TD0 & TD1

Notions abordées :

- Contrôleurs
- Routage
- Vues
- Session et contexte

- Création du repository git

1. Créer un dossier **springboot-tds** ;
2. Publier **springboot-tds** sur github en tant que nouveau repository ;
3. Ajouter **jcheron** à la liste des **colaborators** de ce projet ;
4. Publier (commit and push) régulièrement sur github.

Création

Créer le projet td0 :

- Group : s4.spring
- Artifact : td0
- Packaging : War
- Description : Gestion d'items
- Dependencies : Web, Devtools, Mustache

Configurer le projet dans **application.properties**, pour que le contextPath soit **td0/**, configurer Mustache

Vues

On utilisera Semantic-UI pour la partie présentation.

Créer un template **header.html** :

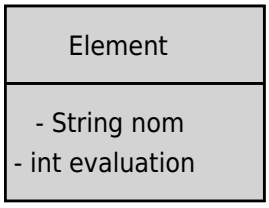
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Gestion d'items</title>
<link rel="stylesheet" type="text/css"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
```

Créer un template **footer.html** :

```
</div><!-- Fermeture de la div ui-container -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.4.1/semantic.min.js"></sc
ript>
</body>
</html>
```

Objet métier

Créer une classe Element dans s4.spring.models :



CREATED WITH YUML

Ajouter :

- Des accesseurs (getters/setters) sur les 2 membres
- Sur-définir la méthode **equals** pour que 2 items du même nom soit considérés comme égaux

Initialisation d'un objet en session

L'annotation @SessionAttributes permet de définir des variables de session. Combinée à @ModelAttribute, elle permet l'initialisation correcte de la variable de session.

```
@SessionAttributes("items")
public class MainController {

    @ModelAttribute("items")
    public List<String> getItems(){
        return new ArrayList<>();
    }
}
```

Routes

Adresse	**Description**
items/	Affiche la liste des éléments stockée en Session + 1 bouton pour ajouter un élément (/items/new)
items/new	Afficher un formulaire d'ajout d'élément (seulement son nom), la validation va vers /items/addNew puis redirige vers /items
items/inc/{nom}	Incrémente l'évaluation de l'élément de 1, puis redirige vers /items

Adresse	Description
items/dec/{nom}	Décrémente l'évaluation de l'élément de 1, puis redirige vers /items

Redirection

Effectue l'ajout, puis redirige la réponse vers la route **/items**

```
@PostMapping("items/addNew")
public RedirectView addNew(@RequestParam String nom) {
    ...
    return new RedirectView("/items/");
}
```

Ajouts/modifications

- Ajouter la route **items/delete/{index}** pour supprimer un item par son index
- Restructurer l'application :
    - Ajouter une classe **s4.spring.models.Categorie** ayant un **libelle** et pouvant comporter une liste d'**items**
    - Modifier la classe **Categorie** pour qu'elle puisse gère ses items (accès, ajout, suppression...)
    - Mettre en session la liste des catégories, initialisée par défaut avec les catégories Amis, Famille, Professionnels
    - Afficher sur la route **/** les catégories et les items contenus (on pourra utiliser le composant Tab)