Java & JEE Training

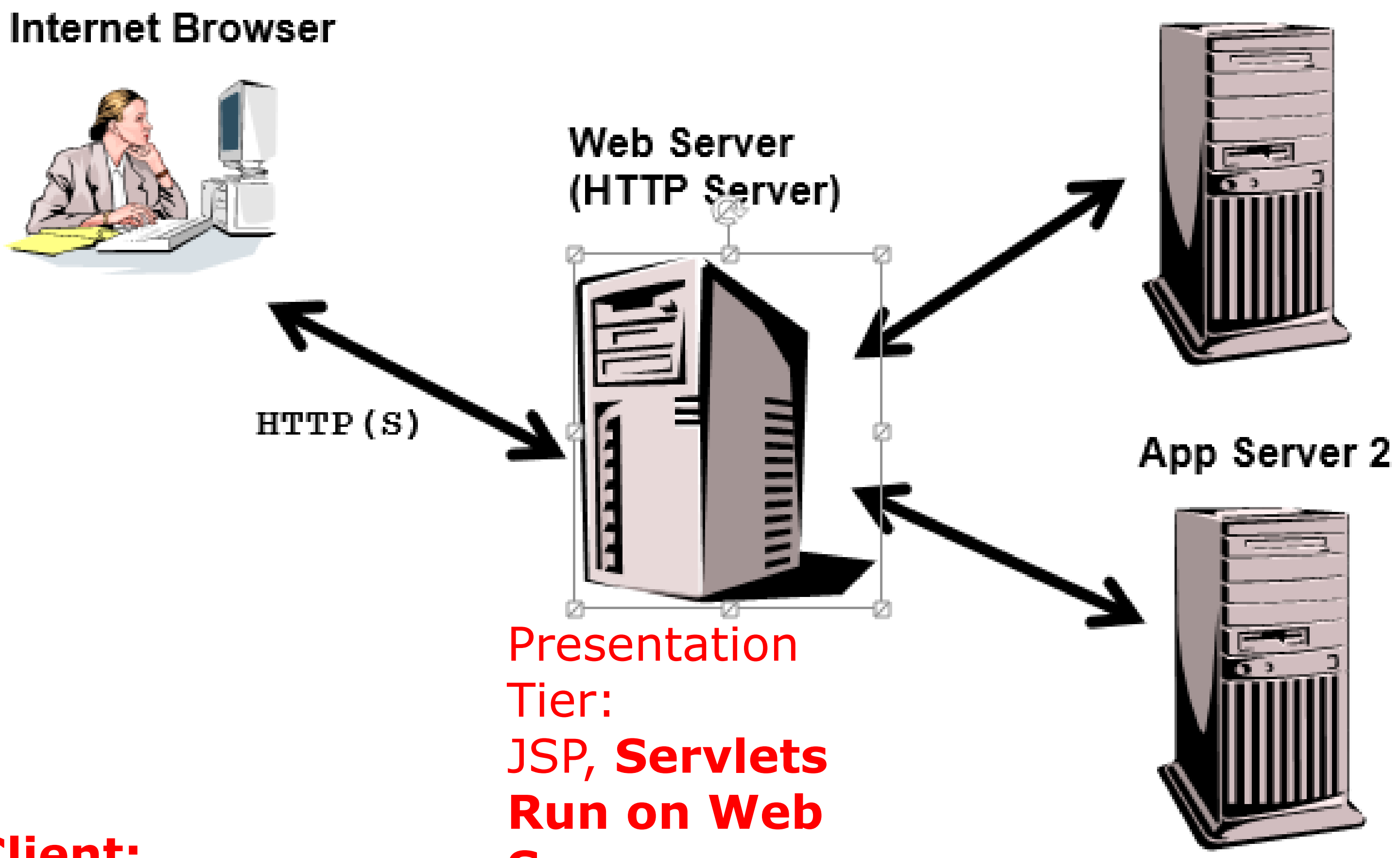
Day 29 – Servlets - Part 5

MindsMapped Consulting

Review of previous session…

….

