**PG FSD End Points and Communication Aug cohort**

Day 1 :06/01/2024

MVC using Servlet and JSP

Course 3

We will develop MVC application using Spring MVC or Spring boot

Web Service : Giving the service for web application when two application running using different technologies. Java req

Amazon or e commerce application -🡪 XML/JSON paypal java

Using Servlet and JSP .net paytm asp.net

Php Phone pay php

Python Google pay python

Credit card / debit cart or net banking

Java is platform independent but language dependent programming language.

Xml : eXtensible markup language

Json : java script object notation

Spring Framework and Spring boot, Web Service and Junit testing

After develop our own rest api those rest api we calling in angular application.

We Develop web application using JEE technologies

Servlet and JSP

MVC

Model bean, service, dao, resource etc.

View : html or jsp css and Javascript

Controller : Servlet

Web container part of web server responsible to create the object of servlet and jsp. In ems model layer not up to that mark

To improve model layer EJB came in picture.

Enterprise Java bean : to develop simple ejb application we need to do more configuration. We need ejb container part of application server. EJB is complex.

For EJB Web application ie Servlet and JSP are client. Using EJB we develop distributed application.

JSP Servlet

JSP Servlet EJB payment gateway

JSP Servlet

JEE

Servlet -🡪 controller layer

JSP -🡪 view layer

EJB -🡪 model layer

Framework : Framework provide set of api which internally connected to each other to perform specific task. Framework internally follow standard rules. Framework provide implementation of design pattern. Design pattern is known as best practice or solution of repeating problem. If we develop any application with help of framework 70 to 80% task taken care framework. But framework is not final product it like a template or protocol.

Angular angular is framework

React with Redux react is library

Angular and React as well as Vue JS which help to develop

SPA application

Angular use MVC and React use V in MVC.

React provide virtual dom and angular doesn’t provide.

Java api, python api : it may be classes or interfaces or function or modules etc.

Rest api

Java frameworks

Struts : Struts is an open source web framework provided by Apache. Struts internally follow MVC architecture design pattern. It provided lot of api to improve view layer, controller layer and model layer. But among that controller layer is very strong. So struts is known as controller centric framework.

JSF JSF is an open source web framework provided by Oracle. JSF internally follow MVC architecture design pattern. It provided lot of api to improve view layer, controller layer and model layer. But among that view layer is very strong. So JSF is known as view centric framework.

Hibernate : Hibernate is an open source framework provided by JBoss. Which help to connect the database. Hibernate is use to improve dao layer. Hibernate use ORM (Object Relation mapping )

Spring framework : spring is an open source light weighted layer or onion architecture framework.

Spring framework provided lot of module which help to improve all type of application or layers.

Spring framework modules.

1. Spring core
2. Spring context
3. Spring dao
4. Spring Web or Spring MVC : it internally follow MVC Architecture design pattern. It provided lot of api to improve view, controller and model layer. Spring MVC is known as model centric framework.

Spring Framework Vs EJB

1. Spring Rest
2. Spring security
3. Spring cloud
4. Spring micro service
5. Spring ORM
6. Spring AOP
7. Spring boot

Etc

IOC : Inversion of Control . it is a concept. It is also known as programming design pattern.

According to IOC in place creating or maintaining any resources explicitly allow to create by container. If container create it will maintain properly. You only need to pull from container use it and leave it.

DI : Dependency Injection : DI is an implementation of IOC.

Spring framework support two type of DI

1. Constructor base
2. Setter base
3. Property base

In Spring Framework we can achieve DI using

1. XML base Configuration
2. Annotation base configuration

Spring container part of jar file which is responsible to create object for normal java bean class. that class in spring framework is known as POJO. Plain Old Java Object. That class not to extends or implements any pre defined class except Object class.

Web Container part of web server it will create the object of class only that type is type of servlet or jsp.

Struts : in Struts web container will create the object that class it class type of struts.

BeanFactory is core interface provided by spring core and context module which help to pull the object from xml.

Singleton design pattern : only one memory need to create.

By default spring container create singleton object for pojo class.

If you want to each time whenever we pull from container new memory then we need to use protype.

Constructor base DI Vs setter base DI

Using constructor base Setter base DI internally use empty

Di we need achieve fully DI. Constructor Base DI. Using setter we override the value

We can’t achieve partial DI in setter order doesn’t matter as well as we

It can be empty or parameter can achieve partial DI.

Empty default value.

Parameter passed value.

Parameterized DI order matter.

Day 7 :07/02/2024

Autowired

Spring framework by default Inject or DI for primitive value implicitly. Like 0,0.0,null, false etc.

But if class contains complex or user defined object it doesn’t inject implicitly we need to inject explicitly using property ref or constructor ref attribute.

Using auto wired features we can achieve di for complex property implicitly rather explicitly using property ref or constructor ref.

byType : if we use auto wired byType then spring container search or scan the xml file if that type bean definition which is part of class present in xml file it automatically inject.

In byType autowired we need only bean definition of that type. If more than one bean

Definition present then we need to use byName

byName: byName id name and reference name must be match(part of pojo class) .

@Component : This annotation we need to use on pojo class. This annotation is equal to

<bean class=”com.Address”></bean>

By default id name is address in using camel naming rules.

If class contains one word then then id name must be in lowercase. If class contains more than one word then id name must be first word lower case and second word onward first letter upper case.

Example

Address then id name is address

EmployeeInfo id name is employeeInfo

CustomerDetails id name is customerDetails

@Component

@Autowired this annotation we need to use on complex property.

@value this annotation is use to set initial value for those property.

By default @Component annotation is disable.

We need to enable using

1. Using xml file
2. Using configuration class with few more annotation.

We need to create class and on that class we need to use the annotation

As

@Configuration this annotation is equal to beans.xml file

@ComponentScan This annotation is use to enable @Compoennt annotation.

@Bean : This annotation we need to write on a method which is responsible to create the object.

ApplicationContext : it is an interface which internally extends BeanFactory interface which provided set of method which help to pull the object from a container.

By default spring container do di with singleton. Using annotation if need to prototype.

@scope annotation : this annotation is use to set the scope for bean.

SpringWithDataSourceFeatures : this example is use to improve the model layer.

Spring Core

Spring Context

Spring jdbc

Mysql connector

Spring JDBC module provided pre defined class ie DriverManagerDataSource. This class provided data connection. This class we can configure in xml or configuration.

create table employee(id int primary key,name varchar(10), salary float);

@Repository : this annotation we need to use on dao layer class. which contains database coding.

@Service : this annotation we need to use on service layer. Which contains business logic.

Day 3 :13/01/2024

Spring DAO : This module provided one of the pre defined API ie JdbcTemplate. JdbcTemplate internally wrap core jdbc code and provided more functionality to improve dao layer using jdbc.

Create maven project

Then in pom.xml file add properties tag to change the version of java as 11.

Then add 4 dependencies.

1. Mysql connector
2. Spring core
3. Spring context
4. Spring jdb

Spring ORM : Object Relation Mapping

Spring ORM doesn’t provide any orm tool. It help to integrate existing ORM tools like

Hibernate or JPA or iBaties etc.

Limitation of JDBC or JdbcTemplate

1. Using JDBC we can’t store Java object as well as we can retrieve java objects.

In DAO layer we need to convert java object into SQL query format and vice-versa.

1. JDBC use SQL language. SQL is Database dependent language.
2. JDBC throw checked exception and exception hierarchy is database dependent.
3. JDBC doesn’t support relationship ie is as and has a.

ORM : Object Relation mapping

Programming language Database

class Employee { Employee -🡪 Table (Relation )

Id,name,salary

}

Employee emp = new Employee();

Mapping

Employee == EMPLOYEE

Id = ID PK

Name = NAME

Salary = SALARY with data type and column contains PK

In JDBC not mandatory table must be map to java bean class.

In JDBC not mandatory table must be contains PK.

In ORM one record is equal to one object. so table must be contains PK.

ORM is a concept.

The implementation of ORM is JPA as well as Hibernate.

JPA is a specification as well as it provide the implementation. JPA is a technology part of EJB. Evern we can say JPA is type of EJB.

Hibernate is a framework part of Jboss. Hibernate is a implementation of JPA.

Hibernate CRUD Operation

JPA CRUD Operation

Spring framework with ORM tool ie Hibernate or JPA.

Hibernate CRUD Operation

We need table ie Employee -🡪 Id(PK), Name, Salary-🡪 in database.

Create maven project as HibernateCrudOperation

Then using properties tag change java version and add mysql and hibernate core dependencies.

In Hibernate we use hibernate.cfg.xml file. (Configuration file) Which contains database details like driver name, url, username, password, dialect class(to convert java object to sql), mapping class(java bean class with few annotation) and hibernate properties.

Now we need to create the mapping class ie JavaBean class with few annotation.

In ORM java bean class is known as entity class.

@Entity class java bean class

@Id annotation on that variable which map to pk.

@Table annotation : if java bean class name and table name is different.

@column : if variable and column name is different then we can use @column annotation

Day 4 :14/01/2024

In ORM Java Bean class is known as entity class.

In Hibernate we can provide configuration details using xml file or using java classes.

hibernate.cfg.xml

Configuration : it is a pre defined class provided by hibernate which help to load the configuration file.

Once file loaded successfully. Then we need to create SessionFactory interface reference.

It is type of interface provided by hibernate which is equal to Connection in jdbc.

Which help to create the Session interface reference.

Session is an interface which provided set of methods which help to do CRUD operation on entity class. it is like a Statement or PreparedStatement in JDBC.

TCL : Transactional Control Language

It is a type of SQL sub language which contains two commands commit or rollback.

If we do any DML Operation like Insert, Delete, and Update (more than one query on single table or multiple table). If all query executed successfully you can say commit or else rollback.

Update account set amount = amount -500 where accno=123;

Update account set amount = amount +500 where accno=567;

Commit

Rollback.

By default if we do any operation using JDBC it is auto commit.

con.setAutocommit(false); by default true. If you want to achieve transaction concept using JDBC we need to setautocommit false.

pstmt.executeUpdate(“DML Operation”);

con.commit() or con.rollback();

if we do any DML operation using ORM tool like hibernate or Jpa by default they are not auto commit. In ORM tool we need to use Transaction concept whenever we are doing DML operation.

