Advanced Java

Lesson 11—SOA and Web Services

Learning Objectives

- ✓ Explain SOA Architecture
- ✓ Explain Web Services
- ✓ List the steps to create a SOAP based Web Service
- ✓ List the steps to create a RESTful Web Service

SOA and Web Services

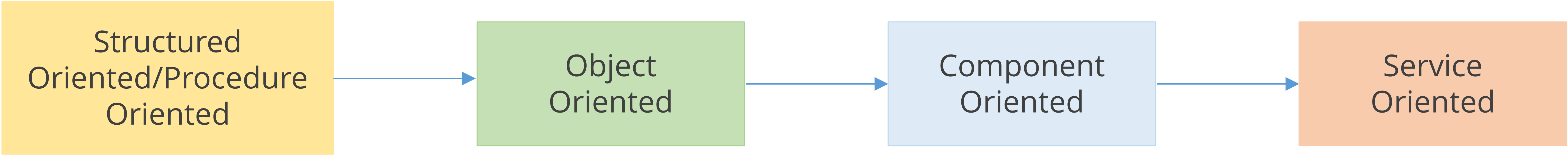
Topic 1—Basics of SOA and SOA Architecture

- Phases of Software Evolution
- What is SOA?
- Advantages of SOA
- SOA Architecture
- Implementing SOA: Example
- SOA Service Description
- Web Service as Implementation of SOA

Phases of Software Evolution

The software evolution has distinct phases. These layers are built up one by one over many years.

Software evolution begins with understanding the concept of 1 and 0 (bits) that gives rise to machine language. This is followed by assembly, procedure oriented, object oriented, component oriented, and service oriented languages.



Structured Oriented/Procedure Oriented

Object Oriented

Component Oriented

Service Oriented

https://www.researchgate.net/publication/215571061

Phases of Software Evolution

STRUCTURE ORIENTED/PROCEDURE ORIENTED

- Structure Oriented/Procedure Oriented
- Object Oriented
- Component Oriented
- Service Oriented

A procedural language is a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program.

It contains a systematic order of statements, functions, and commands to complete a computational task or program.

Phases of Software Evolution

STRUCTURED ORIENTED/PROCEDURE ORIENTED LANGUAGE: LIMITATIONS

- Structured Oriented/Procedure Oriented
- Object Oriented
- Component Oriented
- Service Oriented

- Structure/procedure oriented language is unable to model real world problems.
- It uses global data concept and is thus accessed by all the functions.
- It is difficult to maintain because the line of code is high.

Phases of Software Evolution

OBJECT ORIENTED

Structured Oriented/Procedure Oriented

Object Oriented

Component Oriented

Service Oriented

It refers to a type of computer programming (software design) in which the programmers define the data type of a data structure and the types of operations (functions) that can be applied to the data structure.

Phases of Software Evolution

OBJECT ORIENTED: LIMITATIONS

- Structured Oriented/Procedure Oriented
- Object Oriented
- Component Oriented
- Service Oriented

Although it offers better implementation, data security, code reusability, and flexibility, It is difficult to manage complex applications.

Phases of Software Evolution

COMPONENT ORIENTED

- Structured Oriented/Procedure Oriented
- Object Oriented
- Component Oriented
- Service Oriented

Component-based software engineering (CBSE), also called component-based development (CBD), is a branch of software engineering that emphasizes the separation of concerns with respect to the wide-ranging functionality available throughout a given software system.

A component modifies complexity through the use of a component container and its services.

Phases of Software Evolution

COMPONENT ORIENTED: LIMITATIONS

Structured Oriented/Procedure Oriented

Object Oriented

Component Oriented

Service Oriented

Component development is a complex task and comes with the following limitations:

- Component maintenance costs
- Reliability and sensitivity to changes
- Unsatisfied requirements

Phases of Software Evolution

SERVICE ORIENTED

- Structured Oriented/Procedure Oriented
- Object Oriented
- Component Oriented
- Service Oriented

Web Services are self-describing services that will perform well-defined tasks and can be accessed through the Web.

Service Oriented approach follows an architecture, called SOA (Service Oriented Architecture), that focuses on building systems through the use of different Web Services and integrating them to make up the whole system.

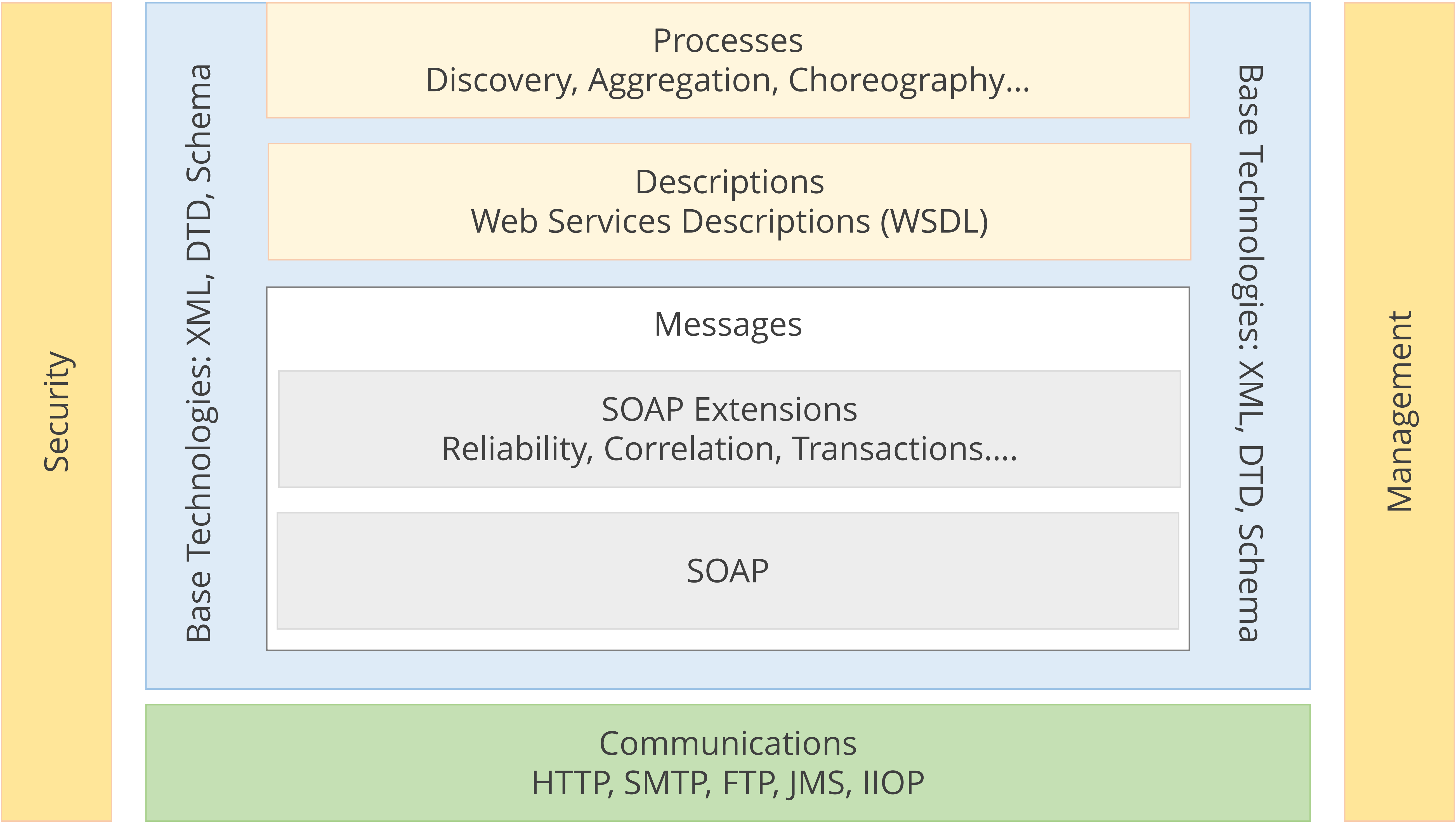
What is SOA?

AN ARCHITECTURAL SOLUTION

SOA is the architectural solution for integrating diverse systems by providing an architectural style that promotes loose coupling reuse.

It is a programming model or paradigm where web service and contracts become~~s~~ a dominant design for interoperability

SOA Architecture



Security

Base Technologies: XML, DTD, Schema

Processes
Discovery, Aggregation, Choreography…

Descriptions
Web Services Descriptions (WSDL)

Messages

SOAP Extensions
Reliability, Correlation, Transactions….

SOAP

Base Technologies: XML, DTD, Schema

Management

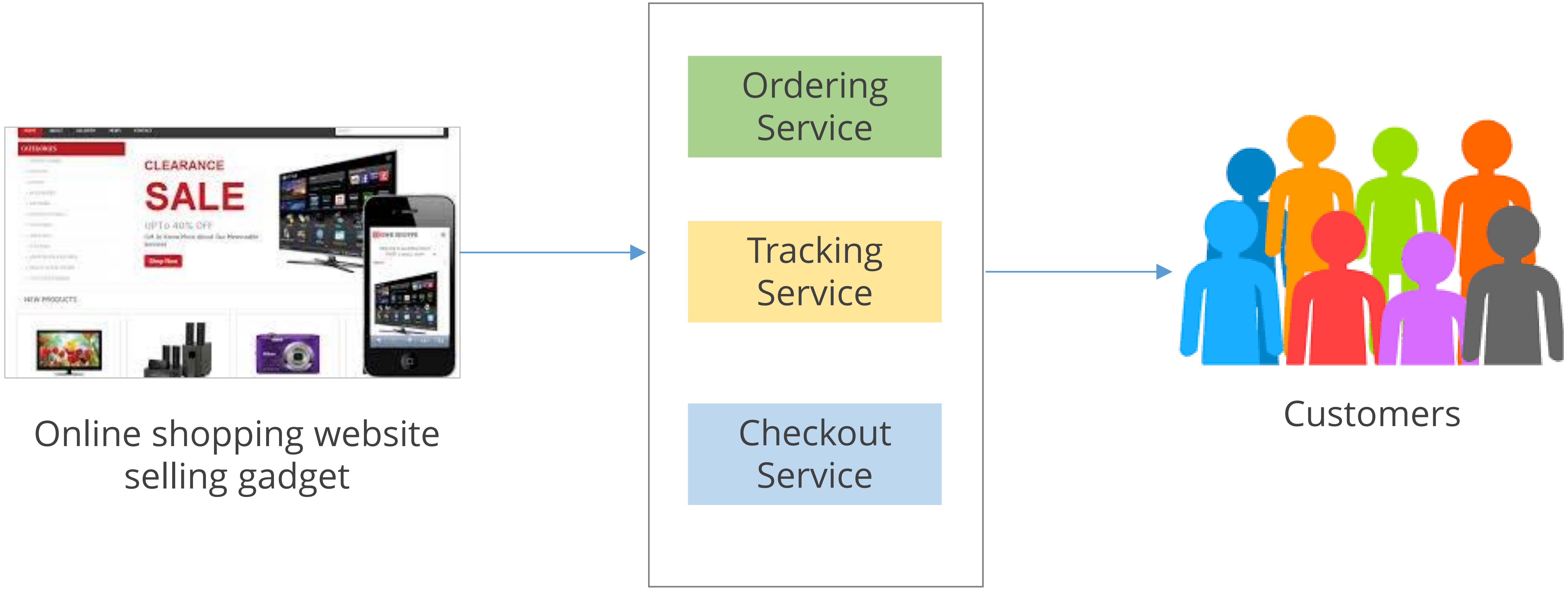
Communications
HTTP, SMTP, FTP, JMS, IIOP

What Is SOA?

A COLLECTION OF SERVICES

SOA is a collection of Services.

The following diagram shows how an Online Shopping website selling gadgets is involved with different services like Ordering, Tracking, and Checkout to give service to its customers.



Ordering Service

Tracking Service

Checkout Service

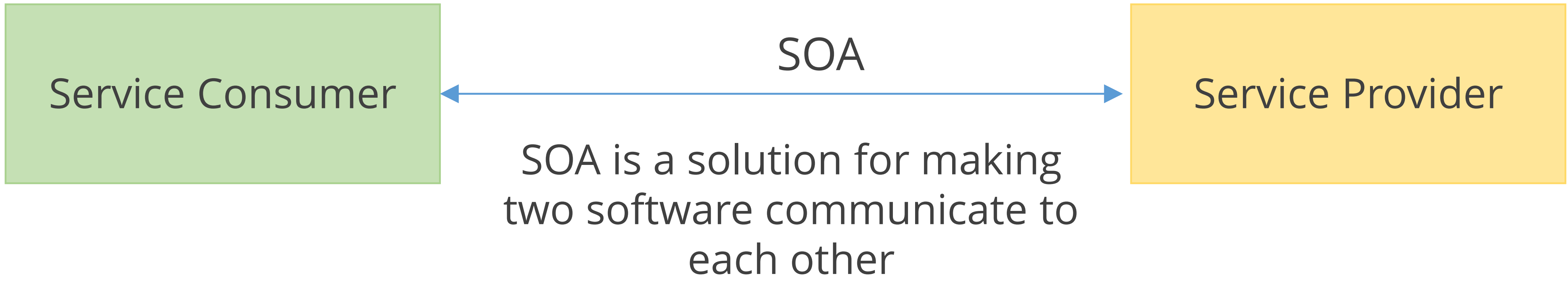
Online shopping website selling gadget

Customers

What Is SOA?

ESTABLISHING COMMUNICATION

It helps in establishing communication between all services (applications).

