Atelier Framework Spring-04
Thymeleaf – Partie 04

Objectifs

- **Fonctionnalités de base**
  - Fonction de recherche
  - Fonction d'insertion et validation

A. Formulaire de recherche

1. Prendre une copie du projet «**jpa-spring-boot-Thymeleaf3**» et le nommer «**jpa-spring-boot-Thymeleaf4**».
2. Ajouter un formulaire au début de la vue « **produits.html** » pour réaliser une recherche selon la propriété « **designation** ». Ainsi, ajouter un bloc « **div** » contenant ce formulaire. Prendre la nouvelle version de la vue :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8"/>
<title>Liste des Produits</title>
<link rel= "stylesheet"
      type ="text/css"
      href= "../static/css/bootstrap.min.css"
      th:href="@{/css/bootstrap.min.css}"
      />
</head>
<body>
<h3>Liste des produits</h3>

<div class="container">
<form th:action="@{index}" method="get" >
   <label>Mot Clé (désignation):</label>
   <input type="text" name ="motCle" th:value="${motCle}"/>
   <button class ="btn btn-primary"> Chercher </button>
</form>
</div>

<div class ="container spacer">
<table class ="table table-striped">
  <thead>
   <tr>
```

```
   <th>ID</th><th>Désignation</th><th>Prix</th><th>Quantité</th><th>Date
Achat</th><th>Action</th>
  </thead>
  <tbody>
        <tr th:each="p:${pageProduits.content}">

          <td th:text="${p.id}" ></td>
          <td th:text="${p.designation}" ></td>
          <td th:text="${p.prix}" ></td>
          <td th:text="${p.quantite}" ></td>
          <td th:text="${p.dateAchat}" ></td>
        </tr>
  </tbody>
</table>
</div>
<div class ="container">
 <ul class ="nav nav-pills"    >
    <li  th:each ="p:${pages}" th:class="${p==pageCourante}?active:''">
     <a th:text ="${p}"  th:href ="@{index(page =${p})}"></a>
    </li>
 </ul>
</div>
</body>
</html>
```

- ✓ La validation du formulaire est réalisée par la méthode « **index** » du contrôleur « **ProduitController** »
- ✓ Le formulaire englobe un champ nommé « **motCle** » qui représente le paramètre passé au contrôleur pour la recherche.

3. Dans l'interface « **ProduitRepository** », déclarer une méthode de recherche par « **designation** » en mode paginé. Prendre la nouvelle version de ce repositoty :

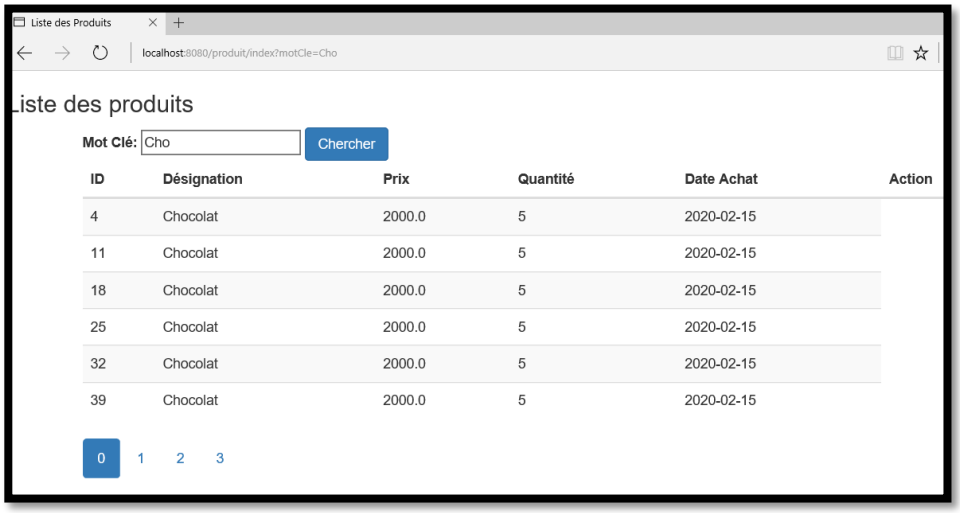
```
package spring.jpa.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import spring.jpa.model.Produit;

public interface ProduitRepository extends JpaRepository<Produit, Long>  {

     // Retourner la page des Produits selon une recherche par designation
  Page<Produit> findByDesignationLike(String mc, Pageable pageable);
}
```

4. Dans le contrôleur, modifier la méthode « **index** » pour recevoir le nouveau paramètre «**motCle**» récupéré du formulaire de recherche de la vue «**produits.html**». Voici la nouvelle version de la méthode «**index**» :

