Bài tập bắt buộc của Tuần XX GV sẽ công bố cuối giờ thực hành. Bài tập bắt buộc sẽ push lên Github với Repository name: Tuan04

Sinh viên nộp sai yêu cầu bài tập Tuần nào xem như không nộp bài tập Tuần đó.

BÀI TẬP TUẦN 4 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA)

1. *Chương 3: Jakarta Server Page - JSP*

Mục tiêu:

- *Hiểu và áp dụng được các cú pháp căn bản của JSPs trong việc xây dựng ứng dụng Web.*
- *Hiểu và áp dụng được các đối tượng ngầm định (implicit objects).*
- *Hiểu và áp dụng được JavaBean vào ứng dụng JSP.*
- *Hiểu và áp dụng được Expression Language vào ứng dụng Web.*
- *JSP Standard Tag Library (JSTL)*

Dependencies:

```
<!-- Jakarta Servlet API -->
<dependency>
    <groupId>jakarta.servlet</groupId>
    <artifactId>jakarta.servlet-api</artifactId>
    <version>6.0.0</version>
    <scope>provided</scope>
</dependency>

<!-- Jakarta Annotations API -->
<dependency>
    <groupId>jakarta.annotation</groupId>
    <artifactId>jakarta.annotation-api</artifactId>
    <version>3.0.0</version>
    <scope>provided</scope>
</dependency>


<!-- JSTL (Jakarta Tags Core) -->
<dependency>
    <groupId>jakarta.servlet.jsp.jstl</groupId>
    <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
    <version>3.0.0</version>
</dependency>

<dependency>
    <groupId>org.glassfish.web</groupId>
    <artifactId>jakarta.servlet.jsp.jstl</artifactId>
    <version>3.0.1</version>
</dependency>

<!-- Mariadb Client -->
<dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>3.5.1</version>
</dependency>

<!-- Jakarta Persistence API -->
<dependency>
    <groupId>jakarta.persistence</groupId>
    <artifactId>jakarta.persistence-api</artifactId>
    <version>3.2.0</version>
</dependency>

<!-- Hibernate core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.6.1.Final</version>
</dependency>

<!-- Hibernate validator -->
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>8.0.2.Final</version>
</dependency>
```

KẾT NỐI DATABASE

1. *Sử dụng @Resource với Datasource*

a. *Tạo file **context.xml** tại folder **webapp/META-INF***

```
<Context>
    <Resource name="jdbc/shopping" auth="Container" type="javax.sql.DataSource"
        maxTotal="100" maxIdle="30" maxWaitMillis="10000"
        username="root" password="123456" driverClassName="org.mariadb.jdbc.Driver"
        url="jdbc:mariadb://localhost:3306/shoppingdb" URIEncoding="UTF-8"/>
</Context>
```

Ref: https://tomcat.apache.org/tomcat-10.1-doc/jndi-datasource-examples-howto.html

b. *Tạo Dao Interface*

```
public interface ProductDAO {
    public List<Product> findAll();
}
```

c. *Implement Dao Interface*

```
public class ProductDAOImpl implements ProductDAO {
    private final DataSource datasource;
    public ProductDAOImpl(DataSource datasource) {
        this.datasource = datasource;
    }
    @Override
    public List<Product> findAll() {
        String sql = "SELECT id, name, price, image FROM product";
        List<Product> list = new ArrayList<>();
        try (
                Connection con = (Connection) this.datasource.getConnection();
                PreparedStatement ps = (PreparedStatement) con.prepareStatement(sql);
                ResultSet rs = ps.executeQuery();) {
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                Double price = rs.getDouble("price");
                String image = rs.getString("image");
                list.add(new Product(id, name, price,image));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }
}
```

d. *Sử dụng **@Resource** với **Datasource** ở Controller*

```
@WebServlet(name = "productController", urlPatterns = "/products")
public class ProductController extends HttpServlet {
    @Resource(name = "jdbc/shopping")
    private DataSource dataSource;

    private ProductDAO productDAO;

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        this.productDAO = new ProductDAOImpl(this.dataSource);
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setAttribute("products", productDAO.findAll());
        req.getRequestDispatcher("views/product/index.jsp").forward(req, resp);
    }
}
```

