Université de Lorraine
DIPLOME: MASTER Informatique
Date : 27 février 2017

Nom:.....

Prénom:.....

GLA

1 Gestion d'une bibliothèque

La bibliothèque de l'université souhaite informatiser la gestion de ses livres en utilisant une approche basée sur la technologie JEE. L'application doit permettre l'affichage du fond bibliographique, l'enregistrement de tout nouveau livre, la soumission (au service administratif-financier) de demandes d'achat et la gestion de réponses reçues, la gestion des messages envoyés aux utilisateurs qui empruntent des livres.

Chaque livre (classe `Book`) a un titre et un certain nombre d'auteurs. On peut avoir deux livres avec le même titre mais pas deux auteurs avec le même nom. On veut connaître à tout moment l'utilisateur (classe `User`) qui emprunte le livre (s'il y en a un) et, le cas échéant, la date de début de l'emprunt. On souhaite également enregistrer le nombre d'emprunts depuis l'enregistrement du livre à la bibliographique.

Il faudra développer les beans nécessaire à la logique métier de cette application; on supposera que la partie Vue (JSF et NamedBean) de l'application est déjà développée ou sera développée par la suite. L'application utilise une basse de données accessible dans le container via la Persistence Unit appelée `ExamBibliothequePU`.

1.1 Le modèle de données

Proposer le modèle de données de l'application en précisant clairement les entités nécessaires et leurs associations. Proposer une implémentation JPA de ce modèle. Pour chaque classe proposée il faut détailler tous les attributs et toutes les annotations (non implicites) nécessaires. On supposera que les getters/setters sont générés automatiquement.

1.2 Ajouter des livres

L'application disposera d'une page permettant d'ajouter des livres au fond bibliographique de la bibliographique. Développer un EJB `BookManagerBean` implantant cette fonctionnalité. Plus précisément, il faudra que le bean implante la méthode

```
void addBook(Book book);
```

qui permettra de persister dans la base de données le livre passé en paramètre. Il faut détailler le code Java ainsi que les annotations nécessaires. Il faut également préciser et détailler les éventuels ajouts qui doivent être faits pour les classes de la question 0.1.

1.3 Emprunter un livre

Le personnel de la bibliothèque dispose d'une interface permettant d'enregistrer les prêts. Dès qu'un livre est emprunté par un abonné de la bibliothèque le nombre de prêts du livre est mis à jour dans la base de données et un mail est envoyé à l'abonné pour confirmer le prêt. Un EJB permettant l'envoi de mails est déjà disponible mais il faut savoir que la méthode permettant d'envoyer des mails peut prendre beaucoup de temps:

```
public class MailSenderBean implements MailSender {

    public void sendMail(String address, String message) {
        // l'envoi de ce mail va prendre beaucoup de temps
        ...
        System.out.println("  Mail to "+ address + " finally sent : " + message);
    }
}
```

Le bean `BookManagerBean` de la question 0.2 doit maintenant implémenter la méthode

```
void borrowBook(Long idBook, Long idUser)
```

qui permettra d'enregistrer l'emprunt du livre avec l'identifiant `idBook` par l'utilisateur avec l'identifiant `idUser` et d'envoyer un mail de confirmation à l'adresse mail de l'utilisateur.

Compléter le bean `BookManagerBean` pour implanter cette fonctionnalité. Il faut détailler le code Java ainsi que les annotations nécessaires pour les classes et méthodes concernées. Compléter **directement sur l'énoncé** `MailSenderBean` avec toutes les annotations nécessaires pour appeler sa méthode `borrowBook` de manière à éviter les attentes inutiles.

1.4 Statistiques

L'application fournit des statistiques (nombre de livres, livre le plus emprunté, etc.) qui sont utilisées dans différentes interfaces de l'application. Les statistiques sont calculées dans la méthode `updateStatistics` du bean `Statistics` et accessible avec les getters supposés disponibles pour tous les attributs du bean.

On suppose que la méthode `getAllBooks` du bean `BookManagerBean` permet de récupérer tous les livres de la bibliothèque mais elle prend potentiellement beaucoup de temps. On doit donc utiliser un mécanisme de cache pour optimiser la vitesse de réponse et donc les statistiques seront calculées au démarrage de l'application ainsi que tous les jours à minuit.

```
public class Statistics {
    private int nbBooks;
    private Book mostDemanded;


    private BookManager bookManager;



    public void updateStatistics() {
        List<Book> res = bookManager.getAllBooks();
        // compute statistics: nbBooks, mostDemanded, ...
        ...
    }
    ...
}
```

Compléter **directement sur l'énoncé** le bean `Statistics` avec les annotations nécessaire pour implanter ce comportement.

1.5 Demandes d'achats

Les demandes d'achats de nouveaux livres sont faites par le personnel de la bibliothèque et envoyées dans un topic JMS pour être traitées de manière asynchrone par l'application SAF (service administratif-financier). On suppose que le bean `BookManagerBean` de la question 0.2 implante une méthode qui permet d'envoyer une demande sous la forme d'un objet de type

```
public class Request implements Serializable {
    private String from;
    private String description;
    private String reply;
    ...
}
```

avec les attributs `from` et `description` remplis en fonction des attributs du livre passé en paramètre. On supposera que tous les getters/setters et constructeurs de la classe `Request` sont déjà implantés.

Les réponses à ces demandes sont des messages avec un contenu de type `Request` envoyées par l'application SAF dans une queue `ExamBibliothequeQueue`. Notre application utilisera un bean `ReplyHandler` pour récupérer ces messages et afficher leur contenu sous la forme:

```
Reply to request message: «description» - «reply»
```

Implanter le bean `ReplyHandler` pour réaliser la réception de réponses du SAF. Il faut détailler le code Java ainsi que les annotations nécessaires pour les classes et méthodes concernées.