

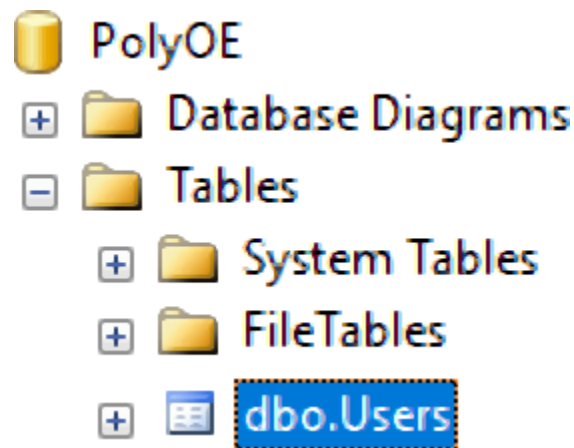
**MỤC TIÊU:**

Kết thúc bài thực hành này bạn có khả năng


- ☐ Khai báo thư viện cần thiết JPA và SQL Server Driver
- ☐ Xây dựng các thực thể mô tả cấu trúc dữ liệu quan hệ
- ☐ Lập trình JPA cơ bản (thao tác và truy vấn dữ liệu)
- ☐ Xây dựng DAO (Data Access Object) chuyên dụng truy xuất dữ liệu

**PHẦN I: CSDL VÀ JPA CƠ BẢN****BÀI 1: TẠO CSDL, CẤU HÌNH PERSISTENCE.XML**

Hãy tạo CSDL có tên là PolyOE trong SQL Server chứa bảng Users có cấu trúc như hình sau:

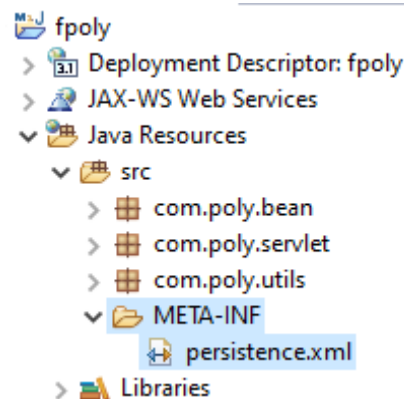


Hình 1: CSDL

THANHVINHCENTER.PolyOE - dbo.Users			
	Column Name	Data Type	Allow Nulls
	Id	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"/>

Hình 2: Cấu trúc bảng Users

Tạo file src/META-INF/persistence.xml để cấu hình kết nối CSDL PolyOE



```
<?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=PolyOE" />
    <property name="javax.persistence.jdbc.user" value="sa" />
    <property name="javax.persistence.jdbc.password" value="*****" />
    <property name="hibernate.show_sql" value="true" />
    <property name="hibernate.format_sql" value="true" />
    </properties>
</persistence-unit>
</persistence>

```

Khai báo thư viện cần thiết JPA và SQL Server Driver

```

<!-- 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>

```

Xây dựng thực thể User mô tả cấu trúc dữ liệu của bảng Users

```

@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
}

```

## BÀI 2: LẬP TRÌNH THAO TÁC DỮ LIỆU

Viết chương trình Java Console chứa các phương thức cho phép thực hiện các công việc sau đây:

- ☐ `create()`: Thêm mới một User
- ☐ `update()`: Tìm và Cập nhật thông tin một User theo id
- ☐ `delete()`: Tìm và Xóa một User theo id
- ☐ `findAll()`: Truy vấn và xuất tất cả các User
- ☐ `findByRole(boolean role)`: Truy vấn theo vai trò
- ☐ `findKeyword(String keyword)`: Truy vấn theo từ khóa
- ☐ `findOne(String username, String password)`: Truy vấn theo id và password
- ☐ `findPage(int page, int size)`: Truy vấn một trang

Hướng dẫn:

- ☐ Tạo `JpaProgram.java` và Tổ chức file `JpaProgram.java`

```
package com.poly.app;

public class JpaProgram {
    public static void main(String[] args) {
        create();
        // update();
        // delete();
        // findAll();
        // findByRole(true);
        // findKeyword("Nguyễn");
        // findOne("TeoNV", "123456");
        // findPage(3, 5);
    }
    private static void create() {}
    private static void update() {}
    private static void delete() {}
    private static void findAll() {}
    private static void findByRole(boolean) {}
    private static void findKeyword(String keyword) {}
    private static void findOne(String username, String password) {}
    private static void findPage(int page, int size) {}
}
```