2. *Sử dụng JPA*

a. *Tạo file **persistence.xml** tại folder **resources/META-INF***

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence version="3.0" xmlns="https://jakarta.ee/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence
https://jakarta.ee/xml/ns/persistence/persistence_3_0.xsd">
    <persistence-unit name="user-management" transaction-type="RESOURCE_LOCAL">
        <!-- Persistence provider -->
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <properties>
            <property name="jakarta.persistence.jdbc.driver"
value="org.mariadb.jdbc.Driver" />
            <property name="jakarta.persistence.jdbc.url"
value="jdbc:mariadb://localhost:3306/userdb" />
            <property name="jakarta.persistence.jdbc.user"   value="root" />
            <property name="jakarta.persistence.jdbc.password" value="123456" />

            <!-- Automatically export the schema -->
            <property name="jakarta.persistence.schema-generation.database.action"
value="none" />

            <!-- Echo all executed SQL to console -->
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
            <property name="hibernate.highlight_sql" value="true" />

        </properties>
    </persistence-unit>
</persistence>
```

b. *Tạo **EntityManagerFactoryUtil***

```
public class EntityManagerFactoryUtil {
    private static final EntityManagerFactory entityManageFactory;

    static {
        try {
            entityManageFactory = Persistence.createEntityManagerFactory("user-
management");
        } catch (Throwable ex) {
            System.err.println("Initial EntityManagerFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static EntityManager getEntityManager() {
        return entityManageFactory.createEntityManager();
    }

    public static void close() {
        if (entityManageFactory.isOpen()) {
            entityManageFactory.close();
        }
    }
}
```

c. *Cách sử dụng*

```
try (EntityManager entityManager = EntityManagerFactoryUtil.getEntityManager()) {
    UserDAO userDao = new UserDAOImpl(entityManager);
    List<User> listUser = userDao.findAll();

 } catch (Exception e) {
    e.printStackTrace();
}
```

Jakarta Bean Validation

Ref: https://jakarta.ee/learn/docs/jakartaee-tutorial/current/beanvalidation/bean-validation/bean-validation.html

1. Tạo class User

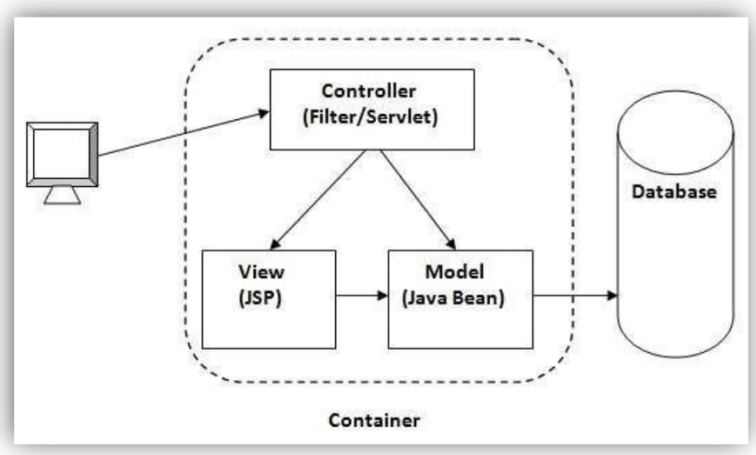
```java
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;

public class User {
    protected int id;

    @NotNull(message = "Name must be not null")
    @NotEmpty(message = "Name must be not empty")
    @Size(min = 8, max = 50, message = "Name must be between 8 and 50 characters")
    protected String name;

    @NotEmpty(message = "Name must be not empty")
    @Email(message = "Email should be valid")
    protected String email;

    protected String country;

    public User() {
    }

    public User(String name, String email, String country) {
        super();
        this.name = name;
        this.email = email;
        this.country = country;
    }

    public User(int id, String name, String email, String country) {
        super();
        this.id = id;
        this.name = name;
        this.email = email;
        this.country = country;
    }

