

# **JAVA PROGRAMMING – SERVLET/JSP & JPA**

---

# GIỚI THIỆU MÔN HỌC

---

## □ Phần 1: Giới thiệu

- ❖ Thế giới web
- ❖ Mô hình ứng dụng web
- ❖ Mô hình công nghệ 3 lớp
- ❖ Mô hình công nghệ Servlet/JSP
- ❖ Cài đặt môi trường phát triển
- ❖ Tạo dự án web động và chạy thử

## □ Phần 2: Khảo sát dự án web động

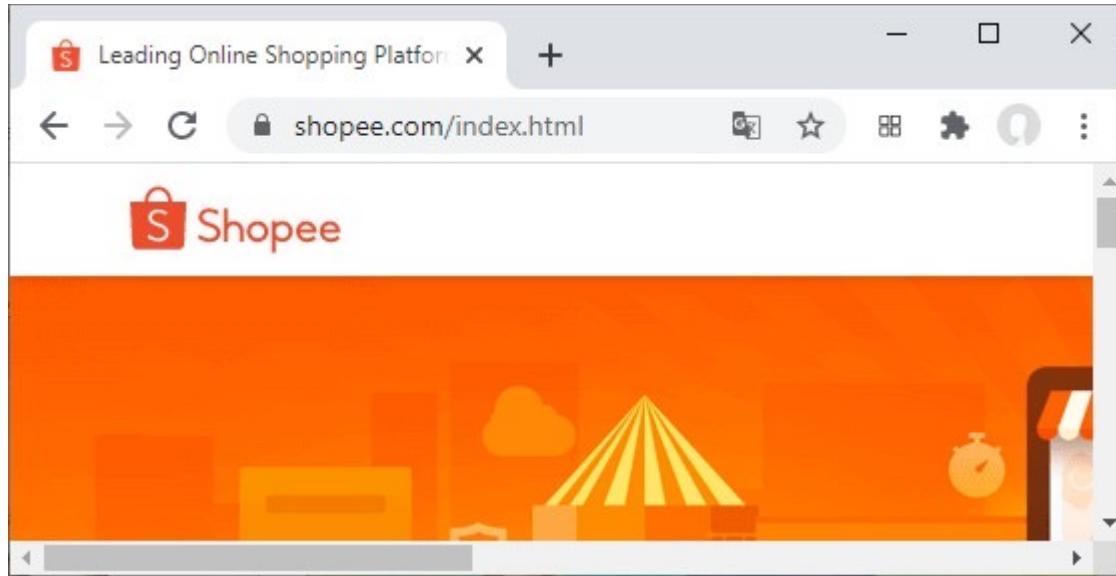
- ❖ Tổ chức dự án web động
- ❖ Tổ chức mô hình MVC
- ❖ Truyền dữ liệu từ Servlet sang JSP
- ❖ Làm việc với tham số đơn giản
- ❖ Đóng gói và triển khai

# ① GIỚI THIỆU

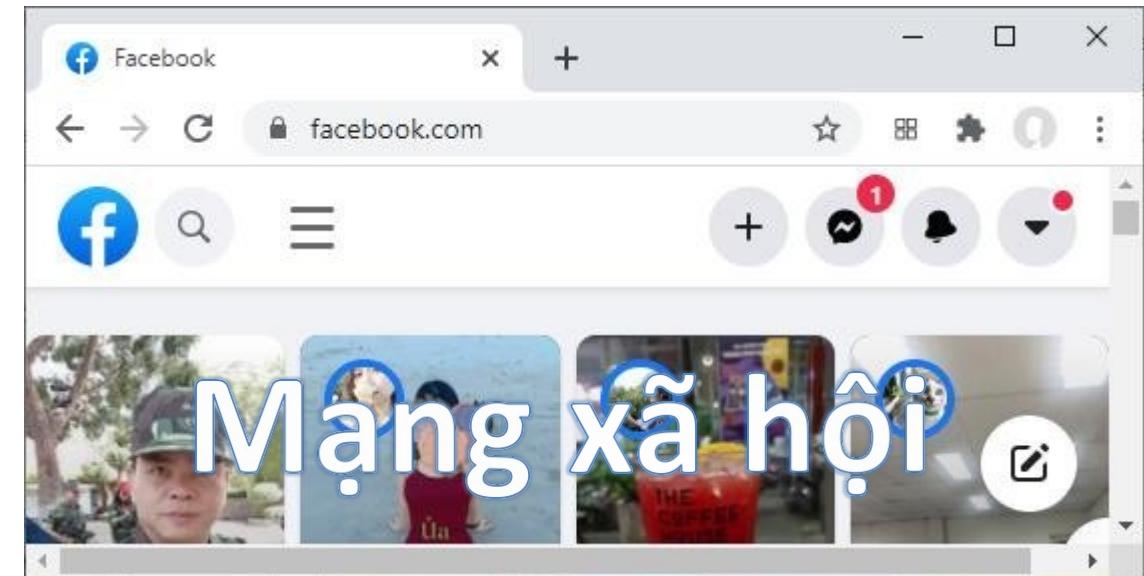
---

# GIỚI THIỆU THẾ GIỚI WEB

---



Mua sắm



# GIỚI THIỆU INTERNET VÀ WEB

---



## Mở đầu

---

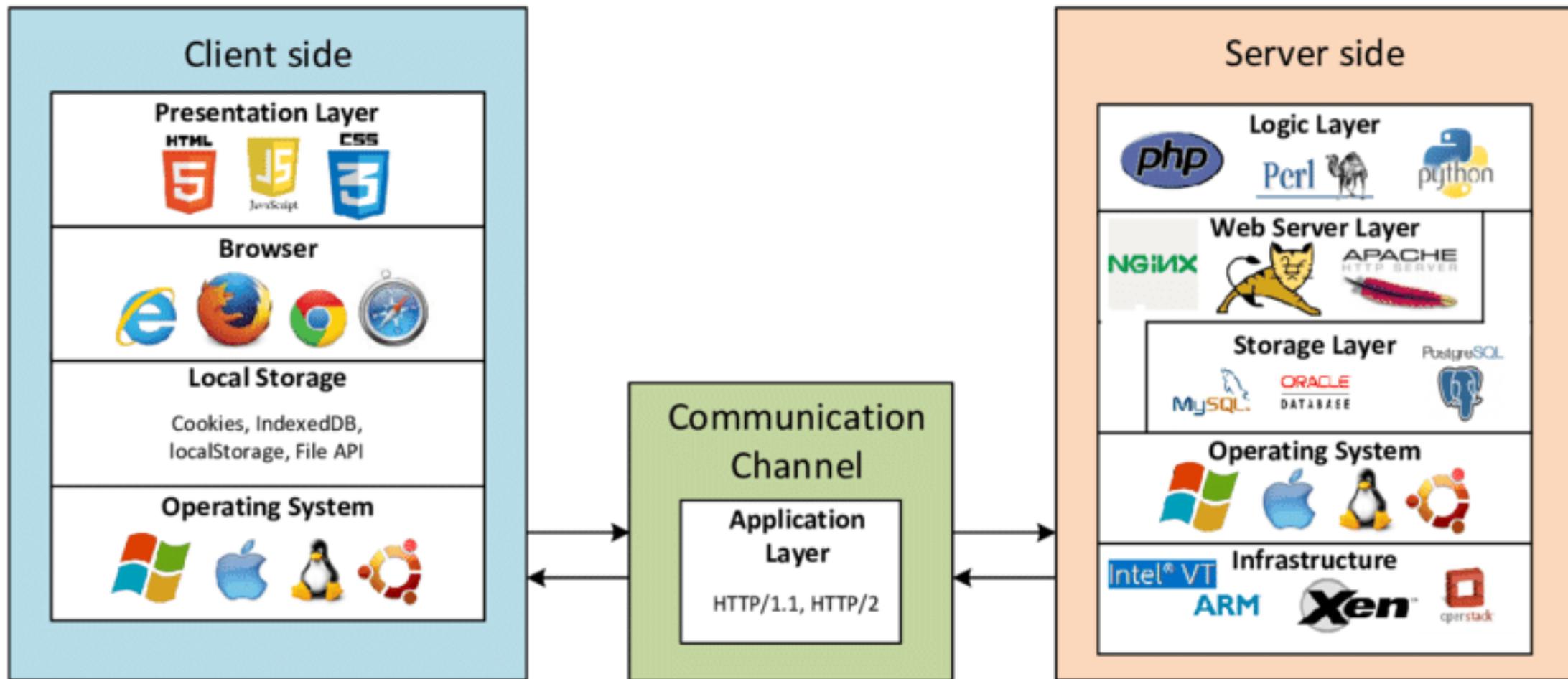
- Máy khách và máy chủ ngày càng trở nên quá tải bởi độ phức tạp và yêu cầu của người dùng. Từ đó phát sinh mô hình phát triển ứng dụng đa tầng (multi-tier).
- Mục tiêu là làm cho máy khách trở nên gọn nhẹ, dễ cấu hình. Tất cả mã nguồn lõi, cài đặt, xử lý đều thực hiện trên máy chủ, do đó chương trình trở nên dễ quản lý, các máy khách luôn được phục vụ với phiên bản chương trình mới nhất
- Web là một ví dụ điển hình nhất của mô hình ứng dụng đa tầng
- Mô hình ứng dụng đa tầng đáp ứng được nhu cầu về mặt tốc độ, bảo mật, cũng như sự đáng tin cậy của ứng dụng

### ❑ Một số thuật ngữ

- ❖ **Web page:** Trang web , tập tin văn bản có nội dung gồm: content + presentation data được viết trong markup language.
- ❖ **Web Browser:** Phần mềm duyệt web, Internet Explorer, FireFox, Chrome,...  
Web browser biết cách kết nối với server để lấy về một web page theo yêu cầu của user. Web browser biết cách thông dịch các thẻ đánh dấu (presentation data) trong web page để thể hiện nội dung ra màn hình.
- ❖ **Web server:** Phần mềm có khả năng đáp ứng yêu cầu lấy thông tin từ Web browser
- ❖ **Database:** Cơ sở dữ liệu lưu trữ nhờ một RDBMS như SQL Server, Oracle.
- ❖ **Protocol:** Giao thức, cách thức truyền thông trên mạng, một tập các quy định về tổ chức vật lý của một đơn vị thông tin được truyền trên mạng.

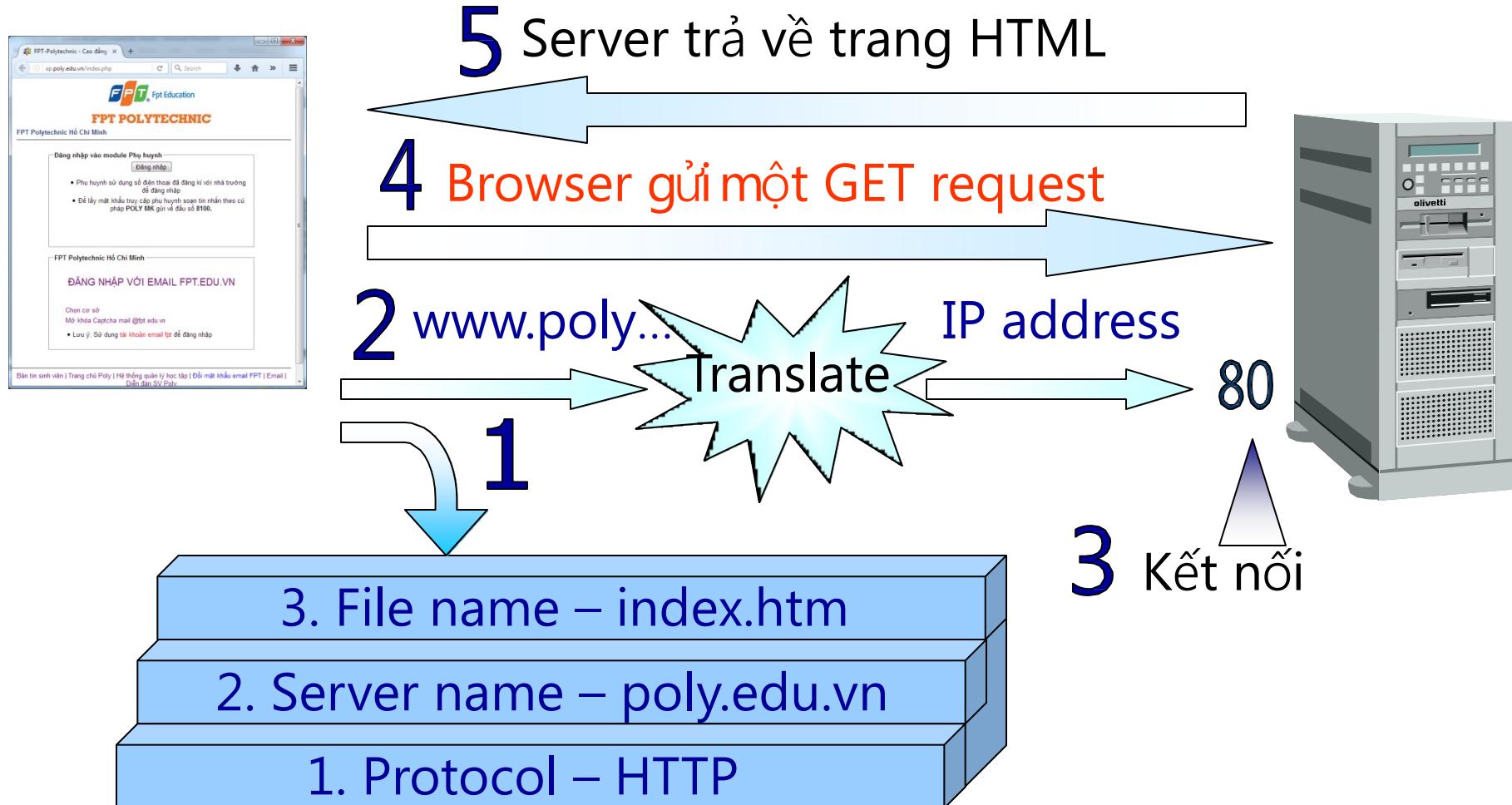
# Giới thiệu

---



# CƠ BẢN VỀ LẬP TRÌNH WEB

## □ Web Server làm việc như thế nào ?



## CƠ BẢN VỀ LẬP TRÌNH WEB

---

### ❑ Cơ chế tương tác Browser – Web Server khi user submit 1 form

- **Bước 1:** Browser thu gom data mà user đã nhập trên form trước khi user submit.
- **Bước 2:** Browser phát sinh một HTTP Request tới web server, trong đó bao gồm các thông tin: data từ user, thông tin về browser, thông tin về loại yêu cầu, độ dài của yêu cầu. Mỗi yêu cầu cần phải mô tả một phương thức (method). Ba phương thức thường dùng:
  - Phương thức **HEAD** : Lấy thông tin phần HEAD của tài liệu.
  - Phương thức **GET** : Yêu cầu thực thi một chương trình trên server, dữ liệu từ user được kèm theo URL ( giống như gửi thư không bao thư).
  - Phương thức **POST** : Yêu cầu thực thi một chương trình trên server, dữ liệu từ user không được kèm theo URL mà được đóng gói truyền ngầm( giống như gửi thư có bao thư).
- **Bước 3:** Web server thực thi một server script để tạo ra đáp ứng (response).
- **Bước 4:** Web server trả kết quả là đáp ứng đã sinh ra về cho browser

# Phương thức Http

```
GET /doc/test.html HTTP/1.1 → Request Line
Host: www.test101.com
Accept: image/gif, image/jpeg, /**
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35
bookId=12345&author=Tan+Ah+Teck → A blank line separates header & body
                                         } Request Message Body
```

The diagram illustrates an HTTP request message. It starts with the method "GET /doc/test.html HTTP/1.1" followed by a blank line. Then come the "Request Headers": "Host: www.test101.com", "Accept: image/gif, image/jpeg, /\*\*", "Accept-Language: en-us", "Accept-Encoding: gzip, deflate", "User-Agent: Mozilla/4.0", and "Content-Length: 35". Another blank line separates the headers from the "Request Message Body", which contains the query string "bookId=12345&author=Tan+Ah+Teck". Brackets on the right group the headers and body under the heading "Request Message Header".

```
HTTP/1.1 200 OK → Status Line
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html
<h1>My Home page</h1> → A blank line separates header & body
                                         } Response Message Body
```

The diagram illustrates an HTTP response message. It begins with the status line "HTTP/1.1 200 OK" followed by a blank line. Then come the "Response Headers": "Date: Sun, 08 Feb xxxx 01:11:12 GMT", "Server: Apache/1.3.29 (Win32)", "Last-Modified: Sat, 07 Feb xxxx", "ETag: \"0-23-4024c3a5\"", "Accept-Ranges: bytes", "Content-Length: 35", "Connection: close", and "Content-Type: text/html". Another blank line separates the headers from the "Response Message Body", which contains the HTML content "<h1>My Home page</h1>". Brackets on the right group the headers and body under the heading "Response Message Header".

## MỘT SỐ CÔNG NGHỆ PHÍA SERVER

---

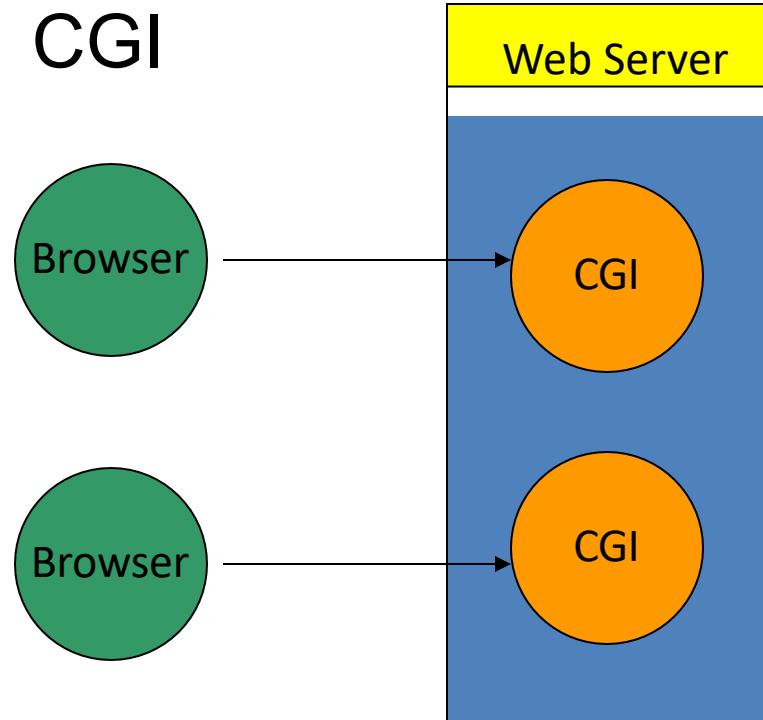
- CGI – Common Gateway Interface
- SSJS – Server Side Java Script
- PHP – Personal Home Page
- ASP – Active Server Page
- Java Servlet
- JSP – Java Server Page

# MỘT SỐ CÔNG NGHỆ PHÍA SERVER

## COMMON GATEWAY INTERFACE (CGI)

- ❑ Browser gửi request, CGI là 1 file .exe viết bằng C , Pascal , Perl....
- ❑ Yếu điểm : mỗi request phải tạo ra 1 CGI
- ❑ instance để phục vụ cho Client

```
#include <stdio.h>
void main()
{
    printf("<HTML>\n");
    printf("<Title>My CGI program</Title>\n");
    printf("<Body><H1> Hello
</H1></Body>\n");
    printf("</HTML>\n");
}
```



# MỘT SỐ CÔNG NGHỆ PHÍA SERVER

---

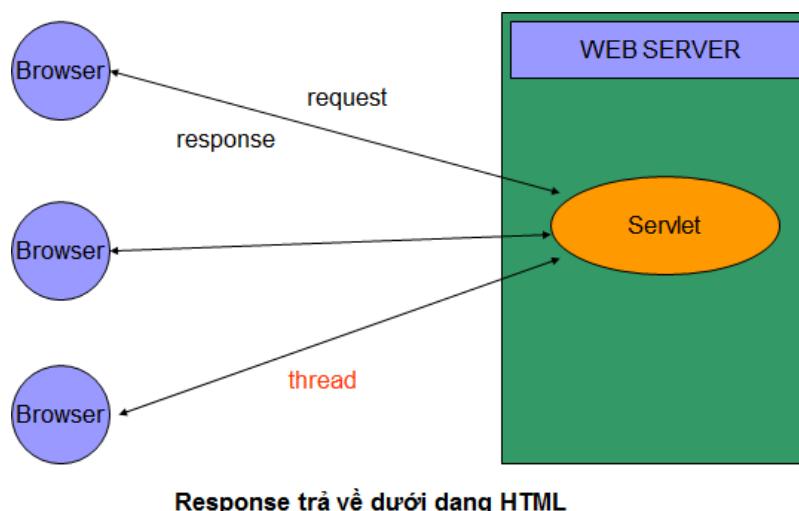
## ISAPI – ASP CỦA MICROSOFT

- ❑ CGI chậm vì mỗi lần có lời request phải load vào memory , Window cung cấp cách thức tạo CGI ở dạng DLL ( gọi là ứng dụng ISAPI : Internet Server API)
  - ❖ ISAPI chỉ nạp 1 lần vào memory khi lần đầu tiên Web Sever triệu gọi chúng. Để viết ISAPI : Visual C++ , Delphi
  - ❖ Nhược điểm : chỉ chạy trên hệ điều hành Window
- ❑ ASP : Active Server Page là 1 ngôn ngữ Script , sẽ thi hành các Script ( ASP.dll) và trả về trang Web
  - ❖ <HTML>
  - ❖ <% Response.write("Hello World !!!") %>
  - ❖ <% Response.write("Today is : ") + Date() %>
  - ❖ </HTML>