```
// nom logique pour accéder à l'action "index" ou méthode "index
     @RequestMapping (value = "/index")
     public String index (Model model,
          // paramètre pour le numéro de la page (0 par défaut)
          @RequestParam (name="page" , defaultValue ="0") int p,
          // paramètre "motCle"
          @RequestParam (name="motCle", defaultValue="") String mc)
     {
          // récupérer la page numero "p" (de taille 6)
          Page <Produit> pg =
produitRepos.findByDesignationLike("%"+mc+"%", PageRequest.of(p, 6));
          // nombre total des pages
          int nbrePages =pg.getTotalPages();
          // créer un tableau d'entier qui contient les numéros des pages
          int [] pages = new int[nbrePages];
          for(int i= 0 ; i< nbrePages; i++)
          {
               pages[i]=i;
          }
          // placer le tableau dans le "Model"
          model.addAttribute("pages", pages);
          // placer la page des produits dans le "Model" comme un attribut"
          model.addAttribute("pageProduits", pg);
          // retourner le numéro de la page courante
          model.addAttribute("pageCourante",p);
          // retourner la valeur du mot clé
           model.addAttribute("motCle", mc);
          //retourner le nom de la vue WEB à afficher
          return "produits";
     }
```

- ✓ Le contrôleur fait appel à la méthode « **findByDesignationLike** » du repository en lui passant deux arguments :
  1. Le mot clé de recherche
  2. Un objet de type « **PageRequest** » pour indiquer la taille de la page à récupérer et son numéro
- ✓ Le mot clé est stocké comme attribut dans le « **Model** » pour être récupéré dans la vue.

5. Enregistrer la modification et relancer l'exécution.

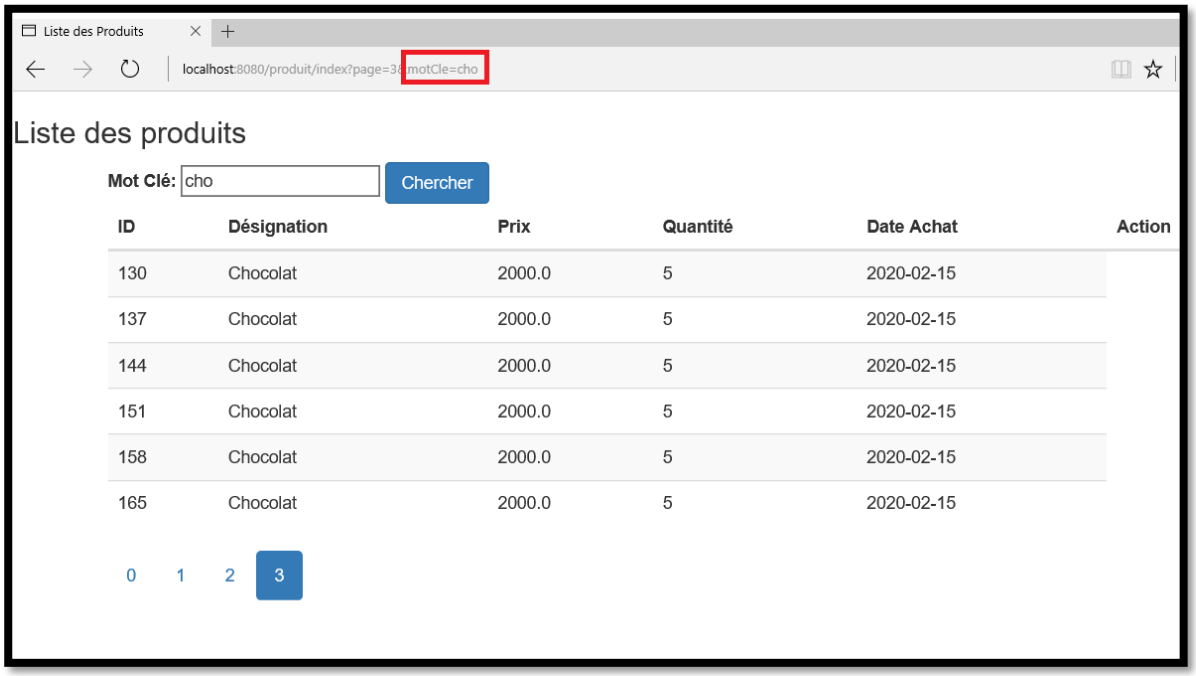


6. En cliquant, dans la barre de navigation, sur une autre puce pour afficher une autre page, remarquer que le mot clé est perdu et la vue affiche tous les produits

- ✓ Il suffit ,donc, d'ajouter dans l'url de chaque puce le paramètre « **motCle** ».
- ✓ Modifier le code de la barre de navigation, au-dessous de la vue, par celui-ci :