    // getter & setter
```

2. *Kiểm tra validate ở Servlet*

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {

        ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
        Validator validator = factory.getValidator();

        String name = req.getParameter("name");
        String email = req.getParameter("email");
        String country = req.getParameter("country");

        User user = new User(name, email, country);

        Set<ConstraintViolation<User>> violations = validator.validate(user);

        if (!violations.isEmpty()) {
            req.setAttribute("errors", violations);
            req.setAttribute("user", user);
            req.getRequestDispatcher("/index.jsp").forward(req, resp);
        } else {
            req.setAttribute("message", "Register successful!");
            req.getRequestDispatcher("/success.jsp").forward(req, resp);
        }
    }
}
```

Bài 1 (Bài 2 - Tuần 3). *JSP - Model View Controller (MVC)*

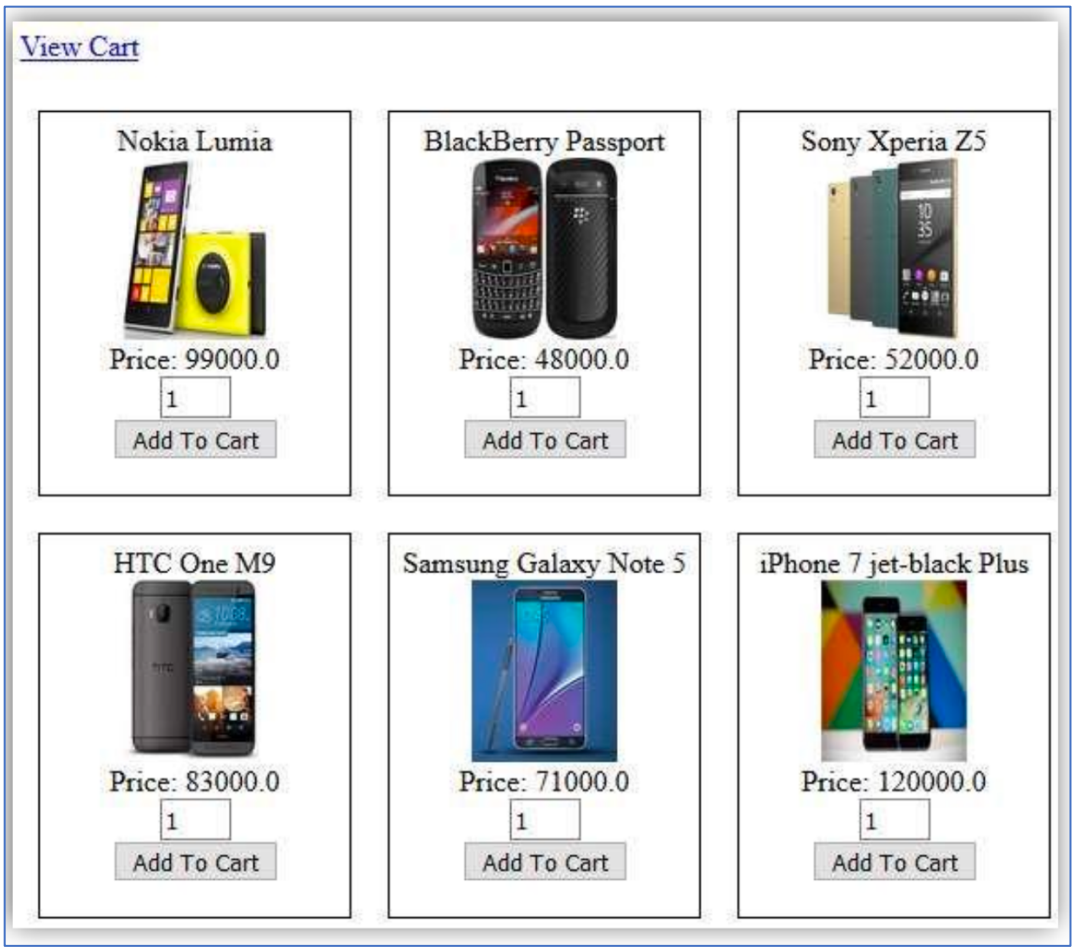
Thực hiện form đăng ký tài khoản với JSPs. Sau khi đăng ký thành công cập nhật danh sách tài khoản (không hiển thị password), Thực hiện theo mô hình MVC.



