Caltech | Center for Technology & Management Education

Post Graduate Program in Full Stack Web Development

Full Stack

Caltech | Center for Technology & Management Education

Implement Frameworks the DevOps Way

Web Services in Java

A Day in the Life of a Full Stack Developer

Joe has performed remarkably in the last sprint. Based on his expertise, the company has asked Joe to develop and implement different operations of the project through web services.

In this sprint, using web services, he has to develop a web application in which the system will generate reports of the premium amount to be paid for the type of insurance policy purchased.

In this lesson, we will learn how to solve this real-world scenario to help Joe complete his task effectively and quickly.

Learning Objectives

By the end of this lesson, you will be able to:

- Explain web services
- Differentiate between SOAP and RESTful web service
- Create a RESTful web service
- Explain advantages of microservices over conventional development
- Implement microservices in a Spring boot application

Web Services

Web Services

Web Services

Acts as a medium of communication between the client and a server.

Allows communication over HTTP using XML, SOAP, WSDL, and UDDI.

Exposes the business logic as services over the Internet.

Uses programmable interfaces and Internet protocols that are used to provide ways to find, subscribe,
and invoke those services.

Importance of Web Services

- 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.

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

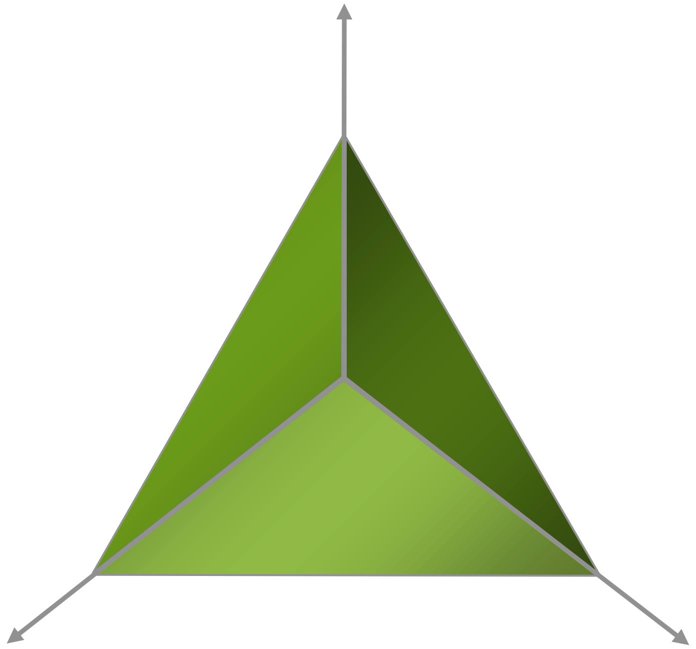
- Web services provide a cross-platform, cross-language solution based on XML messaging.

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

- Web services enable interoperability among heterogeneous applications.

SOAP Web Service

A simple XML-based protocol that lets applications exchange information over HTTP. It is a protocol for accessing a web service.



Platform and operating system independent. Hence, it supports different technologies and programming languages.

Web services built on the SOAP protocol require no customization as SOAP works on the HTTP protocol.

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 focuses on two things:

