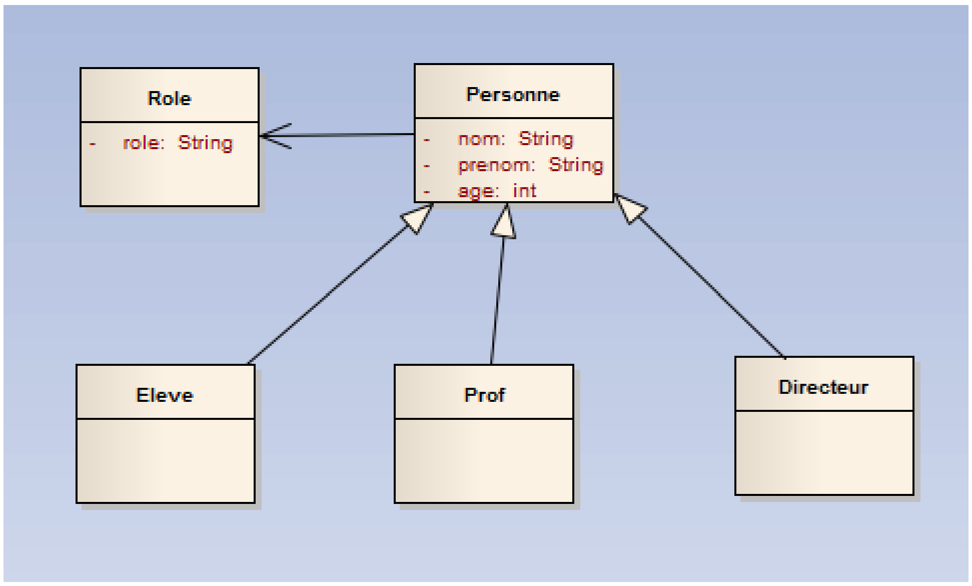
Rappel Séance 5 (week-end 16-17/02/2022)

En utilisant les association JPA, créer et mapper les liens d'héritages suivants



Objectifs de la séance 6 (week-end 22-24/02/2022)

Les objectifs de la séance d'aujourd'hui:

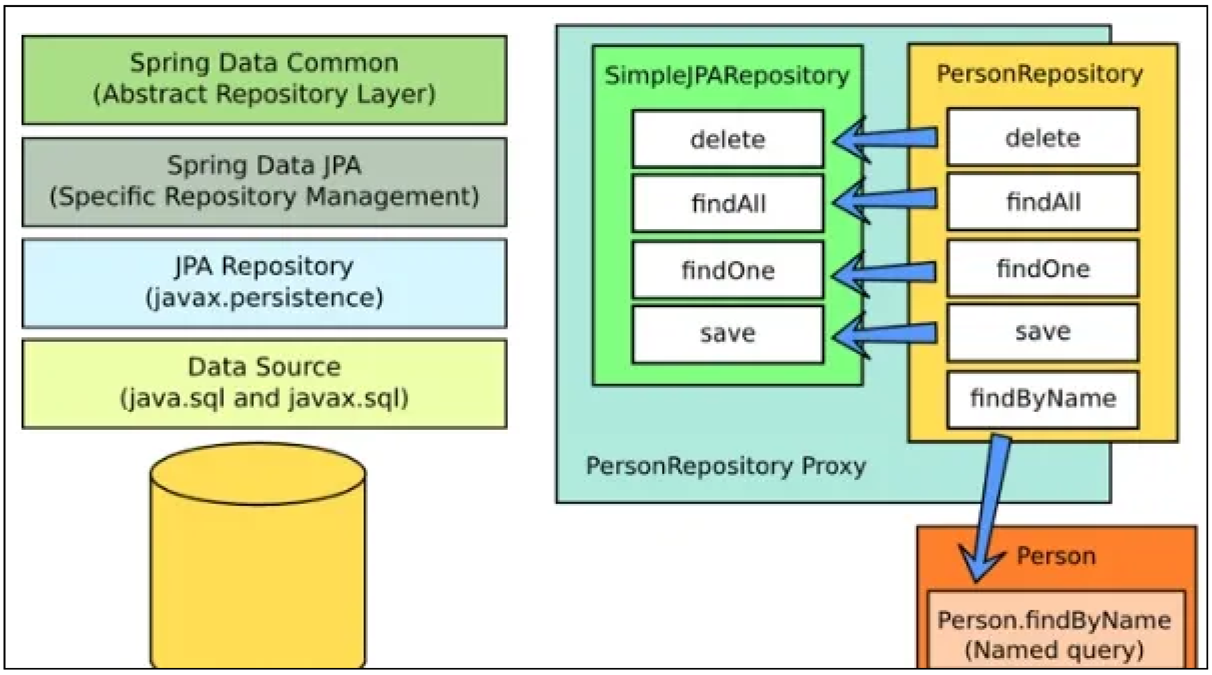
Objectif **6.1** :
Utiliser le framework **Spring Data** pour implémenter la couche DAO