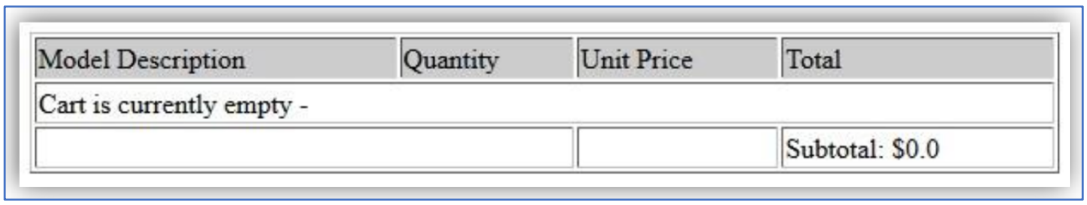
Bài 2 (Bài 4 – Tuần 3). *JSP - Thao tác với JSP Session*

Thực hiện website mua bán sản phẩm đơn giản dùng JSP Session.

Hiển thị danh sách sản phẩm và cho thêm sản phẩm vào giỏ hàng



Khi giỏ hàng không có gì, hiển thị thông báo



Model Description	Quantity	Unit Price	Total
Cart is currently empty -			
			Subtotal: $0.0

Khi giỏ hàng có sản phẩm, cho phép xóa, sửa các sản phẩm.

Model Description	Quantity	Unit Price	Total
PRO02 BlackBerry Passport	1 Update Delete	$48000.0	$48000.0
PRO06 iPhone 7 jet-black Plus	1 Update Delete	$120000.0	$120000.0
			Subtotal: $168000.0

Bài 3: *JSPs - Bài tập tổng hợp - MVC*

Xây dựng ứng dụng quản lý tin tức cho một website trực tuyến, các trang Web cần thực hiện các chức năng: *xem tin tức theo từng danh mục, thêm mới tin tức và xóa tin tức không cần thiết.* Một phần của cơ sở dữ liệu được mô tả như sau:

DANHMUC (MADM, TENDANHMUC, NGUOIQUANLY, GHICHU)

TINTUC (MATT, TIEUDE, NOIDUNGTT, LIENKET, MADM)

Xây dựng ứng dụng dùng Servlet/JSPs kết hợp với SQL Server/MariaDB và Web server Tomcat thực hiện các yêu cầu với mô hình MVC (Các Servlet đóng vai trò Controller):

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình. Cơ sở dữ liệu trong SQL Server lưu tên: QUANLYDANHMUC
- Dùng HTML/CSS tạo Layout chung cho các *trang Web quản lý tin tức trực tuyến*

- Tạo lớp mô tả thông tin *TinTuc.java* (bao gồm các get/set cho các thuộc tính và các constructors) dùng để thao tác với tin tức trực tuyến.
- Tạo lớp *DanhSachTinTucQuanLy.java* (bao gồm các phương thức thao tác với CSDL: lấy tin tức theo danh mục tin tức, thêm mới tin tức và xóa tin tức).

- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu dùng JSPs: (*DanhSachTinTucServlet.java - DanhSachTinTuc.jsp*) Liệt kê dữ liệu danh sách tin tức theo từng danh mục.
- Tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu o Tạo Form cho phép thêm nội dung tin tức với đầy đủ thông tin phù hợp với CSDL (*TinTucFormServlet.java, TinTucForm.jsp*)
    - Kiểm tra dữ liệu nhập phía Client trên Form:
        - Mã TT, Tiêu đề, Liên kết, Nội dung, Thông tin danh mục của tin là bắt buộc nhập.
        - Liên kết bắt đầu bởi "*http://"* (dùng RegularExpression).
        - Nội dung không quá 255 ký tự (dùng RegularExpression).
    - Thông tin nhập vào hợp lệ sau khi nhấn nút "Thêm" sẽ được thêm vào cơ sở dữ liệu và hiển thị dữ liệu trên màn hình (*KetQua.jsp hoặc trả về DanhSachTinTuc.jsp*).
- Tạo giao diện Web cho phép thực hiện chức năng quản lý: thao tác hủy dữ liệu. Khi chọn chức năng này, hiển thị danh sách và kèm theo chức năng hủy tin tức ra khỏi cơ sở dữ liệu (*QuanLyFormServlet.java, QuanLyForm.jsp*).