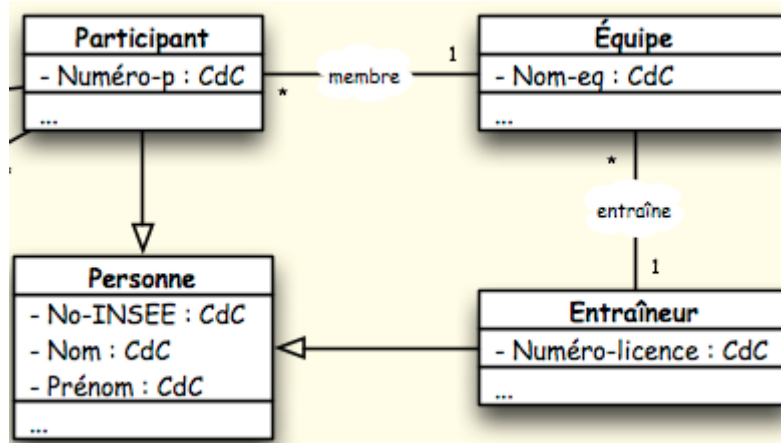


Rappel Séance 4

(01 et 03 /02/2022)

En utilisant les association JPA, créer et mapper les entités du diagram de classes suivant:



Objectifs de la séance 5

(week-end 13/01/2022)

Les objectifs de la séance d'aujourd'hui:

Objectif 5.1 : Améliorer le projet en utilisant **@Repository**, **@Service**, **@Controller**

Objectif 5.2 : Mapper une relation d'héritage le projet en utilisant **@Inheritance**, et les stratégies **SINGLE_TABLE**, **TABLE_PER_CLASS**, **JOINED**

Complément Youtube de cette séance: <https://youtu.be/bFcWhKagARk>

Dans ce TP on suppose que :

- ☒ ~~Vous avez réalisé totalement le TP 3.~~
- ☒ ~~Vous avez réalisé totalement le TP 4.~~

SI CE N'EST PAS LE CAS : FAIRE D'ABORD LE TP3 ET TP4 D'URGENCE

LE RÉSULTAT DU TP4 EST LE SUIVANT:

```

Hibernate:
    drop table if exists Client
Hibernate:
    create table Client (
        id bigint not null auto_increment,
        name varchar(255),
        primary key (id)
    )
Jan 13, 2022 5:28:14 PM org.hibernate.tool.hbm2ddl.SchemaExport execute
INFO: HHH000230: Schema export complete
creation bean dao
Hibernate:
    insert
    into
        Client
        (name)
    values
        (?)
    
```

INTRODUCTION

Le but c'est de réaliser l'inversion of control (IOC) et l'injection de dépendance (ID) par les annotations

Suivantes:

IOC → @Repository , @Service , @Controller

ID → @Autowired

I. AMÉLIORATION DU PROJET POUR UTILISER LES ANNOTATIONS SPRING IOC ET ID

1. AMÉLIORATION DU FICHIER RESOURCES/SPRING.XML :

AVANT AMELIORATION
<pre> <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "https://www.springframework.org/dtd/spring-beans-2.0.dtd"> <beans> <bean id="ctrl1" class="presentation.ClientController"> <property name="service" ref="s1"/> </bean> <bean id="s1" class="service.ClientServiceImpl"> <property name="dao" ref="d1"/> </bean> <bean id="d1" class="dao.ClientDaoImpl"></bean> </beans> </pre>
APRES AMELIORATION
<pre> <?xml version="1.0" encoding="UTF-8" ?> <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" </pre>

```
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/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">

<tx:annotation-driven />

<context:component-scan base-package="dao" />
<context:component-scan base-package="service" />
<context:component-scan base-package="presentation" />

<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>
```

2. AMÉLIORATION DES IMPLEMENTATIONS DE LA COUCHE DAO

AVANT AMELIORATION

```
package dao;

import models.Client;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

// Amélioration 1 : Cette classe doit être annotée par @Repository
public class ClientDaoImpl implements IClientDao {
```

```
//Amélioration 2 :Le code barré sera remplacé par @PersistenceContext
private EntityManagerFactory emf=
Persistence.createEntityManagerFactory("unit_person");
private EntityManager entitymanager=emf.createEntityManager();

@Override
public void save(Client p) {
// Amélioration 3 :Le code barré sera remplacé par @Transactional
// Au niveau de la classe d'implémentation Service
entitymanager.getTransaction().begin();
entitymanager.persist(p);
entitymanager.getTransaction().commit();
}
public ClientDaoImpl() {
    System.out.println("creation bean dao");
}
}
```

APRES AMELIORATION

```
package dao;

import models.Client;
import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.PersistenceContext;

@Repository
public class ClientDaoImpl implements IClientDao {

    @PersistenceContext
    private EntityManager entitymanager;
    @Override
    public void save(Client p) {
        entitymanager.persist(p);
    }
    public ClientDaoImpl() {
        System.out.println("creation bean dao");
    }
}
```

