# JDBC Connection

## 5 steps to connect java to DB:-

1. Load the driver
2. Establish the Connection
3. Create Statement
4. Execute
5. Closed Connection

## Steps in Elaborate:

### Load The Driver:

* Loading the Driver will done by two way but before that we need upload the “MySQL CONNECTOR” JAR file into our project
* Go to any browser and search for “MVN Repository”
* An web page will open, On that Repository web page a search bar will present, So search “MySQL CONNECTOR” on the search bar
* Another page will open, on that page select second option of “MySQL Connector Java”
* Again it will redirect to another page, Scroll down on the page we can see the version of “MySQL CONNECTOR” and select the highest downloaded version, it will open the respective version page, on that page…..
* After download the JAR file go to project right click on project select “Build Path” option and “Build Path configuration”
* An Interface tool box will open, on that

# Servlet Application

**Due to Documentation Process the method names and class names are not followed the convention but mentioned as method**

* What is Web Application
* Type of Web Application
  + Static Web Application
  + Dynamic Web Application
* Dynamic Web Application have two types
  + User Dependent
  + User Independent
* Print-Writer is a Class which extends Writer

Questions Oral Test -06-04-2-24

(

How to create servlet and explain ways

Diff b/w servlet and Generic servlet

Explain Print Writer and how will get Print Writer Object

how to map R-URL with servlet class

)

* Servlet Tag always should be in the Top(top means above the servlet mapping tag) in web.xml file

## Query Parameter: 06-04-2024

* Send data from Frontend it is (Request)
* Send data from Backend it is (Response)
* Resource URL-->https://localhost:8080/ProjectName/URL-Pattern
* Query Parameter-->?key=value&key=value
* req.getParameter("Need to give the keys exactly what we given in query Parameter on URL"):
  + req means (Servlet Request Class)
  + Fetch the data from query parameter
  + The Return Type is String, so store in String type container.

Assignment: A task has been given: -

1. Send data from front end (Using R-URL & Query parameter)
2. Receive from server
3. Save on database

## Servlet Life Cycle: 09-04-2024

* Clients System🡪Request🡪Web Container 🡪(URL Pattern (Inserted on Web.xml file using servlet tag))
* If URL Present 🡪Servlet Life Cycle will start
* If URL Not Present🡪404 Page Not Found Error

**Servlet Life Cycle:**

1. Servlet class Loaded
2. Servlet Class instance created
3. in it method will be invoked:-

* to initialize servlet
* this method will execute first
* Run for only one time
* implicitly object is created

1. Service method will be invoked

* Run for nth time according to Requirement

1. Destroy method will be invoked

* One object is not used then that object is garbage collected
* When destroy method is called when ……the object is collected

Note:-These All Statement is done by the Web Container.

## Servlet Context: 10-04-2024

1. It is used to pass the message or information to all the servlets
2. Another name for “**Web.xml**” is “**Deployment Descriptor**”
3. **Deployment Descriptor** used to Map the URL with Servlet class
4. Use this tags as a first instruction(context-Param) on web.xml

<context-Param>

<Param-name>variable-name</ Param-name>

<Param-value>Data to pass for all servlet</ Param-value >

</context-Param>

1. And Map the context class with mapping tag
2. Enter these Tags in Web.xml(in-between web-app tags as first line)
3. Use below Statement for Create an Object for ServletContext Class on server code page

ServletContext context = getServletContext()—

Context.getInitParameter(“Param-name”)-Used to get the value from context parameter.

1. How the above statement is working without Object refe …..

## Servlet Config:- (Similar to context but one)

1. It is used to pass the message or information to the particular servlets
2. The Below code is written on the server code file

ServletConfig config= getServletConfig();

String email = config.getInitParameter("email");

PrintWriter out = res.getWriter();

out.print("<h1> Email = "+ email + "</h1>");

1. The Below code insert into web.xml:

<servlet>

<servlet-name>c2</servlet-name>

<servlet-class>servlets.Config2</servlet-class>

<init-param>

<param-name>email</param-name>

<param-value>velan@gmail.com</param-value>

</init-param>

</servlet>

## HTTP Servlet: Absent on this day 10-04-2024

(Notes Referred from friends)-

* Extends HTTP servlet class
* Out .print—>Statement only take h1 tag
* Default method is get method so we need to overwrite with “DO Get” Method
* @Web Servlet(“/class name”)

## Servlet Chaining: 11-04-2024

Disadvantage of Generic Servlet:

…….Need to write

Advantage of HTTP Servlet:

It can accept any type of protocol(Not clear), Like

* HTTP
* HTTPS
* SMTP
* FTP
* SSL

### Servlet Chaining:

* One servlet is calling another servlet.
* Request share from one to another [or] request processed through multiple servlet in a chain is called as Servlet Chaining.

