TP Spring n°5
Spring Web épisode 2

Dans ce TP nous allons créer le contrôleur de Product. Afin d'afficher dans une page la liste de tous les produits et dans une autre page un produit en particulier.

1° - ProductController

Créez dans le package controller un nouveau contrôleur pour les produits.
Ajoutez en plus de l'annotation @Controller, l'annotation @RequestMapping("products") afin de définir les routes de ce contrôleur en commençant par /products.

1.2° - Afficher tous les produits

Ajoutez une nouvelle méthode dans votre contrôleur avec le nom getProducts de la façon suivante :

```
@Controller
@RequestMapping("/products")
public class ProductController {

	@GetMapping(value = { "", "/" })
	public String getProducts(Model model) {
		System.out.println("/products : get all products");
		return "products";
	}
}
```

Vous devez maintenant créer la page jsp pour products.

Comme nous n'avons toujours pas de base de donnée, nous devons ajouter à nos bean les produits. Pour cela dans la classe principale (là ou vous avez votre main), on peut récupérer les beans créés afin de les manipuler pour ajouter des produits, des stocks…

Et tout cela va se faire grâce aux annotations. Ajoutez dans votre fichier principal le code suivant permettant de créer un bean afin d'ajouter des produits à votre liste :

```
@Bean
public CommandLineRunner runner(ProductService productService) {
   return args -> {
      Product product1 = new Product(1l, "Produit 1", "desc1", 12d, "url1",2l);
      Product product2 = new Product(2l, "Produit 2", "desc2", 32d, "url2",4l);

      productService.save(product1);
      productService.save(product2);
   };
}
```

Maintenant votre contrôleur doit connaître ce *productService* qui contient les produits. Ajoutez dans votre contrôleur un attribut privé de type *ProductService*, utilisez l'annotation ***@Autowired*** pour le câbler au bean.

Vous pouvez maintenant récupérer dans votre méthode *getProducts* la liste des produits. Ajoutez dans cette méthode la ligne suivante :

```
model.addAttribute("products", productService.getAllProducts());
```

On obtient :

```
@GetMapping(value = { "", "/" })
public String getProducts(Model model) {
       System.out.println("/products : get all products");
       model.addAttribute("products", productService.getAllProducts());
       return "products";
}
```

On va maintenant pouvoir récupérer dans notre page jsp cet attribut *products*, que l'on vient d'ajouter au modèle.
Cet attribut étant une liste, on va avoir besoin de la parcourir afin de créer autant de *div* qu'il y a d'articles dans la liste.

Dans la page jsp on ajoute la ligne qui déclare l'utilisation de l'outil en haut dans le fichier (au dessus de la balise html) :

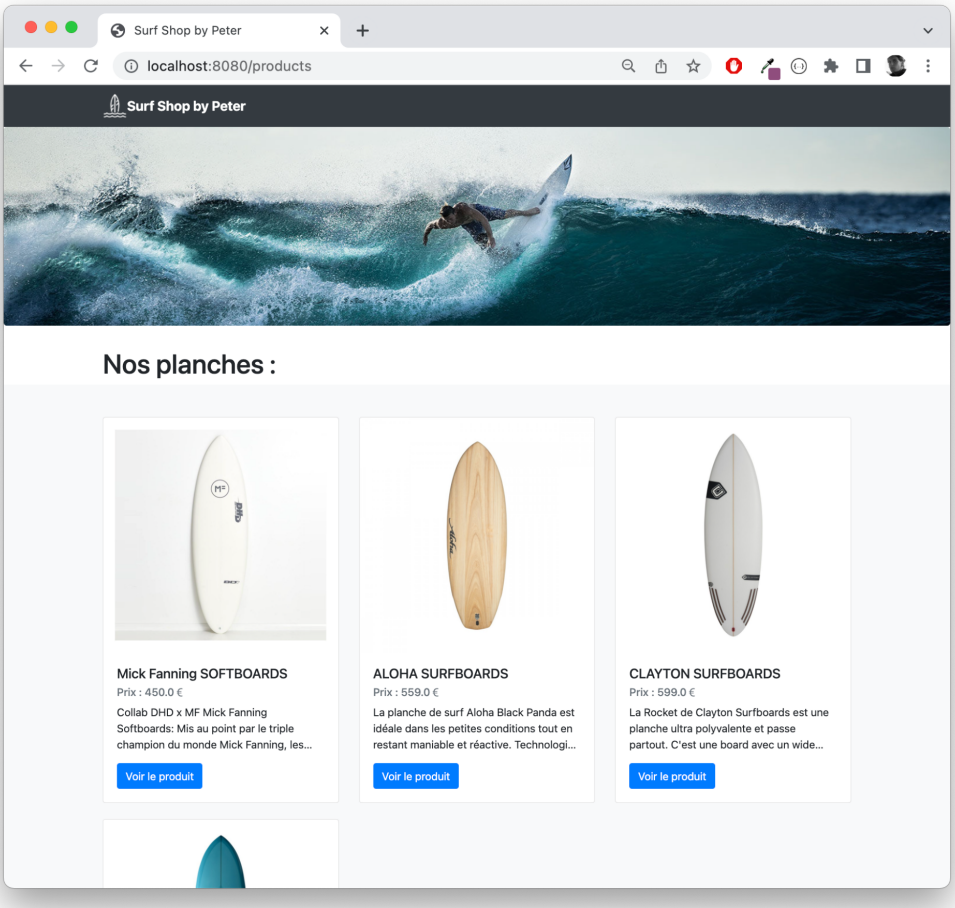
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

On va maintenant pouvoir créer une boucle afin de parcourir tous les *products* et pour créer la div contenant ce produit. Ajoutez dans votre body le code permettant de faire une boucle :

```
<c:forEach items="${products}" var="product">
	<div>
		<p>${product.name}</p>
	</div>
</c:forEach>
```

Relancer votre serveur et rafraîchissez la page sur le navigateur et vous devriez voir la page html s'afficher !

Après 2-3 modification en s'inspirant du template, on obtient un résultat comme celui-ci :



1.2° - Afficher un produit

Sur le même principe affichez la page d'un produit avec ses informations plus la quantité de stock de ce produit. La route doit être par exemple */products/21* pour afficher les informations du produit qui possède l'id 21.

Créez une nouvelle méthode dans le contrôleur avec la route suivante :

```
@GetMapping("/{productId}")
public String getProductById(Model model, @PathVariable("productId") Long id) {
	System.out.println("/products/id : get product by id (" + id + ")");
	// ...
	return "product";
}
```

Récupérez le produit, et également sa quantité de stock. A vous de faire la page jsp, elle sera lancée sur lors du clic sur le bouton *"Voir le produit"* d'un produit.

On obtient :