Web Server and Application Server



Presentation Tier:
JSP, **Servlets Run on Web Server or container**

JDBC on web server or app server

**Client:
HTML JS CSS
(Run on client machine on the browser)**

Database

Demo of complete Web app flow…

- Form which takes two fields… Student ID and Student Name…
- Output should show whether that row exists or not, and if it exists, how many records in the table match the criteria (id, name)

- If match found, then print login successful, else login failed.

Validation in web applications

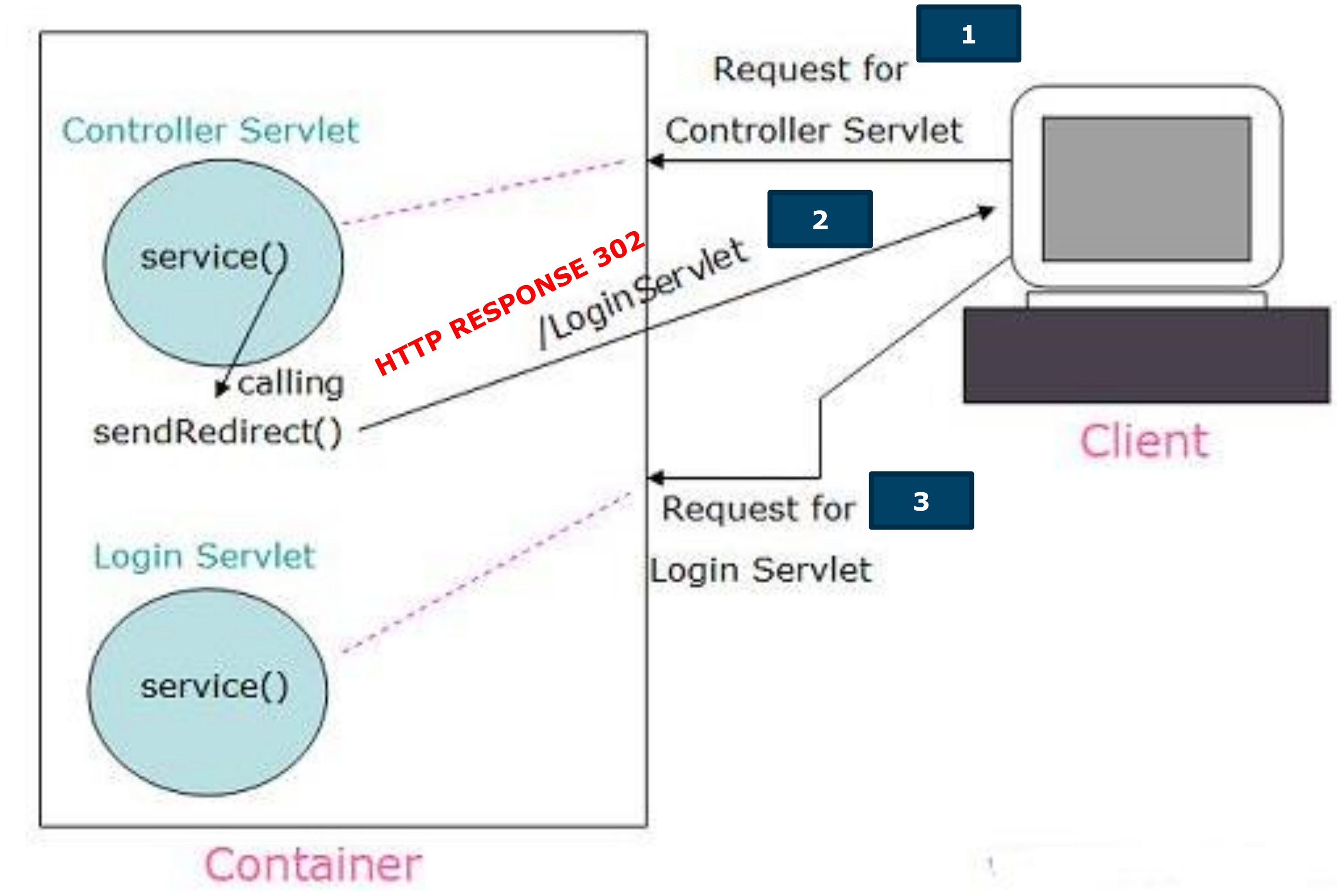
- Validation logic should be implemented BOTH
  - Frontend – JavaScript
  - Backend - Java

