**Phase 3 : 7 days**

**Day 1**

**20-03-2023**

**Spring framework and spring boot**

**Framework. Framework is a collection of classes and interfaces which internally connected to each other to perform a specific task. If we develop any enterprise application without framework we need to do more configuration. But if we develop the application with the help of any framework take core those all configuration. Design pattern. Design pattern means best practise or solution for repeating problems. If we develop any application using any framework we are following standards because all framework internally follow design pattern.**

**Collection framework is like a data structure.**

**MVC : Model View Controller**

**View 🡪 html or jsp**

**Controller 🡪 servlet**

**Model -🡪 javabean class or entity class**

**Service class**

**Dao class**

**Resource class**

**Jsp and servlet object creation is taken care by web container. Web container is a part of web service ie tomcat.**

**But web container doesn’t create object of java bean class, service class, dao class as well as resource class etc.**

**To improve the model class we were use EJB (Enterprise Java Bean).**

**View 🡪jsp and html**

**Controller 🡪 servlet**

**Model 🡪 EJB**

**To run the ejb application we required ejb container. Ejb container is a part of application server application server are heavy weighted server. Ejb is very complex technologies**

**Framework**

**Struts : struts is open source web framework which internally follow mvc architecture. Struts is known as controller centric framework.**

**JSF (Java Server Faces) : jsf is open source web framework part of oracle. It internally follow mvc architecture. JSF is known as View centric framework.**

**JSF Vs Angular / React JS / View JS**

**Hibernate : hibernate is replacement of JDBC.**

**Spring Framework : spring is open source layered architecture or onion architecture framework. Spring is light weighted framework.**

**Spring framework provided lot of modules which help to develop any type of application on demand.**

**Spring framework modules**

**Spring core**

**Spring context**

**Spring web**

**Spring mvc : spring mvc internally follow mvc architecture framework.**

**Spring mvc is known as model centric framework.**

**Spring MVC Vs EJB**

**Spring mvc allow use to integrate with struts or JSF.**

**Spring jdbc**

**Spring orm spring orm allow us to integrate with existing orm tool like hibernate, jpa or iBaties etc.**

**Spring data**

**Spring rest**

**Spring boot**

**Spring security**

**Spring micro service**

**Spring core and context**

**IOC : Inversion of control**

**DI : Dependency Injection**

**IOC is a concept. IOC is a design programming pattern.**

**In place of creating or maintaining any resources explicitly allow to maintain by container. If container create it will maintain the life cycle of those resources. Whenever we need pull from container, use it and leave it. If we create we need to maintain properly.**

**DI : DI is an implementation of IOC.**

**Using two ways we can do the DI in spring framework**

1. **Constructor base di**
2. **Setter base di**

**Using DI we pull the resource from container. Like objects**

**Spring container is light weighted container. If we want to achieve only ioc and di we don’t need to web server or application need to add few jar files inside our project can we can achieve di and ioc. Container are part of those jar files.**

**To achieve these concept we need to configuration using**

1. **Using Xml file**
2. **Using annotation**

**DI using constructor base and setter base with XML Configuration**

**Singleton design pattern : if we want to create only one object of that class but more than one reference then we have to use singleton design pattern.**

**Using constructor base we have to achieve fully dependencies.**

**Using setter base di we can achieve partial dependencies.**

**In parameterized constructor base di number of parameter as well as type of parameter must be match.**

**In setter base number of parameter as well non mandatory to match.**

**Day 2**

**21-03-2023**

**Auto wiring : Auto wired is one the great features of spring framework which help to enable you to inject the complex object dependency implicitly rather than doing explicitly with help ref attribute on property or constructor-arg tags.**

**If we use autoWired=”byteType”. Spring framework automatically scan the xml file if it found bean definition for that complex object it automatically to the di for that complex object.**

**In bytype we need only one bean definition for that complex object to achieve di.**

**If we have more than one bean definition then we have to use typeName.**

**In typename bean id name and complex property reference name must be match.**

**Spring framework do the di for primitive property implicitly. If we want to do the di for complex property we need to use ref attribute.**

**DI using Annotations**

1. **We can use xml file for enable the annotation**
2. **We can use configuration class to enable the annotation.**

**@Component : it is type of annotation provided by spring framework we need to use on pojo or javabean class.**

**It is equal to <bean class=”packagename.className”></bean>**

**By default id is classname in camelnaming rule. Means if class name is one word then id is classname in lower case like**

