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.

$$\begin{tikzcd}
\text{JSP} & \text{Servlet} \arrow[dr] & \\
\text{JSP} & \text{Servlet} \arrow[r] & \text{EJB payment gateway} \\
\text{JSP} & \text{Servlet} \arrow[ur] &
\end{tikzcd}$$

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
5. Spring Rest
6. Spring security
7. Spring cloud
8. Spring micro service
9. Spring ORM
10. Spring AOP
11. 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

Di we need achieve fully DI.

We can't achieve partial DI

It can be empty or parameter

Empty default value.

Parameter passed value.

Parameterized DI order matter.

Setter base DI internally use empty

Constructor Base DI. Using setter we override the value

in setter order doesn't matter as well as we

can achieve partial DI.

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.
2. JDBC use SQL language. SQL is Database dependent language.
3. JDBC throw checked exception and exception hierarchy is database dependent.
4. 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.

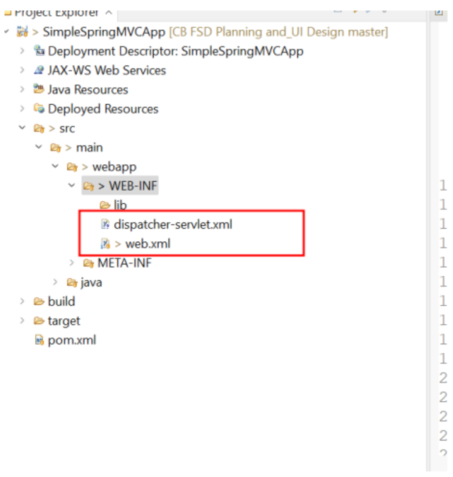
```
<servlet>
     <servlet-name>dispatcher</servlet-name>
     <servlet-
class>org.springframework.web.servlet.DispatcherServlet</
servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
  </servlet-mapping>
```

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.

2. 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.

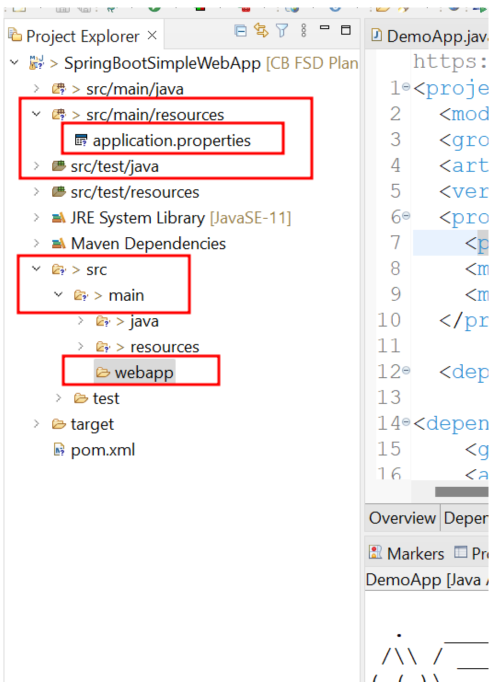
Spring boot application with manual configuration for starter.

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.
3. 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.

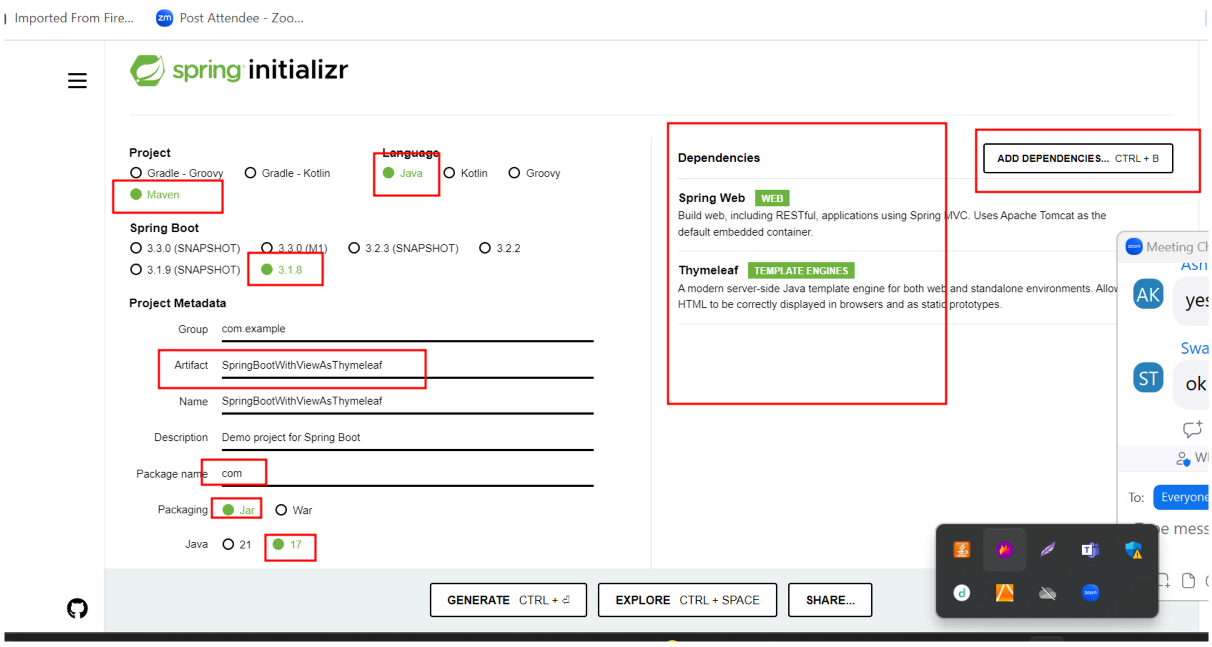
Now we need to provide view resolver details in application.properties files.

