



## LẬP TRÌNH JPA CƠ BẢN

---

### J4 PROGRAMMING – SERVLET/JSP & JPA

**❑ Phần 1: Lập trình CSDL và JPA cơ bản**

- ❖ Xác định hạn chế của lập trình JDBC
- ❖ Mô hình và kiến trúc công nghệ JPA
- ❖ Giới thiệu CSDL mẫu
- ❖ Cấu hình thư viện cần thiết
- ❖ Xây dựng lớp thực thể
- ❖ Lập trình JPA truy vấn và thao tác dữ liệu cơ bản

**❑ Phần 2: Xây dựng ứng dụng CRUD**

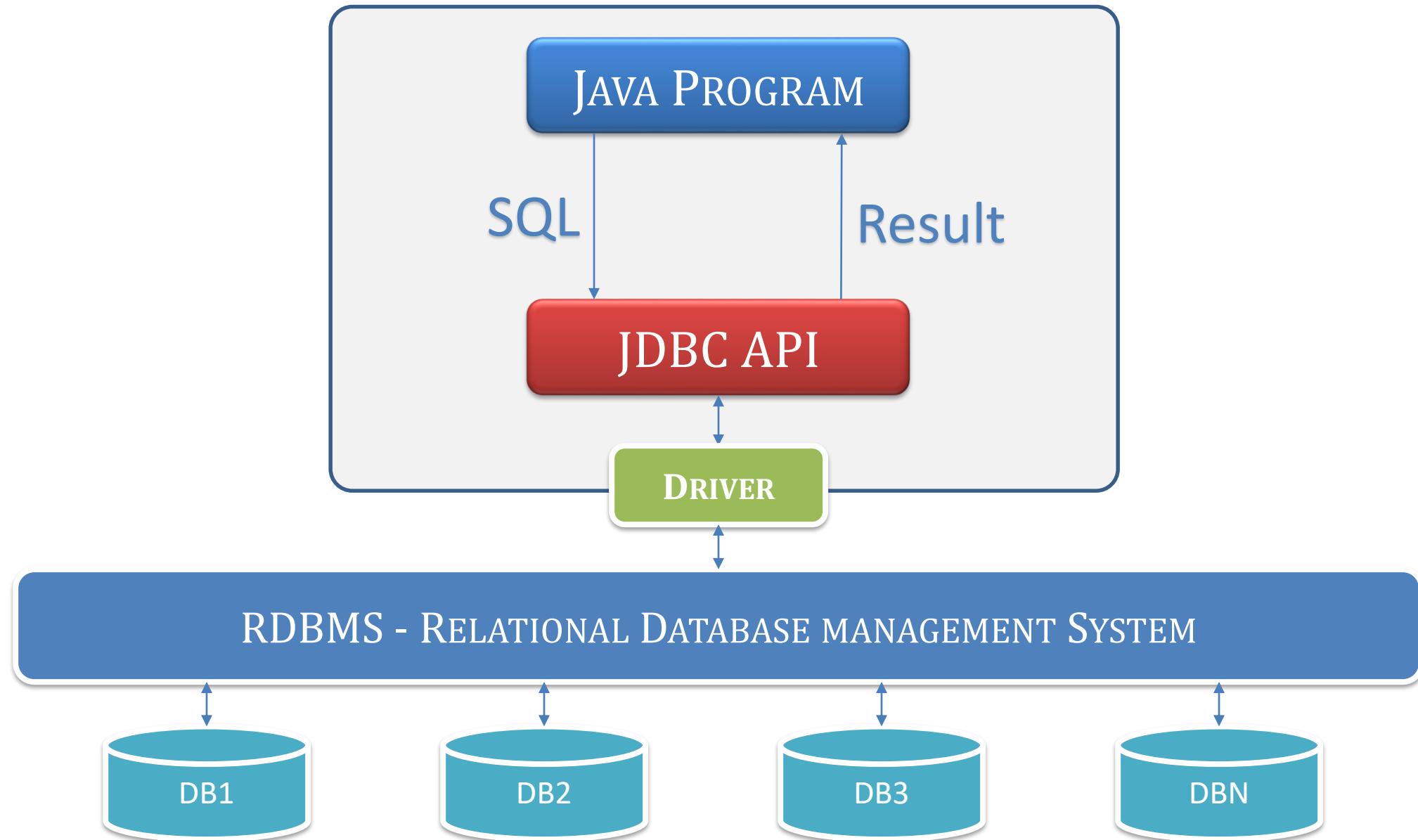
- ❖ Phác thảo giao diện và mô hình công nghệ
- ❖ Xây dựng thư viện tiện ích và DAO
- ❖ Xây dựng Servlet
- ❖ Thiết kế giao diện
- ❖ Hoàn thiện các chức năng





