

# **Developing Java Web Services**

# **Developing Java Web Services**

## **Trainer's Guide**

© 2014 Aptech Limited

All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.

**APTECH LIMITED**

Contact E-mail: [ov-support@onlinevarsity.com](mailto:ov-support@onlinevarsity.com)

First Edition - 2014



*Unleash your potential*



Dear Learner,

We congratulate you on your decision to pursue an Aptech Worldwide course.

**Aptech Ltd. designs its courses using a sound instructional design model – from conceptualization to execution, incorporating the following key aspects:**

- Scanning the user system and needs assessment

**Needs assessment is carried out to find the educational and training needs of the learner.**

**Technology trends are regularly scanned and tracked by core teams at Aptech Ltd. TAG\* analyzes these on a monthly basis to understand the emerging technology training needs for the Industry.**

**An annual Industry Recruitment Profile Survey<sup>#</sup> is conducted during August - October to understand the technologies that Industries would be adapting in the next 2 to 3 years. An analysis of these trends & recruitment needs is then carried out to understand the skill requirements for different roles & career opportunities.**

**The skill requirements are then mapped with the learner profile (user system) to derive the Learning objectives for the different roles.**

- Needs analysis and design of curriculum

**The Learning objectives are then analyzed and translated into learning tasks. Each learning task or activity is analyzed in terms of knowledge, skills and attitudes that are required to perform that task. Teachers and domain experts do this jointly. These are then grouped in clusters to form the subjects to be covered by the curriculum.**

**In addition, the society, the teachers, and the industry expect certain knowledge and skills that are related to abilities such as *learning-to-learn, thinking, adaptability, problem solving, positive attitude etc.* These competencies would cover both cognitive and affective domains.**

A precedence diagram for the subjects is drawn where the prerequisites for each subject are graphically illustrated. The number of levels in this diagram is determined by the duration of the course in terms of number of semesters etc. Using the precedence diagram and the time duration for each subject, the curriculum is organized.

- Design & development of instructional materials

**The content outlines are developed by including additional topics that are required for the completion of the domain and for the logical development of the competencies identified. Evaluation strategy and scheme is developed for the subject. The topics are arranged/organized in a meaningful sequence.**

**The detailed instructional material – Training aids, Learner material, reference material, project guidelines, etc.- are then developed. Rigorous quality checks are conducted at every stage.**

- Strategies for delivery of instruction

**Careful consideration is given for the integral development of abilities like thinking, problem solving, learning-to-learn etc. by selecting appropriate instructional strategies (training methodology), instructional activities and instructional materials.**

**The area of IT is fast changing and nebulous. Hence, considerable flexibility is provided in the instructional process by specially including creative activities with group interaction between the students and the trainer. The positive aspects of Web based learning –acquiring information, organizing information and acting on the basis of insufficient information are some of the aspects, which are incorporated, in the instructional process.**

- Assessment of learning

**The learning is assessed through different modes – tests, assignments & projects. The assessment system is designed to evaluate the level of knowledge & skills as defined by the learning objectives.**

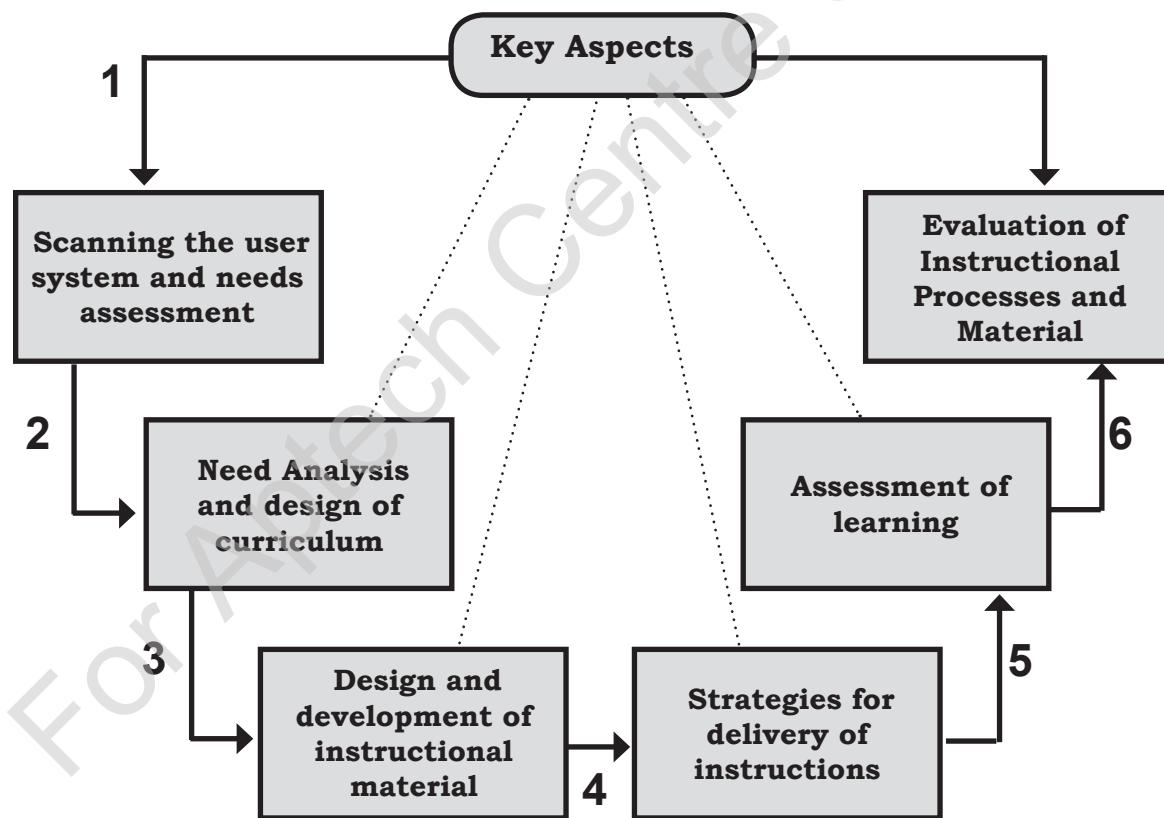
- Evaluation of instructional process and instructional materials

**The instructional process is backed by an elaborate monitoring system to evaluate - on-time delivery, understanding of a subject module, ability of the instructor to impart learning. As an integral part of this process, we request you to kindly send us your feedback in the reply pre-paid form appended at the end of each module.**

\*TAG – Technology & Academics Group comprises of members from Aptech Ltd., professors from reputed Academic Institutions, Senior Managers from Industry, Technical gurus from Software Majors & representatives from regulatory organizations/forums.

Technology heads of Aptech Ltd. meet on a monthly basis to share and evaluate the technology trends. The group interfaces with the representatives of the TAG thrice a year to review and validate the technology and academic directions and endeavors of Aptech Ltd.

### Aptech New Products Design Model



“

A little learning is a dangerous thing,  
but a lot of ignorance is just as bad

”

For Aptech Centres Only

---

## Preface

---

The book ‘Developing Java Web Services’ Trainer’s Guide’ serves understanding on the features and functionalities of Java EE 7 for creating Web services and Web service clients. The faculty/trainer should teach the concepts in the theory class using the slides. This Trainer’s Guide will provide guidance on the flow of the session and also provide tips and additional examples wherever necessary. The trainer can ask questions to make the session interactive and also to test the understanding of the students.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 Certification for Aptech-IT Division, Education Support Services. As part of Aptech’s quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

Design Team

“ Practice is the best of  
all instructors.

For Aptech Centre Use ”

## Table of Contents

### Sessions

1. Introduction to Web Services
2. SOAP, WSDL, and UDDI
3. Web Service Endpoints
4. Designing Web Service Clients
5. JAX-WS
6. RESTful Web Services

**“ The future depends on what  
we do in the present.**

For Aptech Centre Uday

## Session 1: Introduction to Web Services

### **1.1 Pre-Class Activities**

You should familiarize yourself with the concept of Web services. You should be able to explain the Service Oriented Architecture (SOA) and Java APIs for XML Web services (JAX-WS). You should be able to make learners understand how Java APIs can be used for XML Processing (JAXP). In addition, you should also be able to explain concepts of Java APIs for XML Registry (JAXR), SOAP with Attachment API for Java (SAAJ), and Java Architecture for XML Binding (JAXB).

Familiarize yourself with the topics of the current session in-depth.

#### **1.1.1 Objectives**

After the session, learners will be able to:

- Define Web services and describe their purpose
- Describe Service Oriented Architecture (SOA)
- Describe role of XML, SOAP, Registry standards, and WSDL in Web services
- Describe purpose of using JAXP in processing XML documents
- Explain parsing of XML document using SAX and DOM
- Describe JAX-WS
- Explain JAXR architecture, its components, and its interfaces
- Describe the purpose and features of SAAJ
- Describe purpose and limitations of JAXB and its components
- Explain the marshalling and unmarshalling processes

#### **1.1.2 Teaching Skills**

To teach this session, you should be well versed with the concept of Web services, SOA, JAXP, JAX-WS, SAAJ, and JAXB. You should know about the various Web service APIs of Java EE platform that support and implement Web services.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

#### **Tips:**

It is recommended that you test the understanding of the students by asking questions in between the class.

#### **1.1.3 In-Class Activities:**

Follow the order given here during In-Class activities.

#### **Review of Previous Session**

You should begin with a brief recap or review of previous session. Since this is the first session, you can skip doing this.

## Overview of the Session

Then, give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell the students that this session introduces them to the concept of Web services. Tell them that this session also explains in detail the SOA. Tell the students that this session introduces them to the purpose of using JAXP in processing XML documents and use of JAX-WS. It also introduces the students to the JAXR architecture and its components and its interfaces. Further, the session describes the purpose of SAAJ and JAXB.

## 1.2 In-Class Explanations

### Introduction to Web Services

#### Slide 3

**Introduction to Web Services**

A Web service is a service available on the Web. In other words, they are methods made available on the Web.

- ◆ Following figure shows the client-server architecture:

```

graph TD
    Client[Client] -- "ValidateCreditCard  
("3478-6787-5345-3768")" --> Server[Server]
    Server -- "true=Valid Number" --> Client
    Server -- "Credit card validation" --> CreditCardValidation
    Server -- "Email-id verification" --> EmailIdVerification

```

Introduction to Web Services/Session 1      3

Using slide 3, explain the meaning of Web service. Provide or ask a few examples where Web services are used. Tell them what made Web services highly successful. Explain with the help of the figure that Web services generally follow the client-server architecture.

**Characteristics of Web Services**

Slide 4

### Characteristics of Web Services

- ◆ Characteristics of Web services are as follows:

<b>Accessibility</b>	• A Web service is accessible over the Web.
<b>Communication Standards</b>	• Standard Web protocols interact with other Web services or application programs using XML to exchange information.
<b>Integration</b>	• Web services are loosely coupled and integrated only when required.

© Aptech Ltd. Introduction to Web Services/Session 1 4

Using slide 4, explain the main characteristics of a Web service, which are accessibility, communication standards, and integration.

**Uses of Web Services**

Slide 5

### Uses of Web Services

- ◆ Following figure shows the uses of Web services:

© Aptech Ltd. Introduction to Web Services/Session 1 5

Using slide 5, explain in detail about the uses of a Web service. Tell them that the Web services are generally used in Application-to-Application (A2A) integration or Business-to-Business (B2B) Communication. Explain how Web services are used in these applications.

Explain with an example for A2A integration that can be seen in a social media site such as Facebook or Twitter in which the application needs to share data with every other application. Explain that B2B communication happens between a company, its customers, and suppliers for business purposes such as request, purchase order, price, invoice, and payment.

### In-Class Question:



What is a Web service?

### Answer:

A Web service is a service available on the Web. In programming terms, Web service is a method made available on the Web.

### Advantages of Web Services

#### Slide 6

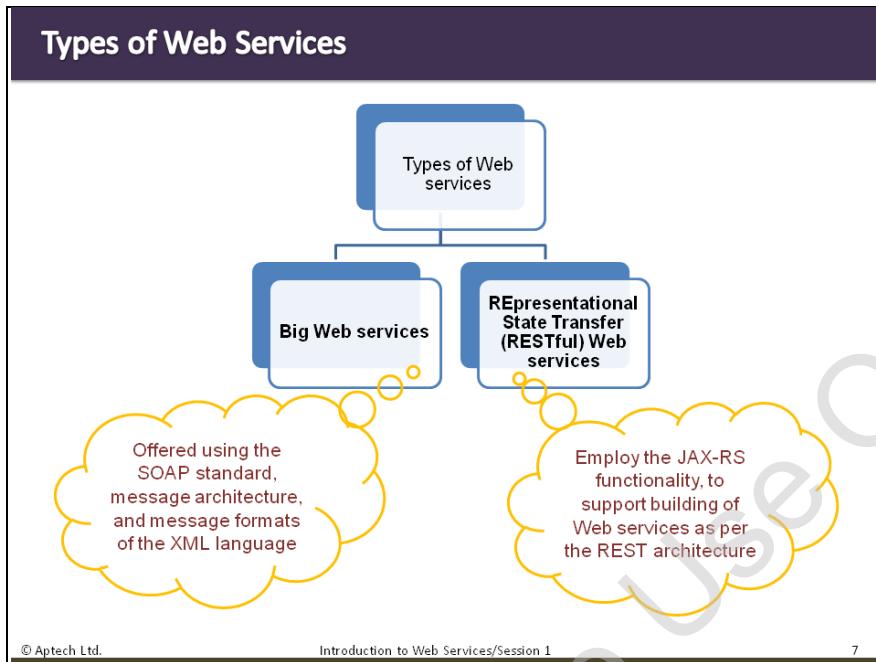
### Advantages of Web Services

- Freedom to use any platform and language for code development**
  - Developers can use Web services on any software platform, architecture, and programming language for development.
- Flexibility to reuse existing software code**
  - Web services allow enterprise application developers to use and reuse software code.
- Availability of wide variety of tools**
  - A large variety of tools is available for developing Web services.
- Wider market for services**
  - Extending applications and services as Web services avail organizations a vast range of clientele.

Using slide 6, discuss the advantages of Web services that have made them popular. Explain how Web services are beneficial to the users.

Types of Web Services

Slide 7



Using slide 7, explain the types of Web services, which are Big Web services and RESTful Web services. Describe the tools, such as NetBeans IDE that simplify the development of Web service applications. Explain WSDL and JAX-RS, which support Web services. Explain how the performance of RESTful Web services can be enhanced by leveraging the caching infrastructure.

Service Oriented Architecture

Slide 8

**Service Oriented Architecture (SOA)**

- Service Oriented Architecture (SOA) involves building an infrastructure that provides location and implementation transparency.
- Following table lists the components of a typical SOA:

Component	Description
Service Broker	Publishes information related organizations and their services and provides information on the accessibility and usage of the services provided.
Service Provider	Allows service consumers to use the services depending on the service cost, availability of the service, and security.
Service Consumer	Is used in an organization internally/externally. It makes use of a service broker to locate one or more services. Then, it connects to the service provider to make use of services.

© Aptech Ltd. Introduction to Web Services/Session 1 8

Using slide 8, explain Service Oriented Architecture (SOA). Describe location and implementation transparency. Also, explain the components of a typical SOA with the help of the table and what it is useful for.

### **SOA Implementation by Web Services**

Slides 9 and 10

### SOA Implementation by Web Services 1-2

Web service is created by service provider and hosted on the server. It uses XML to locate and implement transparency.

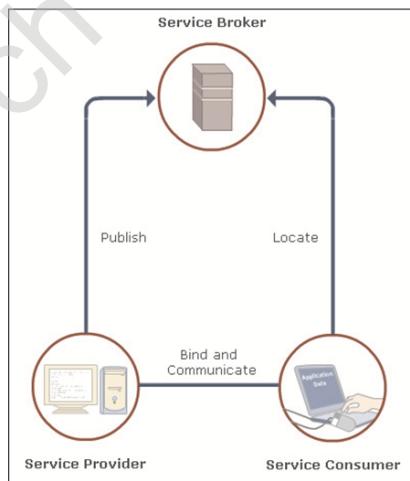
The information related to accessibility and usage is published in registries such as UDDI or ebXML, which serve as service brokers.

The service consumer identifies the required service and gets its information. Based on this information, the service consumer invokes the Web service that is placed on the service provider's server.

XML is used for communication and Hypertext Transfer Protocol (HTTP) is used as the communication protocol.

### SOA Implementation by Web Services 2-2

- ◆ Following figure illustrates the registry:



Using slides 9 and 10, explain how the Web services locate and implement SOA. Using slide 9 explain that the Service Provider registers the Web service in registries such as UDDI or ebXML and these registries serve as the Service Broker. The Service Consumer locates the Web service as per the requirements and invokes it. Using the figure on slide 10, explain the registry in SOA.

Mention that when an employee wants to know the number of leaves he/she can avail in a year. A service consumer can use the service published on the service broker to provide the number of leaves an employee can avail in a year.

## Web Service Standards

### Slide 11

### Web Service Standards

- ◆ Web service standards include the following:
  - ◆ Exchange information over the Web
  - ◆ Transfer data
  - ◆ Maintain registries
  - ◆ Ensure interoperability between different platforms

**Extensible Markup Language (XML)**

- The most common markup language to communicate over the Web.
- The most flexible language that allows applications to communicate irrespective of their geographic locations or software platforms.
- Helps to implement and execute Web services.

© Aptech Ltd.

Introduction to Web Services/Session 1

11

Using slide 11, explain the need for Web service standards. Explain in brief about XML, which is the most common markup language used to communicate over the Web. Inform that XML is very flexible. It is used to develop standards such as SOAP, WSDL, and UDDI, which enables communication between applications at different geographical locations or software platforms.

### In-Class Question:



What is XML?

### Answer:

XML is Extensible Markup Language which is used to communicate over the Web. It is the most flexible language that allows applications to communicate irrespective of their geographic locations or software platforms. It also helps to implement and execute Web services.

## Simple Object Access Protocol

### Slide 12

### Simple Object Access Protocol

Simple Object Access Protocol (SOAP) is a standard protocol that facilitates transfer of XML data among various applications.

- ◆ SOAP message (SOAP envelope) is an XML document.
- ◆ SOAP message has a header and a body that has message data enclosed in an envelope.
- ◆ Following figure shows the structure of SOAP message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope...>
...
<SOAP-ENV:Body>
<po:purchaseOrder
  xmlns:po="http://www.flamingo.com/books/PO">
  <po:name>Wiley</po:name>
  <po:orderdate>2007-09-09</po:orderdate>
  <po:book>
    <po:title>Professional Java User Interfaces</po:title>
    <po:quantity>100</po:quantity>
    <po:wholesale-price>30</po:wholesale-price>
  </po:book>
</po:purchaseOrder>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

© Aptech Ltd. Introduction to Web Services/Session 1 12

Using slide 12, explain SOAP. Describe the advantages of SOAP. Also, explain SOAP Message structure showing header and an envelope with the help of the illustration given in the figure. Mention that a SOAP message mandatorily consists of the Envelope and the Body element. The Envelope element defines the start and end of the message and the Body element contains the data being sent.

## Web Service Registries

### Slide 13

### Web Service Registries

- ◆ Every service provider is listed in the registry.
- ◆ Following figure depicts the Web service registries:

Web service registry is a service that allows service providers to publish services on the Web.

First, create a registry account and add information including name and service of service provider, Web service name, type of service, and contact information of the service provider.

Service Provider ID	Web Service Info
Service Provider ID	Web Service Info
Service Provider ID	Web Service Info
Service Provider ID	Web Service Info

© Aptech Ltd. Introduction to Web Services/Session 1 13

Using slide 13, explain that Web Service Registry is a service that allows service providers to publish their services on the Web. Explain what the user has to do to use the information given in the Web service registry. Show the figure on slide 13 and mention that every

service provider is listed in the registry. If a consumer needs a service, the entire registry is explored to locate the required service.



What is a Web service registry?

**Answer:**

Web service registry is a service that allows service providers to publish services on the Web.

### Web Service Description Language

Slide 14

### Web Service Description Language

- ◆ Web services can be used by first retrieving a document (written in WSDL) from the registry.

**WSDL Document Information**

- Web service location that refers to the WSDL file
- Parameters that have to be passed to the methods
- Methods provided by the Web service
- Syntax to invoke method
- Return type of methods

- ◆ WSDL need not be coded manually as Integrated Development Environments (IDEs) automatically generate the document.

- ◆ Following figure depicts the location of WSDL file:

© Aptech Ltd.      Introduction to Web Services/Session 1      14

Using slide 14, explain WSDL. Explain that it is a document written in WSDL that can be retrieved from the registry. Explain the information contained in the document, which are listed as follows:

- Web service location that refers to the WSDL file
- Parameters that have to be passed to the methods
- Methods provided by the Web service
- Syntax to invoke method
- Return type of methods

Additionally, explain the location of the WSDL document using the illustration given in the figure on slide 14.

**In-Class Question:**



What is WSDL?

**Answer:**

WSDL is a language that is used to describe Web services and how to access the Web service. Generally, the WSDL is written in XML.

**Java EE 7 Web Services Framework****Slide 15**

Java EE 7 Web Services Framework	
Web Service API	Description
Java API for XML-based Web services (JAX-WS)	It enables communication with Java and non-Java Web services.
SOAP with Attachments API for Java (SAAJ)	It helps read, manipulate, create and transmit SOAP messages, and process SOAP header blocks in JAX-RPC. It complies with SOAP 1.1 and the SOAP Messages with Attachments specification.
Java API for XML Registries (JAXR)	It allows an application to access registries such as UDDI or ebXML. It simplifies publishing and querying of Web services.
Java API for XML Processing (JAXP)	It enables users to use DOM2 and SAX2 interfaces. Apart from this API, standard Java APIs can also read, write, and modify XML documents.

Using slide 15, explain Web service APIs in the J2EE platform which are JAX-WS, SAAJ, JAXR, and JAXP and the use of each.

**In-Class Question:**

List the different Java Web services APIs.

**Answer:**

- Java API for XML-based Web services (JAX-WS)
- SOAP with Attachments API for Java (SAAJ)
- Java API for XML Registries (JAXR)
- Java API for XML Processing (JAXP)

**Benefits of Using the Java EE Platform****Slide 16**

**Benefits of Using the Java EE Platform**

The Java EE platform helps to:

Improve application development using component-based model	Take the support of Web service standards and WS-I basic profile	Make portable applications and interoperable services	Expand distributed applications with little effort	Declare component security requirements
---	--	---	--	---

© Aptech Ltd.      Introduction to Web Services/Session 1      16

Using slide 16, explain the benefits of using the Java EE Platform.

**Web Services Technologies of Java EE 7****Slide 17**

**Web Services Technologies of Java EE 7**

- ◆ Following table lists the Web Services Technologies of Java EE 7:

Web Service Technology	JSR	Description
Java API for RESTful Web Services (JAX-RS) 2.0	339	Assists in the creation of an API that supports RESTful Web services in the Java Platform.
Implementing Enterprise Web Services 1.3	109	Defines the programming model and runtime architecture to implement Web services in Java.
Java API for XML-based Web Services (JAX-WS) 2.2	224	Is the next generation Web services API replacing JAX-RPC 1.0.
Web Services Metadata for the Java Platform	181	Defines an annotated Java format that uses Java Language Metadata (JSR 175).
Java API for XML-based RPC (JAX-RPC) 1.1 (Optional)	101	Supports emerging industry XML-based RPC standards.
Java APIs for XML Messaging 1.3	67	Helps to package and transport business transactions. It uses on-the-wire protocols defined by ebXML.org, Oasis, W3C, and IETF.
Java API for XML Registries (JAXR) 1.0	93	Assists a set of distributed Registry Services to enable business-to-business integration between enterprises. It uses the protocols defined by ebXML.org, Oasis, and ISO 11179.

© Aptech Ltd.      Introduction to Web Services/Session 1      17

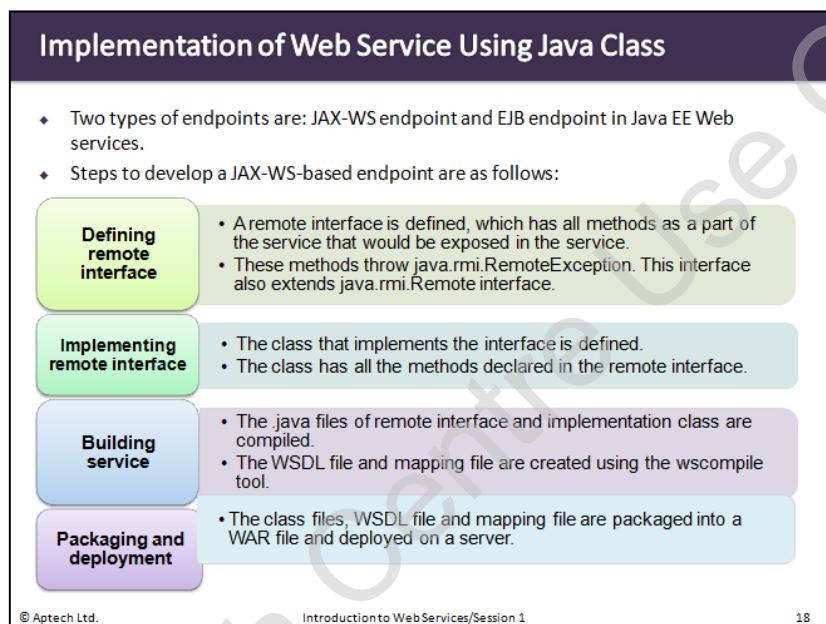
Using slide 17, explain the Web Service Technologies of Java EE 7 and the uses of each.

**In-Class Question:**

What is use of JAXR 1.0?

**Answer:**

It assists a set of distributed Registry Services to enable business-to-business integration between enterprises. It uses the protocols defined by ebXML.org, Oasis, and ISO 11179.

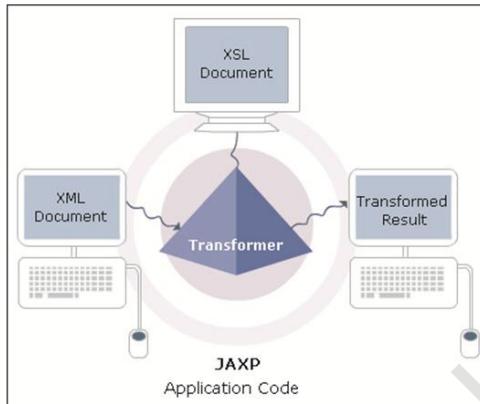
**Implementation of Web Service Using Java Class****Slide 18**

Using slide 18, explain that there can be two types of endpoints namely, JAX-WS endpoint and EJB endpoint in Java EE Web services. Then, explain the four steps to develop JAX-WS based endpoint, which are:

- Defining remote interface
- Implementing remote interface
- Building service
- Packaging and deployment

**JAXP****Slides 19 and 20****JAXP 1-2**

- ◆ JAXP uses Extensible Stylesheet Language Transformation (XSLT) engines to transform XML documents from one format to another.
- ◆ Following figure shows processing of XML documents using JAXP:



© Aptech Ltd.

Introduction to Web Services/Session 1

19

**JAXP 2-2**

- ◆ Following table lists the packages that define the JAX API:

Package	Description
javax.xml.parsers	Defines SAXParserFactory and DocumentBuilderFactory classes, which provide a common interface for SAX and DOM parsers.
javax.xml.transform	Defines the generic APIs to transform source to result.
javax.xml.transform.dom	Implements DOM-specific transformation APIs.
javax.xml.transform.sax	Implements SAX2-specific transformation APIs.

© Aptech Ltd.

Introduction to Web Services/Session 1

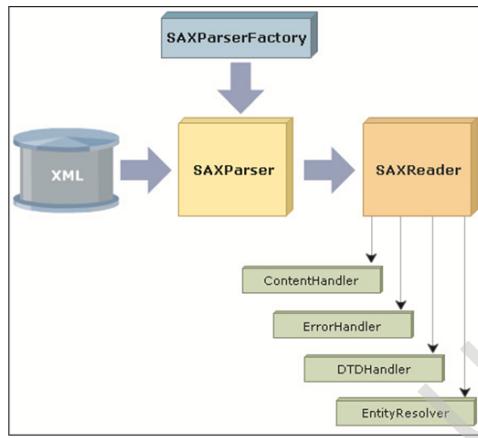
20

Using slides 19 and 20, explain JAXP in brief. Explain the two Java standards that are used to parse XML documents, which are Simple API for XML (SAX) or Document Object Model (DOM) Java standards.

Inform that these APIs can read, write, and modify XML documents by using XML parser. Then, explain processing of XML documents using JAXP using the figure on slide 19. Mention that XSLT engine is used to output the transformation result as a XML document. Explain that a package is a namespace that is attributed to the classes based on category or functionality. Using slide 20, explain the packages that define the JAX API and the use of each.

**SAX Parser****Slides 21 and 22****SAX Parser 1-2**

- ◆ SAX processes the XML documents sequentially into a series of events.
- ◆ SAX is very fast as it does not load the document into memory.
- ◆ The SAX parser is depicted in the following figure:



© Aptech Ltd.

Introduction to Web Services/Session 1

21

**SAX Parser 2-2**

- ◆ Following table lists the key classes/interfaces to parse XML documents:

Package	Description
DefaultHandler	Is present in the <code>org.xml.sax.helpers</code> package. It acts as a default base class for SAX2 event handlers.
SAXParserFactory	Is present in the <code>javax.xml.parsers</code> package. It returns the SAXParser and the exception classes to report errors.
SAXParser	Is present in the <code>javax.xml.parsers</code> . It defines the API that wraps an XMLReader implementation class.
XMLReader	Is present in the <code>java.io</code> package. It is an interface to read an XML document by using callbacks.

© Aptech Ltd.

Introduction to Web Services/Session 1

22

Using slides 21 and 22, explain SAX Parser. Explain that SAX processes the XML documents sequentially. Show the figure of SAX Parser on slide 21 and mention about the relationship between the various components used while parsing an XML document with a SAX parser. SAXParserFactory class is used to generate an instance of the SAX parser. The parser contains a SAXReader object. When the parser's parse() method is invoked, the reader invokes one of the many callback methods implemented in the application. Those methods are defined by the interfaces ContentHandler, ErrorHandler, DTDHandler, and EntityResolver.

Using slide 22, explain the key classes/interfaces that are used to parse XML documents that are listed as follows:

- DefaultHandler
- SAXParserFactory
- SAXParser
- XMLReader

While explaining the key classes, point out the package in which each class is present and what it is useful for.

**In-Class Question:**



List any one advantage of SAX Parser.

**Answer:**

SAX is very fast as it does not load the document into memory.

**SAX Parser Implementation**

Slides 23 and 24

**SAX Parser Implementation 1-2**

- ◆ Parsing an XML document is done with the help of JAXP APIs and SAX.
- ◆ Following Code Snippet shows how to parse an XML document using a SAX-based parser:

```
//Step 1
public class SAXParsing extends DefaultHandler {

    public void readDocument(){
        //Step 2
        SAXParserFactory spFactory = SAXParserFactory.newInstance();
        spFactory.setValidating (true);
        //Step 3
        SAXParser saxp = spFactory.newSAXParser();
        //Step 4
        XMLReader xReader = saxp.getXMLReader();
        //Step 5
        xReader.setContentHandler(this);
        //Step 6
        xReader.parse (XMLDocument);
    }
}
```

## SAX Parser Implementation 2-2

- ◆ Steps performed by the code are as follows:

- 1 • A class named SAXParsing is created. This class extends the DefaultHandler class.
- 2 • An instance of the SAXParserFactory class, named spFactory, is created, and the validation property of this instance is set to true.
- 3 • An instance of the SAXParser class, named saxp, is obtained from the SAXParserFactory instance.
- 4 • The encapsulated SAX XMLReader is retrieved to read character streams.
- 5 • The ContentHandler property of XMLReader is set to allow the application to register a content event handler.
- 6 • The XML document is parsed using the instance of the XMLReader interface.

Using slides 23 and 24, explain the SAX Parser implementation by using first the Code Snippet on slide 23. Using slide 24, explain each of the given steps with reference to the Code Snippet.

## DOM Parser

### Slides 25 to 27

#### DOM Parser 1-3

A DOM parser accesses XML documents randomly and creates a tree from its elements. It splits the documents into more memory, but is easy to create and modify XML documents by using in-memory content tree.

- ◆ Following table lists the packages in DOM parser:

Package	Description
org.w3c.dom	Defines DOM programming interfaces for XML documents. Its primary interfaces are Document and Node. Document is a HTML/XML document and Node is the primary data type for DOM.
javax.xml.parsers	Defines the DocumentBuilder class and DocumentBuilderFactory class to process XML documents.

- ◆ Parsing an XML document can also be done with the help of JAXP APIs and DOM.

## DOM Parser 2-3

- Following Code Snippet shows how to parse an XML document using DOM-based parser:

```
...
public void readDocument() {
// step 1
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
dbFactory.setValidating(true);
// step 2
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
// step 3
Document doc = dBuilder.parse(XMLDocument);
// parse the tree created - node by node
}
}
```

## DOM Parser 3-3

- Steps performed by the code are as follows:



- A class named DOMParsing is created for parsing an XML document. An instance of the DocumentBuilderFactory class named dbFactory is created and the validation property of this instance is set to true.
- A DocumentBuilder object named dBuilder is created using an instance of DocumentBuilderFactory class.
- The instance of the DocumentBuilder class parses the input file by invoking the parse method and passing the document to be parsed named XMLDocument.

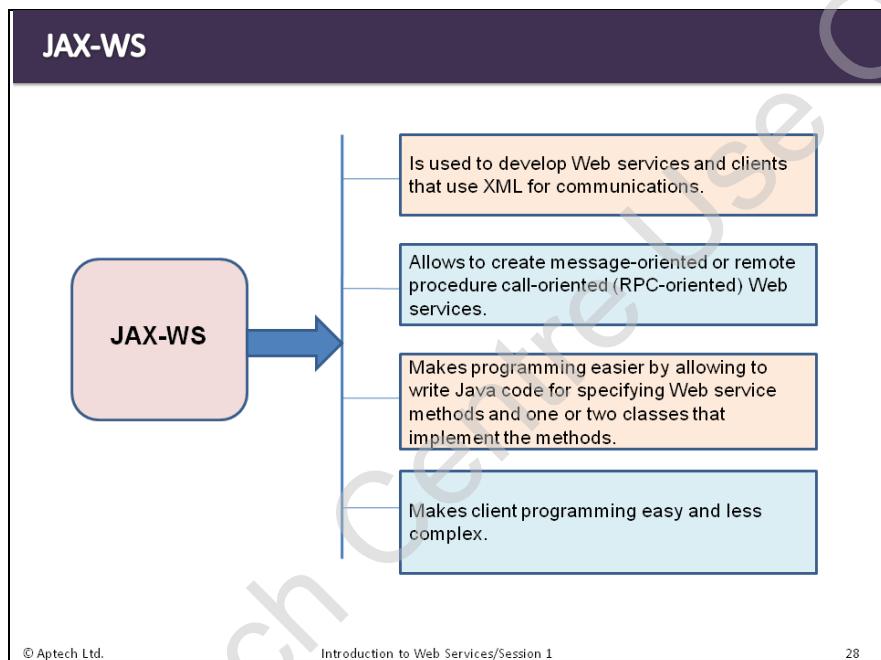
Using slides 25 to 27, explain the DOM Parser and its advantages. Explain the two packages in the DOM Parser, which are org.w3c.dom and javax.xml.parsers. Explain that org.w3c.dom defines DOM programming interfaces for XML documents and javax.xml.parsers defines the DocumentBuilder class and DocumentBuilderFactory class to process XML documents. Also, explain how parsing an XML document can be done. Explain the steps to parse an XML document using a DOM-based parser given on slide 27 with reference to the Code Snippet on slide 26.

**In-Class Question:**

What is DOM Parser?

**Answer:**

A DOM parser accesses XML documents randomly and creates a tree from its elements. It splits the documents into more memory, but is easy to create and modify XML documents by using in-memory content tree.

**JAX-WS****Slide 28**

Using slide 28, explain JAX-WS. Describe the characteristics of JAX-WS and the advantages provided by them. Explain how JAX-WS makes programming easier by:

- Enabling Java code to specify Web service methods and one / two classes that implement these methods
- Enabling JAX-WS client programs to create a proxy / local object for the Web service to utilize and invoke the Web service methods on the proxy
- Making client programming very easy and less complex
- Generating SOAP messages for the API calls and parsing them from responses, by which, code need not be written to generate or parse SOAP messages

**Role of JAX-WS in Web Services**

Slide 29

**Role of JAX-WS in Web Services**

Tasks performed by JAX-WS in a Web service are as follows:

Interoperates with SOAP-based Web services using WSDL	Maps the data types between XML and Java	Invokes methods on the generated stubs	Supports standard Internet protocols such as HTTP	Enables portability of service endpoints and service clients across JAX-WS implementations
---	--	--	---	--

◆ Following figure shows the role of JAX-WS in Web services:

© Aptech Ltd  
Introduction to Web Services/Session 1  
29

Using slide 29, explain the different tasks performed by JAX-WS in a Web service.

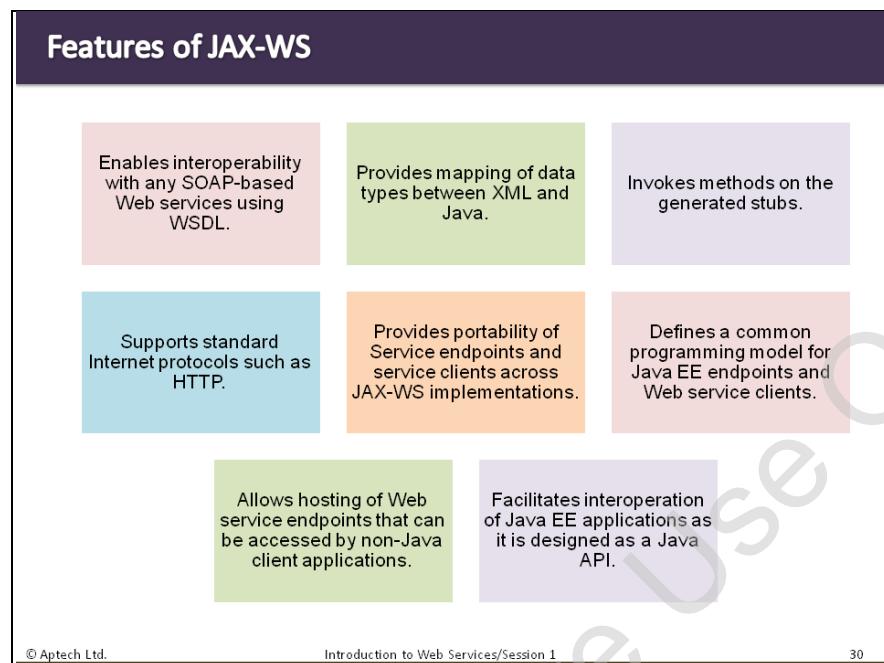
**In-Class Question:**

List any one task performed by JAX-WS in a Web Service.