We can achive two way:

1.Req Dispatcher

2.Send Redirect

1.Request Dispatcher🡪Is used to dispatch request one servlet to another (like calling)

We have two type R/D:-

1. Forward 🡪 used forward the response in next page

2. include 🡪used include the response in same page

Here naming convension

Variable Name should be like statements class name or interface name

Servlet chain code

1.Forward

Sevlet 1 code

@WebServlet("/servlet1")

public class ServletChainning1 extends HttpServlet {

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

RequestDispatcher dispatcher= req.getRequestDispatcher("servlet2");

dispatcher.forward(req, resp);

}

}

Sevlet 2 code

@WebServlet("/servlet2")

public class ServletChainning2 extends HttpServlet{

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

PrintWriter out = resp.getWriter();

out.print("<h1>Welcome to Servlet 2</h1>");

}

}

2.Include code:

Servlet to servlet

Servlet to html

Servlet to jsp

@WebServlet("/LoginWelcomePageLogin2")

public class LoginWelcomePageLogin2 extends HttpServlet{

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

String u\_email=req.getParameter("email");

String u\_pass=req.getParameter("pass");

String db\_email="Santhos@gmail.com";

String db\_pass="123";

if (u\_email.equals(db\_email) && u\_pass.equals(db\_pass)) {

RequestDispatcher dispatcher = req.getRequestDispatcher("welcome");

dispatcher.forward(req, resp);

} else {

PrintWriter out = resp.getWriter();

out.print("<h1>INVALID EMAIL PASSWORD</h1>");

RequestDispatcher dispatcher = req.getRequestDispatcher("login2.html");

dispatcher.include(req, resp);

}

}

}

## Send Redirect: 13-04-2024

sendRedirect is a Method

send redirect is usd tp redirect the responcse from one application to another

Attribute:

Used to send(Tranfer) the object from one servlet to another servlet

Types:

1.Context Attribute

2.Request Attribute

3.Session Attribute

Methods:(Below methods is common for all three types)

setAttribute(String key, Object o)—used to set the object

getAttribute(String key)—

removeAttribute(String key name)—it will remove the name

Scope:

Context 🡪throughout out Application

Session 🡪From login to logout(in between Login and Logout)

Request🡪Throughout the circle were it pass and get (Wereever the Request has been shared)

Page Scope🡪With the Class (Example of Private)

## : 15-04-2024

* HTTP is a Stateless Protocol
* Stateless means Each and every request is consider as new request.(this is disadvantage)
* Stateless Protocol as a disadvantage to overcome we need to convert as State full protocol
* Using “Session Tracking” Technique we can overcome the problem
  + URL Re-Writing
  + Hidden form field
  + HTTP Session
  + Cookies
* State full protocol means considering the request as an old request [or] old user
* Cookies means a Small piece of information
* Cookie is a Class which is present in javax.servlet.HTTP
* Cookie’s will store on the browser
* Create object for cookie class using parameterized constructor which having 2 string argument that is key and values (for example we pass one for email another for password)
* “resp.addCookie” method is used to add cookie and pass the cookie class object reference as argument.
* Finally add web servlet tag on above method.
* One default cookie will be added at sometime

## Fetch Cookies: 14-04-2024

* Use the “get cookie” method from “request” (req.getCookie).The return type is cookie array
* Get cookie methed take all cookies from browser and send as in array form
* Get name method to fetch name(key) present in response
* Get value method to fetch value(Ex:-resp.getValue)

Types of Cookies

Non persistence(By Default all cookies are non persistance)

Once browser closed it will gone-temporary

Persisitance cookies-

To make the cookies as persisitance need to use setMaxAge(60\*60\*24\*30\*6) method

Httpsession

3 methods

Create session

Validate session

Invalidate session(doble)

To create session--HTTPSession session=req.getSeesoin()

Req,getsession is a overloaded method

Req.getSession(true)-if it reurtns true it will create seassion if false it will give presvious session

18

22

Life cycle of jsp

7 stages

1.Transulation of jsp

Demo.jsp---🡪demo\_jsp.java

Jsp file converted in to java file

2.compliation of jsp

Java file will be converted in to .class file

Demo.java----🡪demo\_jsp.class

3.servlet life cycle will load(from 3rd stage to 7th stage total 7 stages)

Jsp\_init()

Jsp\_Service()

Jsp\_Destroy()

Object scope:--- scope means visibility level

Context scope🡪any in an app

Request scope🡪between req and respeonces

Session scope🡪b/w login and logout

Page Scope🡪within the page

Implicit 9 object will created when jsp file craeted

1.jsp writer

2.http servlet request

3.http response

4.Servlet config

5.Servlet context

6.http session

7.Object class super most parent class

8.page context

9.Throwavble

Reference variable for above object

Out

Request

Response

Config

Application

Session

Page

Page context

Exception

## Small project on servlet: 24,25,26-04-2024:

On jsp file we cannot use a tag inside another tage to over come this problem to go for concat

Below code shows thst if block is opem in one tage and closed in another tag

<%

String message=(String) request.getAttribute("msg");

if(message != null){

%>

<h1><%=message %></h1>

<%} %>

# HIBERNATE

## Introduction: 29-04-2024:

* Note:-Absent on this day for Beta-Monks interview
* Only Hibernate Introduction has been happened