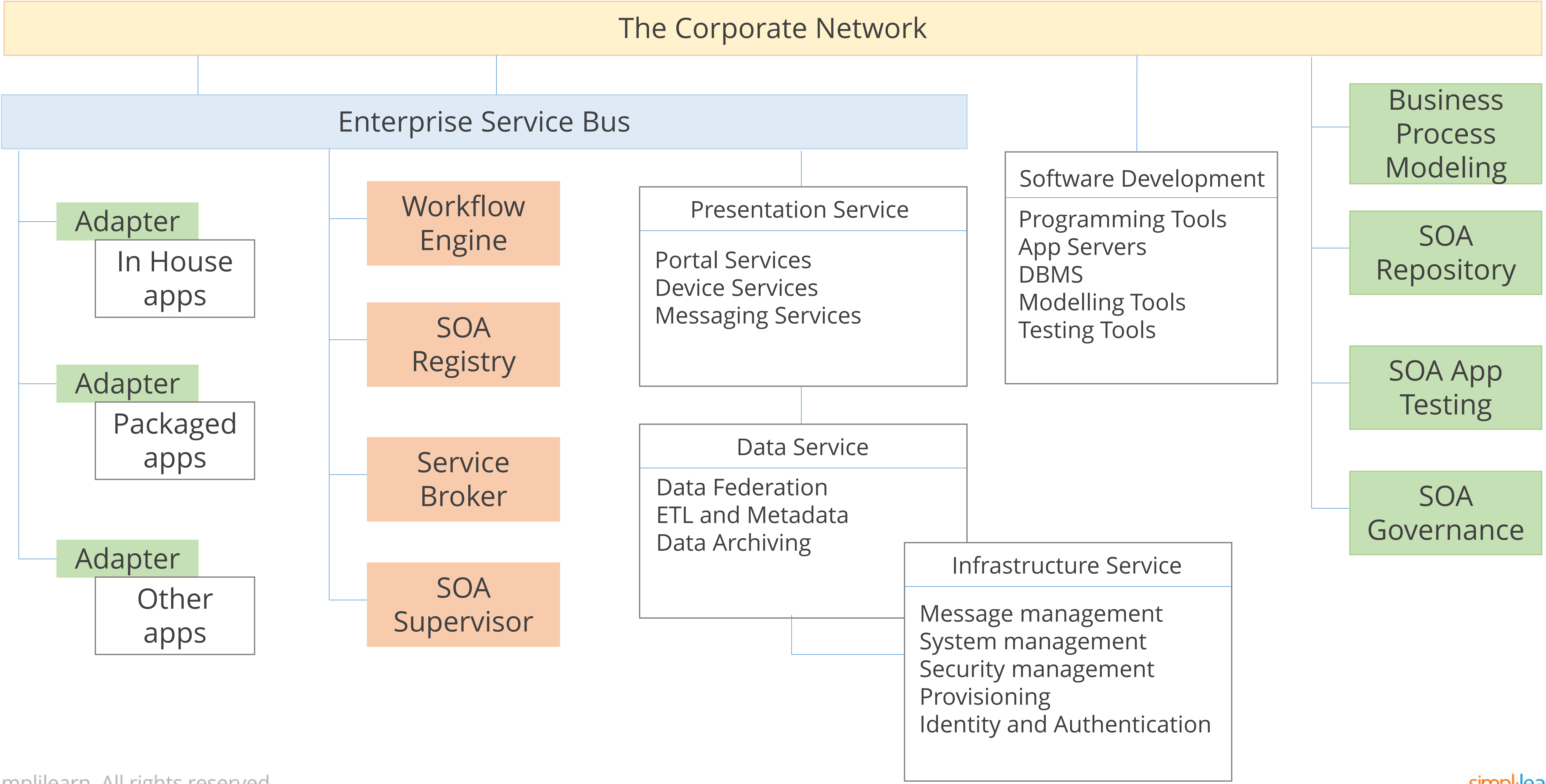


SOA is a solution for making two software communicate to each other

Advantages of SOA

SOA Services	Uses
Service	• Improved information flow • Ability to expose internal functionality • Organizational flexibility
Service Re-use	Lower software development and management costs
Messaging	Configuration flexibility
Message Monitoring	• Business intelligence • Performance measurement • Security attack detection
Message Control	• Application of management policy • Application of security policy
Message Transformation	Data translation
Message Security	Data confidentiality and integrity
Complex Event Processing	• Simplification of software structure • Ability to adapt quickly to different external environments • Improved manageability and security

Implementing SOA: Example



The Corporate Network

Enterprise Service Bus

Adapter

In House apps

Adapter

Packaged apps

Adapter

Other apps

Workflow Engine

SOA Registry

Service Broker

SOA Supervisor

Presentation Service

Portal Services
Device Services
Messaging Services

Data Service

Data Federation
ETL and Metadata
Data Archiving

Infrastructure Service

Message management
System management
Security management
Provisioning
Identity and Authentication

Software Development

Programming Tools
App Servers
DBMS
Modelling Tools
Testing Tools

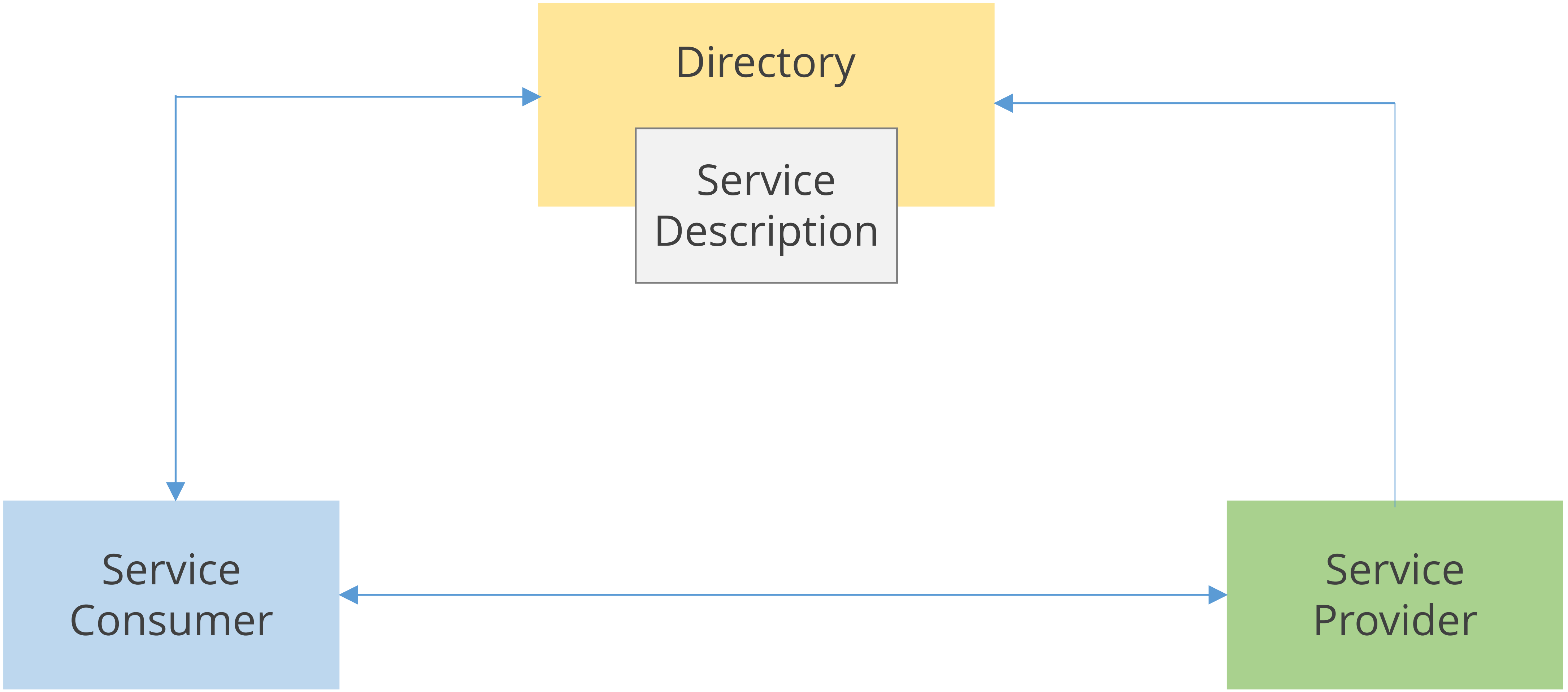
Business Process Modeling

SOA Repository

SOA App Testing

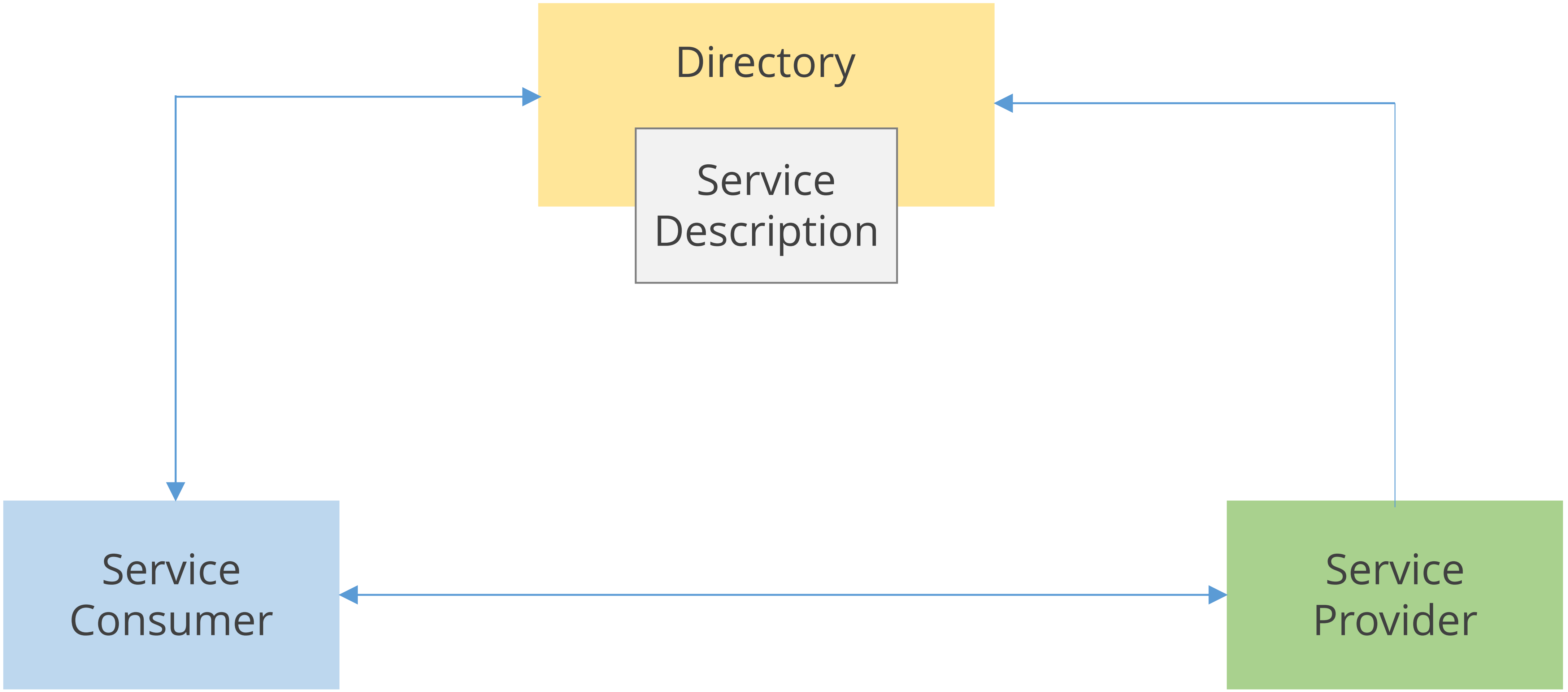
SOA Governance

SOA Service Description



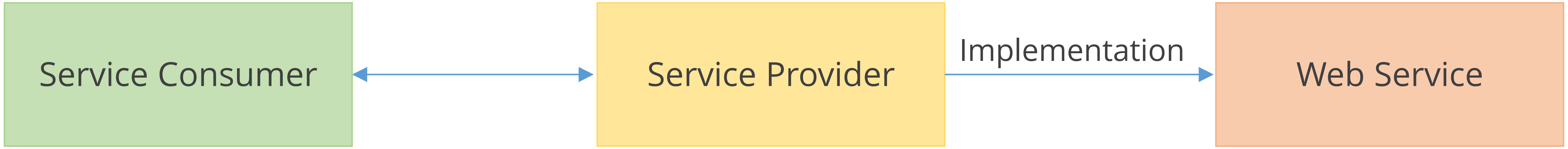
Consumer queries the directory to locate a service and communicate with the provider

SOA Service Description



Service Provider publishes its service description and directory

Web Service as Implementation of SOA



Web Service is an implementation of SOA

SOA and Web Services

Topic 2—Web Services

- Why Use Web Services?
- What Are Web Services?
- Web Services Operational Model
- Web Services: Example
- Types of Web Services

Why Use Web Services?

1. Web applications enable interaction between an end user and a website. Web services are service-oriented and enable application-to-application communication over the Internet and easy accessibility to heterogeneous applications and devices.

2. Web services can be invoked through XML-based RPC mechanisms across firewalls.

3. Web services provide a cross-platform, cross-language solution based on XML messaging.

4. Web services facilitate ease of application integration using a lightweight infrastructure without affecting scalability.

