[Spring] practical notes……

Create maven project

Mandatory-internet should ON

1) file

2) new

3) others(type- maven project)

4) select and click-next button

5)here we have to click on check-box (create simple project)

Because we are not going to develop a web page so. 👆

6) click-next button.

7) it will directly go to the page called [group-id] and [artified-id]

{

This we done in servlet (recalling)

8)if we don't select a simple project [check-box] it will pop-up page

9)below 3 points as I mentioned.

7) "select as" all catalogs (down text-box -type) org.apache.maven

8) scroll down select org.apache.maven.web.application (1.0) or (1.4)

9) click-next button (watch bottom left corner downloading maven-project)

(if it struct at one point plse type (lower case letter) y and enter)

}

8)now we can see our created project in the [PACKAGE EXPLORER]

//

Spring Module

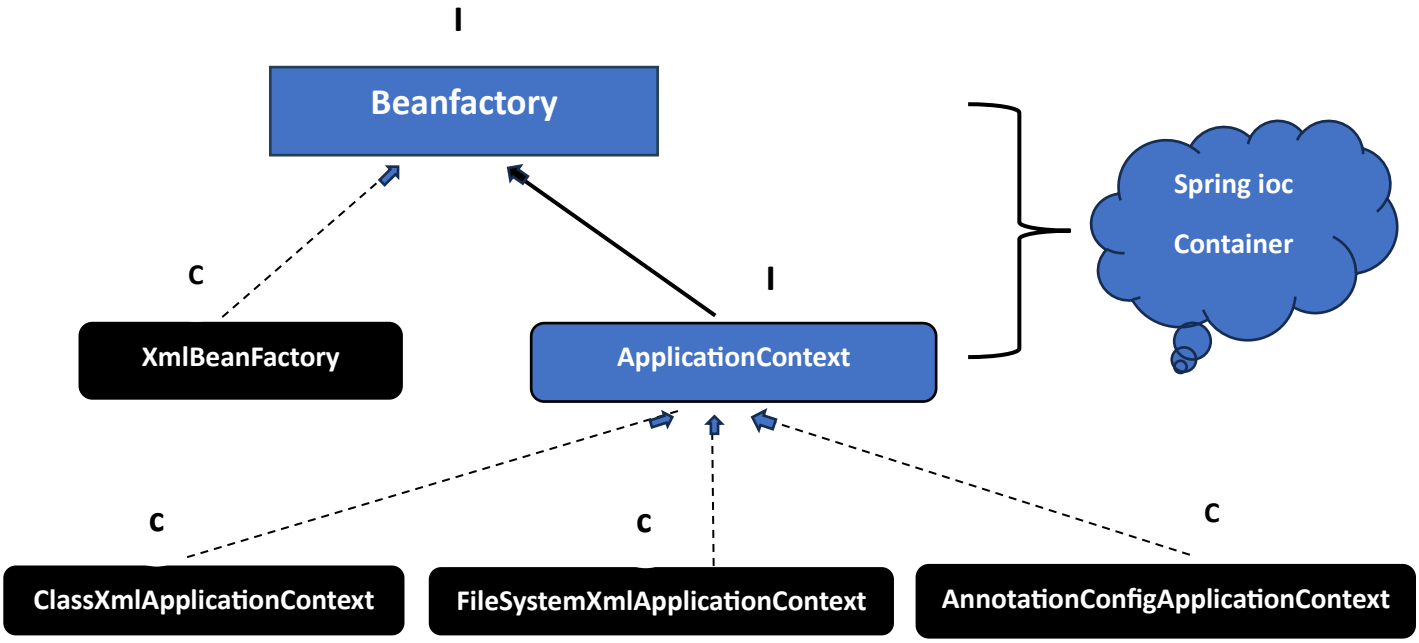
1) Spring IOC (inversion of controller)
2) Spring MVC (model view controller)
3) Spring Boot

1) Spring IOC

Program example : refer a previous page

I = interface

C = class



Maven project check box recall:

1) Jdbc and hibernate " Tick "
2) Servlet " No Tick "
3) Spring IOC " Tick "
4) Spring MVC " No Tick "

Now,

In servlet we are manually creating a object for a non-static method between a class

But, here Spring will take care of creation a object of a particular class.

