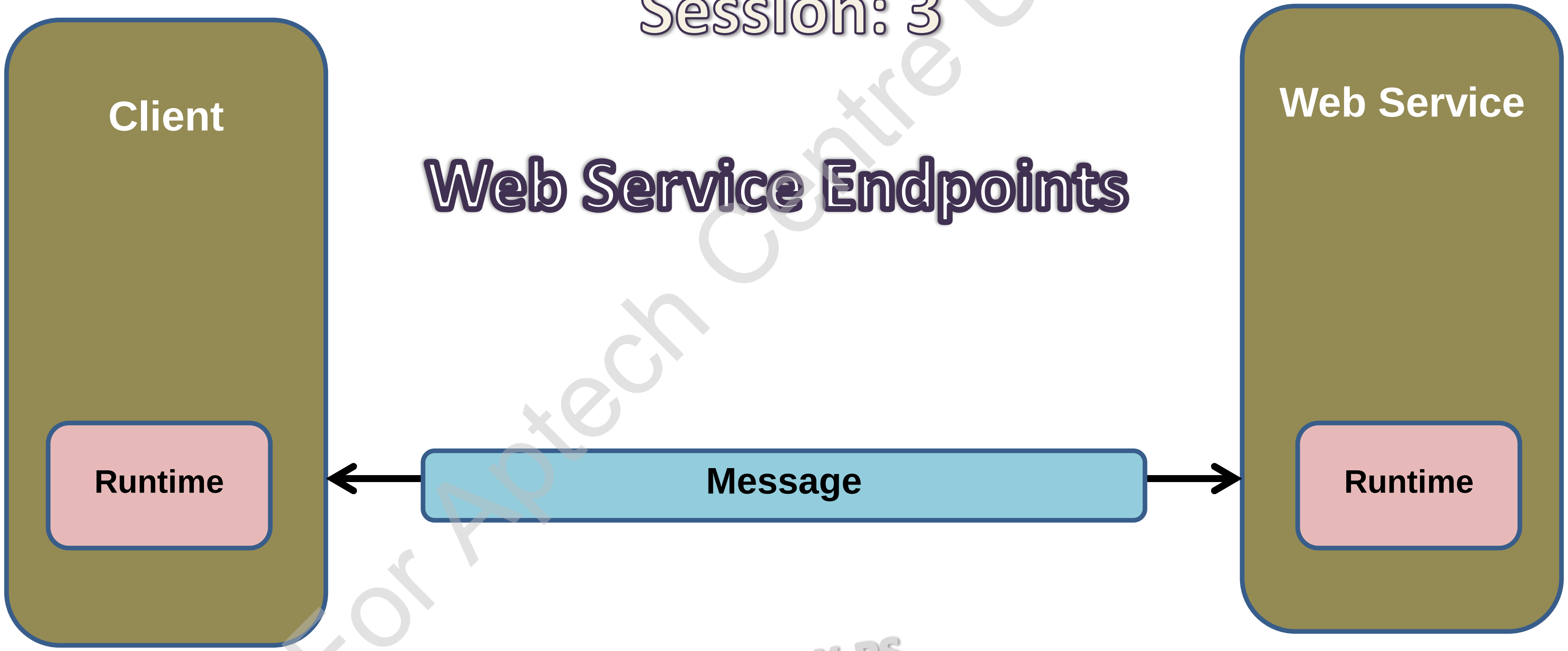
Developing Java Web Services



Session: 3

Web Service Endpoints

Objectives

- Explain the guidelines to design Web service endpoints
- Describe the method to package to deploy a Web service
- Explain the process of invoking Web service

Web Service Endpoints

A Web service endpoint is a program that implements a Web service and carries out Web service requests.

Web Service Endpoints

A URL where the service can be accessed by the client application

Web service design guidelines simplifies the process of creating endpoints

Web Service Design Guidelines

- Developer should understand the nature of Web service
- Implementing class should be annotated with either javax.jws.WebService or javax.jws.WebServiceProvider

Steps to create a Web service and a client are as follows:

1. Code the implementation class
2. Compile the implementation class
3. Package the files into a WAR file
4. Deploy the WAR file
5. Code the client class
6. Generate and compile the Web service artifacts
7. Compile the client class
8. Run the client

Web Service Design Decisions

The Web service is available to the client along with the details of the service.

Decide whether and how to publish a Web Service

Type and nature of client calls | Type of service endpoints used | Level of interoperability

