JavaEE
Architecture des applications

Master 2 Informatique

Parcours type : INGÉNIERIE DU LOGICIEL ET DES DONNÉES (ILD)

Aix-Marseille Université

Gaël Guibon, Jean-Luc Massat, Omar Boucelma

2018-2019

Chapitre 1

Examen : Compléter une application de contrôle 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
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 des favoris (favoritespages.jsp) à l'adresse "localhost :8090/food/favorites".

1.3 Gestion des données

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

1. Représenter une nourriture (*"Food"*) en fonction de cette base de données. Les propriétés "quality" et "type" ont des valeurs contraintes.

```
 // Rappel des enum de Java Standard
 public enum QUALITY {EXCELLENT, GOOD, MEDIOCRE, BAD};
private String quality;
// accès via
ClasseObjet.QUALITY.
```

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 favoris

Le client souhaite pouvoir cliquer sur l'étoile vide présente sur chaque nourriture afin d'en faire une nourriture favorite. Pour cela il faut :

1. Mettre à jour l'élément lors du clic sur l'étoile.
2. Conditionner l'affichage de l'étoile : si l'élément est une nourriture favorite alors l'étoile doit être pleine.

```
// étoile pleine
<i class="fas fa-star star" aria-hidden="true" ></i>
// étoile vide
<i class="fas fa-star star" aria-hidden="true" ></i>
```

3. Voir l'ensemble des éléments favoris à partir de la page "localhost :8090/food/favorites" accessible dans la barre de navigation.

1.5 Graphique dynamique et liste des éléments

Le client souhaite pouvoir filtrer les éléments par leur type via la liste de sélection (élément HTML ayant l'identifiant "typeselect").

1. Les choix de la liste doivent être ceux présents dans l'énumération TYPE (voir squelette de l'objet Food)
2. Chaque changement doit avoir des répercussions sur le graphique : le graphique récupère ses données via une Map<String, Integer> ayant pour clés les valeurs possibles de QUALITY, et pour valeurs le nombre d'éléments associés. Dans la vue, cette Map est nommée "qualitycounts" :

```
// dans la vue
<input type="hidden" id="foodcount4chart" value='${qualitycounts}' />
// dans le controller (pour envoyer la Map en format JSON correct)
ObjectMapper mapper = new ObjectMapper();
mapper.writeValueAsString( monService.method1() ) ;
```

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

1.6 BONUS : JPA et favoris

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

La gestion des favoris est actuellement simple. Le client souhaiterait avoir :

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