Université Nord Américaine privée
Institut International Technologie

Atelier Framework Spring-04
Thymeleaf – Partie 03

Objectifs

- **Amélioration l'affichage d'une vue**
  - Affichage des données par pagination
  - Navigation entre les pages

A. Affichage des données par page

1. Prendre une copie du projet «**jpa-spring-boot-Thymeleaf2 »** et le nommer «**jpa-spring-boot-Thymeleaf3».**
2. Prendre une nouvelle version du contrôleur « **ProduitController**» pour modifier la méthode « **index** » comme suit :

```
package spring.jpa.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import spring.jpa.model.Produit;
import spring.jpa.repository.ProduitRepository;

@Controller       // pour déclarer un contrôleur
@RequestMapping (value = "/produit")  // nom logique dans l'URL pour accéder au contrôleur
public class ProduitController {
      @Autowired       // pour l'injection de dépendances
      private ProduitRepository produitRepos;

      @RequestMapping (value = "/index")  // nom logique pour accéder à l'action "index" ou méthode "index
      public String index (Model model,
            @RequestParam (name="page") int p) // paramètre pour le numero de la page
      {
        // récupérer la page numero "p" (de taille 6)
       Page <Produit> pg = produitRepos.findAll( PageRequest.of(p, 6));
       model.addAttribute("pageProduits", pg);
    // placer la page « pg » des produits dans le "Model" comme un attribut"
            return "produits";  //retourner le nom de la vue WEB à afficher
      }
}
```

✓ Remarquer l'ajout d'un nouvel argument entier « **p** » dans la méthode « **index** ».
✓ L'argument « **p** » représente le numéro de la page à récupérer.
✓ L'argument « **p** » de la méthode est associé à un paramètre de la requête HTTP nommé « **page** ».
✓ Remarquer l'envoi d'un objet de type « **PageRequest** » comme argument dans la méthode « **findAll** » de « **ProduitRepository** » pour spécifier le numéro de la page à récupérer (comportant **6** produits au maximum).
✓ La page récupérée est stockée comme attribut nommé «**pageProduits**» dans l'objet «**Model**»

3. Dans la vue « **produits.html** » modifier l'instruction :

```
<tr th:each="p:${produits}">
```

par celle-ci :

```
<tr th:each="p:${pageProduits.content}">
```

4. Lancer l'exécution de la vue :

http://localhost:8080/produit/index

5. Remarquer la génération de l'erreur suivante (paramètre « page » non présent) :

6. Pour remédier à ce problème, spécifier une valeur par défaut « **0** » pour le paramètre « **page** » dans l'entête de la méthode « index » du contrôleur « **ProduitController** » :

```
public String index (Model model,
      // paramètre pour le numero de la page (0 par défaut çàd la première page)
          @RequestParam (name="page" , defaultValue ="0") int p)
```

7. Enregistrer la modification et relancer l'exécution.
8. Pour afficher le contenu de la deuxième page, par exemple, saisir l'url suivante :

http://localhost:8080/produit/index?page=1

B. Navigation entre les pages (barre de navigation)

9. Prendre une nouvelle version du contrôleur « **ProduitController**» pour modifier la méthode « **index** » comme suit :

```
package spring.jpa.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import spring.jpa.model.Produit;
import spring.jpa.repository.ProduitRepository;

@Controller       // pour déclarer un contrôleur
@RequestMapping (value = "/produit")  // nom logique dans l'URL pour accéder au contrôleur
public class ProduitController
{
      @Autowired       // pour l'injection de dépendances
      private ProduitRepository produitRepos;

      // nom logique pour accéder à l'action "index" ou méthode "index
      @RequestMapping (value = "/index")
      public String index (Model model,
                  // paramètre pour le numero de la page (0 par défaut)
                  @RequestParam (name="page" , defaultValue ="0") int p)
      {
            // récupérer la page numero "p" (de taille 5)
            Page <Produit> pg = produitRepos.findAll( PageRequest.of(p, 6));
      // nombre total des pages
            int nbrePages =pg.getTotalPages();
      // créer un tableau d'entier pour contenir 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 nom de la vue WEB à afficher
            return "produits";
      }
}
```

