

Atelier 03 Spring Boot :

Spring MVC Coté Serveur :

Thymeleaf et Bootstrap

Dans cet Atelier on va apprendre, comment utiliser le moteur de template **Thymeleaf** pour développer la couche présentation (View) de notre application Web MVC.

Objectifs :

1. Ajouter la dépendance *Bootstrap*,
2. Ajouter la dépendance *Thymeleaf*,
3. Ma première page Thymeleaf,
4. Utiliser la pagination,
5. Ajouter un lien permettant de supprimer des produits,
6. Ajouter un menu Navbar.

Ajouter la dépendance Bootstrap

1. Editer le fichier pom.xml de votre application et ajouter la dépendance suivante :

```
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>bootstrap</artifactId>
  <version>4.3.1</version>
</dependency>
```

Ajouter la dépendance Thymeleaf

2. Editer le fichier pom.xml de votre application et ajouter la dépendance suivante :

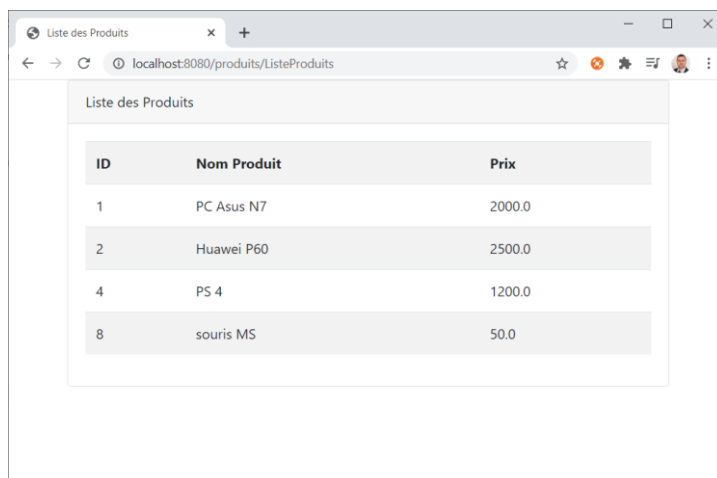
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

Ma première page Thymeleaf

3. Créer, dans le dossier src/main/resource/templates (créer le dossier templates s'il n'existe pas) le fichier listeProduits.html :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<link rel="stylesheet" type="text/css"
href="webjars/bootstrap/4.3.1/css/bootstrap.min.css" />
<head>
<meta charset="utf-8">
<title>Liste des Produits</title>
</head>
<body>
<div class="container mt-5">
<div class="card">
  <div class="card-header">
    Liste des Produits
  </div>
  <div class="card-body">

    <table class="table table-striped">
      <tr>
        <th>ID</th><th>Nom Produit</th><th>Prix</th>
      </tr>
      <tr th:each="p:${produits}">
        <td th:text="${p.idProduit}"></td>
        <td th:text="${p.nomProduit}"></td>
        <td th:text="${p.prixProduit}"></td>
      </tr>
    </table>
  </div>
</div>
</div>
</body>
</html>
```



The screenshot shows a web browser window with the title 'Liste des Produits'. The address bar shows 'localhost:8080/produits/ListeProduits'. The page content is a card with a header 'Liste des Produits' and a table with the following data:

ID	Nom Produit	Prix
1	PC Asus N7	2000.0
2	Huawei P60	2500.0
4	PS 4	1200.0
8	souris MS	50.0

Utiliser la pagination

4. Ajouter à l'interface `ProduitService` :

`Page<Produit> getAllProduitsParPage(int page, int size);`

5. Ajouter son implémentation à la classe `ProduitServiceImpl` :

```
@Override
public Page<Produit> getAllProduitsParPage(int page, int size) {

    return produitRepository.findAll(PageRequest.of(page, size));
}
```

6. Tester la méthode `getAllProduitsParPage` :

```
@Test
public void testFindByNomProduitContains()
{
    Page<Produit> prods = produitService.getAllProduitsParPage(0,2);
    System.out.println(prods.getSize());
    System.out.println(prods.getTotalElements());
    System.out.println(prods.getTotalPages());

    prods.getContent().forEach(p -> {System.out.println(p.toString());
                                   });

