

Rappel Séance 7

(week-end 25-26/12/2021)

Créer une annotation @Table définie par name (String) et size (long) en respectant les conditions suivantes:

- ☐ L'annotation @Table doit être applicable Seulement aux classes
- ☐ L'annotation @Table doit être disponible lors de l'exécution
- ☐ L'annotation @Table doit être héritée par les classes filles.

Objectifs de la séance 8

(week-end 08-09/01/2022)

Les objectifs de la séance d'aujourd'hui:

Objectif 8.1 : Commencer à utiliser les annotations **JPA**

Complément Youtube de cette séance

https://www.youtube.com/watch?v=KsO3Uqf_oxw&t=6115s

I. Les annotations JPA

INTRODUCTION

Ce TP consiste à implémenter la couche "DAO" du projet créé au TP6 en utilisant les annotations JPA. **L'implémentation** utilisée est celle fournie par Hibernate. Il est demandé d'utiliser les annotations de base exigées par la **spécification** JPA.

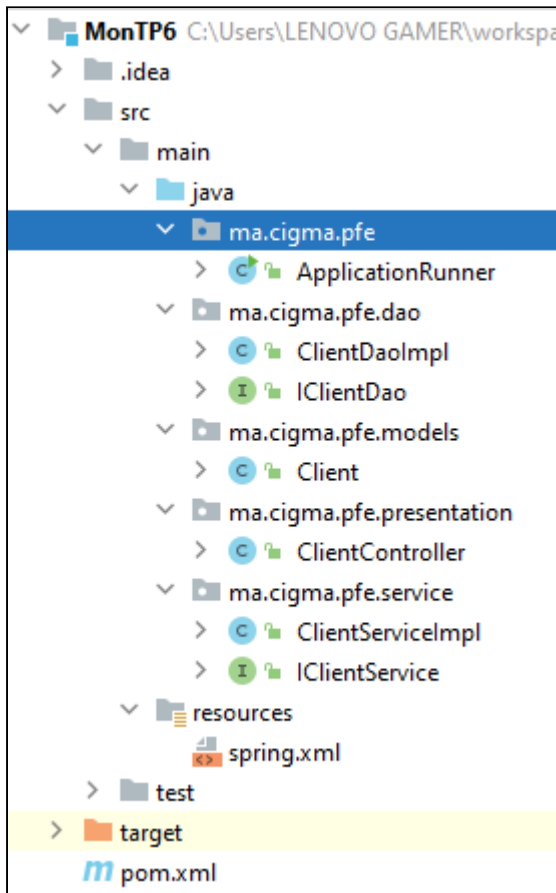
Dans ce TP on suppose que :

- ☒ ~~Vous avez créé votre projet pfe en respectant l'architecture trois couches.~~
- ☒ ~~Vous avez installé la base MySQL sur votre machine~~

SI CE N'EST PAS LE CAS : FAIRE D'ABORD LE TP6 D'URGENCE ET INSTALLER MySQL SERVER

1. LE PROJET CRÉÉ DANS LE TP6

Ci-après l'arborescence du projet du TP6



A CE STADE DE LA FORMATION, LE PROJET TP6 CONTIENDRA OBLIGATOIREMENT LES ÉLÉMENTS SUIVANTS:

- ☐ LE FICHIER POM.XML de Maven pour l'ajout des dépendances du projet
- ☐ LE FICHIER DE CONFIGURATION DU SPRING POUR L'INVERSION DU CONTRÔLE ET AUSSI L'INJECTION DE DÉPENDANCE. [dans src/main/resources/spring.xml]
- ☐ LA CLASSE MODÈLE CLIENT.JAVA [Attributs private @Getter et @Setter de Lombok]
- ☐ LA COUCHE DAO CONTIENT:
[L'INTERFACE IClientDao et la classe d'implémentation ClientDaoImpl]
- ☐ LA COUCHE SERVICE CONTIENT:
[L'interface IClientService et la classe d'implémentation ClientServiceImpl]
- ☐ LA COUCHE PRESENTATION CONTIENT:
[La classe ClientController]