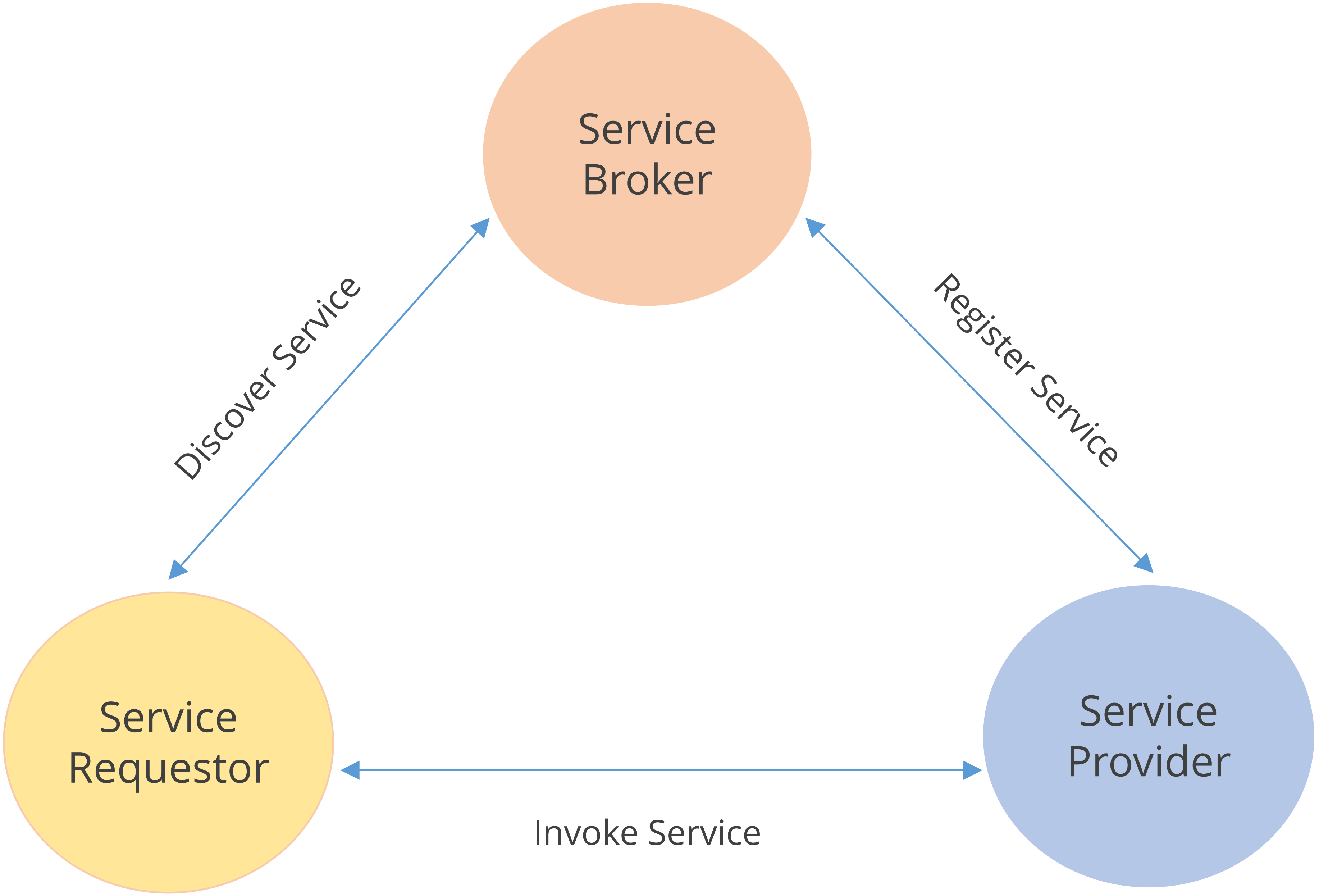
5. Web services enable interoperability among heterogeneous applications.

What Are Web Services?

- Web services are based on the concept of service-oriented architecture (SOA).
- According to Gartner research (June 15, 2001), “Web services are loosely coupled software components delivered over Internet standard technologies.”
- Web services are self-describing and modular business applications that expose the business logic as services over the Internet. This is done through programmable interfaces and Internet protocols to provide ways to find, subscribe, and invoke those services.

Web Services Operational Model

Web Service operational model can be conceptualized as a simple operational model that has a lot in common with a standard communication model:



Operations are conceived as involving three distinct roles and relationships that define the Web Services provider and users.

Web Services: Example

Consider the following scenario:

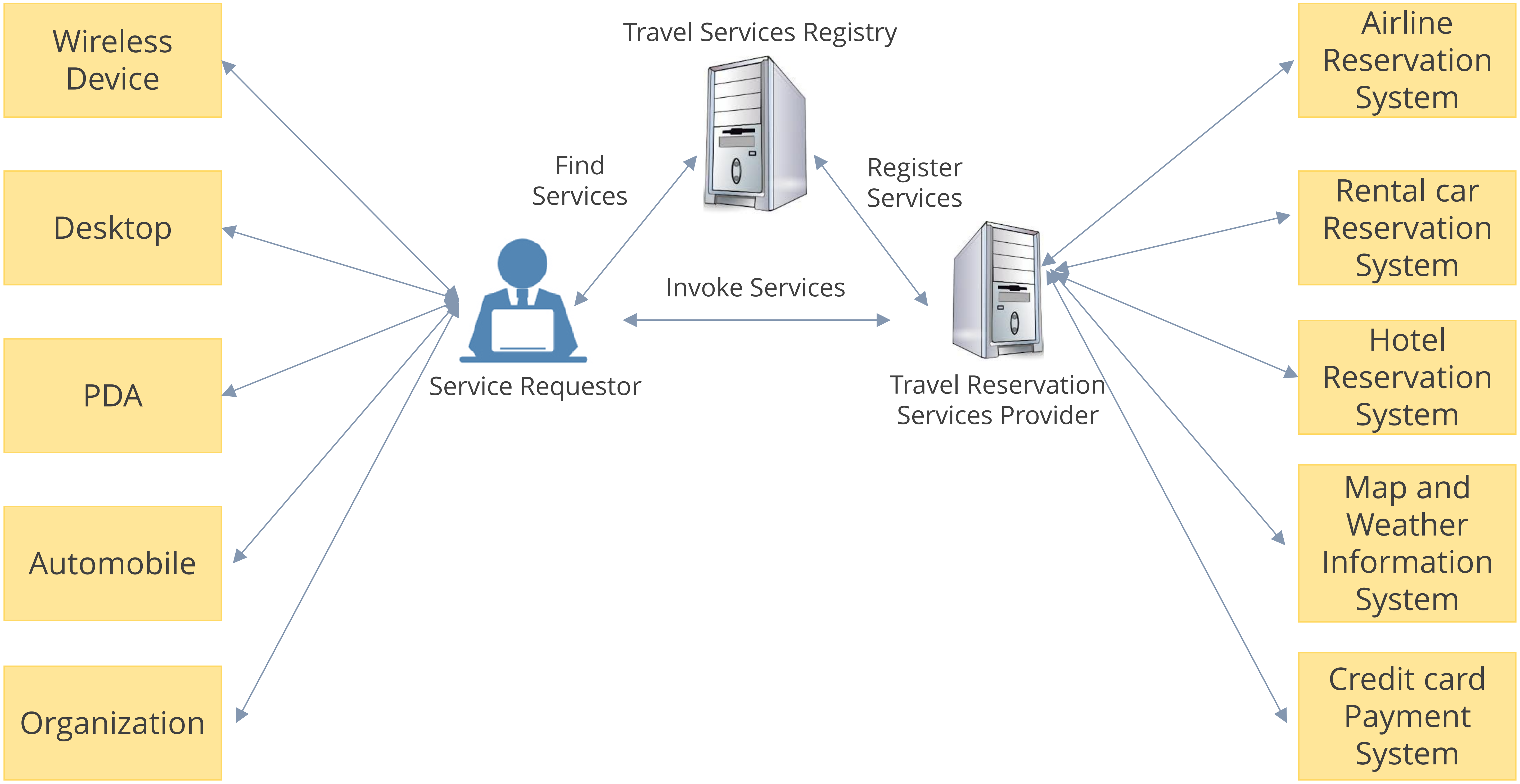
1. A Travel service provider deploys its Web services by exposing the business applications obtained from different travel businesses like airlines, car-rental, hotel accommodation, credit card payment, and so forth.

2. The service provider registers its business services with descriptions using a public or private registry. The registry stores the information about the services exposed by the service provider.

3. The customer discovers the Web services using a search engine or by locating it directly from the registry and then invokes the Web services to perform travel reservations and other functions over the Internet using any platform or device.

4. In the case of large-scale organizations, the business applications consume these Web services to provide travel services to their own employees through the corporate intranet.

Web Services: Example



Wireless Device

Desktop

PDA

Automobile

Organization

Travel Services Registry

Find Services

Register Services

Invoke Services

Service Requestor

Travel Reservation Services Provider

Airline Reservation System

Rental car Reservation System

Hotel Reservation System

Map and Weather Information System

Credit card Payment System

Types of Web Services

Web services are based on the following key standards:

1. Extensible Markup Language (XML) based messaging
2. Universal Description, Discovery, and Integration (UDDI)
3. Web Services Description Language (WSDL)
4. Simple Object Access Protocol (SOAP)

Types of Web Services

EXTENSIBLE MARKUP LANGUAGE (XML)

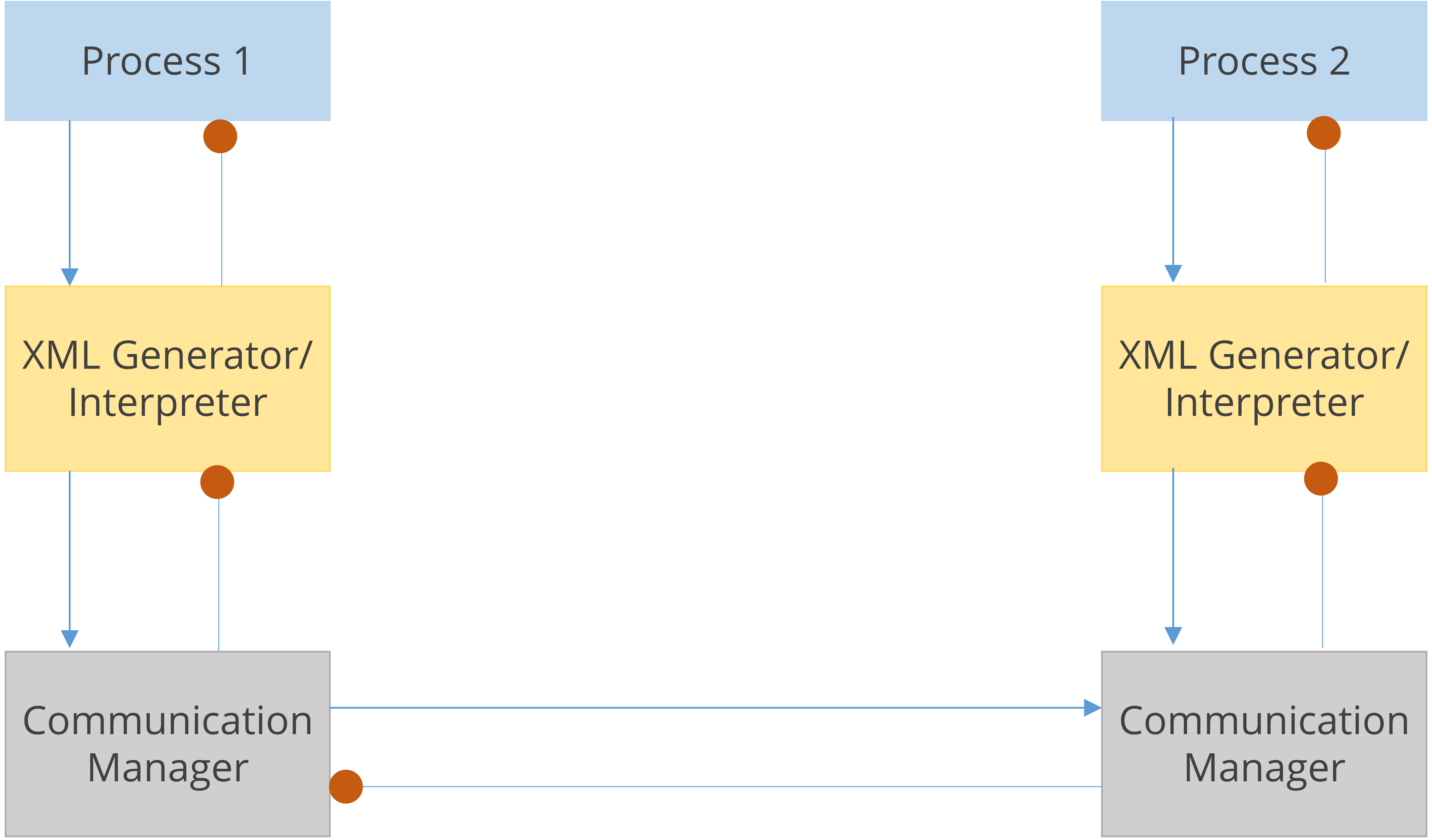
Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)