**Answer:** JAX-WS Maps the data types between XML and Java.

## Features of JAX-WS

### Slide 30



© Aptech Ltd.

Introduction to Web Services/Session 1

30

Using slide 30, explain how JAX-WS functions to communicate even with non-Java platforms. Explain the key features of JAX-WS frame work.

**Benefits of Using JAX-WS**

Slide 31

**Benefits of Using JAX-WS**

- Standardization of SOAP requests and responses.
- Standardization of parameter marshalling and unmarshalling.
- Simplification of developers' role as SOAP creation and marshalling/unmarshalling tasks are offered through a library/tool.
- Support for different mapping scenarios, such as XML to Java, Java to XML, WSDL to Java, Java to WSDL, and WSDL/XML and Java.

© Aptech Ltd. Introduction to Web Services/Session 1 31

Using slide 31, explain that JAX-WS reduces the complexity for developers. Explain the key benefits of using JAX-WS.

**Client Requests to JAX-WS**

Slides 32 to 34

**Client Requests to JAX-WS Service 1-3**

- ◆ Steps performed by the client application to use a Web service are as follows:

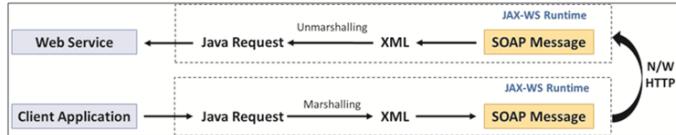
- 1 • The request from client is passed through client-side JAX-WS runtime.
- 2 • The runtime maps the request to XML and converts it to a SOAP form.
- 3 • The message is then sent to a server on the network.
- 4 • At the server, the SOAP message is received by a JAX-WS runtime.
- 5 • The runtime applies the XML to Java mapping and maps the request with its parameters to the corresponding Java method.

- ◆ To make a request, the client application can use the pre-created static classes or classes/interfaces generated at runtime.

© Aptech Ltd. Introduction to Web Services/Session 1 32

### Client Requests to JAX-WS Service 2-3

- Following figure depicts the client requests to JAX-WS service:



- Following Code Snippet shows how to implement JAX-WS:

```

import javax.jws.WebService;
import javax.jws.WebMethod;
@WebService
public class Welcome{
private final String message1 = "Welcome, ";
private final String message2 = "to WebService Course.";
public void Welcome(){}
@WebMethod
public String sayWelcome(String name){
return message1 + name + message2;
}}
    
```

### Client Requests to JAX-WS Service 3-3

- The code displays a welcome message to the users who call the Web service.
- Following two annotations have been used in the Code Snippet:

#### @WebService

- The attributes are: name, targetNamespace, serviceName, wsdlLocation, and endpointInterface.
- All these attributes are optional.

#### @WebMethod

- The attributes are: operationName and action.
- These attributes are optional.

Using slides 32 to 34, explain the steps performed by the client application to use a Web service. Also, explain the client requests to JAX-WS with the help of the figure on slide 33. Use the Code Snippet on slide 33, to implement JAX-WS for displaying a welcome message to the users who call the Web service. Explain the steps that show implementation of JAX-WS. Also, using slide 34, explain the two annotations used in the Code Snippet namely, @WebService and @WebMethod and their attributes.

**Registry Standards**

Slide 35

## Registry Standards

A registry is a place where interfaces are published by the service to make the clients know about the service.

XML-based registries have standards for registering, deregistering, and looking up Web services across different platforms, systems, and languages.

The most popular registries are the Universal Description, Discovery, and Integration (UDDI) and the electronic business XML (ebXML) registries.

© Aptech Ltd. Introduction to Web Services/Session 1 35

Using slide 35, explain registry and the requirements of an XML-based registry. Then explain the uses of XML-based registries. Next, inform that these registries have certain standards for registering, deregistering, and looking up Web services across different platforms, systems, and languages. List the most popular registries such as UDDI and ebXML.

**In-Class Question:**

What are the most commonly used registries?

**Answer:** UDDI and ebXML are the most commonly used registries.

**UDDI Registry Standards**

Slides 36 and 37

### UDDI Registry Standards 1-2

UDDI has a standard mechanism to store information about organizations and their Web services.

- Following figure depicts the UDDI registry:

```

graph TD
    subgraph "Web Service Provider"
        direction TB
        A[Icon: Computer and Phone]
        B[Icon: Hand holding a tablet]
        C[Icon: Database]
        A --> B
        B --> C
        C --> A
    end
    subgraph "Web Service Requestor"
        D[Icon: Computer]
        E[Icon: Hand holding a tablet]
        F[Icon: Database]
        D --> E
        E --> F
        F --> D
    end
    subgraph "UDDI Registry"
        G[Icon: Folders]
        H[Icon: Computer]
        G --> H
        H --> G
    end
    A --> G
    G --> B
    B --> C
    C --> A
    D --> G
    G --> E
    E --> F
    F --> D
    style A fill:#f0e68c,stroke:#800000
    style B fill:#f0e68c,stroke:#800000
    style C fill:#f0e68c,stroke:#800000
    style D fill:#f0e68c,stroke:#800000
    style E fill:#f0e68c,stroke:#800000
    style F fill:#f0e68c,stroke:#800000
    style G fill:#f0e68c,stroke:#800000
    style H fill:#f0e68c,stroke:#800000

```

© Aptech Ltd. Introduction to Web Services/Session 1 36

### UDDI Registry Standards 2-2

- Following table describes the four core elements in the UDDI information model:

Element	Description
businessEntity	Describes all information including the name, description, and contact details of a business.
businessService	Groups related services of an organization.
bindingTemplate	Provides instructions to invoke a remote Web service.
tModel	Contains information such as name, publishing organization, and URL pointers to the actual specifications themselves.

© Aptech Ltd. Introduction to Web Services/Session 1 37

Using slides 36 and 37, explain UDDI registry and for what it is used. Using the figure on slide 36 explain about UDDI registry. Mention that a service provider publishes its services to the UDDI registry. A service requestor in need of a service looks up for the service in the registry and receives the service binding information. The service provider then uses the binding information to invoke the service.

Using slide 37, explain the four key elements of UDDI information model and what each one is useful for.

**In-Class Question:**

What does a businessEntity element of the UDDI information model do?

**Answer:**

It describes all information including the name, description, and contact details of a business.

**ebXML Registry Standards**

Slide 38

### ebXML Registry Standards

The Electronic Business XML (ebXML) is a business-to-business XML framework, which facilitates electronic business over the Internet.

ebXML registry	ebXML information model
<ul style="list-style-type: none"> <li>❑ Is a metadata registry and a repository that can hold arbitrary content.</li> <li>❑ Has repository that can have metadata, technical specifications, and related artifacts.</li> <li>❑ Stores information about Collaboration-Protocol Profile (CPP) and Collaboration-Protocol Agreement (CPA).</li> </ul>	<ul style="list-style-type: none"> <li>❑ Enables data validation for improved integrity of registry data.</li> <li>❑ Facilitates packaging (or grouping) of related registry objects.</li> <li>❑ Allows both synchronous and asynchronous communication.</li> <li>❑ Supports digital-signature-based authentication, validation, and authorization.</li> </ul>

© Aptech Ltd.      Introduction to Web Services/Session 1      38

Using slide 38, explain ebXML and its features. Mention that ebXML facilitates electronic business over the Internet. It can be used to advertise and find out information about businesses. ebXML can also be used to publish Web service descriptions in this registry. Explain ebXML registry and what data it can hold. Additionally, explain the ebXML information model and its advantages.

**JAXR API**

Slide 39

**JAXR API**

Java API for XML Registries (JAXR) API

- Is an abstract uniform Java API.
- Has a single set of APIs that can access many XML registries, including UDDI and ebXML registries.

© Aptech Ltd. Introduction to Web Services/Session 1 39

Using slide 39, introduce the JAXR API.

**JAXR Architecture**

Slide 40

**JAXR Architecture**

- ◆ Following figure shows the JAXR architecture:

The diagram illustrates the JAXR architecture. At the top is the **JAXR Client**, which interacts with a **JAXR Pluggable Provider**. The provider then connects to various **Diverse Registries**. The client interface is divided into **Capability-Specific Interfaces** (C1, C2, ..., Cn) and **Registry-Specific JAXR Provider** interfaces (eb XML Provider, UDDI Provider, Other Provider). The registries themselves are categorized as **eb XML/SOAP**, **UDDI/SOAP**, and **Other**.

- ◆ The JAXR information model is based on the ebXML Registry Information Model (RIM). It also supports UDDI and has all the data types defined in the UDDI Data Structure Specification.

© Aptech Ltd. Introduction to Web Services/Session 1 40

Using slide 40, explain the JAXR architecture and its two parts namely, JAXR client and JAXR provider and for what they are used. Mention that the JAXR provider is a JAXR pluggable provider with underlying implementations of a UDDI-specific JAXR provider and an ebXML-specific provider.

The JAXR pluggable provider exposes capability-specific methods to the JAXR client via the registry interface. The registry returns the response to JAXR provider where the response is transformed to JAXR response. The provider sends this response back to the JAXR client.

Explain the different functionalities from different registry providers and the two capability profiles, namely level 0 and level 1.

### JAXR Information Model

Slide 41

<b>JAXR Information Model</b>	
<b>Interface</b>	<b>Description</b>
Organization	Organization instances are RegistryObjects that provide information on an organization that has been published to the underlying registry.
Service	Service instances are RegistryObjects that provide information on services offered by an organization.
ServiceBinding	ServiceBinding instances are RegistryObjects that represent technical information on how to access a specific interface offered by a Service instance.
Concept	Concept instances represent an arbitrary notion or concept. It can be virtually anything.
Classification Scheme	ClassificationScheme instances represent a taxonomy that can be used to classify or categorize RegistryObject instances.
ExternalLink	ExternalLink instances provide a link to content managed outside the registry using a URI.
ExternalIdentifier	ExternalIdentifier instances provide identification information to a RegistryObject.
Classification	Classification instances classify a RegistryObject instance using a classification scheme.
PostalAddress	PostalAddress instances provide address information for a user and an Organization.

© Aptech Ltd.

Introduction to Web Services/Session 1

41

Using slide 41, explain the ebXML Information Model. Explain its features and benefits. Explain the various interfaces that define the JAXR information model. Provide an insight on how the registry is used.

### In-Class Question:



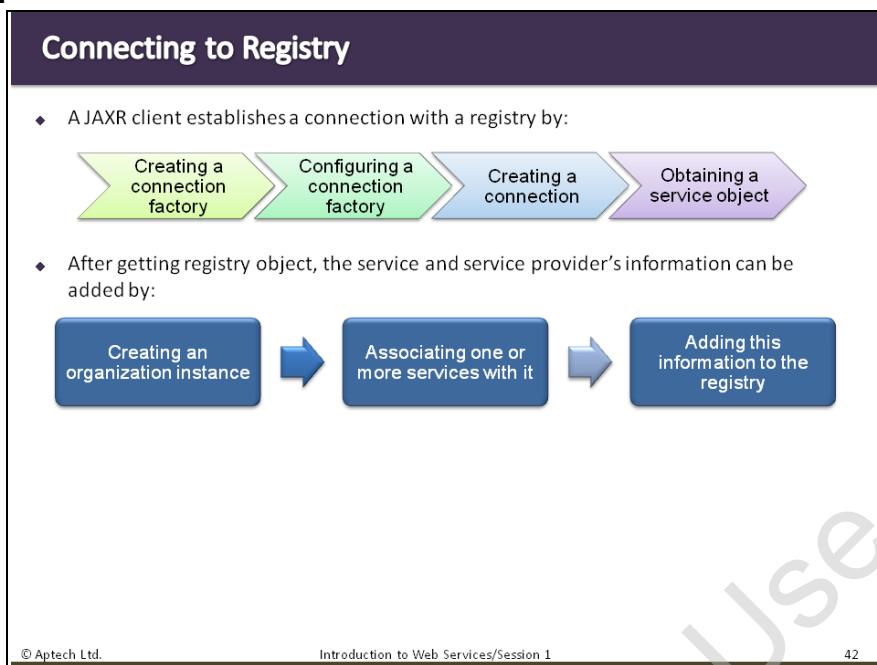
What is JAXR?

### Answer:

Java API for XML Registries (JAXR) API is an abstract uniform Java API. It has a single set of APIs that can access many XML registries, including UDDI and ebXML registries.

**Connecting to Registry**

Slide 42



© Aptech Ltd.

Introduction to Web Services/Session 1

42

Using slide 42, explain how the JAXR client establishes a connection with the registry by the following steps:

- Creating a connection factory
- Configuring connection factory
- Creating a connection
- Obtaining a service object

Explain how service and service provider's information can be added to registry object by the following steps:

- Creating an organization instance
- Associating one or more services with it
- Adding this information to the registry

Use different examples (such as properties of objects, methods, instances of objects, and so on) to explain the above steps for establishing connection as well as adding information to registry.

**Publishing Data on Registry**

Slides 43 to 47

**Publishing Data on Registry 1-5**

- Following Code Snippet demonstrates the steps to publish a Web service to UDDI:

```
public static void main(String[] args) throws JAXRException{
    try {
        //Setting the properties for the ConnectionFactory Properties enviro = new
        Properties(); enviro.setProperty("javax.xml.registry.queryManagerURL",
        QUERY_URL); enviro.setProperty("javax.xml.registry.lifeCycleManagerURL"
        , PUBLISH_URL);
        enviro.setProperty("javax.xml.registry.factoryClass",
        "com.sun.xml.registry.uddi.ConnectionFactoryImpl");
        //creating a connection
        ConnectionFactory confac = ConnectionFactory.newInstance();
        confac.setProperties(enviro);
        Connection conn = confac.createConnection();
        //Authenticating the UDDI userName and Password
        PasswordAuthentication passwdAuth = new
        PasswordAuthentication(uddiUserName,
        uddiPassword.toCharArray());
        Set credentials = new HashSet();
        credentials.add(passwdAuth);
        conn.setCredentials(credentials);
```

**Publish data to the UDDI by  
configuring the properties.  
Then, create a connection.**

**Authenticate the UDDI credentials.**

**Publishing Data on Registry 2-5**

```
//Obtaining a reference to the RegistryService,
//BusinessLifeCycleManager and the BusinessQueryManger
RegistryService registryservice = conn.getRegistryService();
BusinessLifeCycleManager lifecyclemgr =
registryservice.getBusinessLifeCycleManager();
BusinessQueryManager querymgr =
registryservice.getBusinessQueryManager();
//Creating an organization object
Organization company = lifecyclemgr.createOrganization("Aptech");
InternationalString description =
lifecyclemgr.createInternationalString("Leading IT
Trainers");
//Creating a user object
User contact = lifecyclemgr.createUser();
PersonName name = lifecyclemgr.createPersonName("John Miller");
contact.setPersonName(name);
//creating and assigning the users telephone number
TelephoneNumber telnum = lifecyclemgr.createTelephoneNumber();
telnum.setNumber("1800-420-8000");
Collection phonenumbers = new ArrayList();
phonenumbers.add(telnum);
contact.setTelephoneNumber(phonenumbers);
```

**Obtain a reference to the  
RegistryService,  
BusinessLifeCycleManager and the  
BusinessQueryManager classes.**

## Publishing Data on Registry 3-5

```
//creating and assigning the users email address
EmailAddress email = lifecyclemgr.createEmailAddress("enquiry@abcinc.com");
Collection emaillist = new ArrayList();
emaillist.add(email);
contact.setEmailAddresses(emaillist);
//Setting the user as the primary contact for the organization
company.setPrimaryContact(contact);
//Creating the Web service object
Collection servicelist = new ArrayList();
Service service = lifecyclemgr.createService("Courses");
InternationalString serviceDescription =
lifecyclemgr.createInternationalString("Available
courses..");
service.setDescription(serviceDescription);
//Creating the Web service bindings
Collection serviceBindigs = new ArrayList();
ServiceBinding binding =
lifecyclemgr.createServiceBinding();
InternationalString bindingDescription =
lifecyclemgr.createInternationalString("Course Name");
```

Define the data that needs to be published to the UDDI.

Define the Web services to be offered and the Web service bindings for the same.

## Publishing Data on Registry 4-5

```
binding.setDescription(bindingDescription);
binding.setAccessURI("http://www.aptech.org");
boolean b = serviceBindigs.add(binding);
service.addServiceBindings(serviceBindigs);
servicelist.add(service);
company.addServices(servicelist);
//Classifying the organization
ClassificationScheme scheme = querymgr.
findClassificationSchemeByName(servicelist,"Courses");
Classification classification = lifecyclemgr.
createClassification(scheme, "Course name", "Course number");
Collection classificationlist = new ArrayList();
boolean c = classificationlist.add(classification);
company.addClassifications(classificationlist);
Collection organizationlist = new ArrayList();
organizationlist.add(company);
//making the final call to the registry to get a response
BulkResponse response = lifecyclemgr.saveOrganizations(organizationlist);
Collection exceptions = response.getExceptions();
```

Classify the Web services offered into the appropriate category.

## Publishing Data on Registry 5-5

```

if (exceptions == null) {
    Collection keys = response.getCollection();
    Iterator iterator = keys.iterator();
    Key key = (Key) iterator.next();
    String uid = key.getId();
    company.setKey(key);
}
else { // there is an exception
    Iterator iterator = exceptions.iterator();
    while (iterator.hasNext()) {
        Exception exception = (Exception) iterator.next();
        System.out.println("Exception...." + exception);
    }
    conn.close();
}
catch(Exception e) {
    e.printStackTrace();
}
}

```

Publish all the data into the registry and awaits response from the registry.

Registry throws exceptions in case any errors are encountered.

If there are any exceptions, the details of the exceptions is displayed in the console.

Else, the data is published and added to the Web service.

Using slides 43 to 47, explain the Code Snippet that demonstrates the steps to publish a Web service to UDDI. Explain that the code publishes data to the UDDI. It first configures the properties, such as queryManagerURL, lifeCycleManagerURL, and factoryClass for the ConnectionFactory. Then, the code creates a connection and authenticates the UDDI credentials.

Next, the code obtains a reference to the RegistryService, BusinessLifeCycleManager and the BusinessQueryManger classes. The code then defines the data that needs to be published to the UDDI. It also defines the Web services to be offered and the Web service bindings for the same. The code also classifies the Web services offered into the appropriate category.

Finally, the code publishes all the data into the registry and awaits response from the registry. The registry will throw exceptions in case any errors are encountered. If there are any exceptions, the details of the exceptions will be displayed in the console. Else, the data will be published and added to the Web service.

**Querying the Registry**

Slide 48

## Querying the Registry

◆ A registry can be queried by using organization and service names related to the organization that needs information.

Finding Organizations by Name

- A find qualifier collection is created to store find qualifiers.
- A constant is added to show that the search results have to be sorted in descending order.
- A name pattern collection namePat is created and the string of organization name is added to it.
- Using find qualifier and name pattern instances, the findOrganizations() method is passed.

Finding Services and Service Bindings

- An iterator instance is created from the collection returned by getCollection() method.
- A while loop is defined to process all the organization objects in the iterator. Inside the loop, the organization object is retrieved using the next() method.
- An iterator is created from the collection returned by getServices() method.
- While loop is defined to process each service associated with the object.

◆ One can remove any data submitted by using the key returned by the registry as an argument to one of the BusinessLifeCycleManager delete methods.

◆ The different delete methods include deleteOrganizations(), deleteServices(), deleteServiceBindings(), and deleteConcepts().

© Aptech Ltd.      Introduction to Web Services/Session 1      48

Using slide 48, explain how a registry can be queried by using the name of the organization and the name of the service.

**SAAJ**

Slides 49 and 50

## SAAJ 1-2

SOAP with Attachments API for Java (SAAJ) is an API that allows users to create and send SOAP messages with attachments by means of the javax.xml.soap package.

SOAP provides a common message format for Web services. It enables developers to produce and consume messages conforming to the SOAP 1.2 specification and SOAP with Attachments note.

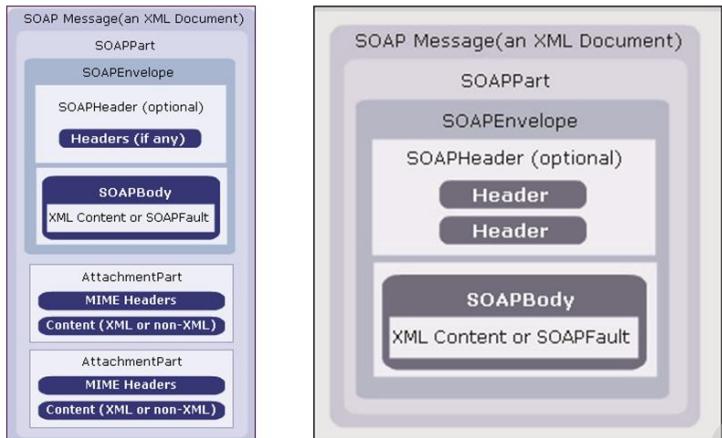
◆ Using SAAJ API, one can create XML messages that adapt to the SOAP 1.2 and WebService-I Basic Profile 2.0 specifications.

◆ There are two perspectives of SAAJ, namely Messages and Connections.

© Aptech Ltd.      Introduction to Web Services/Session 1      49

## SAAJ 2-2

- ◆ There are two types of SOAP messages are those that have attachments and those without attachments, which are as follows:



© Aptech Ltd.

Introduction to Web Services/Session 1

50

Using slides 49 and 50, explain what SAAJ is and what it does. Explain the two perspectives of SAAJ, namely Messages and Connections. Using slide 50, explain the two types of SOAP messages: those that have attachments and those without attachments.

Mention that a SOAP message that does not have attachments has a structure made up of a SOAPPART, a SOAPENVELOPE, an optional SOAPHEADER, and a SOAPBODY. The SOAPPART represents the SOAP document and includes the SOAPENVELOPE objects.

The SOAPENVELOPE represents the Envelope element in the SOAP document and contains SOAPHEADER object and SOAPBODY object which represents the Header and Body elements in the SOAP document.

Besides the SOAPPART, a SOAP message can have one or more attachment parts. However, the SOAP part must contain only XML content. If any content in a message is not in XML format, it is placed in the attachment part.

### In-Class Question:



What is SAAJ?

### Answer:

SOAP with Attachments API for Java (SAAJ) is an API that allows users to create and send SOAP messages with attachments by means of the javax.xml.soap package.

**SAAJ API**

Slide 51

**SAAJ API**

- ◆ The package javax.xml.soap provides the API for creating and building SOAP messages.
- ◆ Following table describes some of the classes that constitute SAAJ API:

Class	Description
AttachmentPart	A single attachment to a SOAPMessage object.
MimeHeader	An object that stores a MIME header name and its value.
SOAPConnection	A point-to-point connection that a client can use for sending messages directly to a remote party.
SOAPConnectionFactory	A factory for creating SOAPConnection objects.
SOAPMessage	The root class for all SOAP messages.

Using slide 51, explain SAAJ API and the classes that constitute SAAJ API.

**SOAP Connection**

Slide 52

**SOAP Connection**

- ◆ In SAAJ API, the connection is represented by a SOAPConnection object, which goes from the one endpoint to another endpoint.
- ◆ Following Code Snippet creates the SOAPConnection object con and then uses it to send the message:

```
SOAPConnectionFactory fact = SOAPConnectionFactory.newInstance();
SOAPConnection con = fact.createConnection();
// create a request message and give it content
java.net.URL endpoint = new URL("http://java.sun.com/javaee/7/docs");
SOAPMessage response = con.call(request, endpoint);
```

- All the messages sent over a SOAPConnection object are sent with the call() method.
- The return value for the call() method is the SOAPMessage object that is the response to the message that was sent.

Using slide 52, explain the connection that is represented by SOAPConnection Object.

Explain why it is called a point-to-point connection. Using the Code Snippet, explain how a connection is established. Explain some of the new features of SAAJ 1.3. They are listed as follows:

- Support for the WS-I Basic Profile 1.1
- Document Object Model integration
- SOAPMessage properties being used to set character set encoding and turn the writing of an XML declaration at the start of the SOAP part of the message

**JAXB**

Slide 53

## JAXB

Java Architecture for XML Binding (JAXB) is a set of interfaces which helps client applications to communicate with code generated from a schema.

- Following table shows the three components of JAXB:

Interface	Description
Binding compiler	Creates Java classes from a given schema.
Binding framework	Provides runtime services such as marshalling, unmarshalling, and validation that have to be performed on the contents classes.
Binding language	Describes schema binding of Java classes using which you can override the default binding rules.

© Aptech Ltd. Introduction to Web Services/Session 1 53

Using slide 53, explain JAXB and the three components or interfaces of JAXB, which are Binding compiler, Binding framework, and Binding language. Describe the usage of each component.

**In-Class Question:**

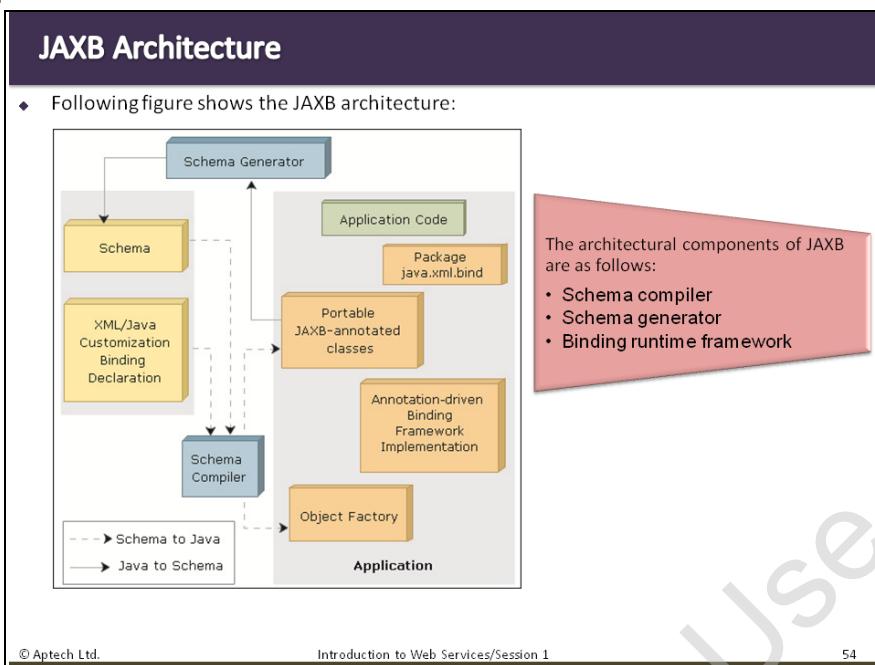
What is JAXB?

**Answer:**

Java Architecture for XML Binding (JAXB) is a set of interfaces which helps client applications to communicate with code generated from a schema.

**JAXB Architecture**

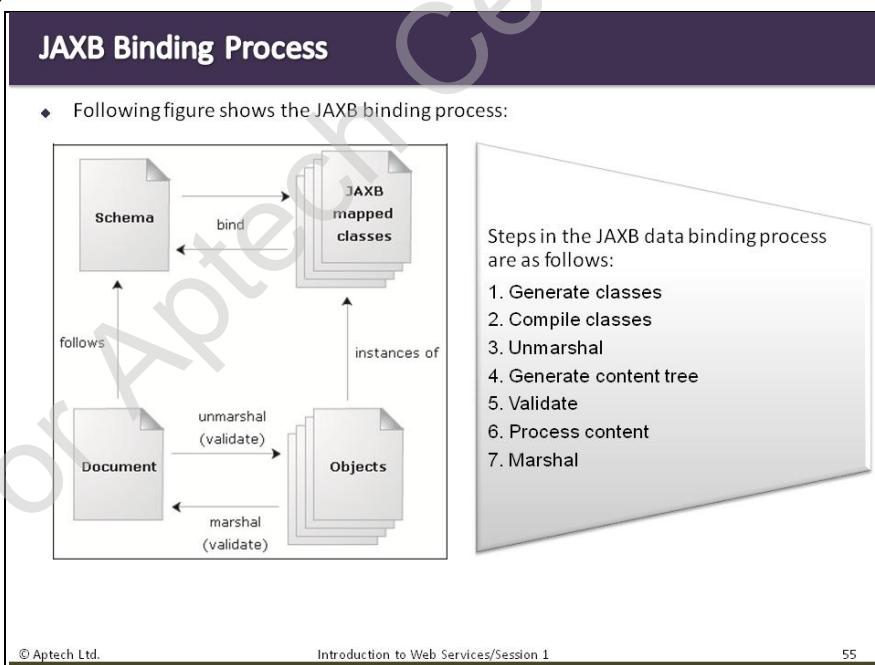
Slide 54



Using slide 54, explain JAXB architecture and the benefit of standard structure. Also, explain the architectural components of JAXB: Schema compiler, Schema generator, and Binding runtime framework and for what they are used.

**JAXB Binding Process**

Slide 55



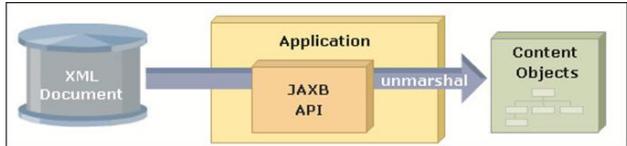
Using slide 55, explain that JAXB presents an XML document in a Java format to the application, which enables easy access to the contents. Explain that the schema is bound into a set of Java classes. Finally, explain the six steps in the JAXB binding process.

**Concept of Unmarshalling**

Slide 56

### Concept of Unmarshalling

- ◆ The process of converting an XML document into a content tree is termed as unmarshalling.
- ◆ Following figure depicts unmarshalling:



- ◆ JAXB allows to access XML data without unmarshalling it.

© Aptech Ltd. Introduction to Web Services/Session 1 56

Using slide 56, explain the concept of Unmarshalling. Explain that the process of converting an XML document into a content tree is termed as unmarshalling. Explain further with the help of the figure.

**In-Class Question:**

What is Unmarshalling?

**Answer:**

The process of converting an XML document into a content tree is termed as unmarshalling.

Unmarshalling

Slide 57

**Unmarshalling**

- Following Code Snippet shows how to parse an XML document using a DOM-based parser:

```
//Step1
import javax.xml.bind.JAXBContext; JAXBContext jc =
JAXBContext.newInstance(
"Information.jaxb");
//Step2
import javax.xml.bind.Unmarshaller;
Unmarshaller unmarshaller = jc.createUnmarshaller();
//Step3
Collection collection = (collection)unmarshaller.unmarshal(new
File("products.xml"));
//Step4
CollectionType.ProductsType productsType = collection.
getProducts();
List productList = productsType.getProduct();
```

1. An object of JAXBContext class is created whose context path is Information.jaxb.

2. An object of the Unmarshaller class is created that controls the process of unmarshalling.

3. The unmarshal() method is invoked, which performs the actual unmarshalling of the XML document, products.xml.

4. The get() method in the schema-derived classes is used to access the XML data.

© Aptech Ltd.

Introduction to Web Services/Session 1

57

Using slide 57, explain using the Code Snippet that during unmarshalling, a DOM tree that represents the content of XML document is created. The content in the tree is in the form of nodes. Explain the steps in the Code Snippet.

Concept of Marshalling

Slide 58

**Concept of Marshalling**

- In the marshalling process, the user has to build an XML document using a content tree.
- Following Code Snippet demonstrates how to create an XML document from a DOM tree:

```
//Step1
import javax.xml.bind.JAXBContext;
JAXBContext jc = JAXBContext.newInstance(
"Information.jaxb");
//Step2
import javax.xml.bind.Marshaller;
Marshaller marsh = jc.createMarshaller();
//Step3
marsh.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, new Boolean(true));
//Step4
marsh.marshal(collection, new FileOutputStream("Output.xml"));
```

1. An object of the JAXBContext class is created and the appropriate context path of the package that contains the classes and interfaces for the bound schema is specified.

2. An object of the Marshaller class is created which controls the process of marshalling.

3. The Marshaller object sets its properties using the setProperty method.

4. The marshal() method is invoked by specifying an object that contains the root of the content tree and the output target.

© Aptech Ltd.

Introduction to Web Services/Session 1

58

Using slide 58, explain the concept of Marshalling. Explain that unlike the unmarshalling process, the user has to build an XML document using a content tree in the marshalling process. Explain the various steps to create an XML document from a DOM tree.

## Data Validation

### Slide 59

**Data Validation**

- ◆ Validation is the process of verifying if an XML document meets all the constraints expressed in the schema.
- ◆ It has to be done stringently while writing out data, but not when reading data.
- ◆ During unmarshalling, `setValidating()` method has to be used to validate source data against the associated schema.
- ◆ If an error is encountered, the data processing need not be stopped.
- ◆ Instead, a validation error report should be generated.
- ◆ JAXB specification authorizes all providers to report validation errors.
- ◆ If an XML document is found invalid, it will still be unmarshalled although the result will not be valid.
- ◆ Validation is not done as part of marshalling process as `setValidating()` method does not exist for marshalling.
- ◆ Validation can be done at one time and marshalling at another time.

© Aptech Ltd.

Introduction to Web Services/Session 1

59

Using slide 59, explain data validation and how it has to be done during unmarshalling. Explain why validation is not done as part of marshalling process.

## Limitations of JAXB

### Slide 60

**Limitations of JAXB**

Limitations of JAXB:

- JAXB cannot be used to process generic XML as it requires a DTD and a subset of XML Schemas.
- Additional work is required to tell JAXB what kind of tree it should construct to simplify the application.
- JAXB does not support the legal DTD constructs.

© Aptech Ltd.

Introduction to Web Services/Session 1

60

Using slide 60, explain the limitations of JAXB.

**Summarize Session****Slide 61**

**Summary**

- ◆ Java EE Web services are platform-independent and language-independent services available on the Web.
- ◆ Service Oriented Architecture (SOA) aids application development by a loosely coupled modular approach. It permits the applications to extend and utilize services.
- ◆ SOAP is a Web service standard for packaging the XML data that are transferred between applications.
- ◆ The Java EE platform has two types of endpoints for Web services. One extends Stateless Session Beans into Web services and the other resembles a plain Java class.
- ◆ The Java API (JAXP) provides a framework for using SAX2 and DOM2 that read, write, and modify XML documents.
- ◆ JAX-WS provides a framework and runtime environment to create and execute XML-based Web services.
- ◆ The Java API for XML Registries (JAXR) API provides a single set of APIs to access a variety of XML registries.
- ◆ The Java Architecture for XML Binding (JAXB) API is a set of interfaces that enable communication through the code generated from a schema.

Using slide 61, summarize the session. Make them revise the following points:

- Java EE Web services are platform-independent and language-independent services available on the Web.
- Service Oriented Architecture (SOA) aids application development by a loosely-coupled modular approach. It permits the applications to extend and utilize services.
- SOAP is a Web service standard for packaging the XML data that are transferred between applications.
- The Java EE platform has two types of endpoints for Web services. One extends Stateless Session Beans into Web services and the other resembles a plain Java class.
- The Java API (JAXP) provides a framework for using SAX2 and DOM2 that read, write, and modify XML documents.
- JAX-WS provides a framework and runtime environment to create and execute XML-based Web services.
- The Java API for XML Registries (JAXR) API provides a single set of APIs to access a variety of XML registries.
- The Java Architecture for XML Binding (JAXB) API is a set of interfaces that enable communication through the code generated from a schema.

**1.3 Post Class Activities for Faculty**

You should familiarize yourself with the topics of the next session. You should also explore the next session which describes SOAP, WSDL, and UDDI.

**Tips:** You can also check the **Articles/Blogs/Expert Videos** uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

## Session 2: SOAP, WSDL, and UDDI

### **2.1 Pre-Class Activities**

You should familiarize yourself with the concepts of SOAP, WSDL, and UDDI. You should be able to explain the advantages of using SOAP in a Web service. You should be able to explain different parts of a SOAP Message. You should also be able to explain purpose of WSDL file and describe its elements. In addition, you should also be able to explain UDDI model and ebXML Registry Standards.

Familiarize yourself with the topics of the current session in-depth.

#### **2.1.1 Objectives**

After the session, learners will be able to:

- Explain the purpose and advantages of using SOAP in Web service
- Describe the different parts of a SOAP message
- Explain SOAP messaging with attachment
- Define Document/Literal SOAP and RPC/Literal SOAP messaging modes
- Explain transportation of SOAP over HTTP protocol
- List sub-elements of SOAP fault element and describe SOAP fault codes
- Explain the purpose of a WSDL file and describe its elements
- Describe UDDI model
- Explain the ebXML Registry standards

#### **2.1.2 Teaching Skills**

To teach this session, you should be well versed with the concept of Simple Object Access Protocol (SOAP) and its advantages. You should know about SOAP messaging thoroughly. You should have sound knowledge on the concepts of Web Service Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI) model, and ebXML Registry Standards. You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

#### **Tips:**

It is recommended that you test the understanding of the students by asking questions in between the class.

#### **2.1.3 In-Class Activities**

Follow the order given here during In-Class activities.

#### **Review of Previous Session**

You should begin with a brief recap or review of previous session which discussed about Web services and their purpose. You should summarize the various Web service standards and APIs discussed in the previous session.