2. AJOUTEZ AU POM.XML DE PROJET TP6 LES DÉPENDANCES SUIVANTES.

```
<properties>
  <hibernate.version>5.0.4.Final</hibernate.version>
  <spring.version>5.3.13</spring.version>
</properties>

<dependencies>
```

```

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>${spring.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-tx</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>${hibernate.version}</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>${hibernate.version}</version>
</dependency>
<dependency>
  <groupId>javax.transaction</groupId>
  <artifactId>jta</artifactId>
  <version>1.1</version>
</dependency>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.6</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.20</version>
  <scope>provided</scope>
</dependency>

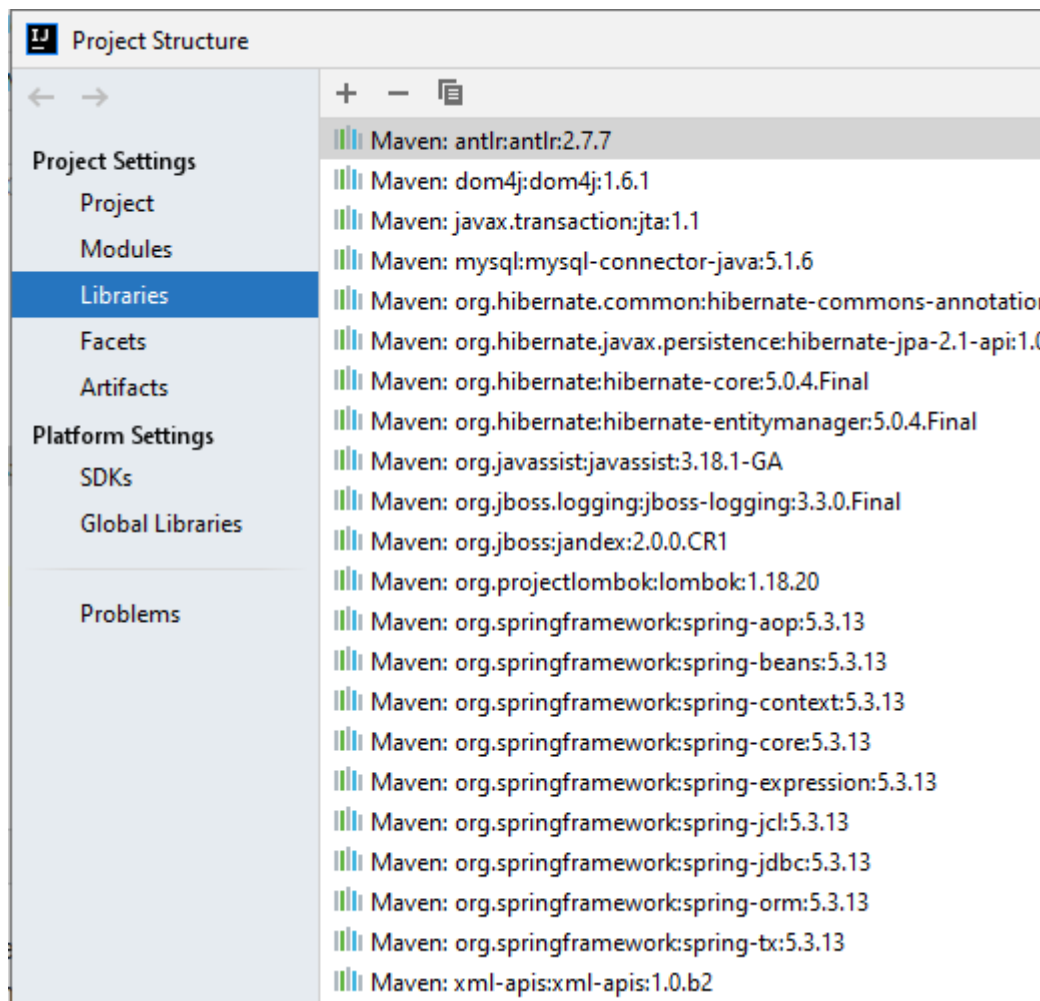
<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
  <groupId>javax.xml.bind</groupId>

```

```
<artifactId>jaxb-api</artifactId>
<version>2.3.1</version>
</dependency>

</dependencies>
```