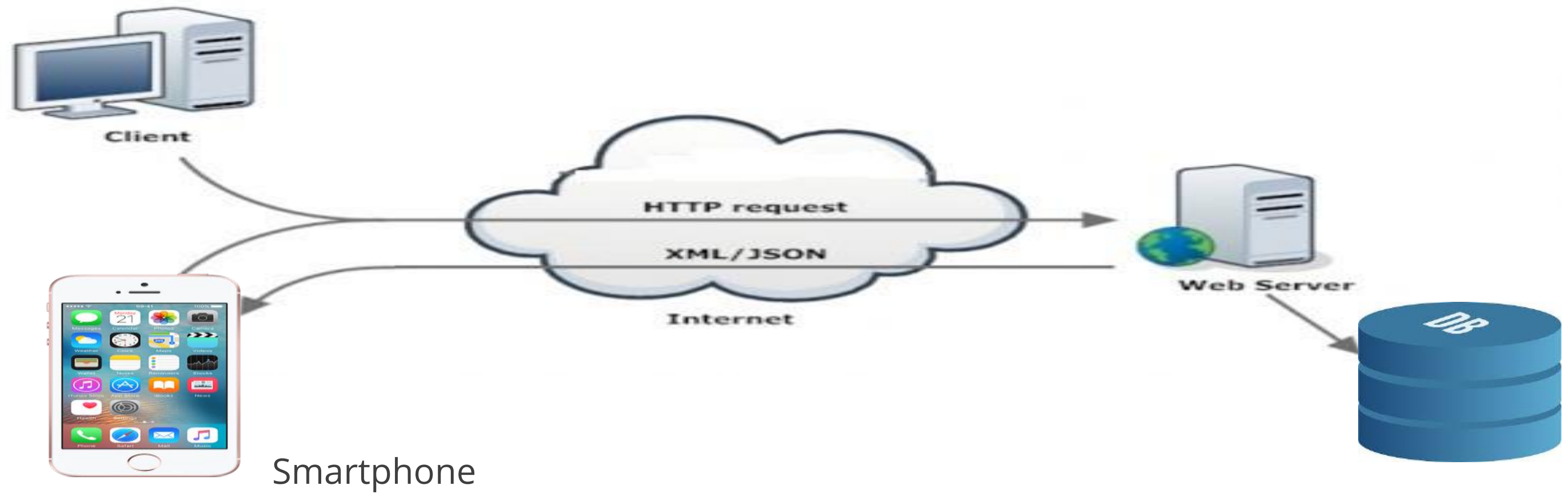
  1. Resources: Its application functionality, which is represented by a unique URL.
  2. Interface: Its HTTP method works as an interface to access the resources.

RESTful Web Services

- 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 return XML, JSON, or even HTML format response.



SOAP vs. RESTful Web Services

SOAP

- The main advantage of SOAP is that it provides a mechanism for services to describe themselves to clients and to advertise their existence.
- SOAP brings its own protocol and focuses on exposing pieces of application logic (not data) as services.
- SOAP-based reads cannot be cached.
- SOAP only permits XML.

