

Name: Quang Thuận

SĐT: 0383789311

E-mail: quangthuansmt@gmail.com

1. Java Servlet 03 - Chu kỳ sống của Servlet và Ví dụ Hello Servlet
2. Java Servlet 04 - Cấu hình Java Servlet bằng XML
3. Java Servlet 05 - Cấu hình Java Servlet bằng Java Annotation
3. Java Servlet 06 - ServletRequest trong Java Servlet
4. Java Servlet 07 - ServletResponse trong Java Servlet
5. Java Servlet 08 - ServletConfig trong Java Servlet
6. Java Servlet 09 - HTTP Response Code trong Java Servlet
7. Java Servlet 10 - Đọc dữ liệu gửi lên từ client qua URL trong Java Web
8. Java Servlet 11 - Đọc dữ liệu từ POST và GET trong HTML Form
9. Java Servlet 13 - Redirect Chuyển hướng trang web trong Java Servlet
10. Java Servlet 15 - ServletContext trong Java Servlet
11. Bài 1 - Giới thiệu về JSP
12. Bài 2 - Jsp - ví dụ Hello World
13. Bài 3 - Jsp - HttpServletRequest
14. Bài 4 - JSp - HttpServletResponse
15. Bài 5 - JSP Form
16. Java Web 17 - Java Servlet - RequestDispatcher trong Java Servlet



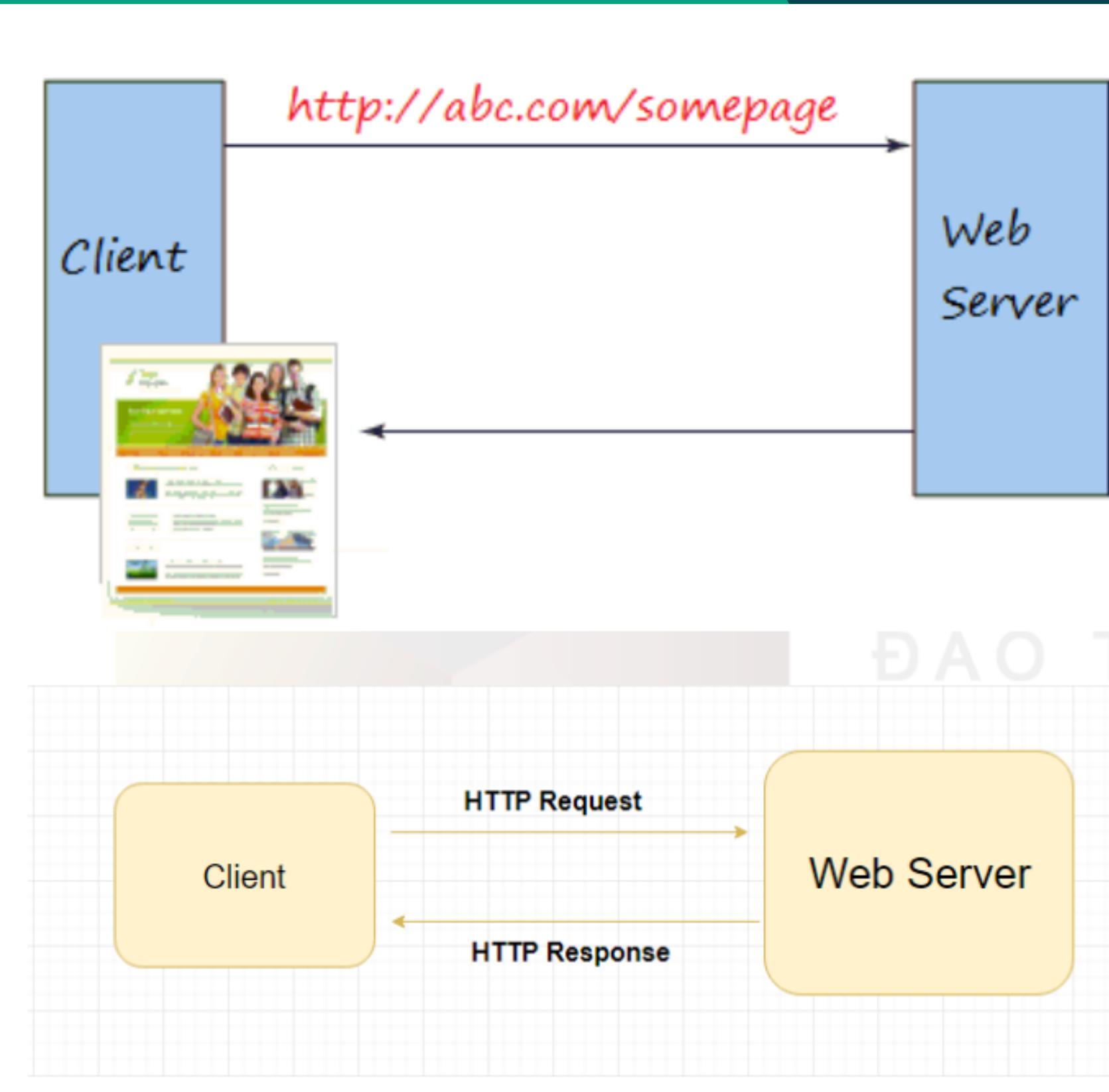
JAVA SERVLET

SERVLET LÀ GÌ ?

SỬ DỤNG SERVLET NHƯ THẾ NÀO ?

MỤC ĐÍCH CỦA SERVLET ?

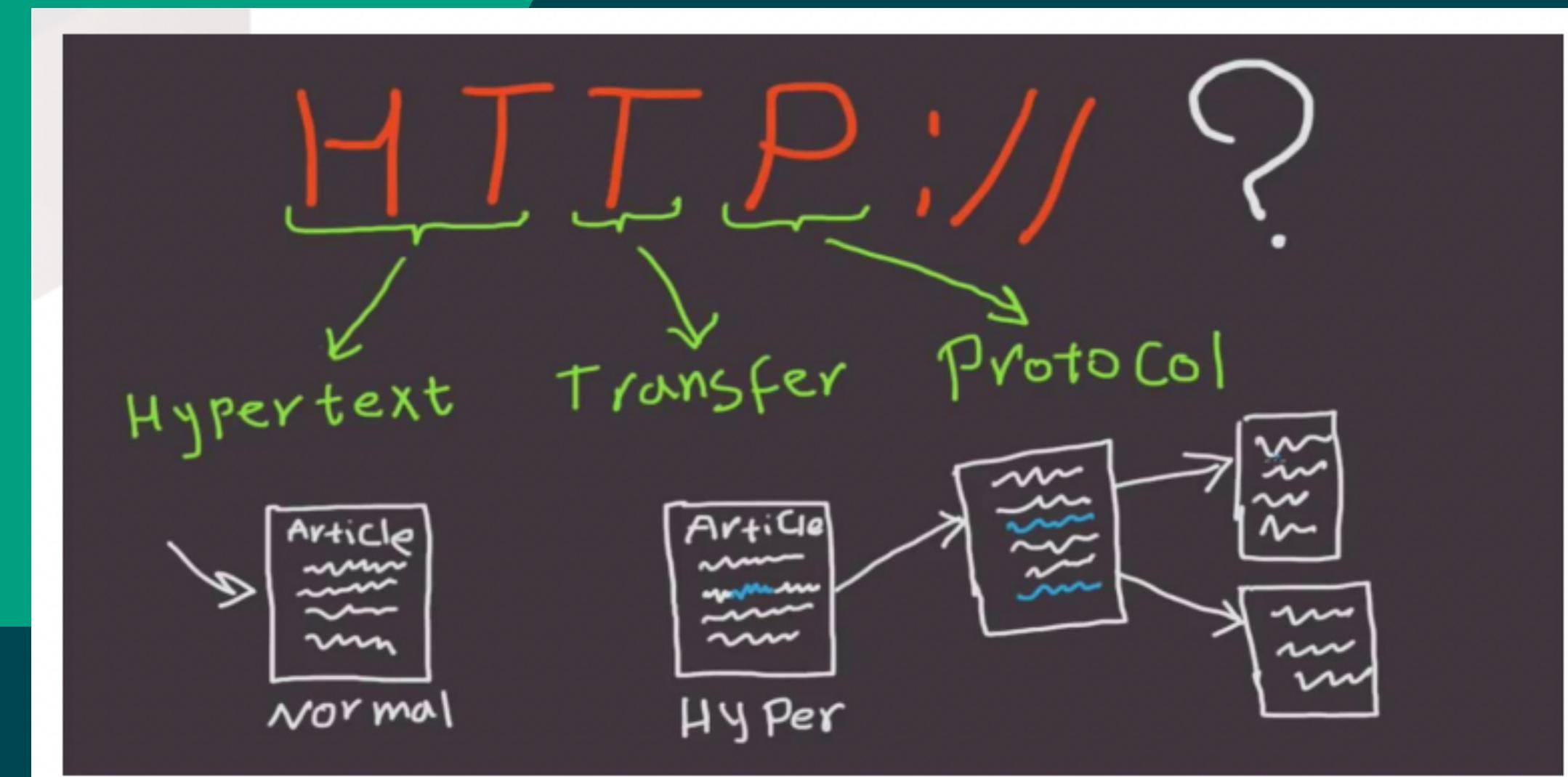
WEB SERVER ?



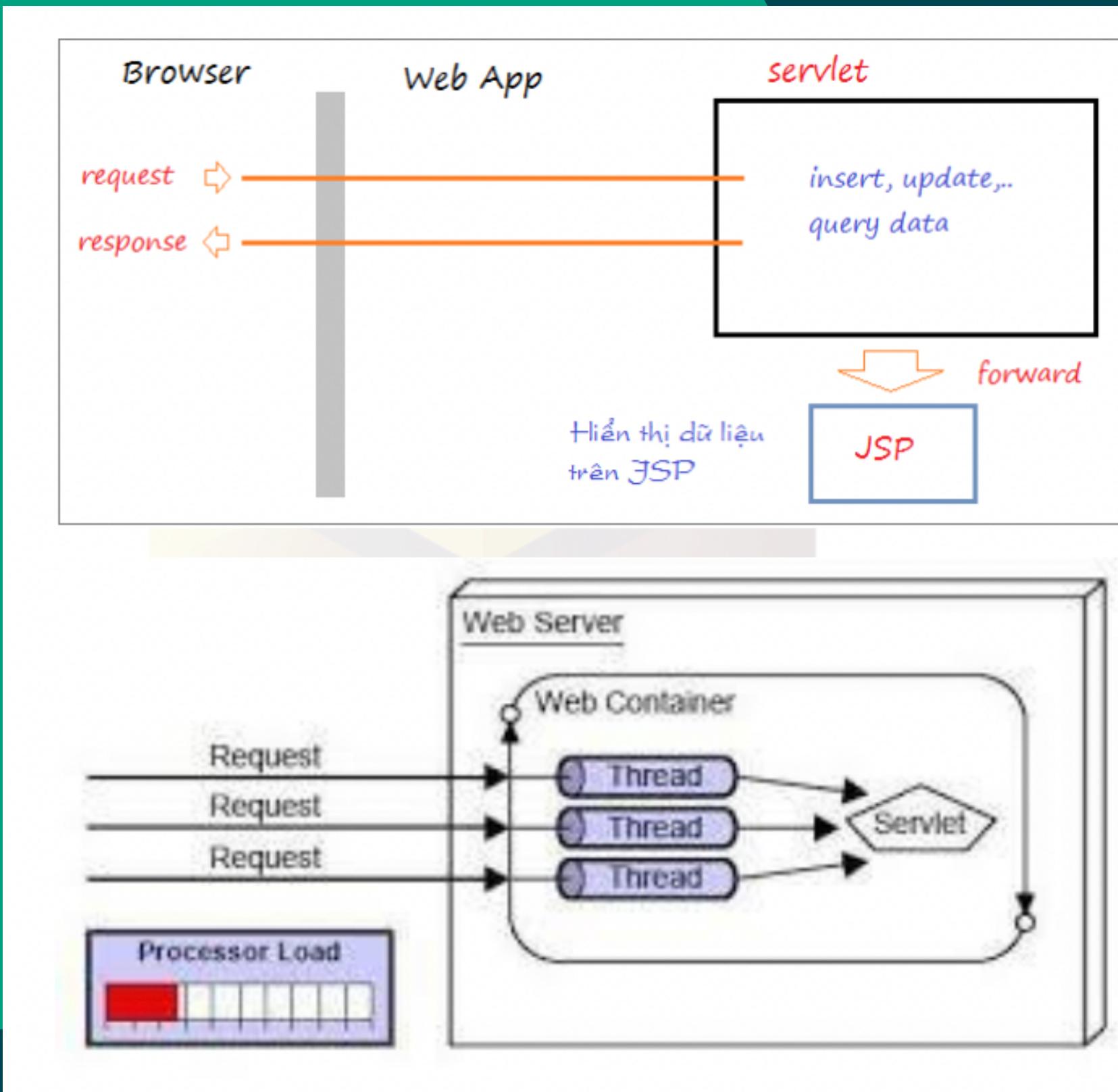
- Web Server là máy chủ cài đặt chương trình (phần mềm) phục vụ cho các ứng dụng web
- Web Server có khả năng tiếp nhận các yêu cầu (request) từ các trình duyệt web (browsers) và gửi phản hồi (response) đến máy khách hàng (client) thông báo giao thức HTTP hoặc các giao thức khác
 - HTTP Request: Yêu cầu từ client
 - HTTP Response: Phản hồi từ 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 phải hồi các yêu cầu của Client qua Response



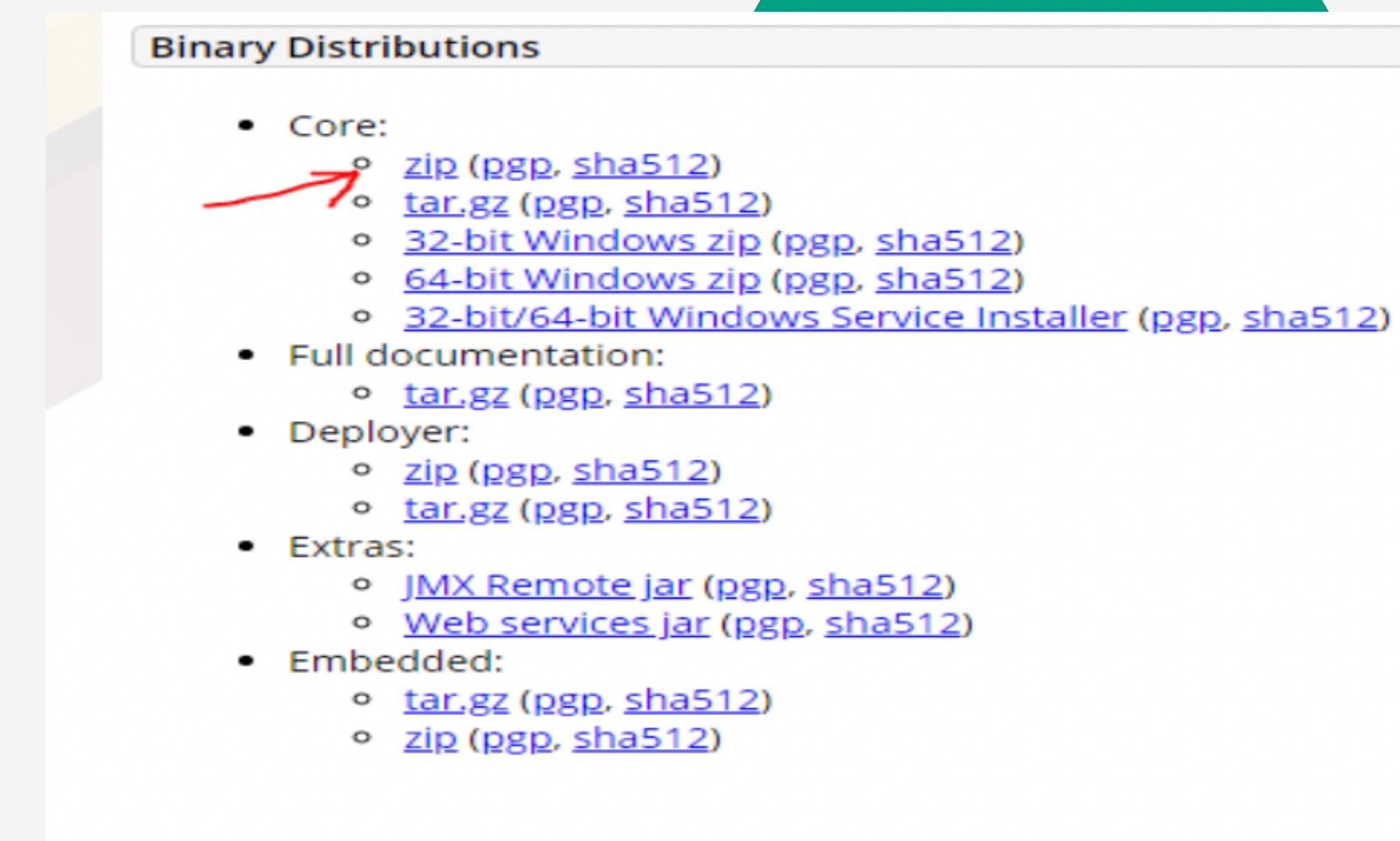
SERVLET - WHAT ? - HOW ? - WHEN ?



- 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
- Servlet mạnh mẽ chịu tải tốt
- Servlet interface khai báo 3 phương thức cần thiết cho 1 vòng đời của một servlet là `init()`, `service()` và `destroy()`. Các method này được cài đặt bởi Servlet và được thực thi bởi Servlet Container

Cài đặt Server

- Để run được servlet chúng ta cần install tomcat vào, nó đóng vai trò như là 1 con server để khi run application có thể build được ứng dụng
- Download Tomcat - bằng đường link sau đây
<https://tomcat.apache.org/download-90.cgi>
- Vào link tải về và giải nén TomCat



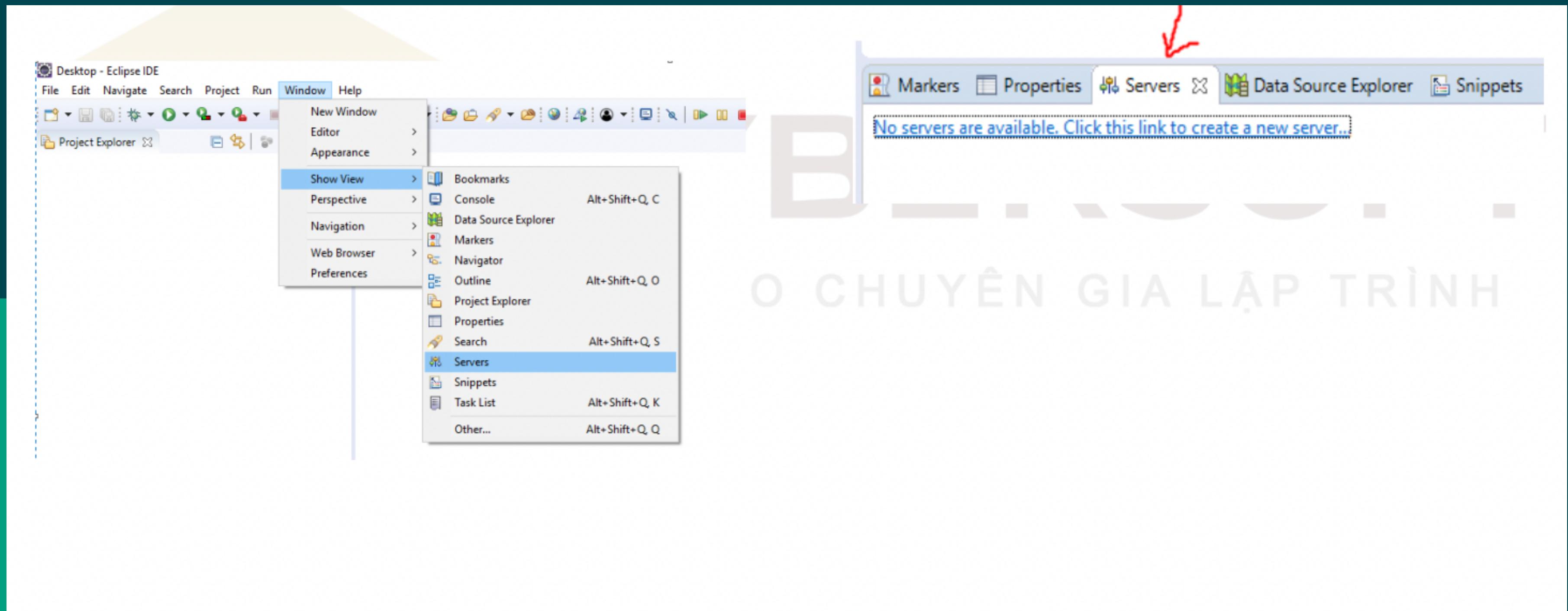
The screenshot shows the 'Binary Distributions' section of the Apache Tomcat download page. It lists several options under 'Core':

- 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 \(pgp, sha512\)](#)

A red arrow points to the first item in the 'Core' list: [zip \(pgp, sha512\)](#).

Cài đặt Server

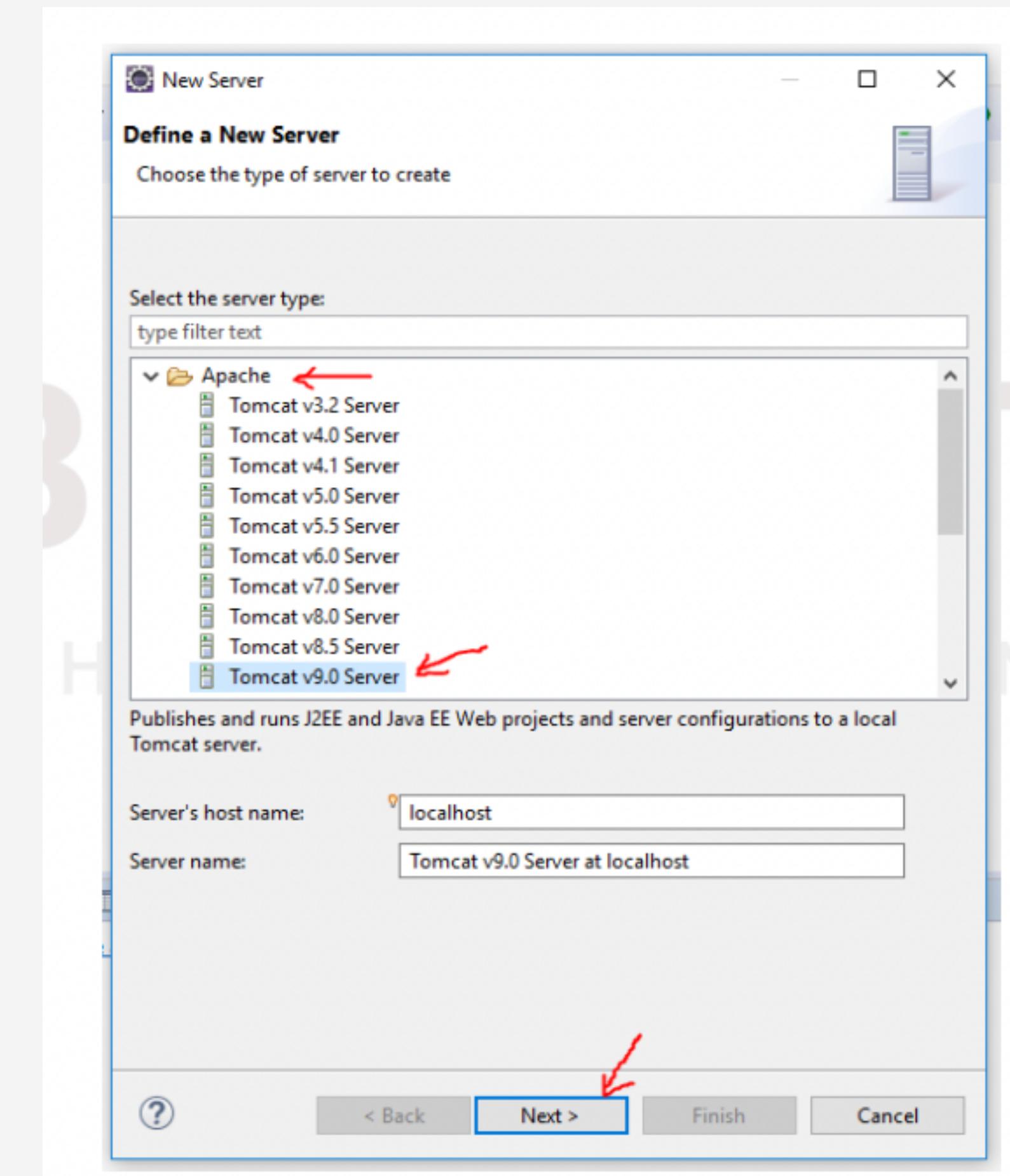
Click vào **Window** -> **Show View** --> click vào tab **Server** --> **View** --> chọn **Server**



Cài đặt server

Click vào chuột phải --> chọn tab Server
--> Chọn New --> Chọn Server

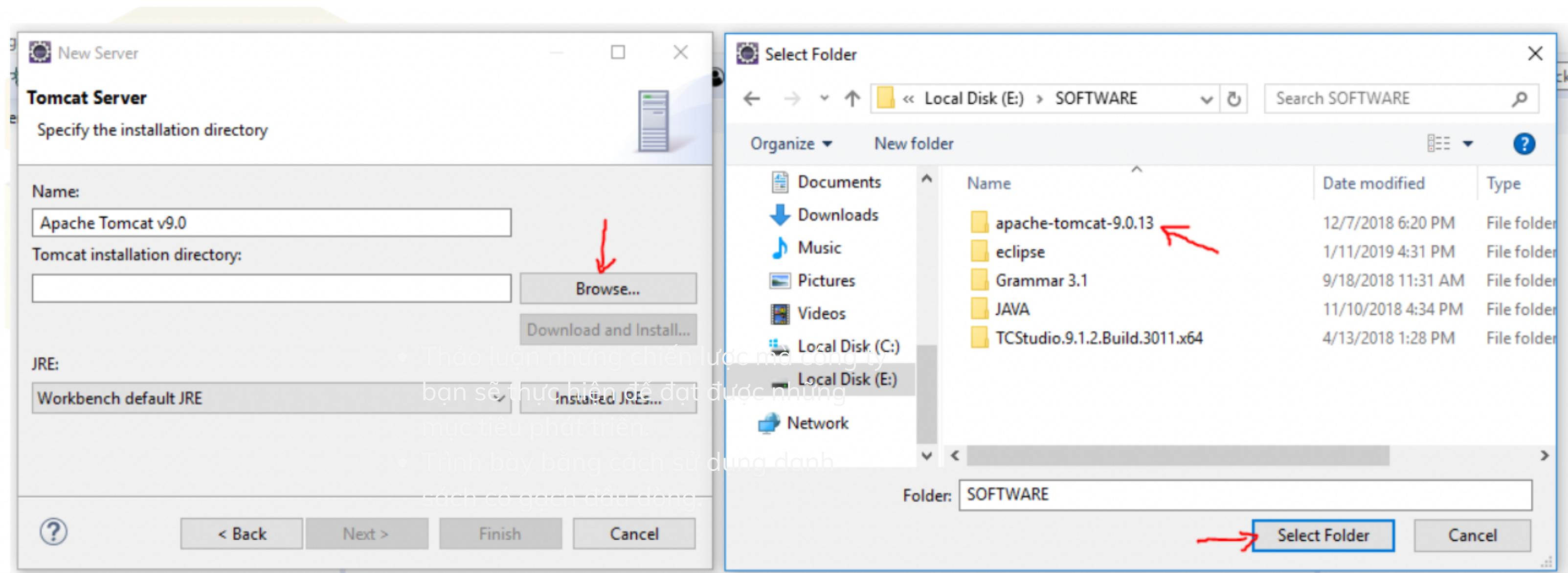
Cửa sổ hiện ra --> Click vào Apache
--> chọn Tomcat có version trùng
với bản vừa tải về --> chọn Next



Cài đặt Server

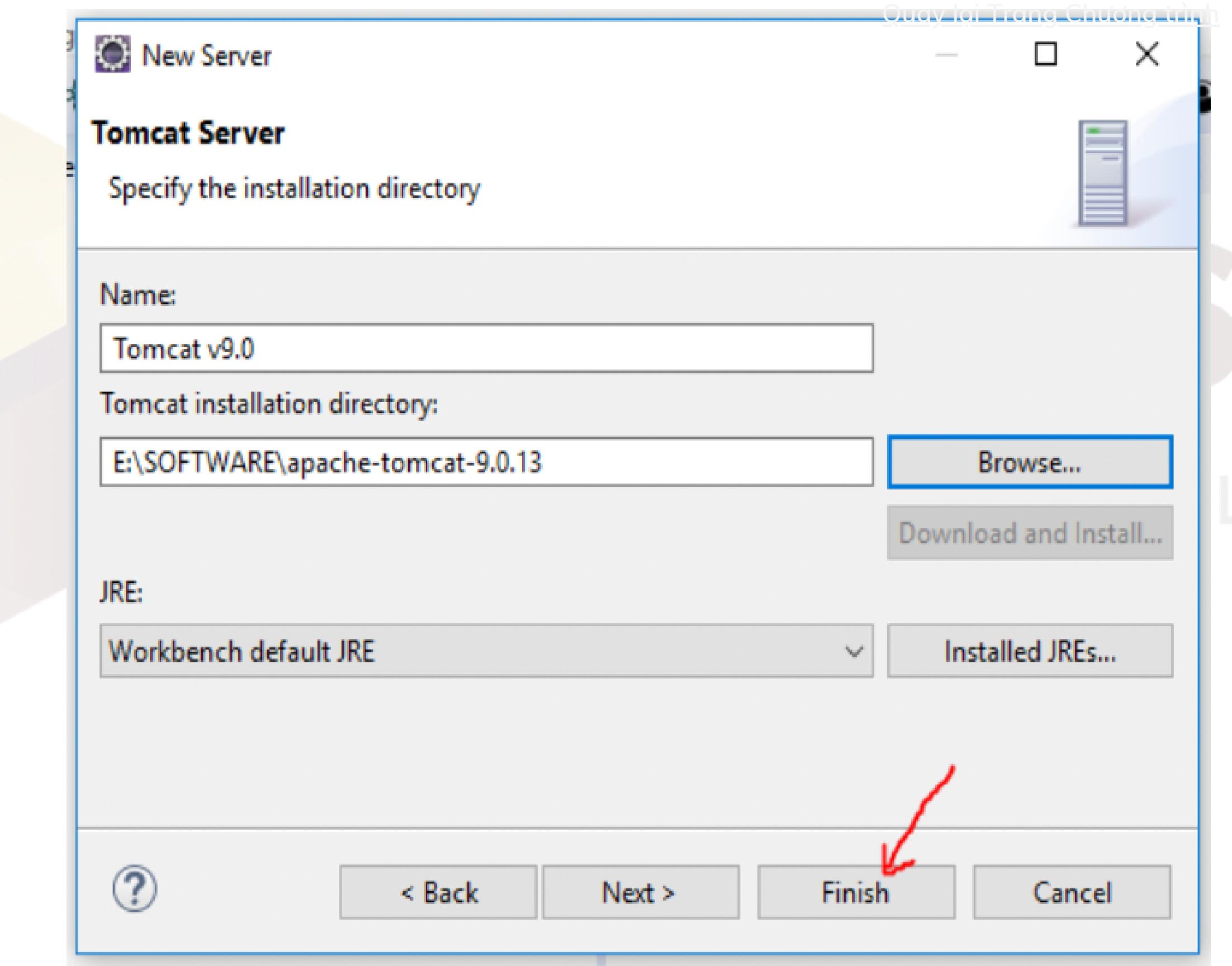
[Quay lại Trang Chương trình](#)

Click vào Browse -> mở thư mục chứa Tomcat vừa giải nén -> chọn thư mục apache-tomcat -> click vào Select Folder



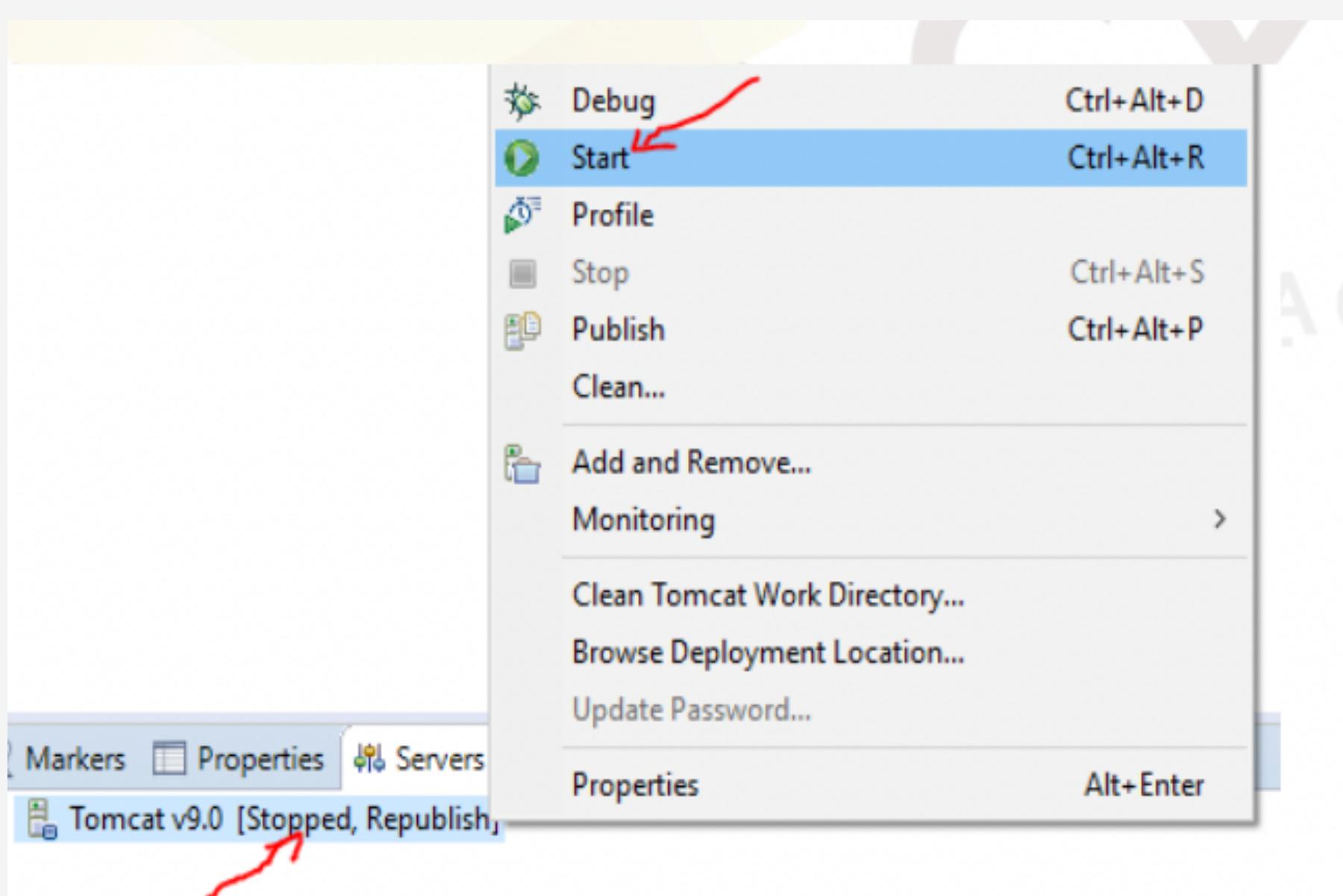
Cài đặt Server

Chọn Finish để hoàn thành

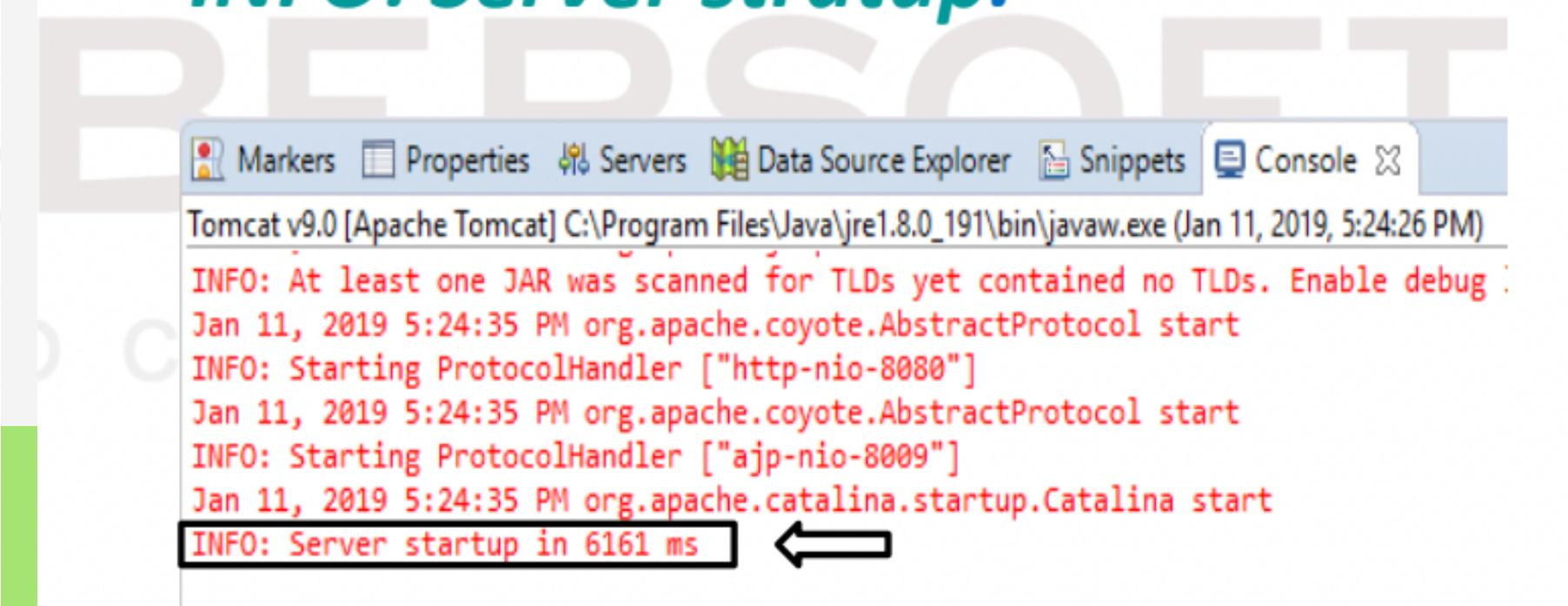


Cài đặt Server

- Chạy thử
- Click vào chuột phải Server vừa cài đặt --> Chọn Start để chạy Server



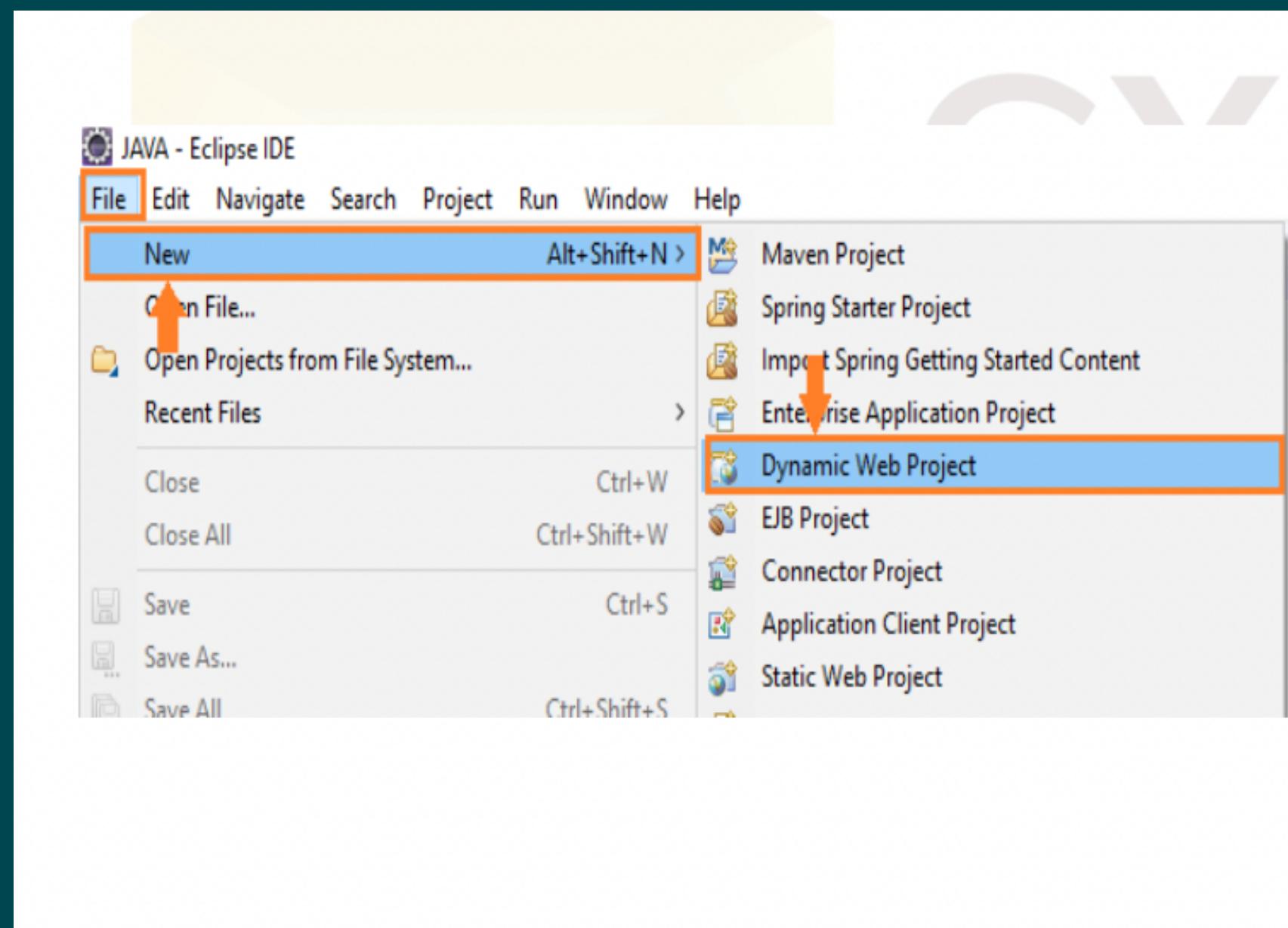
❖ **Server Tomcat** được cài đặt thành công nếu trong tab **Console** xuất hiện thông báo **INFO: Server stratus.**



Tạo mới dự án

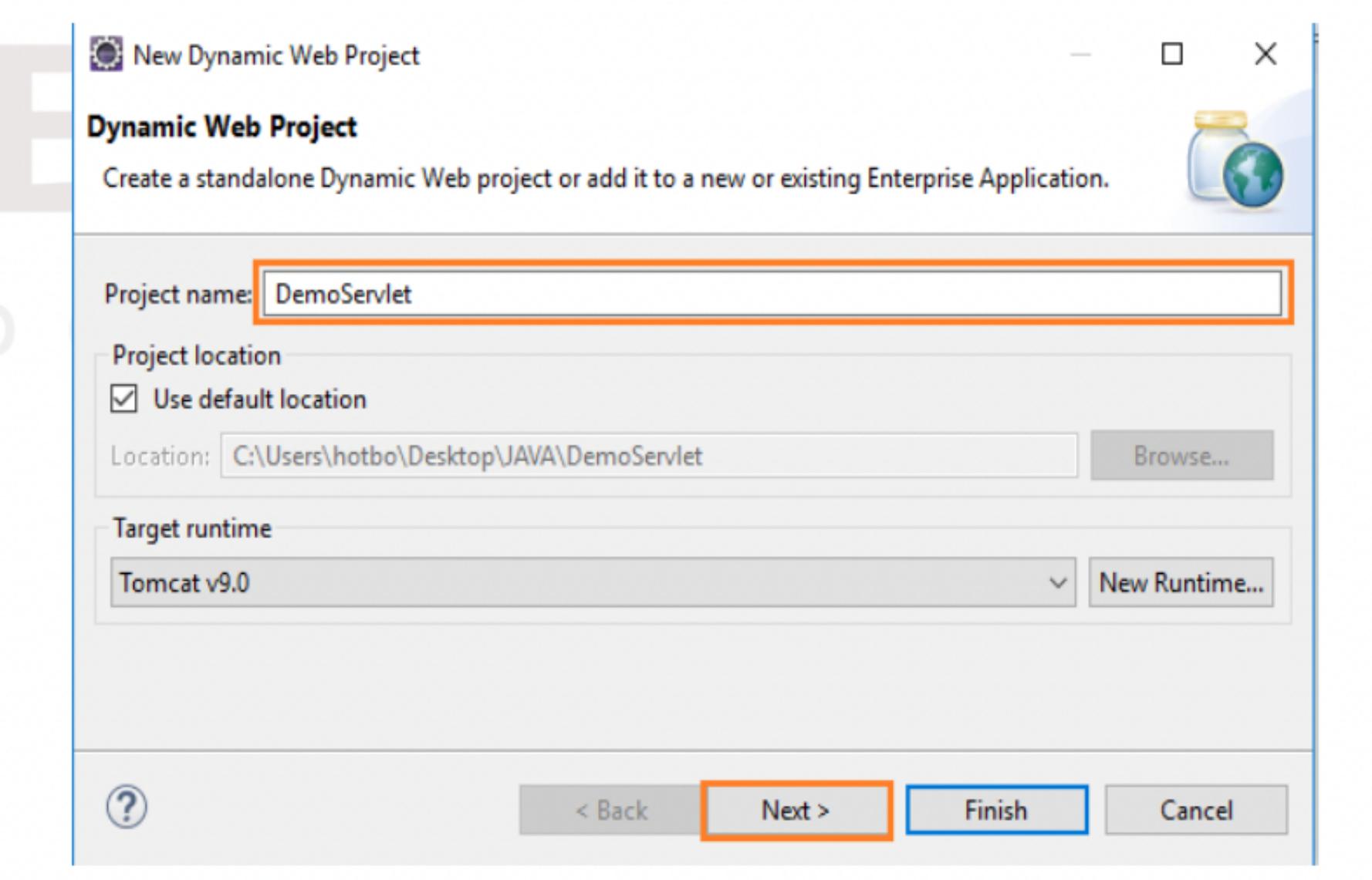
Bước 1:

Click vào **File** -> Chọn **New** -> Chọn **Dynamic Web Project**



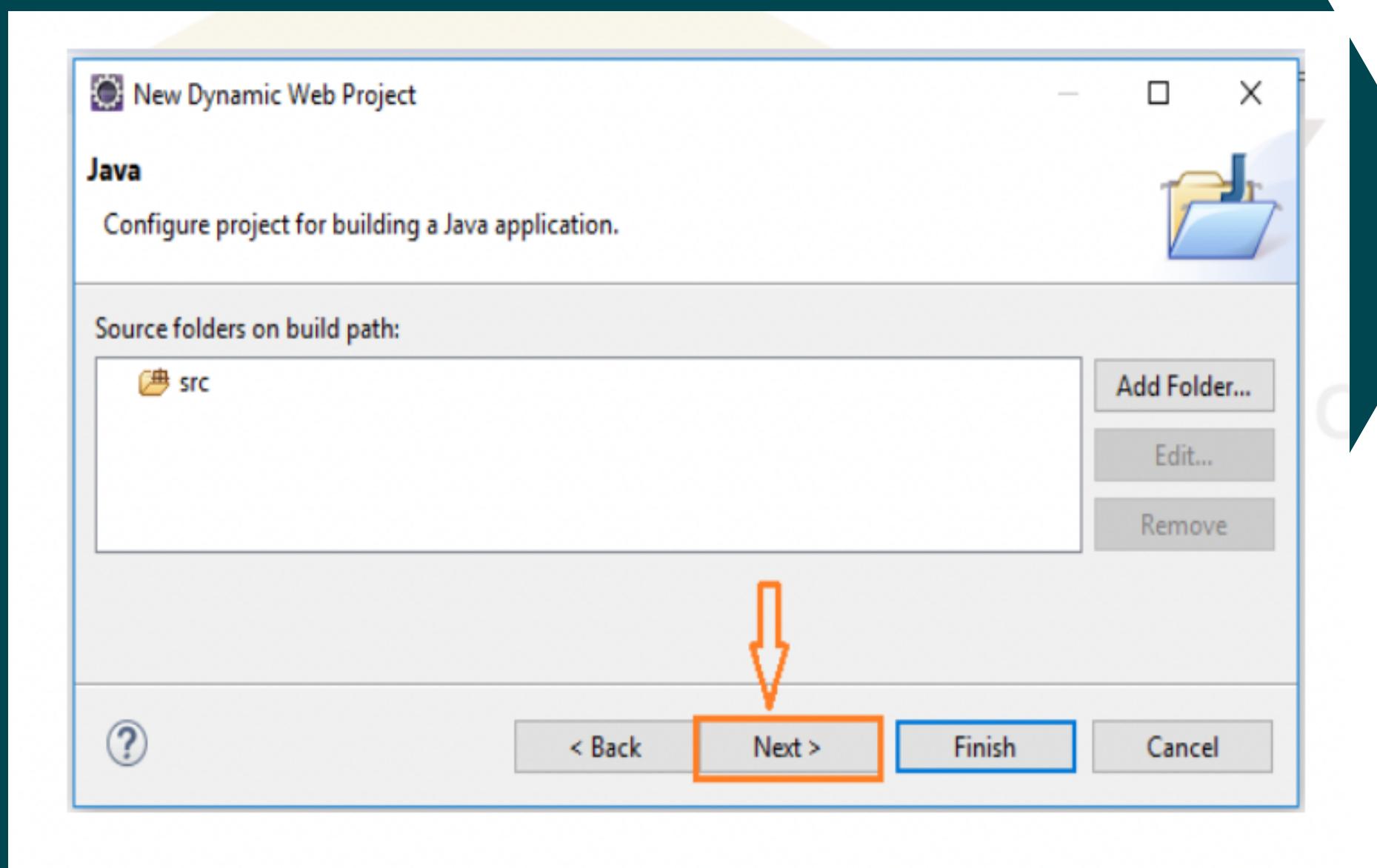
Bước 2:

Đặt tên cho Project --> Chọn **Next**

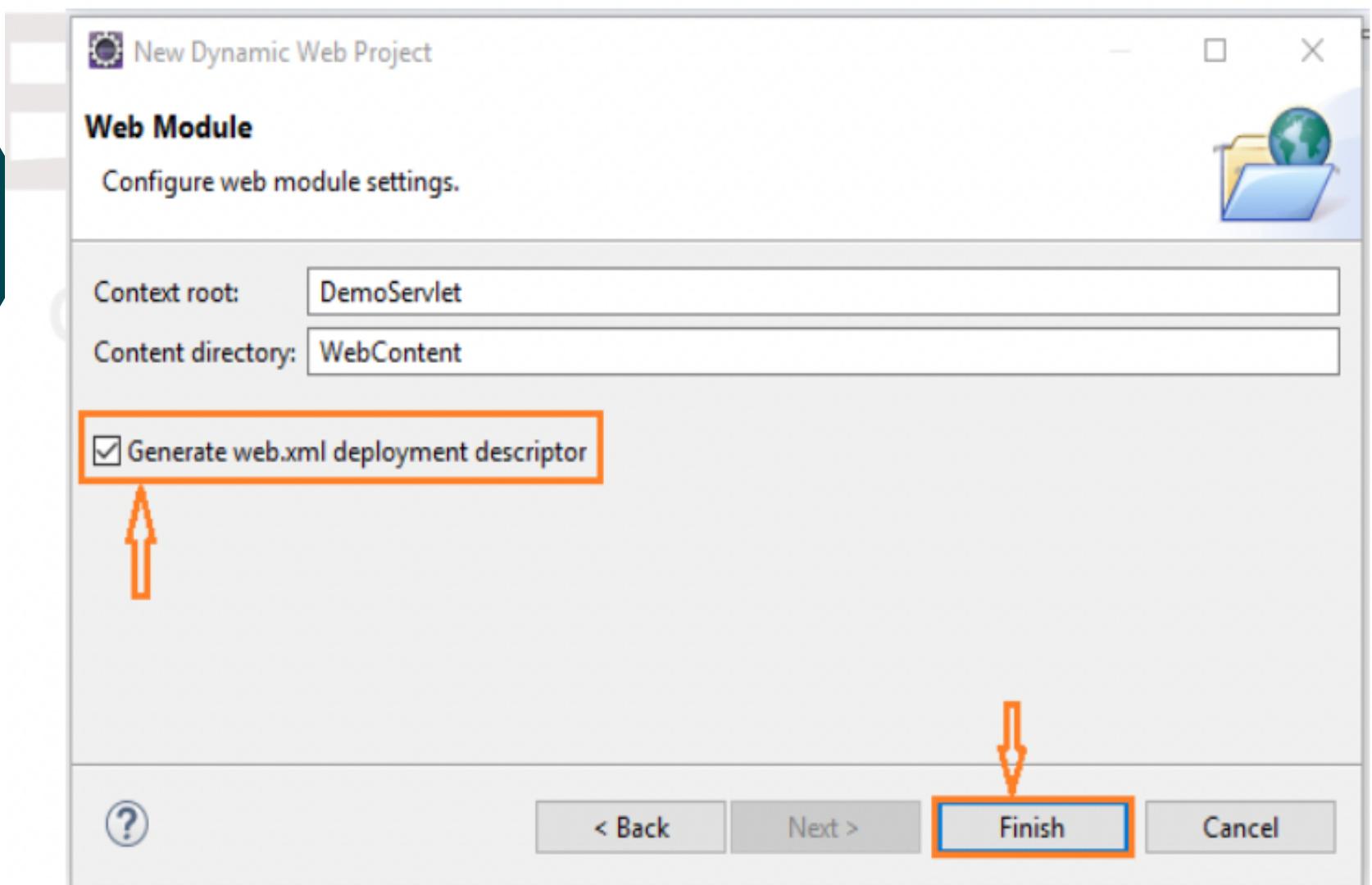


Tạo mới dự án

Bước 3:
Chọn **Next**



Bước 4:
Tích chọn **Generate web.xml** ->
chọn **Finish** để kết thúc



Có 2 cách để cấu hình url trên servlet

Sử dụng annotation

```
1 usage
@WebServlet(name = "helloServlet", value = "/hello-servlet")
public class HelloServlet extends HttpServlet {
    2 usages
    private String message;

    public void init() { message = "Hello World!!!!!!"; }

    6 usages
    public void doGet(HttpServletRequest request, HttpServletResponse
        response.setContentType("text/html");

        // Hello
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>" + message + "</h1>");
        out.println("</body></html>");
    }
}
```

Sử dụng file xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app 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
    version="4.0">

    <servlet>
        <servlet-name>helloServlet</servlet-name>
        <servlet-class>com.example.demo.servlet.HelloServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>helloServlet</servlet-name>
        <url-pattern>/home-page</url-pattern>
    </servlet-mapping>

```

Cấu hình container (web.xml)

Để ứng dụng Servlet có thể hoạt động được thì chúng ta cần cấu hình thông qua 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

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
    <display-name>DemoServlet</display-name>
    ...
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.jsp</welcome-file>
        <welcome-file>default.htm</welcome-file>
    </welcome-file-list>
    ...
    <servlet>
        <servlet-name>HomeServlet</servlet-name>
        <servlet-class>com.demo.servlet.HomeServlet</servlet-class>
    </servlet>
    ...
    <!-- Map all request to DispatcherServlet for handling -->
    <servlet-mapping>
        <servlet-name>HomeServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

- **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
- **wellcome-list** -> liệt kê danh sách file khi một ứng dụng được chạy lên

Container Handle Request

localhost:8080/DemoServlet/trang-chu

```
@WebServlet("/HomeServlet")  
public class HomeServlet extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
    private String message;  
    /** * @see HttpServlet#HttpServlet()  
    */  
    public HomeServlet() {  
        super();  
        // TODO Auto-generated constructor stub  
    }  
    /** * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse)  
    */  
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {  
        // TODO Auto-generated method stub  
        response.getWriter().append("Served at: ").append(request.getContextPath());  
        // Set response content type  
        response.setContentType("text/html");  
        // Actual logic goes here.  
        PrintWriter out = response.getWriter();  
        out.println("<h1>" + message + "</h1>");  
    }  
}
```

```
<!-- Map all request to DispatcherServlet for handling -->  
<servlet-mapping>  
    <servlet-name>HomeServlet</servlet-name>  
    <url-pattern>/trang-chu</url-pattern>  
</servlet-mapping>
```

Container Handle Request

```
/*
 * .implementation.class.HomeServlet
 */
//@WebServlet("")
@WebServlet(name = "/HomeServlet", urlPatterns = {"/HomeServlet/*"})
public class HomeServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private String message;
    ...
    /**
     * @see HttpServlet#HttpServlet()
     */
    public HomeServlet() {
        super();
        // TODO Auto-generated constructor stub
    }
    ...
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ...
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
        // Set response content type
        response.setContentType("text/html");
        ...
        // Actual logic goes here
    }
}
```

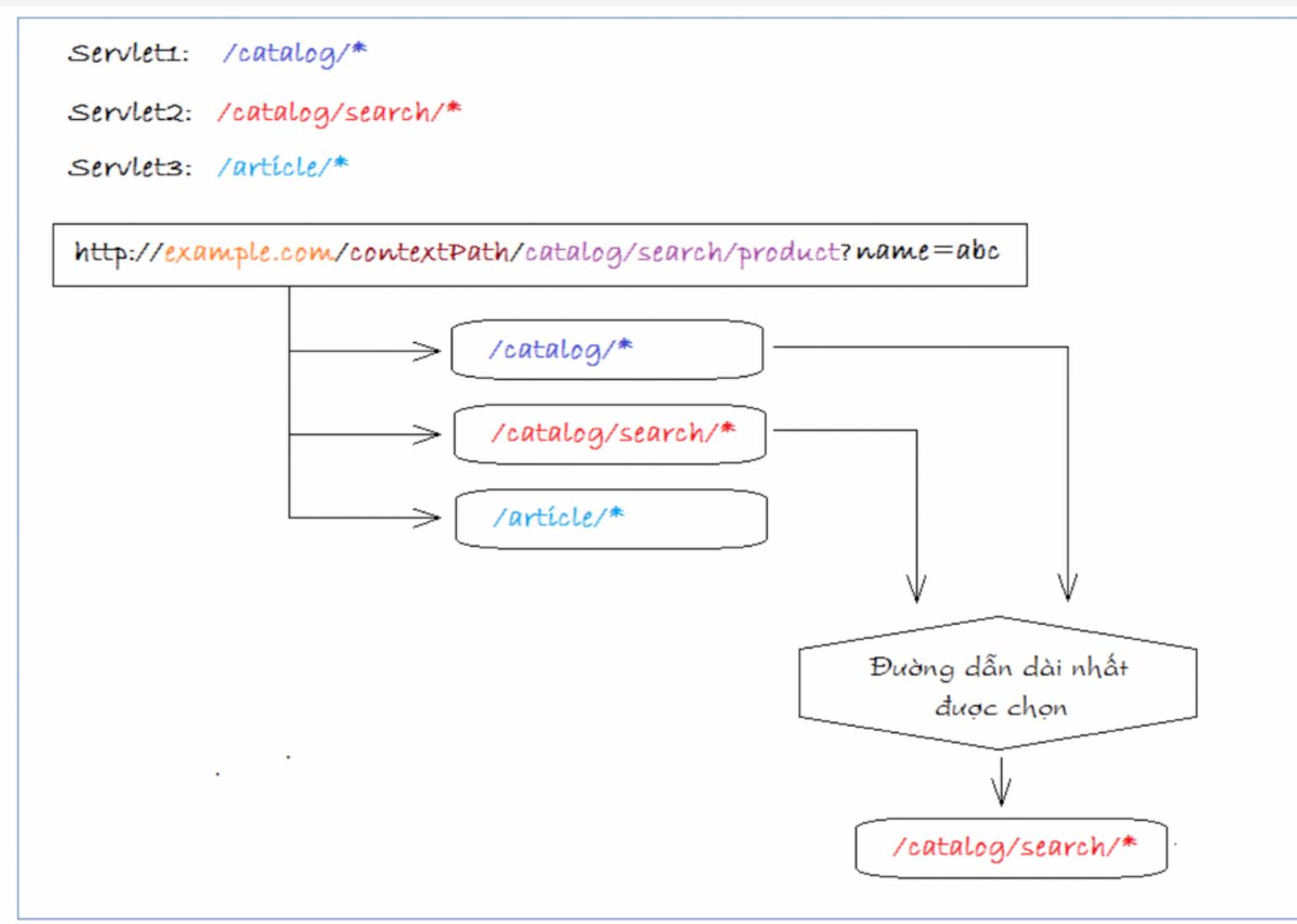
name -> định danh khi ta muốn truy cập vào controller đó theo một cái tên khác, không dựa vào file web.xml

urlPattern -> thiết lập một pattern để có thể truy cập vào url đó theo nhiều định dạng

Nếu ta để HomeServlet hoặc Homervlet/1234 thì đều có thể truy cập vào được vì có sử dụng

Container Handle Request

Khi người dùng nhập vào 1 đường dẫn trên trình duyệt, nó sẽ gửi vào WebContainer, WebContainer sẽ quyết định servlet nào thực hiện request đó



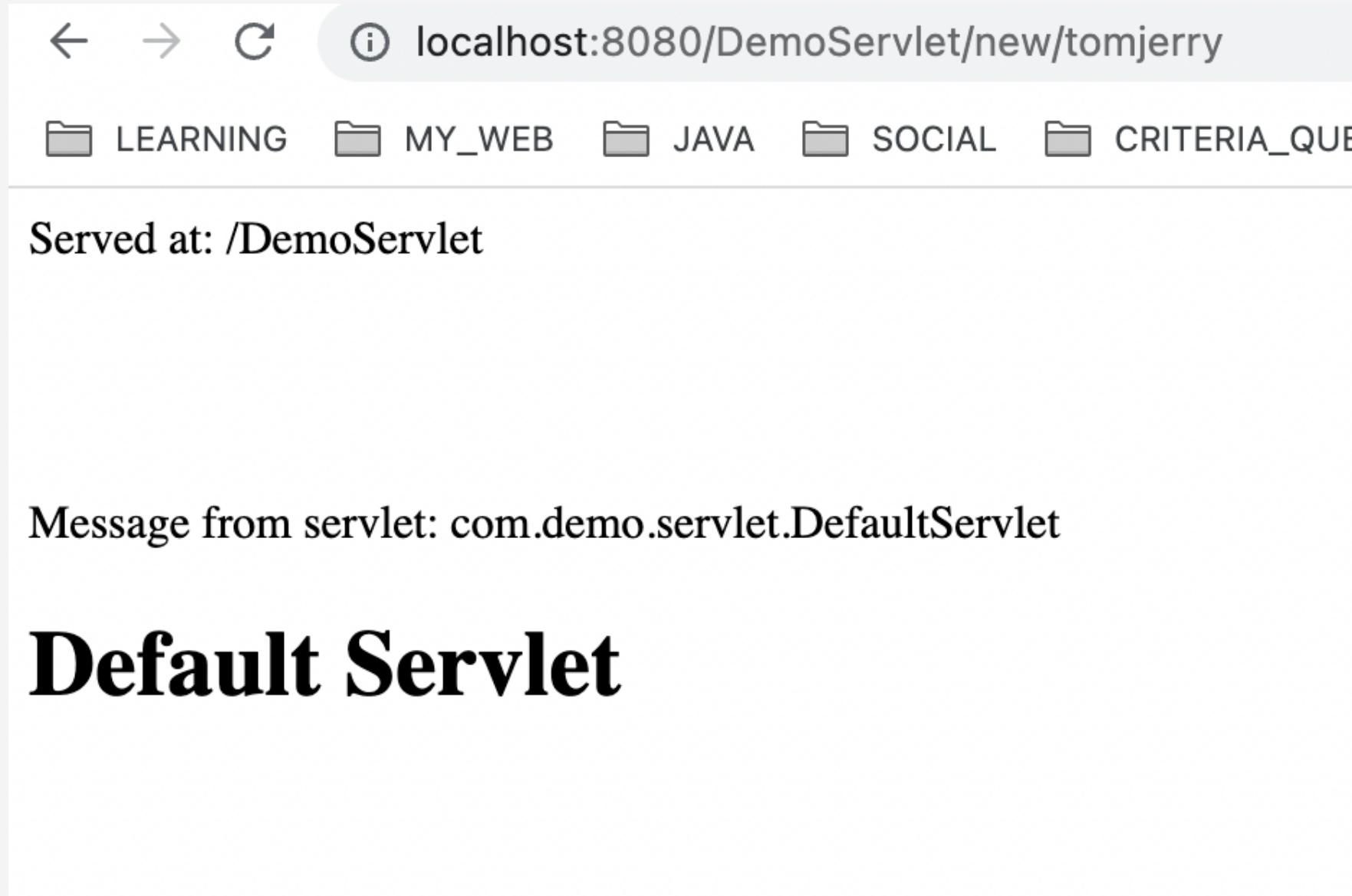
Nếu ta tạo 2 controller mapping và để chung 1 value cho path thì dẫn tới khi compile time, chương trình sẽ bị lỗi và không thể run được như dưới

```
... 21 more
Caused by: org.apache.catalina.LifecycleException: Failed to start component [StandardEngine[...
    at org.apache.catalina.util.LifecycleBase.handleSubClassException(LifecycleBase.java:...
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:198)
    at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1396)
    at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1386)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at org.apache.tomcat.util.threads.InlineExecutorService.execute(InlineExecutorService...
    at java.base/java.util.concurrent.AbstractExecutorService.submit(AbstractExecutorServ...
    at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:919)
    ... 21 more
Caused by: java.lang.IllegalArgumentException: The servlets named [com.demo.servlet.HomeServlet] ...
    at org.apache.tomcat.util.descriptor.web.WebXml.addServletMappingDecoded(WebXml.java:...
    at org.apache.tomcat.util.descriptor.web.WebXml.addServletMapping(WebXml.java:336)
    at org.apache.catalina.startup.ContextConfig.processAnnotationWebServlet(ContextConfig...
    at org.apache.catalina.startup.ContextConfig.processClass(ContextConfig.java:2359)
    at org.apache.catalina.startup.ContextConfig.processAnnotationsStream(ContextConfig.ja...
    at org.apache.catalina.startup.ContextConfig.processAnnotationsWebResource(ContextConfig...
    at org.apache.catalina.startup.ContextConfig.processAnnotationsWebResource(ContextConfig...
    at org.apache.catalina.startup.ContextConfig.processAnnotationsWebResource(ContextConfig...
    at org.apache.catalina.startup.ContextConfig.processAnnotationsWebResource(ContextConfig...
    at org.apache.catalina.startup.ContextConfig.processClasses(ContextConfig.java:1398)
    at org.apache.catalina.startup.ContextConfig.webConfig(ContextConfig.java:1303)
    at org.apache.catalina.startup.ContextConfig.configureStart(ContextConfig.java:986)
    at org.apache.catalina.startup.ContextConfig.lifecycleEvent(ContextConfig.java:303)
    at org.apache.catalina.util.LifecycleBase.fireLifecycleEvent(LifecycleBase.java:123)
    at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5135)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:102)
```

Container Handle Request

Default Servlet: Servlet với url-pattern mặc định = /

Là một servlet mặc định, thì servlet này dùng để xử lý những request mà không nằm trong bất kì một url-pattern nào của các Servlet được khai báo trong ứng dụng của bạn



The screenshot shows a browser window with the URL `localhost:8080/DemoServlet/new/tomjerry`. The page content includes a sidebar with links to LEARNING, MY_WEB, JAVA, SOCIAL, and CRITERIA_QUERY. The main content area displays the message: "Served at: /DemoServlet" and "Message from servlet: com.demo.servlet.DefaultServlet". Below this, the word "Default Servlet" is displayed in large, bold, black font.

```
/***
 * .*.Servlet implementation class DefaultServlet
 */
@WebServlet(urlPatterns = {"/*"})
public class DefaultServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private String message;

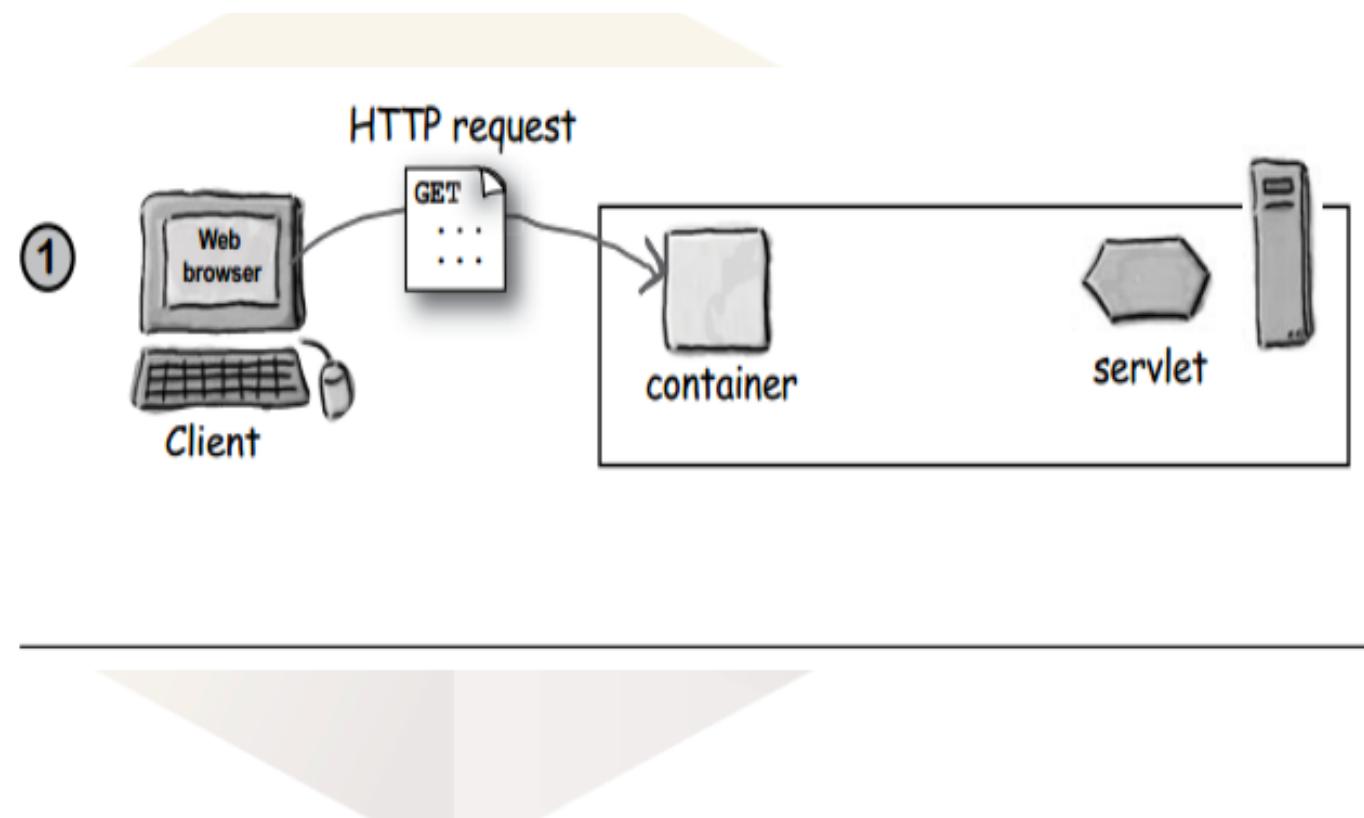
    /**
     * @see HttpServlet#HttpServlet()
     */
    public DefaultServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // TODO Auto-generated method stub

        response.getWriter().append("Served at: ").append(request.getContextPath());
        // Set response content type
        response.setContentType("text/html");
        // Actual logic goes here
    }
}
```

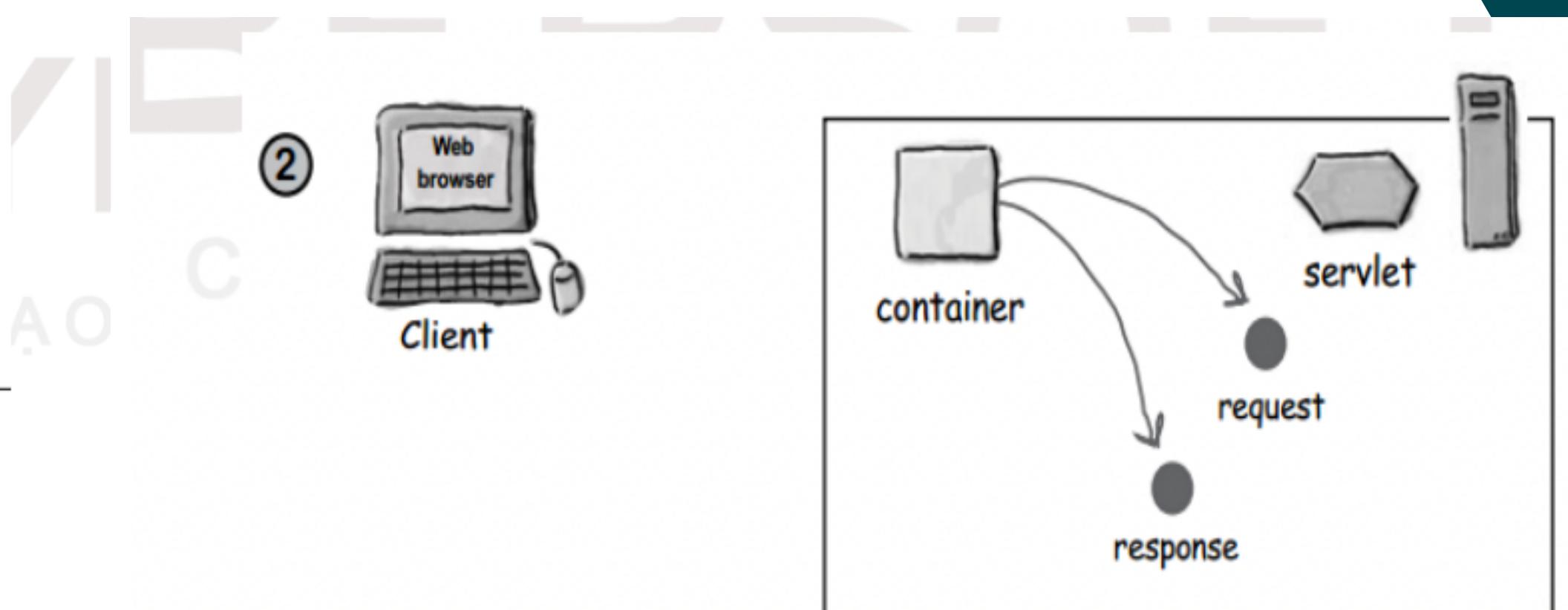
Container Handle Request

B1. Người dùng click vào đường link đến servlet



B2. Container nhận request cho Servlet -> tạo ra 2 đối tượng là

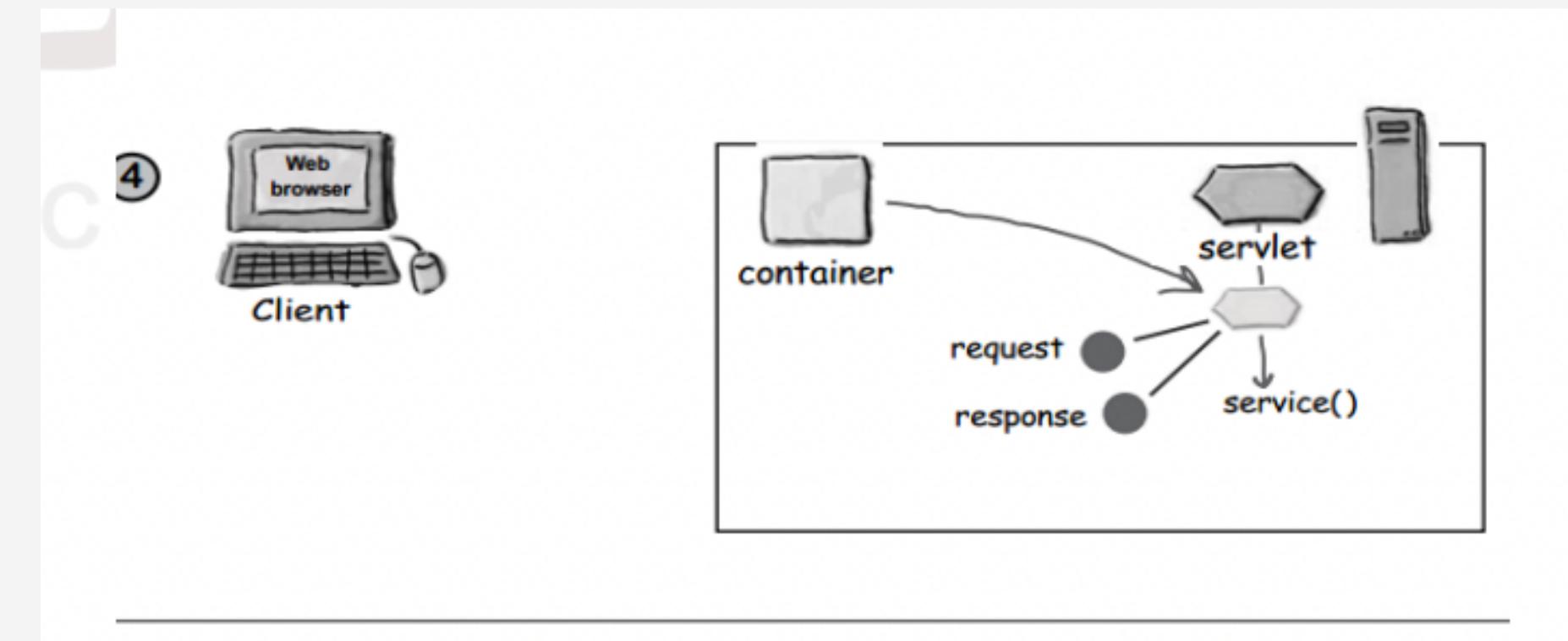
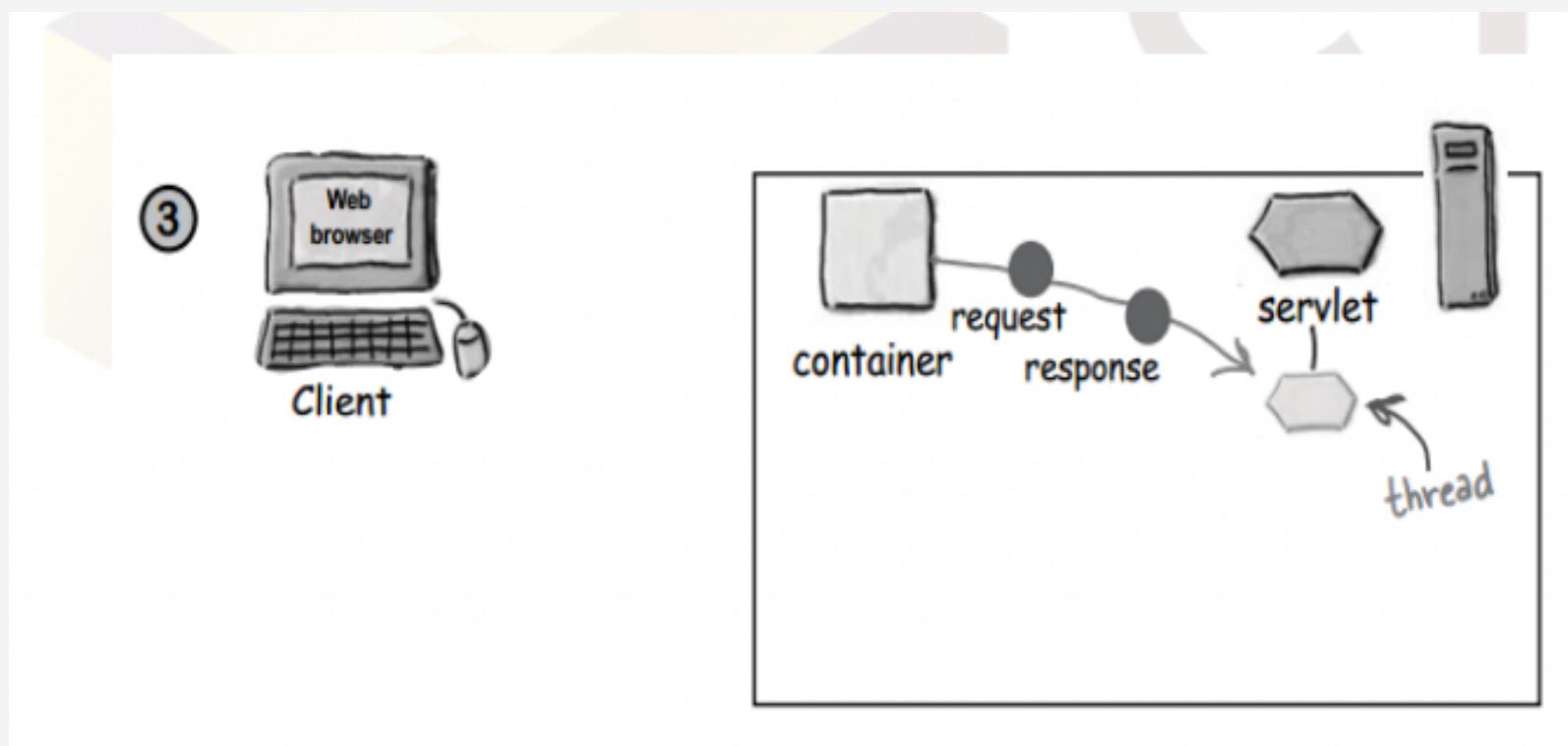
- **ServletRequest**
- **ServletResponse**



Container Handle Request

B3. Container tìm Servlet dựa trên URL của request -> Tạo và cung cấp 1 luồng (thread) cho request đó -> Đưa 2 đối tượng **ServletRequest** và **ServletResponse** vào trong **thread** này

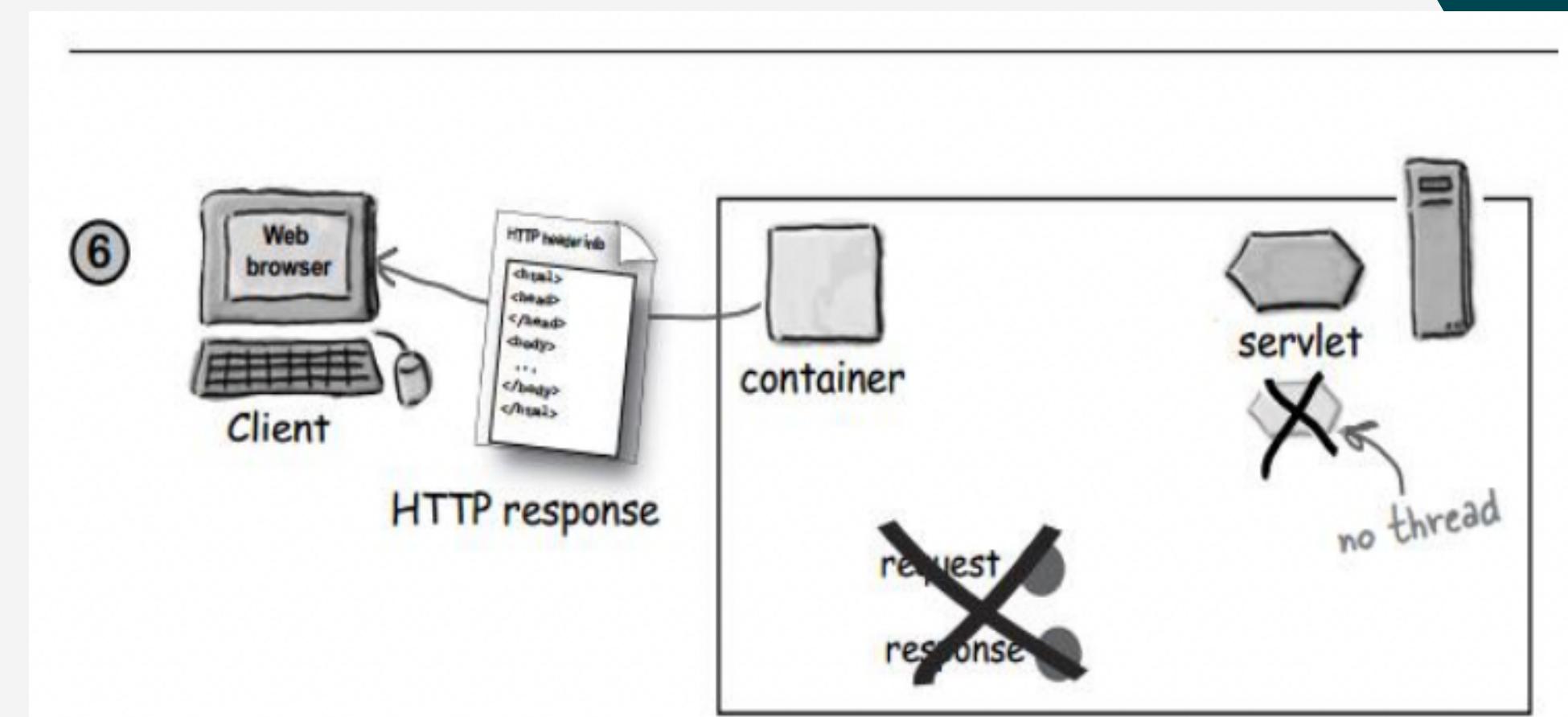
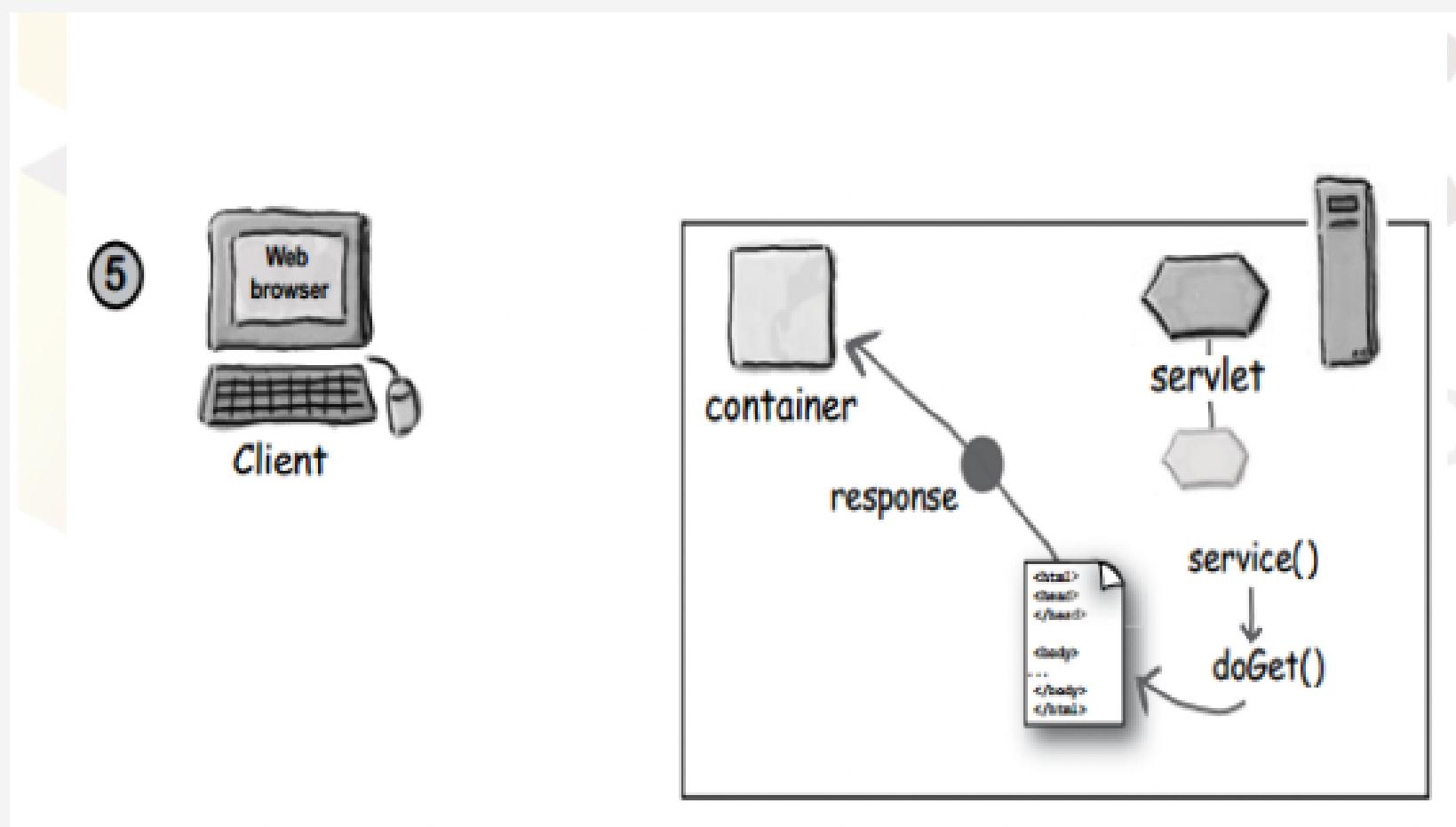
B4. Khi **servlet** được gọi thì nó gọi method **init()** ban đầu, sau đó gọi method **service()** của **servlet** -> method **service()** gọi method **doGet()** hoặc **doPost()** tương ứng vào trong **thread** này



Container Handle Request

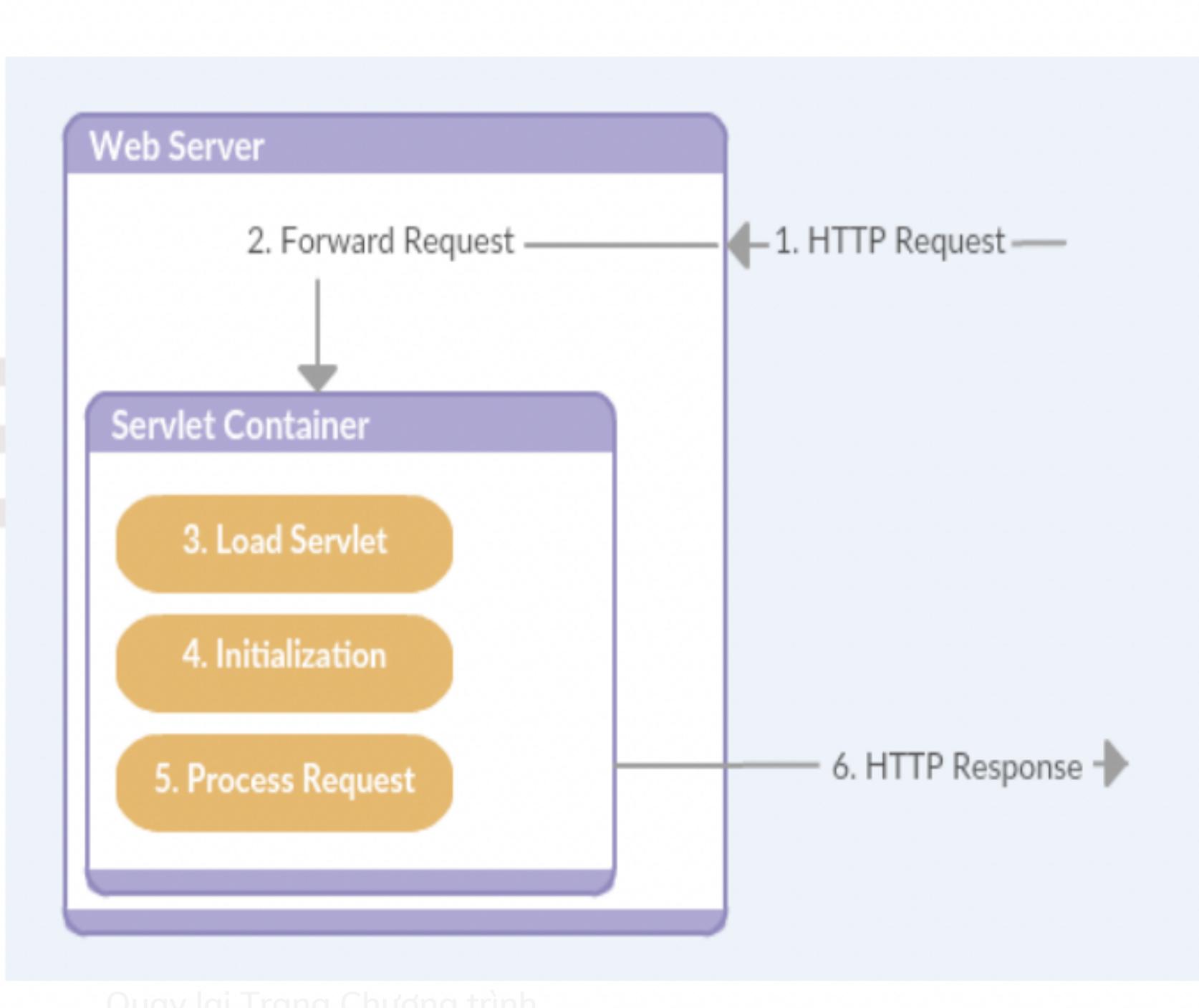
B5. Phương thức **doGet()** tạo ra 1 trang động (**dynamic Page**) và đưa vào đối tượng **ServletResponse**

B6. **Thread** hoàn tất, **Container convert** đối tượng **ServletResponse** thành **HttpServletResponse** 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 tạo ra 1 page nội dung content dynamic
- Tạo và gửi response cho client hoặc forward request cho Servlet khác

LifeCycle Servlet

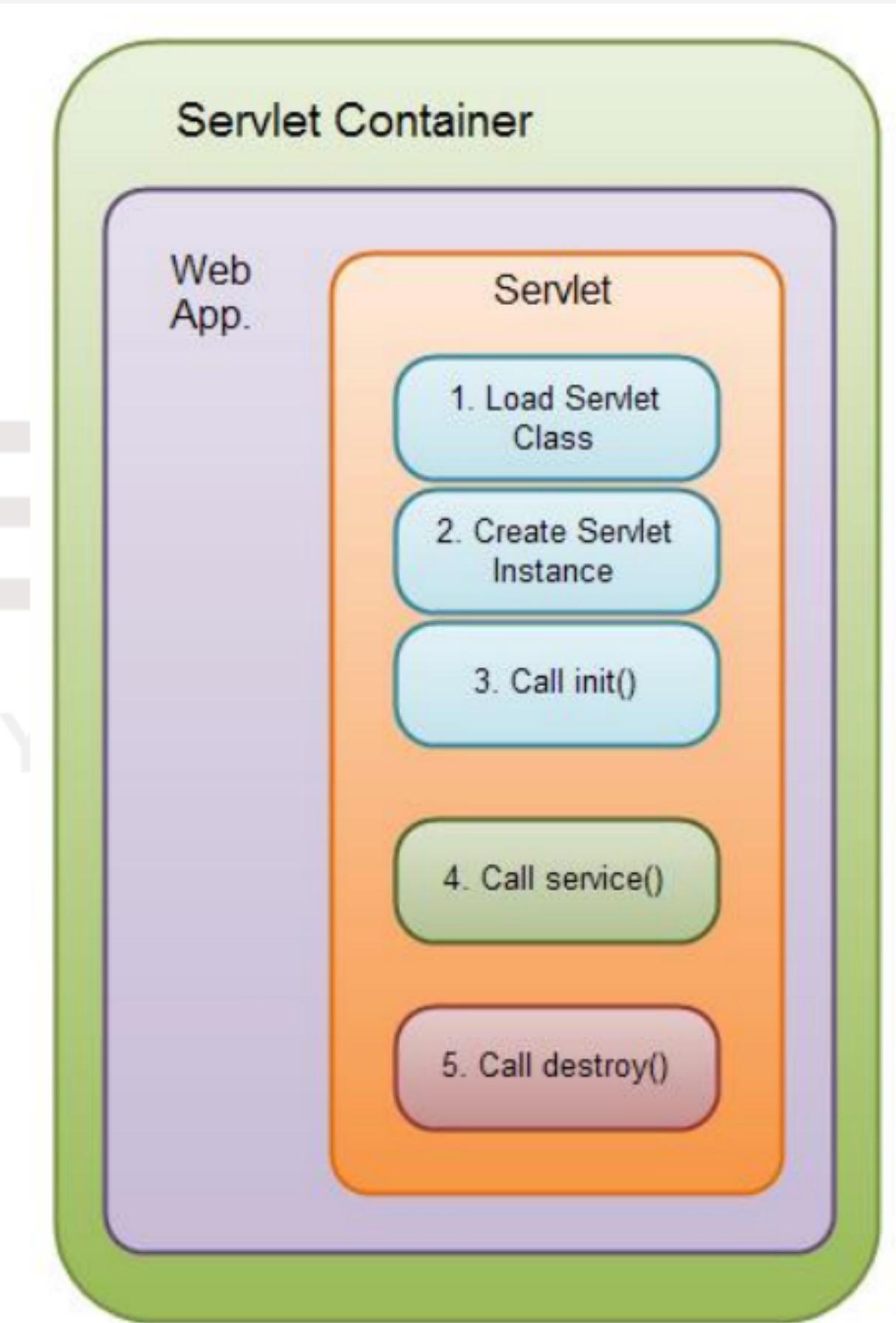
Có 5 bước

1. Tải **Servlet class** vào bộ nhớ
2. Tạo đối tượng **Servlet**
3. Gọi method **init()** của **Servlet**
4. Gọi method **service()** của **Servlet**
5. Gọi method **destroy()** của **servlet**

Bước 1 - 2 - 3 chỉ được thực hiện 1 lần duy nhất khi **Servlet** được nạp lần đầu tiên

Bước 4 được thực hiện nhiều lần khi nhận được request từ người dùng

Bước 5 được thực thi khi bộ chứa **Servlet** (**Servlet Container**) gỡ bỏ tải **Servlet**



Send request tới URL (Redirect hoặc là forward)

sendRedirect

- **Send Redirect:** Send Redirect là method của đối tượng `HttpServletResponse.sendRedirect(String url)`
- Method này được sử dụng để chuyển hướng request tới 1 servlet khác để sử dụng tiếp các chức năng
- Yêu cầu này xem như là tạo 1 request mới
- Không đính kèm data khi redirect qua 1 page mới

```
    " 
»  response.sendRedirect(request.getContextPath() .+ "/HomeServletV2"); ¶
»  ¶
»  ¶
```

Forward

- Method này được khai báo trong RequestDispatcher (forward(ServletRequest request, ServletResponse response))
- Method này được sử dụng để chuyển tiếp đến các tài nguyên khác trong cùng 1 máy chủ, tài nguyên khác ở đây có thể là bất kì file servlet nào, trang jsp nào)
- sử dụng forward thì có thể đính kèm data, sau đó sử dụng request.getAttribute(name) để nhận value về

Include

- Sử dụng include để load resource của current servlet hiện tại vào servlet cần chuyển tiếp

```
>> final String url = "/HomeServletV2";  
>>   
>> request.setAttribute("data", "From HomeServlet");  
>> RequestDispatcher dispatcher = request.getRequestDispatcher(url);  
>>   
>> dispatcher.forward(request, response);
```

Redirect và forward dùng khi nào ?

- Trong một ứng dụng java web, khi chúng ta thực hiện 1 hành động nào đó **mà không muốn gửi dữ liệu đi, hoặc muốn chuyển sang một trang mới mà không có bất kì xử lý dữ liệu gì** thì chúng ta nên dùng `sendRedirect`
- **ví dụ:** Khi nhấn nút thực hiện logic thì khi xử lý xong, chúng ta sẽ chuyển hướng đến 1 trang nào mong muốn
- Trong một ứng dụng java web, khi chúng ta muốn thực hiện 1 hành động nào đó **mà muốn gửi dữ liệu đi, hoặc muốn chuyển sang 1 trang mới mà xử lý dữ liệu** thì chúng ta dùng `forward`.
- **ví dụ:** Khi nhấn nút thực hiện logic thì nếu người dùng đăng nhập sai, chúng ta gửi thông báo cho người dùng, lúc này bạn có thể dùng `request.setAttribute` để gửi dữ liệu đi