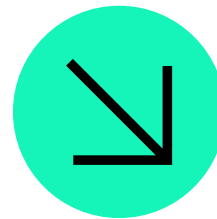




Gestion de la persistance de la base de données Oracle à travers : Hibernate, Spécification JPA et Spring Data

Let's Get Started

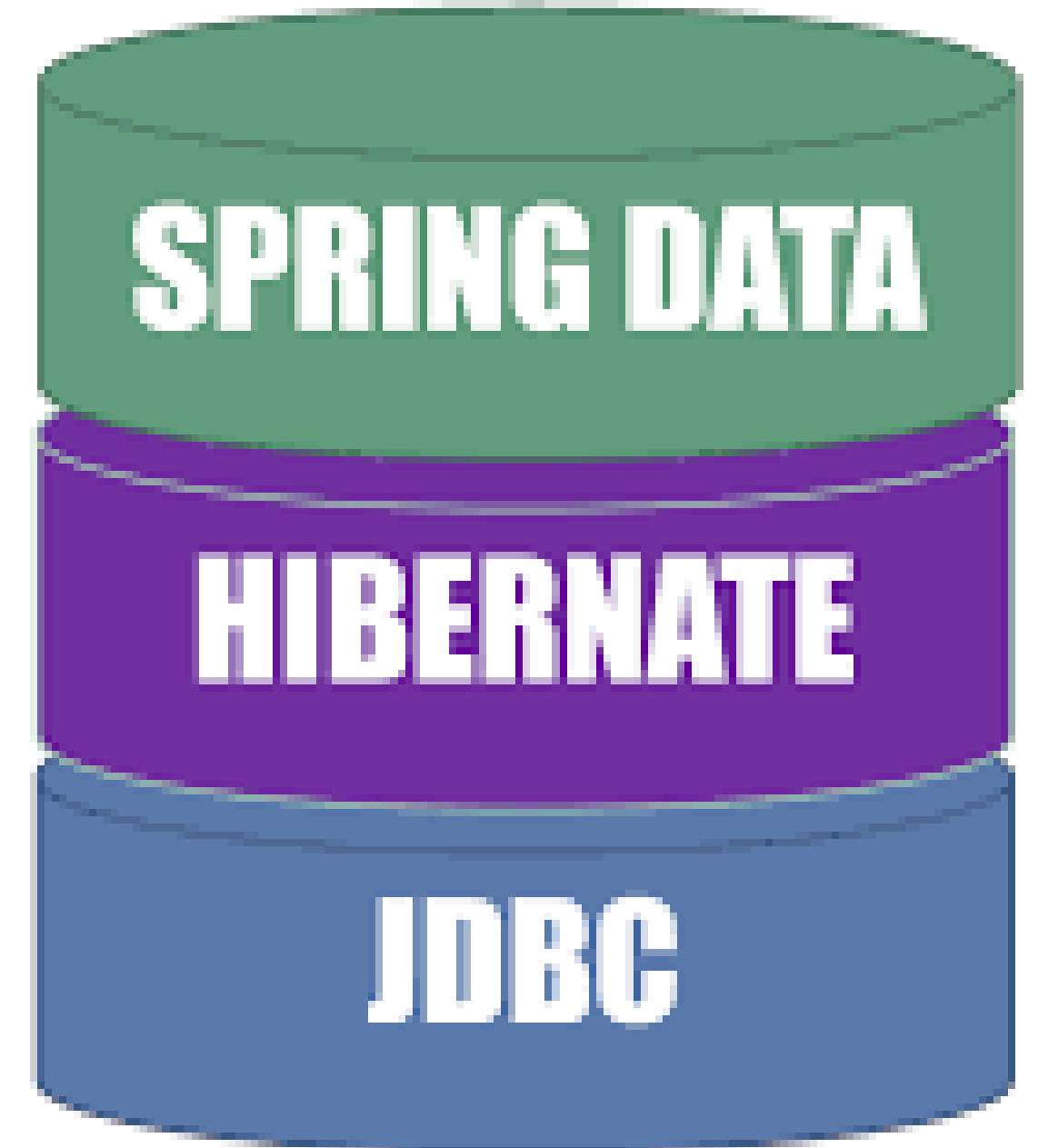


Présenté par

josué KPATCHA
Othman EL HADRATI

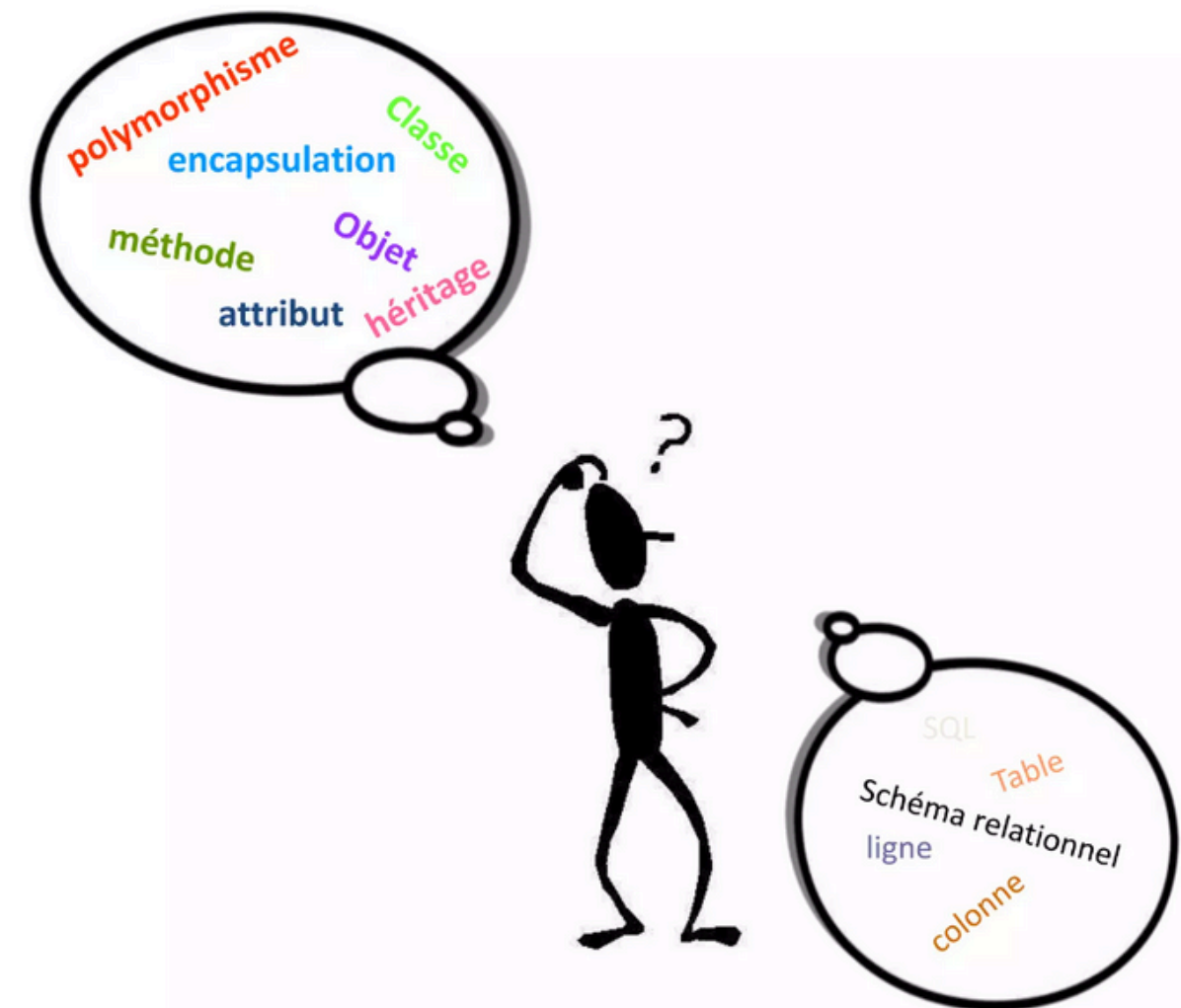
Encadré par :

P.CHERRADI



Introduction

la correspondance entre le modèle relationnel et le modèle d'objet dans les systèmes de gestion de bases de données et les applications JAVA, et comment les technologies comme JPA, Hibernate et Spring Data simplifient ce processus.



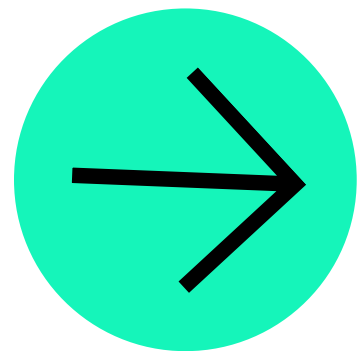
SOMMAIRE

- 1 JDBC limites et Importance de la persistance
- 2 La Persistance des données & ORM
- 3 Présentation de JPA
- 4 ORM : Hibernate
- 5 Intégration de Spring Data avec JPA
- 6 Partie questions
- 7 Conclusion

JDBC (Java DataBase Connectivity)

L'utilisation de JDBC demande fréquemment la rédaction de lignes de code nombreuses et répétitives.

```
*OracleCon.java ×
1 package OracleCon;
2
3 import java.sql.*;
4 class OracleCon{
5     public static void main(String args[]){
6         try{
7             //step1 on recupere le pilote compatible avec Oracle
8             Class.forName("oracle.jdbc.driver.OracleDriver");
9
10            //step2 connexion avec la base de donnee
11            Connection con=DriverManager.getConnection(
12                "jdbc:oracle:thin:@localhost:1521:xe","system","password");
13
14            //step3 executeur des queries
15            Statement stmt=con.createStatement();
16
17            //step4 execute query
18            ResultSet rs=stmt.executeQuery("select * from emp");
19
20            while(rs.next())
21                System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
22
23            //step5 close the connection object
24            con.close();
25
26        }catch(Exception e){ System.out.println(e);}
27    }
28 }
29 }
```