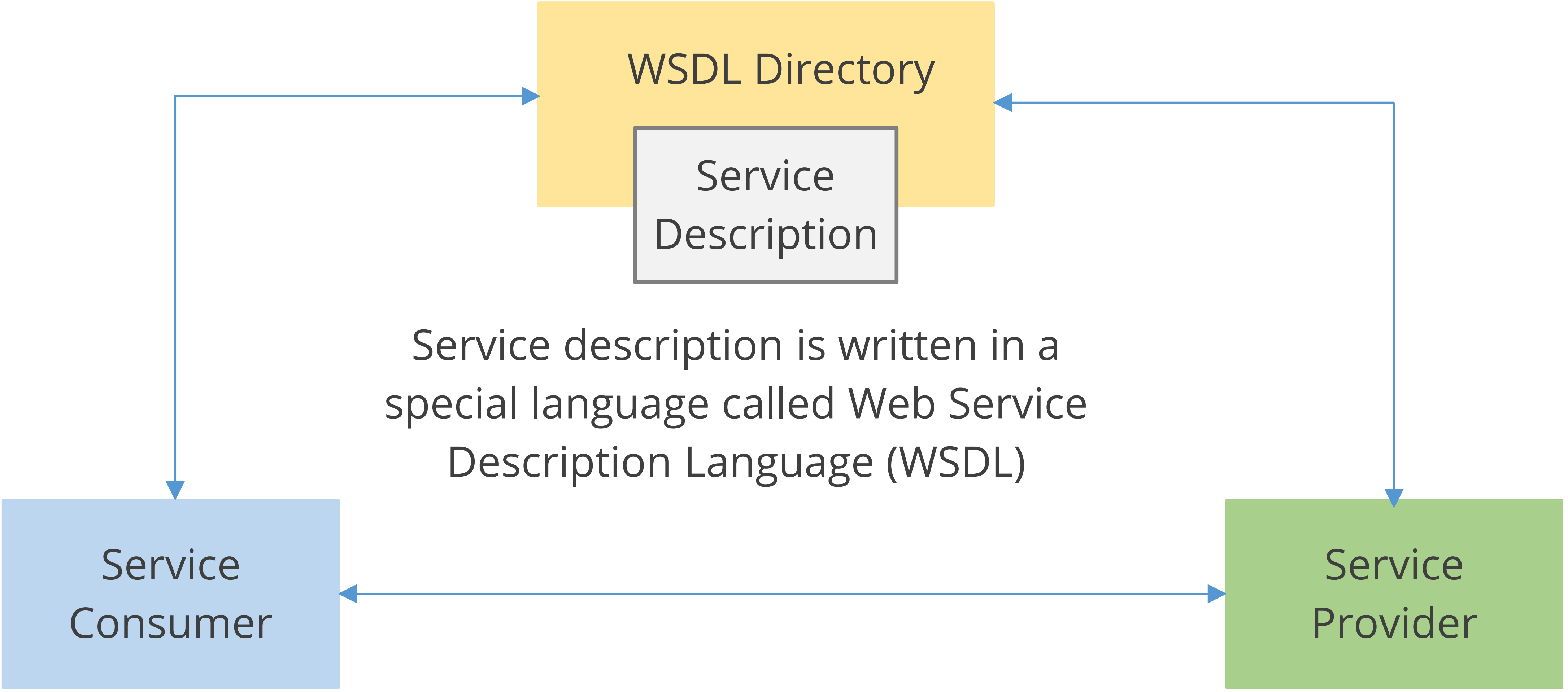
RESTful

- In RESTful Web Services, clients have to know what to send and what to expect.
- REST is focused on accessing named resources through a single consistent interface.
- REST has better performance and scalability. REST reads can be cached.
- REST permits many different data formats.

WSDL

- A WSDL (Web Services Description Language) is a document in XML format.
- It describes the interface of a web service. It provides information on how to access a web service and which operations it will perform.
- It is an integral part of UDDI (Universal Description, Discovery, and Integration).
- 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.

