

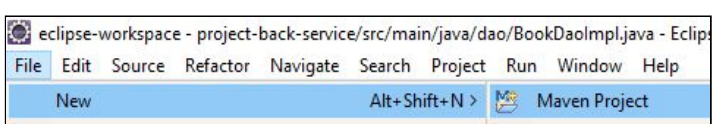
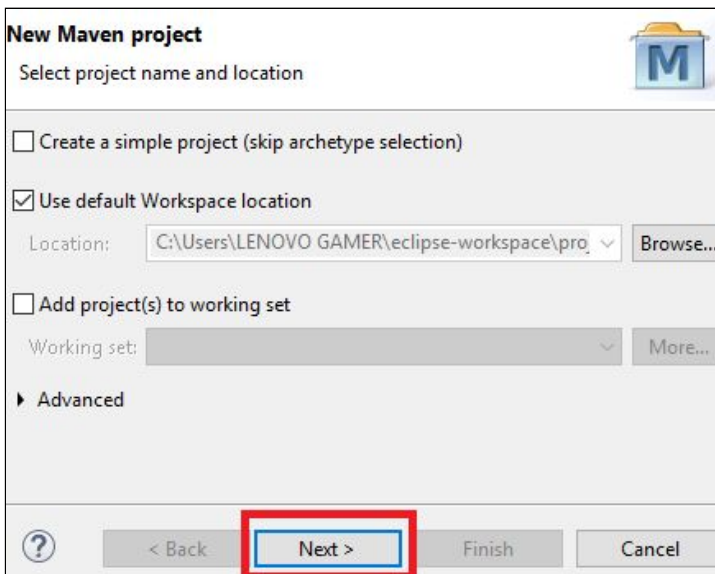
TRAVAUX PRATIQUE : JPA N°1

INTRODUCTION

CE TP CONSISTE À IMPLÉMENTER LA COUCHE “DAO” D’UN PROJET BACK OFFICE EN UTILISANT L’IMPLÉMENTATION HIBERNATE DE LA SPÉCIFICATION JPA. IL EST DEMANDÉ D’UTILISER LES ANNOTATIONS DE BASE EXIGÉES PAR LA SPÉCIFICATION JPA.

Nb : IL EST À SAVOIR QUE CES TRAVAUX PRATIQUES SERONT NOTÉES.

I. CREATION DU PROJET MAVEN

ETAPE 1: NEW PROJECT MAVEN	
ETAPE 2: WORKSPACE LOCATION	

ETAPE 3: SELECT ARCHETYPE

New Maven Project

New Maven project

Select an Archetype

Catalog: New Maven Catalogue 1

Filter: org.apache.maven 2

Group Id	Artifact Id
org.apache.maven.archetypes	maven-archetype-portlet
org.apache.maven.archetypes	maven-archetype-profiles
org.apache.maven.archetypes	maven-archetype-quickstart
org.apache.maven.archetypes	maven-archetype-simple
org.apache.maven.archetypes	maven-archetype-site
org.apache.maven.archetypes	maven-archetype-site-simple
org.apache.maven.archetypes	maven-archetype-site-skin
org.apache.maven.archetypes 3	maven-archetype-webapp

An archetype which contains a sample Maven Webapp project.
https://repo1.maven.org/maven2

☒ Show the last version of Archetype only ☐ Include snapshot archetypes Add Archetype...

Advanced

< Back Next > 4 Finish Cancel

ETAPE 4 : YOUR PROJECT AND YOUR PACKAGE NAME

New Maven Project

New Maven project

Specify Archetype parameters

Group Id: ma.fst.test 1 nom du package

Artifact Id: BackProject 2 nom du projet

Version: 0.0.1-SNAPSHOT

Package: ma.fst.test.BackProject

Properties available from archetype:

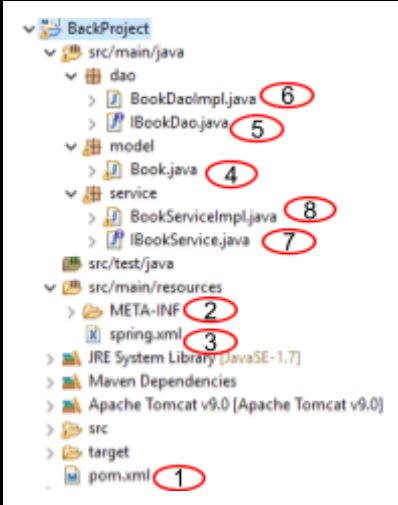
Name	Value

Add... Remove

< Back Next > 3 Terminer Finish Cancel

II. L'ARCHITECTURE DU PROJET À RÉALISER

CI-APRÈS L'ARBORESCENCE DU PROJET À CRÉER AVEC L'EXPLICATION DE CHAQUE COMPOSANT:

	<p>1 : pom.xml pour ajouter les dépendances du projets</p> <p>2: META-INF/persistence.xml : la configuration JPA pour l'accès à la base de données ainsi que les paramètres relatifs aux requette SQL généré et aussi les paramètres de génération du modèle physique des données.</p> <p>3: Le fichier de configuration du spring pour l'inversion du contrôle et aussi l'injection de dépendance.</p> <p>4: La classe modèle considérée comme entité dans ce projet</p> <p>5: L'interface de la couche DAO</p> <p>6: L'implémentation de la couche DAO</p> <p>7: L'interface de la couche Service</p> <p>8: L'implémentation de la couche Service</p>
---	---

POM.XML : DEPENDENCES DU PROJET

AJOUTER AU POM.XML DE VOTRE PROJET LES DÉPENDANCES SUIVANTES.

```

<properties>
    <hibernate.version>5.0.4.Final</hibernate.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>

```

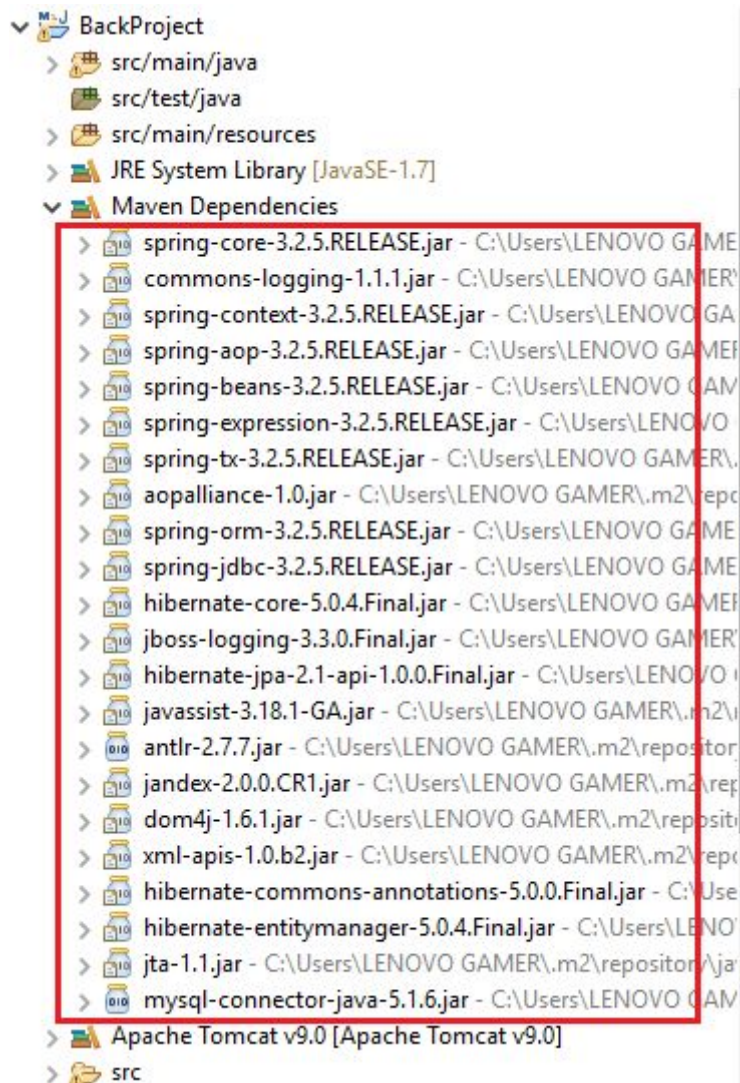
```

</dependency>
<dependency>
    <groupId>javax.transaction</groupId>
    <artifactId>jta</artifactId>
    <version>1.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java
-->

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