    /*ou bien
    for (Produit p : prods)
    {
        System.out.println(p);
    } */
}
```

7. Au niveau de la classe `ProduitController`, modifier la méthode `listeProduits` pour qu'elle prenne en charge la pagination :

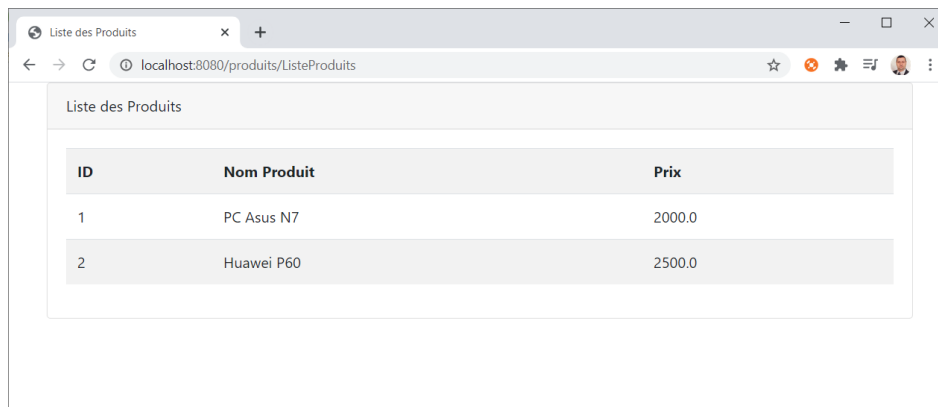
```
@RequestMapping("/ListeProduits")
public String listeProduits(ModelMap modelMap,
    @RequestParam (name="page",defaultValue = "0") int page,
    @RequestParam (name="size", defaultValue = "2") int size)
{
    Page<Produit> prods = produitService.getAllProduitsParPage(page, size);
    modelMap.addAttribute("produits", prods);
    modelMap.addAttribute("pages", new int[prods.getTotalPages()]);
    modelMap.addAttribute("currentPage", page);
    return "listeProduits";
}
```

8. Modifier la page `listeProduits.html` :

```
...
</tr>

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

9. Tester



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/produits/ListeProduits'. The page title is 'Liste des Produits'. Below the title is a table with three columns: 'ID', 'Nom Produit', and 'Prix'. The table contains two rows of data.

ID	Nom Produit	Prix
1	PC Asus N7	2000.0
2	Huawei P60	2500.0

Ajouter quelques produits à chaque démarrage de l'application

10. Modifier la classe ProduitsApplication comme suit :

```
package com.nadhem.produits;

import java.util.Date;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.nadhem.produits.entities.Produit;
import com.nadhem.produits.service.ProduitService;

@SpringBootApplication
public class ProduitsApplication implements CommandLineRunner {

    @Autowired
    ProduitService produitService;

    public static void main(String[] args) {
        SpringApplication.run(ProduitsApplication.class, args);
    }

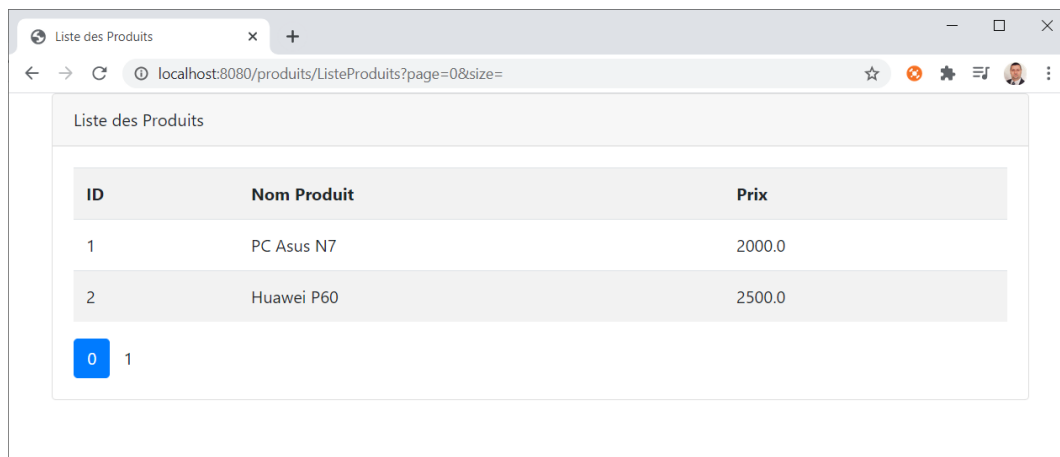
    @Override
    public void run(String... args) throws Exception {
        produitService.saveProduit(new Produit("PC Dell", 2600.0, new Date()));
        produitService.saveProduit(new Produit("PC Asus", 2800.0, new Date()));
        produitService.saveProduit(new Produit("Imprimante Epson", 900.0, new Date()));
    }
}
```

Ajouter la barre de navigation entre les pages

11. Modifier la page *listeProduits.html* comme suit :

```
...
<table class="table table-striped">
...
</table>
<ul class="nav nav-pills">
  <li th:each="page,status:${pages}">
    <a th:class="${status.index==currentPage?'btn btn-primary':'btn' }"
      th:href="@{ ListeProduits(page=${status.index}, size=${size} ) }"
      th:text="${status.index }"></a>
  </li>
</ul>
...
```

12. Tester :



Ajouter un lien permettant de supprimer des produits

13. Editer la page *listeProduits.html* :

```
...
<td th:text="${p.prixProduit}"></td>
<td><a class="btn btn-danger" th:href="@{supprimerProduit(id=${p.idProduit},
page=${currentPage},size=${size})}">Supprimer</a></td>
...