WSDL



WSDL Directory

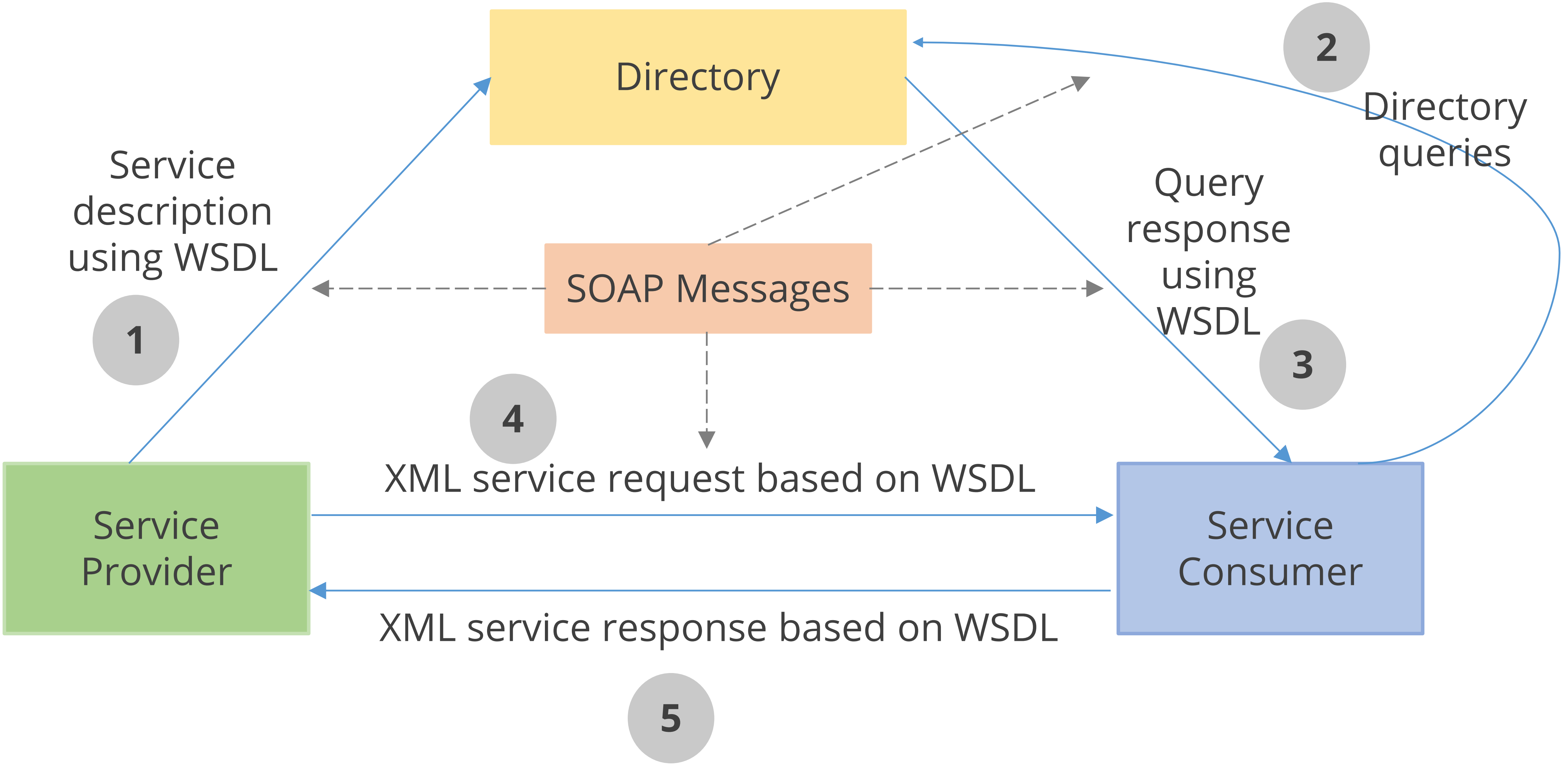
Service Description

Service description is written in a special language called Web Service Description Language (WSDL)

Service Consumer

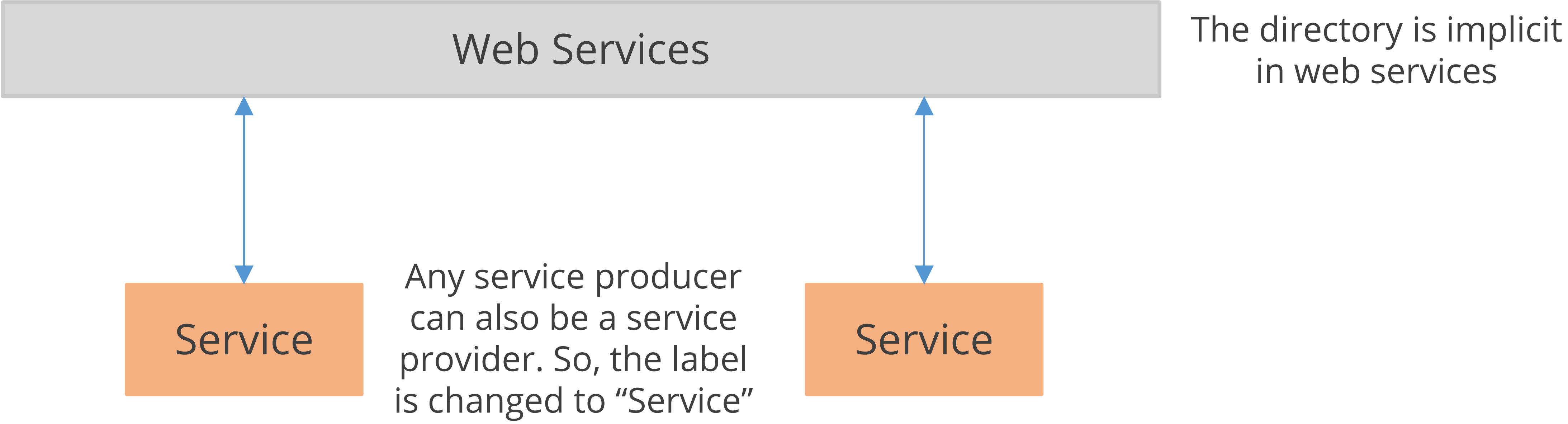
Service Provider

WSDL Flow


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

UDDI

- 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, business processes, and its preferred methods for conducting business.