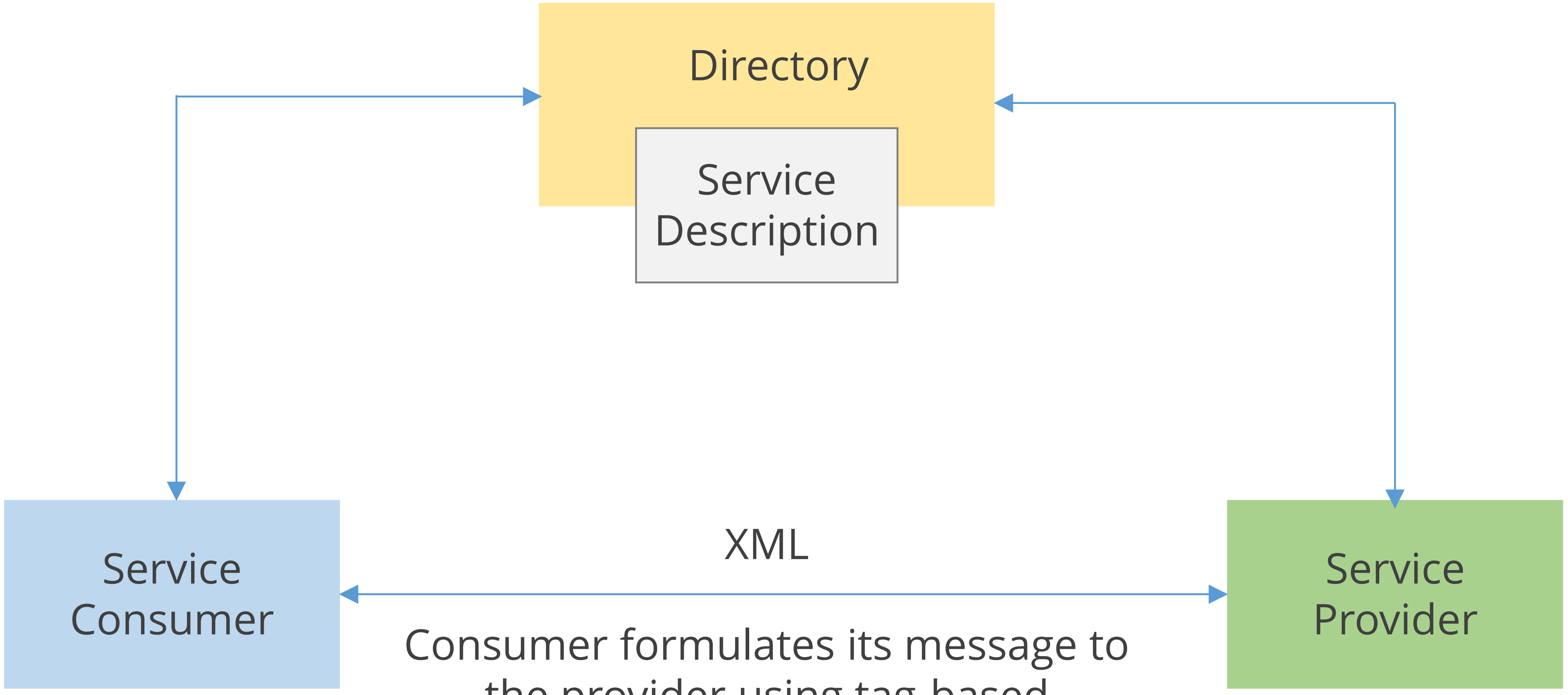
These technologies are interoperable at a core level as they use XML as data representation layer for all web service protocols and technologies that are created



Types of Web Services

WEB SERVICES WITH XML

- Extensible Markup Language (XML)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)
- Simple Object Access Protocol (SOAP)



Consumer formulates its message to the provider using tag-based language called XML.

Types of Web Services

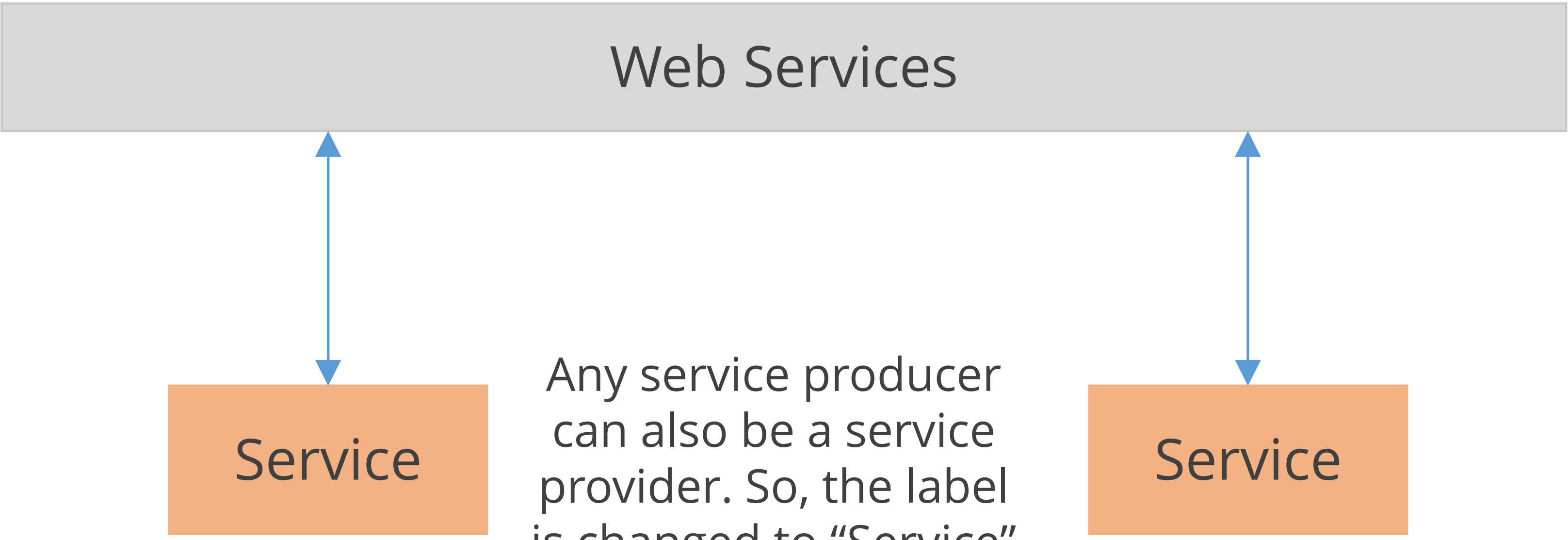
UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI)

- Extensible Markup Language (XML)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)
- Simple Object Access Protocol (SOAP)

UDDI provides a world-wide registry of web services for advertisements, discovery, and integration purposes.

UDDI could be dynamic or static; Business Analysts and technologists could use UDDI to search for available web services.

The UDDI Business Registry provides a place for a company to programmatically describe its services and business processes and its preferred methods for conducting business.



The directory is implicit in web services

Any service producer can also be a service provider. So, the label is changed to “Service”

Types of Web Services

WEB SERVICES DESCRIPTION LANGUAGE (WSDL)

- Extensible Markup Language (XML)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)
- Simple Object Access Protocol (SOAP)

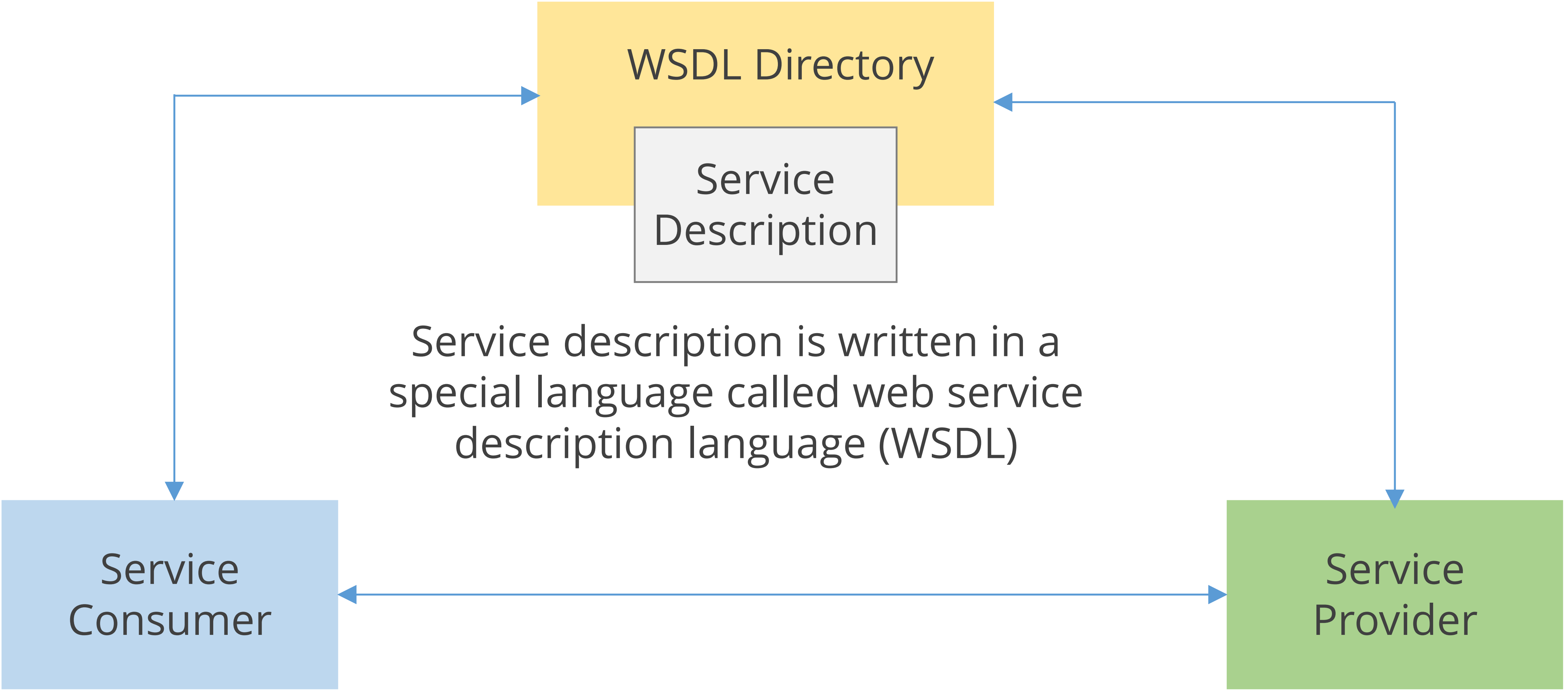
WSDL describes the interface of the Web Service. It is comparable to the concept of “Remote Interface” of Java RMI or the “Interface Definition Language File (IDL)” of RPC.

It standardizes how a web service represents the input and output parameters of an invocation externally, that is, the function structure, the nature of invocation (in, in/out,etc), and the service protocol binding.

Types of Web Services

WEB SERVICES WITH WSDL

- Extensible Markup Language (XML)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)
- Simple Object Access Protocol (SOAP)



Service description is written in a special language called web service description language (WSDL)

Types of Web Services

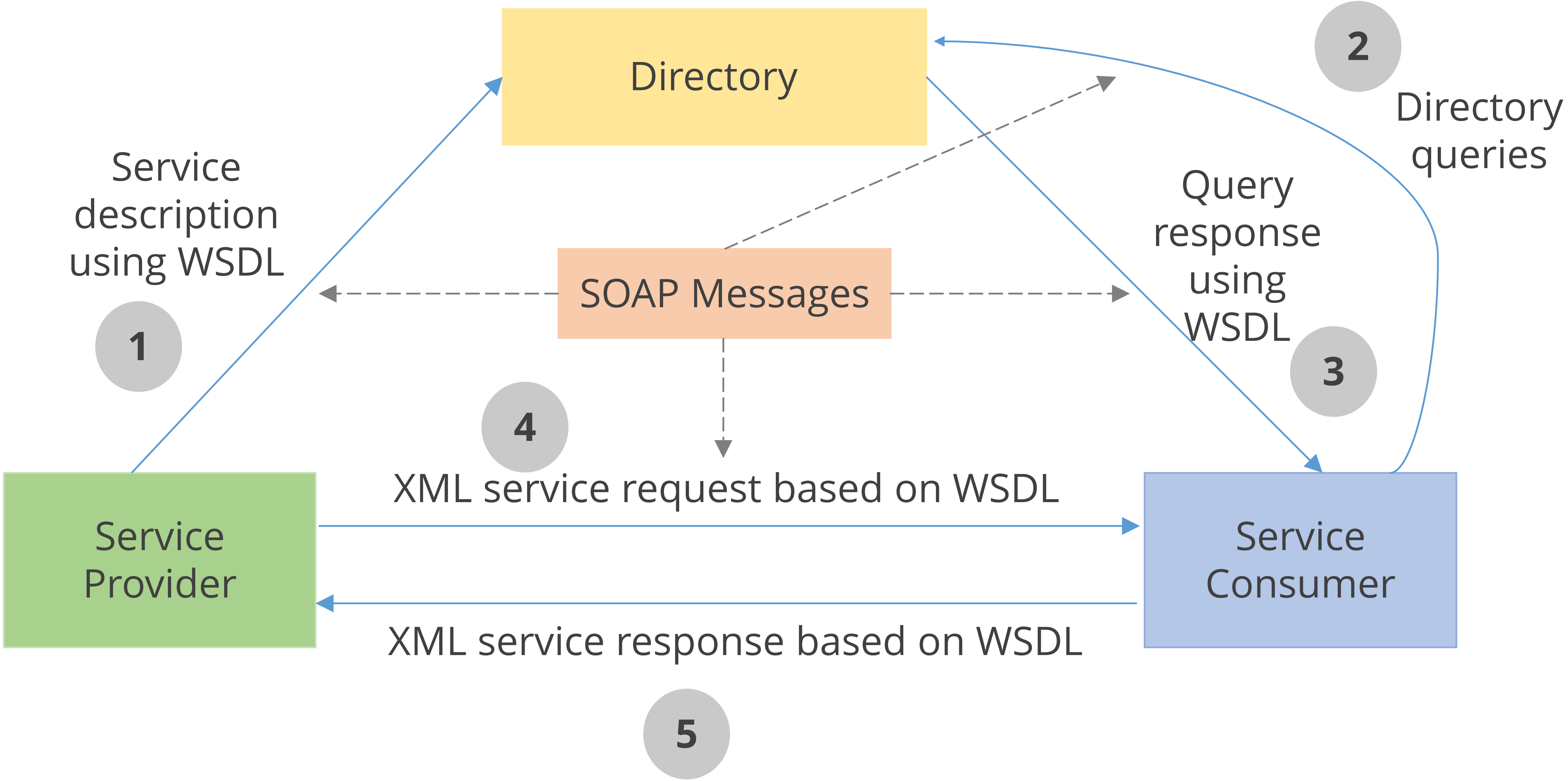
USE OF WSDL

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)


Directory
2
Directory queries
Service description using WSDL
1
Query response using WSDL
3
SOAP Messages
4
XML service request based on WSDL
Service Provider
Service Consumer
XML service response based on WSDL
5

Types of Web Services

SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)

SOAP is a simple XML-based protocol to let applications exchange information over HTTP.