↓

Determine how requests are received

Client request converted into an internal format

↓

Identify the protocol for delegating request

Requests sent to business protocol with less time and no discrepancy

↓

Decide processing of requests

Designing the interface to handle Web service requests

↓

Decide format of response to client

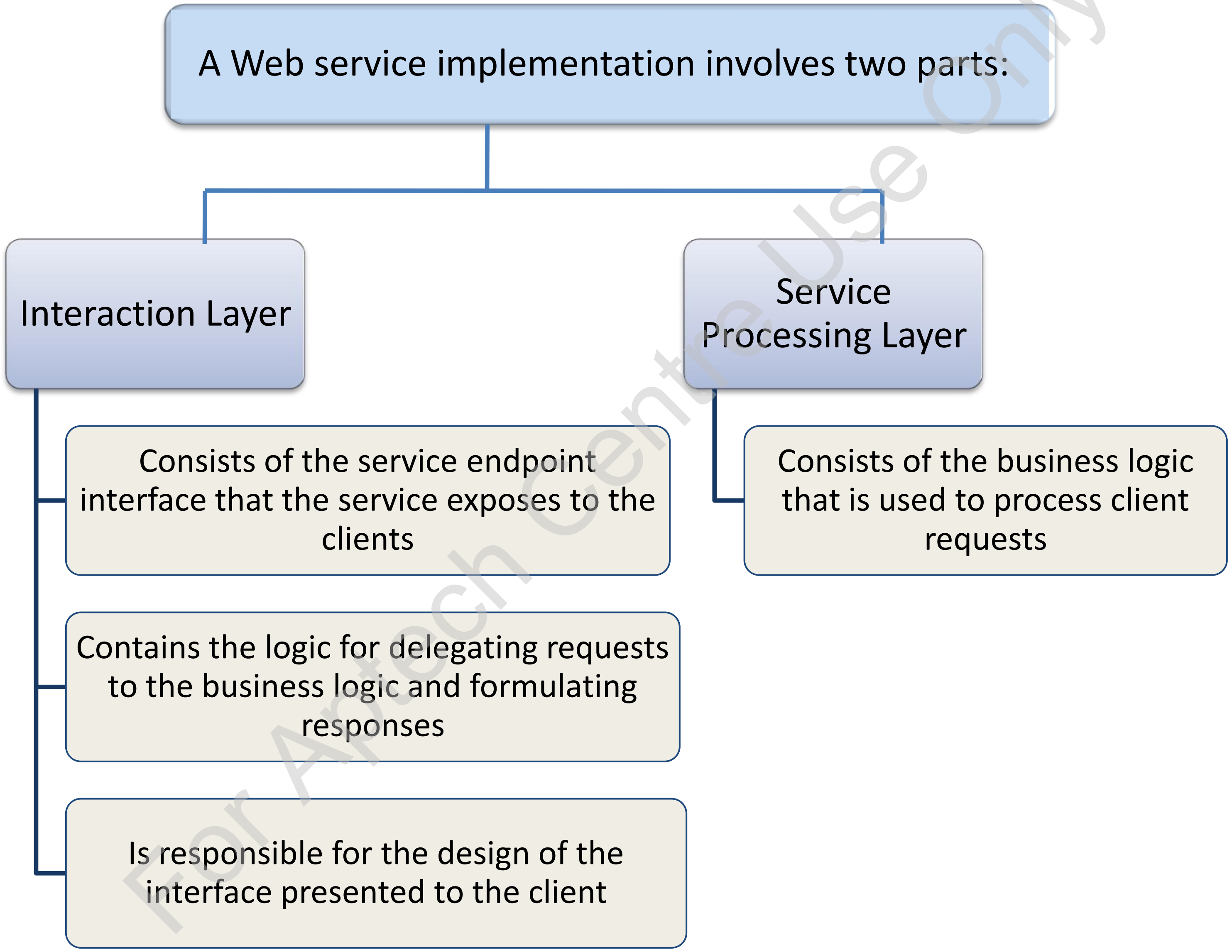
Helping the client to understand the response message

↓

Determine how problems are reported

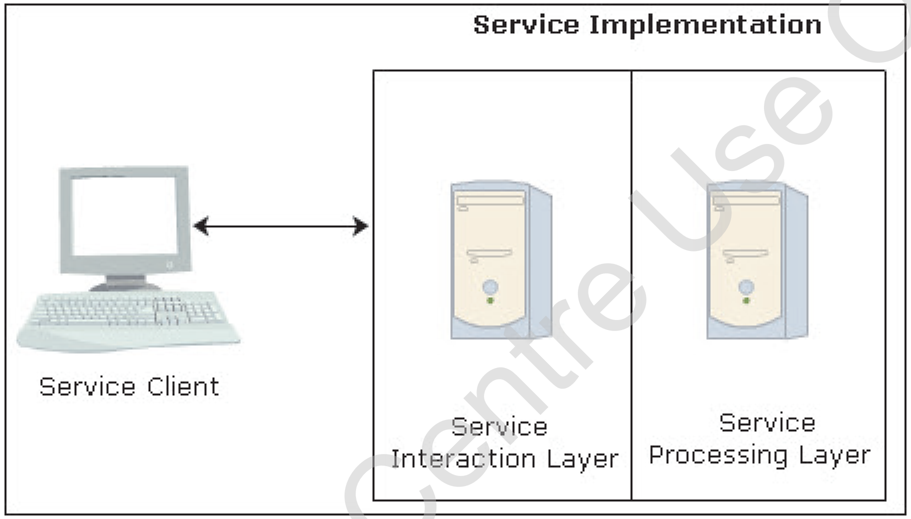
Planning to recover from errors and exceptions

Layered View of Web Service 1-4


A Web service implementation involves two parts:
Interaction Layer
Service Processing Layer
Consists of the service endpoint interface that the service exposes to the clients
Contains the logic for delegating requests to the business logic and formulating responses
Is responsible for the design of the interface presented to the client
Consists of the business logic that is used to process client requests