## Introduction: 30-04-2024:

### What is framework?

Take notes from java part 3

### Why we go for Hibernate:

* We go to Hibernate to overcome the dis-advantages of JDBC
* Dis-Advantages of JDBC:-

1. We cannot pass another object for Already created object (like if we created is student object but if we pass employee object it won’t accept)this is disadvantage
2. Possibilities of value mismatch🡪we can pass same string value for different two columns (like inter-change) which can accept string as input. It won’t show error because both column will accept the string

### Hibernate:

* Hibernate is an open source light weight frame work used to develop a java application on interacting with database
* Hibernate is an ORM tool and Hibernate will implement the specification of JPA
* We did not learn Hibernate we learn Hibernate with JPA and it is a specification

### Hibernate with JPA (Java persistence API):

* Why we learn hibernate with JPA:
* In feature, if I want to switch one framework to another framework with all our old code, it will be easy. For this purpose we learn hibernate with JPA
* JPA is a Rule given to ORM tool
* ORM-Object Relational Mapping (ORM common for all the programming language)
* Object🡪is nothing but JAVA OBJECT and Relational🡪stands for RDBMS and Mapping🡪is Connection
* The connection between the java object and database is known as ORM
* Here the class is a Table[In DB] and Attributes is a column[In DB]
* ORM is a programming technique where one object will be mapped with one row in the table

### Hibernate project creation:

* New–>Project—>Maven project—>click create simple project (skip archy..)—>Enter Group id and Artifice id(Package name and Class name)
* Go to POM.xml file🡪Create “<Dependencies>< Dependencies /> “tag
* Open any browser search “MVN Repository” an website will open
* And search for My SQL connector on search box and select second option which is
* Observe highest download number on any version and click the version
* It will redirect to another page, then scroll down and copy that Dependencies tag codes and past it in inside dependencies tag(we created) on POM.XML file
* Again search for “Hibernate core” and choose “Hibernate core Relocation”🡪Observe highest download number on any version and click the version
* It will redirect to another page, then scroll down and copy that Dependencies tag codes and past it in inside dependencies tag(we created) on POM.XML file
* Save the file and check maven dependencies(check the “dependencies” spelling is correct) are added in the project
* If any error occurs update the project--> Finally update the maven project
* Then mouse over on SRC/main/resource file—>right click🡪New folder---->Given name of “META-INF” to the folder—>Then mouse over on newly created folder🡪Right click on folder🡪Select New file–>Select Other and search for xml—>click xml file🡪Give name as “persistence.xml”
* And copy the code given by the teacher(SURIYA Sir on group)then past it on the “persistence File” Save it(if persistence is open in the form of table got to source..ned to check the db name )
* If any error occurs update the project--> Finally update the maven project

**Setuping Project: 01-05-2024:**

* On the persistence xml file we only change the DB name(User created DB)
* Hibernate will work only when the table have any primary key(hibernate will work based on primary key logic)
* Hibernate create the table implicitly by the logic of “ONE OBJECT IS ONE ROW”
* Don’t use special characters while creating object blue print
* Because one object is save as one row in DB. So variables of an object will be the column name for the DB
* For this reason we can’t able to provide special characters in the variables and here we need to follow the method name convention for variable (only for this scenario)

### Annotations:

* Hibernate will create table for “Entity class”
* Annotation are nothing but indication to the compiler—>we use annotation above the data members
* On which data member the annotation rules to be applied for that we need to mention the annotation above the specified data member(Variable or methods)
* Hibernate interfaces and classes are present in the package of JAVAX.PERSISTENCE.\*;
* @Entity:
* Entity annotation is a class level annotation and the purpose of this is to make the class as entity class because hibernate will create table for entity class
* @Id:
* ID is the annotation used to make the variable as primary key
* If we mention this ID annotation in above of any variable then that below variable consider as primary keyed variable
* The above two annotation are must want to create(Mandatory) for performing with hibernate

### Creation of Class:

* Entity Manager-it’s a interface—used to perform all the database operation
* Create a Blue Print with Annotation and all variables with private key word along with getter and setter method.
* And Create another class file with the class name of what the function we need to do with DB(if we need to save the data to DB then name the file as Save-Hibernate)
* After creating the class create main method and use below predefined method of hibernate classes

1. Persist()-used to save
2. Find()-used to fetch data based on primary key
3. Remove()-used to delete
4. Merge()-used to update

* These methods are present in “Entity-Manager-Factory” interface. To use these above method we need to create object for interface class with implementing the interface class
* Entity Manager Factory used to establish the connection between java and DB. Based on persistence unit name which is basically “DEV” which help us to create an object of Entity Manager interface and Persistence class is the implementation class for entity manager factory

Persistance.craeteEntityManagerFactory(persistence unit name)

Emf.createEntityManager();

EntityTransaction et=em.getTransaction();

EntityTransactoin is usd to manage the transaction between java and db

First of all we need to start the transaction then do all other process—et.begin –to start thr transaction

At last use the et.commit method

To Fetch the data create the class for Fetch

To fetch the data Entity transaction is not needed

Em.find(User.class,1);--return type is specified class(Based on entity class)