It is a protocol for accessing a Web Service.

Types of Web Services

WHY USE SIMPLE OBJECT ACCESS PROTOCOL (SOAP)?

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

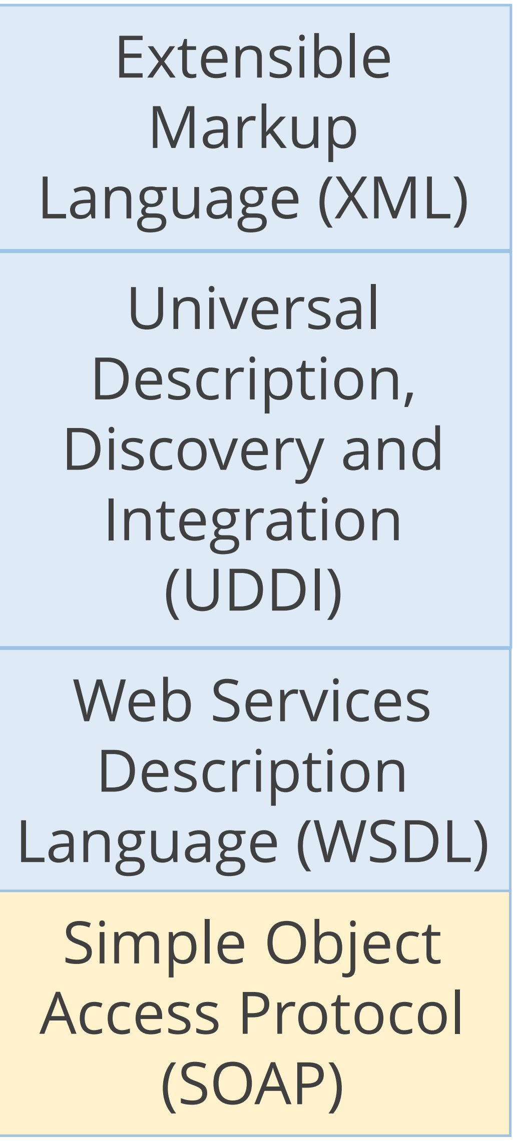
Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)

- It is important to have Internet communication between programs for application development.
- Firewalls and proxy servers generally block traffic that creates a compatibility and security problem, even if they are used for application communication like RPC (Remote Procedure Call).
- HTTP is supported by all Internet browsers and servers. However, SOAP was created to accomplish better communication between applications.
- SOAP supports different technologies and programming languages.

Types of Web Services

SOAP ARCHITECTURE

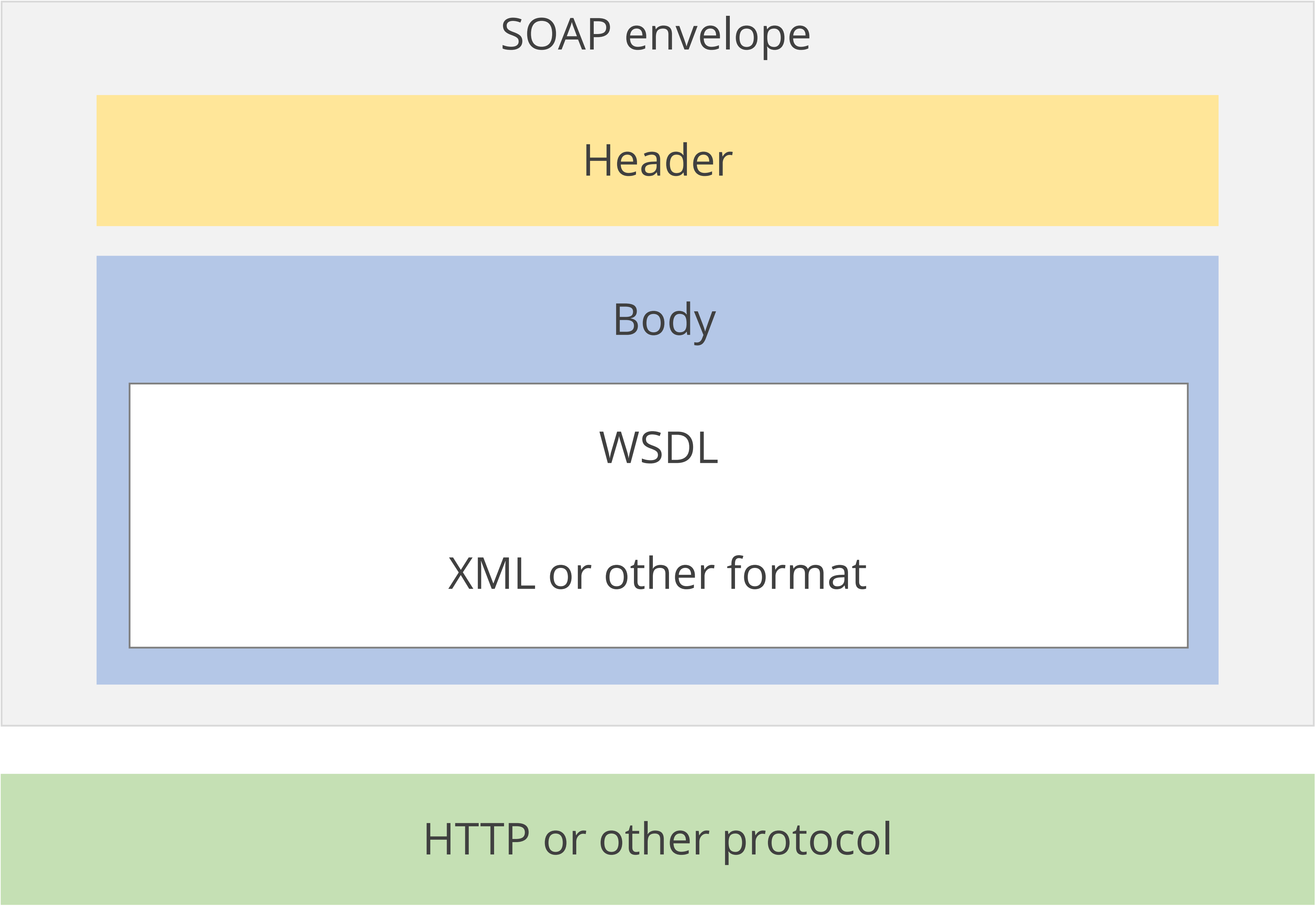


Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)



SOAP envelope

Header

Body

WSDL

XML or other format

HTTP or other protocol

Types of Web Services

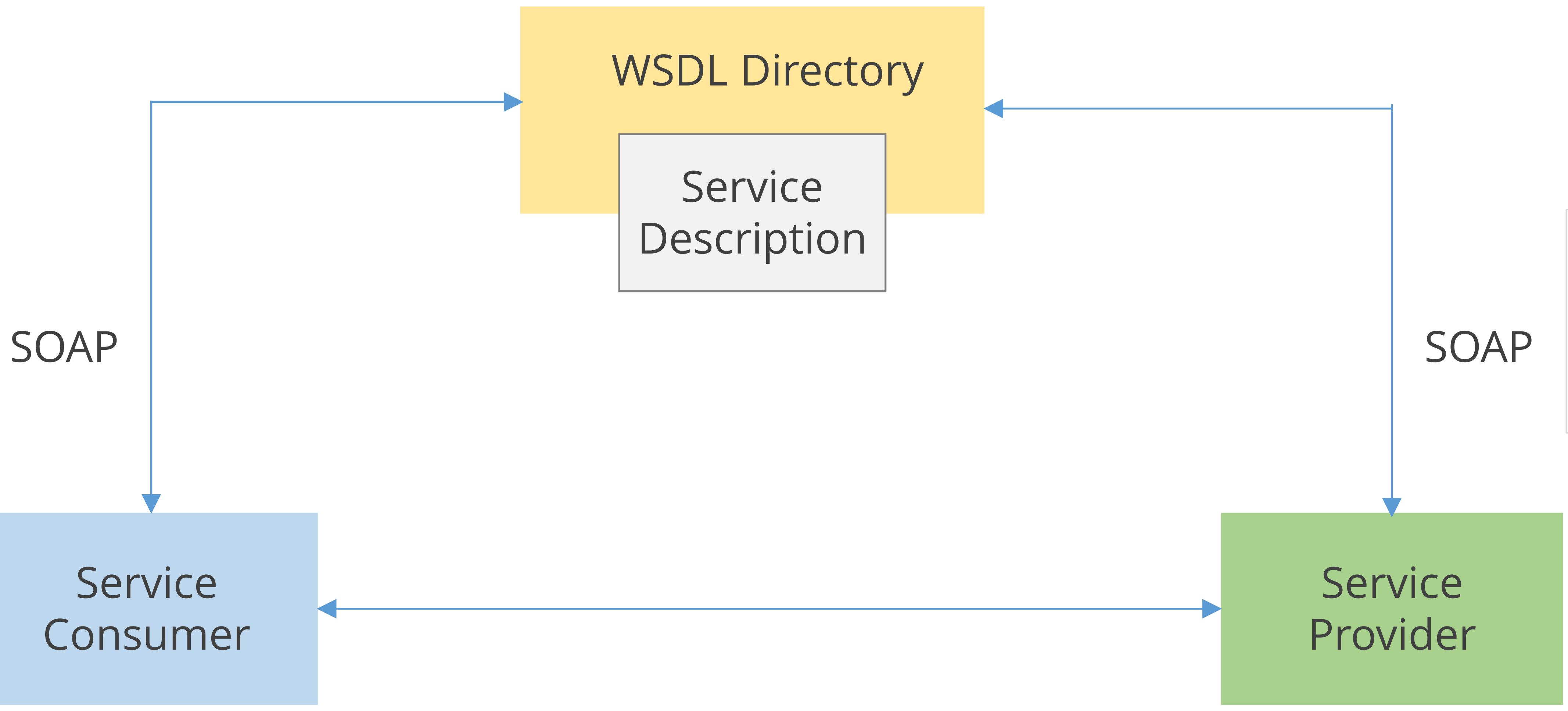
WEB SERVICES WITH SOAP

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)



Messages are sent and received in a directory through SOAP language

Types of Web Services

SOAP ELEMENTS

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)

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

Types of Web Services

SOAP ELEMENTS: ENVELOPE

- It is the root element of a SOAP message
- It defines the XML document as SOAP message

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)

Envelope	Header	Body	Fault

Types of Web Services

SOAP ELEMENTS: HEADER

- Extensible Markup Language (XML)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)
- Simple Object Access Protocol (SOAP)

- The optional SOAP Header element contains application-specific information (like authentication, payment, etc.) about the SOAP message.

- If the Header element is present, it must be the first child element of the Envelope element.

All immediate child elements of the Header element must be namespace-qualified.

Envelope	Header	Body	Fault

Types of Web Services

SOAP ELEMENTS: BODY

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)

- The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.
- Immediate child elements of the SOAP Body element may be namespace-qualified.
- This is the SOAP Fault element used to indicate error messages.

Envelope	Header	Body	Fault

Types of Web Services

SOAP ELEMENTS: FAULT

Extensible Markup Language (XML)

Universal Description, Discovery and Integration (UDDI)

Web Services Description Language (WSDL)

Simple Object Access Protocol (SOAP)

- The optional SOAP Fault element is used to indicate error messages.
- The SOAP Fault element holds errors and status information for a SOAP message.
- If a Fault element is present, it must appear as a child element of the Body element. A Fault element can only appear once in a SOAP message.
- The SOAP Fault element has the following sub elements:

Sub Element	Description
<faultcode>	A code for identifying the fault
<faultstring>	A human readable explanation of the fault
<faultactor>	Information about who caused the fault to happen
<detail>	Holds application specific error information related to the Body element

Envelope | Header | Body | Fault

SOA and Web Services

DEMO—SOAP Application

Java APIs for Web Service

There are two main APIs defined by Java for developing web service applications (since Java EE 6):

1. JAX-WS: for SOAP web services.
2. JAX-RS: for RESTful web services.

SOA and Web Services

Topic 3—Creating a SOAP based Web Service

- Axis
- Web Service in an Axis Environment
- Creating a SOAP based Web Service

Axis

Axis facilitates easy deployment and undeployment of services using XML-based Web services deployment descriptors (WSDDs).