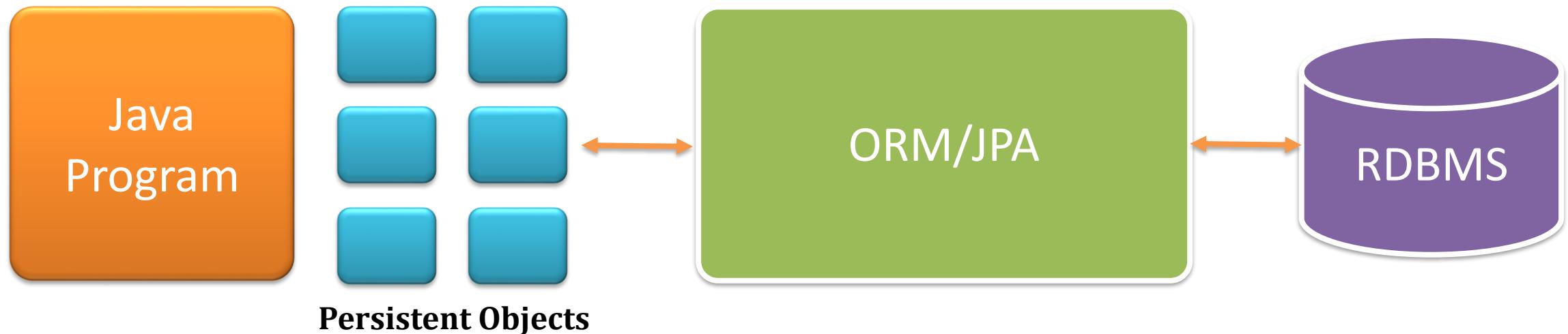
# ① LẬP TRÌNH CSDL VÀ JPA CƠ BẢN

# MÔ HÌNH ỨNG DỤNG JDBC

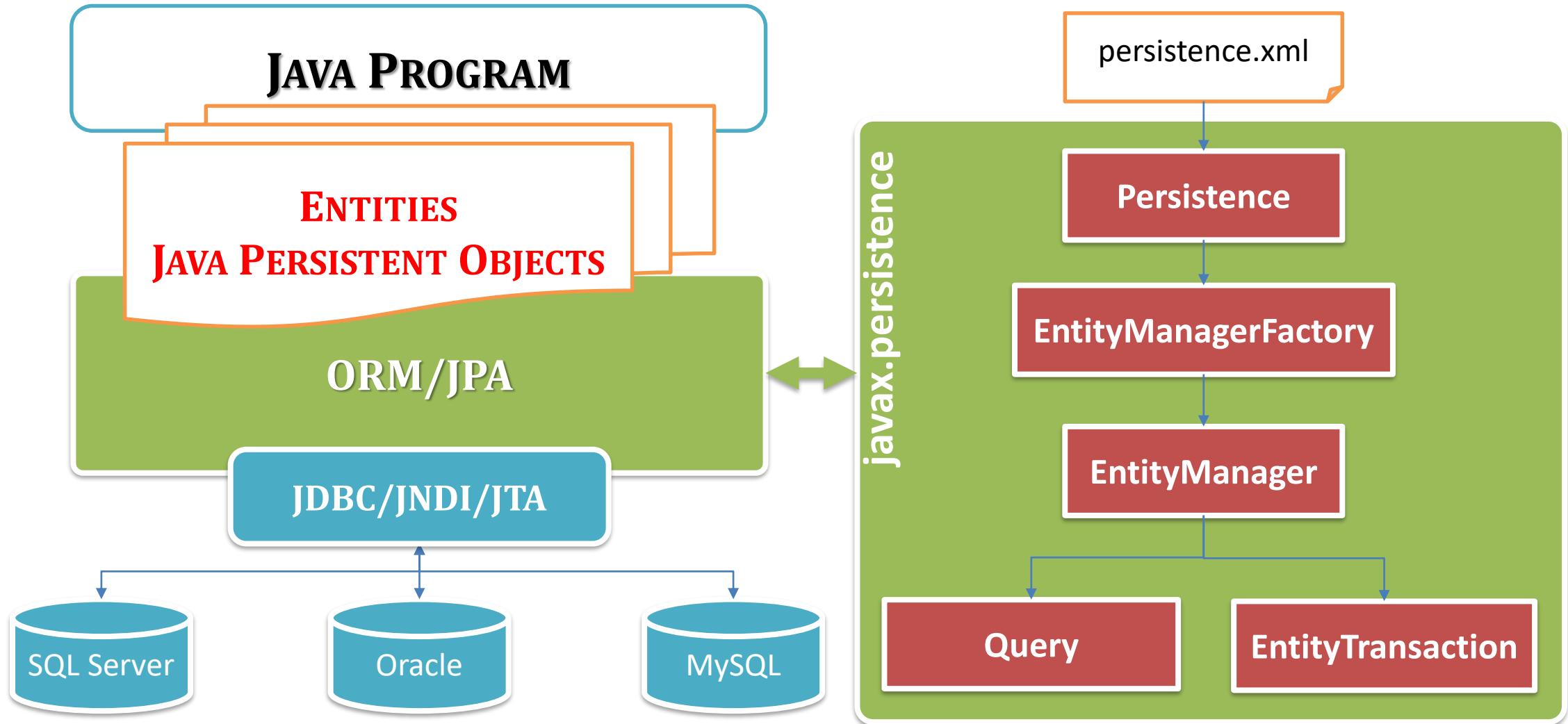


- ❑ Viết nhiều câu lệnh SQL => tốn nhiều thời gian, dễ vấp sai sót.
- ❑ Viết quá nhiều mã java cho việc truy vấn và thao tác dữ liệu (vì phải chuyển đổi từ đối tượng thành SQL và ngược lại từ ResultSet thành Collection)
- ❑ Khá khó khăn trong việc điều khiển Transaction
- ❑ Nâng cấp khó khăn vì SQL phụ thuộc vào hệ quản trị CSDL, nếu thay đổi hệ quản trị CSDL thì phải viết lại mã
- ❑ Không được hỗ trợ bởi các dịch vụ nền (Connection Pool, Instance Pool, Transaction...)