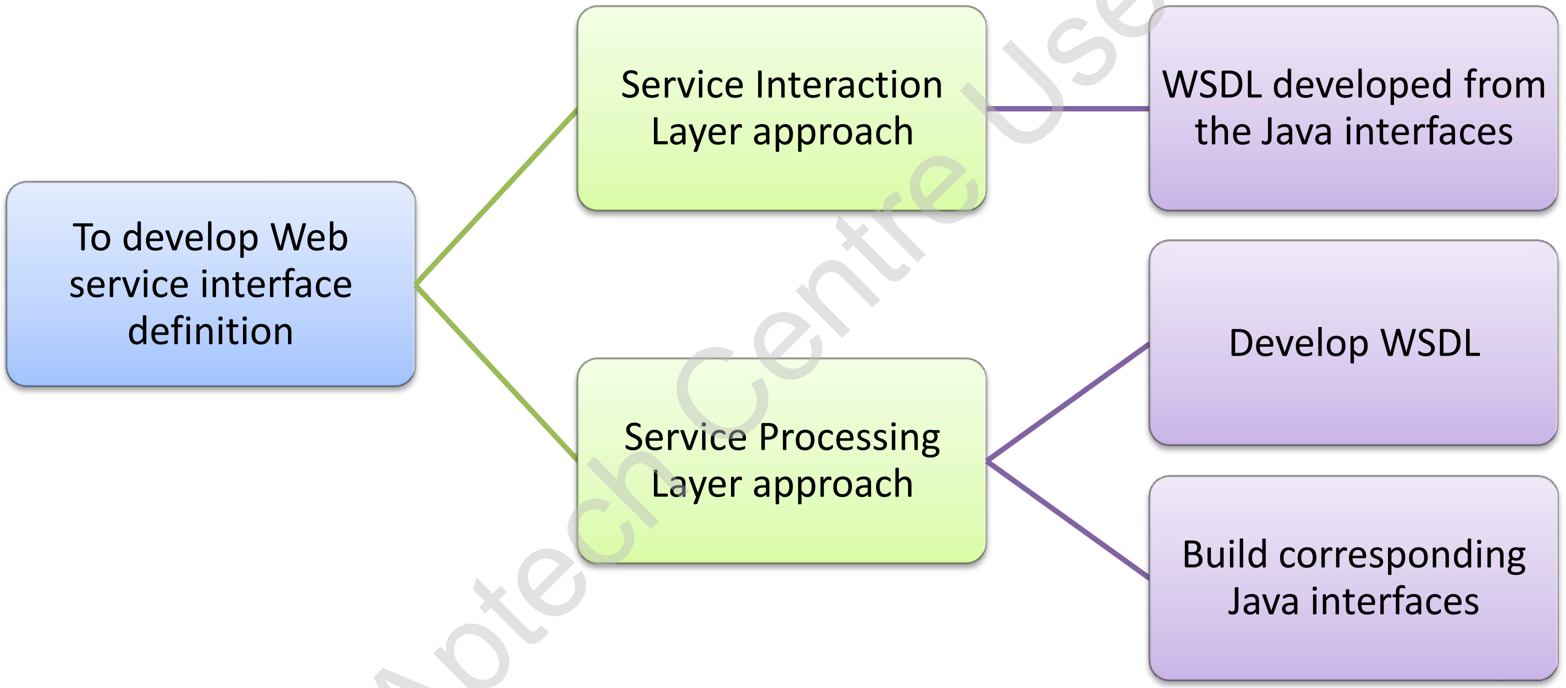
Layered View of Web Service 2-4

- Following figure shows the layered view of the Web service:



Implementing the Web service into a layered view helps to:		
Get clarity on the division of responsibilities	Designate a single location for all request processing logic	Describe existing business logic as a Web service

Layered View of Web Service 3-4


To develop Web service interface definition
Service Interaction Layer approach
WSDL developed from the Java interfaces
Service Processing Layer approach
Develop WSDL
Build corresponding Java interfaces

Layered View of Web Service 4-4

- Following are the factors that influence the design of the interface:

Choice of interface endpoint type

- JAX-RPC service endpoint used when the processing happens within the Web tier
- EJB service endpoint used when the processing happens on the EJB tier

Granularity of service

- A trade-off between client-side flexibility and Web service performance

Parameter types for Web service operations

- Mapping of call parameters and return values to Java objects, XML, or other types

Interfaces with overloaded methods

- Avoiding overloaded methods in WSDL descriptions

Other Design Considerations

The interaction layer receives client requests in the form of SOAP messages and delegates them to the Web service business logic.

There are other factors that influence the design of this layer.

Receiving requests

- On receiving requests, security checks, logging, auditing, and input validation are done

Delegating requests to processing layer

- Requests are processed synchronously or asynchronously

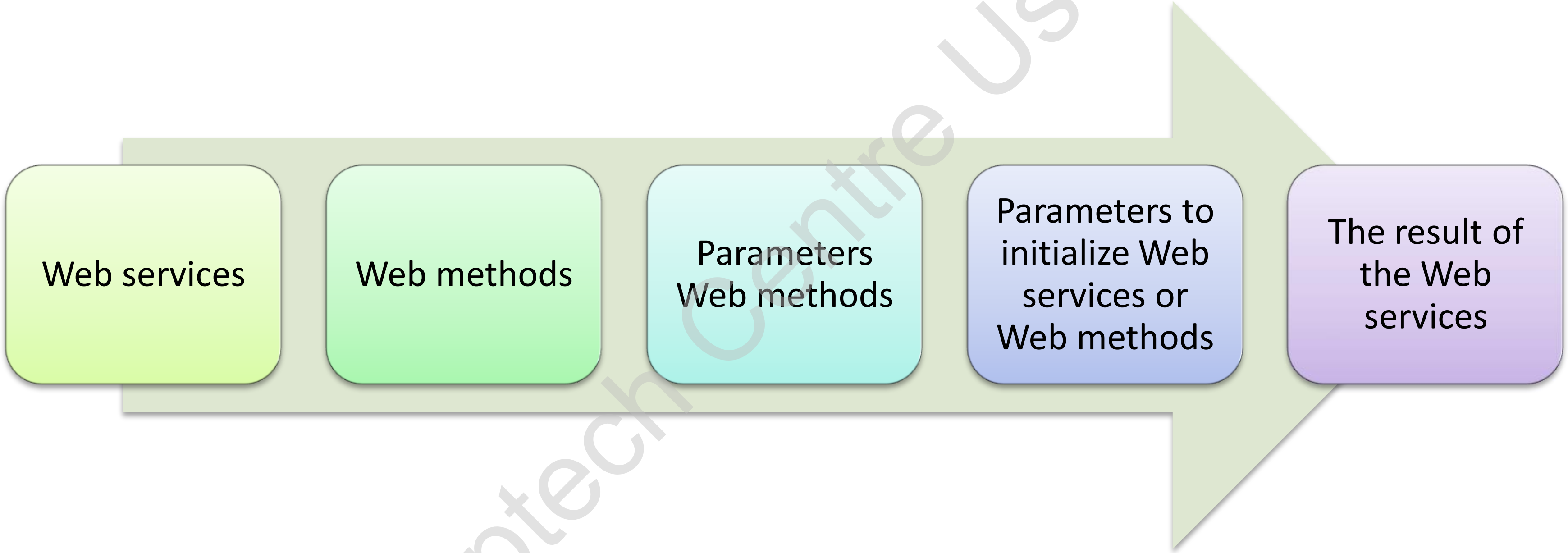
Formulating response

- Response to a method call comes as an XML document with return values

Web Service Annotations 1-2

For operations of a Web service, deployment descriptors and supporting files have to be packaged and deployed. These descriptors have been replaced with annotations in Java EE platform.