**Address class name then id is address.**

**If class name more than one word then**

**AddressInfo class name then id is addressInfo**

**@Autowired annotation we need to write on complex object.**

**@Value : this annotation we need to use to primitive property to set default value.**

**@Configuration : this annotation is replacement of beans.xml file.**

**Spring with XML and annotation configuration with Database using DataSoruce and JdbcTemplate**

**This project help use to improve model layer.**

**JavaBean class**

**Service class**

**Dao class**

**Resource class**

**Create the maven project**

**SpringwithJDBC**

**DataSource : it provide database connection from application server with high secure environment and singleton connection object.**

**But now a day we can achieve same concept with help of spring framework with few jar file and those configuration we will write inside a spring configuration xml file.**

**To do this spring framework provided pre defined class ie DriverManagerDataSource. This class configuration we have to do in xml file or in java class and provide four details like drivername, url, username and password.**

**@Repository : this is specific annotation we have to write on DAO layer or class. We can write JDBC code or hibernate code inside a DAO class.**

**@Service : this annotation we have to write on service class or layer.**

**Day 3**

**Spring DAO**

**Spring provided pre defined API ie JdbcTemplate which internally jdbc code and provided lot of pre defined methods to the jdbc code very easily.**

**To use JdbcTemplate we need to configurate jdbctemplate in xml configuration file and in dao layer do the di for JdbcTemplate rather than DataSource.**

**Spring MVC :**

**Spring MVC (Model View Controller). Spring MVC module help us to develop web application. It internally follow MVC Architecture framework.**

**Spring MVC provided pre defined class ie DispatcherServlet. This class follow front controller design pattern.**

**If we want to implements Spring MVC concept with Spring boot. We need to configure FrontController ie DispatcherServlet in web.xml file which is consider as FrontController.**

**index.jsp 🡪view**

**Spring MVC means web project with tomcat server and we need to convert this project to maven to add the dependencies.**

**Now we need to add spring mvc dependencies.**

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

**Index.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h2>Spring MVC Simple Project</h2>

<a href=*"hello"*>Click Here</a>

</body>

</html>

**When we click on hyper link the request pass to web.xml file. In web.xml file we configure url-pattern as /. Means any request receive by DispatcherServlet.**

**After receive the request by DispatcherServlet. It search spring configuration with name as dispatcher-servlet.xml (here dispatcher is servlet name in servlet-name tag).**

**servletname-servlet.xml**

**A-servlet.xml**

**This file must be inside web-inf folder.**

**Normal servlet**

**class Demo extends HttpServlet {**

**public void doGet(HttpServletRequest req, HttpServletResponse res) {**

**}**

**public void doGet(HttpServletRequest req, HttpServletResponse res) {**

**}**

**}**

**@Controller : this class consider as servlet and controller Specific class**

**class MyController {**

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

**public ModelAndView sayHello() {**

**// do coding**

**ModelAndView mav = new ModelAndView();**

**mav.setViewName(“display.jsp”);**

**return mav;**

**}**

**}**

**@RequesMapping is a type of annotation which is use to map the request which we receive from view**

**Day 5**

**Spring boot**

**Spring boot is another module which help to develop the application very easily with less configuration. Spring boot provides RAD features (Rapid Application Development)**

**In short**

**Spring boot = all spring modules (like spring core, context, mvn, dao, rest, aop etc) – No XML file + in build web server it tomcat or jetty + few annotation.**

**Spring boot itself is standalone project which help to develop any type of projects.**

**Spring boot components**

1. **Spring boot starter :**

**Spring boot starter main responsibility to combine all group of common or related jar which help to develop type of modules.**

**Spring boot web starter**

**Spring boot jdbc starter**

**Spring boot jpa stater**

**Spring boot testing**

**Spring boot security starter**

**Spring boot log4j starter**

1. **Spring boot auto configuration**

**Before spring boot we configuration spring application using xml or annotation file. But spring boot automatically do all configuration base upon starter with jar file added in project classpath.**

**In spring boot we use application.properties file which provide all configuration details in form json.**

**@SpringBootApplication = @ComponentScan + @Configuration + @EnableAutoConfiguration annotation.**

**In spring boot no web.xml file no spring configuration file.**

1. **Spring boot cli**

**We develop spring boot mvc**

**We need to create maven simple project.**

**Spring boot web starter internally use tomcat server which by default run on port number 8080**

[**http://localhost:8080/**](http://localhost:8080/)