- It enables deploying and undeploying services and also Axis-specific resources like handlers and chains using an administration utility ‘AdminClient’ provided as part of the Axis toolkit.
- It helps deploy a service, ensures that the AXIS CLASSPATH is set, and runs the following command:

```
java org.apache.axis.client.AdminClient deploy.wsdd
```

- It helps undeploy a service, ensures that the AXIS CLASSPATH is set, and then runs the following command:

```
java org.apache.axis.client.AdminClient undeploy.wsdd
```

Web Service in an Axis Environment

To create a Web service in an Axis environment, the following Models are used:

1. Create the service provider (server application)

2. Create the service requester (client applications)

Creating a SOAP based Web Service

1. Create a new dynamic web project and name it “SimpleSOAPExample”

2. Create a new package named “com.webservice”

3. Create a simple Java class named “HelloWorld.java” which works as Service

4. Right click project → new → web service

5. Click next

   In service implementation text box, write the qualified class name of created class (HelloWorld.java), move both above slider to maximum level (Test service and Test Client level), and click finish. You are done. A new project named “SimpleSOAPExampleClient” will be created in your work space.

6. Click start server

7. After clicking start server, eclipse will open test web service API. With this test API, you can test your web service

Creating a SOAP based Web Service

Create a new dynamic web project and name it “SimpleSOAPExample”

Create a new package named “com.webservice”

Create a simple Java class named “HelloWorld.java”

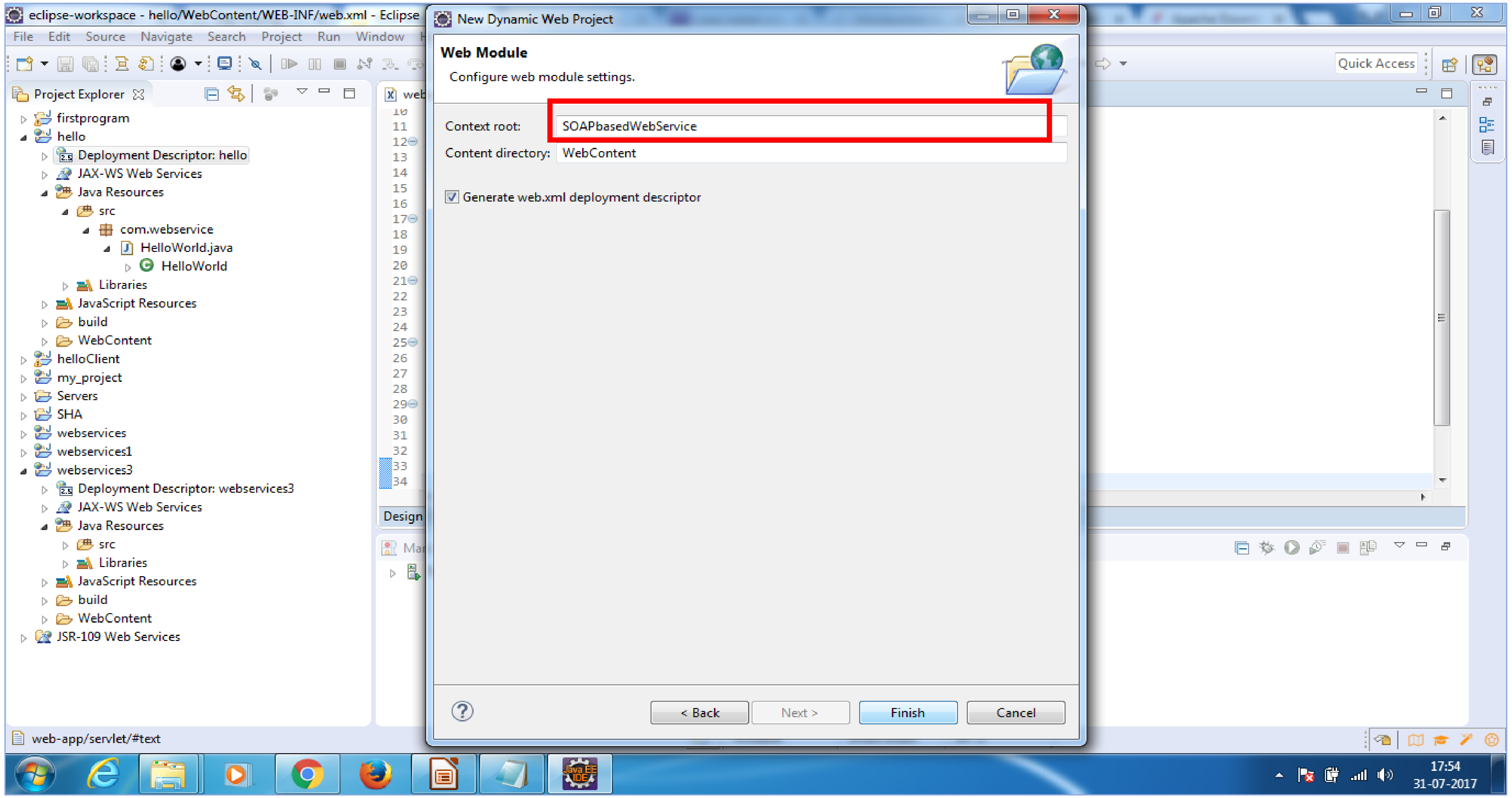
Right click project → new → web service

Click next

Click start server

Test your Web service

STEP 1: Create a new dynamic web project and name it “SimpleSOAPExample”



Web Module

Configure web module settings.

Context root: SOAPbasedWebService

Content directory: WebContent

☑ Generate web.xml deployment descriptor

Creating a SOAP based Web Service

Create a new dynamic web project and name it “SimpleSOAPExample”

Create a new package named “com.webservice”

Create a simple Java class named “HelloWorld.java”

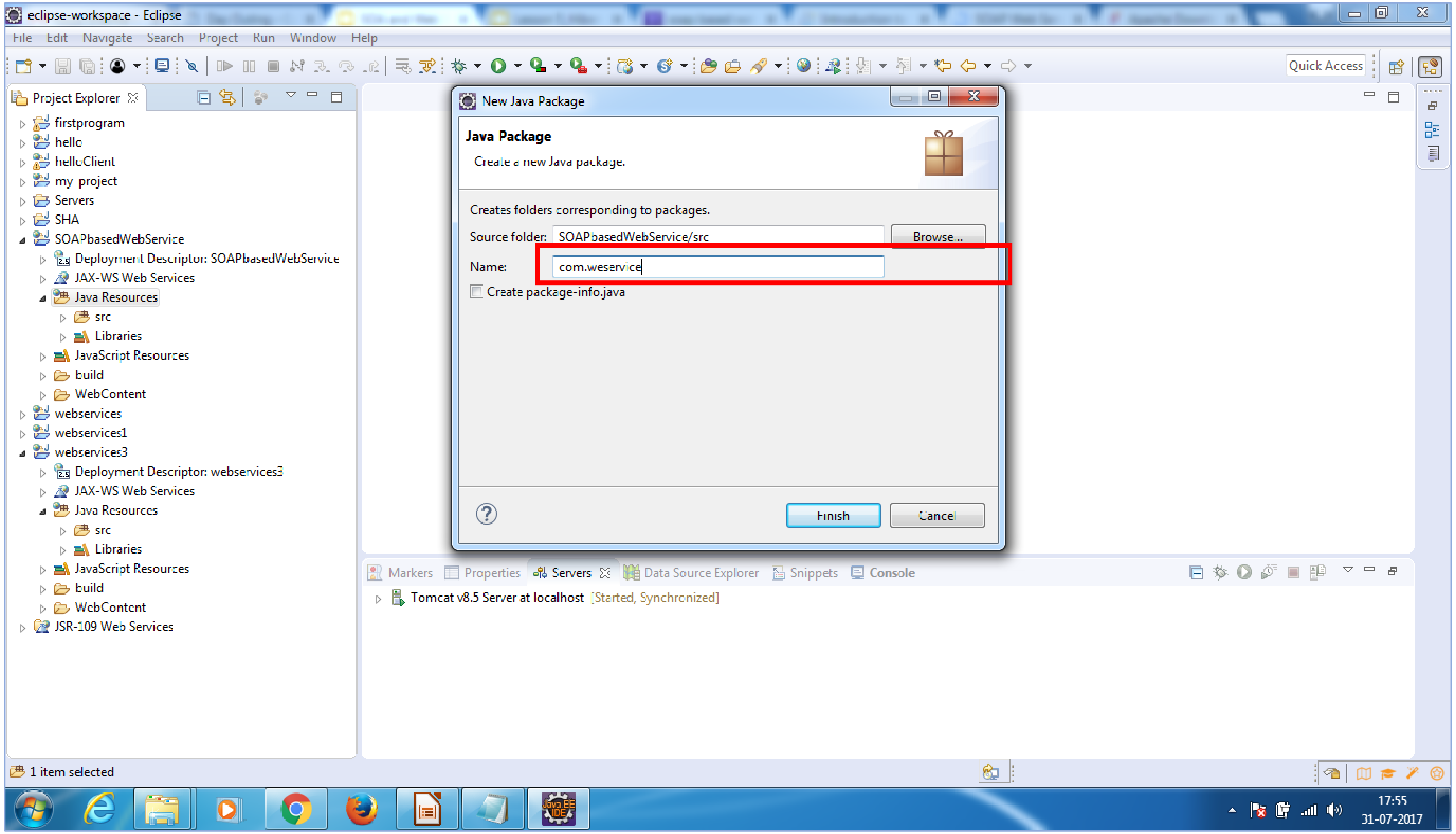
Right click project → new → web service

Click next

Click start server

Test your Web service

STEP 2: Create a new package named “com.webservice”



New Java Package

Java Package

Create a new Java package.

Creates folders corresponding to packages.

Source folder: SOAPbasedWebService/src [Browse...]

Name: com.weservice

□ Create package-info.java

[Finish] [Cancel]

Creating a SOAP based Web Service

- Create a new dynamic web project and name it “SimpleSOAPExample”
- Create a new package named “com.webservice”
- Create a simple Java class named “HelloWorld.java”
- Right click project → new → web service
- Click next
- Click start server
- Test your Web service

STEP 3: Create a simple Java class named “HelloWorld.java”

The following example is of a simple service that accepts a string parameter, invokes a method justSayHello, and returns a String parameter:

```
package com.weservice;

public class HelloWorld {

       public String sayHelloWorld(String name)
       {
       return "Hello world from "+ name;
       }
      }
```

Creating a SOAP based Web Service

Create a new dynamic web project and name it “SimpleSOAPExample”

Create a new package named “com.webservice”

Create a simple Java class named “HelloWorld.java”

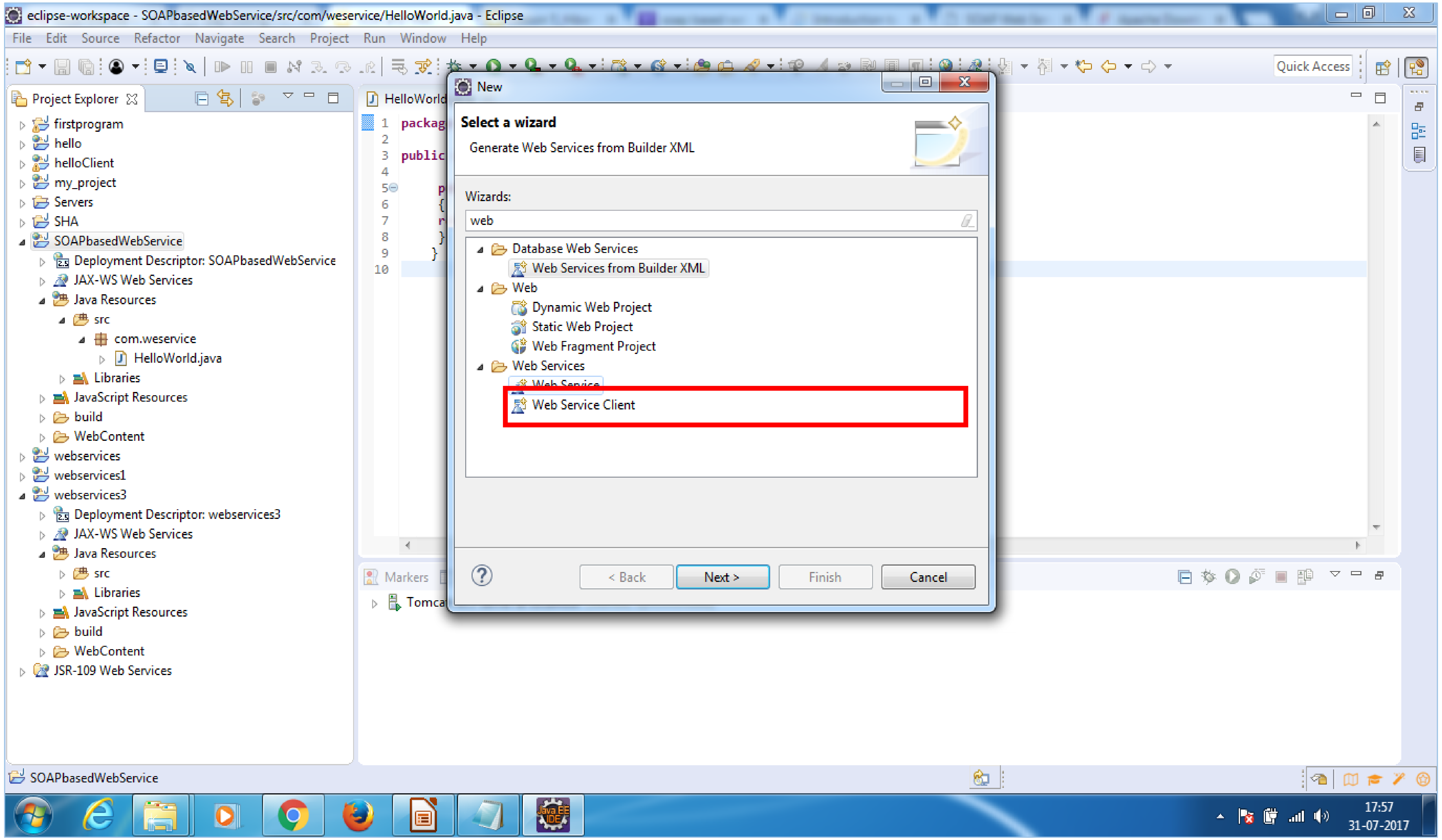
Right click project → new → web service

Click next

Click start server

Test your Web service

STEP 4: Right click project → new → web service



Select a wizard

Generate Web Services from Builder XML

Wizards:

web

Creating a SOAP based Web Service

Create a new dynamic web project and name it "SimpleSOAPExample"