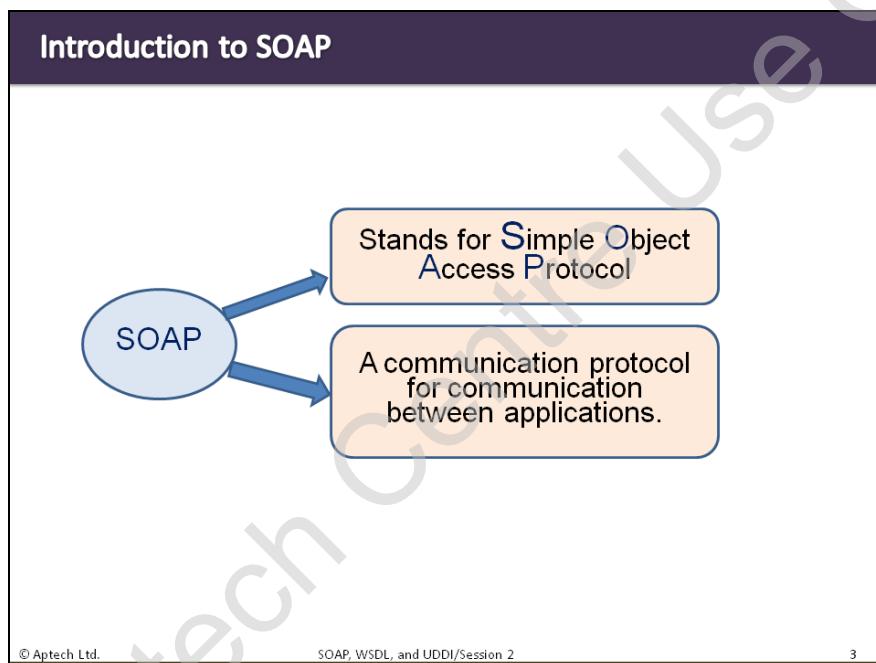
## Overview of the Session

Then, give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell the students that this session explains the concept of SOAP, WSDL, and UDDI. Tell them that this session also explains in detail the purpose and advantages of SOAP. The session introduces them to the purpose of WSDL and also explains about UDDI model and ebXML Registry Standards.

## 2.2 In-Class Explanations

### Introduction to SOAP

#### Slide 3



Using slide 3, provide the full form of SOAP. Then, explain the definition of SOAP. Inform the students that SOAP is based on XML and W3C recommendations.

## Information Exchange Approaches

### Slide 4

### Information Exchange Approaches

The diagram consists of a central title 'Information Exchange Approaches' at the top. Below it are two blue rounded rectangular boxes. The top box is labeled 'EDI' and contains a bulleted list of steps for sending an EDI document. The bottom box is labeled 'RPC and RMI' and contains a bullet point describing its function. At the bottom of the slide, there is copyright information and a page number.

- To send an EDI document, perform the following steps:
  - Install translation software on your system. The translation software converts business documents into X12 format.
  - Set up a private wide area network to send and receive the documents.
- The Remote Procedure Call (RPC) approach involves invoking remote methods. These remote methods allowed information exchange in the form of parameters and returned values.

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      4

Using slide 4, explain the two common approaches for information exchange between distributed systems, that is, Electronic Data Interchange (EDI) and Remote Procedure Call (RPC). Explain that EDI is used to exchange business documents between business partners. Explain the steps to send or receive EDI documents. Also, explain the drawbacks of using this approach.

Then, discuss RPC and Remote Method Invocation (RMI). Tell the students that these approaches are used to invoke remote methods, which allow information exchange. Explain the drawbacks of RPC and RMI approach.

## SOAP

### Slide 5

## SOAP

- ◆ SOAP provides interoperability between applications using XML and HTTP.
- ◆ XML is platform and language independent.
- ◆ A Java program on a Linux platform can easily interpret a SOAP message created by a C# program on a Windows platform.
- ◆ Following figure shows the use of SOAP:

Requires no special firewall configuration. Traffic can be carried by any protocol.

Corporate A    Internet    Business Partner B

SOAP message (XML Infoset)

© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

5

Using slide 5, describe SOAP. Explain to the uses of SOAP with the help of the figure. Tell them that SOAP provides interoperability between Extensible Markup Language (XML) and Hypertext Transfer Protocol (HTTP) applications. Hence, Business Corporate can easily exchange business information using SOAP Messages through different platforms.

#### In-Class Question:



What is SOAP?

#### Answer:

SOAP stands for Simple Object Access Protocol. It is a communication protocol used between different applications.

## Advantages of SOAP

### Slide 6

## Advantages of SOAP

- Vendor Neutral**
  - Most vendors implement products as per the specifications or standards defined by W3C.
- Transport Protocol Independent**
  - SOAP messages can be transmitted over HTTP, Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and Post Office Protocol (POP).
- Platform Independent**
  - XML is platform independent. SOAP message are constructed using XML Infoset, hence, SOAP is also platform independent.
- Language Independent**
  - SOAP messages are XML Infosets. Several languages, such as Java, Perl, C++ can be used to create these messages.
- Interoperability**
  - SOAP transports XML Infosets using various protocols. This enables distributed applications to communicate without any issues.
- Simple**
  - SOAP does not require any change in network infrastructure or security configurations and hence is simple to use.

© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

6

Using slide 6, discuss the advantages of SOAP. Explain the various advantages of SOAP such as Vendor Neutrality, Transport Protocol Independence, Platform Independence, Language Independence, Interoperability, and Simplicity. Explain why these are the advantages of SOAP. Tell the students that from version 1.2, the acronym for SOAP, 'Simple Object Access Protocol', has been dropped. Also, tell that SOAP uses other application layer protocols such as HTTP for transmission of messages.

**SOAP 1.1 vs. SOAP 1.2****Slides 7 to 9****SOAP 1.1 vs. SOAP 1.2 1-3**

- Following table provides a comparison between the features provided by SOAP 1.1 and SOAP 1.2:

Feature	SOAP 1.1	SOAP 1.2
SOAP messages	Is based on XML 1.0	Is based on XML Infosets
SOAP elements	Allows additional elements to be used after the <body> element	Does not allow additional elements to be used after the <body> element
actor attribute	Indicates the receiver of a header element	Renamed to role and indicates the receiver of a header element
next role	Can be applied to the intermediary SOAP nodes	Can be applied to the intermediary SOAP nodes and the ultimate receiver
none role	Not supported	Used to indicate that the header block should not be processed by any of the SOAP nodes
ultimateReceiver role	Not supported	Used to indicate a header block must be processed only by the ultimate receiver of the SOAP message

**SOAP 1.1 vs. SOAP 1.2 2-3**

Feature	SOAP 1.1	SOAP 1.2
Fault codes	Supports the VersionMismatch, MustUnderstand, Client, and Server fault codes	Supports the DataEncodingUnknown, VersionMismatch, MustUnderstand, Sender (previously, Client), and Receiver (previously, Server) fault codes
Fault code extensions	Allows a dot notation to be used to indicate fault code extensions	Provides an XML-like structure for the fault code extensions
Fault structure	Only the root element was namespace qualified (e:fault); the other elements are faultcode, faultstring, faultfactor, and detail	All the elements are namespace qualified with e:fault as the root element and e:Code, e:Subcode, e:Value, e:Reason, e:Node, e:Role, and e:detail as the child elements
Fault semantics	Allows the body of the fault message to include multiple child elements	Allows the body of the fault message to include only one child element
Misunderstood header	Not supported	Used for providing additional information in the MustUnderstand faults

### SOAP 1.1 vs. SOAP 1.2 3-3

Feature	SOAP 1.1	SOAP 1.2
Upgrade header	Not supported	Used to specify the supported envelope versions when the VersionMismatch fault is reported by a node
Binding Framework	Supports single binding to HTTP	Provides an abstract binding framework to support all protocols
encodingStyle attribute	Allows the attribute to be used on any element in the envelope	Allows the attribute to be used only on the child elements of the Body, Header, and fault Detail elements
Multi-reference values	Encodes multi-reference values in top-level elements	Allows in-place encoding of multi-reference values

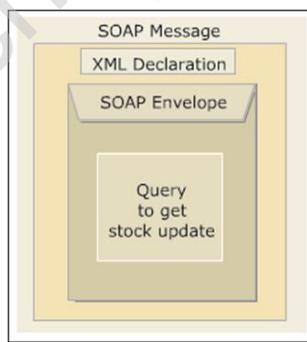
Using slides 7 to 9, explain the difference between the features of SOAP 1.1 and SOAP 1.2.

### SOAP Messages

#### Slide 10

### SOAP Messages

- ◆ A SOAP message contains the following parts:
  - ◆ XML declaration
  - ◆ Envelope: Comprises Header and Body. The data is enclosed in the Body part of the SOAP message. For example, a request to receive stock update will be part of the Envelope.
- ◆ Following figure shows the SOAP message structure:



Using slide 10, explain SOAP message structure. Begin the explanation by providing few examples of SOAP messages such as request to receive stock update, list of available flights, request to get a price quote, query to a search engine. Then, explain the different parts of SOAP message. Give an example for Envelope such as a request to receive stock update will be part of the Envelope and then explain the parts of SOAP Envelope.

## XML Declaration

### Slide 11

### XML Declaration

- ◆ Following Code Snippet demonstrates the XML declaration tag of a SOAP message:

```
<? Xml version= "1.0" encoding = "UTF-8"?>
```

**Note –**

- It is not mandatory to have an XML declaration in a SOAP message.
- Unicode is an encoding standard that can represent the characters, digits, and symbols of world's major languages.
- UTF-8 is a Unicode character-encoding format created to provide backward compatibility with ASCII.
- UTF-8 and UTF-16 encoding formats can represent any character in Unicode character set.

Using slide 11, explain XML Declaration part of a SOAP Message. Emphasize that a SOAP message always begins with an XML Declaration. Explain the Code Snippet that shows the XML Declaration tag of a SOAP Message. The declaration states the version of XML and the encoding used in the message. The XML version has to be 1.0 and the encoding value can be (UTF) -8 or (UTF)-16. If no encoding attribute is present, then by default the document uses UTF-8 encoding and the document version is considered as 1.0.

**SOAP Message Envelope****Slides 12 to 17****SOAP Message Envelope 1-6**

- ◆ The SOAP envelope contains a message from one application to be sent to another application.
- ◆ The envelope part of the SOAP message acts as a container for the message.
- ◆ The Envelope element:
  - ◆ is the root element of the message and is mandatory
  - ◆ acts as a processing node to the receiving application. For example, the <Envelope> tag indicates the start of a SOAP message and the </Envelope> tag indicates the end. Once the receiving application encounters the </Envelope> tag, it starts processing the message
  - ◆ contains two child elements, an optional <Header> element and a mandatory <Body> element



SOAP 1.1 allows additional elements to be added after the SOAP Body element, SOAP 1.2 does not allow these.

**SOAP Message Envelope 2-6**

- ◆ Following Code Snippet demonstrates the Envelope tag in a SOAP message:

```
<Envelope ...>
  <Header>
    0 or more headers
  </Header>
  <Body>
    message body
  </Body>
</Envelope>
```

**SOAP Namespaces**

- ◆ Following Code Snippet demonstrates a SOAP message with its elements qualified with respective namespaces:

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV= "http://www.w3.org/2003/05/soap-envelope"
  xmlns:po= "http://www.flamingo.com/books/PO">
  ...
  <!--SOAP Message goes here -->
  <po:orderDate> 2014/07/06 </po:orderDate>
  ...
</SOAP-ENV:Envelope>
```

## SOAP Message Envelope 3-6

### SOAP Message Header

- ◆ Header is the child element of Envelope element.
- ◆ Headers can also include information such as digital signatures for password-protected service, authentication, authorization, transaction management, and routing path.

### role Attribute

- ◆ Following Code Snippet demonstrates a SOAP message with various header entries:

```
<SOAP-ENV: Envelope
  xmlns:SOAP-ENV= "http://www.w3.org/2003/05/soap-envelope"
  xmlns:mid= "http://www.flamingo.com/books/message-id"
  xmlns:m= "http://www.flamingo.com/books/monitored-by"
  xmlns:md = "http://www.flamingo.com/books/monitoring-date"
  xmlns:mu = "http://www.flamingo.com/books/mark"
    <SOAP-ENV: Header>
      <mid:message-id
        SOAP-ENV:role="http://www.flamingo.com/logger">
          11546544ea:b134534:f3sdas5342:4354
        </mid:message-id>
      <m:monitored-by SOAP-ENV:role = "
```

## SOAP Message Envelope 4-6

```
http://www.w3.org/2003/05/soap-envelope/role/next">
<node>
  <time> 1078753670000  </time>
  <identity>austria</identity>
</node>
</m:monitored-by>
<md:monitoring-date SOAP-ENV:role = " http://www.w3.org/2003/05/soap-
envelope/role/none">
...
</md:monitoring-date>
<mu:mark SOAP-ENV:role = "http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver">
...
</mu:mark>
</SOAP-ENV: Header>
</SOAP-ENV: Envelope>
```

## SOAP Message Envelope 5-6

### mustUnderstand Attribute

- Following Code Snippet demonstrates an example of a header entry with the mustUnderstand attribute.

```
<!-- SOAP Message Structure -->
<?xml version = "1.0" encoding="UTF-8" ?><SOAP-ENV:Envelope
    xmlns:SOAP-ENV= "http://www.w3.org/2003/05/soap-envelope">
<SOAP-ENV:Header xmlns:au="http://www.flamingo.com/books.authentication">
<au:Requestor mustUnderstand="true"
    <name> John Smith </name>
</au:Requestor>
    ...
</SOAP-ENV:Header>
    ...
</SOAP-ENV:Envelope>
```

## SOAP Message Envelope 6-6

### Body Element

- The Body element is the mandatory element of Envelope element. The immediate child elements of Body element must be namespace-qualified. The Body element must be placed as follows:
  - If the Header element is not present, the Body element should be the immediate child of Envelope element.
  - If the Header element is present, the Body element should immediately follow the Header element.

### Need of Attachment

- SOAP messages contain XML fragment. However, at times you may want to send data that is not XML.
- Messages that require binary data are converted to a Multipurpose Internet Mail Extensions (MIME) message format and then sent.
- A MIME message can contain multiple parts and supports binary data as well.

Using slides 12 to 17, explain SOAP Message Envelope, its usage, and contents. Give examples to explain these concepts. Then, use the Code Snippet on slide 13 to explain the Envelope tag in a SOAP message. Using Code Snippets from 13 to 16, explain the students about SOAP Namespaces, SOAP Message Header, role Attribute, and mustUnderstand Attributes.

While discussing about SOAP Message Header, describe the information included in Header. The students may be asked to provide a few examples. While explaining about role attribute, explain the roles that are important in a SOAP message namely, next, none, and ultimateReceiver.

Next, proceed to explain about the Body Element. Describe its contents and how it should be placed. Then, provide a few examples of data in the Body element such as billing address or parameters to a method call.

Finally, explain the Need of Attachment. Tell that SOAP messages need not always be in XML hence, messages that require binary data are converted to Multipurpose Internet Mail Extensions (MIME). Explain by providing scenario-based examples. A scenario can be one where you want to allow customers to book a room in a hotel and how this can be achieved through SOAP-based Web Service.

**In-Class Question:**



What is MIME?

**Answer:**

MIME stands for Multipurpose Internet Mail Extensions that is used to convert binary data to a MIME message and then sent across different platforms.

**MIME Message Structure**

Slides 18 to 20

**MIME Message Structure 1-3**

**MIME header**

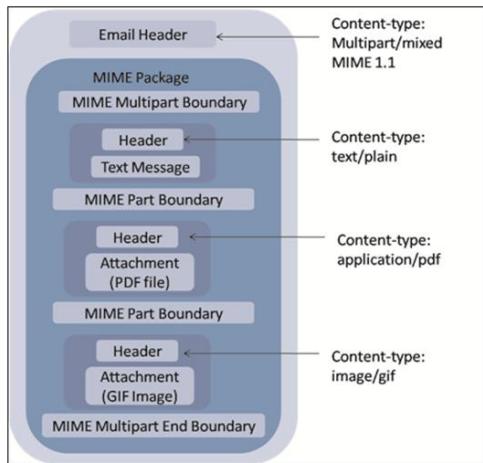
- Identifies the version of MIME and content type. A content type as Multipart/Mixed indicates that the email contains multiple parts of varying types

**MIME package**

- Can contain one or more MIME parts

## MIME Message Structure 2-3

- Following figure shows different types of message attachments:



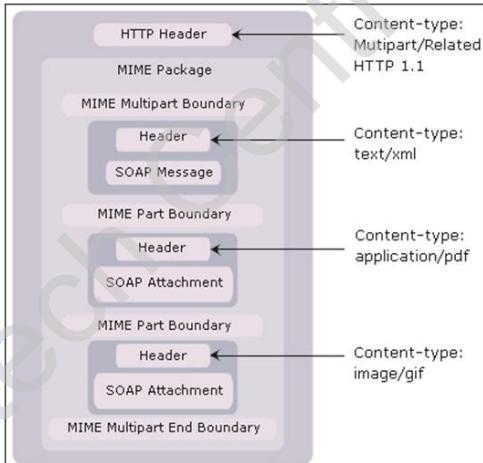
© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

19

## MIME Message Structure 3-3

- Following figure shows a SOAP message with two attachments:



© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

20

Using slides 18 to 20, explain MIME message structure. Explain the difference between the MIME header and the MIME package.

Explain about an email with attachment containing two or more MIME parts. Emphasize that each MIME message contains a header, which identifies the content type. Show the different types of message attachments using the figure on slide 19.

Then, explain about MIME message with a SOAP attachment. Tell that SOAP cannot have binary data in the message. However, SOAP messages can be sent along with binary data using SOAP with Attachments (SwA) standard. Using the figure on slide 20, explain a SOAP message with two attachments.

## SOAP Message Modes

### Slide 21

### SOAP Message Modes

- ◆ Based on the messaging styles and encoding types, SOAP defines following four messaging modes:
  - Document/Literal**
  - RPC/Literal**
  - Document/Encoded**
  - RPC/Encoded**

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      21

Using slide 21, list the messaging modes based on messaging styles and encoding types. Mention that Document/Encoded and RPC/Encoded messaging modes are not WS-I compliant.

#### In-Class Question:



What are the various SOAP Messaging Modes?

#### Answer:

- Document/Literal
- RPC/Literal
- Document/Encoded
- RPC/Encoded

**Document / Literal Messaging Mode****Slides 22****Document / Literal Messaging Mode**

- Following Code Snippet demonstrates a SOAP message with a Body element with the details about the Spiderman Digital Versatile/Video Disc (DVD):

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap/envelope">
    ...
    <SOAP-ENV:Body>
        <d:dvd xmlns:d="http://www.flamingo.com/dvds/">
            <d:title>Spider-Man</d:title>
            <d:language>English</d:language>
            <d:discs>2</d:discs>
            <d:runtime>
                <d:minutes>121 </d:minutes>
            </d:runtime>
            <d:price>9.49</d:price>
        </d:dvd>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Using slide 22, explain the Document / Literal Messaging Mode. Explain that Document / Literal messaging mode is used when XML fragments have to be sent in a SOAP message. Give examples of XML fragment such as movie details, purchase order, billing address, book details. Tell that the schema and namespace of XML fragment is different from that of SOAP elements. Use the Code Snippet to explain a SOAP message on Spiderman Digital Versatile/Video Disc (DVD).

**In-Class Question:**

What are the messaging modes that WS-I Basic Profile (WS-I BP) 2.0 does not support?

**Answer:**

Document/Encoded and RPC/ Encoded messaging modes.

**RPC / Literal Messaging Mode**

Slide 23

**RPC / Literal Messaging Mode**

- ◆ The RPC/Literal messaging mode is based on sending request messages and receiving response messages.
- ◆ Following Code Snippet demonstrates SOAP request message:

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=" http://www.w3.org/2003/05/soap/envelope"
    ...
    <SOAP-ENV:Body>
        <m:GetOrderStatus xmlns:m="http://www.flamingo.com/methods">
            <m:OrderNo>34347</m:OrderNo>
            </m:GetOrderStatus>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

- ◆ Following Code Snippet demonstrates SOAP response message:

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=" http://www.w3.org/2003/05/soap/envelope"
    ...
    <SOAP-ENV:Body>
        <m:GetOrderStatusResponse xmlns:m="http://www.flamingo.com/methods">
            <m:OrderStatus>Shipped on 2014-08-09 </m:OrderStatus>
        </m:GetOrderStatusResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Using slide 23, explain the RPC / Literal Messaging mode. Explain that it is based on sending request messages and receiving response messages. The request message represents a method with one or more parameters and the response message represents a value returned by the method. Tell that the requested and resultant information are sent as part of Body element.

Using the Code Snippets, explain the SOAP request message and the SOAP response message.

**Transport Protocols**

**Slide 24**

## Transport Protocols

**SOAP**

- ◆ Is an XML-based protocol.
- ◆ Follows the HTTP request and response model to transmit client request and obtain Web Service response.

SOAP messages over HTTP help in exchange of messages between clients and Web services, all running on different platforms, and at various locations on the Internet.

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      24

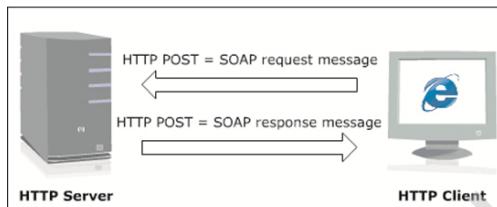
Using slide24, explain what SOAP is based on. Explain what model it follows to transmit client request and obtain Web Service response and the benefit of SOAP messages over HTTP.

**SOAP HTTP Binding****Slides 25 and 26****SOAP HTTP Binding 1-2**

- ◆ SOAP messages are transmitted as a payload of an HTTP message.
- ◆ A payload of an HTTP message contains form data such as username, password, credit card number, and so on.

**HTTP**

- ◆ Is a request-response protocol. A SOAP message is carried as a payload in an HTTP POST message.
- ◆ Following figure shows the HTTP POST message:

**SOAP HTTP Binding 2-2**

- ◆ Features of HTTP 1.1 are as follows:
  - ◆ Requires a Host header. This header allows a message to be routed through proxy servers and helps the Web server to distinguish between various sites on the same server.
  - ◆ Supports persistent connections, which allows multiple request/response on the same HTTP connections.
  - ◆ Provides the OPTIONS method that helps determine the capabilities of the HTTP server.
  - ◆ Provides the entity tag, which expands on the caching support.
  - ◆ Provides additional conditional headers such as If-Unmodified-Since, If-Match, and If-None-Match.

Using slides 25 and 26, explain about SOAP HTTP binding. Explain HTTP tunneling and how it helps a message pass through any firewall without any obstruction. Explain about payload of an HTTP message and what type of data it contains.

Describe the HTTP POST message using the figure on slide 25. Explain that SOAP messages sent over HTTP are placed in the payload of an HTTP request or response. By saying that HTTP is a Request/Response protocol, we mean that the sender expects a response from the receiver. The response can be data or even an error code. HTTP requests are typified by

the messages that your browser sends to a Web server to request a Web page or submit a form.

Mention that a request for a Web page is usually made in an HTTP GET message, while submission of a form is done with an HTTP POST message.

Explain that being an application-level protocol, HTTP is used in distributed, collaborative, and hypermedia information systems. Explain the limitations of HTTP 1.0. Tell that the limitations of HTTP 1.0 are overcome by HTTP 1.1 version. Then proceed to explain the features of HTTP 1.1.

### In-Class Question:



Which version of Hypertext Transfer Protocol (HTTP) is most commonly used?

### Answer:

HTTP 1.1

### SOAP Request over HTTP

#### Slide 27

#### SOAP Request over HTTP

- ◆ A SOAP message sent using HTTP POST message comprises the HTTP header and the SOAP message.
- ◆ Following Code Snippet demonstrates SOAP request over HTTP:

```
POST/orderstatus HTTP/1.1
Host:www.flamingo.com:80
Content-Type: text/xml; charset=utf-8
Content-Length:482
SOAPAction:"http://www.flamingo.com/books/getOrderStatus"
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV=" http://www.w3.org/2003/05/soap/envelope">
...
<SOAP-ENV:Body>
<m:GetOrderStatus
 xmlns:m="http://www.flamingo.com/methods">
 <m:OrderNo>34347</m:OrderNo>
</m:GetOrderStatus>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Using slide 27, explain SOAP Request Message over HTTP. Also, explain the Code Snippet in detail.

## SOAP Response over HTTP

### Slide 28

#### SOAP Response over HTTP

- ◆ A SOAP message sent using HTTP also contains the HTTP header and the SOAP message.
- ◆ Following Code Snippet demonstrates a SOAP response message embedded within an HTTP POST message:

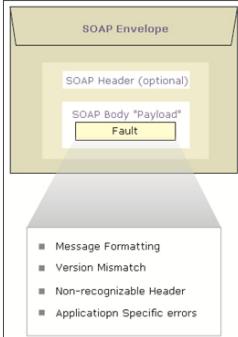
```
HTTP/1.1 200 OK
Connection:close
Content-Length: 659
Content-Type:text/xml; charset=utf-8
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV: Envelope
    xmlns:SOAP-ENV=" http://www.w3.org/2003/05/soap/envelope">
    ...
    <SOAP-ENV:Body>
        <m:GetOrderStatusResponse
            xmlns:m="http://www.flamingo.com/methods">
            <m:OrderStatus>
                Shipped on 2014-08-09
            </m:OrderStatus>
        </m:GetOrderStatusResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Using slide 28, explain how SOAP Response message is sent over HTTP. Explain the Code Snippet and tell that the value 200 OK in the first line is an HTTP response success code which indicates that the message was received and processed successfully.

**Fault Element****Slide 29**

## Fault Element

- ◆ A SOAP fault is generated when an error occurs in transmission of message.
- ◆ In SOAP, errors are generated by the nodes in the message path while processing a message.
- ◆ The causes for errors in SOAP are improper message formatting, version mismatch, trouble processing a header and application specific errors.
- ◆ Following figure shows SOAP message with Fault element:



The diagram illustrates the structure of a SOAP message. It consists of a 'SOAP Envelope' containing a 'SOAP Header (optional)' and a 'SOAP Body "Payload"'. Within the 'SOAP Body' is a 'Fault' element. Below the message structure, a list of causes for errors is provided:

- Message Formatting
- Version Mismatch
- Non-recognizable Header
- Application Specific errors

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      29

Using slide 29, explain Fault Element. Explain how SOAP Fault is generated. Explain the causes for errors in SOAP. Explain that an error is enclosed in a Fault element. Point out that a Fault element is always a child of Body element and it appears only once in the Body element. The Body element has only the Fault element. Using the figure, explain the structure of SOAP message with Fault element.

**Fault Subelements****Slides 30**

## Fault Subelements

- ◆ Following table describes the Fault subelements:

Fault Subelement	Description
env:Code	Identifies an error in a SOAP message. This subelement contains two child elements e:Value and e:Subcode
env:Value	Indicates a value that identifies the type of error that occurred.
env:Subcode	Indicates additional error information.

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      30

Using slide 30, explain the subelements of Fault element. Explain each of the Subelement and its functions.

**Fault Codes****Slide 31**

**Fault Codes**

- ◆ Following table describes the fault code:

Fault Code	Description
VersionMismatch	Indicates that receiving application received a different version of SOAP message.
MustUnderstand	Indicates that receiving node or application was unable to process the header entry targeted for it. <b>Note:</b> If the mustUnderstand attribute in the message received was set to 1, the receiving node sends a fault message.
Sender	Indicates an error in the message or its data. That is, it indicates an error on the message sender's end.
Receiver	Indicates that the intermediary node or the ultimate receiver was unable to process the message.
DataEncodingUnknown	Indicates that the received messages are using an unrecognized value of the encodingStyle attribute.

© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

31

Using slide 31, explain that a faultcode subelement provides error information in the form of fault codes. Tell about different Fault Codes and what each indicates.

**In-Class Question:**

What is the function of the Fault subelement env:Code?

**Answer:** Fault Subelement env:Code identifies an error in a SOAP message. This subelement contains two child elements e:Value and e:Subcode.

**SOAP Fault over HTTP****Slide 32****SOAP Fault over HTTP**

- ◆ An error in a SOAP message is communicated to the sender using a SOAP fault message.
- ◆ Following Code Snippet demonstrates an HTTP response message indicating mismatch in the version of SOAP message received:

```
HTTP/1.1 500 Internal Server Error
Connection: close
Content-Length:659
Content-Type: text/xml; charset=utf-8
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <Code>SOAP-ENV:VersionMismatch</Code>
            <Reason>Not a SOAP 1.2 message. </Reason>
            <Detail>Version MisMatch </Detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV: Body>
</SOAP-ENV:Envelope>
```

© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

32

Using slide 32, explain how an error in a SOAP message is sent to the sender.

Explain using the Code Snippet how an HTTP response message indicates a mismatch in the version of SOAP message received.

**Basics of WSDL****Slide 33****Basics of WSDL**

WSDL is a specification defined to describe information about a Web Service in XML format.

WSDL provides the common methods to represent the data types passed in the service, the functions that are available in the service, and the mapping of service onto the network protocol.

WSDL 2.0 divides the description of the abstract functionality offered by a service from the specific description of the service description such as 'how' and 'where' that functionality is offered.

© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

33

Using slide 33, explain WSDL. Describe the common methods of WSDL and how abstract functionality differs from the specific description of the service description.

**In-Class Question:**

What is WSDL?

**Answer:** WSDL stands for Web Services Description Language. It is a language that describes Web services and how to access them.

**WSDL for Describing Web services****Slide 34**

### WSDL for Describing Web Services

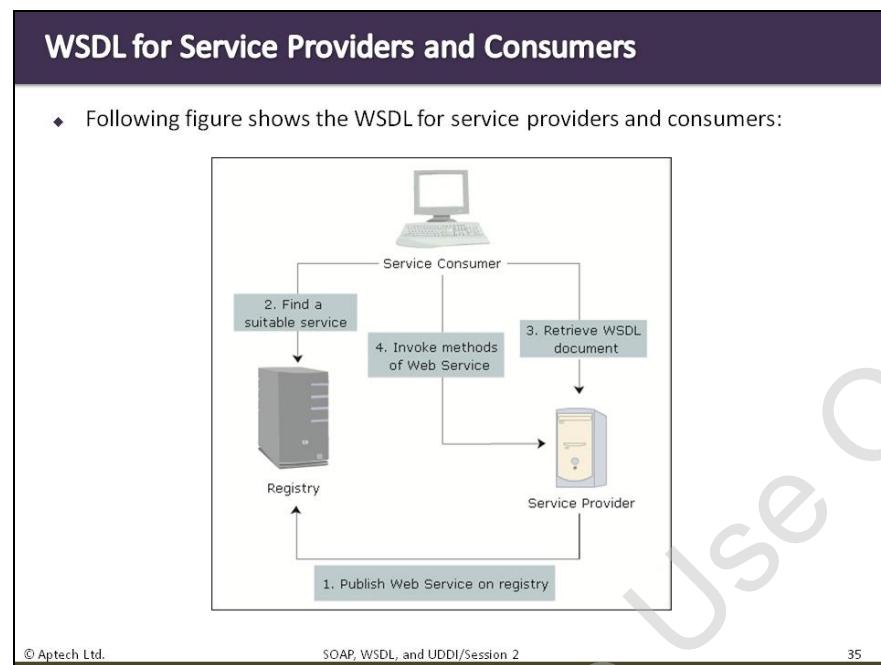
- ◆ A typical WSDL document contains the following information about the Web Service:
  - Available methods
  - Type of protocol to be used
  - Parameters and return type of methods
  - Location of Web Service

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      34

Using slide 34, explain what a typical WSDL document contains.

**WSDL for Service Providers and Consumers**

Slide 35



Using slide 35, explain how WSDL helps the Service Providers and Consumers using the figure. Explain that a Service Provider creates the WSDL document and publishes it along with other information to a registry. Next, a service consumer searches the registry to locate service of interest. The service consumer downloads the WSDL document after finding relevant service. Explain how the service consumer's application de-serializes the WSDL document and creates Java classes from them. Explain how Java classes can now invoke the methods of the Web service and start using the Web service.

**WSDL Document Structure**

Slides 36 to 39

**WSDL Document Structure 1-4**

- ◆ Following table describes the WSDL document elements:

Elements	Description
description	<ul style="list-style-type: none"> <li>• Acts as a container for the elements types, interface, binding, and service.</li> <li>• Defines the name of the Web Service and also one or more namespaces used by its child elements.</li> </ul>
types	<ul style="list-style-type: none"> <li>• Defines the data type of the information exchanged between applications.</li> <li>• Is mandatory only if the data type is other than the built-in data types of XML Schema. Example of XML schema's built-in types are string, integer, and so on.</li> </ul>
interface	<ul style="list-style-type: none"> <li>• Describes the operations that the Web service includes and the input/output messages that are exchanged for each operation.</li> <li>• Describes the fault messages.</li> </ul>
binding	<ul style="list-style-type: none"> <li>• Describes how the input and output messages of each operation will be transmitted over the Internet from one application to another.</li> </ul>
service	<ul style="list-style-type: none"> <li>• Defines the address for invoking the Web Service. In other words, it provides the URL of the service provider's server on which the Web Service is hosted. The service consumer's application uses this URL to invoke the methods of the Web Service.</li> </ul>
import	<ul style="list-style-type: none"> <li>• Used to import XML Schemas or WSDL document. This is an optional element.</li> </ul>

**WSDL Document Structure 2-4**

- ◆ Following Code Snippet shows a sample WSDL document:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<description targetNamespace="http://ws.soap.syskan.com/" 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: tns="http://ws.soap.syskan.com/"
  xmlns: xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsoap="http://www.w3.org/ns/wsdl/soap">
  <types>
    <xsd:schema>
      <xsd:import
        namespace="http://ws.soap.syskan.com/"
        schemaLocation="http://localhost:8080/SOAPWebService/SOAPWS?xsd=1"/>
```

## WSDL Document Structure 3-4

```

</xsd:schema>
</types>
<interface name="SOAPWS">
<operation name="add"
pattern="http://www.w3.org/ns/wsdl/in-out">
<input element="tns:add"
wsam:Action="http://ws.soap.syskan.com/SOAPWS/addRequest"/>
<output element="tns:addResponse"
wsam:Action="http://ws.soap.syskan.com/SOAPWS/addResponse"/>
</operation>
</interface>

<binding name="SOAPWSPortBinding"
interface="tns:SOAPWS"
type="http://www.w3.org/ns/wsdl/soap"
wsoap:version="1.1"
wsoap:protocol="http://www.w3.org/2006/01/
soap11/bindings/HTTP/">
<operation ref="tns:add">
<input/>

```

## WSDL Document Structure 4-4

```

<output/>
            </operation>
</binding>
<service name="SOAPWS" interface="tns:SOAPWS">
    <endpoint name="SOAPWSPort"
binding="tns:SOAPWSPortBinding"
address="http://localhost:8080/
SOAPWebService/SOAPWS"/>
</service>
</description>

```

Using slides 36 to 39, explain the structure of WSDL document. Explain the important elements of the WSDL document and what each are used for. Also, explain a sample WSDL Document using the given Code Snippet on slides 37 to 39.

**What is UDDI?**

Slides 40 to 42

**What is UDDI 1-3****Web Service Registry**

- ◆ A dictionary consists of searchable collection of words in one or more specific languages.
- ◆ Similarly, a service registry consists of searchable collection of descriptions of Web services.

**Universal Description, Discovery, and Integration (UDDI)**

- ◆ Is a platform-independent, XML based registry for businesses worldwide to list the business on the Internet.
- ◆ Provides standard mechanisms for businesses to describe and publish their Web Services, discover published Web Services, and use them.

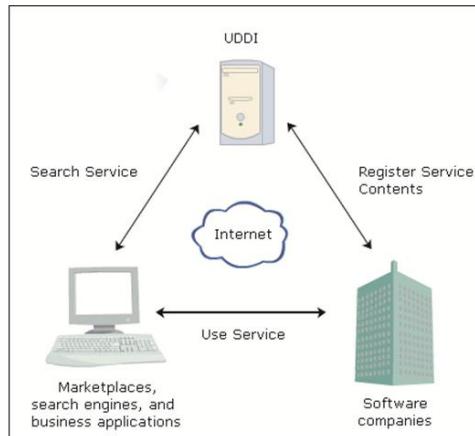
**What is UDDI 2-3**

To use the registry following steps are required:

- UDDI contains references to specifications called as Technical Models, or tModels.
- The tModels describe the working of Web Services. They are built upon a programming model and schema that are platform and language independent.
- The software companies populate the registry by describing various tModels and specifications common to a business.
- UDDI programmatically assigns a Unique Universal Identifier (UUID) to each tModel and business registration.
- Market places, search engines, and business applications query the registry to discover services of other companies and integrate this data with each other over the Web.

### What is UDDI 3-3

- Following figure shows the UDDI registry:



© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

42

Using slides 40 to 42, explain Web service registry and UDDI. Also, using the figure on slide 42 explain how UDDI can be used.

Give an example of the airline industry. Assume that the industry published an UDDI standard for flight rate checking and reservation. Now, all airlines can register their services into an UDDI directory. Travel agencies can then search the UDDI directory to find the airline's reservation interface. When the interface is found, the travel agency can communicate with the service immediately because it uses a well-defined reservation interface.

Consider a travel company XYZ wants to register its contact information, service description, and online service access information with UDDI. The following steps are necessary:

1. Select an appropriate operator (such as Microsoft) and obtain a UDDI client, such as those provided by the operator.
2. Obtain an authentication token from the operator.
3. Register all the important information about the business so that it helps when searching for matches.
4. Release the authentication token.
5. Fill in the tModel information. This is important because when someone wants to search for a given service can find your business as one of the service providers.
6. Update the information to reflect changing business contact information and new service details.

#### In-Class Question:



What is UDDI?

**Answer:** UDDI or Universal Description, Discovery, and Integration is a registry where businesses can list themselves on the Internet.

**UDDI Data Structure**

Slides 43 to 45

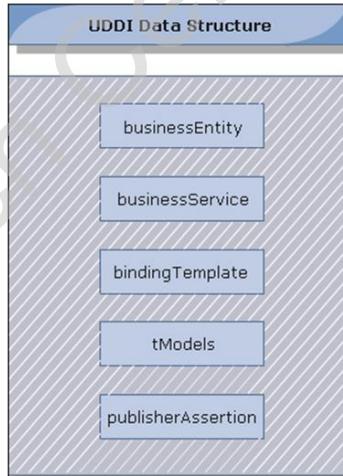
**UDDI Data Structure 1-3**

- ◆ UDDI provides five core data structures as follows:

<b>businessEntity</b>	• Represents all the information about the business or the organization that provides the Web Service.
<b>businessService</b>	• Represents a Web Service.
<b>bindingTemplate</b>	• Represents the binding of a Web Service with its URL and its tModels.
<b>tModel</b>	• Provides information about a Web Service. It specifies the name of the service, brief description of the service, and a unique code that is used to identify a service in the registry.
<b>publisherAssertion</b>	• Represents a relationship between two business entities or service providers.

