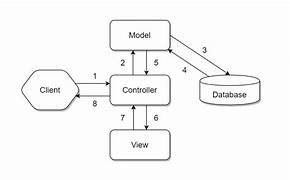
**JSP MVC jdbc (Mô hình MVC)**



# **Bước 0: Chuẩn bị -setup**

* Maven Project 🡺 Add Dependencies

-7 cái:

B1 :Vào trang wed : <https://mvnrepository.com/tags/mvc>

B2: Lưu ý SQL thì phải phải đúng jdk

* + - * Cách xem : Project Facets

<https://mvnrepository.com/artifact/jakarta.servlet/jakarta.servlet-api>

<https://mvnrepository.com/artifact/jakarta.servlet.jsp/jakarta.servlet.jsp-api>

<https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core>

<https://mvnrepository.com/artifact/jakarta.servlet.jsp.jstl/jakarta.servlet.jsp.jstl-api>

<https://mvnrepository.com/artifact/org.glassfish.web/jakarta.servlet.jsp.jstl>

<https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client>

<https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc>

https://mvnrepository.com/artifact/jakarta.annotation/jakarta.annotation-api

# **Bước 1 : Tạo các package**

M(MODEL) :các entity

V(VIEW) :.jsp,.html

C(CONTROLLER) :dao, daoImpl

Đầu tiên tạo Entity hoặc Models chứa các class đối tượng

# **Bước 2 : Tạo DataSource (Chuỗi kết nối)**

Tạo file : Context.xml (webapp\META-INF)

Tạo database :

|  |
| --- |
| create database bai4\_Shopping  use bai4\_Shopping  create table product(  id INT IDENTITY(1,1),  name NVARCHAR(20) not null,  price float not null,  image NVARCHAR(20) not null,  ) |

# **Bước 3: Tạo package dao**

-Trung gian

= Tạo class interface

= Xác định có những phương thức nào : lấy tất cả, thêm, lấy 1

# **Bước 4 : Tạo package daoImpl**

-Package duy nhất làm việc với Database

B1: Tạo DataSource : để kết nối tới Database

B2: Hoàn thiện các phương thức kết nối đến Database

# **Bước 5 : Tạo package Controller**

Tạo class Servlet;

Tạo 2 biến :DataSource và biến đại diện class Dao;

Cấu hình cho DataSource @Resource(name = "jdbc/shooping")

Tạo method init : gõ chữ init sau đó enter

|  |
| --- |
| @Override  **public** **void** init(ServletConfig config) **throws** ServletException {  **super**.init(config);  **try** {  System.***out***.println("Init ProductController");  System.***out***.println("DataSource: " + **this**.dataSource.getConnection()); // .getConnection dùng để kiểm tra kết  // nối đến database  } **catch** (SQLException e) {  e.printStackTrace();  }  productDAO = **new** ProductDaoImpl(**this**.dataSource);  } |

Trong Java Servlet, phương thức init() có công dụng khởi tạo servlet và thực hiện các thao tác cần thiết trước khi servlet xử lý các yêu cầu (HTTP requests). Đây là phương thức được gọi một lần duy nhất khi servlet được khởi tạo bởi servlet container (ví dụ: Tomcat, Jetty) và trước khi bất kỳ yêu cầu nào được xử lý.

Tóm lại : chuẩn bị các tài nguyên cần thiết như

* + - * kết nối cơ sở dữ liệu,
      * cấu hình đối tượng DataSource
      * khởi tạo các đối tượng phụ thuộc như DAO
      * thiết lập cấu hình từ file cấu hình

# **Bước 6 : Code ở hàm doGet**

Trong ngữ cảnh gửi dữ liệu từ một trang web đến server:

* **doPost** thường được sử dụng để gửi dữ liệu, đặc biệt khi cần gửi các thông tin nhạy cảm (như mật khẩu) hoặc khi dữ liệu lớn hơn giới hạn của URL. Phương thức này cho phép gửi dữ liệu trong body của yêu cầu, giúp bảo mật hơn và không bị giới hạn kích thước như trong GET.
* **doGet** có thể cũng gửi dữ liệu, nhưng dữ liệu sẽ được thêm vào URL (query string), không phù hợp cho các yêu cầu gửi thông tin nhạy cảm hoặc dữ liệu lớn.