# MỘT SỐ CÔNG NGHỆ PHÍA SERVER

## SERVLET – JSP CỦA SUN

- ❑ Đổi nghịch lại ISAPI của Microsoft , Sun đưa ra 1 khái niệm tương tự : Servlet và tương đương ASP là JSP
- ❑ JSP (Java Server Page)
  - ❖ <HTML>
  - ❖ <% out.println("Hello World !!!") %>
  - ❖ <% Date date = new Date() %>
  - ❖ <% out.println("Today is : ") + date %>
  - ❖ </HTML>



# MỘT SỐ CÔNG NGHỆ PHÍA SERVER

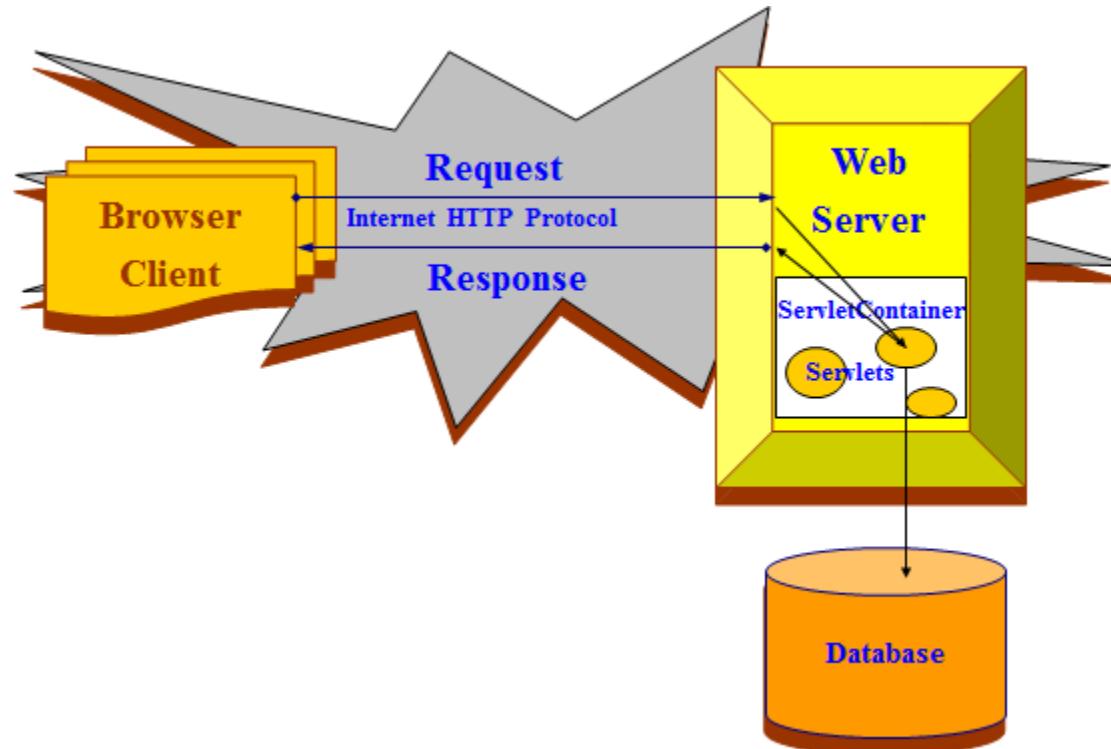
---

## SERVLET – JSP CỦA SUN

- ❑ Chương trình Java thực thi trên Web Server và JVM. (khắc phục các nhược điểm của CGI)
- ❑ Lớp trung gian giữa Web Browser hay HTTP Client với các ứng dụng hay CSDL trên HTTP server.
- ❑ Thực hiện bằng cơ chế Web Application. servlet đón nhận request từ client thực hiện xử lý theo các phương thức/services được implement trong servlet và kết xuất response trả về theo định dạng của trang HTML.
- ❑ Chức năng: đọc dữ liệu HTML form, kết nối CSDL, vận hành cookies, sessions, bảo mật ...
- ❑ Cung cấp các thành phần API để thực hiện tương tác với các ứng dụng Web

# MỘT SỐ CÔNG NGHỆ PHÍA SERVER

## SERVLET MODEL



*Cơ chế giao tiếp Client với Java Web server*

## Java EE là gì

---

- Java EE là viết tắt của Java Platform, Enterprise Edition là nền tảng tiêu chuẩn và mở để xây dựng, phát triển các ứng dụng doanh nghiệp lớn, bao gồm: ứng dụng mạng, web, đa tầng, phân tán... J2EE là mở rộng của J2SE
- Các phiên bản
  - J2EE 1.2 (December 12, 1999)
  - J2EE 1.3 (September 24, 2001)
  - J2EE 1.4 (November 11, 2003)
  - Java EE 5 (May 11, 2006)
  - Java EE 6 (December 10, 2009)
  - Java EE 7 (May 28, 2013)

## Java EE là gì

---

- J2EE cung cấp các API cho việc phát triển ứng dụng nhằm:
  - Giảm thời gian phát triển ứng dụng
  - Giảm độ phức tạp của ứng dụng
  - Tăng hiệu suất ứng dụng

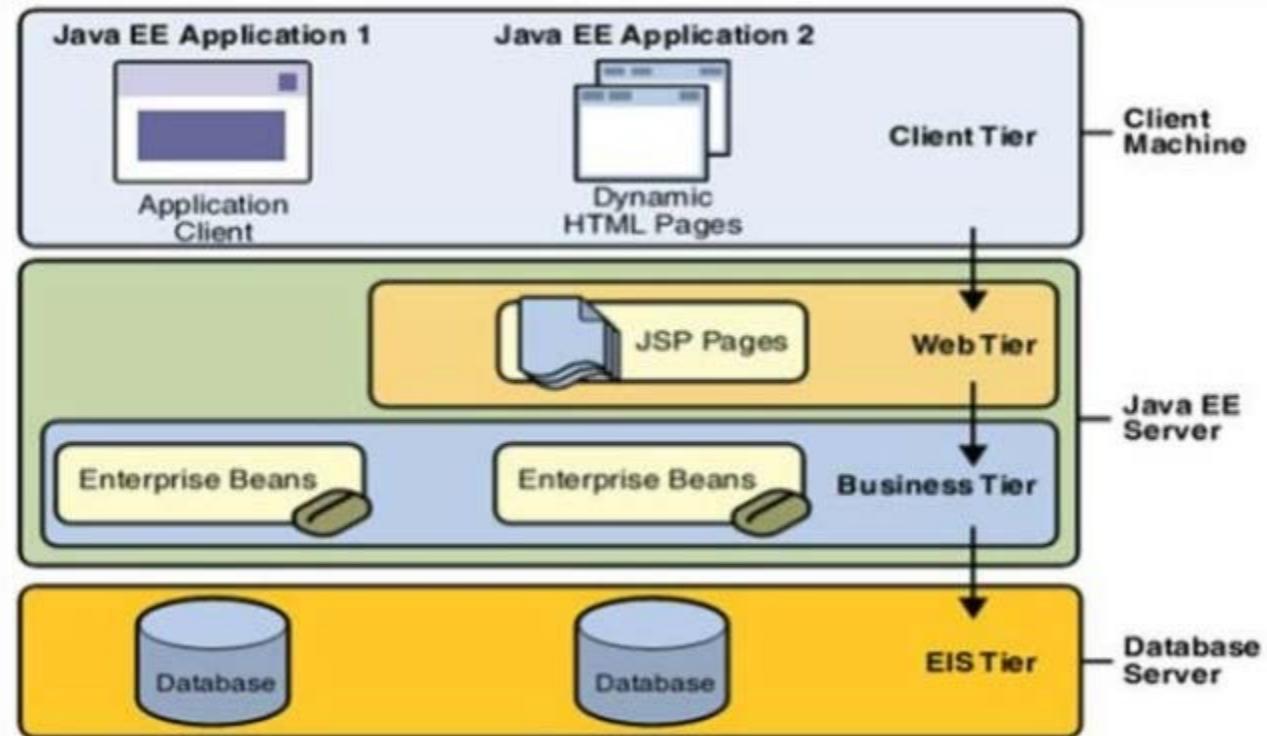
## Java EE là gì

---

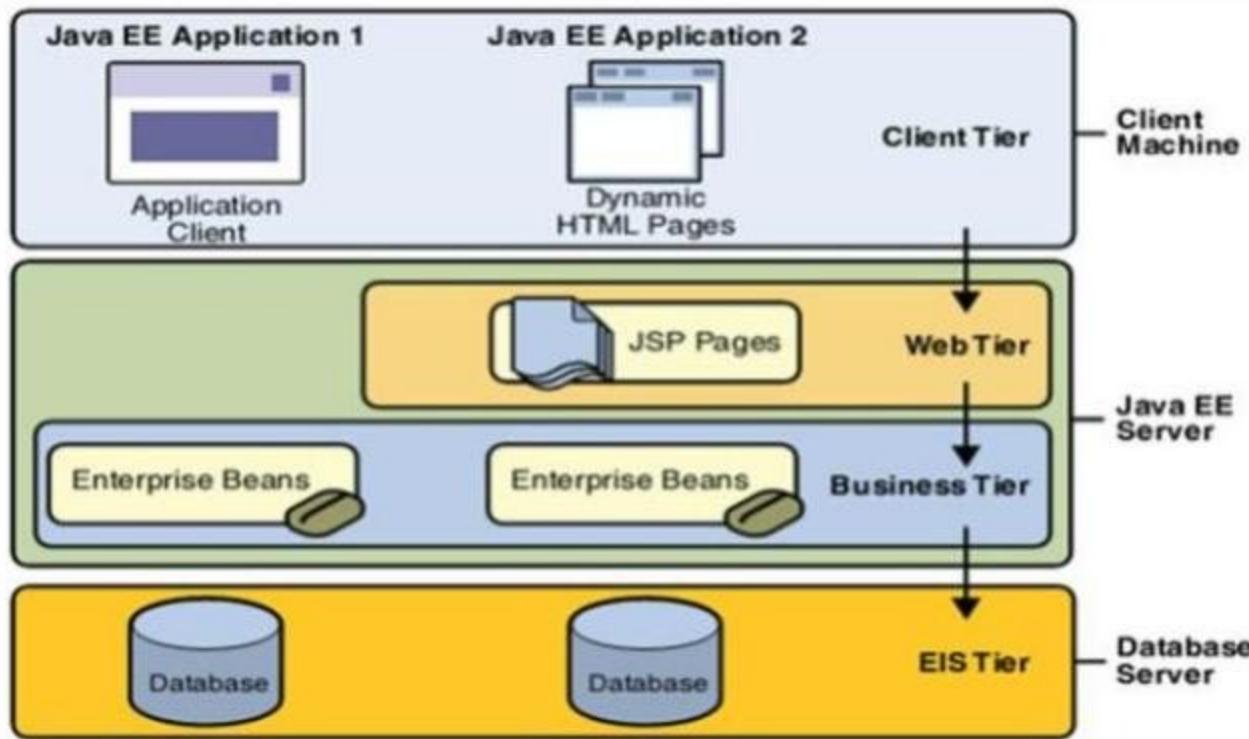
- Là kiến trúc ứng dụng đa tầng với ưu điểm:
  - Khả năng mở rộng
  - Khả năng truy cập
  - Khả năng quản lý
- Mô hình kiến trúc chia làm 2 tầng:
  - Tầng trình diễn
  - Tầng nghiệp vụ

# Các thành phần Java EE

---



# CÁC THÀNH PHẦN JAVA EE

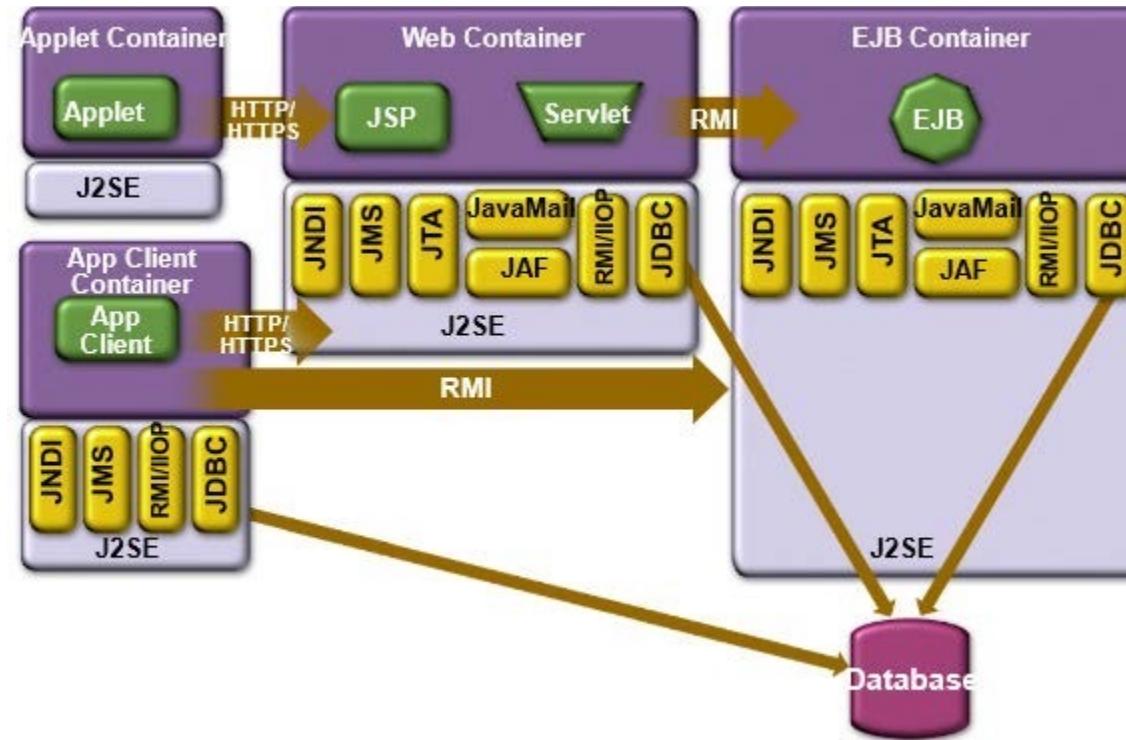


## Nhận xét:

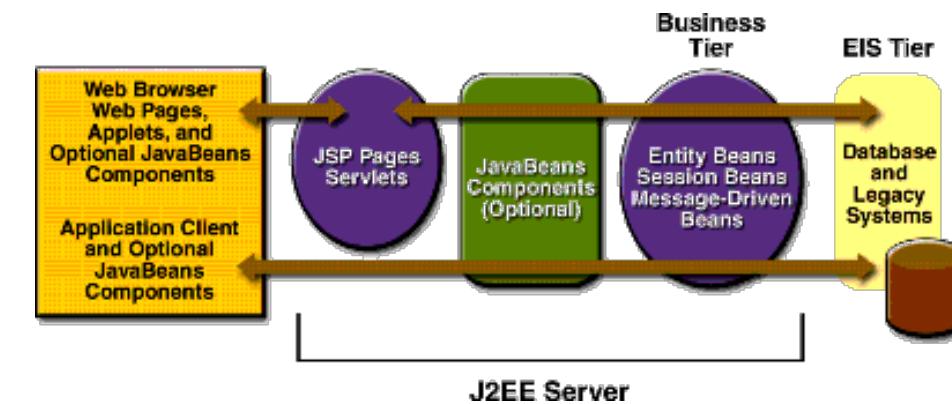
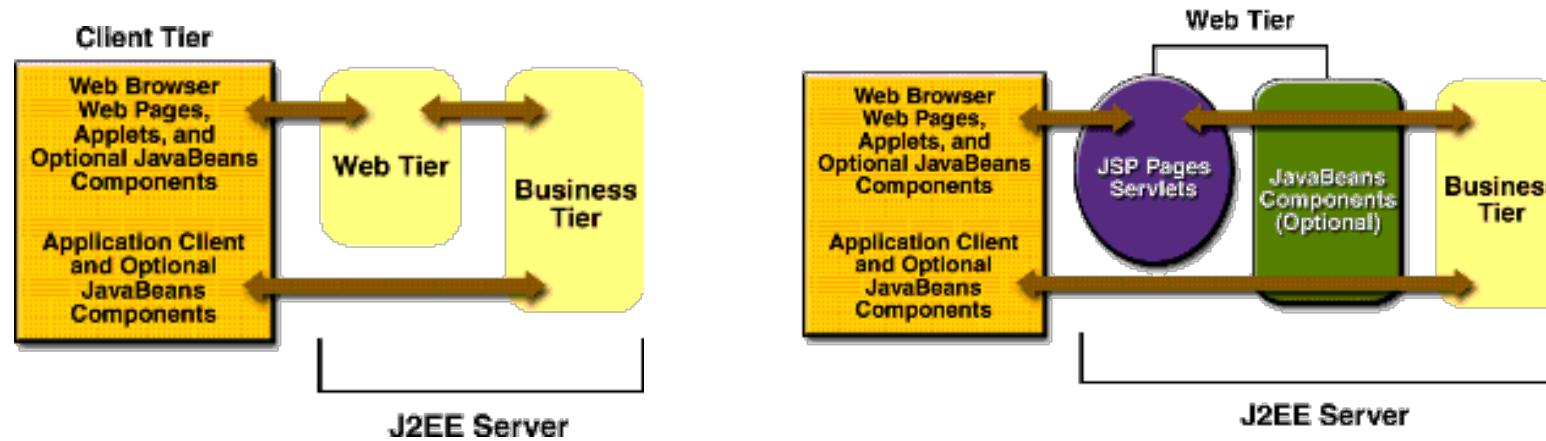
4 lớp nhưng chỉ cài đặt trên 3 máy: Client machine, J2EE Server machine, Database server machine → vẫn là three-tiered Application.

# Java EE Container

---



# CÁC THÀNH PHẦN JAVA EE



**Business Tier và EIS – Enterprise Information System**

## Java EE Container

---

- Container cung cấp các dịch vụ :
  - Dịch vụ bảo mật (security service)
  - Dịch vụ giao dịch (transaction service)
  - Dịch vụ JNDI (JNDI lookup service)

## Java EE Container

---

- Java Application – component này là 1 chương trình standalone chạy bên trong Application Client container. Application Client container cung cấp những API hỗ trợ cho messaging, remote invocation, database connectivity và lookup service. Application Client container đòi hỏi những API sau: J2SE, JMS, JNDI, RIM-IIOP và JDBC. Container này được cung cấp bởi nhà cung cấp application server
- Applet – Applet component là java applet chạy bên trong Applet container, chính là web browser có hỗ trợ công nghệ Java. Applet phải hỗ trợ J2SE API.

## Java EE Container

---

- Servlet và JSP – đây là Web-based component chạy ở bên trong Web container, được hỗ trợ bởi Web Server. Web container là một môi trường run-time cho servlet và jsp. Web Container phải hỗ trợ những API sau: J2SE, JMS, JNDI, JTA, JavaMail, JAF, RIM-IIOP và JDBC. Serlet và JSP cung cấp một cơ chế cho việc chuẩn bị, xử lý, định dạng nội dung động
- Enterprise JavaBean (EJB) – EJB component là business component chạy bên trong EJB container. EJB component là phần nhân, cốt lõi của ứng dụng J2EE. EJB container cung cấp các dịch vụ quản lý transaction, bảo mật, quản lý trạng thái, quay vòng tài nguyên (resource pooling). EJB container phải hỗ trợ những API sau: J2SE, JMS, JNDI, JTA, JavaMail, JAF, RIM-IIOP và JDBC.