1) add a XML file exactly in the src/main/java.

not inside a package (it will throw an error)

- **file**
- **new**
- **other**
- **search xml file**
- **name it as ------.xml**
- **click next 👍**

2) Add dependency(exactly opening tag of <dependencies>)

- **type a tag<dependencies> </dependencies>**
- **Browser (maven repository)**
- **In search box type as " spring context "**
- **Select highest user platform**
- **In that search 5.0.18**
- **For me this version is not (run) working properly**
- **I just copied a version of 5.3.19**
- **Copy dependency**
- **Paste it in pom.xml**

//

To understand the spring concept

- **Create one package**
- **Inside the package**
- **Create 3 classes**
- **By naming**
- **1) Musicpalyer (non-static())**
- **2) phone1 (write main())**
- **3) phone2 (write main())**

Normally, we used to create a object by using a new operator, but using Spring ,

We have to give the object creation - class name to- XML file

- **Inside XML file**
- **There is a tag name called <bean> </bean>**
- **Inside that tag we have the attribute called class="" and id=""**
- **We have to copy the qualified path of Musicplayer calss name(point the cursor->right click->there is a option called copy qualified path).**
- **And paste it in a class attribute { class=" paste " }.**
- **And give the id name as well in the id attribute { id="name" } it is a user-defined-name**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation = "http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
       <bean id="myphone" class="springioc_basics_basic1.Musicplayer"></bean>
       ➢  </beans>
       ➢  (the code given by sir)
```

Inside a phone class we have to type this :

1st way of getting object

- **Using one of the spring IOC container, we are creating a object now**
- **Using a interface called AplicationContext we are giving a location of the object present.**
- **For that we use a class called ClassPathxmlapplicationContext(“”)’**
- **It accept a String type of argument (“ ”)**
- **This path we are going to store it in the AplicationContext interface- reference-variable=**

- **After,,**
- **we successfully create a object**
- **Using a reference-variable of a ApplicationContext interface**
- **We call the a method called a getbean();**
- **We going to pass a id=””, name**
- **It is also accept a String type of argument (“ ”);**
- **i.e location.getbean(“id=””””);**

- **now,,**
- **to use a method (music) present in the Musicplayer calss**
- **we have to downcasting a class (Musicplayer) {explicitly}**
- **and store it in a reference-variable**
- **than using a reference-variable { . } we can call the (music) method()**
- **run the program we can see a output.** 😍

```
ApplicationContext location=new ClassPathXmlApplicationContext("ioc.xml");
Musicplayer address=(Musicplayer)location.getBean("myphone"); //downcasting
address.startMusic();
```

- **job done.**

2nd way of getting object

- Using one of the spring IOC container, we are creating a object now
- Using a interface called BeanFactory we used to get an object.
- We know that we cant create a object for interface so,
- With the help of the class we create a object and store it in a interface as a reference-variable
- BeanFactory beanFactory=new XmlBeanFactory(resource); // middle line ----------)
- The type XmlBeanFactory is deprecated
- This line says that this is oldest way of getting a object , currently no one is using this type of way of getting a object.
- But the drawback of this interface we cannot give a xml path directly as in the ApplicationContext
- It will receive a resource's only//
- For that we have to use a another class called ClassPathResource
- Creating a object for this class,(in order to get a xml location)
- It accept a String type of argument (" ")
- We going to pass a reference-variable(resource) to the BeanFactory method();
- And store it in a BeanFactory as a reference-variable

- After,,
- Using a reference-variable of a BeanFactory interface
- We call the a method called a getbean();
- We going to pass a id="", name
- It is also accept a String type of argument (" ");
- i.e location.getbean("id="""");

- now,,
- we successfully create a object
- to use a method (music) present in the Musicplayer calss
- we have to downcasting a class (Musicplayer) {explicitly}
- and store it in a reference-variable
- than using a reference-variable { . } we can call the (music) method()
- run the program we can see a output. 😍
-