**if we want view as a jsp then we need to configure few configuration in application.properties file and we need to add one dependency.**

**In spring if we want view as jsp then we need to add one dependency**

**<dependency>**

**<groupId>org.apache.tomcat.embed</groupId>**

**<artifactId>tomcat-embed-jasper</artifactId>**

**<version>9.0.38</version>**

**</dependency>**

**We need to provide vew resolver details in application.properties file.**

**application.properties file**

**spring.mvc.view.prefix: ./**

**spring.mvc.view.suffix: .jsp**

**In Spring boot we can connect to database using**

1. **JdbcTemplate : jdbc starter**
2. **JPA (In Spring boot hibernate is deprecated).**
3. **Spring JPA Data for both we need to use jpa starter**

**Spring MVC with JPA (as ORM tool) and View we use as JSP.**

[**https://start.spring.io/**](https://start.spring.io/)

**using this official website we can create the spring boot project with all required starter with build tool can be maven or gradle. This project contains pom.xml file (if build tool is maven) and created main class with @SpringBootAplication annotation and required files added project.**

**Spring boot with View**

1. **JSP**
2. **Thymeleaf**

**Thymeleaf : Thymeleaf is server side view engine develop using Java technologies.**

**It is upon html and xml base with HTML4 features.**

**With help of thymeleaf we can do dynamic task in html page.**

**In thymeleaf we use expression language**

1. **${…}**
2. **\*{…}**
3. **@{…}**
4. **#{…}**

**Inside with we can do the dynamic task.**

**Spring boot with thymeleaf as a View.**

**Spring Boot with View as Thymeleaf and Connecting database using Spring JPA Data with database mysql.**

[**https://start.spring.io/**](https://start.spring.io/)

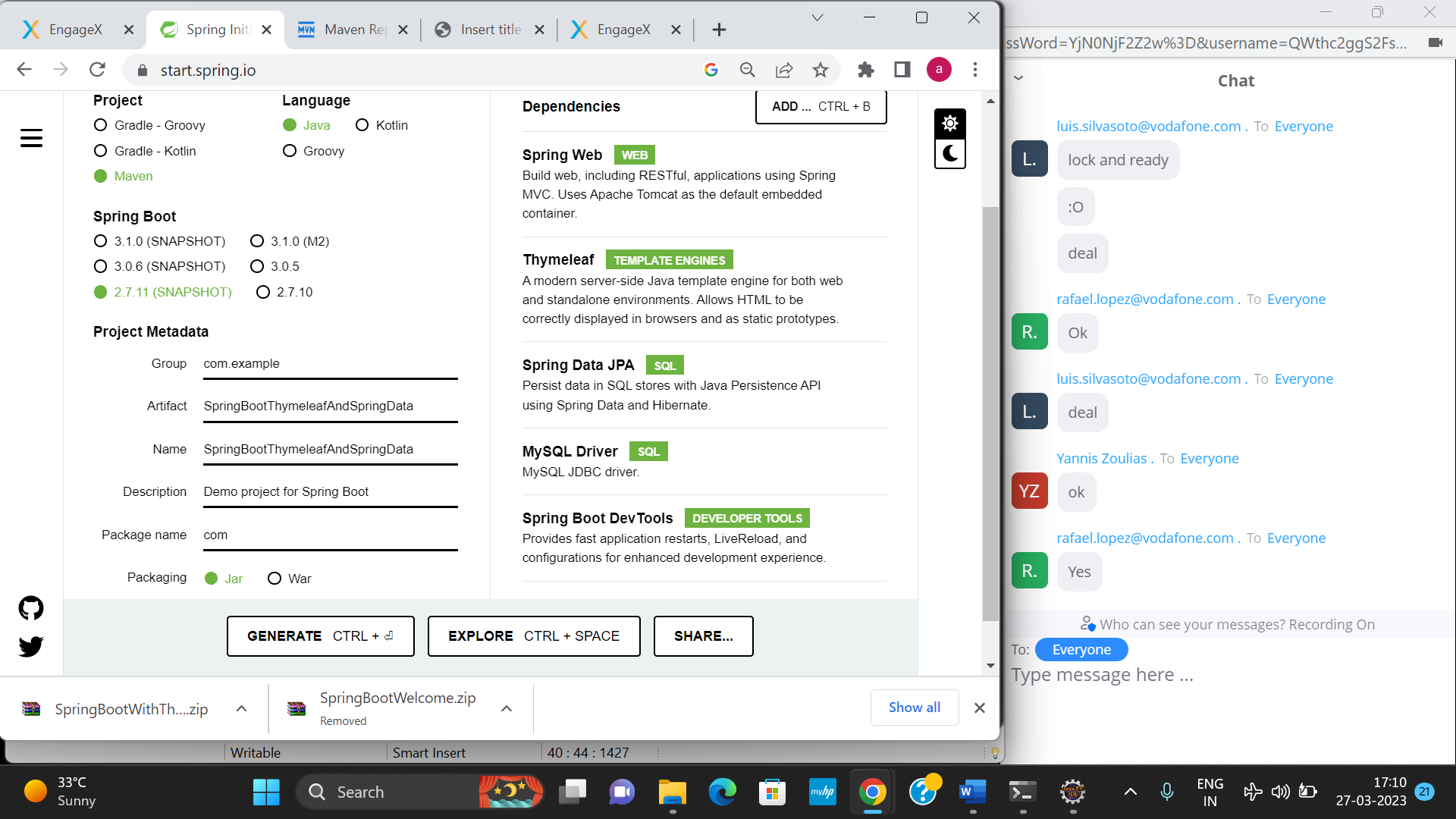
**web starter 🡪for spring boot with web**