3. AMÉLIORATION DES IMPLEMENTATIONS DE LA COUCHE SERVICE

AVANT AMELIORATION

```
package service;

import dao.IClientDao;
import models.Client;

// Amélioration 1 : Cette classe doit être annotée par @Service
public class ClientServiceImpl implements IClientService {

// Amélioration 2 : l'objet (dao) doit être injecté (ID) par @Autowired
    private IClientDao dao;

// Amélioration 3 : On a plus besoin du setter de l'attribut 'dao'
    public void setDao(IClientDao dao) {
        this.dao = dao;
    }
    public ClientServiceImpl() {
        System.out.println("creation bean service");
    }
    @Override
    public void save(Client c) {
        dao.save(c);
    }
}
```

APRES AMELIORATION

```
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;

@Service
@Transactional
public class ClientServiceImpl implements IClientService {

    @Autowired
    private IClientDao dao;

    public ClientServiceImpl() {
        System.out.println("creation bean service");
    }
}
```

```
@Override
public void save(Client c) {
    dao.save(c);
}
}
```

4. AMÉLIORATION DES CONTRÔLEURS DE LA COUCHE PRESENTATION

AVANT AMELIORATION

```
package presentation;

import models.Client;
import service.IClientService;

// Amélioration 1 : Cette classe doit être annotée par @Controller ou
// @RestController. Pour le moment nous allons utiliser @Controller
public class ClientController {
    // Amélioration 2 : l'objet (service) doit être injecté par @Autowired
    private IClientService service;

    // Amélioration 3 : On a plus besoin du setter de l'attribut "service"
    public void setService(IClientService service) {
        this.service = service;
    }

    public void save(Client person) {
        service.save(person);
    }

    public ClientController() {
        System.out.println("creation bean controller");
    }
}
```

APRES AMELIORATION

```
package presentation;

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

@Controller("ctrl1")
public class ClientController {

    @Autowired
```

```
private IClientService service;

public void save(Client person) {
    service.save(person);
}

public ClientController() {
    System.out.println("creation bean controller");
}
}
```

II. MAPPING DE L'HERITAGE EN JPA

SUPPOSONS QUE LES CLIENTS SONT DE DEUX TYPE CLIENTVIP (PRÉFÉRENCES COMME ATTRIBUTS) ET CLIENTNORMAL (IMPORTANCE LEVEL COMME ATTRIBUT)

@Inheritance(strategy = InheritanceType.JOINED)

5. CRÉER CES DEUX CLASSES ET UN LIEN D'HÉRITAGE AVEC LA CLASSE CLIENT

CLIENT CLASS

```
package models;

import lombok.Data;

import javax.persistence.*;
import java.util.List;

@Entity
@Data

@Inheritance(strategy = InheritanceType.JOINED)

public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;

    public Client(String name) {
        this.name = name;
    }

    public Client() {
    }
}
```

CLIENTVIP CLASS

```
package models;

import lombok.Data;

import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;

@Entity
@Data
@PrimaryKeyJoinColumn(name="vip_id")

public class ClientVip extends Client{
    private String preferences;

    public ClientVip(String name, String preferences) {
        super(name);
        this.preferences = preferences;
    }
}
```

CLIENTNORMAL CLASS

```
package models;
import lombok.Data;

import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;

@Entity
@Data
@PrimaryKeyJoinColumn(name="normal_id")

public class ClientNormal extends Client{
    private int importanceLevel;

    public ClientNormal(String name, int importanceLevel) {
        super(name);
        this.importanceLevel = importanceLevel;
    }
}
```

NOTER L'UTILISATION DES ANNOTATIONS SUIVANTES :



@Inheritance(strategy = InheritanceType.**JOINED**) → Super Class



@PrimaryKeyJoinColumn(name="**normal_id**") → Sub Class



@PrimaryKeyJoinColumn(name="**normal_id**") → Sub Class

6. EXÉCUTER LA CLASSE APPLICATION RUNNER POUR AJOUTER UN CLIENT PUIS VÉRIFIER LES TABLES CRÉÉES POUR REPRÉSENTER LE LIEN D'HÉRITAGE (**DANS JOINED STRATÉGIE**).

