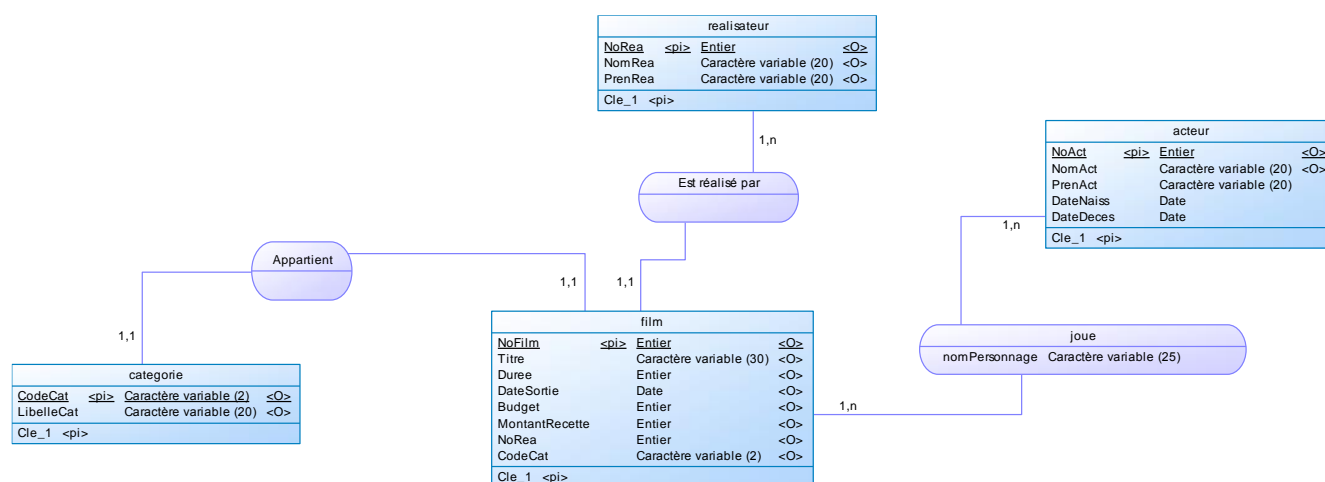


# Projet de synthèse : Cinéma

## Présentation du sujet

### CinemaEpul

Le site web **cinemaEpul.fr** permet de consulter une présentation des films français. Il utilise une base de données, dont voici le schéma conceptuel :



La base de données : au schéma conceptuel vous est fournie :

- Cinema.sql

## Fonctionnalités de l'application

L'application Service doit gérer les objets métiers

- **Ajout – Modification - Suppression**
  - o Film
  - o Joue
- **Recherche/Affichage**
  - o Films sur critère
  - o Acteurs avec leur rôle dans le film
  - o .....

Le client consommateur des services recevra des flux de données par des appels RestFul et affichera les résultats dans la couche de présentation.

## Equipe

Le travail en équipe ne veut pas dire comme on le voit trop souvent se mettre à deux ou trois devant un ordinateur, un qui tape, l'autre qui commente et le troisième qui consulte ses textos !

On peut parfaitement imaginer deux équipes, une qui se concentrera sur le web service et l'autre sur le client. Le code commun sera mis sur la forge logicielle que vous utiliserez.

## Remise des TPS

Les TP devront être remis sous forme du dossier du projet de l'application zippé et renommés de la forme suivante : nom des auteurs, exemple :

DUPONT\_MARTIN\_BRUN\_DURAND.zip

L'utilisateur de la BD sera : userepul/epul.

Si vous modifiez la base de données, fournissez le script de la base de données.

Le mode de dépôt et sa date limite seront fournis oralement.

Le fichier décompressé DOIT être immédiatement exploitable tel quel dans l'environnement annoncé sans autre manipulation, faites donc le test avant de le déposer.

## Environnement de travail

Cette application sera réalisée avec :

- l'IDE de votre choix
- projet Spring, SpringBoot ou ....
- java JDK 1.8
- base de données Mysql
- Web Service RestFull générant un flux json
  - o Persistance avec Entity
  - o Dépendances avec Maven
  - o Serveur Tomcat ou WildFly
- Client consommateur
  - o Toute technologie de présentation est acceptée
    - Pages JSP
    - Angular JS
    - .....

## Annexe

### Projet Maven

Pour enlever l'erreur de votre fichier pom.xml qui stipule que vous n'avez pas de fichier web.xml, ajoutez les lignes suivantes

