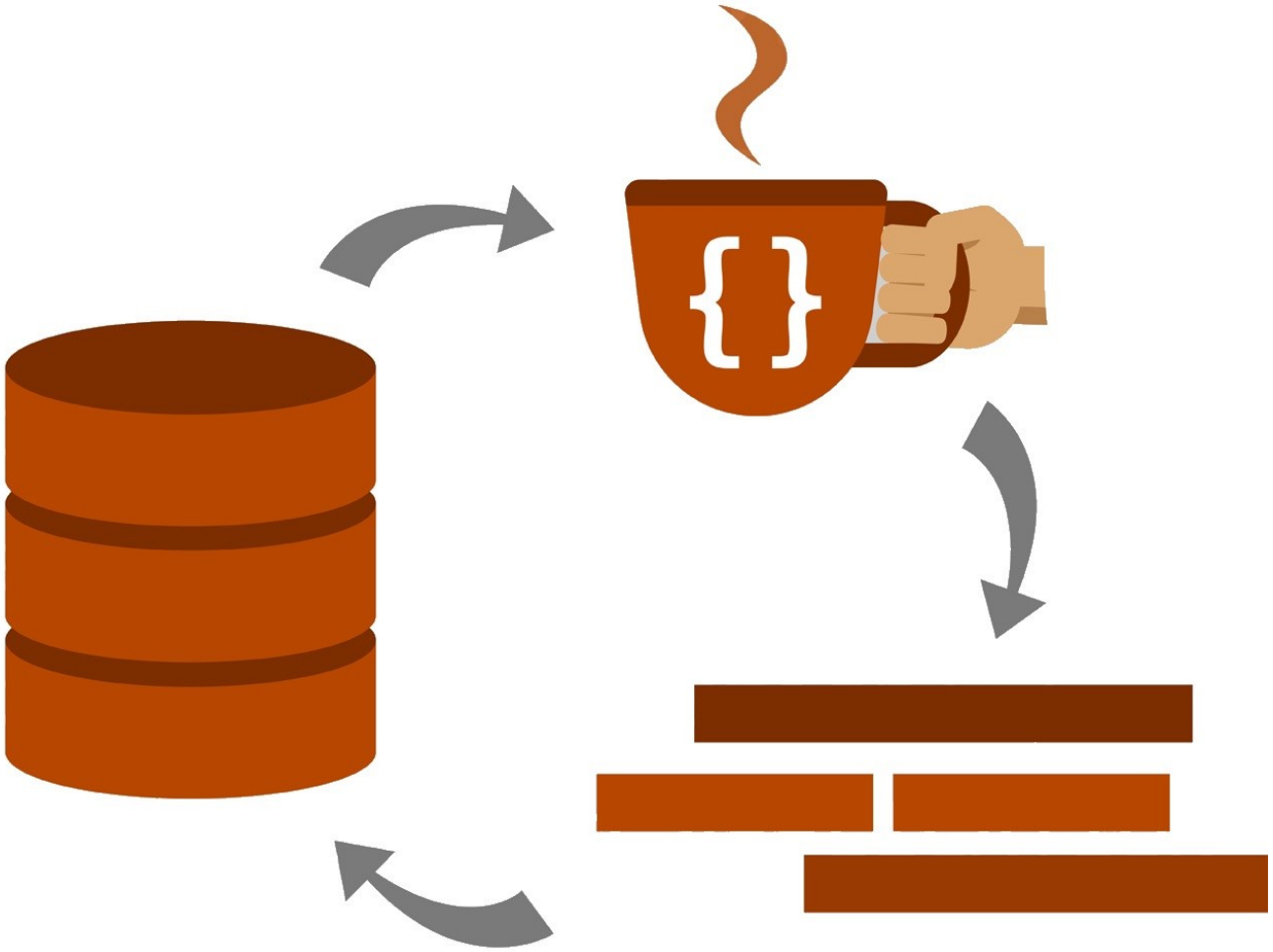
IUT Paris Descartes

2018-2019

Rapport Technologique

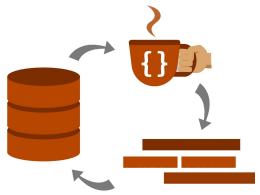
JPA et les ORM



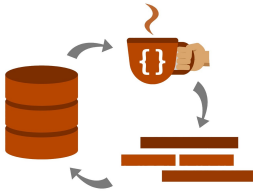
Thomas TURBIDE - Aroun RADJAVELOU - Christian JACQUOT - Yohan ALVES - Arsène LAPOSTOLET

Groupe 205

Sommaire



I/ Introduction



Qu'est ce que JPA ?

JPA pour Java Persistence API est une norme intégrée à JavaEE qui propose une abstraction pour la création d'ORM en Java. L'API est packagée dans *javax.persistence*.

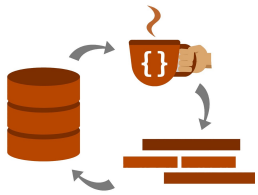
Persistance de donnée

La persistance de donnée désigne tous les mécanismes qui permettent à un programme de stocker des données qui ne seront pas perdues à l'issue de son exécution. On peut citer comme exemple le stockage des informations dans un fichier ou dans une base de données.

Object Relational Mapping (ORM)

Le mappage objet-relationnel est un mécanisme qui permet de stocker des objets instanciés par un programme directement dans une base de données sans passer par des requêtes SQL. L'utilisation d'un ORM dans un projet permet de gagner beaucoup de temps sur la partie Modèle en ne codant pas des requêtes JDBC, code souvent redondant.

II/ Implémentations



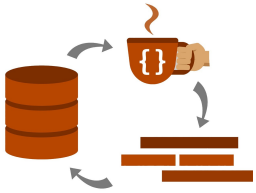
EclipseLink est une implémentation Open-Source de JPA. Il est développé par la fondation Eclipse. Il permet de stocker les objets Java dans des bases de données relationnelles et des documents XML.

Oracle TopLink est une implémentation professionnelle de JPA. Il est très puissant et propose des binding JSON, XML, le mapping avec des bases de données relationnelles et objets, ainsi que des outils de mise à l'échelle via la plateforme Oracle WebLogic.

Hibernate est une autre implémentation professionnelle de JPA. Elle est développée par RedHat et très adaptable. Hibernate offre des méthodes de haut niveau pour le mapping objet relationnel. Il permet donc de gagner beaucoup de temps sur le développement mais présente une capacité de mise à l'échelle moyenne.

OpenJPA est une autre implémentation Open Source de JPA. Il est développé par The Apache Foundation et propose le mapping avec des bases de données relationnelles et objets.

III/ Principe clé



Les Annotations

Dans le langage Java, et ce, depuis le Java SE 1.5, une annotation est une forme de métadonnée syntaxique qui peut être ajoutée au code source. Les classes, les variables, les paramètres et les packages peuvent être annotés. Les annotations sont compilées par le compilateur JAVAC. Les annotations les plus connues sont : *@Override*, *@Deprecated*. Il est possible de développer ses propres annotations.

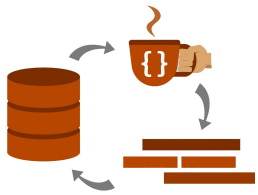
L'API Entity

JPA utilise l'API Entity pour mapper les objets avec la base. En identifiant les champs des objets java par des champs de la base de donnée relationnelle.

```
@Entity
@Table(name = "personnes")
public class Personne implements Serializable {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private Integer id;
	@Column(length = 32)
	private String nom;

	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getNom() {
		return nom;
	}
	public void setNom(String nom) {
		this.nom = nom;
	}
}
```

IV/ Mise en place



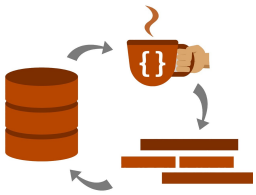
Le fichier persistence.xml

Il est tout d'abord nécessaire de configurer la connexion à la base de données et l'initialisation de l'ORM. Cela se fait via le fichier *persistence.xml.* On y précise l'adresse de connexion à la base, les IDs de connexion, le Driver JDBC. La configuration peut varier selon l'implémentation de JPA, le SGBD et le Driver. Voici un exemple avec TopLink et MySQL.

```
<?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="jpatest">
            <properties>
                  <property name="topLink.jdbc.url" value="jdbc:mysql://      local-
host:3306/jpatest"/>
                  <property name="topLink.jdbc.user" value="root"/>
                  <property name="topLink.jdbc.driver" value="com.mysql.jdbc.Driver"/>
                  <property name="topLink.jdbc.password" value=""/>
                  <property name="topLink.ddl-generation" value="drop-and-create-
tables"/>
                  <property name="topLink.ddl-generation.output-mode" va-
lue="database">
                  <property name="topLink.logging.level" value="FINE"/>
            </properties>
      </persistence-unit>
</persistence>
```

Entity Manager

Ensuite, il suffit de développer les DAO en utilisant la classe *EntityManager* qui permet, grâce à ses méthodes *persist(), find()* et *remove(),* de réaliser les opérations CRUD sur des objets annotés *@Entity* et dont les attributs sont configurés. Cela permet de réduire considérablement la quantité de code dans les DAO.



Voici un exemple de DAO :

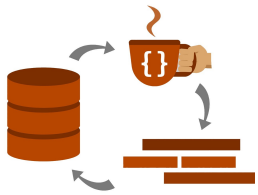
```
@Statefull
public class PersonneDao {
    private static final String JPQL_SELECT_PAR_NOM = "SELECT p FROM Personnes p WHERE
p.nom=:nom";
    private static final String PARAM_EMAIL           = "nom";

    // Injection du manager, qui s'occupe de la connexion avec la BDD
    @PersistenceContext( unitName = "jpatest" )
    private EntityManager em;

    // Enregistrement d'une nouvelle personne
    public void creer( Personne p ) throws DAOException {
        try {
            em.persist( p );
        } catch ( Exception e ) {
            throw new DAOException( e );
        }
    }

    // Recherche d'une personne à partir de son nom
    public Utilisateur trouver( String nom ) throws DAOException {
        Personne p = null;
        Query requete = em.createQuery( JPQL_SELECT_PAR_NOM );
        requete.setParameter( PARAM_EMAIL, nom );
        try {
            p = (Personne) requete.getSingleResult();
        } catch ( NoResultException e ) {
            return null;
        } catch ( Exception e ) {
            throw new DAOException( e );
        }
        return p;
    }
}
```

V/ Vers les frameworks



Le suite logique pour la programmation d'applications professionnelles JEE est l'utilisation des frameworks MVC d'entreprise. Ce sont des outils complexes et puissants qui prennent en charge de nombreuses fonctionnalités, comme un ORM, ou encore l'injection de dépendances.

spring

JavaServer™ Faces JSF

STRUTS

GWT

vaadin