Redirecting requests to other sources

MindsMapped Consulting

Redirecting Requests to Other Resources

- Servlet **`RedirectServlet`**
  - Redirects the request to a different resource
  - Why not just have links to those resources?
    - Could determine browser type and choose link accordingly
    - Could add parameters before redirecting (those sent originally are included automatically)
  - `response.sendRedirect(URL);`
    - redirects to that URL
    - immediately terminates current program
    - Use relative paths whenever possible to make Website portable


1
Request for
Controller Servlet
Controller Servlet
service()
calling
sendRedirect()
HTTP RESPONSE 302
/LoginServlet
2
Client
Request for
3
Login Servlet
Login Servlet
service()
Container

RedirectServlet redirecting requests to other resources.

```
// RedirectServlet.java
// Redirecting a user to a different Web page.

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class RedirectServlet extends HttpServlet {

   // process "get" request from client
   protected void doGet( HttpServletRequest request,
      HttpServletResponse response )
         throws ServletException, IOException
   {
      String location = request.getParameter( "page" );

      if ( location != null )

         if ( location.equals( "deitel" ) )
            response.sendRedirect( "http://www.deitel.com" );
         else
            if ( location.equals( "welcome1" ) )
               response.sendRedirect( "welcome1" );

```

Obtains the **page** parameter from the request.

Redirects the request to www.deitel.com.

Redirects the request to the servlet `WelcomeServlet`.

