



# SERVLET – BUỔI 1

CYBERSOFT.EDU.VN



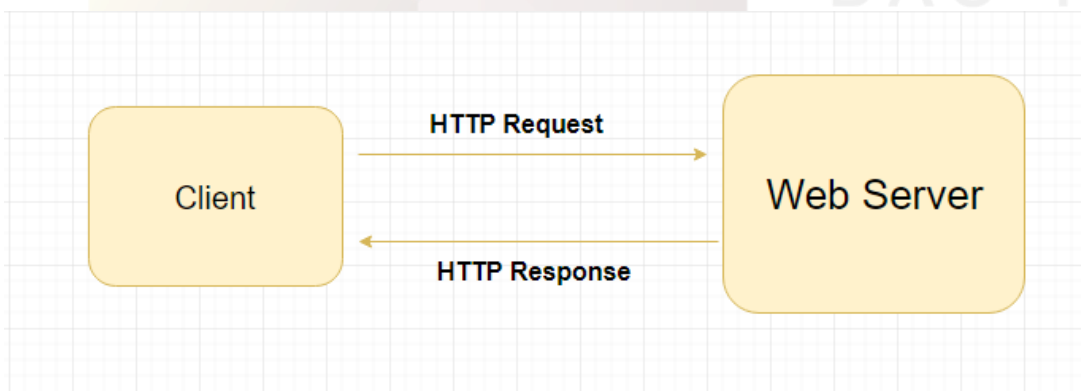
# Nội dung

- ☐ Tạo dự án sử dụng Servlet
- ☐ Cấu hình Servlet
- ☐ Http Header
- ☐ Vòng đời của Servlet
- ☐ Request – Response
- ☐ Phương thức GET – POST
- ☐ Redirect – Forward
- ☐ Bài tập demo

CYBERSOFT

ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

# Web Server



## ❑ Định nghĩa.

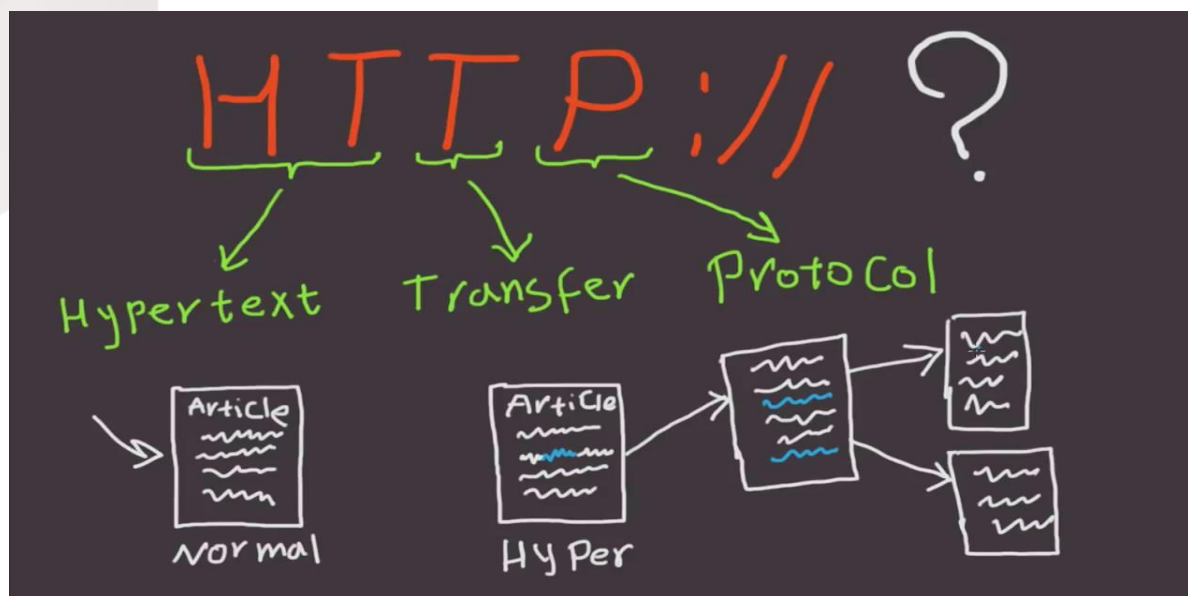
- ❖ Web Server là **máy chủ cài đặt chương trình** (phần mềm) phục vụ các ứng dụng web.
- ❖ Web Server có khả năng **tiếp nhận các yêu cầu** (request) từ các trình duyệt web (browser) và **gửi phản hồi** (response) đến các máy khách (client) thông qua **giao thức HTTP** hoặc các giao thức khác.

✓ **HTTP Request:** Yêu cầu từ client.

✓ **HTTP Response:** Phản hồi của server.

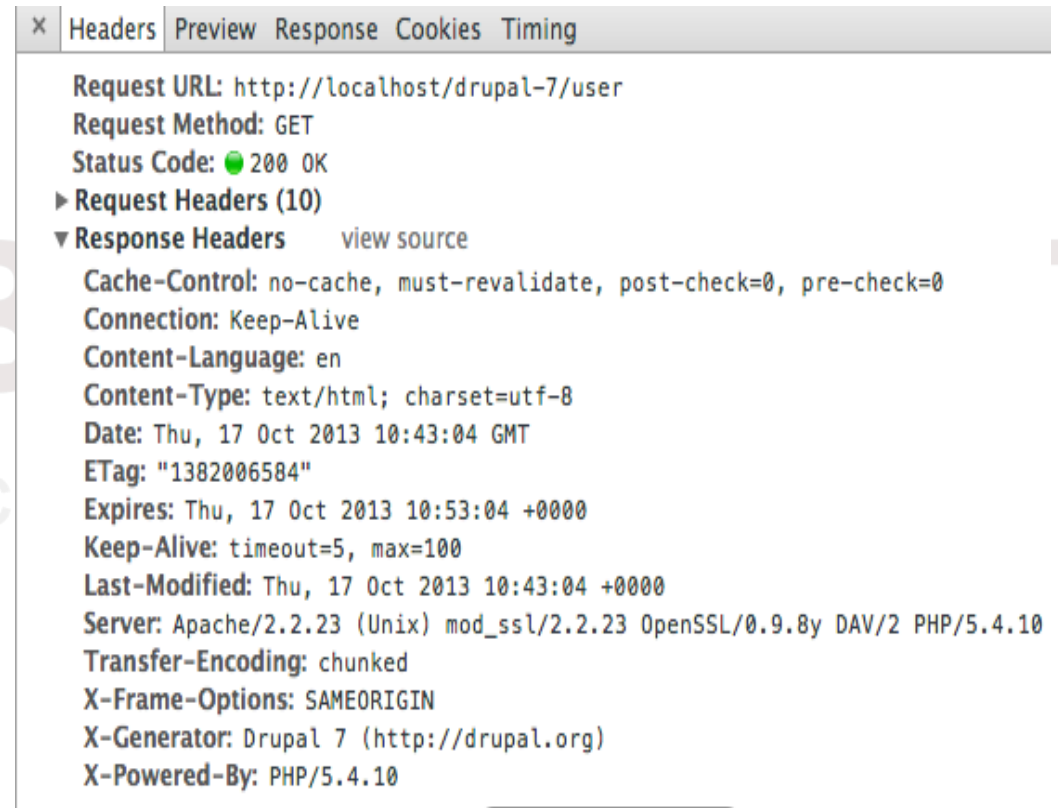
# HTTP là gì?

- ❑ **Http** (HyperText Transfer Protocol) là một **tập hợp các quy tắc, quy ước** dùng để trao đổi thông tin, truyền tải dữ liệu giữa Client và Server.
- ❑ **Http** được thiết kế theo **mô hình Client – Server**, giao tiếp giữa Client và Server dựa vào một cặp **Request** và **Response**.
- ❑ **Client** đưa ra các yêu cầu (request) và **Server** phản hồi các yêu cầu của Client qua response.



# Http Header

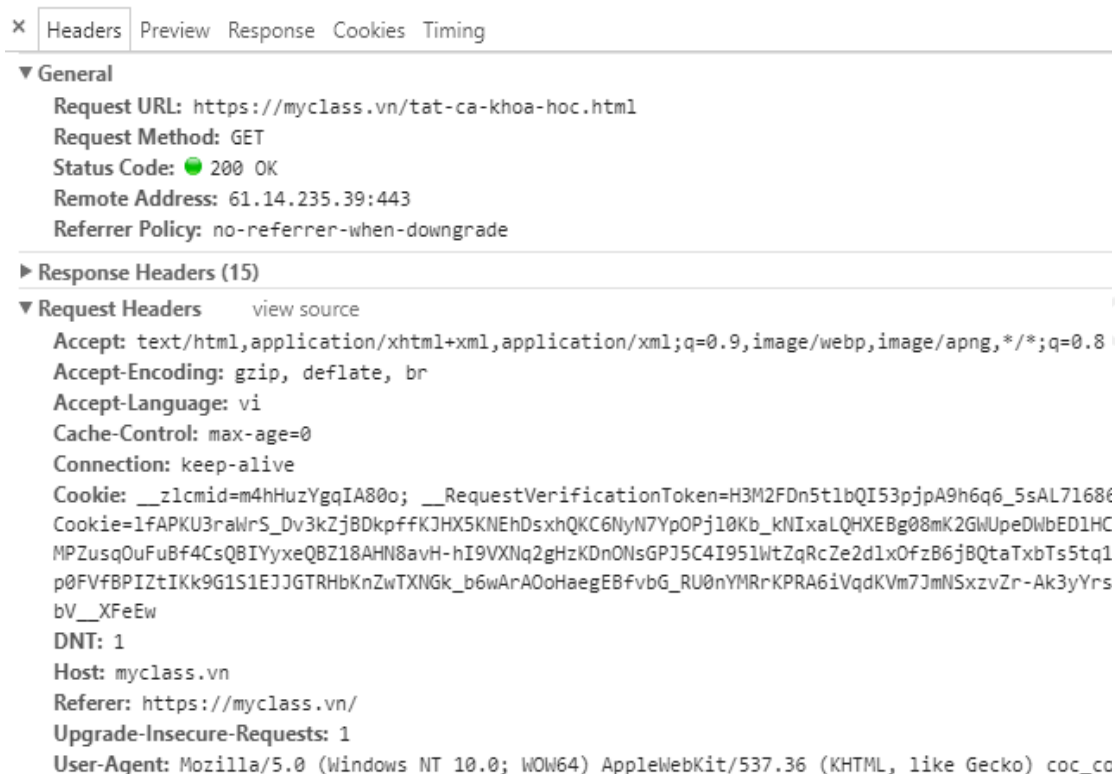
- ❑ **Http Header** là phần đầu của **Http**, chứa **thông tin yêu cầu trong mỗi Request** mà Client gửi đi **cũng như thông tin phản hồi trong mỗi Response** mà Server trả về.
- ❑ **Http Header** chứa thông tin chủ yếu như **thông tin về trình duyệt, thông tin cấu hình server, ngày tháng, kiểu dữ liệu truyền tải,...** Server và trình duyệt sẽ dựa vào các thông tin này để trả dữ liệu và hiển thị thông tin cho phù hợp.



```
X Headers Preview Response Cookies Timing
Request URL: http://localhost/drupal-7/user
Request Method: GET
Status Code: 200 OK
▶ Request Headers (10)
▼ Response Headers view source
Cache-Control: no-cache, must-revalidate, post-check=0, pre-check=0
Connection: Keep-Alive
Content-Language: en
Content-Type: text/html; charset=utf-8
Date: Thu, 17 Oct 2013 10:43:04 GMT
ETag: "1382006584"
Expires: Thu, 17 Oct 2013 10:53:04 +0000
Keep-Alive: timeout=5, max=100
Last-Modified: Thu, 17 Oct 2013 10:43:04 +0000
Server: Apache/2.2.23 (Unix) mod_ssl/2.2.23 OpenSSL/0.9.8y DAV/2 PHP/5.4.10
Transfer-Encoding: chunked
X-Frame-Options: SAMEORIGIN
X-Generator: Drupal 7 (http://drupal.org)
X-Powered-By: PHP/5.4.10
```

# Http Header

❑ **Request Header** chứa các thông tin kèm theo khi Client gửi yêu cầu lên Server.