## Web components

---

- Web components: Servlet hoặc JSP (cùng với JavaBean và custom tags)
- Các web components chạy trên 1 web container
- Các web container phổ biến: Tomcat, Resin
- Web container cung cấp các dịch vụ hệ thống (system services) cho các Web components
  - Request dispatching, security, và quản lý vòng đời

## Web Application

---

- Web Application là 1 gói triển khai, gồm:
  - Web components (Servlet và JSP)
  - Tài nguyên tĩnh như images
  - Helper classes (sử dụng bởi web components)
  - Thư viện Libraries
  - Deployment descriptor (web.xml)
- Web Application có thể được tổ chức thành:
  - Phân cấp các thư mục và files ( dạng chưa đóng gói)
  - File \*.WAR: dạng đóng gói ( bên trong có phân cấp ),  
sử dụng khi muốn triển khai trên một remote machine

## Cấu hình ứng dụng web:

---

- Thông số cấu hình được đặc tả trong file **web.xml** (Web Applications Deployment Descriptor)

## Cấu hình ứng dụng web:

---

- Tất cả tài liệu XML cần có prolog

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app  
    xmlns="http://java.sun.com/xml/ns/javaee"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-  
    instance"  
    xsi:schemaLocation="http://java.sun.com/xml/n  
    s/javaee  
    http://java.sun.com/xml/ns/javaee/web-  
    app_3_0.xsd    version="3.0">
```

## Alias Path

---

- Khi 1 Servlet container nhận 1 request, nó cần biết Web component nào trong ứng dụng Web sẽ xử lý request này.
  - Thực hiện map **URL path** trong request tới 1 Web component
- Alias Path có thể có 2 dạng
  - /alias-string (cho servlet) hoặc
  - /\*.jsp (cho JSP)

## Chú ý

---

- Phân biệt chữ hoa-thường
- Phải đúng thứ tự như sau:
  - icon, display-name, description, distributable
  - context-param, filter, filter-mapping
  - listener, servet, servlet-mapping, session-config
  - mime-mapping, welcome-file-list
  - error-page, taglib, resource-env-ref, resource-ref
  - security-constraint, login-config, security-role
  - env-entry, ejb-ref, ejb-local-ref

## Tham số khởi tạo và ngữ cảnh (Context and Initialization Parameters)

---

- Biểu diễn ngữ cảnh trong ứng dụng
- Có thể được dùng chung bởi các Web components trong cùng 1 file WAR

```
<web-app>
    ...
    <context-param>
        <param-name>
            javax.servlet.jsp.jstl.fmt.localizationContext
        </param-name>
        <param-value>messages.BookstoreMessages</param-value>
    </context-param>
    ...
</web-app>
```

## Event Listeners

---

- Nhận các events trong vòng đời của servlet

```
<listener>
    <listener-class>
        listeners.ContextListener
    </listener-class>
</listener>
```

## Filter Mappings

---

- Chỉ ra filters nào được áp dụng cho request nào, và theo thứ tự nào

```
<filter>
    <filter-name>OrderFilter</filter-name>
    <filter-class>filters.OrderFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>OrderFilter</filter-name>
    <url-pattern>/receipt</url-pattern>
</filter-mapping>
```

## Error Mappings

---

- Ánh xạ (Map)

- mã trạng thái (status code) trả về trong 1 HTTP response do có 1 ngoại lệ Java
- với 1 Web resource (ví dụ các trang error page)

```
<error-page>
    <exception-type>exception.OrderException</exception-
        type>
    <location>/errorpage.html</location>
</error-page>
```

## Reference

---

S

- Sử dụng khi có web components muốn tham chiếu tới các tài nguyên (database), môi trường.
- Ví dụ: khai báo 1 tham chiếu tới data source

```
<resource-ref>
    <res-ref-name>jdbc/BookDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

## WEB APPLICATION

---

❑ Web Application là 1 gói triển khai, gồm:

- ❖ Web components (Servlet và JSP)
- ❖ Tài nguyên tĩnh như images
- ❖ Helper classes (sử dụng bởi web components)
- ❖ Thư viện Libraries
- ❖ Deployment descriptor (web.xml)

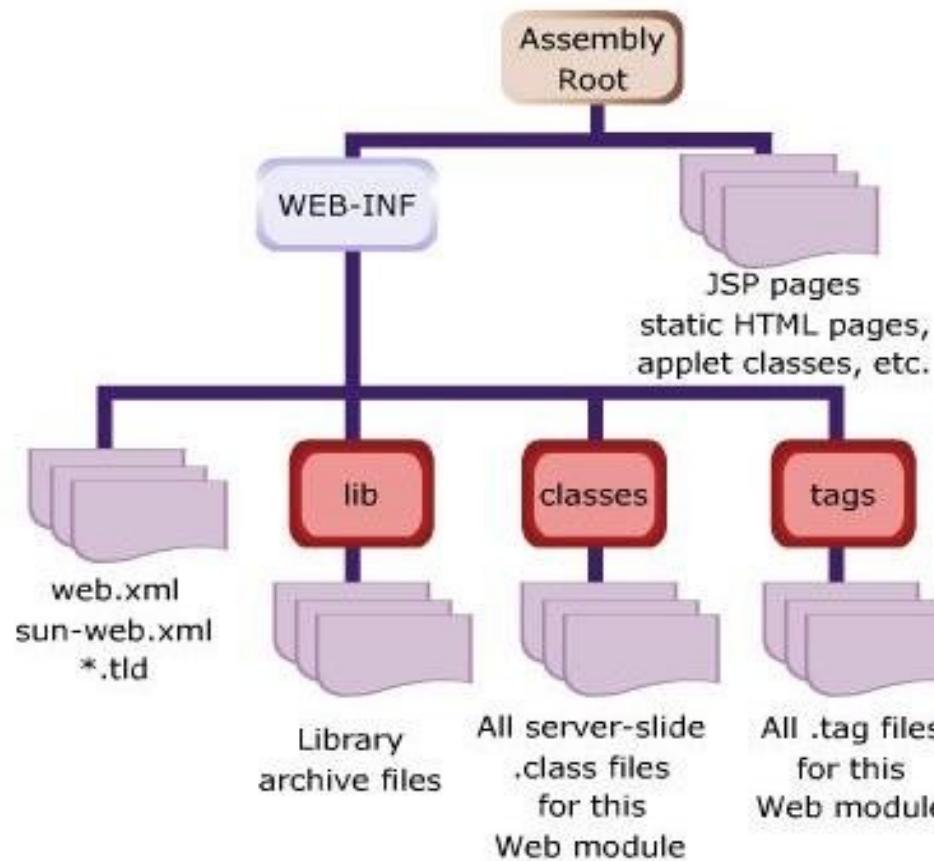
❑ Web Application có thể được tổ chức thành:

- ❖ Phân cấp các thư mục và files ( dạng chưa đóng gói)
- ❖ File \*.WAR: dạng đóng gói ( bên trong có phân cấp ),  
sử dụng khi muốn triển khai trên một remote  
machine

# WEB APPLICATION

---

## CẤU TRÚC CỦA WEB JAVA APPLICATION



Cấu trúc này được đóng gói thành tập tin \*.war để deploy trên Web Server

## CẤU HÌNH ỨNG DỤNG WEB:

- Thông số cấu hình được đặc tả trong file **web.xml** (Web Applications Deployment Descriptor)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <servlet-name>Servlet name</servlet-name>
        <servlet-class>[package.]classname</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Servlet name</servlet-name>
        <url-pattern>/context Path/root</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>default page to show</welcome-file>
    </welcome-file-list>
</web-app>
```

## CẤU HÌNH ỨNG DỤNG WEB:

---

### □ Ví dụ:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>servlet.sample.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/HelloServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>HelloServlet</welcome-file>
    </welcome-file-list>
</web-app>
```

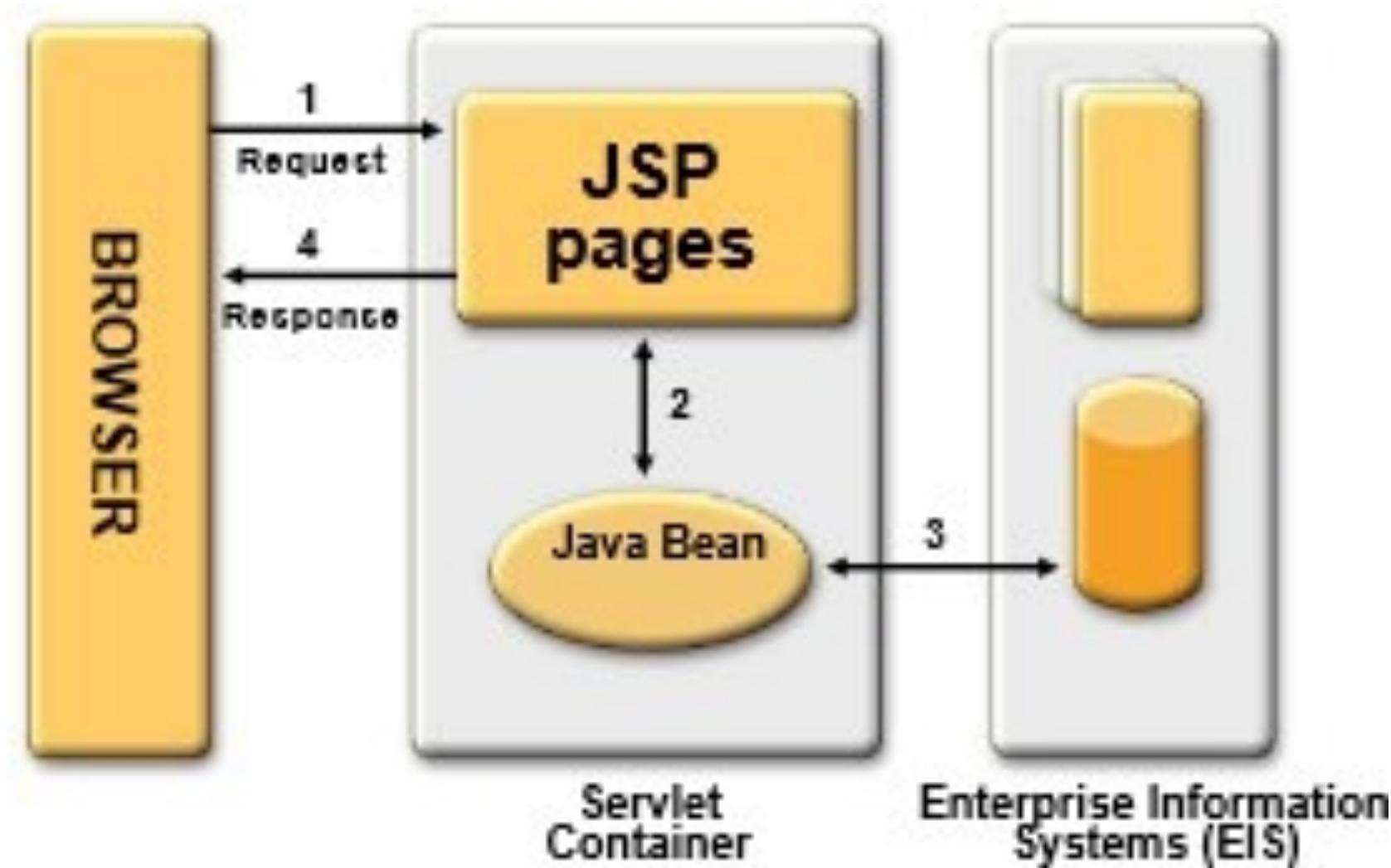
## Quá trình phát triển của kiến trúc ứng dụng Web

---

- No MVC
- MVC Model 1 (**Page-centric**)
- MVC Model 2 (**Servlet-centric**)
- Web application frameworks
  - Struts
- **Standard-based** Web application framework
  - JavaServer Faces (JSR-127)

## MVC 1: Page Centric

---



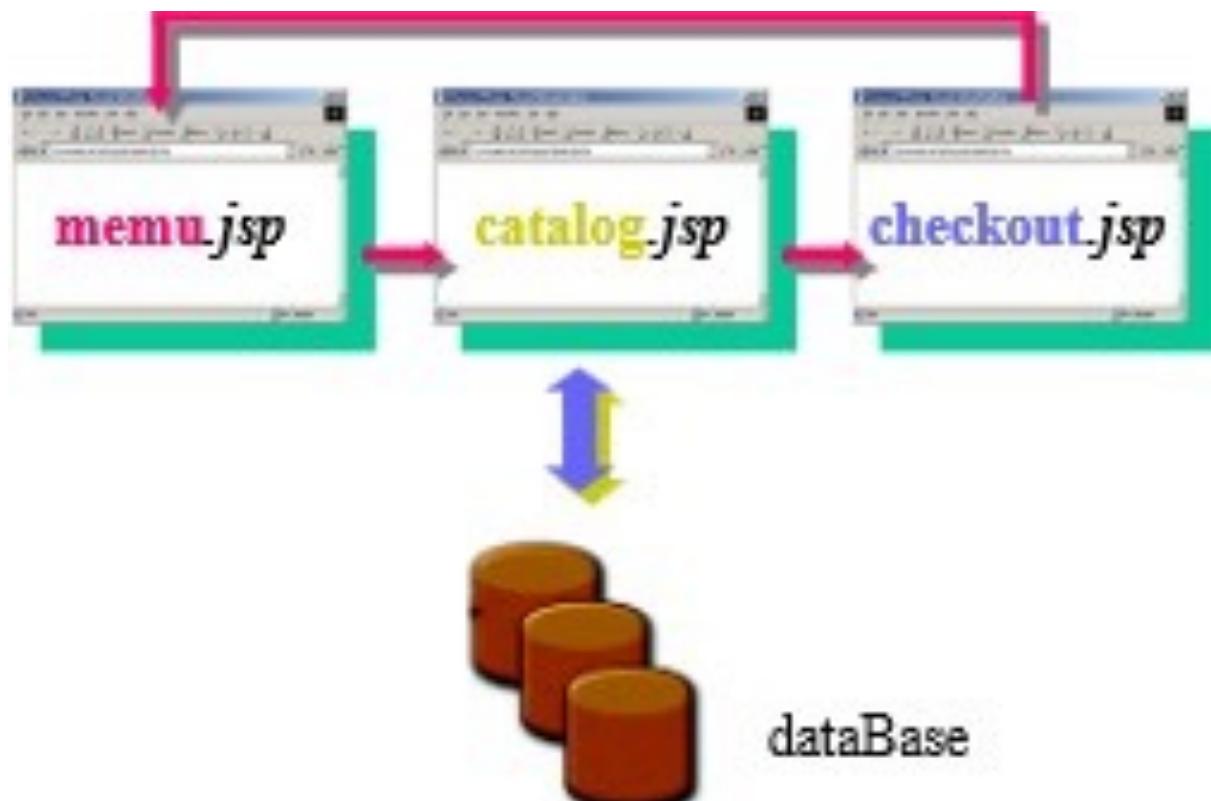
## MVC 1: Page Centric

---

- Bao gồm 1 loạt các trang JSP có liên hệ chặt chẽ với nhau
  - Các trang JSP xử lý tất cả: **presentation, control, và business process**
- **Business process logic** và **control** được **CODE CỨNG** trong các trang JSP
  - Dưới dạng JavaBeans, scriptlets, expression
- Chuyển trang được thực hiện
  - Khi user click vào 1 liên kết. Ví dụ: <A HERF="find.jsp">
  - Qua hành động submit form. Ví dụ: <FORM ACTION="search.jsp">

## MVC 1: Page Centric

---

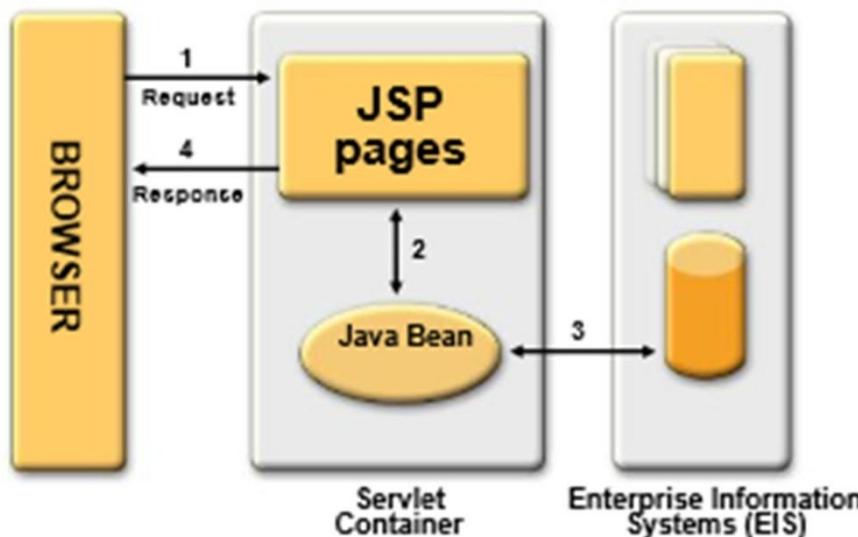


page–centric catalog application

# CÁC MÔ HÌNH ỨNG DỤNG SỬ DỤNG TRONG JSP

## ❑ Mô hình 1

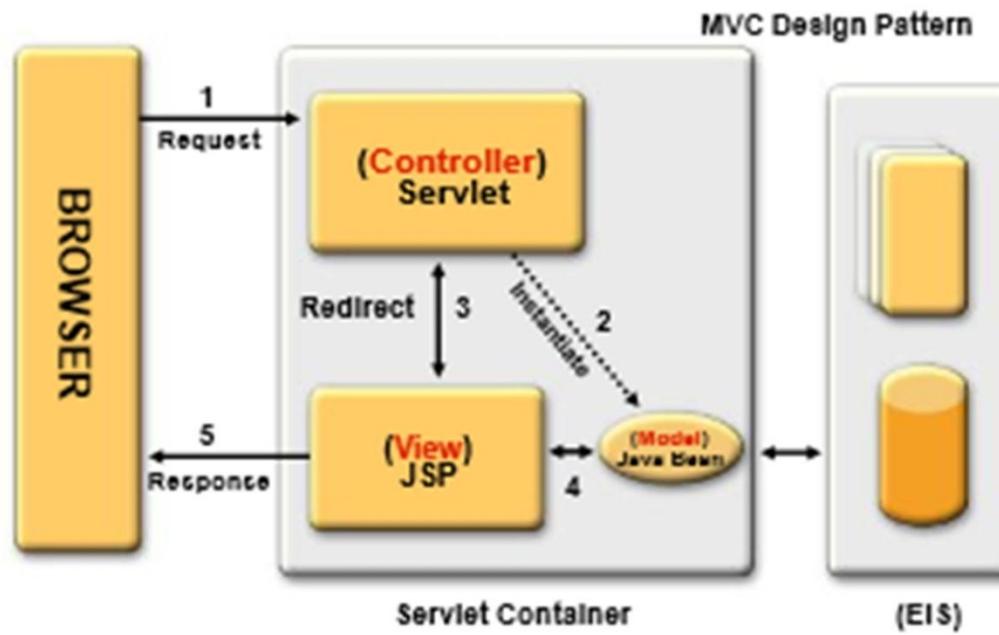
- JSP xử lý request thông qua các thành phần như scriptlet, custom tag và scripting
- Trang JSP dùng JavaBean hay JDBC để truy cập CSDL (nếu có), thực hiện xử lý và trả kết quả (response) cho client.
- Áp dụng cho ứng dụng đơn giản
- Nhược điểm: không linh hoạt và khó khăn khi chỉnh sửa



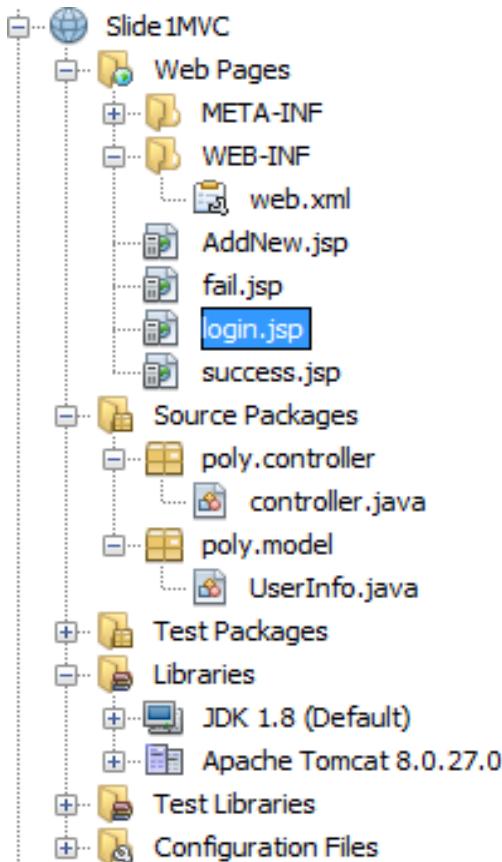
# CÁC MÔ HÌNH ỨNG DỤNG SỬ DỤNG TRONG JSP

## ❑ Mô hình 2

- Phát triển ứng dụng trên cơ chế điều phối xử lý (controller hay dispatcher) tách biệt giữa giao diện (view) và xử lý.
- Việc xử lý request do JSP và servlet thực hiện
- Áp dụng cho ứng dụng lớn và phức tạp.
- Ưu điểm: khả năng tái sử dụng (reuse) và linh hoạt (flexible)