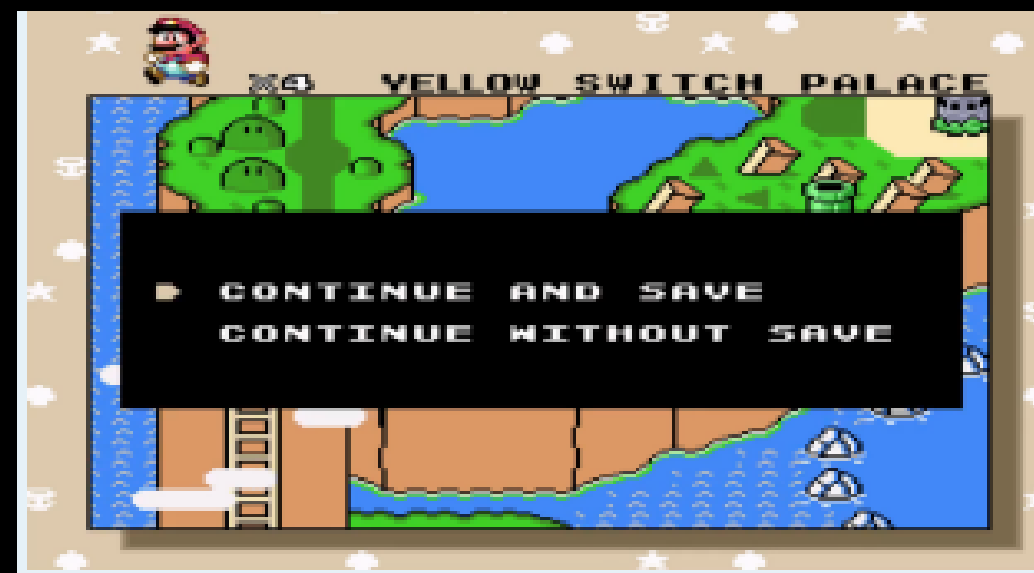


L'importance de Persistance

Pas de persistance
sur Game boy.



Persistance sur
Mario

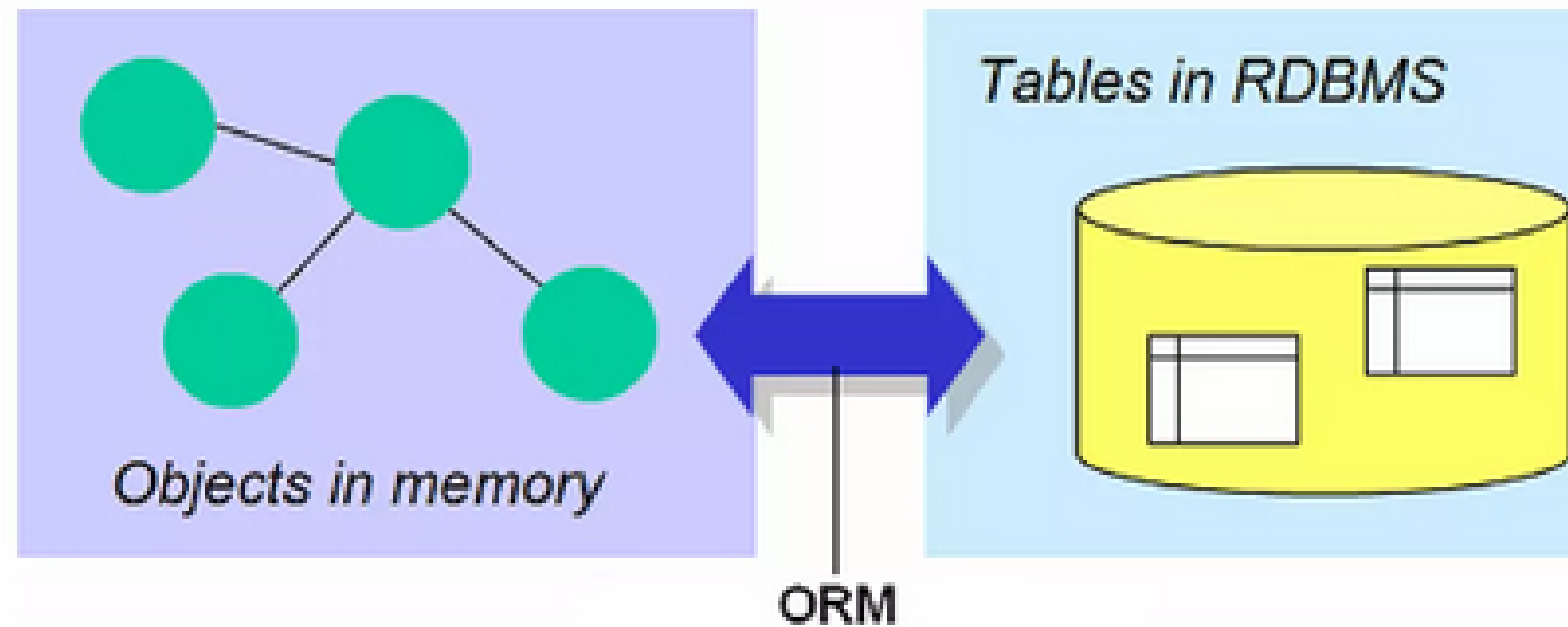


La Persistance des Données & ORM

La Persistance des données est la notion qui traite de la sauvegarde des données contenues dans les objets des applications dans un support informatique comme les fichiers , les bases de données

Le Mapping Objet/Relationnel (Object-Relational Mapping ORM) consiste à réaliser la correspondance entre une application modélisée en conception orienté objet et une base de données relationnelle

L'ORM a pour but d'établir la correspondance entre : une table d'une base de donnée et une classe du modèle objet



FONCTIONNALITES DE ORM

Assurer le mapping des objets

Fournir une interface qui permet l'implémentation des actions de types CRUD

Proposer un langage de requetes indépendant de la base de données cible

Avantage de l'utilisation des outils de mapping

Les solutions de ORM permettent de réduire la quantité de données à produire

Portabilité de votre application

Propose des méthodes d'accès aux bases de données plus efficaces

Les frameworks ORM

1

Toplink

2

Eclipselink

3

Hibernate

4

Apache OpenJPA



Java Persistence API:JPA

Java Persistence API (JPA) est un standard essentiel faisant partie intégrante de la plateforme Java EE. Il définit une spécification permettant la gestion de la correspondance entre les objets Java et une base de données, facilitant ainsi la gestion de la persistance des données