**UDDI Data Structure 2-3**

- ◆ Following figure show the UDDI data structure:



### UDDI Data Structure 3-3

- ◆ Features of UDDI version 3 are as follows:
  - ◆ It serves as a meta service to find Web services by enabling robust queries against rich metadata.
  - ◆ It provides specification for building flexible, interoperable XML Web services registries useful in both private and public deployments.
  - ◆ It consists of multi-registry topologies.
  - ◆ It provides increased security features and improved WSDL support.
  - ◆ It provides a new subscription API and core information model advances.
  - ◆ It offers clients and developers a broad and complete blueprint of a description and discovery foundation for a diverse set of Web services architectures.

Using slides 43 to 45, explain the core data structures of UDDI. Explain the benefit of each data structure. Also, explain the key features of UDDI version 3.

#### In-Class Question:



List the core Data Structures of UDDI.

#### Answer:

- businessEntity
- businessService
- bindingTemplate
- tModel
- publisherAssertion

**UDDI Publisher API**

Slides 46 and 47

**UDDI Publisher API 1-2**

Following table describes some of the methods available under UDDI Publisher API:

Methods	Description
Add_publisherAssertions	Adds relationship assertions to the existing set of assertions
save_business	Adds or updates one or more businessEntity entries
save_service	Adds or updates one or more businessService entries
save_binding	Adds or updates one or more bindingTemplate entries
save_tModel	Adds or updates one or more tModel entries
delete_business	Deletes one or more businessEntity entries
delete_service	Deletes one or more businessService entries
delete_binding	Deletes one or more bindingTemplate entries
delete_tModel	Deletes (or hides) one or more tModel entries
delete_publisherAssertions	Deletes specific publisher assertions from the assertion collection managed by a specific publisher

**UDDI Publisher API 2-2**

Methods	Description
get_authToken (deprecated)	Logs you into the registry
discard_authToken (deprecated)	Logs you out of the registry
find_relatedBusinesses(deprecated)	Finds matching publisherAssertion entries
get_publisherAssertions	Gets a list of publisherAssertion entries
get_registeredInfo	Gets an abbreviated list of businesses and tModels currently being managed by a given publisher
get_assertionStatusReport	Gets a summary of publisherAssertion entries
set_publisherAssertions	Saves the entire set of publisher assertions for an individual publisher

Using slides 46 and 47, explain about Publisher API and Inquiry API in UDDI. Then, explain the different methods available under UDDI Publisher API.

**UDDI Inquiry API**

Slides 48 and 49

**UDDI Inquiry API 1-2**

Following table describes some of the methods available under the UDDI Inquiry API:

Methods	Description
find_business	Finds matching businessEntity entries
find_service	Finds matching businessService entries
find_binding	Finds matching bindingTemplate entries
find_tModel	Finds matching tModel entries
find_relatedBusinesses	Finds information about businessEntity registrations that are related to a specific business entity whose key is passed in the inquiry
get_businessDetail	Gets businessEntity entries
get_serviceDetail	Gets businessService entries
get_bindingDetail	Gets bindingDetail entries
get_tModelDetail	Gets tModel entries

**UDDI Inquiry API 2-2**

Methods	Description
get_operationalInfo	Gets operational information related to one or more entities in the registry
get_registeredInfo (deprecated)	Gets an abbreviated list of businessEntity and tModel entries
add_publisherAssertions (deprecated)	Adds one or more publisherAssertion entries
set_publisherAssertions (deprecated)	Updates one or more publisherAssertion entries
delete_publisherAssertions (deprecated)	Deletes one or more publisherAssertion entries

Using slides 48 and 49, explain the benefit of using the Inquiry API. Then, explain the various methods available under UDDI Inquiry API.

**Electronic Business Scenario**

Slides 50 and 51

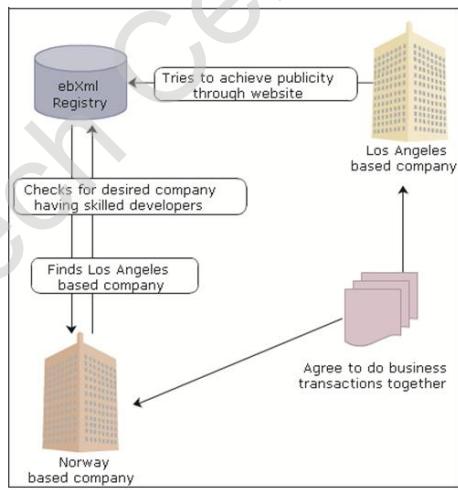
**Electronic Business Scenario 1-2**

Consider the case of two companies situated far away from each other and looking forward to start some business. Following are the details:

1. A small company in Los Angeles has a few skilled mobile application developers.
2. To establish a good international reputation, the company tries to achieve through Website.
3. Another small company in Norway is involved in mobile application development, but it does not have skilled developers.
4. When the Norway based company gets a new project, it looks for appropriate company.
5. The Norway based company checks the ebXML Registry for the desired company and finds the company in Los Angeles.
6. A contract is negotiated and both companies agree to do business together.
7. The Los Angeles based company sends the first mobile application back to Norway, which is accepted by Norway based company.
8. In return, the company transfers the money to Los Angeles.

**Electronic Business Scenario 2-2**

Following figure shows an E-business scenario:



Using slides 50 and 51, explain the concept of electronic business. Then, explain an E-business scenario in which two companies situated in remote locations want to start some business. Explain how the companies provide business details to each other and how ebXML enables the companies to work together in future.

**Electronic Business Process**

Slides 52 and 53

**Electronic Business Process 1-2**

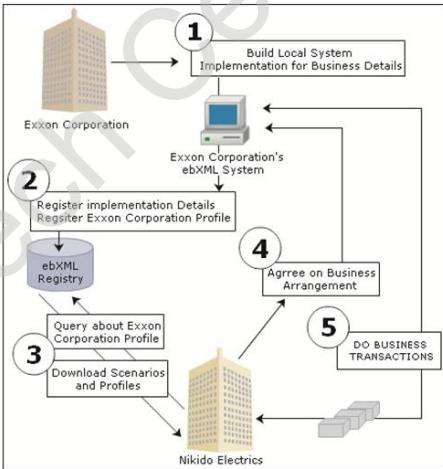
Consider a scenario of how two companies, Exxon Corporation, a Los Angeles-based company and Nikido Electrics, a Norwegian company could implement a business that adheres to ebXML processes.

Following are the steps both the companies must do to do electronic business in ebXML format:

1. Exxon Corporation creates a Collaboration Protocol Profile (CPP) by defining itself in an ebXML compliant application.
2. Exxon Corporation submits its CPP to the registry and tries to achieve publicity through its Website.
3. Nikido Electrics, already registered at the ebXML registry/repository, is looking for new trading partners. It starts a new query and receives the CPP of Exxon Corporation. Using the CPP of Exxon Corporation and that of Nikido Electrics, ebXML derives a document called Collaboration Protocol Agreement (CPA).
4. Nikido Electrics contacts Exxon Corporation directly and sends the newly created CPA for acceptance.
5. After signing the contract, both the companies do electronic business directly following the business processes defined in the CPA.

**Electronic Business Process 2-2**

Following figure shows the use of ebXML:



Using slides 52 and 53, explain using a scenario how companies can implement a business that adheres to ebXML processes. Explain the steps the companies need to take up for doing electronic business in ebXML format using the figure on slide 53.

Begin the explanation of the figure starting from step 1 stating Exxon Corporation builds Local System Implementation for Business Details and submits the CPP to the registry. Then, Register implementation details of Exxon Corporation profile and query about EXXON Corporation Profile and ensure that both the companies (Exxon Corporation and Nikido Electrics) agree on Business arrangement and finally do business transactions.

**In Class Question:**

What is ebXML?

**Answer:** ebXML is an XML based standard, which enables business organizations to use business information available on the Internet in a secure way for their benefits.

**ebXML Registry and Repository**

Slides 54 and 55

## ebXML Registry and Repository 1-2

- ◆ The ebXML architecture consists of components such as:
  - ◆ ebXML Registry and Repository
  - ◆ ebXML Business Processes
  - ◆ ebXML Collaboration Protocol Profiles
  - ◆ Collaboration Protocol Agreements
  - ◆ ebXML Core Components
  - ◆ ebXML Messaging Service
- ◆ The clients communicate with the registry using the following two interfaces:
  - ◆ **Object Manager:** Provides the methods to create new objects within the registry and affect state transitions on existing objects.
  - ◆ **Object Query Manager:** Provides the methods to find and access the objects created within the registry.

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      54

## ebXML Registry and Repository 2-2

Following figure shows the ebXML architecture:

```

graph TD
    CC[Core Component] --- BP[Business Process]
    BP --- RR[Registry and Repository]
    RR --- OM[Object Manager]
    RR --- OQM[Object Query Manager]
    RR --- CPP[Collaboration Protocol Profile]
    RR --- CPA[Collaboration protocol Agreement]
    CPP --- MS[Messaging Service]
  
```

The diagram illustrates the ebXML architecture as a stack of five horizontal layers. From top to bottom, the layers are: Core Component (pink), Business Process (light blue), Registry and Repository (yellow), Collaboration Protocol Profile / Collaboration protocol Agreement (orange), and Messaging Service (grey). The Registry and Repository layer contains two boxes: Object Manager and Object Query Manager.

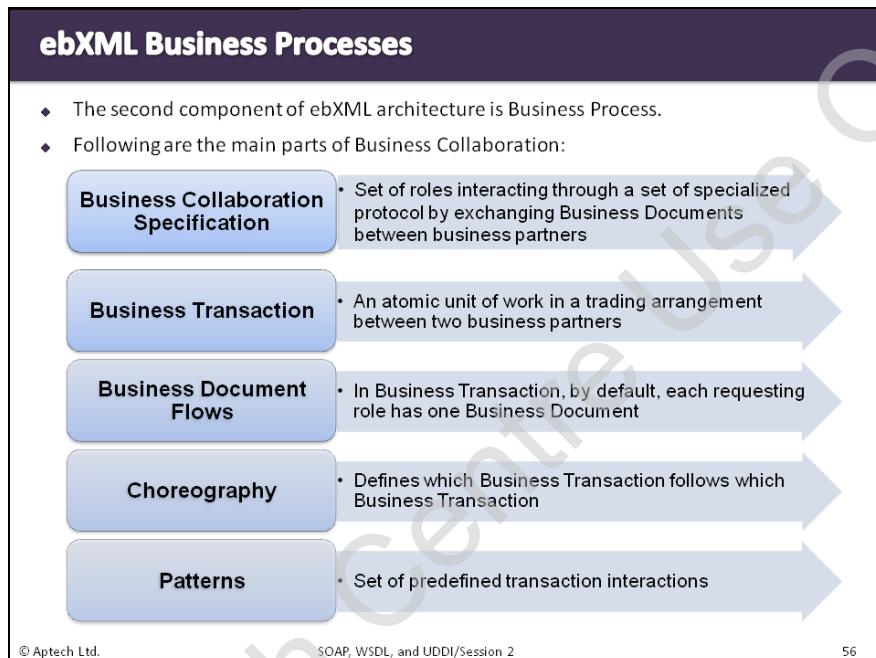
© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      55

Using slides 54 and 55, explain the components of ebXML architecture. Inform that ebXML registry and repository is a superset of UDDI. Then, explain what information is stored in

ebXML registry. Point out that the registry is connected internally to the repository. Explain the interfaces through which clients can communicate with the registry. Explain for what each interface is used. Show the figure on slide 55 and explain each block. For example, mention that a core component is the basic building block that contains information representing a business concept. A Business Process involves the exchange of information between two or more Trading Partners, and so on.

### **ebXML Business Processes**

**Slide 56**



Using slide 56, explain the second component of ebXML architecture that is Business Process. Explain how it defines the Business Collaboration. Additionally, describe the main parts of Business Collaboration, which are:

- Business Collaboration Specification
- Business Transaction
- Business Document Flows
- Choreography
- Patterns

**ebXML CPP and CPA**

Slides 57 to 59

**ebXML CPP and CPA 1-3**

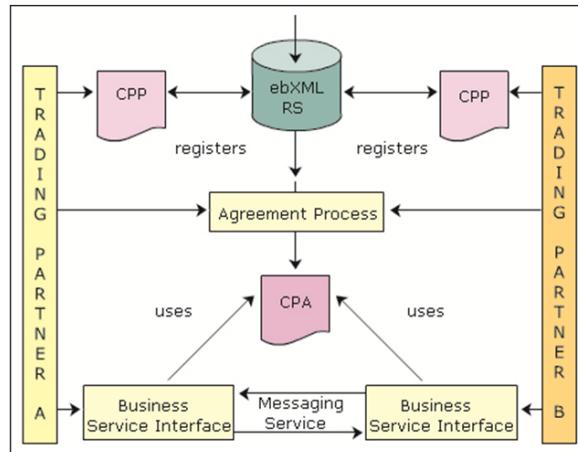
- ◆ The third component of ebXML architecture includes the Collaboration-Protocol Profile (CPP) and Collaboration-Protocol Agreement (CPA) documents.
  - ◆ **CPP:** Is an XML document that contains information about a business and the way it exchanges information with other businesses . Is an intersection of two CPP documents.
  - ◆ **CPA:** Is derived from two or more CPPs.

**ebXML CPP and CPA 2-3**

- ◆ The working of CPP and CPA is as follows:
  1. Two companies or trading partners create CPPs by adding information about the company.
  2. Both companies submit their CPPs to the ebXML registry.
  3. The CPPs are stored in the repository.
  4. A company then queries the ebXML registry to get a suitable CPP from a potential trading partner.
  5. Using the CPP of the two companies, ebXML derives the CPA.
  6. The company sends the CPA to the potential trading partner for acceptance.
  7. Upon agreement, the CPA gets registered in ebXML registry, and both the companies start the business.
  8. After registering the CPA in the ebXML registry, both companies configure their Business Service Interface (BSI) software with the newly created CPA.
  9. After the configuration, the BSI software is used to carry out electronic business.

### ebXML CPP and CPA 3-3

- Following figure shows ebXML CPP and CPA:



© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

59

Using slides 57 to 59, explain the next component of ebXML architecture, which includes the Collaboration-Protocol Profile (CPP) and Collaboration – Protocol Agreement (CPA) documents. Explain their working with the help of the given figure on slide 59.

Mention that the Messaging Service helps to exchange messages between organizations and the registries is a database of items such as XML schemas of business documents, definitions of library components for business process modelling, and trading partner agreements which support doing business electronically. The CPA is derived from the CPP's of the trading partners.

#### In-Class Question:



What is CPP?

**Answer:** CPP is an XML document that contains information about a business and the way it exchanges information with other businesses. It is an intersection of two CPP documents.

**Core Components and Messaging Service**

Slide 60

## Core Components and Messaging Service

- ◆ Two components of ebXML architecture are: Core Components and the Messaging Service.

<p style="text-align: center;"><b>Core Component</b></p> <ul style="list-style-type: none"> <li>• Is the last entity that gets referenced in a CPP document.</li> <li>• Is also called Aggregated Component.</li> <li>• Is a reusable building block containing information about a business concept. For example, for a concept such as purchase order, the core components are date of purchase order, sales tax, and total amount.</li> </ul>	<p style="text-align: center;"><b>ebXML Messaging Service (ebXML MS)</b></p> <ul style="list-style-type: none"> <li>• Provides the message exchange functionality within the ebXML infrastructure.</li> <li>• Is an entity used by the Business Service Interface software to send and receive XML messages from one point to another</li> <li>• Is based on TCP/IP, FTP, HTTP, and SMTP protocols</li> <li>• Uses packages data in SOAP messages and transmits them using HTTP, TCP/IP, FTP, or SMTP protocol</li> </ul>
--	---

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      60

Using slide 60, explain the components of ebXML architecture, which are, Core Components and the Messaging Service. Explain how ebXML supports re-usability. Also, explain about ebXML Messaging Service.

**Java API for XML Registries (JAXR)**

Slides 61 and 62

## Java API for XML Registries (JAXR) 1-2

- ◆ JAXR is an abstraction-based API therefore; Java programmers can use this API to develop registry client programs that can be used across different target registries.
- ◆ Following Code Snippet shows how to use UDDI to publish and query a Web service:

```

public class publishWebService{
    public static void main(String[] args) throws
        JAXREException{

        //Configure the ConnectionFactory to use JAXR Provider for UDDI
        System.setProperty("javax.xml.registry.
        ConnectionFactoryClass", "com.ibm.xml.registry.
        uddi.ConnectionFactoryImpl");
        ConnectionFactory connFac = ConnectionFactory.newInstance();

        //Configure URLs for UDDI inquiry and publish APIs.
        Properties p = new Properties();
        p.setProperty("javax.xml.registry.queryManagerURL", query_url);
        p.setProperty("javax.xml.registry.lifeCycleManagerURL", publish_
        url);
        connFac.setProperties(p);

        //Create a connection to the UDDI registry
        Connection conn = connFac.createConnection();
    }
}
```

© Aptech Ltd.      SOAP, WSDL, and UDDI/Session 2      61

## Java API for XML Registries (JAXR) 2-2

```

//Specify credentials for accessing UDDI registry.
PasswordAuthentication passwdAuth = new
PasswordAuthentication("Aptech_user", new char[]
{'p', 'a', 's', 's', '@', '1', '2', '3'});
Set userCredentials = new HashSet();
userCredentials.add(passwdAuth);
conn.setCredentials(userCredentials);
//Retrieve the javax.xml.registry.BusinessLifeCycleManager interface
RegistryService regServ = conn.getRegistryService();
BusinessLifeCycleManager bLCM = regServ.getBusinessLifeCycleManager();
//Create an Organization named "Aptech".
Organization org = bLCM.createOrganization("Aptech");
//Add the organization to a Collection
Collection coll = new ArrayList();
coll.add(org);
//Save the Organization to the UDDI registry.
BulkResponse bRes = bLCM.saveOrganizations(coll);
//Obtain the Organization's Key from the response.
if (bRes.getExceptions() == null) {
Collection res = bRes.getCollection();
Key orgaKey = (Key)res.iterator().next();
System.out.println("\nOrganization Key = " + orgKey.getId());
}
}
}

```

© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

62

Use slides 61 and 62, explain about Java API for accessing XML registries.

Point out that JAXR has a unified JAXR information model that describes data and metadata in the registries. Explain how JAXR helps Java programmers. Also, explain about JAXR metadata. Point out that JAXR works well with JAXP, JAX-WS, and JAXM to enable Web services in Java EE7.

Then, explain the Code Snippet which shows how to use UDDI to publish and query a Web service.

### Summarize Session

#### Slide 63

### Summary

- SOAP overcomes the problems of EDI and RPC by using XML and HTTP.
- The structure of a SOAP message consists of Envelope element, Header element and body element.
- Document/Literal messaging mode is used to transmit data in XML fragments, whereas RPC/Literal messaging mode is used to transmit method call and the values returned by a method.
- Transport Protocols deals with transportation of SOAP messages over HTTP.
- Errors in SOAP messages are sent as fault messages over HTTP.
- A WSDL document is an XML document that adheres to the WSDL XML schema. It generally consists of six elements namely definitions, types, message, portType, binding and service.
- An ebXML registry is similar to the UDDI registry only with the difference that along with storing information about the WSDL document it also stores the document too.

© Aptech Ltd.

SOAP, WSDL, and UDDI/Session 2

63

Using slide 63, summarize the session. Make them revise the following points:

- SOAP overcomes the problems of EDI and RPC by using XML and HTTP.
- The structure of a SOAP message consists of Envelope element, Header element, and body element.
- Document/Literal messaging mode is used to transmit data in XML fragments, whereas RPC/Literal messaging mode is used to transmit method call and the values returned by a method.
- Transport Protocols deals with transportation of SOAP messages over HTTP.
- Errors in SOAP messages are sent as fault messages over HTTP.
- A WSDL document is an XML document that adheres to the WSDL XML schema. It generally consists of six elements namely, definitions, types, message, portType, binding, and service.
- An ebXML registry is similar to the UDDI registry only with the difference that along with storing information about the WSDL document it also stores the document too.

### **2.3 Post Class Activities for Faculty**

You should familiarize yourself with the topics of the next session. You should also explore the next session which describes Web service Endpoints and guidelines to design a Web service endpoint.

**Tips:** You can also check the **Articles/Blogs/Expert Videos** uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

## Session 3: Web Service Endpoints

### **3.1 Pre-Class Activities**

You should familiarize yourself with the guidelines to design Web service endpoints. You should know the guidelines to design a Web service. You should be well-versed with Web service annotations. You should also be able to explain the method to package and deploy a Web service. You should also know thoroughly the process of invoking a Web service.

Familiarize yourself with the topics of the current session in-depth.

#### **3.1.1 Objectives**

After the session, learners will be able to:

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

#### **3.1.2 Teaching Skills**

To teach this session, you should be well versed with Web Service Endpoints, deploying a Web service, and the process of invoking a Web service. You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

#### **Tips:**

It is recommended that you test the understanding of the students by asking questions in between the class.

#### **3.1.3 In-Class Activities:**

Follow the order given here during In-Class activities.

#### **Review of Previous Session**

You should begin with a brief recap or review of previous session. You should summarize the concepts of SOAP, WSDL, and UDDI along with their purpose and advantages that were discussed in the previous session.

#### **Overview of the Session**

Then, give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell the students that this session explains Web Service Endpoints in detail. Tell them that this session also explains the methods to deploy the Web service and the process of invoking Web service.

### 3.2 In-Class Explanations

#### Web Service Endpoints

Slide 3

The diagram illustrates the components of a Web service endpoint. At the top, a blue circle labeled "Web Service Endpoints" is connected by a line to a larger shape below it. This shape is divided into two horizontal sections: a light blue section at the top and an orange section at the bottom. The light blue section contains the text: "A URL where the service can be accessed by the client application". The orange section contains the text: "Web service design guidelines simplifies the process of creating endpoints".

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

© Aptech Ltd.      Web Service Endpoints/Session 3      3

Using slide 3, explain a Web service endpoint. List the guidelines a developer has to consider while designing the Web service. Tell that Web service design guidelines simplify the creation of Web service endpoints.

#### Web Service Design Guidelines

Slide 4

The diagram lists the steps to create a Web service and a client. It starts with a list of developer requirements, followed by a title "Steps to create a Web service and a client:", and then a numbered list of 8 steps with corresponding icons.

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:

- 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

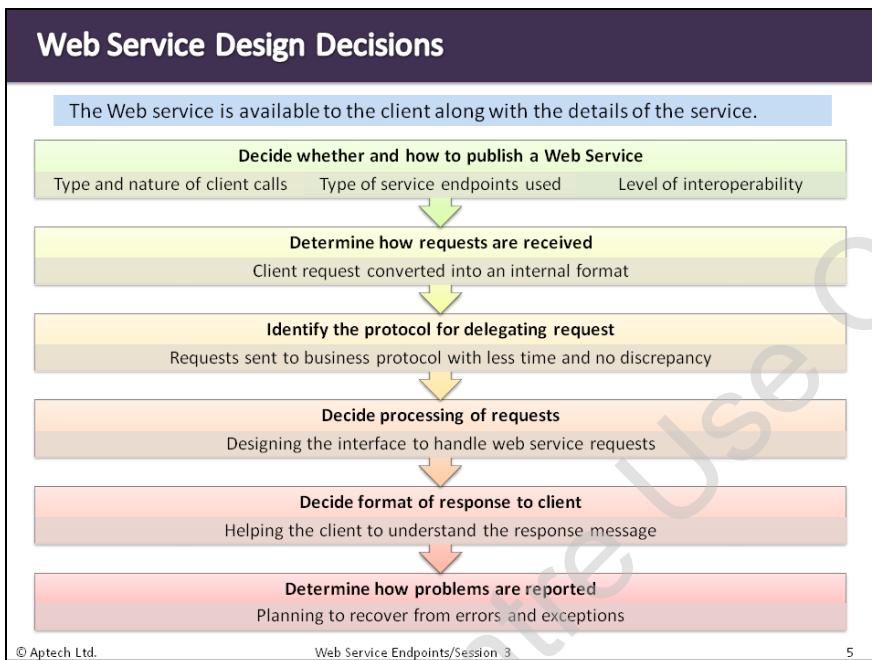
© Aptech Ltd.      Web Service Endpoints/Session 3      4

Using slide 4, explain the steps to create a Web service and a client. Explain that first the developer has to understand the nature of the service and the implementation class has to

be annotated with `javax.jws.WebService` or the `javax.jws.WebServiceProvider` annotation. Then, proceed to explain the various steps to create a Web service and a client.

### Web Service Design Decisions

Slide 5



Using slide 5, explain briefly how the Web service is made available to the client. Then, explain the different steps to decide the design of the Web service. Discuss the various factors on which the interface of the Web service depends. Explain about the importance of handling exceptions and errors and the need for a recovery plan.

#### In-Class Question:



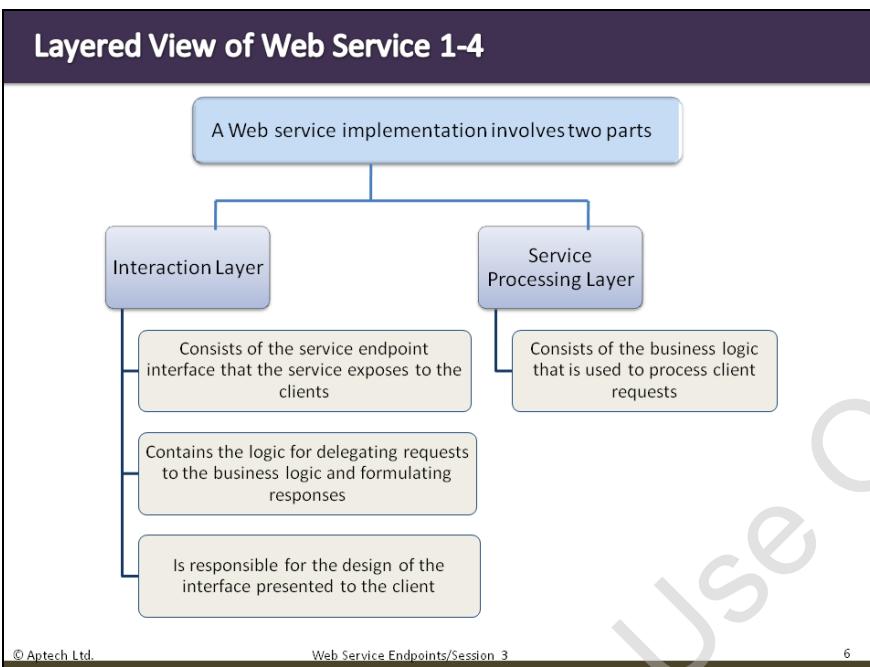
What is a Web service endpoint?

#### Answer:

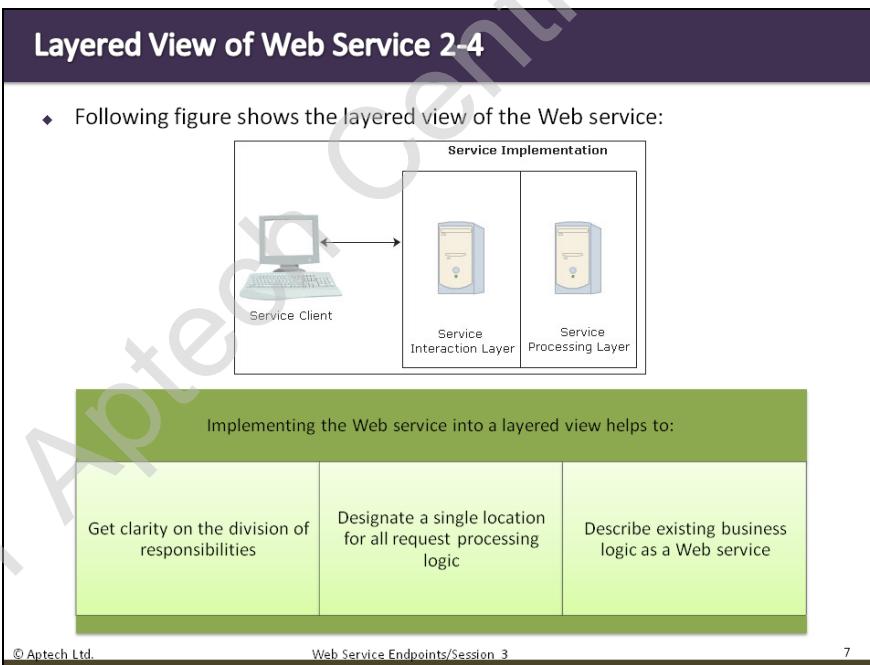
A Web service endpoint is a program that implements a Web service and carries out Web service requests.

**Layered View of Web Service**

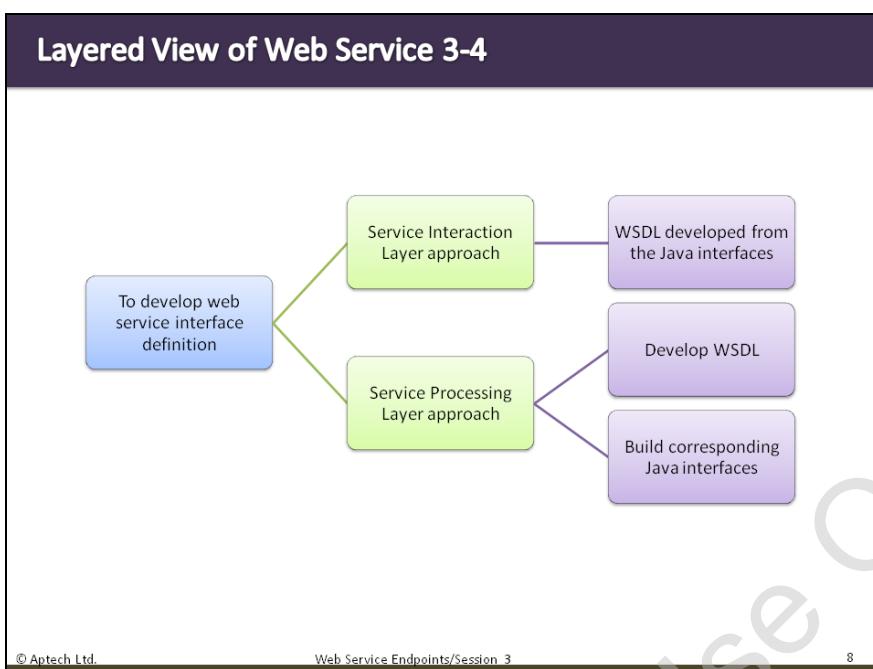
Slides 6 to 9



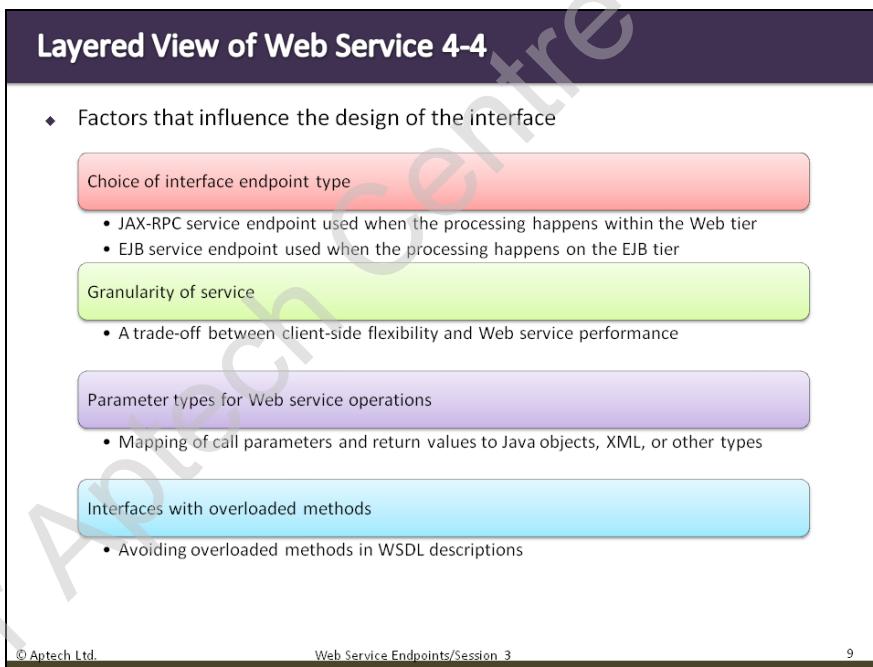
Using slide 6, discuss about the two parts that are involved in the Web service implementation. Explain the benefits of implementing the Web service into a layered view.



Using slide 7, explain the figure that shows the layered view of the Web service. Using the figure, explain how the Service clients interact with Service Interaction Layer and Service Processing Layer.



Using slide 8, discuss how Web service interface definition can be developed using Service Interaction Layer approach or Service Processing Layer approach.



Using slide 9, explain the factors that influence the design of the interface.

#### In-Class Question:



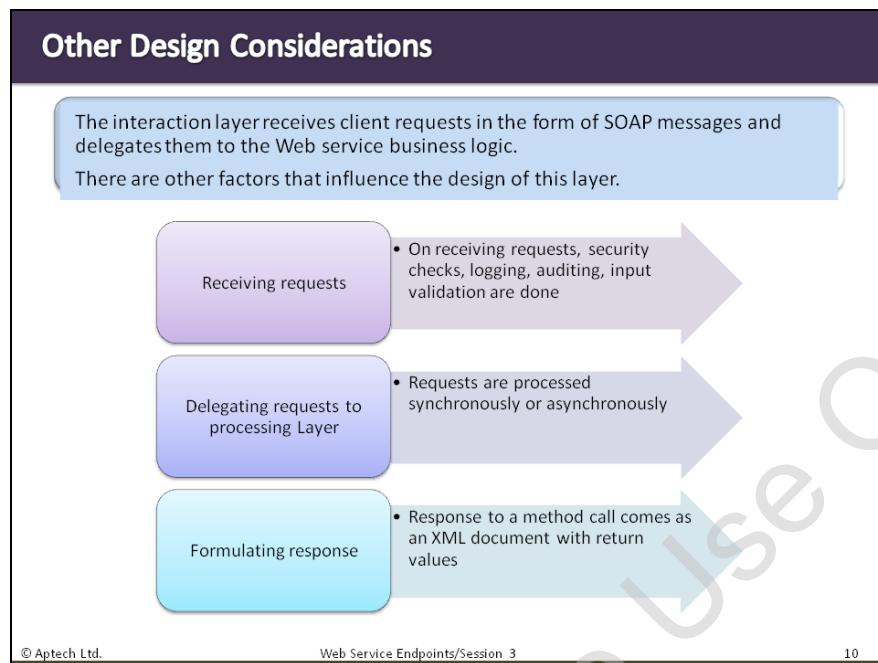
What are the two layers involved in the implementation of Web service?

#### Answer:

Service Interaction Layer and Service Processing Layer.

Other Design Considerations

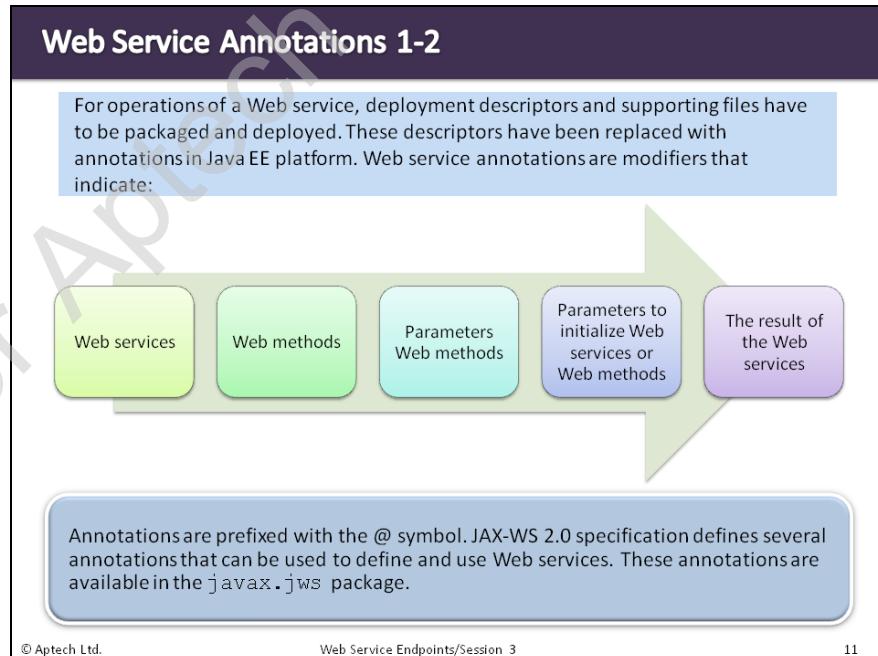
Slide 10



Using slide 10, explain that the interaction layer receives client requests in the form of SOAP messages. Describe how the interaction layer delegates the messages to the Web service business logic. Finally, explain the other factors that influence the design of the service interaction layer.

Web Service Annotations

Slides 11 and 12



## Web Service Annotations 2-2

Some of the annotations in the javax.jws package are:

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

Using slides 11 and 12, explain how the Web service becomes available to clients. Explain that the different files needed for operating a Web service including deployment descriptors are packaged into archive files before deployment. Explain that the deployment descriptors used in J2EE 1.4 are replaced by annotations in Java EE platform. Describe the Web service annotations. Then, proceed to explain about the annotations in the javax.jws package.