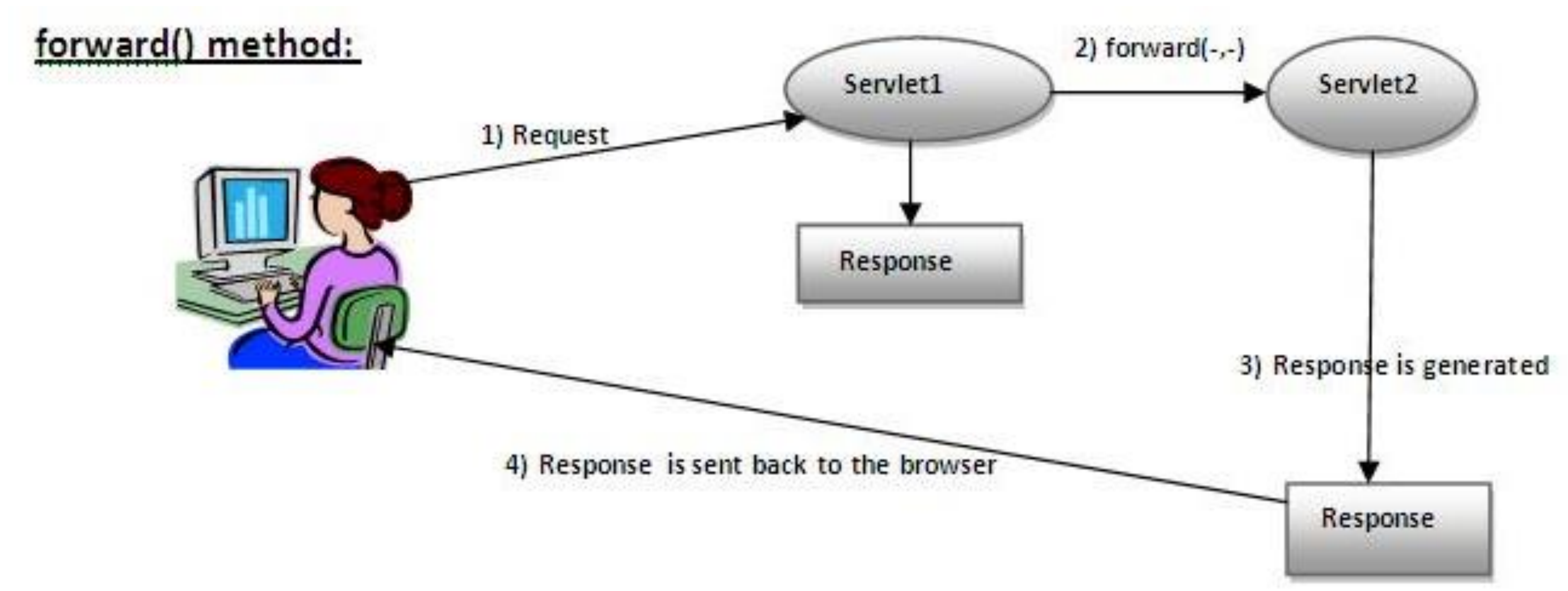
CRUD…

- Select
- Update
- Insert
- Delete

- 1. /Hello?op=Insert OR /Hello?op=Delete

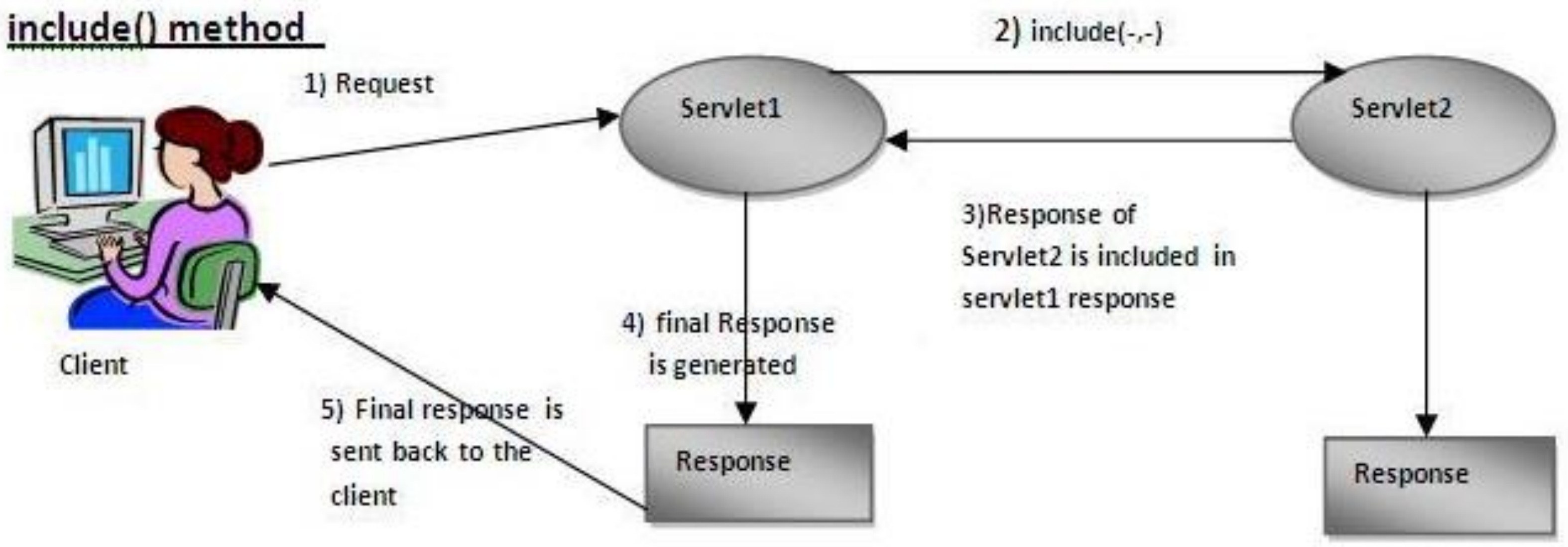
- 2. /Hello/Insert OR /Hello/Delete

RequestDispatcher forward()



