Labo Spring 3

Formulaire

Objectif : Créer des formulaires et les valider

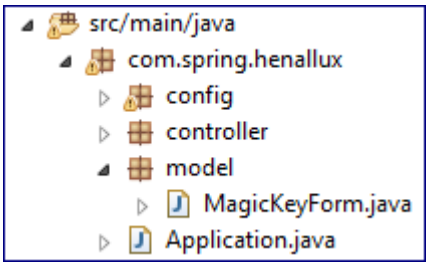
Un formulaire va être ajouté dans la page welcome.jsp afin de demander à l'utilisateur une clé (un code) permettant d'accéder à la suite de l'application Web.

Etape 1 – Classe modèle *MagicKeyForm*

Créez le package *model* dans *src/main/java/com.spring.henallux*.

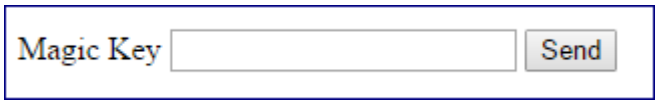
Créez-y la classe *MagicKeyForm* qui contient une variable d'instance privée de type String appelée *magicKey*.

Prévoyez les gettor/settor publiques pour cette variable d'instance ainsi qu'au moins le constructeur sans argument.



Etape 2 – Formulaire dans la page *welcome.jsp*

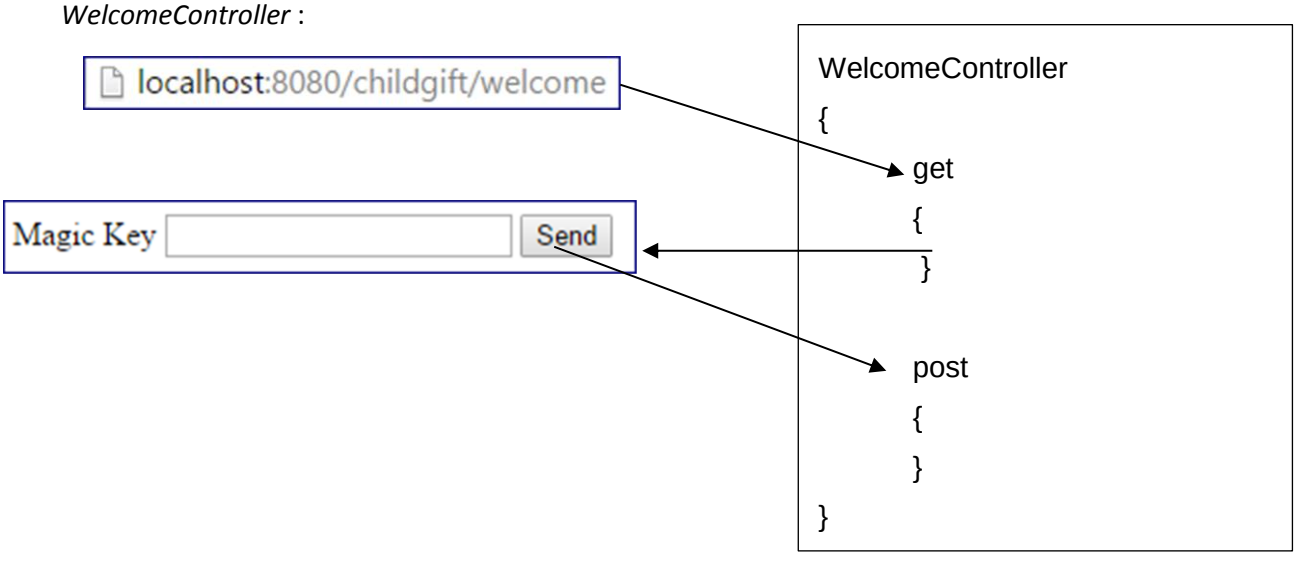
Ajoutez le formulaire ci-dessous dans la page *welcome.jsp.* Placez dans le formulaire un bouton appelé *Send*.



Utilisez la balise <form :form> pour créer le formulaire ainsi que les balises <form :label> pour le label, <form :input> pour la zone de saisie et <form :button> pour le bouton.