User u=em.find(User.class,1);--if id is present it will return object or it will return null

**JPQL: 02-05-2024:**

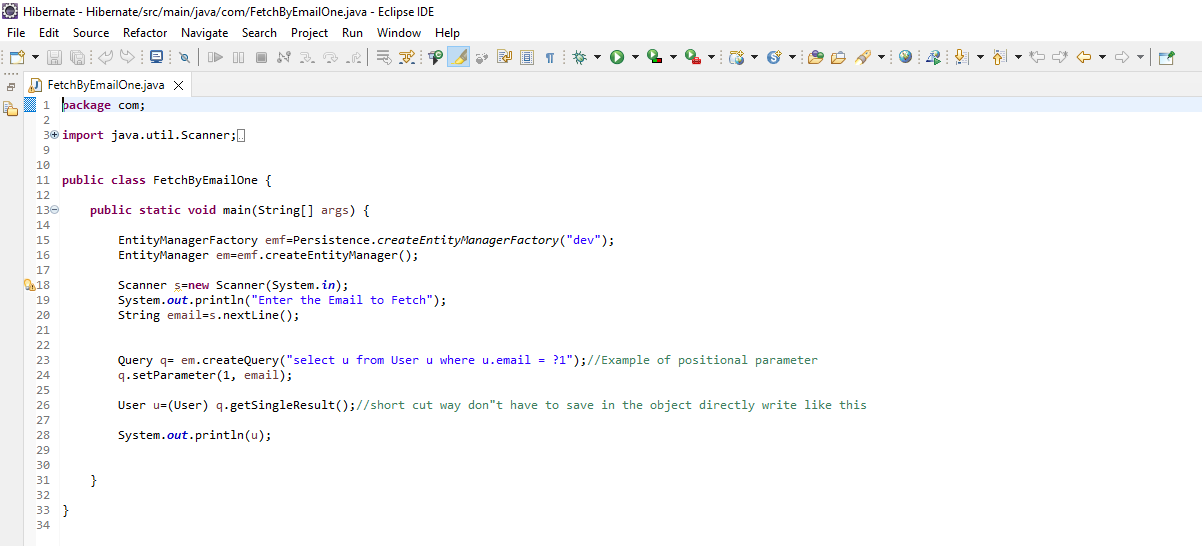
* JPQL🡪JAVA PERSISTENCE QUERY LANGUAGE
* It is an OOPS and Platform independent Query Language
* It is DB independent
* SQL is fully based on Tables and Column Names
* But JPQL fully Based on Entity Class/variables
* If we want to use other methods (Except above 4 methods), Other than above mentioned four method, we need to use JPQL with respective methods.
* **Syntax for JPQL:**
* For SQL🡪select \* from EMP;
* For JPQL🡪select u from User u;
* Here the u is alias of User and in the place of \* we need to give the alias name
* **Syntax for JPQL to specific:**
* For SQL🡪select ENAME from EMP;
* For JPQL🡪select U.ENAME from User u where U.ID;
* Here the u is alias of User and in the place of \* we need to give the alias name
* Use CREATEQUERY method from Entity Manager and Return Type is Query.
* **Example Program:**
* EntityManagerFactory emf=Persistence.createEntityManagerFactory("dev");
* EntityManager em=emf.createEntityManager();
* Query q=em.createQuery("select u from User u");
* To know any methods return type, we have short cut key word of control + 2………
* List<User> l=q.getResultList();
* for (User u : l) {System.out.println(u);}

## JPQL: 03-05-2024:

* To fetch Single data use the method of “query . get Single Result”
* To fetch Multiple data use the method of “query . get Result List”

## Parameter: 04-05-2024:

* Two Types of Parameter
  1. Positional Parameter
  2. Named Parameter
* **Positional Parameter**:
* Giving the position number after the placeholder is known as position parameter Ex:-**select u from User u where u. email=?1**
* In Hibernate to set the value to the placeholder we have a method called “**q. set Parameter**”(q is query) which will take 2 argument Ex:-**q. set Parameter(1, email)**
* We can use this method for all data types
* **Example Program:**

****

### *Named Parameter:*

* Same as Positional but a small change in placeholders
* Instead of (?) we use (:) followed by any name
* And while using set parameter method need to use the named parameter in the form of string like mentioned below



## *Mapping: 06-05-2024:*

* Another name for **Is a Relationship** is “**Inheritance**”
* Another name for **Has a Relationship** is “**Association**”
* Association means [one object mapped or Associated with another object is called as mapping in hibernate]
* **One to one** 🡪 One object Associated with another only one object is called as one to one mapping
* **One to Many** 🡪 One object Associated with many object is called as one to many mapping
* **Many to one** 🡪 Many object Associated with another only one object is called as Many to one mapping
* **Many to Many** 🡪 many object Associated with many object is called as Many to many mapping
* For using this operation 3 annotations is mandatory
  1. @Entity
  2. @ID
  3. @ManyToOne or OneToOne or OneToMany or ManyToMany
* ? create Database If Not Exist = true