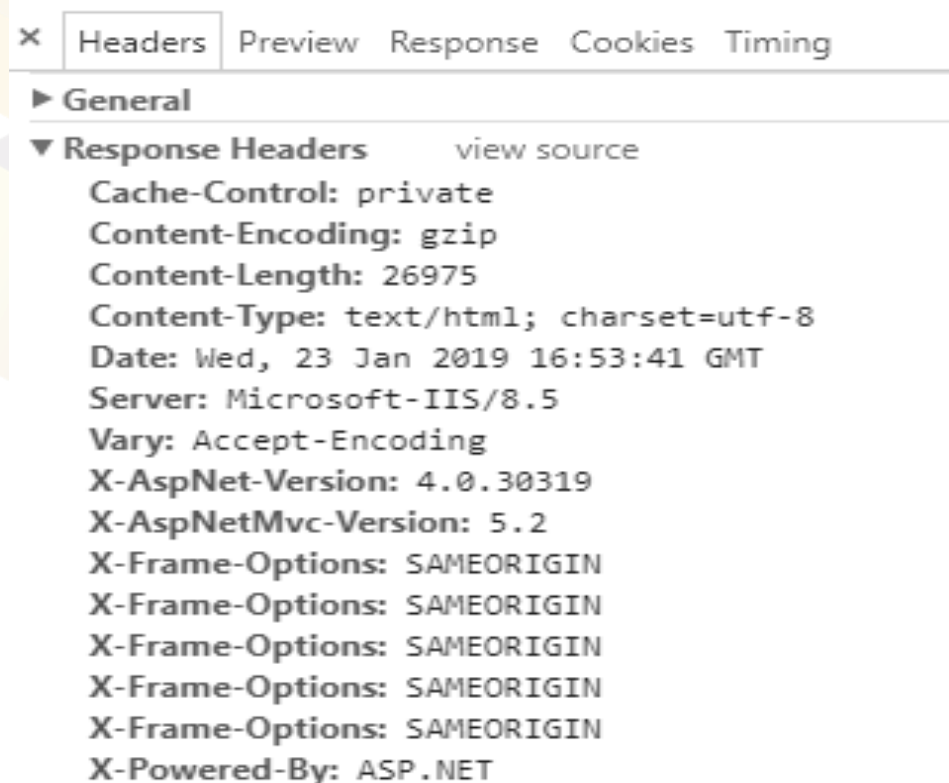


❑ **Thông tin cơ bản**

- ✓ **Host:** Địa chỉ trang web mà Client muốn gửi request.
- ✓ **Accept-Encoding:** Báo cho Server biết client chấp nhận loại định dạng mã hóa nào.
- ✓ **User-Agent:** Chứa thông tin về hệ điều hành và trình duyệt người dùng đang sử dụng.
- ✓ **Cookie:** Chứa thông tin về các cookie đã được lưu trên trình duyệt.
- ✓ **Accept-Language:** Thông tin ngôn ngữ mặc định của client.

# Http Header

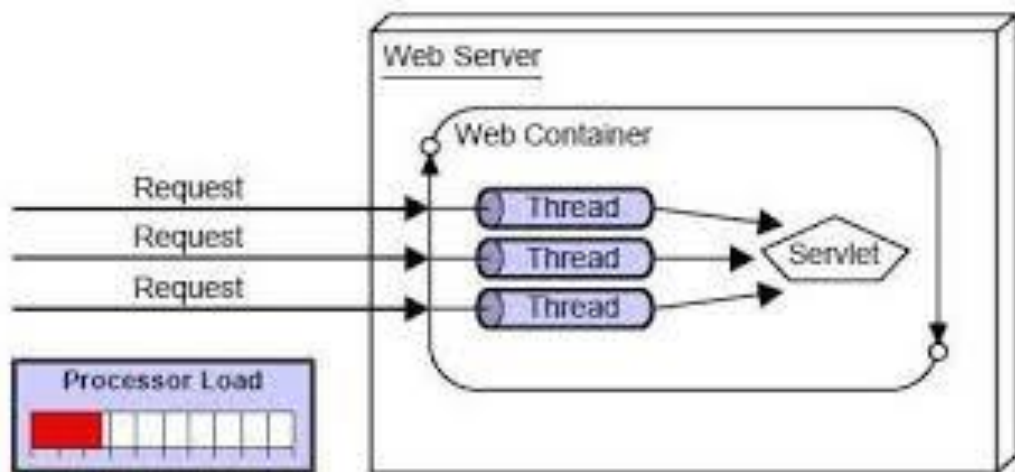
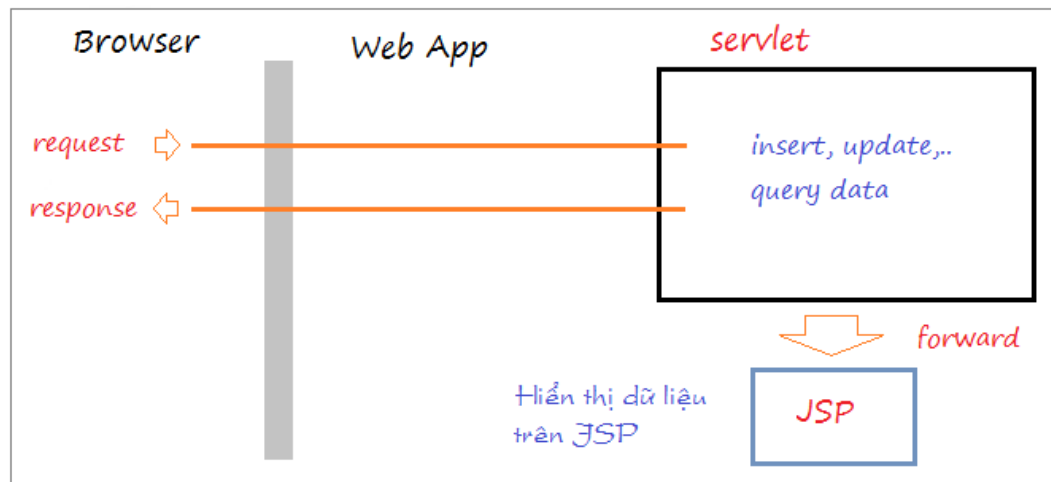
- ❑ **Response Header** chứa các thông tin kèm theo khi Server gửi phản hồi về cho Client.



- ❑ **Thông tin cơ bản**

- ✓ **Set-Cookie:** Server gửi cookie cho Client lưu lại và Client sẽ gửi lại cookie cho Server trong các request tiếp theo.
- ✓ **Expires:** Thông báo cho trình duyệt biết nội dung có hiệu lực trong bao lâu.
- ✓ **Content-Length:** Dung lượng gói tin gửi cho Client.
- ✓ **Content-Type:** Thông tin định dạng văn bản.
- ✓ **Last-Modified:** Thông báo lần cuối cùng văn bản được chỉnh sửa.

# Java Servlet



## ❑ Servlet

- Servlet là **bộ thư viện của Java** dùng để tạo các ứng dụng web (website/ web service).
- Servlet hỗ trợ nhiều **interface** và **class** trong lập trình web.
- Servlet cho phép nhà phát triển phần mềm **thêm những nội dung động** vào web server sử dụng nền tảng Java (html/xml).
- Servlet mạnh mẽ và chịu tải tốt.
- **Servlet Interface** khai báo 3 phương thức cần thiết cho một vòng đời của một Servlet là **init()**, **service()** và **destroy()**. Các phương thức được cài đặt bởi Servlet và được thực thi bởi **Servlet Container**.



# Cài đặt Server

## ❑ Download Tomcat

- Vào link tải về và giải nén Tomcat
- <https://tomcat.apache.org/download-90.cgi> .



