Data Rendering

=>The process of passisng data from controller class to view comp by keeping in one or another scope is called Data Rendering.

=>For Data Rendering, we need to keep the Data in Model Attributes of controller class..

and we need to read these model attributes in jsp view comp with the support of EL (Expression Language)

to

=>For Every_request given Spring MVC App the the DispatcherServlet creates one special object called (shared memory) BindingAwareModelMap object having request scope.. The handler methods of Controller class can accesss that object as param of type Map<String, Object> collection as shown below

request (a1)

(a2) (a5) DispatcherServlet

(83) HandlerMapping comp

note:: if we design handler methods of controller/handler class with possible set of params (20+) then creating respective objects based on the type of the params and calling the handler methods having respective objs as the arguments will be taken care by DipsatcherServlet

Using EL of jsp page, we can perform the arithemetic

and logical operations in the jsp page with out using java code

(DispatcherServlet actually creates

this shared Memory (BindingAware ModelMap obj) only when it gets BindingAwareModelMap class hierarchy class,interface as the parameter type in the controller class handler method)

+5

✓ Object

✓

AbstractMap<K, V>

✓

HashMap<K, V>

✓

ModelMap

java.util.Map(1)

implements

Model (1)

a) (Shared Memory)

(.../process)

response

dynamic webpage (a16)

(a17)

attr1-val1 (#a8)

attr2-val2

@Controller

public class MyController{

(a4?)

BindingAwareModei Map object

(request scope) (shared memory)

(a7)

val1

val2

model

(a10)

`(a13) (a15)`

attribute values

@RequestMapping("/process")

public String process(Map<String,Object> map){

(a8)map.put("attr1","val1");

map.put("attr2","val2");

(a11)

InternalResourceViewResolver

|-->prefix: /WEB-INF/pages/

model

attr name

attr value

return "show_data"; (a9)

attributes are added to shared memory

|-->suffix: .jsp

}

LVN

View object (a12)

}

/WEB-INF/pages/show_data.jsp

LinkedHashMap<K, V>

implements

✓ Extended ModelMap BindingAwareModelMap

creates

note:: if DispatcherServlet obj observers the like above hierarchy upto Map(1) classes or interfaces as the parameter types them the DispatcherServlet BindingAware ModelMap class obj as the special obj(sharable obj) having request scope.. In this object the handler method keeps the results and jsp view comps reads the results using EL code

WEB-INF/pages/show_data.jsp (a14)

${attr1} (a14) ${attr2}

(reading Model attributes using EL)

java.util.Map<K,V> (1)

implements

with respect diagram

(a5)(a6)&& (a7) :: DS gets handler method signature whose request path

gets

is "/process" and also controller class bean id DS creates Shared Memory (BindingAware ModelMap class obj) by seeing Map<String,Object> type param in the handler method signature and DS also calls

handler method having BindingAwareModelMap obj as the argument

value.. which refered by Map<String, Object> type param in the handler method

(a8) the data kept using map collection goes to SharedMemory

as Model attributes (indirectly request scope attributes)

(a14) The view comp show_data.jsp execute and reads the Model attributes from request scope using EL Support ${attr1}, ${attr2}

✓ Object

A AbstractMap<K, V>

HashMap<K, V>

org.sf.ui.Model(1)

LinkedHashMap<K, V>

C

ModelMap

ExtendedModelMap

BindingAwareModelMap

=>The BindingAwareModelMap object(Shared memory having request scope) can be referred by either one its hierarchy class or interface reference variable becoz super class ref variable can refer one of its sub class obj or interface ref variable can refer one its impl class object

(shared memory)

The handler method of controller class can refer BindingAwareModelMap object by taking the parameter of the following type

(loosely coupled programming)

a) Map<k,v> (1)Best) becoz it supports more of non-invasive programming

b) HashMap (c)

c) LinkedHashMap (c)

d) ModelMap (c)

e) Model(1)

f) ExtendedModelMap (c)

g) BindingAwareModelMap (c)