Web service annotations are modifiers that indicate the following:


Web services
Web methods
Parameters Web methods
Parameters to initialize Web services or Web methods
The result of the Web services

Annotations are prefixed with the @ symbol. JAX-WS 2.0 specification defines several annotations that can be used to define and use Web services. These annotations are available in the javax.jws package.

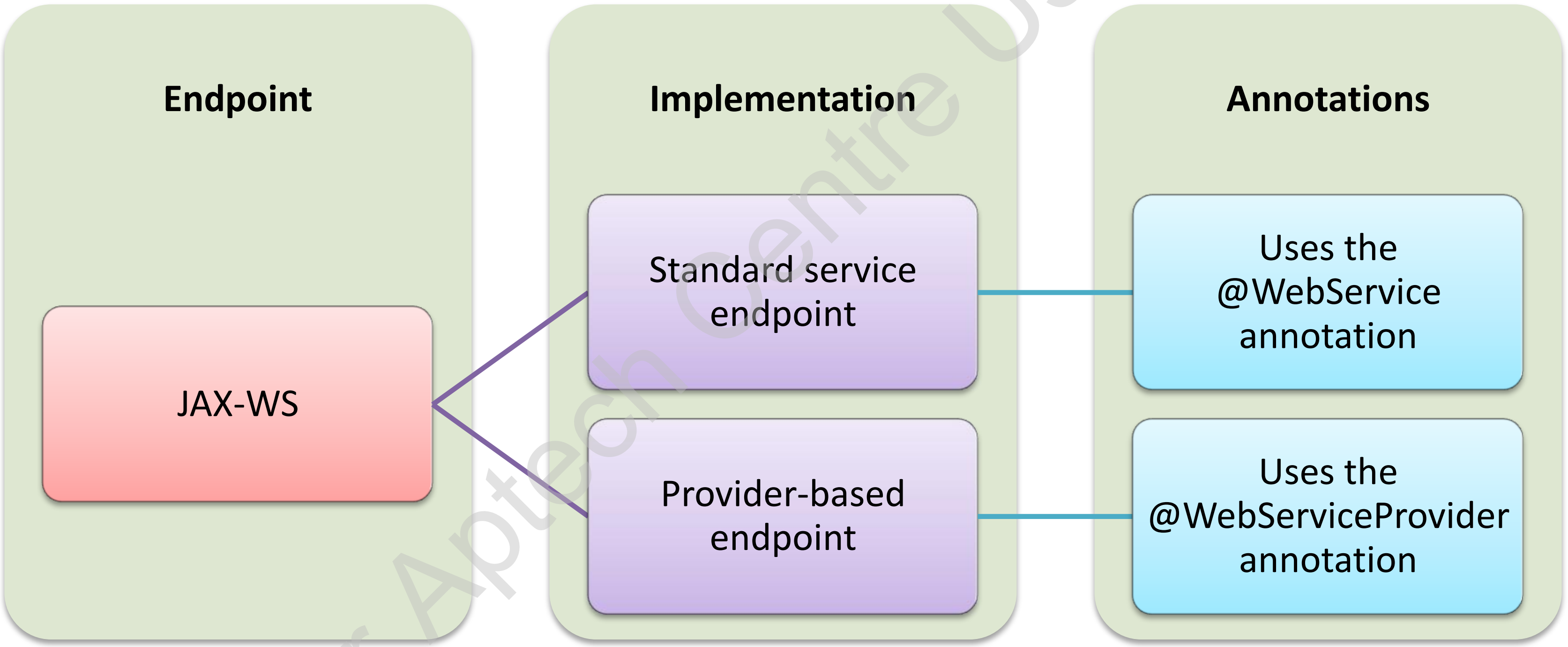
Web Service Annotations 2-2

Some of the annotations in the `javax.jws` package are as follows:

Annotations	Description
javax.jws.WebService (@WebService)	Specifies that the Java Web Service (JWS) file implements a Web service
javax.jws.WebServiceProvider (@WebServiceProvider)	Specifies that a Web service is provided in the Provider implementation class.
javax.jws.WebMethod (@WebMethod)	Specifies that the method is a public operation offered by the Web service
javax.jws.WebParam (@WebParam)	Specifies the parameters required by the Web service and the behavior of the parameters
javax.jws.WebResult (@WebResult)	Specifies the parameter that is returned by the Web service
javax.jws.soap.SOAPBinding (@SOAPBinding)	Specifies the mapping of the Web service with the SOAP message protocol
javax.jws.soap.SOAPMessageHandler (@SOAPMessageHandler)	Specifies a SOAP message handler in a `SOAPMessageHandler` array
javax.jws.soap.initParams (@initParams)	Specifies the array of name/value pairs that are passed to the handler during initialization

JAX-WS Endpoint 1-2

JAX-WS service endpoint is implemented by annotating Java classes. This procedure does not require WSDL files. The information in these files are specified by using the attributes of the annotations.



