LAB- Spring Mvc -Validation

In this lab you will be understanding how to do form validation

STEP -1

1) Open RequestInfoController and observe the method requestInfoPost() which handles post requests for /requestInfo.htm

2) When post request comes, the form data is binded to RequestInfoData object. We want to apply some validation rules for on that object. Annotate the method argument with @Valid .

If a model attribute is annotated with **@Valid** , then validators will be invoked to validate that object based on annotations.

So, we need to annotate the fields of RequestInfoData as per how you want to validate the form data.

Open RequestInfoData.java.

We want to make sure that name is not empty . So, annotate it with @NotEmpty
W want email to be a valid email address. So, annotate it with @Email

we want age to be between 0 and 100 so annotate age with @Min(0) and @Max(100)

We want dob to be a date in dd-MM-yyy format . So, annotate it with
@DateTimeFormat(pattern="dd-MM-yyyy")

To write error messages, we have already configured a messageSource with base package as errormessages.

Open errormessages.properties and observe that we have written error messages to be displayed. (If you dont understand how to write keys, please see my video on again).

Deploy the application and try giving wrong values for the fields for which you have applied validation. You should see the corresponding error messages.

Did you register any Validator beans for validation?

No

So, as you have written <mvc:annotation-driven /> validator is registered.

WAY2LEARN

Custom Validator

1) We want a validator which can validate size of a given integer. We already have @Min and @Max for this purpose. But we want a single annotation for both.

So, create an annotation as shown below :

```
@Target( { ElementType.FIELD })
@Retention(RetentionPolicy.RUNTIME)
public @interface MySize {

	String message() default " Invalid!!";
	Class<?> []groups() default {};
	Class<? extends Payload>[] payload() default {};

	int min();
	int max();
}
```

Now create a class implementing ContraintValidator as shown below :

```
public class MySizeValidator  implements ConstraintValidator<MySize, Integer>{

	int min;
	int max;

	@Override
	public void initialize(MySize mySize) {
		min=mySize.min();
		max=mySize.max();

	}

	@Override
	public boolean isValid(Integer value, ConstraintValidatorContext context) {
		System.err.println("MySizeValidator.isValid()");
		if(value<min || value>max){
			return false;
		}
		return true;
	}

}
```

understand the logic inside it

Now got to MySize Annotation and add the below annotation on top of it.

```
@Constraint(validatedBy=MySizeValidator.class)
```

This says that MysizeValidator has the logic for validation when ever @MySize is applied to a field.

Now got to RequestInfoData and apply MySize annotation on age field to make sure that age is between 0 and 100.

Deploy the application and test it.

Congratulations !! You know how to do form validation