VÉRIFIER QUE MAVEN A PROCÉDÉ À LA RÉCUPÉRATION DES JAR NÉCESSAIRES POUR VOTRE PROJET



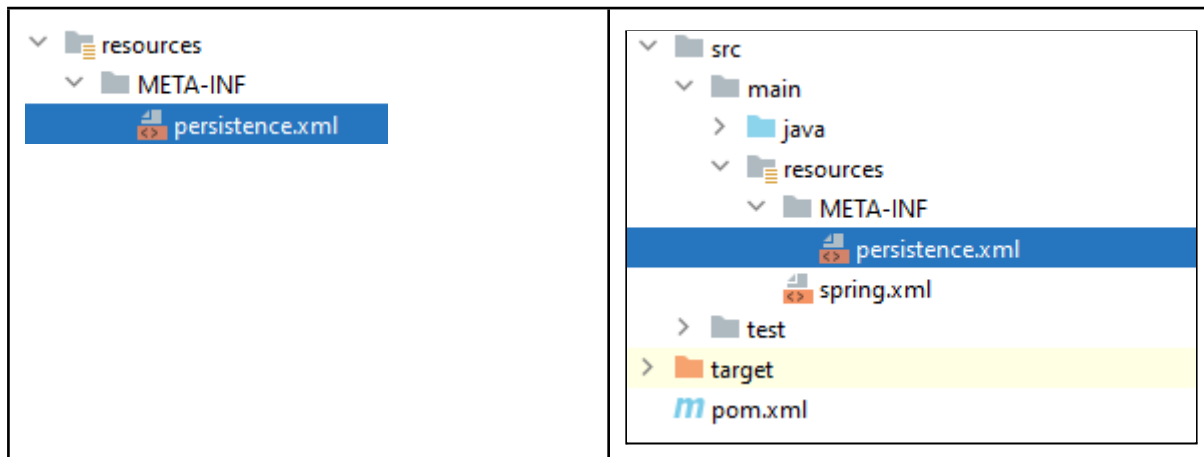
VÉRIFIER LES JARS DANS LOCAL REPOSITORY : C:\Users\VOTRE USER\.m2

PERSISTENCE.XML : PARAMETRES DU MODELE PHYSIQUE DES DONNÉES

CRÉER UN "SOURCE FOLDER" NOMMÉ "src/main/resources"

CRÉER DANS "SRC/MAIN/RESOURCES", un "folder" nommée "META-INF" en majuscule

CRÉER UN FICHIER PERSISTENCE.XML [EN MINUSCULE] DANS src/main/resources/META-INF.



METTRE DANS LE FICHIER PERSISTENCE.XML LES PARAMÈTRES DE VOTRE BASE DE DONNÉES COMME SUIVANT: [IL FAUT INSTALLER LE SERVEUR MYSQL DANS VOTRE MACHINE]

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
             version="2.0">
  <persistence-unit name="unit_clients">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/pfe_base?createDatabaseIfNotExist=true" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password" value="root" />
    </properties>
    <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver" />
    <property name="hibernate.show_sql" value="true"/>
    <property name="hibernate.format_sql" value="true" />
    <property name="hibernate.hbm2ddl.auto" value="create" />
    <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQL5Dialect" />
  </persistence-unit>
</persistence>
```

MODIFIER LA CLASSE MODÈLE : MA.CIGMA.PFE.MODELS.CLIENT

UTILISER LA ANNOTATIONS SUIVANTES DANS LA CLASSE ENTITÉ CLIENT ET SE RAPPELER DU ROLE DE CHAQUE ANNOTATION @ENTITY @TABLE @ID @GENERATEDVALUE @COLUMN @TRANSIENT

@ENTITY	DÉFINIR UNE CLASSE MODÈLE COMME ENTITÉ À GÉRER PAR L'IMPLÉMENTATION HIBERNATE, SINON VOUS AUREZ L'EXCEPTION UNKNOWN ENTITY
---------	--

@TABLE	DONNER LE NOM DE TABLE ÉQUIVALENTE À LA CLASSE ENTITÉ AU NIVEAU DE LA BASE DE DONNÉE
@Id	DÉFINIR LA COLONNE ÉQUIVALENT À LA CLÉ PRIMAIRE DANS VOTRE TABLE. IL AUSSI IMPORTANT DE CONNAÎTRE @IdCLASS ET @EMBEDDEDID QUI SERONT TRAITÉS DANS LE PROCHAIN TP
@GeneratedValue	A UTILISER SI LA CLÉ DOIT ÊTRE GÉNÉRÉE ET NON PAS AFFECTÉ DANS LES OBJETS DE LA CLASSE ENTITY
@COLUMN	À UTILISER SI LE NOM DE LA COLONNE EST DIFFÉRENT DU NOM DE L'ATTRIBUT DE LA CLASSE ENTITY.

VOUS TROUVER CI-APRÈS LA CLASSE CLIENT.JAVA APRÈS MODIFICATION