Dans ce TP on suppose que :

- ☑ ~~*Vous avez réalisé totalement le TP3.*~~
- ☑ ~~*Vous avez réalisé totalement le TP4.*~~
- ☑ ~~*Vous avez réalisé totalement le TP5.*~~

Si ce n'est pas le cas : Faire d'abord le TP3, TP4 et le TP5 d'urgence

INTRODUCTION



I. UTILISATION DE SPRING DATA FRAMEWORK

1. AJOUTER LES DÉPENDANCES DU SPRING DATA DANS LE FICHIER POM.XML

```
<!--
https://mvnrepository.com/artifact/org.springframework.data/spring-data-jpa
-->
<dependency>
   <groupId>org.springframework.data</groupId>
   <artifactId>spring-data-jpa</artifactId>
   <version>2.5.0</version>
</dependency>
```

2. SUPPRIMER LES CLASSES D'IMPLÉMENTATION DE LA COUCHE DAO
3. MODIFIER LES INTERFACES DE LA COUCHE DAO EN AJOUTANT LE LIEN D'HÉRITAGE AVEC L'INTERFACE SPRING DATA "CRUDREPOSITORY"

```
package dao;

import models.Client;
import org.springframework.data.repository.CrudRepository;

@Repository
public interface IClientDao extends CrudRepository<Client,Long> {
}
```

4. MODIFIER LE FICHIER DE CRÉATION DES BEANS RESOURCES/SPRING.XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:data="http://www.springframework.org/schema/data/jpa"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:tx="http://www.springframework.org/schema/tx"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.0.xsd
   http://www.springframework.org/schema/data/jpa
   http://www.springframework.org/schema/data/jpa/spring-jpa.xsd
   http://www.springframework.org/schema/tx
   http://www.springframework.org/schema/tx/spring-tx.xsd">

   <data:repositories base-package="dao" />
   <context:component-scan base-package="service" />
   <context:component-scan base-package="presentation" />
   <tx:annotation-driven />

   <bean id="entityManagerFactory"
         class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
       <property name="persistenceUnitName" value="unit_person" />
   </bean>

   <bean id="transactionManager"
         class="org.springframework.orm.jpa.JpaTransactionManager">
       <property name="entityManagerFactory"
                 ref="entityManagerFactory" />
   </bean>
</beans>
```

5. Maintenant, puisque les méthodes de la couche Dao sont implémentées par spring data framework, Compléter la couche service en ajoutant toutes les méthodes nécessaires pour la gestion d'un client.

L'interface de la couche service

```java
package service;

import models.Client;
import java.util.List;

public interface IClientService {
    Client save(Client clt);
    Client modify(Client clt);
    void remove(long idClt);
    Client getOne(long idClt);
    List<Client> getAll();
}
```

La classe d'implémentation de la couche service

```java
package service;

import dao.IClientDao;
import models.Client;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
public class ClientServiceImpl implements IClientService {
   @Autowired
   private IClientDao dao;
   @Override
   @Transactional
   public Client save(Client clt) {
        return dao.save(clt);
   }

   @Override
   @Transactional
   public Client modify(Client newClt) {
        Client oldClt = dao.findById(newClt.getId()).get();
        oldClt.setName(newClt.getName());
        return dao.save(oldClt);
   }

   @Override
   @Transactional
   public void remove(long idClt) {
         dao.deleteById(idClt);
   }

   @Override
   public Client getOne(long idClt) {
        return dao.findById(idClt).get();
   }

   @Override
   public List<Client> getAll() {
        return (List<Client>) dao.findAll();
   }
}
```

6. Modifier le contrôleur pour appeler toutes les méthodes de la couche service.