- ☐ Viết mã thao tác

Thao tác là hành động làm thay đổi dữ liệu. Mã thao tác cần phải điều khiển transaction và được tổ chức theo cấu trúc như sau:

```
// Nạp persistence.xml và tạo EntityManagerFactory
EntityManagerFactory emf = Persistence.createEntityManagerFactory("PolyOE");
// Tạo EntityManager để bắt đầu làm việc với CSDL
EntityManager em = emf.createEntityManager();
try {
    em.getTransaction().begin(); // Bắt đầu Transaction
    // MÃ THAO TÁC
    em.getTransaction().commit(); // Chấp nhận kết quả thao tác
    System.out.println("Thêm mới thành công!");
} catch (Exception e) {
    em.getTransaction().rollback(); // Hủy thao tác
    System.out.println("Thêm mới thất bại!");
}
em.close();
emf.close();
```

Phần highlight màu xanh là bắt buộc cho cả thao tác và truy vấn.

Phần highlight màu vàng chỉ dành cho thao tác

- ☐ Viết mã cho create()

```
// Tạo Entity
User entity = new User();
entity.setId("TeoNV");
entity.setFullname("Nguyễn Văn Tèo");
entity.setEmail("teonv@gmail.com");
entity.setPassword("123456");

// Insert vào CSDL
em.persist(entity);
```

- ☐ Viết mã cho update()

```
// Truy vấn thực thể theo id
User entity = em.find(User.class, "TeoNV");
// Thay đổi thông tin thực thể
entity.setPassword("poly@2020");
entity.setAdmin(true);
// Cập nhật thực thể
em.merge(entity);
```

- ☐ Viết mã cho delete()

```
User entity = em.find(User.class, "TeoNV");
em.remove(entity);
```

□ Viết mã cho findAll()

```
// Câu lệnh truy vấn JPQL
String jpql = "SELECT o FROM User o";
// Tạo đối tượng truy vấn
TypedQuery<User> query = em.createQuery(jpql, User.class);
// Truy vấn
List<User> list = query.getResultList();
// Hiển thị kết quả truy vấn
for (User user: list) {
    System.out.println(">>Fullname: " + user.getFullname());
    System.out.println(">>Is Admin: " + user.isAdmin());
}
```

□ Viết mã cho findByRole(boolean role)

```
// Câu lệnh truy vấn JPQL
String jpql = "SELECT o FROM User o WHERE o.admin=:role";
// Tạo đối tượng truy vấn
TypedQuery<User> query = em.createQuery(jpql, User.class);
query.setParameter("role", role);
// Truy vấn
List<User> list = query.getResultList();
// Hiển thị kết quả truy vấn
```

□ Viết mã cho findKeyword(String keyword)

```
// Câu lệnh truy vấn JPQL
String jpql = "SELECT o FROM User o WHERE o.fullname LIKE ?0";
// Tạo đối tượng truy vấn
TypedQuery<User> query = em.createQuery(jpql, User.class);
query.setParameter(0, keyword);
// Truy vấn
List<User> list = query.getResultList();
// Hiển thị kết quả truy vấn
```

□ Viết mã cho findOne(String username, String password)

```
// Câu lệnh truy vấn JPQL
String jpql = "SELECT o FROM User o WHERE o.id=:id AND o.password=:pw";
// Tạo đối tượng truy vấn
TypedQuery<User> query = em.createQuery(jpql, User.class);
query.setParameter("id", username);
query.setParameter("pw", password);
// Truy vấn một thực thể
User user = query.getSingleResult();
```

```
// Hiển thị kết quả truy vấn
```

```
System.out.println(">>Fullname: " + user.getFullname());
System.out.println(">>Is Admin: " + user.isAdmin());
```

- ☐ Viết mã cho findPage(int page, int size)

```
// Câu lệnh truy vấn JPQL
```

```
String jpql = "SELECT o FROM User o";
```

```
// Tạo đối tượng truy vấn
```

```
TypedQuery<User> query = em.createQuery(jpql, User.class);
```

```
query.setFirstResult(page * size);
```

```
query.setMaxResults(size);
```

```
// Truy vấn
```

```
List<User> list = query.getResultList();
```

```
// Hiển thị kết quả truy vấn
```

```
for (User user: list) {
    System.out.println(">>Fullname: " + user.getFullname());
    System.out.println(">>Is Admin: " + user.isAdmin());
}
```