Hibernate provided their own query language ie HQL (Hibernate Query Language)

SQL VS HQL

SQL is database dependent. It retrieve record as string format.

Select \* from employee. \* all columns and employee is table name. SQL is not case sensitive.

Select \* from employee where id = 100;

Select \* from employee where salary > 15000; id and salary are column name.

Select name from employee retrieve only one column

Select salary from employee retrieve only one column

Select name ,salary from employee retrieve more than one column

HQL is database independent. It retrieve record as object.

Select emp from Employee emp; Employee is entity class name. it is case sensitive. emp is object. using emp we are retrieving all variables.

Select emp from Employee emp where emp.id = 100; emp object and id is variable name

Select emp from Employee emp where emp.salary > 15000 emp object and salary is variable name

Select emp.name from Employee emp retrieve only name variable values.

Select emp.salary from Employee emp retrieve only salary variable values

Select emp.name,emp.salary from Empoyee emp retrieve name and salary variable values

Relationship

One to many relationship

Trainer and student

Product and Orders

Projects and Employee

Database side

One trainer can handle multiple students.

If we want to create table in database : 1st option or

Using hibernate property we can create table with relationship 2nd option

Trainer

PK

TId TName tech

1 Raj Java

2 Ravi Python

3 Ramesh Angular

Student

PK FK

SID SName age tsid

100 Seeta 21 1

101 Veeta 22 1

102 Leeta 23 2

103 Meeta 24 null

create table trainer(tid int primary key,tname varchar(20), tech varchar(20));

create table student(sid int primary key,sname varchar(20), age int, tsid int,

foreign key(tsid) references trainer(tid));

Please create maven project as HibernateRelationship.

Then add properties to change the version of java.

Then add two dependencies mysql connector and hibernate core

And add previous project hibernate.cfg.xml file

Hibernate JPA

Both use same annotation

hibernate.cfg.xml file persistence.xml

which contains database details.

SessionFactory EntityManagerFactory

Session EntityManager

Transaction EntityTransaction

HQL JPQL

save persist

delete remove

update merge

get find

Day 5 :20/01/2024

JPA : Java Persistence API

JPA is also one of implementation of ORM.

In JPA persistence unit name part of persisten.xml file which hold the database details as well as entity class information.

ORM doesn’t provide features of IOC and DI.

Spring framework doesn’t provide any ORM tools. It provide to integrate with existing ORM tool like Jpa or Hibernate.

Spring ORM : with hibernate or JPA.

Spring JPA Data.

Spring with JPA

Create maven project then change the java version.

Then add dependencies.

Spring core and context IOC and DI @Component, @Service @Repository and @Autowired

Spring jdbc DriverManagerDataSource

Mysql connector to connect mysql database.

Jpa or hibernate

Two dependencies only hibernate core

Spring orm help to integrate with jpa or hibernate.

*LocalContainerEntityManagerFactoryBean*

It is a pre defined class provided by spring orm module which help to integrate with Jpa orm tools.

This class help us to do DI in DAO for EntityManagerFactory.

Spring core

Spring context using xml and annotation

Spring jdbc

Hibernate

Jpa

Spring orm with jpa

Day 6 :21/01/2024

Spring MVC (model view controller).

Before Spring MVC controller we are/were using servlet as a controller.

Limitation of servlet

1. To make normal java class as servlet. We need implements or extends type of servlet.
2. We need to override pre defined method provided by HttpServlet or GenericServlet.
3. Inside one servlet we can’t write more than one doGet or doPost method.
4. doGet and doPost always take two parameter is request and response.

In Spring MVC we need to create normal java class with @Controller annotation.

Like

We can write user defined method with @RequestMapping annotation with few attribute to behave that method like doGet or doPost. Method name can any thing but return type can be ModelAndView(part of spring mvc) or String. @RequestMapping annotation help us to map the request.

@Controller this class behave like Servlet.

public class MyController {

@RequestMapping(value=”hello”) by default method consider as Get

public ModelAndView sayHello() {

// coding..

ModelAndView mav = new ModelAndView();

mav.setViewName(“display.jsp”); // like Requestdispatcher ie forward.

return mav;

}

@RequestMapping(value=”info”,method=RequestMethod.GET)

public ModelAndView sayHi() {

// coding..

ModelAndView mav = new ModelAndView();

mav.setViewName(“info.jsp”); // like Requestdispatcher ie forward.

return mav;

}

}

By default @Controller annotation not enable. We need to enable this annotation in spring configuration file. In spring core we need to load spring configuration file. But in Spring MVC container is going to load the xml file automatically.

Spring MVC internally follow front controller design pattern.

Front controller is a type of servlet which is responsible to keep the track about all controller flow.

FrontController---🡪 keep the track account all controller it is a type of servlet.

Normal Servlet program

module module module

LoginController AccountController ManagerController

LogingService AccountService ManagerService

Login Account Manager

LoginDao AccountDao ManagerDao

Spring internally follow front controller design pattern. Spring MVC provided pre defined class is DispatcherServlet. It is a type of servlet which behave like a front controller.

This class we need to configure in web.xml file or normal java class.

We will develop Simple Spring MVC project.

1. We need to create dynamic web project with version 2.5 or 3.1 but web.xml file required..
2. Convert this project to maven.
3. Create normal java class with @Controller annotation and write more than one method with @RequestMapping annotation and return type must be ModelAndView
4. Then in web.xml file please add front controller ie Dispatcherservlet.

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

DispatcherServlet is a pre defined class provided by Spring MVC which behave like front controller. This class we need to configure in web.xml file. Whenever any request pass from view ie html or jsp. DispatcherServlet receive the request and search spring configuration file start with name as servlet name part of servlet-name tag followed by -servlet.xml file

File name must be dispatcher-serlvet.xml file . this file must be present with web.xml file (inside WEB-INF folder).

Abc-servlet.xml

Demo-servlet.xml file



Index.jsp -----🡪web.xml file (DispatcherServlet class as FrontController) receive the request -🡪 then this class load spring configuration file with name as dispatcher-servlet.xml file. This file enable @Controller annotation using component scan tag. Then flow move to controller class. controller class can contains more than one method with @RequestMapping annotation. This annotation contains value attribute to map the request. By default method is get consider. Inside this method we can do coding or we can call service layer then using ModelAndView class reference we can redirect to specific view.

Spring MVC with core JDBC

jdbcTemplate

hibernate

jpa

View as JSP

Controller as Spring Controller

Spring MVC with JPA for Product Management System

Add product, delete product, update and view product.

Dynamic web project with web.xml file (2.5 or 3.x version)

Then convert this project to maven

Then dependencies

Spring mvc @controller, @service, @repository @autowired

Mysql connector

Spring Jdbc

Spring orm

Jpa two dependencies.

In web.xml file we need to configure DispatcherServlet as front controller

dispathcher-serlvlet.xml file we need to configure database details and di for EntityManagerFactory.

Product table

create table product(pid int primary key auto\_increment,pname varchar(30), price float, url blob);

Day 7 :27/01/2024

Spring MVC with Database

1. Normal JDBC with data source
2. JdbcTemplate
3. Hibernate
4. JPA

In Angular

class MyService {

Constructor(public http:HttpClient) {} // DI

}

Spring boot

Spring boot is a type of spring module which help to do the bootstrap for spring framework or spring modules. This module is not use for specific purpose.

Spring boot = all spring modules – no xml file (beans.xml or dispatcher-servler.xml or any other xml , no hibernate.cfg.xml or persistenc.xml ) + few annotation+ in built embedded server ie tomcat or jetty server.

Spring boot itself is core java project or standalone application or project which help to create any type of projects. In spring boot we don’t want external tomcat server.

Spring boot components.

1. Spring boot starter : spring boot provided lot of different types of starter which help to download more than one jar file depending upon type of project we are creating.

Spring 1.x version

Core

Context

Mvc

Jackson which help to convert java object to json

Then external tomcat etc.

Starter is a combination of more than one jar file which help to develop type of projects.

1. Spring boot auto configuration : without spring boot we need to configure database related code , security, testing in xml. But in spring boot no xml file. Base upon type of starter it automatically ready to provide us IOC and DI for those resources.

Spring boot provided one annotation ie

@SpringBootApplication = @Configuration + @ComponentScan + @AutoConfiguration

Spring boot with web starter

1. No external tomcat required.
2. No web.xml file required.
3. No DispatcherServlet frontcontroller need to add in xml file
4. No dispatcher-servlet.xml file required.

We need to create normal java class and on that class write @SpringBootApplication annotation. In main method use pre defined class as SpringApplication.run(MainClassName,args)

Spring boot with Maven tool : pom.xml file is part maven build tool.

Spring boot with Gradle tool : no xml file not even pom.xml file.

Spring boot application with manual configuration for starter.

In spring if we need any configuration like database details, port number etc. we need to create normal file with name as application.properties or application.yml.

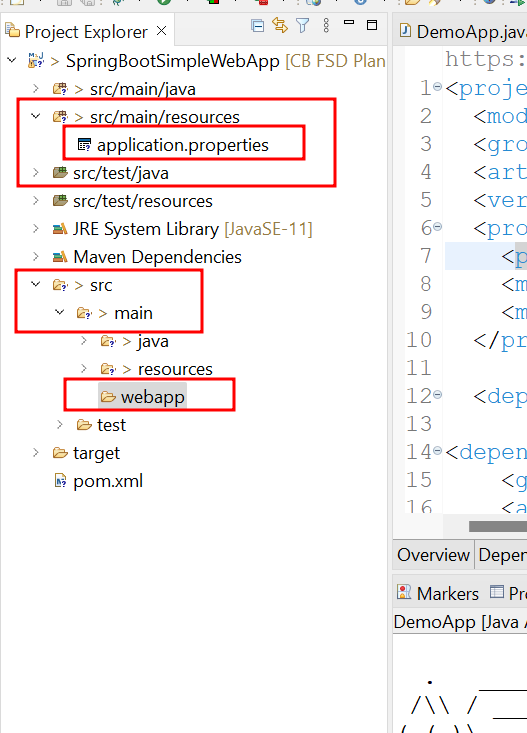
Spring boot with View as

1. If we want View as JSP. Spring boot doesn’t provide jsp starter
2. Spring boot provided view engine as Thymeleaf . But they provided thymeleaf starter

Dynamic html.

1. If we use spring boot with rest api then our view as Angular.

If we need view as JSP in spring boot we need to add jasper dependencies.