## Remove Both: 07-05-2024:

* If we try to remove the child row from child table it will throw an error, because there was a connection between parent and child
* But we can delete parent row
* So if we want to delete the child row either we need to delete the parent class with whom it mapped or break the relation between child and parent
* **Short cut key🡪**Control shift o for auto import
* **Short cut key🡪**Control 2 L for auto return type
* **Short cut key🡪**Control shift A and Control shift F for auto align the code
* Two type of direction
* One to one 🡪UNI-Direction[Both advantage and disadvantage]
* One to one 🡪Bi-Direction

## One to Many BI-DIRECTION: 07-05-2024:

* @Join Column insert after one to one annotation
* @One To One (mapped By=”object reference”)

## Many To One: 08, 09-05-2024:

* Just Program No theory

## Many To One: 10-05-2024:

* Just Program No theory

## Cascade: 11-05-2024:

* Any action taken for Parent class it will reflect on child class also
* If we Delete Parent class child class also delete which whom it mapped using mapped attribute, if we save parent class child class also save
* To use this we need to insert the below case cade attribute
* @One to Many (mapped By ="batch", cascade = Cascade Type. ALL)
* private List<Student> Student;
* The process of loading or fetching the data is known as Fetch type
  1. Lazy
  2. Eager

### Fetch Types:

* Lazy🡪we try to fetch the parent class then only parent class data will receive means it is known as Lazy
* Eager🡪Along with parent child data also receive means it is Eager
* Default🡪When we did not mention fetch type default will work
* The Default fetch type for one to one is eager
* The Default fetch type for one to Many is lazy

## Catch Memory: 13-05-2024:

* It will store the data received from the DB and give result to UI or To Client or To browser.
* Two Types of catch memory are there

1. First Level:

* It is an Entity Manager Level memory
* It Enable by default

1. Second Level:

* By default it not enable, we need to enable explicitly
* It is common for all the entity managers

# *SPRING*

## *Introduction of spring: 17-05-2024:*

* Spring Frame is a Frame work of Frame works which not maintained by oracle, Because spring is the 3rs parties application implementing java
* Before spring “EJB” was used after that all converted into spring.
* The full form of EJB is “Enterprise Java Bean” and it is a tightly couple application, if one changes made on any one object it will affect all the object to overcome this problem we go for spring
* EJB is following the “**Has-A-Relationship**” on Composition type🡪that is one object is completely depend upon another object.
* But **spring** is the “loosely couple” application framework

## Setup of spring: 18-05-2024:

* Create Maven Simple project(Tick--skip archer and create simple project)
* Create dependencies tag on pom.xml file. After search on browser for “MVN REPOSITORY” and search for **spring context** and check for higher download without any issue history and copy the dependencies and paste it on pom.xml file inside dependency tag close the with dependencies tag
* Force Update the project
* Using spring we can develop an “Loosely Coupled Application”
* The man feature in spring is “**IOC**”. The Full form of IOC is “INVERSION OF CONTROL”. Inversion means **Reverse**
* The process of transferring the control of object creation from user to spring frame work is known as ……

### Spring Container:

* Two type of spring containers are present. Both are “**Interface**”.

1. Bean Factory
2. Application Context

* Here **Bean Factory** is like a **Parent** [old] and **Application Context** is like a **Child** [New]
* Application context has a 4 sub class [Context].

1. Class Path XML Application Context
2. Annotation Conflict Application Context
3. Web Application Context
4. File System Application Context

* To connect the “user spring container” and object we need configuration file
* We can create in two ways

1. By XML file creation way
2. By Annotation creation way

* Object another name in spring is “Bean”

### Configuration by xml way

* Create configuration file on SRC/main/resource🡪config.xml and paste the code sir shared on group

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

<http://www.springframework.org/schema/context/spring-context.xsd>">

</beans>

* In CONFIG xml file we can create object by using <Bean ></Bean> Tags. These tags want to present inside the <Beans ></Beans> tag
* Create these attribute on bean tag

1. Id=”any id or reference variable name for object”
2. Class=”fully qualified class name”



## Accessing Non Static Method: 20-05-2024:

* Factory Return Pattern – >Based on user requirement it will return Object
* We can access the non-static method of blue print using “**Get Bean (“Object reference id [mentioned on bean tag]”)**” method
* The Return type of the get bean method is object, so we need to type cast or downcast and save in blue print object name
* **Dependency Injection**: The Process of Injecting one object into another object and Injecting the value for dependencies is known “Dependencies Injection“[setting the value to the variable is called Injecting the value for dependency]
* Dependency is nothing but Data members
* We can inject the value for dependency in two ways[dependency injecting in two ways]

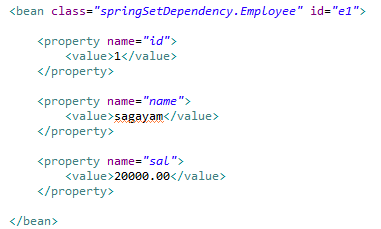
1. By Constructor Injection
2. By setter Injection