```
package presentation;

import models.Client;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import service.IClientService;
import java.util.List;

@Controller(value = "ctrl1")
public class ClientController {
   @Autowired
   private IClientService service;

   public Client save(Client clt) {
       return service.save(clt);
   }
   public Client modify(Client clt) {
       return service.modify(clt);
   }
   public void remove(long idClt) {
        service.remove(idClt);
   }
   public Client getOne(long idClt) {
       return service.getOne(idClt);
   }
   public List<Client> getAll() {
       return service.getAll();
   }
}
```

7. Tester les méthodes du contrôleur dans la classe Application Runner

```
import models.Client;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;
import presentation.ClientController;

public class ApplicationRunner {

   public static void main(String[] args) {
       ApplicationContext ctx=new
ClassPathXmlApplicationContext("spring.xml");
        ClientController ctr= (ClientController) ctx.getBean("ctrl1");
        Client client1 = new Client("Omar");
        Client client2 = new Client("Said");
        Client client3 = new Client("Ahmed");

        // Test1 => save 3 Clients
        client1=ctr.save(client1);
        client2=ctr.save(client2);
        client3=ctr.save(client3);

        // Test2 => getAll Clients before modify and remove
        ctr.getAll().stream()
                .forEach(i-> System.out.println(i));

        // Test3 => getOne Client service
        System.out.println(ctr.getOne(1));

        // Test4 => modify Client service
        client1.setName("Hassan");
        ctr.modify(client1);

        // Test5 => remove Client service
        ctr.remove(2);
        // Test getAll Client after modify and remove
        ctr.getAll().stream()
                .forEach(i-> System.out.println(i));
    }
}
```

8. Analyser les requêtes genre par Spring Data

```
                      // Test1 => save 3 Clients
Hibernate: select tbl.next_val from hibernate_sequences tbl where tbl.sequence_name=?
for update
Hibernate: insert into hibernate_sequences (sequence_name, next_val)  values (?,?)
Hibernate: update hibernate_sequences set next_val=?  where next_val=? and
sequence_name=?
Hibernate: insert into Client (name, id) values (?, ?)
Hibernate: select tbl.next_val from hibernate_sequences tbl where tbl.sequence_name=?
for update
Hibernate: update hibernate_sequences set next_val=?  where next_val=? and
sequence_name=?
Hibernate: insert into Client (name, id) values (?, ?)
Hibernate: select tbl.next_val from hibernate_sequences tbl where tbl.sequence_name=?
for update
Hibernate: update hibernate_sequences set next_val=?  where next_val=? and
```

```
sequence_name=?
Hibernate: insert into Client (name, id) values (?, ?)
Jan 27, 2022 7:40:44 PM org.hibernate.hql.internal.QueryTranslatorFactoryInitiator
initiateService
INFO: HHH000397: Using ASTQueryTranslatorFactory
```

// Test2 => get All Clients before modify and remove

```
Hibernate: select client0_.id as id1_1_, client0_.name as name2_1_, client0_1_.status as
status1_2_, client0_2_.preferences as preferen1_3_, case when client0_1_.id is not null
then 1 when client0_2_.id is not null then 2 when client0_.id is not null then 0 end as
clazz_ from Client client0_ left outer join Normal client0_1_ on
client0_.id=client0_1_.id left outer join Vip client0_2_ on client0_.id=client0_2_.id
Hibernate: select addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_0_, addresses0_.id as
id1_0_0_, addresses0_.id as id1_0_1_, addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses0_.description as descript2_0_1_ from Address addresses0_ where
addresses0_.FK_CLIENT_ID=?
Hibernate: select addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_0_, addresses0_.id as
id1_0_0_, addresses0_.id as id1_0_1_, addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses0_.description as descript2_0_1_ from Address addresses0_ where
addresses0_.FK_CLIENT_ID=?
Hibernate: select addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_0_, addresses0_.id as
id1_0_0_, addresses0_.id as id1_0_1_, addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses0_.description as descript2_0_1_ from Address addresses0_ where
addresses0_.FK_CLIENT_ID=?
Client(id=1, name=Omar, addresses=[])
Client(id=2, name=Said, addresses=[])
Client(id=3, name=Ahmed, addresses=[])
```

// Test3 => get One Client by Id