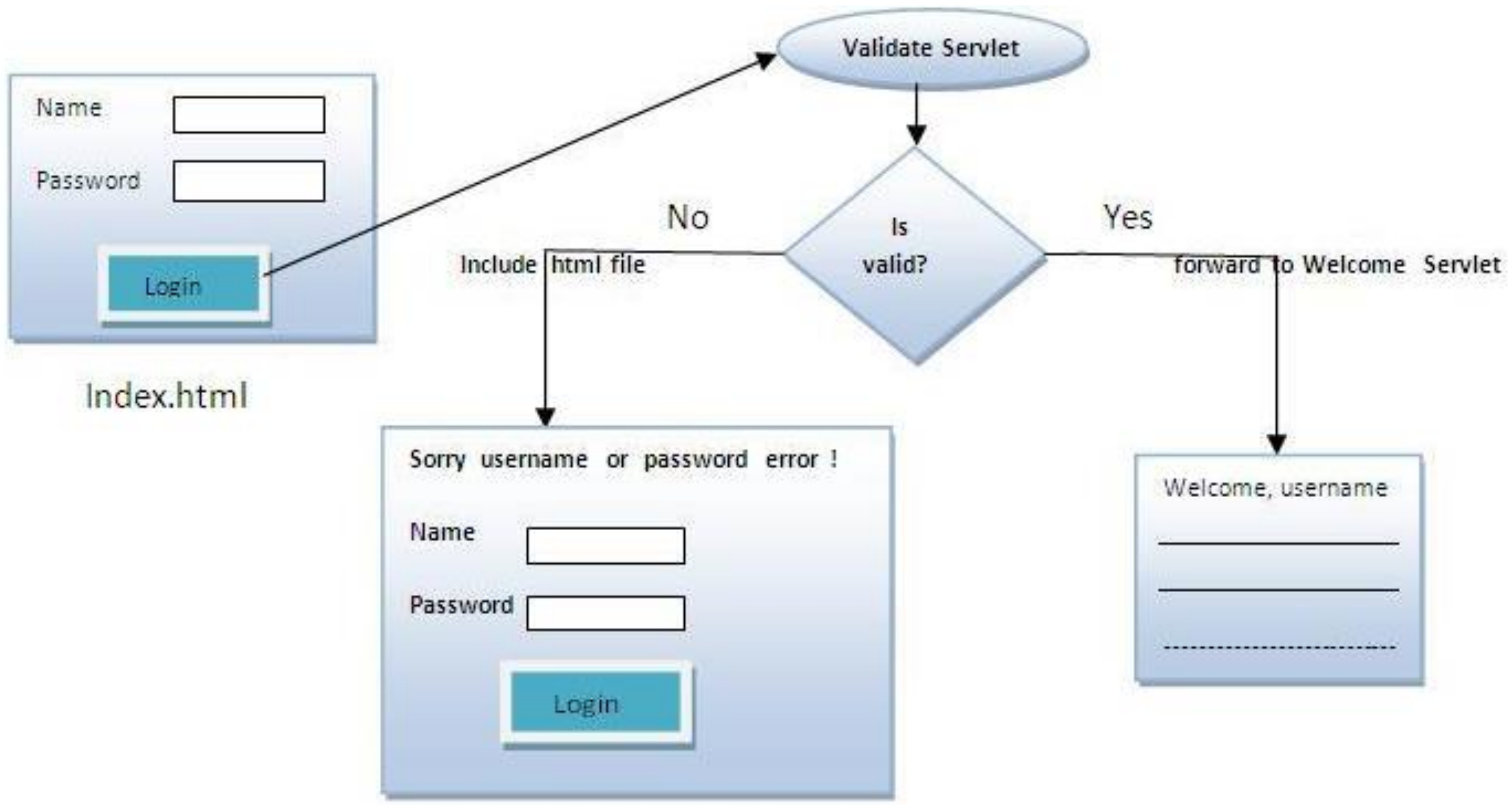
```
RequestDispatcher rd=request.getRequestDispatcher(“<another servlet>");
rd.forward(request, response);
```

RequestDispatcher include()


include() method
1) Request
2) include(-,-)
Servlet1
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
5) Final response is sent back to the client
Client
Response
Response

```
out.print("<Your message to include goes here>");
RequestDispatcher rd=request.getRequestDispatcher("nameOfHTMLorJSP");
rd.include(request, response);
```