```

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



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

PERSISTENCE.XML : PARAMETRES DU MODELE PHYSIQUE DES DONNÉES

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

CREER DANS "SRC/MAIN/RESOURCES", UN "FOLDER" NOMEE "META-INF"

CRÉER UN FICHIER PERSISTENCE.XML DANS SRC/MAIN/RESOURCES/META-INF.

METTRE DANS PERSISTENCE.XML LES PARAMÈTRES DE VOTRE BASE DE DONNÉES SOUS FORME DE PERSISTENCE-UNIT.

```
<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="books">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/books" />
      <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.dialect"
value="org.hibernate.dialect.MySQL5Dialect" />
    </properties>
  </persistence-unit>
</persistence>
```

SPRING.XML : INVERSION DU CONTRÔLE ET INJECTION DES DÉPENDANCES

DANS "SRC/MAIN/RESOURCES" CRÉER LE FICHIER DE CONFIGURATION DE SPRING. NOMMER LE SPRING.XML : CE FICHIER PERMET PRINCIPALEMENT DE DÉFINIR LES PACKAGES POUR SCANNER LES COMPOSANTS SPRING. INSTANCIER UN OBJET DE LA CLASSE ENTITYMANAGERFACTORY PAR SPRING IOC.

```
<?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:context="http://www.springframework.org/schema/context"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:p="http://www.springframework.org/schema/p"
  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" />
```

```

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

```

MODEL.BOOK.JAVA : CLASSE MODELE ET ENTITE

UTILISER LA ANNOTATIONS SUIVANTES DANS LA CLASSE ENTITÉ 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 BOOK.JAVA POUR NOTRE CAS

```

package model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="tbooks")
public class Book {
    @Id // simple primary key
    //@IdClass @EmbeddedId composite key
    @GeneratedValue (strategy=GenerationType.IDENTITY)

```

```

        private int id;
        @Column(name="title")
        private String title;

        public Book() {
            super();
        }
        public Book(String title) {
            super();
            this.title = title;
        }
        public int getId() {
            return id;
        }
        public void setId(int id) {
            this.id = id;
        }
        public String getTitle() {
            return title;
        }
        public void setTitle(String title) {
            this.title = title;
        }
    }
}

```

DAO.IBOOKDAO.JAVA : INTERFACE DE LA COUCHE DAO

DÉFINIR LES MÉTHODES QUE LA COUCHE SERVICE PEUT APPELER DANS L'INTERFACE DAO.IBOOKDAO. DANS UN PREMIER TEMPS UNE SEULE MÉTHODE INSERT(BOOK B) SERA EXPOSÉE POUR LA PARTIE SERVICE. CI-APRES L'INTERFACE EN QUESTION:

```

package dao;
import model.Book;
public interface IBookDao {
    void insert(Book b);
}

```

DAO.BOOKDAOIMPL.JAVA : IMPLÉMENTATION DE LA COUCHE DAO

IMPLÉMENTER LA MÉTHODE : INSERT(BOOK B) DANS UNE CLASSE DAO.BOOKDAOIMPL.JAVA
 UTILISER L'ANNOTATION @REPOSITORY POUR DÉCLARER CETTE CLASSE COMPOSANT SPRING.
 UTILISER L'ANNOTATION @PERSISTENCECONTEXT POUR INJECTER UN OBJET DE TYPE ENTITYMANAGERFACTORY ET PAR LA SUITE UN OBJET ENTITYMANAGER.

```

package dao;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.stereotype.Repository;
import model.Book;
@Repository
public class BookDaoImpl implements IBookDao {

```

```

        @PersistenceContext
        EntityManager em;
        @Override
        public void insert(Book b) {
            em.persist(b);
        }
    }
}

```

IL EST A NOTER QUE **@PersistenceContext** REMPLACE LES DEUX INSTRUCTIONS JAVA:

```

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

```

ESSAYER D'INSTANCIER L'ENTITY MANAGER PAR LES DEUX LIGNES CI-DESSOUS.