=>

super class reference variable: can refer its sub class obj

=> interface reference variable can refer its impl class obj

=>if the handler method is designed having one of the listed parameter type then the DispatcherServlet creates BindingAwareModelMap class obj as the SharedMemory one 1 per request given to handler method basis

Example code (In Controller calss)

```
@RequestMapping("/process")
Best of way representing SharedMemory becoz
Good Pratice
it makes the code non-invasive
public String process (Map<String,Object> map) { System.out.println("ShowHomeController.process():: ShareMemory Object class name::"+map.getClass());
//add model attributes to shared Memory
map.put("attr1","val1");
map.put("sysDt", LocalDateTime.now());
//return LVN
return "show_data";
}
```

In show_data.jsp(WEB-INT/pages folder)

```
<%@ page isELIgnored="false" %>
<h1 style="color:red;text-align:center">Show_data.jsp</h1>
Model attributes are :: ${attr1}, ${sysDt}
request url ::
localhost:2525/MVCBootProj1-SX +
←
localhost:2525/MVCBootProj1-Showing HomePage/process
```

Show_data.jsp

rst App

Model attributes are :: val1, 2022-09-25T11:36:38.056241900

#2

Example code

In controller class

```
@RequestMapping("/process")
This is bad pratice becoz Model(I) is spring api specific interface
public String process(Model model) {
i.e makes the code as invasive code
System.out.println("ShowHomeController.process():: ShareMemory Object class name::"+model.getClass());
```

```
//add model attributes to shared Memory
model.addAttribute("attr1","val1");
model.addAttribute("sysDt", LocalDateTime.now());
}
//return LVN
return "show_data";
```

In WEB-INF/pages/show_data.jsp

```
<%@ page isELIgnored="false" %>
<h1 style="color:red;text-align:center">Show_data.jsp</h1>
Model attributes are :: ${attr1}, ${sysDt}
```

#3

Example code

@RequestMapping("/process") Not good pratice becoz ModelMap is spring api class i.e copde becomes invasive

```
public String process(ModelMap map) {
System.out.println("ShowHomeController.process():: ShareMemory Object class name::"+map.getClass());
//add model attributes to shared Memory
map.addAttribute("attr1","val1");
map.addAttribute("sysDt", LocalDateTime.now());
}
//return LVN
return "show_data";
```

WEB-INF/pages/show_data.jsp

```
<%@ page isELIgnored="false" %>
<h1 style="color:red;text-align:center">Show_data.jsp</h1>
Model attributes are::${attr1}, ${sysDt}
```

localhost:4041/MVCFirstApp1/pm X +

localhost:4041/MVCFirstApp1/process

Model attributes are :: val1, 2022-09-25T11:51:32.982232600

Show_data.jsp

Taking handler method return type as Model(1)

-> Here we need to create SharedMemory to place the model attributes manually, more over

the LVN will be request path itself.

note:: The handler method of controller class generated results and gathered results from service, DAO classes will be passed to View comps in the form of model attributes Using Data Rendering Cocepts

In controller class

```
@RequestMapping("/process")
public Model process() {
```

}

Here the LVN is request path (process)

Model model=new BindingAwareModelMap(); // manually creted shared Memory //add model attributes to shared Memory

model.addAttribute("attr1","val1");

model.addAttribute("sysDt", LocalDateTime.now()); return model;

=>if the controller class handler method is not returning the LVN then the handler method request path it self becomes the default LVN..

For example, if the request path is "/home" the default LVN is "home".