```
        // (2) bean==obj  obj==bean
          ClassPathResource resource=new ClassPathResource("ioc.xml");
          BeanFactory beanFactory=new XmlBeanFactory(resource);
        Musicplayer musicplayer=(Musicplayer)beanFactory.getBean("myphone");
//downcasting
          musicplayer.startMusic();
          Musicplayer musicplayer1=(Musicplayer)beanFactory.getBean("myphone");
//downcasting
          musicplayer1.startMusic();       // scope concept {singleton and
prototype}

//       sample s1=new demo();   upcasting.
//       Demo d1=(demo)s1;       downcasting.
```

job done.

To prove whether an object is created (or) not

Xml file store the class name (fully qualified path) and as well as id="";

(For Which class we are planning to create an object.)

To prove

Take a constructor in the object creation class(MusicPlayer)

And write a print statement to as (object is created)

Now, while running a program we will get a output what ever present in the music method()

Along with a constructor statements 👆.

In ioc xml file we have three attributes name as:

- Id=" "
- Class=" "
- Scope=" "

```
<bean id="myphone" class="springioc_basics_basic1.Musicplayer"
scope="prototype"></bean>
```

- Class // attribute is used to give a object creation "class name";
- Id //attribute -the data which is present in the (object class) i ioc xml file
- We store it in a id attribute
- We can give any name in id=" "
- It is user defined
- Scope //attribute is used to give a count of the object creation (in the Spring ioc)

We know that SPRING is by default a SINGLETON PATTERN

(IT is avoid the creation of object more than one time)

In scope attribute if we mention as 'prototype' - IT WILL show that how many time we have got a object in the program ex: <scope="prototype">

If in case we mention as ' singleton ' – even though if we call more than one time a object –" IT WILL show only a one time of an object creation 💙 " <scope="singleton">

Disclaimer (try to understand a statement which I have an wrote 🤓 because its not understood by me only)

The main difference between the ApplicationContext and BeanFactory

ApplicationContext	BeanFactory
• it is currently using for getting a object of a class	• it is deprecated
• by passing the path of the ioc xml file in the Application interface itself is going to get a object of a class even before using a getbean( ) method we got a object//	• but here for getting a path of ioc xml file we have to use a class(create a object) and we have to pass the resource to the BeanFactory method(" resource") than finally using a getbean( ) method we get a conformation of getting a object of a class .

• ex:
ApplicationContext location=new ClassPathXmlApplicationContext("ioc.xml");
	• ex:
	`ClassPathResource resource=new ClassPathResource("ioc.xml");`
	`BeanFactory beanFactory=new XmlBeanFactory(resource);`
	`Musicplayer musicplayer=(Musicplayer)beanFactory.getBean("myphone"); //downcasting`

• Points to remember: interface (file path) getbean() (id=" ") downcasting (2) lines of code	• Points to remember: Class (file path) BeanFactory interface (resource) getbean() (id=" ") downcasting (3) lines od code

SETTER INJECTION

Initializing a value for the global variable between the class by creating a object and passing a value through ioc xml inside the bean tag (<bean> </bean>) by taking a property tag (<property> </property>)

```
<bean id="lp" class="setter.injection.Laptop" scope="prototype">
      <property name="brand" value="dell"></property>
      <property name="colour" value="black"></property>
      <property name="cost" value="55000"></property>
```

Here, name attributes defines variable name { CASE SENCITIVE }

Value attribute defines the value which we are going to pass to the respective variable

Ex:

```
private String brand;
  private String color;
     private int cost;
```

using a value=" " we are setting a data to the name variable;

(it is not a definition of setter injection // it's a work process of setter injection)

Remember :

While running the object code – if we get an hexa-decimal-values;

Don't forget to take a toString() method; and override it. ❤

(alt+shift+s)

Ex: core java (concept)

Its like a ENCAPSULATION CONCEPT

(setting a value to private member of the class by taking a "getter()-setter()" method creating a object (manually through new operator)between the classes)

Note this:[**setter injection= <bean> </bean> tag**

Class and id attributes

<property> </property> tag

Name and value attribute **]**

Dependency Injection

We can inject the value in " 3 " ways:

1) Setter injection.
2) Constructor injection.
3) Variable injection.

Constructor injection

Through constructor we are injecting the value to the Global variable

Initializing a value for the global variable through constructor between the class by creating a object and passing a value through ioc xml inside the bean tag (<bean> </bean>) by taking a

<constructor-arg></constructor-arg>