# MÔ HÌNH ỨNG DỤNG JPA



- ❑ **Java Program** chỉ làm việc với các đối tượng (trong suốt SQL)
- ❑ **ORM** (Object Relational Mapping) ánh xạ các đối tượng với các bản ghi trong CSDL quan hệ thông qua các lớp thực thể (Entity Class)
- ❑ **JPA** (Java Persistence API) thực hiện các công việc chuyển đổi các đối tượng sang SQL (trên cơ sở ORM) để thao tác, truy vấn dữ liệu và ngược lại một cách tự động



### ❑ File **persistence.xml**

- ❖ File cấu hình khai báo CSDL

### ❑ Persistence

- ❖ Nạp persistence.xml từ đó tạo ra đối tượng EntityManagerFactory

### ❑ EntityManagerFactory

- ❖ Đối tượng này cho phép tạo ra EntityManager để bắt đầu lập trình truy vấn và thao tác dữ liệu.

### ❑ EntityManager

- ❖ Cho phép thao tác (thêm, sửa, xóa) và truy vấn 1 thực thể

### ❑ EntityTransaction

- ❖ Được tạo ra từ EntityManager để điều khiển transaction các thao tác dữ liệu (thêm, sửa, xóa)

### ❑ Query/TypedQuery<T>

- ❖ Được tạo ra từ EntityManager để truy vấn dữ liệu sử dụng câu lệnh JPQL (Java Persistence Query Language)

# Khai báo thư viện cần thiết - POM.XML

```

<!-- Hibernate/JPA -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.12.Final</version>
</dependency>
<!-- SQL Server Driver -->
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <version>8.2.2.jre8</version>
</dependency>

```

- JPA chỉ là bản đặc tả và có khá nhiều nhà cung cấp thư viện thực hiện đúng bản đặc tả này, Hibernate là một trong số đó
- Bạn cũng cần khai báo JDBC Driver làm việc với hệ quản trị CSDL

## Libraries

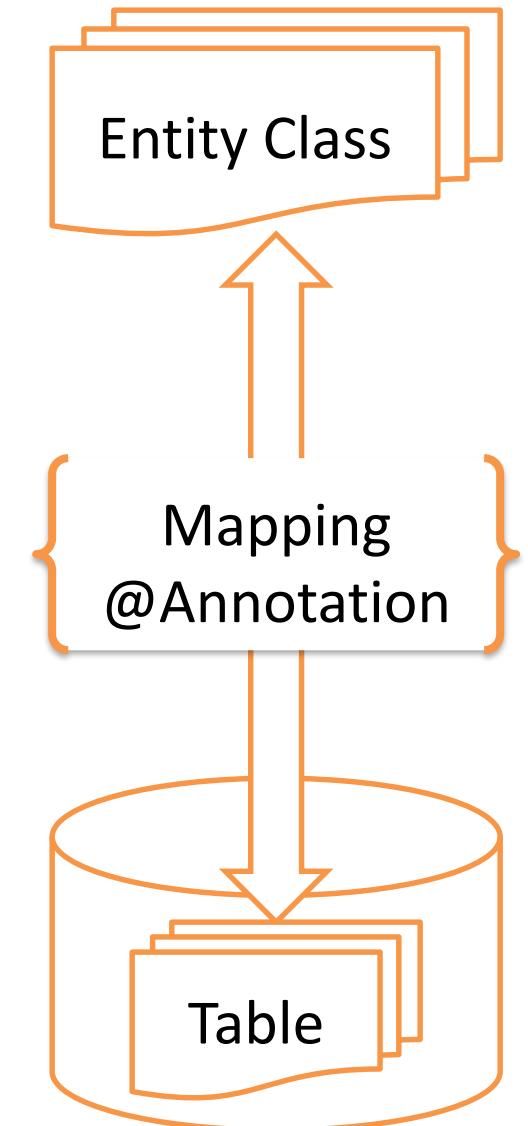
- > Apache Tomcat v8.5 [Apache Tomcat v8.5]
- > JRE System Library [JavaSE-1.8]
- > Maven Dependencies
  - > hibernate-core-5.2.12.Final.jar - C:\Users\
  - > jboss-logging-3.3.0.Final.jar - C:\Users\A\
  - > hibernate-jpa-2.1-api-1.0.0.Final.jar - C:\U\
  - > javassist-3.20.0-GA.jar - C:\Users\Admin\
  - > antlr-2.7.7.jar - C:\Users\Admin\.m2\rep\
  - > jboss-transaction-api\_1.2\_spec-1.0.1.Final
  - > jandex-2.0.3.Final.jar - C:\Users\Admin\.r\
  - > classmate-1.3.0.jar - C:\Users\Admin\.m2\
  - > dom4j-1.6.1.jar - C:\Users\Admin\.m2\re\
  - > hibernate-commons-annotations-5.0.1.Fi
  - > mssql-jdbc-8.2.2.jre8.jar - C:\Users\Admin\

- ❑ Ánh xạ đối tượng quan hệ là công việc tạo các lớp thực thể (Entity Class) mô tả cấu trúc dữ liệu của các bảng trong CSDL quan hệ.