### In-Class Question:



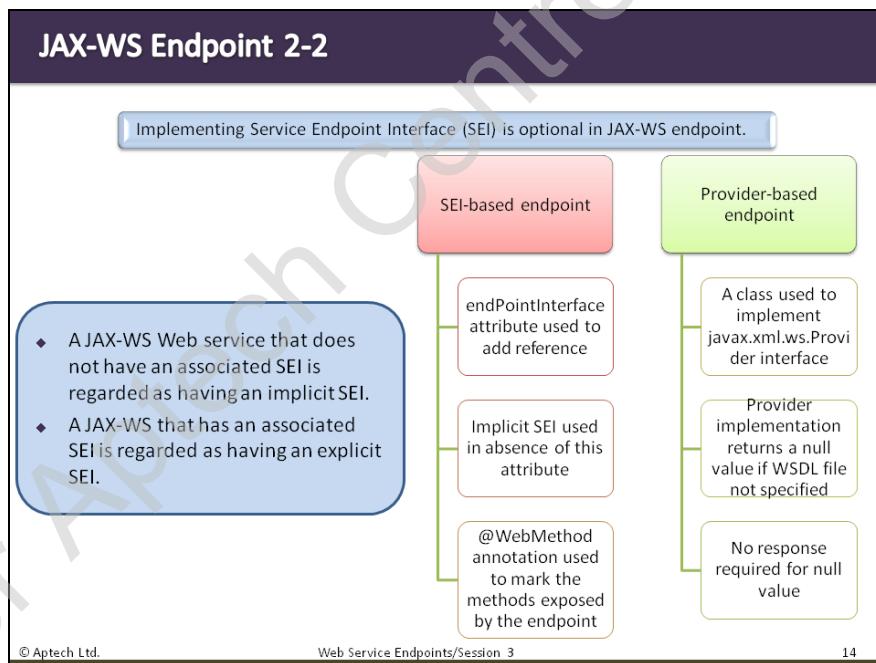
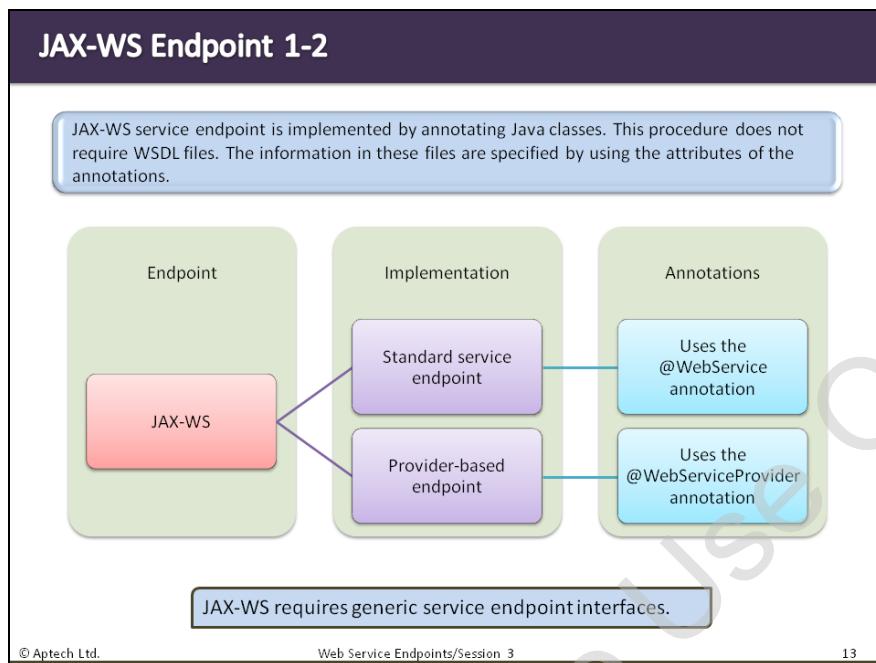
Name the Web service annotation that specifies that the Java Web Service (JWS) file implements a Web service.

### Answer:

The javax.jws.WebService (@WebService) annotation is used to specify that the Java Web Service (JWS) file implements a Web service. This annotation has five optional attributes namely, name, targetNamespace, serviceName, wsdlLocation, and endpointInterface.

**JAX-WS Endpoint**

Slides 13 and 14



Using slides 13 and 14, explain that in JAX-WS, the basis for implementing an endpoint is the standard service endpoint or a provider-based endpoint that uses the @WebServiceProvider annotation. Describe the association of a Service Endpoint Interface (SEI) with JAX-WS. Explain the reason for considering a JAX-WS as an implicit or an explicit Service Endpoint Interface (SEI). Explain the method to annotate a Java class when developing a JAX-WS Web service.

Also, explain the process to use endPointInterface attribute while creating a SEI-based endpoint. Finally, explain the points to consider while creating a Provider-based endpoint.

**Deployment Process**

Slides 15 and 16

### 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 :

```

graph LR
    A[JAX-WS guidelines used to create WSDL file] --> B[Details on binding and service elements provided separately]
    B --> C[JAX-WS is used to develop the endpoint interface and implementation]
    C --> D[Developer]
    D --> E[Generate endpoint interface and implementation]
    E --> F[Generate WSDL by mapping  
Methods → <portType>  
Parameters → <message>, <part>]
  
```

© Aptech Ltd. Web Service Endpoints/Session 3 15

### Deployment Process 2-2

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

```

graph LR
    A[A WSDL document neutral in the XML types, idioms, and error handling capabilities created] --> B[Code to marshall SOAP messages to endpoint invocations generated]
    B --> C[An implementation class for the endpoint generated]
    C --> D[Developer]
    D --> E[Creates the WSDL document]
    E --> F[Generate interface for the endpoint]
    F --> G[Generate actual implementation class containing business logic]
  
```

© Aptech Ltd. Web Service Endpoints/Session 3 16

Using slides 15 and 16, explain the factors on which the process of deployment of the Web service depends. Describe the different approaches in which a Web service can be created. Then, explain how the process of deployment is different for each of the approaches. Explain the steps to create the service implementation and then develop WSDL. Further, explain the steps to develop the WSDL and then create service implementation.

Publishing Web Service

Slides 17 to 21

### 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.

<ul style="list-style-type: none"> <li>• Used by general public</li> <li>• Used by trusted business partners</li> <li>• Used with two entities of the same organization</li> </ul>	<b>Web services</b> <ul style="list-style-type: none"> <li>❑ Description present in a WSDL document in the registry</li> <li>❑ May hold XML schemas referenced by the service description in the registry</li> <li>❑ Undeploying involves disabling and removing a service endpoint from the Web container, removing associated files, and freeing other server resources</li> </ul>
--	--

© Aptech Ltd.

Web Service Endpoints/Session 3

17

Using slide 17, describe the registries in which a Web service is published. Mention that Web services published on registries can be used by the general public, trusted business partners, or just two entities within the same organization. Explain the factors that are involved in publishing a Web service. Mention that description about the Web service is made available in the WSDL document which is published in the registry. Then, explain about undeploying a Web service.

### 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

© Aptech Ltd.

Web Service Endpoints/Session 3

18

## 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:wspl_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>
```

© Aptech Ltd.

Web Service Endpoints/Session 3

19

## 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

© Aptech Ltd.

Web Service Endpoints/Session 3

20

## 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

Using slides 18 to 21, explain the Code Snippet that demonstrates a simple Web service called 'CalculatorWS' with a Web method named add(). Explain the code line by line to show that the method returns the sum of two integers when two integer parameters are passed. Explain the Code Snippet that shows how the WSDL document defines the name of the Web service as CalculatorWS. Explain further to show how the service element is used to map the binding to the port.

### In-Class Question:



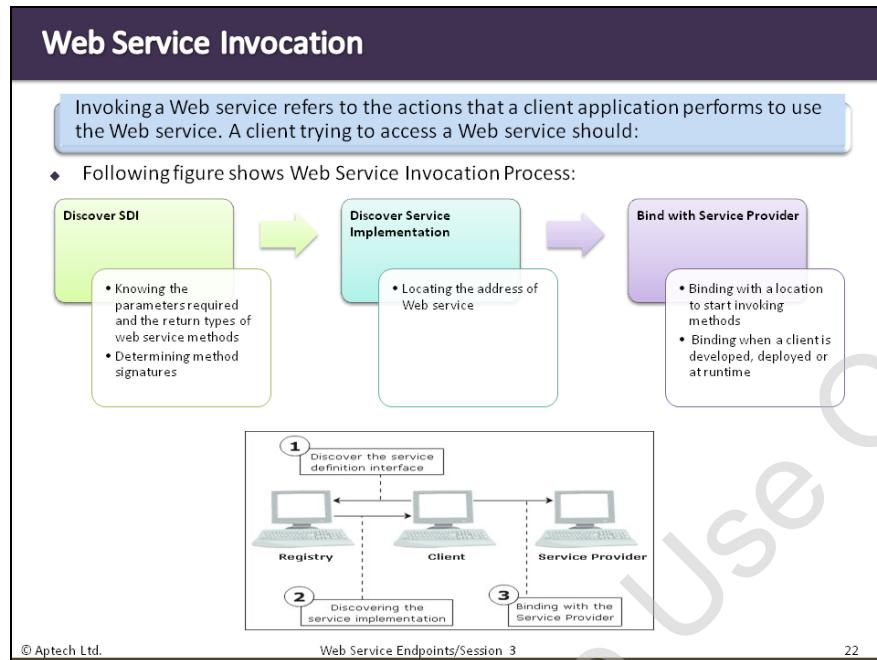
Name the three registries in which a Web service can be published.

### Answer:

Depending on the intended clientele for the Web service, the service is published on either a public registry, private registry (inter-enterprise registry), or an intra-enterprise registry.

**Web Service Invocation**

Slide 22



Using slide 22, explain Metadata Annotations of JAX-WS. Describe the annotations that are present in the `javax.jws` package. Discuss the process for invoking a Web service. Explain the steps a client must perform while trying to access a Web service. Explain the figure that shows the Web service invocation process. Explain how the client discovers the service definition interface and then process of discovering the service implementation, and finally the binding of the client with the service provider.



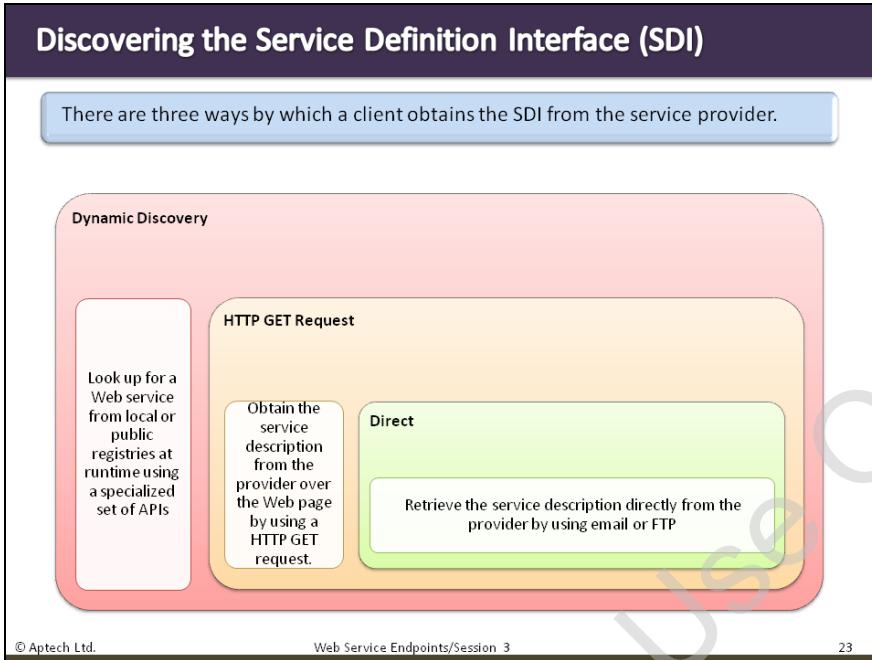
What is Web Service Invocation?

**Answer:**

Invoking a Web service refers to the actions that a client application performs to use the Web service.

Discovering the Service Definition Interface

Slides 23

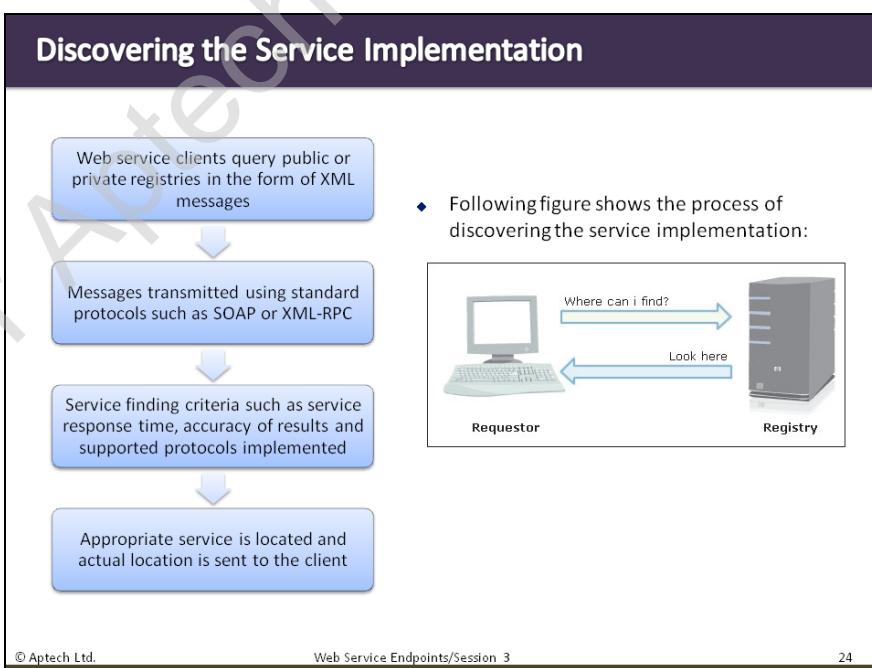


Using slide 23, explain the three ways by which a client obtains the SDI from the service provider, which are as follows:

- Direct
- HTTP GET Request
- Dynamic Discovery

Discovering the Service Implementation

Slide 24



Using slide 24, explain how the Web service clients query public or private registries. Explain the common criteria used to find a service. Describe how the actual service implementation

is obtained by the service requestor. Using the figure, describe the process of discovering the service implementation between the requestor and the registry.

### Binding with the Service Provider

Slides 25 to 27

### 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:

© Aptech Ltd. Web Service Endpoints/Session 3 25

Using slide 25, explain how the client creates a message, which is sent to the provider by using the network protocols as given in WSDL documents and how the client makes calls to the Web service. Then, proceed to explain the process of binding to a service using the figure.

### 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 using the Web service using the `getPort()` method of the Web service.

© Aptech Ltd. Web Service Endpoints/Session 3 26

### 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.

Using slides 26 and 27, explain the Code Snippet that demonstrate how to invoke a Web service using a JSP client.

### Summarize Session

#### Slide 28

### 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.

Using slide 28, summarize the session. Make them revise the following points:

- 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.

### **3.3 Post Class Activities for Faculty**

You should familiarize yourself with the topics of the next session. You should also explore the next session which describes Designing Web Service Clients.

**Tips:** You can also check the **Articles/Blogs/Expert Videos** uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

## Session 4: Designing Web Services Clients

### **4.1 Pre-Class Activities**

You should familiarize yourself with the concepts of designing Web service clients. You should know the different Modes of Communication. You should also know how to locate and access the Web services. Further, you should know how to handle Web service exceptions. Familiarize yourself with the topics of the current session in-depth.

#### **4.1.1 Objectives**

After the session, learners will be able to:

- Explain Web Service Clients
- Describe the Modes of Communication
- Explain how to Locate and Access the Web Services
- Explain how to handle Web Service Exceptions

#### **4.1.2 Teaching Skills**

To teach this session, you should be well versed with Web service clients. You should have knowledge about Dynamic client and Standalone client through which users access a Web service. You must be familiar with the features and modes of communication of Dynamic Proxy Clients, DII, Standalone Clients, and Web Tier Clients. You should be able to demonstrate how to locate and access a Web service. Further, you should be able to teach them how to handle the different Web service exceptions. For teaching in the class, you are expected to use slides and LCD projectors.

#### **Tips:**

It is recommended that you test the understanding of the students by asking questions in between the class.

#### **4.1.3 In-Class Activities:**

Follow the order given here during In-Class activities.

#### **Review of Previous Session**

You should begin with a brief recap or review of previous session. You should summarize the creation of Web service endpoints and the process of packaging, deploying, and invoking a Web service.

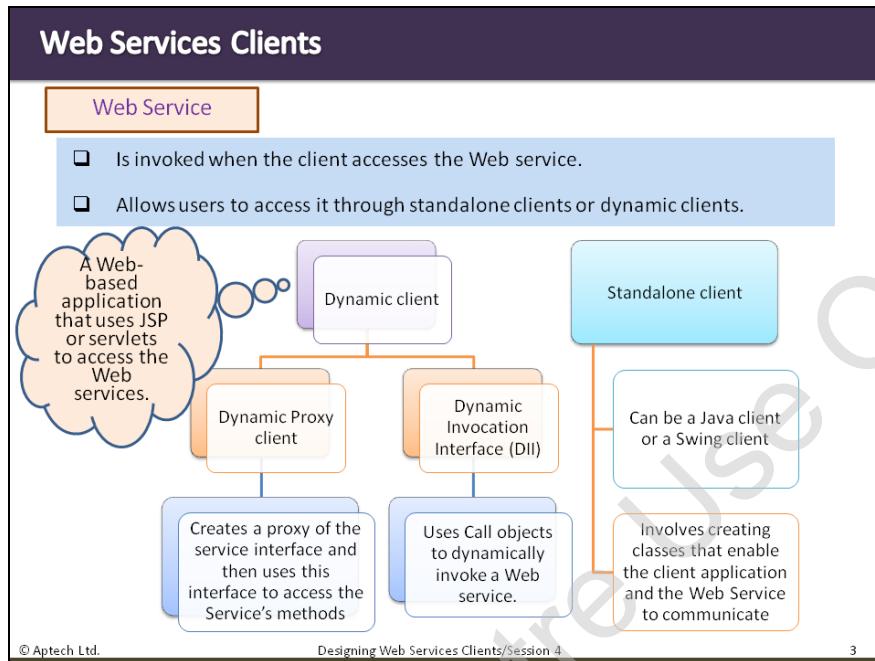
#### **Overview of the Session**

Then, give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell the students that this session explains about the Web service clients. The session describes the various modes of communication that enables the client application and the Web service to communicate. Tell them that the session explains the process of locating and accessing the Web service. Further, the session explains the method to handle Web service endpoints.

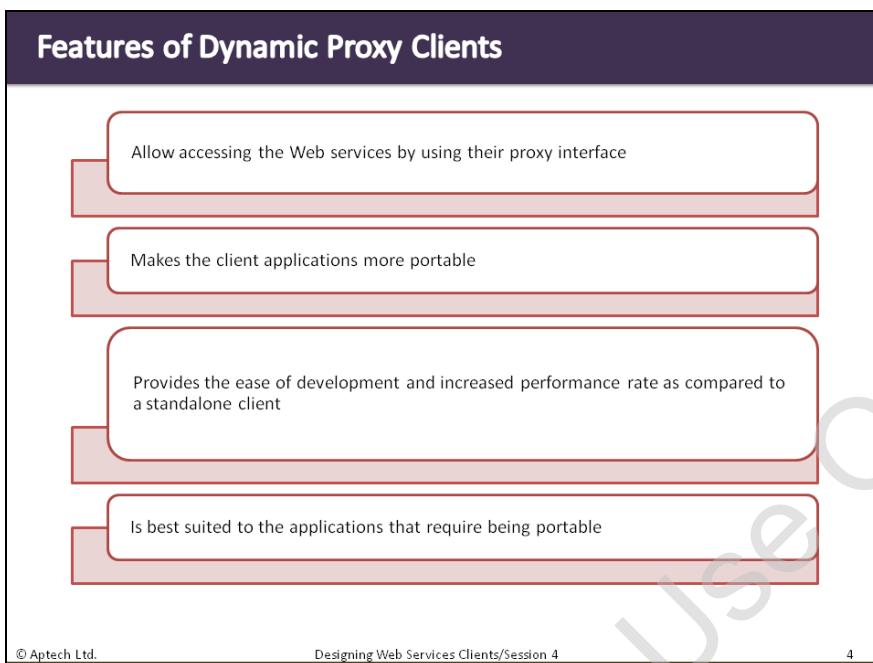
## 4.2 In-Class Explanations

### Web Service Clients

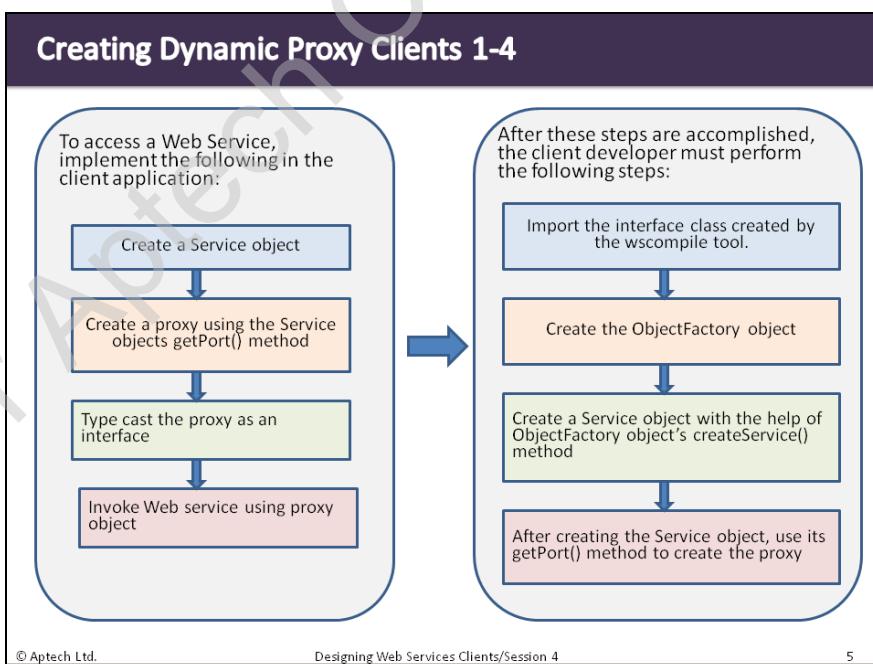
#### Slide 3



Using slide 3, explain how to invoke and access a Web service. Explain about Dynamic client and its types, namely, Dynamic Proxy client, and Dynamic Invocation Interface (DII). Also, list the types of Standalone clients and its features.

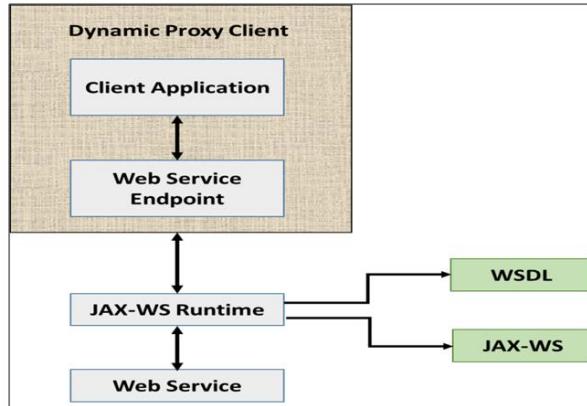
**Features of Dynamic Proxy Clients****Slide 4**

Using slide 4, explain the features of Dynamic Proxy Clients. Explain how proxy interface is better than standalone clients to access Web services. Finally, explain the reason for suitability of Dynamic Proxy for applications.

**Creating Dynamic Proxy Clients****Slides 5 to 8**

## Creating Dynamic Proxy Clients 2-4

- Following figure shows the use of Dynamic Proxy Client:



## Creating Dynamic Proxy Clients 3-4

- Following Code Snippet demonstrates the use of ObjectFactory object:

```

@XmlRegistry
public class ObjectFactory
{
    private final static QName _Add_QNAME = new
    QName("http://ws.soap.djws.com/", "add");
    private final static QName _AddResponse_QNAME = new
    QName("http://ws.soap.djws.com/", "addResponse");

    /**
     * Create a new ObjectFactory that can be used to create new instances
     * of schema derived classes for package: com.djws.soap.ws */
    public ObjectFactory()
    {
    }

    /**
     * Create an instance of {@link Add } */
    public Add createAdd()
    {
        return new Add();
    }

    /**
     * Create an instance of {@link AddResponse } */
  
```

## Creating Dynamic Proxy Clients 4-4

```

public AddResponse createAddResponse()
{
    return new AddResponse();
}
/** Create an instance of {@link JAXBELEMENT }{@code <}{@link Add }{@code
>} */ 
@XmlElementDecl(namespace = "http://ws.soap.djws.com/", name =
"add")
public JAXBELEMENT<Add> createAdd(Add value)
{
    return new JAXBELEMENT<Add>(_Add_QNAME, Add.class, null, value);
}
/** Create an instance of {@link JAXBELEMENT }{@code <}{@link AddResponse
} {@code >} */
@XmlElementDecl(namespace = "http://ws.soap.djws.com/", name =
"addResponse")
public JAXBELEMENT<AddResponse> createAddResponse(AddResponse value)
{
    return new JAXBELEMENT<AddResponse>(_AddResponse_QNAME,
AddResponse.class, null, value);
}
}

```

Using slides 5 to 8, explain the steps to access a Web service using the Dynamic Proxy approach in the client application. Then, explain the steps to be performed by the client developer.

Using the figure on slide 6, explain how the client uses Dynamic Proxy approach to access a Web service. Using slide 7 and 8, explain the use of ObjectFactory class.

### In-Class Question:



List the steps to access the Web service using Dynamic Proxy approach.

**Answer:** The steps to access the Web service using Dynamic Proxy approach are as follows:

1. Create a Service object
2. Create a proxy using the Service objects getPort() method
3. Type cast the proxy as an interface
4. Invoke Web service using proxy object

**Features of DII**

Slide 9

### Features of DII

- Enables clients to invoke the methods of Web services, which is unknown at the compile time.
- Allows the client application to lookup methods dynamically and access them through Call objects.
- Provides the client developer with complete control over the client application.
- Is the only method that supports one-way invocation.

**Drawback**

- The client developer needs to develop very complicated code.
- The effort for the same is much greater than the Static Stub and Dynamic Proxy methods.

© Aptech Ltd. Designing Web Services Clients/Session 4 9

Using slide 9, explain the features of DII Client. Explain about DII methods by highlighting that DII method is the only method that supports one-way invocation. Describe the need for developing highly complicated code by the client developer for DII.

**Creating DII Clients**

Slides 10 to 13

### Creating DII Clients 1-4

- To access a Web service using the DII method, perform the following tasks:
  - 1 • Create a Service object.
  - 2 • Create a QName class instance using the generated interface class.
  - 3 • Create the Call object and set its address.
  - 4 • Set the Call object properties.
  - 5 • Set Web service operation name.
  - 6 • Invoke Web service method.
- Following figure shows the use of DII client:

© Aptech Ltd. Designing Web Service Clients/Session 4 10

## Creating DII Clients 2-4

- Following code snippet shows creation of a DII client:

```
WebServiceClient(name = "CalculatorWS", targetNamespace =
"http://ws.soap.syskan.com/", wsdlLocation =
"http://localhost:8080/SOAPWebService/CalculatorWS?wsdl")

public class CalculatorWS_Service extends Service{

private final static URL CalculatorWS_WSDL_LOCATION;
private final static WebServiceException CalculatorWS_EXCEPTION;
private final static QName CalculatorWS_QNAME = new
QName("http://ws.soap.djws.com/", "CalculatorWS");

static {
URL url = null;
WebServiceException e = null;
try {
url = new URL("http://localhost:8080/SOAPWebService/CalculatorWS?wsdl");
}
catch (MalformedURLException ex) {
e = new WebServiceException(ex);
}
}
```

## Creating DII Clients 3-4

```
CalculatorWS_WSDL_LOCATION = url;
CalculatorWS_EXCEPTION = e;
}
public CalculatorWS_Service(){
super(__getWsdlLocation(),CalculatorWS_QNAME);
}
public CalculatorWS_Service(WebServiceFeature... features) {
super(__getWsdlLocation(),CalculatorWS_QNAME, features);
}
public CalculatorWS_Service(URL wsdlLocation) {
super(wsdlLocation, CalculatorWS_QNAME);
}
public CalculatorWS_Service(URL wsdlLocation, WebServiceFeature...
features) {
super(wsdlLocation, CalculatorWS_QNAME, features);
}
public CalculatorWS_Service(URL wsdlLocation, QName serviceName) {
super(wsdlLocation, serviceName);
}
public CalculatorWS_Service(URL wsdlLocation, QName serviceName,
WebServiceFeature... features) {
```

## Creating DII Clients 4-4

```

super(wsdlLocation, serviceName, features);
}
/** @return returns CalculatorWS*/
@WebEndpoint(name = "CalculatorWSPort")
public SOAPWS getCalculatorWSPort() {
    return super.getPort(new QName("http://ws.soap.djws.com/",
"CalculatorWSPort"), CalculatorWS.class);
}
/** @param features - A list of {@link javax.xml.ws.WebServiceFeature} to
configure on the proxy. Supported features not in the <code>features</code>
parameter will have their default values. * @return returns CalculatorWS*/
@WebEndpoint(name = "CalculatorWSPort")
public SOAPWS getCalculatorWSPort(WebServiceFeature... features) {
    return super.getPort(new QName("http://ws.soap.syskan.com/",
"CalculatorWSPort"), CalculatorWS.class, features);
}
private static URL __getWsdlLocation(){
    if (CalculatorWS_EXCEPTION!= null) {
        throw CalculatorWS_EXCEPTION;
    }
    return CalculatorWS_WSDL_LOCATION;
}

```

Using slide 10, explain the different tasks to access a Web service using the DII method. The different tasks are listed here:

- Create a Service object
- Create a QName class instance using the generated interface class
- Create the Call object and set its address
- Set the Call object properties
- Set Web Service operation name
- Invoke Web Service method

Explain the methods to be used, what input parameters are to be given, and what properties are to be set. Explain the use of DII Client using the figure. Explain how WSDL uses JAXR to locate the services.

Using slides 11 to 13, explain the Code Snippet that shows the creation of a DII client.

## Features of Standalone Clients

### Slide 14

### Features of Standalone Clients

- Provides the best performance compared to the other two dynamic approaches of accessing Web services
- Allows developers to directly code their clients against the class
- Very easy to code and implement

**Drawbacks**

- Major changes in the service interface could result in the development of the entire client application all over again.
- Using the standalone client approach in applications where the service interface frequently changes could heavily cost client developers in terms of development effort.

© Aptech Ltd.      Designing Web Service Clients/Session 4      14

Using slide 14, explain the features of Standalone Client. Also, explain the drawbacks of standalone client.

#### In-Class Question:



Among Dynamic Proxy Clients, DII Clients, and Standalone Clients, which clients provide the best performance?

#### Answer:

Standalone Clients provide the best performance when compared to the other two dynamic approaches of accessing Web Services.

Creating Standalone Clients

Slides 15 and 16

### Creating Standalone Clients 1-2

- ◆ To access a Web service using the standalone client, a client developer needs to implement the following in the client application:
  - 1 • Create a Java class to invoke the Web service method.
  - 2 • Typecast the object as an interface.
  - 3 • Set the endpoint address for the generated object.
  - 4 • Access Web service methods using generated object
- ◆ Following figure shows a standalone client:

© Aptech Ltd. Designing Web Service Clients/Session 4 15

### Creating Standalone Clients 2-2

- ◆ Following code snippet shows creation of a static stub client:

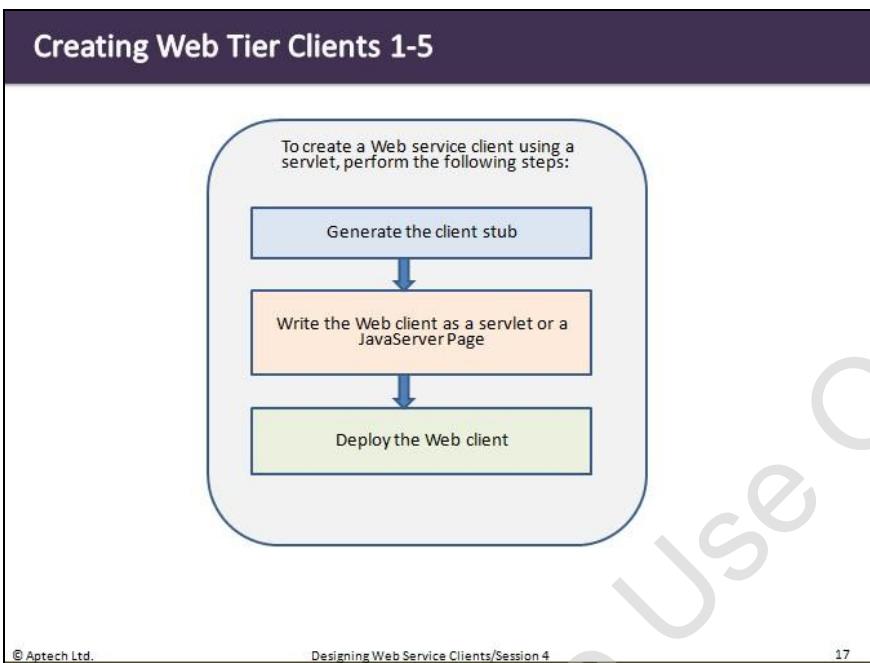
```
//stand alone client
...
public static void main(String[] args) {
try {
int num1 = 5;
int num2 = 7;
int result = addNum(num1, num2);
System.out.println("Result = " + result);
} catch (Exception ex){ System.out.println("Exception: " + ex);
}
private static int addNum(int i, int j)
{
com.djws.soap.ws.CalculatorWS_Service service = new
com.djws.soap.ws.CalculatorWS_Service();
com.djws.soap.ws.CalculatorWS port = service.getCalculatorWSPort();
return port.add(i, j);
}
...
```

© Aptech Ltd. Designing Web Service Clients/Session 4 16

Using slide 15, explain the steps to access a Web Service using the standalone client. Describe the standalone client using the figure. Using slide 16, explain the Code Snippet that shows the creation of a static stub client.

Creating Web Tier Clients

Slides 17 to 21



Using slide 17, explain a scenario of Web client accessing the CalculatorWS Web service. Inform that two numbers have to be entered by the client on the browser page of the Web service. Tell that the Web Service returns the sum and the Web component plays the role of the Web service client.

**Creating Web Tier Clients 2-5**

- Following code snippet shows creation of Web client as a servlet:

```

// Servlet Web client

...
@WebServlet(name = "JAXWSServlet", urlPatterns = {"/*"})
public class JAXWSServlet extends HttpServlet {
    @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_8080/CalculatorWS/CalculatorWS.wsdl")
    private CalculatorWS_Service service;

    /**
     * Processes requests for both HTTP <code>GET</code> and
     <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
}
  
```

## Creating Web Tier Clients 3-5

```

protected void processRequest(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet JAXWSServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form method=\"post\" action=\"\">");
        out.println("<h2>Consuming CalculatorWS Web Service Using a
Servlet Client</h2>");
        out.println("First number: <input type=\"text\" id=\"txtbox1\""
name="num1" size="6" maxlength="6" value="">");
        out.println("<br><br>");
        out.println("Second number: <input type=\"text\" id=\"txtbox2\""
name="num2" size="6" maxlength="6" value="">");
        out.println("<br><br>");
        out.println("<input type=\"submit\" name=\"btn\" value=\"Add\"/>");
    }
}

```

## Creating Web Tier Clients 4-5

```

out.println("<br><br>");
int i=0;
int j=0;
if (request.getParameter("btn").equalsIgnoreCase("Add")){
    if(request.getParameter("num1").isEmpty() ||
    request.getParameter("num2").isEmpty()){
        out.println("Specify the two numbers to be added");
    }else {
        i = Integer.parseInt(request.getParameter("num1"));
        j = Integer.parseInt(request.getParameter("num2"));

        int result = addNum(i,j);
        out.println("<h3><u>Result</u></h3> ");
        out.println("Sum of the two numbers is " + result);
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}

```

## Creating Web Tier Clients 5-5

```

    }
}

...
private int addNum(int num1, int num2) {
    // Note that the injected javax.xml.ws.Service reference as well
    // as port objects are not thread safe.
    // If the calling of port operations may lead to race condition some
    // synchronization is required.
    com.djws.CalculatorWS port = service.getCalculatorWSPort();
    return port.add(num1, num2);
}
}

```

Using slides 18 to 21, explain the Code Snippet that shows the creation of Web client as a servlet. Explain about exception handling using try-catch block in the code.

### Modes of Communication

Slide 22

## Modes of Communication

Client can communicate with Web services using:

Dynamic Proxy

Dynamic Invocation Interface

Standalone Client

Using slide 22, explain the key methods by which the client can communicate with the Web services.

**In-Class Question:**

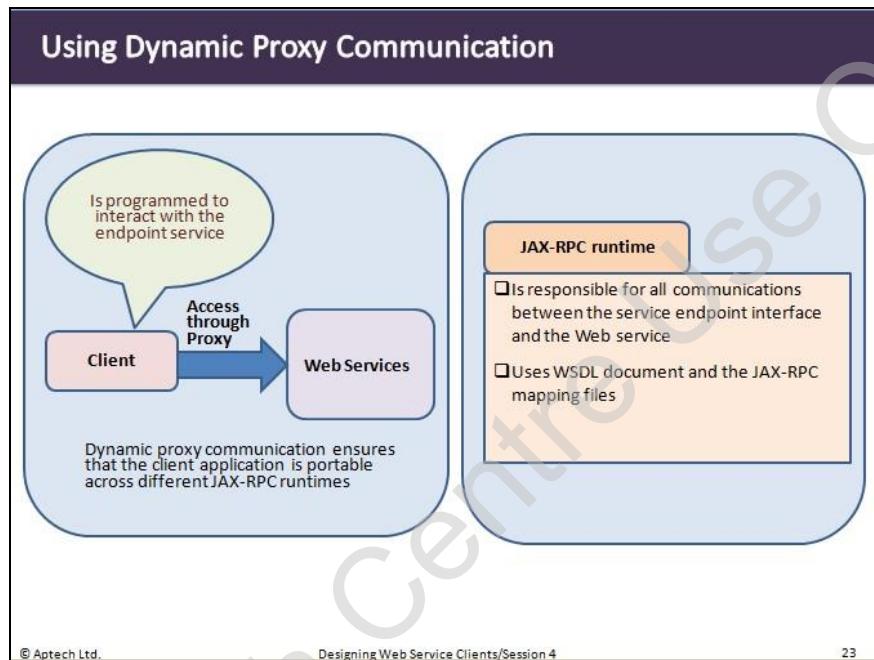
Name the three modes of communication.

**Answer:**

The client can communicate with the Web services using three methods. They are Dynamic Proxy, Dynamic Invocation Interface, and Standalone Client.

**Using Dynamic Proxy Communication**

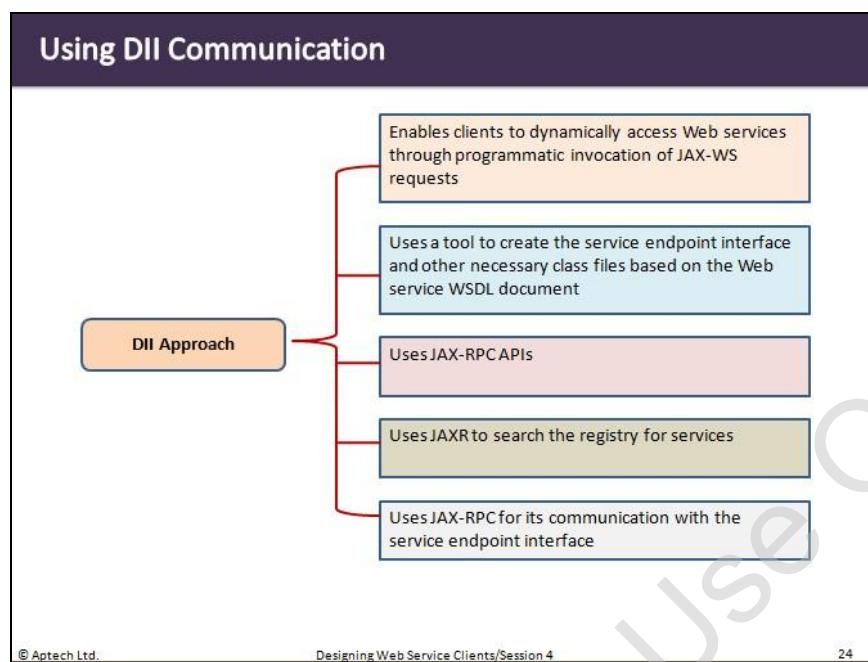
Slide 23



Using slide 23, explain Dynamic Proxy Communication. Explain how the client accesses the Web Service through its proxy. Describe how JAX-RPC runtime uses WSDL document and the JAX-RPC mapping files for all communications between the service endpoint interface and the Web Service. Explain the method to use the invocation mode. Explain the classes that developers must create when using dynamic proxies.

Using DII Communication

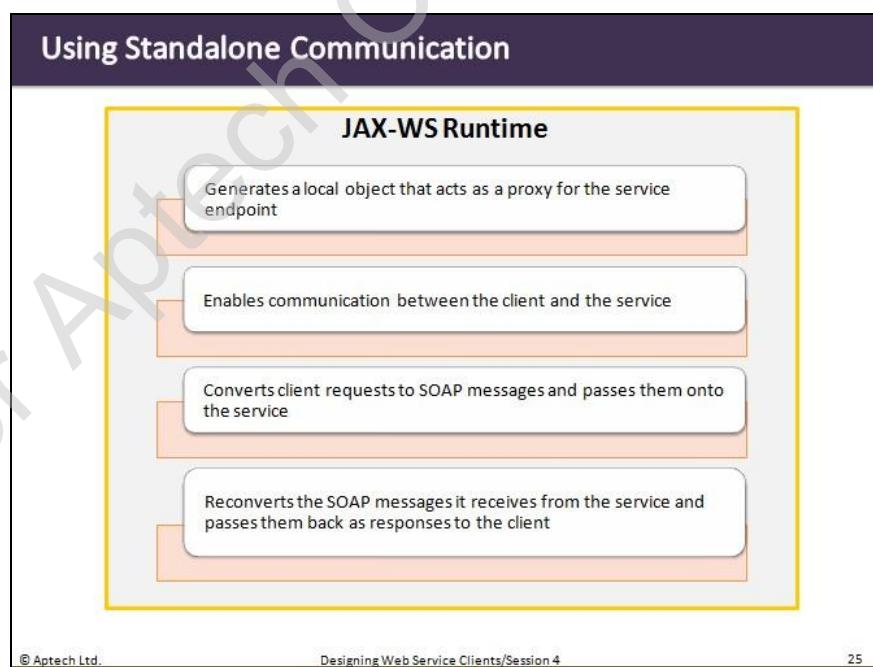
Slide 24



Using slide 24, explain DII communication. Describe about the DII approach. Explain the files that are created by a built-in tool based on the Web Service WSDL document. Explain about using JAX-RPC APIs in DII. Explain how the DII client searches the registry for services.

Using Standalone Communication

Slide 25



Using slide 25, explain how JAX-WS runtime helps in Standalone communication.

Locating and Accessing Web Services

Slide 26

### Locating and Accessing Web Services

- Following code snippet demonstrates how to locate and access a Web service:

```

@WebMethod
@WebResult(targetNamespace = "")
@RequestWrapper(localName = "add", targetNamespace =
"http://ws.soap.djws.com/", className =
"com.djws.soap.ws.Add")
@ResponseWrapper(localName = "addResponse", targetNamespace =
"http://ws.soap.djws.com/", className =
"com.djws.soap.ws.AddResponse")
@Action(input = "http://ws.soap.djws.com/CalculatorWS/addRequest", output =
"http://ws.soap.djws.com/CalculatorWS/addResponse")
public int add(
@WebParam(name = "num1", targetNamespace = "")
int num1;
@WebParam(name = "num2", targetNamespace = "")
int num2;
);

```

© Aptech Ltd.      Designing Web Service Clients/Session 4      26

Using slide 26, explain the procedure to locate and access a Web service using the Code Snippet. Describe the details on Web methods and its parameters. Discuss the annotations that have been used in the Code Snippet.

Web Service Exceptions

Slides 27 and 28

### Web Service Exceptions 1-2

- Exceptions are errors or unexpected events encountered by an executing program.
- When an exception occurs during the execution of Web service, the Web service is expected to capture not only the exception but also inform the client about the exception.