Endpoint
Implementation
Annotations
JAX-WS
Standard service endpoint
Provider-based endpoint
Uses the @WebService annotation
Uses the @WebServiceProvider annotation

JAX-WS requires generic service endpoint interfaces.

JAX-WS Endpoint 2-2

Implementing Service Endpoint Interface (SEI) is optional in JAX-WS endpoint.

- A JAX-WS Web service that does not have an associated SEI is regarded as having an implicit SEI.
- A JAX-WS that has an associated SEI is regarded as having an explicit SEI.

SEI-based endpoint

- endPointInterface attribute used to add reference
- Implicit SEI used in absence of this attribute
- @WebMethod annotation used to mark the methods exposed by the endpoint

Provider-based endpoint

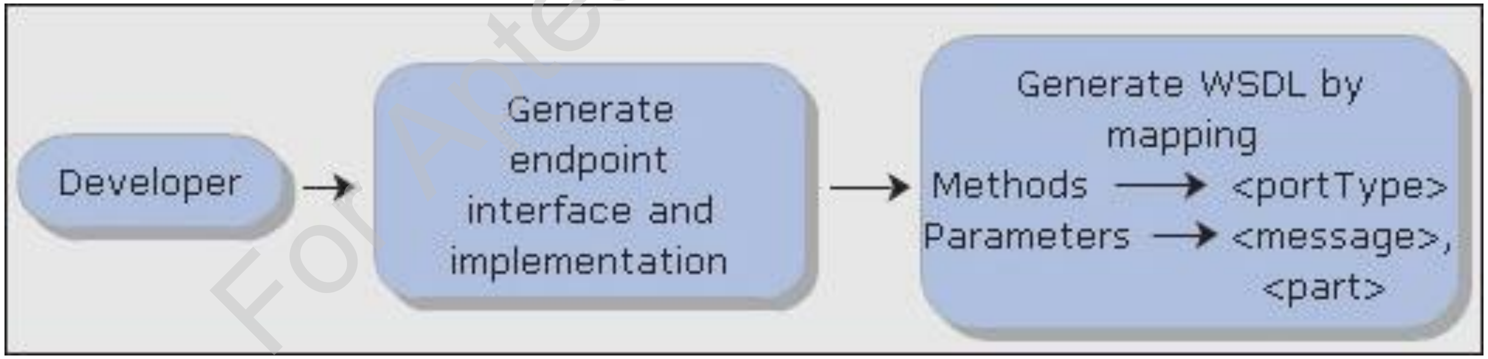
- A class used to implement javax.xml.ws.Provider interface
- Provider implementation returns a null value if WSDL file not specified
- No response required for null value

Deployment Process 1-2

The process of deployment of the Web service depends on the sequence of the two actions – developing WSDL and creating service implementation.

If the process is to first create service implementation and then develop WSDL, then:

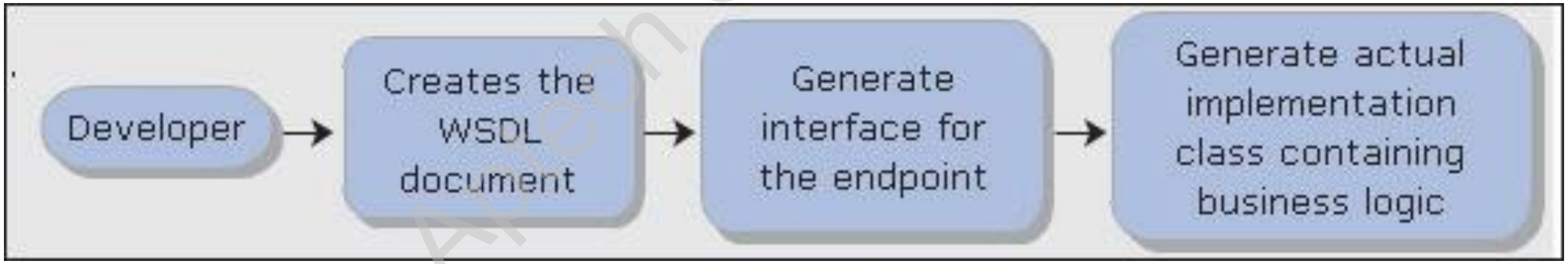
- JAX-WS is used to develop the endpoint interface and implementation
- JAX-WS guidelines used to create WSDL file
- Details on binding and service elements provided separately


Developer
Generate endpoint interface and implementation
Generate WSDL by mapping
Methods → <portType>
Parameters → <message>, <part>