```
Hibernate: select client0_.id as id1_1_0_, client0_.name as name2_1_0_,
client0_1_.status as status1_2_0_, client0_2_.preferences as preferen1_3_0_, case when
client0_1_.id is not null then 1 when client0_2_.id is not null then 2 when client0_.id
is not null then 0 end as clazz_0_, addresses1_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses1_.id as id1_0_1_, addresses1_.id as id1_0_2_, addresses1_.FK_CLIENT_ID as
FK_CLIEN3_0_2_, addresses1_.description as descript2_0_2_ from Client client0_ left
outer join Normal client0_1_ on client0_.id=client0_1_.id left outer join Vip client0_2_
on client0_.id=client0_2_.id left outer join Address addresses1_ on
client0_.id=addresses1_.FK_CLIENT_ID where client0_.id=?
Client(id=1, name=Omar, addresses=[])
```

// Test4 => Modify Client

```
Hibernate: select client0_.id as id1_1_0_, client0_.name as name2_1_0_,
client0_1_.status as status1_2_0_, client0_2_.preferences as preferen1_3_0_, case when
client0_1_.id is not null then 1 when client0_2_.id is not null then 2 when client0_.id
is not null then 0 end as clazz_0_, addresses1_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses1_.id as id1_0_1_, addresses1_.id as id1_0_2_, addresses1_.FK_CLIENT_ID as
FK_CLIEN3_0_2_, addresses1_.description as descript2_0_2_ from Client client0_ left
outer join Normal client0_1_ on client0_.id=client0_1_.id left outer join Vip client0_2_
on client0_.id=client0_2_.id left outer join Address addresses1_ on
client0_.id=addresses1_.FK_CLIENT_ID where client0_.id=?
Hibernate: update Client set name=? where id=?
```

// Test5 => Remove Client by Id

```
Hibernate: select client0_.id as id1_1_0_, client0_.name as name2_1_0_,
client0_1_.status as status1_2_0_, client0_2_.preferences as preferen1_3_0_, case when
client0_1_.id is not null then 1 when client0_2_.id is not null then 2 when client0_.id
```

```
is not null then 0 end as clazz_0_, addresses1_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses1_.id as id1_0_1_, addresses1_.id as id1_0_2_, addresses1_.FK_CLIENT_ID as
FK_CLIEN3_0_2_, addresses1_.description as descript2_0_2_ from Client client0_ left
outer join Normal client0_1_ on client0_.id=client0_1_.id left outer join Vip client0_2_
on client0_.id=client0_2_.id left outer join Address addresses1_ on
client0_.id=addresses1_.FK_CLIENT_ID where client0_.id=?
Hibernate: delete from Client where id=?
           // Test6 => get All Client after removing and updating
Hibernate: select client0_.id as id1_1_, client0_.name as name2_1_, client0_1_.status as
status1_2_, client0_2_.preferences as preferen1_3_, case when client0_1_.id is not null
then 1 when client0_2_.id is not null then 2 when client0_.id is not null then 0 end as
clazz_ from Client client0_ left outer join Normal client0_1_ on
client0_.id=client0_1_.id left outer join Vip client0_2_ on client0_.id=client0_2_.id
Hibernate: select addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_0_, addresses0_.id as
id1_0_0_, addresses0_.id as id1_0_1_, addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses0_.description as descript2_0_1_ from Address addresses0_ where
addresses0_.FK_CLIENT_ID=?
Hibernate: select addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_0_, addresses0_.id as
id1_0_0_, addresses0_.id as id1_0_1_, addresses0_.FK_CLIENT_ID as FK_CLIEN3_0_1_,
addresses0_.description as descript2_0_1_ from Address addresses0_ where
addresses0_.FK_CLIENT_ID=?
Client(id=1, name=Hassan, addresses=[])
Client(id=3, name=Ahmed, addresses=[])
```

Cas pratique pour l'utilisation de Spring Data

(ce cas pratique sera noté sur 5 points dans la note des TPs)

1- Ajouter les méthodes nécessaires dans les trois couches pour trouver les clients par leur nom.

Indication : dans la couche Dao, il suffit d'ajouter la signature de la méthode suivante au niveau de l'interface ClientDao

```
  List<Client> findByName(String name);
  // Name est un attribut dans la classe Client
```