```

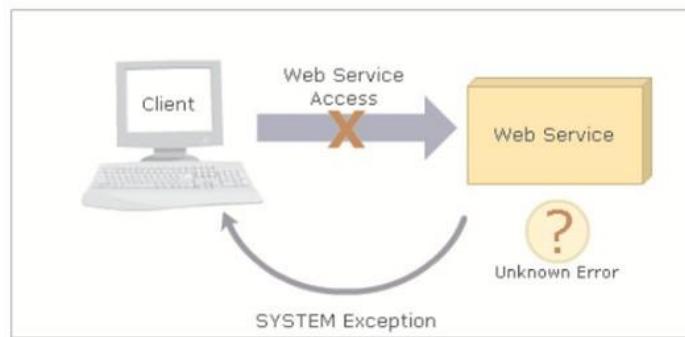
graph TD
    A[Web Service Endpoint Exceptions] --> B[System]
    A --> C[Service-Specific]
    B --> D["Are all exceptions thrown by errors beyond the control of the application"]
    B --> E["Are all unanticipated errors that can occur at the time of service method invocation"]
    C --> F["Can be handled by the client applications by:  
□ prompting the user to retry  
□ displaying the kind of exception occurred  
□ translating the system exception to an unchecked exception"]

```

© Aptech Ltd.      Designing Web Service Clients/Session 4      27

## Web Service Exceptions 2-2

- Following figure shows the types of Web service exceptions:



© Aptech Ltd.

Designing Web Service Clients/Session 4

28

Using slides 27 to 28, describe exceptions. Explain about the types of exceptions thrown by the Web Service endpoint. Explain the exceptions that are categorized as System exceptions. Provide a few examples of unanticipated errors that give rise to these exceptions. Explain how these errors can be handled by client applications.

Use the figure in slide 28 to explain the types of Web Service exceptions. Explain how the Web service throws System exception.

### In-Class Question:



What are the two types of Web service exceptions?

### Answer:

The two types of exceptions that can be thrown by a Web Service endpoint are System and Service-specific.

## System Exceptions in Web Service

Slides 29 and 30

### System Exceptions in Web Service 1-2

- ◆ Different types of system exceptions are as follows:

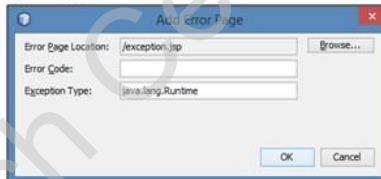
javax.xml.ws.WebServiceException	• Are thrown by the getPort() method due to insufficient data to create the proxy
javax.xml.ws.IOException	• Results due to unavailability of service, network failures, and so on give rise to IOException
javax.xml.ws.WebServiceException	• Results due to use of invalid property names and setting them with invalid values
javax.xml.ws.WebServiceException	• Occurs due to configuration errors such as invalid property names, invalid property values, type mismatch

- ◆ Following code snippet shows how runtime exceptions are handled in deployment descriptor of a Web application:

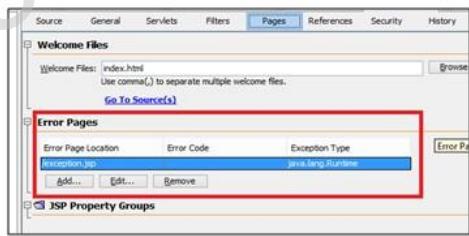
```
<error-page>
<exception-type>java.lang.Runtime</exception-type>
<location>/exception.jsp</location>
</error-page>
```

### System Exceptions in Web Service 2-2

- ◆ Following figure shows a dialog box to add exception page in the Deployment Descriptor (Web.xml) using NetBeans IDE:



- ◆ Following figure shows the result after setting up Error Page for Web.xml in NetBeans:



Using slides 29 and 30, explain about system exceptions in Web service. Using slide 29, list the different issues faced by the Web service clients during execution. Mention that these issues lead to different types of System exceptions. A javax.xml.ws.WebServiceException is thrown by dynamic proxy clients. IOException and javax.xml.ws.WebServiceException are thrown by DII Call interfaces and Standalone clients mostly face problems during the configuration of the stub. Explain the Code Snippet on slide 29 that shows how runtime exceptions are handled in deployment descriptor of a Web application.

Using slide 30, explain the figures to show how to add exception page in the Deployment Descriptor using NetBeans IDE. Show the figure that displays the result after setting up Error Page for Web.xml in NetBeans.

Service Exceptions During Web Service Calls

Slides 31 and 32

### Service Exceptions During Web Service Calls 1-2

**Service Exceptions**

- Are thrown by faults or errors generated by the client application itself
- Are also called checked exceptions in client applications
- Are a result of improper data passed to the Web service
- Are easy to determine as they are generated by the application itself
- Are generally listed as operation elements in a WSDL file and are known as wsdl:fault elements

© Aptech Ltd. Designing Web Service Clients/Session 4 31

Using slides 31, explain about Service exceptions. Explain the reason for the occurrence of these errors. Explain the reason why these errors are easy to determine and handle.

### Service Exceptions During Web Service Calls 2-2

**JAX-WS Tools**

- Can be used to map faults or errors to Java objects.
- Generate necessary exception classes and parameters to handle.

Following code snippet demonstrates the use of illegalArgumentException:

```
<fault name="illegalArgumentException" message="tns:illegalArgumentException"/>
```

© Aptech Ltd. Designing Web Service Clients/Session 4 32

Using slide 32, explain about JAX-WS tools and their usage. Explain how the client application handles these checked exceptions and recovers from such exceptions. Mention that a client accessing the CalculatorWS Web service pass two non-numeric values to the method. Explain about `illegalArgumentException` that is received by the client because the error message is defined in the WSDL document. Finally, explain the Code Snippet that shows the use of `illegalArgumentException`.

**Exception Handling Mechanism****Slides 33 to 36**

### Exception Handling Mechanism 1-4

To handle service exceptions, developers can:

- Provide appropriate mechanisms to recover from exceptions
- Can resort to boundary checking for input values
- Can use JavaScript to validate the boundaries before sending requests to a service

 In case of Java EE 7 Web Components, clients may handle service exceptions as unchecked applications, such as javax.servlet.ServletException or may divert it to an error page.

© Aptech Ltd.      Designing Web Service Clients/Session 4      33

Using slides 33, explain the role of client application developer in handling Service exceptions. Explain how clients handle service exceptions in Java EE 7 Web Components. Explain how the clients must perform boundary checking using an example such as credit card numbers consisting 16-digit integers of particular range. Point out the uses of validation, which are:

- Minimized multiple trips to the service
- Reduction of network traffic
- Increase in service access speed

### Exception Handling Mechanism 2-4

- ◆ Following code snippet demonstrates how illegalArgumentException is handled in case of non-numeral character entry:

```
try {
    com.djws.soap.ws.CalculatorWS_Service service = new
    com.djws.soap.ws.CalculatorWS_Service();
    com.djws.soap.ws.CalculatorWS port = service.getCalculatorWSPort();
    int result = port.add(num1, num2);
}
catch (illegalArgumentException cc) {
    System.out.println(cc);
}
```

- ◆ Following code snippet shows an error page:

```
<error-page>
<exception-type> illegalArgumentException</exception-type>
<location>/exception.jsp</location>
</error-page>
```

© Aptech Ltd.      Designing Web Service Clients/Session 4      34

### Exception Handling Mechanism 3-4

- Following figure shows the output which the Servlet Web service client code will produce when the page is first loaded:

- Following figure shows an error message if the **Add** button is clicked without entering any numbers in the text boxes:

© Aptech Ltd.

Designing Web Service Clients/Session 4

35

### Exception Handling Mechanism 4-4

- Following figure shows the Web page after the numbers to be added are entered in the text boxes:

- Following figure shows the sum of two numbers after the **Add** button is clicked:

© Aptech Ltd.

Designing Web Service Clients/Session 4

36

Using slides 34 to 36, explain the Code Snippet which demonstrates how an `IllegalArgumentException` in the case of non-numeral character entry is handled. Then, explain the Code Snippet that shows the error page to which the Web-tier clients are redirected in case of an exception. Finally, show the figures on slides 35 and 36 to explain the output after executing the code.

**In-Class Question:**

How do clients handle service exceptions for Java EE 7 Web Components?

**Answer:**

In case of Java EE 7 Web Components, clients may handle service exceptions as unchecked applications, such as `javax.servlet.ServletException` or may divert it to an error page.

**Summarize Session****Slide 37**

### Summary

- ◆ A Web service is invoked when the client accesses the Web service. Users can access a Web service through standalone clients or dynamic clients.
- ◆ A Dynamic client is a Web-based application that uses JSP or servlets to access the Web services. This type of client can be of two types – Dynamic Proxy client and Dynamic Invocation Interface (DII).
- ◆ A Dynamic Proxy client creates a proxy of the service interface and then uses this interface to access the Service's methods.
- ◆ A DII uses Call objects to dynamically invoke a Web service. This enables the client to invoke methods of a service even without knowing their endpoint addresses until runtime.
- ◆ A Standalone client can be a Java client or a Swing client.
- ◆ There are two types of exceptions that can be thrown by a Web service endpoint, System and Service-specific.
- ◆ All exceptions thrown by errors beyond the control of the application can be categorized as System exceptions.
- ◆ Service-specific exceptions are thrown by faults or errors generated by the client application itself.

Using slide 37, summarize the session. Make them revise the following points:

- A Web service is invoked when the client accesses the Web service. Users can access a Web service through standalone clients or dynamic clients.
- A Dynamic client is a Web-based application that uses JSP or servlets to access the Web services. This type of client can be of two types – Dynamic Proxy client and Dynamic Invocation Interface (DII).
- A Dynamic Proxy client creates a proxy of the service interface and then uses this interface to access the Service's methods.
- A DII uses Call objects to dynamically invoke a Web service. This enables the client to invoke methods of a service even without knowing their endpoint addresses until runtime.
- A Standalone client can be a Java client or a Swing client.
- There are two types of exceptions that can be thrown by a Web service endpoint, System and Service-specific.

- All exceptions thrown by errors beyond the control of the application can be categorized as System exceptions.
- Service-specific exceptions are thrown by faults or errors generated by the client application itself.

#### **4.3 Post Class Activities for Faculty**

You should familiarize yourself with the topics of the next session. You should also explore the next session which describes JAX-WS.

**Tips:** You can also check the **Articles/Blogs/Expert Videos** uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

## Session 5: JAX-WS

### **5.1 Pre-Class Activities**

You should familiarize yourself with the concept of the Java API for XML based Web services (JAX-WS). You should be able to explain the need for JAX-WS. You should also be able to explain the features of JAX-WS. In addition, you should also be able to explain JAX-WS architecture, annotations, and programming model.

Familiarize yourself with the topics of the current session in-depth.

#### **5.1.1 Objectives**

After the session, learners will be able to:

- Explain Web services on Java Enterprise Edition platform
- Explain Java API for XML based Web services (JAX-WS)
- Explain the need of JAX-WS
- Explain the features and standards of JAX-WS
- Explain the JAX-WS architecture
- Explain JAX-WS annotations
- Explain JAX-WS programming model

#### **5.1.2 Teaching Skills**

To teach this session, you should be well versed with the concept of Java API for XML based Web services (JAX-WS), its need and features along with JAX-WS architecture, standards, annotations, and programming model.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

#### **Tips:**

It is recommended that you test the understanding of the students by asking questions in between the class.

#### **5.1.3 In-Class Activities**

Follow the order given here during In-Class activities. Give a summary of Web service clients, communication modes, and handling Web service exceptions discussed in the previous session.

#### **Review of Previous Session**

You should begin with a brief recap or review of previous session.

#### **Overview of the Session**

Then, give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell the students that this session explains JAX-WS in detail. Tell them that this session also explains the need, features, and

standards of JAX-WS. Finally, the session introduces them to the JAX-WS architecture and describes JAX-WS annotations and JAX-WS programming model.

## 5.2 In-Class Explanations

### Enhancements of Web Services on Java EE Platform

#### Slide 3

**Enhancements of Web Services on Java EE Platform**

- ◆ Today, there is a need to develop enterprise applications with fewer resources in lesser time and effort.
- ◆ One of the platforms used to develop enterprise applications and Web services is Java Enterprise Edition (Java EE).
- ◆ It has more annotations, lesser XML configurations, more Plain Old Java Object (POJO), and simple packaging.
- ◆ Following table shows some of the new technologies included on the Java EE platform:

Technology	Description
Java API for XML based Web Services (JAX-WS)	It is a Java API, to create Web services on a Java platform.
Java API for RESTful Web Services (JAX-RS)	It is a Java API to support creation of Web services based on REpresentational State Transfer (REST) architecture.
Dependency Injection	It is a technique that provides the required objects to the software components.

© Aptech Ltd.
JAX-WS/Session 5

3

Using slide 3, introduce Java Platform, Enterprise Edition (Java EE) by emphasizing the need to develop enterprise applications with fewer resources in lesser time and effort. Tell that Java EE is one such platform that satisfies today's requirements. Then provide the gist of the features and advantages of Java EE Platform. Tell how Java EE platform helps the developer. Also, explain some of the new technologies included on the Java EE platform.

**'Big' Web Services****Slide 4**

## 'Big' Web Services

- ◆ 'Big' Web services or 'SOAP-based' services are based on JAX-WS in Java EE.
- ◆ SOAP is a standard protocol that defines the architecture and message formats in XML language.
- ◆ It follows HTTP protocol for request-and-response model on the Web.
- ◆ Following figure shows SOAP-based communication:

The diagram shows a Client computer sending a 'Request' message (containing XML elements like <message>...) through a 'SOAP Library' to a 'Server'. The 'Server' contains a 'Web Service' which returns a 'Response' message (containing XML elements like <ResponseMessage>...) back through another 'SOAP Library' to the 'Client'.

© Aptech Ltd. JAX-WS/Session 5 4

Using slide 4, first explain that Web services are classified as 'Big' Web services and REpresentational State Transfer or RESTful Web services. Tell that this classification is based on their implementation method. Then, explain why JAX-WS in Java EE forms the basis for 'Big' Web services' or 'Simple Object Access Protocol (SOAP) based' services. Explain WSDL and that it is used to describe Web services. Using the figure, explain SOAP-based communication.

**RESTful Web Service****Slides 5 and 6**

## RESTful Web Services 1-2

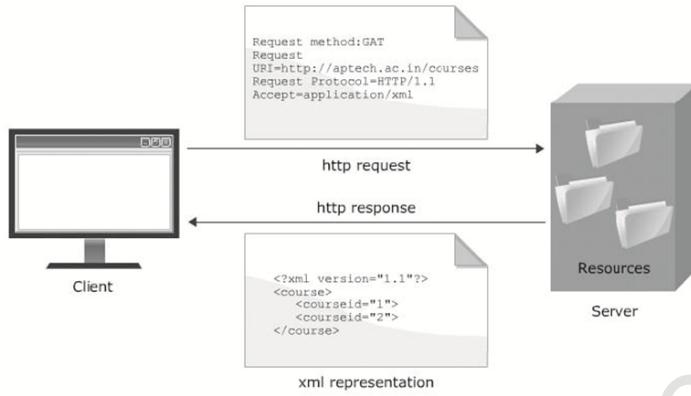
- ◆ Is a Web application that is based on client-server architecture called as the REST architecture.
- ◆ Has a request-and-response model.
- ◆ Uses HTTP, which is based on Uniform Resource Identifier (URI) on the hyperlink, to access resources.
- ◆ Following table shows the mappings of HTTP methods to their CRUD actions:

HTTP Method	CRUD Action
POST	Create a new resource from the request data
GET	Retrieve a resource
PUT	Update a resource from the request data
DELETE	Delete a resource

© Aptech Ltd. JAX-WS/Session 5 5

## RESTful Web Services 2-2

- ◆ In REST-based communication, a client initiates the request for the resource. In response, the client receives representation of the resource.
- ◆ Following figure shows REST-based communication:



© Aptech Ltd.

JAX-WS/Session 5

6

Using slides 5 and 6, explain what RESTful Web service is. Tell about the architecture on which it is based on. Point out what it uses to access resources. Tell about the various types of resources that can be accessed. Tell about the four main Hyper Text Transfer Protocol (HTTP) methods that are used to Create, Retrieve, Update, and Delete (CRUD) resources. Then detail the mappings of HTTP methods to their CRUD actions. Explain that REST uses Web standards such as HTTP, URL, and XML and JAX-RS provides the functionality for RESTful Web services. Using the figure on slide 6, explain REST-based communication.

### In-Class Question:



What are the two types of Web services?

### Answer:

'Big' Web services and RESTful Web services are the two types of Web services.

## Java API for XML Web Services (JAX-WS)

### Slide 7

### Java API for XML Web Services (JAX-WS)

- ◆ Web services support a few standards that help to develop interoperable applications.  
The key standards are as follows:

- WSDL – To define the service
- SOAP protocols – To exchange XML messages over different transport protocols (HTTP, SMTP, and JMS)
- Java API for XML-based Remote Procedure Call (JAX-RPC) – To invoke a Web service on Java platform

© Aptech Ltd.      JAX-WS/Session 5      7

Using slide 7, discuss the key standards that help to develop interoperable applications. Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) protocols, and Java API for XML-based Remote Procedure Call (JAX-RPC). Explain the usage of these standards.

Point out that JAX-RPC has a few APIs based on its own data bindings, which support SOAP protocol over HTTP to exchange XML messages. Tell that to support new industry standards, JAX-RPC was modified and renamed with JAX-WS, which makes the development of Web service applications simple and easy. Finally, explain the advantages of JAX-WS.

**JAX-WS Standards****Slide 8**

JAX-WS Standards	
◆ Following table shows the new standards of JAX-WS programming model:	
Standard	Description
SOAP 1.2	It is a lightweight protocol for structured information exchange of text and binary data.
XML/HTTP	It assists in transforming XML messages over transport protocol, HTTP without SOAP.
Web Services Interoperability (WS-I)	It has WS-I Basic Profile document that has specifications for SOAP and WSDL. WS-I Basic Profile 2.0 has the specified encoding styles, proxy generations, and dynamic invocation of interfaces.
Data Mapping Model	It specifies the mapping of XML elements to Java classes. JAXB 2.0 promotes the mappings for all XML schemas. It is supported by JAX-WS for data mapping.
Interface Mapping Model	It is used to map service interfaces with service implementation classes.
Dynamic Programming Model	It is used for message-oriented invocations and asynchronous invocations.
Handler Model	It processes the SOAP message before and after the messages are sent over the network in the Web service development.

Using slide 8, explain the new standards of JAX-WS programming model. Explain each standard and its usage.

**Features of JAX-WS****Slide 9**

Features of JAX-WS	
	<b>Platform Independence</b>
	<b>Annotations</b>
	<b>Invocation of Web Services</b>
	<b>Data Binding with Java Architecture for XML Binding (JAXB 2.0)</b>
	<b>Support for SOAP 1.2</b>
	<b>Support for Message Transmission Optimized Mechanism (MTOM)</b>
	<b>Dynamic and Static Clients</b>

Using slide 9, explain the key features of JAX-WS, which are as follows:

- Platform Independence
- Annotations
- Invocation of Web services

- Synchronous invocation
- Asynchronous invocation Data Binding with Java Architecture for XML Binding (JAXB 2.0)
- Support for SOAP 1.2
- Support for Message Transmission Optimized Mechanism (MTOM)
- Dynamic and Static Clients
- Development Tools

While explaining Platform Independence, describe the delegate class that is generated internally by JAX-WS that helps to achieve better platform independence. While explaining Annotations, tell that they help to expose Java artifacts as Web service. While explaining about Invocation of Web services, point out the two types of client-side invocations supported by JAX-WS, namely, synchronous and asynchronous invocation.

Discuss the two types of asynchronous invocation approaches: Polling approach and Callback approach. Tell for what JAXB 2.0, SOAP 1.2, Message Transmission Optimized Mechanism (MTOM), and dynamic client API called javax.ws.Dispatch are used. Discuss the constants in the dispatch client. Tell about PAYLOAD mode and MESSAGE mode. Point out the two command tools for Web service development, namely, wsimport tool and wsgen tool.

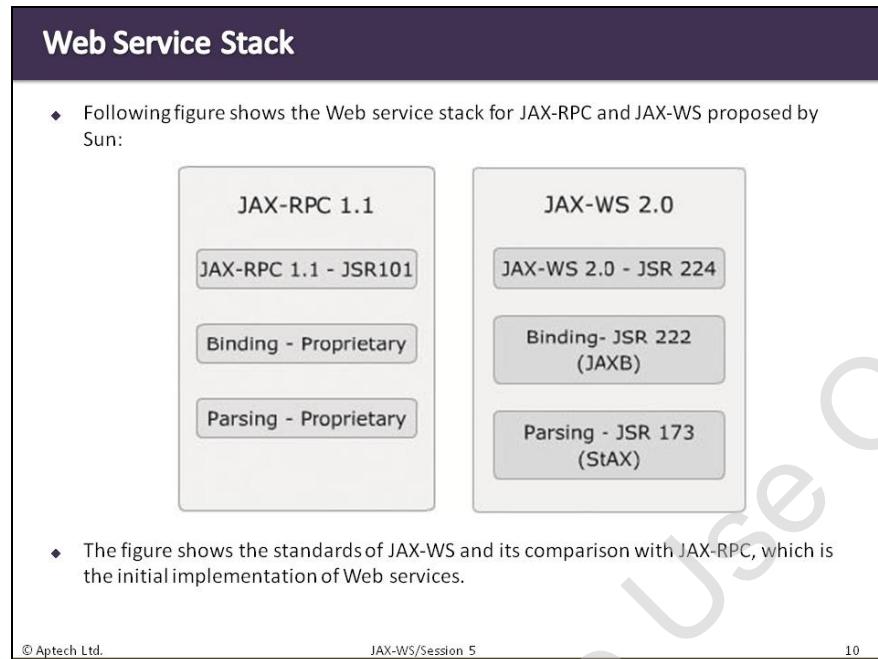
#### In-Class Question:



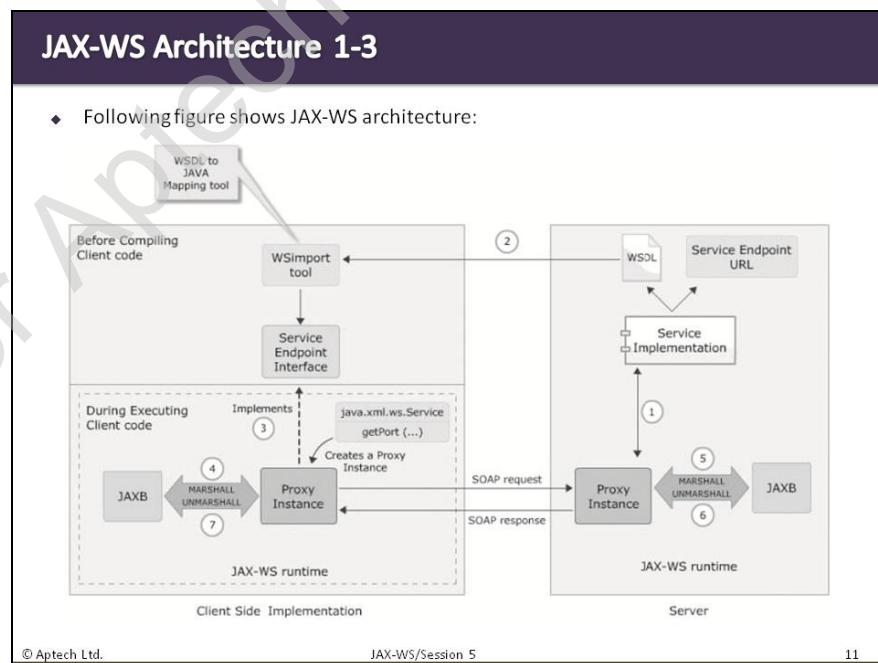
Name the mechanism supported by JAX-WS for optimized transmission of binary data in Web service.

#### Answer:

Support for Message Transmission Optimized Mechanism (MTOM). MTOM is a mechanism supported by JAX-WS for optimized transmission of binary data in Web service.

**Web Service Stack****Slide 10**

Using slide 10, explain that JAX-RPC has its own data binding and parsing mechanism, which are based on Java Architecture for XML Binding (JAXB) and Streaming API for XML (StAX). Using the figure, compare JAX-WS with JAX-RPC, which is the initial implementation of Web services.

**JAX-WS Architecture****Slides 11 to 13**

## JAX-WS Architecture 2-3

- ◆ Following are the steps to invoke the Web:

  1. **At the server-side:** The development of JAX-WS starts with an annotated Web service implementation class. This implementation class is deployed on an application server. Then, the wsigen tool generates the WSDL file, which has information on service methods and the SEI. While deploying, a server-side proxy instance is generated to handle the request and response.
  2. **At the client-side:** The Web service is invoked using a command tool called wsimport. This tool is a WSDL to Java mapping tool. It generates the SEI on the client-side. Then, the client code for accessing the Web service is compiled and executed.
  3. **After the execution of client application code:** JAX-WS runtime generates a proxy class instance. This instance internally implements the generated SEI. The proxy class is invoked by the getPort() method from javax.xml.ws.Service class. The generated proxy instance is used for invoking the Web service method.
  4. **After invoking the method on Web service by the client:** The JAX-WS runtime uses JAXB to marshal the parameters of the method. Then, these objects are encapsulated in SOAP message in XML format. These are sent as a SOAP request across the network. The parameters are passed to proxy instance to be passed to the service present at the server.

## JAX-WS Architecture 3-3

5. **After the SOAP request is received by the server:** The JAX-WS runtime environment uses JAXB to unmarshal the SOAP request. Unmarshalling converts the SOAP message having XML schema elements into Java objects. The message is given to the server-side proxy to execute the Web service method.
6. **At the server-side:** The result of the Web service method is marshalled again using JAXB. JAXB wraps the response in the SOAP message, to enable the server-side proxy to send it back to the client.
7. **At the client-side:** The JAX-WS runtime unmarshalls the SOAP response message using JAXB. Then, it passes the unmarshalled Java object to the proxy. The proxy returns the result from the Java object to the client application.

Using slides 11 to 13, explain that the generation of dynamic proxy class instance is the basis for JAX-WS architecture. Tell that this instance sends and receives the SOAP request and SOAP response to the client and server. Then, explain the steps to invoke the Web service.

**JAX-WS Annotations****Slide 14****JAX-WS Annotations**

JAX-WS has a library of annotations to simplify Web services development.

Annotations enable easy conversion of a POJO class to a Web service.

At the server, annotations describe the way of accessing an implementation class as a Web service.

At the client, annotations describe the way a client-side Java class accesses Web service.

The application uses the metadata information defined on service implementation class or service interface. Specialized tools process the metadata to generate the underlying code.

Annotations provide attributes that can be used to keep the required information in the implementation class itself.

Therefore, it is optional to have webservices.xml deployment descriptor file.

Using slide 14, explain the JAX-WS Annotations. Explain the benefits of using JAX-WS annotations. Discuss the usage of annotations at the server-side and at the client-side.

**Web Services Metadata Annotations****Slides 15 to 19****Web Services Metadata Annotations 1-5****@WebService Annotation**

- Following table shows the attributes of javax.jws.WebService annotation:

Attribute	Description
name	It specifies the name of the Web service. It is represented by wsdl:portType. Its default value is the name of Java class/interface.
targetNamespace	It specifies the XML namespace of the XML elements. Its default value is the namespace mapped to a package name of Web service.
serviceName	It specifies the service name of the Web service. It is mapped in the WSDL file using wsdl:service. Its default value is the Java class/interface and service.
endpointInterface	It helps to separate the service interface from the implementation class. If this property is not specified, then the Web service interface is generated by the implementation class.
wsdlLocation	It indicates the relative/absolute Web address of the WSDL file that defines the Web service.

## Web Services Metadata Annotations 2-5

### @WebMethod Annotation

- Following table shows the attributes of javax.jws.WebMethod annotation:

Attribute	Description
operationName	It maps the method to wsdl:operation element in the WSDL file. It defines the operation type performed by the Web service method. The default value is the name of the Java method as a string.
action	It defines the action for this operation. It finds out SOAP action header value for SOAP bindings. Its default value is a string.

### @WebResult Annotation

- Following table lists the attributes of javax.jws.WebResult annotation:

Attribute	Description
name	It specifies the variable name that is returned after invoking the Web service method. In document style operations, the name parameter is the local name of XML element, which is the return value.
targetNamespace	It specifies the XML namespace for the value returned. When the return value maps to an XML element, this is used to bind documents. The default value of targetNamespace is empty namespace, represented as "".

## Web Services Metadata Annotations 3-5

### @WebParam Annotation

- Following table shows the attributes of javax.jws.WebParam annotation:

Attribute	Description
name	It specifies the name of the parameter.
targetNamespace	It specifies the XML namespace for the parameter. This attribute has to be applied only when the operation is document-style or the parameter maps to a header. Its default value is the Web service's namespace.
mode	It shows the direction in which the parameters are passed and returned in the method. Its values can be IN, OUT, and INOUT. Its default value is IN.

## Web Services Metadata Annotations 4-5

### @SOAPBinding Annotation

- Following table lists the attributes of javax.jws.soap.SOAPBinding annotation:

Attribute	Description
style	It defines the message encoding style of Web services. Its value can either be DOCUMENT or RPC. Its default value is javax.jws.soap.SOAPBinding.Style.DOCUMENT.
use	It defines the message formatting style in Web services. Its default value is LITERAL (javax.jws.soap.SOAPBinding.Use.LITERAL).
parameterStyle	It determines the way the method parameters are placed in the message body. Its default value can be BARE or WRAPPED. If the value is set to BARE, it implies that each parameter is placed as a child element in the message body. If the value is set to WRAPPED, it means that all input/output parameters are in a single element in the respective request/response message. Its default value is javax.jws.soap.SOAPBinding.ParameterStyle.WRAPPED.

## Web Services Metadata Annotations 5-5

### @SOAPMessageHandler Annotation

- Following table lists the attributes of javax.jws.soap.SOAPMessageHandler annotation:

Attribute	Description
name	Specifies the name of SOAP message handler. The default value of this attribute is the name of the class that implements the Handler interface.
className	Specifies name of the Handler class.
initParams	Specifies the array of the name/value pairs that will be passed to the handler during initialization.
roles	Specifies the list of SOAP roles that the handler implements.
headers	Specifies the list of headers that the handler processes.

### @initParams Annotation

- Following table lists the attributes of javax.jws.soap.initParams annotation:

Attribute	Description
name	Specifies the name of the initialization parameter.
value	Specifies the value of the initialization parameter.

Using slides 15 to 19, explain Web service metadata. Point out that these annotations are supported for Web both JAX-WS and JAX-RPC application. Explain the different annotations present in the javax.jws package, which are as follows:

- @WebService Annotation
- @WebMethod Annotation
- @WebParam Annotation
- @WebResult Annotation
- @SOAPBinding Annotation
- @SOAPMessageHandler Annotation
- @initParams Annotation

For each of the mentioned annotations, explain the different attributes and what each attribute specifies or defines.

**In-Class Question:**

What is @WebService Annotation?

**Answer:**

@WebService annotation marks a Java class as a Web service or a Java interface as a Web service interface. This is present in all endpoint Java implementation classes.

**Core JAX-WS API Annotations**

Slides 20 to 23

### Core JAX-WS API Annotations 1-4

There are a few core annotations of JAX-WS API that specify the metadata associated with Web service implementations.

At runtime, these annotations simplify the process of developing Web services. They help to map Java to WSDL and schema.

They also respond to Web service invocations and help to control the JAX-WS runtime processes.

© Aptech Ltd. JAX-WS/Session 5 20

### Core JAX-WS API Annotations 2-4

**@RequestWrapper Annotation**

- Following table lists the attributes of javax.xml.ws.RequestWrapper annotation:

Attribute	Description
localName	It describes the local name of the wrapper element in the message request. It is in the form of XML. Its default value is the name of the method annotated with @WebMethod.
targetNamespace	It indicates the namespace of the request wrapper element. Its default value is the targetNamespace of the SEI.
className	It indicates the name of the request wrapper class.

**@ResponseWrapper Annotation**

- Following table shows the attributes of javax.xml.ws.ResponseWrapper annotation:

Attribute	Description
localName	It describes the wrapper element's local name in the XML message response. Its default value is the method name annotated with @WebMethod.
targetNamespace	It indicates the namespace of the request wrapper element. Its default value is the targetNamespace attribute value of the SEI.
className	It indicates the name of the response wrapper class.

© Aptech Ltd. JAX-WS/Session 5 21

## Core JAX-WS API Annotations 3-4

### @WebServiceProvider Annotation

- Following table lists the attributes of javax.xml.ws.WebServiceProvider annotation:

Attribute	Description
wsdlLocation	It is a mandatory attribute. It defines the Web location of the WSDL file that describes the Web service.
portName	It defines the service end point and maps to the wsdl:port element of WSDL file.
serviceName	It defines the service endpoint and maps to the wsdl:service element of WSDL file. Its default value is the unqualified name of the Java class, which is appended with the service keyword.
targetNamespace	It specifies the WSDL generated from Web service and the XML namespace of the XML elements. Its default value is the namespace, which is mapped to a package name of the Web service.

## Core JAX-WS API Annotations 4-4

### @ServiceMode Annotation

- This annotation is used to develop JAX-WS Web services implementation class with the help of the Web service provider.

### @WebServiceRef Annotation

- Following table lists the attributes of javax.xml.ws.WebServiceRef annotation:

Attribute	Description
name	It specifies the JNDI name of the resource. It maps the annotated method to its related Java bean property name. It maps to the default field name in case of field annotation.
type	It specifies the resource of Java type. It maps the annotated method type to its related Java bean property type. It maps to the default field name in case of annotated field.
mappedName	It specifies that a resource is mapped to a product specific name.
value	It specifies that javax.xml.ws.Service is extended by the service class. Its reference is of SEI.

Using slides 20 to 23, list and explain the Core JAX-WS API Annotations. Point out the uses of these annotations. They are:

- Simplify Web services development
- Map Java to WSDL and schema
- Respond to Web service invocations
- Help to control the JAX-WS runtime processes

Explain the different annotations, which are listed here:

- @RequestWrapper Annotation
- @ResponseWrapper Annotation
- @WebServiceProvider Annotation
- @ServiceMode Annotation

- @WebServiceRef Annotation

For each of these annotations, explain the different attributes and what each attribute specifies or defines.

**In-Class Question:**



For what is @serviceMode Annotation used?

**Answer:**

This annotation is used to develop JAX-WS Web services implementation class with the help of the Web service provider.

**Design Requirements of a JAX-WS Application**

Slide 24

**Design Requirements of a JAX-WS Application**

- ◆ The two types of service endpoint implementations supported by JAX-WS, which make services to work at XML message level are as follows:
  - The standard JavaBeans service endpoint interface
  - A new Provider interface

© Aptech Ltd.      JAX-WS/Session 5      24

Using slide 24, explain the two types of service endpoints implementation supported by JAX-WS. Further, point out that javax.jws.WebService or JAXWS javax.jws.WebServiceProvider annotation can be used to develop JAX-WS Web service. Add that @WebService can be used for JavaBeans endpoints and @WebServiceProvider can be used for provider endpoints.

## Requirements of Service Endpoints in JAX-WS

Slide 25

### Requirements of Service Endpoints in JAX-WS

- Following table lists the requirements for JAX-WS service endpoints as Dos and Don'ts:

Dos	Don'ts
<ol style="list-style-type: none"> <li>Use javax.jws.WebService or javax.jws.WebServiceProvider annotation to annotate the Web service implementation class.</li> <li>Declare the business methods as public.</li> <li>Use javax.jws.WebMethod annotation to declare the business methods.</li> <li>javax.jws.WebMethod annotation should have JAXB compatible parameters and return types. This has to be done for business methods that are exposed to Web service clients.</li> <li>The life cycle event callback methods can be contained in implementing class. The methods include javax.annotation.PostConstruct or javax.annotation.PreDestroy.</li> </ol>	<ol style="list-style-type: none"> <li>Do not mention the endpoint interface element explicitly in @WebService annotation as the SEI is implicitly defined. An implementation class can reference the SEI.</li> <li>Do not declare business methods as static or final in the implementation class.</li> <li>Do not declare the implementing class as abstract or final.</li> <li>Do not have finalize() method in the implementing class.</li> </ol>

Using slide 25, explain the dos and don'ts of the requirements of JAX-WS service endpoints.

## JAX-WS Client Model

Slides 26 and 27

### JAX-WS Client Model 1-2

- Following are the types of clients in Web Service programming model:

#### Dynamic proxy client

- It is also called as static stub client model in JAX-WS.
- SEI is used to invoke Web services.
- Dynamic Proxy Client is used to implement Web services.
- In JAX-RPC, tools are used to generate stub client.

#### Dispatch client

- It is also called as dynamic dispatch client model in JAX-WS.
- This model is flexible to write XML constructs and is generic.

- Data can be sent by the dispatch client API to construct XML message in any of the following modes:

#### PAYLOAD mode

- In the javax.xml.ws.Service.Mode.PAYLOAD mode, the dispatch client provides the content of the soap:Body element.
- The JAX-WS includes the payload in the soap:Envelope element.

#### MESSAGE mode

- In the javax.xml.ws.Service.Mode.MESSAGE mode, the dispatch client provides the entire SOAP envelope.
- It includes the soap:Envelope, soap:Header, and soap:Body elements.

## JAX-WS Client Model 2-2

- Following table lists the methods of javax.xml.ws.Dispatch interface:

Attribute	Description
T invoke(T msg)	It specifies synchronous invocation of Web service operation. It has a parameter msg. This parameter represents a message object that can have the entire message or payload section of the message. The msg object is marshalled as per the underlying protocol binding.
Response<T> invokeAsync(T msg)	It specifies asynchronous invocation of the Web service operation. It returns to the client without waiting for the response from the Web service method. The response is received by polling.
java.util.concurrent.Future<?> invokeAsync(Tmsg,AsyncHandler<T> handler)	It specifies asynchronous invocation of the Web service operation. It returns to the client without waiting for the response from the Web service method. The response is received by the handler object.

Using slides 26 and 27, explain the types of clients in Web service programming model which are Dynamic Proxy Client and Dispatch Client. Explain why they are used. Tell that the data can be sent by Dispatch client using PAYLOAD mode or MESSAGE mode. Also, explain the methods of javax.xml.ws Dispatch interface. Tell what each method specifies, the parameters in the method, and how the response is received.

### In-Class Question:



What are the two types of clients in Web service programming model?

### Answer:

Dynamic Proxy client and Dispatch Client

## Creating a Simple Web Service Application Using JAX-WS

Slides 28 and 29

### Creating a Simple Web Service Application Using JAX-WS 1-2

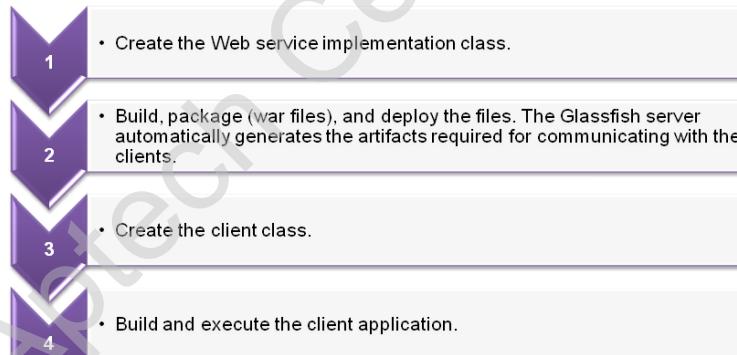
- ◆ A Web service application can be developed using JAX-WS by annotating the Java class with the @WebService annotation.
- ◆ By annotating, the class can be defined as a Web service endpoint. The files required to develop a Web service application using JAX-WS implementation are:
  - ◆ Service Interface
  - ◆ Service Implementation Class

Consider a scenario where employee details, such as employee ID, employee name, and employee salary are stored in a database table named 'Employee', as shown in the following figure:

select * from APP.EMPLOYEE...			
#	EMPID	EMPLOYEENAME	EMPLOYEESALARY
1	EMP111	John Smith	8000.0
2	EMP112	Michelle Sawyer	8500.0
3	EMP113	Brenda Taylor	9000.0
4	EMP114	Alex Johnson	9500.0

### Creating a Simple Web Service Application Using JAX-WS 2-2

- ◆ Following are the steps to create a Web service application and client:



Using slides 28 and 29, explain that @WebService annotation can be used to develop a Web service application. Tell about the two files that are required to develop a Web service application using JAX-WS implementation.

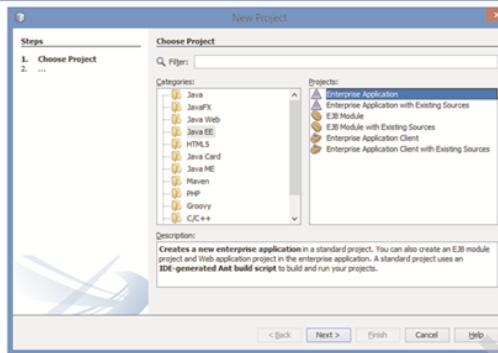
Clarify when it is necessary and when it is not necessary to have service implementation class to implement service interface. Explain the four important steps to create a Web service application and client by using a scenario, where employee details, such as employee ID, employee name, and employee salary are stored in 'Employee' database.

## Create a Web Application Implementation Class

Slides 30 to 39

### Create a Web Service Implementation Class 1-10

- To create the Web service implementation class, perform the following steps:
- In the NetBeans IDE, click File → New Project → Others and then, in the New Project wizard, select Java EE from the Categories list and Enterprise Application from the Projects list, as shown in the following figure:



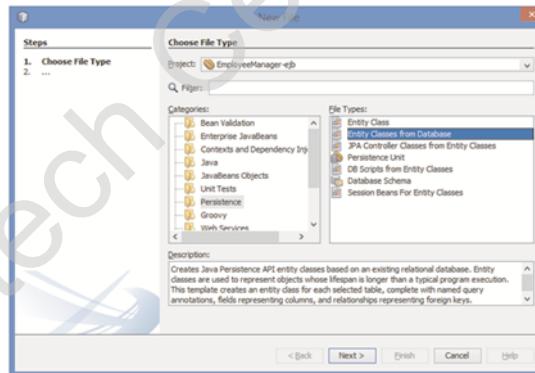
© Aptech Ltd.

JAX-WS/Session 5

30

### Create a Web Service Implementation Class 2-10

To use entity beans to create the Web service, entity classes need to be created from the Employee database. To do this, right-click EmployeeManager-ejb and then, click New → Other. Then, in the New File wizard, select Persistence in the Categories list and Entity Classes from Database in the File Types list, as shown in the following figure:



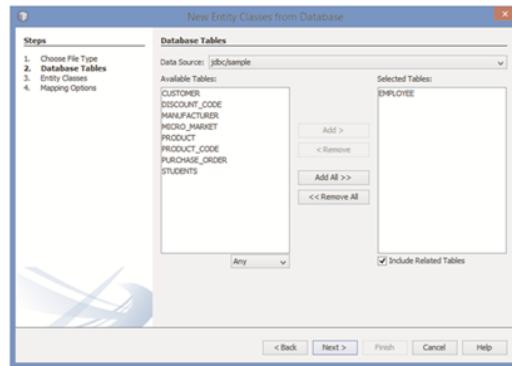
© Aptech Ltd.

JAX-WS/Session 5

31

## Create a Web Service Implementation Class 3-10

Next, to select the database to use, in the Database Source drop-down list, select jdbc/sample. All the tables associated with the selected source will be displayed in the Available Tables list. Select EMPLOYEE and then, click the Add button to add the table to the Selected Tables list, as shown in the following figure:



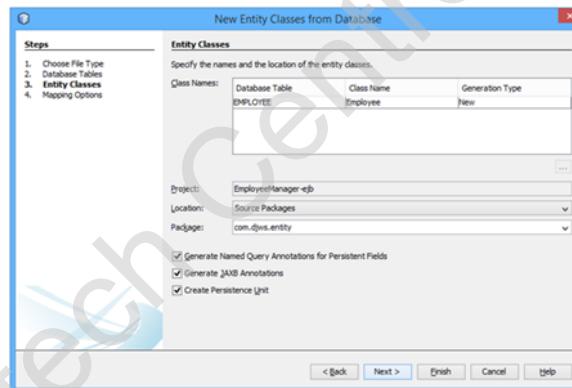
© Aptech Ltd.

JAX-WS/Session 5

32

## Create a Web Service Implementation Class 4-10

On the next page of the wizard, specify a package for the entity classes and then, click Finish, as shown in the following figure:



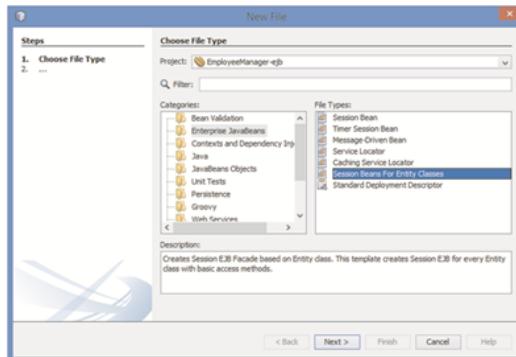
© Aptech Ltd.

JAX-WS/Session 5

33

## Create a Web Service Implementation Class 5-10

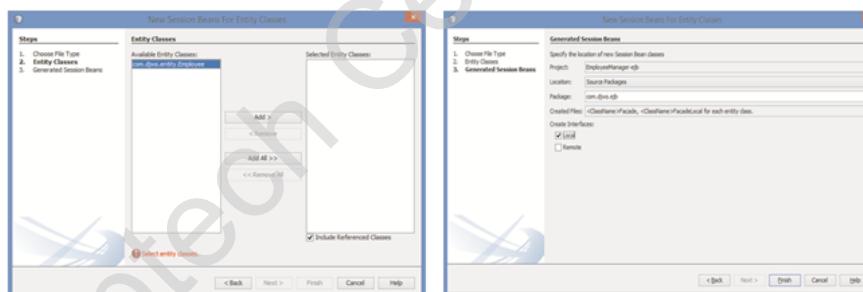
To create session beans from the entity classes, right-click EmployeeManager-ejb and then, click New → Others. Then, in the New File wizard, select Enterprise JavaBeans from the Categories list and Session Beans for Entity Classes from the File Types list, as shown in the following figure:



## Create a Web Service Implementation Class 6-10

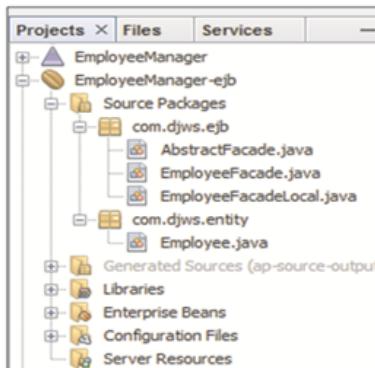
On the next page of the wizard, select the required entity classes from the Available Entity Classes list and then, click the Add button, as shown in the following figure:

Next, specify a package name for the session beans and select Local to create local interfaces, as shown in the following figure:

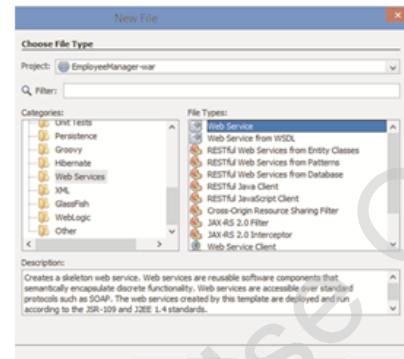


## Create a Web Service Implementation Class 7-10

Three Java classes will be created. In this example, the three Java classes are AbstractFacade, EmployeeFacade, and EmployeeFacadeLocal, as shown in the following figure:



To create a Web Service from the sessions beans, right-click EmployeeManager-war, and then, click New → Other. Then, in the New File wizard, select Web Services in the Categories list and Web Service in the File Types list, as shown in the following figure:



© Aptech Ltd.

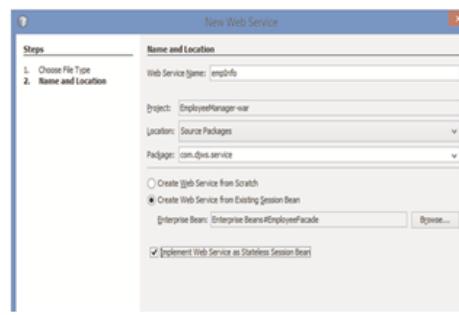
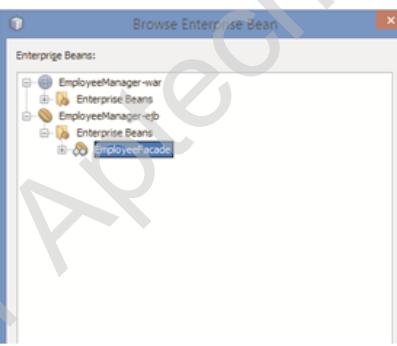
JAX-WS/Session 5

36

## Create a Web Service Implementation Class 8-10

Specify a name for the Web service, a package name for the Web service, select the Create Web Service from Existing Session Bean option, and then, click Browse. In the Browse Enterprise Bean dialog box, select the bean to be used, as shown in the following figure:

Select the Implement Web Service as Stateless Session Bean check box, as shown in the following figure:



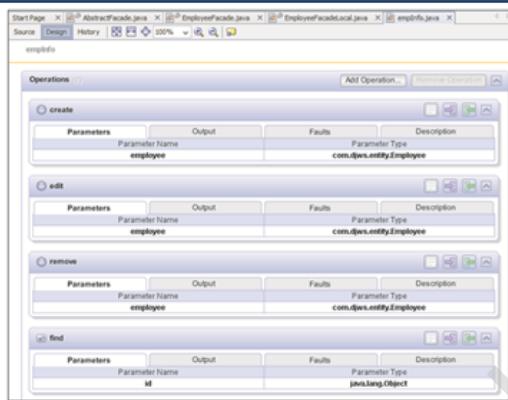
© Aptech Ltd.

JAX-WS/Session 5

37

## Create a Web Service Implementation Class 9-10

- ♦ Alternatively, you can create a session bean from scratch. By default, the Web service is implemented as a stateful session bean. To create a stateless session bean, you need specify explicitly by selecting the check box provided. The .java file for the Web service is created.
- ♦ By default, this Web service contains code for several operations, such as create, edit, remove, find, findAll, findRange, and count, which can be viewed by clicking the Design tab as shown in the following figure:



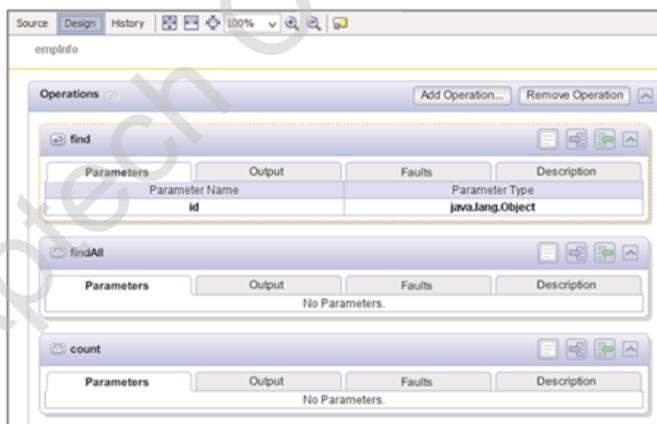
© Aptech Ltd.

JAX-WS/Session 5

38

## Create a Web Service Implementation Class 10-10

To remove some of these default operations, in the Design tab, click the operation name bar to select the operation and click Remove Operation button. Retain the count, find, and findAll operations, as shown in the following figure:



© Aptech Ltd.

JAX-WS/Session 5

39

Using slides 30 to 39, explain what the Web service implementation class contains, namely, the method definition or implementation code for the method, which is accessed by the client. Demonstrate the various steps to create the Web service implementation class using the figures given on slides 30 to 39.

## Build, Package, and Deploy the Web Service

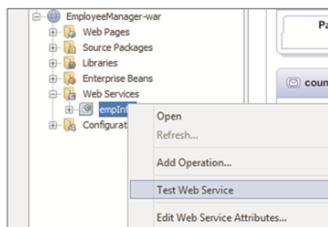
Slides 40 to 42

### Build, Package, and Deploy the Web Service 1-3

- The generation and deployment of the Web service is done in NetBeans IDE. The steps to do are as follows:

1. • Select the project in NetBeans IDE.
2. • Right-click to view the context menu.
3. • Select Deploy from the context menu.

The command Deploy builds and packages the Web application into the WAR file. Then, it deploys the file on the application server, that is, GlassFish Server. To test the Web Service, right-click the Web Service and then, in the context menu, click Test Web Service, as shown in the following figure:



© Aptech Ltd.

JAX-WS/Session 5

40

### Build, Package, and Deploy the Web Service 2-3

The tester Web page appears as shown in the following figure:

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

**Methods :**