```
package ma.cigma.pfe.models;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

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

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

    @Override
    public String toString() {
        return "Client{" +
            "id=" + id +
            ", name='" + name + '\'' +
            '}';
    }
}
```

II. Save a client using JPA

INTERFACE DE LA COUCHE DAO ET SON IMPLÉMENTATION

DÉFINIR LES MÉTHODES QUE LA COUCHE DAO PEUT CONTENIR DANS L'INTERFACE
`MA.CIGMA.PFE.DAO.IClientDao`

DANS UN PREMIER TEMPS UNE SEULE MÉTHODE `SAVE(CLIENT C)` SERA EXPOSÉE POUR LA PARTIE SERVICE. CI-APRÈS L'INTERFACE EN QUESTION:

```
package ma.cigma.pfe.dao;
import ma.cigma.pfe.models.Client;
public interface IClientDao {
    Client save(Client c);
}
```

IMPLÉMENTER LA MÉTHODE : `SAVE(CLIENT B)` DANS UNE CLASSE
`MA.CIGMA.PFE.DAO.CLIENTDAOIMPL`

UTILISER UN OBJET DE TYPE `ENTITYMANAGERFACTORY` ET PAR LA SUITE UN OBJET
`ENTITYMANAGER`.

```
EntityManagerFactory emf=
Persistence.createEntityManagerFactory("unit_clients");
EntityManager em=emf.createEntityManager();
```

UTILISER UNE TRANSACTION DANS LA MÉTHODE `SAVE` COMME SUIVANT:

```
@Override
public Client save(Client c) {
    em.getTransaction().begin();
    em.persist(c);
    em.getTransaction().commit();
    return null;
}
```

LA CLASSE `MA.CIGMA.PFE.DAO.CLIENTDAOIMPL` DEVIENT:

```
package ma.cigma.pfe.dao;

import ma.cigma.pfe.models.Client;

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

public class ClientDaoImpl implements IClientDao{

    EntityManagerFactory emf=
```

```
Persistence.createEntityManagerFactory("unit_clients");
EntityManager em=emf.createEntityManager();

public ClientDaoImpl() {
}
@Override
public Client save(Client c) {
    em.getTransaction().begin();
    em.persist(c);
    em.getTransaction().commit();
    return null;
}
}
```

MAINTENANT MODIFIER LA CLASSE `ma.cigma.pfe.ApplicationRunner`

```
package ma.cigma.pfe;

import ma.cigma.pfe.models.Client;
import ma.cigma.pfe.presentation.ClientController;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class ApplicationRunner {

    public static void main(String[] args) {
        ApplicationContext context= new
        ClassPathXmlApplicationContext("spring.xml");
        ClientController ctrl = (ClientController)
        context.getBean("idCtrl");
        Client clt = new Client("OMAR");
        ctrl.save(clt);
    }
}
```

EXÉCUTER LA CLASSE `ma.cigma.pfe.ApplicationRunner` ET VÉRIFIER LA CONSOLE


```
Run: ApplicationRunner
INFO: HHH10001001: Connection properties: {user=root, password=****}
Dec 24, 2021 10:10:25 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Dec 24, 2021 10:10:25 AM org.hibernate.engine.jdbc.connections.internal.PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
Dec 24, 2021 10:10:25 AM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
Dec 24, 2021 10:10:25 AM org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl useContextualLobCreation
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3] less than 4
Dec 24, 2021 10:10:26 AM org.hibernate.tool.hbm2ddl.SchemaExport execute
INFO: HHH000227: Running hbm2ddl schema export
Hibernate:
    drop table if exists Tclients
Hibernate:
    create table Tclients (
        id bigint not null auto_increment,
        name varchar(255),
        primary key (id)
    )
Dec 24, 2021 10:10:26 AM org.hibernate.tool.hbm2ddl.SchemaExport execute
INFO: HHH000230: Schema export complete
RG Service Layer Level ...
Hibernate:
    insert
    into
        Tclients
        (name)
    values
        (?)
```

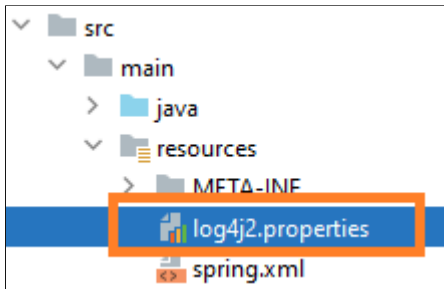
III. Add logs to your application : SLF4J or LOG4J

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

