MVC Validation des formulaires

Champs Requis

Validation : pas à pas

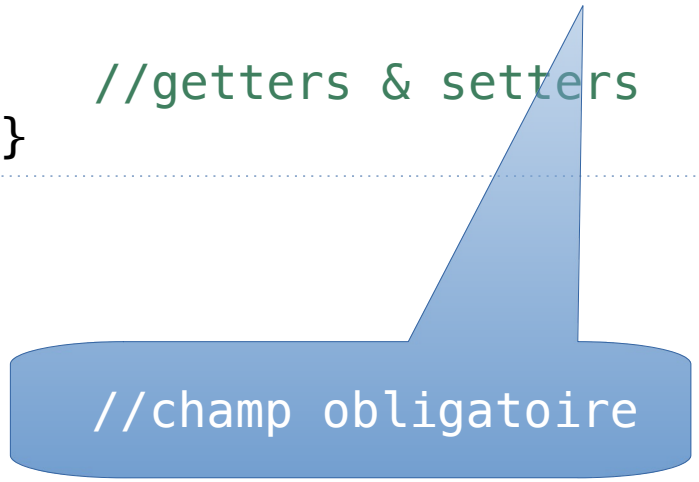
- Dans la classe d'une entité qui sera manipulée par un formulaire : Ajouter les règles de validation et les messages d'erreurs.
- Insérer les balises d'erreur dans le formulaire
- Dans le contrôleur qui réceptionne le formulaire ,on va procéder au contrôle de la validité des champs du formulaire soumis.
- Afficher le résultat après soumission d'un formulaire valide

Valider l'objet à valider ?

```
public class Personnage {

    private String prenom;  // taille de la chaine <20
    private String   nom;// obligatoire
    private enum genre; //homme ou femme
    private String role; //paysan, magicien,  militaire ...
    private int pointDeVie; // de 1 à 10
    private List<String> equipements; //épee, couteau, potion, ...
    private LocalDate creaDate; //compris entre 1920  et 2020.
    private boolean majeur;//majorité à 18 ans

    //getters & setters
}
```

@NotNull, @Size, @max, @min

```java
public class Personnage {

    private String prenom;
    private String   nom;

    //getters & setters
}
```

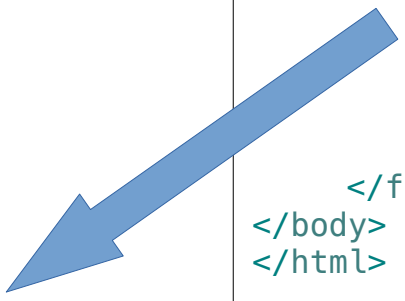



```java
public class Personnage {


private String prenom;

@NotNull
@Size(min=1,message="nom obligatoire)
private String   nom;

//getters & setters
}
```

Insérer des balises <form:errors ...> dans le formulaire pour afficher les messages d' erreurs

personnage-form.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head><style> .error{color:red} </style></head>
<body>
     <form:form action="processForm" modelAttribute="personnage">

          Prénom: <form:input path="prenom" />

          <br><br>

          Nom (*): <form:input path="nom" />
          <form:errors path="nom" cssClass="error"/>

          <br><br>

          <input type="submit" value="Valider" />
     </form:form>
</body>
</html>
```



```
Nom: <form:input path="nom" />
<form:errors path="nom" cssClass="error"/>
```

• contrôler la validité des champs du formulaire soumis

```
@Controller
@RequestMapping("/personnages")
public class PersonnageControleur {

    @RequestMapping("/afficheForm")
    public String showForm(Model theModel){
        theModel.addAttribute("personnage",new Personnage());
        return "personnage-form";
        }