### Setter Injection:

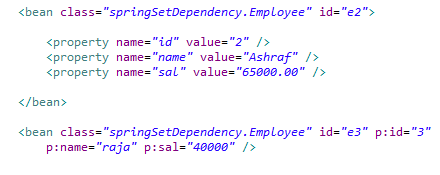
* We can Inject the value in two way using setter injection
  + 1. Value as a tag inside the Property tag
    2. Value as an attribute inside Property open tag
    3. Property schema method
* **Value as Tag**:- On configuration file we need to create the tags

1. Property tag
   1. <property name=”variable name ”></ property >
2. Value tag
   1. < value >value to inserted here for variable mentioned on property tag</ value >

* Value tag need to insert inside the property tag
* Example Program:-



* **Value as Attribute & Property schema:**

****

## Batch or Collection dependency: 21-05-2024:

* Value tag or attribute will accept only primitive data types, on Non primitive only string will accept.

## Object Dependency Injection &

## Constructor Dependency Injection 22-05-2024:

## Constructor Schema & Component Class & Packaging Structure 23-05-2024:

## Annotation Context & Value as Annotation 24-05-2024:

* Auto Wired Annotation

## Annotation Context 25-05-2024:

* Primary Annotation
* Qualifier Annotation
* Bean Annotation

## Annotation Context 27-05-2024:

* Collection Method Bean Creation
* @Scope Annotation

## Scope of Bean:

1. Single Ton Design Pattern🡪By Default all Bean is single ton
2. Prototype🡪Used to create multiple object
3. Session
4. Request
5. Application

## Life Cycle of spring:

1. Spring container will be started
2. Instance created
3. Dependencies will be injected
4. Custom (in it) method will be invoked
5. Custom method () will be invoked
6. Custom destroy method will be invoked

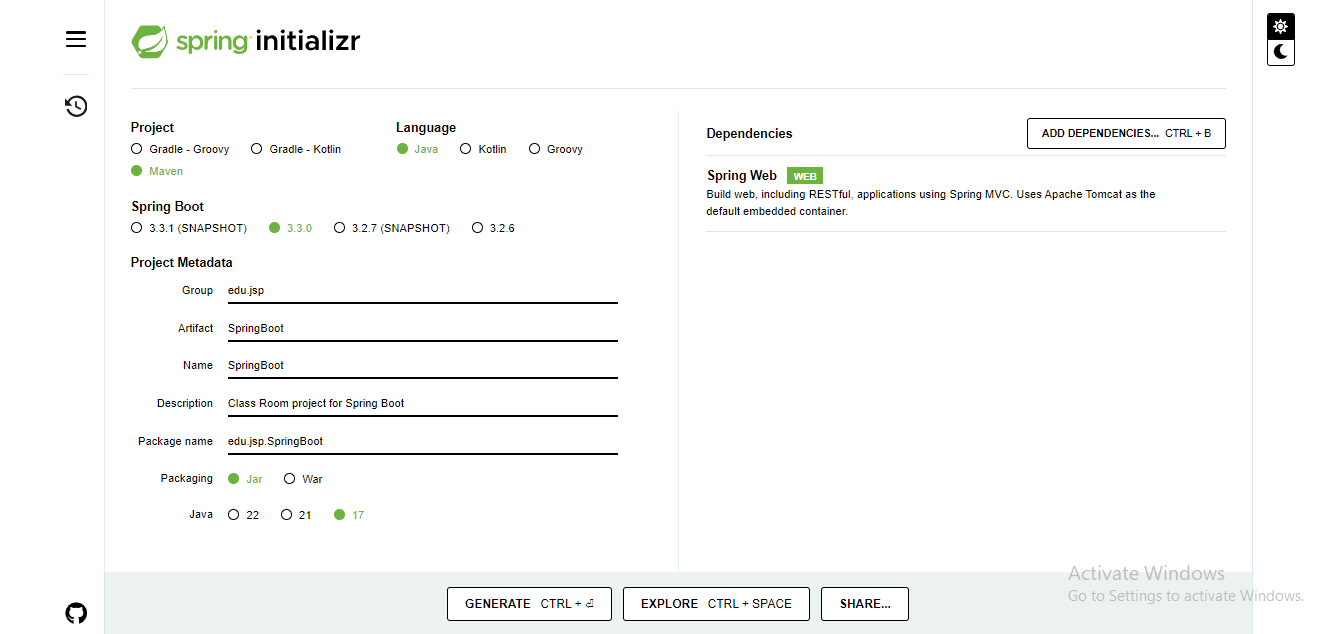
## Annotation Context 28-05-2024:

* MVN Repository🡪Search the word Annotation 🡪JAVAX Annotation API
* @Post Construct🡪to mention on custom in it method
* @Pre destroy🡪to mention on custom destroy method

# SPRING BOOT

## Setup and Introduction of Spring Boot 29-05-2024:

* Create work space for Spring Boot and open it
* Open browser and search “**Spring initializer**”
* And select the option like given below