**Tóm lại:**

* **doPost**: Được sử dụng để gửi dữ liệu đến server (thích hợp cho form).Hoặc có gì thay đổi
* **doGet**: Chủ yếu dùng để truy vấn thông tin từ server.

Do bài này chúng ta chỉ cần load tất cả sản phẩm từ database lên trang jsp cho người dùng xem nên mình dùng doGet là được r. Việc này k quan trọng và cũng k sợ lộ gì đó nên dùng doGet.

|  |
| --- |
| //Them danh sach san pham vao request  request.setAttribute("products", list);  //Chuyen toi trang product.jsp request.getRequestDispatcher("views/product/product.jsp").forward(request, response); |

Giải thích:

//Dòng thứ 1

**Chức năng**: Dòng này lưu trữ danh sách sản phẩm (list) vào trong đối tượng request với một khóa (key) là "products".

**Mục đích**: Điều này cho phép các thông tin (trong trường hợp này là danh sách sản phẩm) được truyền từ servlet sang JSP (JavaServer Page) để hiển thị. Khi bạn gọi getAttribute("products") trong JSP, bạn có thể truy cập danh sách sản phẩm đã được lưu.

//Dòng thứ 2

Dòng này là cầu nối giữa servlet và JSP, cho phép servlet gửi dữ liệu (danh sách sản phẩm) đến JSP để hiển thị mà không cần thực hiện yêu cầu mới. Điều này giúp cho việc phát triển ứng dụng web trở nên dễ dàng và mạch lạc hơn, vì bạn có thể xử lý logic nghiệp vụ trong servlet và hiển thị giao diện người dùng trong JSP.

# **Bước 7 :Tạo folder views (chứa file jsp)/ resources (chứa ảnh)**

= Trong webapp/views : folder này chỉ dùng để chứa các view trả về

Đầu các trang jsp trong views thêm câu sau

|  |
| --- |
| <%@ taglib uri="jakarta.tags.core" prefix="c"%> |

= Trong webapp/resources/images/ : folder này chỉ dùng để chứa ảnh

# **Bước 8 : Code view**

=Tạo file index.jsp để tạo trang view chính (trang này người dùng tương tác đầu tiên)

=<jsp:forward page="ProductController"></jsp:forward>

Trong đó ProductController là trên Servlet hay JSP mà trang index muốn trỏ tới

|  |
| --- |
| <c:forEach items="${products}" var="product">  <tr>  <td>${product.id}</td>  <td>${product.name}</td>  <td><img src="${pageContext.request.contextPath}/resources/images/${product.image}" width="80" height="80"></td>  <td>${product.price}</td>  <td><a href="${pageContext.request.contextPath}/buy?id=${product.id}">BUY</a></td>  </tr>  </c:forEach> |

Với câu lệnh sau để lấy đường dẫn ${pageContext.request.contextPath}/

# **Giải thích**

* Session

|  |
| --- |
| **HttpSession session = request.getSession(); là gì?**   1. **Session (phiên làm việc)**:    * Khi bạn truy cập vào một trang web, trang web đó sẽ tạo ra một **phiên làm việc** (session) cho bạn.    * Phiên làm việc này giống như một **chiếc hộp** mà website dùng để lưu thông tin của bạn trong suốt thời gian bạn đang sử dụng website.    * Ví dụ: Khi bạn thêm sản phẩm vào giỏ hàng, thông tin về sản phẩm sẽ được lưu trong session. Nếu bạn truy cập các trang khác nhau của website, giỏ hàng của bạn vẫn giữ nguyên nhờ vào session. 2. **request.getSession() làm gì?**    * Lệnh này yêu cầu website **lấy** cái hộp (session) của bạn.    * Nếu bạn **đã có hộp** từ trước (khi truy cập website lần đầu), nó sẽ lấy cái hộp cũ của bạn.    * Nếu bạn **chưa có hộp** (lần đầu truy cập), lệnh này sẽ **tạo một cái hộp mới** để lưu trữ thông tin trong suốt thời gian bạn sử dụng trang web. 3. **Ví dụ thực tế**:    * Bạn vào một cửa hàng online và thêm sản phẩm vào giỏ hàng.    * Website cần một nơi để **lưu giữ thông tin về sản phẩm bạn đã thêm**, và cái nơi đó chính là **session**.    * Lệnh HttpSession session = request.getSession(); là để lấy cái "hộp" này, sau đó bạn có thể **lưu** giỏ hàng của mình vào trong đó. |