```

public abstract int com.djws.service.EmpInfo.count()
count 0

public abstract com.djws.service.Employee com.djws.service.EmpInfo.find(java.lang.Object)
find EMP111

public abstract java.util.List com.djws.service.EmpInfo.findAll()
findAll 0

```

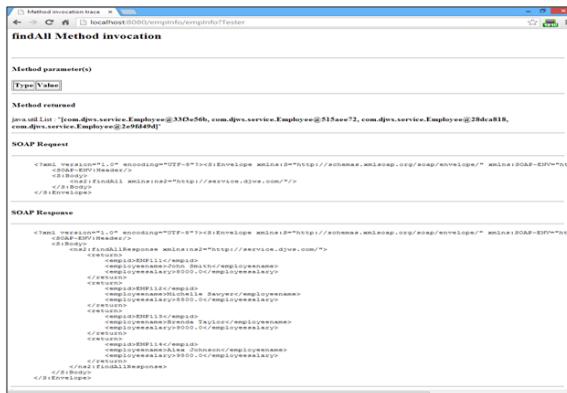
© Aptech Ltd.

JAX-WS/Session 5

41

### Build, Package, and Deploy the Web Service 3-3

- ◆ The Web Service Tester displays the operations that were retained—count, find, and findAll.
  - ◆ The count operation will display the number of records in the Employee table. The find operation will display the details of the employee whose employee ID has been specified in the box.
  - ◆ The findAll operation will display the details of all employees stored in the Employee table.
- Following figure shows the output of the findAll operation:



The screenshot shows a Windows application window titled "Method invocation trace" with the URL "localhost:8080/wmpinfo/employeeInfoTester". The window is titled "findAll Method invocation". It contains tabs for "Method parameters(s)" and "Type/Value". Under "Method returned", it lists a Java ArrayList containing three Employee objects with IDs 515 and 206. Below this is a "SOAP Request" section with XML code and a "SOAP Response" section with multiple XML responses, each containing employee details like name, salary, and department.

© Aptech Ltd.

JAX-WS/Session 5

42

Using slides 40 to 42, explain how to build, package, and deploy the Web service. Demonstrate the steps to deploy the Web service, which are:

- Select the project in NetBeans IDE.
- Right-click to view the context menu.
- Select Deploy from the context menu.

Also, explain about tester Web page using the figures on slides 41 and 42.

#### In-Class Question:



List the steps to deploy the Web service.

#### Answer:

1. Select the project in NetBeans IDE.
2. Right-click to view the context menu.
3. Select Deploy from the context menu.

## Create Web Client

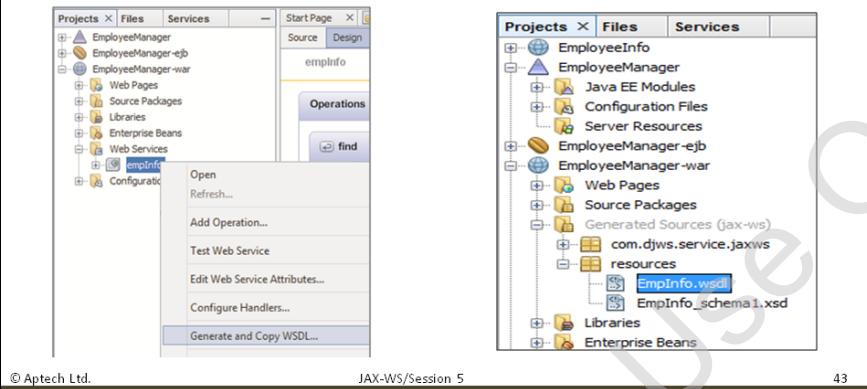
Slides 43 to 48

### Create Web Client 1-6

- A Web client can be created using JSP, Servlet, and so on. The WSDL file is generated based on the Web service.

The WSDL can also be copied to the project. To do this, right-click the Web service and then, click Generate and Copy WSDL as shown in the following figure:

A new folder is created in the project named Generated Sources (jax-ws). To view the WSDL file, expand Generated Sources (jax-ws) and then expand resources, as shown in the following figure:



© Aptech Ltd.

JAX-WS/Session 5

43

### Create Web Client 2-6

The WSDL file is used to generate the Web service client. Perform the following steps to generate the Web service client:

1. Create a new Web Application project in NetBeans IDE.
2. Ensure that the Glassfish Server is selected and the Context Path is set.
3. Right-click the project and then click New → Others and then in the New File Wizard, select Web Services in the Categories list and Web Service Client in the File Types list.
4. To specify the project where the WSDL file is located, click Browse next to the Project box.
5. Browse to the Web service for which the Web service client is being created, as shown in the following figure:



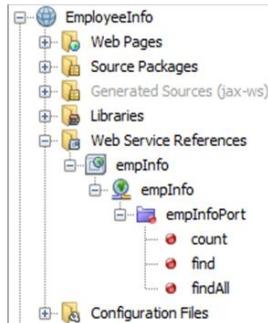
© Aptech Ltd.

JAX-WS/Session 5

44

### Create Web Client 3-6

A Web Service References folder is created in the project. This folder contains the method exposed by the Web service, as shown in the following figure:



### Create Web Client 4-6

6. • Add a new JSP file by right-clicking the project and then clicking New → JSP.

7. • Name the JSP file as index.

Drag the find() method from the Web Service Reference folder to the index.jsp file and place it after the h1 heading. The updated index.jsp file appears, as shown in the following figure:

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Hello World</h1>      <!-- start web service invocation --><hr/>
    <%
    try {
      com.djws.service.EmpInfo_Service service = new com.djws.service.EmpInfo_Service();
      com.djws.service.EmpInfo port = service.getEmpInfoPort();
      // TODO: implement this call to invoke arguments here
      java.lang.Object id = null;
      // TODO: implement result here
      com.djws.service.Employee result = port.find(id);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    <!-- end web service invocation --><hr/>
  </body>
</html>

```

## Create Web Client 5-6

Update the title of the JSP page and h1 heading, add an input text box for entering the employee ID and a button, add the code to use the Web service to retrieve and display the relevant employee details and add the exception handling code in the catch block as shown in the following code snippet:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
        <link href="style.css" rel="stylesheet" type="text/css"/>
    </head>
    <body>
        <h1>Employee Information</h1>
        <%-- start Web service invocation --%><hr/>
        <form action="index.jsp">
            Enter Employee ID:
            <input type="text" id="txtbox" name="empid" size="6"
maxlength="6">
            <br><br>
            <input type="submit" value="Get Details">
            <br><br>
        <%>
```

## Create Web Client 6-6

```
try {
    com.djws.service.EmpInfo_Service service = new
com.djws.service.EmpInfo_Service();
    com.djws.service.EmpInfo port = service.getEmpInfoPort();

// TODO initialize WS operation arguments here
    if(request.getParameter("empid")!=null){
java.lang.String id =request.getParameter("empid");

        // TODO process result here
        com.djws.service.Employee emp = port.find(id);
        out.println("The details of the employee are as follows:");
        out.println("<br>");
        out.println("<b>ID: </b>" + emp.getEmpid());
        out.println("<br>");
        out.println("<b>Name: </b>" + emp.getEmployeeName());
        out.println("<br>");
        out.println("<b>Salary: </b>" + emp.getEmployeeSalary());
        }
    }
catch (Exception ex) {
    out.println("Exception : " + ex);
}
%>
<%-- end Web service invocation --%><hr/>
</form>    </body>    </html>
```

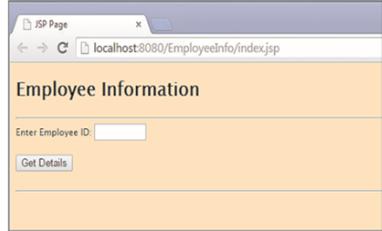
Using slides 44 to 48, explain that WSDL file is generated based on the Web service. Explain the code to generate WSDL by the Web eservice. Demonstrate the steps to generate the WSDL Web Client. Using the figures on slides 43 to 46, explain the creation of index.jsp file. Explain the addition of the exception handling code and the style.css file specified in the Code Snippet on slides 47 and 48.

## Build Package and Deploy the JSP Client

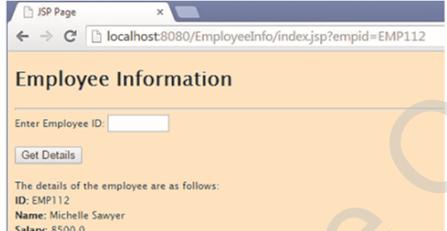
### Slide 49

**Build Package Deploy the JSP Client**

To build, package, and deploy the client application, right-click the project name and select Run from the context menu. The output of the client application is as shown in the following figure:



When an employee ID is specified and the Get Details button is clicked, the name and salary of the employee whose ID has been specified is displayed on the Web page as shown in the following figure:



© Aptech Ltd.      JAX-WS/Session 5      49

Using slide 49, demonstrate the steps to build, package, and deploy the JSP client. Use the figures to show the output of the client application and the retrieved employee details that are displayed on the Web page.

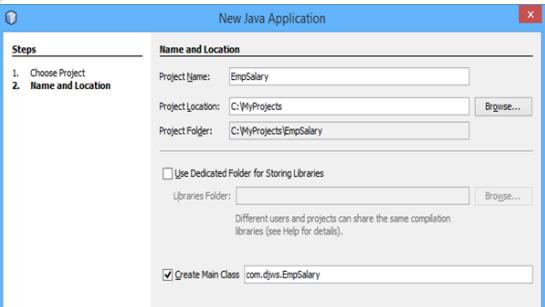
## Create an Application Client

### Slides 50 to 57

**Create an Application Client 1-8**

You can also create a Java application client for the Web service. To do this, perform the following steps:

1. Create a new Java Application project in NetBeans IDE by clicking File → New Project and then, in the New Project wizard, select Java in the Categories list and Java Application in the Projects list.
2. Specify a name and location for the project and ensure that the Create Main Class check box is selected as shown in the following figure:

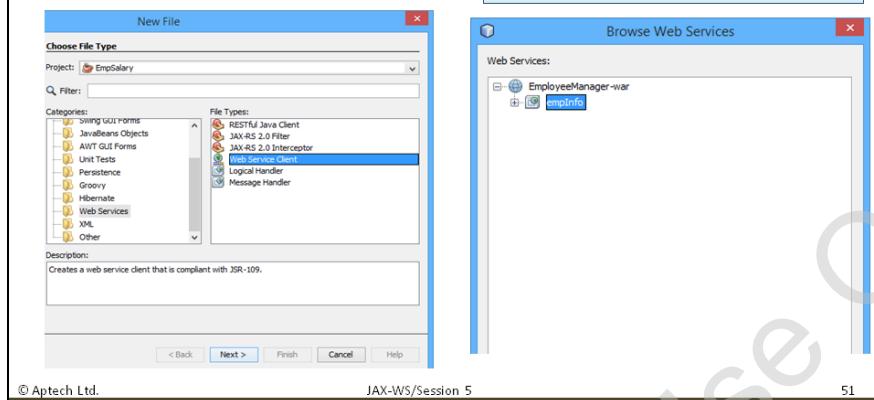


© Aptech Ltd.      JAX-WS/Session 5      50

## Create an Application Client 2-8

3. To create a Web service client, right-click the Java Application project and then click New → Others. Then, in the New File wizard, select Web Services in the Categories list and Web Service Client in the File Types list, as shown in the following figure:

4. To specify the project that contains the Web service, in the New Web Service Client wizard, click Browse, next to the Project box, and then in the Browse Web Services dialog box, select the Web service for which the client is being created, as shown in the following figure:



© Aptech Ltd.

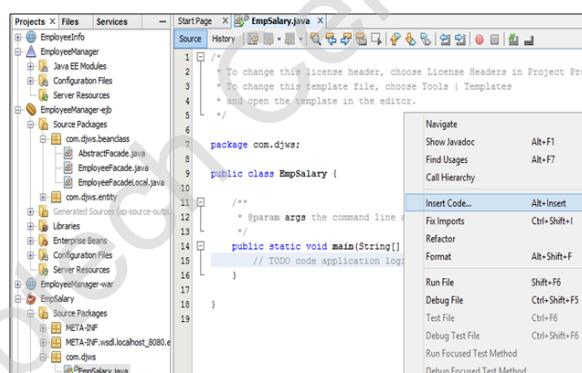
JAX-WS/Session 5

51

## Create an Application Client 3-8

5. The Web Service code will be generated. Build the Web service client project to compile all the generated resources.

6. Open the EmpSalary.java file, right-click inside the main() method, and then click Insert Code as shown in the following figure:



© Aptech Ltd.

JAX-WS/Session 5

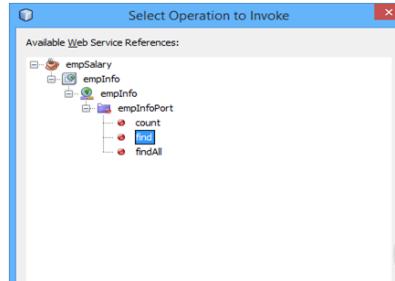
52

### Create an Application Client 4-8

7. In the context menu that appears, click Call Web Service Operation, as shown in the following figure:



8. In the Select Operation to Invoke dialog box, select the find() method, as shown in the following figure:



### Create an Application Client 5-8

The IDE adds the code required to invoke the find() method, as shown in the following figure:

```

11
12  public class EmpSalary {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20
21      private static Employee find(java.lang.Object id) {
22          com.djws.service.EmpInfo_Service service = new com.djws.service.EmpInfo_Ser
23          com.djws.service.EmpInfo port = service.getEmpInfoPort();
24          return port.find(id);
25      }
26
27  }

```

## Create an Application Client 6-8

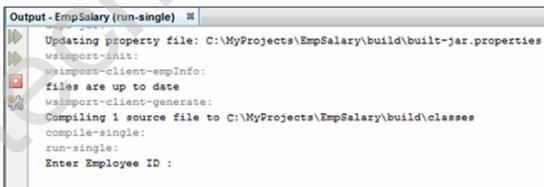
9. To accept user input and display the output, add the code given as shown in code snippet, in the main() method of the EmpSalary class.