In view comp (jsp comp) (process.jsp)

<%@ page isELIgnored="false" %>

<h1 style="color:red;text-align:center">process.jsp</h1>

Model attributes are :: ${attr1}, ${sysDt}

Taking Map<String, Object> as the return type of handler method

code in Controller class

@RequestMapping("/process")

public Map<String, Object> process() {

//create SharedMemory

|

Here the LVN is request path (process)

Map<String, Object> map=new HashMap();

//add model attributes to shared Memory map.put("attr1","val1");

map.put("sysDt", LocalDateTime.now()); return map;

Here the HashMap object acts

as the shared Memory

}

In process.jsp(WEB-INF/pages folder)

<%@ page isELIgnored="false" %>

<h1 style="color:red;text-align:center">process.jsp</h1>

Model attributes are: ${attr1}, ${sysDt}

if handler method is not having "String" return type and finds no

facility

for returning LVN then it takes

the request path itself as the LVN.

For example.. if the return of handler method is Map<k,v> or Model or HashMap or ... (other than string)

then the request path given in the @RequestMapping(-) automatically becomes LVN

Taking ModelAndView (legacy style) as the return type of Handler method (old style)

In Controller class

@RequestMapping("/process")

public ModelAndView process() { //create SharedMemory

bad approach

be

ModelAndView used to the fixed return type for handler methods

ModelAndView mav-new ModelAndView(); //add model attributes to shared Memory mav.addObject("attr1", "val1");

mav.addObject("sysDt", LocalDateTime.now());

// place LVN to MAV object

mav.setViewName("show_data");

return mav;

show_data.jsp(WEB-INF/pages folder)

<%@ page isELIgnored="false" %>

<h1 style="color:red;text-align:center">Show_data.jsp</h1>

Model attributes are::${attr1}, ${sysDt}

in spring mvc applications of

xml driven cfgs.. So we can say

it is legacy style of taking return type.

Here the ModelAndView class obj

acts the Shared Memory

Limitations of taking Model,Map<String, Object>, HashMap and etc.. as the return type of

Handler method

==========

(other than String)

==========

=========

a) We need to create and return SharedMemory having model attributes manually in the handler method

b) No control on the LVN i.e it takes request path of handler method itsents the logical view name

c) The DS created default SharedMemory (BindingAwareModelMap obj) will be wasted completely (if created)

take

d) we can not request path of the handler method as the "/"

type

note: if the return of handler method is ModelAndView .. all the above problems are there except (b) problem. (ModelAndView is legacy class

Advantages of taking Map<String,Object>,Model and etc.. as the parameter types of handler method

==========

==========

==========

a) we need not to create the sharedMemory manually to place the model attributes (DS will create it)

b) we can use the DS created Shared Memory to place the Model attributes i.e the DS created SharedMemory(BindingAware ModelMap obj) will not be wasted

c) By taking the returnt type as java.lang.String, we can get control on the LVN.

and not recomanded to use)

d) we can take the request path as the "/"

is

What happens if the return type of Handler method void?

Ans) the request path of handler method will be taken as the LVN

```
@RequestMapping("/process")
public void process(Map<String, Object> map) {
//add model attributes to shared Memory
map.put("attr1","val1");
map.put("sysDt",new Date());
}
```

The best signature of the handler method in most of the situations is

public String <method name>(Map<String,Object> map, ...,....,...){

It takes "process" (request path) as the Logical vie name

}

What happens if the handler method returns null?

Ans) the request path of handler method will be taken as the LVN

```
@RequestMapping("/process")
public String process(Map<String, Object> map) {
map.put("attr1","val1");
//add model attributes to shared Memory
map.put("sysDt",new Date());
```

It takes "process" (request path) as the Logical vie name

```
return null;
 }
```

...

return type

any name

represents shared memory

...

we can give

other params

How can we forward the request from one handler method to another handler method?

is

a) This called handler method chaining i.e the request given to one handler method will communicate with another handler method of same or different controller class

What is handler method chaining? In how many ways we can do it?