Please see the [README](#) file for packaging information. It explains wh

### Binary Distributions

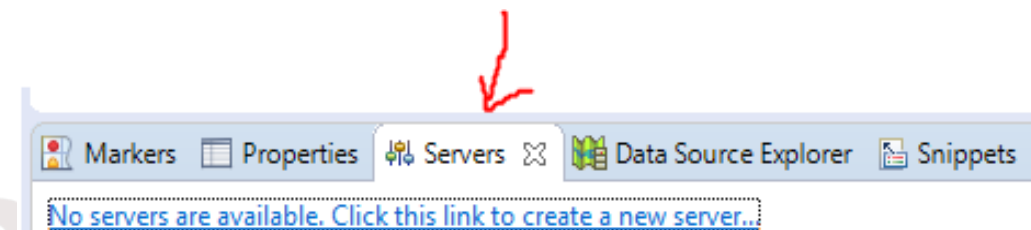
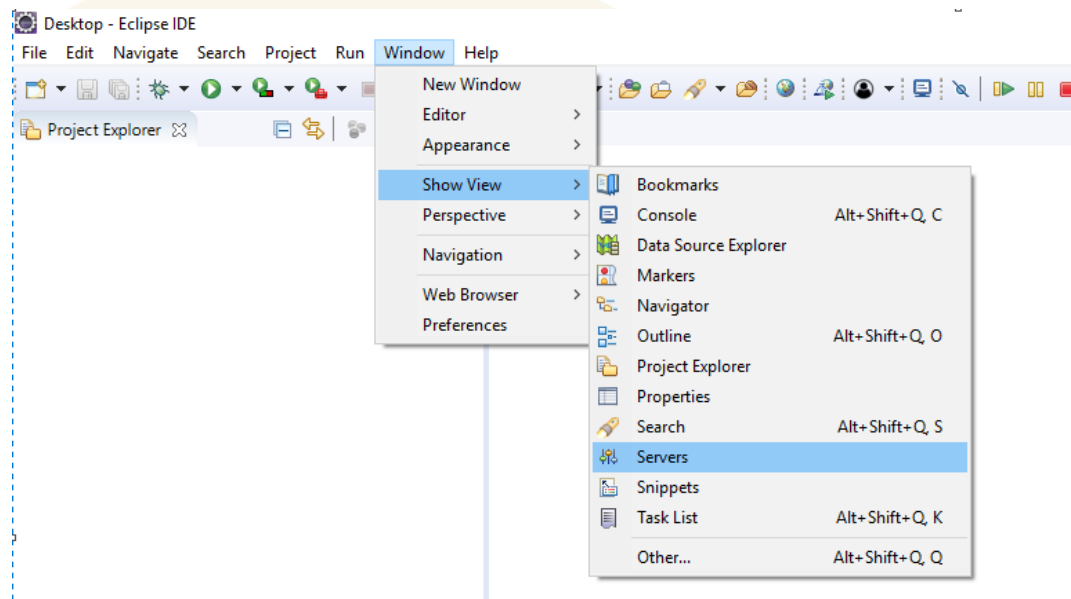
- Core:
  - [zip](#) (pgp, sha512)
  - [tar.gz](#) (pgp, sha512)
  - [32-bit Windows zip](#) (pgp, sha512)
  - [64-bit Windows zip](#) (pgp, sha512)
  - [32-bit/64-bit Windows Service Installer](#) (pgp, sha512)
- Full documentation:
  - [tar.gz](#) (pgp, sha512)
- Deployer:
  - [zip](#) (pgp, sha512)
  - [tar.gz](#) (pgp, sha512)
- Extras:
  - [JMX Remote jar](#) (pgp, sha512)
  - [Web services jar](#) (pgp, sha512)
- Embedded:
  - [tar.gz](#) (pgp, sha512)
  - [zip](#) (pgp, sha512)

OFT  
ÁP TRÌNH

# Cài đặt Server

❑ Click vào **Window** → Chọn **Show View** → Chọn **Servers**.

❑ Click vào tab **Servers**.

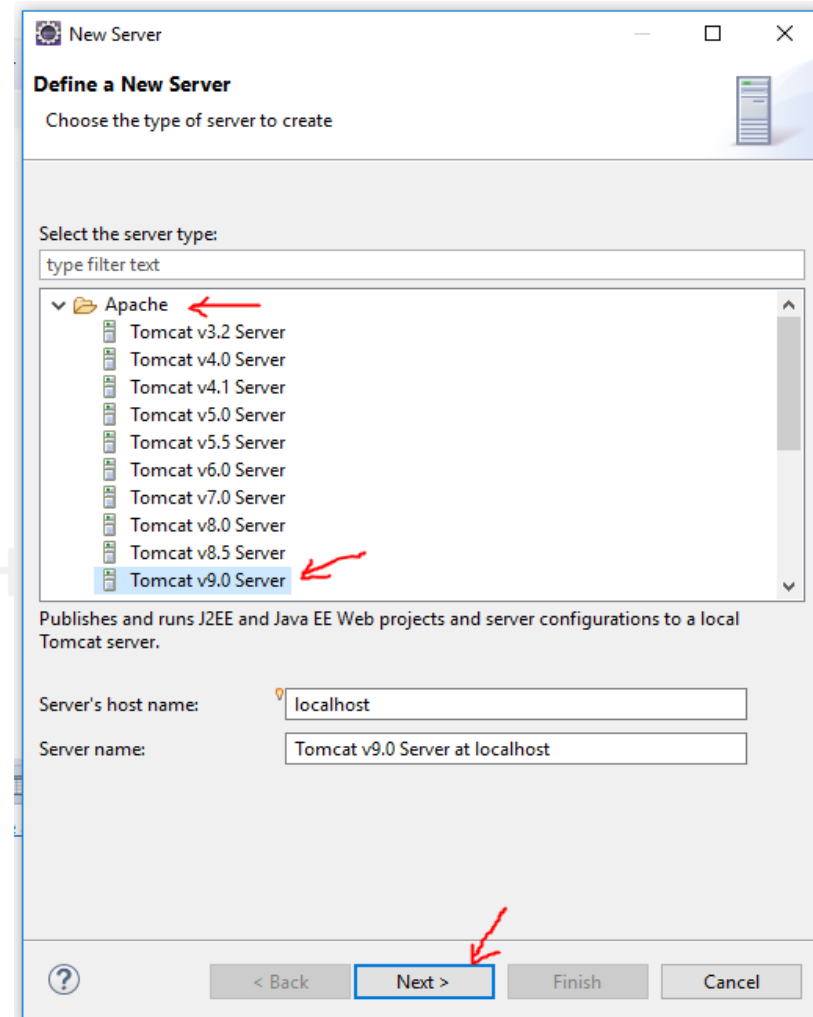


O CHUYÊN GIA LẬP TRÌNH

# Cài đặt Server

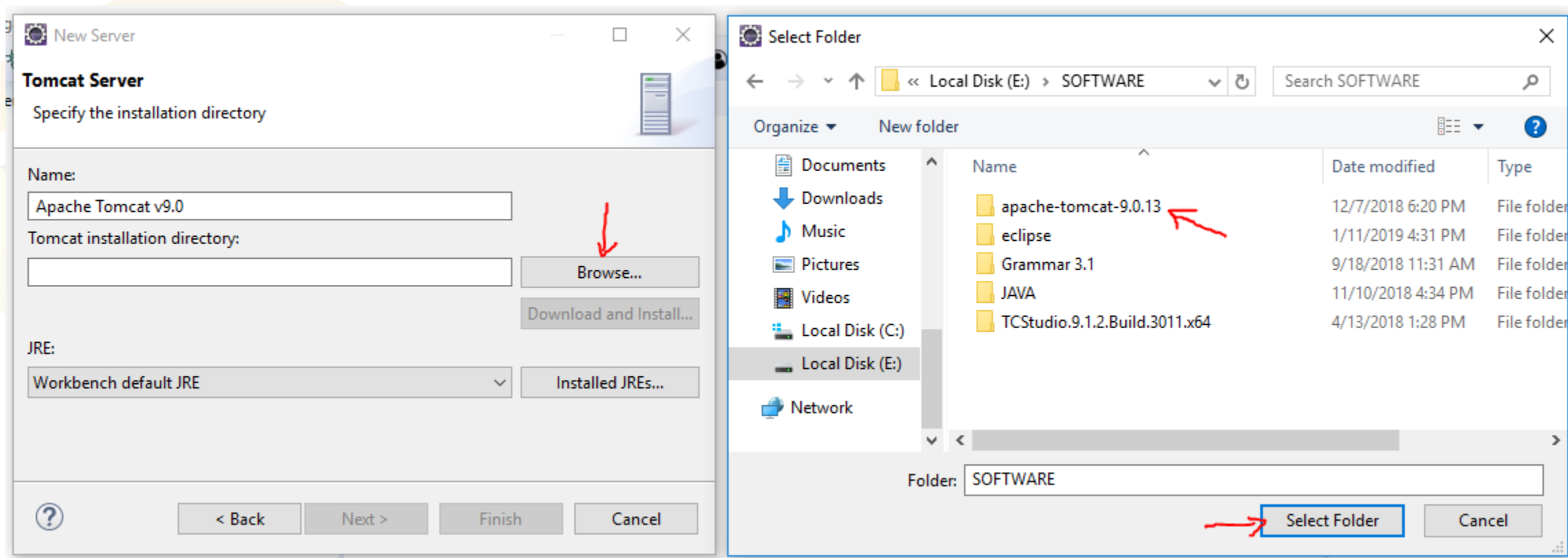
## ❑ Tạo mới server

- Click chuột phải vào tab Servers  
➔ Chọn New ➔ Chọn Server.
- Cửa sổ hiện ra ➔ click Apache  
➔ Chọn Tomcat có version trùng với bản bạn vừa tải về ➔ Chọn Next.



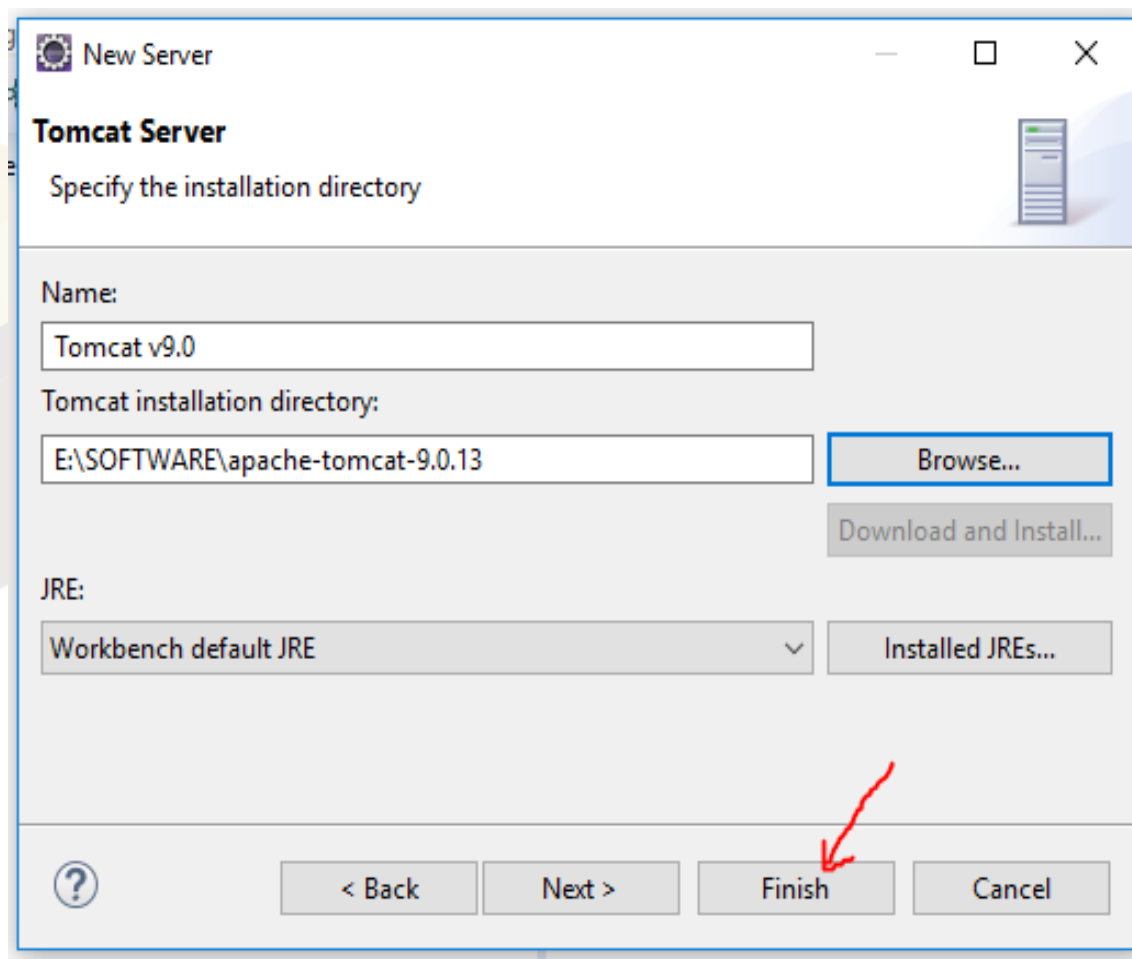
# Cài đặt Server

- ❑ Click vào **Browse** → Mở **thư mục** chứa **Tomcat** bạn vừa giải nén → Chọn thư mục **apache-tomcat** → Click vào **Select Folder**.