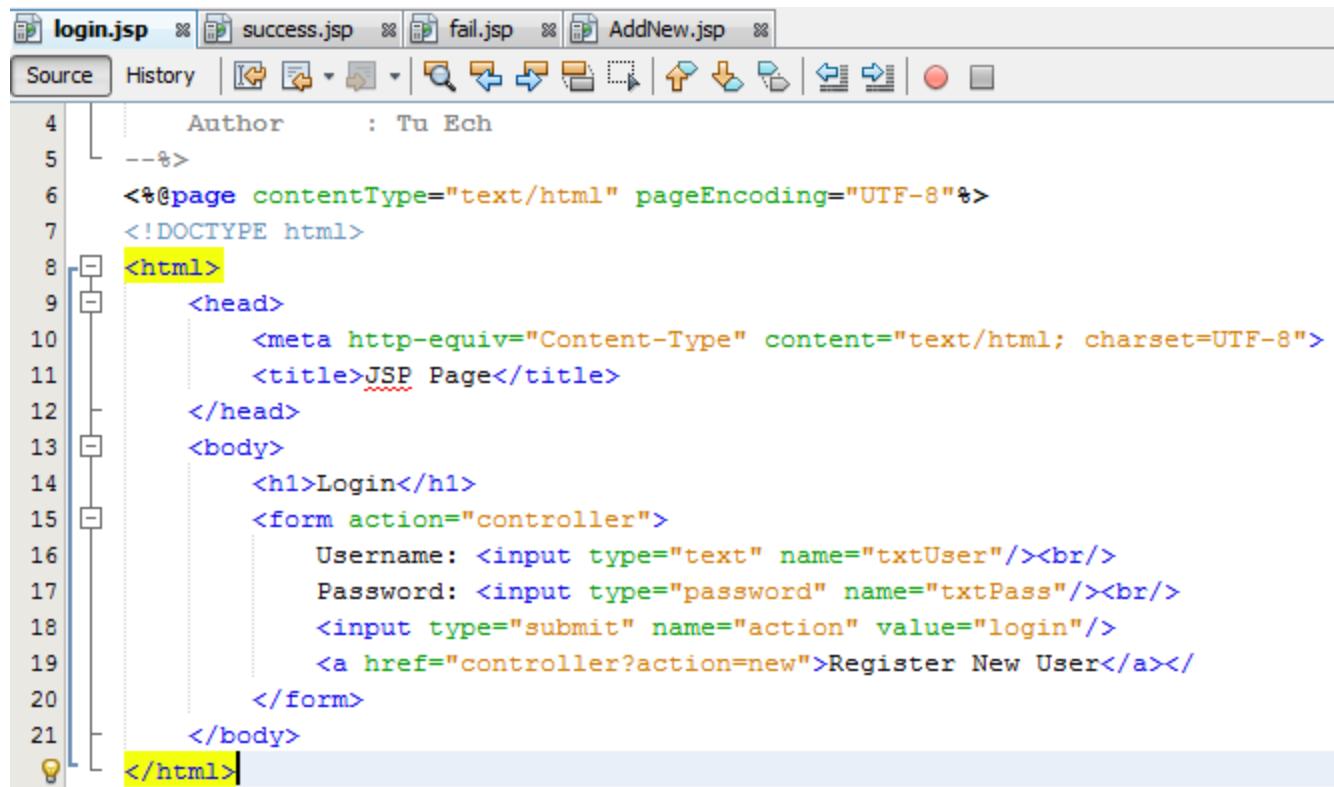
## ❑ Model



```
public class UserInfo {
    private String username;
    private String password;
    public UserInfo() { ...4 lines }
    public UserInfo(String username, String password) {
        this.username = username;
        this.password = password;
    }
    public String getUsername() { ...3 lines }
    public void setUsername(String username) { ...3 lines }
    public String getPassword() { ...3 lines }
    public void setPassword(String password) { ...3 lines }
    public boolean CheckLogin(){
        if(username.equals("fpt") && password.equals("poly"))
            return true;
        return false;
    }
}
```

## VÍ DỤ

### View



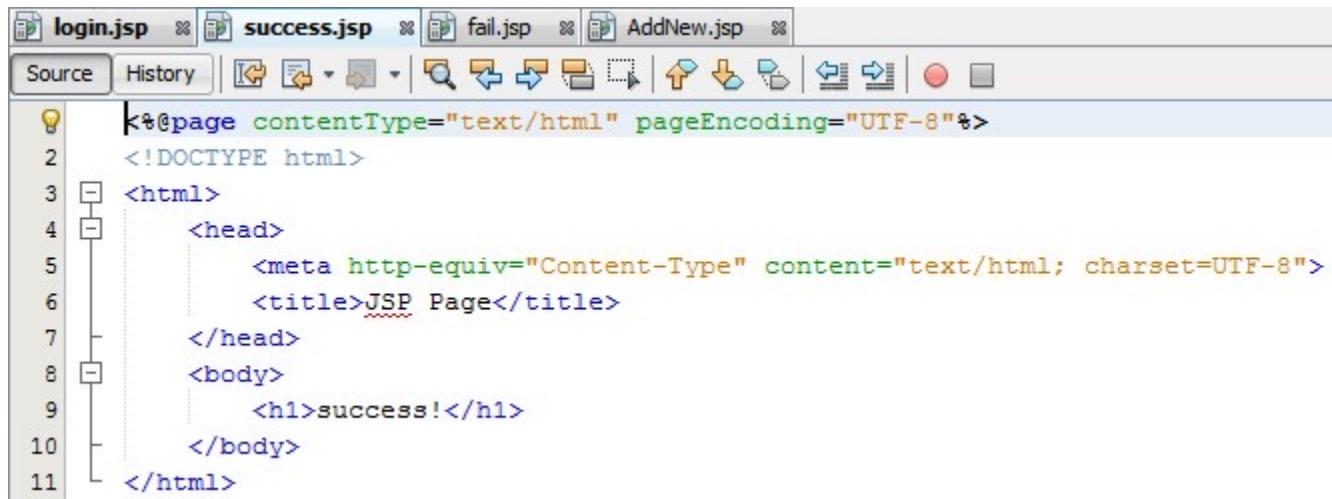
The screenshot shows a Java IDE interface with the 'login.jsp' file open. The code editor displays the following JSP code:

```
Author : Tu Ech
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Login</h1>
        <form action="controller">
            Username: <input type="text" name="txtUser"/><br/>
            Password: <input type="password" name="txtPass"/><br/>
            <input type="submit" name="action" value="login"/>
            <a href="controller?action=new">Register New User</a>
        </form>
    </body>
</html>
```

The code is color-coded for syntax highlighting, and the IDE interface includes tabs for other files like 'success.jsp', 'fail.jsp', and 'AddNew.jsp'. The code editor has a toolbar with various icons for file operations.

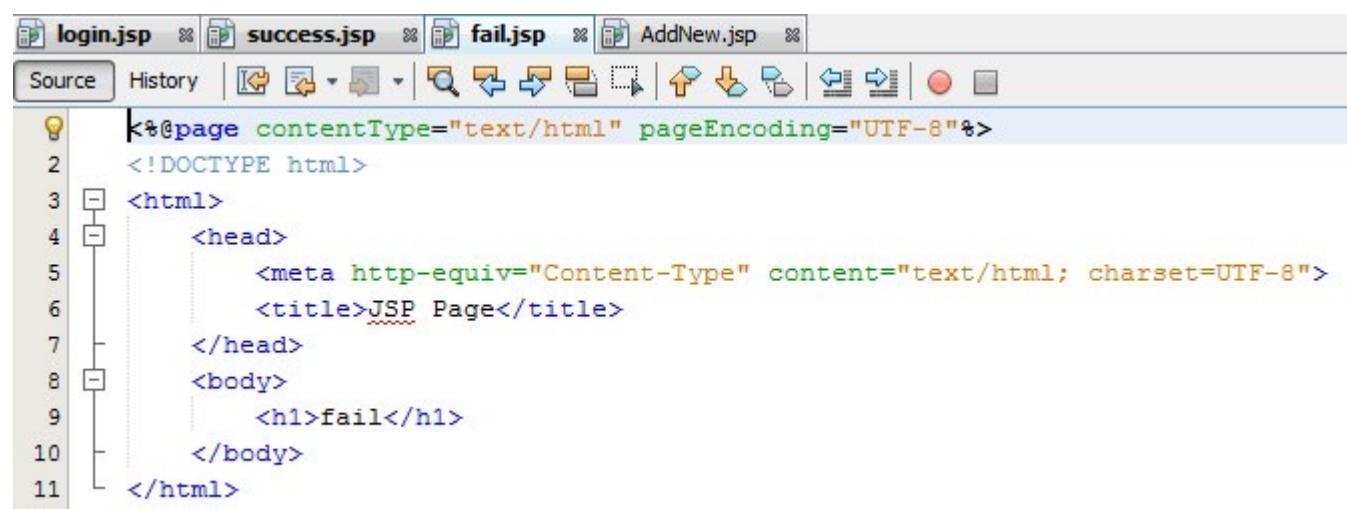
# VÍ DỤ

## □ View



The screenshot shows a Java IDE interface with multiple tabs at the top: login.jsp, success.jsp (which is currently selected), fail.jsp, and AddNew.jsp. Below the tabs is a toolbar with various icons. The main area displays the source code for success.jsp:

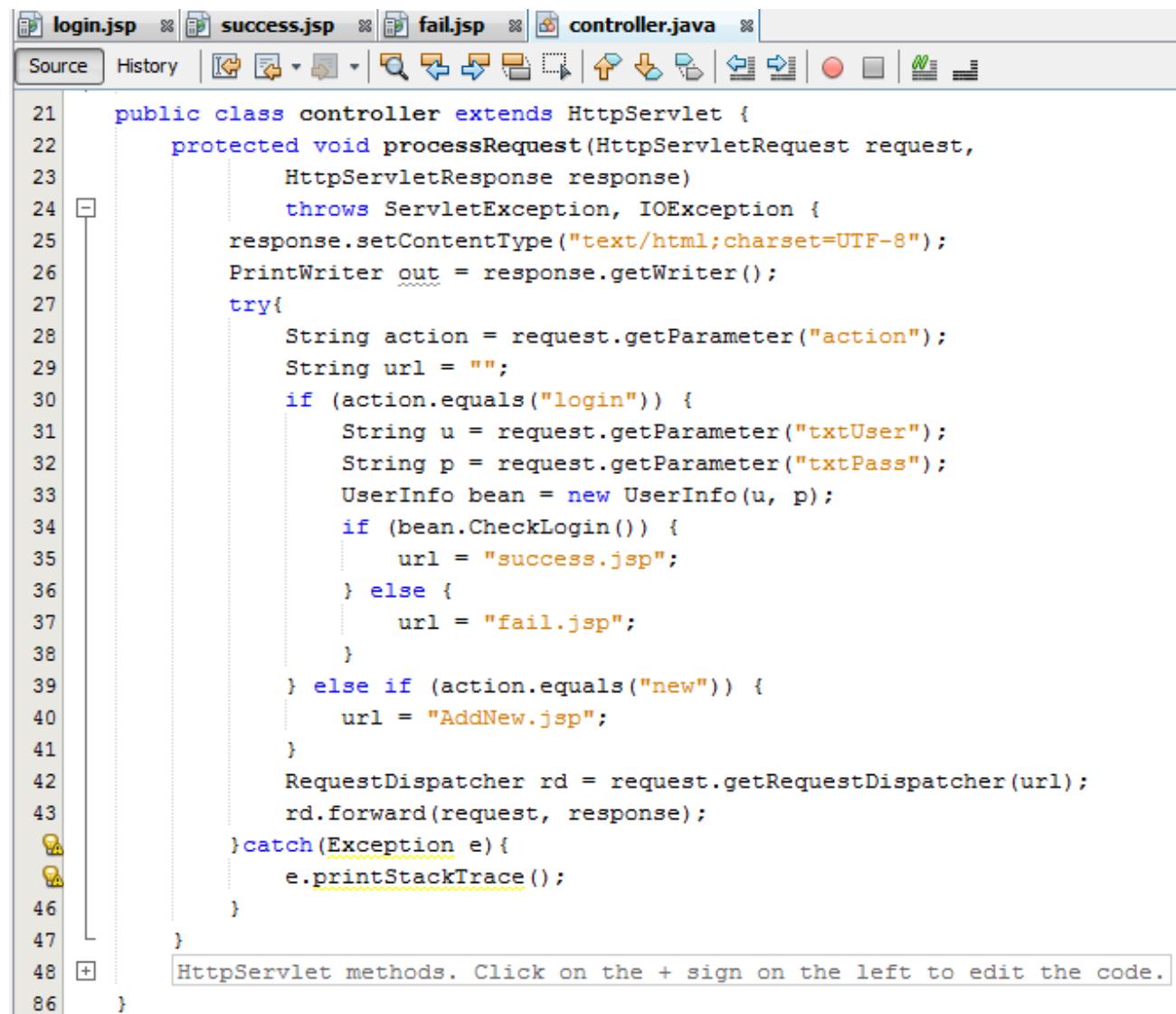
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>success!</h1>
    </body>
</html>
```



The screenshot shows the same Java IDE interface with the tabs: login.jsp, success.jsp, fail.jsp (selected), and AddNew.jsp. The main area displays the source code for fail.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>fail</h1>
    </body>
</html>
```

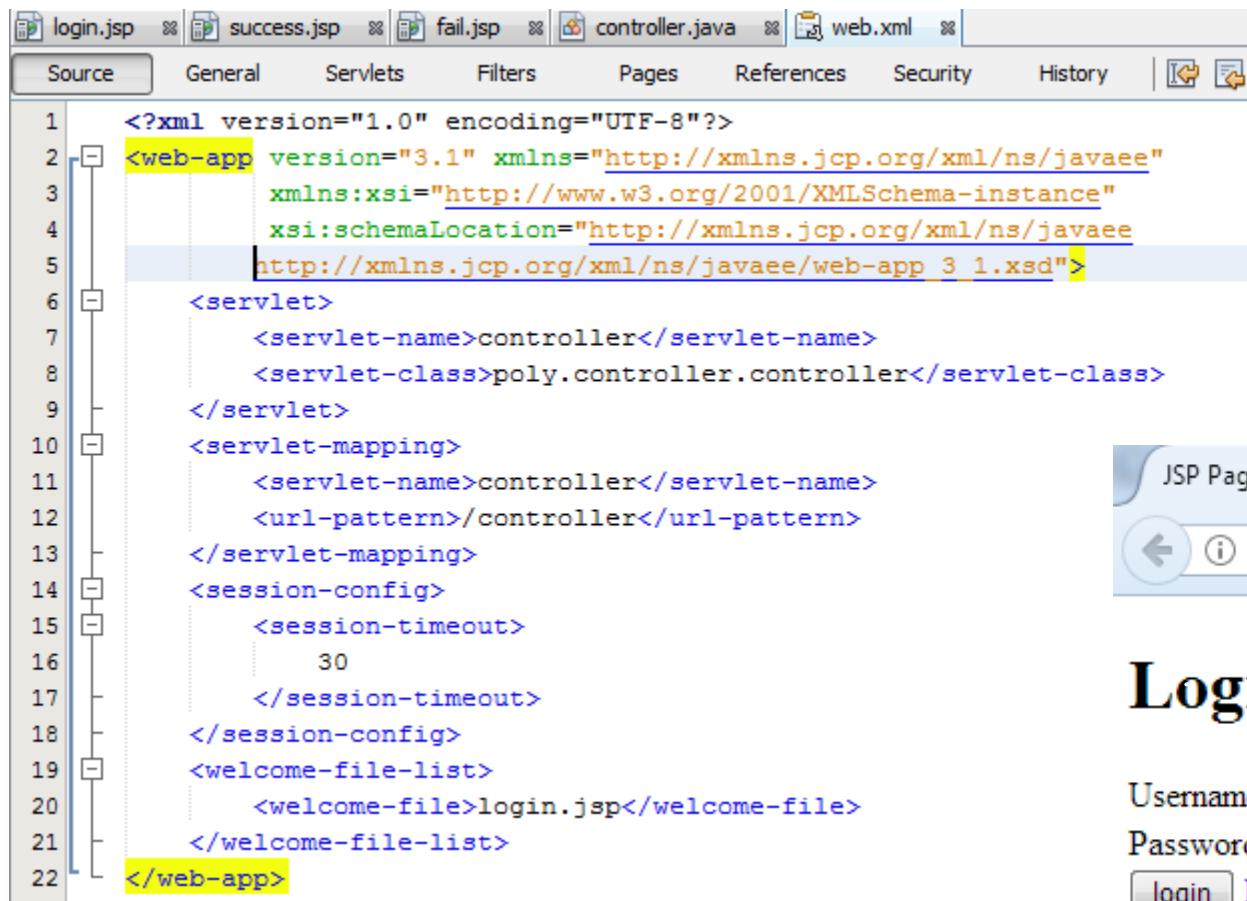
## Controller



The screenshot shows a Java code editor with the file `controller.java` open. The code implements a `HttpServlet` to handle requests for login and new user addition, and to forward them to corresponding JSP pages. It also handles exceptions by printing the stack trace.

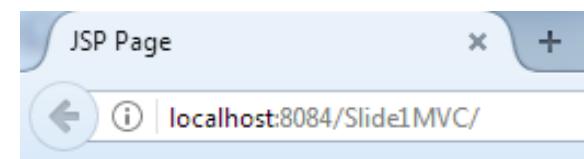
```
21  public class controller extends HttpServlet {
22      protected void processRequest(HttpServletRequest request,
23                                     HttpServletResponse response)
24                                     throws ServletException, IOException {
25         response.setContentType("text/html;charset=UTF-8");
26         PrintWriter out = response.getWriter();
27         try{
28             String action = request.getParameter("action");
29             String url = "";
30             if (action.equals("login")) {
31                 String u = request.getParameter("txtUser");
32                 String p = request.getParameter("txtPass");
33                 UserInfo bean = new UserInfo(u, p);
34                 if (bean.CheckLogin()) {
35                     url = "success.jsp";
36                 } else {
37                     url = "fail.jsp";
38                 }
39             } else if (action.equals("new")) {
40                 url = "AddNew.jsp";
41             }
42             RequestDispatcher rd = request.getRequestDispatcher(url);
43             rd.forward(request, response);
44         }catch(Exception e){
45             e.printStackTrace();
46         }
47     }
48
49     // ...
50
51     /**
52      * HttpServlet methods. Click on the + sign on the left to edit the code.
53     */
54 }
```

## □ web.xml



The screenshot shows an IDE interface with several tabs at the top: login.jsp, success.jsp, fail.jsp, controller.java, and web.xml. The web.xml tab is active, displaying its XML code. The code defines a web application with a servlet named 'controller' that maps to the class 'poly.controller.controller'. It also sets a session timeout of 30 minutes and specifies 'login.jsp' as the welcome file.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <servlet>
        <servlet-name>controller</servlet-name>
        <servlet-class>poly.controller.controller</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>controller</servlet-name>
        <url-pattern>/controller</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>login.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```



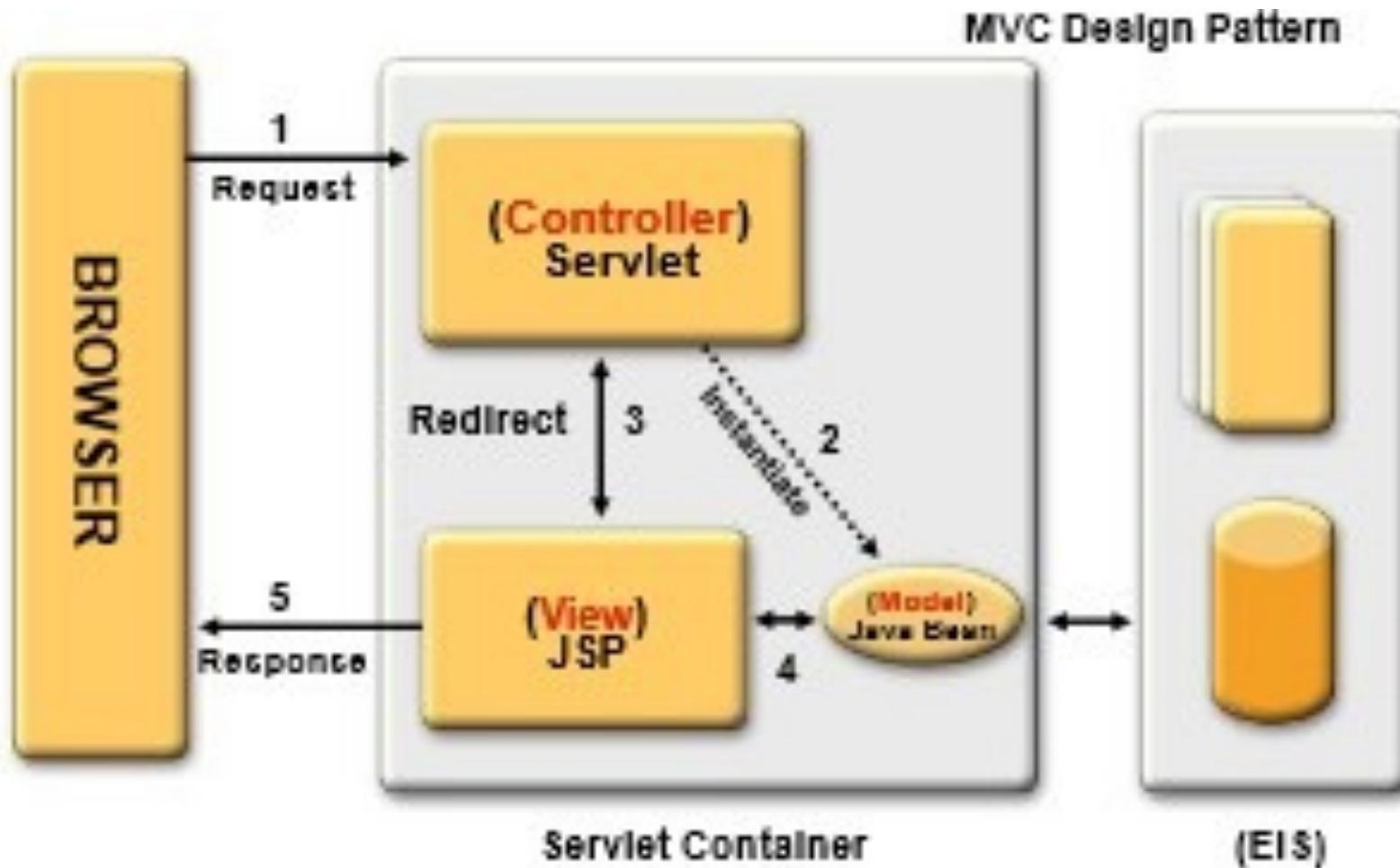
## Login

Username:

Password:  ••••

[Register New User](#)

## MVC 2: Server Centric



## MVC 2: Server Centric

---

- Nếu muốn biểu diễn các trang JSP khác nhau, tùy theo dữ liệu nhận được?
  - Riêng JSP với JavaBeans và custom tags (Model 1) chưa xử lý tốt được
- Giải pháp
  - Sử dụng đồng thời Servlet và JSP (Model 2)
  - Servlet xử lý request gửi tới, xử lý 1 phần dữ liệu, thiết lập các beans, forward kết quả cho 1 trong nhiều trang JSP nào đó