- ❖ Entity Class  $\Leftrightarrow$  Table
- ❖ Field  $\Leftrightarrow$  Column
- ❖ Association  $\Leftrightarrow$  Relationship
- ❖ Constraints...

- ❑ Entity Class phải tuân theo qui ước của JavaBean

- ❖ Public class
- ❖ Constructor mặc định (không tham số)
- ❖ Getters/Setters





- + Database Diagrams
- Tables
  - + System Tables
  - + FileTables
  - + **dbo.Users**

THANHVINHCENTER.PolyOE - dbo.Users

Column Name	Data Type	Allow Nulls
<b>Id</b>	nvarchar(20)	<input type="checkbox"/>
Password	nvarchar(50)	<input type="checkbox"/>
Fullname	nvarchar(50)	<input type="checkbox"/>
Email	nvarchar(50)	<input type="checkbox"/>
Admin	bit	<input type="checkbox"/>

```

@Entity
@Table(name="Users")
User
@Id
@Column(name="Id")
id: String
@Column(name="Password")
password: String
@Column(name="Fullname")
fullname: String
@Column(name="Email")
email: String
@Column(name="Admin")
admin: Boolean
  
```

```
@Entity  
@Table(name = "Users")  
public class User {  
    @Id  
    @Column(name = "id")  
    String id;  
    @Column(name = "password")  
    String password;  
    @Column(name = "fullname")  
    String fullname;  
    @Column(name = "email")  
    String email;  
    @Column(name = "admin")  
    Boolean admin = false;  
  
    getters/setters  
}
```

❑ Các @Annotation ánh xạ đã sử dụng

❖ **@Entity**

➢ Đây là lớp thực thể (Entity Class)