TABLES:

```
Hibernate: create table Client (id bigint not null
auto_increment, name varchar(255), primary key (id))
ENGINE=InnoDB
Hibernate: create table ClientNormal (importanceLevel integer not
null, normal_id bigint not null, primary key (normal_id))
ENGINE=InnoDB
Hibernate: create table ClientVip (preferences varchar(255),
vip_id bigint not null, primary key (vip_id)) ENGINE=InnoDB
```

CONSTRAINTS:

```
Hibernate: alter table ClientNormal add constraint
FK2vxnowgo3nbe9t0yaalcynjvl foreign key (normal_id) references
Client (id)
Hibernate: alter table ClientVip add constraint
FK80cpt7le1jgof9ccm9si8xath foreign key (vip_id) references
Client (id)
```

@Inheritance(strategy = InheritanceType.SINGLE_TABLE)

7. CHANGER LA STRATÉGIE D'HÉRITAGE DANS LES TROIS CLASSES PRÉCÉDENTE COMME SUIVANT

CLIENT CLASS

```
package models;

import lombok.Data;

import javax.persistence.*;
import java.util.List;

@Entity
@Data

@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="client_type")
@DiscriminatorValue("client")

public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private long id;
private String name;

public Client(String name) {
    this.name = name;
}

public Client() {
}
}
```

CLIENTVIP CLASS

```
package models;

import lombok.Data;
import javax.persistence.Entity;

@Entity
@Data
@DiscriminatorValue("vip")

public class ClientVip extends Client{
    private String preferences;

    public ClientVip(String name, String preferences) {
        super(name);
        this.preferences = preferences;
    }
}
```

CLIENTNORMAL CLASS

```
package models;
import lombok.Data;


import javax.persistence.Entity;
@Entity
@Data
@DiscriminatorValue("normal")

public class ClientNormal extends Client{
    private int importanceLevel;

    public ClientNormal(String name, int importanceLevel) {
        super(name);
        this.importanceLevel = importanceLevel;
    }
}
```

NOTER L'UTILISATION DES ANNOTATIONS SUIVANTES :

→ **Super Class : Client**

 `@Inheritance(strategy = InheritanceType.SINGLE_TABLE)`

 `@DiscriminatorColumn(name="client_type")`

 `@DiscriminatorValue("client")`

→ **Sub Class : ClientVip and ClientNormal**

 `@DiscriminatorValue("vip")`

 `@DiscriminatorValue("normal")`

8. EXÉCUTER LA CLASSE APPLICATION RUNNER POUR AJOUTER UN CLIENT PUIS VÉRIFIER LES TABLES CRÉÉES POUR REPRÉSENTER LE LIEN D'HÉRITAGE (**DANS SINGLE_TABLE STRATÉGIE**).

TABLES:

Hibernate: create **table Client** (**client_type** varchar(31) not null, id bigint not null auto_increment, name varchar(255), **preferences** varchar(255), **importanceLevel** integer, primary key (id))
ENGINE=InnoDB

@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)

9. CHANGER LA STRATÉGIE D'HÉRITAGE DANS LES TROIS CLASSES PRÉCÉDENTE COMME SUIVANT


CLIENT CLASS

```
package models;

import lombok.Data;

import javax.persistence.*;
import java.util.List;

@Entity
@Data

@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS) 

public class Client {
    @Id
    // GenerationType.IDENTITY n'est pas permise si
    // La stratégie d'héritage est TABLE_PER_CLASS
```

```
@GeneratedValue(strategy = GenerationType.TABLE)
private long id;
private String name;

public Client(String name) {
    this.name = name;
}

public Client() {
}
}
```

CLIENTVIP CLASS

```
package models;

import lombok.Data;
import javax.persistence.Entity;

@Entity
@Data
public class ClientVip extends Client{
    private String preferences;

    public ClientVip(String name, String preferences) {
        super(name);
        this.preferences = preferences;
    }
}
```

CLIENTNORMAL CLASS

```
package models;
import lombok.Data;

import javax.persistence.Entity;

@Entity
@Data
public class ClientNormal extends Client{
    private int importanceLevel;

    public ClientNormal(String name, int importanceLevel) {
        super(name);
        this.importanceLevel = importanceLevel;
    }
}
```

NOTER L'UTILISATION DES ANNOTATIONS SUIVANTES :

→ **Super Class : Client**

Dossier des travaux pratiques. Module : JEE FRWS .Années scolaire 2021/2022. Niveau : Master FST Settat

Professeur : M. Boulchahoub Hassan hboulchahoub@gmail.com

Mise à jour 1 Feb 2022



@Inheritance(strategy = InheritanceType.**TABLE_PER_CLASS**)

→ **Sub Class : ClientVip and ClientNormal (Aucune annotation)**

10. EXÉCUTER LA CLASSE APPLICATION RUNNER POUR AJOUTER UN CLIENT PUIS VÉRIFIER LES TABLES CRÉÉES POUR REPRÉSENTER LE LIEN D'HÉRITAGE (**DANS SINGLE_TABLE STRATÉGIE**).

TABLES :

```
Hibernate: create table Client (id bigint not null, name  
varchar(255), primary key (id)) ENGINE=InnoDB  
Hibernate: create table ClientNormal (id bigint not null, name  
varchar(255), importanceLevel integer not null, primary key (id))  
ENGINE=InnoDB  
Hibernate: create table ClientVip (id bigint not null, name  
varchar(255), preferences varchar(255), primary key (id))  
ENGINE=InnoDB
```