



Spring Formulaire

Les balises



Data binding

Traduction de "to bind" :

- attacher, unir une chose à une autre, relier.

Définition de "data binding" :

En anglais : Automatically set and retrieve data from java object.

Définition en français :

C'est un moyen de lier la partie vue à la partie logique



Spring propose des balises de formulaires qui permettent le data binding

- Grâce aux balises fournies par spring pour les formulaires , nous pourrons réaliser une liaison entre les données présentes coté vue et les données correspondantes coté java.
- Cette liaison facilitera la saisie et l'édition, la validation des données (aka la vérification que les données renseignées dans la vue sont conformes à des règles que l'on va définir pour chacun des champs de formulaire).

Spring form tags

Balise de formulaire	Description
form:form	Contient un formulaire
form:input	Champ texte
form:textarea	Champ texte multi-ligne
form:checkbox	Case à cocher
form:radio	Bouton radio
form:select	Menu déroulant
...etc	

<https://docs.spring.io/spring/docs/5.0.2.RELEASE/spring-framework-reference/web.html#mvc-view-jsp-tags>

- Pour disposer des balises spring dans une jsp, on doit utiliser au préalable une taglib :

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

Ensuite on développe un formulaire spring

utilisateur-form.jsp

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

<form:form action="processForm" >
Prénom: <form:input />

Nom: <form:input />

<input type="submit" value="valider"/>
</form:form>
```

Schéma du flux à coder

main-menu.jsp

[Formulaire Utilisateur](#)

UtilisateurControleur.java

```
@RequestMapping("/afficheUtilisateurForm")  
@RequestMapping("/processForm")
```

utilisateur-form.jsp

Nom :

Prénom :

Valider

utilisateur-vue.jsp

L'utilisateur est vérifié : Juste Leblanc

Utilisateur

Utilisateur

Utilisateur

UtilisateurControleur.java

- la méthode qui affiche le formulaire, y injecter un objet Utilisateur
- Pourquoi ? Dans un formulaire on ne sait pas créer d'objet, mais on sait les manipuler (éditer, visualiser)
- Le formulaire va servir principalement à éditer ou à créer des Utilisateurs.
- Donc on veut pouvoir transmettre un Utilisateur (entier) depuis ce formulaire vers le contrôleur.
- Il faut donc fournir à la vue un Utilisateur en amont (fut-il vide pour la création) pour pouvoir via ce formulaire le peupler ou l'éditer puis le transmettre au Contrôleur.

Contrôleur (suite)

Model permet de passer des données entre contrôleur et vue

```
@Controller
public class UtilisateurControleur {

    @RequestMapping("/afficheUtilisateurForm")
    public String afficheForm(Model leModel) {
        leModel.addAttribute("utilisateur", new Utilisateur());
        return "utilisateur-form";
    }

    @RequestMapping("/processForm")
    public String processForm(
        @ModelAttribute("utilisateur") Utilisateur unUtilisateur){
        //log la donnée entrante unUtilisateur.getNom();
        return "utilisateur-vue";
    }
}
```

clé

valeur

Il s'agit de la même clé qu'on utilisera dans le formulaire pour référencer l'utilisateur

On revient sur le formulaire spring, configurer le data binding

utilisateur-form.jsp

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

<form:form action="processForm" modelAttribute="utilisateur" >
  Prénom: <form:input path="prenom" />
  <br><br>
  Nom: <form:input path="nom" />
  <br><br>
  <input type="submit" value="valider"/>
</form:form>
```

Nom :

Prénom :

Valider

Explications

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

<form:form action="processForm" modelAttribute="student">
First name: <form:input path="firstName" />
<br><br>
LastName: <form:input path="lastName" />
<br><br>
<input type="submit" value="Submit"/>
</form:form>
```

On termine le contrôleur

```
@Controller
public class UtilisateurControleur {

    @RequestMapping("/afficheUtilisateurForm")
    public String afficheForm(Model leModel) {
        leModel.addAttribute("utilisateur", new Utilisateur());
        return "utilisateur-vue";
    }

    @RequestMapping("/processForm")
    public String processForm(
        @ModelAttribute("utilisateur") Utilisateur unUtilisateur){
        //log la donnée entrante unUtilisateur.getNom();
        return "utilisateur-vue";
    }
}
```

Clé ou nom de l'attribut

Argument de la méthode

L'objet est peuplé par spring en arrière plan avec les données du formulaire

La classe Utilisateur.java

```
public class Utilisateur {  
  
    private String prenom;  
    private String nom;  
  
    public String getPrenom() {  
        return prenom;  
    }  
    public void setPrenom(String prenom) {  
        this.prenom = prenom;  
    }  
    public String getNom() {  
        return nom;  
    }  
    public void setNom(String nom) {  
        this.nom = nom;  
    }  
}
```

Utilisateur-vue.jsp

```
<html>
<head>
<meta charset="UTF-8">
<title> Utilisateur - Vue</title>
</head>
<body>
L'utilisateur est vérifié : ${utilisateur.prenom}
${utilisateur.nom}
</body>
</html>
```

Clé ou
nom de l'attribut

Clé ou
nom de l'attribut

Synthèse

Spring MVC balises de Formulaire
@ModelAttribute
Spring MVC Data Binding

- Créer une classe pour le modèle (Utilisateur ici)
- Créer un contrôleur avec qui les pages vont communiquer
- Créer un formulaire dans une page
- Coder en java les méthodes qui affichent et traitent le formulaire
- Créer la vue de confirmation des traitements réalisés sur l'objet modèle.