JPA propose un ensemble d'interfaces décrivant comment respecter ce standard. Cependant, pour mettre en œuvre JPA, il est nécessaire d'utiliser un framework ou une solution qui réalise cette implémentation. JPA est défini dans le package `javax.persistence`

Créatioin des entités

Le mapping se fait généralement de deux façons:

- En utilisant des fichiers XML
- En utilisant des annotations JPA

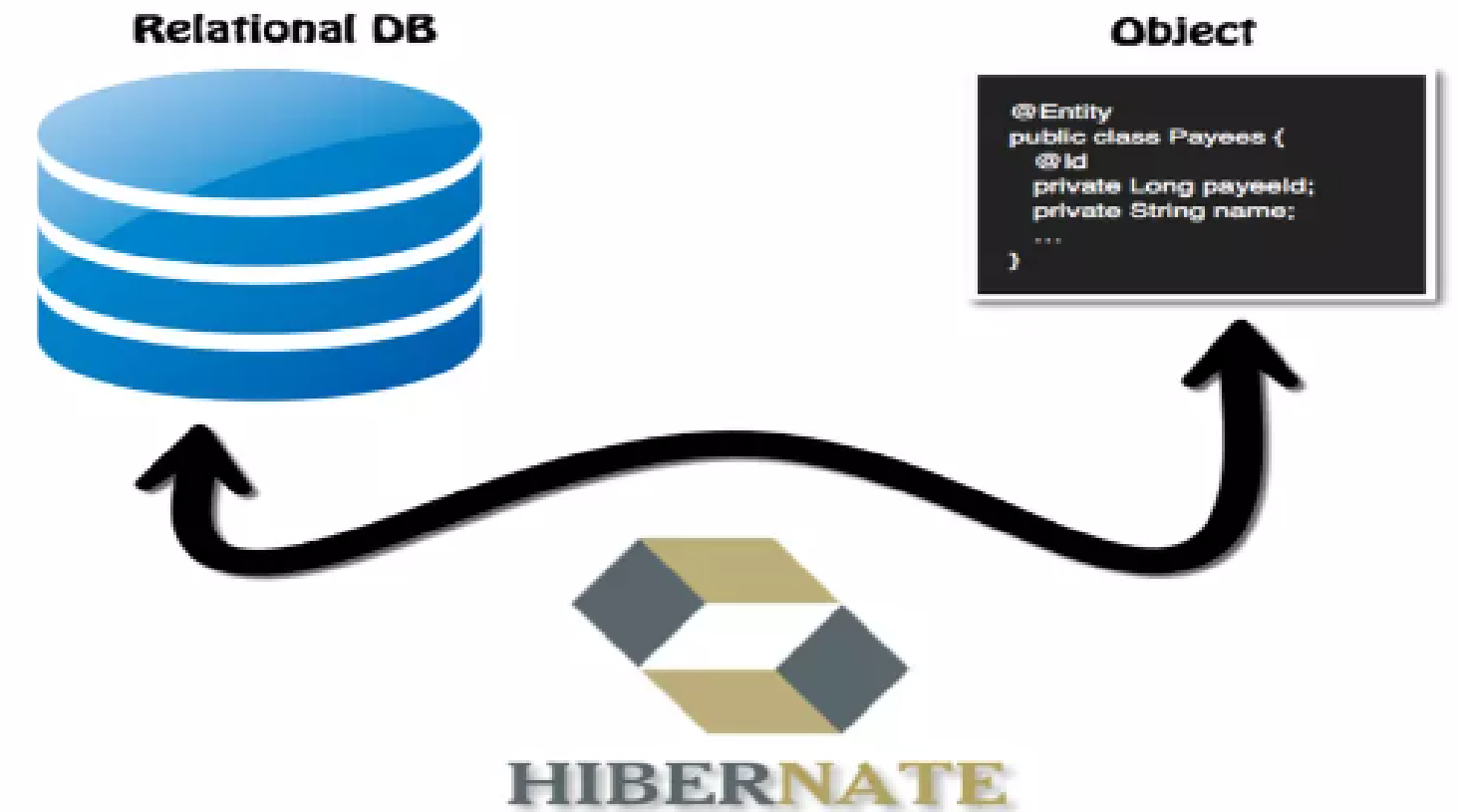
La structure d'un fichier XML configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://xmlns.jcp.org/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence/orm
    http://xmlns.jcp.org/xml/ns/persistence/orm 2.2.xsd"
  version="2.2">
  <!-- Déclaration de l'entité -->
  <entity class="chemin.vers.MaClasse">
    <!-- Définition de la table associée -->
    <table name="NOM_DE_LA_TABLE"/>
    <!-- Définition des attributs de l'entité -->
    <attributes>
      <!-- Attribut ID -->
      <id name="id">
        <generated-value strategy="IDENTITY"/>
      </id>
      <!-- Attribut simple -->
      <basic name="nomAttribut">
        <column name="NOM_COLONNE"/>
      </basic>
      <!-- Attribut Many-to-One -->
      <many-to-one name="nomAttributManyToOne" fetch="LAZY">
        <join-column name="NOM_COLONNE_JOINTURE"/>
      </many-to-one>
      <!-- Autres attributs... -->
    </attributes>
  </entity>
</entity-mappings>
```

Presentation du FrameWork Hibernate

- **C'est quoi exactement
Hibernate ?**

Definition :



Hibernate est une framework ORM qui simplifie le développement d'applications Java .

La methode d'Annotations :

Annotation	Rôle
@javax.persistence.Table	Préciser le nom de la table concernée par le mapping
@javax.persistence.Column	Associé à un getter, il permet d'associer un champ de la table à la propriété
@javax.persistence.Id	Associé à un getter, il permet d'associer un champ de la table à la propriété en tant que clé primaire
@javax.persistence.GeneratedValue	Demander la génération automatique de la clé primaire au besoin
@javax.persistence.Transient	Demander de ne pas tenir compte du champ lors du mapping

Configuration de Hibernate

- L'ajout des dépendances nécessaires pour utiliser jpa , Hibernate et Oracle .

Configuration de Hibernate

- L'ajout des dépendances nécessaires pour utiliser Jpa Hibernate et Oracle .
- Creation d'une classe qui respect JAVA BEAN

Exemple D'application :

Class java bean pour
représenter une personne