In spring boot all jsp pages must be part of webapp folder. And that folder must be insider src/main.

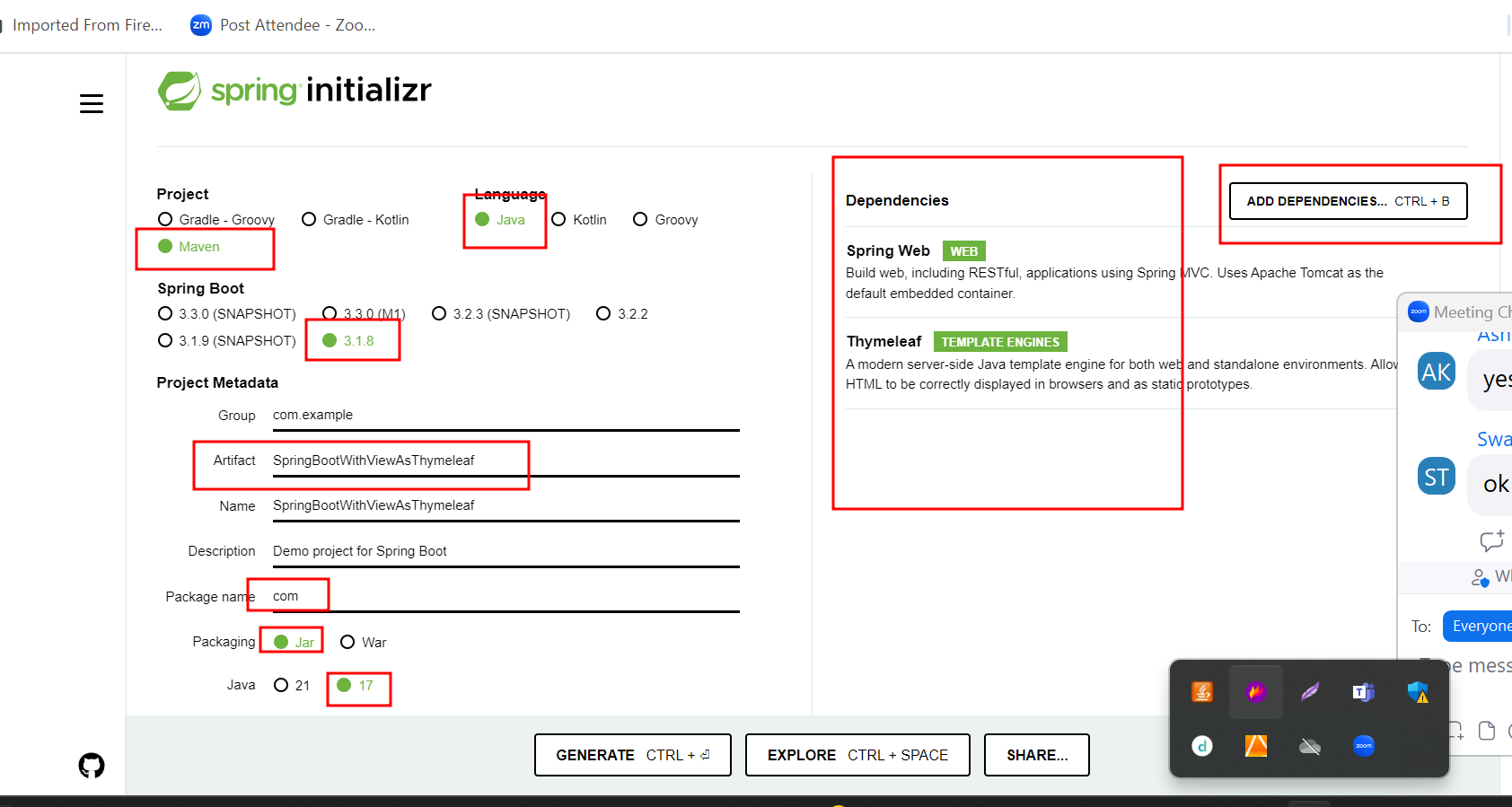
Now we need to provide view resolver details in application.properties files.

Spring boot with web application Ie web starter and view as thymeleaf(Dynamic HTML Page).

We will spring boot project using official web site provided by spring framework ie spring initializer

<https://start.spring.io/>

Spring boot 3.x min java version 17 required.



Whenever we create any spring boot project with spring initializer it automatically add testing starter.

If we add manually more than one starter. Those starter version can different. So it can make some problem.

So spring boot use parent tag part of pom.xml file to maintain common version between more than one starter.

Spring boot 3.x version minimum java version must be 17.

If we want to use view as thymeleaf we need to create html pages with thymeleaf plugin and those pages must be part of templates folder insider resources.

With normal HTML form or JSP form using submit button we are passing the value from view to controller.

But thymeleaf provide a features as bind the form value with bean objects like Model Driven form or reactive form in angular.

Spring framework or spring boot provided Model class which help to share the data between controller to view like ModelAndView.

Day 8 :27/01/2024

Spring boot with MVC

View as Thymeleaf, or JSP

Connecting database ie Mysql using Spring JPA Data.

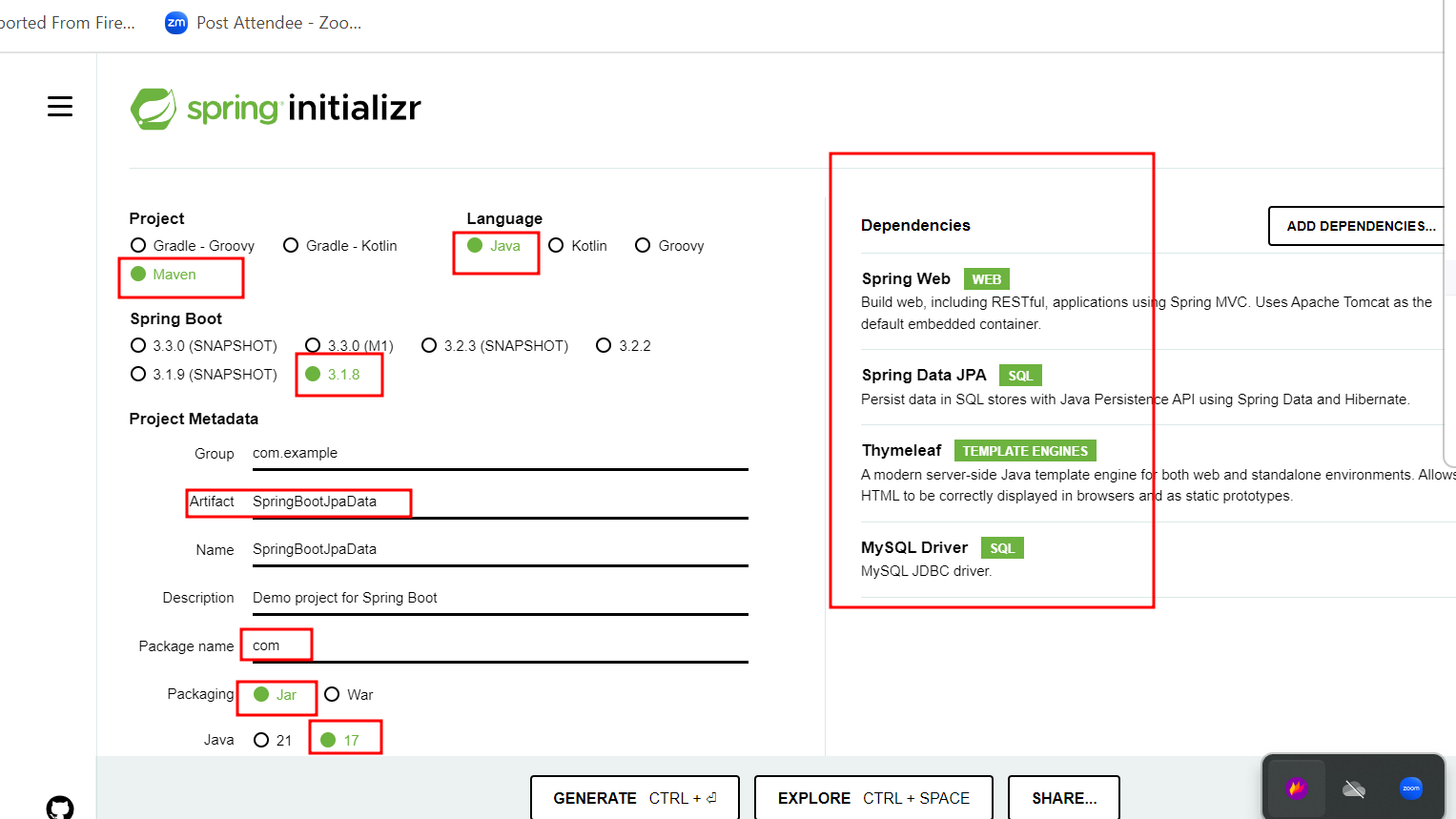
We need to create the project using spring initializer website.

Web starter

Jpa starter

Mysql connector dependencies.

Thymeleaf starter



In DAO

1. DI for DataSource
2. DI for JdbcTemplate
3. DI for SessionFactory
4. DI for EntityManagerFactory

Spring JPA Data is base upon JPA orm. Spring JPA Data provide two interfaces

1. CrudRepository : Super interface ie Generic interface
2. JpaRepository : Sub interface specific for Jpa.

These two interface base upon Java 8 features. Which internally provide implementation of all Standard DAO methods.

In Dao layer we need to create normal interface and that interfaces extends CrudRepository or JpaRepository with Generics as first parameter entity class and second parameter data type of that column which column hold primary key.

@Repository

public interface ProductRepository extends JpaRepository<Product,Integer> {

}

@SpringBootApplication

This annotation enable @Controller, @Service, @Repository annotation only if all the classes part of same package or sub package of same package.

Day 9 :03/02/2024

Web Service Giving the service for web application when more than one application running using different technologies.

Java or Spring boot

Amazon

On - line shopping --🡪

Payment Paytm spring boot

Paypal asp.net

Gpay php

Phonepay python

Net banking

Credit / debit card

Java is platform independent programming language but language dependent. Java can communicate with only java technologies.