* Project 🡪 Maven
* Language 🡪 Java
* Spring Boot Version 🡪 Select other than snapshot
* Group 🡪 Enter the package name
* Artifact 🡪 Enter the Project Name
* Packaging 🡪 Jar
* Java version 🡪 17
* Dependencies 🡪Below

🡪 Click Add Dependency Button and search “Web”

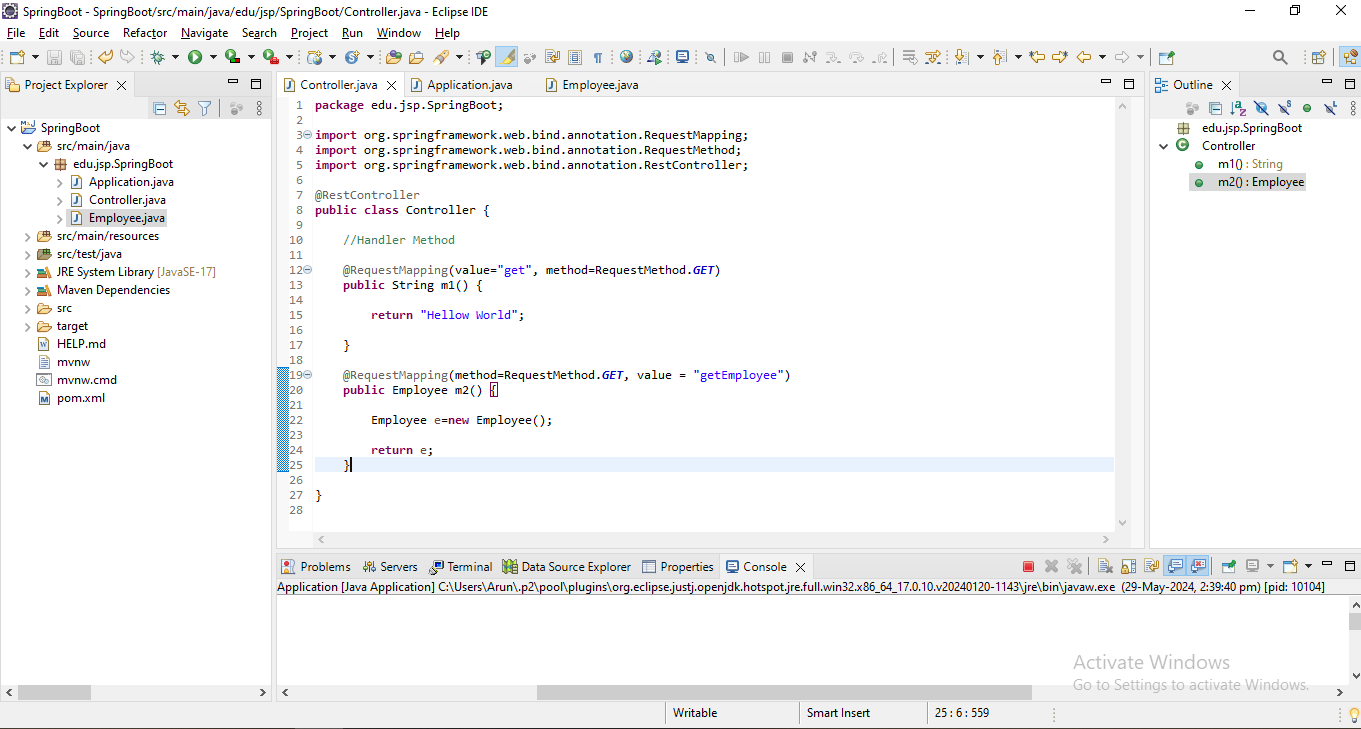
* Select “Spring Web” and Click “Generate” Button
* An jar file will download then go to download in our system
* Select the Downloaded file and extract it [A Folder will be Extract]
* Go to Eclipse and go File🡪Import🡪Search for Maven
* Select Existing Maven Project🡪Browse🡪Select the Extracted Folder and click Finish
* It will take time to import and Build Project. After Project creation done
* Update the Project🡪Select the project🡪Right Click🡪Maven🡪Update Project
* A Tool box will open, on that tick the “Force update” box and click ok

## Annotations:

* @Spring Boot Application is the Annotation which is the combination of 3 annotation. They are @Configuration, @Component scan and @Enable Auto Configuration
* @Spring Boot Application 🡪 To make the file as configuration file
* @Rest controller 🡪 To create an object or to make the class as component class
* @Request Mapping (value=”get”, method=Request Method. GET)
* @Request Mapping (value=”get”, method=Request Method. POST)
* @Request Mapping (value=”get”, method=Request Method. DELETE)
* @Request Mapping (value=”get”, method=Request Method. PUT)
* @Request Mapping (value=”get”, method=Request Method. PATCH)
* Verb is nothing but HTTP Methods

