TP REST

Objectif

L'objectif de ce TP est de mettre en pratique les services Web REST, afin d'exposer une API REST accessible et manipulable depuis n'importe quel client.

Prérequis

Ce TP nécessite Netbeans 8.2 Java EE incluant GlassFish Server 4.1.1.

Création de l'application Web

1. Sélectionnez File > New Project. Sous la catégorie "Java Web", choisissez "Web Application".
2. Donnez un nom à votre projet, sélectionnez le serveur sur lequel l'application sera déployée, puis cliquez sur Finish.
3. Créez un service web REST en cliquant sur New > Other > Web Service > RESTful Web Services from Pattern. Sélectionnez "Simple Root Resource". Dans la fenêtre suivante, il faut rentrer les informations permettant de configurer le service Web REST :
    a. Un nom de package (Resource Package)
    b. Un chemin d'accès qui permet d'accéder au service Web lorsqu'il est déployé (Path) et qui est à ajouter à l'URL donnée lors du déploiement de l'application Web.
    c. Un nom de class (Class Name)
    d. Un type MIME pour le format de réponse du service Web. Modifiez MIME Type en sélectionnant « text/plain ».
4. Une nouvelle classe est créée par défaut « GenericResource ». Vous allez supprimer le code par défaut et le remplacer par le code suivant :

```
import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.PathParam;
import javax.ws.rs.Consumes;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.GET;
import javax.ws.rs.Produces;

@Path("Mypath")
public class GenericResource {

    @GET
    @Produces("text/plain")
    public String getText() {
        return "Mon premier service REST";
    }
}
```

a. Il s'agit d'un service REST simple qui invoque la méthode `getText` qui retourne une réponse sous format texte "Mon premier service REST".
b. L'annotation `@Path` désigne d'une part l'URI d'accès au service et à ses sous-services, et d'autre part la présence de paramètres dans cette URI.
c. Cinq annotations sont définies, qui peuvent annoter certaines méthodes d'un service REST : `@GET`, `@POST`, `@PUT`, `@DELETE`, et `@HEAD`. Elles correspondent aux cinq méthodes HTTP qui portent le même nom. On ne peut poser chaque annotation qu'une seule fois sur une unique méthode dans une classe donnée, pour un chemin d'accès donné.

Testez votre Web Service

Une fois votre application REST créée, déployez-la sur le serveur Glassfish en faisant un clic-droit sur le projet, puis sur Run.

Une fois le projet déployé, sous la partie "RESTful Web Services" de vos projets, vous pouvez faire un clic-droit sur "GenericResource", puis "Test Resource Uri".

Travail à faire

Vous allez créer un service Web REST permettant de gérer plusieurs listes de clients. La principale différence avec le service Web précédent réside dans le fait que les états de ce nouveau Web service sont stockés sur le serveur.

1. Créez une nouvelle application Web qui sera déployée sur le serveur Glassfish que vous nommerez « RestBank ».

2. Définissez une classe Client qui sera munie de deux attributs (nom et prénom) et définissez une classe ListeClient pour gérer une structure de données de type liste. La classe ListClients va comprendre 5 méthodes : ajouterClientdansListe, modifierClient, consulterClient, consulterListeClient et supprimerClient.
3. Ajoutez la classe du service Web permettant d'implémenter les méthodes HTTP (Restful Web Services from Patterns). Vous allez définir les méthodes exposées par le service pour:
    a. Créer sa propre liste de clients
    b. Ajouter des clients dans cette liste
    c. Modifier les informations relatives à un client dans cette liste
    d. Consulter les informations relatives à un client de cette liste
    e. Obtenir les informations sur tous les clients de la liste
    f. Supprimer un client de cette liste

    N'oubliez pas de redéfinir les méthodes HTTP pour chacune des méthodes exposées par le service Web.

    g. Vérifiez que votre application est bien déployée sur le serveur Glassfish.

Vous pouvez utiliser l'application Postman ou Insomnia pour invoquer vos Web Services avec les méthodes GET, POST, PUT et DELETE.

Pour aller plus loin, connectez votre application à la base de données en utilisant JPA pour stocker les clients dans la base de données.

Bon courage !