Spring RVC based Layered Application

jsp --------> DispatcherServlet ------

(view)

logic)

(presenntation

(FrontController)

navigation

logic

-> controller

(controller)

(delegation

(MiniProject (dao)

--> service class (Mode layer) ↓

--> Repository----> Db s/w

(Model layer) ¯

(b.logic)

(persinstence logic)

logic +exception

handling)

Dynamic webpage

(MVC architecture based web application as the Layered App) Mini Project Performing the CURD Operations

===

Employees Info

home page

eno

ename job

sal

operations

(edit)

edit_employee.jsp

Generate Report

101

raja clerk 9000

edit delete no=101 ?no=101

(select)

102 ramesh manager 8000 edit delete-

Pno=102 ?no=102

(delete)

register_employee.jsp

emp name:

emp job:

salary::

Tegister

000

clerk

90000

to delete the record?

Add Employee

(insert)

Double confirmation

are u sure that u want

rakesh

employee number :: employee name :: employee job :: employee salary ::

101 (not editable)

raja

Perform Edit Employee

editable

clerk

Operation in Db Table (use DS, controller, Service, Repository)

9000

Edit Employee

note:: After adding Lombok api starter, if we are getting exception then we need to comment <annotationPath> tag of pom.xml file (optional in latest eclipse IDEs)

is available

insert employee record to

db table by generated eno

dyamically

(use DS, controller, Service, Repository)

record deletion from DB

(use DS, controller, Service,Repository)

step1) make sure that "emp" db table in oracle Db s/w

step2) make sure that "emp_id_seq" sequence is created in oracle Db s/w

having ability to generate emp no dynamically (Optional to create becoz by configuring Sequence generator for the @Id property of Entity class, we can get this kind of sequence dynamically)

Schema: SYSTEM

Name: EMP_ID_SEQ

Properties DDL

Start With:

100

Increment:

1

Min Value:

100

Max Value: Cache:

10000

<Not Specified>

Cache Size:

Cycle:

<Not Specified>

Order:

<Not Specified>

note:: Using Entity class for gathering inputs and for displaying outputs is really bad practice becoz

it not only properties for col data .. it also contains properties for metadata properties ...So it is better to take another Java Bean nothing but VO (Value Object) class for the same

note:: The java bean using which we collect inputs from enduser and display outputs for enduser is called VO class

=> CREATE SEQUENCE "SYSTEM"."EMP_ID_SEQ" MINVALUE

note1: The Java Bean whose objs represent inputs/outputs is called VO class note2: The Java Bean whose objs represent Shippable /transferable data

100 MAXVALUE 10000 INCREMENT BY 1 START WITH 100 CACHE 20 NOORDER NOCYCLE;

with in the project or across the multiple projects is called DTO class note3: The Java Bean whose obj represent the record of DB table is called Entity class =>All these are basically helper class cum model class playing different roles

step3) create spring boot starter Project having the following dependencies

web, data jpa, lombok api, tomcat-embeeded-jasper, driver (collect seperately)

oracle

apache ,jstl

note:: collect tomcat-embedded-jasper and

File menu --->new ---> Project ---> spring startter

(collect seperetly)

apache JSTL from mvnrepository.com

Service URL

https://start.spring.io

Name

BootMVCProj07-MiniProject-CURDOperations

Use default location

G:\Worskpaces\Spring\NTSPBMS615-BOOT-Ext\BootMVCProj07- Browse

Location

Type:

Maven

Packaging:

War

Java Version:

11

✓ Language:

Java

Group

nit

Artifact

Version

Description

BootMVCProj07-MiniProject-CURDOperations

0.0.1-SNAPSHOT

Demo project for Spring Boot

Package Working sets

com.nt

Add project to working sets

Working sets:

New... Select...

-->next ---> select the following dependenices

web, lombok, spring data jpa, oracle driver

add extra dependencies in pom.xml by collecting them from mvnrepoistory.com

<!-- https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-jasper -->

<dependency>

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

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

<scope>provided</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.eclipse.jetty/apache-jstl -->

<dependency>

<groupId>org.eclipse.jetty</groupId>

<artifactId>apache-jstl</artifactId>

<version>10.0.20</version>

</dependency>

step4) create the following packages in the Project

#src/main/java

>com.nt

com.nt.controller com.nt.model

com.nt.repo com.nt.service

src/main/resources

static templates

application.properties.

packages

to be created

step5) add the following entries in application.properties

=>For data source cfg

=>For jpa-hibernate properties

=>For view resovlers

=>For Embedded Ports (server)