```
package entity;

public class Personne {
    private int id;
    private String nom;
    private String prenom;
    private int age;
    public Personne() {}
    // les getters et les setters
}
```


Table 'Personne' dans la base
de données

Table Personne			
	Name	Type	
1	<u>Id Person</u>	Int(11)	Primary Key
2	person_nom	varchar(45)	
3	person_prenom	varchar(45)	
4	person_age	Int(11)	

Exemple D'application :

L'ajout des Annotations JPA
pour faire le mapping

`@GeneratedValue(strategy=GenerationType.IDENTITY)`



```
import javax.persistence.*;
@Entity
public class Personne {
    @Id
    @Column(name="id_person")
    private int id;
    @Column(name="person_nom")
    private String nom;
    @Column(name="person_prenom")
    private String prenom;
    @Column(name="person_age")
    private int age;
    // constructeurs
    public Personne() { }
    // les getters et les setters
}
```

Configuration de Hibernate

- L'ajout des dépendances nécessaires pour utiliser jpa , Hibernate et Oracle .
- Creation d'une classe qui respect JAVA BEAN
- Creation le fichier du persistence : persistence.xml

Exemple D'application :

Configuration du fichier de persistence : persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="UP_CAT">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <properties>
      <property name="hibernate.dialect" value="org.hibernate.dialect.OracleDialect"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.connection.driver_class" value="oracle.jdbc.driver.OracleDriver"/>
      <property name="hibernate.connection.username" value="system"/>
      <property name="hibernate.connection.password" value="system"/>
      <property name="hibernate.connection.url" value="jdbc:oracle:thin:@localhost:1521:xe"/>
    </properties>
  </persistence-unit>
</persistence>
```

Exemple D'application :

Creation d'une interface pour les opérations :

```
import java.util.List;

public interface IPersonne {

    public void save(Personne p);
    public List<Personne> findAll() ;
    public List<Personne> findByName() ;
    public Personne findById( Long id );
    public void update(Personne p) ;
    public void deleteById(Long id);

}
```

Configuration de Hibernate

- L'ajout des dépendances nécessaires pour utiliser jpa , Hibernate et Oracle .
- Creation d'une classe qui respect JAVA BEAN
- Creation le fichier du persistence : persistence.xml
- Utilisation des interfaces JPA :
 - EntityManagerFactory
 - EntityManager
 - EntityTransaction

Exemple D'application :

Implementation d'interface ImpPersonne:

```
import java.util.List;
import javax.persistence.* ;

public class ImpPersonne implements IPersonne {
    // declaration d'objet EntityManager
    //pour la gestion des entites
    private EntityManager entityManager ;
    // Cons
    public ImpPersonne (){
        // vcreation de entity manager pour lier le java application avec la base de donnee
        EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("UP_CAT");
        // creation de l'entity manager qui va interagir avec le BD

        entityManager = entityManagerFactory.createEntityManager() ;
    }
}
```


Exemple D'application :

Overriding des methodes:

```
@Override
public void save(Personne p){
    EntityTransaction transaction = entityManager.getTransaction();
    transaction.begin();
    try{
        entityManager.persist(p);
        transaction.commit() ;
    }catch(Exception e){
        transaction.rollback();
        e.printStackTrace();
    }
}
```

```
@Override
public Personne findById(int idP) {
    return entityManager.find(Personne.class,idP);
}
```

```
@Override
public List<Personne> findAll(){
    Query query=entityManager.createQuery("select p from Personne p");
    return query.getResultList();
}
```

```
@Override
public List<Personne> findByName(String nameP){
    Query query=entityManager.createQuery("select p from Personne p where p.name like : x");
    query.setParameter("x", "%" + nameP + "%") ;
    return query.getResultList();
}

@Override
public void delete(Integer idP) {
    Personne p = this.findById(idP);
    entityManager.remove(p);
}
```