Etape 3 – Classe *WelcomeController*

Un clic sur le bouton *Send* devra avoir pour effet d'appeler la méthode *post* de la classe *WelcomeController* :



Ce controller doit créer un objet de la classe *MagicKeyForm* et être implémenté de sorte que la valeur introduite par l'utilisateur soit placée dans la variable *magicKey* de cet objet.

Pour ce faire, prévoyez l'instruction de création et d'ajout de cet objet au *Model* dans la méthode *get* :

```
@Controller
@RequestMapping(value="/welcome")
public class WelcomeController {

    @RequestMapping(method=RequestMethod.GET)
    public String home(Model model) {
        model.addAttribute("magicKeyForm", new MagicKeyForm());
        return "integrated:welcome";
    }
```

Récupérez ensuite cet objet du *Model* comme argument de la méthode *post* :

```
@RequestMapping(value="/send", method=RequestMethod.POST)
public String getFormData(Model model,
                          @ModelAttribute(value="magicKeyForm") MagicKeyForm form) {
```

La variable *magicKey* de l'objet *form* sera remplie avec la valeur introduite par **l'utilisateur à condition que** dans la balise *<form :form>* de la page jsp les bonnes valeurs aient été placées dans les attributs *action* (path pour appeler la méthode *post* du controller) et *modelAttribute* (lien vers l'objet modèle).

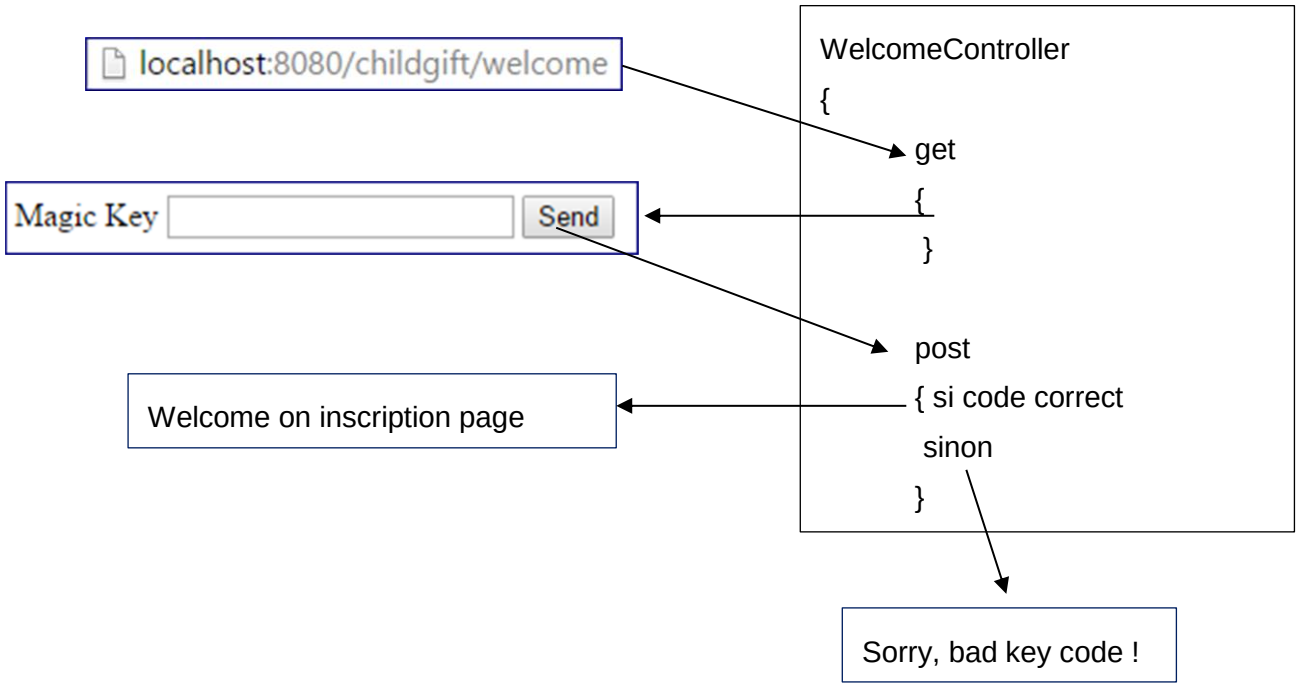
```
<form:form id="form"
           method="POST"
           action ="/childgift/welcome/send"
           modelAttribute="magicKeyForm">
```

De même, il faut que les champs du formulaire soient reliés aux variables d'instance de l'objet modèle via l'attribut *path*.

