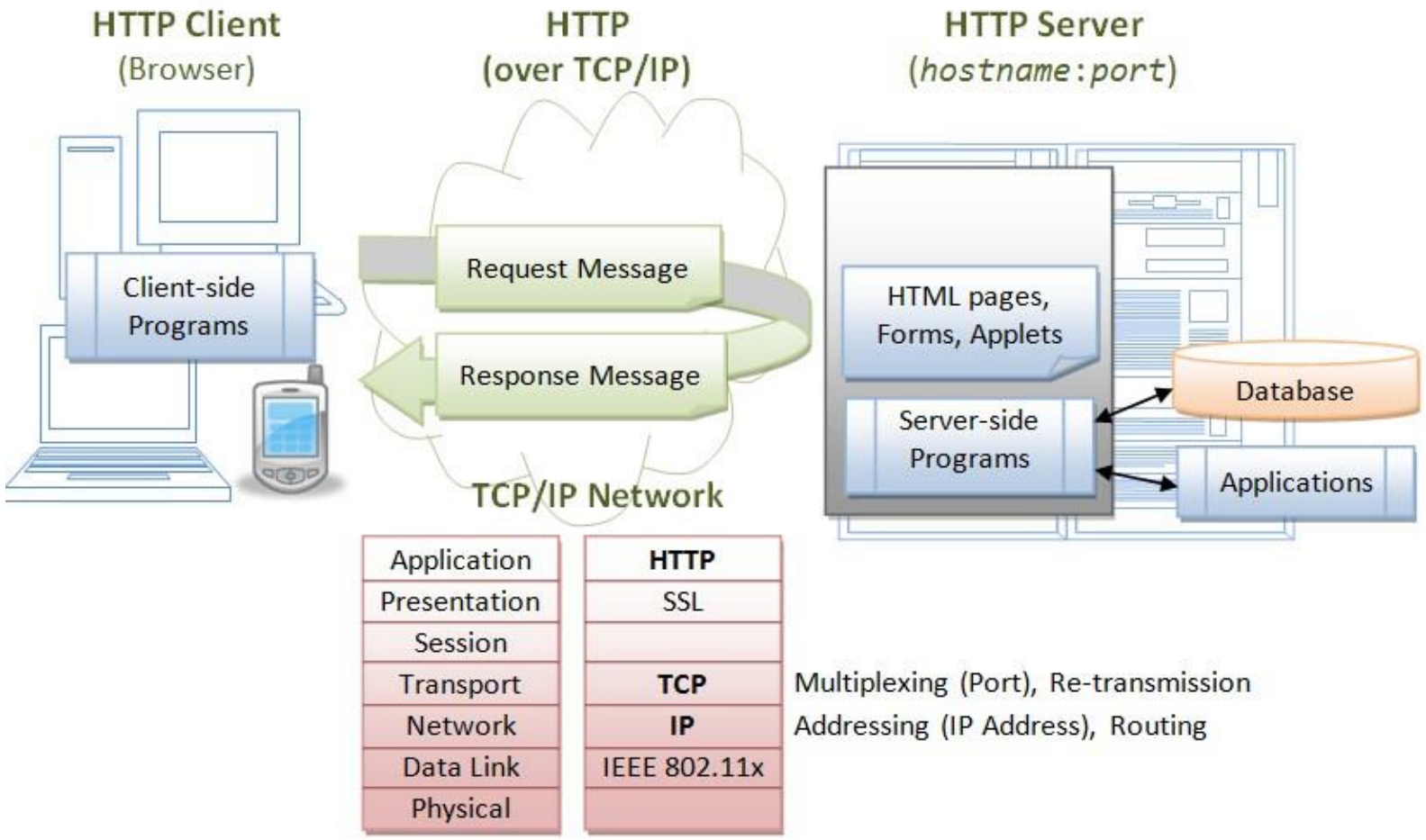
Lecture 10: INTRODUCTION TO SERVLETS AND WEB CONTAINERS

Actions in Accord to All the Laws of Nature

Maharishi University of Management -Fairfield, Iowa © 2019

Servlet Architecture



Overview of Web Dynamics

- Interaction between a browser (client) and a web server:
  - Client requests a resource (file, picture, etc)
  - Server returns the resource, or declares it's unavailable

- Steps:

1. User clicks a link or button in browser
2. Browser formats request and sends to server
3. Server interprets request and attempts to locate resource
4. Server formats a response and sends to browser
5. Browser renders response into a display for user

Web Dynamics (continued)

- Clients and servers communicate using the HTTP protocol
- Browsers know how to transform HTML markup into an HTTP request. They also know how to extract HTML from an HTTP response and render it as a displayable HTML page
- Servers know how to translate an HTTP request into an action of locating a resource. They also know how to produce an HTTP response that contains HTML and gives information about the requested resource.

HTTP (Hypertext Transfer Protocol)

- A request/response protocol that uses TCP/IP to send and receive messages.
- IP routes packets of a message from one host to another; TCP is responsible for arranging these packets into the original message at the destination.
- HTTP protocol is stateless; it does not remember any data that was sent or received in previous conversations.

HTTP Requests

- POST Request – A request for a resource that includes data sent from an HTML form. There is no limit to the amount of data that can be sent in the request.

- GET Request – A request to get a resource specified in the URL. Although some form data can be sent also, it is very limited and very insecure

- Other types of requests that are rarely used in web apps: PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT

Anatomy of an HTTP Request

- An HTTP Request consists of the following:

A request line, for example
GET /images/logo.png HTTP/1.1,

- which requests a resource called /images/logo.png from the server.
- GET may include a query string e.g. `?name=Joe&job=programmer`
- Request headers, such as `Accept-Language: en`
- An empty line.
- An optional message body (not used in GET requests but contains form data in a POST)

HTTP Response

- A response message consists of the following:

- A Status-Line (for example
  HTTP/1.1 200 OK
  which indicates that the client's request succeeded)

- Response headers, such as
  Content-Type: text/html
  May include a "set-cookie" command to maintain a session (more later)
- An empty line
- An optional message body (which often contains the HTML code to be displayed)

Example

- Conversation between an HTTP client and an HTTP server running on
- www.example.com, port 80.
- Request:

  GET /index.html HTTP/1.1
  Host: www.example.com

- Response:

  HTTP/1.1 200 OK
  Date: Mon, 23 May 2005 22:38:34 GMT
  Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
  Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
  ETag: "3f80f-1b6-3e1cb03b"
  Content-Type: text/html; charset=UTF-8
  Content-Length: 131
  Accept-Ranges: bytes
  Connection: close

  <html>
  <body> Hello World, this is a very simple HTML document. </body>
  </html>

What Do Web Servers Serve?

Without making use of some kind of helper application, a web server serves static content – files, images, pdfs, videos, exactly as they are on the server machine.

Responses cannot be customized based on input data passed in by the client or modified at the time they are delivered.

As a consequence:

1. A web server on its own can't handle behavior like log in, online shopping, data lookups, and computations related to input data -- over the web -- require more than a web server.
2. A web server on its own can't save data to the server.

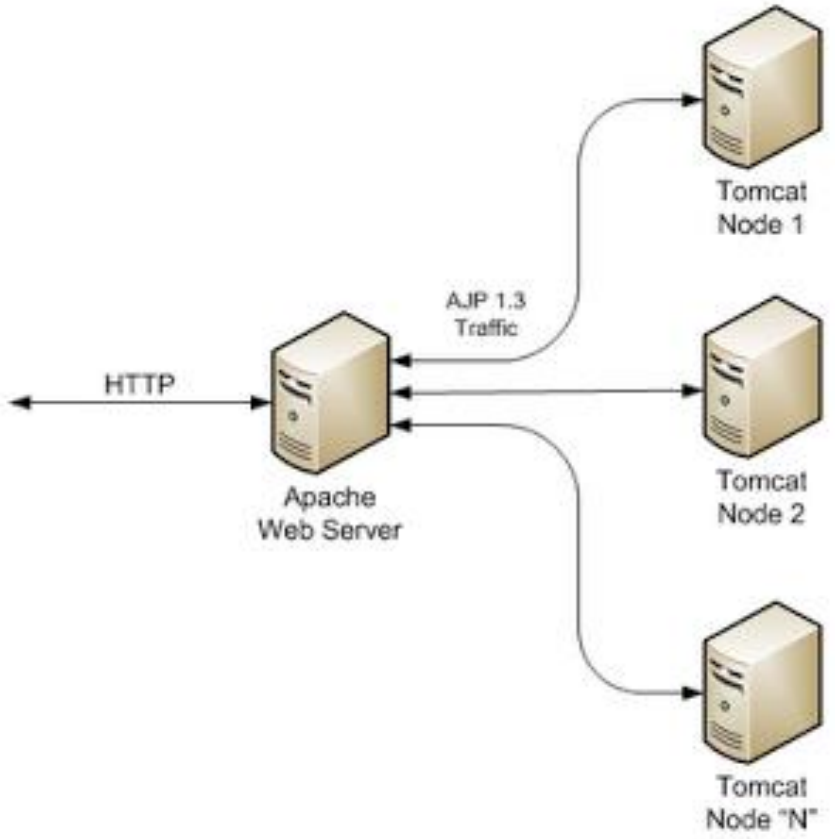
Servlets: Add Dynamic Content

- A servlet is server-side java code that can handle http requests and return dynamic content.

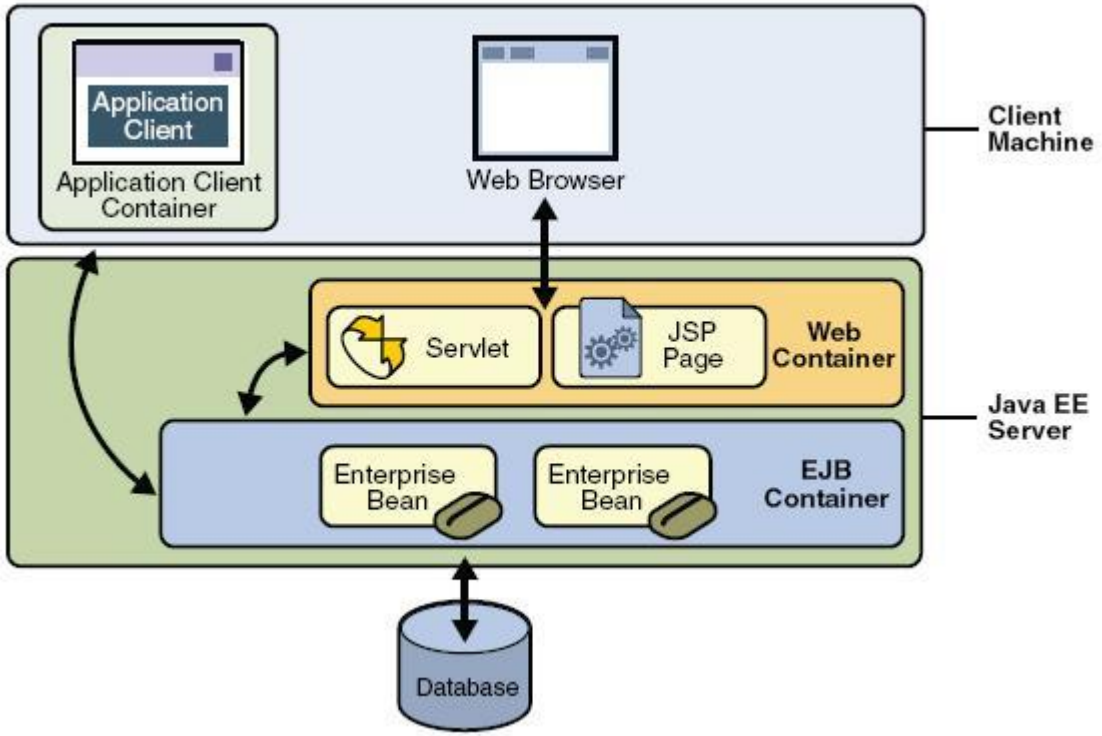
- Servlets are managed by a servlet engine or container. Each request that comes in results in the spawning of a new thread that runs a servlet (eliminating the cost of creating a new process every time).

Web server vs web container

- Most commercial web applications use Apache
  - proven architecture and free license.
- Tomcat can act as simple web server
  - for production environments it may lack features like load-balancing, redundancy, etc.
- Glassfish is like Tomcat, but is also JEE container
  - TomEE ?



Web container and servlet architecture



A servlet is a Java class that extends the capabilities of servers that host applications access by means of a request-response programming model.

SimplestServlet

```
public class SimplestServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        PrintWriter out = response.getWriter();
        out.print("<html><head><title>Test</title></head><body>");
        out.print("<p>Postback received</p>");
        out.print("</body></html>");

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        PrintWriter out = response.getWriter();
        out.print("<html><head><title>Test</title></head><body>");
        out.print("<form method='post'>");
        out.print("<p>Please click the button</p>");
        out.print("<input type='submit' value='Click me'/>");
        out.print("</form>");
        out.print("</body></html>");

    }
}
```

Specifying a Servlet

Servlets are given names that are used by different parts of a web app:

1. Each servlet is a Java class and so has a fully qualified class name. Example: mum.Hello
2. Another name for the servlet is used in an HTML form to tell the Web Server that a servlet is being requested and to tell the Container ***which*** servlet is needed.
    1. Example: hello. This is the name used by client in the form (action = "hello") to refer to servlet.
3. A third name is the ***internal name*** that the container uses to keep track of the servlets it is managing. This is the value specified as the servlet-name in the web.xml configuration file.

Three Names of a Servlet

```
<servlet>
   <servlet-name>hello</servlet-name>
   <servlet-class>mum.Hello</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>hello</servlet-name>
   <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

- servlet-name is the internal name of the servlet
- serlvet-class is the Java name of the class
- url-pattern is the way the servlet is specified in the HTML form
- <form action="*hello*" *method*="*get*">

web.xml

```
<web-app …>
    <servlet>
        <servlet-name>SimplestServlet</servlet-name>
        <servlet-class>mum.cs545.SimplestServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SimplestServlet</servlet-name>
        <url-pattern>/SimplestServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        …
    </session-config>
</web-app>
```

Mapping Requests

- The THREE types of <url-pattern> elements
- **1) EXACT match**
  - *Example:*
    <url-pattern>/Beer/SelectBeer</url-pattern>
  - MUST begin with a slash (/).
- **2) DIRECTORY match**
  - *Example:*
    <url-pattern>/Beer/*</url-pattern>
  - MUST begin with a slash (/).
  - Always ends with a slash/asterisk (/*).
- **3) EXTENSION match**
  - *Example:*
    <url-pattern>*.jsp</url-pattern>
  - MUST begin with an asterisk (*) (NEVER with a slash).
  - After the asterisk, it MUST have a dot extension (.do, .jsp, etc.)

- https://docs.oracle.com/cloud/latest/as111170/WBAPP/configureservlet.htm#WBAPP135. (examples)

XML vs Annotations

```
<servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>edu.mum.cs.SimplestServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

```
@WebServlet(name = "ServletDemo", urlPatterns = {"/", "/index", "welcome"})
public class SimplestServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    }
}
```

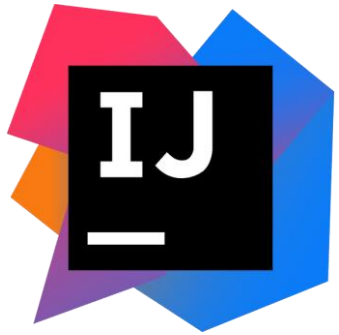
XML vs Annotations

- **XML**
  - Centralized file to change the project
  - You need to restart after deploy

- **Annotations**
  - Annotations are in the same file which makes it easy to track and understand the purpose of the file
  - Change should be done in all files
  - No need to restart after change, the container will compare the servlet file version with the one in memory and update it accordingly.

Setting up our Development Environment

- Java JDK
- IntelliJ IDE
- Tomcat (Container)



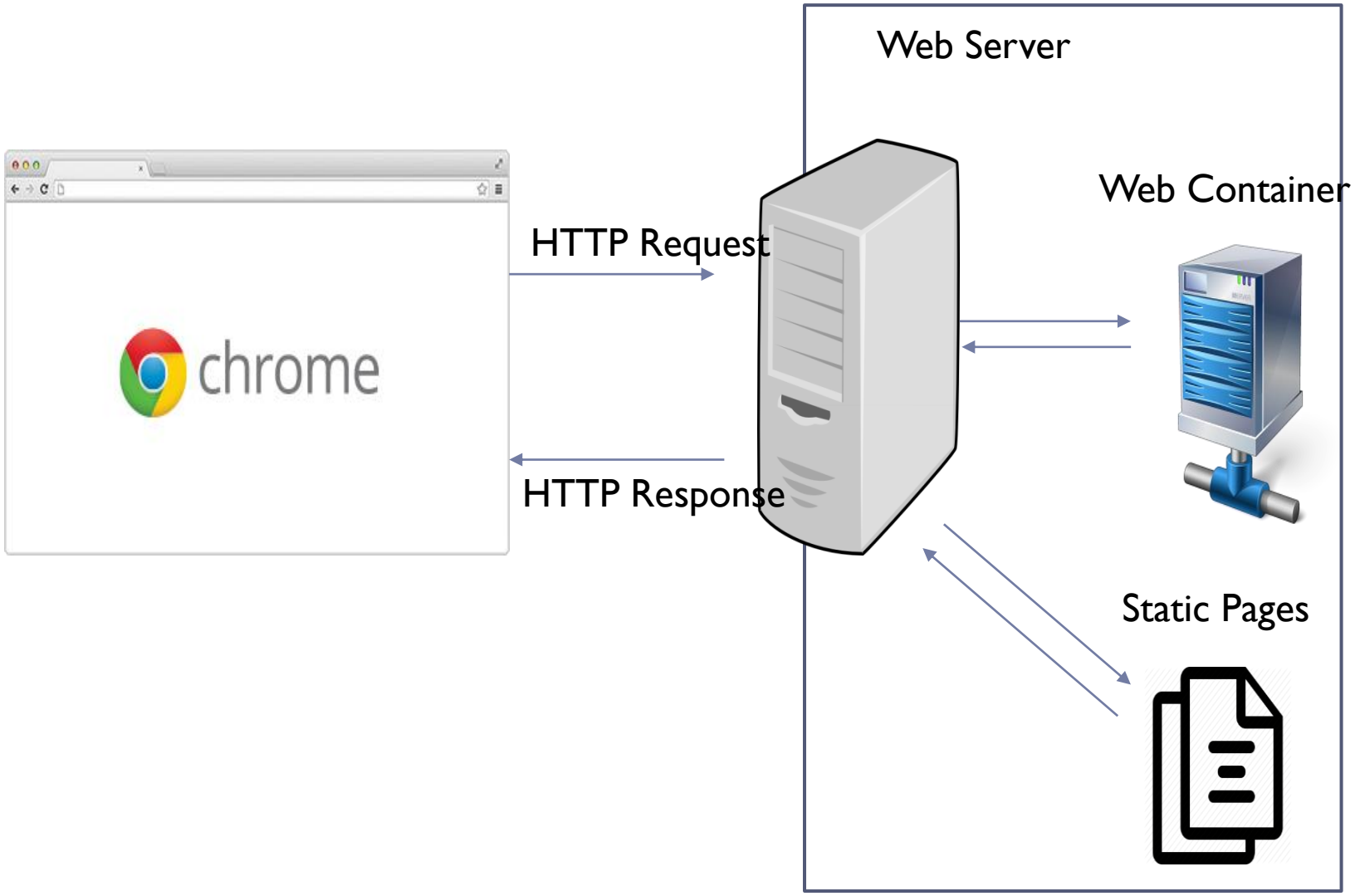
Main Point 1

Servlets are the basis of dynamic web applications. Servlets process information from a request object and generate new information in a response object.

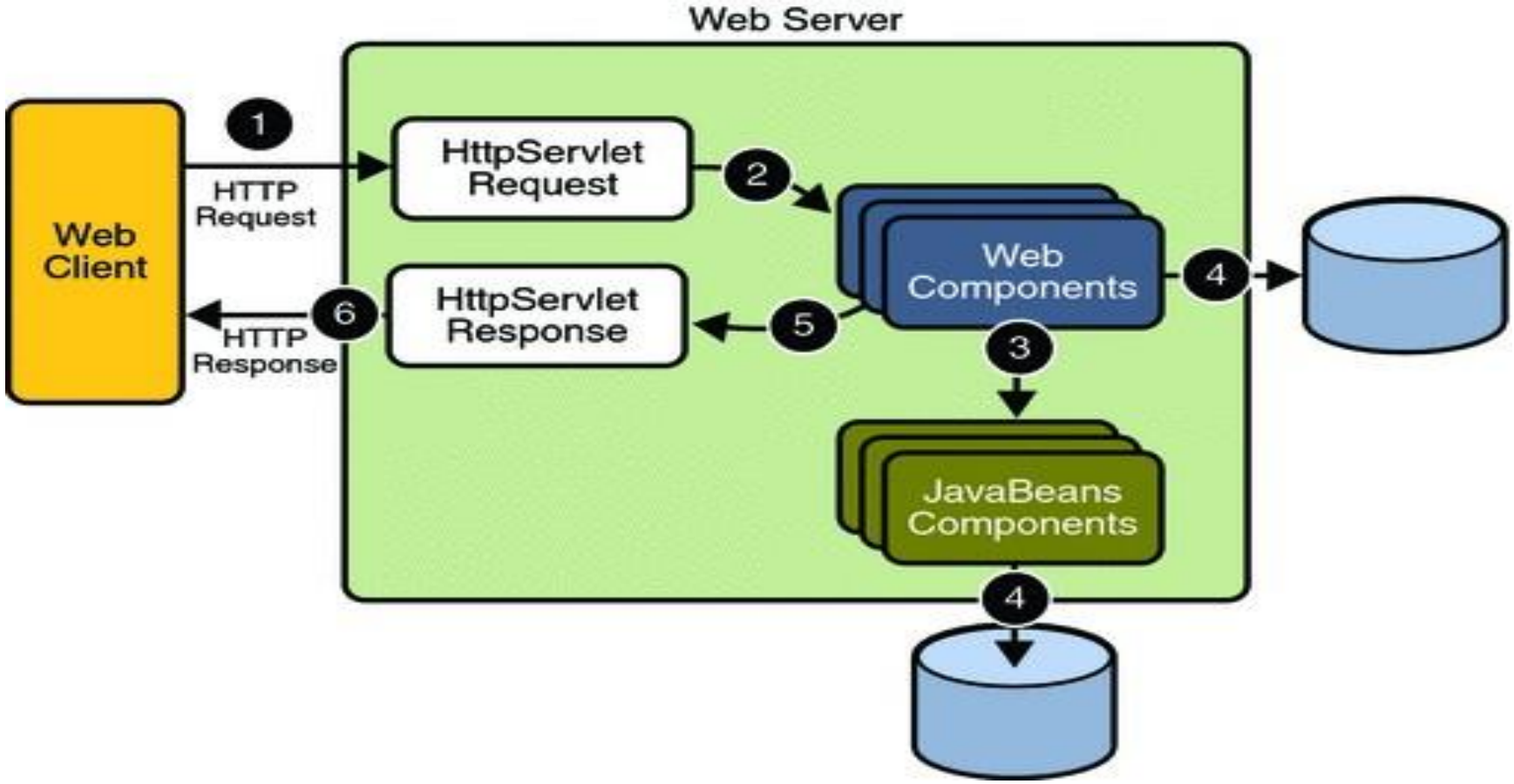
Science of Consciousness:

Analogously, humans also operate by giving responses to requests. Response are more likely to be correct and appropriate to the extent that one has broad awareness to consider all relevant information and fine focus to understand the request.

Dynamic versus static pages



Web server with Servlet container



Web Components are servlets and JSPs in the servlet container JavaBeans Components represent enterprise JavaBeans, EJB

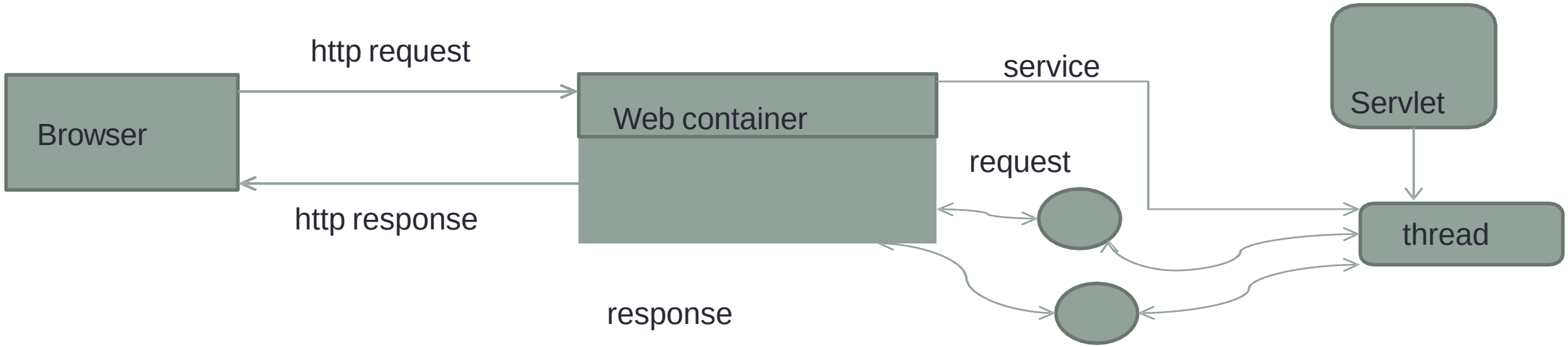
The Container

- Servers that support servlets have as a helper app a servlet container.
- When a request comes to the web server, if the server sees the request is for a servlet, it passes the request data to the servlet container.
- The servlet container locates the servlet, creates request and response objects and passes them to the servlet, and returns to the web server the response stream that the servlet produces.
- The web server sends the response back to the client browser to be rendered.

Containers provide fundamental support

- network communications
  - communicating with web server
- lifecycle management
  - no "main" method in a servlet, …
- concurrency
- state management
  - session and context attributes
- security
- Support for JSP (JSF, JPA, JTA, EJB, …)

How container handles an HTTP request



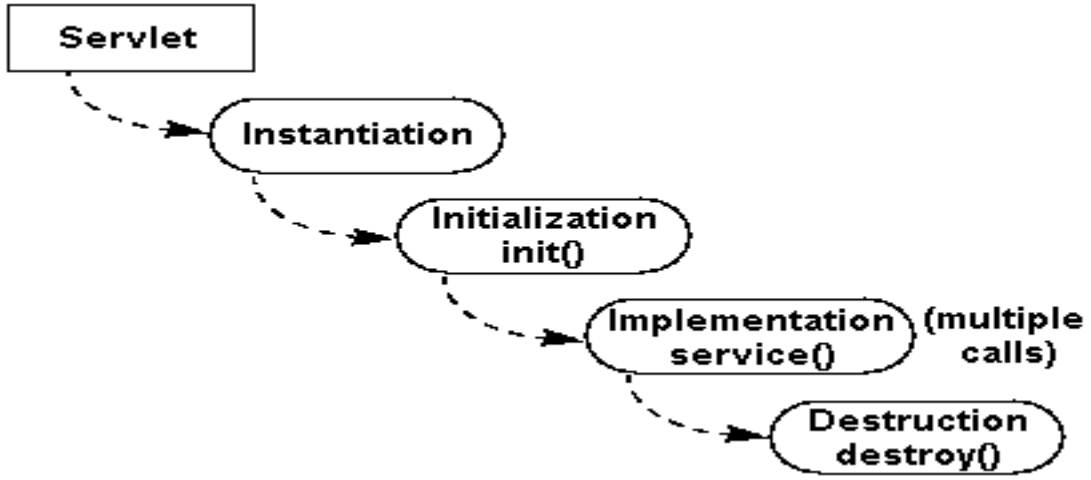
- Container receives new request for a servlet
- Creates HttpServletRequest and HttpServletResponse objects
- Calls service method on HttpServlet object in thread
- When thread completes, converts response object into HTTP response message

Main Point 2

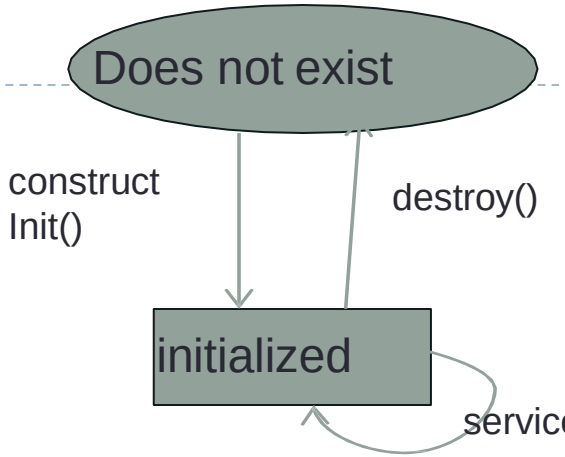
- Web containers provide essential support services for servlets.

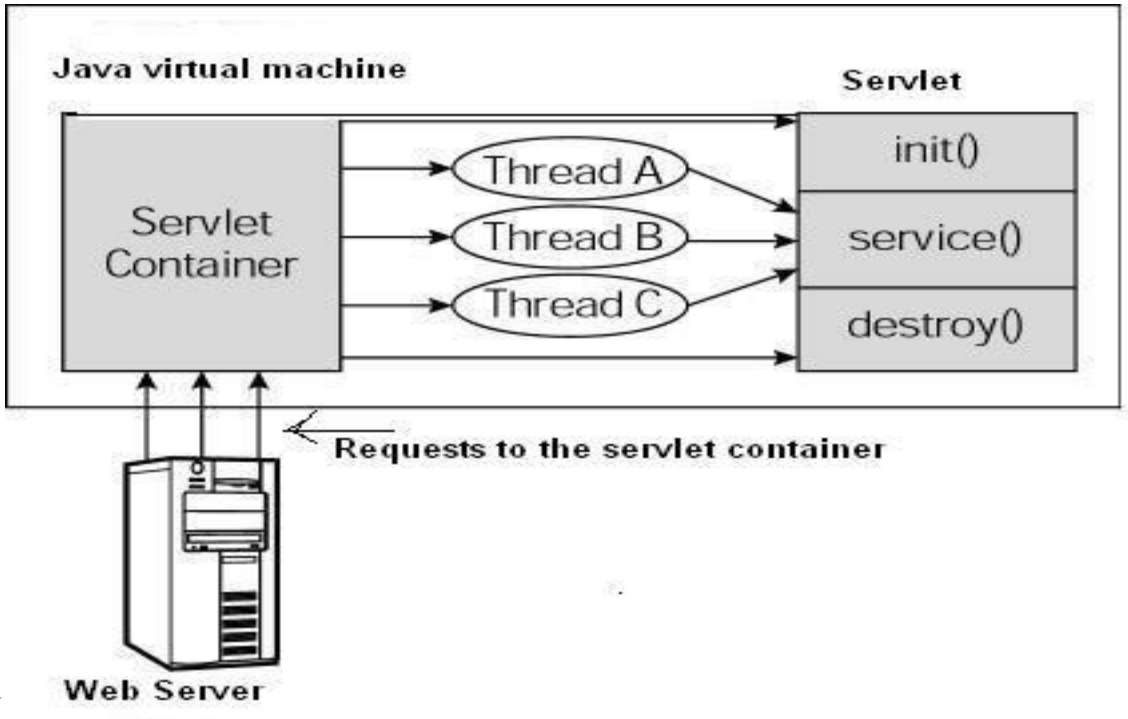
- Science of Consciousness: Our experience of pure consciousness provides essential support services that enable us to think clearly and act effectively.

Servlet life cycle



. Load
. Create
. Init
. Service
. Destroy





- 1 instance of servlet
- new thread created for every request
  - Then service() called on the thread
- All threads share instance variables
- Each thread has own stack for local variables

The service() method

- The servlet container (Tomcat) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.
- Each time the server receives a request for a servlet, the server starts a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.
- Threads share the same instance (and instance variables) of the servlet class. Every method call (e.g., service) in a thread will have its own space on the call stack and own local variables.

HOW A WEB CONTAINER HANDLES HTTP REQUESTS

```
public void handleHTTPRequest(Socket s) throws Exception {
    // these are used to encapsulate the HTTP request and response
    Hashtable<String, String> requestHeaders = new Hashtable<String, String>();
    Hashtable<String, String> formParameters = new Hashtable<String, String>();
    Hashtable<String, String> responseHeaders = new Hashtable<String, String>();
    StringBuilder responseContent = new StringBuilder();
    Pattern p = null;
    Matcher m = null;
    byte[] b = null;
    FileInputStream fis = null;
    Mylet mylet = null; // will hold reference to Mylet named
    String method = null;
    String myletName = null;
```

HOW A WEB CONTAINER HANDLES HTTP REQUESTS

```
try {
    InputStream is = new BufferedInputStream(s.getInputStream());
    s.setSoTimeout(2000);      // block on read for this many milliseconds

    String requestLine = readLine(is);
    p = Pattern.compile("^(\\w+)\\s+/([^\\s]+)");
    m = p.matcher(requestLine);
    if (m.find()) {
        method = m.group(1).toLowerCase();
        myletName = m.group(2);
        mylet = getMylet(myletName);
        System.out.println("mylet: " + mylet);
    } else
        throw new Exception(String.format("Something wrong with %s\r\n",
requestLine));
```

HOW A WEB CONTAINER HANDLES HTTP REQUESTS

```
StringBuilder sb = new StringBuilder();
String line = null;
int contentLength = -1;
// read headers and save them in a Hashtable
while ((line = readLine(is)).length() > 0) {
    if (line.toLowerCase().startsWith("content-length")) {
        // Remember length of the content.
        p = Pattern.compile("\\d+$");
        m = p.matcher(line);
        if (m.find()) {
            contentLength = Integer.parseInt(m.group());
        }
    }
    String header[] = line.split(":");
    requestHeaders.put(header[0].trim(), header[1].trim());      // save header in a hash table
}

// Save parameters in a hash table—will be in body if it is a post (get params ignored?)
if (contentLength != -1) {
    b = new byte[contentLength];
    is.read(b);
    String params = new String(b, "UTF-8");
    String fields[] = params.split("&"); // params = name1=value1;name2=value2;name3=value3&...
    for (String field : fields) {
        String nv[] = field.split("=");// field is name=value
        formParameters.put(nv[0], nv[1]);// name-->value
    }
}
```

HOW A WEB CONTAINER HANDLES HTTP REQUESTS

```
// invoke service method of mylet, passing in encapsulated HTTP request & response
        // Should run this in its own thread!
        mylet.service(method, requestHeaders, formParameters, responseHeaders, responseContent);
        StringBuilder httpResponse = new StringBuilder();
        // httpResponse.append(status line): should be able to set status
        // httpResponse.append(headers in responseHeaders): should set headers
        OutputStream os = s.getOutputStream();
        os.write(responseContent.toString().getBytes("UTF-8"));
    } catch (Exception ex) {
        System.out.format("%s: %s", ex.getClass().getName(), ex.getMessage());
    } finally {
        s.close();
        fis.close();
    }
}

// Use reflection to instantiate named Mylet or reuse an already existing one
private Mylet getMylet(String name) throws Exception {
    Mylet mylet = mylets.get(name);
    if (mylet == null)  // if first time anybody has asked for this servlet since last
    {                   // time that the container was started.
        Class c = Class.forName("container." + name);
        mylet = (Mylet) c.newInstance();  //need reflection because containers won't know servlet names in advance
        mylets.put(name, mylet);  // save reference in hashtable so it can be reused
        mylet.init();
    }
    return mylet;
}
```

Servlet life cycle

1. Load servlet class (Class c = Class.forname(...))
2. Instantiate servlet (c. getDeclaredConstructor().newInstance())
3. init() called only once in the servlet's life
   - Must complete before Container can call service()
4. service() (called for each request, each request runs in a separate thread)
5. destroy() (called only once)

Main Point 3

Web containers manage the life cycle of servlets.

Science of Consciousness:

The unified field manages the entire universe, and by experiencing this field of awareness on a regular basis we spontaneously act more and more in accord with all the laws of nature.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Web Containers: Actions in Accord with All the Laws of Nature

1. Developers override the doGet or doPost methods of servlets to implement the request-response functionality of the web application.

2. The web container is responsible for calling the service methods as well as managing the lifecycle of the servlet and exchanging all information over the network.

---

3. Transcendental consciousness is the experience of the home of all the laws of nature. Having this experience structures one's awareness to be in accord with all the laws of nature.

4. Impulses within the Transcendental Field: Servlets represent specific impulses of intelligence that are supported by the general-purpose services of the web container. In a similar manner, thoughts connected with the transcendental field are supported by all the laws of nature.

5. Wholeness moving within itself: In unity consciousness, thoughts and actions arise from this level of thought, and daily life is lived in terms of this experience of wholeness and integration. This is like the effects of integration and correctness that are produced in web applications due to the underlying wholeness and integration of the web container.