```
<bean id="con_mobile" class="constructor.injection.Mobile" scope="prototype">
        <constructor-arg index="0" value="vivo"></constructor-arg>
        <constructor-arg index="1" value="250000"></constructor-arg>
        <constructor-arg index="2" value="black"></constructor-arg>
        </bean>
```

Ex:

```
   String mobile_name;
   String mobile_cost;
   String mobile_colour;
public Mobile(String mobile_name, String mobile_cost, String mobile_colour) {
        super();
        this.mobile_name = mobile_name;     // index o
        this.mobile_cost = mobile_cost;     // index 1
        this.mobile_colour = mobile_colour; // index 2
}
```

Tracing:

Remember while injecting a value through ioc xml file for the constructor everything stored in the form of index based 0,1,2……

- We are passing a value by taking a index="" attribute and value="" attribute through a tag called <constructor-arg></constructor-arg>
- Here, the constructor receiving the value through (arguments) ,
- And passing to the global variable using this. Keyword.

If we run the code we will get a object address, to over come this

Just override a toString() method,

Note this: [constructor injection= <bean> </bean> tag

Class and id attributes

<constructor-arg> </constructor-arg> tag

index and value attribute]

make a proper notes :

configuration , component, Autowired

just a overview: [@Component @Autowired @Configuration @ComponentScan (not : Scans)

If we not mentioned the ” @primary ”

We will get an error called

`org.springframework.beans.factory.NoUniqueBeanDefinitionException:`

If we give more than one “ **@primary** ”

We will **get an error called**

`more than one 'primary' bean found among candidates: [kitkat, milky_Bar]`

“”

“ **we have to mentioned only one primary anotation**”

Interface class **-> implement class -> implements class -> utilization class (autowired() and one method() {**

Void store() }) -> **controller class** **main class to handle all the activity.**

(normal way)

Now, have to make use of Configuration class