**jpa starter 🡪 orm features**

**thymeleaf starter 🡪 view**

**mysql connector 🡪 mysql connection jar file**

**devtool starter 🡪 if we do any changes automatically refresh the project.**



**If we are planning to connect the database using spring data**

**We need to create one interface and that interface must be extends JpaRepository interface and this interface is base upon java 8 features which internally provided default implementation of all CURD Operation.**

**Day 7**

**28-03-2023**

**index.html (view thymeleaf)**

**emilid**

**password**

**typeof user admin customer**

**submit reset**

**signup link**

**customer.html**

**emialid**

**password**

**signup**

**admin user no signup**

**admin can login with** [**admin@gmail.com**](mailto:admin@gmail.com) **and admin@123**

**customer can do signup as well as signin**

**after login**

**customer can move customer home page**

**admin can move admin home page**

**admin can**

**addCategory.html**

**cid**

**cname**

**addProduct.html**

**pid**

**pname**

**cid**

**price**

**qty**

**viewCategory.html**

**viewProduct.html**

**viewCustomer.html**

**viewOrder.html**

**customer can**

**viewProduct.html**

**orderProduct.html (like update product code)**

**create table category(cid int primary key,cname varchar(25));**

**class Category {**

**cid**

**cname**

**@OneToMany**

**@JoinColumn(name=”cid”)**

**List<product> listOfProduct;**

**}**

**create table product(pid int primary key,pname varchar(25), cid int, price float, qty int, foreign key(cid) references category(cid));**