=>For Context path of web application

while runing in Embedded Tomcat server

application.properties

#View Resolver cfg

spring.mvc.view.prefix=/WEB-INF/pages/

spring.mvc.view.suffix=.jsp

#Embedded server port number

server.port=4041

#Context path

server.servlet.context-path=/Employee-CURDOperations

#DataSource cfg

spring.datasource.driver-class-name=oracle.jdbc.driver.Oracle Driver

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=system

spring.datasource.password=manager

#Hibernate -JPA properties

spring.jpa.hibernate.ddl-auto-update

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect. Oracle Dialect

step6) creating the following folders in src/main/webapp folder

webapp (standard folder)

✓ images (not s standard folder)

WEB-INF (standard folder)

pages (not a standard folder)

step7) Write the neccessary code to display the home page

i) place report icon as png or jpeg image in images folder

The non-standard folders in webapp ar

css -> to place all css files

js ----> to place java script files images ---> to place all image files videos ---> to place all video files audios ---> to place all audio files

ii) Develop controller class having handler method with request path "/"

```
@Controller
public class EmployeeOperationsController { (partial code)
@GetMapping("/")
public
String showHome() { return "home";
}
}
```

e

iii) place home.jsp in WEB-INF/pages folder

home.jsp (WEB-INF/pages)

```
<%@ page isELIgnored="false" %>
<h1 style="color:red;text-align:center"><a href="report">Get Employee Data</a></h1> <br><br>
<h1 style="color:red;text-align:center"><a href="report"><img src="images/report .png" alt="image not found"/></a></h1>
```

step8) Develop Model/Entity class

Employee.java

```
package com.nt.model;
import java.io.Serializable;
import javax.persistence.Column; import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType; import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
```

```
}
import lombok.Data;
@Table(name="emp")
@Entity
@Data
public class Employee implements Serializable {
@Id
@SequenceGenerator(name = "gen1",sequenceName = "emp_id_seq",initialValue = 1, allocationSize = 1)
@GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
private Integer empno;
@Column(length = 20)
private String ename;
@Column(length = 20)
private String job;
private Float sal;
private Integer deptno;
```

step9) Develop the Repository interface for Employee Model class in "com.nt.repository" pkg

```
//IEmployeeRepository.java
package com.nt.repository;
import org.springframework.data.repository.CrudRepository;
import com.nt.model.Employee;
public interface IEmployee Repository extends Crud Repository<Employee, Integer> {
}
```

step10) Develop the Service Interface, Service Impl class having logics related to fetching all the records to generate the report

```
//Service Interface
package com.nt.service;
import com.nt.model.Employee;
public interface IEmployee MgmtService {
public Iterable<Employee> getAllEmployees();
}
//Service Impl class
package com.nt.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.nt.model.Employee;
import com.nt.repository.IEmployeeRepository;
```

```
@Service("empService")
public class Employee MgmtServiceImpl implements IEmployeeMgmtService {
@Autowired
private IEmployee Repository empRepo;
@Override
public Iterable<Employee>getAllEmployees() {
return empRepo.findAll();
}
```

step11) add Handler method in controller class having request path "/emp_report" using the the above service method

```
@Controller
public class EmployeeOperationsController { @Autowired
private IEmployeeMgmtService empService;
@GetMapping("/") //To show the home page
public String showHome() {
//return LVN
return "home";
}
@GetMapping("/emp_report")
public String showEmployee Report (Map<String, Object> map) {
//use service
Iterable<Employee> itEmps=empService.getAllEmployees();
// put result in model attribute
map.put("empsList",itEmps);
//retunr LVN
return "show_employee_report";
}
}
```

step12) add show_employee_report.jsp in WEB-INF/pages folder having jstl tags based logic to generate the report content

show_employee_report.jsp

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@page isELIgnored="false" %>
<c:choose>
<c:when test="${!empty empsList}">
<h1 style="color:red;text-align:center"> Employees Report </h1>
<table border="1" align="center" bgcolor="cyan">
```

