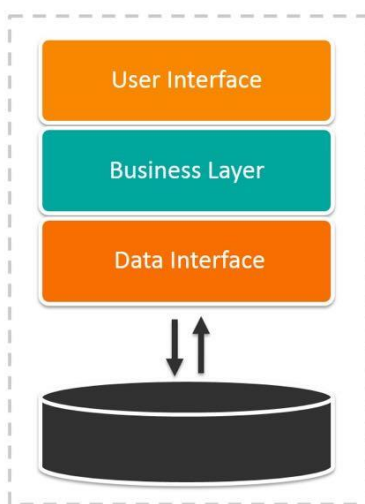
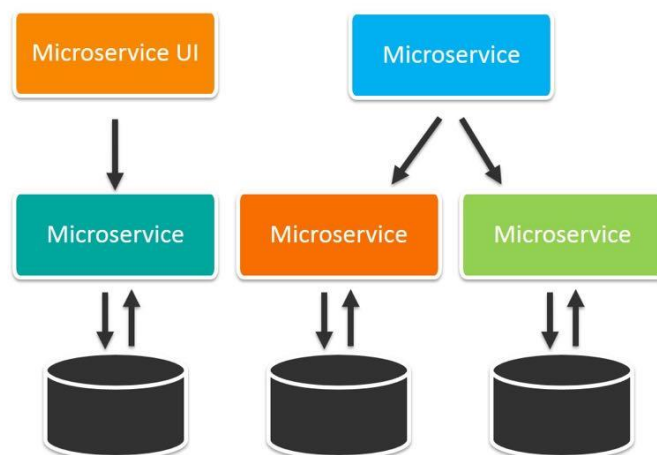


Micro Service : Micro means very small service.

Monolithic Architecture



Microservices Architecture



In **Monolithic web service** contains more than one modules like login modules(login controller, login service, login dao, login bean class. customer modules (customer controller, customer service, customer dao or repository and customer bean), account, transaction etc.

Dev1	Dev2	Dev3	Dev4
Login module	Account module	customer,	Feedback module

We need to combine all those modules and need create single **jar or war**. Then that jar or war file we need to deploy or run on actual server ie production server environment.

CI And CD

Limitation of Monolithic web service.

1. All modules develop using same language.

2. One module depends upon another modules.
3. If we need to any changed in any one of the modules we need to do the changes and re-deploy whole application.

Login Module -→ Dev1

Using spring boot

LoginController

Login

LoginService

LoginRepository

Database -→ mysql (mydb1)

They deploy the application.

8181

They communicate using rest api

Dashboard module dev2

Controller, service, repository and bean --→ database mysql (mydb2)

Then deploy the application

8282

If we want to implements application using Java as micro service

Spring framework provide two modules

Spring cloud

Spring micro service

Spring Framework provided one of the web server ie Eureka Server. This server help us to deploy more than one micro service develop using spring framework.

Multi project application

1st project –(Eureka Server)→ contains web starter and **Eureka Server** starter which help to run the Eureka server.

The screenshot shows the Spring Initializr web application interface. The 'Project' section has 'Maven' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '3.1.8' selected. The 'Project Metadata' section has 'Artifact' set to 'Spring-Eureka-Server', 'Name' set to 'Spring-Eureka-Server', 'Description' set to 'Demo project for Spring Boot', 'Package name' set to 'com', 'Packaging' set to 'jar', and 'Java' set to '17'. The 'Dependencies' section has 'Spring Web' and 'Eureka Server' selected. The 'Eureka Server' dependency is highlighted with a red box. The 'Spring Web' dependency is also highlighted with a red box. The 'Eureka Server' dependency is listed as 'spring-cloud-netflix Eureka Server'.

In spring boot whenever we add web starter by default tomcat run on default port number 8080. Eureka server also run on default port number 8080.

If we do changes in eureka server project as 8761. The by default all micro service deploy on same port number.

But whenever we are going to deploy more than one micro service each service check default port number of eureka ie **8761** but if eureka server running on 8080 port number we need to mention in each micro service project eureka server running on 8080.

```
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false
```

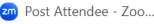
Eureka server project is a server application. This project is not providing any service it help us to deploy more than one micro service project.


```
@EnableEurekaServer
```

This annotation is use to enable eureka server features.

2nd project –(Eureka Client) → contains web starter and **Eureka Client** starter which help to deploy that project in Eureka Server.

Micro service project. (this project only provide simple rest full message as welcome to spring boot micro service).

orted From Fire... 



Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ **Maven**

Language

☒ **Java** ☐ Kotlin ☐ Groovy

Spring Boot

☐ 3.3.0 (SNAPSHOT) ☐ 3.3.0 (M1) ☐ 3.2.3 (SNAPSHOT) ☐ 3.2.2 ☐ 3.1.9 (SNAPSHOT) ☒ **3.1.8**

Project Metadata

Group

com.example

Artifact

First-Micro-Service

Name

First-Micro-Service

Description

Demo project for Spring Boot

Package name

com

Packaging

☒ **Jar** ☐ War

Java

☐ 21 ☒ **17**

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web **WEB**


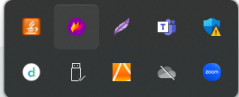
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Eureka Discovery Client **SPRING CLOUD DISCOVERY**

A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

Spring Boot DevTools **DEVELOPER TOOLS**

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

Port number 8080 (default port number).

3rd project –(Eureka Client) → contains web starter and **Eureka Client** starter which help to deploy that project in Eureka Server.

Micro service project. (this project only provide simple rest full message as welcome to spring boot micro service).

The screenshot shows the Spring Initializr web application interface. The browser address bar displays 'start.spring.io'. The page title is 'spring initializr'. The configuration is as follows:

- Project:** ☒ Maven
- Language:** ☒ Java
- Spring Boot:** ☒ 3.1.9 (SNAPSHOT)
- Project Metadata:**
 - Group: com.example
 - Artifact: Second-Micro-Service
 - Name: Second-Micro-Service
 - Description: Demo project for Spring Boot
 - Package name: com
 - Packaging: ☒ Jar
 - Java: ☒ 17
- Dependencies:**
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Eureka Discovery Client** (SPRING CLOUD DISCOVERY): A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.
 - Spring Boot DevTools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Port number 8080 (default port number).

4th project –(Eureka Client) → contains web starter and Eureka Client starter , jpa starter, mysql connector and devtools etc which help to deploy that project in Eureka Server. This project interact with database.

Account-micro-service

The screenshot shows the Spring Initializr web application interface for an 'Account-micro-service'. The configuration is as follows:

- Project:** ☒ Maven
- Language:** ☒ Java
- Spring Boot:** ☒ 3.1.9 (SNAPSHOT)
- Project Metadata:**
 - Group: com.example
 - Artifact: account-micro-service
 - Name: account-micro-service
 - Description: Demo project for Spring Boot
 - Package name: com
 - Packaging: ☒ Jar
 - Java: ☒ 17
- Dependencies:**
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Eureka Discovery Client** (SPRING CLOUD DISCOVERY): A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.
 - Spring Boot DevTools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
 - Spring Data JPA** (JPA): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
 - MySQL Driver** (JDBC): MySQL JDBC driver.

Port 8282

@RequestMapping Vs @GetMapping, @PostMapping, @DeleteMapping,
@PutMapping

@RequestMapping annotation we can use on class level to check base path as well as method level with method attribute get, post, put and delete etc. this is generic annotation

@GetMapping : this annotation we need to use on method level and it is consider as Get method.