❖ **@Table**

➢ Lớp thực thể ánh xạ với bảng

❖ **@Column**

➢ Trường ánh xạ với cột

❖ **@Id**

➢ Trường ánh xạ với cột khóa chính

❑ Các @Annotation phải đặt ngay trên các class, field, method được ánh xạ

❑ Phải có getters và setters cho tất cả các trường

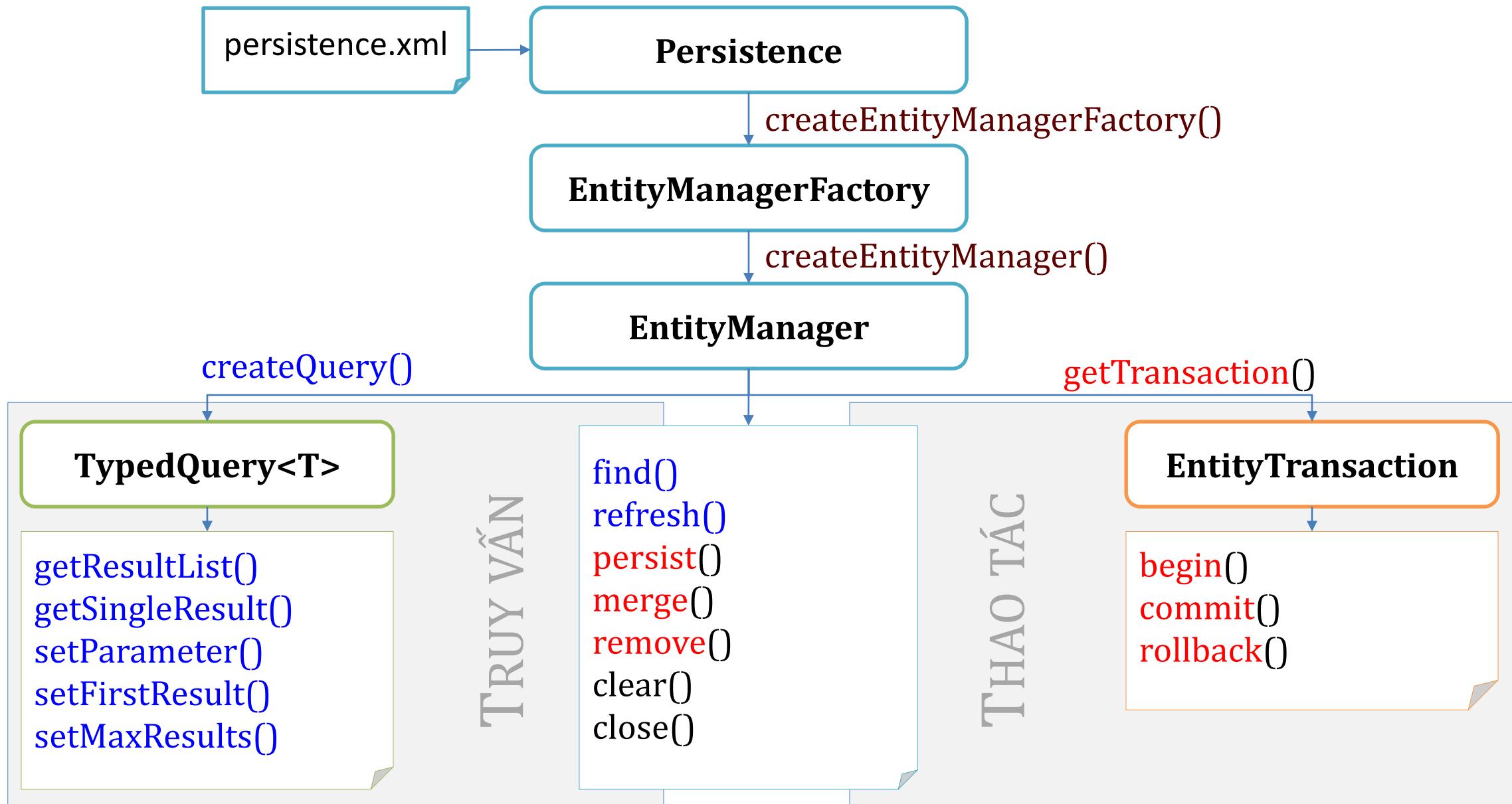
## SỬ DỤNG KIỂU DỮ LIỆU

Java type	ANSI SQL Type
int or java.lang.Integer	INTEGER
long or java.lang.Long	BIGINT
short or java.lang.Short	SMALLINT
double or java.lang.Double	FLOAT
java.math.BigDecimal	NUMERIC
char or java.lang.Character	CHAR(1)
java.lang.String	VARCHAR
byte or java.lang.Byte	TINYINT
boolean or java.lang.Boolean	BIT
byte[]	VARBINARY (or BLOB)

- ❑ Kiểu của Java phải tương thích với kiểu của CSDL
- ❑ Chú ý:
  - ❖ Kiểu nguyên thủy không cho phép null, kiểu lớp bao cho phép null
  - ❖ Kiểu nguyên thủy luôn luôn khởi tạo với giá trị zero-based, kiểu lớp bao luôn luôn khởi tạo null
  - ❖ Vì vậy cột cho phép null thì field phải sử dụng lớp bao

## ☐ src/META-INF/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="PolyOE">
        <properties>
            <property name="javax.persistence.jdbc.driver" value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:sqlserver://localhost;database=DB" />
            <property name="javax.persistence.jdbc.user" value="user" />
            <property name="javax.persistence.jdbc.password" value="password" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
        </properties>
    </persistence-unit>
</persistence>
```



// 1. Nạp persistence.xml và tạo EntityManagerFactory

```
EntityManagerFactory factory = Persistence.createEntityManagerFactory("PolyOE");
```

// 2. Tạo EntityManager chuẩn bị lập trình CSDL

```
EntityManager em = factory.createEntityManager();
```

```
<persistence-unit name="PolyOE">
```

// 3.1. Lập trình thao tác dữ liệu

```
em.getTransaction().begin();
try {
    em.persist(entity); // em.merge(entity); | em.remove(entity);
    em.getTransaction().commit();
} catch (Exception e) {
    em.getTransaction().rollback();
}
```

// 3.2 Lập trình truy vấn một thực thể theo khóa chính

```
User user = em.find(User.class, "NghiemN");
```

// 4. Đóng EntityManager và kết thúc lập trình CSDL

```
em.close();
```

Lập trình thao tác dữ liệu cần được điều khiển Transaction.

Transaction = “**None or All**” (được ăn cả, ngã về không)

```
EntityManagerFactory factory = Persistence.createEntityManagerFactory("PolyOE");
EntityManager em = factory.createEntityManager();
```

```
// Câu lệnh JPQL truy vấn thực thể
String jpql = "SELECT o FROM User o";
// Tạo Query/TypedQuery<T> để truy vấn
TypedQuery<User> query = em.createQuery(jpql, User.class);
// Truy vấn danh sách thực thể
List<User> list = query.getResultList();
// Truy vấn một thực thể (nếu JPQL đảm bảo chỉ có 1 thực thể)
User entity = query.getSingleResult();

em.close();
```

JPQL là câu lệnh truy vấn đối tượng (*các thành phần bên trong là Entity Class và Property chứ không phải là Table và Column*)

**TypedQuery<T>**

- getResultList(): List<T>
- getSingleResult(): T

Hãy căn cứ mệnh đề SELECT để xác định T

```
String jpql = "SELECT o FROM User o";
TypedQuery<User> query = em.createQuery(jpql, User.class);
query.setFirstResult(10); // vị trí bắt đầu
query.setMaxResults(8); // số lượng thực thể tối đa
List<User> list = query.getResultList();
```

**Phân trang**

- setFirstResult(int)
- setMaxResults(int)

```
String jpql = "SELECT o FROM User o WHERE o.admin=:role";
TypedQuery<User> query = em.createQuery(jpql, User.class);
query.setParameter("role", true); // cấp giá trị cho tham số :role
List<User> list = query.getResultList();
```

**Tham số**

- setParameter(String, Object)
- setParameter(int, Object)

