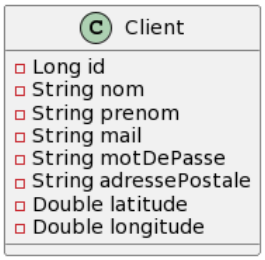
L'objectif de ce TD est de mettre en place une architecture SOA telle que vue en cours et de développer différents services métiers autour d'une entité simple. Vous utiliserez l'environnement de développement Netbeans sur le Bureau Virtuel "3IF-DASI TP Netbeans".

1. Créez un projet d'application "Java with Maven" (cf tutos techniques sous Moodle)

2. Créez une classe `Client` (classe "POJO") avec un constructeur (nom, prénom, mail et adresse postale) et des getter/setter



3. Créez une classe `Main` et faites un premier test très simple de création et d'affichage de 3 clients en redéfinissant la méthode `toString()` de la classe `Client`.
   Exemple d'affichage :

```
Client: id=null;nom=Hugo;prenom=Victor;mail=vhugo@paris.fr;motDePasse=toto;adressePostale=Paris;latitude=null;
longitude=null;
Client: id=null;nom=Yourcenar;prenom=Marguerite;mail=myourcenar@gmail.com;motDePasse=titi;adressePostale=Tooloose;
latitude=null;longitude=null;
Client: id=null;nom=Zola;prenom=Emile;mail=ezola@gmail.com;motDePasse=tutu;adressePostale=Lyon;latitude=null;
longitude=null;
```

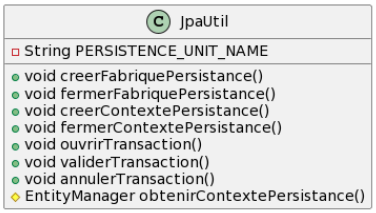
4. Créez une base de données "DASI-TD"de type Java DB (cf tutos techniques sous Moodle)

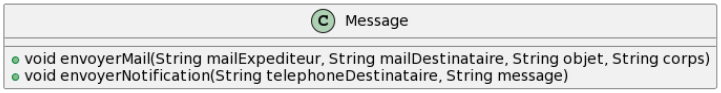
5. Vous allez maintenant transformer la classe `Client` en classe d'entité et développer la fonctionnalité permettant d'inscrire un nouveau client (voir la description du service dans l'encadré ci-dessous). Testez ce service métier sur les 3 clients et affichez à chaque fois le résultat du service métier. Observez le schéma et le contenu des tables créées dans la BD.
   Exemple d'affichage :

```
Inscription réussie pour HUGO Victor <vhugo@paris.fr>
Inscription échouée pour YOURCENAR Marguerite <vyourcenar@gmail.com>
Inscription réussie pour ZOLA Emile <ezola@gmail.com>
```

> **Signature du service** : `inscrireClient(Client client) : Boolean`
> **Description du service** : Ce service obtient les coordonnées GPS à partir de l'adresse postale du client, enregistre ce client (avec les coordonnées GPS), puis envoie un mail de confirmation au client si tout s'est bien passé. Si l'adresse postale n'est pas valide ou si une erreur est apparue lors de l'enregistrement, le service envoie un mail d'infirmation. Ce service retourne Vrai si l'inscription s'est bien passée, Faux dans le cas contraire.

💡 Utilisez la classe `JpaUtil` pour gérer le gestionnaire d'entité, la classe `Message` pour simuler l'envoi du mail et la classe `GeoNetApi` pour obtenir les coordonnées GPS (disponibles sous Moodle). Pour pourvoir utiliser la classe `GeoNetApi`, vous devrez ajouter à votre projet les dépendances Maven indiquées dans les commentaires au début du fichier `GeoNetApi.java`.





?

6. Exprimez sous forme d'annotation JPA une contrainte d'unicité du mail des clients. Testez l'inscription d'un nouveau client avec une adresse mail déjà utilisée.

7. Développez la fonctionnalité d'authentification d'un client à partir de son mail et de son mot de passe.

   Signature du service : `authentifierClient(String mail, String motDePasse) : Client`
   Description du service : Ce service identifie un client à partir de son adresse mail, puis vérifie si le mot de passe indiqué correspond au mot de passe enregistré. Ce service renvoie l'entité Client si l'authentification a réussie, ou null en cas d'échec.

8. Testez dans une méthode du `Main` un scénario complet comprenant l'inscription d'un client, suivie de l'authentification de ce client puis de l'affichage de son profil.
   Exemple d'affichage :
   ```
   Client #151: HUGO Victor <vhugo@paris.fr>, habitant à Paris [48.85298934, 2.34988063]
   ```

9. Développez la fonctionnalité de recherche de client par ID.

   Signature du service : `rechercherClientParID(Long id) : Client`
   Description du service : Ce service identifie un client à partir de son ID et renvoie l'entité Client correspondante, ou null si aucune entité ne correspond.

   Testez en affichant le résultat de la recherche (i.e. le profil du client trouvé ou "Aucun client correspondant")

10. Développez la fonctionnalité de consultation de la liste de tous les clients.

    Signature du service : `consulterListeClients() : List<Client>`
    Description du service : Ce service renvoie toutes les entités Client triées par ordre alphabétique (nom/prénom).

    Testez.
    Exemple d'affichage :
    ```
    - Client #151: HUGO Victor [48.85298934, 2.34988063]
    - Client #153: ZOLA Emile [45.812604220, 3.17972409472]
    ```

11. Modifiez le fichier persistence.xml pour passer en mode CREATE et relancez votre application. Que constatez-vous ?

12. Créez de nouvelles méthodes de tests pour l'inscription, l'authentification et la recherche de clients, avec une saisie interactive des valeurs en utilisant la classe `Saisie` disponible sous Moodle.

最后修改: 2024年03月6日 星期三 07:51

◄ IHM - Support de cours rédigé - Sylvie SERVIGNE

跳至...

Générer des diagrammes UML avec PlantUML ►

?