Java Phase 3 :

11 classes

Day 1

16-08-2022 Spring Framework and spring boot

JEE : Servlet, JSP and EJB.

View -🡪 HTML and JSP.

Controller -🡪 Servlet

Model -🡪 JavaBean or Entity class

Service class

Dao class

Web Container is a part of server responsible to take care execution of servlet and jsp. Web container will create the object of servlet and jsp and call life cycle methods.

EJB : Enterprise Java Bean : EJB container which is part of application server is responsible to improve the model layer. EJB help us to improve the model layer.

View 🡪HTML or JSP

Controller -- > Servlet

Model 🡪 EJB

EJB is very heavy component as well as very complex to run the application. To run the ejb application we required application server.

Solution for EJB application spring framework.

EJB Vs spring framework.

If we develop application with the help of servlet and jsp. It is not mandatory we have follow MVC design pattern or architecture.

Design pattern : Best practice or solution for repeating problem.

27 design pattern GOF

Creational design pattern

Structural design pattern

Behavior design pattern

Framework : Framework contains lot of pre-defined classes and interface which internally connected to each other to perform specific task. Framework also known as prototype or template but not a final product. If we develop any application using any framework 70 to 80% task taken care by framework. Framework internally allow all design pattern. The implementation of design pattern is taken care by framework.

Struts : struts is open source web framework provided by Apache. Struts internally follow MVC design pattern. It provided lot of classes to improve controller and model layer. Struts internally provided FrontController design pattern.it provide whole application flow. Struts internally provided pre-defined FrontController class.

Struts known as Controller centric framework.

JSF : Java Server Faces : JSF is open source web framework provided by Oracle. JSF internally follow MVC design pattern. It provided lot of classes to improve controller and model layer. JSF internally provided FrontController design pattern.it provide whole application flow. JSF internally provided pre-defined FrontController class.

JSF is known as View Centric Framework.

JSP -🡪 JSF VS Angular Framework or React JS

Hibernate : Hibernate is a ORM framework which help to improve DAO layer. Hibernate is a replacement of JDBC.

Spring framework : spring is a open source light weighted layer architecture framework. Spring also known as onion architecture framework.

Spring provided lot of pre-defined modules which help to improve all layer in the application. On demand we can use any module to improve that specific layer.

Spring core

Spring context

Spring web

Spring MVC : spring mvc it follow MVC architecture framework. Provided lot of classes and interfaces to improve controller and model layer. It provide FrontController design pattern.

Spring MVC is model centric framework.

EJB VS spring Framework

Spring DAO

Spring ORM

Spring AOP

Spring REST

Spring cloud

Spring security

Spring micro service

Spring expression

Spring integration etc

Spring Core and Context

IOC : Inversion of Control

DI : Dependency Injection

IOC : Inversion of control : IOC is a concept. It is programming design pattern. In place of creating or maintaining any resource explicitly allow to container create and maintain. If we create we are responsible to maintain the life of the resource. If container will create it will maintain properly. From container pull any resource whenever required use it and leave it.

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

Using DI concept container inject the any external resource which we want. So we have to pull from container whenever we required.

Spring framework support two type of DI

1. Constructor base DI
2. Setter base DI

We have to configure constructor base as well as setter base DI using

1. Using XML file configuration
2. Using Annotation base configuration

We configure spring plugin in eclipse IDE

Constructor Base DI using XML file configuration

Now days we are developing spring framework application using two build tool

1. Maven
2. Gradle

Spring Provided to achieve DI for POJO class. Plain Old Java Object.

Spring container is a part of jar file. Which will help us to create the object for Plain Old Java Object. This is a normal class not to extends or implements Pre-defined class.

Day 2

17-08-2022 Spring Framework and spring boot

Constructor base DI using parameterized number of parameter and order of parameter must be match. We can’t partial DI using constructor base.

Setter base DI number of parameter and type of parameter not mandatory must be match. Using setter base DI we can achieve partial DI.

Spring framework do inject ie DI for primitive property implicitly

Autowired : autowired is a spring framework features which help to inject ie DI for complex property Implicitly rather than explicitly using property ref or constructor-arg ref attribute.

If we use auto wired features spring framework automatically scan for that complex object in xml file.

To Enable auto wired features using xml we have to use an attribute as autowire.

1. byType : in xml it must be contains only one bean declaration for that type of class.
2. byName : if we want more than one then we have to use byName. In byName option id name and reference name must be match.

Constructor and setter base di using Annotation

@component : This annotation we can write on class level. This annotation we will write on POJO or javaBean class. By default id name is classname in camel naming rules. If class contains one world in id must lower case. If class contains more than one word then second word onward first letter upper case.

@Autowired : This annotation we can use on complex property level, constructor level or setter method level.

By default @Component annotation is not enable. We can enable using

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

Using xml file.

Using configuration file

@Value : This annotation we have to use to property level to set default value.

Using java configuration class

@Scope : this annotation we can write on class level to make singleton or protocol. By default singleton.

Spring MVC : Model View Controller :

Spring MVC internally follow MVC architecture framework. It provided front controller pre-defined class ie DispatcherServlet. Which we have to configure in web.xml file.



@Controller : this class now behave like Servlet.

class MyController {

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

public ModelAndView sayHello() {

ModelAndView mav = new ModelAndView();

mav.setViewName(“display.jsp”); /RequestDispatcher

return mav;

}

}

Day 3

22-08-2022 Spring Framework and spring boot

First we have to create dynamic web project with tomcat server

Then we have to convert this project into maven project . right click on project and select the option as configure and then convert to maven.

Then open the pom.xml file and which java version in release tag may sure 1.8 or 11 or 15 etc.

Then we have to add the spring mvc dependencies.

Then in web.xml file add the dispatcher servlet as front controller.

<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>

url pattern is /. It means url can be hello, hi etc.

Then create spring configuration file start with pre-fix servlet name present inside <servlet-name> tag followed by servlet.xml ie dispatcher-servlet.xml

And while creating file please select the option as context.

dispatcher-servlet.xml

<context:component-scan base-package=*"com"*></context:component-scan>

It is use to enable @Controller annotation.

Then we have to make normal java class with Annotation @Controller and inside that class we can write more than one method with return type is ModelAndView. On those method we have to write @RequestMapping annotation with path and method options.

package com.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

@Controller // this class like servlet.

public class MyController {

// doGet method with meaningful method name

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

public ModelAndView sayHello() {

ModelAndView mav = new ModelAndView();

System.out.println("I Came here");

mav.setViewName("display.jsp"); // RD forward

return mav;

}

}

Flow of the application

Index.html --🡪 web.xml 🡪dispatcher-servlet.xml -🡪MyController 🡪 sayHello with path inside @RequestMapping annotation -🡪 Display.jsp