Deployment Process 2-2

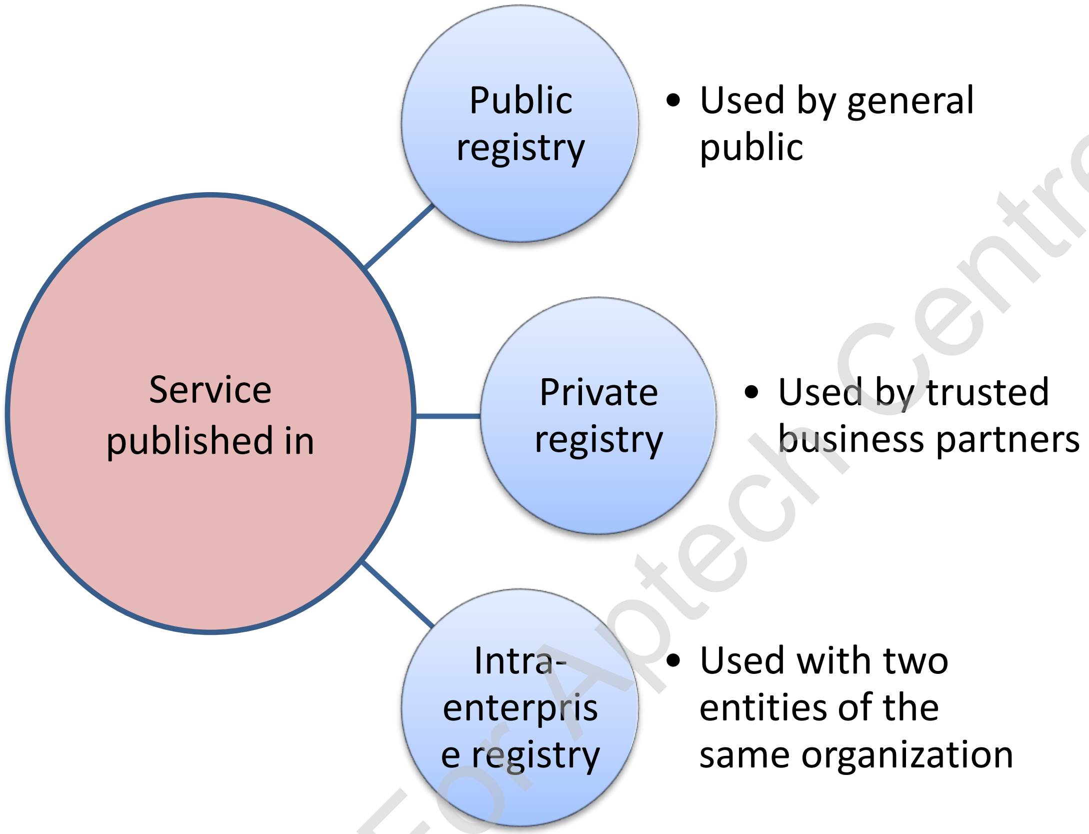
If the process is to first develop WSDL and then create service implementation:

- A WSDL document neutral in the XML types, idioms, and error handling capabilities created
- Code to marshall SOAP messages to endpoint invocations generated
- An implementation class for the endpoint generated
- Endpoint methods implemented


Developer
Creates the WSDL document
Generate interface for the endpoint
Generate actual implementation class containing business logic

Publishing Web Service 1-5

Publishing a Web service involves making the details about the Web service such as its interfaces, methods, parameters, and service location available to clients through a registry. The registry depends on the client.



Service published in
Public registry
Used by general public
Private registry
Used by trusted business partners
Intra-enterprise registry
Used with two entities of the same organization

Web services

- ❑ Description present in a WSDL document in the registry
- ❑ May hold XML schemas referenced by the service description in the registry
- ❑ Undeploying involves disabling and removing a service endpoint from the Web container, removing associated files, and freeing other server resources

Publishing Web Service 2-5

- Following code snippet demonstrates a simple Web service that takes two integer parameters and provides the sum of the two integers:

```
@WebService(serviceName = "CalculatorWS")
 public class CalculatorWS
{
   /**
     * Web service operation
     */
    @WebMethod(operationName = "add")
    public int add(@WebParam(name = "num1") int num1,
@WebParam(name = "num2") int num2)
{
      int sum = num1 + num2;
         return sum;
    }
 }
```

- ❑ CalculatorWS is the Web service name.
- ❑ num1 and num2 are integers given by user.
- ❑ sum is the variable that stores the value.

Publishing Web Service 3-5

- Following code snippet demonstrates the WSDL file of the CalculatorWS Web service:

```
This XML file does not appear to have any style information associated
with it. The document tree is shown here.

<!—

 Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is
Metro/2.3 (tags/2.3-7528; 2013-04-29T19:34:10+0000) JAXWS- RI/2.2.8
JAXWS/2.2 svn-revision#unknown.

 -->

 <!—

 Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is
Metro/2.3 (tags/2.3-7528; 2013-04-29T19:34:10+0000) JAXWS- RI/2.2.8
JAXWS/2.2 svn-revision#unknown.

 -->

<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/ oasis-
200401-wss-wssecurity-utility-1.0.xsd"xmlns:wsp="http:// www.w3.org/ns/ws-
policy" xmlns:wsp1_2="http://schemas.xmlsoap.
org/ws/2004/09/policy"xmlns:wsam="http://www.w3.org/2007/05/
addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/
wsdl/soap/"xmlns:tns="http://DJWS.com/" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/
wsdl/"targetNamespace="http://DJWS.com/" name="CalculatorWS">

<types>

<xsd:schema>

 <xsd:import namespace="http://DJWS.com/" schemaLocation="

 http://localhost:8080/CalculatorWS/CalculatorWS?xsd=1"/>

 </xsd:schema>

 </types>
```

Publishing Web Service 4-5

```
<message name="add">
 <part name="parameters" element="tns:add"/>
 </message>
 <message name="addResponse">
 <part name="parameters" element="tns:addResponse"/>
 </message>
<portType name="CalculatorWS">
 <operation name="add">
 <input wsam:Action="http://DJWS.com/CalculatorWS/
 addRequest" message="tns:add"/>
 <output wsam:Action="http://DJWS.com/CalculatorWS/
 addResponse" message="tns:addResponse"/>
 </operation>
</portType>
<binding name="CalculatorWSPortBinding" type="tns:CalculatorWS">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
```

- message element defines the messages mapped to the method invocation.
- portType element maps the add operation of the Web service to the input and output endpoints.
- binding element defines the protocols and data formats for the messages and the operations.

Publishing Web Service 5-5

```
<operation name="add">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
</binding> <service name="CalculatorWS">
<port name="CalculatorWSPort" binding="tns:
CalculatorWSPortBinding">
<soap:address location="http://localhost:8080/
CalculatorWS/
CalculatorWS"/>
</port>
</service>
</definitions>
```

❑ service element maps the binding to the port.