Java XML/JSON python, php, asp.net

XML : eXtensible Markup language

JSON : Java Script object notation

Web Service

Two types

SOAP Based web service : Simple object access protocol. SOAP web service is base upon SOA. Service Oriented architecture. In SOAP web service we need to consume and produce the data only in the form XML.

Rest Full web service : Representation state Transfer. Exposing our resources servlet, jsp or spring controller as web service. Rest Full web service allow use to expose the data in any format on demand. Like xml, json, plain text, html etc.

In Spring boot @Controller annotation replace by @RestController.

If my controller is normal controller with @Controller annotation then view can be html, or jsp or thymeleaf. So View technologies tightly coupled with Java technologies.

If my controller is rest controller with @Restcontroller annotation then view can be any other technologies. Asp.net, php, python, rest client

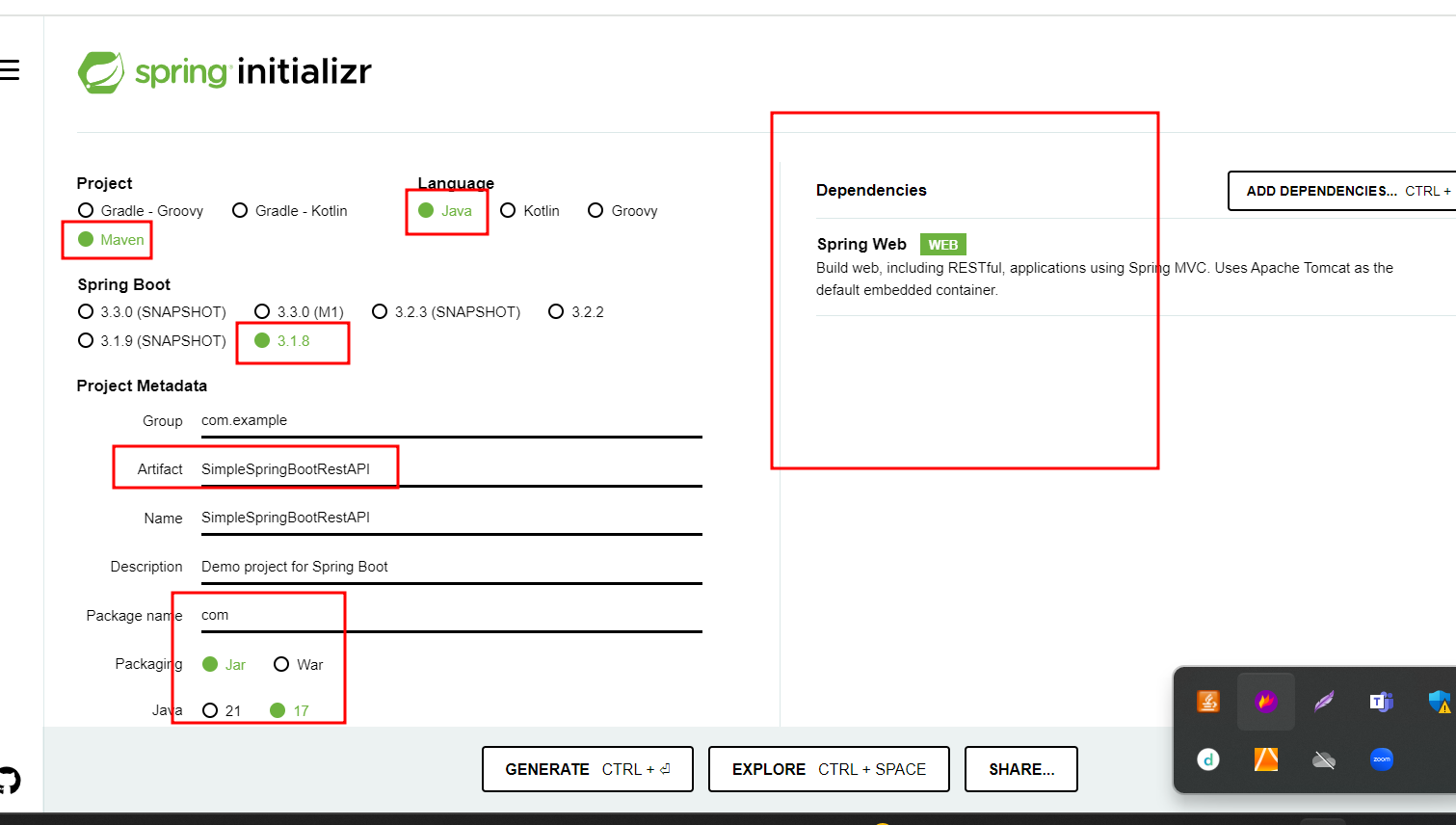
Now view for as Angular or react or plain java script

Rest api (JSON)

Frontend backend database

Angular Spring boot with rest controller mysql

Simple Rest API using Spring boot



Using Servlet also we can create rest api with help of JAX\_RS using Jersey open source tool.

We can create spring MVC with rest api. We need add more than one dependencies manually.

With help of spring boot with only one starter web starter.

Rest API

1. Get Method : Get the resources : like employee, customer, manager, order, product etc.

Retrieve query or select query

1. Get data in plain text format
2. Get data in html format
3. Get data in xml format
4. Get employee object in json format
5. Get employees object in json format.

Using Get method we can pass the data to rest api using two ways

1. Query param
   1. Passing single value : URL?key=value
   2. Passing multiple value : URL?key=value&key=value&key=value

By default all html or jsp form with get method internally they use query param technique.

1. Path param
   1. Passing single value : URL/value1
   2. Passing multiple value : URL/value1/value2/value3
2. Post method : create the resource : Like insert query with database.

We can store employee, customer, order, product or account details.

1. Delete method : delete the resource : like delete query

Delete employee information using employee id, delete account details using accno etc. this information we can receive using query param or path param.

1. Put method or patch method : update the resource like update query

Update employee salary using id, update amount in account using accno etc.

Put is use to update whole object

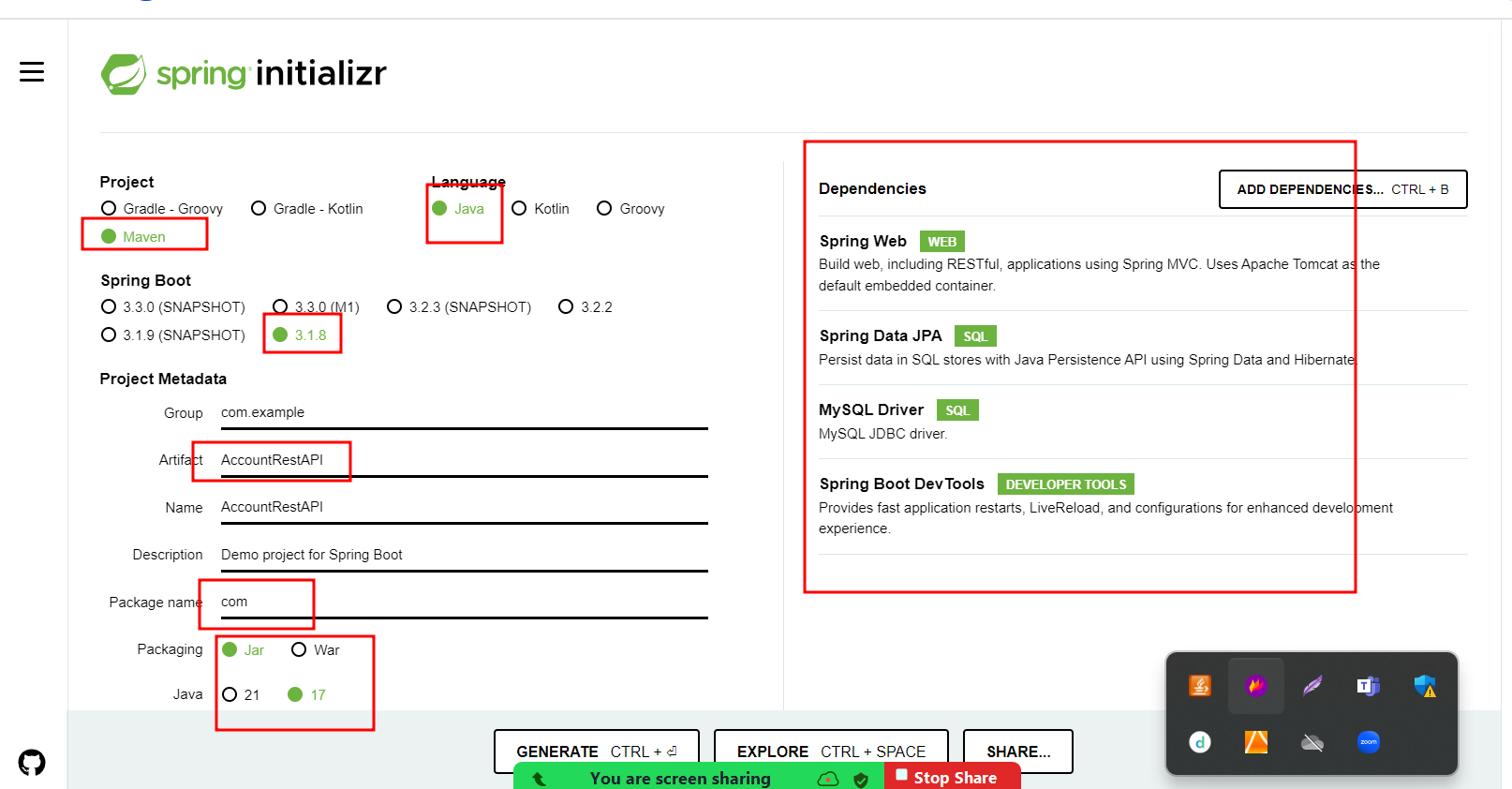
Path is use to update partial object.

Employee table id, name, salary we want update salary or name using id then we can use patch method.

If we want to update name and salary using id then we can put method.

Creating spring boot rest api with database using spring data

AccountRestAPI



Using browser we can test only get method

We can’t test post, put , patch , delete methods.

Post man client plugin

Day 10 :04/02/2024

In post man client we need to create workspace.

Inside workspace we can create more than one collection.

Collection is like container or directory which hold more than one rest api url.

Put and patch

Put use to update whole object.

Patch use to update partial object.

@RequestMapping : This annotation we can use on class level as well as method level with method option as get, post, put, patch, delete etc.