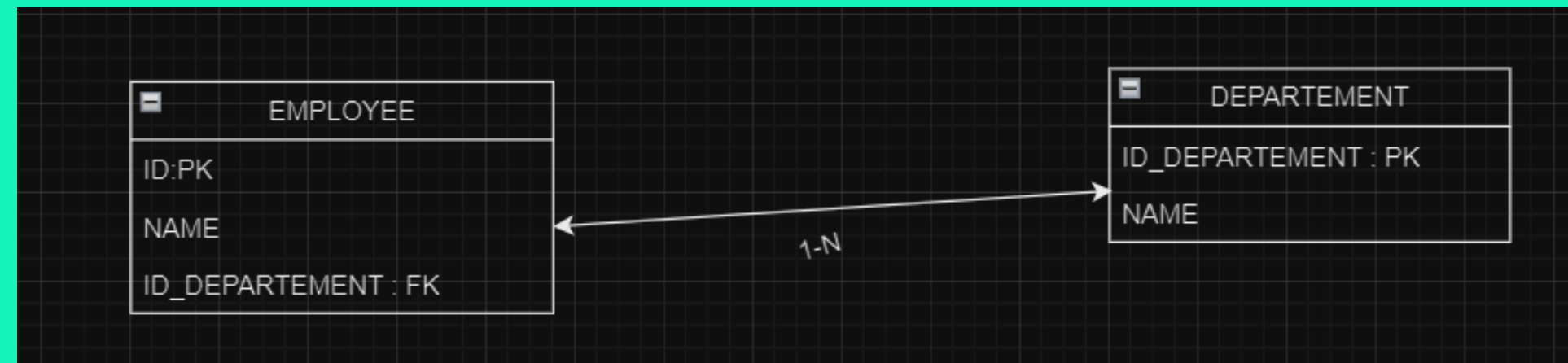
Les Relations entre Les Classes :

Types :

- relation de type **1 - 1** : Annotation associe **@OneToOne**
- relation de type **1 - n** : Annotation associe **@OneToMany**
- relation de type **n - 1** : Annotation associe **@ManyToOne**
- relation de type **n - n** : Annotation associe **@ManyToMany**

Exemple : Les Relations entre Les Classes

Schema Entite-Association



Class Employee

```
1 import javax.persistence.*;
2
3 @Entity
4 public class Employee {
5     @Id
6     @GeneratedValue(strategy = GenerationType.IDENTITY)
7     private Long id;
8
9     private String name;
10
11     @ManyToOne
12     @JoinColumn(name = "department_id")
13     private Department department;
14
15     // Constructeurs, getters et setters
16 }
```

Class Departement

```
import javax.persistence.*;
import java.util.List;

@Entity
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

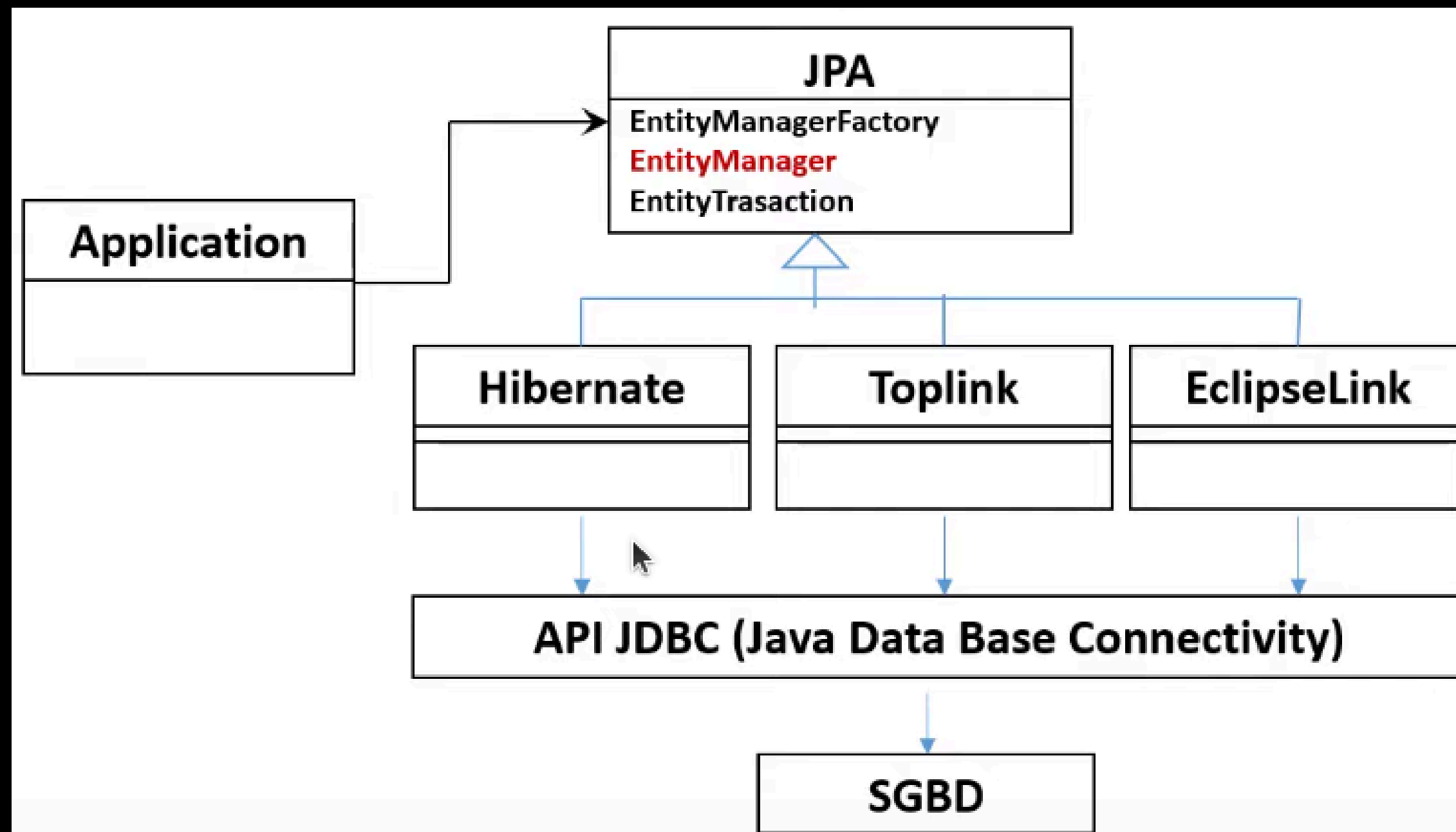
    private String name;

    @OneToMany(mappedBy = "department")
    private List<Employee> employees;

    // Constructeurs, getters et setters
}
```