    @RequestMapping("/processForm")
    public String processForm(
        @Valid  @ModelAttribute("personnage") Personnage unPersonnage,
            BindingResult leBindingResult){

            if(leBindingResult.hasErrors()) {return "personnage-form";}
            else        return "personnage-vue";
            }
}
```

Création d'une vue qui affiche un résultat valide

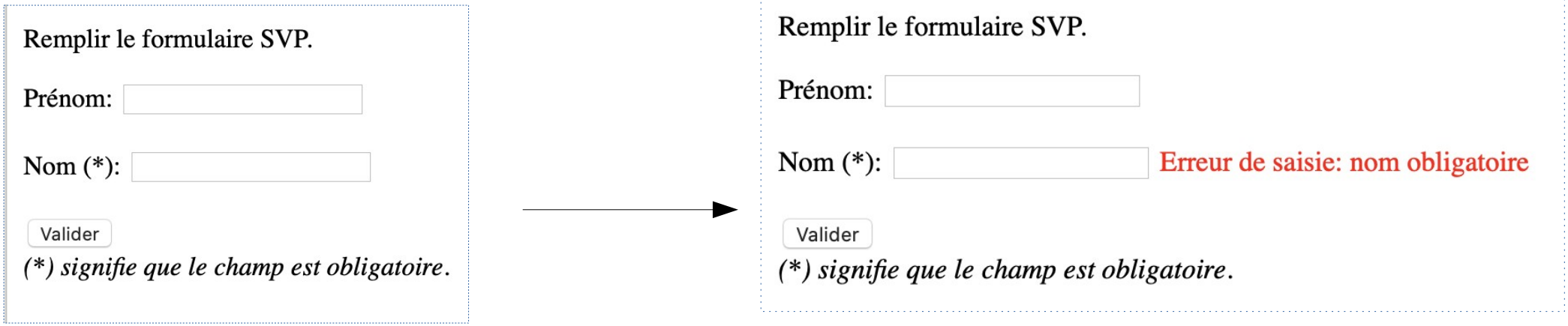
personnage-vue.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<title>personnage valide - vue</title>
</head>
<body>
Le personnage soumis est  valide  : ${personnage.nom} $
{personnage.prenom}
</body>
</html>
```

main-menu.jsp

```
<a href="personnages/affichеForm">Formulaire Personnage</a>
```

J'execute pour tester la validation



ON A LE RÉSULTAT ATTENDU

On essai de trouver un défaut

- si nous testons avec un nom composé uniquement d'espace
- le formulaire passe la validation
- Comment remédier à ce cas

D'abord ajoutons un log de monitoring dans le contrôleur

```
@Controller
@RequestMapping("/personnages")
public class PersonnageControleur {

…

@RequestMapping("/processForm")
public String processForm(
@Valid  @ModelAttribute("personnage") Personnage unPersonnage,
BindingResult leBindingResult){

System.out.println("Nom: |" + unPersonnage.getNom()+"|");

 if(leBindingResult.hasErrors()) return "personnage-form";
 else return "personnage-vue";


}
}
```

Advanced Spring solution : @InitBinder

- On va appliquer la méthode trim() sur les champs du formulaire
- L'annotation @InitBinder va réaliser un traitement sur chaque requête entrante adressée au contrôleur PersonnageControleur.java
- Dans ce traitement nous allons spécifier l'action de trim().

@InitBinder

```
@Controller
@RequestMapping("/personnages")
public class PersonnageControleur {

// déclarer un  initBinder  pour traiter trim() les chaines entrantes
// résoudre un souci de validation des espaces dans les chaînes

@InitBinder
public void initBinder(WebDataBinder dataBinder) {
StringTrimmerEditor stringTrimmerEditor=new StringTrimmerEditor(true);
dataBinder.registerCustomEditor(String.class, stringTrimmerEditor);
}

… etc
```

On teste à nouveau la soumission d'un formulaire avec :

- **un nom vide,**
- **un nom rempli**
- **et un nom composé d'espaces.**

@NotNull

Il s'agit d'un message d'erreur par défaut pour les champs soumis mais qui valent null

- Nous pouvons spécifier un message particulier pour expliquer l'erreur à l'utilisateur du formulaire

- `@NotNull( message="champ obligatoire")`

Maintenant , passons aux autres règles de validation ...