## PHẦN II: KẾT HỢP JPA VÀ SERVLET/JSP

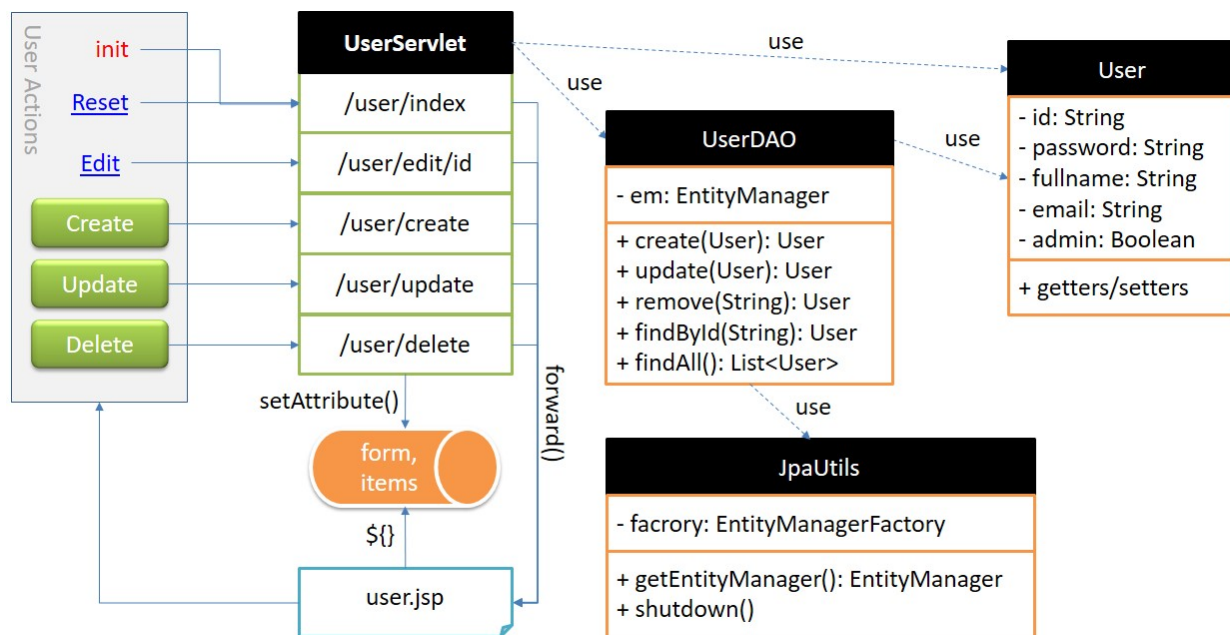
### BÀI 3: XÂY DỰNG ỨNG DỤNG CRUD

Xem lại hướng dẫn bài giảng phần 2 để xây dựng ứng dụng CRUD quản lý User như hình sau.

Username?					
Password?					
Fullname?					
Email Address					
<input type="radio"/> Admin <input checked="" type="radio"/> User					
<div> <input type="button" value="Create"/> <input type="button" value="Update"/> <input type="button" value="Delete"/> <a href="#">Reset</a> </div>					
NghiemnN	songlong	Nguyễn Nghiệm	nghiemn@fpt.edu.vn	Admin	<a href="#">Edit</a>
TeoNV	123456	Nguyễn Văn Tèo	teonv@fpt.edu.vn	User	<a href="#">Edit</a>

Hướng dẫn:

Để thực hiện bài thực hành này bạn cần phải xây dựng các thành phần như sơ đồ sau đây:



- ☐ Lớp thực thể User
- ☐ Lớp truy xuất dữ liệu UserDAO và JpaUtils
- ☐ Servlet điều khiển hành vi người sử dụng UserServicelet
- ☐ Giao diện user.jsp

#### BÀI 4: QUẢN LÝ TÀI KHOẢN NGƯỜI SỬ DỤNG

Xây dựng AccountServlet và ánh xạ với các URL để người sử dụng quản lý tài khoản thành viên của họ:

```

@WebServlet({
    "/account/sign-up",
    "/account/sign-in",
    "/account/sign-out",
    "/account/forgot-password",
    "/account/change-password",
    "/account/edit-profile"
})

```

Trong đó:



- **“/account/sign-up”**: Đăng ký thành viên
  - *Thêm mới*
- **“/account/sign-in”**: Đăng nhập
  - *Truy vấn theo id, so sánh mật khẩu, duy trì trong session*
- **“/account/sign-out”**: Đăng xuất
  - *Xóa khỏi session*
- **“/account/forgot-password”**: Lấy lại mật khẩu đã quên qua email
  - *Truy vấn và gửi email*
- **“/account/change-password”**: Đổi mật khẩu
  - *Truy vấn và cập nhật mật khẩu*
- **“/account/edit-profile”**: Cập nhật thông tin tài khoản
  - *Lấy user từ session, hiển thị lên form và cập nhật*

Yêu cầu hoàn thành các chức năng tương ứng với các URL có **màu đỏ** ở trên.