```
String jpql = "SELECT o FROM User o WHERE o.admin=?0";
TypedQuery<User> query = em.createQuery(jpql, User.class);
query.setParameter(0, true); // cấp giá trị cho dấu ? đầu tiên
List<User> list = query.getResultList();
```



## ② XÂY DỰNG ỨNG DỤNG CRUD

---

# XÂY DỰNG ỨNG DỤNG QUẢN LÝ USER

DANH SÁCH	CẬP NHẬT					
ID	PASSWORD	FULLNAME	EMAIL	ROLE	ACTIVE?	
TeoNV	123456	Nguyễn Văn Tèo	teonv@gmail.com	admin	yes	<a href="#">Edit</a>

Action	Description
Initialize	Hiển thị lên bảng
Edit	Hiển thị user lên form
Create	Thêm user mới vào CSDL
Update	Cập nhật user đang xem
Delete	Xóa user đang xem
Reset	Xóa trống form

DANH SÁCH	CẬP NHẬT
-----------	----------

Username

Password

Fullname

Email Address

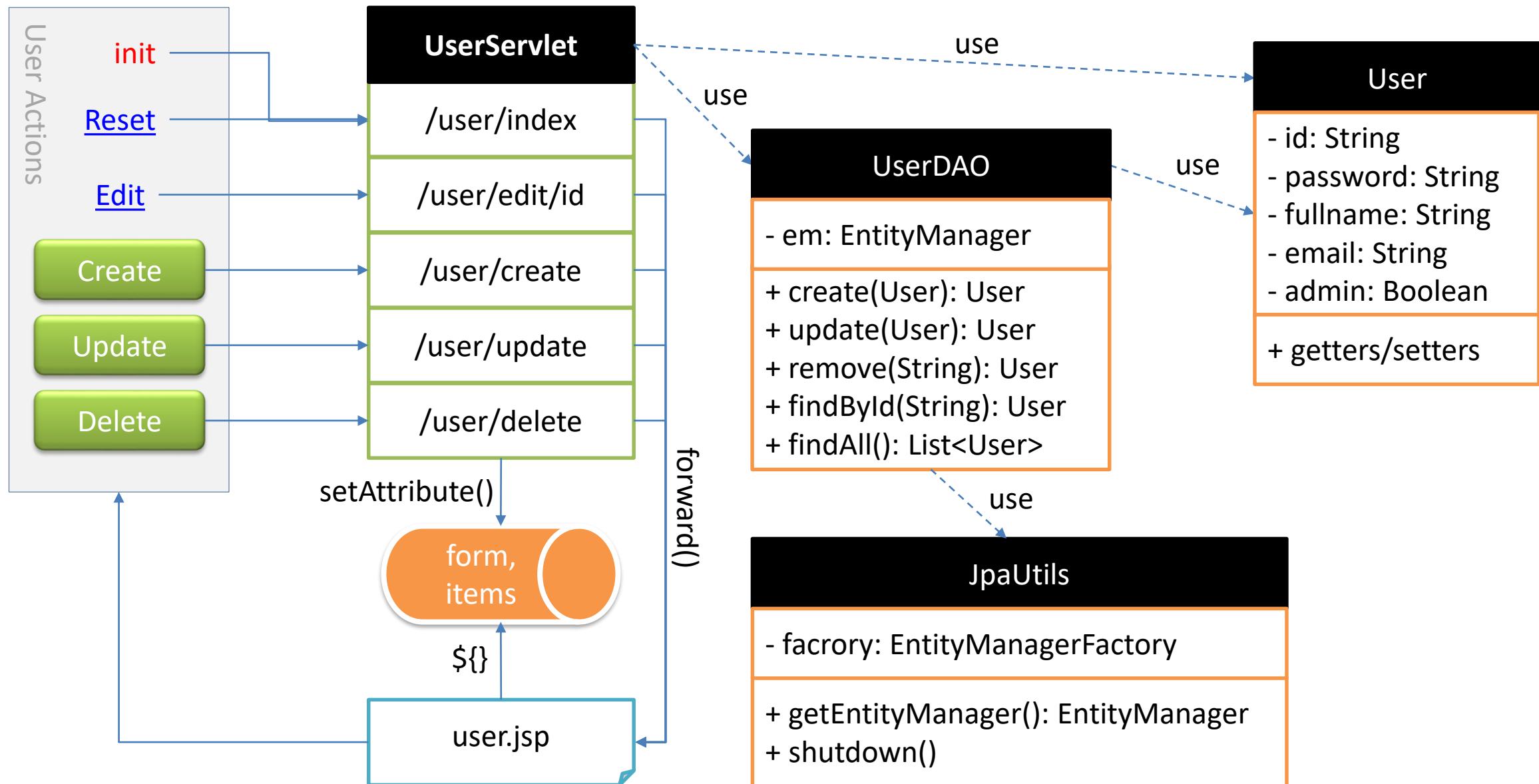
  

Role

 Admin  User      Active?  
 Yes  No

[Create](#) [Update](#) [Delete](#) [Reset](#)

# TỔ CHỨC ỨNG DỤNG QUẢN LÝ USER



```
public class JpaUtils {  
    private static EntityManagerFactory factory;  
    static public EntityManager getEntityManager(){  
        if(factory == null || !factory.isOpen()) {  
            factory = Persistence.createEntityManagerFactory("PolyOE");  
        }  
        return factory.createEntityManager();  
    }  
    static public void shutdown() {  
        if(factory != null && factory.isOpen()) {  
            factory.close();  
        }  
        factory = null;  
    }  
}
```