```
<div class ="container">
 <ul class ="nav nav-pills"    >
    <li  th:each ="p:${pages}" th:class="${p==pageCourante}?active:''">
     <a th:text ="${p}"  th:href ="@{index(page =${p} , motCle =${motCle})}"></a>
    </li>
 </ul>
</div>
```

7. Enregistrer la modification et relancer l'exécution. Remarquer l'ajout du paramètre « **motCle** » dans l'url :



B. Insertion d'un nouveau produit

8. Définir une nouvelle méthode «**formProduit**» dans le contrôleur « **ProduitController**» pour qui permet d'accéder au formulaire de saisie d'un nouveau produit :

```
@RequestMapping(value="/form",method=RequestMethod.GET)
        public String formProduit (Model model)
        {
            //placer un objet de type "Produit" dans le modèle
              model.addAttribute("produit", new Produit());
            //retourner le nom de la vue WEB à afficher (le formulaire)
              return "formProduit";
          }
```

NB : Ajouter l'instruction « import » nécessaire

9. Créer une nouvelle vue «**formProduit.html**» (dans le dossier « **templates** »)

10. Prendre le code de cette vue :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8"/>
<title>Fiche Produit</title>
<link rel= "stylesheet"
      type ="text/css"
      href= "../static/css/bootstrap.min.css"
      th:href="@{/css/bootstrap.min.css}"
      />
</head>
<body>
<div class="col-md-6 col-sm-6 col-xs-12 spacer col-md-offset-3" >
      <div class="panel panel-default">
            <div class ="panel-heading">Saisie d'un produit  </div>
            <div class= "panel-body">

                  <form th:action="@{save}" method="post" th:object ="${produit}">
                  <div class="form-group">
                     <label class="control-label">Désignation</label>
                     <input class="form-control" type="text"
                           th:field="*{designation}"/>
                  </div>
                  <div class="form-group">
                     <label class="control-label">Prix</label>
                     <input class="form-control" type="text"
                            th:field="*{prix}"/>
                  </div>
                  <div class="form-group">
                     <label class="control-label">Quantité</label>
                     <input class="form-control" type="text"
                            th:field="*{quantite}"/>

                  </div>

                  <div class="form-group">
                     <label class="control-label">Date Achat</label>
```

