

Objectifs TP3

(week-end 16/01/2021)

Les objectifs de la séance d'aujourd'hui:

Objectif 3.1 : Création des **annotations** en Java

Objectif 3.2 : Rappel **Three Layered Architecture**

Objectif 3.3 : Rappel **Inversion Of Control, Dependency Injection**

Objectif 3.4 : Commencer à utiliser les annotations **JPA**

Complément Youtube de cette séance

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

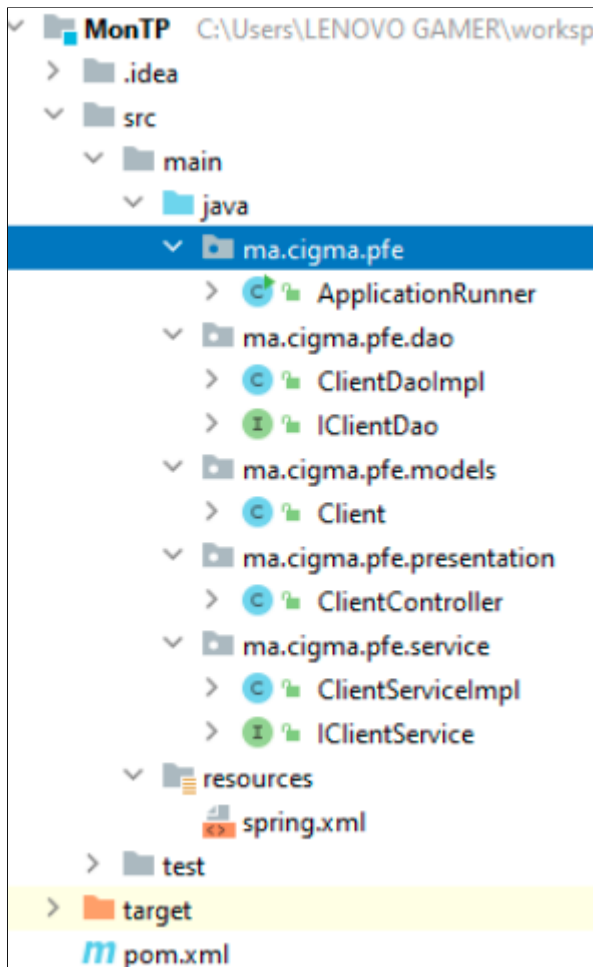
I. JPA et ses annotations

INTRODUCTION

CE TP CONSISTE À IMPLÉMENTER LA COUCHE "DAO" D'UN PROJET [CRÉER DANS LE TP2] EN UTILISANT L'IMPLÉMENTATION HIBERNATE DE LA SPÉCIFICATION JPA. IL EST DEMANDÉ D'UTILISER LES ANNOTATIONS DE BASE EXIGÉES PAR LA SPÉCIFICATION JPA.

1. OUVRIR LE PROJET CRÉÉ DANS LE TP2

CI-APRÈS L'ARBORESCENCE DU PROJET DU TP2



A CE STADE DE LA FORMATION, LE PROJET TP2 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]
- ☐ 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 TP2 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>

