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 cette partie, on a du faire quelles configurations dans le application.propertities. J'ai du ajouter 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 et interfaces au squelette fournie.
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 front-end et notre back-end.

Finalement, on a des pages web permettant la création de user.
Il est possible de lister tous les users, supprimer des users et créer des workers si l'utilisateur est un administrateur. Une autre page a été faite pour pouvoir mettre à jour le salaire d'un job.