```
<!-- 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

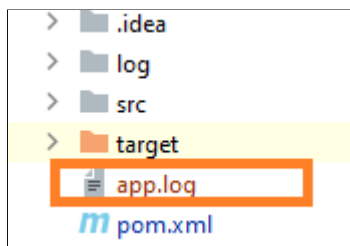
# Root logger level
rootLogger.level = debug
# Root logger referring to console appender
rootLogger.appenderRef.stdout.ref = consoleLogger
rootLogger.appenderRef.rolling.ref = fileLogger
```

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



IV. Update a client using JPA

Maintenant nous allons ajouter le cas d'utilisation suivant à notre application:

→ *Modifier un client*

1. MODIFICATION DE L'INTERFACE DAO → ICLIENTDAO

Apporter la modification suivante à l'interface IClientDao

```
package ma.cigma.pfe.dao;

import ma.cigma.pfe.models.Client;

public interface IClientDao {
    Client save(Client c);
    Client update(Client c);
}
```

2. MODIFICATION DE L'IMPLEMENTATION DAO → CLIENTDAOIMPL

Apporter la modification suivante à la classe ClientDaoImpl

```
package ma.cigma.pfe.service;

import ma.cigma.pfe.dao.IClientDao;
import ma.cigma.pfe.models.Client;

public class ClientServiceImpl implements IClientService {

    private IClientDao dao;

    public void setDao(IClientDao dao) {
        this.dao = dao;
    }

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

    @Override
    public Client modify(Client c) {
        return dao.update(c);
    }
}
```

3. MODIFICATION DE L'INTERFACE SERVICE → ICLIENTSERVICE

Apporter la modification suivante à l'interface service → IClientService

```
package ma.cigma.pfe.service;

import ma.cigma.pfe.models.Client;

public interface IClientService {
    Client save(Client c) ;
    Client modify(Client c);
}
```

4. MODIFICATION DE L'IMPLEMENTATION SERVICE → CLIENTSERVICEIMPL

Apporter la modification suivante à la classe d'implémentation service → ClientServiceImpl

```
package ma.cigma.pfe.service;
```

```
import ma.cigma.pfe.dao.IClientDao;
import ma.cigma.pfe.models.Client;

public class ClientServiceImpl implements IClientService {

    private IClientDao dao;

    public void setDao(IClientDao dao) {
        this.dao = dao;
    }

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

    @Override
    public Client modify(Client c) {
        return dao.update(c);
    }
}
```

5. MODIFICATION DU CONTROLLER PRESENTATION → CLIENTCONTROLLER

Apporter la modification suivante au controller presentation → ClientController

```
package ma.cigma.pfe.presentation;

import ma.cigma.pfe.models.Client;
import ma.cigma.pfe.service.IClientService;

public class ClientController {

    private IClientService service ;

    public void setService(IClientService service) {
        this.service = service;
    }

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

    public void modify(Client c){
        service.modify(c);
    }
}
```

6. MODIFICATION DU APPLICATION RUNNER PFE→APPLICATIONRUNNER

```
package ma.cigma.pfe;

import ma.cigma.pfe.models.Client;
import ma.cigma.pfe.presentation.ClientController;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class ApplicationRunner {

    public static void main(String[] args) {
        ApplicationContext context =new
        ClassPathXmlApplicationContext("spring.xml");

        ClientController ctrl = (ClientController)
        context.getBean("idCtrl");
        // Test save use case for three clients
        ctrl.save(new Client("OMAR"));
        ctrl.save(new Client("SIHAM"));
        ctrl.save(new Client("AHMED"));
        ctrl.save(new Client("FARAH"));

        // Test modify use case for client with id==1
        ctrl.modify(new Client(1,"new Name"));

    }
}
```

7. EXECUTER APPLICATION RUNNER PFE→APPLICATIONRUNNER

Exécuter Application Runner et vérifier les logs au niveau de la console