```
@Repository
public interface IClientDao extends
  CrudRepository<Client,Long> {
    List<Client> findByName(String name);
 }
```

2- Par analogie, Ajouter les méthodes nécessaires pour la gestion d'une autre entité à votre choix : Facture, Commande, Adresse…

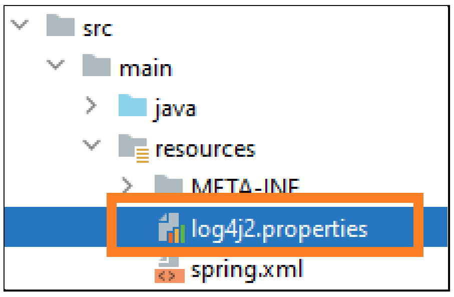
II. Adding logs to your application : SLF4J or LOG4J

Pour suivre l'exécution de votre application correctement, il est indispensable d'ajouter des logs en utilisant les dépendances nécessaires à votre pom.xml

Ajouter les dépendances suivante à votre pom.xml

```xml
<!-- Added to construct our application logs -->
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-api</artifactId>
   <version>2.7</version>
</dependency>
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-core</artifactId>
   <version>2.7</version>
</dependency>
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-slf4j-impl</artifactId>
   <version>2.7</version>
</dependency>
```

Ajouter le fichier log4j2.properties à votre src/main/resources



Le contenu de log4j2.properties est le suivant:

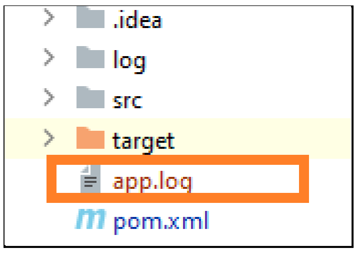
```
# Extra logging related to initialization of Log4j
# Set to debug or trace if log4j initialization is failing
status = warn
# Name of the configuration
name = ConsoleLogConfigDemo

# Console appender configuration
appender.console.type = Console
appender.console.name = consoleLogger
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = %d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L -
%m%n

# RollingFileAppender will print logs in file which can be rotated based
on time or size
appender.rolling.type = RollingFile
appender.rolling.name = fileLogger
appender.rolling.fileName= app.log
appender.rolling.filePattern= ${basePath}app_%d{yyyyMMdd}.log.gz
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = %d{yyyy-MM-dd HH:mm:ss.SSS} %level [%t]
[%c] [%M] [%l] - %msg%n
appender.rolling.policies.type = Policies
```

```
# Root logger level
rootLogger.level = debug
# Root logger referring to console appender
rootLogger.appenderRef.stdout.ref = consoleLogger
rootLogger.appenderRef.rolling.ref = fileLogger
```

Lancer ApplicationRunner et remarquer la différence dans les traces affichées dans la console

```
2022-01-06 16:05:02 DEBUG IdentifierGeneratorHelper:74 - Natively generated
identity: 1
2022-01-06 16:05:02 DEBUG ResourceRegistryStandardImpl:104 - HHH000387:
ResultSet's statement was not registered
2022-01-06 16:05:02 DEBUG TransactionImpl:62 - committing
2022-01-06 16:05:02 DEBUG AbstractFlushingEventListener:132 - Processing
flush-time cascades
2022-01-06 16:05:02 DEBUG AbstractFlushingEventListener:174 - Dirty
checking collections
2022-01-06 16:05:02 DEBUG AbstractFlushingEventListener:106 - Flushed: 0
insertions, 0 updates, 0 deletions to 1 objects
2022-01-06 16:05:02 DEBUG AbstractFlushingEventListener:113 - Flushed: 0
(re)creations, 0 updates, 0 removals to 0 collections
2022-01-06 16:05:02 DEBUG EntityPrinter:102 - Listing entities:
2022-01-06 16:05:02 DEBUG EntityPrinter:109 -
ma.cigma.pfe.models.Client{name=OMAR, id=1}
2022-01-06 16:05:02 DEBUG TransactionImpl:51 - begin
2022-01-06 16:05:02 DEBUG ActionQueue:231 - Executing identity-insert
immediately
2022-01-06 16:05:02 DEBUG SQL:92 -
    insert
    into
        TClients
        (name)
    values
        (?)
Hibernate:
    insert
    into
        TClients
        (name)
    values
        (?)
```

Actualiser votre projet et remarquer la création du fichier app.log dans la racine de l'application


.idea
log
src
target
app.log
pom.xml