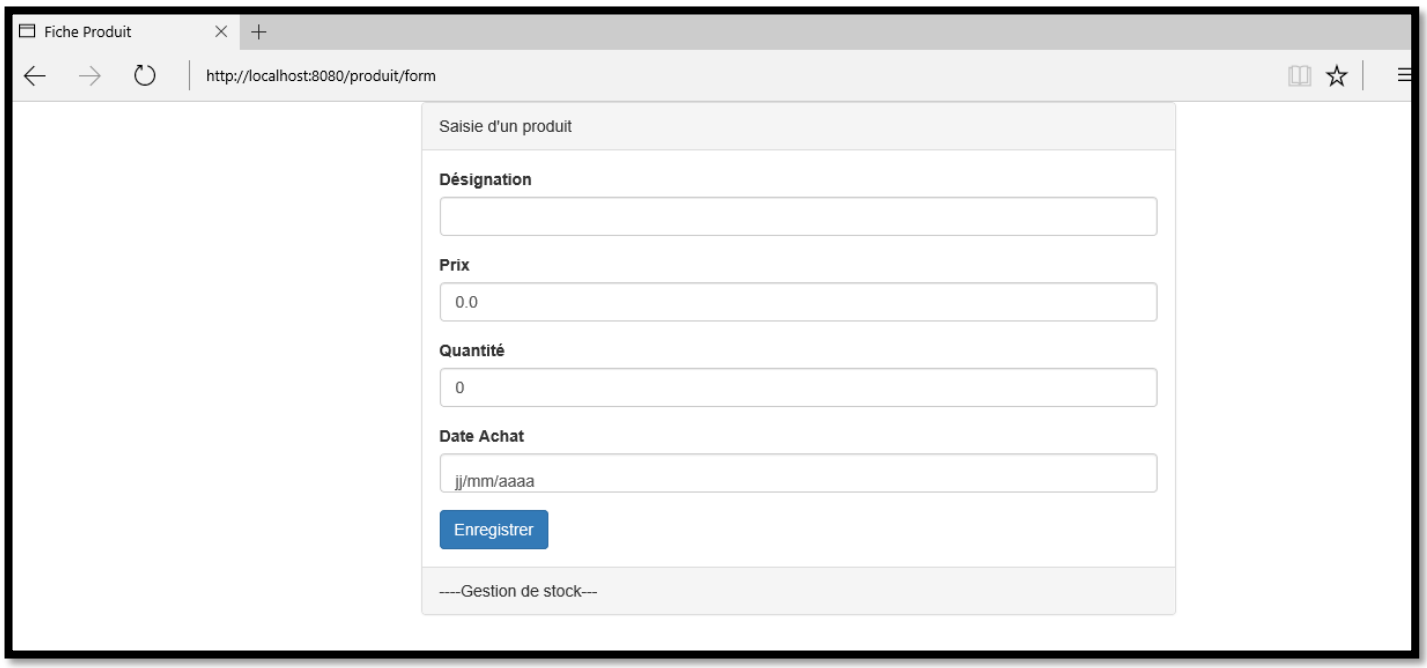
```
                    <input class="form-control" type="date"
                           th:field="*{dateAchat}"/>
                </div>

                <div >

                    <button class="btn btn-primary" type="submit"  >Enregistrer</button>

                </div>
                </form>
            </div>
            <div class="panel-footer"> ----Gestion de stock---        </div>

        </div>

</div>
</body>
</html>
```

- ✓ Tous les champs du formulaire sont associés aux attributs de l'objet « **Produit** » placé comme attribut dans le « **Model** » :
  1. L'entité est associée au formulaire via l'attribut « **th :object** »
  2. Une propriété de l'entité est associée à un champ du formulaire via l'attribut « **th :field** »
- ✓ La soumission du formulaire déclenche l'appel de l'action « **/save** » du contrôleur courant « **ProduitController** ».

11. Enregistrer la modification et lancer l'url suivante :

http://localhost:8080/produit/form

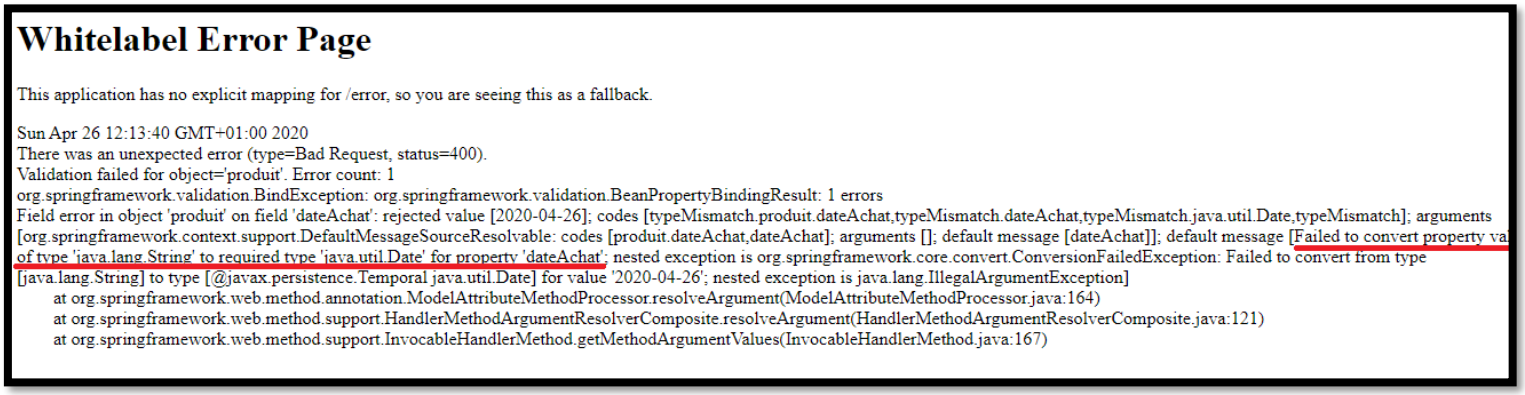
12. Le formulaire est similaire celui-ci :