## MVC 2: Server Centric

---

- JSP chỉ được sử dụng để biểu diễn kết quả (**presentation**)
  - Xử lý điều khiển (Control) thực hiện bởi servlets
- Servlet hoạt động như một **gatekeeper**
  - Cung cấp các services thông dụng, như **authentication**, **authorization**, **login**, **error handling**, ...
- Servlet hoạt động như một **central controller**
  - Quyết định logic phù hợp để xử lý các request, sẽ gửi request đến những nơi nào, ...
  - Thực hiện việc điều hướng (redirecting)

## Web Application Framework

---

- Dựa trên kiến trúc MVC Model 2
- Hầu hết các ứng dụng Web phải cung cấp các chức năng
  - Nhận (**receive**) và gửi tiếp (**Dispatching**) HTTP requests
  - Gọi các phương thức từ tầng model
  - Tổng hợp và chọn ra các views trả về cho client
- Cung cấp các classes và interfaces cho lập trình viên sử dụng/mở rộng

## Web Application Framework

---

- Phân tách tầng presentation và các business logic thành các components
- Cung cấp 1 điểm điều khiển trung tâm
- Cung cấp các tính năng mở rộng
- Dễ dàng kiểm thử unit (**unit-testing**) và bảo trì
- Nhiều công cụ hỗ trợ
- Ổn định
- Có cộng đồng hỗ trợ mạnh mẽ
- Đơn giản hóa chế độ đa ngôn ngữ (internationalization)
- Đơn giản hóa việc validate đầu vào

## Web Application Framework

---

- Frameworks đang phát triển mạnh mẽ
- JSP/Servlets vẫn còn khó sử dụng
- Frameworks định nghĩa các components chuẩn, cho phép tái sử dụng.
- Frameworks còn chỉ rõ cách thức phối hợp các components trong 1 ứng dụng

# Một số Web Application Frameworks

---

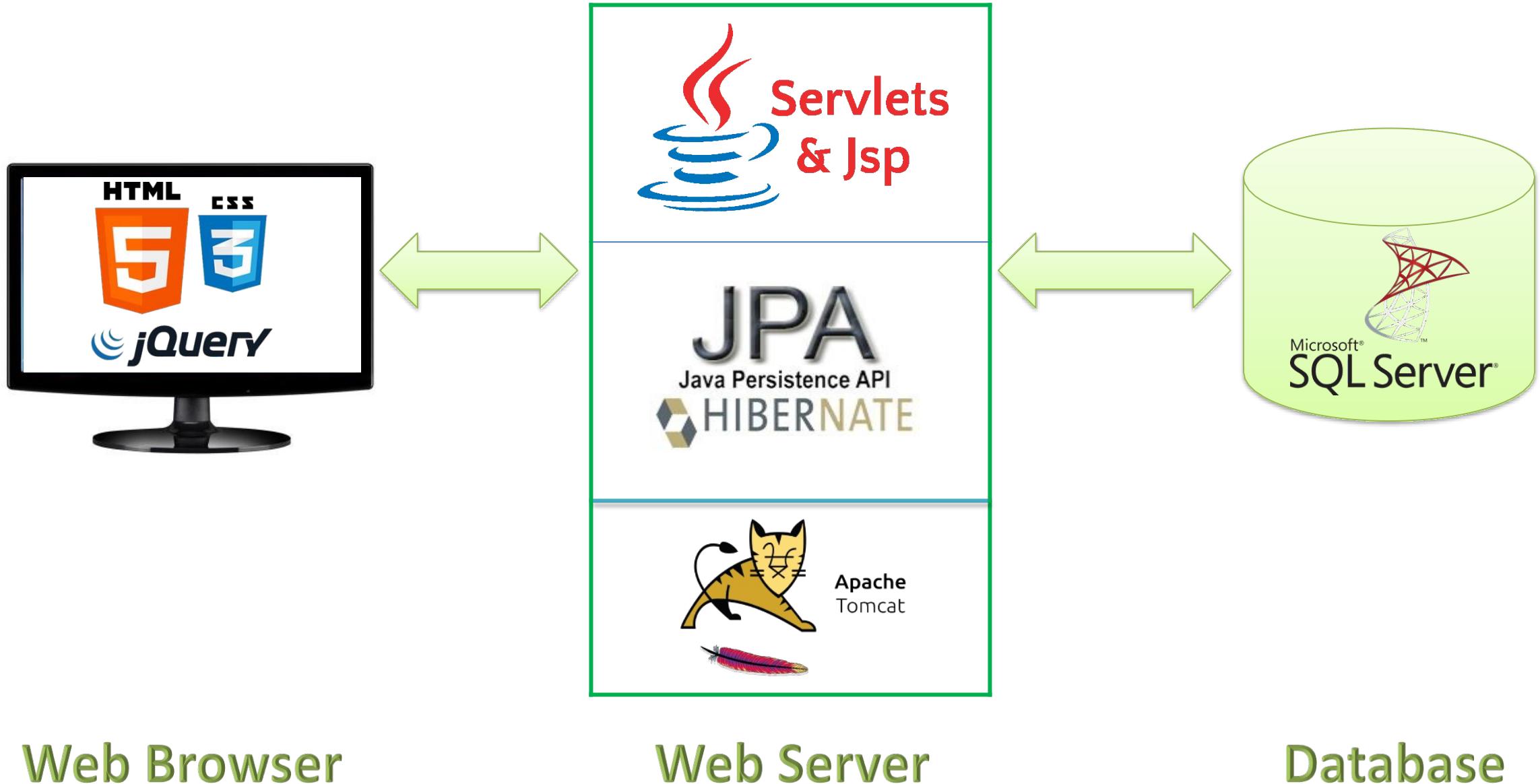
- Apache Struts I and II
- Spring Framework MVC
- JavaServer Faces
- Echo
- Tapestry
- Wicket
- ...

# GIỚI THIỆU MÔ HÌNH 3 LỚP (THREE TIER MODEL)

---



# GIỚI THIỆU CÔNG NGHỆ ÁP DỤNG CHO MÔN HỌC



# GIỚI THIỆU CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN

---



DB Server Web Server



Web Browser

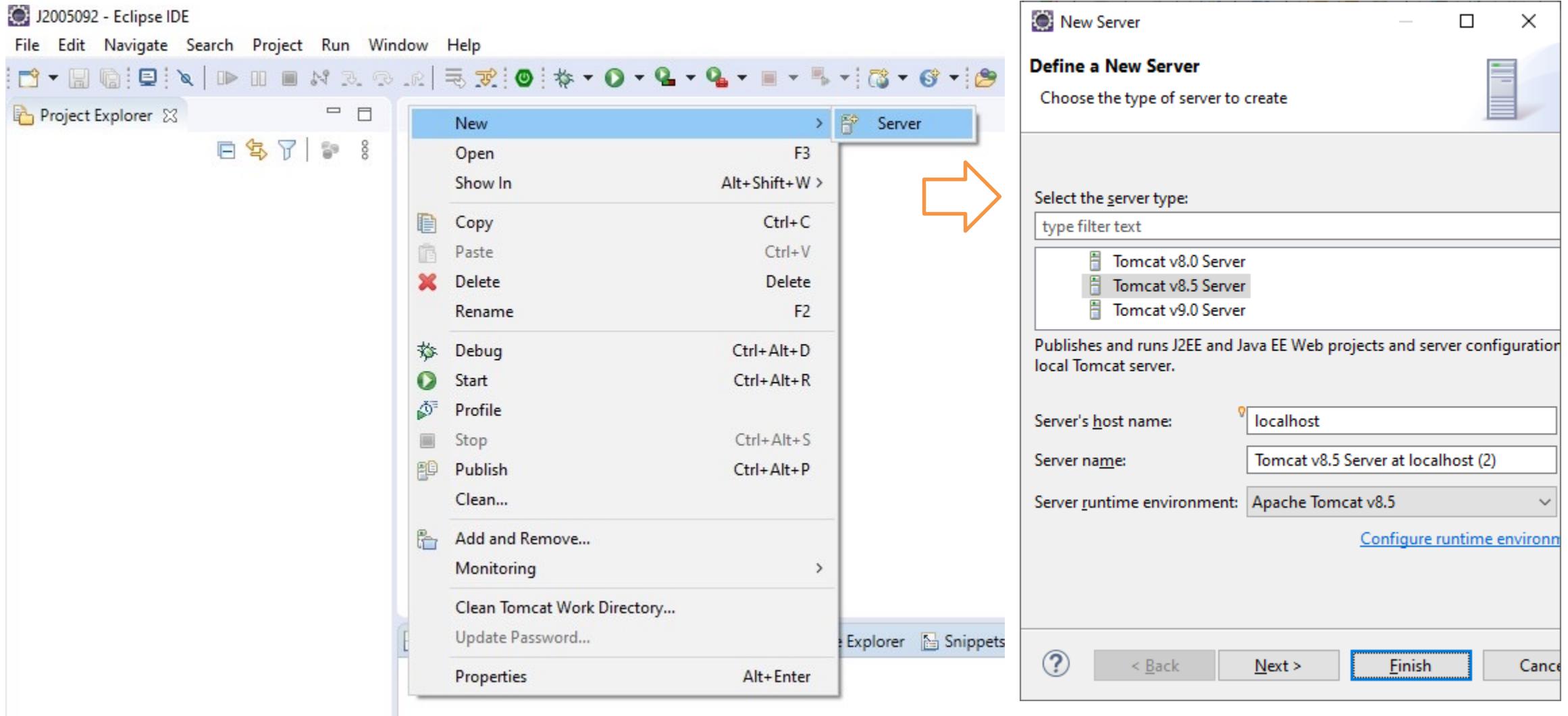


## TÍCH HỢP TOMCAT VÀO ECLIPSE

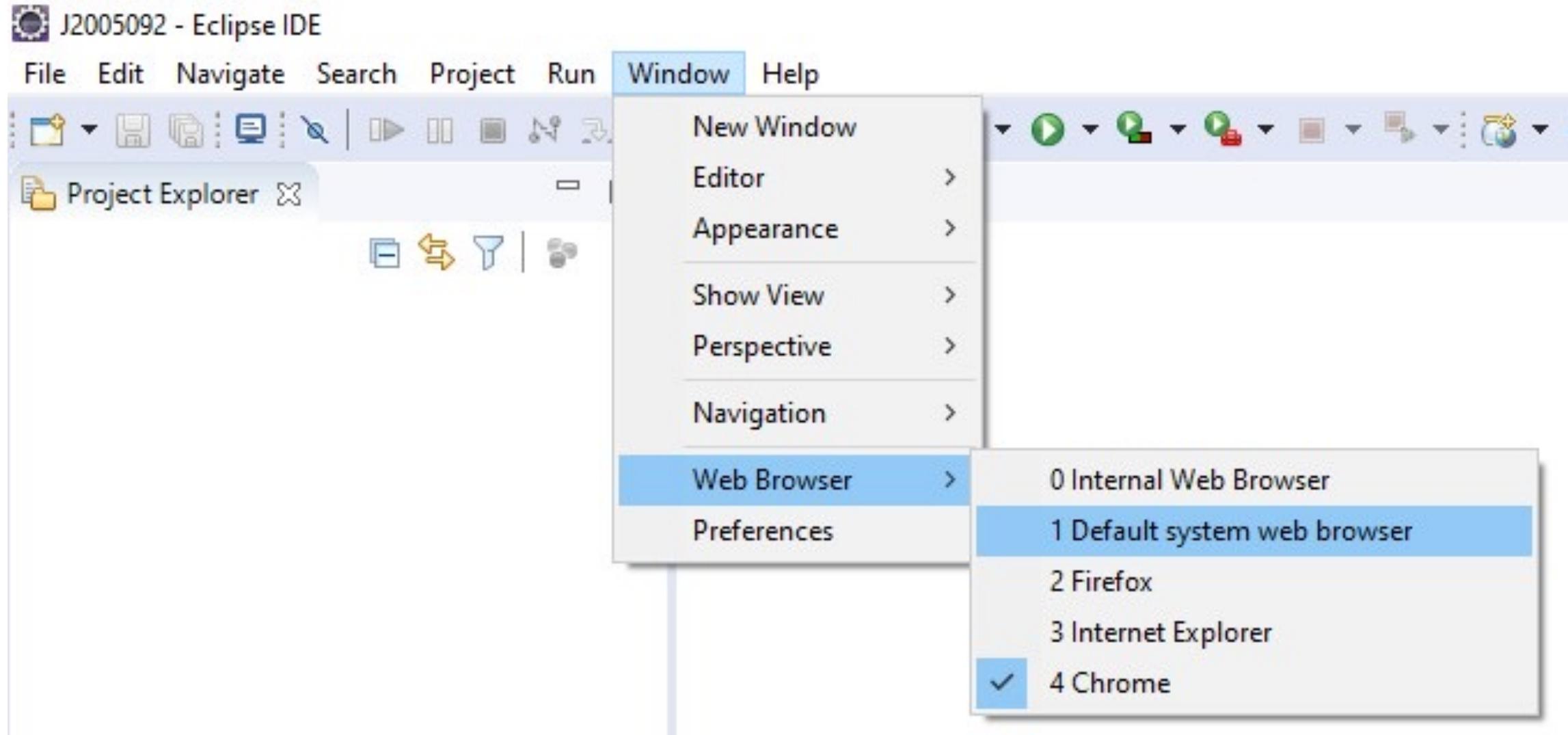
---

- ❑ Tomcat là Web Server, Eclipse là công cụ phát triển ứng dụng web.  
Ứng dụng web sau khi phát triển cần đóng gói và triển khai vào Tomcat mới có thể chạy được
- ❑ Tích hợp Tomcat vào eclipse là để đơn giản hóa quy trình nói trên.  
Không cần phải đóng gói ứng dụng mà ta có thể chạy ứng dụng ngay lúc đang phát triển
- ❑ Tích hợp sẽ giúp chúng ta:
  - ❖ Dễ dàng trong debug
  - ❖ Đơn giản việc test nhanh ứng dụng

# TÍCH HỢP TOMCAT VÀO ECLIPSE

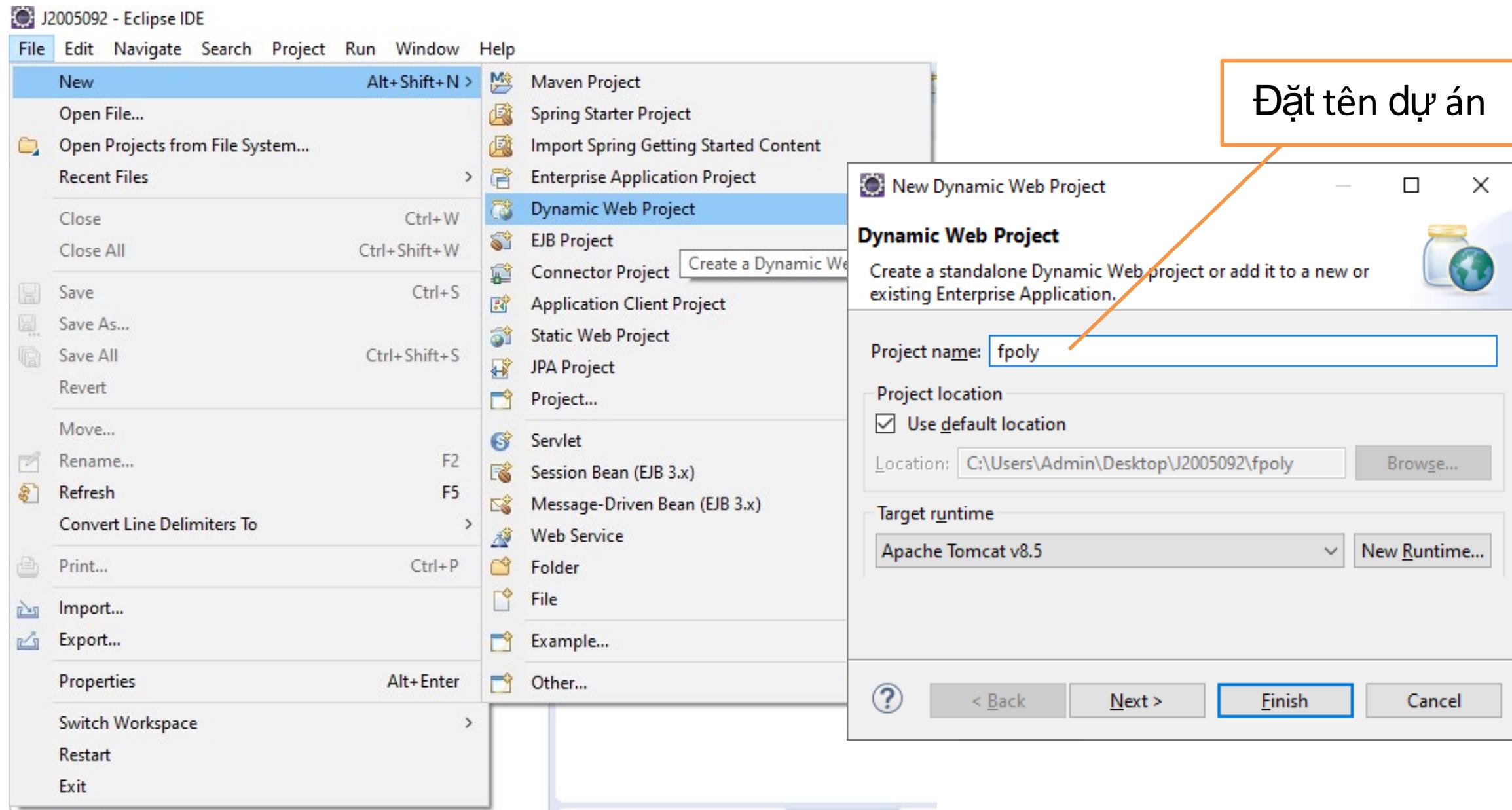


# CHỌN TRÌNH DUYỆT NGOÀI

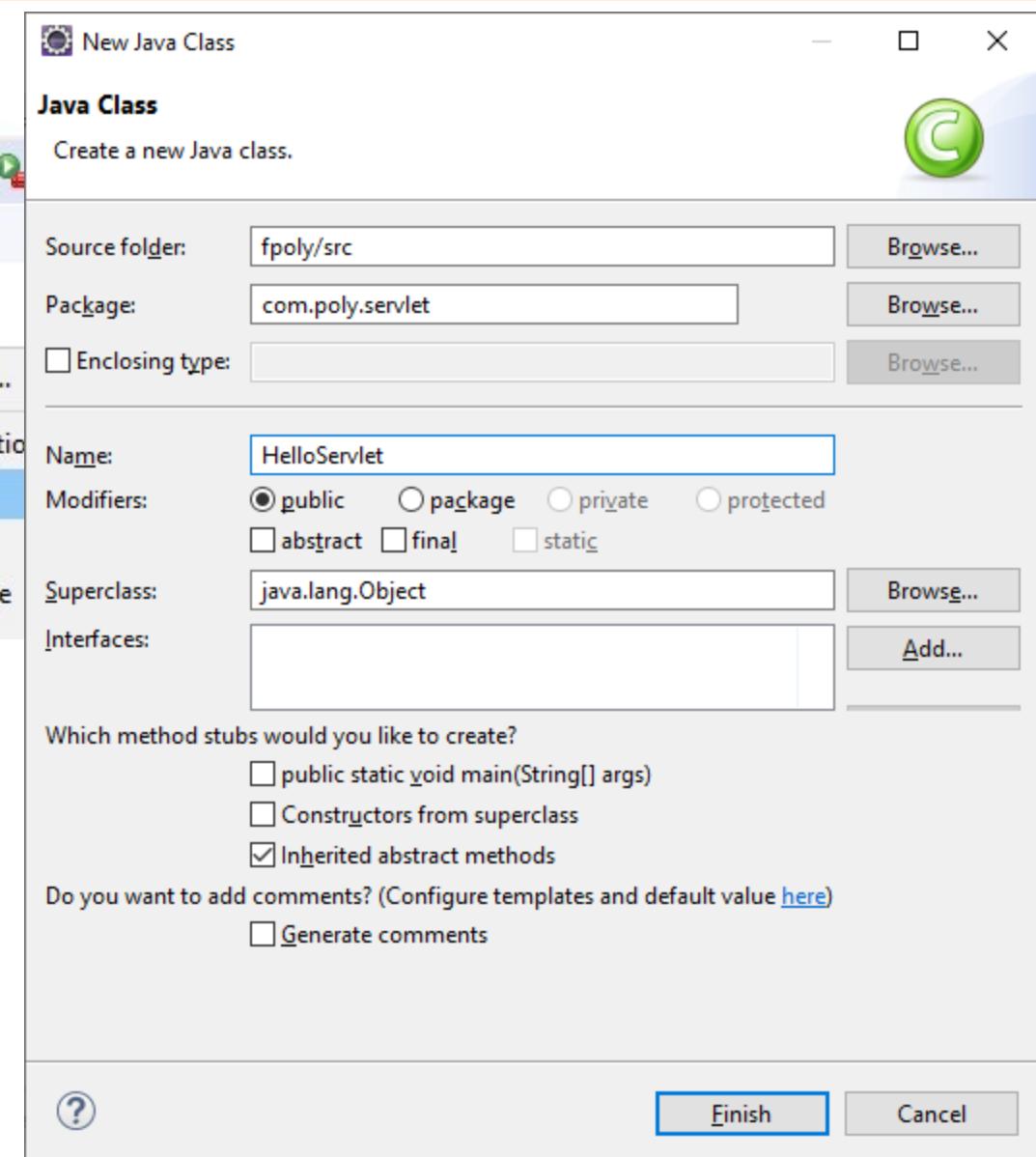
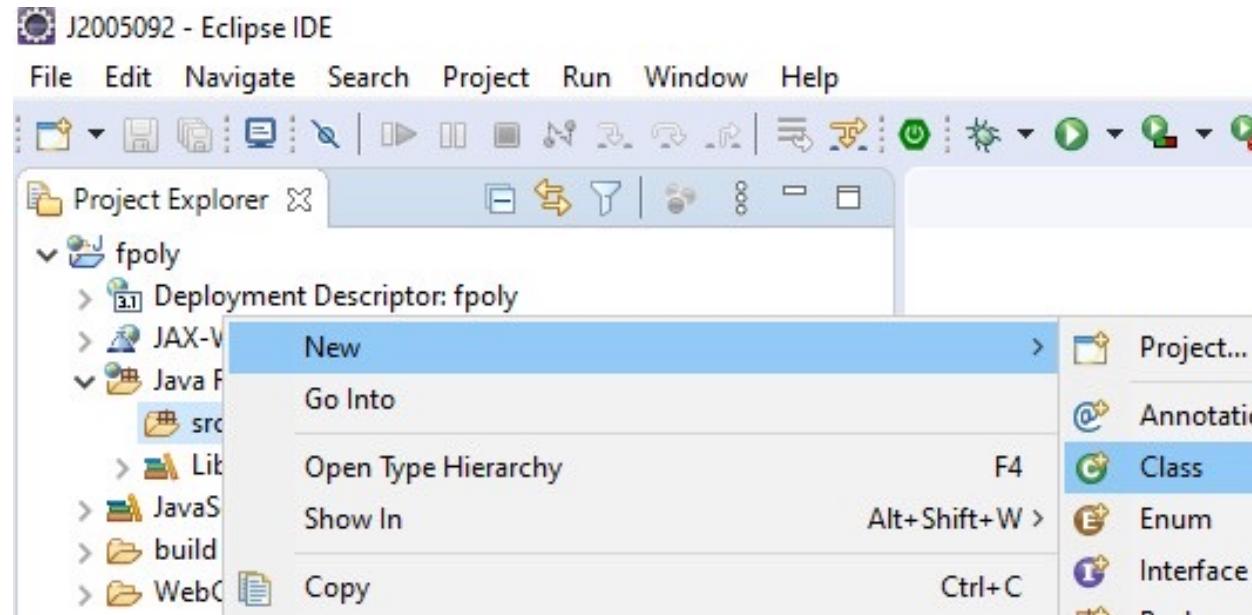