Create package inside a package itself (not in java/source) -> configuration class -> pass the package address to -> controller class (ex.class)-> in getbean(ex.class) receive a utilization class name]

Diagram for AnnotationconfigApplicationContext

DEPENDENCY INJECTION

We are injecting the data by using " getter and setter (or) constructor args tag inside the bean tag " through the external file called ioc.xml file with out touching a java code Is called as Dependency injection.

For example : refer dependency injection Concept 🤓

why ? Spring ioc called as inversion of control

(or)

Purpose of inversion of control

We are going to give control to the external file for getting an object without typing a NEW operator by using a inversion of control

- **We can get a object in two way using Spring ioc container called**
- **1) BeanFactory interface**
- **2) ApplicationContext interface**

If you ask which one is most preference for getting an object means:

- **i.e ApplicationContext**
- **Because passing a xml path address is enough to get a object, no need to wait for to call a getbean() method .**
- **But in BeanFactory**
- **We have to pass the resource for that we have to use a one more class to get a xml path address, than have to pass the (resource()) and than have to call the getBean() method than only we get an object of the class.**

- **So comparing this we can say that Application]Context is most preference**

2) Spring MVC

Servlet

In Servlet each and every CRUD operation we have to create different // different clases

(or)

Each and every request we have to create different classes

//

Spring MVC

In spring MVC for each and every request we going to mapping for " only one class " and manage all the request in a single class -> (achieve: by creating a MULTIPLE ( ) METHOD in a class)

//

We can map from front-end to middleware in " 2 " ways :

1) @WebServlet("/")

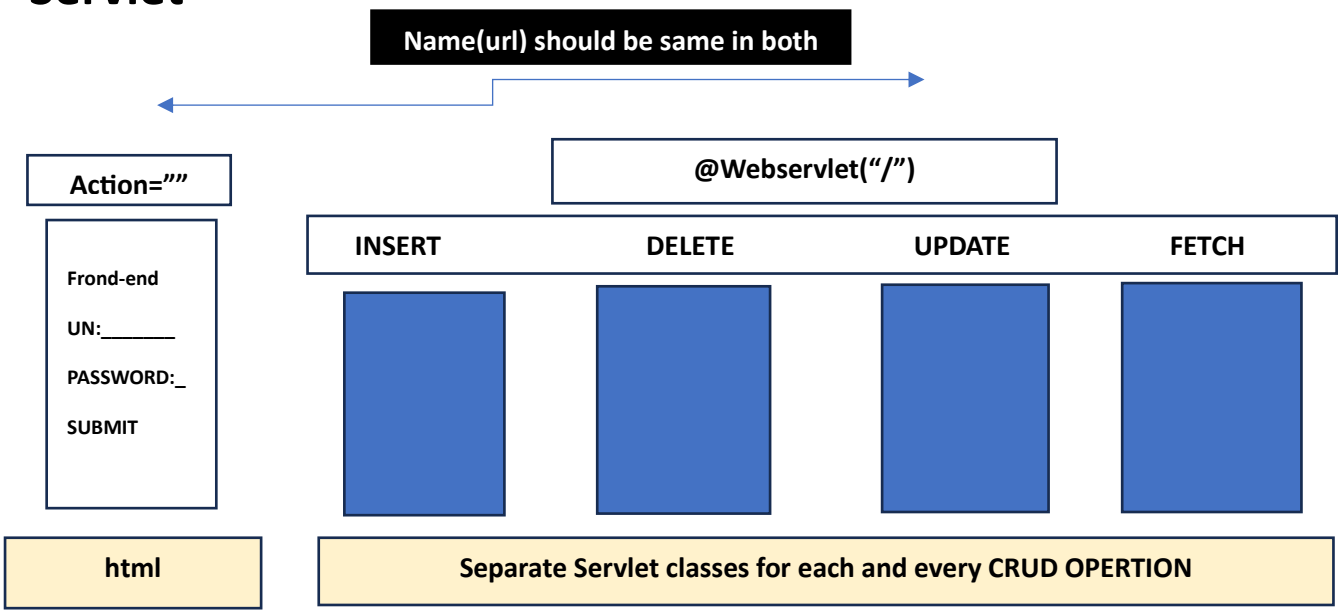
2) Web.xml

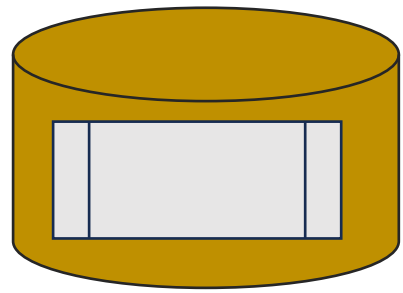
Inside a web app tag( )