**class Product {**

**@Id**

**pid**

**pname**

**price**

**qty**

**}**

**Phase3 Project**

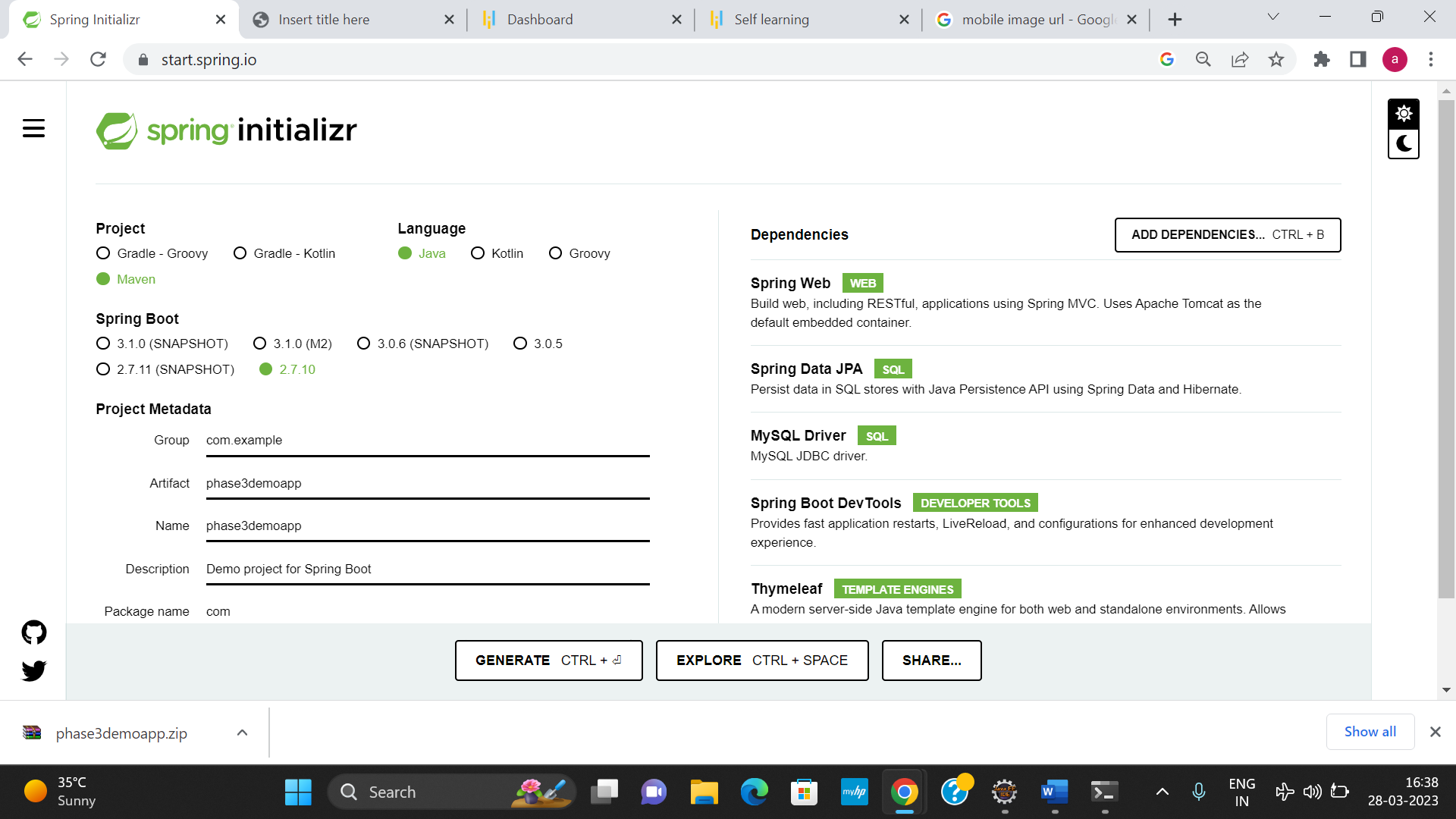
**Dao**

**jdbcTempalte**

**jpa**

**Spring data**

**Phase 3 spring io**



**Day 8**

**03-04-2023 (Extra Session )**

**Admin can do some operation**

**Add the product**

**Can view all order details.**

**Customer can do some operation**

**Customer can view the product**

**Customer can order the product**

**Spring MVC or Spring Boot with MVC :**

**View can be JSP or HTML or thymeleaf or any other engine.**

**If we use these view technologies. Our view become rightly coupled with backend technologies.**

**Amazon (Java Technologies) --🡪 Payment gateway**

**Paypal asp.net**

**Paytm python**

**Google Pay node js**

**Credit or Debit cart in other banking**

**English**

**Java Request to XML/JSON paypal or paytml or any other application**

**Asp.net**

**XML :eXtensible markup language**

**JSON : JavaScript object notation.**

**Web Service : Giving the service for web application when both application running using different technologies.**

**2 types of web service**

1. **SOAP base web service: Simple Object Access Protocol. : In SOAP web service we can consume and produce the data only in the form of XML.**
2. **Rest full web service : Representational state transfer**

**In Rest full web service we can consume and produce the data in any other format like xml, json, plain text, html or medial type etc.**

**In controller layer @Controller annotation is replaced by @RestController.**

**If we use @Controller annotation then our view must be jsp or html or thymeleaf or any other engine.**

**If we can @RestController then our view can be any other technologies like Aps.net, php, angular , react js or any rest client.**

**Using this we have to create rest api.**

**Those api to store data, delete date, update data and retrieve data.**