The directory is implicit in web services

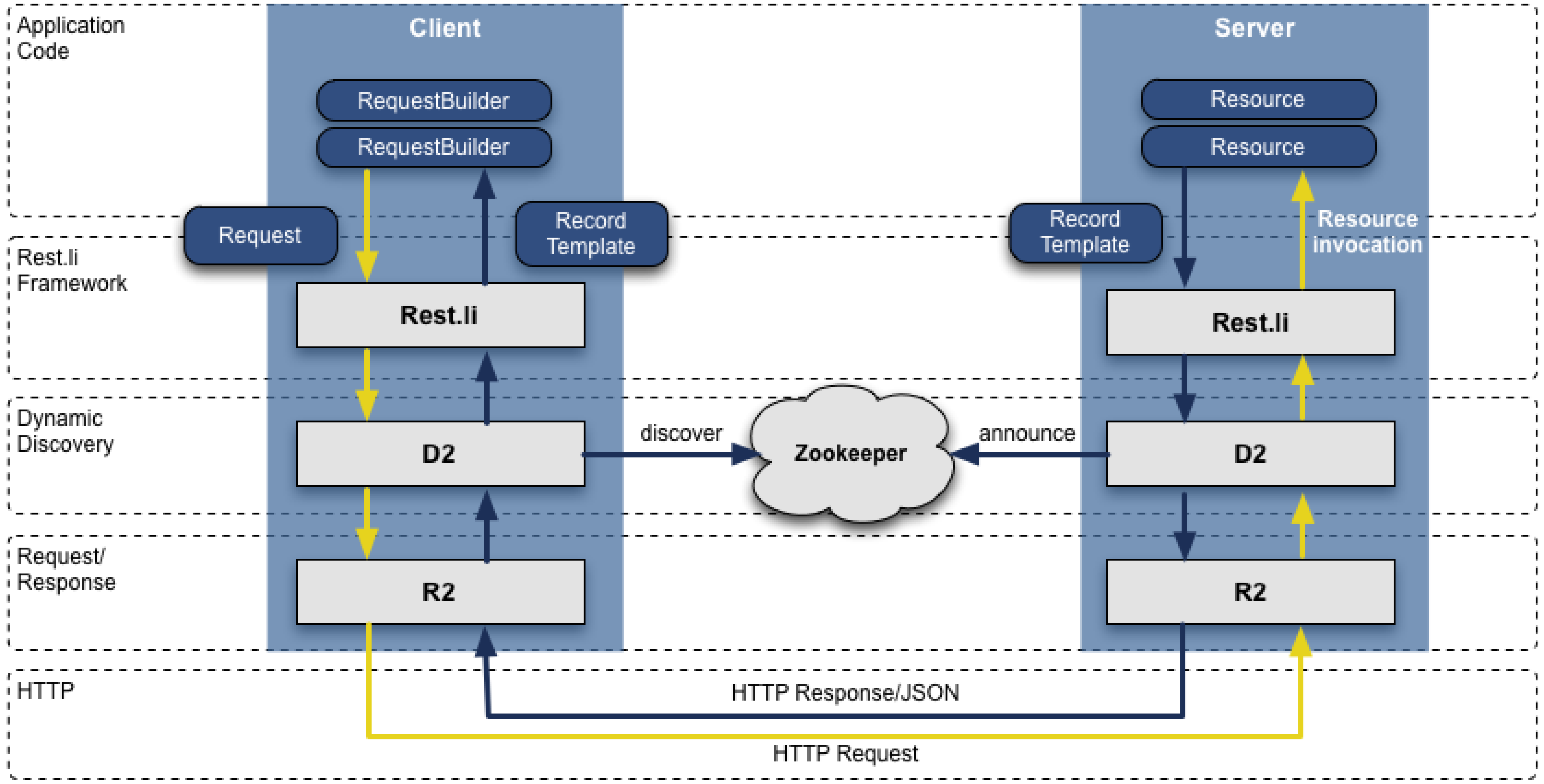
Any service producer can also be a service provider. So, the label is changed to “Service”

RESTful with Spring Boot

Advantages of RESTful

- Highly scalable
- Flexible and portable
- Independent due to the separation between client and server
- Simple to build and is adaptable
- Client does not require routing information
- Clients can use a listener interface for notifications

RESTful Architecture



Source: https://engineering.linkedin.com/architecture/restli-restful-service-architecture-scale

Limitations of SOAP

- Does not have a built-in messaging system
- Does not have any security like SOAP

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

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. If 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) //  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

Assisted Practice

Assisted Practice: Build RESTful with Spring Boot

Duration: 60 min.