*Eclipse cung cấp sẵn trình duyệt nội bộ, trình duyệt này không đủ mạnh để hỗ trợ JS và CSS*

# TAO DỰ ÁN WEB ĐỘNG

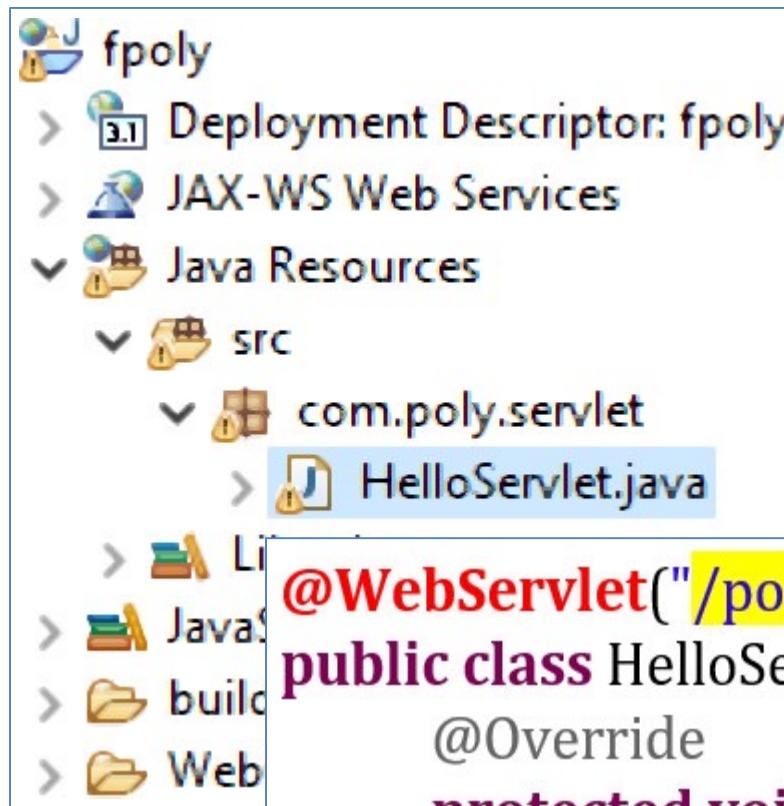


# TẠO LỚP MỚI



## Đặt tên

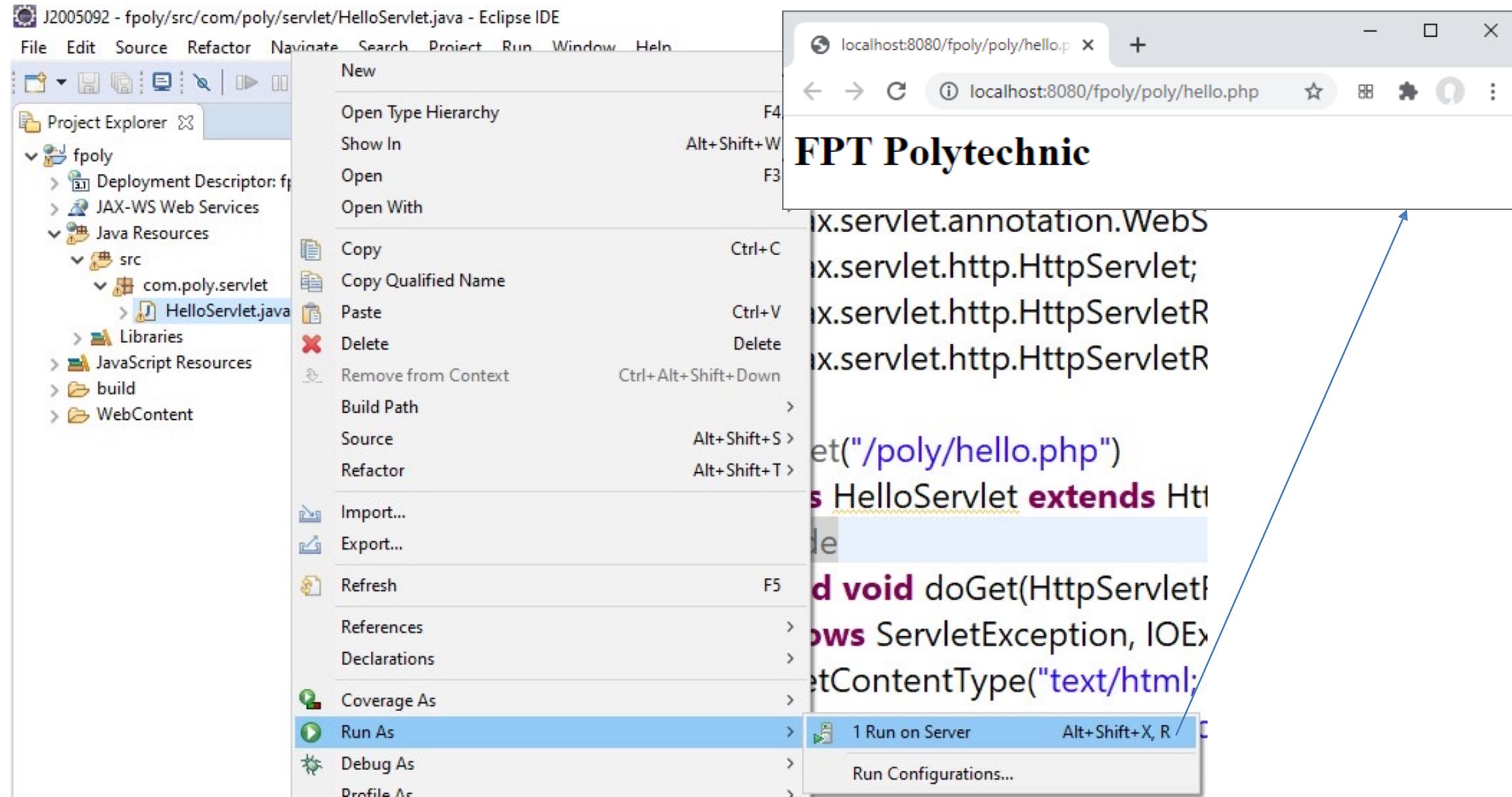
- ❖ Package: com.poly.servlet
- ❖ Class: HelloServlet



- @WebServlet("/poly/hello.php")
- extends HttpServlet
- doGet(...)
- resp.getWriter().print(...)

```
@WebServlet("/poly/hello.php")
public class HelloServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        resp.getWriter().println("<h1>FPT Polytechnic</h1>");
    }
}
```

# CHẠY SERVLET



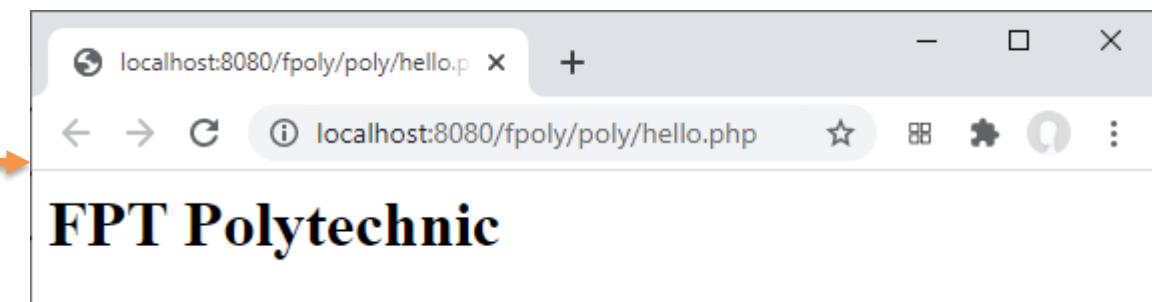
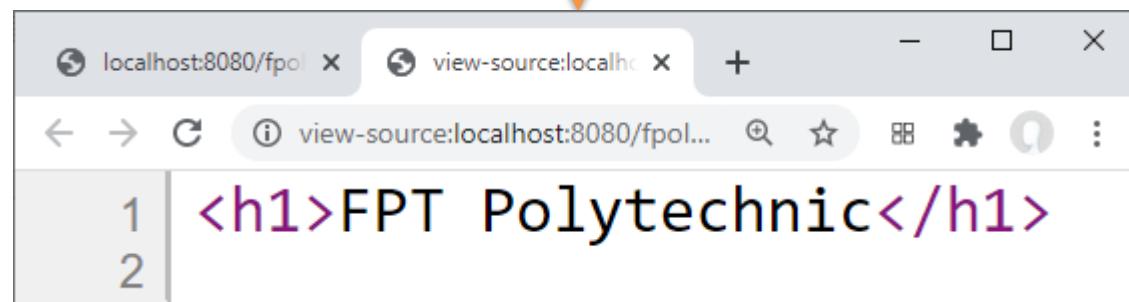
# GIẢI THÍCH QUY TRÌNH XỬ LÝ REQUEST

http://localhost:8080/fpoly/poly/hello.php

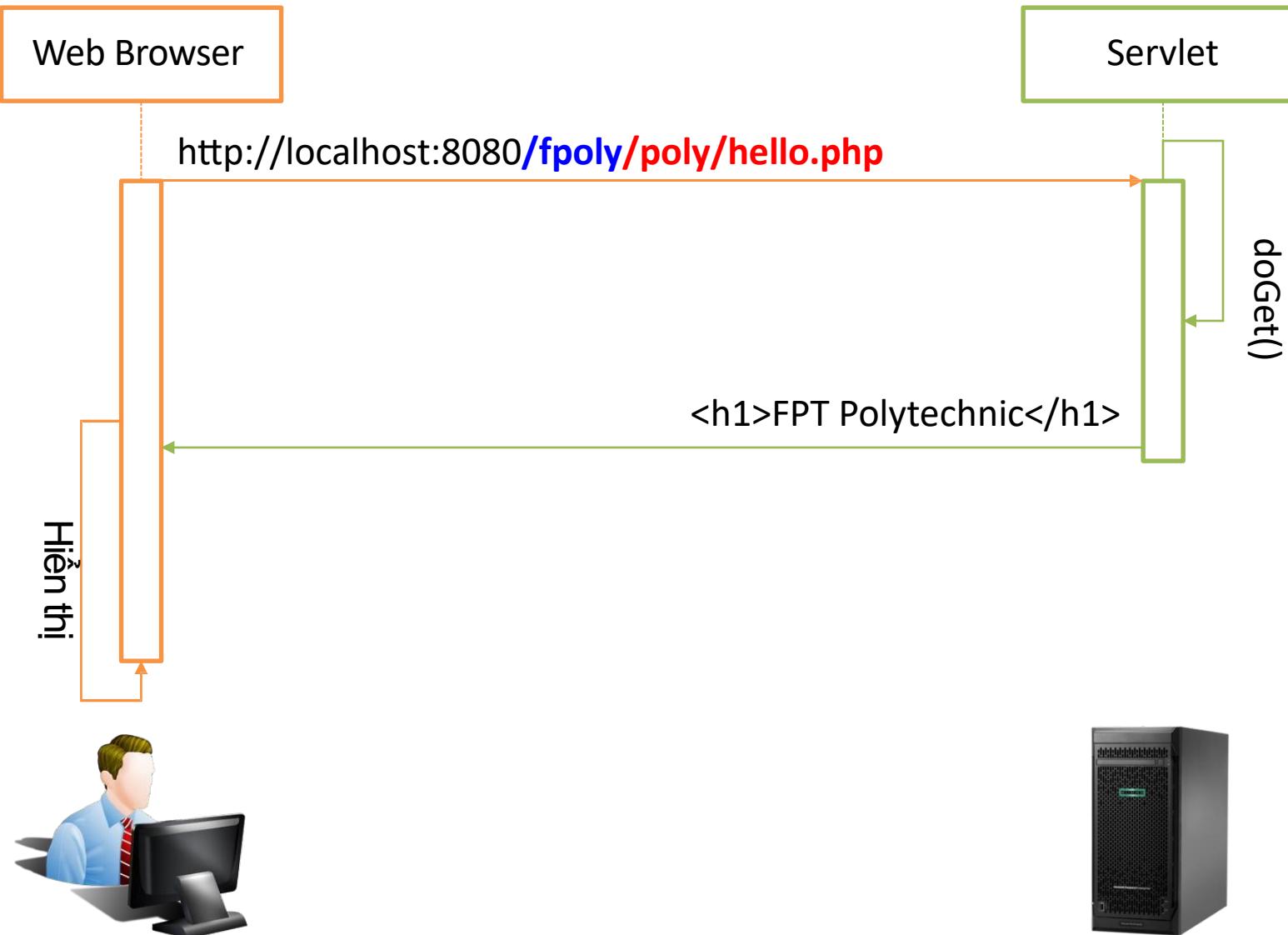
Request

```
@WebServlet("/poly/hello.php")
public class HelloServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        resp.getWriter().println("<h1>FPT Polytechnic</h1>");
    }
}
```

