WEB ARCHITECTURE

Apache Tomcat

1

Web Architecture

2

1. Client/Server

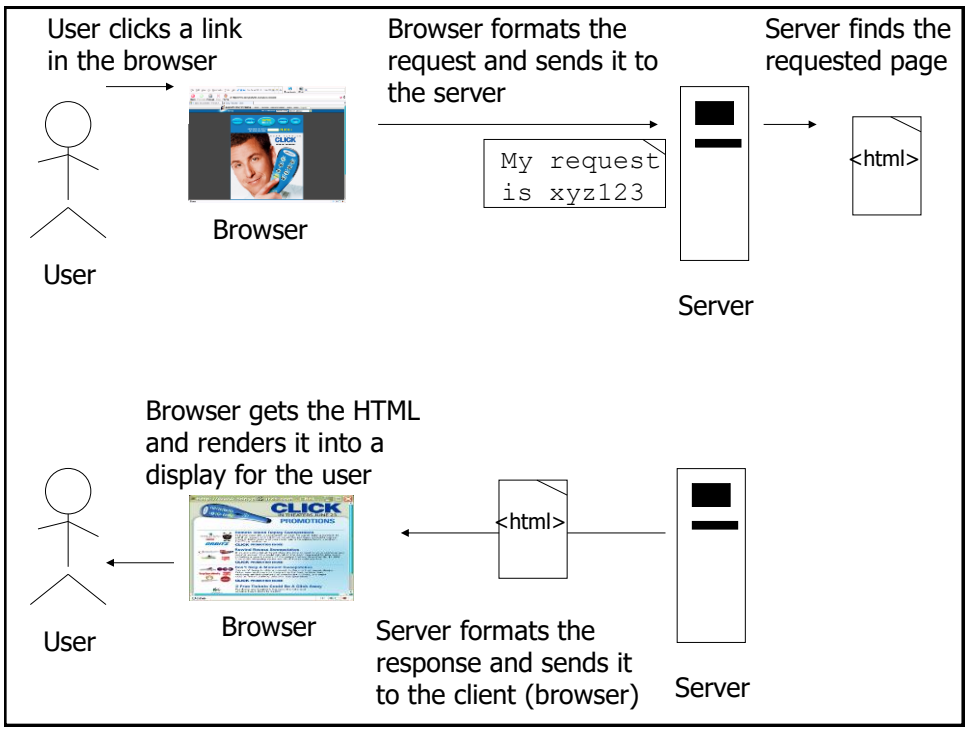
- *What does your web server do?*

A web server processes client requests and responds with the requested data, such as a web page, an API response, or other resources.

- *What does your web client do?*

A web client allows users to send requests to a web server and displays the server's response, such as a webpage or retrieved data.

3



4

1.1. Clients and servers know HTML and HTTP

- *HTML*

When a server answers a request, the server usually sends some type of content to the browser so that the browser can display is. Servers often send the browser a set of instructions written in HTML. All web browsers know what to do with HTML.

- *HTTP*

The client sends an HTTP request, and the server answers with a HTTP response. When a web server sends an HTML page to the client, it sends it using HTTP (HyperText Transfer Protocol).

5

1.1.1. What is the HTTP protocol?

Key elements of the request stream:
-HTTP method (the action to be performed)
-The page to access (a URL)
- Form parameters (like arguments to a method)

Browser

HTTP request

HTTP response

Server

Key elements of the response stream:
- A status code (for whether the request was successful)
- Content-type (text, picture, HTML, etc.)
- The content (the actual HTML, image, etc.)

6

1.1.2. What is in the request?

- The first thing you will find is an **HTTP method** name. These aren't JAVA methods, but the idea is similar.

The method name tells the server the kind of request that's being made, and how the rest of the message will be formatted.

The HTTP protocols has several methods, but the ones you'll use most often are **GET** and **POST**.

