TP 2 : Architecture 3-tiers

Objectifs

Mise en place d'une architecture trois tiers à l'aide des EJBs.

Compte rendu du TP

Répondre aux questions suivantes après avoir réalisé le TP :

- Discuter les avantages et inconvénients d'une architecture 3-tiers en vous appuyant sur des exemples concrets du TP. Vous expliquerez notamment les problèmes de sécurité liés à l'architecture 2-Tiers présentée dans le TP1, et les avantages sur ce point du TP2.
- Joindre au compte rendu de TP les sources Java de vos programmes.

Les comptes rendus des TP1 et TP2 sont à rendre par voie électronique à olivier.coupelon@univ-bpclermont.fr au plus tard le 17/11/2008.

Ressources

- Plateforme Java™
- Eclipse
- Glassfish V2
- Ressources disponibles http://olivier.coupelon.net/t/tpmiddleware.
- Fichiers jar : appserv-rt.jar, javaee.jar, ojdbc6.jar

Partie 1 : Premiers pas avec les EJBs

Dans cette partie du TP vous allez réaliser un premier EJB session très simple qui consiste à implémenter une méthode Hello qui prend en entrée une chaîne de caractère et renvoie en sortie une chaîne de caractères contenant un message de bienvenue. Vous développerez également un client test qui permet de tester votre EJB.
Il existe deux types d'EJB session : Les EJB Stateful qui conservent leur état entre différents appels de méthodes par un même client, et les EJB Stateless qui ne conservent aucune informations entre deux appels successifs. Nous allons créer et utiliser un EJB Stateless. Pour ce faire, deux étapes sont nécessaires : la réalisation de l'interface distante de l'EJB, qui va permettre au serveur d'application de connaître les méthodes distantes disponibles, et l'implémentation réel de l'EJB stateless.

1. Sous Eclipse, créer un projet ***HelloEJB***
2. Ajouter les deux jars suivants au classpath du projet : ***appserv-rt.jar*** et ***javaee.jar***
3. Créer l'interface de l'EJB, nommée ***Hello*** ayant comme seule méthode ***sayHello*** qui prend comme argument une chaine de caractères et renvoie comme résultat une autre chaine de caractères. Pour cela, n'oubliez pas de rajouter l'annotation @*Remote* à votre interface, qui indiquera a Glassfish que cette interface pourra être utilisée a distance.

4. Créer une classe ***HelloImpl*** permettant d'implémenter votre EJB stateless. Indiquer comme nom JNDI de votre bean : ***ejb/EJBHello***. Utilisez pour cela l'annotation ***@Stateless(name = «MonEJBHello», mappedName="ejb/HelloEJB")***. Cette annotation permet d'indiqué au serveur d'application que notre classe est un EJB Stateless, que son nom est MonEJBHello, et qu'il devra la rendre disponible par l'intermédiare du nom JNDI ejb/HelloEJB.
5. Déployer votre EJB dans le serveur d'applications Glassfish. Pour celà, exporter le projet ***HelloEJB*** dans un jar (Choisir Export>Java>Jar file) et en déployant ensuite le jar à partir de Glassfish (depuis l'interface d'administration). Vou pouvez vérifier le nom JNDI de votre EJB au niveau du serveur.
6. Créer un client java qui permet de tester votre EJB. Pour cela, il faut créer un nouveau projet sous Eclipse nommé ***ClientHelloEJB***, en ajoutant une dépandance avec le projet ***HelloEJB***.
7. Implémenter le code du client. Pour éviter de coder en dur les paramètres de connexion JNDI, vous pouvez créer un fichier de ressources ***jndi.properties*** (à l'image du fichier utilisé dans le TP1) et le mettre à la racine du projet client. Le code pour le lookup JNDI est toujours le meme, mais l'objet retourné est cette fois de type Hello.

Partie 2 : Migration de la solution du TP1 vers une architecture 3-tiers

Lors du TP1, partie 2, on vous demandais d'utiliser une Datasource pour vous connecter à votre base. Comme vous avez pu le constater, ceci pose un certain nombre de problèmes. En pratique, ce n'est pas la bonne méthode. A vous de me nous dire pourquoi dans le compte rendu.
L'objectif est donc de créer une interface sur le serveur, qui se charge d'accéder à la base, et de faire les opérations demandées par le client. Pour ce faire, vous allez implémenter un EJB qui se chargera d'accéder à la base par l'intermédiaire de la Datasource. Vous allez ensuite créer le client à cet EJB qui affichera les couples *nom de l'employé → nom du département auquel l'employé est affecté*.