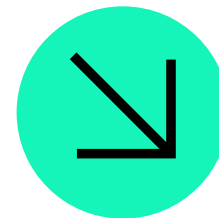
Intégration de Spring Data avec JPA

Résumé de l'Architecture de la partie



Intégration de Spring Data avec JPA

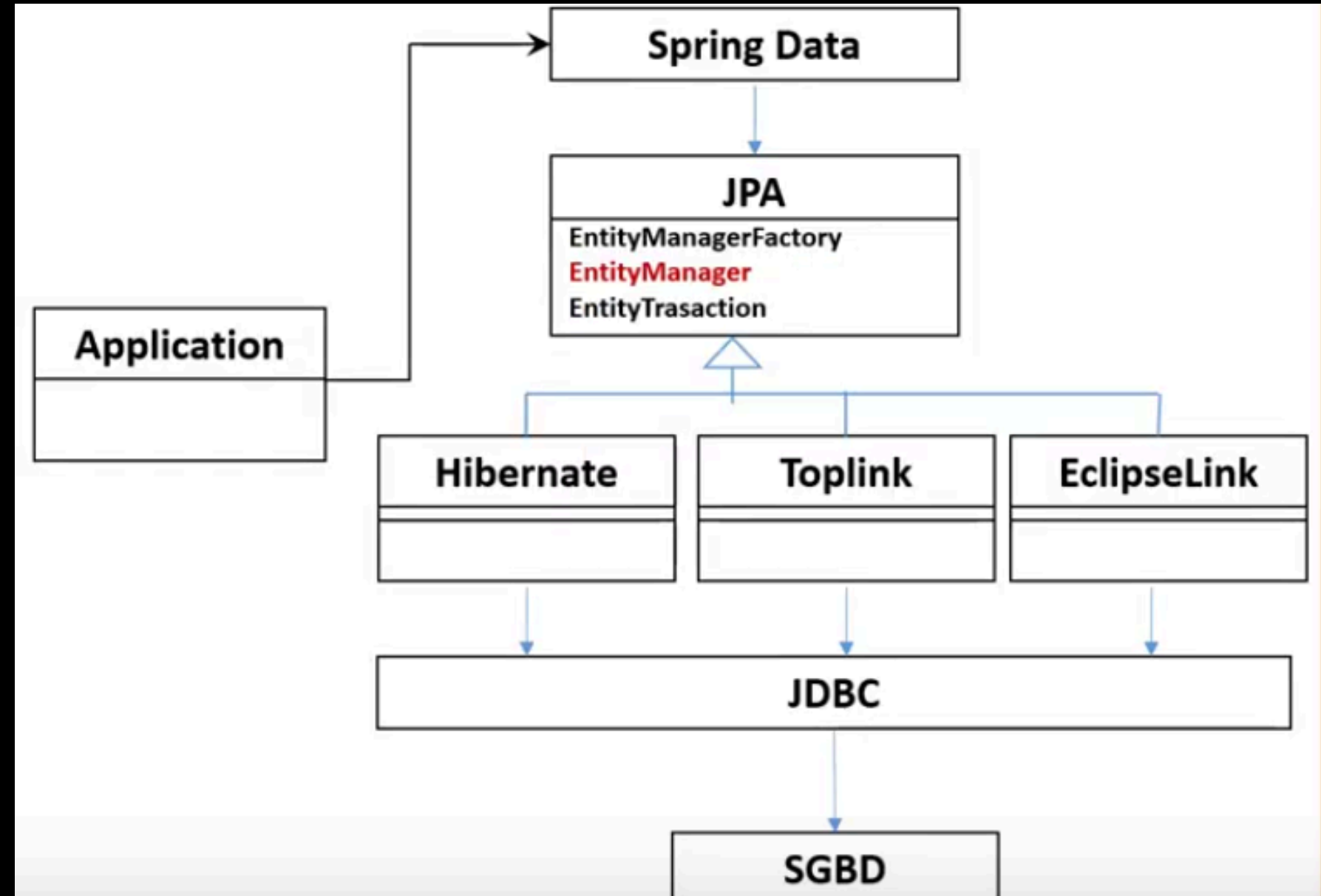
Ajout d'une nouvelle couche à l'architecture précédente qui va simplifier plus le développement des couches d'accès aux données



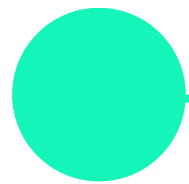
Spring Data fournit une abstraction au dessus des technologies de persistance de données : gestion de la persistance des entités de manière automatique

Intégration de Spring Data avec JPA

Nouvelle Architecture



Intégration de Spring Data avec JPA



Définition & fonctionnalités

Spring Data est un projet Spring qui a pour objectif de simplifier l'interaction avec différents systèmes de stockage de données : qu'il s'agisse d'une base de données relationnelle, d'une base de données NoSQL, d'un système Big Data ou encore d'une API Web.

génération automatique de requêtes, la pagination, le tri, et la gestion des transactions, tout en favorisant l'intégration transparente avec une variété de technologies de persistance de données.

