TP 3 : Le framework Spring

Les 2 premières parties du TP permettent de comprendre l'injection de dépendances de Spring, notamment avec les Beans. La 2ème partie permet de s'habituer la programmation par aspect avec spring AOP. La dernière partie permet de créer des services web avec Spring data.

Partie 1 : Injection de dépendances

Comme demandé dans le sujet, on a créé nos Beans toutes associées à un acteur, à savoir le client, le magasin, la banque et le fournisseur. Pour chaque acteur, on a créé une classe qui implémente les Beans. Le client implémente l'interface IRun qui va être appelé à l'exécution du code.

Partie 2 : Spring AOP

A partir des mêmes classes et interfaces de la 1ère partie, on a ajouté la classe ServiceMonitor fournie. C'est une classe annoté Aspect. On a ajouté dedans des méthodes permettant de logger les appels de nos méthodes. On a pu utiliser différentes annotations à savoir @Before, @Arond, @After et @AfterReturn. Les différentes annotations permettent de faire un système de log complet.

Partie 3 : Spring-data

Pour la dernière partie, il faut intégrer le modèle du tp1 (prise de rendez-vous avec des professionnels) dans Spring. Pour cette partie, on a du configurer le fichier application.propertities. J'ai ajouté une dépendance que mon binôme n'avait pas besoin, il s'agit de la dépendance jaxb.

```
<dependency>
  <groupId>org.glassfish.jaxb</groupId>
  <artifactId>jaxb-runtime</artifactId>
  <version>2.3.1</version>
</dependency>
```

On a ajouté nos classes métiers au format spring jpa.
Pas besoin d'intégrer les DAOs puisque l'injection de dépendance s'occupe de l'implémentation.
On a juste créé des interfaces héritant de JpaRepository.
On a fait nos webServices dans des classes taggées controller.
Comme avec jaxRS, on peut définir plusieurs services dans une seule classe.
Ici on s'est servi de thymeleaf pour faire du data-binding entre notre back-end et notre front-end.

Dans notre prototype, on ne laisse pas la possibilité à l'utilisateur de créer des jobs. Il existe un ensemble prédéfini de jobs à utiliser.

Finalement, on a des pages web permettant de lister tous les users, de créer des user et d'en supprimer. Même chose pour les appointments et les workers.
D'autres pages ont été faites pour pouvoir lister et mettre à jour le salaire d'un job.
On a créé une classe aspect qui gère le log dès qu'on accède à un service.