```
2022-01-06 17:55:40 DEBUG TransactionImpl:51 - begin
2022-01-06 17:55:40 DEBUG TransactionImpl:62 - committing
2022-01-06 17:55:40 DEBUG AbstractFlushingEventListener:132 - Processing flush-time
cascades
2022-01-06 17:55:40 DEBUG AbstractFlushingEventListener:174 - Dirty checking collections
2022-01-06 17:55:40 DEBUG AbstractFlushingEventListener:106 - Flushed: 0 insertions, 1
updates, 0 deletions to 4 objects
2022-01-06 17:55:40 DEBUG AbstractFlushingEventListener:113 - Flushed: 0 (re)creations,
0 updates, 0 removals to 0 collections
2022-01-06 17:55:40 DEBUG EntityPrinter:102 - Listing entities:
2022-01-06 17:55:40 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=AHMED,
id=3}
2022-01-06 17:55:40 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=FARAH,
id=4}
2022-01-06 17:55:40 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=new Name,
id=1}
2022-01-06 17:55:40 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=SIHAM,
id=2}
2022-01-06 17:55:40 DEBUG SQL:92 -
```

```
update
  TClients
set
  name=?
where
  id=?
Hibernate:
update
  TClients
set
  name=?
where
  id=?
```

V. Remove a client by id using JPA

Maintenant nous allons ajouter le cas d'utilisation suivant à notre application:

→ *Supprimer un client par son id*

1. MODIFICATION DE L'INTERFACE DAO → ICLIENTDAO

Apporter la modification suivante à l'interface IClientDao

```
package ma.cigma.pfe.dao;

import ma.cigma.pfe.models.Client;

public interface IClientDao {
    Client save(Client c);
    Client update(Client c);
    void deleteById(long idClient);
}
```

2. MODIFICATION DE L'IMPLEMENTATION DAO → CLIENTDAOIMPL

Apporter la modification suivante à la classe ClientDaoImpl

```
package ma.cigma.pfe.dao;

import ma.cigma.pfe.models.Client;

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

public class ClientDaoImpl implements IClientDao{

    EntityManagerFactory emf=
    Persistence.createEntityManagerFactory("unit_clients");
    EntityManager em=emf.createEntityManager();
```

```

@Override
public Client save(Client c) {
    em.getTransaction().begin();
    em.persist(c);
    em.getTransaction().commit();
    return null;
}

@Override
public Client update(Client newClient) {
    em.getTransaction().begin();
    Client currentClient =
em.find(Client.class, newClient.getId());
    currentClient.setName(newClient.getName());
    em.persist(currentClient);
    em.getTransaction().commit();
    return null;
}

@Override
public void deleteById(long idClient) {
    em.getTransaction().begin();
    Client clientInDataBase = em.find(Client.class, idClient);
    em.remove(clientInDataBase);
    em.getTransaction().commit();
}
}

```

3. MODIFICATION DE L'INTERFACE SERVICE → ICLIENTSERVICE

Apporter la modification suivante à l'interface service → IClientService

```

package ma.cigma.pfe.service;

import ma.cigma.pfe.models.Client;

public interface IClientService {
    Client save(Client c) ;
    Client modify(Client c);
    void removeById(long id);
}

```

4. MODIFICATION DE L'IMPLEMENTATION SERVICE → CLIENTSERVICEIMPL

Apporter la modification suivante à la classe d'implémentation service → ClientServiceImpl

```

package ma.cigma.pfe.service;

import ma.cigma.pfe.dao.IClientDao;

```



```
import ma.cigma.pfe.models.Client;

public class ClientServiceImpl implements IClientService {

    private IClientDao dao;

    public void setDao(IClientDao dao) {
        this.dao = dao;
    }

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

    @Override
    public Client modify(Client c) {
        return dao.update(c);
    }

    @Override
    public void removeById(long id) {
        dao.deleteById(id);
    }
}
```

5. MODIFICATION DU CONTROLLER PRESENTATION → CLIENTCONTROLLER

Apporter la modification suivante au controller presentation → ClientController

```
package ma.cigma.pfe.presentation;

import ma.cigma.pfe.models.Client;
import ma.cigma.pfe.service.IClientService;

public class ClientController {

    private IClientService service ;

    public void setService(IClientService service) {
        this.service = service;
    }

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

    public void modify(Client c){
        service.modify(c);
    }

    public void removeById(long id){
        service.removeById(id);
    }
}
```

```
}
```

6. MODIFICATION DU APPLICATION RUNNER PFE → APPLICATIONRUNNER

```
package ma.cigma.pfe;

import ma.cigma.pfe.models.Client;
import ma.cigma.pfe.presentation.ClientController;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class ApplicationRunner {

    public static void main(String[] args) {
        ApplicationContext context =new
        ClassPathXmlApplicationContext("spring.xml");