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.


Imported From Fire... Post Attendee - Zoo...

spring initializr

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 SpringBootWithViewAsThymeleaf
Name SpringBootWithViewAsThymeleaf
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.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allow HTML to be correctly displayed in browsers and as static prototypes.

GENERATE CTRL + ⏎ EXPLORE CTRL + SPACE SHARE...

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.

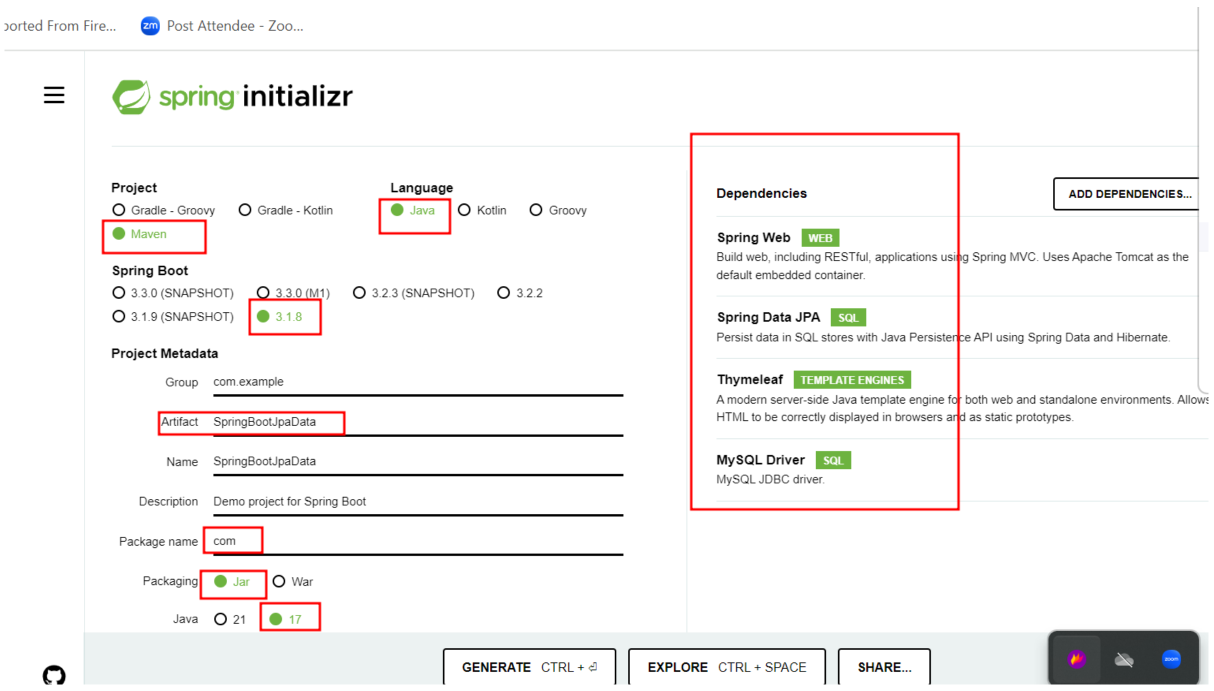
We need to create the project using spring initializer website.

Web starter

Jpa starter

Mysql connector dependencies.

Thymeleaf starter



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	SpringBootJpaData
Name	SpringBootJpaData
Description	Demo project for Spring Boot
Package name	com
Packaging	[✓ Jar] War
Java	21 [✓ 17]

Dependencies

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

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

MySQL Driver SQL
MySQL JDBC driver.

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.