7

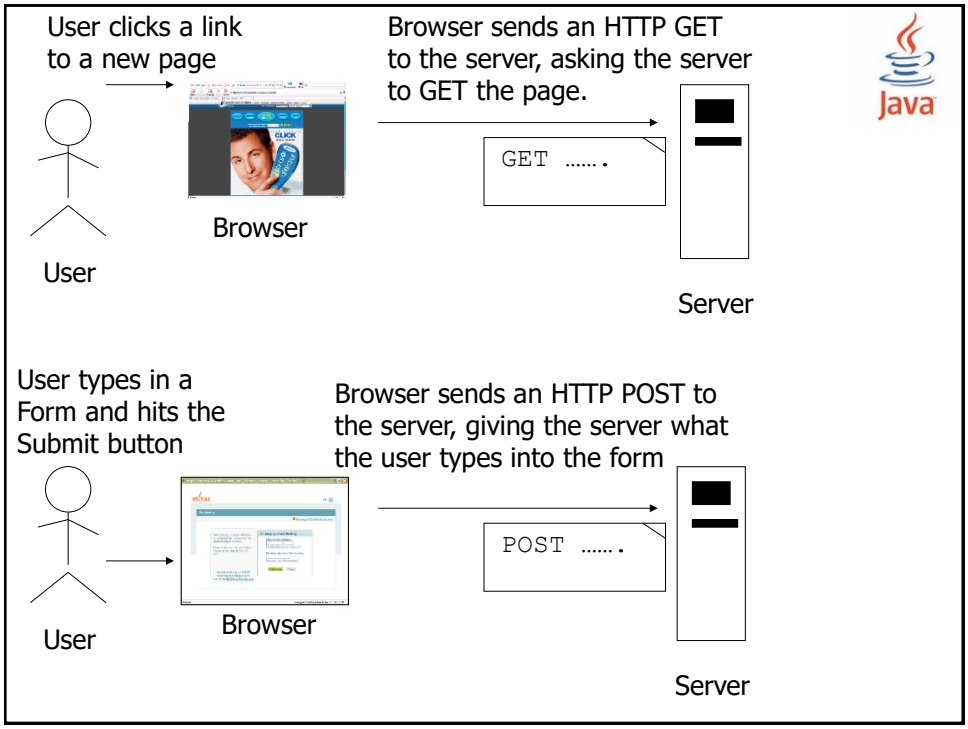
1.1.3. GET and POST

The two most common HTTP request types (= request methods) are **get** and **post**.

- A **get** request gets *(or retrieves)* information from a server. Common uses of **get** requests are to retrieve an HTML document or an image.
- A **post** request *posts* (or *sends)* data to a server. Common uses of **post** requests typically send information, such as authentication information or data from *a form* that obtains user input, to a server.

8

8



9

THE DIFFERENCE BETWEEN GET AND POST

GET requests can be bookmarked, always produce the same result for the same URL, and do not change the server state.

POST requests cannot be bookmarked, as they are used to send data that modifies server state.

GET is meant for retrieving data (e.g., fetching a webpage or API response).

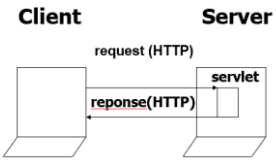
POST is meant for sending data to be processed or stored (e.g., submitting a form, uploading a file).

10

1.1.4. What is a SERVLET?

A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model.

Although servlets can handle various types of requests, they are most commonly used to extend web applications running on a web server. Java Servlet technology provides HTTP-specific servlet classes for such applications.



11

11

1.1.5. Container

- Servlets don't have a main() method. They are under the control of another Java application called a **Container**.

- **Tomcat** is an example of a **Container.** When your web server application (like Apache) gets a request for a servlet, the server hands the request not to the servlet itself, but to the Container in which the servlet is deployed.

12

1.1.5. Container

- **WildFly** is another example of a **Container.**

WildFly, formerly known as JBoss AS, or simply JBoss, is an application server written by JBoss, now developed by Red Hat. WildFly is written in Java and implements the Java Platform, Enterprise Edition (Java EE) specification.It runs on multiple platforms.

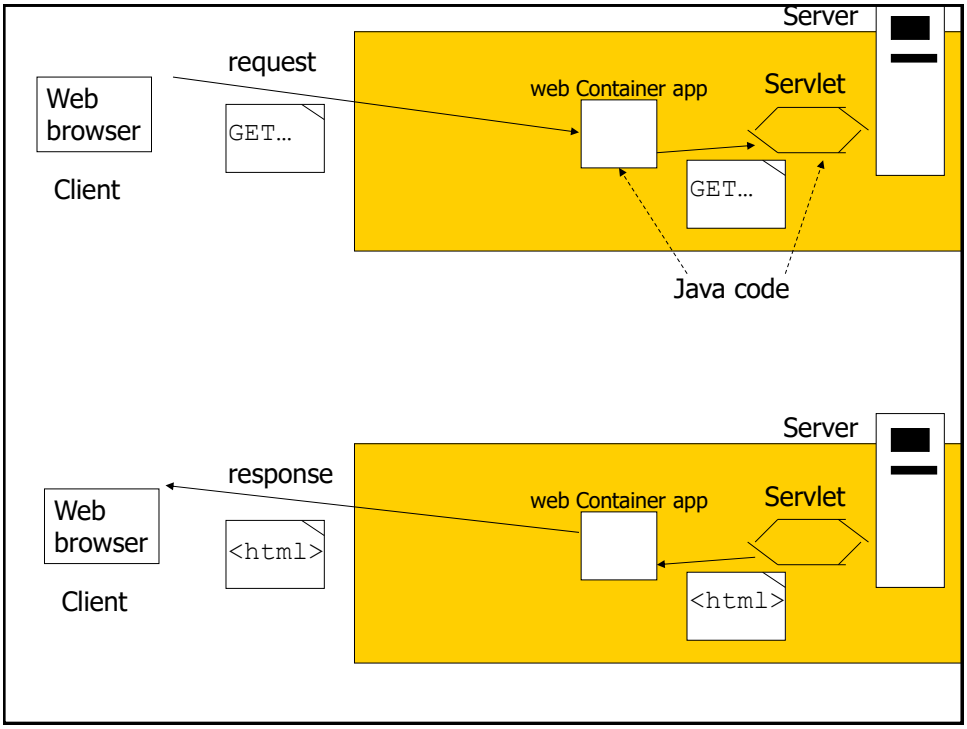
WildFly is free and open-source software.

13

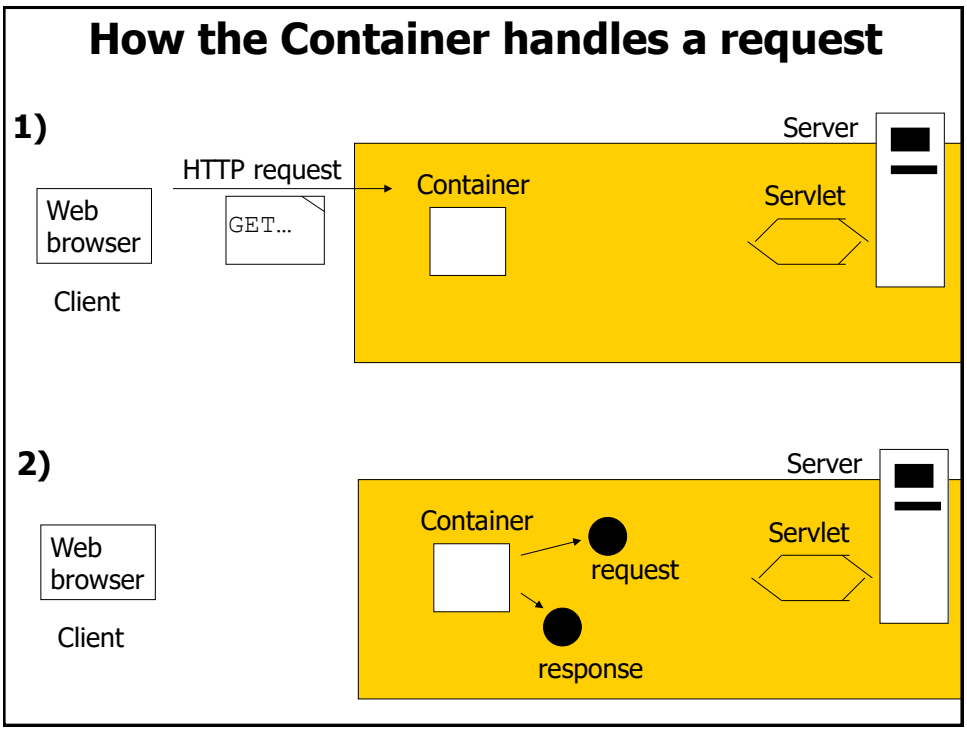
1.1.5. Container

- It is the Container that gives the servlet the HTTP request and response, and it is the Container that calls the servlet's method (like **doPost()** and **doGet()**).

14

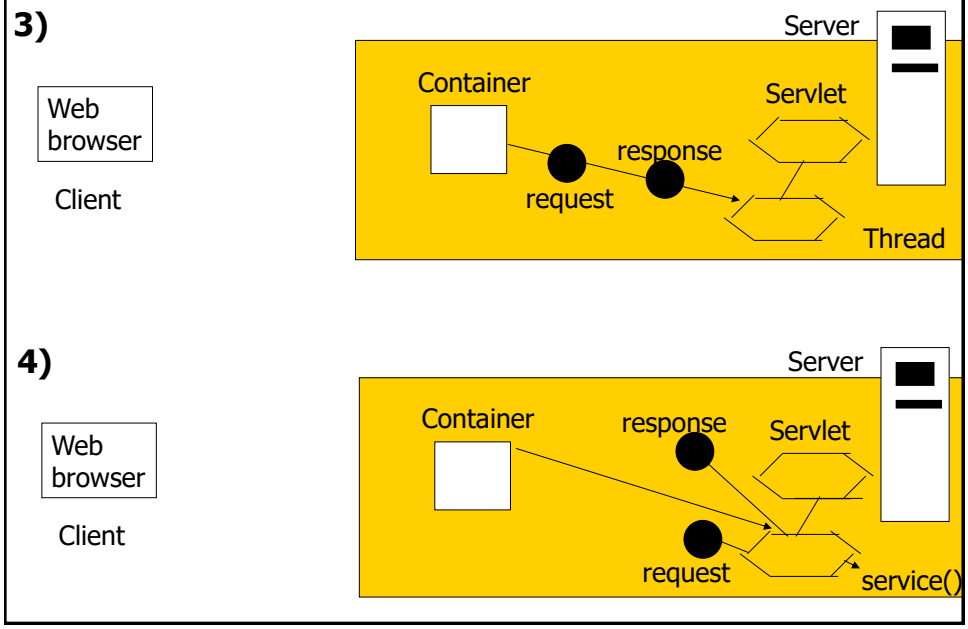

Server
request
Web browser
GET…
Client
web Container app
Servlet
GET…
Java code
Server
response
Web browser
<html>
Client
web Container app
Servlet
<html>

15


How the Container handles a request
1)
Server
HTTP request
Container
Web browser
GET…
Servlet
Client
2)
Server
Container
Web browser
request
Servlet
Client
response

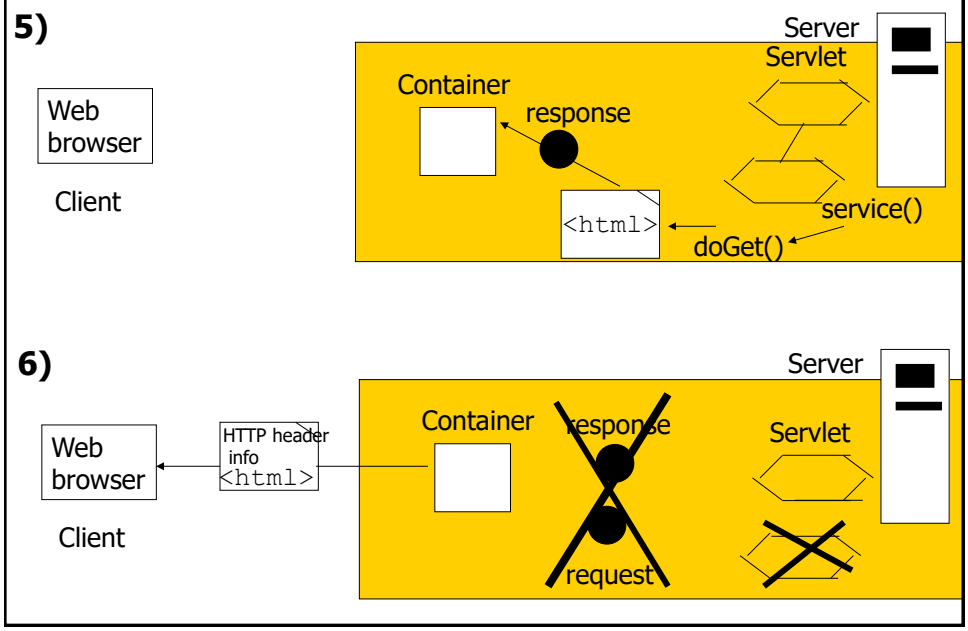
16

How the Container handles a request



17

How the Container handles a request



18

How the Container handles a request

- 1) User click a link that has a URL to a servlet instead of a static page.

- 2) The container "sees" that the request is for a sevlet, so the container creates two objects: HttpServletResponse and HttpServletRequest.

- 3) The container finds the correct servlet based on the URL in the request, creates or allocates a thread for that request, and passes the request and response objects to the servlet thread.

19

How the Container handles a request

- 4) The container calls the servlet's service() method. Depending on the type of request, the service() method calls either the doGet() or doPost() method. For this example, we'll assume the request was an HTTP GET.

- 5) The doGet() method generates the dynamic page and stuffs the page into the response object. Remember, the container still has a reference to the response object!

20

How the Container handles a request

- 6) The thread completes, the container converts the response object into an HTTP response, sends it back to the client, then deletes the request and response objects.

21

1.1.6. Example Servlet


```
package servlet;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
```

Extends **HttpServlet** to handle HTTP **get** requests and HTTP **post** requests.

```
public class WelcomeServlet extends HttpServlet
{
```

22

```
// process "get" requests from clients
@Override
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response )
        throws ServletException, IOException
{  response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   out.println("<h1>Welcome to the Servlet!</h1>");
}

// process "post" requests from clients
@Override
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response )
        throws ServletException, IOException
{  response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   String name = request.getParameter("name"); … }
```


23