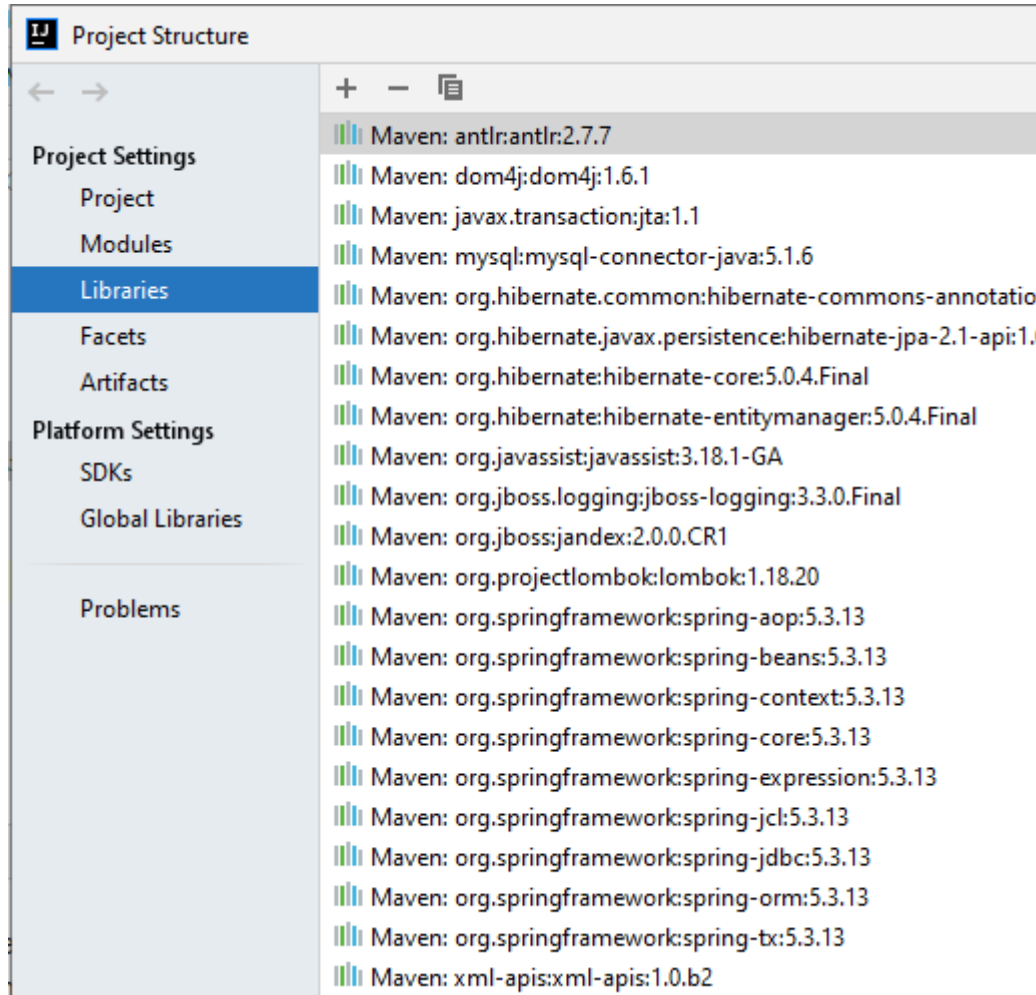
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <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>
```

```
</dependencies>
```

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



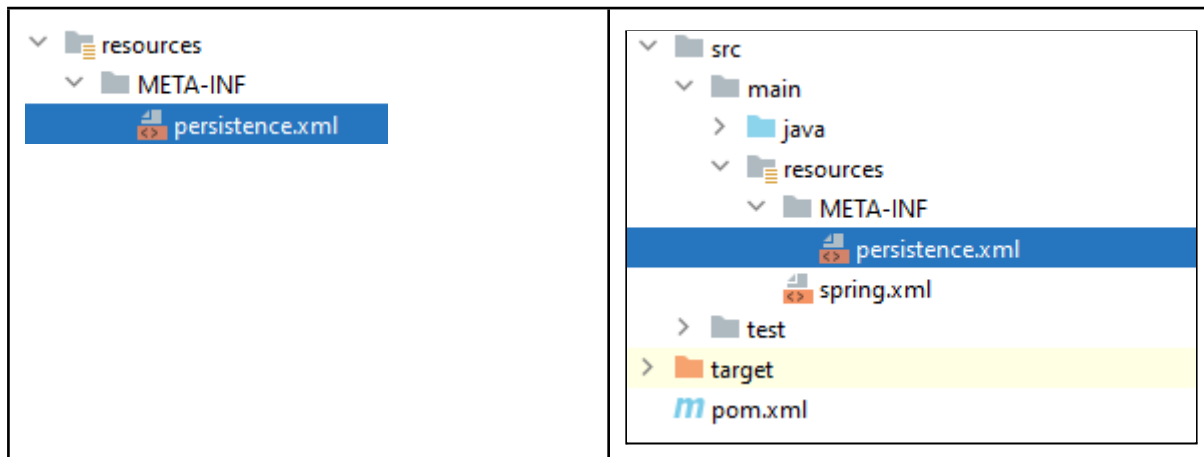
VÉRIFIER LES JARS DANS LOCAL REPOSITORY : C:\Users\VOTRE_USER\.m2

PERSISTENCE.XML : PARAMETRES DU MODELE PHYSIQUE DES DONNÉES

CRÉER UN "SOURCE FOLDER" NOMMÉ "SRC/MAIN/RESOURCES"

CRÉER DANS "SRC/MAIN/RESOURCES", UN "FOLDER" NOMMÉ "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"
/>
            <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" />
        </properties>
    </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 + '\'' +
            '}';
    }
}
```

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 {
    boolean 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 boolean save(Client c) {
    em.getTransaction().begin();
    em.persist(c);
    em.getTransaction().commit();
    return true;
}
```

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 boolean save(Client c) {
    em.getTransaction().begin();
    em.persist(c);
    em.getTransaction().commit();
    return true;
}
}
```

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 x
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
        (?)
```