```
<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <servlet>
  <servlet-name>ds</servlet-name>
  <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
  </servlet>

  <servlet-mapping>
  <servlet-name>ds</servlet-name>
  <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

FLOW_DIAGRAM

Servlet





database

1) action=""

2) @WebServlet("/ ")

Name(url) should be same in both

3) Database

Spring MVC

@CONTROLLER

```
@Controller
public class Doctor
{
@ResponseBody
@RequestMapping("/insert")
public String insert()
{
        System.out.println("data is inserted");
        return " data is inserted ";
}
@ResponseBody
@RequestMapping("/delete")
public String delete()
{
        System.out.println("data is deleted");
        return " data is deleted ";
}
}
```

ACTION=”URL”

Front-end

UN:_____

LN:_____

PASSWORD:_____

EMAIL_ID:_____

PHONE NO:_____

SUBMIT CANCEL

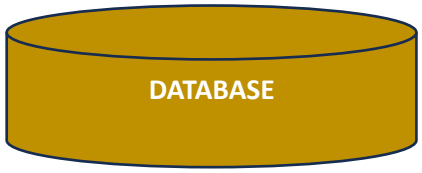
FRONT_END

SINGLE CLASS

MULTIPLE METHOD ()

MIDDLEWARE

```
<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <servlet>
  <servlet-name>ds</servlet-name>
  <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  </servlet>

  <servlet-mapping>
  <servlet-name>ds</servlet-name>
  <url-pattern>/ </url-pattern>
  </servlet-mapping>

</web-app>
```



Note this:

To check the servlet class name is perfectly placed or not :

Do this [cntrl+(move a cursor towards)servlet class name]

If it is blink as a link than consider it is working properly

If it is not:

Do this: add a dependency in the POM.xml

Exactly after the <dependencies> </dependencies> tag

Now< definitely it's going to blink 😍

//

WORKING OF SPING MVC

1) <ACTION="">

2)WEB.XML

4)Hore-gere (only j-spider KIRAN SIR students understood)

3)DISPATCHER SERVLET (It is a inbuild class with extending of Spring-framework)

4)we have to create a one more xml file

- **Inside a web-app not web-INF**
- **File name should be lowercase**
- **(web.xml)Servlet name and xml file name should be same with the extension of ds-servlet.xml**

5) add dependency sir has given the code

6) their we have to give the PACKAGE-NAME

7)then control go to the @ controller class

8)their it will mapping by the URL with a particular method()

- **For mapping we have to mention a annotation as @RequestMapping("/insert")**
- **Exactly above the method declaration**

9) If we want to see a output In the browser means :

- **Mention a annotation as @ResponseBody**
- **Exactly above the method declaration**

Note:

In servlet we have to create a object for setting a DTO class

But in spring MVC it will atomatically create a object of DTO class to set a data given by the front-end.

To achive this have to give the same name as given in the DTO class attribute to the JSP name attribute than only it will set the value of a particular variable;

Ex: private school; [DTO]

<input type="text" name="school"> [jsp]

important

But in Spring MVC

HTML file is not going to support: only a JSP FILE we are going to use

Example : scenario is given below to understood

Before that add the dependency of HIBERNATE MYSQL

Dto

- **Create a new package**
- **Create a new class , name as dto**
- **Declare a global variable**
- **Provide a getter setters**
- **@GeneratedValue(strategy = GenerationType.*IDENTITY*)**
- **Automatically generate a id number**
- **@Entity**
- **@ID**

JSP

- **Create a jsp file in web-app**
- **Write a form**
- **[for validation new attribute is there that is " pattern=" [A-Za-z0-9]+ " "]**
- **+ indicate more than one character should be mentioned.**
- **And enter the url to mapping**
- **<form action="insert">**

@CONTROLLER CLASS

- **@controller**
- **@**`@RequestMapping("/insert")`
- **@ModelAttribute (automatically create a object of a DTO to set a value)**
- **Ex:** `public void insert(@ModelAttribute DoctorDto d1 )`
- **Using this we can get a data of a DTO members**
- **To check just use a System.*out*.println(d1.getName());**

Insert data into database

Database

- **Create a database**
- **Give a connection through persistence.xml**
- **Src/main/web-app/WEB-INF**
- **CREATE a folder as META-INF**
- **INSIDE CREATE A PERSISTENCE.XML**

Front-end

- **Create a jsp file**
- **Inside a web-app**
- **<form> </form>**

Middleware

- **@controller class**
- **to set a data use**
- **@ModelAttribute (automatically set a data into dto)**
- **To check received or not use one sys.out statement**

Ex:
```
public String insert(@ModelAttribute DoctorDto d1 )
 {
//System.out.println("data is inserted");
   //   System.out.println(d1.getName());
```

Now to insert into database

- **Create a dao**
- **Give a object to controller class**
- **@compound and @AuotWired**

- **Create a reference variable dao class**

Ex: @Autowired

- Doctor_Dao doctordao;
- **Using this reference variable pass the dto data to dao class uing a method and perform a database operation**

Ex: controller class

```
@Controller
public class Doctor
 {
  @Autowired
  Doctor_Dao doctordao;

  @ResponseBody
  @RequestMapping("/insert")
 public String insert(@ModelAttribute DoctorDto d1 )
  {
  String msg=doctordao.insert(d1);
  return msg;
        }
```