Create a new package named "com.webservice"

Create a simple Java class named "HelloWorld.java"

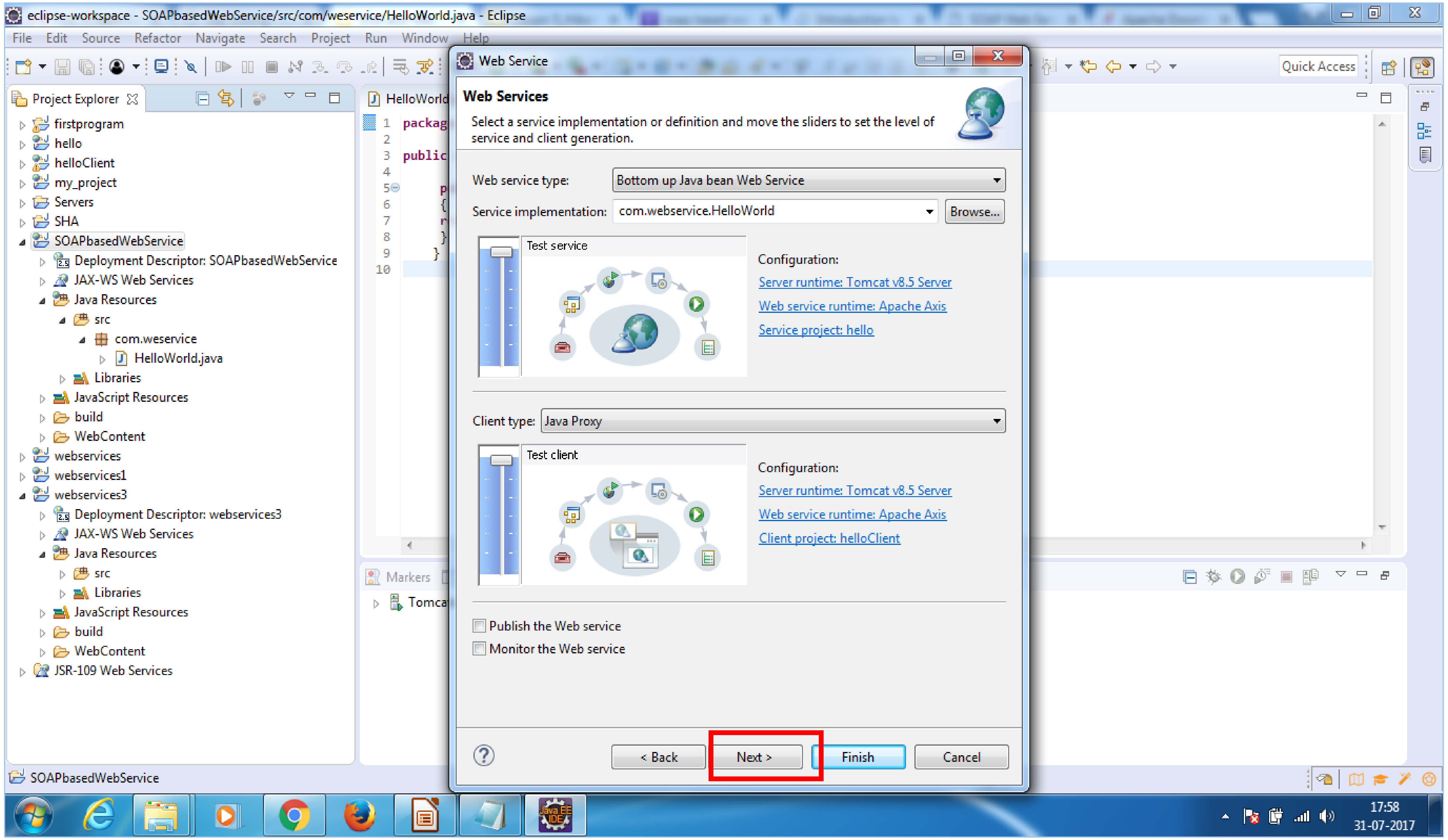
Right click project → new → web service

Click next

Click start server

Test your Web service

STEP 5: CLICK NEXT



Web Services

Select a service implementation or definition and move the sliders to set the level of service and client generation.

Web service type: Bottom up Java bean Web Service

Service implementation: com.webservice.HelloWorld

Configuration:
Server runtime: Tomcat v8.5 Server
Web service runtime: Apache Axis
Service project: hello

Client type: Java Proxy

Configuration:
Server runtime: Tomcat v8.5 Server
Web service runtime: Apache Axis
Client project: helloClient

□ Publish the Web service
□ Monitor the Web service

Creating a SOAP based Web Service

Create a new dynamic web project and name it “SimpleSOAPExample”

Create a new package named “com.webservice”

Create a simple Java class named “HelloWorld.java”

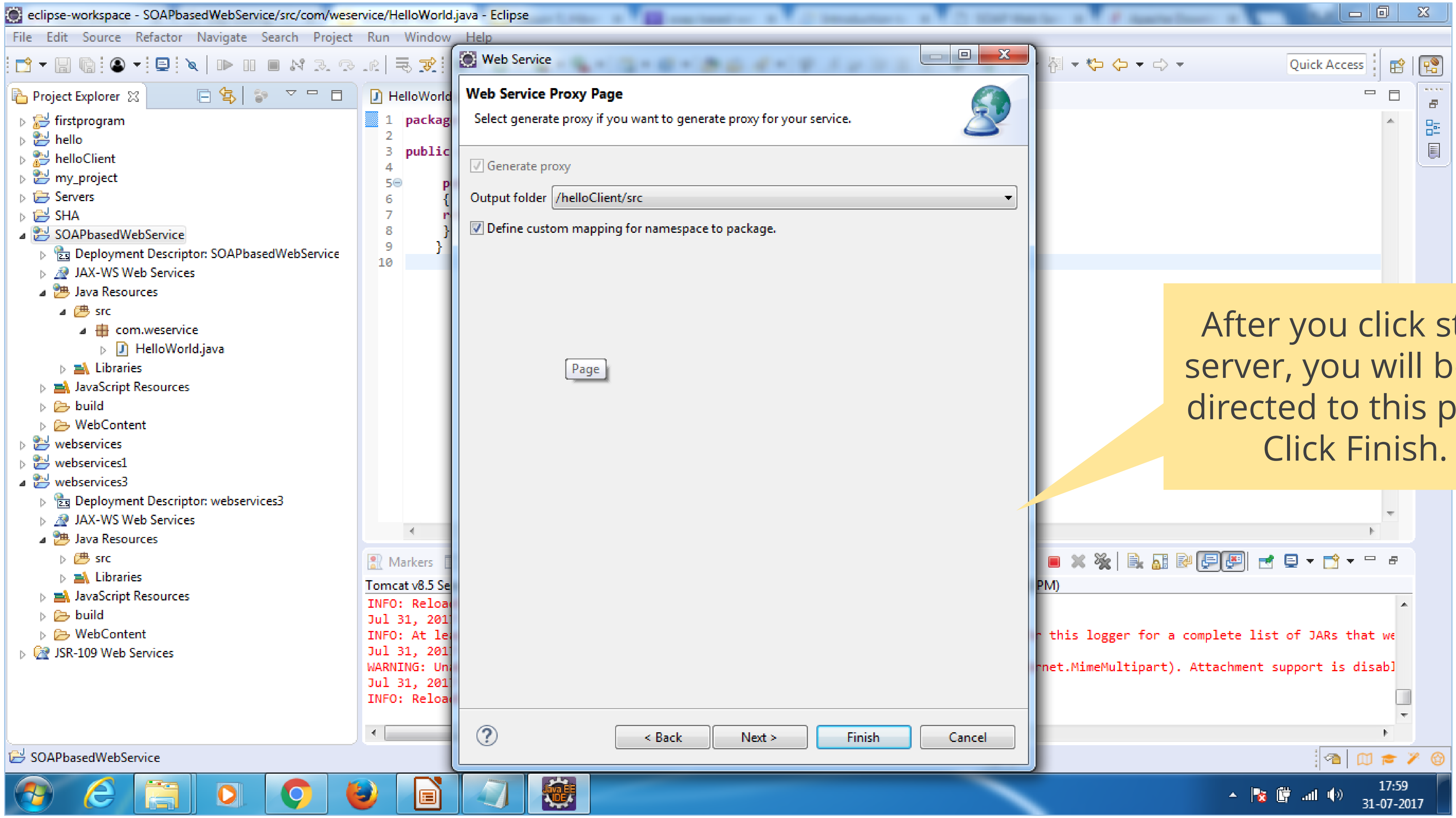
Right click project → new → web service

Click next

Click start server

Test your Web service

STEP 6: CLICK START SERVER



eclipse-workspace - SOAPbasedWebService/src/com/weservice/HelloWorld.java - Eclipse

Web Service

Web Service Proxy Page

Select generate proxy if you want to generate proxy for your service.

☑ Generate proxy

Output folder /helloClient/src

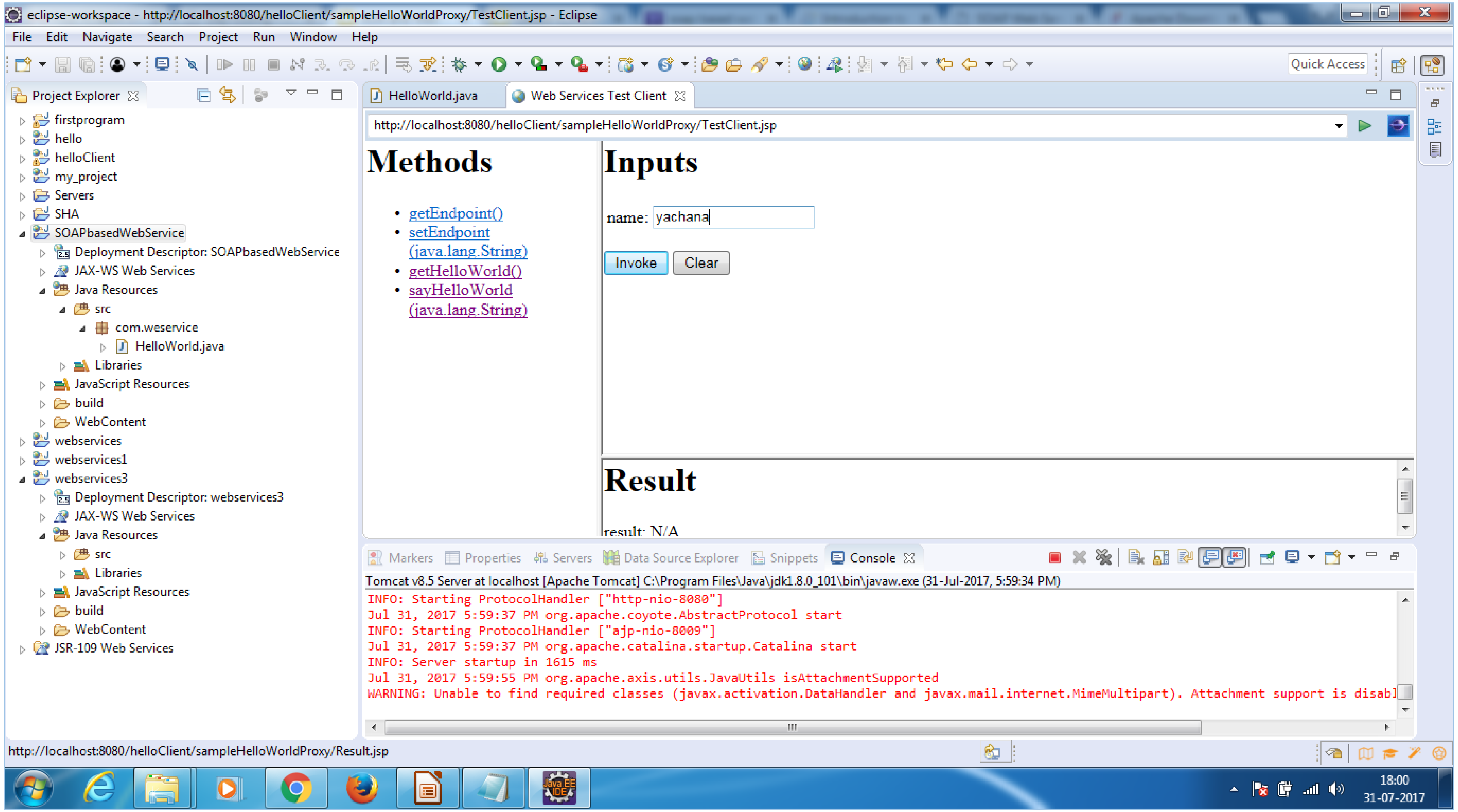
☑ Define custom mapping for namespace to package.