```
<tr style="color: red"><th>empno </th><th> emp name</th><th> Job </th> <th>salary </th><th> deptno</th>
</tr>
<c:forEach var="emp" items="${empsList}">
<tr style="color: blue">
<td>${emp.empno }</td>
<td>${emp.ename}</td>
<td>${emp.job }</td>
<td>${emp.sal}</td> <td>${emp.deptno}</td>
</tr>
</c:forEach>
</table>
</c:when>
<c:otherwise>
<h1 style="color:red;text-align:center"> Employees Not found </h1>
</c:otherwise>
</c:choose>
```

step13) add edit, delete and Add, home hyperlinks to the Report page

show_employee_report.jsp

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@page isELIgnored="false" %>
<c:choose>
<c:when test="${!empty empsList}">
<table border="1" align="center" bgcolor="cyan">
<h1 style="color:red;text-align:center"> Employees Report </h1>
<tr style="color: red"><th>empno </th><th> emp name</th><th> Job </th> <th>salary </th><th> deptno</th>
<th> operations</th> </tr>
<c:forEach var="emp" items="${empsList}">
<tr style="color: blue">
<td>${emp.empno }</td>
<td>${emp.ename}</td>
<td>${emp.job}</td>
<td>${emp.sal}</td>
<td>${emp.deptno}</td>
<td><a href="emp_edit?no=${emp.empno}"><img src="images/edit.png" width="30px" height="30px"></a>
<a href="emp_delete?no=${emp.empno}"><img src="images/delete.png" width="30px" height="30px"></a>
</td>
</tr>
</c:forEach>
```

```
</table>
</c:when>
<c:otherwise>
<h1 style="color:red;text-align:center"> Employees Not found </h1>
</c:otherwise>
</c:choose>
<center>
<a href="emp_add"><img src="images/add.png" width="40px" height="50px"> Add Employee</a>
   
<a href="./"><img src="images/home.png" width="40px" height="50px">Home</a>
</center>
```

step14) Perform the following opeations to complete add employee activity

(a) make sure that add employee hyperlink is placed in the report page

In show_employee_report.jsp

```
<a href="emp_add"><img src="images/add.png" width="40px" height="50px"> Add Employee</a>
```

(b) Add the following code in Service Interface and In service Impl class to insert the record

In service Interface

```
public String registerEmployee(Employee emp);
```

In service Impl class

```
@Override
public String registerEmployee(Employee emp) {
}
return "employee is saved with id value:"+empRepo.save(emp).getEmpno();
```

(c) Add handler method in controller class to show the form page of Add Employee operation

```
@GetMapping("/emp_add") // for form launching
public String showFormForsaveEmployee(@ModelAttribute("emp") Employee emp) {
//return LVN
return "register_employee";
}
```

(d) develop register_employee.jsp page in WEB-INF/pages folder

register_employee.jsp

```
<%@ page isELIgnored="false" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="frm" %>
<h1 style="color:red;text-align:center">Register Employee</h1>
<frm:form modelAttribute="emp">
<table align="center" bgcolor="cyan">
<tr>
```

```
<td> Employee Name::</td>
<td><frm:input path="ename"/> </td>
</tr>
<tr>
<td> Employee Desg::</td>
<td><frm:input path="job"/> </td>
</tr>
<tr>
<td> Employee salary::</td>
<td><frm:input path="sal"/> </td>
</tr>
<tr>
<td> dept no </td>
<td><frm:input path="deptno"/> </td>
</tr>
<tr>
<td> <input type="submit" value="submit"></td>
<td> <input type="reset" value="cancel"> </td>
</tr>
</table>
</frm:form>
```

note:: if the <form> is taken with out action attribute.. then the form will be submitted to a url

using which the form page launching is done

note:: if the <a> is taken with out href attribute.. then the hyperlink generated request

will be sent to a url

using which the hyperlink page launching is done

step15) Perform the add employee taks related form submission operation

i) place @PostMapping("/emp_add") handler method in controller class calling service class method that inserts the record in db table

In controller class

```
@PostMapping("/emp_add")
public String saveEmployee(@ModelAttribute("emp") Employee emp,
//use Service
Map<String,Object> map) {
String msg=empService.registerEmployee(emp);
Iterable<Employee> itEmps=empService.getAllEmployees();
//keep the result in ModelAttribute
```

map.put("resultMsg", msg);

Limitations of this code

map.put("empsList", itEmps);

// return LVN

}

return "show_employee_report";

ii) add statement in show_employee_report.jsp page to read and display

"resultMsg" model attribute value

<h2 style="color:green;text-align:center">${resultMsg}</h2>

a) Missing code modularity/reusability i.e we are placing the logics of showReport() method also in saveEmployee(-) handler method with out going for reusability

b) Raises Double posting/ Duplicate form submission problem as discussed below

What is "Double Posting" (or) "Form Duplication problem" and how can we solve it?

Ans) if u press the "refresh" button on the result page of form submission the form

activity will be repeated unnecessary and leads problems like duplicate record insertion with new id, duplicate money deduction from Credit Card/Debit card and etc..