# Cài đặt Server

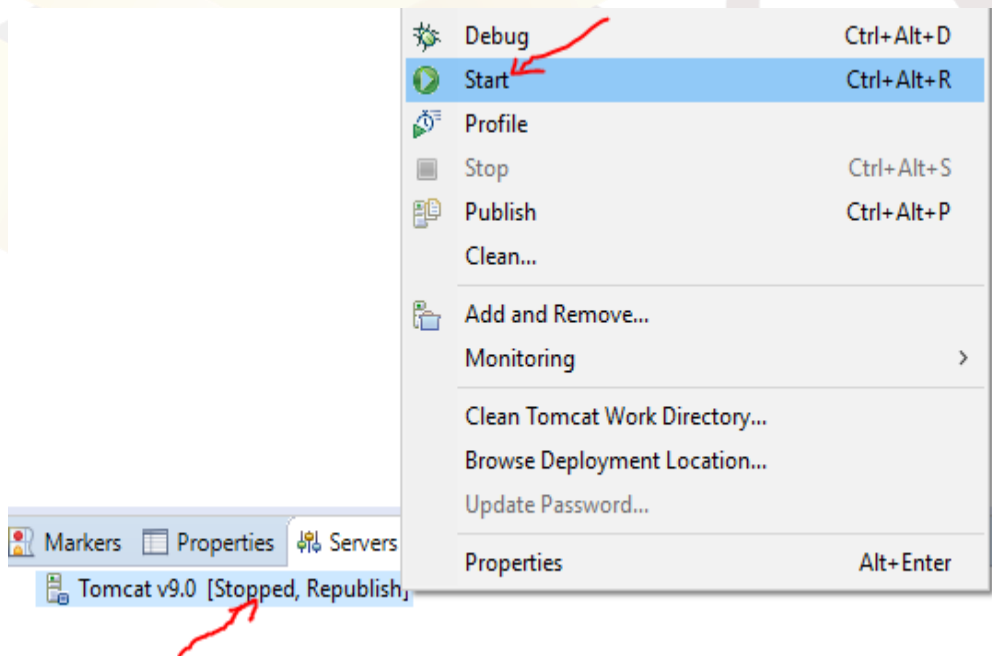
- ❑ Chọn **Finish** để hoàn thành



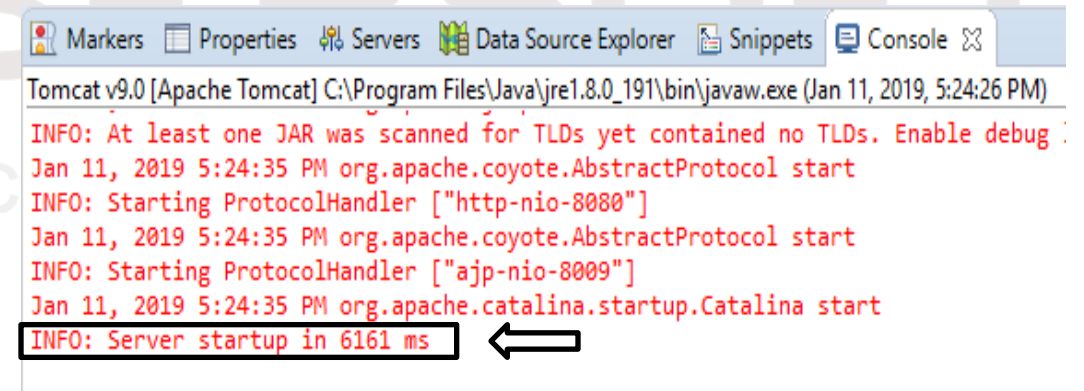
# Cài đặt Server

## ❑ Chạy thử

- Click chuột phải vào Server vừa cài đặt → Chọn **Start** để chạy Server.



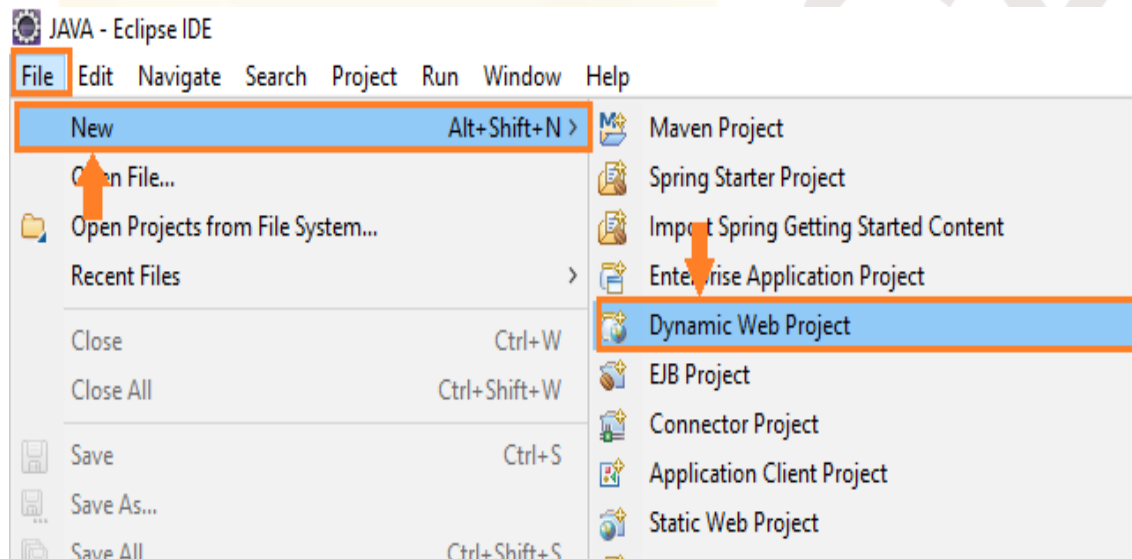
- ❖ **Server Tomcat** được cài đặt thành công nếu trong tab **Console** xuất hiện thông báo **INFO: Server startup**.



# Tạo mới dự án

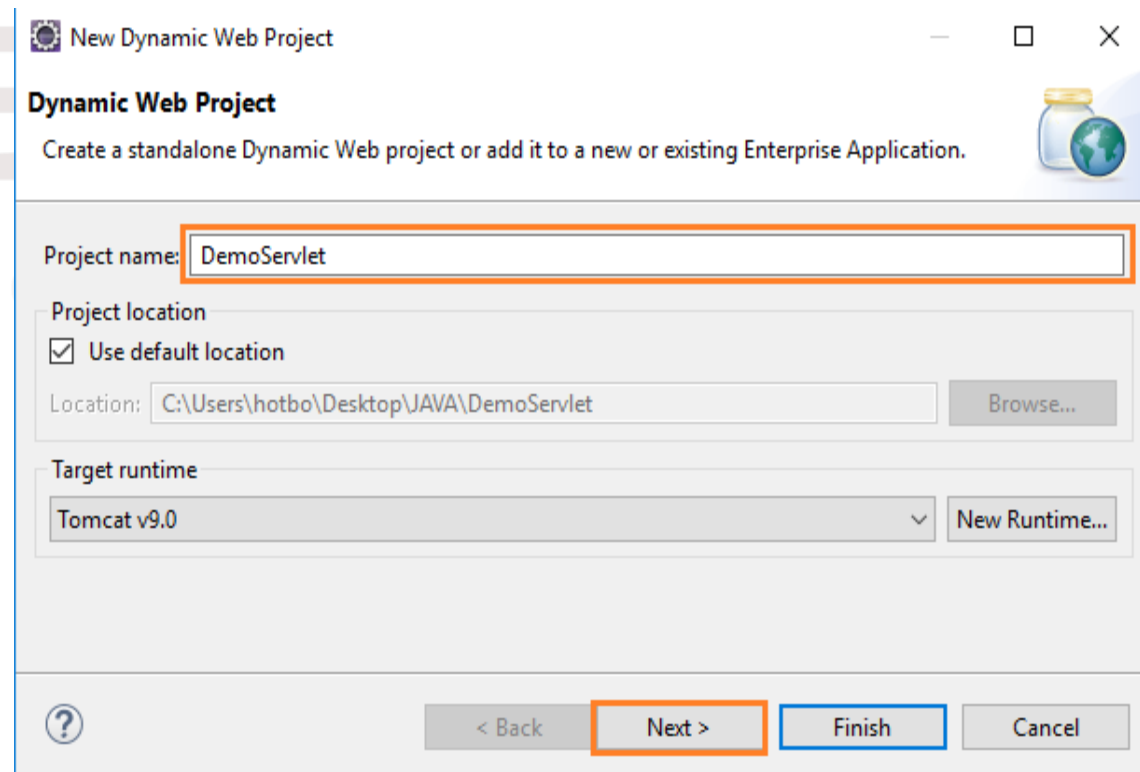
## ❑ Bước 1

- Click vào **File** → Chọn **New** → Chọn **Dynamic Web Project**.