Problem Statement:

You are given a project to demonstrate RESTful with Spring Boot.

Assisted Practice: Guidelines

Steps to demonstrate RESTful with spring boot:

1. Create a new project using Spring Initializer

2. Select the required project dependencies

3. Create an Entity and a Repository class

4. Create a Service and a Controller class

5. Execute the project as *Spring Boot App*

6 Push the code to GitHub repositories

Microservices with Spring Boot

Microservice

Microservices can be understood using the three characteristics:

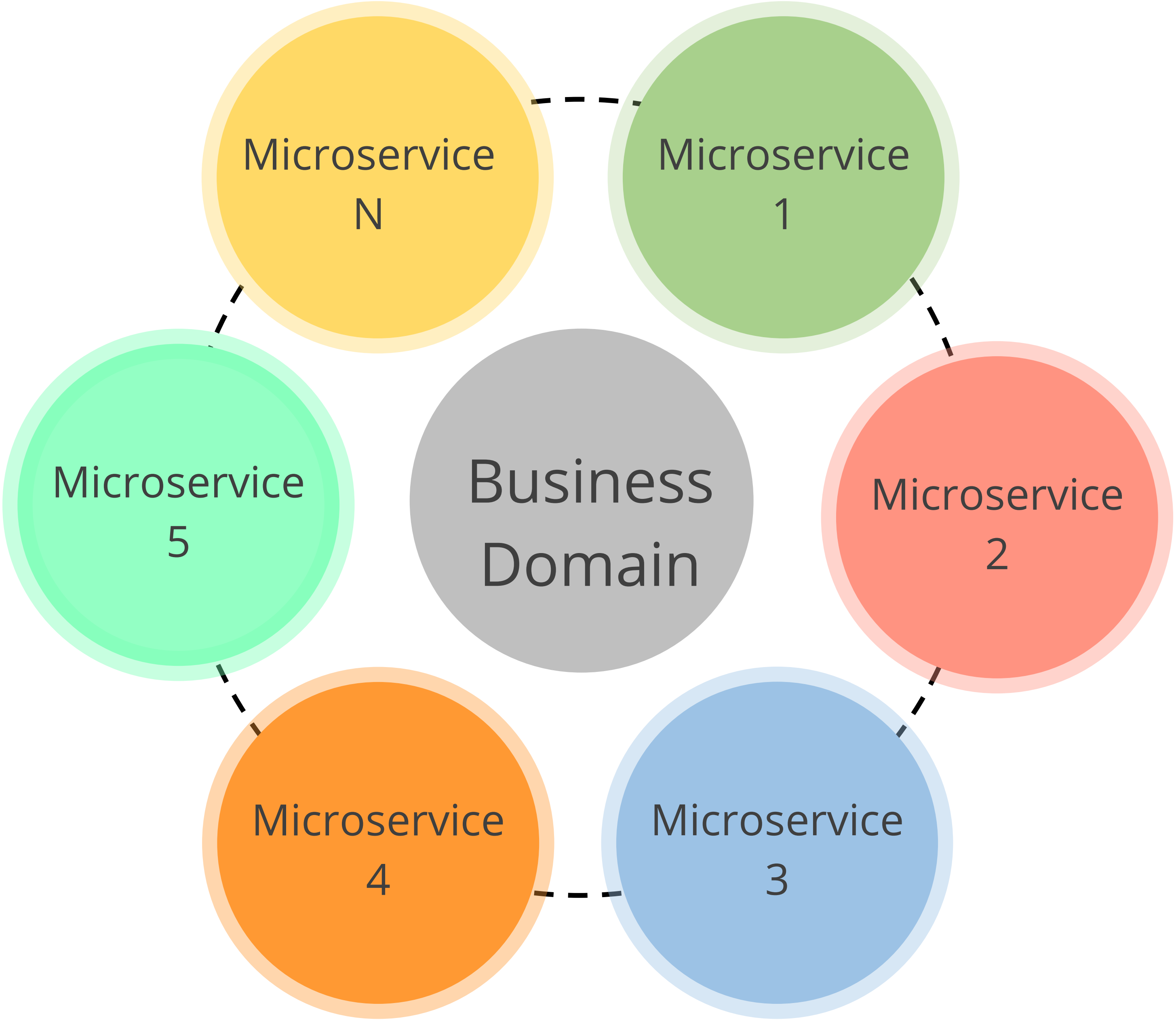
REST
It is built majorly around RESTful web services.