Ans) The process of making the request received by one handler method of one controller passing its request to another handler method of same or different controller class is called handler method chaining

In Controller class

```
@RequestMapping("/process")
public String process() {
}
System.out.println("ShowHomeController.process()");
return "forward:report";
```

fixed keyword request path of dest handler method

```
@RequestMapping("/report")
public String showReport() {
Source handler method
System.out.println("Show HomeController.show Report()"); Dest handler method
return "show_data";
}
```

=>Here the source handler method directly communicates with dest handler method .. It internally uses rd.forward(-,-) where the soruce handler method and dest handler method will use same req,res objs.

so the source handler method model attributes can be used in dest handler method and its view comp

How can we redirect one handler method reqeust to anonther handler method?

ans) This is also called HandlerMethod chaining.. but the source Handler method redirect the request to dest handler method after having network round trip with browser. So the source handler method and the dest handler method will not use same req,res objs. Due to this the model attributes of source handler method can not read and used in dest handler method and its view comp, But to pass additional data Westend query string to the "redirect:<path>" as shown here (redirect:report?p1=val1&p2=p2=val2)

class

```
@RequestMapping("/process")
public String process(HttpServletRequest req) {
System.out.println("ShowHomeController.process()::"+req.hashCode());
req.setAttribute("attr1", "val1");
return "redirect:report";
}
```

It can be done in two ways

a) Forwarding mode of chaining (forward:<path>)

b) Redirecting mode of chaining (redirect: path)

```
public String showReport(HttpServletRequest req) {
```

@RequestMapping("/report")

System.out.println("req attribute ::"+req.getAttribute("attr1")); gives null

return "show_data";

}

System.out.println("ShowHomeController.showReport()");

note1:: use forward request mode handler method chaining if source handler method and dest handler method are there

same

in the web application and to share some data

note2:: use redirection mode handler method chaining if source handler method and dest handler method are there

in two different web application and do not want share data..

of same or different servers/machines

Can we place return statement in the method whose return type is void?

Ans) Yes.. we place return statement with out value.

for LVN

return; i.e return <with out value>

What is the best signature for handler method?

Ans) public String <handler method>(Map<String,Object> map){

b/w

What is difference forward request mode of handler method chaining and redirection mode handler method handler method chaining?

Ans) Calling one handler method from another handler method is called handler method chaining

In "forward:<path>" (forwarding request mode method chaining)

a) The source handler method and dest handler method use same req, res objs b) The source handler method directly communicates with dest handler method

c) The source handler method and dest handler method must be

d) interally uses rd.forward(-,-)

there in the same web application

(e) The model attributes/request attributes of source handler method can be accessed

and used in the dest handler method and its view comp

In "redirect:<path>" (redirecting request mode of and chaining)

a) The source handler method and dest handler method do not use same req, res objs

b) The source handler method communicates with dest handler method through browser

c) The source handler method and dest handler method can be

there in the same web application or in

two different web applciations of same server

or different servers belong to same machine or different machines

....

...

//return LVN

}

represents the SharedMemory

How to get request, response objects in the handler methods?

Ans) we can take them as the parameters of the handler methods.. So the DispatcherServlet

can pass them as arguments while calling the method

```
@RequestMapping("/process")
public String process(HttpServletRequest req,HttpServletResponse res) {
System.out.println("ShowHomeController.process()::"+req.hashCode());
req.setAttribute("attr1", "val1");
return "show_data";
}
```

How to pass HttpSession object to handler method of controller class?

Ans) we can take HttpSession as the parameter value of the handler method

```
@RequestMapping("/process")
public String process(HttpSession ses) {
ses.setAttribute("attr1", "val1");
return "show_data";
}
```

=>The EL Code kept in jsp page can read and display given

attribute values by searching in multiple scopes in the following

order

a) page scope b) request scope c) session scope

d) application (note:: model attributes scope is

request scope)

How to pass ServletConfig object and ServletContext objs to controller class handler methods?