        ClientController ctrl = (ClientController)
        context.getBean("idCtrl");
        // Test save use case for three clients
        ctrl.save(new Client("OMAR"));
        ctrl.save(new Client("SIHAM"));
        ctrl.save(new Client("AHMED"));
        ctrl.save(new Client("FARAH"));

        // Test modify use case for client with id==1
        ctrl.modify(new Client(1L,"new Name"));

        // Test remove use case for client with id==1
        ctrl.removeById(1L);
    }
}
```

7. EXECUTER APPLICATION RUNNER PFE → APPLICATIONRUNNER

Exécuter Application Runner et vérifier les logs au niveau de la console

```
2022-01-06 18:20:15 DEBUG EntityPrinter:102 - Listing entities:
2022-01-06 18:20:15 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=AHMED,
id=3}
2022-01-06 18:20:15 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=FARAH,
id=4}
2022-01-06 18:20:15 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=new Name,
id=1}
2022-01-06 18:20:15 DEBUG EntityPrinter:109 - ma.cigma.pfe.models.Client{name=SIHAM,
id=2}
2022-01-06 18:20:15 DEBUG SQL:92 -
delete
from
TClients
```

```

where
    id=?
Hibernate:
delete
from
    TClients
where
    id=?

```

VI. Select one client by id using JPA

Maintenant nous allons ajouter le cas d'utilisation suivant à notre application:

→ *Chercher un client par son id*

1. MODIFICATION DE L'INTERFACE DAO → ICLIENTDAO

Apporter la modification suivante à l'interface IClientDao

```

package ma.cigma.pfe.dao;

import ma.cigma.pfe.models.Client;

public interface IClientDao {
    Client save(Client c);
    Client update(Client c);
    void deleteById(long idClient);
    Client findById(long idClient);
}

```

2. MODIFICATION DE L'IMPLEMENTATION DAO → CLIENTDAOIMPL

Apporter la modification suivante à la classe ClientDaoImpl

```

package ma.cigma.pfe.dao;

import ma.cigma.pfe.models.Client;

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

public class ClientDaoImpl implements IClientDao{

    EntityManagerFactory emf=
    Persistence.createEntityManagerFactory("unit_clients");
    EntityManager em=emf.createEntityManager();

    @Override
    public Client save(Client c) {
        em.getTransaction().begin();
        em.persist(c);
        em.getTransaction().commit();
        return null;
    }
}

```

```

@Override
public Client update(Client newClient) {
    em.getTransaction().begin();
    Client currentClient =
em.find(Client.class, newClient.getId());
    currentClient.setName(newClient.getName());
    em.persist(currentClient);
    em.getTransaction().commit();
    return null;
}

@Override
public void deleteById(long idClient) {
    em.getTransaction().begin();
    Client clientInDataBase = em.find(Client.class, idClient);
    em.remove(clientInDataBase);
    em.getTransaction().commit();
}

@Override
public Client findById(long idClient) {
    return em.find(Client.class, idClient);
}
}

```

3. MODIFICATION DE L'INTERFACE SERVICE → ICLIENTSERVICE

Apporter la modification suivante à l'interface service → IClientService

```

package ma.cigma.pfe.service;

import ma.cigma.pfe.models.Client;

public interface IClientService {
    Client save(Client c) ;
    Client modify(Client c);
    void removeById(long id);
    Client getById(long id);
}

```

4. MODIFICATION DE L'IMPLEMENTATION SERVICE → CLIENTSERVICEIMPL

Apporter la modification suivante à la classe d'implémentation service → ClientServiceImpl

```

package ma.cigma.pfe.service;

import ma.cigma.pfe.dao.IClientDao;
import ma.cigma.pfe.models.Client;

public class ClientServiceImpl implements IClientService {

```

```
private IClientDao dao;

public void setDao(IClientDao dao) {
    this.dao = dao;
}

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

@Override
public Client modify(Client c) {
    return dao.update(c);
}

@Override
public void removeById(long id) {
    dao.deleteById(id);
}

@Override
public Client getById(long id) {
    return dao.findById(id);
}
}
```

5. MODIFICATION DU CONTROLLER PRESENTATION → CLIENTCONTROLLER

Apporter la modification suivante au controller presentation → ClientController

```
package ma.cigma.pfe.presentation;

import ma.cigma.pfe.models.Client;
import ma.cigma.pfe.service.IClientService;

public class ClientController {

    private IClientService service ;

    public void setService(IClientService service) {
        this.service = service;
    }

