**Sever**

System.out.println("Start build Server site");

SecurityManager securityManager = System.getSecurityManager();

if(securityManager == null) {

System.setProperty("java.security.policy", "policy\\policy.policy");

System.setSecurityManager(new SecurityManager());}

ICinemaDAO cinemaDAO = new CinemaDAO();

IMovieDAO movieDAO = new MovieDAO();

LocateRegistry.createRegistry(10016);

Naming.rebind("rmi://localhost:10016/cinemaRemote", cinemaDAO);

Naming.rebind("rmi://localhost:10016/movieRemote", movieDAO);

System.out.println("Server is ready to connect on port 10016 ");

**---------------------------------------------------------------------------------**

**Client**

SecurityManager securityManager = System.getSecurityManager();

if(securityManager == null) {

System.setProperty("java.security.policy", "policy\\policy.policy");

System.setSecurityManager(new SecurityManager());}

ICinemaDAO cinemaDAO = (ICinemaDAO)

Naming.lookup("rmi://localhost:10016/cinemaRemote");

IMovieDAO movieDAO = (IMovieDAO)

Naming.lookup("rmi://localhost:10016/movieRemote");

**---------------------------------------------------------------------------------**

<provider>org.hibernate.ogm.jpa.HibernateOgmPersistence</provider>

<class>entity.Cinema</class>

<class>entity.Show</class>

<class>entity.Movie</class>

<properties>

<property name="hibernate.ogm.datastore.provider"

value="mongodb" />

<property name="hibernate.ogm.datastore.database"

value="moviedb" />

<property name="hibernate.ogm.datastore.host"

value="localhost" />

<property name="hibernate.ogm.mongodb.port"

value="27017" />

<property name="hibernate.ogm.datastore.create\_database"

value="true" />

</properties>

@Entity

@Table(name = "cinemas")

public class Cinema implements Serializable {

@Id

private Long id;

@ElementCollection(fetch = FetchType.EAGER)

private List<Show> listShow;

@Embeddable

public class Show implements Serializable {

private LocalDateTime timing;

@ManyToOne

@JoinColumn(name = "movieID")

private Movie movie;

public interface ICinemaDAO extends Remote{

public boolean addCinema(Cinema cn) throws RemoteException;

public class CinemaDAO extends UnicastRemoteObject implements ICinemaDAO {

private EntityManager em;

public CinemaDAO() throws RemoteException {

em = MovieEntityManager.getInstance().getEntityManager();

public boolean addCinema(Cinema cn) throws RemoteException {

boolean result = false;

EntityTransaction tr = em.getTransaction();

try { tr.begin();em.persist(cn);tr.commit();

// update em.merge(cn);

// find em.find(Cinema.class, id);

result = true;}

catch (Exception e) { tr.rollback();e.printStackTrace();}

return result;}  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

private static MovieEntityManager instance = null;

private EntityManager em;

private MovieEntityManager() {

em =Persistence.createEntityManagerFactory("SERVICE").createEntityManager();}

public synchronized static MovieEntityManager getInstance() { if(instance == null)

instance = new MovieEntityManager();

return instance; }

public EntityManager getEntityManager() {

return em;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

public Cinema getCinema(Long id)

throws RemoteException {

return em.find(Cinema.class, id);}