## ❑ Bước 2

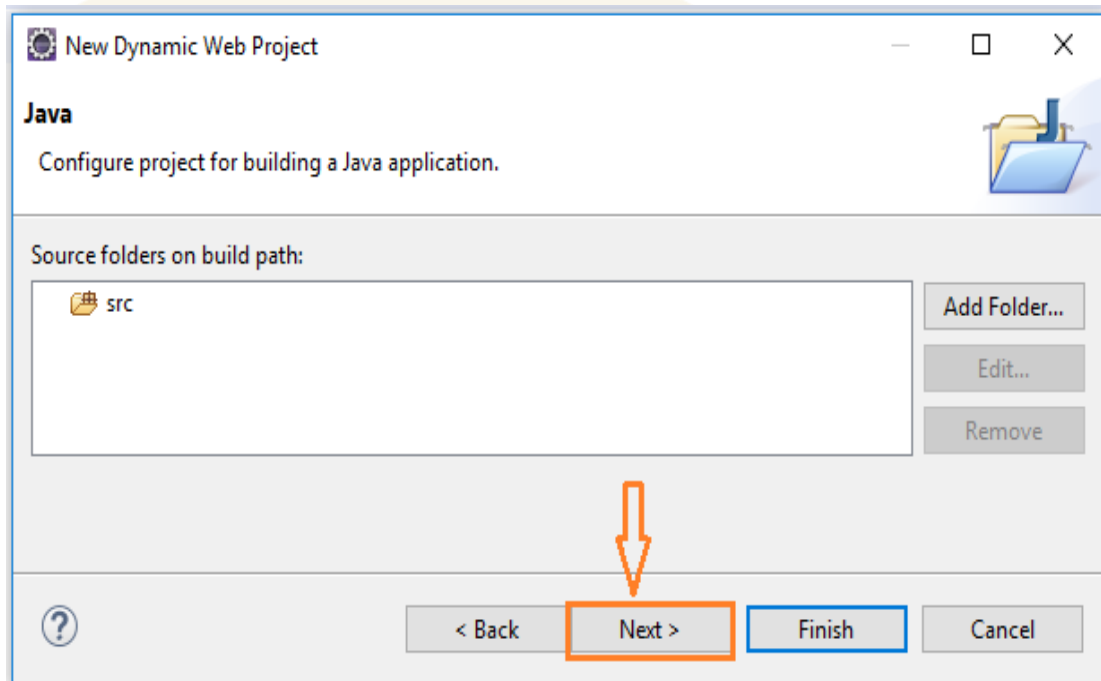
- Đặt tên cho **Project** → Chọn **Next**.



# Tạo mới dự án

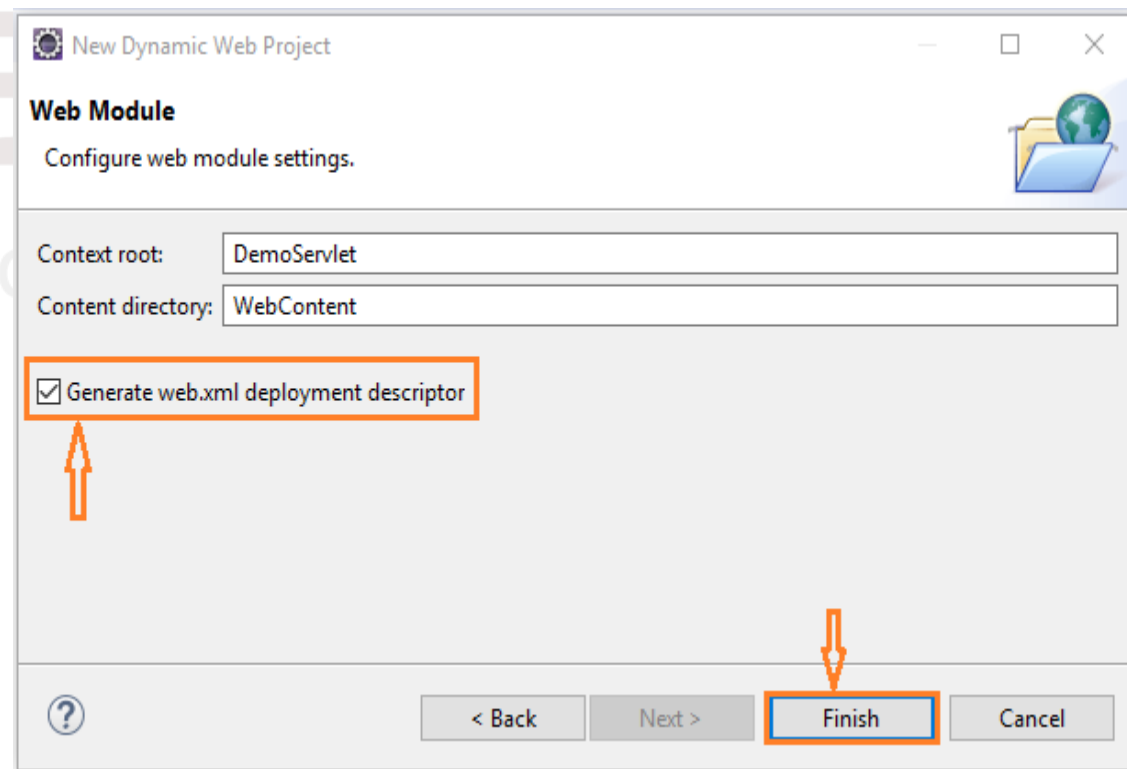
## ❑ Bước 3

➤ Chọn **Next**



## ❑ Bước 4

➤ Tích chọn **Generate web.xml** →  
Chọn **Finish** để kết thúc.

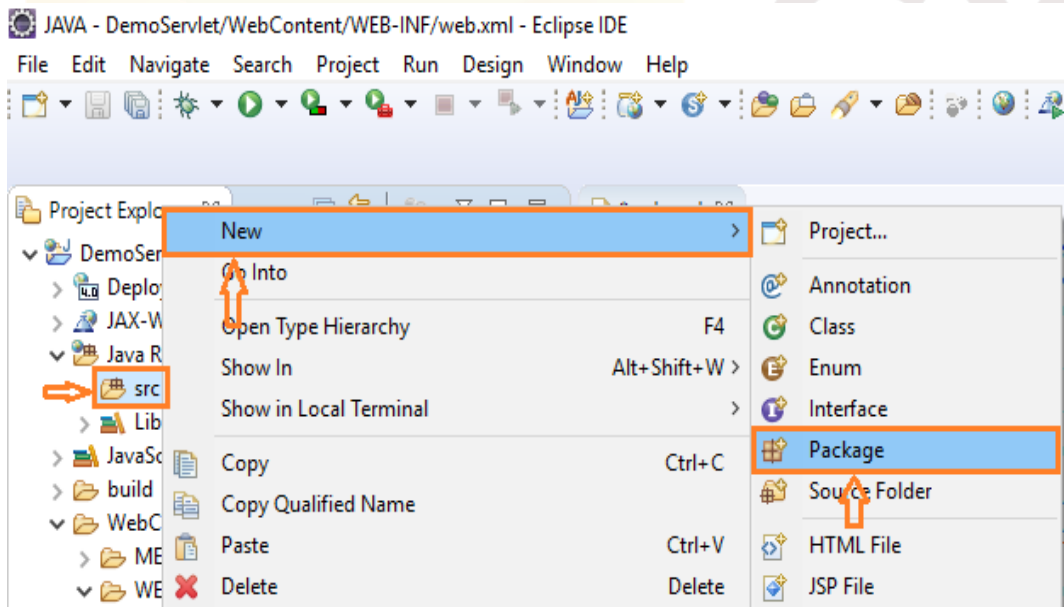




# Tạo mới Servlet

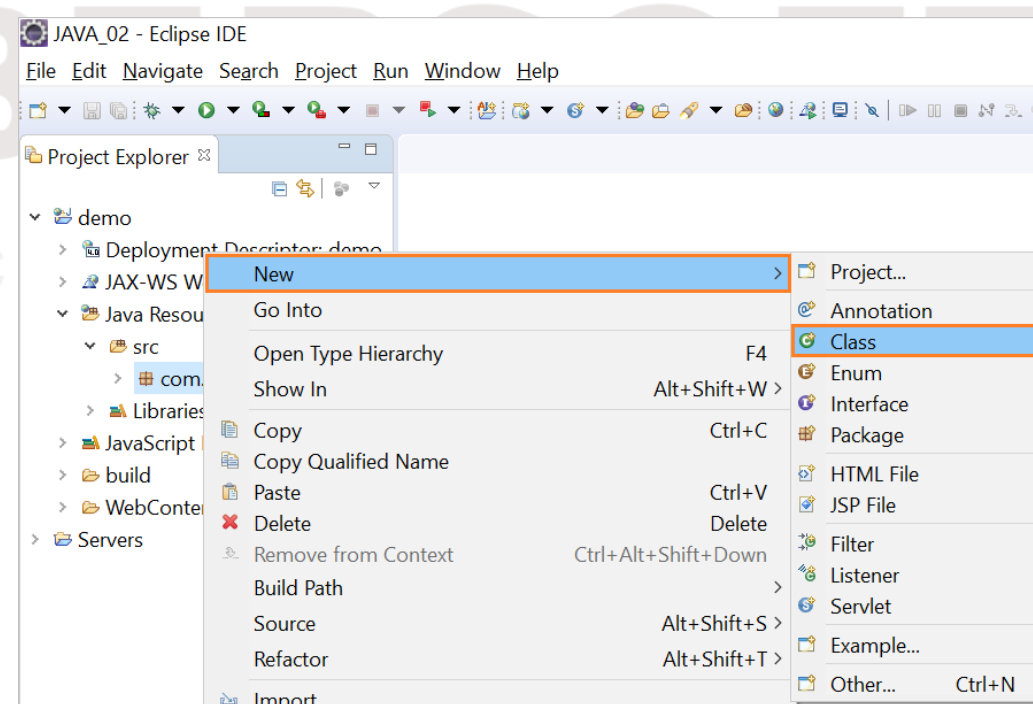
## ❑ Bước 1

- Click chuột phải vào thư mục **src** → Chọn **New** → Chọn **Package** để tạo Package.



## ❑ Bước 2

- Click chuột phải vào **Package** vừa tạo → Chọn **New** → Chọn **Class** để tạo 1 class mới → Đặt tên **HomeServlet**.



