Module SOR
Examen session 2, juillet 2015
Systèmes à objets répartis
Durée : 1h20 minutes
Tous documents autorisés

Singhoff Frank
singhoff@univ-brest.fr
C-203

NB : ce document contient 2 feuilles recto/verso.

Exercice n^o 1 : Questions de cours sur .NET (5 points)

- Question 1 : En C#, comment passe-t-on un argument par référence ?
- Question 2 : Quelles sont les différences entre une classe MarshalByRefObject et une classe [Serializable].
- Question 3 : Quelles sont les principales différences entre les objets répartis .NET et les objets répartis Java RMI ?
- Question 4 : Quelles sont les principales différences entre la technologie à objets répartis .NET et les web services .NET ?
- Question 5 : Pourquoi, par défaut, n'est-il pas possible de mémoriser au sein du serveur de web services .NET des données applicatives provenant du client ? Quelle commande doit-on utiliser afin de pouvoir stocker des données applicatives dans une classe Web service .NET grâce à une session ?

Exercice n^o 2 : Mise en oeuvre d'un service afin de gérer l'occupation d'un gymnase avec CORBA (8 points)

Dans cet exercice on souhaite implanter un serveur permettant à une mairie de gèrer l'occupation d'un gymnase. L'interface de ce serveur est donnée en annexe.

L'occupation du gymnase est décrite mois par mois. Pour chaque mois, on souhaite mémoriser une liste de séances. Le serveur doit stocker deux informations pour chaque séance : le sport concerné ainsi que l'animateur qui gère cette séance. Les services offerts par le serveur sont décrits par deux interfaces :

1. L'interface *gymnase* constitue le point d'entrée de l'application. On suppose que le serveur instancie une seule fois cette interface. Cette interface offre un service d'usine à objets CORBA : en effet, un employé de mairie qui souhaite enregistrer les séances de sport pour un mois donné, doit préalablement utiliser cette interface pour allouer un objet CORBA de type *mois*.

   L'utilisateur (exemple : un programme client) récupère la référence de l'objet *mois* instancié par la variable *ref* de la méthode *ajouter_mois*. Chaque mois a un numéro unique ; le numéro de mois est fourni par l'utilisateur grâce à l'argument *numero*. L'exception *dejaExistant* doit être levée par la méthode *ajouter_mois* lorsqu'un client demande la création d'un mois déjà enregistré.
2. L'interface *mois* permet à un client de décrire la liste des séances associées à un mois. Un objet CORBA de type *mois* est associé à chaque mois dans l'objet *gymnase*. L'ajout d'une séance est effectué par le client en invoquant la méthode *ajouter_seance*. La variable *s* de type *seance* décrit le cours à ajouter, et en particulier le sport assuré ainsi que l'animateur assurant la séance.

   L'attribut *numero* stocke le numéro de mois donné par le client lors de la création de l'objet CORBA de type *mois*. Enfin, l'attribut *les_seances* permet au client de consulter la liste des séances enregistrées pour le mois.

Question 1 :
A partir du fichier IDL en annexe, le compilateur IDL va produire plusieurs fichiers. Donnez la liste de ces fichiers ainsi que leur fonction ou utilisation.

Question 2 :
Donnez les deux classes Java qui implantent les interfaces IDL *mois* et *gymnase*.

Annexe : l'interface IDL du serveur

```
module gestion_gymnase {

    exception dejaExistant {};

    struct seance {
        string sport;
        string animateur;
    };

    typedef sequence<seance>  liste_des_seances;

    interface mois {
        readonly attribute long numero;
        readonly attribute liste_des_seances les_seances;
        void ajouter_seance(in seance s);
    };

    interface  gymnase {
        void ajouter_mois(in long numero, inout mois ref)
             raises (dejaExistant);
    };

};
```