WAY2LEARN

LAB- Spring Mvc -Form Handling

In this lab you will be understanding how to render forms and handle form submits

STEP -1

In this step, you will be working on **01mvc-forms-start** project

1) Open requestinfo.jsp in /WEB-INF/jsp/application and observe the code written for a form.

Open RequestInfoController and observe that when GET request is given for /requestInfo.htm, it is returning the view name as "requestInfo".

Open tiles.xml and observe the tile definition for "requestInfo" . The body of this tile definition displays requestInfo.jsp

Deploy the application . On home page, click on RequestInfo link in the topmenu. You will be show with a form. Try to submit it by filling information.

You will observe that **requestInfoSuccess.jsp** is shown. Can you tell why only Name, email and ages are displayed?

2)when the form is submitted, the request goes to requestInfoPost() method which is just returning the view name.

Change the method such that the request parameters name, email,age,city,dob,mobile,query are injected into method arguments .(Hint : Use @RequestParam)

In the method, create an Object of RequestInfoData using all these request parameters and add it to model with name "**requestInfoData**".

The method should look like below :(Dont copy paste. Analyze the code and type it)

```
@RequestMapping(value={"/requestInfo.htm","/s/requestInfo.htm"},method=RequestMethod.PO
ST)
	public String requestInfoPost(String name,String email,int age,String city,
			Date dob, long mobile, String query,Map<String,Object> model){

		RequestInfoData requestInfoData= new RequestInfoData();
		requestInfoData.setName(name);
		requestInfoData.setEmail(email);
		requestInfoData.setAge(age);
		requestInfoData.setCity(city);
		requestInfoData.setDob(dob);
		requestInfoData.setMobile(mobile);
```

```
                requestInfoData.setQuery(query);
                model.put("requestInfoData", requestInfoData);

                return "requestInfoSuccess";
        }
```

Now Open requestInfoSuccess.jsp and observe that the code is already written for you to display data from **requestInfoData** for most of the fields. Change values for Name, email, Age in jsp such that they will be displayed from **requestInfoData** instead of params.

Deploy the application . On home page, click on RequestInfo link in the topmenu. You will be shown with a form. Try to submit it by filling information.

You should observe Http Status 400. If you observe the console of server, you can see that conversion of string to Date failed.

If you want conversion to happen, annotate Date argument with
`@DateTimeFormat(pattern="yyyy-MM-dd")`

If you want the conversion to happen, the corresponding formatter has to be registered.

As we have configured <mvc:annotation-driven /> in mvc-config.xml , the formatter are already registered.

Now redeploy and submit the form with date in the above format . Data Should be submitted properly and you should see the success page with submitted information.

3) We are not happy with the method signature as the method is taking lot of arguments. What if there are 50 form fields which are submitted?

You can make the method to take RequestInfoData as argument and annotate it with @ModelAttribute("requestInfoData")

Automatically, all the request parameters will be mapped to RequestInfoData Object and it is injected into method.

Remove all method arguments and make the method look as shown below :

```
@RequestMapping(value={"/requestInfo.htm","/s/requestInfo.htm"},method=RequestMethod.PO
ST)
public String requestInfoPost(@ModelAttribute("requestInfoData") RequestInfoData
requestInfoData ){

                return "requestInfoSuccess";
        }
```

The process of binding the request data into model attribute is **DataBinding**

Can you tell who is responsible for this data binding?

Make the above change , deploy and test by giving dob in expected format.   Result should be same as earlier.

4)   What if date is given in wrong format ?
Databinding will fail and you will GET status 400.

How to handle databinding errors?
 Make you method to take Binding result as argument as shown below :

```
public String requestInfoPost(@ModelAttribute("requestInfoData") RequestInfoData
requestInfoData,BindingResult result ){

             return "requestInfoSuccess";
      }
```

If there are errors in databinding, Binding result will be populated and will be passed as argument.

Now, if there are binding errors, you can render the previous view. write your method as shown below :

```
public String requestInfoPost(@ModelAttribute("requestInfoData") RequestInfoData
requestInfoData,BindingResult result ){
             if(result.hasErrors()){
                    return "requestInfo";
             }

             return "requestInfoSuccess";
            }
```

Deploy the application and try to submit the form with wrong date format. You should observe that same form is displayed.

But the whole form is now empty . User has to fill the whole form again just because of one wrong entry. This is bad. We want to show the same page with earlier data and also error messages.

If you want to retain the previous data and show error messages, you have to use spring's form tag library.

Now you have to replace all html form elements with spring's form tags.

Rename requestInfo.jsp to requestInfoWithoutSpringTags.jsp
Rename requestInfowithspringtags.jsp to requestInfo.jsp

Observe how we are displaying errors using <form:errors /> tag.
Also observe that we gave modelAttribute="requestInfoData" in <form:form > tag.

Can you tell why it is required ? (See the video once again if you can't remember)

Now deploy the application click on RequestInfo link in topmenu of home page.
You will see exception stack trace. Can you find out the reason ? (See the video if you can't)

Modify the requestInfo() as below :

```
@RequestMapping(value={"/requestInfo.htm","/s/requestInfo.htm"},method=RequestMethod.GE
T)
	public String requestInfo(Map<String,Object> map){
		map.put("requestInfoData", new RequestInfoData());
		return "requestInfo";
	}
```

Now deploy the application click on RequestInfo link in topmenu of home page.You should see the page. Submit the page with wrong date format. You should see error messages.

But the error messages are not friendly.

If you need to display friendly error messages, you need to write errormessages in a .properties files.