13. Passer, maintenant, à écrire le code de validation du formulaire. Ajouter dans le contrôleur une méthode « **save** » :

```
@RequestMapping(value="/save",method=RequestMethod.POST)
        public String save (Model model,  Produit produit)
        {
           //enregistrer le produit dans la BD
           produitRepos.save(produit);
           //retourner au formulaire
             return "formProduit";
        }
```

14. Enregistrer la modification, relancer l'exécution et saisir un nouveau produit. Suite à la validation du formulaire, une erreur de non correspondance de type de date est déclenchée :



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun Apr 26 12:13:40 GMT+01:00 2020
There was an unexpected error (type=Bad Request, status=400).
Validation failed for object='produit'. Error count: 1
org.springframework.validation.BindException: org.springframework.validation.BeanPropertyBindingResult: 1 errors
Field error in object 'produit' on field 'dateAchat': rejected value [2020-04-26]; codes [typeMismatch.produit.dateAchat,typeMismatch.dateAchat,typeMismatch.java.util.Date,typeMismatch]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [produit.dateAchat,dateAchat]; arguments []; default message [dateAchat]]; default message [Failed to convert property va
of type 'java.lang.String' to required type 'java.util.Date' for property 'dateAchat'; nested exception is org.springframework.core.convert.ConversionFailedException: Failed to convert from type [java.lang.String] to type [@javax.persistence.Temporal java.util.Date] for value '2020-04-26'; nested exception is java.lang.IllegalArgumentException]
    at org.springframework.web.method.annotation.ModelAttributeMethodProcessor.resolveArgument(ModelAttributeMethodProcessor.java:164)
    at org.springframework.web.method.support.HandlerMethodArgumentResolverComposite.resolveArgument(HandlerMethodArgumentResolverComposite.java:121)
    at org.springframework.web.method.support.InvocableHandlerMethod.getMethodArgumentValues(InvocableHandlerMethod.java:167)

15. Pour remédier à ce problème, il est nécessaire de spécifier à Spring le format de la date à utiliser ; il suffit d'ajouter l'annotation suivante avant la déclaration de l'attribut « **dateAchat** » dans l'entité « **Produit** » :

```
@DateTimeFormat (pattern = "yyyy-MM-dd")
```

NB : Ajouter l'instruction « import » nécessaire

16. Enregistrer la modification, relancer l'exécution et vérifier le succès d'ajout d'un nouveau produit en accédant à la liste des produits (passer à la dernière page) :

http://localhost:8080/produit/index

17. Il est possible de rediriger l'affichage vers la vue « **produits.html** » directement après l'enregistrement du nouveau produit. Prendre la nouvelle version de la méthode « **save** » dans le contrôleur :

```
@RequestMapping(value="/save",method=RequestMethod.POST)
        public String save (Model model,  Produit produit)
        {
            //enregistrer le produit dans la BD
            produitRepos.save(produit);
            //rediriger vers la vue « produits.html »
              return  "redirect:index";
        }
```

18. Enregistrer la modification et relancer l'exécution.

C. Procédure de validation

Pour réaliser un contrôle automatique sur les valeurs saisies dans le formulaire, utiliser le module de validation de Hibernate :

19. Ajouter la dépendance de validation au projet :

```
<dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

20. Ajouter des annotations (appartenant au package « **javax.validation.constraints** ») dans la définition de l'entité « **Produit** » :

- ✓ Ajouter les deux annotations suivantes pour l'attribut « **designation** »