SERVICE.IBOOKSERVICE.JAVA : INTERFACE DE LA COUCHE SERVICE

DÉFINIR LES MÉTHODES QUE LA COUCHE PRESENTATION PEUT APPELER DANS L'INTERFACE SERVICE.IBOOKSERVICE. DANS UN PREMIER TEMPS UNE SEULE MÉTHODE ADD(BOOK B) SERA EXPOSÉE POUR LA PARTIE PRESENTATION. CI-APRÈS L'INTERFACE EN QUESTION.

```

package service;
import model.Book;
public interface IBookService {
    void add(Book b);
}

```

SERVICE.BOOKSERVICEIMPL.JAVA : IMPLÉMENTATION DE LA COUCHE SERVICE

IMPLÉMENTER LA MÉTHODE : ADD(BOOK B) DANS UNE CLASSE SERVICE.BOOKSERVICEIMPL.JAVA
 UTILISER L'ANNOTATION **@Service** POUR DÉCLARER CETTE CLASSE COMPOSANT SPRING.
 UTILISER L'ANNOTATION **@Autowired** POUR INJECTER UN OBJET DE TYPE IBOOKDAO PAR L'IOC.
 UTILISER L'ANNOTATION **@Transactional** POUR GÉRER LES COMMIT ET LES ROLLBACK DES OPÉRATIONS AU NIVEAU DE LA BASE DE DONNÉES.

```

package service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import dao.IBookDao;
import model.Book;
@Service
public class BookServiceImpl implements IBookService {
    @Autowired
    IBookDao dao;

    @Transactional
    @Override
    public void add(Book b) {
        dao.insert(b);
    }
}

```


SERVICE.TEST.JAVA : TEST DE LA COUCHE SERVICE ET LA COUCHE DAO

CRÉER UNE CLASSE POUR TESTER LA MÉTHODE ADD (BOOK B) DE LA COUCHE SERVICE ET LA MÉTHODE INSERT (BOOK B) DE LA COUCHE DAO. CI-APRÈS LA CLASSE DE TEST:

```
package service;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import model.Book;
public class Test {
    public static void main(String[] args) {
        ApplicationContext ap= new
ClassPathXmlApplicationContext("spring.xml");
        IBookService service= (IBookService)ap.getBean("bookServiceImpl");
        service.add(new Book("JAVA"));
    }
}
```

EXERCICE : COMPLÉMENT DE LA COUCHE DAO ET LA COUCHE SERVICE POUR COUVRIR TOUTES LES OPÉRATIONS CRUD (CREATE READ UPDATE DELETE)

1) COMPLÉTER L'INTERFACE DAO.IBOOKDAO EN AJOUTANT LES MÉTHODES SUIVANTES:

```
package dao;
import model.Book;
public interface IBookDao {
    void insert(Book b);
    void update(Book b);
    void delete(Book b);
    List<Book> selectAll();
}
```

2) COMPLÉTER L'IMPLÉMENTATION DE L'INTERFACE DAO.IBOOK

3) COMPLÉTER L'INTERFACE SERVICE.IBOOK SERVICE EN AJOUTANT LES MÉTHODES SUIVANTES:

```
package service;
import model.Book;
public interface IBookService {
    void add(Book b);
    void modifier(Book b);
    void supprimer(Book b);
    List<Book> chercherAll();
}
```

4) TESTER VOS MÉTHODES DANS LA CLASSE DE TEST: