La persistance des données dans les applications : DAO, JPA, Hibernate...

Plan

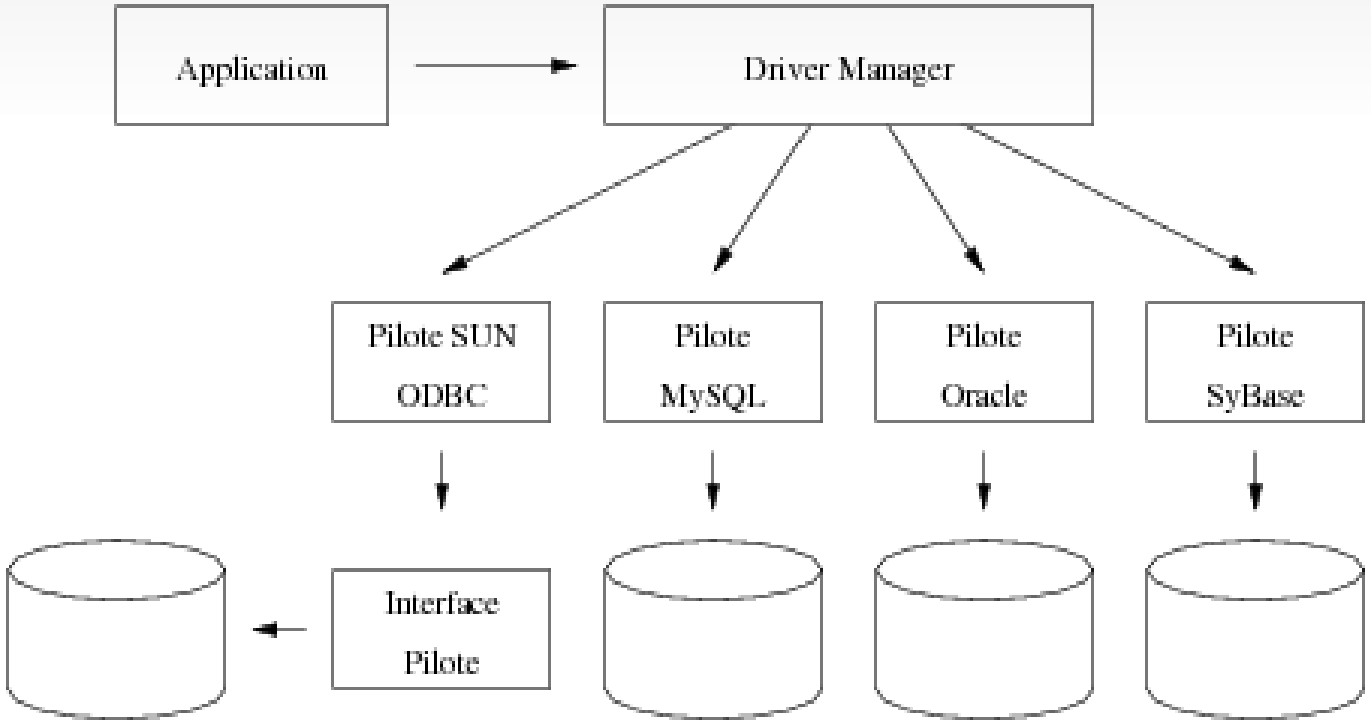
1. Généralités sur la persistance des données dans les applications
2. La connection des applications aux Bds (JDBC/ODBC)
3. La problématique du mapping objet/relationnel
4. Le patron de conception DAO
5. Les ORM : des cadres applicatifs pour automatiser le mapping objet/relationnel (JPA, Hibernate)
6. Integration dans Spring
7. Exemple d'application, de projet utilisant cette mise en oeuvre ORI/OAI, ESUP
8. Hibernate avancé: création et manipulation de schéma
9. Conclusion

1. Généralités sur la persistance des données dans les applications

- Lancement, exécution et arrêt d'une application
- Stockage des données, des résultats dans des fichiers, dans des bases de données
- Quels intérets des BDs (structuration, gestion, rapidité, ...)?
- Meilleure gestion du cycle de vie des données
- Intégrité référentielle, cohérence par le transactionnel
- Architecture distribuée, réseau
- Exploitation, maintenance

2. La connection des applications aux BDs

- Via ODBC (Open DataBase Connectivity)
- Via JDBC (Java DataBase Connectivity)
- Indépendance du code applicative vis à vis du moteur



2. La connection des applications aux BDs

Déclaration du pilote JDBC

```
import java.sql.*;

public class ExempleJDBC
{
public static void main(String[] Args)
   {
   try {
      Class.forName("org.gjt.mm.mysql.Driver");
      }
   catch (Exception E) {
      System.err.println("Pas de pilote !");
      }
   ... connexion et utilisation de la base ...
   }
}
```

Connexion à la base de données

```
try {
   String url = "jdbc:mysql://localhost/dbessai";
   Connection conn =
      DriverManager.getConnection(url, "user",
      "password");

   ... utilisation de la base ...
   }
catch (SQLException E) {
   System.err.println(E.getMessage()); }
```

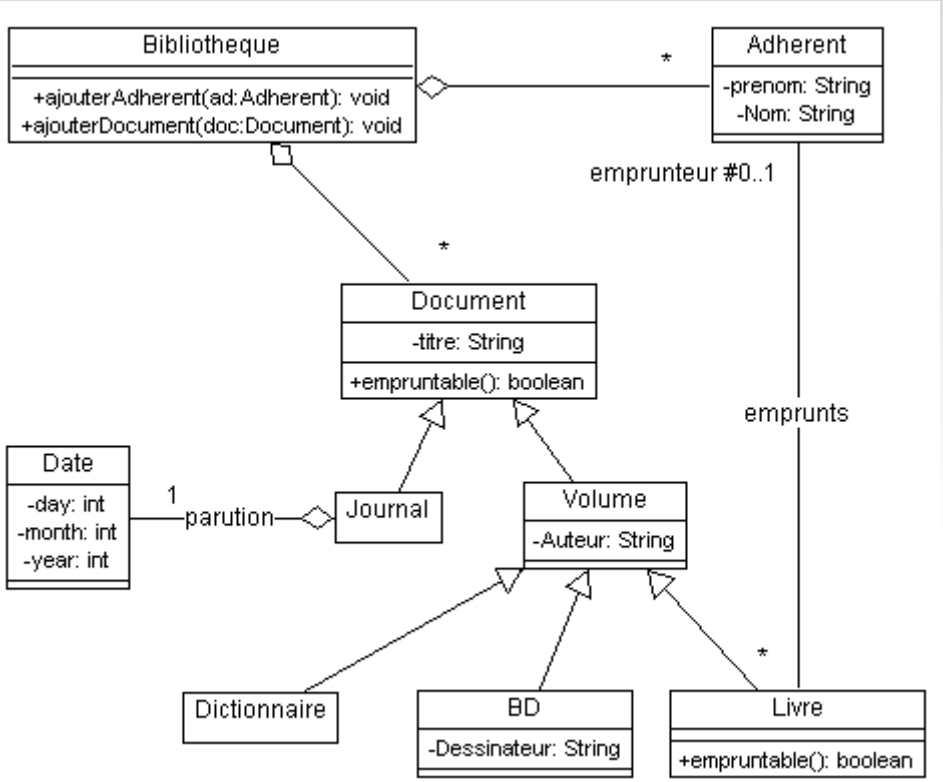
Les requêtes en JDBC

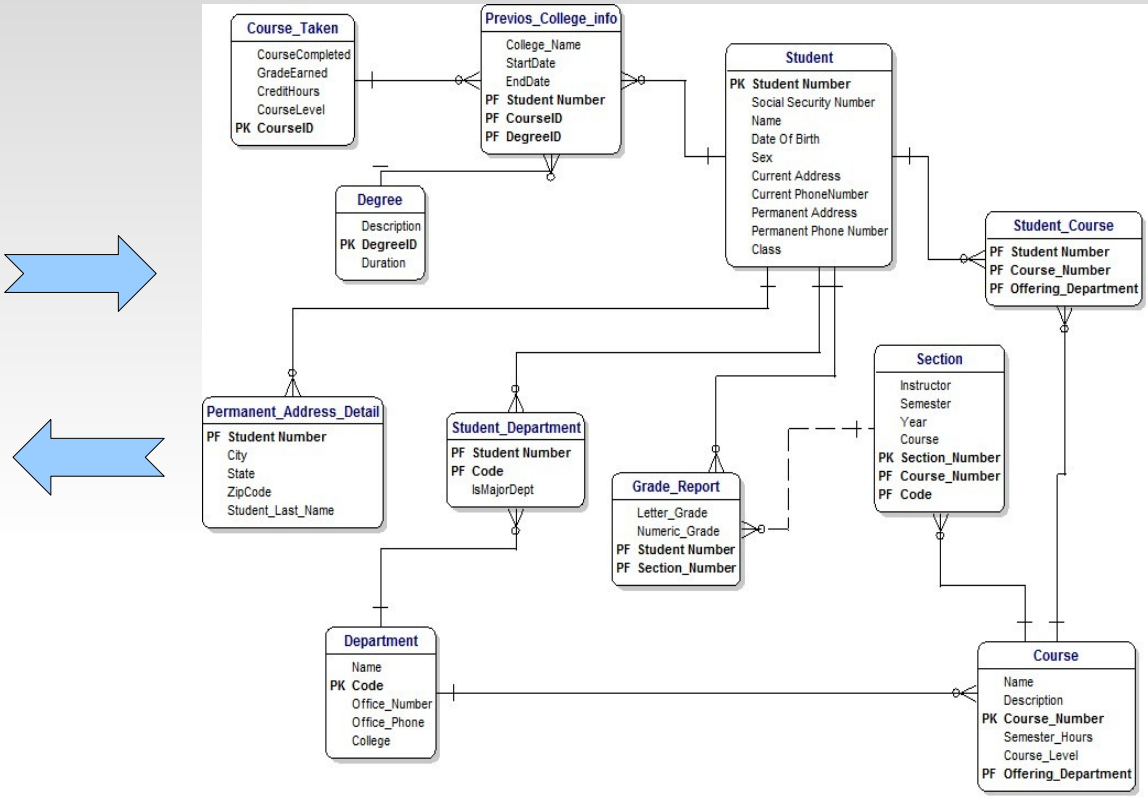
```
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery(
   "SELECT Nom,Prenom,Age FROM personne " +
   "ORDER BY age"
   );
while (rs.next()) {
   System.out.println("Nom    : " + rs.getString(1));
   System.out.println("Prénom : " + rs.getString(2));
   System.out.println("Age    : " + rs.getString(3));
   }
rs.close(); st.close(); conn.close(); // Fermeture
```

Insertion de lignes

```
Statement st = conn.createStatement();
int nb = st.executeUpdate(
   "INSERT INTO personne(Nom,Age) " +
   "VALUES ('" + nom + "', " + age + ")"
   );
System.out.println(nb + " ligne(s) insérée(s)");
st.close();
```

3. La problématique du mapping objet/relationnel





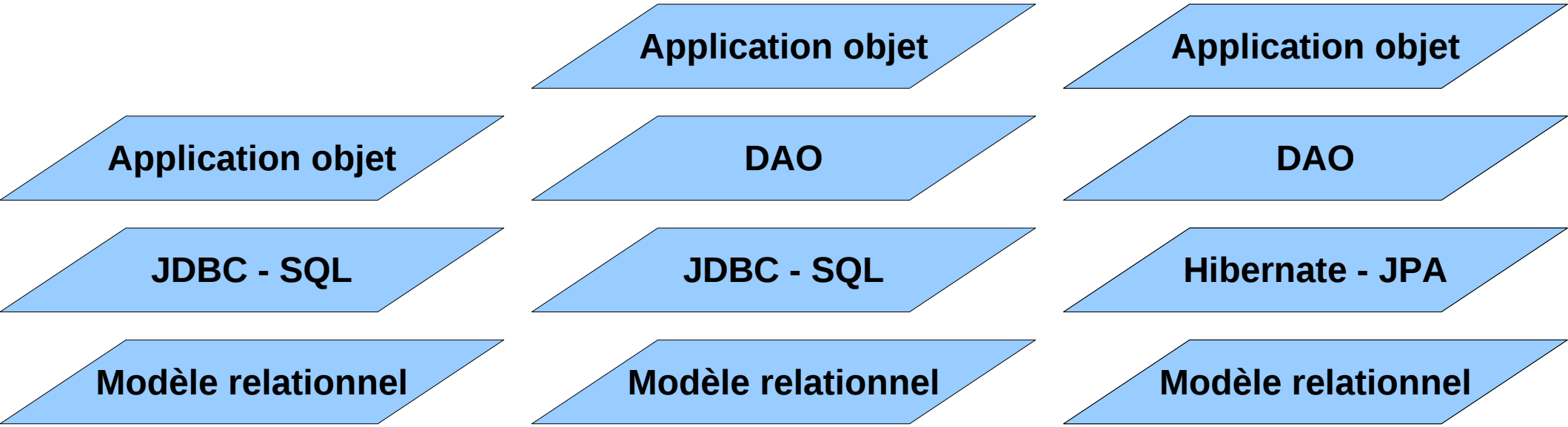
- Application: Modèle objet => diagramme de classe UML
- Base de données: Modèle relationnel => EER (Enhanced Entity Relationship)

4. Le patron de conception DAO

- DAO (Data Access Object): patron de conception pour définir une couche d'accès au données (CRUD)
- Classe metier Livre → DAO LivreDAO.

5. Les ORM (Object Relational Mapping) : des cadres applicatifs pour faciliter le mapping objet/relationnel

- 3 modèles architecturaux: sans DAO, avec DAO, avec DAO + ORM



- JPA, Java Persistence API
  - spécification ORM et volonté de standardisation des concepts qui font le succès d'Hibernate
- Hibernate : l'ORM à succès
  - Intégration dans le conteneur léger Spring
  - Mapping par fichier XML ou annotations Java 1.5

5. Deux formes de mapping

- Hibernate : Mapping par fichier XML

```
<class name="person" table="PERSON">
  <id name="id" column="PERSON_ID">
    <generator class="foreign">
      <param
 name="property">employee</param>
    </generator>
  </id>
  <one-to-one name="employee"
    class="Employee"
    constrained="true"/>
</class>
```

- JPA/EJB3 : Mapping par annotations Java

```
@Entity
public class Flight implements
   Serializable {
   Long id;
   @Id
   public Long getId() { return
   id; }
   public void setId(Long id)
   { this.id = id; }
}
```

- JDBC + Hibernate + Spring
  - Définition d'une source de donnée :

```
<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName">
        <value>${hibernate.connection.driver_class}</value>
    </property>
    <property name="url">
        <value>${hibernate.connection.url}</value>
    </property>
    <property name="username">
        <value>${hibernate.connection.username}</value>
    </property>
    <property name="password">
        <value>${hibernate.connection.password}</value>
    </property>
    <property name="connectionProperties">
        <props>
            <prop key="hibernate.dialect">${hibernate.properties.dialect}</prop>
        </props>
    </property>
</bean>
```

7. Exemple d'applications, de projets utilisant cette mise en oeuvre (Hibernate/Spring)

- Portail ESUP (http://www.esup-portail.org)
- ORI-OAI(http://www.ori-oai.org/display/ORIOAI/ORI-OAI.ORG)
- CEPIA(https://listes.cru.fr/wiki/cepia-dev/)

8. Hibernate avancé: création et manipulation de schéma

- Création automatique du schéma
- Mise à jour du schéma (changement de struture)
- Facilités pour les migrations de données

```
<bean
    id="createSessionFactory"
    parent="abstractHibernateSessionFactory"
    lazy-init="true"
    >
    <description>
        This bean is used to create the database structures.
        Caution: leave inherited attribute lazy-init to true or the database will
        be re-created from scratch each time the application starts.
    </description>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.hbm2ddl.auto">create</prop>
        </props>
    </property>
</bean>
```

9. Conclusion

- Couches d'abstraction :
  - Connection : JDBC/ODBC
  - Couche de persistance (DAO)
  - Conteneurs légers : Spring /EJB3
  - Relationnel – Objet (ORM) : JPA/Hibernate

Références

- Une introduction à JDBC: http://www.dil.univ-mrs.fr/~massat/ens/jee/jdbc.html
- JPA(wikipedia)
- Tutoriel JPA(http://tahe.developpez.com/java/jpa/)
- CRUD avec Spring, JSF et Hibernate(http://beuve.developpez.com/tutoriel/j2ee/Spring/CRUD/)
- Hibernate - Référence en français