```
...
import java.util.Scanner;
...
{
public static void main(String[] args)
{
    String empId = null;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter Employee Id :");
    empId = in.nextLine();
    Employee emp = new Employee();
    emp = find(empId);
    System.out.println("Employee ID: " + emp.getEmpid());
    System.out.println("Employee Name: " +
    emp.getEmployeeName());
    System.out.println("Employee Salary: " +
    emp.getEmployeeSalary());
}
}
```

## Create an Application Client 7-8

```
private static Employee find(java.lang.String id)
{
    com.djws.EmpInfo_Service service = new
        com.djws.EmpInfo_Service();
    com.djws.EmpInfo port = service.getEmpInfoPort();
    return port.find(id);
}
```

10. Build and run the EmpSalary project. The user will be prompted to enter the employee ID as shown in the following figure:



### Create an Application Client 8-8

11. Specify employee ID EMP114 and then, press ENTER. The employee name and employee salary is displayed as shown in the following figure:

```

Output - EmpSalary (run-single) ✘
Compiling 1 source file to C:\MyProjects\EmpSalary\build\classes
compile-single:
run-single:
Enter Employee ID :
EMP114
Employee ID: EMP114
Employee Name: Alex Johnson
Employee Salary: 9500.0
BUILD SUCCESSFUL (total time: 1 minute 37 seconds)
  
```

Using slides 50 to 57, demonstrate the steps to create a Java application client for the Web service. Use the figures to point out what option(s) need to be selected in the menu or dialog boxes to get the final output of the Web services.

### Summarize Session

#### Slide 58

### Summary

- ◆ Web services can be developed using annotations and Plain Old Java Objects (POJOs). They are available on JEE platform, which provides the right environment for easy development and deployment of Web services.
- ◆ There are two types of Web services. They are 'Big' Web services and RESTful Web services.
- ◆ JAX-WS supports annotations that make development of Web service applications simple and easy.
- ◆ JAX-WS data binding and parsing are based on the JAXB and StAX.
- ◆ JAX-WS architecture is based on the generation of dynamic proxy class instance, which is responsible to send/receive SOAP request/response on the client/server.
- ◆ The two types of service endpoint implementations supported by JAX-WS are the standard JavaBeans service endpoint interface and a new Provider interface.
- ◆ The different types of clients in Web service client programming model supported by JAX-WS are Dynamic proxy client and Dispatch client. They support synchronous and asynchronous invocations on the client-side.

Using slide 58, summarize the session. Make them revise the following points:

- Web services can be developed using annotations and Plain Old Java Objects (POJOs). They are available on JEE platform, which provides the right environment for easy development and deployment of Web services.

- There are two types of Web services. They are ‘Big’ Web services and RESTful Web services.
- JAX-WS supports annotations that make development of Web service applications simple and easy.
- JAX-WS data binding and parsing are based on the JAXB and StAX.
- JAX-WS architecture is based on the generation of dynamic proxy class instance, which is responsible to send/receive SOAP request/response on the client/server.
- The two types of service endpoint implementations supported by JAX-WS are the standard JavaBeans service endpoint interface and a new Provider interface.
- The different types of clients in Web service client programming model supported by JAX-WS are Dynamic proxy client and Dispatch client. They support synchronous and asynchronous invocations on the client-side.

### **5.3 Post Class Activities for Faculty**

You should familiarize yourself with the topics of the next session. You should also explore the next session which describes the RESTful Web services.

**Tips:** You can also check the **Articles/Blogs/Expert Videos** uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

## Session 6: RESTful Web Services

### **6.1 Pre-Class Activities**

You should familiarize yourself with RESTful Web services. You should be able to explain Java API for RESTful Web Services Application Program Interface (JAX-RS API). You should also know how to build a RESTful Web service with JAX-RS API.

Familiarize yourself with the topics of the current session in-depth.

#### **6.1.1 Objectives**

After the session, learners will be able to:

- Explain RESTful Web Services
- Describe JAX-RS API
- Explain how to build a RESTful Web Service with JAX-RS API

#### **6.1.2 Teaching Skills**

To teach this session, you should be well versed with RESTful Web Services. You should know about REpresentational State Transfer (REST) architecture guidelines and the constraints applied to these architectures. You should know the requirements for developing RESTful Web services. You should be able to describe the JAX-RS API, its annotations, and how to build a RESTful Web Service with JAX-RS API. You should know well how to use RESTful implementation class. You should have sound knowledge on how to build, package and deploy the RESTful Web service. You should also know all the steps to create a Web client for the RESTful Web service. For teaching in the class, you are expected to use slides and LCD projectors.

#### **Tips:**

It is recommended that you test the understanding of the students by asking questions in between the class.

#### **6.1.3 In-Class Activities:**

Follow the order given here during In-Class activities.

#### **Review of Previous Session**

You should begin with a brief recap or review of previous session which defined JAX-WS and explained the need, features, and standards of JAX-WS. You should summarize about JAX-WS architecture and JAX-WS annotations and JAX-WS programming model discussed in the previous session.

#### **Overview of the Session**

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell the students that this session explains RESTful Web services. Tell that the session also describes JAX-RS API and explains how to build a RESTful Web Service with JAX-RS API.

## 6.2 In-Class Explanations

### Introduction to RESTful Web Services

#### Slide 3

**Introduction to RESTful Web Services**

- Web application development is based on HTTP protocol.  
HTTP is a network protocol used for data communication on the World Wide Web.
- The other technologies that are used in Web application development are HTML, JavaScript, XML, and Asynchronous JavaScript and XML (AJAX).
- Web services are Web applications based on XML standards and transport protocol, HTTP.
- The development of Web services can be classified based on the technologies and the architectures available on the Web.
- The two approaches used for designing the Web services are Standards-based approach and REST-based approach.

© Aptech Ltd. RESTful Web Services/Session 6 3

Using slide 3, introduce the learners to RESTful Web Services. Mention that RESTful Web Services are based on Hyper Text Transfer Protocol (HTTP) protocol, which implements a request-response pattern in the client-server architecture. List the other technologies such as HTML, JavaScript, XML, and Asynchronous JavaScript and XML (AJAX) that are used in Web application development.

Explain that Web services are classified on the basis of technologies. Cite few examples such as Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL), Extensible Markup Language (XML), security, Uniform Resource Locator (URL), and JAX-RPC. Point out that Web services are also based on the Web architectures such as client-server architecture.

Mention that in these architectures, a server transfers the information requested by the client, which is presented to the client. Discuss about the Standards-based approach and REST-based approach, the two approaches used for designing the Web services. Mention that the Uniform Resource Identifiers (URIs) that are accessible over HTTP protocol identify these resources.

**REST**  
**Slide 4**

## REST

REST is a set of guidelines or principles applied to the architectures available in a network system.

REST is an architecture-style on which systems are designed consisting of protocols, data components, hyperlinks, and clients.

The different constraints applied to the architectures based on the REST style are as follows:

- Client-Server
- Stateless
- Cache
- Layered components
- Uniform interface
- Code on demand

© Aptech Ltd. RESTful Web Services/Session 6 4

Using slide 4, explain the concept of REST. Mention that the World Wide Web, a system of interlinked hypertext documents, is an example of the REST architecture-style. Explain the different constraints applied to the architectures based on the REST style.

**In-Class Question:**

What is REST?

**Answer:**

REST is a set of guidelines or principles applied to the architectures available in a network system.

## **RESTful Web Services**

### **Slide 5**

### **RESTful Web Services**

RESTful Web services are Web services based on REST architecture and are accessed using HTTP protocol on the Web.

- ◆ RESTful Web services are similar to SOAP-based Web services.

The requirements for developing RESTful Web services based on the REST architecture style are as follows:

The requirements for developing RESTful Web services based on the REST architecture style are as follows:

Resources	Representation of a Resource	URI	HTTP methods
-----------	------------------------------	-----	--------------

© Aptech Ltd. RESTful Web Services/Session 6 5

Using slide 5, explain about RESTful Web services. Tell that RESTful Web services are platform and language independent. Explain the similarity in characteristics of RESTful Web services and SOAP-based Web services. Then, explain the difference between RESTful Web services and SOAP-based Web services. Mention that RESTful Web services use Uniform Resource Identifier (URI) to send data directly to the service over the underlying protocol, HTTP whereas SOAP-based Web services use SOAP protocol over HTTP. Explain the requirements for developing RESTful Web services.

#### **In-Class Question:**



List the requirements for developing RESTful Web services based on the REST architecture style.

#### **Answer:**

The requirements for developing RESTful Web services based on the REST architecture style are as follows:

- Resources
- Representation of a Resource
- URI
- HTTP methods

**Resources****Slide 6**

## Resources

Resource can be defined as any kind of information exposed over the Web. Information can be a document, an image, or an entity such as a book, or a flight detail from a database.

Examples of some RESTful Web services resources are as follows:

- ◆ Courses offered by a university
- ◆ A CNN news story board
- ◆ A search result for any article in the search engine
- ◆ Flight information stored in the flight database

© Aptech Ltd. RESTful Web Services/Session 6 6

Using slide 6, define Resource and provide a few examples of RESTful Web services.

**In-Class Question:**

What is a Resource?

**Answer:**

Resource can be defined as any kind of information exposed over the Web.

**Representation of a Resource****Slide 7**

## Representation of a Resource

Representation can be defined as the format in which the resource is sent and received during client and server interaction on the Web. The format of the resource representation, returned as a response to the client, represents the temporary state of the resource. Following figure shows the handling of request and response of a RESTful Web service:

© Aptech Ltd. RESTful Web Services/Session 6 7

Using slide 7, define Representation. Tell about the different data formats in which a resource can be represented and they can be accessed by different clients. Explain the figure, which shows a client application requesting for a resource on a URI. Explain that client application receives an XML document containing the details of the product in an XML format. Point out that based on the client requirement, the format in which the response is returned varies.

### URI

#### Slide 8

In the RESTful Web service, a resource is identified by a URI.  
A URI is a combination of name and address of the resource, which helps the client and server to exchange data formats during interaction.

© Aptech Ltd. RESTful Web Services/Session 6 8

Using slide 8, explain URI and mention that URI helps the client and server to exchange data formats during interaction.

#### In-Class Question:



What is URI?

#### Answer:

A URI is a combination of name and address of the resource, which helps the client and server to exchange data formats during interaction.

## HTTP

### Slide 9

## HTTP

HTTP is a stateless transport protocol used to exchange data over the Web. Web browsers support HTTP methods to send and receive data from the server. Most commonly used HTTP methods for transferring data to the server in a request object are GET and POST. RESTful Web services support these HTTP methods for sending and receiving the data represented as a resource on the Web.

- ◆ HTTP methods are also known as **verbs**.
- ◆ The different types of actions are create, read, update, and delete, also known as **CRUD actions**.
- ◆ Following table describes the Mapping of HTTP methods with their Matching CRUD Actions:

HTTP Method	Action
POST	Create a new resource from the incoming data in the request
GET	Retrieve a resource
PUT	Update a resource from the incoming data in the request
DELETE	Delete a resource

© Aptech Ltd. RESTful Web Services/Session 6 9

Using slide 9, explain HTTP and its uses. Tell that GET and POST are the most commonly used HTTP methods for transferring data to the server. Explain that the other HTTP methods are PUT, HEAD, and DELETE. Explain why HTTP methods are also known as 'verbs'. Explain the mapping of HTTP methods with their corresponding CRUD (Create, Read, Update, Delete) Actions.

## JAX-RS API

### Slide 10

## JAX-RS API

JAX-RS specification is an official API on Java EE platform for creating Web services based on REST architecture. JAX-RS is a Java programming API, designed to simplify the development process of RESTful Web service using annotations.

JAX-RS API allows creating two types of resources, which are as follows:

- ◆ Root resource class
- ◆ Sub resources

© Aptech Ltd. RESTful Web Services/Session 6 10

Using slide 10, explain JAX-RS API. Tell that it simplifies the development process of RESTful Web service by using annotations. Discuss the two types of resources that are created by JAX-RS API. Explain the role of @Path annotation in the resources of JAX-RS API.

## JAX-RS API Annotations

Slides 11 to 17

### JAX-RS API Annotations 1-7

JAX-RS API annotations simplify the development process of RESTful Web services. These annotations help to identify the resources and the actions to be performed on the identified resources.

Annotations available in the `java.ws.rs` package are categorized as follows:

- `@Path` annotation
- HTTP method annotations
- Annotations for injecting data from request URI
- Data representation type annotations

Using slide 11, explain JAX-RS API annotations and the uses of JAX-RS API annotations, which can be listed as follows:

- Simplifying the development process of RESTful Web services
- Identifying the resources and the actions to be performed on the identified resources

Discuss the action of Annotation Processing Tool (APT) at runtime.

List the annotations available in the `javax.ws.rs` package, which are as follows:

- `@Path` annotation
- HTTP Method Annotations
- Injecting Data from Request URI Annotations
- Data Representation Type Annotations

## JAX-RS API Annotations 2-7

- ◆ **@Path Annotation**

The @Path annotation specifies the URI of the Web service class, hosted as a resource on the server. The @Path annotation has a single attribute which accepts a String value.

- ◆ Following code snippet demonstrates the use of @Path annotation with the URI template applied at class level to access the Web service class deployed on the Web server:

```
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
//Sets the path to base URL + /Welcome
@Path("/welcome")
public class Welcome{
// This method is called if TEXT_PLAIN is requested
@GET
@Produces(MediaType.TEXT_PLAIN)
public String sayPlainTextWelcome(){
return "Welcome to the Learning Portal";
}
}
```

Using slide 12, explain @Path annotation. Discuss about the levels at which @Path annotation can be used in the Web service class.

Use the Code Snippet to explain the use of @Path annotation with the URI template applied at class level, which accesses the Web service class deployed on the Web server.

## JAX-RS API Annotations 3-7

- ◆ The URI template containing the embedded variables specified with the @Path annotation can be written as follows:

```
@Path("resourcePath/{param1}/.../{paramN}")
```

- ◆ Following code snippet demonstrates the URI template with the embedded variables to access the sub resource from the Web Service class on the Web server:

```
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
//Sets the path to base URL + /Welcome
@Path("/welcome/{username}")
public class Welcome{
// This method is called if TEXT_PLAIN is requested
@GET
@Produces(MediaType.TEXT_PLAIN)
public String sayPlainTextWelcome(@PathParam("username") String userName){
return "Welcome to the Learning Portal " + userName;
}
}
```

Using slide 13, mention that URI template can be embedded with the variables for accessing the resources. Using the Code Snippet explain the usage of the URI template with the embedded variables to access the sub resource from the Web Service class on the Web server.

## JAX-RS API Annotations 4-7

```

@GET
@Path("students/{id}")
@Produces({"application/xml", "application/json"})
public Student find(@PathParam("id") String id) {
    return super.find(id);
}
}

```

- ◆ **HTTP Method Annotations:** The annotations in the javax.ws.rs packages corresponding to the equivalent HTTP methods such as GET, POST, PUT, and DELETE are as follows:

@GET annotation maps to the HTTP GET method

@POST annotation maps to the HTTP POST method

@PUT annotation maps to the HTTP PUT method

@DELETE annotation maps to the HTTP DELETE method

Using slide 14, explain HTTP method annotations. List the annotations in the javax.ws.rs packages corresponding to the equivalent HTTP methods such as GET, POST, PUT, and DELETE.

## JAX-RS API Annotations 5-7

- ◆ **Injecting Data from Request URI Annotations** – extract the data from the request URI into the resource class.
- ◆ Following code snippet demonstrates the use of the @PathParam annotation on the parameter of a method of the Web service class present on the Web server:

```

@PUT
@Path("{id}")
@Consumes({"application/xml", "application/json"})
public void edit(@PathParam("id") String id, Student entity) {
    super.edit(entity);
}

```

- ◆ Following code snippet demonstrates the use of @QueryParam annotation with method parameters in the Web service class:

```

...
@Path("/mess")
public String getMessage(@QueryParam("str")String studentName) {
    return "Welcome " + studentName + " ....";
}
...

```

Using slide 15, explain Injecting Data from Request URI Annotations and its use. Describe @PathParam annotation. Using the Code Snippet, explain the use of the @PathParam annotation on the parameter of a method of the Web service class present on the Web server.

Then, explain about @QueryParam annotation and the Code Snippet that demonstrates the use of @QueryParam annotation with method parameters in the Web service class. Use the Code Snippets to explain how they are used.

## JAX-RS API Annotations 6-7

- ◆ **Data Representation Type Annotations** –specifies the data type to be produced or consumed by the Web Service class.
- ◆ Following code snippet demonstrates the use of @Produces annotation applied to a method of the Web service class:

```
@GET
@Override
@Produces({"application/xml", "application/json"})
public List<Student> findAll() {
    return super.findAll();
}
```

- ◆ Following code snippet demonstrates the use of @Consumes annotation applied at the method level of the Web service class:

```
@POST
@Override
@Consumes({"application/xml", "application/json"})
public void create(Student entity){
    super.create(entity);
}
```

Using slide 16, explain Data Representation Type annotations. Describe @Produces annotation, and explain the Code Snippet that shows the use of @Produces annotation applied to a method of the Web service class. Describe @Consumes annotation, and explain the Code Snippet that shows the use of @Consumes annotation applied at the method level of the Web service class.

## JAX-RS API Annotations 7-7

- ◆ Following code snippet demonstrates the use of @Provider annotation applied at the method level of the Web service class:

```
//Use of @Provider annotation in MessageBodyReader
@Consumes("application/x-www-form-urlencoded")
@Provider
public class FormReader implements MessageBodyReader<NameValuePair> {
.....
.....
.....
}

//Use of @Provider annotation in MessageBodyWriter
@Produces("text/html")
@Provider
public class FormWriter implements
    MessageBodyWriter<Student<String studentName, String studentGrade>>{
.....
.....
}
```

Using slide 17, explain @Provider annotation and state the different types of data representations consumed and produced by the Web service methods. Using Code Snippet, describe the use of @Provider annotation applied at the method level of the Web service class. Mention that the @Provider annotation helps to map MessageBodyReader and MessageBodyWriter to the HTTP request and HTTP response, respectively, using JAX-RS runtime.

**In-Class Question:**

What is @QueryParam annotation used for?

**Answer:**

This annotation is used to extract the query parameter from the request URI sent by the client as a name-value pair.

**Building RESTful Web Service with JAX-RS API****Slide 18****Building RESTful Web Service with JAX-RS API**

The steps to design and build a RESTful Web service using JAX-RS API are as follows:

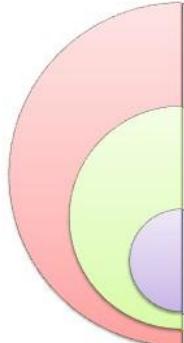
- 1 • Code the implementation class which will act as a root resource class
- 2 • Define the methods within the main implementation class that act as sub resources
- 3 • Annotate the class with JAX-RS annotations
- 4 • Build, package, and deploy the files on the application server
- 5 • Access the resource in the client browser application

Using slide 18, explain the steps to build a RESTful Web Service with JAX-RS API.

**RESTful Web Implementation Class****Slides 19 to 23****RESTful Web Implementation Class 1-5**

The RESTful implementation class is a resource which can be accessed by the client through a URI.

Criteria to be followed by a class to behave as a root resource class:



The RESTful implementation class must be annotated with `@Path`, which specifies it as a base URI of the resource implemented by the service.

The implementation class must have a public constructor so that it can be invoked by the JAX-RS runtime.

The HTTP methods accept different types of request such as GET, POST, PUT, or DELETE for handling various types of requests.

**RESTful Web Implementation Class 2-5**

- Following code snippet demonstrates implementation of the StudentFacadeREST Web service class:

```
package com.syskan.studentdb.entities.service;
import com.syskan.studentdb.entities.Student;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
@Stateless
@Path("com.syskan.studentdb.entities.student")
public class StudentFacadeREST extends AbstractFacade<Student>{
    @PersistenceContext(unitName = "com.syskan_StudentDB_war_1.0-SNAPSHOTPU")
    private EntityManager em;
```

## RESTful Web Implementation Class 3-5

```

public StudentFacadeREST() {
    super(Student.class);
}
@POST
@Override
@Consumes({"application/xml", "application/json"})
public void create(Student entity) {
    super.create(entity);
}
@PUT
@Path("{id}")
@Consumes({"application/xml", "application/json"})
public void edit(@PathParam("id") Integer id, Student entity) {
    super.edit(entity);
}
@DELETE
@Path("{id}")
public void remove(@PathParam("id") Integer id) {
    super.remove(super.find(id));
}

```

## RESTful Web Implementation Class 4-5

```

@GET
@Path("{id}")
@Produces({"application/xml", "application/json"})
public Student find(@PathParam("id") Integer id){
    return super.find(id);
}

@GET
@Override
@Produces({"application/xml", "application/json"})
public List<Student> findAll() {
    return super.findAll();
}
@GET
@Path("{from}/{to}")
@Produces({"application/xml", "application/json"})
public List<Student> findRange(@PathParam("from") Integer from,
@PathParam("to") Integer to) {
    return super.findRange(new int[]{from, to});
}

```

## RESTful Web Implementation Class 5-5

```
@GET  
@Path("count")  
@Produces("text/plain")  
public String countREST() {  
    return String.valueOf(super.count());  
}  
@Override  
protected EntityManager getEntityManager(){  
    return em;  
}
```

Using slides 19 to 23, explain RESTful Web Implementation Class. Using slide 19, define RESTful implementation class and explain the criteria to be followed by a class to behave as a root resource class. Using slides 20 to 23, explain the Code Snippet that demonstrates the implementation StudentsFacadeREST Web service. Mention that that this code uses the Web service to retrieve a list of the student names based on count, by id, or by using the functions findAll(), find(), and findRange() with proper parameters.

### In-Class Question:



What is RESTful Web implementation class?

### Answer:

The RESTful implementation class is a resource which can be accessed by the client through a URI.

**Build, Package, and Deploy the RESTful Web Service****Slides 24 to 31**

### Build, Package, and Deploy the RESTful Web Service 1-8

To create a RESTful Web service based on a database table, perform the following steps:

1. In the NetBeans IDE, create a new Web application project. To do this, click File → New and then in the New Project wizard, select Java Web from the Categories list and Web Application from the Projects list.
2. Specify a name and location for the project.
3. Ensure that GlassFish Server is selected and the Context Path is set.
4. To create a RESTful Web service in the Web application project, right-click the project node and then click New → Other.
5. In the New File wizard, select Web Services from the Categories list and RESTful Web Services from Database from the File Types list.

♦ Following figure shows a STUDENTS table:

© Aptech Ltd.

RESTful Web Services/Session 6

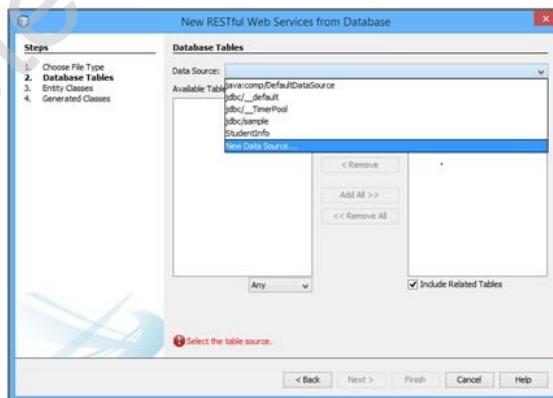
24

Using slides 24 to 31, demonstrate the steps to build, package, and deploy RESTful Web Service. Using slide 24, demonstrate the steps to create a RESTful Web service based on STUDENTS database table. Show the screenshot of STUDENTS database and mention that the STUDENTS table will be used to explain the steps.

**Build, Package, and Deploy the RESTful Web Service 2-8**

6. In the New RESTful Web Service from Database dialog box, from the Data Source drop-down list, select New Data Source.

- ♦ Following figure shows how to create a new data source:



© Aptech Ltd.

RESTful Web Services/Session 6

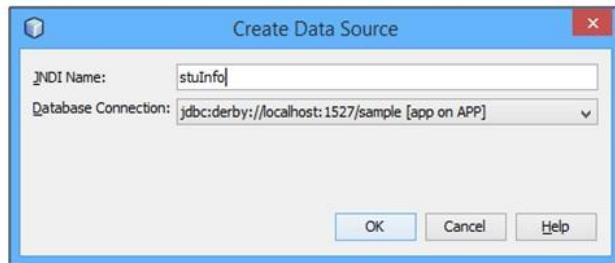
25

Using slide 25, demonstrate the creation of new data source.

## Build, Package, and Deploy the RESTful Web Service 3-8

7. In the Create Data Source dialog box, in the Database Connection drop-down list, select `jdbc:derby://localhost:1527/sample` and specify a JNDI name for the data source.

- Following figure shows how to name the new data source:



© Aptech Ltd.

RESTful Web Services/Session 6

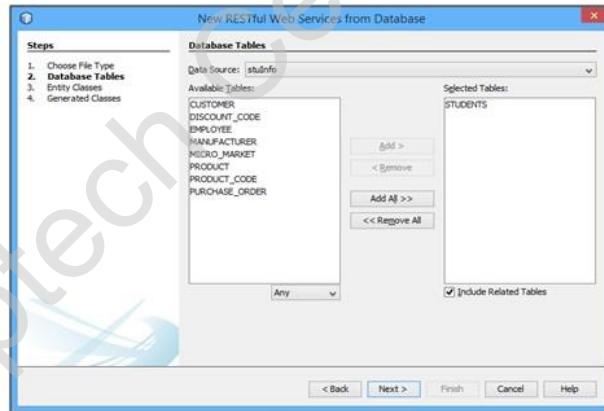
26

Using slide 26, demonstrate the naming the new data source.

## Build, Package, and Deploy the RESTful Web Service 4-8

8. Select the Students database and click the Add button. The Students database is added to the Selected Tables list.

- Following figure shows how to select the database table:



© Aptech Ltd.

RESTful Web Services/Session 6

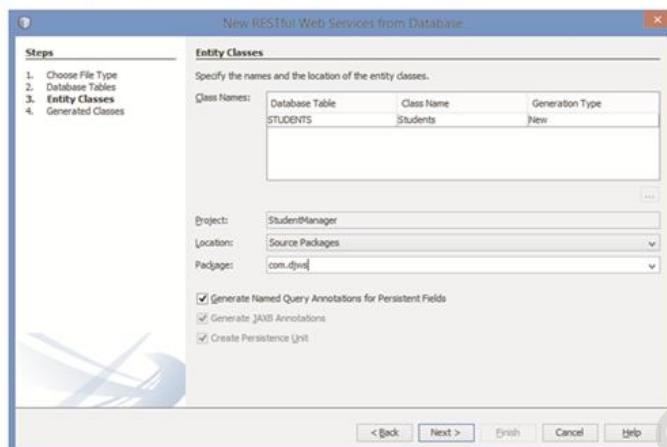
27

Using slide 27, demonstrate adding the STUDENTS table to the selection of the database table.

## Build, Package, and Deploy the RESTful Web Service 5-8

9. Specify a package name for the Web service.

- Following figure shows how to specify the package name:



© Aptech Ltd.

RESTful Web Services/Session 6

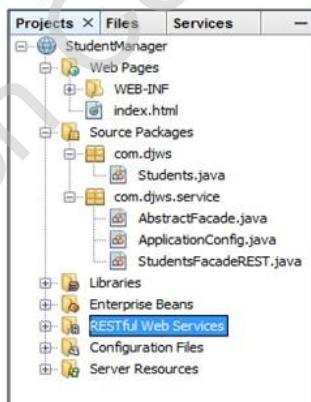
28

Using slide 28, demonstrate how to specify the package name.

## Build, Package, and Deploy the RESTful Web Service 6-8

10. On the next page of the wizard, accept the defaults and click Finish. The IDE generates the RESTful Web service. The generated entity classes are listed in the com.djws package and the services are listed in the com.djws.service package.

- Following figure shows entity classes and services:



© Aptech Ltd.

RESTful Web Services/Session 6

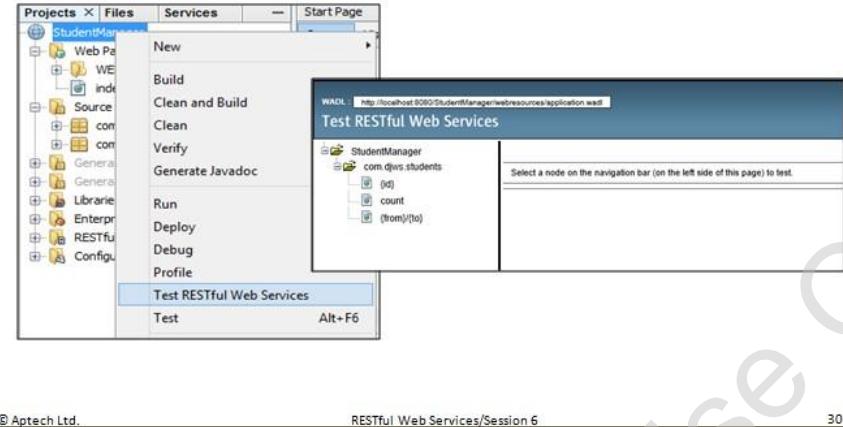
29

Using slide 29, explain the generated entity classes are listed in the com.djws package and the services are listed in the com.djws.service package.

## Build, Package, and Deploy the RESTful Web Service 7-8

11. To test the Web service, right-click StudentManager project, and then click Test RESTful Web Services. The Web service opens in the browser.

- Following figure shows Test RESTful Web Services:



© Aptech Ltd.

RESTful Web Services/Session 6

30

Using slide 30, demonstrate how to test the Web service and the figure that shows how the Web service opens in the browser.

## Build, Package, and Deploy the RESTful Web Service 8-8

12. To find the details of a specific student, in the left pane, click {id}.

13. In the right pane, in the id box, enter the ID of the student and then, click Test. The details of the student whose ID was specified is displayed.

- Following figure shows the output of the RESTful Web Service:



© Aptech Ltd.

RESTful Web Services/Session 6

31

Using slide 31, demonstrate how to input student ID and view the output of the RESTful Web Service.

## Implementing a Web Client

Slides 32 to 45

### Implementing a Web Client 1-14

To create a Web client for the RESTful Web service, perform the following steps:

1. To create a Java client for the RESTful Web service in the Java application project, right-click the project node and then, click New → Other. In the New File wizard, select Web Services from the Categories list and RESTful Java Client from the File Types list.
- ◆ Following figure shows how to choose a RESTful Java Client:

© Aptech Ltd. RESTful Web Services/Session 6 32

Using slides 32 to 45, demonstrate the steps to implement a Web Client. Using slide 32, demonstrate how to create a RESTful Java client for the RESTful Web service.

### Implementing a Web Client 2-14

2. Specify a name for the client.
3. To select the REST resource to be used for the client, under Select REST Resource, ensure that the From Project option is selected and then, click Browse.
4. In the Available REST Resources dialog box, select the appropriate REST Web service as shown in the following figure:

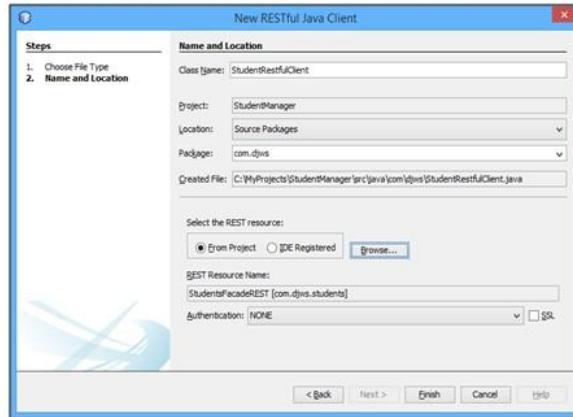
© Aptech Ltd. RESTful Web Services/Session 6 33

Using slide 33, demonstrate how to select the appropriate REST Web service.

## Implementing a Web Client 3-14

5. Select the package for the client.

- Following figure shows how to specify package for Java client:



© Aptech Ltd.

RESTful Web Services/Session 6

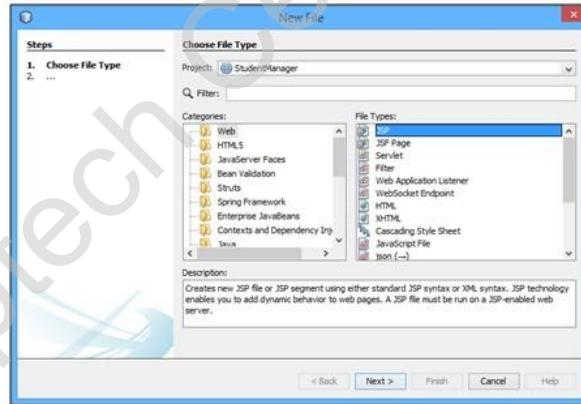
34

Using slide 34, demonstrate how to specify the package for the Java client.

## Implementing a Web Client 4-14

6. To create a JSP file for the client, right-click the StudentManager project and then, click New → Other. Then, in the New File wizard, select Web from the Categories list and JSP from the File Types list.

- Following figure shows how to create a JSP file:



© Aptech Ltd.

RESTful Web Services/Session 6

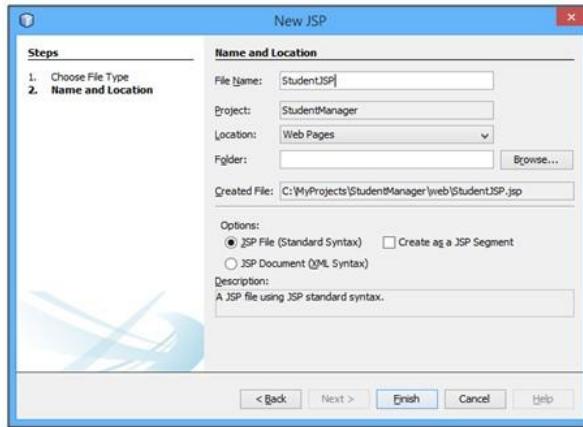
35

Using slide 35, demonstrate how to create a JSP file.

## Implementing a Web Client 5-14

7. Specify a name for the JSP file.

◆ Following figure shows how to specify a name for the JSP file:



© Aptech Ltd.

RESTful Web Services/Session 6

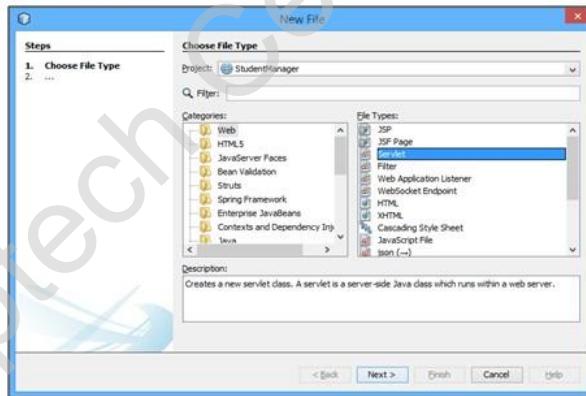
36

Using slide 36, demonstrate how to specify a name for the JSP file.

## Implementing a Web Client 6-14

8. To create a Servlet for the client, right-click the StudentManager project and then, click New → Other. Then, in the New File wizard, select Web from the Categories list and Servlet from the File Types list.

◆ Following figure shows how to create a servlet:



© Aptech Ltd.

RESTful Web Services/Session 6

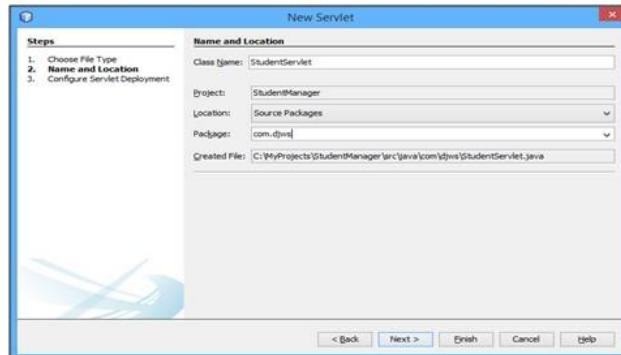
37

Using slide 37, demonstrate how to create a servlet for the client.

## Implementing a Web Client 7-14

9. Specify a name and package for the servlet.

- Following figure shows how to specify a name for servlet:



© Aptech Ltd.

RESTful Web Services/Session 6

38

Using slide 38, demonstrate how to specify a name and package for the servlet in the wizard.

## Implementing a Web Client 8-14

10. To add the servlet information to the web.xml page, select the Add information to deployment descriptor (web.xml) check box.

- Following figure shows how to add servlet information to web.xml:



© Aptech Ltd.

RESTful Web Services/Session 6

39

Using slide 39, demonstrate how to add the servlet information to the web.xml page.

## Implementing a Web Client 9-14

11. To add a text box and a Submit button to the JSP file, update the code as shown in the following code snippet:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action ="StudentServlet">

            <h1>Student Details</h1>
            Enter Student ID: <input type="text" name="txtID" value="" />
            <input type="submit" value="Submit" name="Submit" />
        </form>
    </body>
</html>
```

Using slide 40, explain the Code Snippet to add a text box and a submit button to the StudentJSP file.

## Implementing a Web Client 10-14

12. To use the Web service to retrieve the student information from the database and display it on the Web page, update the code in the StudentServlet file as shown in the following code snippet:

```
...
protected void processRequest(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet StudentServlet</title>");
        out.println("</head>");
        out.println("<body>");
        String id = request.getParameter("txtID");
        StudentRestfulClient stuInfo = new StudentRestfulClient();
        Students stu = stuInfo.find_XML(Students.class, id);
        out.println("Student ID: " + stu.getStudentid() + "<br/>");
        out.println("Student Name: " + stu.getStudentname() + "<br/>");
        out.println("Student Grade: " + stu.getStudentgrade());
```

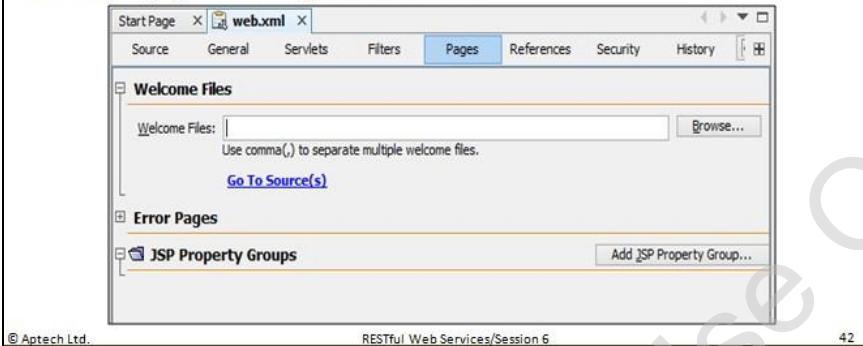
Using slide 41, explain how to update the code in the StudentServlet file in order to use the Web service to retrieve the student information from the database and display it on the Web page.

## Implementing a Web Client 11-14

```
stuInfo.close();
out.println("</body>");
out.println("</html>");
}
}
```

13. To specify the startup file for the project, open web.xml and then, click the Pages tab.

- Following figure shows Pages tab of web.xml:



42

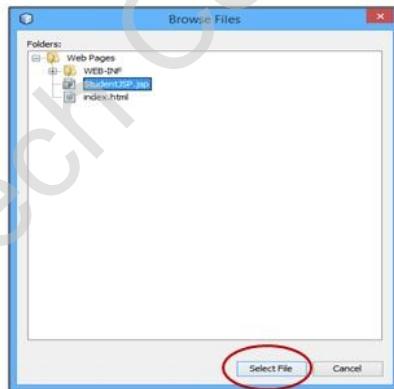
Using slide 42, demonstrate how to specify the startup file for the project.

## Implementing a Web Client 12-14

14. Click the Browse button. In the Browse Files dialog box, select the StudentJSP file.

15. Click Select File button.

- Following figure shows how to select the JSP file:



43

Using slide 43, demonstrate the steps to select the JSP file.

## Implementing a Web Client 13-14

16. Save, build, and run the application. The Web page opens displaying a text box to enter the student ID and a Submit button.

- Following figure shows the JSP page for entering Student ID:

A screenshot of a web browser window titled "JSP Page". The address bar shows "localhost:8080/StudentManager/". The main content area has a title "Student Details". Below it is a form with a text input field containing "Enter Student ID:" and a "Submit" button.

© Aptech Ltd.

RESTful Web Services/Session 6

44

Using slide 44, describe the JSP page that opens up after saving, building, and running the application.

## Implementing a Web Client 14-14

17. Enter the Student ID.

- Following figure shows how to enter the Student ID:

A screenshot of a web browser window titled "JSP Page". The address bar shows "localhost:8080/StudentManager/". The main content area has a title "Student Details". Below it is a form with a text input field containing "STU004" and a "Submit" button.

18. Click the Submit button. The student's details are displayed.

- Following figure shows the output of Student Details:

A screenshot of a web browser window titled "Servlet StudentServlet". The address bar shows "localhost:8080/StudentManager/StudentServlet?txtID=STU004&Submit=Submit". The main content area displays the following text:  
Student ID: STU004  
Student Name: Emma Rogers  
Student Grade: Excellent

© Aptech Ltd.

RESTful Web Services/Session 6

45

Using slide 45, demonstrate how to enter the student ID in the JSP page and how student details are displayed when Submit button is clicked.

**Summarize Session****Slide 46**

**Summary**

- ◆ Web services are Web applications based on XML standards and transport protocol, HTTP.
- ◆ Web services can be developed either using SOAP based approach or REST based approach.
- ◆ REST is an architecture style used for developing Web services. It is a set of guidelines or principles for developing a network-system.
- ◆ RESTful Web services are Web services based on REST principles and accessed over HTTP protocol on the Web.
- ◆ The requirements for developing RESTful Web services based on the REST architecture style are resources, data representation, URI, and HTTP methods.
- ◆ JAX-RS is a Java programming API designed to simplify the development process of RESTful Web service.
- ◆ JAX-RS API is based on annotations such as @Path, @GET, @POST, @PUT, @DELETE, @Produce, @Consumes, and so on which are present in javax.ws.rs package.

Using slide 46, summarize the session. Make them revise the following points:

- Web services are Web applications based on XML standards and transport protocol, HTTP.
- Web services can be developed either using SOAP based approach or REST based approach.
- REST is an architecture style used for developing Web services. It is a set of guidelines or principles for developing a network-system.
- RESTful Web services are Web services based on REST principles and accessed over HTTP protocol on the Web.
- The requirements for developing RESTful Web services based on the REST architecture style are resources, data representation, URI, and HTTP methods.
- JAX-RS is a Java programming API designed to simplify the development process of RESTful Web service.
- JAX-RS API is based on annotations such as @Path, @GET, @POST, @PUT, @DELETE, @Produce, @Consumes, and so on which are present in javax.ws.rs package.

**6.3 Post Class Activities for Faculty**

You should familiarize yourself with the topics of the next session. Since this is the last session, you can skip doing this.

**Tips:** You can check the **Articles/Blogs/Expert Videos** uploaded on the OnlineVarsity site to gain additional information related to the topics covered in this session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.