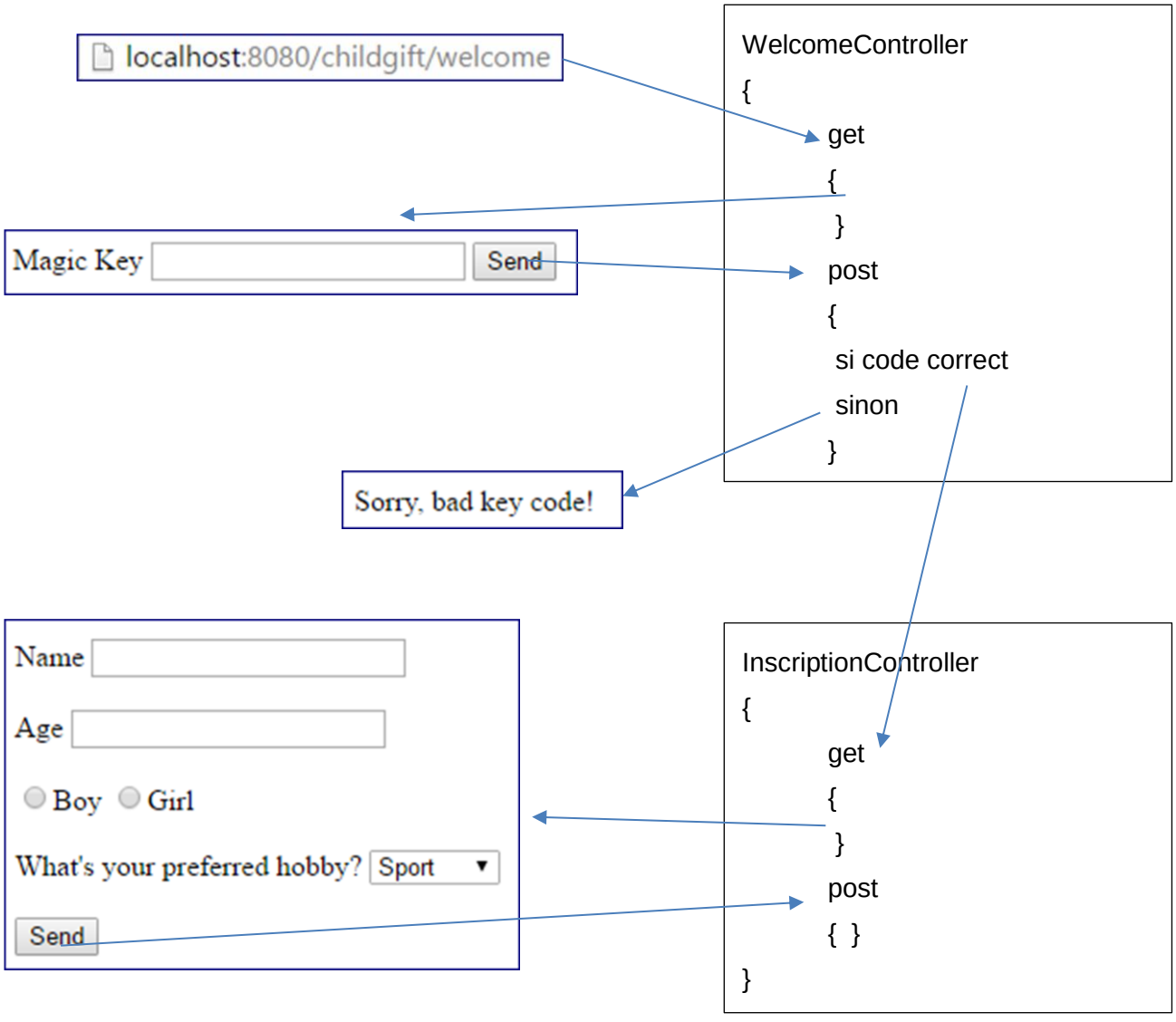
```
<form:input path="magicKey"/>
```

La méthode *post* doit ensuite vérifier que la valeur introduite par l'utilisateur (qui se trouve donc dans la variable *magicKey* de l'objet *form*) est une valeur correcte. Si oui, la page *userInscription.jsp* est affichée. Cette page contient pour l'instant seulement un message de bienvenue. Si non, une page d'erreur (*keyError.jsp*) est affichée.