Response



# GIẢI THÍCH QUY TRÌNH XỬ LÝ REQUEST

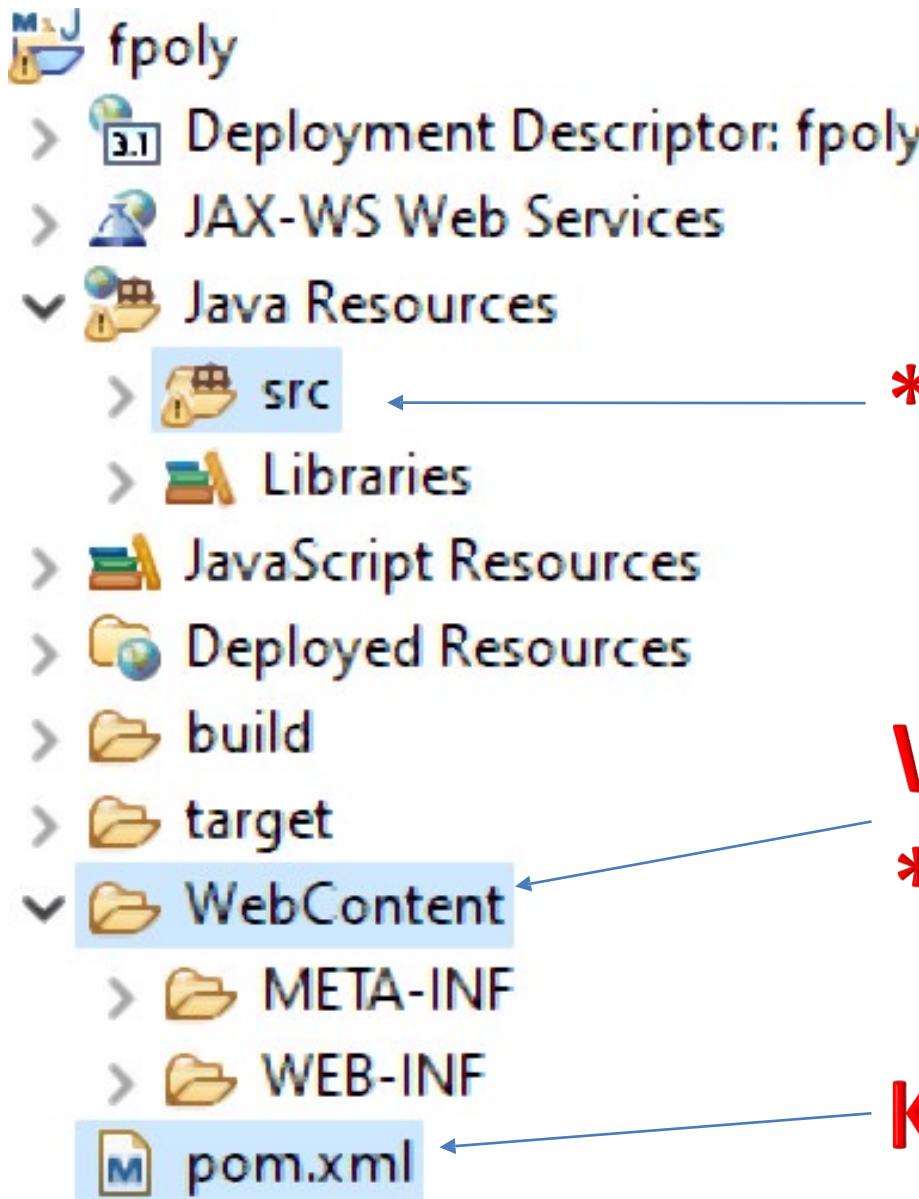


②

## DYNAMIC WEB PROJECT

---

# KHẢO SÁT CẤU TRÚC TỔ CHỨC DYNAMIC WEB PROJECT

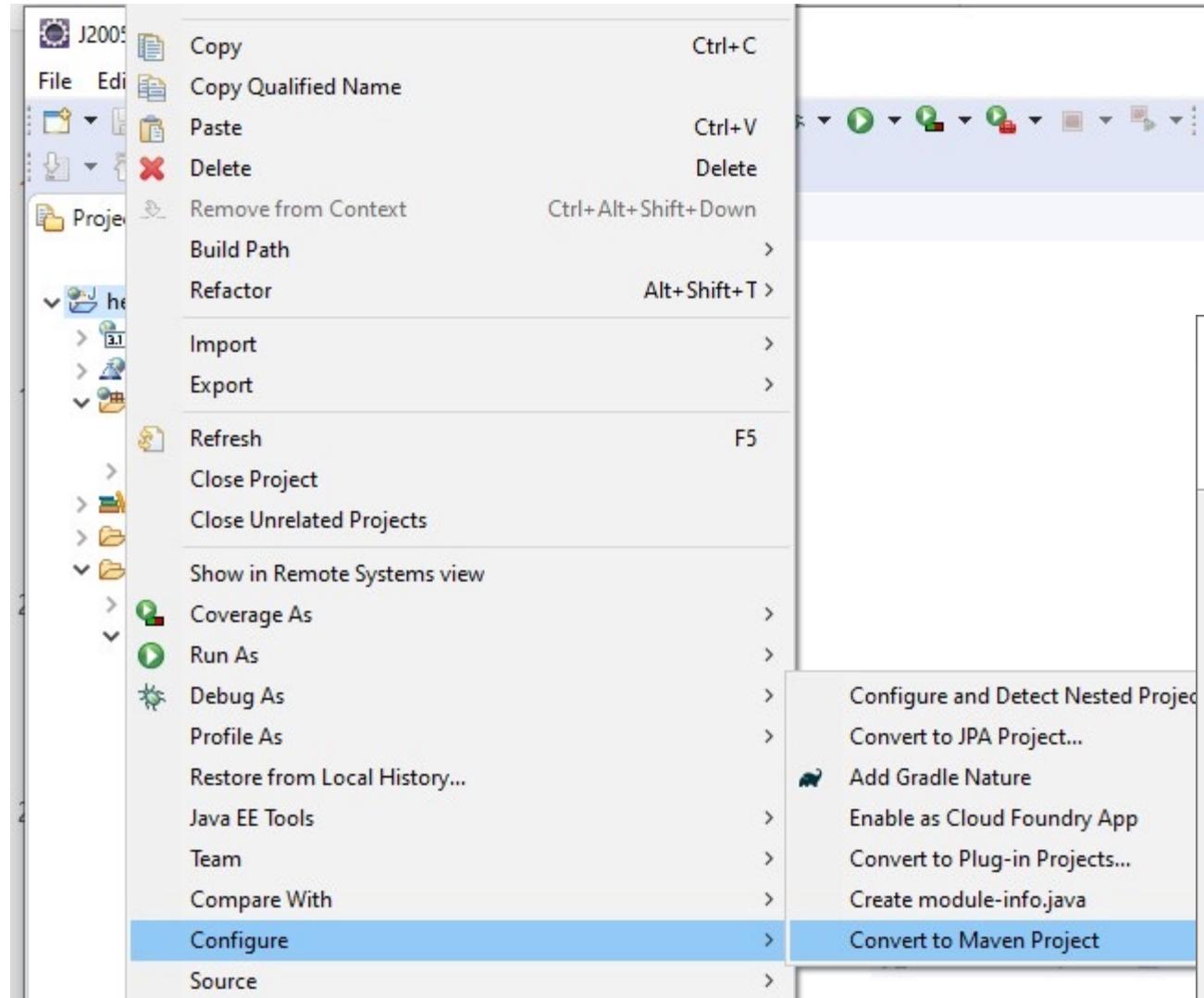


**\*.java: mã nguồn java**

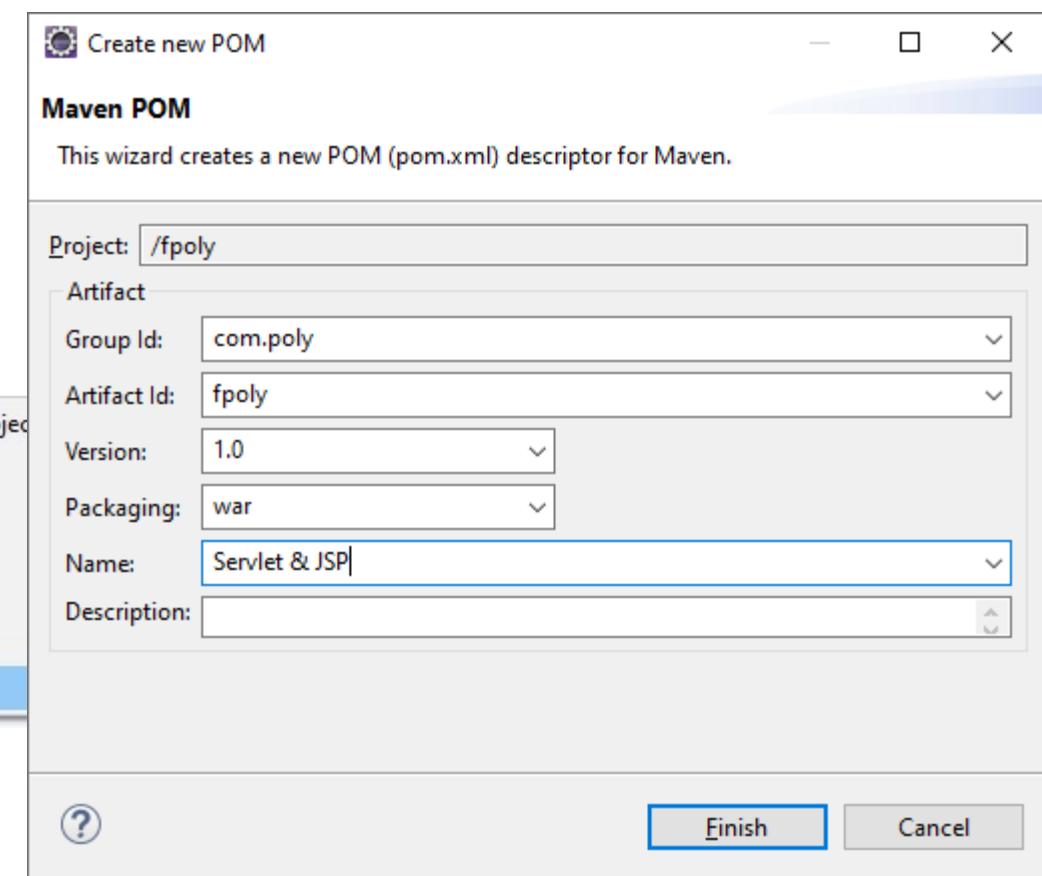
**Webroot: (css, js, images...)**  
**\*.jsp**

**Khai báo thư viện cần thiết**

# TÍCH HỢP VỚI MAVEN



☐ Maven giúp quản lý thư viện cần thiết của dự án

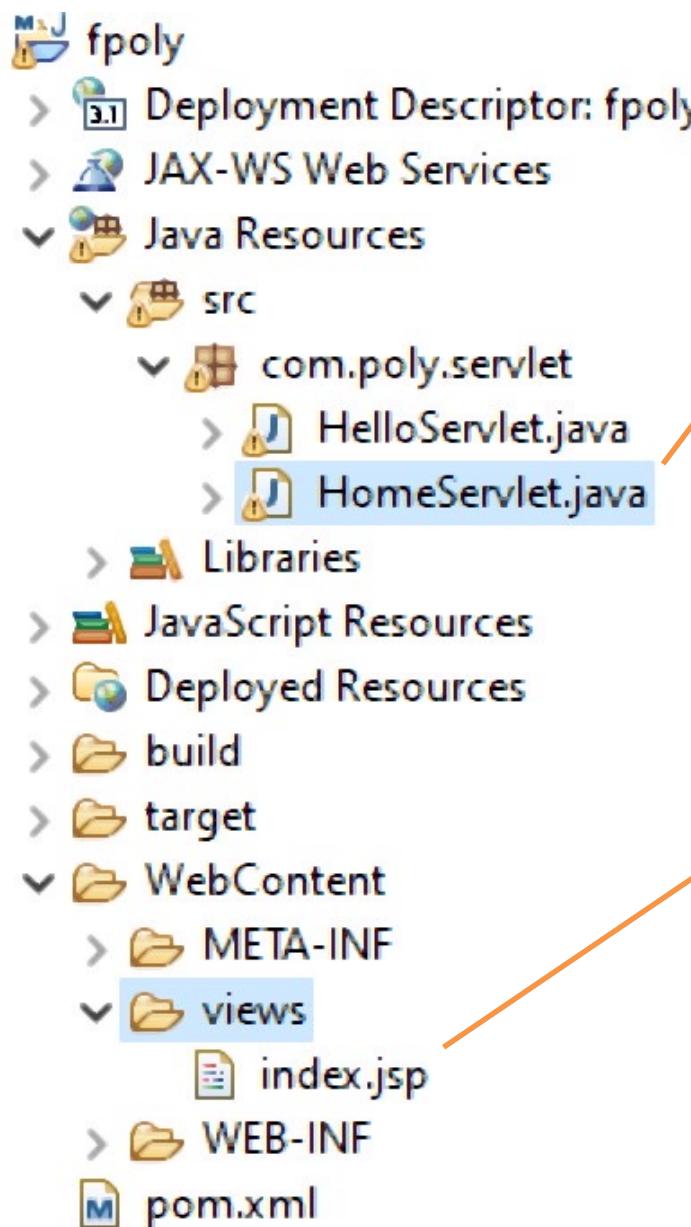


## POM.XML – KHAI BÁO THƯ VIỆN CẦN THIẾT

```
<packaging>war</packaging>
<dependencies>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
</dependencies>
<build>
    <sourceDirectory>src</sourceDirectory>
```

- Khai báo để nạp các thư viện cần thiết cho dự án từ Internet
  - ❖ JSTL
  - ❖ BeanUtils
  - ❖ Java Mail
  - ❖ Database Driver
  - ❖ ...

# TỔ CHỨC WEB DYNAMIC PROJECT



Nơi đặt Servlet  
Thêm HomeServlet vào đây

Nơi đặt JSP  
Thêm index.jsp vào đây

# TỔ CHỨC MÃ HOMESERVLET.JAVA

```
package com.poly.servlet;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/home/index")
public class HomeServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        req.getRequestDispatcher("/views/index.jsp").forward(req, resp);
    }
}
```

## ❑ **@WebServlet**

- ❖ @WebServlet("url")

## ❑ **HttpServlet**

- ❖ **doGet(HttpServletRequest, HttpServletResponse)**

## ❑ **HttpServletRequest**

- ❖ getRequestDispatcher(view)



## TỔ CHỨC MÃ INDEX.JSP

---

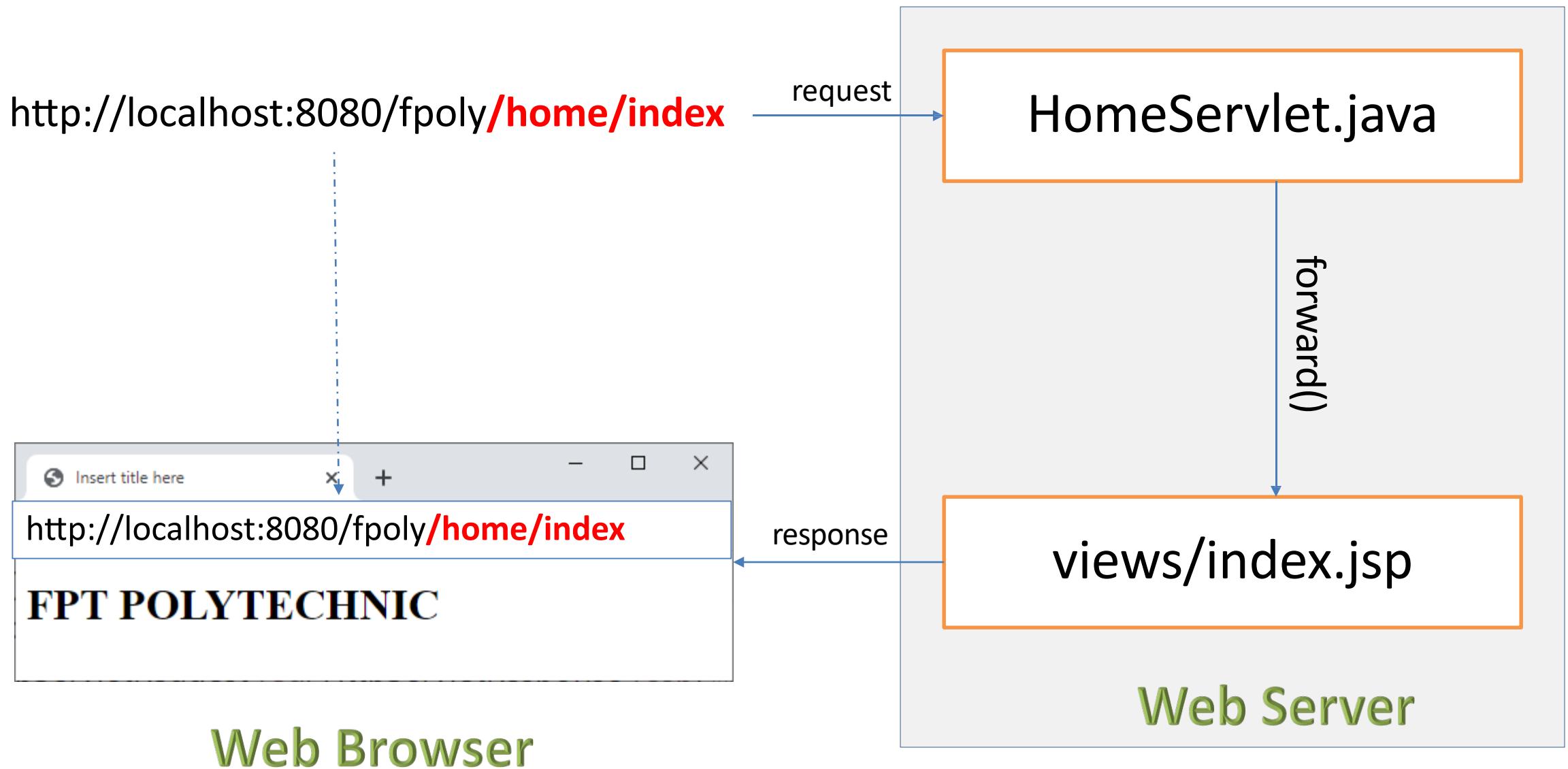
```
<%@ page pageEncoding="utf-8"%>
```

} Tiếng việt

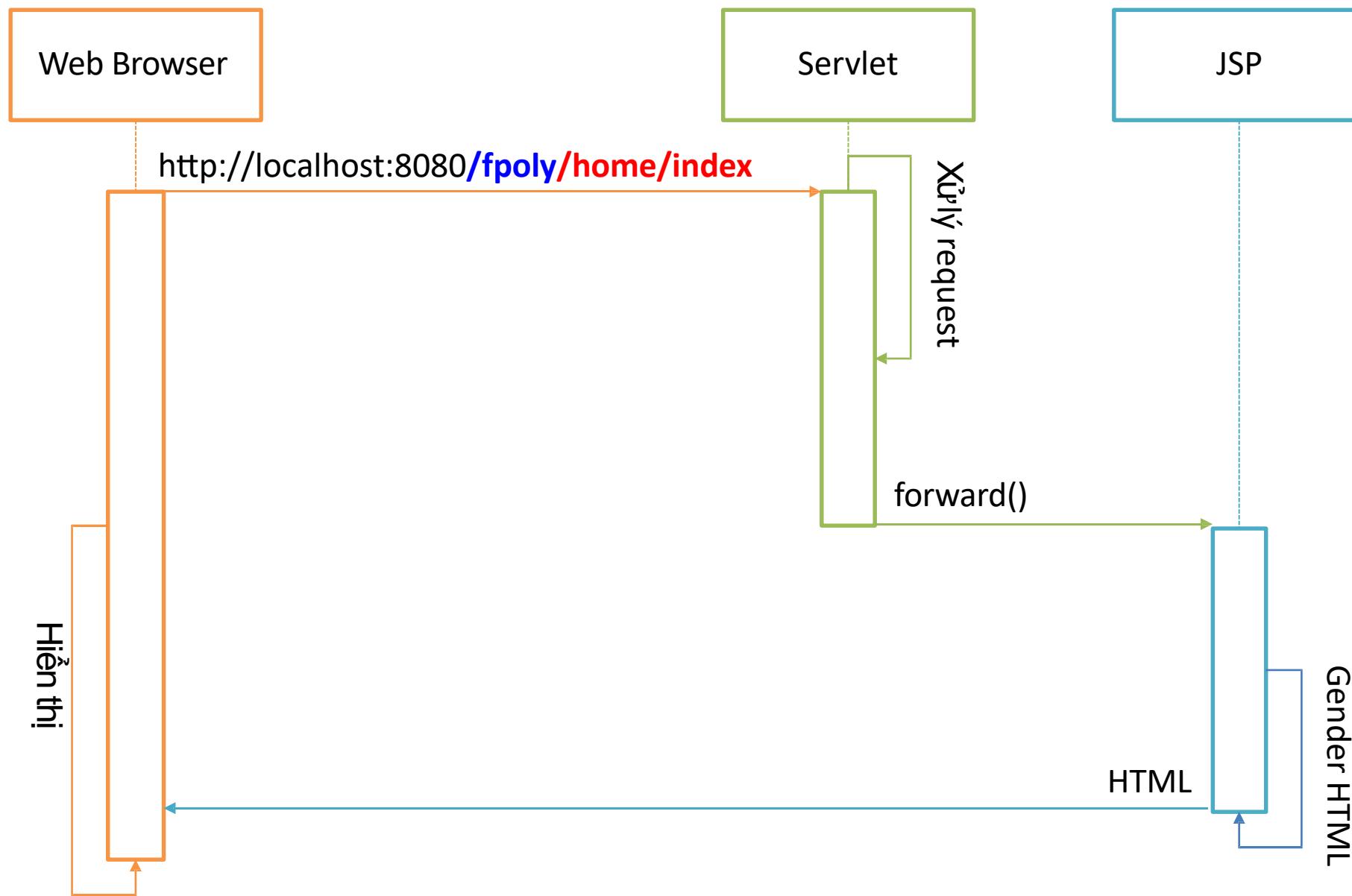
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Insert title here</title>
</head>
<body>
    <h1>FPT POLYTECHNIC</h1>
</body>
</html>
```

} HTML5

# CHẠY VÀ GIẢI THÍCH QUY TRÌNH XỬ LÝ REQUEST

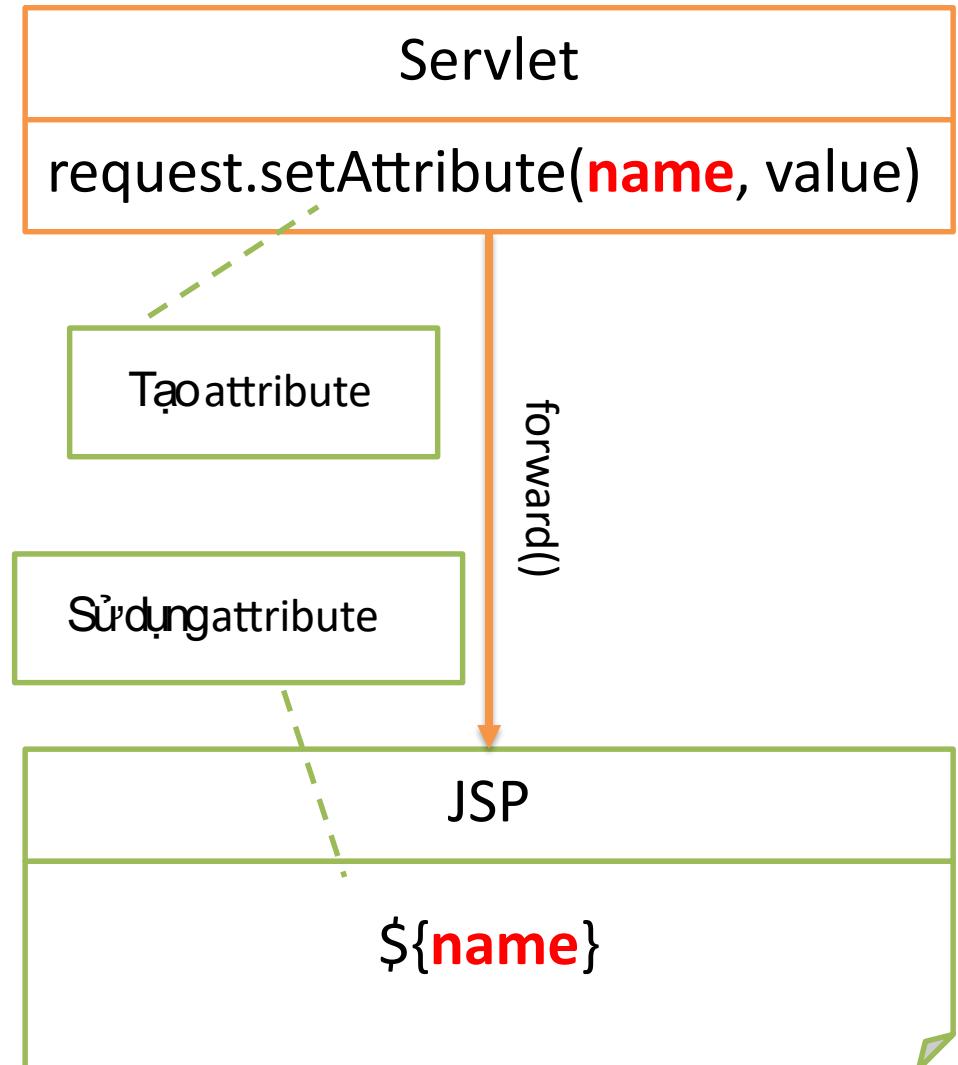


# QUY TRÌNH XỬ LÝ REQUEST



# TRUYỀN DỮ LIỆU TỪ SERVLET SANG JSP

- ❑ Servlet sẽ nhận và xử lý yêu cầu từ người sử dụng
- ❑ JSP đóng vai trò phản hồi người sử dụng
- ❑ JSP cần dữ liệu chia sẻ từ kết quả xử lý của Servlet để sinh giao diện phù hợp
- ❑ Chia sẻ dữ liệu
  - ❖ Servlet sẽ đặt dữ liệu vào request trước khi chuyển tiếp sang JSP.
  - ❖ JSP lấy và hiển thị



# TRUYỀN DỮ LIỆU TỪ SERVLET SANG JSP

The screenshot shows a Java IDE with two files open: `HomeServlet.java` and `index.jsp`. The `HomeServlet.java` code defines a servlet that sets an attribute `message` to "Chào thế giới Servlet/JSP" and then forwards the request to `/views/index.jsp`. The `index.jsp` page displays the message "FPT POLYTECHNIC" and "Chào thế giới Servlet/JSP". An orange arrow points from the line `req.setAttribute("message", "Chào thế giới Servlet/JSP");` in the servlet code to the displayed message in the browser output.