@GetMapping

@PostMapping

@DeleteMapping

@PutMapping

@PatchMapping

All annotation we can use only on method level with @GetMapping is like get method so no method attribute present inside that annotation.

@RestController

@RequestMapping(“account”)

public class AccountController {

// http://localhost:8080/account/findAll

@GetMapping(value=”findAll”) or @RequestMapping(value=”findAll”,method=RequestMethod.GET)

public List<Account> findAll() {

return accountService.findAllAccount();

}

}

Micro Service : small rest full web service.

Monolithic and Micro Service

In monolithic rest full web service

Account Module -🡪 AccountController, Account, AccountService, AccountRepository 🡪database

Login Module -🡪 LoginCotnroller, Login , LoginService, LoginRepository -🡪database

Customer module 🡪Customer Controller, Customer, Customer Service, Customer Repository -🡪database.

We need to integrate all these modules. And create single jar or war file. Then deploy this application on actual server ie production server environment.

* + 1. We need to depends upon each team to finish their task.
    2. If any error generate in any one of module we need to re-check, re compile, recreate and re deploy.
    3. If any changes required in any one of the module we need to re deploy whole application once again after changes done.
    4. All modules develop using only one language with one database.

Micro service.

In micro service architecture we can create small modules using language or different language using same database or different database. And we can deploy alone that module using different port number.

Login module

All those module are interacting with each other using Rest API call.



Angular application

sessionStorage.setItem(“accno”,accno);

AccountService -🡪 Micro Service Project Java 8181

Spring boot

Accno, name, emaild, amount

Rest API produce and consume

Paytm micro service Php

Rest API produce and consume

GPay Service --🡪 Micro Service project 8282

Spring boot

Gpayid, emailed

Spring framework provided two modules

1. Spring cloud
2. Spring micro service

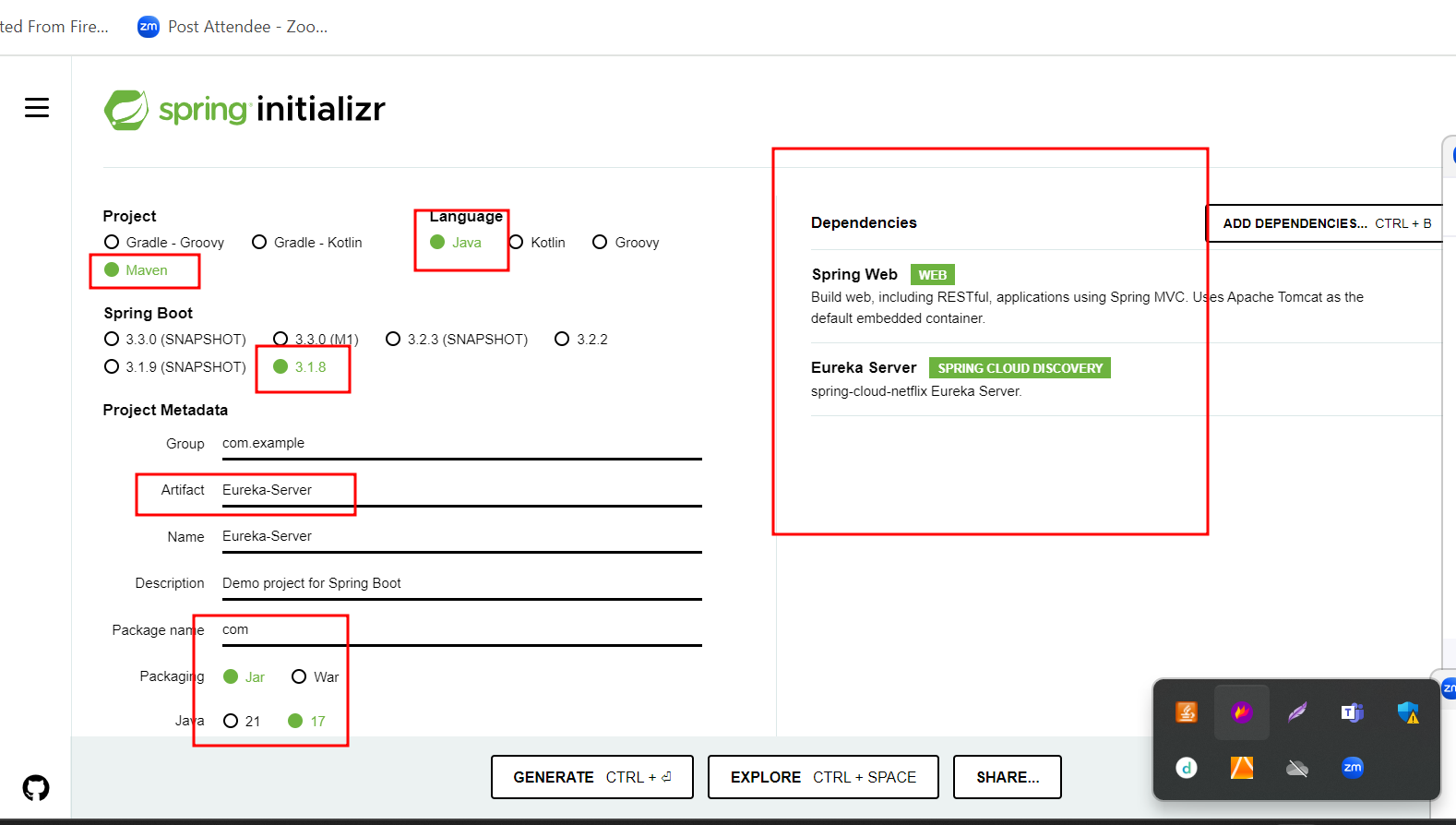
Spring framework provided pre defined service ie web service Eureka server.

Eureka Server is an open source server provided by spring framework which help to deploy more than one micro service project which develop using spring. Eureka Server is use to maintain more than one micro service health.

1. 1st Project : Eureka Server

Starter -🡪 Web starter

eureka server



Whenever we run micro service ie eureka client application by default it will search in port number 8761. Default port number for eureka server is 8761.

But tomcat run on 8080 port number.

application.properties

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

main class

**package com;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;**

**@SpringBootApplication**

**@EnableEurekaServer**

**public class EurekaServerApplication {**

**public static void main(String[] args) {**

**SpringApplication.run(EurekaServerApplication.class, args);**

**System.err.println("Eureka Server up on port number 8761");**

**}**

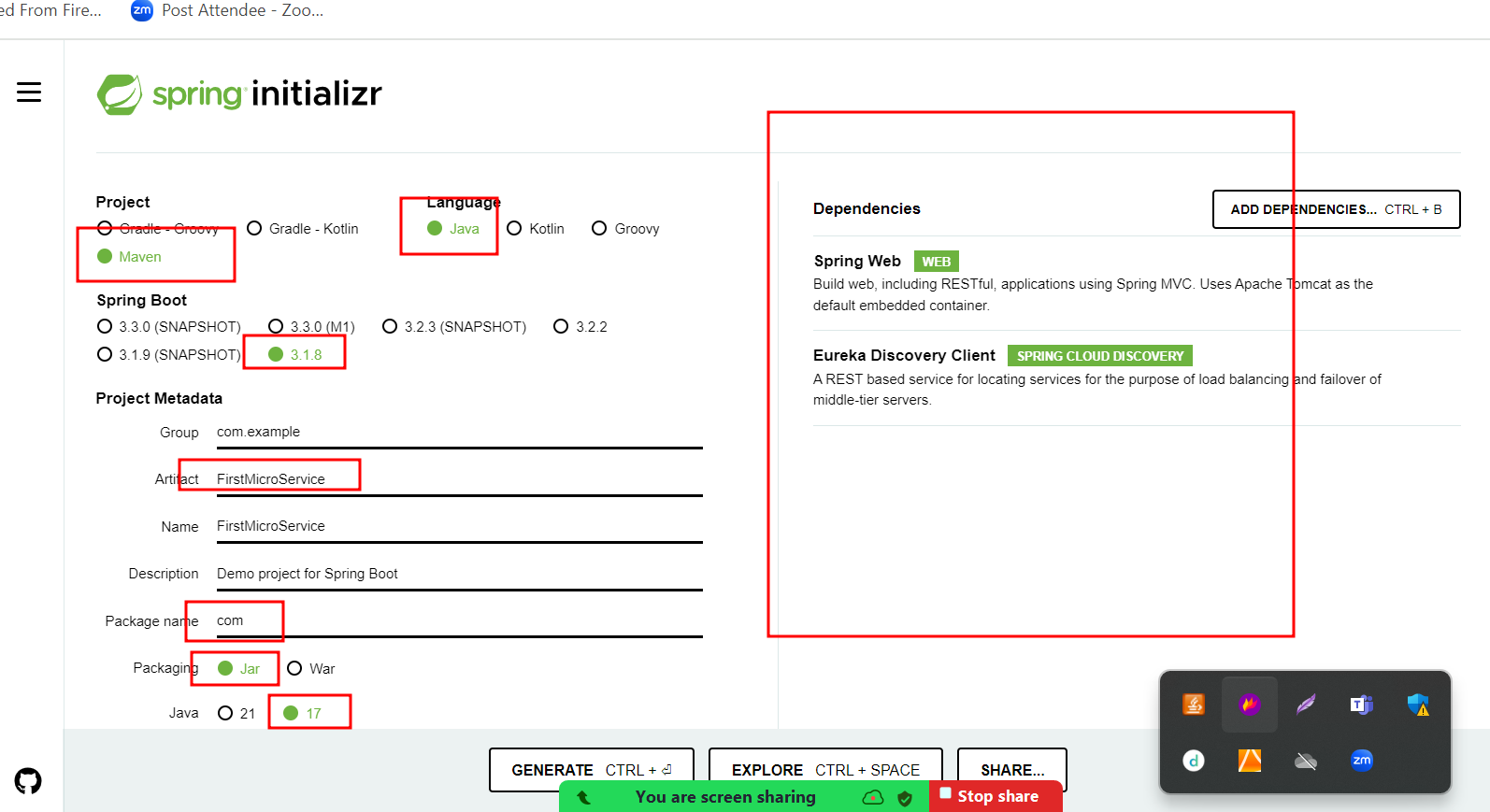
**}**

2nd project : FirstMicroService project which expose simple rest api.