Idéalement, la valeur introduite par l'utilisateur devrait être comparée à des valeurs stockées dans une base de données (cf exercice ultérieur). Pour l'instant, la valeur introduite est juste comparée à une liste de valeurs hardcodée dans le controller.



Dans la suite de l'exercice, la page *userInscripton* va contenir un formulaire qui permettra de saisir les coordonnées de l'enfant (étapes 4 et 5). Par conséquent, un nouveau controller va gérer la page *userInscription*, soit la classe *InscriptionController* (étape 6). La **méthode *post* de *WelcomeController*** ne devra plus retourner directement la page *userInscription*, mais **faire appel à la méthode *get de InscriptionController*** (étape 7).



Etape 4 – Classe modèle *User*

Dans le package *model* , créez la classe *User* qui contient les variables d'instance privées suivantes :

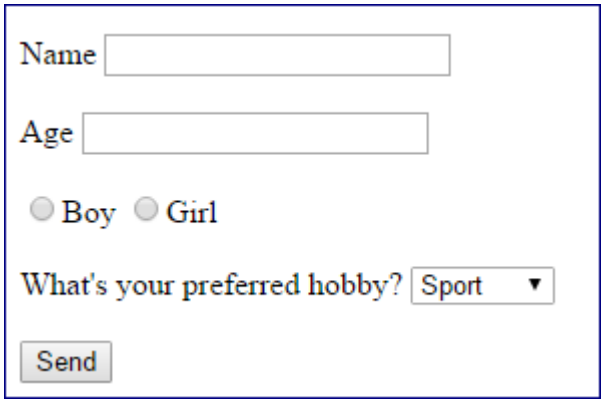
- name : de type String
- age : de type **Integer**
- male : de type **Boolean**
- hobby : de type String

Prévoyez les gettor/settor publiques ainsi qu'au moins le constructeur sans argument.

Etape 5– Page *userInscription*

Adaptez la page *userInscription.jsp*.

Celle-ci doit contenir le formulaire suivant :



Pour les boutons radio, utilisez la balise *<form :radiobutton>* avec les attributs *path* (cf DataBinding), *value* ("true" ou "false") et *label* ("Boy" ou "Girl").

Pour la liste, utilisez la balise *<form:select>* avec l'attribut *path* (cf DataBinding) et *<form:option>* avec les attributs *value* (valeur réelle qui sera récupérée dans la variable d'instance de l'objet modèle) et *label* (valeur affichée dans la page). Placez-y au moins les hobbies suivants : *Sport, Nature, Reading* et *Music.*

```
<form:select path="...">
        <form:option value="..." label="..." />
        <form:option value="..." label="..." />
        <form:option value="..." label="..." />
</form:select>
```

Etape 6 – Classe *InscriptionController*

Créez la classe *InscriptionController* dans le package *controller*.

Dans la méthode *get*, créez un objet de la classe *User*, placez-le dans le *Model* et retournez la page *userInscription.jsp*.

Implémentez la méthode *post* de sorte que les valeurs introduites par l'utilisateur dans les champs du formulaire soient placées dans les variables d'instance de l'objet modèle de type User (quand clic sur le bouton *send*).

Etape 7 - Redirection

Attention, la page *userInscription.jsp* contient désormais un formulaire qui attend un objet modèle. **Il faut donc que la méthode *get* du controller soit appelée avant chaque affichage de la page *userInscription*** pour que l'objet modèle soit créé.
Il faut donc **modifier la méthode *post* de *WelcomeController*** !!! La méthode *post* ne retourne plus directement une page jsp (*return "integrated :userInscription*") mais doit appeler le controller de cette page. L'instruction à écrire est donc une **redirection** vers un controller.

Si le *path* pour appeler *InscriptionController* est *"/inscription*",

```
@Controller
@RequestMapping(value="/inscription")
public class InscriptionController {
```

alors, le retour de la méthode *post* de *WelcomeController* doit être "***redirect:/inscription***".

```
@Controller
@RequestMapping(value="/welcome")
public class WelcomeController {

    @RequestMapping(value="/send", method=RequestMethod.POST)
    public String getFormData(Model model,
                              @ModelAttribute(value="magicKeyForm") MagicKeyForm form) {
        ...
        else return "redirect:/inscription";
    }
```

Testez que les valeurs introduites par l'utilisateur dans les champs du formulaire sont bien placées dans les variables d'instance de l'objet modèle de type *User* après que l'utilisateur a cliqué sur le bouton *send*. Par exemple en affichant à la console (juste le temps du test) les valeurs des variables d'instance de l'objet modèle de type *User*.