Bounded
It is small, but well deployable unit of component.

Dynamic Scaling
It is effective while scaling up the application or team as it is cloud enabled.

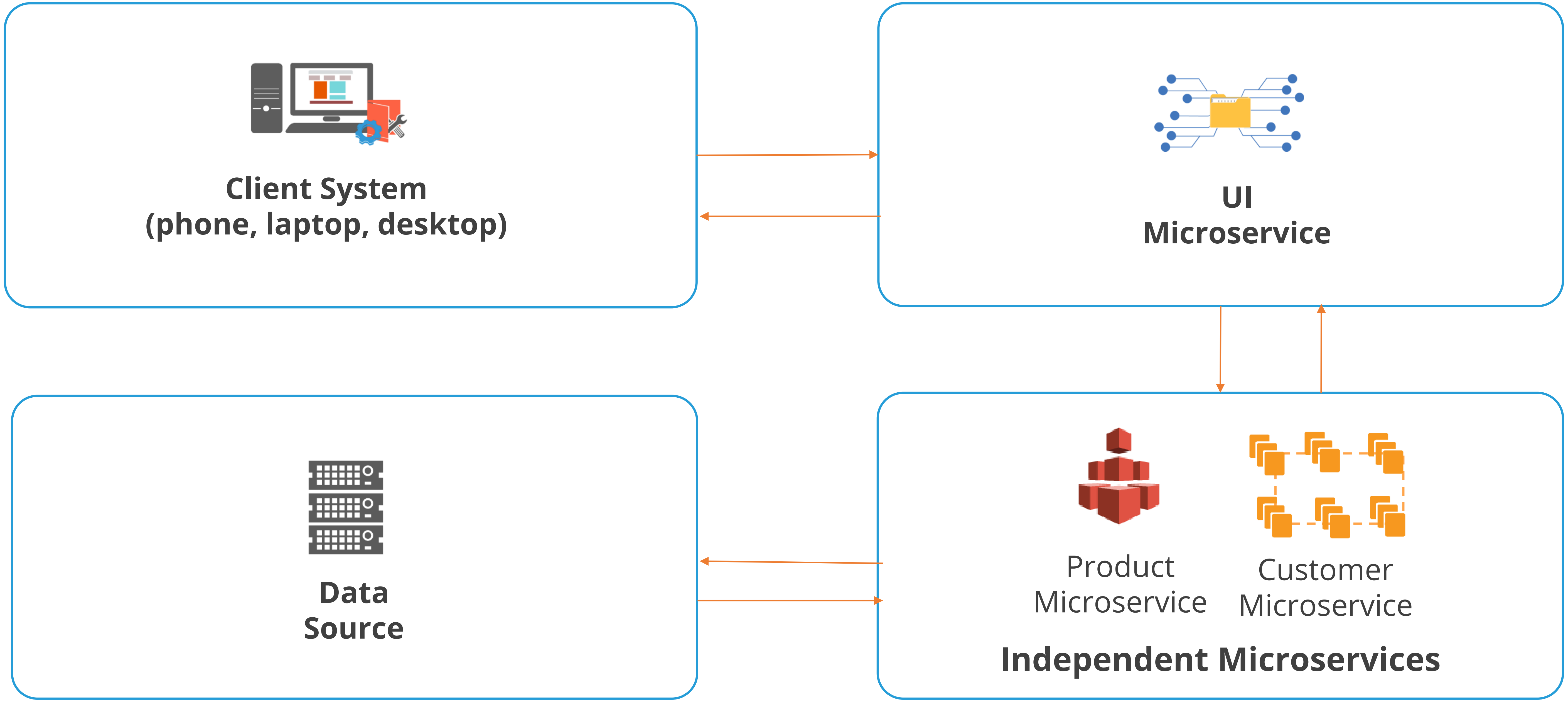
Microservice

- The diagram represents the natural behavior of a microservice.
- It works in collaboration with the application and the complete business logic is divided into smaller chunks of deployable units.
- Every single functionality like sign-in, sign-up, payment can be a microservice working independently.
- This helps the developer to use the same skeleton of a functionality multiple times in the same application helping in code reusability and central request handler.