Starter -🡪 web starter

Eureka discovery client

Port number 8181



application.properties=8181

main class

package com;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication(scanBasePackages = "com")

@EnableDiscoveryClient

public class FirstMicroServiceApplication {

public static void main(String[] args) {

SpringApplication.run(FirstMicroServiceApplication.class, args);

System.err.println("First micro service running on port number 8181");

}

}

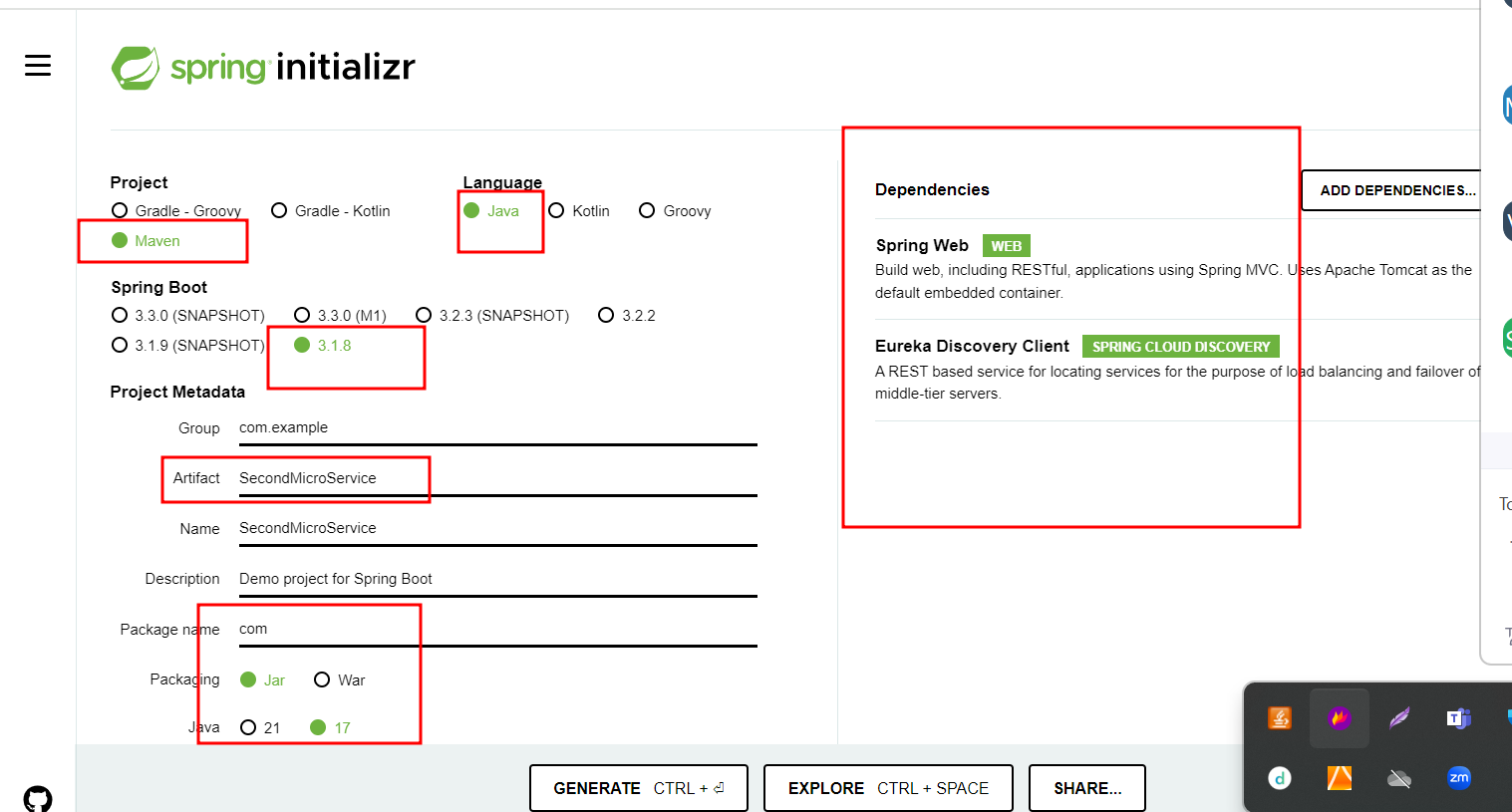
Then create more than one rest controller

3rd project : SecondMicroService

Starter 🡪 web

Eureka Discovery client

8282



4th project account-micro-service : it is going to interact with database.

Starter 🡪 web

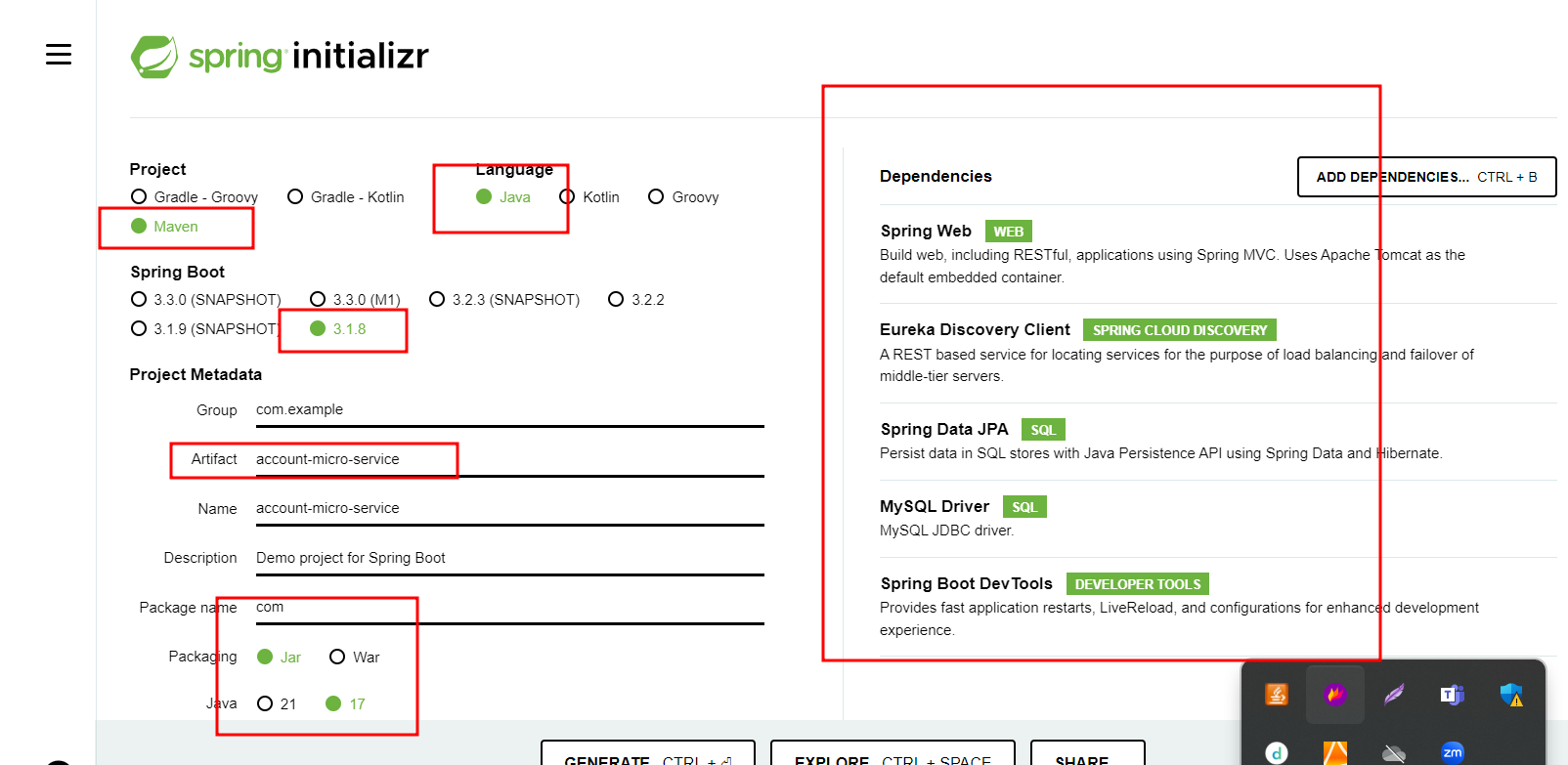
Eureka discovery client

Jpa provide orm features

Mysql help to connect database

Devtool refresh the project.

Port 8383



**Day 11 :**

**10/02/2024**

**Account Table -🡪 different database**

**Accno, name, emaild, amount**

**1 Ravi** [**ravi@gmail.com**](mailto:ravi@gmail.com) **12000**

**GPay -🡪 different database**

**Gpayid emaild**

**100** [**ravi@gmail.com**](mailto:ravi@gmail.com)

**Gpayid we want account details part of different project and running using different application with different port number.**