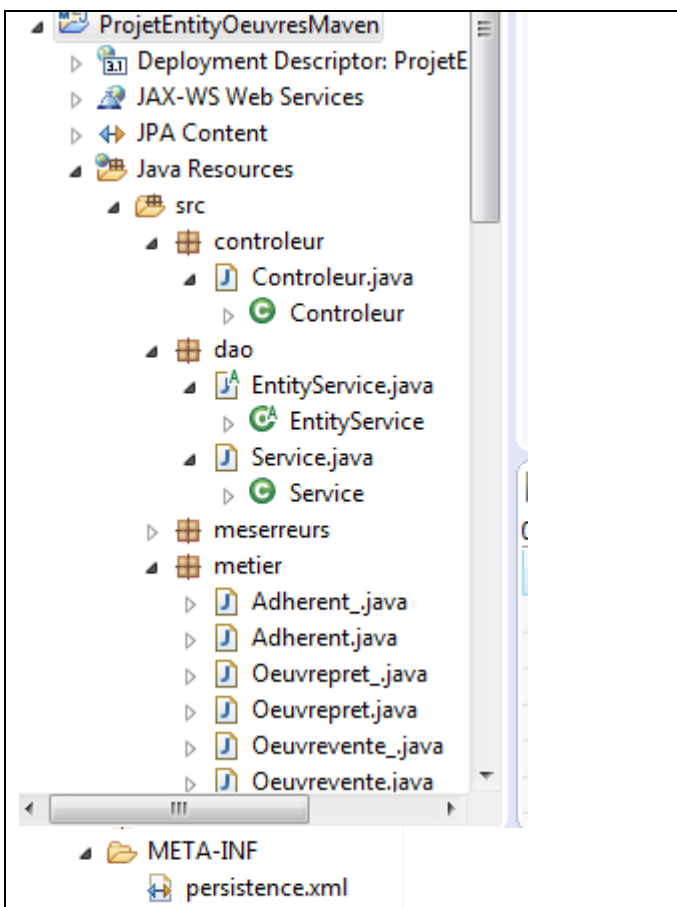
## Pom.xml

```
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-war-plugin</artifactId>
<configuration>
<failOnMissingWebXml>>false</failOnMissingWebXml>
</configuration>
</plugin>
</plugins>
</build>
```

## Annexe

### Projet Entity/maven

#### Architecture

	<p>La classe DAO contient la classe abstraite EntityService qui chaîne sur le fichier persistence.xml Et la classe Service qui offre les méthodes d'accès aux données</p>
--	---

## EntityService

```
package dao;

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

public abstract class EntityService
{
    protected EntityManager entitymanager;
    protected EntityManagerFactory emf;

    public EntityTransaction startTransaction()
    {
        emf=Persistence.createEntityManagerFactory("POeuvre");
        entitymanager=emf.createEntityManager();
        return entitymanager.getTransaction();
    }
}
```

## *Exemple d'appel*

```
package dao;

import meserreurs.MonException;
import java.util.*;

import javax.persistence.EntityTransaction;
import metier.*;

public class Service extends EntityService {

    // ajout 'un adhérent

    public void insertAdherent(Adherent unAdherent) throws MonException {

        try {

            EntityTransaction transac = startTransaction();
            if (!entitymanager.contains(unAdherent)) {
                transac.begin();
                entitymanager.persist(unAdherent);
                entitymanager.flush();
                transac.commit();
            }
            entitymanager.close();
        } catch (Exception e) {
            new MonException("Erreur d'insertion", e.getMessage());
        }
    }
}
```

}

## *persistence.xml*

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" 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">
  <persistence-unit name="POeuvre">
    <class>metier.Adherent</class>
    <class>metier.Oeuvrepret</class>
    <class>metier.Oeuvrevente</class>
    <class>metier.Proprietaire</class>
    <class>metier.Reservation</class>
    <class>metier.ReservationPK</class>
    <class>metier.Sequence</class>
    <properties>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQLDialect" />
      <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/baseoeuvre" />
      <property name="javax.persistence.jdbc.user" value="userepul" />
      <property name="javax.persistence.jdbc.password" value="epul" />
      <property name="hibernate.show_sql" value="true"/>
    </properties>
  </persistence-unit>
</persistence>

```

## *pom.xml*

```

<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.36</version>
  </dependency>
  <dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>4.2.7.Final</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>4.2.7.Final</version>
    <scope>compile</scope>
  </dependency>
</dependencies>

```

# Méthode Post

## Passage de paramètre : FormParam

On utilise **FormParam** pour passer un paramètre en méthode POST  
package ws;

```
import javax.ws.rs.*;
import javax.ws.rs.FormParam;
import com.google.gson.Gson;
```

```
import metier.Activite;
import metier.Client;
import metier.Emplacement;
import metier.Sejour;
import service.EntityServiceImpl;
```

```
import java.util.*;
import java.text.*;
```

```
@Path("/ressources")
public class WSResource {
    // @Context
    // private UriInfo context;
```

```
    /** Creates a new instance of WsSalutation */
    public WSResource() {
    }
```

```
    @POST
    @Path("/addSejour")
    @Consumes("application/x-www-form-urlencoded")
    @Produces("application/json")
```

```
    public String addSejour(@FormParam("unSej") String unSejour )
        throws ParseException {
        EntityServiceImpl ems = new EntityServiceImpl();
        // on récupère le séjour dans un flux json et on le parse en objet
        Gson gson = new Gson();
        Sejour unSej = gson.fromJson(unSejour, Sejour.class);
        Try
        {
        }
        Catch {}
    }
```

## Client

```
import javax.ws.rs.core.Form;
```

```
public String postAjoutJson(String action, Object unObj)
{
    Response uneChaine;

    WebTarget target = Consommateur.get().target;
    target = target.path(action);
    System.out.println(" uri :" + target.getUri());

    Form fjson = new Form();
    Gson gson = new Gson();
    // on encode au format json
    String json=gson.toJson(unObj);
    fjson.param("unSej", json);
    try
    {
        uneChaine = target.request().accept(MediaType.APPLICATION_JSON).
post(Entity.entity(fjson, MediaType.APPLICATION_FORM_URLENCODED),Response.class);
        ....
    }
    Catch {}

}
```