JavaEE
Architecture des applications

Master 2 Informatique

Parcours type : INGÉNIERIE DU LOGICIEL ET DES DONNÉES (ILD) & Fiabilité et sécurité informatique (FSI)

Aix-Marseille Université

Gaël Guibon, Jean-Luc Massat, Omar Boucelma

2018-2019

Chapitre 1

Examen : Compléter une application d'étiquetage de l'alimentation

1.1 Résumé

Pour cet examen vous devez compléter les parties serveur de l'application JEE ainsi que la communication entre la vue et le serveur. La modification de la vue sera donc nécessaire uniquement pour l'EL et le JSTL. Le javascript (pour le graphique) est déjà pré-configuré, tout comme la configuration du projet maven, afin que vous puissiez vous concentrer sur l'essentiel : compléter l'application.
Il vous faut donc :

1. Récupérer le squelette de l'application à l'adresse suivante : https://github.com/gguibon/CoursesJavaEE/exam/
2. Implémenter les fonctionnalités demandées (voir ce PDF ou rechercher 'TODO :' dans le squelette)
3. Rendre votre projet (src+pom.xml) par mail ou clef usb. Vous pouvez également fournir le jar exécutable ("mvn package") + un fichier texte de commentaires si nécessaire.

Pour vous aider, des tests non exhaustifs sont mis en place. A vous de les utiliser, regarder ou de les désactiver. Ils sont commentés et donc ignorés par défaut.
N'oubliez pas de garder une copie de votre projet après chaque fonctionnalité réussie !

1.2 Gestion des pages

Le client souhaite naviguer dans l'application à l'aide de :

1. La page principale (index.jsp) à partir de "localhost :8090".
2. Une centralisation des actions liées aux nourritures (Food) via "localhost :8090/food" + action (exemple : localhost :8090/food/add)
3. La page de progression détaillée (progresspage.jsp) à l'adresse "localhost :8090/progress".

1.3 Gestion des données

Le client possède déjà une base de données (schema.sql + data.sql) [1]. Tout en persistant les données dans cette base il souhaite pouvoir :

1. Représenter une nourriture (*"Food"*) en fonction de cette base de données. La propriété "tag" a des valeurs contraintes.

```
// Rappel des enum de Java Standard
public enum TAG {NONE, EXCELLENT, GOOD, MEDIOCRE, BAD};
private String tag;
// accès via
NomDeLaClasse.TAG
```

1. sous échantillon filtré d'OpenFoodFacts https://fr.openfoodfacts.org/data

2. Voir la liste des nourritures dans la balise "<ul>" à l'identifiant "foodlistview" en reprenant la vue incomplète.
3. Ajouter une nourriture à l'aide du modal de la vue (bouton ayant l'identifiant "btnAddFood"), tout en récupérant la clé auto générée.
4. Supprimer une nourriture à l'aide du bouton prévu (identifiant : "buttonRemove").

1.4 Étiquetage des produits : qualité nutritionnelle (*tag*)

Le client souhaite pouvoir sélectionner une étiquette (TAG) pour chaque élément (food). L'étiquette est alors sauvegardée et l'élément est considéré comme traité (*DONE*). Un bouton indicateur permuttant entre DONE et TODO permet de changer. Pour cela il faut :

1. Mettre à jour l'élément (TAG et DONE) lors de la sélection d'une étiquette.
2. Conditionner l'affichage du statut (s'il a été traité ou non). S'il est traité le bouton est vert, sinon jaune. Aussi, si l'élément a une étiquette 'NONE', alors il ne peut pas être considéré comme traité.

```
  <!-- TODO : Aspect du bouton selon si l'element est fait ou reste a faire -->
<!-- button si l'element est traité --><button name="buttonFav" type="submit" class="btn btn-success btn-xs"
     ><i class="fas fa-check"></i> DONE</button>
<!-- bouton si l'element reste a traiter --><button name="buttonFav" type="submit" class="btn btn-warning
     btn-xs" ><i class="fas fa-exclamation"></i> TODO</button>
```

3. Voir les deux listes d'éléments, ceux traités et ceux restant à faire, à partir de la page "localhost :8090/progress" accessible dans la barre de navigation du haut.
   — Dans cette page "localhost :8090/progress", une sélection de filtre par étiquette (tag) doit être présente (élément ayant l'identifiant "tagselectform")

1.5 Graphique dynamique et liste des éléments

Le client souhaite pouvoir filtrer les éléments par leur statut à faire, fait ou les deux, via les checkbox (dans le formulaire ayant l'identifiant "formdonefilter").

1. Chaque utilisation de ce filtre par checkbox doit avoir des répercussions sur le graphique : le graphique récupère ses données via une Map<String, Integer> ayant pour clés les états possibles ("TODO", "DONE"), et pour valeurs le nombre d'éléments associés. Dans la vue, cette Map est nommée "progresscounts" :

```
// dans la vue
<input type="hidden" id="foodprogress4chart" value='${progresscounts}' />
// dans le controller (pour envoyer la Map en format JSON correct)
ObjectMapper mapper = new ObjectMapper();
mapper.writeValueAsString( monService.method1() ) ;
```

2. Chaque changement de filtre des éléments doit se répercuter dans le graphique via les données de "progresscounts".

1.6 BONUS : JPA et favoris

Sauvegardez une copie de votre projet en DAO. Décommentez des dépendances JPA dans le pom.xml.

La gestion de la TODO list est actuellement simple et un peu trop liée aux aliments. Le client souhaiterait donc avoir :

1. La même application mais utilisant du JPA avec annotations.
2. Une vraie relation entre les éléments Food et les éléments Todo et Done.