Microservice Architecture

Below is the architecture that depicts the way microservices works:



Advantages of Microservice

Independent Development

Each microservice is developed serving a specific functionality and thus can be used independently across the application.

Independent Deployment

Each service can also be deployed individually, because they are independent in nature.

Fault Isolation

Independent nature of microservices also make it easy for developers to debug the faults because even if one service fails, there is very less impact on the application.

Multiple Stacks

A developer has the freedom to choose any technology stack to create a microservice.

Scaling Up

One can easily scale up the services as per the need of the application or business.

Microservice Frameworks

Spring boot

Spring boot is one of the most widely used framework as it provides all round coverage over coding.

Dropwizard

It works with Java libraries and creates a light-weight package which helps centralize the functionality.

Restlet

It helps Java developers to develop web APIs based on REST framework.

Spark

It helps in creating web applications using Java and kotlin.

Why Spring Boot for Microservices?

Spring boot as a framework for microservices is winning the market with its various services for developing microservices in less coding compared to its competitors. There of the major service are given below:

Central Configuration

Spring boot uses **Spring Cloud** as a service that provides **@EnableConfigServer** for central web services for all microservices.

Dependency Injection

@Component annotation is the heart of any spring boot application that injects all the dependencies.

Service Discovery

Eureka REST service is provided by spring boot that becomes the central point for all your microservices registry.

Assisted Practice

Assisted Practice: Microservices Communication with Spring Boot

Duration: 60 min.

Problem Statement:

Create a project to demonstrate microservices with Spring Boot.

Assisted Practice: Guidelines

Steps to demonstrate microservices with spring boot:

1. Create a new project using Spring Initializer and add all the dependencies
2. Create an Entity and a Repository class
3. Create a Controller and a Response class
4. Execute the project as *Spring Boot App*
5. Create a new project and add an Entity, Repository, Service, Controller, and Response class
6. Set the port number and execute the project as *Spring Boot App*
7. Push the code to GitHub repositories

Key Takeaways

- Web service is a medium to allow communication between the client and the server.
- SOAP is a simple XML-based protocol to let applications exchange information over HTTP.
- REST is highly scalable and portable.
- Microservices break the conventional way of developing a web application with minimal coding and central configuration.

Lesson-End Project

Duration: 60 min.

Implement Spring Security with Authentication

Problem Statement:

As a developer, build Authentication Provider in Spring Security.

Before the next Class

You should know:

- The fundamentals of Spring Boot
- Dependency Injection
- Basics of Spring MVC
- Testing using JUnit 5
- Web Services in Java

Phase-End Project

Duration: 240 min.

Make an E-commerce Website for Sporty Shoes

Problem Statement:

As a Full Stack Developer, complete the features of the application by planning the development and pushing the source code to the GitHub repository.

Background of the Project Statement

Sporty Shoes is a company that manufactures and sells sports shoes. They have a walk-in store, and now, they wish to launch their ecommerce portal sportyshoes.com.

You Are Asked to Do

You're asked to develop a prototype of the application. It will then be presented to the relevant stakeholders for budget approval. Your manager has set up a meeting where you're asked to do the following:

- Present the specification document which has the product's capabilities, appearance, and user interactions.
- Set up Git and GitHub account to store and track your enhancements of the prototype
- Explain the Java concepts used in the project
- Discuss the generic features of the product:
  - There will be an admin to manage the website. An administrator login will be required to access the admin page. The admin should be able to change his password if required
  - He should be able to:
    - Manage the products in the store including categorizing them
    - Browse the list of users who have signed up and be able to search users
    - See purchase reports filtered by date and category

You Must Use the Following

IDE: Eclipse or IntelliJ

Programming language: Java

Specification document

Git and GitHub