Ans) Inject them to @Controller class using @Autowired annotation

```
@Autowired
private ServletContext sc;
```

we are Injecting them to

```
@Autowired
```

Controller class by collecting them from

```
private ServletConfig cg;
@RequestMapping("/process")
```

DispatcherServlet as spring beans

given by AutoConfiguration

```
public String process(Map<String, Object> map) {
System.out.println("web application context path::"+sc.getContextPath());
System.out.println("DS Logical name::"+cg.getServletName());
map.put("attr1", "val1"); //Model attribute
return "show_data";
}
```

d) internally uses res.sen Redirect(-) method

e) if source handler methods wants pass some data to dest handler method then we need to append query String to the

url placed for redirection (redirect:<path>?param1=val1&param2=val2)

Use case s handler method chaining

=========Q£==========

ing

In any Project, the handler method perform insert,update and delete operations on DB table records will redirect/forward the request to another handler method that deals with show Report operation (select operation)

note: if any parameter type is not there in the list of allowed parameters of

handler method and that object available through DispatcherServlet then

go for @Autowired based Injections to controller classes.. otherwise

take them as the param types of the handler methods.

eg: request,response, HttpSession, Shared Memory (BindingAwareModelMap) and etc.. (as the param types)

note:: Generally the objects that are specific each reuquest directly or indirectly take them

as handler method args..similarly the objs that are visible across the multiple requests

take them as @Autowired based Injections.

eg: SErvletConfig, ServletContext objs

How to send output from Handler method directly to browser with out involving ViewResolver and view comps?

Ans) This concept is required in two situations

a) While performing Filedownloading activity where the selected file content goes

to browser as response having downloading ability

b) while developing the controller class as the RestController (Restful web applications) (future discussions)

For this we need to take HttpServletResponse as the param of handler method and we need to use

PrintWriter stream obj pointing to response object to write the messsages to browser directly

```
@RequestMapping("/process")
public void process(HttpServletResponse res) throws Exception{
PrintWriter pw=res.getWriter();
// Since the PrintWriter is used
```

to write the message to browser

//get PrintWriter

//set response content type

res.setContentType("text/html");

//write data to browser s/w

pw.println(" directly from handler method ");

}

directly.. So the option of

taking request path name as LVN

is gone.

note:: content-disposition is the special response header that gives instructions to browser towards displaying the recived response on to browser.. "inline" is the default value to display response directly on to the browser, "attachment" value makes the recived response content as the downloadable file content
res.setHeader("Content-Disposition","attachment; fileName=abc.html");

By adding this line in the handler method that is having HttpServletResponse obj as the param, we can send handler method content as the downloadable file to the browser

home page

find season

Developing spring boot MVC app having Service class with B.method

Story board

==========

(a)

season page

Welcome to Winter Season

home

http://localhost:2525/MVCProj2-DataRendering

(i) (z)

InternalResourceViewResolver

|--->prefix: /WEB-INF/pages/ |--->suffix: .jsp

(FrontController)

(c) (q)

RequestMappingHandlerMapping

//serivce interface

(b) (e) (h)

public interface ISeasonFinderService{

public String findSeason();

DispatcherServlet

[/]

```
@Autowired
(k) (m) (p) (s)
@Controller
public class Season FinderOperationsController{
private ISeasonFinderService seasonService;
(y) (b1) (d1)
(d?)
}
//service Impl class
@Service("seasonService")
@RequestMapping("/") (f)
public class Season FinderServiceImpl
public String showHome(){
implements ISeasonFinderService{
return "welcome"; (g)
}
(v)
public String findSeason(){
(r?)
@RequestMapping("/season")
...
public String showSeason (Map<String, Object> map){
(t)
//use service
(n)
View object (j) /WEB-INF/pages/welcome.jsp
View obj
(a1)
/WEB-INF/pages/display.jsp
webpage (n)
(1)
welcome.jsp(WEB-INF/pages folder)
<a href="season"> get sesson </a>
get season
(0)
display.jsp(WEB-INF/pages folder) (ċ 1)
webpage (e1)
```

season name: rainy season

(w) String msg=seasonService.findSeason();

//kee the result in model attribute

map.put("resultMsg", msg);

//return LVN

}

return "display"; (x)

Season name :: ${resultMsg}

**
**

**home **

home