```
@NotNull   // interdire une valeur null
@Size(min=3, max=50)//spécifier la taille acceptée
```

- ✓ Ajouter une annotation pour l'attribut « **prix** »

```
    @DecimalMin("0.1")  // pour spécifier une valeur minimale pour le prix
```

21. Prendre une nouvelle version de la méthode « **save** » dans le contrôleur :

```
@RequestMapping(value="/save",method=RequestMethod.POST)
        public String save (Model model, @Valid Produit produit , BindingResult bindingResult)
        {
              if (bindingResult.hasErrors())
                    // en cas d'erreurs de validation
                    return "formProduit";
             //sinon
            //enregistrer le produit dans la BD
              produitRepos.save(produit);
              //Afficher une page pour confirmer l'enregistrement
              return "confirmation";
        }
```

- ✓ L'annotation « **@Valid** » indique que l'objet est soumis à la procédure de validation
- ✓ Lobjet « **BindingResult** » permet de gérer les erreurs
- ✓ En cas d'erreur, l'affichage est redirigé vers le formulaire de saisie, les erreurs vont être récupérées par le formulaire de saisie dans une zone spécifique
- ✓ En cas de succès, l'affichage est redirigé vers une page de confirmation

22. Prendre le code de la vue «**confirmation.html**» (dans «le dossier « **templates** ») :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8"/>
<title>Liste des Produits</title>
<link rel= "stylesheet"
      type ="text/css"
      href= "../static/css/bootstrap.min.css"
      th:href="@{/css/bootstrap.min.css}"
      />
</head>
<body>
<div class="container" >
    <div class="panel panel-default">
        <div class ="panel-heading">Confirmation  </div>
        <div class= "panel-body">
            <div class="form-group">
               <label>ID:</label>
               <label th:text="${produit.id}"></label>
            </div>
            <div class="form-group">
               <label>Désignation:</label>
               <label th:text="${produit.designation}"></label>
            </div>
            <div class="form-group">
               <label>Prix:</label>
               <label th:text="${produit.prix}"></label>
            </div>
            <div class="form-group">
               <label>Quantité:</label>
               <label th:text="${produit.quantite}"></label>
            </div>

            <div class="form-group">
               <label>Date Achat:</label>
```

```
                <label th:text="${produit.dateAchat}"></label>
            </div>
        </div>
    <div class="panel-footer"> ----Gestion de stock---     </div>

    </div>

</div>
</body>
</html>
```

23. Prendre une nouvelle copie de la vue « **formProduit.html** » (ajout des zones d'affichage des erreurs éventuelles) :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8"/>
<title>Fiche Produit</title>
<link rel= "stylesheet"
      type ="text/css"
      href= "../static/css/bootstrap.min.css"
      th:href="@{/css/bootstrap.min.css}"
      />

  <link rel= "stylesheet"
  type ="text/css"
  href= "../static/css/mon_style.css"
  th:href="@{/css/mon_style.css}"
  />
</head>
<body>
<div class="col-md-6 col-sm-6 col-xs-12 spacer col-md-offset-3" >
      <div class="panel panel-default">
            <div class ="panel-heading">Saisie d'un produit  </div>
            <div class= "panel-body">
                  <form th:action="@{save}" method="post" th:object ="${produit}">
                  <div class="form-group">
                     <label class="control-label">Désignation</label>
                     <input class="form-control" type="text"
                           th:field="*{designation}"/>
                      <span class ="red" th:errors ="*{designation}"></span>
                  </div>
                  <div class="form-group">
                     <label class="control-label">Prix</label>
                     <input class="form-control" type="text"
                            th:field="*{prix}"/>
                     <span class ="red" th:errors ="*{prix}"></span>
</div>
                  <div class="form-group">
                     <label class="control-label">Quantité</label>
                     <input class="form-control" type="text"
                            th:field="*{quantite}"/>
                      <span class ="red" th:errors ="*{quantite}"></span>
```

```
                    </div>

                    <div class="form-group">
                       <label class="control-label">Date Achat</label>
                       <input class="form-control" type="date"
                              th:field="*{dateAchat}"/>
                      <span class ="red" th:errors ="*{dateAchat}"></span>
                    </div>

                    <div >
                       <button class="btn btn-primary" type="submit"  >Enregistrer</button>