**GPaymicro service or PayPal or Paytml or PhonePay micro service**

**5th project gpaymicroservice**

**Starter 🡪 web**

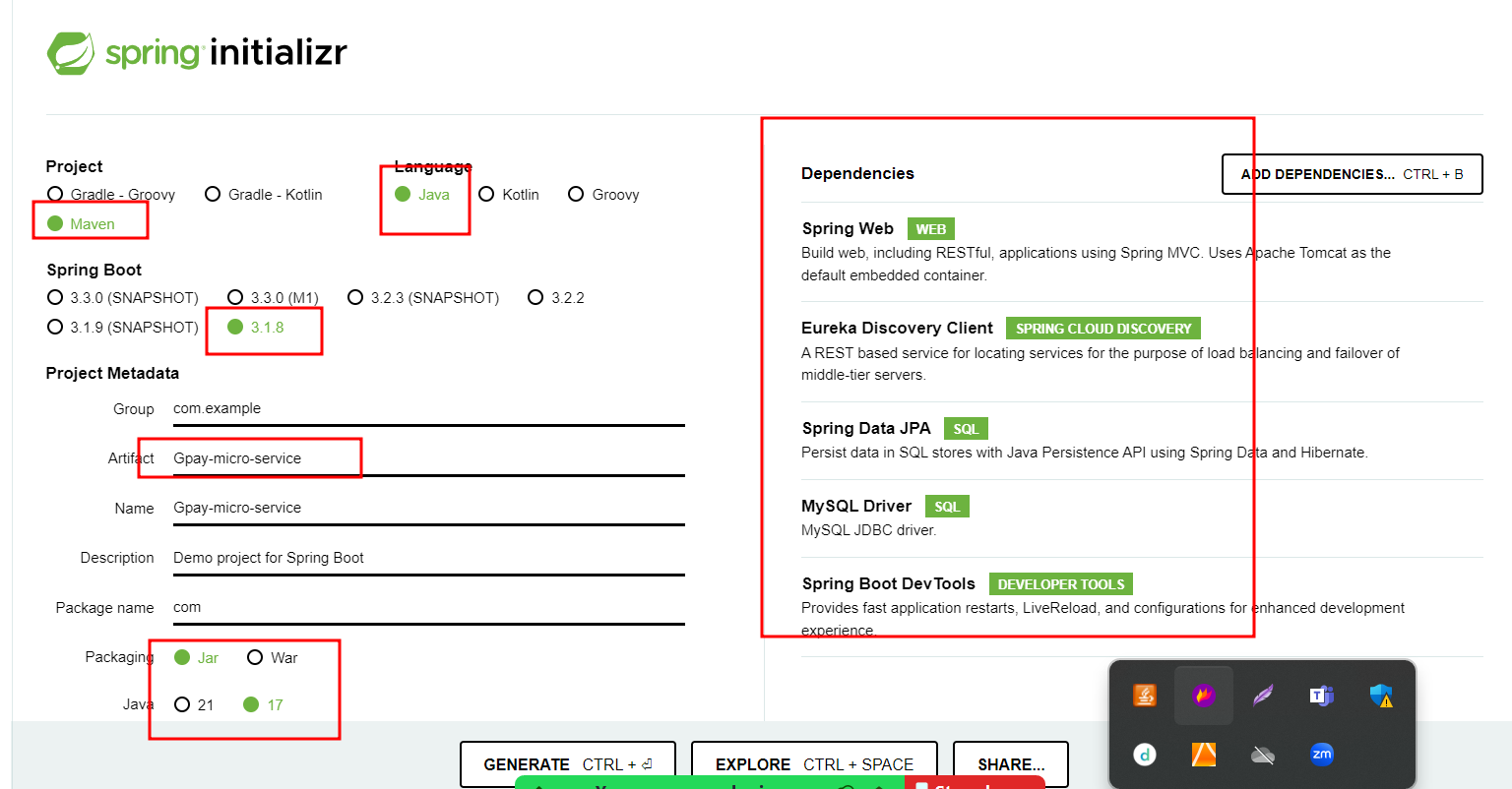
**Eureka discovery client**

**Jpa**

**Mysql**

**Devtool**

**Port number 8484**



**Account micro service running on port number 8383**

**Account details accno, name, emailid and amount**

**Gpay micro service running on port number 8484**

**Gpay details with email id**

**If we pass gpay id then we get emailed and emailed we will pass the account micro service to get the account details ie amount.**

**Through spring boot application if we want to call another rest api or micro service spring provided pre defined api is RestTemplate. This class help us to call another rest api or micro services.**

**Couse 3 project**

**API Endpoints And Communication**

Course-end Project 1

Description

* To plan and develop the Microservice Architecture using Spring Boot
* To perform unit testing using JUnit
* Implement Spring IOC

**Prerequisites:**

* JUnit
* Spring
* SpringBoot
* WebServices
* MicroServices

**Description:**

This assignment is designed to help understand how to plan and develop the back end for a given problem. Further, to gain hands-on experience in designing the microservice architecture for the project and finally perform unit testing for the code.

**Problem Statement:**

George owns a Travel Company where he books cabs for his customers. To manage these bookings, he needs Travel Management Software. He meets Kia, who runs a Software Solution Company and tells her the requirement.

Kia aims to develop the Project using Spring and Spring Boot Framework. She needs to create microservices to provide a solution using REST architecture. Finally, perform the unit testing for various web methods.

**Tasks:**

Perform the below activities as a solution:

* Create a Spring Boot Application in Eclipse EE
* Configure Spring Web dependency in the project
* Develop HTML Web Pages for the front end
* Web Page to Book a Cab
* Web Page to View all the Bookings
* Create an ApplicationController with suitable Annotations from Spring
* Create Microservice to book the cab
* Create Microservice to calculate the fare for cab
* Write Unit Tests to test the microservices
* Build run and test the project on Postman
* Test the project with front end web pages
* Create an executable jar file using maven wrapper

**Course 3 : 3 projects**

1. **Eureka Server**

**Starter : Web**

**Eureka Server**

1. **Booking-cab-service 8181**

**Web**

**Eureka discovery client**

**Thymeleaf**

**Jpa**

**Mysql**

**Devtool**

**View -🡪 thymeleaf dynamic html page**

**Form**

**CName, FromLocation ToLocation typeOfCab**

**CabBooking entity class with variable and setter and getter method.**

**Controller must normal controller not rest controller @Controller**

**One method using HttpServletRequest take value of cname,fromlocation, tolocation and typeofcab and set to CabBookign entity class and pass this object to service layer**

**Service layer do the DI for Repository and RestTemplate and call CabFare micro service pass information as fromlocation, tolocation, typeofcab, numberOfseats and get the amount.**

**CabBooking**

**cabId is pk with auto\_increment,**

**from location**

**tolocation**

**cname**

**numberofseats**

**using localdateandtime find the system data and time.**

**and amount we will get from cabfare if we pass correct information else we can say cab is not available. If correct then we can book the cab.**

**In service layer we can call pre defined method to save booking details.**

**Database**

**CabBooking(PK) CName DateAndTime, fromlocation, tolocation, typeofcab amount**

1. **Cab-fare-service 8282**

**Web**

**Eureka Discovery client**

**JPA**

**MySQL**

**Devtool**

**Entity Cabfare ciid,fromlocation,tolocation,typeocab,numberofseats, amount**

**Setter and getter method**

**Repository**

**Save pre defined**

**Find pre defined method**

**Find amount custom method because we need to find amount with where clause is fromlocation, tolocation, typeofcab and numberofseats.**

**Service**

**RestController we are creating three rest api no view as jsp or thymeleaf.**

**Post method to store cab details**

**Get method to find all cab details**

**Find cabfare with fromlocation, tolocation, typeofcab, numberofseats : we will return amount**

**Car-Fare : Table**

**CFID(PK) fromLocation toLocation typeofCab numberOfSeats Amount**

**1 A B Cab1 7 1000**

**2 A C Cab2 5 1200**

**C A D Cab3 7 1500**

**Day 12 :**

**17/02/2024**

**Testing :** testing is use to find the defect or error or bugs in the application.

If we run the application using main method or main function testing is not required.

Read a, b

Sum = a-b

Write or display sum

**View layer, controller layer, bean or entity layer, service layer, dao layer , resource layer. Etc.**

**Checking the code without main function or main method is also known as testing.**

Testing mainly divided into 2 types.

* 1. Black box testing

Input Process Output

* 1. White box testing

Input Process Output

Unit testing : unit testing is a type of software testing which help test function functionality working or not. Unit means smallest which work independently which written inside function or methods. Unit testing we do early stage.

jUnit : junit is an open source testing framework provided by third party which help to do the unit testing for java application.

Junit testing version

3.x with pre classes and interfaces.

4.x with annotation

5.x with annotation, support java 8 features and combination of more than one module.

Core Java testing using jUnit 5.x version

Spring boot testing

jUnit test case : it is a type of testing class which contains more than one test function with @Test annotation which help to test the function functionality working or not.

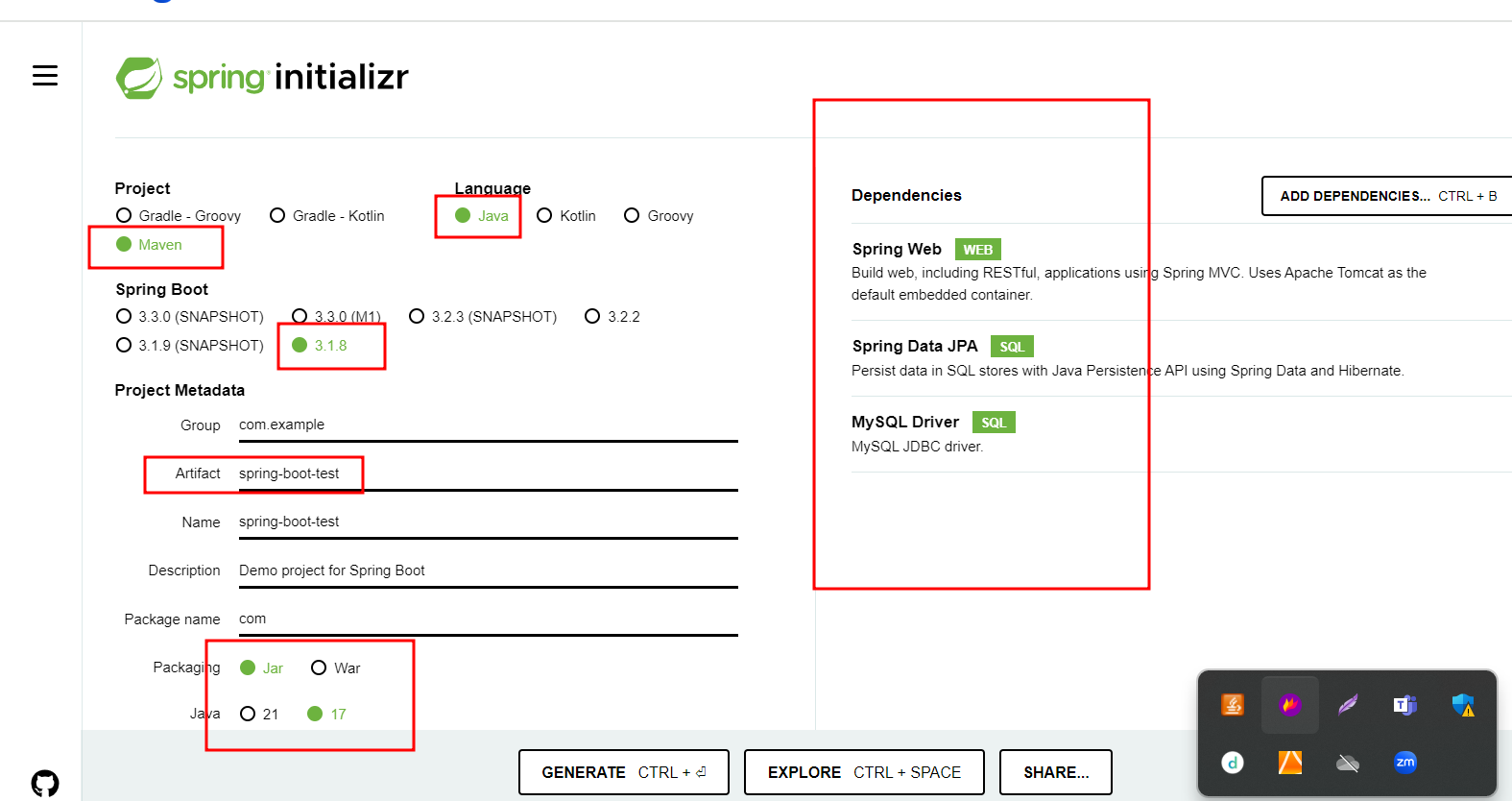
Unit provided lot of assertXXX method which help check actual and expectation output.

