In spring MVC Supports multiple ways to validate the form data. Some of them are below.

# Java Validation API.

* To use Java Validation API we need to add below two jars or maven dependency in the application
* validation-api-1.0.0.[JSR-303]
* hibernate-validator-4.2.0.Final
* Java Validation API suggests us to use annotation on the model class for each attribute to specify the validation constraints.

public class Customer {

    @NotNull

    @Email

    private String emailId;

    @NotNull

    @Size(min=8,max=15)

    private String password;

//getters  and setters

}

**Notes**:

* **@NotNull** to specify that emailId and password fields should not be null.
* **@Email** – specifies that , emailId field is getting validated as per the email standard provided by Hibernate validator.
* **@Size** specifies that password field should be minimum of 8 characters and maximum of 15 characters.
* There are many other annotations to validate the fields in the form.
* In the controller class, specify the model object with @Valid annotation to ensure that form values gets binded with this model object and do the validation.

@Controller

public class LoginController {

     @RequestMapping(value = "/doLogin", method = RequestMethod.POST)

        public String doLogin(@Valid User user, BindingResult result) {

            // login logic here

        }

}

* If any validation error occurs as per our validation defined in the model class, then BindingResult will have the errors. So we can check the binding results for any errors.
* We can display the error messages on the jsp by using spring’s form errors tag as follows

<form:errors path="emailId" />

* We can also specify the error message while specifying the validation constraints in the model class with annotation as below.

@NotEmpty(message = "Email Id is Required.")

private String emailId;

* We can specify the error messages in the properties file. In property file we have to follow the below syntax.

<ConstraintName>.<modelAttributeName>.<propertyName>=<validation error message>

NotEmpty.customer.emailId=Email Id is Required.

Sample Code:

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>

<form:form method="POST" modelAttribute="customer" action="doLogin">

        <table>

            <tr>

                <td>Enter your E-mail:</td>

                <td><form:input path="emailId" /></td>

                <td><form:errors path="emailId" cssStyle="color: #ff0000;" /></td>

            </tr>

            <tr>

                <td><input type="submit" name="submit" value="Register"></td>

            </tr>

        </table>

    </form:form> **login.jsp**

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class LoginController {

     @RequestMapping(value = "/login", method = RequestMethod.GET)

        public String viewLoginPage(Model model) {

          Customer customer = new Customer();

            model.addAttribute("customer", customer);

            return "login";

        }

     @RequestMapping(value = "/doLogin", method = RequestMethod.POST)

        public String doLogin(@Valid Customer customer, BindingResult result,Model model) {

         model.addAttribute("customer",customer);

          if(result.hasErrors()){

              return "login";

          }

          return "home";

        }

} LoginController.java

NotEmpty.customer.emailId=Email Id is required.

Email.customer.emailId=valid email id is required.

NotEmpty.customer.password=Password is required.

Size.customer.password=Password should be minimum of 8 and maximum of 15 characters.

messages.properties

# Custom validator:

* We will not get annotations for all our **business validations** on the form fields.
* To write custom validator we need to follow below steps.
* Create the custom validator class for model class on which we validate and implement the validator interface, make custom validator class as spring bean.
* Override supports(Class clazz) method
* Override validate(Object target,Errors errors) method
* After this we need to inject this custom validator in the controller class and call its validate() method
* Then check the binding result for any errors and return the appropriate view.
* We will have annotations for basic validation and we will use custom validator for business validation.
* Sample for Custom validator.

public class Customer {

private int age;

//getters and setters

}

import org.springframework.stereotype.Component;

import org.springframework.validation.Errors;

import org.springframework.validation.ValidationUtils;

import org.springframework.validation.Validator;

@Component

public class CustomerValidator implements Validator {

    public boolean supports(Class<?> clazz) {

        return Customer.class.isAssignableFrom(clazz);

    }

    public void validate(Object target, Errors errors) {

        Customer customer = (Customer)target;

        int age = customer.getAge();

        String password = customer.getPassword();

        String confPassword = customer.getConfPassword();

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "age", "customer.age.empty");

        //Business validation

        if(!password.equals(confPassword)){

            errors.rejectValue("password","customer.password.missMatch");

        }

        if(age < 18 || age > 60){

            errors.rejectValue("age", "customer.age.range.invalid");

        }

    }

}

* We can validate the fields by using **ValidationUtils OR Errors class.**
* Finally errors are getting added to the Binding result object.