Web Service Invocation

Invoking a Web service refers to the actions that a client application performs to use the Web service. A client trying to access a Web service should:

Discover SDI

- Knowing the parameters required and the return types of Web service methods
- Determining method signatures

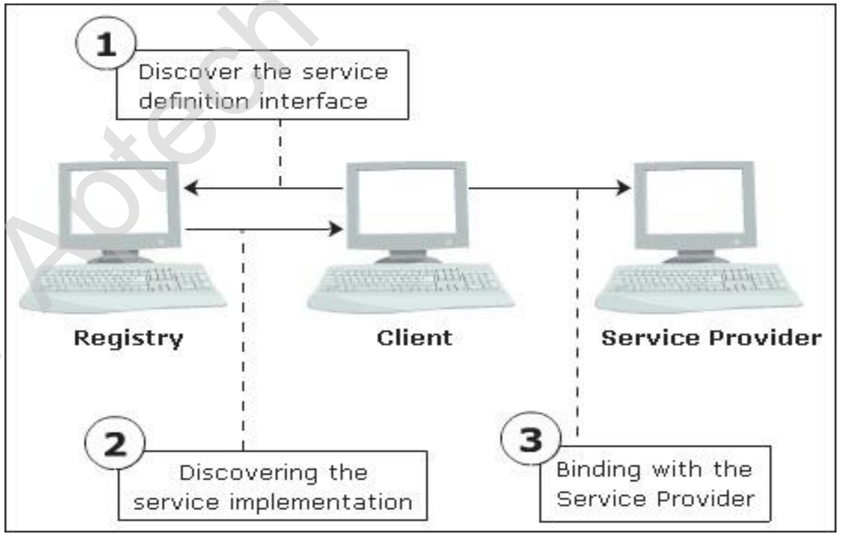
Discover Service Implementation

- Locating the address of Web service

Bind with Service Provider

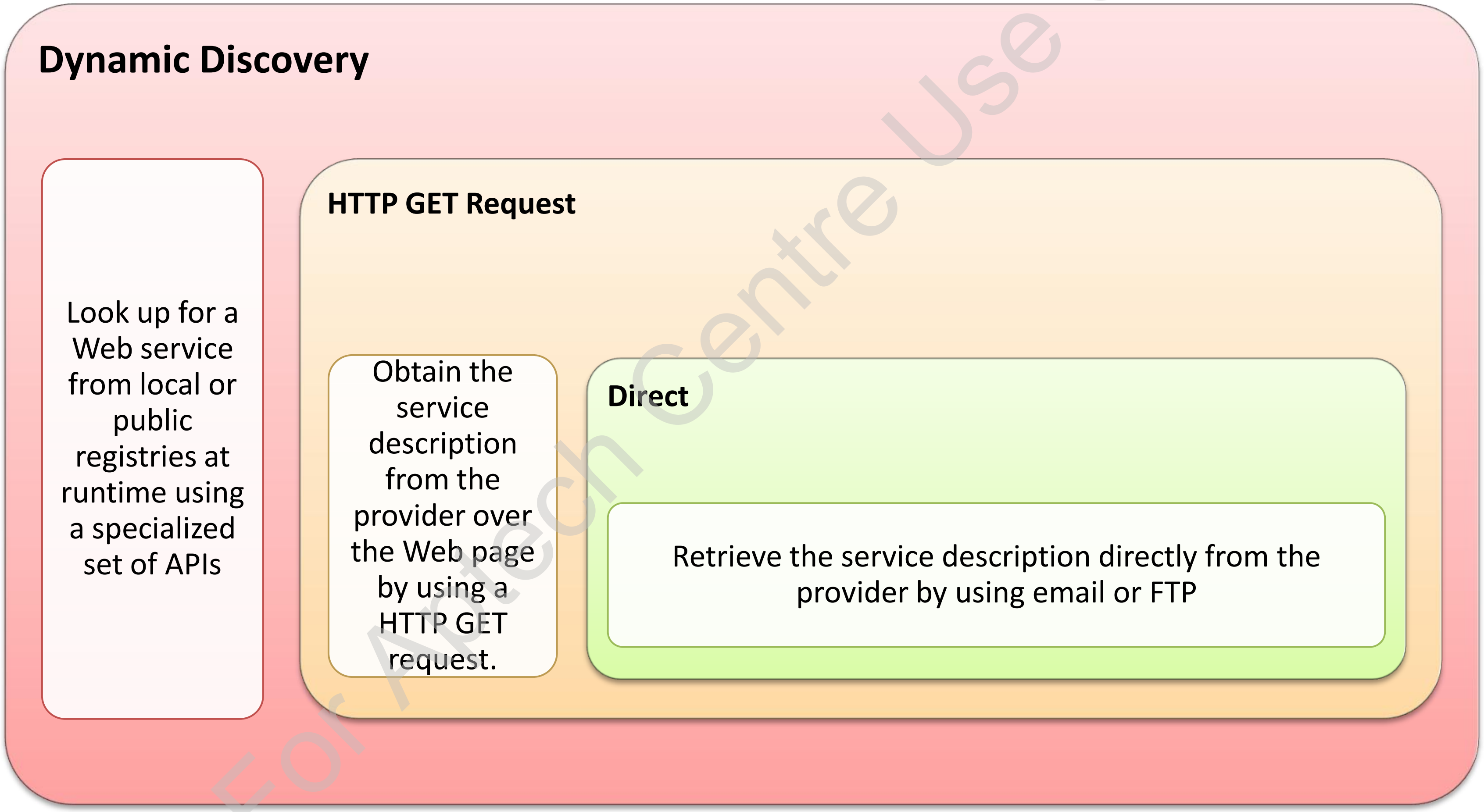
- Binding with a location to start invoking methods
- Binding when a client is developed, deployed or at the time of runtime

- Following figure shows Web service invocation process:


1
Discover the service definition interface
Registry
Client
Service Provider
2
Discovering the service implementation
3
Binding with the Service Provider

Discovering the Service Definition Interface (SDI)

There are three ways by which a client obtains the SDI from the service provider.