Etape 8 – Service *HoobiesService*

La liste des hobbies peut être remplie avec des valeurs fournies par un service (ultérieurement provenant d'une base de données).

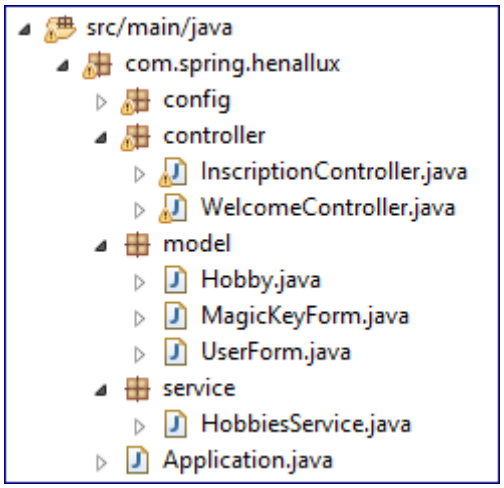
Créez la classe *Hobby* dans le package *model*.

Cette classe contient les variables d'instance privées de type String : *id* et *name*.

Créez le package *service* dans *src/main/java/com.spring.henallux*.

Créez-y la classe *HobbiesService* qui contient une variable d'instance privée de type *ArrayList<Hobby>*. Prévoyez les gettor/settor publiques pour cette variable d'instance ainsi qu'au moins le constructeur sans argument. Granissez cette liste dans le constructeur sans argument de *HobbiesServices* (ajoutez au moins 4 hobbies).

Cette classe doit être annotée *@Service*. Ceci permettra d'y avoir accès dans d'autres classes par **injection de dépendance**.




```java
import java.util.ArrayList;
import org.springframework.stereotype.Service;
import com.spring.henallux.model.Hobby;

@Service
public class HobbiesService {
    private ArrayList<Hobby> hobbies;
```

Dans *InscriptionController*, récupérez par injection de dépendance une référence vers un objet de la classe *HobbiesService*. Ajoutez la liste des hobbies fournie par ce service comme attribut dans le *Model*, afin de pouvoir avoir accès à la liste des hobbies dans la page *userInscription.jsp.*

```java
@Controller
@RequestMapping(value="/inscription")
public class InscriptionController {

    @Autowired
    private HobbiesService hobbiesService;

    @RequestMapping(method=RequestMethod.GET)
    public String home(Model model) {
        model.addAttribute("hobbies",hobbiesService.getHobbies());
```

```jsp
<form:select path="hobby">
      <form:options items="${hobbies}" itemValue="id" itemLabel="name" />
</form:select>
```

Etape 9 – Validation du formulaire

Les champs du formulaire peuvent être validés automatiquement.

Validations à imposer au formulaire de la page *userInscription* :

- Entre 4 et 15 caractères pour le nom
- Age obligatoire et compris entre 1 et 12 ans

Ajoutez une dépendance dans le fichier pom.xml :

```xml
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
</dependency>
```

Annoter les variables d'instance de la classe modèle *User* que vous souhaitez valider. Utilisez *@NotNull, @Size* (avec les attributs *min* et *max*) pour les chaînes de caractères, *@Min ou @Max* (avec l'attribut *value*) pour les nombres …

Dans la méthode *post* de *InscriptionController*, annoter l'attribut modèle de type *User* avec *@Valid* .

Déclarez un argument de type *BindingResult* dans la méthode *post*. Testez ce dernier. S'il n'y a pas d'erreur, la page *gift.jsp* est affichée. Cette page contient pour l'instant seulement un message de félicitation. S'il y a des erreurs, réaffichez la page *userInscription*.

```
@RequestMapping(value="/send", method=RequestMethod.POST)
public String getFormData(Model model,
                          @Valid @ModelAttribute(value="user") User user,
                          final BindingResult errors) {
    if (!errors.hasErrors())
         {
```