# HomeServlet

```
package com.myclass.servlet;

import java.io.IOException;

public class HomeServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        resp.setContentType("text/html");
        resp.setCharacterEncoding("UTF-8");

        PrintWriter writer = resp.getWriter();
        writer.print("<h1>Xin chào</h1>");

        writer.close();
    }
}
```

# Cấu hình Container (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://
  <display-name>demo</display-name>
  <servlet>
    <servlet-name>homeServlet</servlet-name>
    <servlet-class>com.myclass.servlet.HomeServlet</servlet-class>
  </servlet>

  <!-- Map all requests to the DispatcherServlet for handling -->
  <servlet-mapping>
    <servlet-name>homeServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

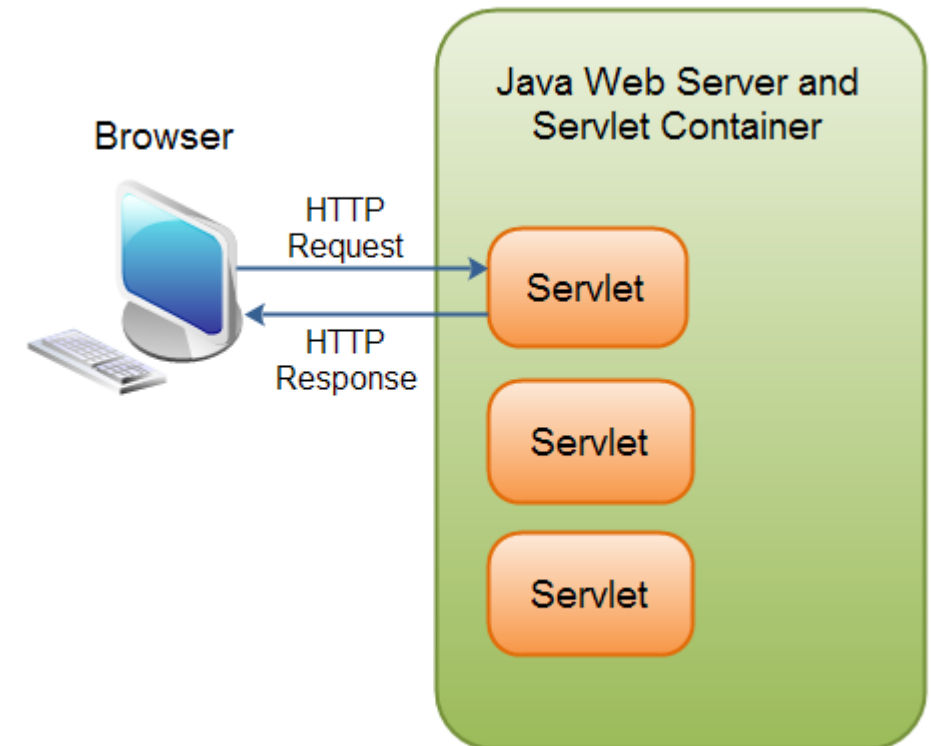
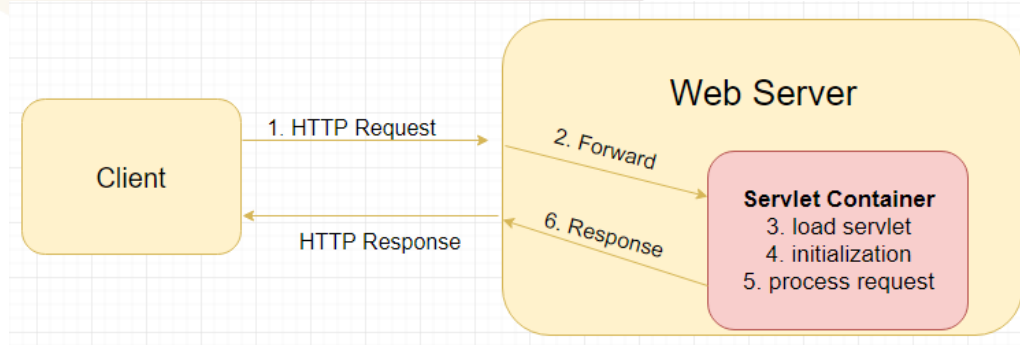
</web-app>
```

# Servlet Container

## ❑ Servlet container

- Servlet **không có phương thức main()**. Nó được điều khiển bởi một Java app được gọi là **Container**.
- Servlet container thực chất là **một phần của web server** tương tác với các servlet.

- Servlet container giống như một cái thùng chứa, **chứa tất cả các servlet** bên trong.



# Servlet Container

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  <display-name>demo</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <servlet>
    <servlet-name>homeServlet</servlet-name>
    <servlet-class>com.myclass.servlet.HomeServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>homeServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

## ☐ Chú thích

### ❖ <servlet>

- Phần tử đại diện cho servlet

### ❖ <servlet-name>

- Phần tử con của servlet và đại diện cho tên Servlet

### ❖ <servlet-class>

- Phần tử con của servlet và đại diện cho lớp Servlet

### ❖ <servlet-mapping>

- Phần tử được sử dụng để ánh xạ Servlet

### ❖ <url-pattern>

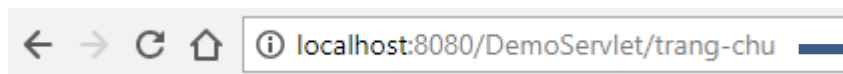
- Phần tử ánh xạ tới url của website

### ❖ <welcome-list>

- Liệt kê danh sách file khi một ứng dụng chạy lên.

# Container Handle Request

## 1. Browse



## 2. Web.xml (Container)

```
<servlet-mapping>
  <servlet-name>demoServlet</servlet-name>
  <url-pattern>/trang-chu</url-pattern>
</servlet-mapping>
```

## 3. Class Servlet

```
public class TrangChuServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

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

        resp.setContentType("text/html");
        PrintWriter print = resp.getWriter();

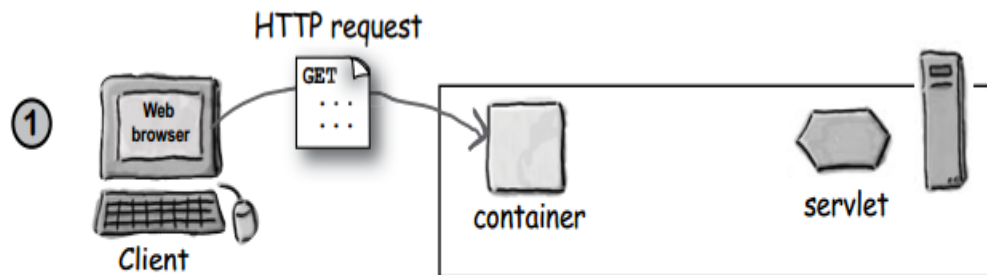
        print.println("<h1>Hello</h1>");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        super.doPost(req, resp);
    }
}
```

```
<servlet>
  <servlet-name>demoServlet</servlet-name>
  <servlet-class>com.tienhoang.servlet.TrangChuServlet</servlet-class>
</servlet>
```

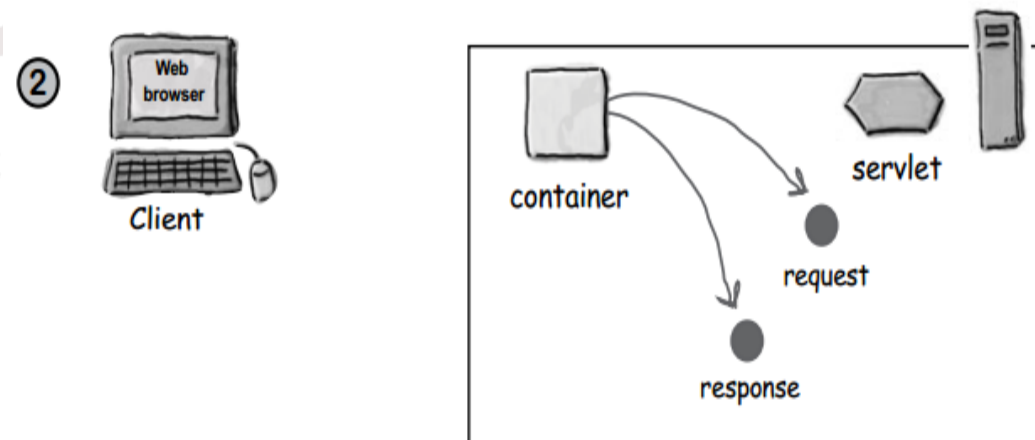
# Container Handle Request

- ❑ **Bước 1:** Người dùng **click** vào đường **link** đến **servlet**.



- ❑ **Bước 2:** **Container** nhận **request** cho **Servlet** → Tạo ra hai **đối tượng**:

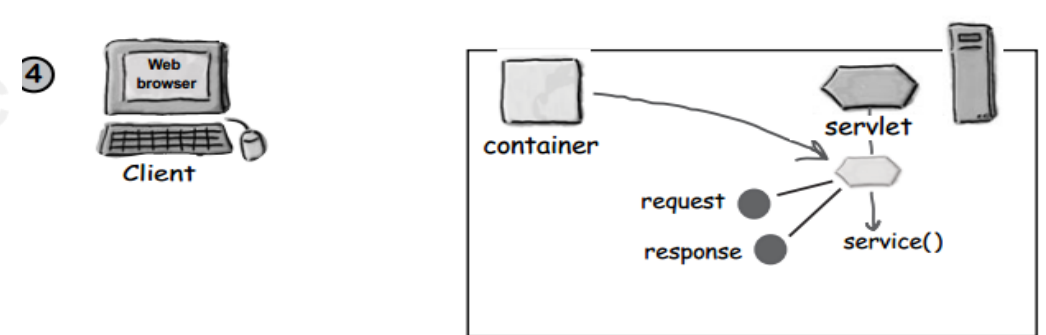
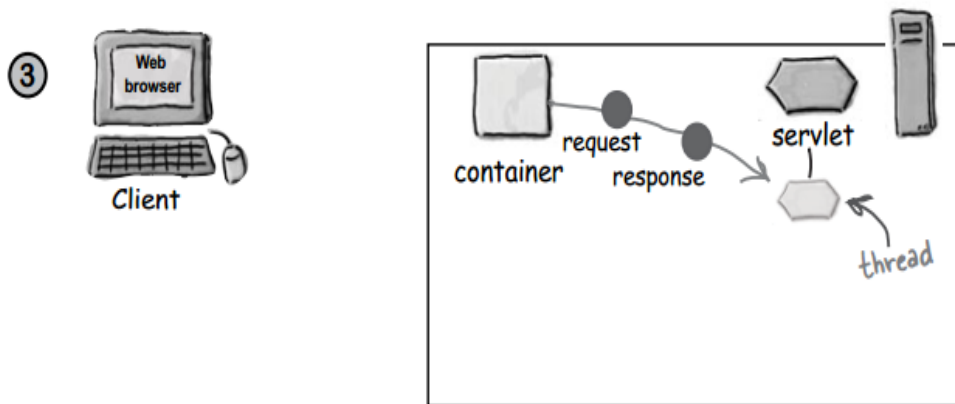
- **ServletRequest**
- **ServletResponse**



# Container Handle Request

- ❑ **Bước 3:** Container tìm Servlet dựa trên URL của request → Tạo và cung cấp 1 luồng (thread) cho request đó → Đưa hai đối tượng ServletRequest và ServletResponse vào thread này.

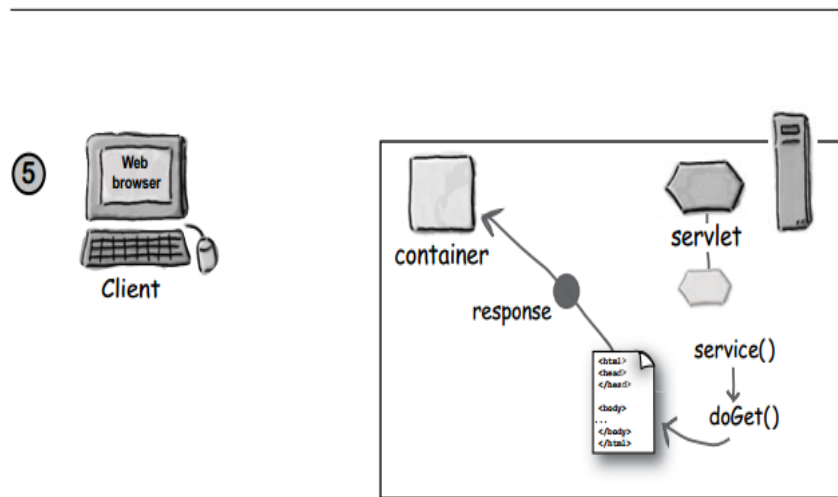
- ❑ **Bước 4:** Container gọi phương thức service() của servlet → Phương thức service() gọi phương thức doGet() hoặc doPost() tương ứng.
  - Ở đây chúng ta dùng GET.



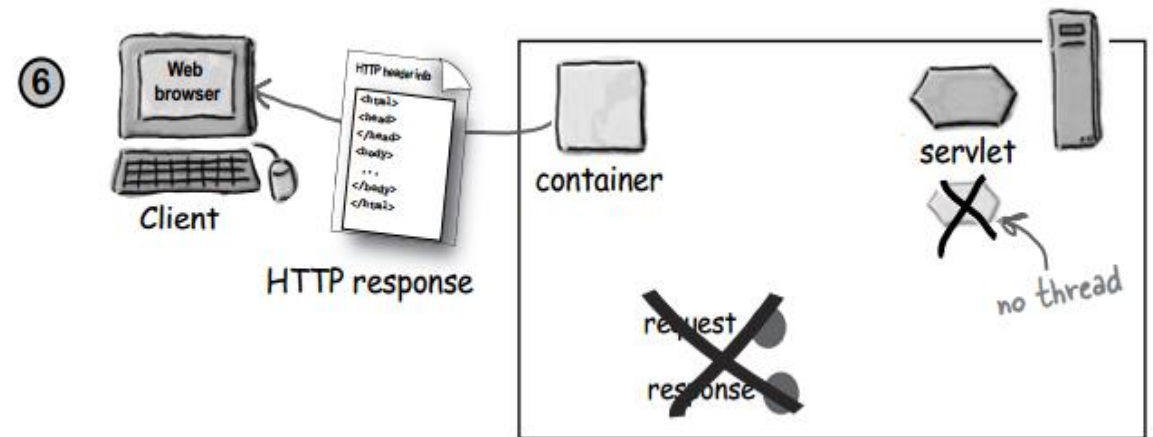


# Container Handle Request

- ❑ **Bước 5:** Phương thức `doGet()` tạo ra một trang động (`dynamic page`) và đưa vào đối tượng `ServletResponse`.



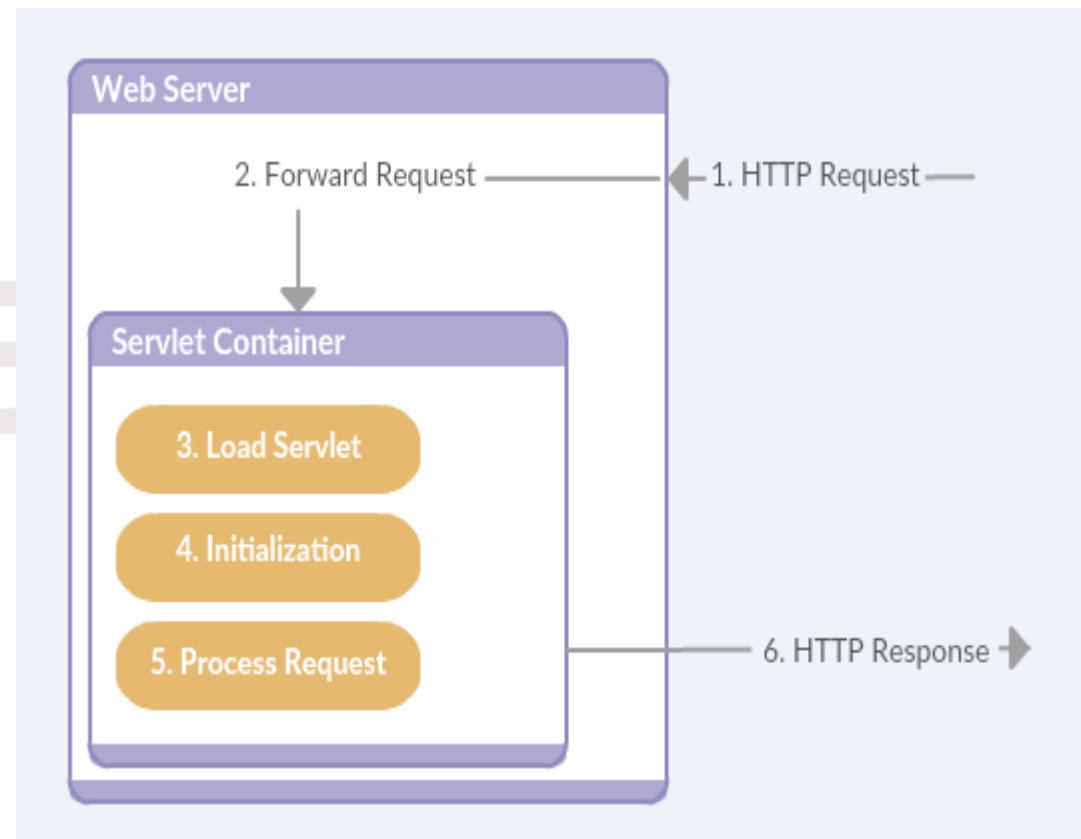
- ❑ **Bước 6:** Thread hoàn tất, Container convert đối tượng `ServletResponse` thành `Http Response` và gửi trả lại cho client. Cuối cùng xóa bỏ đối tượng `ServletRequest` và `ServletResponse`.



# Java Servlet

## ❑ Nhiệm vụ của Servlet

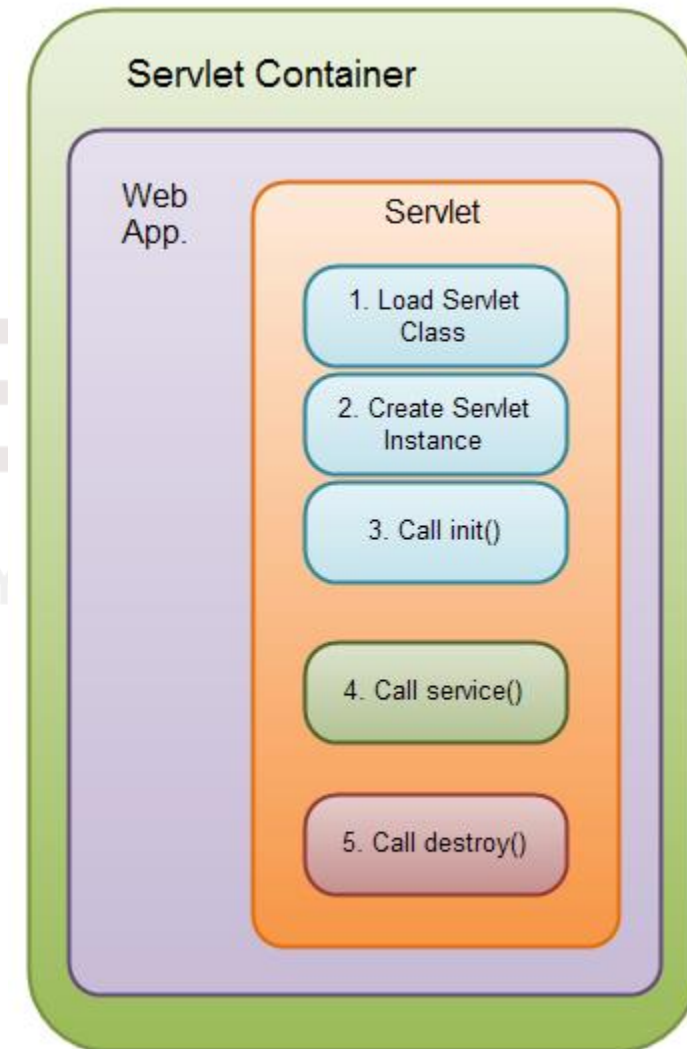
- ✓ Nhận client request.
- ✓ Trích xuất một số thông tin từ request
- ✓ Xử lý nghiệp vụ (truy vấn CSDL, gọi model,...) hoặc sinh nội dung động.
- ✓ Tạo và gửi response cho client hoặc forward request cho Servlet khác.



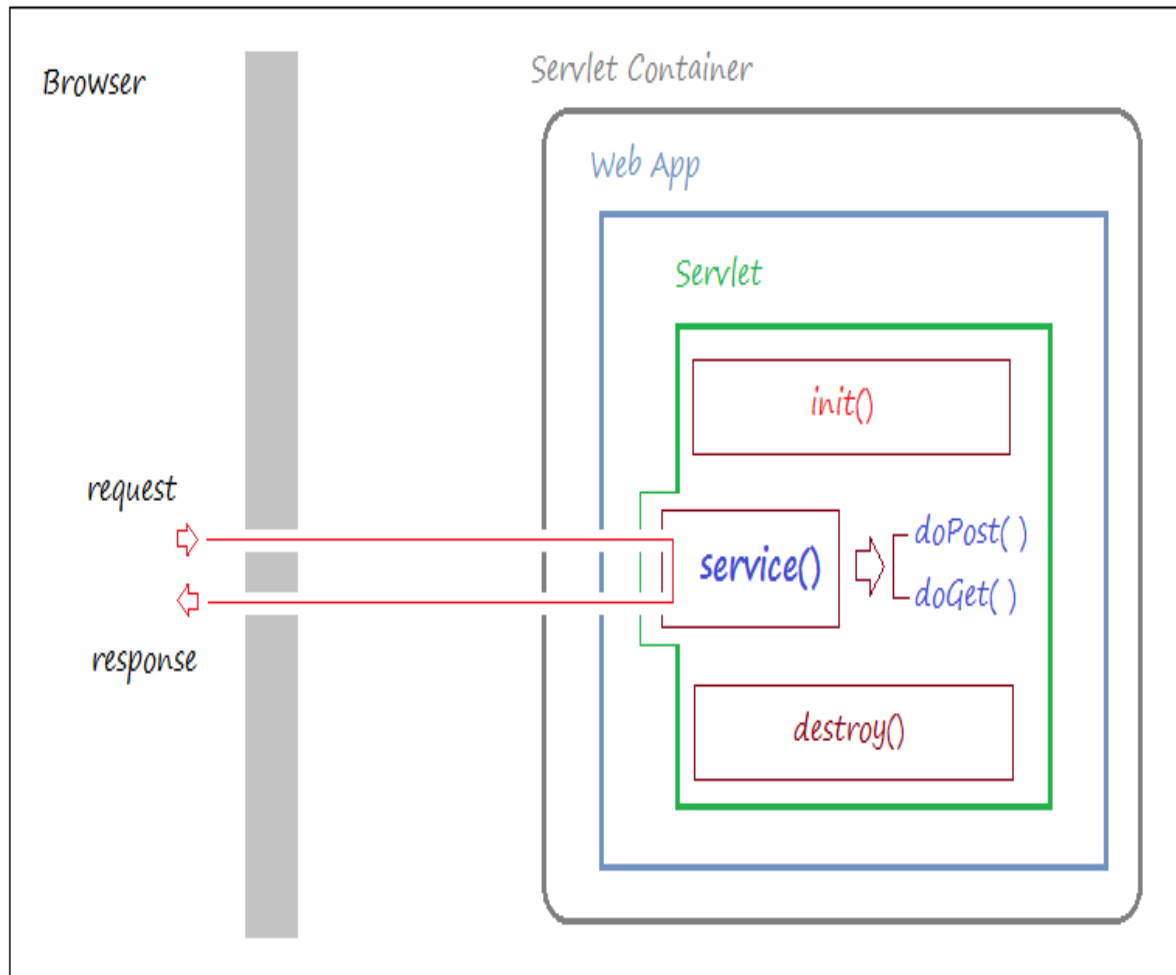
# LifeCycle Servlet

## ❑ Có 5 bước:

1. Tải **Servlet class** vào bộ nhớ.
  2. Tạo đối tượng **Servlet**.
  3. Gọi phương thức **init()** của Servlet.
  4. Gọi phương thức **service()** của Servlet.
  5. Gọi phương thức **destroy()** của Servlet.
- **Bước 1, 2, 3** được thực hiện **1 lần duy nhất khi Servlet được nạp lần đầu**.
  - **Bước 4** được **thực thi nhiều lần** khi nhận được yêu cầu từ người dùng.
  - **Bước 5** được thực thi khi bộ chứa servlet (Servlet Container) **gỡ bỏ tải Servlet**.



# LifeCycle Servlet



❖ **Bước 4 : Phương thức `service()` được gọi nhiều lần.**

➤ Ở bước này **trong mỗi request** phương thức `service()` sẽ **gọi một trong hai phương thức `doPost()` hoặc `doGet()`** tùy theo yêu cầu của người dùng.

- **`doGet()`** : Một **yêu cầu GET** xuất phát từ một yêu cầu bình thường cho một URL hoặc từ Form.
- **`doPost()`** : Một **yêu cầu POST** xuất phát từ một Form cụ thể.

# LifeCycle Servlet

## SERVLET

1. LOAD SERVLET CLASS

2. CREATE SERVLET INSTANCE

3. CALL INIT()

4. CALL SERVICE()

+ doPost()

+ doGet()

5. DESTROY

## CLASS InitParamServlet

```
12 public class InitParamServlet extends HttpServlet {
13
14     private static final long serialVersionUID = 1L;
15
16     private String myname1;
17
18     public InitParamServlet() {
19     }
20     @Override
21     public void init(ServletConfig config) throws ServletException {
22         super.init(config);
23         this.myname1 = config.getInitParameter("myname1");
24     }
25
26     @Override
27     protected void doGet(HttpServletRequest request, HttpServletResponse response)
28         throws ServletException, IOException {
29         String myname2 = this.getServletConfig().getInitParameter("myname2");
30         String name1 = this.myname1;
31         String name2 = myname2;
32         request.setAttribute("myObject1", name1);
33         request.setAttribute("myObject2", name2);
34         RequestDispatcher view = request.getRequestDispatcher("/lab1/list/InitParamServlet.jsp");
35         view.forward(request, response);
36     }
37
38     @Override
39     protected void doPost(HttpServletRequest request, HttpServletResponse response)
40         throws ServletException, IOException {
41         this.doGet(request, response);
42     }
43
44 }
```

# Phương thức GET - POST

## ❑ Phương thức GET

- Phương thức **GET** gửi dữ liệu từ client lên server thông qua các **parameter (tham số) trên url**.
- Các **tham số** mà phương thức **GET** gửi lên sẽ bắt đầu bằng **dấu ?** và nối với nhau giữa các khóa bằng **dấu &**.
- Phương thức **GET** giới hạn **tối đa 2048 ký tự**.

## ❑ Ví dụ:

- <http://abc.com.vn?hoten=cybersoft>
- <http://abc.com.vn?hoten=cybersoft&tuoi=20>

## ❑ Phương thức POST

- Phương thức **POST** gửi thông tin thông qua **Http Header** và thường được **gửi dưới dạng form**.

```
<form method="POST" action="myclass.vn">  
    Tên đăng nhập: <input type="text" name="username" />  
    Mật khẩu: <input type="password" name="email" />  
    <input type="submit" value="Gửi" />  
</form>
```

- Dữ liệu của phương thức POST được **gửi ngầm** không đưa lên url.
- Các **tham số được truyền trong request body** nên có thể truyền dữ liệu lớn.
- Có thể gửi dữ liệu **nhị phân, hình ảnh**.

# Servlet Request

## ❑ Đối tượng HttpServletRequest

- Cung cấp các **phương thức dùng để đọc các thông tin từ client** gửi lên trong **Header** của Http Request.

## ❑ `getParameter(String name)`

- Trả về kết quả từ form của trình duyệt gửi về.
- Ví dụ: `request.getParameter("ho_ten")`

## ❑ `getContextPath()`

- Trả về url của request
- Ví dụ : `request.getContextPath()`

## ❑ `getServerName()`

- Trả về địa chỉ của server
- Ví dụ: `request.getServerName()`

## ❑ `getMethod()`

- Trả về tên phương thức Http mà request thực hiện (GET, POST, ...)
- Ví dụ: `request.getMethod()`

## ❑ `getQueryString()`

- Trả về chuỗi truy vấn chứa trong Url của request .
- Ví dụ: `request.QueryString()`

## ❑ `getCookies()`

- Trả về 1 mảng chứa tất cả các đối tượng Cookie mà client đã gửi trong request.

## ❑ `getSession()`

- Trả về đối tượng HttpSession liên kết với request do client gửi lên.

# Servlet Response

## ❑ Đối tượng HttpServletResponse

Cung cấp các **phương thức dùng để gửi các thông tin phản hồi** từ server về cho client.

### ❑ void **addCookie**(Cookie **cookie**)

Thêm cookie được chỉ định vào phản hồi.

### ❑ void **sendRedirect**(String **location**)

Gửi một phản hồi chuyển hướng trang web.

➤ Ví dụ: `response.sendRedirect("trang-chu")`

### ❑ int **getStatus**()

Trả về mã trạng thái hiện tại của Response

➤ Ví dụ: `response.getStatus()`

### ❑ String **getHeader**(String **name**)

➤ Trả về giá trị cho header.

### ❑ **setHeader**(String **name**, String **value**)

➤ Đặt giá trị cho header

### ❑ void **setStatus**(int **status**)

➤ Đặt mã trạng thái cho response

➤ Ví dụ: `response.setStatus(200)`

### ❑ void **sendError**(int **status**, String **msg**)

➤ Gửi một lỗi đến client.

➤ Ví dụ: `response.sendError(404, "Không tìm thấy!")`



# sendRedirect

❑ **sendRedirect** là phương thức của đối tượng **HttpServletResponse**.

***sendRedirect(String url);***

- ❑ Phương pháp này được sử dụng để **chuyển hướng yêu cầu** của khách hàng cho một số Servlet khác để sử dụng tiếp các chức năng,
- ❑ Yêu cầu này là có thể xem như **tạo một request mới**.

```
resp.sendRedirect("/login");
```

Url của servlet cần chuyển hướng

# Forward

- ❑ Phương thức này được khai báo trong RequestDispatcher.

***forward***(ServletRequest request, ServletResponse response);

- ❑ Phương pháp này được sử dụng **chuyển tiếp các yêu cầu đến tài nguyên khác** để thực hiện các yêu cầu tiếp theo trong cùng một máy chủ, tài nguyên khác ở đây có thể là bất kỳ file servlet, trang jsp bất kỳ loại quá trình file.

```
RequestDispatcher rd = request.getRequestDispatcher("/welcome");  
rd.forward(request, response);
```

# Redirect và Forward dùng khi nào?

- ❑ Trong một ứng dụng java web ,khi chúng ta thực hiện một hành động nào đó mà **không muốn gửi dữ liệu đi,hoặc muốn chuyển sang một trang mới mà không có bất kỳ xử lý dữ liệu gì** thì chúng ta nên dùng **sendRedirect**.
- ❖ **Ví dụ:** khi nhấn nút thực hiện login thì khi xử lý xong, chúng ta sẽ chuyển hướng đến một trang nào mong muốn.
- ❑ Trong một ứng dụng java web ,khi chúng ta thực hiện một hành động nào đó mà **muốn gửi dữ liệu đi,hoặc muốn chuyển sang một trang mới xử lý dữ liệu** thì chúng ta nên dùng **forward**.
- ❖ **Ví dụ:** khi nhấn nút thực hiện login thì nếu người dùng đăng nhập sai,chúng ta gửi thông báo cho người dùng,lúc này các bạn có thể dùng `request.setAttribute` để gửi dữ liệu đi.

- ❑ Sử dụng Servlet tạo trang đăng nhập và trang chào mừng.
- ❑ Thiết kế form cho trang đăng nhập cho phép người dùng nhập thông tin username và password.
  - Lấy thông tin người dùng nhập vào bằng phương thức POST.
  - Kiểm tra thông tin đăng nhập:
    - + Nếu username = “admin” và password trùng với “123456” thì chuyển hướng về trang welcome.
    - + Nếu không đúng thì quay lại trang đăng nhập.

- ❖ **Bước 1:** Tạo dự án có tên BaiTap01
- ❖ **Bước 2:** Tạo Servlet Login và Welcome.
- ❖ **Bước 3:** Tạo trang form đăng nhập.
- ❖ **Bước 4:** Cấu hình web container.
- ❖ **Bước 5:** Trong Servlet Login, override lại phương thức doPost → Sử dụng request.getParameter để lấy dữ liệu người dùng gửi lên.

# Tổng kết

- ☐ Hiểu về mô hình và cách hoạt động của một trang web.
- ☐ Hiểu về cách truyền dữ liệu giữa server và client.
- ☐ Nắm được cách hoạt động của Java Servlet.
- ☐ Tạo dự án sử dụng Servlet
- ☐ Cấu hình Servlet Vòng đời của Servlet
- ☐ Request – Response
- ☐ Phương thức GET – POST
- ☐ Phân biệt Redirect – Forward