Hướng dẫn:

- Sử dụng lại UserDao và User của bài 3 để thực hiện tiếp bài này
- Tạo AccountServlet.java và các jsp cần thiết (sign-in.jsp, sign-up.jsp và edit-profile.jsp...)
- Tổ chức AccountServlet.java

```
@WebServlet({
    "/account/sign-in",
    "/account/sign-up",
    "/account/sign-out",
    "/account/forgot-password",
    "/account/change-password",
    "/account/edit-profile"
})
public class AccountServlet extends HttpServlet{
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        String uri = req.getRequestURI();
        if(uri.contains("sign-in")) {
            this.doSignIn(req, resp);
        }
        else if(uri.contains("sign-up")) {
            this.doSignUp(req, resp);
        }
    }
}
```

```

    }
    else if(uri.contains("sign-out")) {}
    else if(uri.contains("forgot-password")) {}
    else if(uri.contains("change-password")) {}
    else if(uri.contains("edit-profile")) {
        this.doEditProfile(req, resp);
    }
}

private void doSignIn(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    String method = req.getMethod();
    if(method.equalsIgnoreCase("POST")) {
        // TODO: ĐĂNG NHẬP
    }
    req.getRequestDispatcher("/views/account/sign-in.jsp").forward(req, resp);
}

private void doSignUp(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    String method = req.getMethod();
    if(method.equalsIgnoreCase("POST")) {
        // TODO: ĐĂNG KÝ
    }
    req.getRequestDispatcher("/views/account/sign-up.jsp").forward(req, resp);
}

private void doEditProfile(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    User user = (User) req.getSession().getAttribute("user");
    String method = req.getMethod();
    if(method.equalsIgnoreCase("POST")) {
        // TODO: CẬP NHẬT
    }
    req.getRequestDispatcher("/views/account/edit-profile.jsp").forward(req, resp);
}
}

```

□ Thiết kế giao diện

- sign-in.jsp
  - <input name="username">
  - <input name="password" type="password">
- sign-up.jsp
  - <input name="id">
  - <input name="password" type="password">
  - <input name="fullname">
  - <input name="email">

- `<input name="admin" type="hidden" value="false">`
- edit-profile.jsp
  - `<input name="id" value="${user.id}">`
  - `<input name="password" value="${user.password}">`
  - `<input name="fullname" value="${user.fullname}">`
  - `<input name="email" value="${user.email}">`
  - `<input name="admin" type="hidden" value="${user.admin}">`
- Viết mã cho AccountServlet
  - **TODO: ĐĂNG NHẬP**

```
String id = req.getParameter("username");
String pw = req.getParameter("password");

try {
    UserDAO dao = new UserDAO();
    User user = dao.findById(id);
    if(!user.getPassword().equals(pw)) {
        req.setAttribute("message", "Sai mật khẩu!");
    }
    else {
        req.setAttribute("message", "Đăng nhập thành công!");
        req.getSession().setAttribute("user", user);
    }
} catch (Exception e) {
    req.setAttribute("message", "Sai tên đăng nhập!");
}
```

- **TODO: ĐĂNG KÝ**

```
try {
    User entity = new User();
    BeanUtils.populate(entity, req.getParameterMap());

    UserDAO dao = new UserDAO();
    dao.create(entity);
    req.setAttribute("message", "Đăng ký thành công!");
} catch (Exception e) {
    req.setAttribute("message", "Lỗi đăng ký!");
}
```

- **TODO: CẬP NHẬT**

```
try {
    BeanUtils.populate(user, req.getParameterMap());
}
```

```
UserDAO dao = new UserDAO();  
dao.update(user);  
req.setAttribute("message", "Cập nhật tài khoản thành công!");  
} catch (Exception e) {  
    req.setAttribute("message", "Lỗi cập nhật tài khoản!");  
}
```

## BÀI 5: TRUY VẤN CÓ ĐIỀU KIỆN

GV cho thêm nhằm nâng cao truy vấn có tham số, phân trang...