UPEM - MASTER2 INFO - CORBA
td3 : Any / Intercepteurs

13 février 2014

1 *Type ANY*

Reprendre l'interface Forum pour qu'il puisse contenir plusieurs sortes de message (texte, audio,vidéo). Pour cela, on utilise le type générique any. Vous allez intégrer les structures AudioMessage et VideoMessage, modifier le type MessageSet et les opérations postMessage() et getMessage(). Vérifiez en codant un Client qui va poster et lire des messages de type AudioMessage et VideoMessage.

```
struct AudioMessage { string title;
string author ; string author ;
string codageAudio ;
string body;
}
struct VideoMessage { string title;
string author ; string author ;
string codageVideo ;
string body;
}
typedef sequence <any > MessageSet;
interface Forum {
......
void postMessage(in string title ,in any m) raises ( Reject );
any getMessage(in string title) raises ( Reject );
......
};
```

2 *Intercepteurs de requêtes*

Vous allez développer un intercepteur de requêtes (Intercepteur Client et Serveur). Le Client va obtenir l'identifiant (username) de l'utilisateur et l'insérer dans la requête (service contexte). L'objectif ici est de créer un fichier de LOG sur le Serveur. Pour cela on va mettre en oeuvre :

1. un intercepteur des requêtes sortantes (client) pour insérer le nom (identifiant) de l'utilisateur et le nom de la machine.

   ```
   System.getProperty("user.name");
   String computername=InetAddress.getLocalHost().getHostName();
   ```

2. un intercepteur des requêtes entrantes (serveur). La requête sera validée si le username de l'appelant est présent dans la requête (service contexte). Dans le cas contraire la requête sera rejetée (cas ou le client n'a pas utilisé d'intercepteur). L'intercepteur serveur va extraire les les informations : username, nom de l'opération invoquée et les stocker dans un ficher LOG en ajoutant la date.

Pour la mise en oeuvre des intercepteurs on suivra les étapes suivantes :

1. écriture d'un initialiseur d'intercepteurs (client et serveur). Déclaration à l'initalisation de l'ORB.
2. écriture des classes IntercepteurClient et IntercepteurServeur.

2.1 *Développelement de la classe d'Initialisation des intercepteurs*

C'est cette classe qui va être chargée et qui va permettre de créer les intercepteurs (méthode pre_init).

```
public class InterceptorClientInit extends LocalObject
implements ORBInitializer{

public class InterceptorServerInit extends LocalObject
implements ORBInitializer{
```

Cette classe doit être spécifiée grâce à un objet "Properties" qui sera transmis en paramètre de la création de l'objet orb. On développera une classe pour le client et une classe pour le serveur.

```
ORB.init(args, properties).
```

Coté Serveur :

```
Properties props = new Properties();
props.put(
"org.omg.PortableInterceptor.ORBInitializerClass.projetRobot.InterceptorServerInit",
"projetRobot.InterceptorServerInit"
);
```

Coté Client :

```
Properties props = new Properties();
props.put(
"org.omg.PortableInterceptor.ORBInitializerClass.projetRobot.InterceptorClientInit",
"projetRobot.InterceptorClientInit"
);
```

2.2 *Développelement des classes Intercepteur Client et Intercepteur Serveur*

Coté Client on développera la classe InterceptorClient et plus particulièrement la méthode send_request().

```
public class InterceptorClient extends org.omg.CORBA.LocalObject implements
ClientRequestInterceptor{

public void send_request(ClientRequestInfo ri) throws ForwardRequest {
```

Dans cette méthode on insèrera un objet de type "ServiceContext" dans l'objet "ri (request)" de type "ClientRequestInfo".

```
ServiceContext context = new ServiceContext();
context.context_data = computername.getBytes();
context.context_id = 12;
ri.add_request_service_context(context, false);
```

Coté serveur on développera la classe InterceptorServer et plus particulièrement méthode receive_request().

```
public class InterceptorServer extends org.omg.CORBA.LocalObject implements
ServerRequestInterceptor{

public void receive_request(ClientRequestInfo ri) throws ForwardRequest {
```

Dans cette méthode on va extraire les informations de l'objet "ri (request)" de type "ClientRequestInfo".

```
ServiceContext serviceContext = ri.get_request_service_context(12);
String message = new String(serviceContext.context_data);
```