Examen 2007 - 2 h

PAR : Programmation d'Applications Réparties

Troisième année

Tous documents autorisés. Rédigez les parties RMI et EJB sur des copies séparées.

★ Exercice 1. RMI (8pts).

On souhaite réaliser un service (simpliste) de calendrier partagé. Il est composé d'un serveur central sur lequel un objet `CalServeur` est invocable a distance. L'interface de cet objet est donnée plus bas. Chaque calendrier d'utilisateur (créé par un appel distant à CalServeur.createCal) est représenté par un objet de type `Cal` (placé sur le serveur).

Pour simplifier l'implémentation de la classe `Cal`, on ne représente pas des rendez-vous heure par heure, mais simplement si les gens sont libres un jour donné.

CalServeurInterface

```
1 package exam;
2 import java.rmi.Remote;
3
4 public interface CalServeurInterface extends Remote {
5   public Cal createCal(String nom) throws RemoteException;
6 }
```

CalInterface

```
1 package exam;
2 import java.rmi.Remote;
3
4 public interface CalInterface extends Remote {
5   public Boolean estLibre(int jour) throws RemoteException;
6   public Boolean libere(int jour) throws RemoteException;
7   public Boolean occupe(int jour) throws RemoteException;
8 }
```

▷ **Question 1.** Quel est le concept mis en œuvre par la méthode CalServeur.createCal ?

▷ **Question 2.** Écrivez les classes `CalServeur` et `Cal` implémentant les interfaces données. Vous etes libre d'ajouter aux classes les champs nécessaires à cela. En particulier, conseillé de doter `Cal` du champ `Boolean jourLibre[365]`

▷ **Question 3.** On souhaite maintenant ajouter un service permettant de vérifier si un groupe de personne est libre à un instant donné. Ajoutez pour cela à la classe `CalServeur` la methode suivante : `Boolean intersectionLibre(String[ ] nom, int jour)` Vous pourrez être amenés à modifiez le reste de la classe.

▷ **Question 4.** Implémentez une méthode à `CalServeur` pour poser une réunion à un groupe de personne. `void reserve(String[ ] nom, int jour)`

En cas de conflit d'emploi du temps, une exception adéquate devra être levée.

Il est rappelé que les objets RMI s'exécutent en parallèle les uns des autres, et votre solution devra tenir compte de la possibilité qu'une personne peut être disponible au début de l'exécution de `reserve()`, et être retenue par ailleurs en cours d'exécution (par exemple parce qu'un autre client a fait un appel à `occupe()` de cet objet entre temps).

▷ **Question 5.** Écrivez une classe de test cliente créant trois calendrier sur le serveur, et testant les différentes fonctionnalités.

★ **Exercice 2. EJB (12 pts).**

▷ **Question 6.** Ecrivez le code de l'implantation de l'Entity Bean correspondant à cette description

```
<entity id="ejbs.ProduitBean">
        <ejb-name>Produit</ejb-name>
        <local-home>ejbs.ProduitLocalHome</local-home>
        <local>ejbs.ProduitLocal</local>
        <ejb-class>ejbs.ProduitBean</ejb-class>
        <persistence-type>Container</persistence-type>
        <prim-key-class>java.lang.Integer</prim-key-class>
        <reentrant>true</reentrant>
        <cmp-version>2.x</cmp-version>
        <abstract-schema-name>Produit</abstract-schema-name>
        <cmp-field id="CMPFLD_Produit_id">
                <field-name>id</field-name>
        </cmp-field>
        <cmp-field id="CMPFLD_Produit_nom">
                <field-name>nom</field-name>
        </cmp-field>
        <cmp-field id="CMPFLD_Produit_prixUnitaire">
                <field-name>prixUnitaire</field-name>
        </cmp-field>
        <primkey-field>id</primkey-field>
</entity>
```

▷ **Question 7.** Ecrivez le code de la méthode getProduct du Session Bean suivant (2pts)

```
package tp;
public class GestionProduitBean implements SessionBean, GestionProduitRemoteBusiness{
  private SessionContext context;
  public void setSessionContext(SessionContext aContext) {
      context = aContext;
  }
  public void ejbActivate() {}
  public void ejbPassivate() {}
  public void ejbRemove() {}
  public void ejbCreate() {}

  // retourne la référence de l'entity bean qui a la clé key
  public ProductRemote getProduct(Integer key) {
    //TODO implement getProduct
  }

  private ProductRemoteHome lookupProductBeanRemote() {
    try {
      Context c = new InitialContext();
      Object remote = c.lookup("java:comp/env/ejb/ProductBeanRemote");
      ProductRemoteHome rv = (ProductRemoteHome) PortableRemoteObject.narrow(remote,
                                                          ProductRemoteHome.class);
      return rv;
    }
    catch(NamingException ne) {
      Logger.getLogger(getClass().getName()).log(Level.SEVERE,"exception caught",ne);
      throw new RuntimeException(ne);
    }
  }
}
```

▷ **Question 8.** Proposez un attribut transactionnel pour la méthode getProduct. Justifiez le. (1pt)

▷ **Question 9.** Un client veut appeler la méthode getProduct. Faites le diagramme de séquence de cette appel.

▷ **Question 10.** Ecrivez le code de l'appel depuis le client.

▷ **Question 11.** On veut ajouter une méthode permettant de retrouver un produit à partir de son nom. Que faut-il ajouter dans le descripteur de déploiement ?