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class RegistrationController {

    @Autowired

    CustomerValidator customerValidator;

     @RequestMapping(value = "/register", method = RequestMethod.GET)

        public String viewRegistrationPage(Model model) {

          Customer customer = new Customer();

            model.addAttribute("customer", customer);

            return "register";

        }

     @RequestMapping(value = "/doRegister", method = RequestMethod.POST)

        public String doLogin(@Valid Customer customer, BindingResult result,Model model) {

         model.addAttribute("customer",customer);

         customerValidator.validate(customer, result);

          if(result.hasErrors()){

              return "register";

          }

          return "home";

        }

    public CustomerValidator getCustomerValidator() {

        return customerValidator;

    }

    public void setCustomerValidator(CustomerValidator customerValidator) {

        this.customerValidator = customerValidator;

    }

}

* In the above controller , we have **autowired** the customerValidator and called validate() method by passing model object and BindingResult object, which will validate the fields.

NotEmpty.customer.emailId=Email Id is required.

Email.customer.emailId=valid email id is required.

Size.customer.password=Password should be minimum of 8 and maximum of 15 characters.

Size.customer.confPassword=Password should be minimum of 8 and maximum of 15 characters.

customer.age.empty = Age is required

customer.age.range.invalid = Age should be between 18 to 60

customer.password.missMatch = password and confirm password do not match

# initBinder in Custom validator:

* When we use **initBinder** for our custom validator , default spring validator will not called hence, we need to add all the validations in the custom validator.
* We need to remove all the annotation from model class.
* Need to write custom logic to validate each field .
* Need to write initBinder(WebDataBinder wdb) in controller class.
* **InitBinder** will call the custom validator automatically on form submit. So no need to call validator class manually from controller.

import org.springframework.stereotype.Component;

import org.springframework.validation.Errors;

import org.springframework.validation.ValidationUtils;

import org.springframework.validation.Validator;

@Component

public class CustomerValidator implements Validator {

    public boolean supports(Class<?> clazz) {

        return Customer.class.isAssignableFrom(clazz);

    }

    public void validate(Object target, Errors errors) {

        Customer customer = (Customer)target;

        int age = customer.getAge();

        String password = customer.getPassword();

        String confPassword = customer.getConfPassword();

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "emailId", "customer.emailId.empty");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "age", "customer.age.empty");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "password", "customer.password.empty");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "confPassword", "customer.confPassword.empty");

        //Business validation

        if(!password.equals(confPassword)){

            errors.rejectValue("password","customer.password.missMatch");

        }

        if(age < 18 || age > 60){

            errors.rejectValue("age", "customer.age.range.invalid");

        }

        if(!customer.getEmailId().matches("^[\_A-Za-z0-9-\\+]+(\\.[\_A-Za-z0-9-]+)\*@" + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)\*(\\.[A-Za-z]{2,})$")){

            errors.rejectValue("emailId", "customer.emailId.invalid");

        }

    }

}

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class RegistrationController {

    @Autowired

    CustomerValidator customerValidator;

    @InitBinder

    public void initBinder(WebDataBinder webDataBinder){

        webDataBinder.setValidator(customerValidator);

    }

     @RequestMapping(value = "/register", method = RequestMethod.GET)

        public String viewRegistrationPage(Model model) {

          Customer customer = new Customer();

            model.addAttribute("customer", customer);

            return "register";

        }

     @RequestMapping(value = "/doRegister", method = RequestMethod.POST)

        public String doLogin(@Valid Customer customer, BindingResult result,Model model) {

         model.addAttribute("customer",customer);

         //customerValidator.validate(customer, result);

          if(result.hasErrors()){

              return "register";

          }

          return "home";

        }

    public CustomerValidator getCustomerValidator() {

        return customerValidator;

    }

    public void setCustomerValidator(CustomerValidator customerValidator) {

        this.customerValidator = customerValidator;

    }

}

customer.emailId.empty=Email Id is required.

customer.emailId.invalid=valid email id is required.

customer.password.empty=Password is required

customer.confPassword.empty=Confirm Password is required

customer.age.empty = Age is required

customer.age.range.invalid = Age should be between 18 to 60

customer.password.missMatch = password and confirm password do not match