                    </div>
                    </form>
                </div>
                <div class="panel-footer"> ---- Gestion de stock ---</div>
            </div>
</div>
</body>
</html>
```

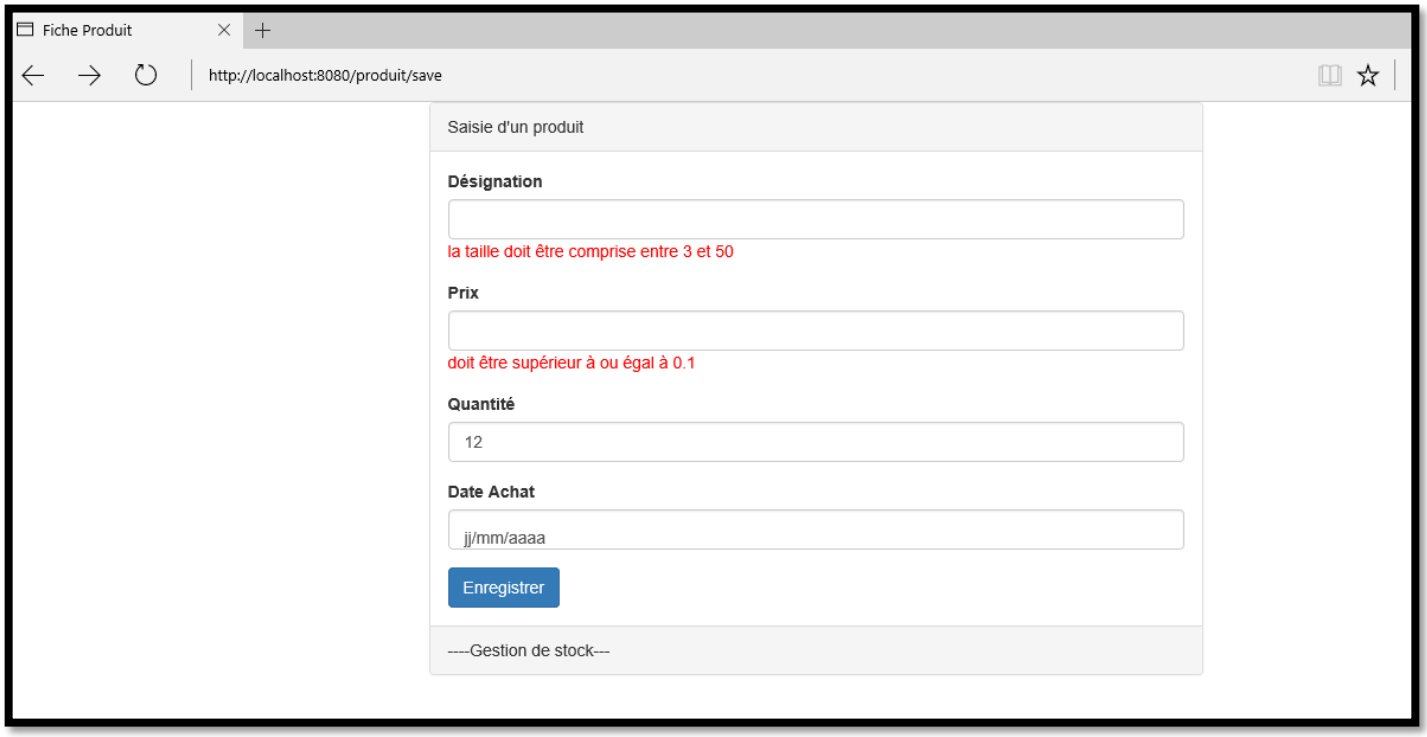
- Remarquer que les erreurs sont gérées dans des balises « **span** » et sont récupérées via l'attribut « **th:errors** » associé à une propriété de l'objet saisi.
- Pour que l'erreur soit affichée en rouge, on utilise une classe CSS « **red** » à définir dans le fichier « **mon_style.css** ».

24. Voici la nouvelle version du fichier « **mon_style.css** » :

```
.red{

   color :red;
}
spacer {
    margin-top: 40px
}
```

25. Enregistrer la modification et lancer l'url suivante (saisir des valeurs erronées et remarquer l'affichage des erreurs):

http://localhost:8080/produit/form



- Voici un exemple de saisie avec succès d'un nouveau produit :