After you click start server, you will be re-directed to this page. Click Finish.

Creating a SOAP based Web Service

- Create a new dynamic web project and name it "SimpleSOAPExample"
- Create a new package named "com.webservice"
- Create a simple Java class named "HelloWorld.java"
- Right click project → new → web service
- Click next
- Click start server
- **Test your Web service**

STEP 7: TEST YOUR WEB SERVICE



eclipse-workspace - http://localhost:8080/helloClient/sampleHelloWorldProxy/TestClient.jsp - Eclipse

```
firstprogram
hello
helloClient
my_project
Servers
SHA
SOAPbasedWebService
    Deployment Descriptor: SOAPbasedWebService
    JAX-WS Web Services
    Java Resources
        src
            com.weservice
                HelloWorld.java
        Libraries
    JavaScript Resources
    build
    WebContent
webservices
webservices1
webservices3
    Deployment Descriptor: webservices3
    JAX-WS Web Services
    Java Resources
        src
        Libraries
    JavaScript Resources
    build
    WebContent
JSR-109 Web Services
```

http://localhost:8080/helloClient/sampleHelloWorldProxy/TestClient.jsp

Methods

- getEndpoint()
- setEndpoint (java.lang.String)
- getHelloWorld()
- sayHelloWorld (java.lang.String)

Inputs

name: yachana

Invoke Clear

Result

result: N/A

Markers Properties Servers Data Source Explorer Snippets Console

```
Tomcat v8.5 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk1.8.0_101\bin\javaw.exe (31-Jul-2017, 5:59:34 PM)
INFO: Starting ProtocolHandler ["http-nio-8080"]
Jul 31, 2017 5:59:37 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-nio-8009"]
Jul 31, 2017 5:59:37 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 1615 ms
Jul 31, 2017 5:59:55 PM org.apache.axis.utils.JavaUtils isAttachmentSupported
WARNING: Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabl
```

http://localhost:8080/helloClient/sampleHelloWorldProxy/Result.jsp

simplilearn

SOA and Web Services

Topic 4—Creating a RESTful Web Service

- Introduction to RESTful Web Services
- RESTful Web Services: Working
- Creating a RESTful Web Service
- SOAP vs. RESTful Web Services

Introduction to RESTful Web Services

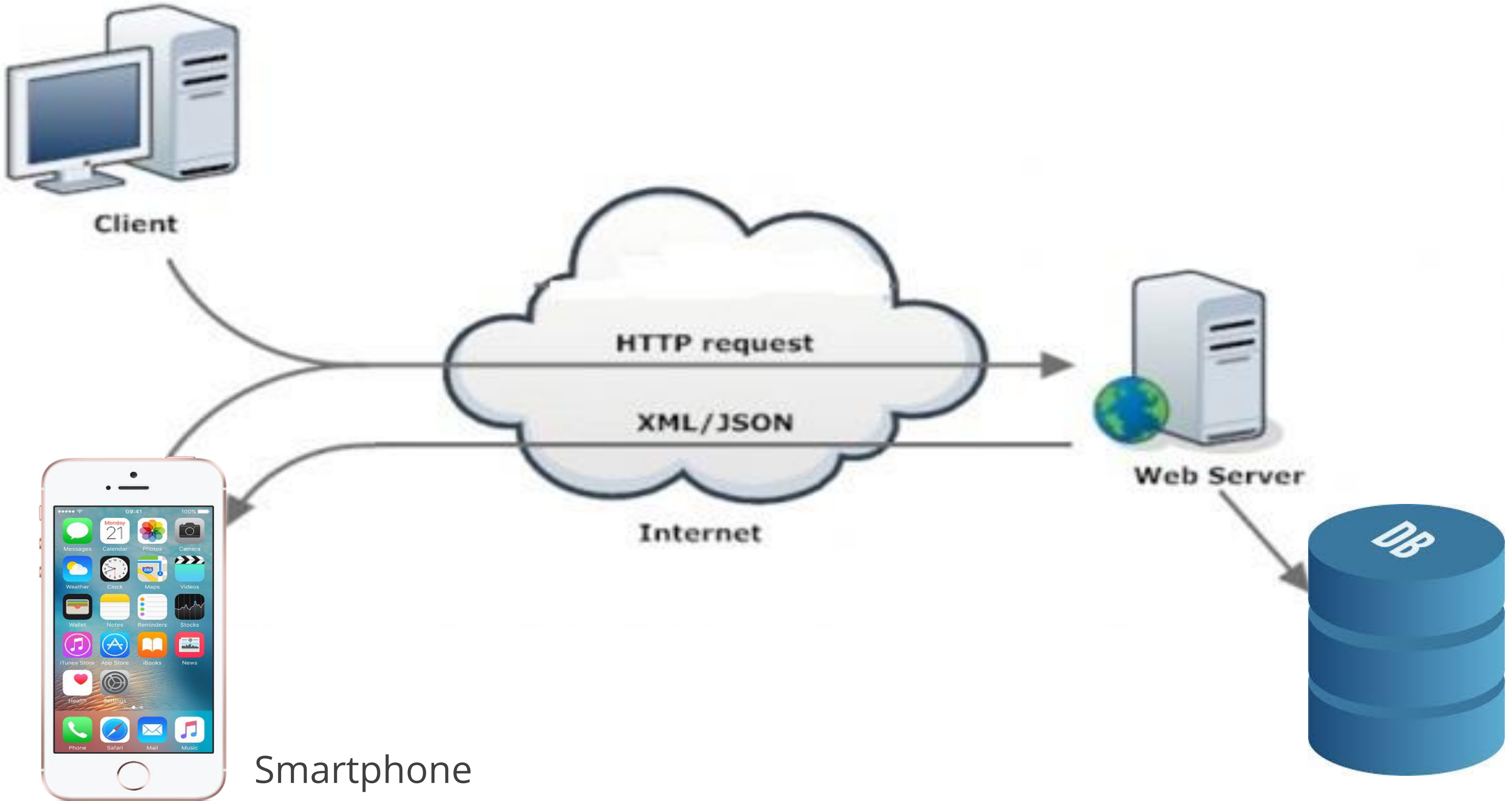
RESTful web services are lightweight, highly scalable, and maintainable Web Services. They are basically based on REST (Representational State Transfer) Architecture.

REST architecture is basically focuses on two things:

1. Resources: Its application functionality, which is represented by a unique URL.
2. Interface: Its HTTP method works as an interface to access the resources.

RESTful Web Services: Working

- A RESTful web service usually defines a URI (Uniform Resource Identifier), which is a service that provides resource representation such as JSON and a set of HTTP Methods.
- There is no need to use XML data interchange format for request and response.
- The REST web services can be return XML, JSON, or even HTML format response.



Creating a RESTful Web Service

FUNCTIONALITIES

Create a web service called UserLog Management with the following functionalities:

HTTP Method	URI	Operation	Operation Type
GET	/UserLogService/users	Get list of users	Read Only
GET	/UserLogService/users/1	Get User with Id 1	Read Only
PUT	/UserLogService/users/2	Insert User with Id 2	Idempotent
POST	/UserLogService/users/2	Update User with Id 2	N/A
DELETE	/UserLogService/users/1	Delete User with Id 1	Idempotent
OPTIONS	/UserLogService/users	List the supported operations in web service	Read Only

Creating a RESTful Web Service

ENVIRONMENT SETUP

1. Set up Java Development Kit (JDK)

2. Set up Eclipse IDE

3. Set up Jersey Framework Libraries

4. Download the latest version of Jersey framework binaries from the following link: https://jersey.java.net/download.html

5. Set up Apache Tomcat

Creating a RESTful Web Service

WRITING RESTful WEB SERVICE WITH JERSEY FRAMEWORK

1. Create a Dynamic Web Project named UserLogManagement using Eclipse IDE.

2. Add Jersey Framework and its dependencies (libraries) in your project.

3. Create UserLogService.java and User.java,UserLogDao.java files under your dynamic project

4. Create a Web XML Configuration file to specify Jersey framework servlet for your application.

5. Export your application as a war file and deploy the same in tomcat.

Creating a RESTful Web Service

Writing User.Java class

Class User has id, name, and designation member variable, and it uses @XmlRootElement and @XmlElement annotations.

```
//When a top level class  is annotated with the @XmlRootElement annotation,
then its value is //represented as XML element in an XML document.
@XmlRootElement(name = "user")
public class User implements Serializable {
   private static final long serialVersionUID = 1L;
   private int id;
   private String name;
   private String profession;
  @XmlElement //Maps a JavaBean property to a XML element derived from
property name.
//setter methods
//getter methods
//constructor
}
```

Creating a RESTful Web Service

Writing UserLogDAO Class

It uses two methods to get user details. First, it checks for the file name User.dat. It it does not exist, it adds one user data to UserList. It then opens User's file and reads data.

```
public List<User> getAllUsers(){
            List<User> userList = null;
               File file = new File("Users.dat");
         if (!file.exists()) {
            User user = new User(1, "John", "Worton");
            userList = new ArrayList<User>();
            userList.add(user);
            saveUserList(userList);
                 else{
            FileInputStream fis = new
FileInputStream(file);
            ObjectInputStream ois = new
ObjectInputStream(fis);
            userList = (List<User>) ois.readObject();
            ois.close();
return userList;
}
```

```
private void saveUserList(List<User>
userList){
      try {
         File file = new
File("Users.dat");
         FileOutputStream fos;
         fos = new
FileOutputStream(file);
         ObjectOutputStream oos = new
ObjectOutputStream(fos);
         oos.writeObject(userList);
        oos.close();
}
```

Creating a RESTful Web Service

Writing UserLogservice Class

```
Path("/UserLogService") //Identifies the URI path that a resource class or class
method will serve //requests for.

public class UserLogService {
   UserDao userDao = new UserDao();
   @HttpMethod(value="GET")  // Indicates that the annotated method responds to
HTTP GET requests
   @Path("/users")
   @Produces(MediaType.APPLICATION_XML) //  is used to specify the MIME
(Multipurpose //Internet Mail Extensions)media types or representations a
resource can produce and send back to the //client.
   public List<User> getUsers(){
      return userDao.getAllUsers();
   }
}
```

Creating a RESTful Web Service

Writing web.xml File

```
web-app>
<servlet>
      <servlet-name>Jersey RESTful Application</servlet-name>
      <servlet-
class>org.glassfish.jersey.servlet.ServletContainer</servletclass>
      <init-param>
         <param-name>jersey.config.server.provider.packages</param-name>
         <param-value>com.tutorialspoint</param-value>
      </init-param>
   </servlet>
   <servlet-mapping>
      <servlet-name>Jersey RESTful Application</servlet-name>
      <url-pattern>/rest/*</url-pattern>
   </servlet-mapping>
</web-app>
```

URL to run the program:
http://localhost:8080/UserLogManagement/rest/UserLogService/users

SOAP vs. RESTful Web Services

SOAP	RESTful
The main advantage of SOAP is that it provides a mechanism for services to describe themselves to clients and to advertise their existence.	In RESTful Web Services, clients have to know what to send and what to expect.
SOAP brings its own protocol and focuses on exposing pieces of application logic (not data) as services.	REST is focused on accessing named resources through a single consistent interface.
SOAP-based reads cannot be cached.	REST has better performance and scalability. REST reads can be cached,
SOAP only permits XML.	REST permits many different data formats

SOA and Web Services

DEMO—RESTful Web Services

Key Takeaways

- ✓ Software evolution begins with understanding the concept of 1 and 0 (i.e.,bits) that gives rise to machine language, followed by assembly language, procedure oriented, object oriented, component oriented, and service oriented.
- ✓ Service Oriented approach follows an architecture, called SOA (Service Oriented Architecture), that focuses on building systems through the use of different Web Services and integrating them to make up the whole system.
- ✓ Web applications enable interaction between an end user and a website. Web services are service-oriented and enable application-to-application communication over the Internet and easy accessibility to heterogeneous applications and devices.
- ✓ RESTful web services are lightweight, highly scalable, and maintainable Web Services. They are basically based on REST (Representational State Transfer) Architecture.

? Quiz

QUIZ

1

UDDI stands for

a. Universal Description Discovery and Integration

b. Unified Description Directory and Integration

c. Uniform Data Dictionary and Identification

d. Uniform Data Dictionary and Integration

QUIZ

1

UDDI stands for

a. Universal Description Discovery and Integration

b. Unified Description Directory and Integration

c. Uniform Data Dictionary and Identification

d. Uniform Data Dictionary and Integration

The correct answer is **a.**

UDDI stands for Universal Description Discovery and Integration

QUIZ

2

Which of the following is correct about WSDL?

a. WSDL is the standard format for describing a web service.

b. WSDL definition describes how a web service can be accessed and what operations it can perform

c. WSDL is a language for describing how to interface with XML-based services

d. All of the above

QUIZ 2

Which of the following is correct about WSDL?

a. WSDL is the standard format for describing a web service.

b. WSDL definition describes how a web service can be accessed and what operations it can perform

c. WSDL is a language for describing how to interface with XML-based services

d. All of the above

The correct answer is **d. All of the above**

WSDL is the standard format for describing a web service. WSDL definition describes how a web service can be accessed and what operations it can perform. WSDL is a language for describing how to interface with XML-based services

Thank You