```

14. Modifier, au niveau de la classe *Produit Controller*, la méthode *supprimerProduit* :

```
@RequestMapping("/supprimerProduit")
public String supprimerProduit(@RequestParam("id") Long id,
                                ModelMap modelMap,
                                @RequestParam (name="page",defaultValue = "0") int page,
                                @RequestParam (name="size", defaultValue = "2") int size)
{
    produitService.deleteProduitById(id);
}
```

```

    Page<Produit> prods = produitService.getAllProduitsParPage(page,
size);
    modelMap.addAttribute("produits", prods);
    modelMap.addAttribute("pages", new int[prods.getTotalPages()]);
    modelMap.addAttribute("currentPage", page);
    modelMap.addAttribute("size", size);
    return "listeProduits";
}

```

15. Ajouter la confirmation de suppression

```

<a class="btn btn-danger" onclick="return confirm('Etes-vous sûr ?') " ...

```

Ajouter un menu Navbar

16. Ajouter la dépendance thymeleaf-layout-dialect nécessaire pour les templates thymeleaf :

```

<dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
</dependency>

```

17. Ajouter la dépendance JQuery nécessaire pour faire fonctionner le dropdown list de la navbar bootstarp :

```

<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>3.5.1</version>
</dependency>

```

18. Créer la page template.html :

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">

<link rel="stylesheet" type="text/css"
href="webjars/bootstrap/4.3.1/css/bootstrap.min.css" />
<script src="webjars/jquery/3.5.1/jquery.min.js"></script>
<script type="text/javascript"
src="webjars/bootstrap/4.3.1/js/bootstrap.min.js"></script>

<head>
<meta charset="utf-8">
<title>Gestion des Produits</title>
</head>
<body>
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <!-- Brand -->
    <a class="navbar-brand" href="#">Gestion Produits</a>

    <!-- Links -->
    <ul class="navbar-nav">
        <li class="nav-item">
            <a class="nav-link" th:href="@{ListeProduits}" >Home</a>

```

```

</li>

<!-- Dropdown -->
<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-toggle="dropdown">
    Produits
</a>
<div class="dropdown-menu">
    <a class="dropdown-item" th:href="@{showCreate}">Ajouter</a>
    <a class="dropdown-item" th:href="@{ListeProduits}">Lister</a>
</div>
</li>
</ul>

<ul class="navbar-nav ml-auto" >
<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-toggle="dropdown">
    [username]
</a>
<div class="dropdown-menu">
    <a class="dropdown-item" href="#">login</a>
    <a class="dropdown-item" href="#">logout</a>
    <a class="dropdown-item" href="#">Profile</a>
</div>
</li>
</ul>
</nav>
</body>
</html>

```

19. Modifier la page listeProduit.html :

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<link rel="stylesheet" type="text/css"
href="webjars/bootstrap/4.3.1/css/bootstrap.min.css" />
<head>
<meta charset="utf-8">
<title>Liste des Produits</title>
</head>
<body>
    <div th:replace="template"></div>
<div class="container">
<div class="card">
    <div class="card-header">
        Liste des Produits
    </div>
<div class="card-body">
        <table class="table table-striped">
            <tr>
                <th>ID</th><th>Nom Produit</th><th>Prix</th>
            </tr>
            <tr th:each="p:${produits.content}">
                <td th:text="${p.idProduit}"></td>
                <td th:text="${p.nomProduit}"></td>
                <td th:text="${p.prixProduit}"></td>
                <td><a class="btn btn-danger" onclick="return confirm('Etes-vous
sûr ?')">

```

```

                th:href="@{supprimerProduit(id=${p.idProduit},
page=${currentPage},size=${size})}">Supprimer</a></td>
            </tr>
        </table>
        <ul class="nav nav-pills">
            <li th:each="page,status:${pages}">
                <a th:class="${status.index==currentPage?'btn btn-primary':'btn' }"
th:href="@{ ListeProduits(page=${status.index}, size=${size} ) }"
th:text="${status.index }"></a>
            </li>
        </ul>
    </div>
</div>
</div>
</body>
</html>

```