- **Ex: DAO CLASS**

```
@Component
public class Doctor_Dao
{
EntityManagerFactory entityManagerFactory=Persistence.createEntityManagerFactory("dev");
EntityManager entityManager=entityManagerFactory.createEntityManager();
EntityTransaction entityTransaction=entityManager.getTransaction();

public String insert(DoctorDto dto )
{
  entityTransaction.begin();
  entityManager.persist(dto);
  entityTransaction.commit();
  return "data is inserted";
```

- }

Flow: vvvvvvvvvvvvvvv important

- ➔ **Front-end**
- ➔ **Web.xml**
- ➔ **/(dispature class)**
- ➔ **Ds-servlet.xml (for package sacanning)**
  - **We can scan in 2 ways**
  - **1) writing aa similar name in all present pakage**
  - **2) individually paste a qualified name of package**
- ➔ **@controller class**
  - **@requestMapping(“/insert”)**
  - **@ModelArtified dto class**
  - **@compound and @AutoWired Dao class**
- ➔ **DTO**
- ➔ **DAO**
- ➔ **DATABASE**

Delete a single row in database

Front-end

- Jsp

Middleware

- @controller class
- Receive an argument
- Pass to DAO class
- Check by primary key
- The data is present or not
- Using a find method()
- Use condition
- Return back the result for configuration to the end user

➔ Same working flow as insert (refer) 🤓

Delete All the record present in the database

Front-end

- No need to write a jsp file

Middleware

- @controller class
- Call a method present in the DAO using @AutoWired reference-variable
- Before deleting all the records have to fetch
- So, use a Query interface
- And store it using a getREsultSet(); method
- Return type is List<> generic class
- Than , use a condition
- To delete one by one use a for each loop
- Return back the String for configuration to the end user

Fetch the single entry from the database

Front-end

- jsp file

Middleware

- @controller class
- Receive an argument (@REsponseBody)
- Pass to DAO class
- Check by primary key (find())
- The data is present or not
- Using a find method()
- Use condition
- Return back the result for configuration to the end user

Fetch all the entry from the database

Front-end

- No need to write a jsp file

Middleware

- @controller class
- Call a method present in the DAO using @AutoWired reference-variable
- For fetch all, use a Query interface
- And store it using a getREsultSet(); method
- Return type is List<> generic class
- Than , use a condition
- Return back the Object for configuration to the end user
- DAO 👇

```
public List<DoctorDto> fetchAll()
{
        Query q=entityManager.createQuery("select a from DoctorDto a");
        List<DoctorDto> multiple=q.getResultList();
        if(multiple.isEmpty())
        {
                return null;
        }else
    return multiple;
}
```

Display the data in the table structure

In servlet we use a RequestDispature and resp.sendRidirect("facebook.com") for forward the data to another file to display in a table format

But: " here ",

In Spring MVC we use a class to forward a data to another file to display in a table format.

```
ModelAndView view=new ModelAndView("download.jsp");
        view.addObject("objects", msg);
        return view;
```

the return type of view is ModelAndView

while recieveing in the download file everything same as servlet process create a table <th> <% downcasting> <% for each loop>

<%= get item> that's it😎

To add a column ..

**Same as <th> <a href=" url ? (where) pk=<%= get item> "> **

//

@ Spring IOC

@Component

@AutoWired

@Configuration

@@ComponentScan(basePackages="ioc_annotation")

@Primary

//

@ Spring MVC

@Controller

@ResponseBody

@RequestMapping("/insert")

@ModelAttribute DoctorDto d1)

@RequestParam -> name attribute and local variable of a argument name should be same ex: public void fetch(@RequestParam int pk) === < ….name=”pk”>

@PathVariable

//

@ Database related

@Entity

@Data (Dependency Lombok)

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

@Column(nullable=false,unique=false)

Maven project check box recall:

1) **Jdbc and hibernate** **“ Tick ”**
2) **Servlet** **“ No Tick ”**
3) **Spring IOC** **“ Tick ”**
4) **Spring MVC** **“ No Tick ”**