You have been provided with errormessages.properties in src/main/resources.
Observe how we are giving keys .

What are the various possible keys ? (See the video if you dont remeber)

You have to tell about this properties file by configuring a ResourceBundleMessageSource as shown below   in mvc-config.xml

```
	<bean id="messageSource"
class="org.springframework.context.support.ResourceBundleMessageSource">
		<property name="basename" value="errormessages" />
	</bean>
```

Now deploy the application and submit the page with wrong data for age and dob. You should see friendly error messages.

5) Now for city, field we want to display a dropdown with prepopulated cities.

So, we can use form:select tag as shown below :

```
<form:select path="city">
        <form:option value="blr" label="BANGALORE"/>
        <form:option value="del" label="DELHI"/>
        <form:option value="mum" label="MUMBAI"/>
</form:select>
```

But we are hard coding cities here.

We want the cities to come dynamically.

So, Modify the requestInfo() Method as shown below :

```
        public String requestInfo(Map<String,Object> map){
                map.put("requestInfoData", new RequestInfoData());

                List<String> cities= Arrays.asList("Bangalore","Hyderabad","Delhi");
                map.put("cities", cities);

                return "requestInfo";
        }
```

Now modify the form tag for city as shown below

```
        <form:select path="city" items="${cities}" cssClass="form-control" />
```

Deploy the application and Click RequestInfo link on home page. You should observe that the cities are populated.

But now the label and value of select options are same. What if you want Label and value to have different value?

Modify the requestInfo() Method as shown below :

```
@RequestMapping(value={"/requestInfo.htm","/s/requestInfo.htm"},method=RequestMethod.GE
T)
        public String requestInfo(Map<String,Object> map){
                map.put("requestInfoData", new RequestInfoData());

                List<City> cities=Arrays.asList(new City("Hyderabad", "hyd"),
                                new City("Bangalore", "blr"),
                                new City("Delhi", "del"));
                map.put("cities", cities);

                return "requestInfo";
        }
```

Observe the City.java Once.

Now change the <form:select> tag for city also as shown below :

```
<form:select path="city" items="${cities}"
                                  itemLabel="cityName" itemValue="cityCode"
cssClass="form-control" />
```

Now deploy the application and observe that city dropdown is populated with cities with different label and value. View the source code of displayed html to confirm this

But submit the form with wrong date or give age value as string. You will be shown the same page with error message. Observe the dropdownbox for city. It is not populated because there is no model attribute with name cities when POST request is made. In the method that is executed for GET request, we have populated model attribute "cities". But not in a method for POST.

If we want some model data to be populated for any request coming to the controller, we can write a separate method in controller class annotated with @ModelAttribute as shown below :

```
@ModelAttribute("cities")
	public List<City> getCities(){
		System.err.println("RequestInfoController.getCities()");
		List<City> cities=Arrays.asList(new City("Hyderabad", "hyd"),
				new City("Bangalore", "blr"),
				new City("Delhi", "del"));


		return cities;
	}
```

Write the above method in the RequestInfoController. Remove the code which is adding "cities" as model attribute in requestInfo() method.

The method annotated with @ModelAttribute will be executed for any request coming to the controller.

Deploy the application and test it by submitting wrong data. When error page is displayed, you should observe that cities are populated.

But now this getCities() method is executed for every request coming to this controller. I want this method to be executed once per session. I want the model attributes to be exported to session on the first time.

So, Annotate the Controller class as shown below :

```
@Controller
@SessionAttributes({"cities"})
public class RequestInfoController {
```

Deploy the application and make sure that getCities() is called only once per session.
How do you test it? keep some sysout in the method and make sure that it is printed only once.

POST-REDIRECT-GET

1) Deploy the application , click the Request Info Link on home page and submit the form with correct data. It should display the requestInfoSuccess page.

What if you refresh the page ? Again POST request goes to same URL which may cause in duplicate data being submitted.

So, it is a good practise to redirect to a different page after POST request is succesful.

So, modify the return value of requestInfoPost() method as

```
return "redirect:requestInfoSuccessRedirect";
```

Now write the below method in the controller to handle GET requests fot **requestInfoSuccessRedirect**

```
	@RequestMapping("requestInfoSuccessRedirect")
	public String requestInfoSuccessRedirect(){
		
		return  "requestInfoSuccess";
	}
```

Deploy the application , click the Request Info Link on home page and submit the form with correct data. you should be redirected to URL ending with **requestInfoSuccessRedirect**

But is the data displayed in success page? No

If you want some data to be added as query parameters in the redirect URL, you can change the method as shown below :

```
	@RequestMapping(value={"/requestInfo.htm","/s/requestInfo.htm"},method=RequestMe
thod.POST)
	public String requestInfoPost(@ModelAttribute("requestInfoData") RequestInfoData
requestInfoData,
							BindingResult result, RedirectAttributes
redirectAttributes){
```

```
            if(result.hasErrors()){
                  return "requestInfo";
            }

            redirectAttributes.addAttribute("name",requestInfoData.getName());
            redirectAttributes.addAttribute("email",requestInfoData.getEmail());

            redirectAttributes.addFlashAttribute("requestInfoData", requestInfoData);


            return "redirect:requestInfoSuccessRedirect";
      }
```

Can you tell What is difference between addAttribute and addFlashAttribute?(See my video
)

Deploy the application and submit the form with correct data. You should see that you are redirected to requestInfoSuccessRedirect and there will be name and email as query parameters. Flash Attribute will be stored in session until redirect and will be removed after succesful rendering.

Congratulations !! You know how to Handle Forms in Spring MVC