```
1 package com.poly.servlet;
2
3+import java.io.IOException;
10
11 @WebServlet("/home/index")
12 public class HomeServlet extends HttpServlet{
13     @Override
14     protected void doGet(HttpServletRequest req,
15             throws ServletException, IOException {
16         req.setAttribute("message", "Chào thế giới Servlet/JSP");
17         req.getRequestDispatcher("/views/index.jsp").forward(req, r
18     }
19 }
```

index.jsp

Insert title here

localhost:8080/hello/home/index

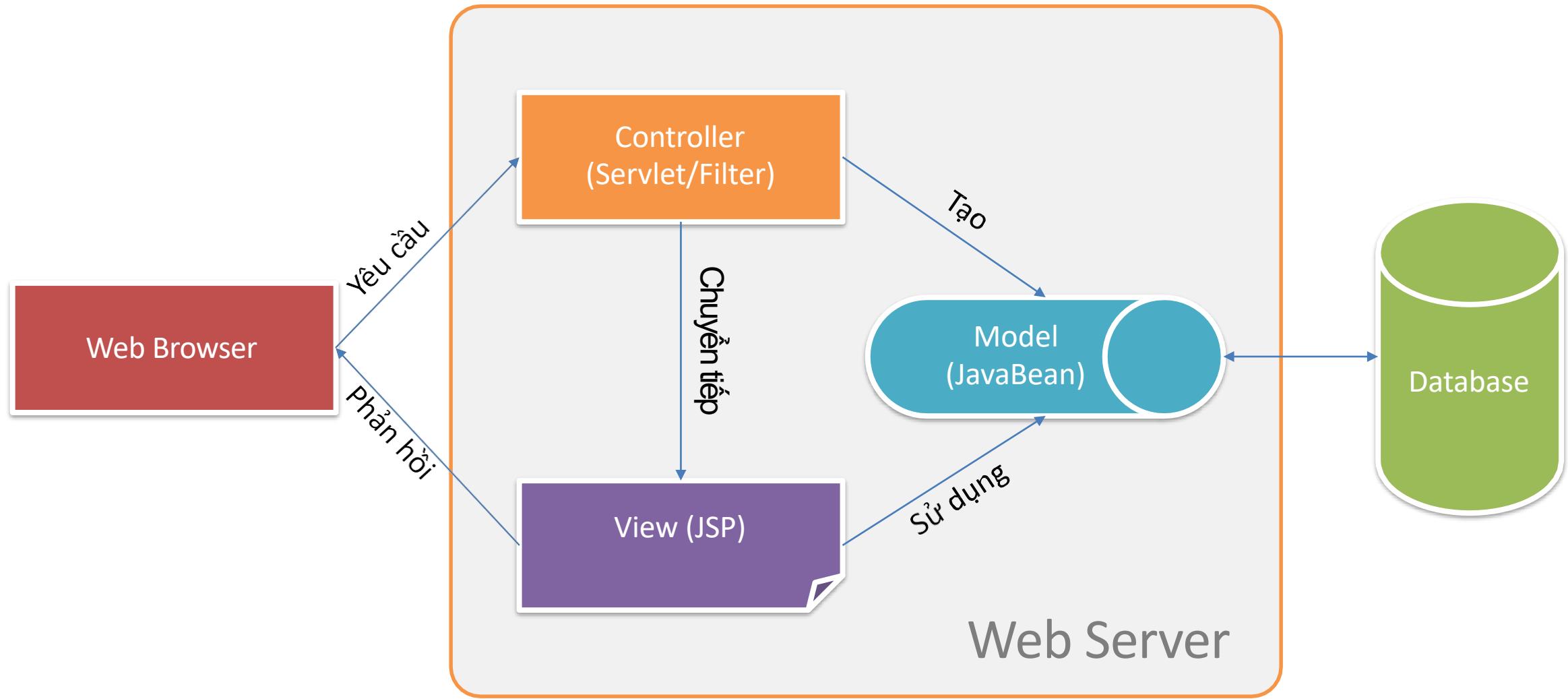
# FPT POLYTECHNIC

Chào thế giới Servlet/JSP

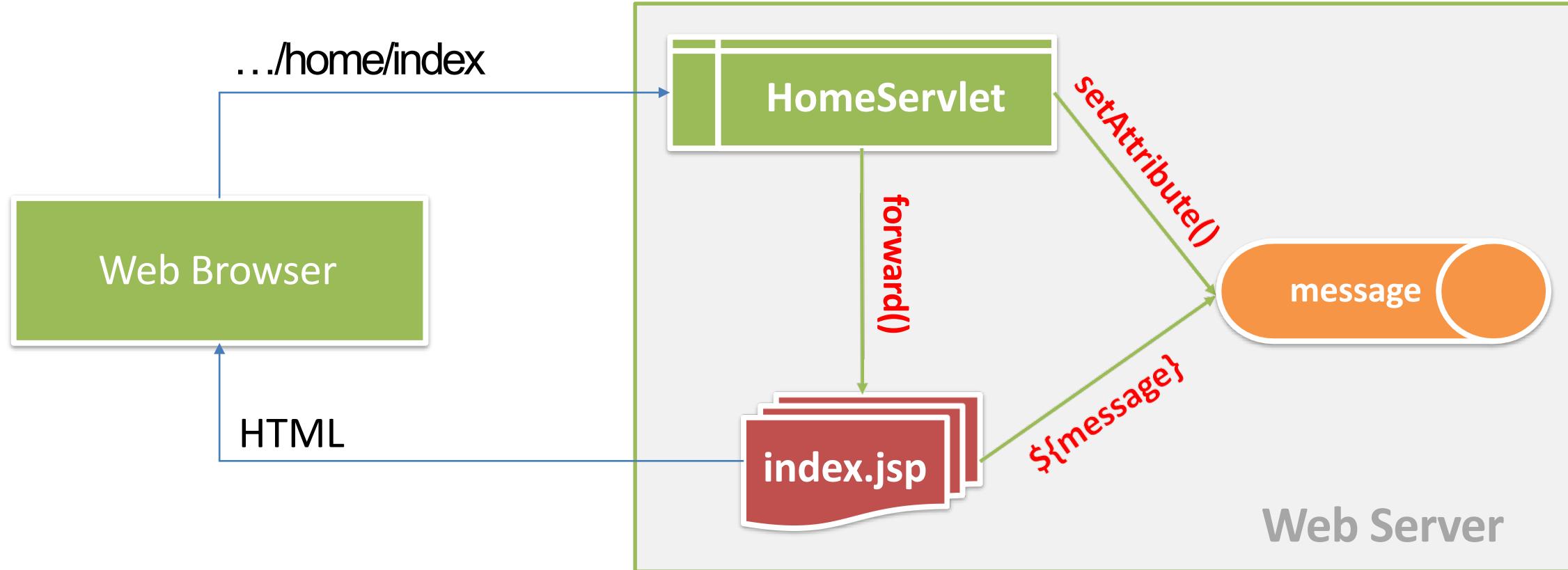
```
9<body>
10<h1>FPT POLYTECHNIC</h1>
11<h4>${message}</h4>
12</body>
13</html>
```

- ❑ `HttpServletRequest.setAttribute(name, Object)`
- ❑  `${name}`

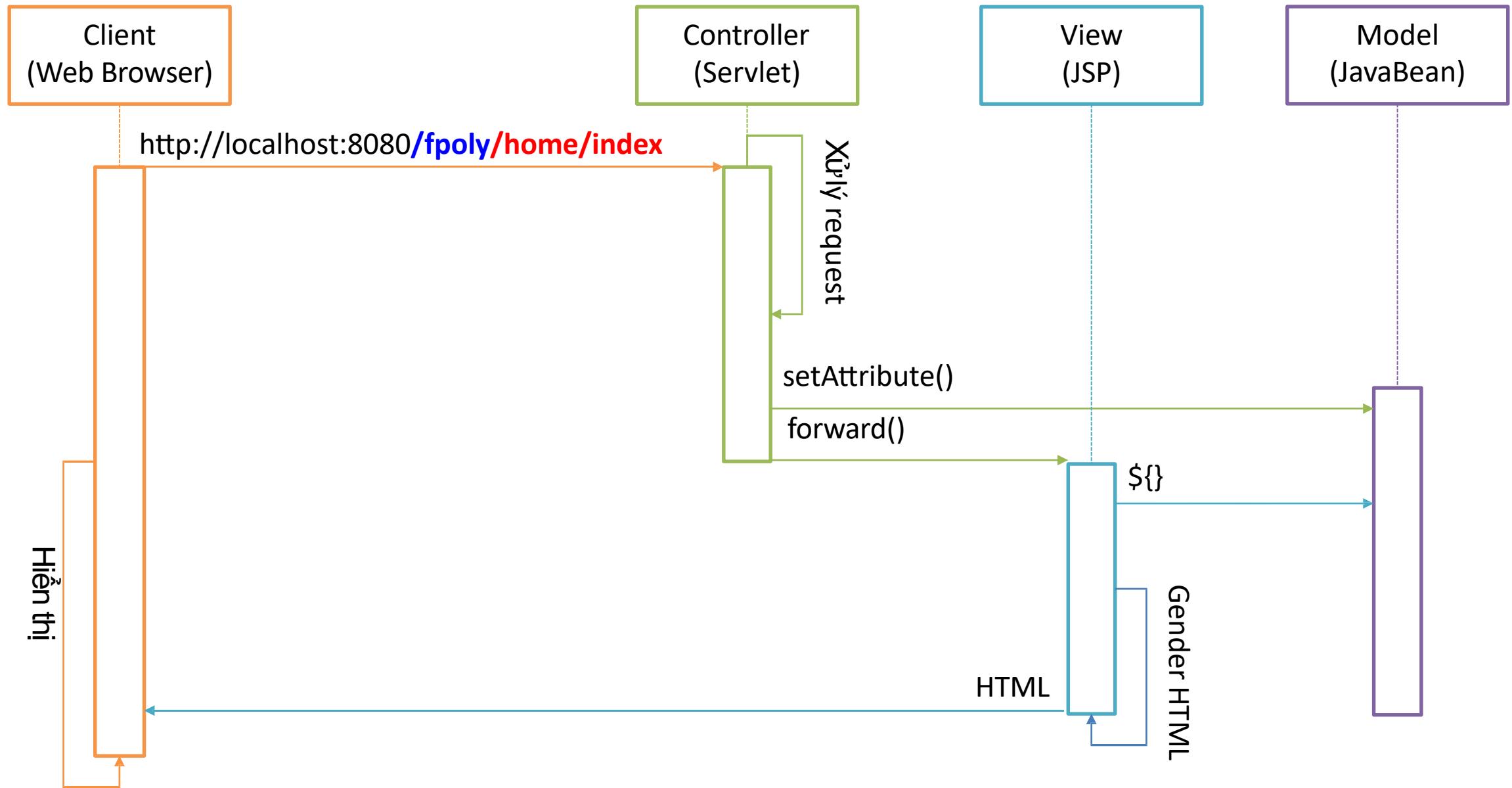
# MÔ HÌNH MVC



## TỔ CHỨC THEO MÔ HÌNH MVC



# QUY TRÌNH XỬ LÝ MVC



## THAM SỐ (PARAMETER)

- Tham số (parameter) là dữ liệu gửi đến servlet từ người sử dụng thông qua chuỗi truy vấn (Query String) hoặc form nhập

```
<a href="/home/index?fullname=Nguyễn Văn  
    Tèo"> Liên kết  
</a>
```

Liên kết

```
<form action="/home/index">  
    <input name="fullname" value="Nguyễn Văn  
    Tèo">  
    <button>Submit</button>  
</form>
```

Submit

@WebServlet("/home/index")

## ĐỌC VÀ XỬ LÝ THAM SỐ

---

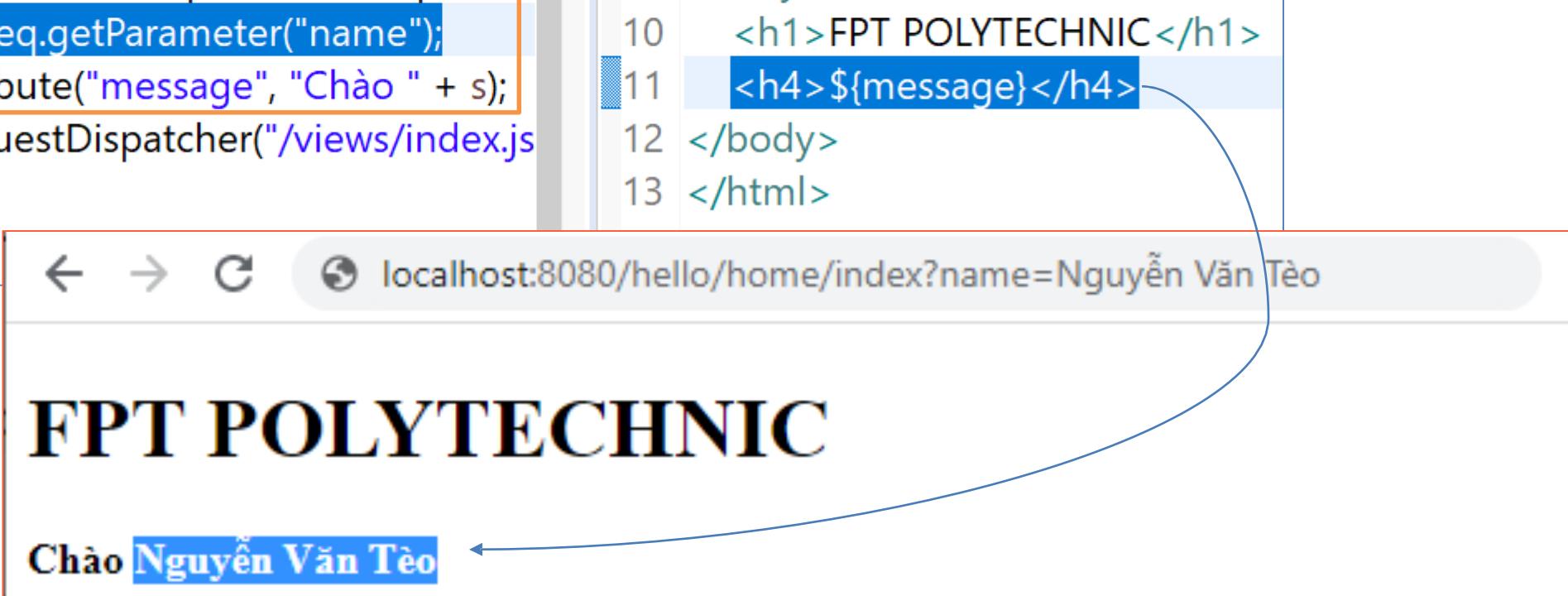
- ❑ Để đọc tham số, trong phương thức doGet(request, response) bạn sử dụng lệnh sau:
  - ❖ String hoten = **request.getParameter("fullname");**
- ❑ Với phương thức getParameter() thì kết quả nhận được là giá trị của tham số được gửi đến servlet hoặc null nếu tham số không tồn tại.
- ❑ Vì giá trị tham số đọc được luôn luôn là một chuỗi, vì vậy để xử lý bạn cần phải đổi sang một số kiểu phù hợp
  - ❖ **Integer.parseInt()**
  - ❖ **Double.parseDouble()**
  - ❖ **Boolean.parseBoolean()**
  - ❖ ...

## ĐỌC GIÁ TRỊ THAM SỐ

http://localhost:8080/fpoly/home/index?name=Nguyễn Văn Tèo

```
@WebServlet("/home/index")
public class HomeServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        String s = req.getParameter("name");
        req.setAttribute("message", "Chào " + s);
        req.getRequestDispatcher("/views/index.jsp").forward(req, resp);
    }
}
```

5<sup>o</sup> <head>
6 <meta charset="utf-8">
7 <title>Insert title here</title>
8 </head>
9<sup>o</sup> <body>
10 <h1>FPT POLYTECHNIC</h1>
11 <h4>\${message}</h4>
12 </body>
13 </html>



FPT POLYTECHNIC

Chào Nguyễn Văn Tèo

### ❑ Đóng gói ứng dụng web

- ❖ Export => \*.war

### ❑ Triển khai ứng dụng web

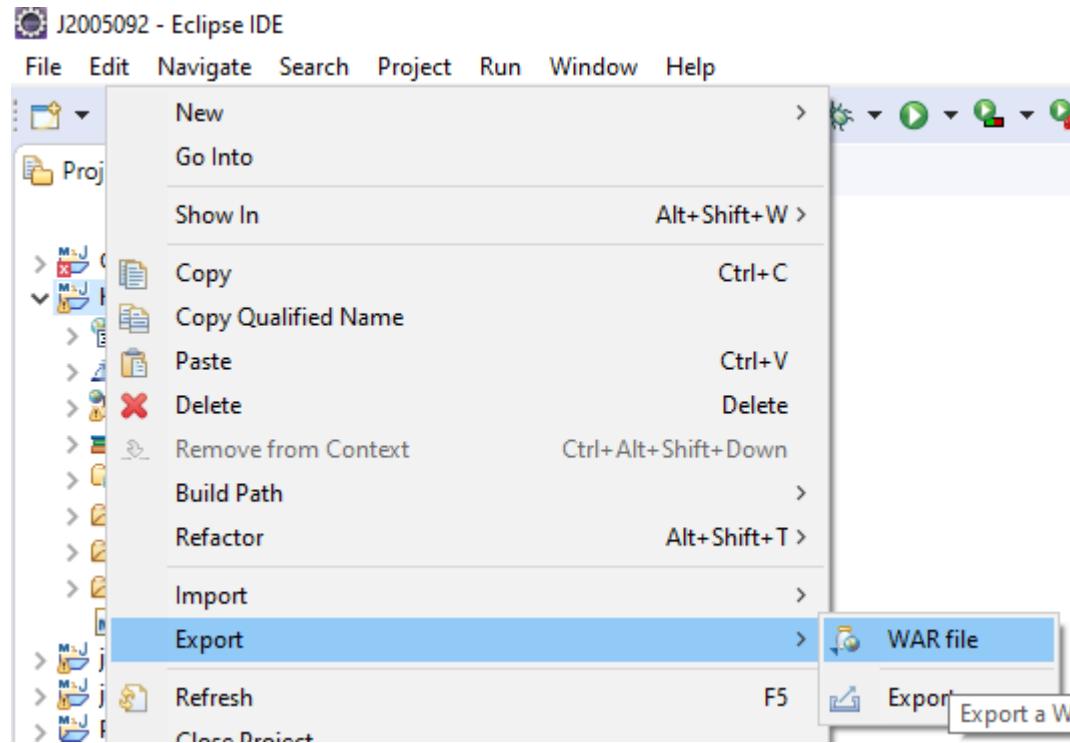
- ❖ Start Tomcat (nếu chưa start, với hosting thì Tomcat đã sẵn có)
- ❖ Chép sản phẩm đóng gói (\*.war) vào thư mục **webapps**

Chú ý: đợi một lúc sẽ thấy file \*.war được giải nén thành một thư mục

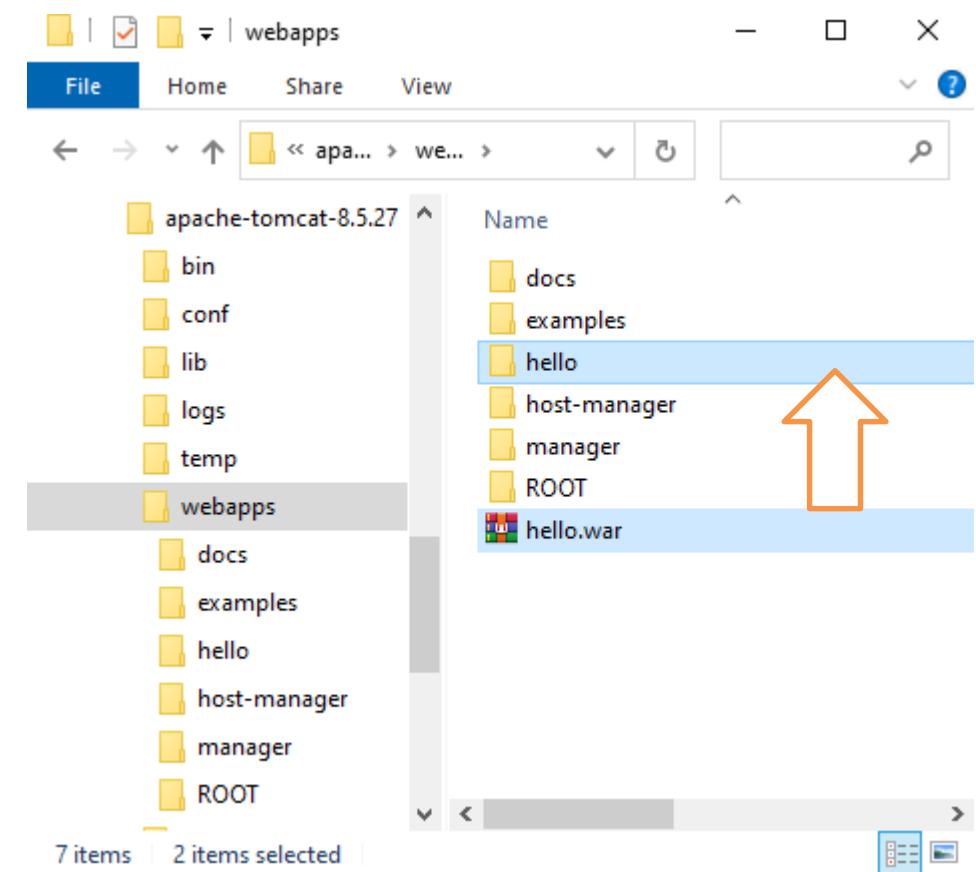
### ❑ Chạy

- ❖ Mở trình duyệt và nhập địa chỉ trang chủ để chạy

# ĐÓNG GÓI VÀ TRIỂN KHAI



Đóng gói



Triển khai

- Thế giới web
- Mô hình ứng dụng web
- Mô hình công nghệ 3 lớp
- Mô hình công nghệ  
Servlet/JSP
- Cài đặt môi trường phát triển
- Dự án web động
- Tổ chức theo mô hình MVC
- Đóng gói và triển khai