Gestion de la persistance avec Spring Data et JPA Hibernate

Dans un projet Maven, pour utiliser Spring Data pour une base de données relationnelles avec JPA, il faut déclarer la dépendance suivante dans le fichier **pom.xml**

```
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-jpa</artifactId>
  <version>2.2.2.RELEASE</version>
</dependency>
```


Gestion de la persistance avec Spring Data et JPA Hibernate

Le fichier **persistenc.xml** sera remplacé par **application.properties**

Spring Data s'organise autour de la notion de repository et fournissent des interfaces génériques qui offrent un ensemble de méthodes pour interagir avec une base de données. Pour une intégration de Spring Data avec JPA, il existe l'interface JpaRepository<T, ID>

Gestion de la persistance avec Spring Data

```
package org.sid.dao; import java.util.List;
import org.sid.entities.Personne;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface PersonneRepository extends JpaRepository<Personne,Long>{
    /*hérites de toutes les méthodes pour gérer les entités*/
    /*Exemple d'ajout de méthodes */
    public List<Personne> findByDesignation(Integer age);
    public List<Personne> findByDesignationContains(String name);
    /*Pour les méthodes plus complexe utilisé des requetes HQL */
}
```

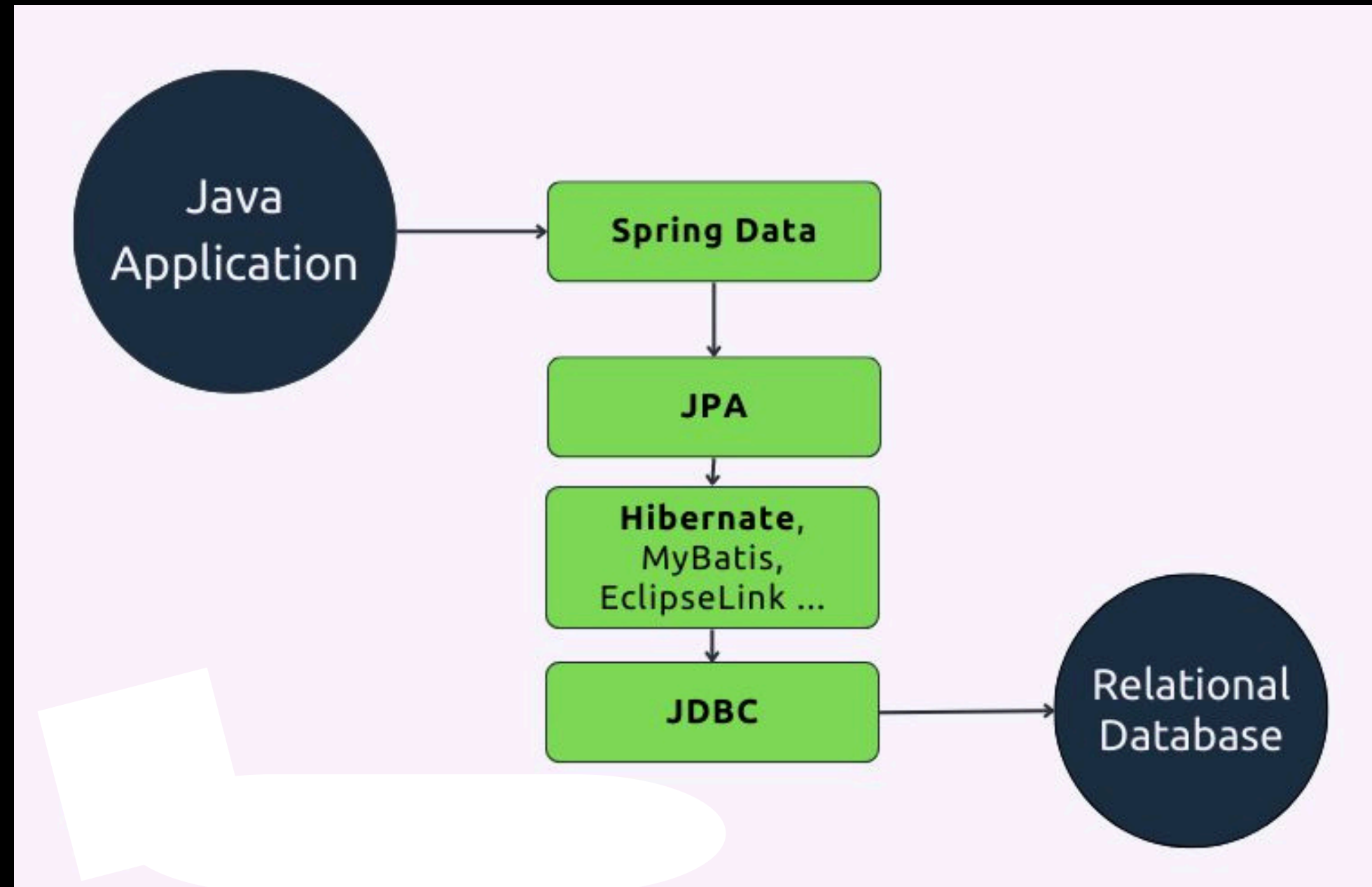
Ajouter des méthodes au besoin juste en les déclarant Sauf si la méthode est relativement complexe



Partie Questions

- C'est quoi un Objet Persistant ?
- Donner deux avantages d'utilisations des Frameworks de ORM plutôt que d'utiliser directement l'API JDBC ?
- C'est quoi la différence fondamentale entre JPA et Hibernate ?
- C'est quoi la difference entre la Persistence Et le mapping?

Conclusion



**Merci pour votre
Attention**