To solve this problem .. take support of PRG (Post Redirect Get) Pattern

PRG Pattern Implementation

==========

Ans) Redirect the request to GET Mode Handler method from POST mode handler method using handler method chainig concept.. So when we press refresh button the GET Mode request handler method will be repeated ..which will not give any side effects becoz the Get Mode reuquest handler method performs generally select the data operation which is not going to be a problem.

Example1 Code for PRG Pattern

==========

In Controller class

P-POST

@PostMapping("/emp_add") //form submission related to add employee operation

public String saveEmployee(@ModelAttribute("emp") Employee emp,

Map<String, Object> map) {

System.out.println("EmployeeOperationsController.saveEmployee()");

//use Service

String msg=empService.registerEmployee(emp);

//keep the result in ModelAttribute

map.put("resultMsg", msg);

// return LVN

}

return "redirect:emp_report"; (R--Redirect)

```
@GetMapping("/emp_report")
public String showEmployeeReport(Map<String, Object> map) {
System.out.println("EmployeeOperationsController.showEmployee Report()");
Iterable<Employee> itEmps=empService.getAllEmployees();
//use service
G-GET
// put result in model attribute
map.put("empsList",itEmps);
//retunr LVN
return "show_employee_report";
}
```

Limitation of this code

(Best)

=>Since the the source handle mostMapping method) and destination

handler method (@GetMapping method) are not using same req,res objs beco2'send redirection, So the model attributes (request scope) of Source Handler method can not be used/accessed in the Dest handler method and its LVN based jsp page.

=>To solve this problem take the suppport of "Flash Attributes" by placing

in "RedirectAttributes" object.

(another shared with Redirection scope)

Memory

Example 2 Code for PRG Pattern (Using flash attributes of RedirectAttribute obj)

=========

========

(shared memory).

=> The flashAttributes kept in RedirectAttributes object are visible and accessible

in

the dest handler method and its LVN related to view comp (jsp page) only

till the end of Redirction activity.. i.e next request given after redirection activity

to dest handler method the flashAttributes are not visible.

In controller class

```
@PostMapping("/emp_add") //form submission related to add employee operation
public String saveEmployee(@ModelAttribute("emp") Employee emp,
RedirectAttributes attrs) {
System.out.println("EmployeeOperationsController.saveEmployee()");
//use Service
String msg=empService.registerEmployee(emp);
//keep the result as flashAttribute
```

attrs.addFlashAttribute("resultMsg",msg);

=>SharedMemory scope (BindingAwareModelMap) is request scope

=> RedirectAttributes scope is rediection scope i.e from source request to

destination request of redirection

=> Session scope is client scope (browser scope)

request mode

=> if ur using "forward:<path>" for forwarding request mode based method chaining then source handler method must match with dest handler method request mode

=> if ur using "redirect:<path>" for redirection mode based method chaining then the source handler method request mode need not to match with dest handler method request mode

Special shared Memory which holds the attributes

```
// return LVN
return "redirect:emp_report";
}
```

only during coarse of

Redirection.. i.e once the redirection is over.. attributes

in this special shared memory

will be vanised.. So these

atributes are called FlashAttributes

```
@GetMapping("/emp_report")
public String showEmployeeReport(Map<String, Object> map) {
System.out.println("EmployeeOperationsController.showEmployeeReport()");
//use service
Iterable<Employee> itEmps=empService.getAllEmployees();
// put result in model attribute
map.put("empsList",itEmps);
//retunr LVN
return "show_employee_report";
}
```

Limitation

(Ignorable)

=> if we give next request to Dest method of redirection by using refresh button ..

the earlier displayed "flash attributes" related message will not come...

(if u want to continue that message take the support of session attributes)

Example 3 Code for PRG Pattern

========

(Using Session obj and Session attributes)

are

to

=>HttpSession object and its session attributes specific each browser s/w of Client machine...

So session attributes are visible in all the handler methods as along as request is coming from that browsers/w.

Controller class

```
@PostMapping("/emp_add") //form submission related to add employee operation
public String saveEmployee(@ModelAttribute("emp") Employee emp,
HttpSession ses) {
System.out.println("EmployeeOperationsController.saveEmployee()");
//use Service
String msg=empService.registerEmployee(emp);
//keep the result as flashAttribute
ses.setAttribute("resultMsg",msg);
// return LVN
return "redirect:emp_report";
}
@GetMapping("/emp_report")
public String showEmployeeReport(Map<String, Object> map) {
System.out.println("EmployeeOperationsController.showEmployeeReport()");
//use service
Iterable<Employee> itEmps=empService.getAllEmployees();
//put result in model attribute
map.put("empsList",itEmps);
//retunr LVN
return "show_employee_report";
}
```