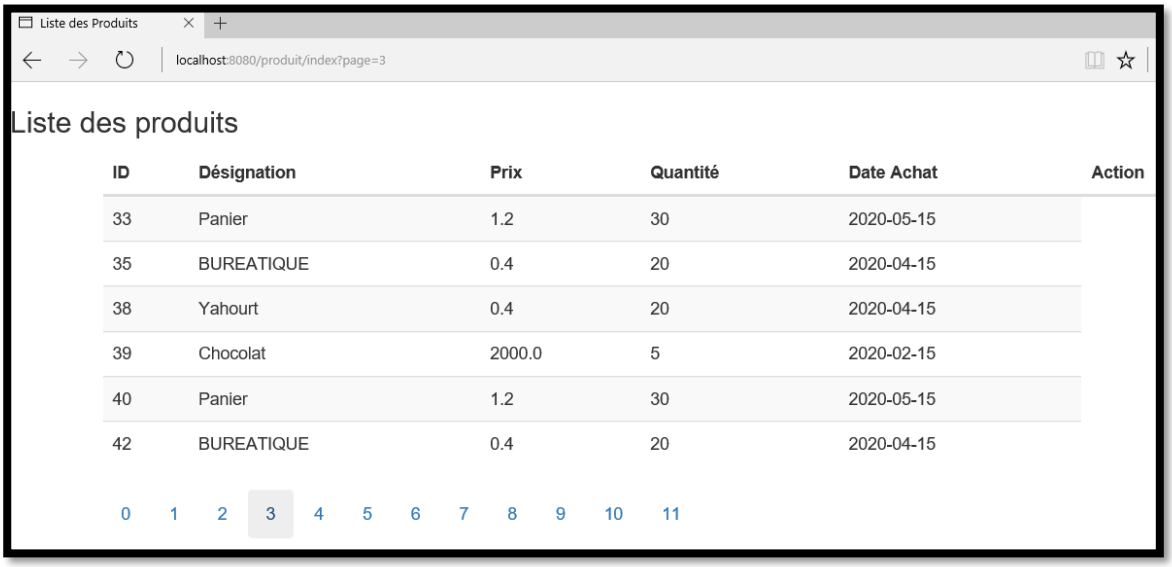
10. Prendre une nouvelle version de la vue « **produits.html**» pour ajouter un bloc « **div** » pour naviguer entre les différentes pages de produits :

```
<!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 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}" >
     <a th:text ="${p}"  th:href ="@{index(page =${p})}"></a>
    </li>
 </ul>
</div>
</body>
</html>
```

- ✓ Selon le nombre de pages (taille du tableau « **pages** » placé comme attribut dans le « **Model** », le code en gras, ci-dessus, permet d'afficher des puces sous forme de liens hypertextes vers une page spécifiée par le paramètre « **page** » passé en paramètre.
- ✓ Le symbole **@{ }** permet de se positionner sur le contexte courant (à savoir : http://localhost:8080/produit/).

11. Relancer l'exécution et remarquer l'insertion de la barre de navigation en dessous de la page web comme suit :



12. Afin de sélectionner le puce courant (sélectionner le numéro correspondant à la page affichée), modifier la balise « **li** » dans le bloc « **div** » de navigation comme suit:

```
<li  th:each ="p:${pages}" th:class="${p==pageCourante}?active:''" >
```

Il s'agit de réaliser un test sur le puce qui porte le numéro de la page courante. Si c'est le cas, alors activer le puce en question.

13. Dans la méthode « **index** » du contrôleur, placer dans le modèle un attribut nommé « **pageCourante** » qui prend comme valeur le numéro de la page courante :

```
model.addAttribute("pageCourante",p);
```

14. Relancer l'exécution et remarquer que la puce portant le numéro de la page courante est maintenant colorée dans la barre de navigation en dessous de la page web comme suit :