Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Compte rendu

Filière :

« Génie du Logiciel et des Systèmes Informatiques Distribués »

GLSID

ORM avec Spring Data JPA

Réalisé par :

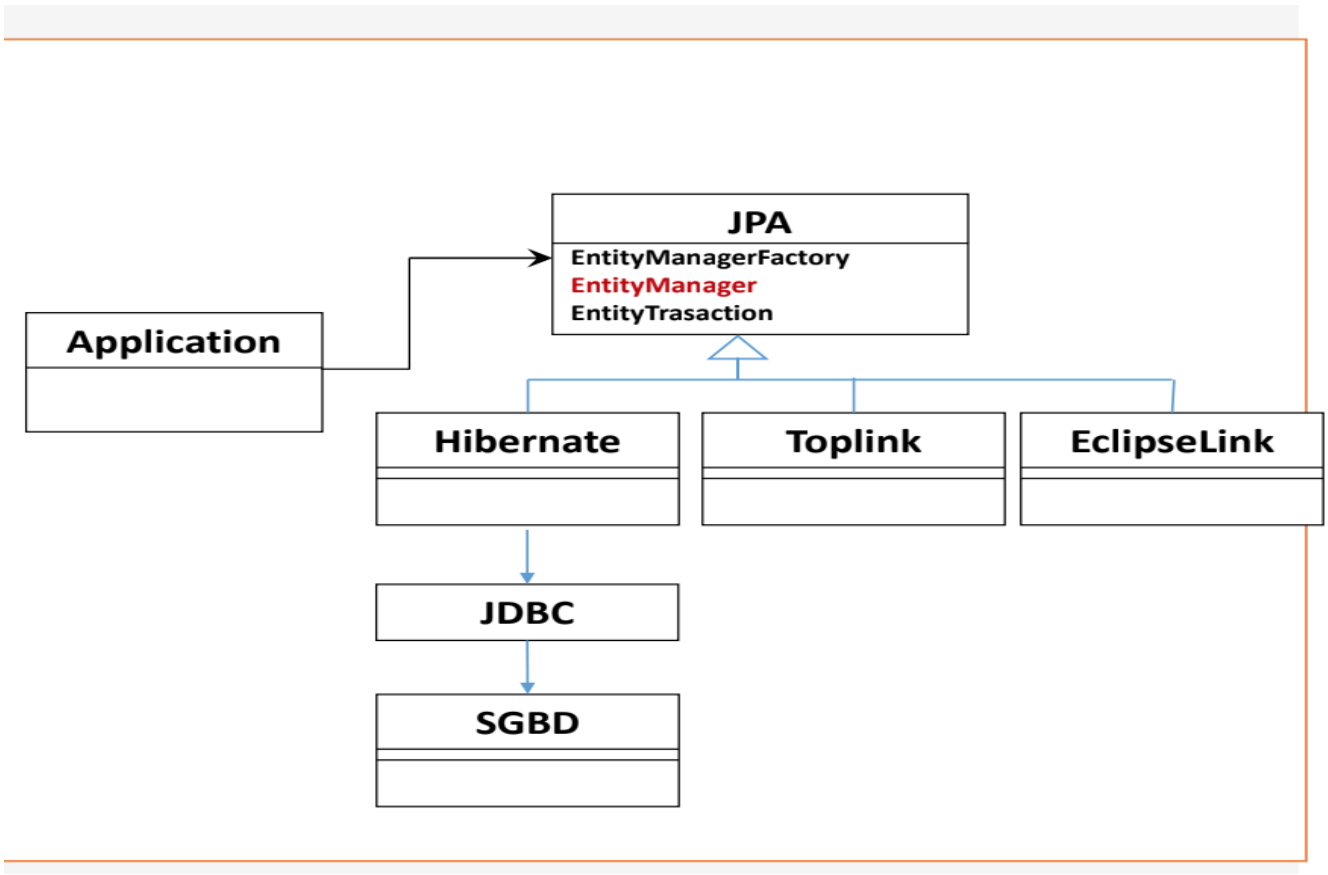
Hassan ELMAKHLOUFI

Encadré par :

Pr. Mohamed Youssfi

Année Universitaire : 2021-2022

Dans ce TP on va faire le premier pas dans Spring data ci-dessous l'architechture globle de l'application:



En premier Lieu j'ai créé l'entités Patient qui sera persisté :

```java
import java.util.Date;
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Patient {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY) // clé primaire
    private Long id;
    @Column(length = 50)
    private String nom;
    @Temporal(TemporalType.DATE)
    private Date dn;
    private boolean malade;
    private int score;
}
```

En utilisant Spring Data, vous n'aurez plus besoin de faire appel à l'objet EntityManager pour gérer la
persistance. Spring Data le fait à votre place.
Spring Data nous évite de créer les interfaces et les implémentation JPA de la couche DAO.

```java
package ma.enset.jpaap.Repositories;

import ma.enset.jpaap.entities.Patient;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.Date;
import java.util.List;

public interface PatientRepository extends JpaRepository<Patient,Long> {
    public List<Patient> findByMalade(boolean m);
    public List<Patient> findByMaladeAndScoreLessThan(boolean m,int score);
    public List<Patient> findByMaladeIsTrueAndScoreLessThan(int score);
    public Page<Patient> findByMalade(boolean m, Pageable pageable);
    public List<Patient> findByDnBetweenAndAndMaladeIsTrueOrNomLike(Date d1, Date d2, String name);
    @Query("select p from Patient p where p.dn between :x and :y or p.nom like :z")
    public List<Patient> chercherPatient(@Param("x") Date d1,@Param("y") Date d2,@Param("z") String name);
}
```

Classe main dans laquelle nous allons ajouter des donnée dans la base de données H2 :

```java
import java.util.Date;

@SpringBootApplication
public class JpaApApplication implements CommandLineRunner {

    @Autowired
    private PatientRepository patientRepository;
    public static void main(String[] args) {
        SpringApplication.run(JpaApApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        for (int i=0;i<100;i++) {
            patientRepository.save(new Patient( id: null, nom: "hassan",new Date(), malade: false, score: 5));
        }
```

Pagination :
Récupérer les données d'une manière paginer

```
Page<Patient> patients = patientRepository.findAll(PageRequest.of( page: 0, size: 10));

System.out.println("Total pages : " + patients.getTotalPages());
System.out.println("Total elements : " + patients.getTotalElements());
System.out.println("Nombre des elements dans chaque page  : " + patients.getNumberOfElements());

for (Patient p:patients) {
    System.out.println(p);
}
```

Rechercher des éléments par des critère spécifique

```
Patient patient = patientRepository.findById(1L).orElseThrow(()->new RuntimeException("Patient Not Found"));
System.out.println(patient);

System.out.println("*************************************************");

patient.setScore(0);
patientRepository.save(patient);
System.out.println(patient = patientRepository.findById(1L).orElseThrow(()->new RuntimeException("Patient Not F

System.out.println("*************************************************");

Page<Patient> patientList = patientRepository.findByMalade( m: false,PageRequest.of( page: 0, size: 4));
for (Patient p:patientList) {
    System.out.println(p);
}
```

Se connecter au console de base de donnees H2 :