```
public class UserDAO {  
    private EntityManager em = JpaUtils.getEntityManager();  
    @Override  
    protected void finalize() throws Throwable {  
        em.close(); // đóng EntityManager khi DAO bị giải phóng  
        super.finalize();  
    }  
    public User create(User entity) {...}  
    public User update(User entity) {...}  
    public User remove(String id) {...}  
    public User findById(String id) {...}  
    public List<User> findAll() {...}  
}
```

```
public User findById(String id) {
    User entity = em.find(User.class, id);
    return entity;
}

public List<User> findAll() {
    String jpql = "SELECT o FROM User o";
    TypedQuery<User> query = em.createQuery(jpql, User.class);
    List<User> list = query.getResultList();
    return list;
}
```

```
try {  
    em.getTransaction().begin();  
    em.???(entity);  
    em.getTransaction().commit();  
    return entity;  
} catch (Exception e) {  
    em.getTransaction().rollback();  
    throw new RuntimeException(e);  
}
```

```
public User create(User entity) {  
    ???=persist ...  
}  
}  
  
public User update(User entity) {  
    ???=merge ...  
}  
}  
  
public User remove(String id) {  
    User entity = this.findById(id);  
    ???=remove ...  
}
```

# TỔ CHỨC CỦA USERSERVLET.JAVA

```
@WebServlet({  
    "/user/index", "/user/edit/*", "/user/create", "/user/update", "/user/delete"  
})  
public class UserServlet extends HttpServlet{  
    @Override  
    protected void service(HttpServletRequest req, HttpServletResponse resp)  
        throws ServletException, IOException {  
        UserDao dao = new UserDao();  
        User user = new User();  
        String uri = req.getRequestURI();  
        if(uri.contains("edit")) { //user/edit/id  
        } else if(uri.contains("create")) { //user/create  
        } else if(uri.contains("update")) { //user/update  
        } else if(uri.contains("delete")) { //user/delete  
        }  
        req.setAttribute("form", user); // form: User  
        req.setAttribute("items", dao.findAll()); // items: List<User>  
        req.getRequestDispatcher("/views/user.jsp").forward(req, resp);  
    }  
}
```

**\${form}:** Hiển thị lên form nhập  
 **\${items}:** Hiển thị lên bảng

## THIẾT KẾ GIAO DIỆN VIEWS/USER.JSP

- Views/user.jsp cần được thiết kế như hình sau. Để làm cho giao diện đẹp, thân thiện bạn cần kết hợp với Bootstrap hoặc CSS

Username?	password?	Fullname?	Email Address	Gender:	Admin	User
<a href="#">Create</a> <a href="#">Update</a> <a href="#">Delete</a> <a href="#">Reset</a>						
NghiemN	songlong	Nguyễn Nghiêm	nghiemn@fpt.edu.vn	Admin	<a href="#">Edit</a>	
TeoNV	123456	Nguyễn Văn Tèo	teonv@fpt.edu.vn	User	<a href="#">Edit</a>	

```
<%@ page pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c" %>
```

```
<!--Thông báo-->
${message}
<c:url var="url" value="/user"/>
<!--Form-->
<form action="${url}/index" method="post">
...
</form>
```

**<c:url/>**: được sử dụng để lấy đường dẫn URL bao gồm cả đường dẫn ứng dụng (context path)

```
<!--Bảng-->
<table border="1" style="width:100%">
...
</table>
```

## THIẾT KẾ FORM VIEWS/USER.JSP

```
<!--Form-->
<form action="${url}/index" method="post">
    <input value="${form.id}" name="id" placeholder="Username?"><br>
    <input value="${form.password}" name="password" placeholder="Password?"><br>
    <input value="${form.fullname}" name="fullname" placeholder="Fullname?"><br>
    <input value="${form.email}" name="email" placeholder="Email Address"><br>
    <input ${form.admin?'checked':''} name="admin" type="radio" value="true">Admin
    <input ${form.admin?':checked'} name="admin" type="radio" value="false">User<br>
    <hr>
    <button formaction="${url}/create">Create</button>
    <button formaction="${url}/update">Update</button>
    <button formaction="${url}/delete">Delete</button>
    <a href="${url}/index">Reset</a>
</form>
```

The image shows a user registration form with the following fields and controls:

- Text input field for "Username?"
- Text input field for "Password?"
- Text input field for "Fullname?"
- Text input field for "Email Address"
- Two radio buttons for "Admin" and "User". The "User" option is selected.
- Four buttons at the bottom: "Create", "Update", "Delete", and "Reset".

## THIẾT KẾ BẢNG VIEWS/USER.JSP

```
<!--Bảng-->
<table border="1" style="width:100%">
    <c:forEach var="item" items="${items}">
        <tr>
            <td>${item.id}</td>
            <td>${item.password}</td>
            <td>${item.fullname}</td>
            <td>${item.email}</td>
            <td>${item.admin?'Admin':'User'}</td>
            <td><a href="#">Edit</a></td>
        </tr>
    </c:forEach>
</table>
```

NghiemN	songlong	Nguyễn Nghiêm	nghiemn@fpt.edu.vn	Admin	<a href="#">Edit</a>
TeoNV	123456	Nguyễn Văn Tèo	teonv@fpt.edu.vn	User	<a href="#">Edit</a>

# HOÀN THIỆN MÃ CHO USERSERVLET.JAVA

EDIT