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

    public void modify(Client c){
        service.modify(c);
    }
}
```

```
public void removeById(long id){
    service.removeById(id);
}

public Client getById(long id){
    return service.getById(id);
}
}
```

6. MODIFICATION DU APPLICATION RUNNER PFE → APPLICATIONRUNNER

```
package ma.cigma.pfe;

import ma.cigma.pfe.models.Client;
import ma.cigma.pfe.presentation.ClientController;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class ApplicationRunner {

    public static void main(String[] args) {
        ApplicationContext context =new
        ClassPathXmlApplicationContext("spring.xml");

        ClientController ctrl = (ClientController)
        context.getBean("idCtrl");
        // Test save use case for three clients
        ctrl.save(new Client("OMAR"));
        ctrl.save(new Client("SIHAM"));
        ctrl.save(new Client("AHMED"));
        ctrl.save(new Client("FARAH"));

        // Test modify use case for client with id==1
        ctrl.modify(new Client(1L,"new Name"));

        // Test remove use case for client with id==1
        //ctrl.removeById(1L);

        // Test find use case for client with id==1
        Client found = ctrl.getById(1L);

    }
}
```

7. EXECUTER APPLICATION RUNNER PFE → APPLICATIONRUNNER

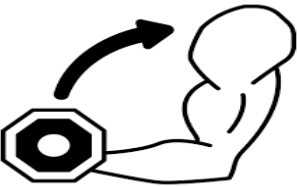

Modifier la stratégie de génération de la base au niveau du fichier META-INF/persistence.xml

```
<property name="hibernate.hbm2ddl.auto" value="update" />
```

Exécuter Application Runner et vérifier les logs au niveau de la console

```
Hibernate:
select
  client0_.id as id1_0_0_,
  client0_.name as name2_0_0_
from
  TClients client0_
where
  client0_.id=?
2022-01-06 18:47:40 DEBUG ResultSetProcessorImpl:110 - Starting ResultSet row #0
2022-01-06 18:47:40 DEBUG EntityReferenceInitializerImpl:126 - On call to
EntityIdentifierReaderImpl#resolve, EntityKey was already known; should only happen on
root returns with an optional identifier specified
2022-01-06 18:47:40 DEBUG TwoPhaseLoad:144 - Resolving associations for
[ma.cigma.pfe.models.Client#2]
2022-01-06 18:47:40 DEBUG TwoPhaseLoad:274 - Done materializing entity
[ma.cigma.pfe.models.Client#2]
2022-01-06 18:47:40 DEBUG ResourceRegistryStandardImpl:73 - HHH000387: ResultSet's
statement was not registered
2022-01-06 18:47:40 DEBUG AbstractLoadPlanBasedEntityLoader:189 - Done entity load :
ma.cigma.pfe.models.Client#2
Client{id=2, name='SIHAM'}
```

Cas pratiques

	<p>Ajouter le use case</p> <ul style="list-style-type: none"> • <code>findAll()</code> : Qui affiche tous les clients <p>Pour le faire ajouter et implémenter les méthodes suivantes:</p> <p><u>Client Dao Layer</u></p> <ul style="list-style-type: none"> → Ajouter <code>List<Client> findAll()</code>; à l'interface → Implémenter cette signature dans la classe <p><u>Client Service Layer</u></p> <ul style="list-style-type: none"> → Ajouter <code>List<Client> getAll()</code>; à l'interface → Implémenter cette signature dans la classe <p><u>Client Controller Layer</u></p> <ul style="list-style-type: none"> → Ajouter <code>List<Client> getAll()</code> au controller <p><u>Application Runner Class</u></p> <ul style="list-style-type: none"> → Tester votre méthode dans <code>ApplicationRunner</code> → Afficher la liste dans la console
	<p>Cas pratique pour l'utilisation des annotations JPA</p> <p><i>(ce cas pratique sera noté sur 10 points dans la note des TP8)</i></p> <p>En ce basant sur le TP8 :</p> <p>Créer toutes les classes et les interfaces nécessaires pour les CRUD (Create, Read, Update and Delete) de l'entité Facture.</p> <p>Pour le moment, une facture est définie par:</p> <ul style="list-style-type: none"> • <code>id</code> (long) • <code>date</code> (<code>java.util.Date</code>) • <code>amount</code> (double) <p><i>Les tests seront vérifié au niveau de votre base MySQL</i></p>