### Run the File:

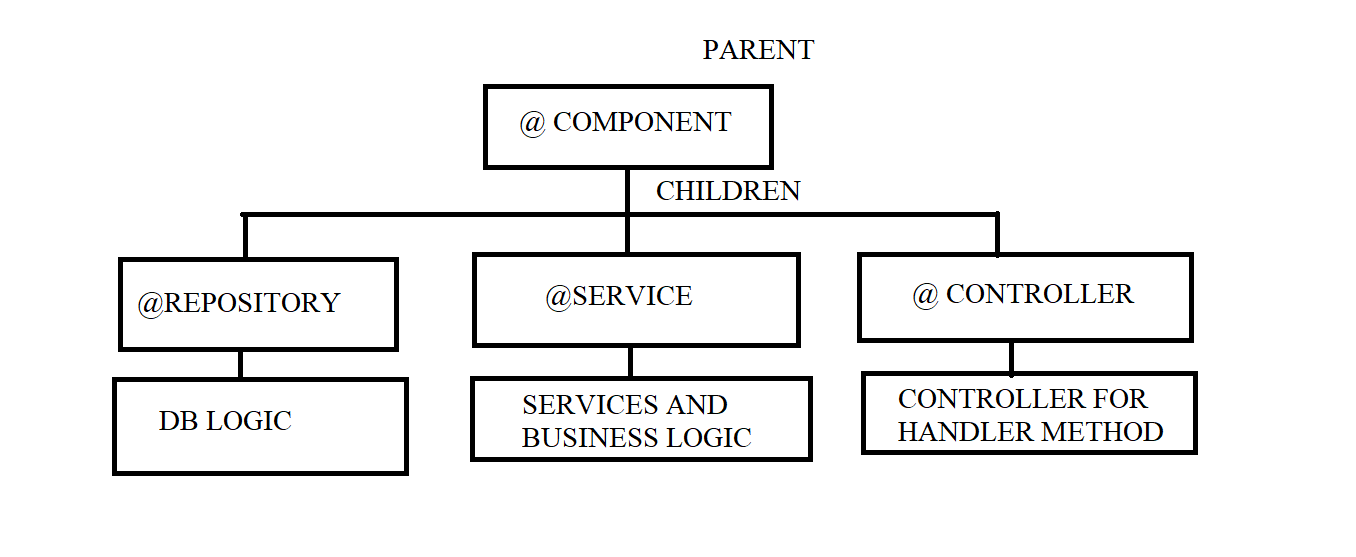
* To run the project create java file inside the project🡪SRC/main/java🡪package and give name as “Controller”. The controller java file image mentioned below
* Mention the controller class with @Rest controller annotation on above the class
* And write logics using methods and mention the method with the annotation of @Request Mapping (value=”mention the URL here (Example: https: //local host/**method Name (get Details)**)”, method= Request Method. GET)
* For Get process use (method= Request Method. GET)
* For Post process use (method= Request Method. POST)
* Do same for put, patch, delete
* Always run the project from the Application file which holding the main method
* After saw the in-build server running open the browser and search the URL mentioned on the server console



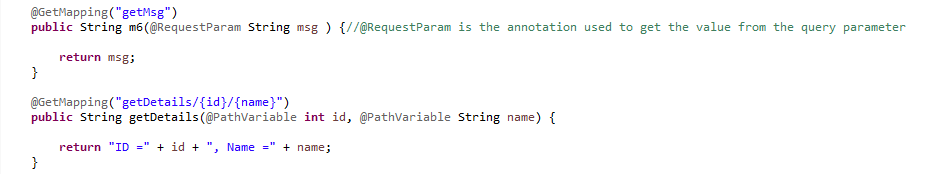
* To view the JSON object in a clear format add JSON chrome Extension

## Layers of Spring 30-05-2024:

* First Layer - DTO[Data Transfer Object]🡪 Consist of Entities
* Second Layer - DAO[Data Access Object]🡪 Consist of Data Base Logic
* Service Layer – Consist of Business Logic
* Controller class consist of handler methods
* Handler methods handle the web requests by method



* @RequestMapping
* @Get Mapping - It is used to map get request with the handler methods.
* @Get Mapping (“/URL or method Name”)
* @Post Mapping - It is used to map post request with the handler methods
* @Request Param: This annotation is used to get the parameter from the request (Front End)
* @Path variable: This annotation is used to get the data from the request



## Post Man Application Request Creation & New Project based on JDBC connection 31-05-2024:

* Dependency - >My SQL Driver[Advance version of My SQL Connector]
* Spring Web-to use front end methods get post
* Spring Boot DEV Tools—Auto restart server
* Spring Data JPA-To use hibernate

## P 01-06-2024:

* @Repository—Mention this annotation which java class holding the DAO concepts and DB methods
* @Service
* @Controller—it return view
* @RequestBody used to bind the json object with java object(used to convert json to java)
* @ResponseBody 🡪java to json 4

## Status code, Update ,Update All, find by attribute 03-06-2024:

* Refer the status code on post man and google it know better.
* @Rest Controller🡪
* 1.combination of @controller and @Response Body
* 2.it will return the domin

## Mapping One To One 04-06-2024:

## Operation on sub class 05-06-2024:

## Updating child class on one to many 06-06-2024: