Introduction to Servlets and Web Containers

Actions in Accord with All the Laws of Nature

Main Point 1 Preview

Every platform for web applications has a mechanism to dynamically generate web pages containing information from the server. On the Java platform Servlets are the Java classes that provide this dynamism.

Science of Consciousness:

Pure consciousness is a field of infinite dynamism. We experience this as restful alertness during the practice of the TM Technique that give a basis for dynamic activity.

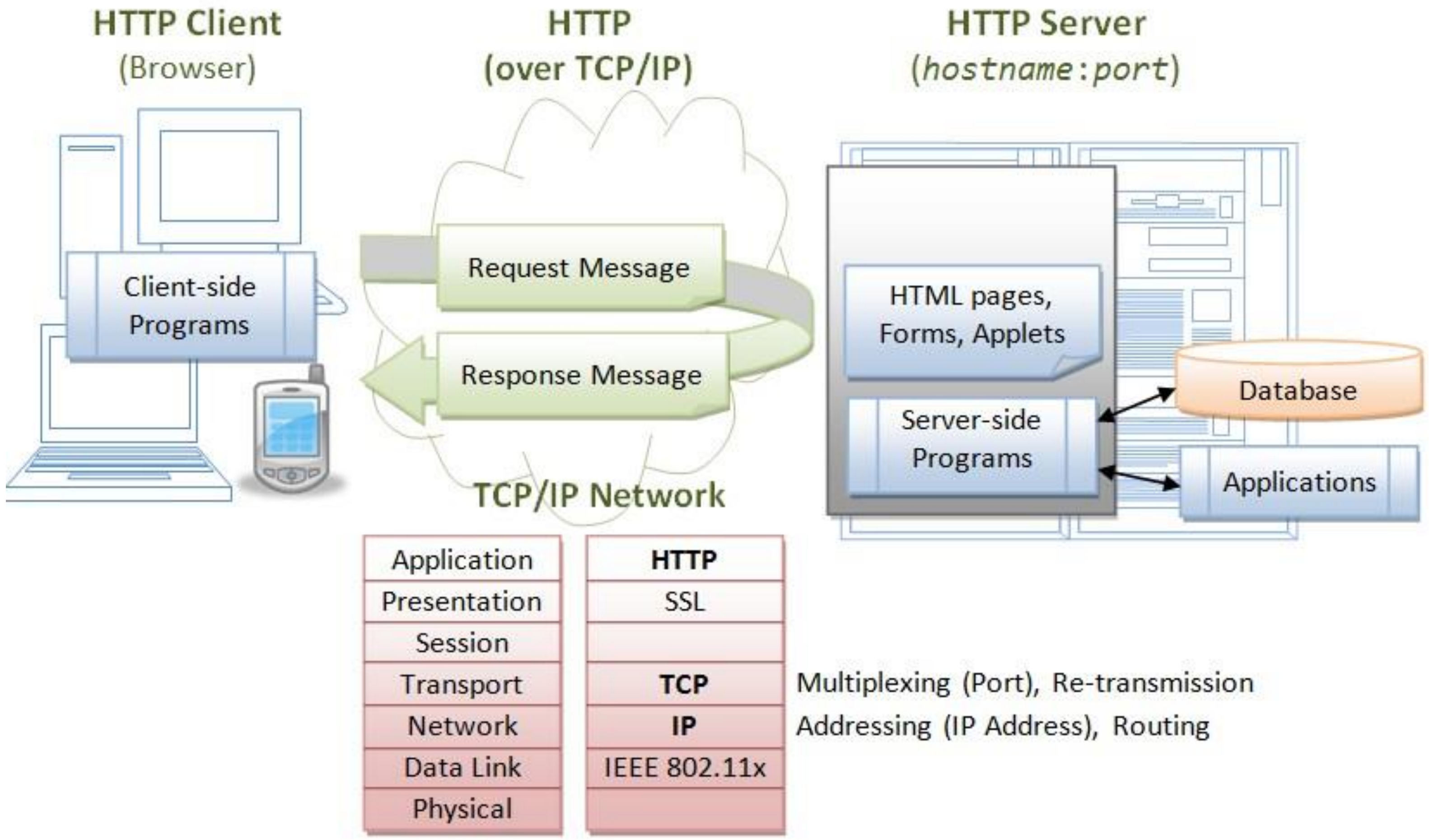
Overview of Web Dynamics

- 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

Servlet Architecture



Application	**HTTP**	
Presentation	SSL	
Session		
Transport	**TCP**	Multiplexing (Port), Re-transmission
Network	**IP**	Addressing (IP Address), Routing
Data Link	IEEE 802.11x	
Physical		

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

- GET Request – A request to get a resource specified in the URL. The purpose of a get request is to retrieve new information from the server. It should not change state in the server.
- POST Request – A request that sends data from an HTML form. There is no limit to the amount of data that can be sent in the request. The intention of a post request is to save or update something on the server with this data.
- 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,
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

  HTTP/1.1 200 OK

(200 indicates that the client's request succeeded)

- Response headers, such as

  Content-Type: text/html

  May include a “set-cookie” header 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.

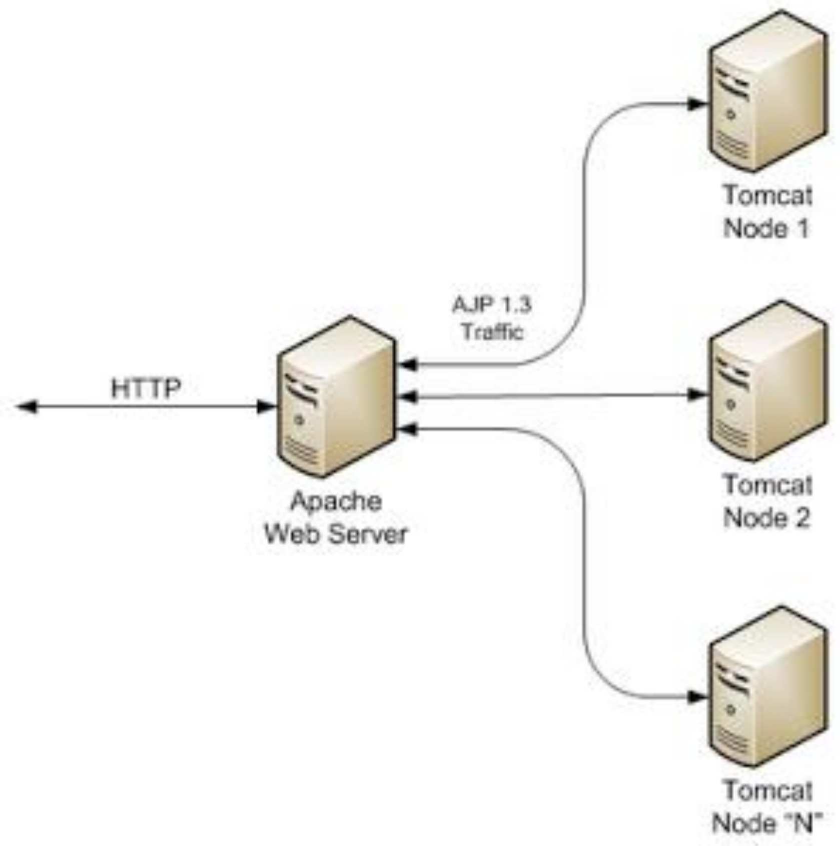
Servlets: Add Dynamic Content

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

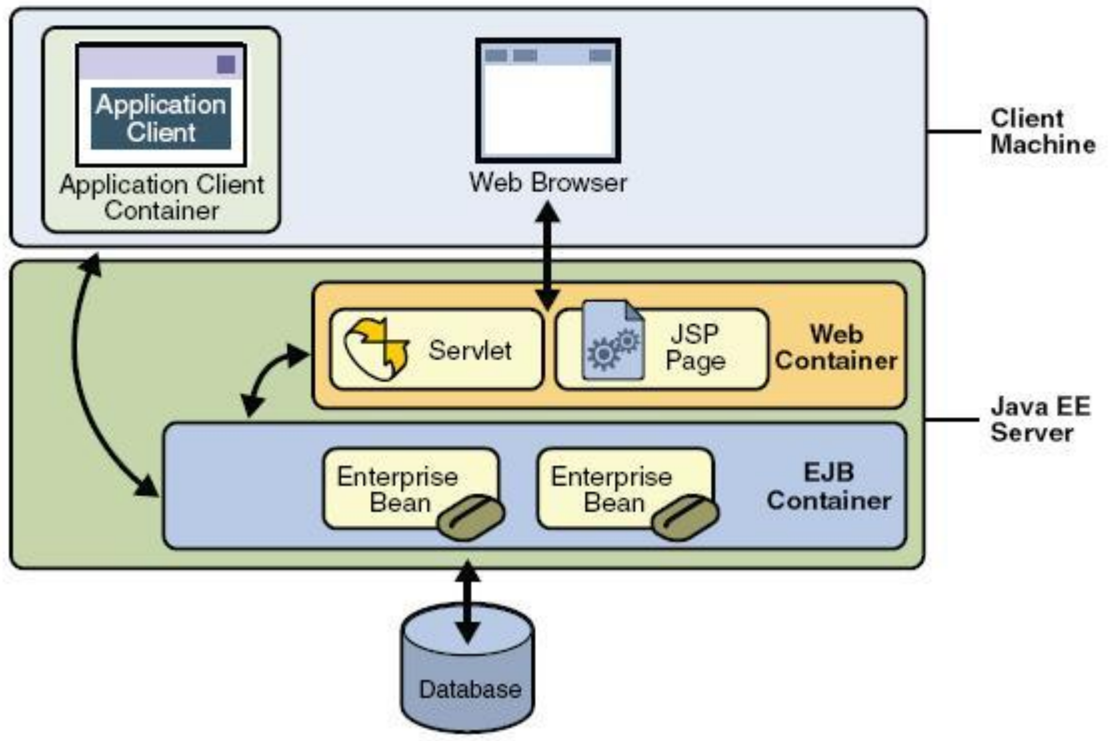
- Servlets are managed by a servlet engine or container.
    - Each request that comes in results in the spawning of a new thread that runs a servlet (eliminating the cost of creating a new process every time).

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, WildFly are like Tomcat, but are also JEE containers
  - TomEE ?



Web container and servlet architecture



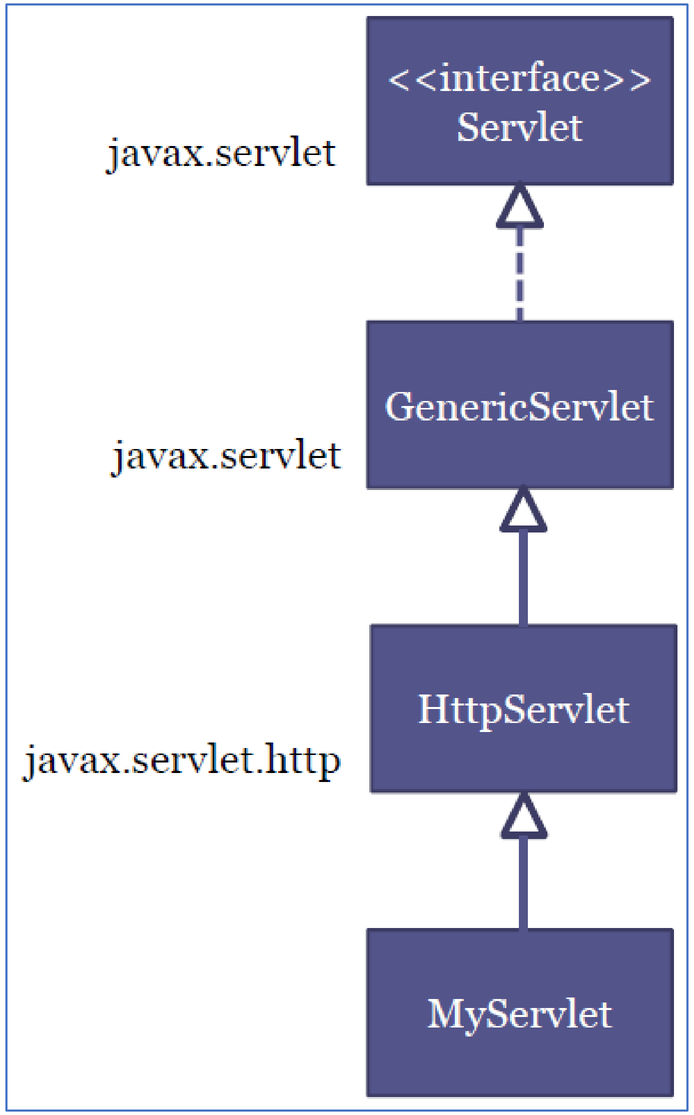
A servlet is a Java class that extends the capabilities of servers that host applications access by means of a request-response programming model.

Setting up our Development Environment

- Java JDK
- IntelliJ IDE
- Tomcat (Container)

Servlet Hierarchy

- Servlet
  - service()
  - init()
  - destroy()
- GenericServlet
- HttpServlet
  - doGet()
  - doPost()
  - doPut()
  - doHead()
  - doDelete()
  - doTrace()



My Servlet

```java
// MyServlet.java

public class MyServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
         PrintWriter out = resp.getWriter();
         out.print("Hello from my first servelt.");
    }
}
```

SimplestServlet

```
public class SimplestServlet extends HttpServlet {
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>");
  }
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>");  }}
```

Simple Servlet Using IntelliJ IDE and Tomcat Web Container/Server

- Very carefully follow the steps in the tutorial for creating a Maven servlet web application in IntelliJ
- Start by previewing the entire sequence and try to understand what you will be doing
- sakai .. Resources> lab helpers>Install tomcat application server in intellij.pdf
- sakai .. Resources> lab helpers>Day3.1 SimplestServlet_IntelliJ.pdf

- Note the following:
  - What is a Maven project and why is it useful?
  - Be sure your project matches the p4 screen before proceeding to p. 5
  - The html page (why is it called index.jsp?)
  - What will happen when you click on the hyperlink?
    <a href=‘hello'> Start the simplest servlet app </a>
  - When will doGet be called and what will it do?
  - When will doPost be called and what will it do?
- Try implementing it (follow the steps carefully)
- Now you know how to implement a web app in IntelliJ!

Three Names of a Servlet

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

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

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

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.

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.do</url-pattern>
- MUST begin with a slash (/).
- Can have an extension (like .do), but it's not required.
- **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>*.do</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>cs472.mum.HelloServlet</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>HelloServlet</servlet-name>
     <url-pattern>/hello</url-pattern>
  </servlet-mapping>
```

- Servlets can be declared and mapped using annotations instead of xml

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    ...
}
```

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.

Note: XML configurations will overwrite Annotations

Context Init parameters

- Context *initialization parameters* usually shortened to *init parameters*.
  - Can be put to any number of uses, from defining connection to database, to providing support email address.
  - You declare context init parameters using `<context-param>` tag with in the `web.xml` (no annotation alternative, you have to use DD)

```xml
<context-param>
    <param-name>support-email</param-name>
    <param-value>cstech.mum.edu</param-value>
</context-param>
```

```java
ServletContext sc = this.getServletContext();
sc.getInitParameter("support-email");
```

Servlet Init Parameters

- You can also obtain inti parameter from the `ServletConfig` object.

```
ServletConfig sc = this.getServletConfig();
sc.getInitParameter("database");
```

```
<servlet>
    <servlet-name>...</servlet-name>
    <servlet-class>...</servlet-class>
    <init-param>
        <param-name>database</param-name>
        <param-value>dbCustomer</param-value>
    </init-param>
</servlet>
```

- Creates init parameter specific to this Servlet.
    - Can opt to use annotation instead.
        - drawback is, you need to recompile the application after every change.

Main Point 1

Every platform for web applications has a mechanism to dynamically generate web pages containing information from the server. On the Java platform Servlets are the Java classes that provide this dynamism.

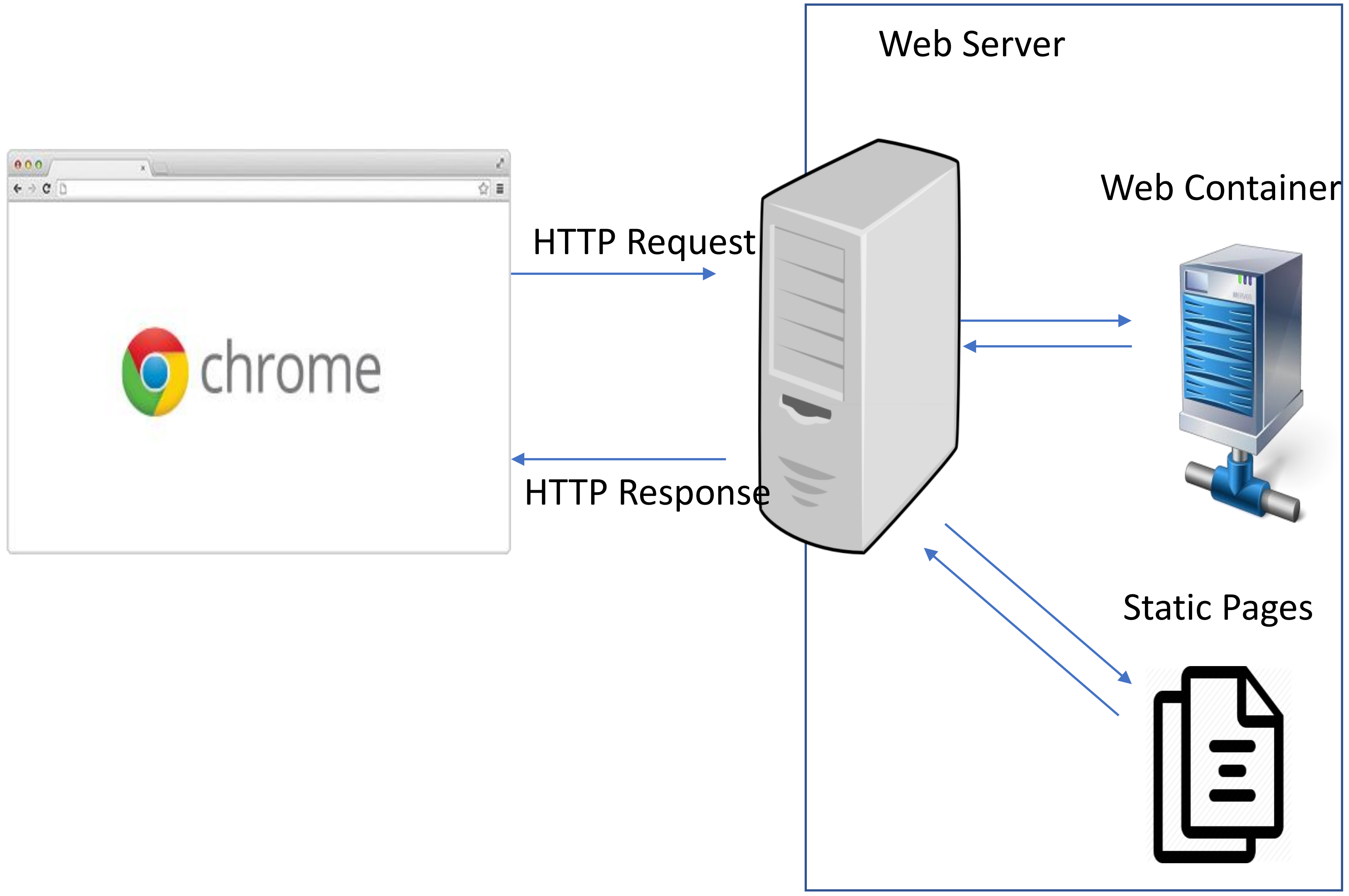
Science of Consciousness:

Pure consciousness is a field of infinite dynamism. We experience this as restful alertness during the practice of the TM Technique that give a basis for dynamic activity.

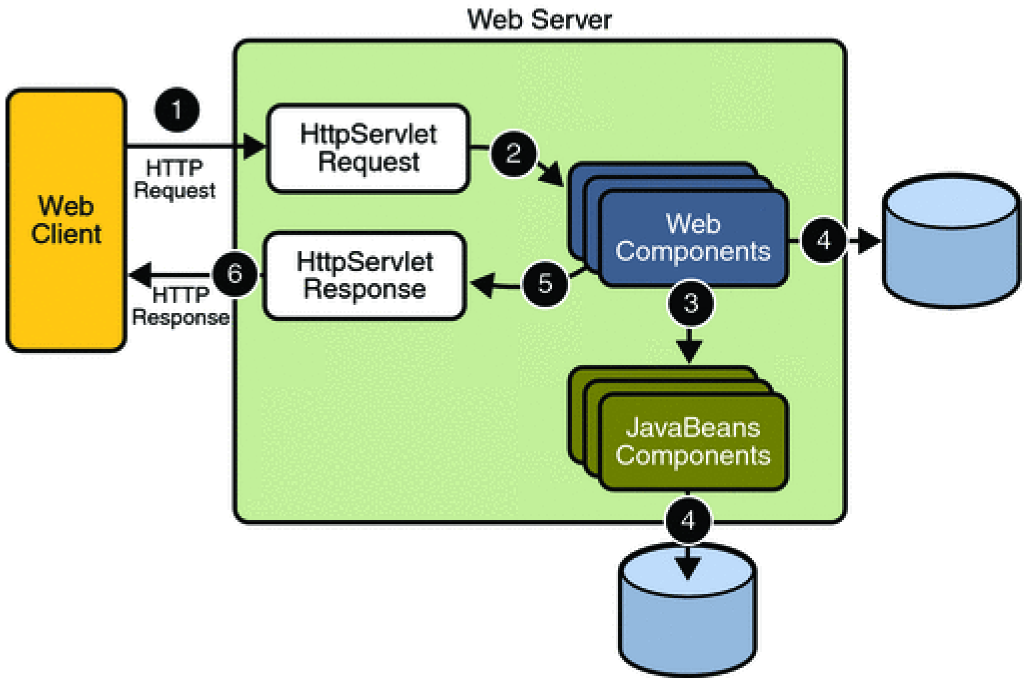
Main Point 2 Preview

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.

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

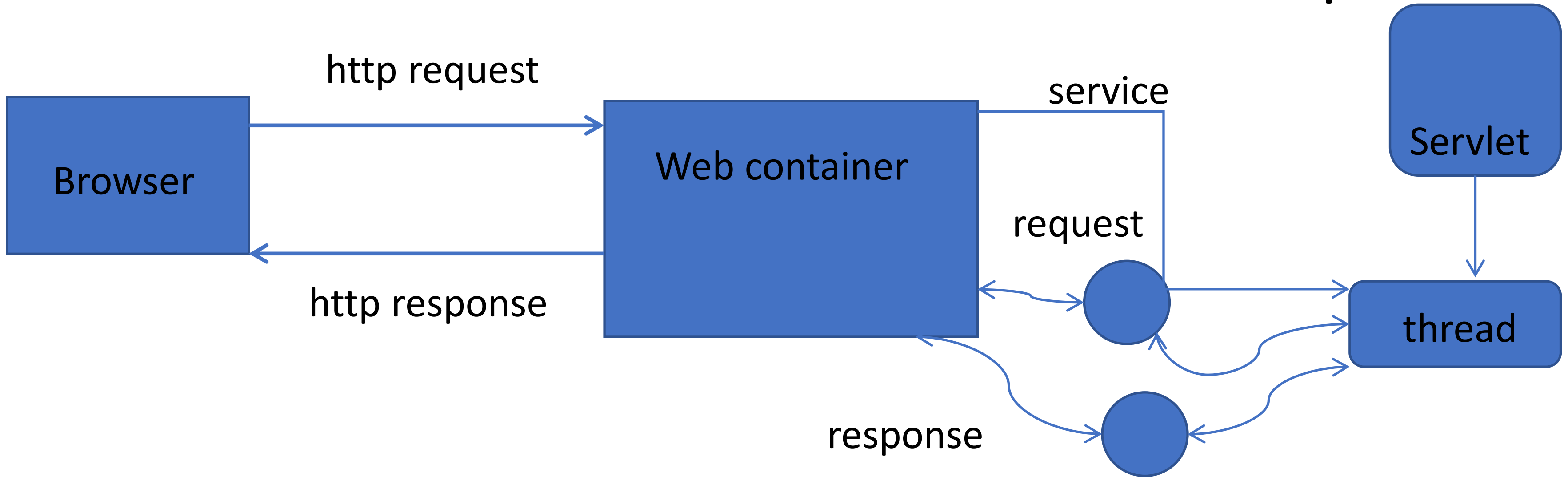
The Container

- 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

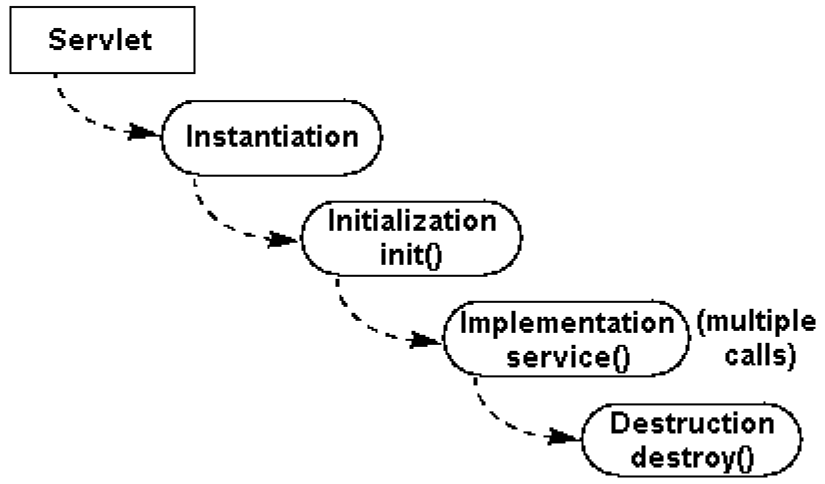


- Container receives new request for a servlet
- Creates HttpServletRequest and HttpServletResponse objects
- Obtains a new thread and calls service method on HttpServlet object in thread
- When thread completes, converts response object into HTTP response message

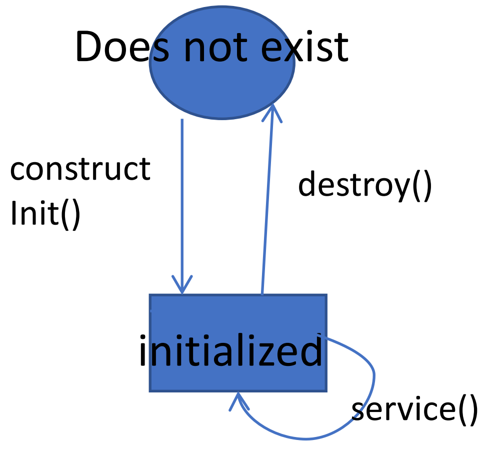
The `service()` method

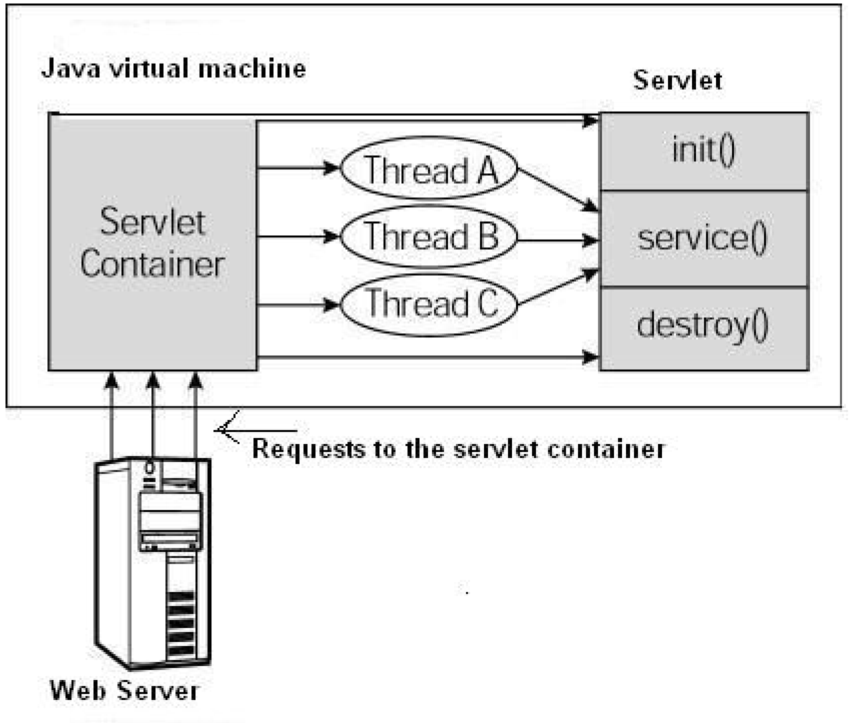
- The servlet container (Tomcat) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.
- Each time the server receives a request for a servlet, the server starts a new thread and calls service. The service() method checks 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.

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
- Compare with mylets

Using the Initializer and Destroyer

- `init` and `destroy` methods do nothing by default.
    - But you can override then to perform some actions:

```
@Override
public void init() throws ServletException {
    System.out.println("Servlet " + this.getServletName() + " has started.");
}

@Override
public void destroy() {
    System.out.println("Servlet " + this.getServletName() + " has stopped.");
}
```

Using the Initializer and Destroyer

- `init` is called after the Servlet is constructed but before it can respond to the first request.
  - access to the `ServletConfig` and `javax.servlet.ServletContext`
  - read a properties file or connect to database etc.

- `destroy` is called immediately after the Servlet can no longer accept the request.
  - when the web application is stopped or undeployed or container shuts down.
  - don't have to wait for garbage collector before cleaning of resources.

When does servlet get instantiated?

- Usually, servlet is instantiated and init called when the first request arrives for the Servlet
  - sufficient for most uses.
- if the `init` does many things, Servlet startup might become a time-intensive process
  - could make first request take several seconds or even several minutes!
  - A simple tweak to the servlet configuration can start servlet immediately when the web application starts:

```
<servlet>
    <servlet-name>...</servlet-name>
    <servlet-class>...</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

Service Parameters

- `HttpServletRequest` implements `ServletRequest`.
- `HttpServletResponse` implements `ServletResponse`.

- Managed by the container, passed to the servlet.
- Servlet sends them back to container to help it send the response back to the browser.

`HttpServletRequest`

- Using the request, the service method can tell if the http request was a GET or POST and delegate to the appropriate method in the servlet class.
- Get header information
  - `request.getHeader(name);`
- Get Session & Cookie related to the request.
- Get User information
- Get Path and url

Getting Request Parameters

- The `getParameter` method returns a single value for a parameter.
    - If the parameter has multiple values, `getParameter` returns the first value
    - `getParameterValues` returns an array of values for a parameter.
        - If the parameter has only one value, this method returns an array with one element in it.

- HTML

```
<input name="userName" type="text" />
```

- Servlet

```
String inputName = request.getParameter("userName");
```

```
String[] inputNames = request.getParameterValues("checkbox_name");
```

Using `HttpServletResponse`

- `HttpServletResponse` interface extends `ServletResponse`
- You use the response object to do this such as:
  - set response headers

```
response.setContentType("text/html");
response.setCharacterEncoding("UTF-8");
```

  - write to the response body

```
PrintWriter out = response.getWriter();
out.print("<html><head><title>Test</title></head><body>");
```

  - redirect the request
  - set the HTTP status code
  - send cookies back to the client

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.

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.