```
// uri = .../user/edit/id
```

```
String id = uri.substring(uri.lastIndexOf("/") + 1);
user = dao.findById(id);
```

try {

```
String id = req.getParameter("id");
dao.remove(id);
req.setAttribute("message", "Xóa thành công");
} catch (Exception e) {
    req.setAttribute("message", "Xóa thất bại");
}
```

try {

```
BeanUtils.populate(user, req.getParameterMap());
dao.create(user);
req.setAttribute("message", "Thêm mới thành công");
```

DELETE

```
} catch (Exception e) {
```

```
    req.setAttribute("message", "Thêm mới thất bại");
}
```

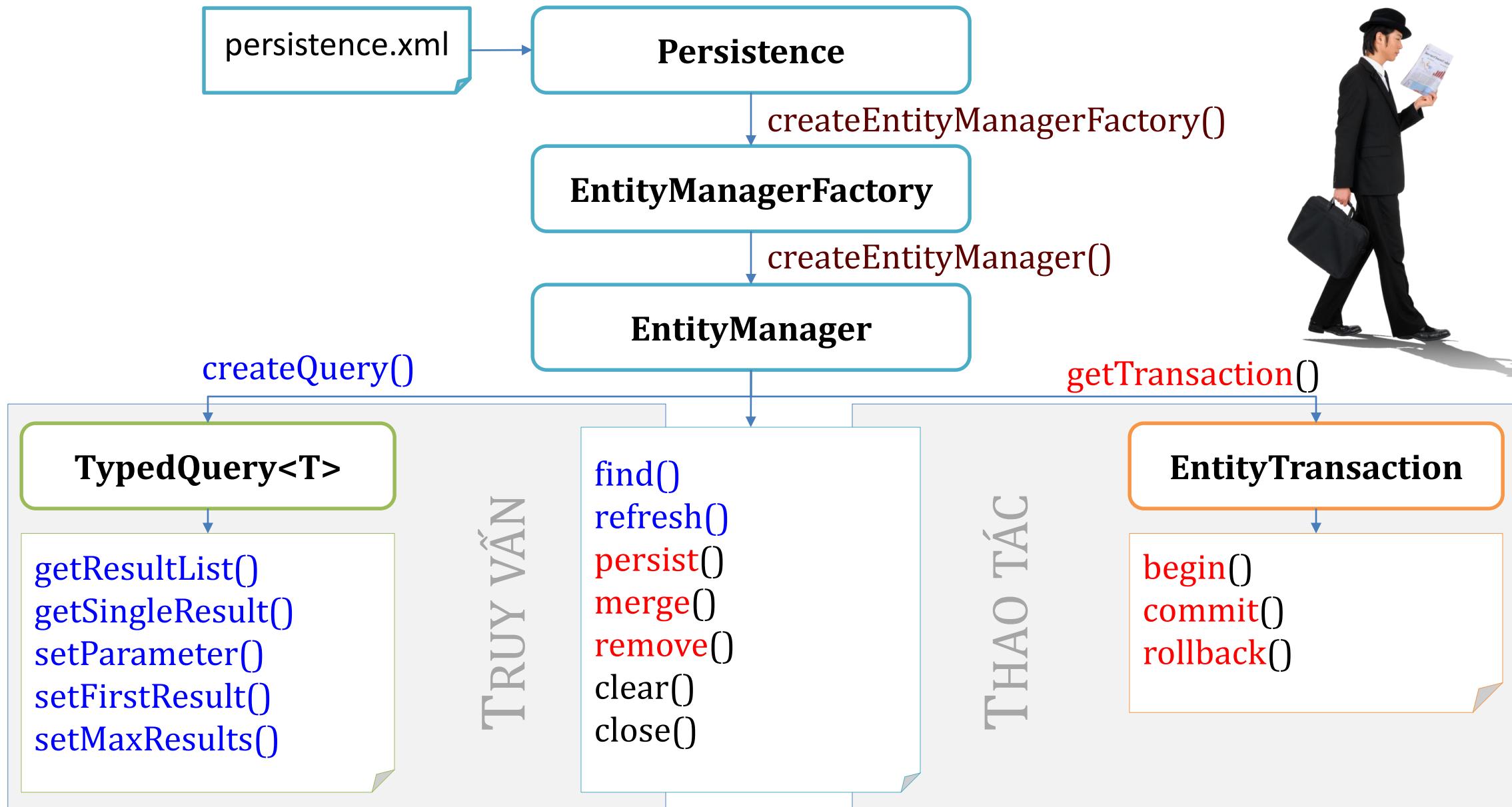
UPDATE

}

INSERT

try {

```
BeanUtils.populate(user, req.getParameterMap());
dao.update(user);
req.setAttribute("message", "Cập nhật thành công");
} catch (Exception e) {
    req.setAttribute("message", "Cập nhật thất bại");
}
```





Cảm ơn