Junit hook functions.

Test suite : test suite is responsible to execute more than one test case classes.

Suite: collection of more than one test case.

Spring boot testing



Mock the object or provide proxy object or provide fake objects.

View layer depends upon controller layer or rest controller layer ie view layer can make mock for rest api.

Controller or rest controller depends upon service layer ie controller can make mock of service layer

Service layer depends upon dao or repository layer ie service layer can make mock for dao layer

Repository layer depends upon db.

Spring boot testing internally use Mockito library to create the mock of the objects.

Day 14 : 18/02/2024

Frontend and backend communication using rest api with database mysql db

Frontend is angular application

Backend is spring boot with data with rest api with micro service concept.

e-commerce or on-shopping

admin -🡪 user : no need to signup only sign

admin dashboard --🡪 product -🡪 add, delete, update and view

view all customer order details

view all customer information

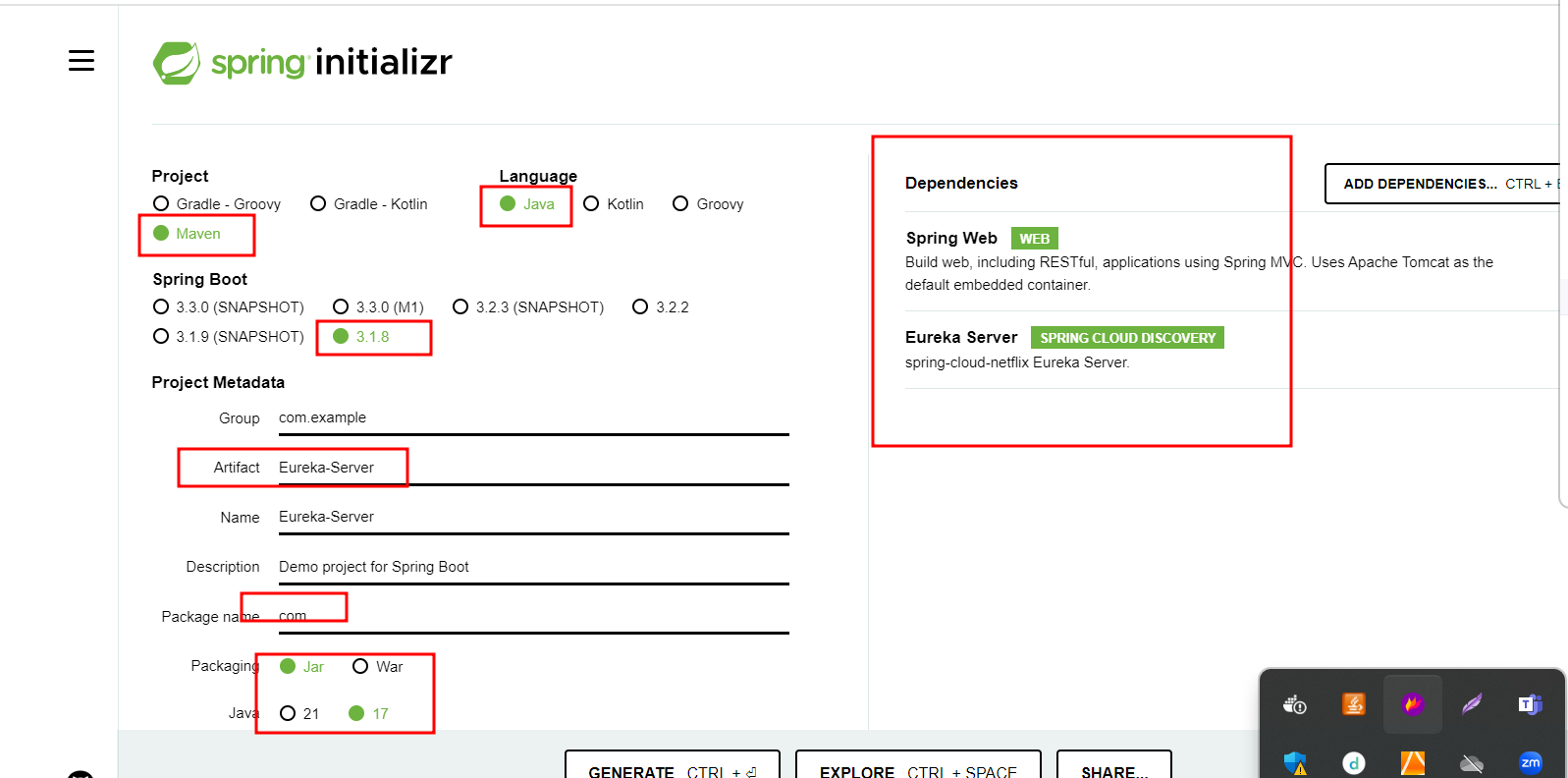
customer -🡪 user : we can SignIn and SignUp

customer dashboard -🡪 view the product and add those product in cart and place the order

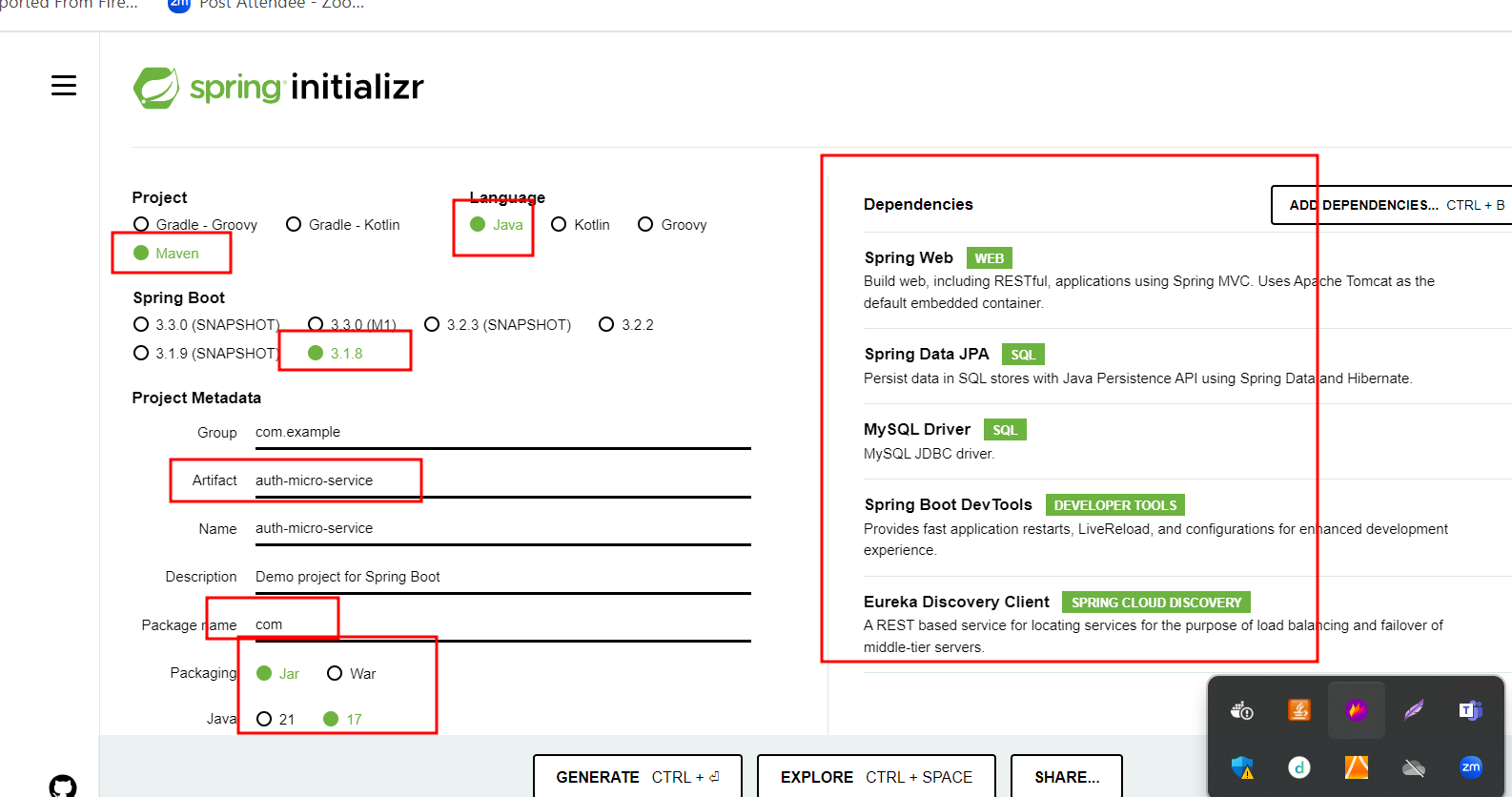
---🡪 it an view its own order details.

Backend -🡪

1st -🡪 Eureka Server to run Eureka Server



2nd -🡪 auth-micro service application



Frontend : Angular

ng new shopping-app

Routing -🡪yes

Styling 🡪 css

Ng g c login component

Ng g c signup component

Ng g class login model class which help to backend json data.

Ng g s login service to call rest api

Ng g c admindashboard component

Ng g c customerdashboard component

In angular HttpClient all methods like get, post, put and delete return by default json format.