Dynamic Discovery

Look up for a Web service from local or public registries at runtime using a specialized set of APIs

HTTP GET Request

Obtain the service description from the provider over the Web page by using a HTTP GET request.

Direct

Retrieve the service description directly from the provider by using email or FTP

Discovering the Service Implementation

Web service clients query public or private registries in the form of XML messages

↓

Messages transmitted using standard protocols such as SOAP or XML-RPC

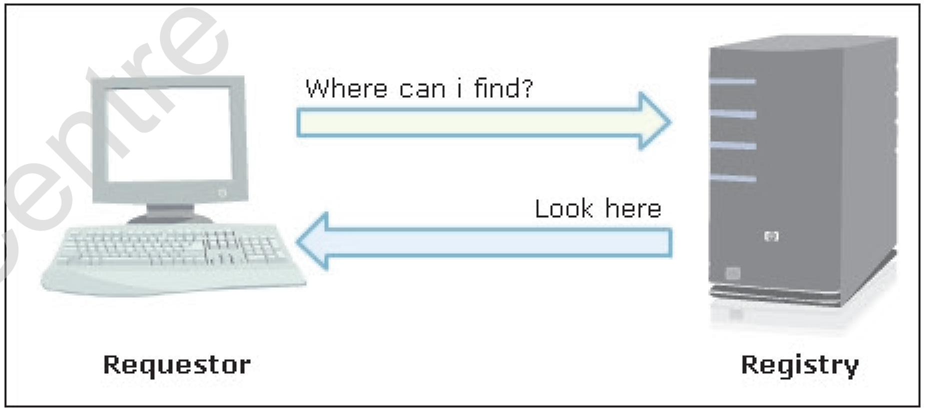
↓

Service finding criteria such as service response time, accuracy of results, and supported protocols implemented

↓

Appropriate service is located and actual location is sent to the client

- Following figure shows the process of discovering the service implementation:



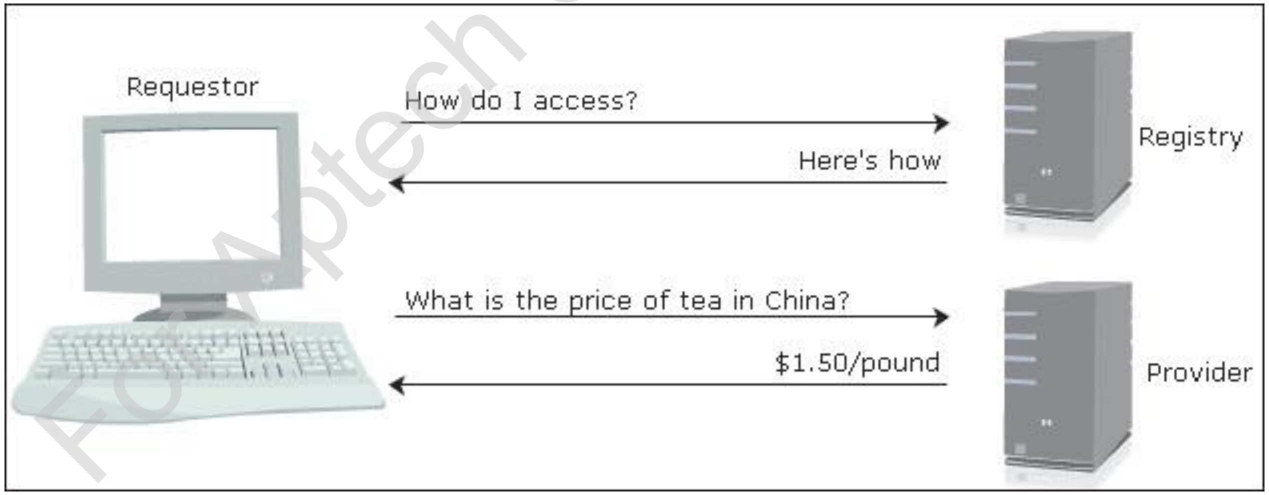
Binding with the Service Provider 1-3

After locating the service implementation, the client creates a message to be sent to the service provider

This message is sent to the provider by using the network protocols specified in the WSDL documents

The client of a Web service makes calls to the Web service using the API specified in the WSDL document

- Following figure shows the process of binding to a service:



Binding with the Service Provider 2-3

- Following code snippet demonstrates how to invoke a Web service using a JSP client:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>JAXWS Web Service Client </title>
    </head>
    <body>
        <h1>Accessing JAXWS Web Service CalculatorWS.</h1>
            <%-- start Web service invocation --%><hr/>
 <%
 try
{ //Instantiating the service and the port
com.djws.CalculatorWS_Service service = new
com.djws.CalculatorWS_Service();
com.djws.CalculatorWS port =
 service.getCalculatorWSPort();
```

- ❑ Instantiate the Web service using the Service() method of the Web service, CalculatorWS.

- ❑ Initialize the port for the Web service using the getPort() method of the Web service.

Binding with the Service Provider 3-3

```
// initializing WS operation arguments
     int num1 = 25;
     int num2 = 15;
// processing result
   int result = port.add(num1, num2);
   out.println("Result = "+result);
  }
 %> <%-- end Web service invocation --%><hr/>
 </body>
 </html>
```

- Invoke the Web service by calling the add() method of the Web service using the port instance.

Summary

- A Web service endpoint is a program that implements a Web service and carries out Web service requests.
- To design an efficient Web service, the developer needs to understand the nature of the service.
- The Web service designed should be dynamic to work in all the applications efficiently.
- A Web service is available to clients only after packaging the required files in the proper folders and deploying them on a server.
- The process of deployment of the Web service depends on the sequence of the development of WSDL and creation service implementation.
- Invoking a Web service refers to the actions that a client application performs to use the Web service.