Request Dispatcher Demo



Java & JEE Training

Understanding scope: parameters, attributes

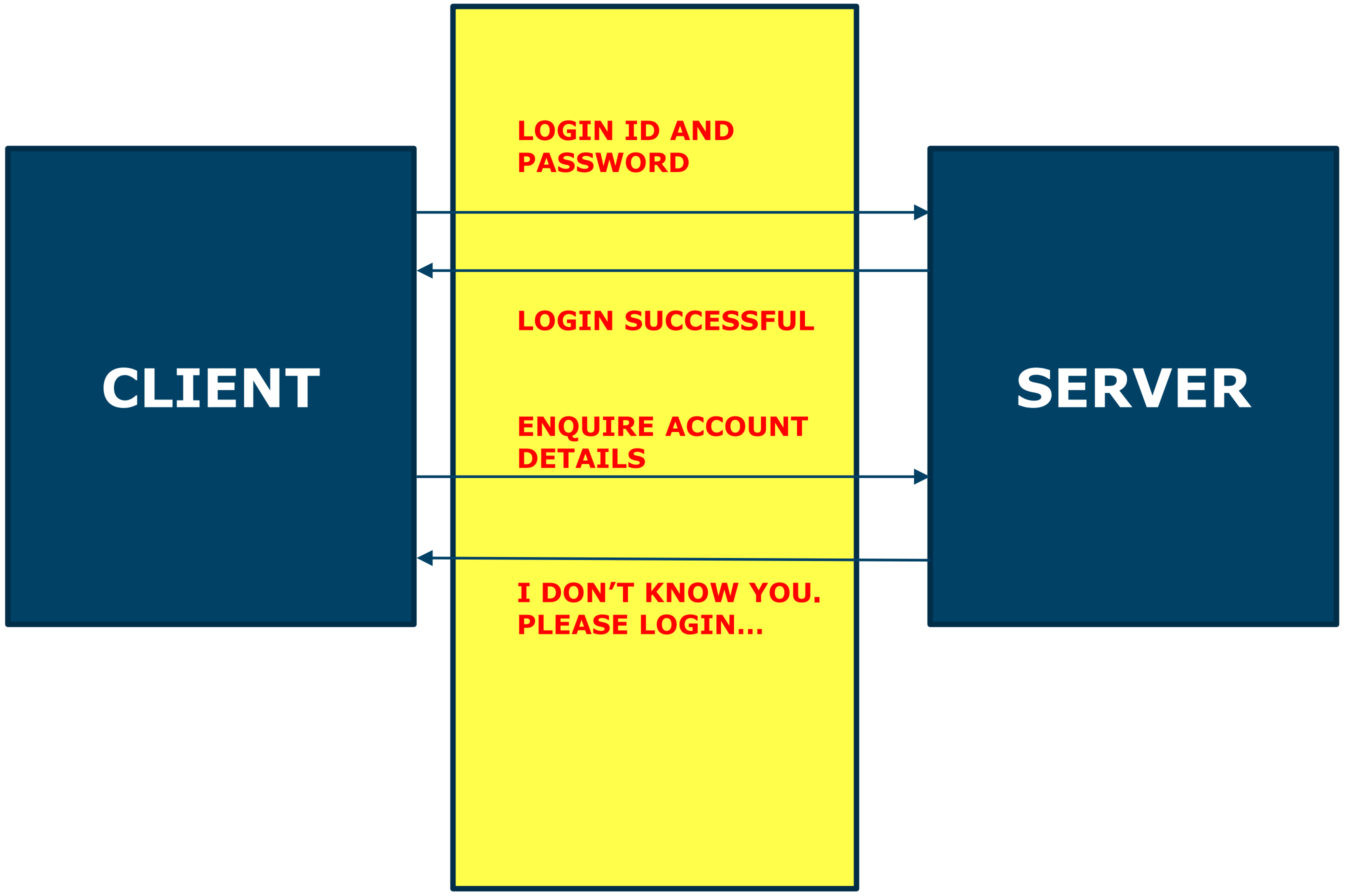
MindsMapped Consulting

Requests, Sessions, Application

- **Request**: The client sending an HTTP request to the server.
- **Session**: Is a set of requests coming in from the client from the point that the client logs in to the point that the client logs out.
- **Application**: The time that the application (or server) is up and running.

- An application may have multiple sessions.
- A session may have multiple requests.

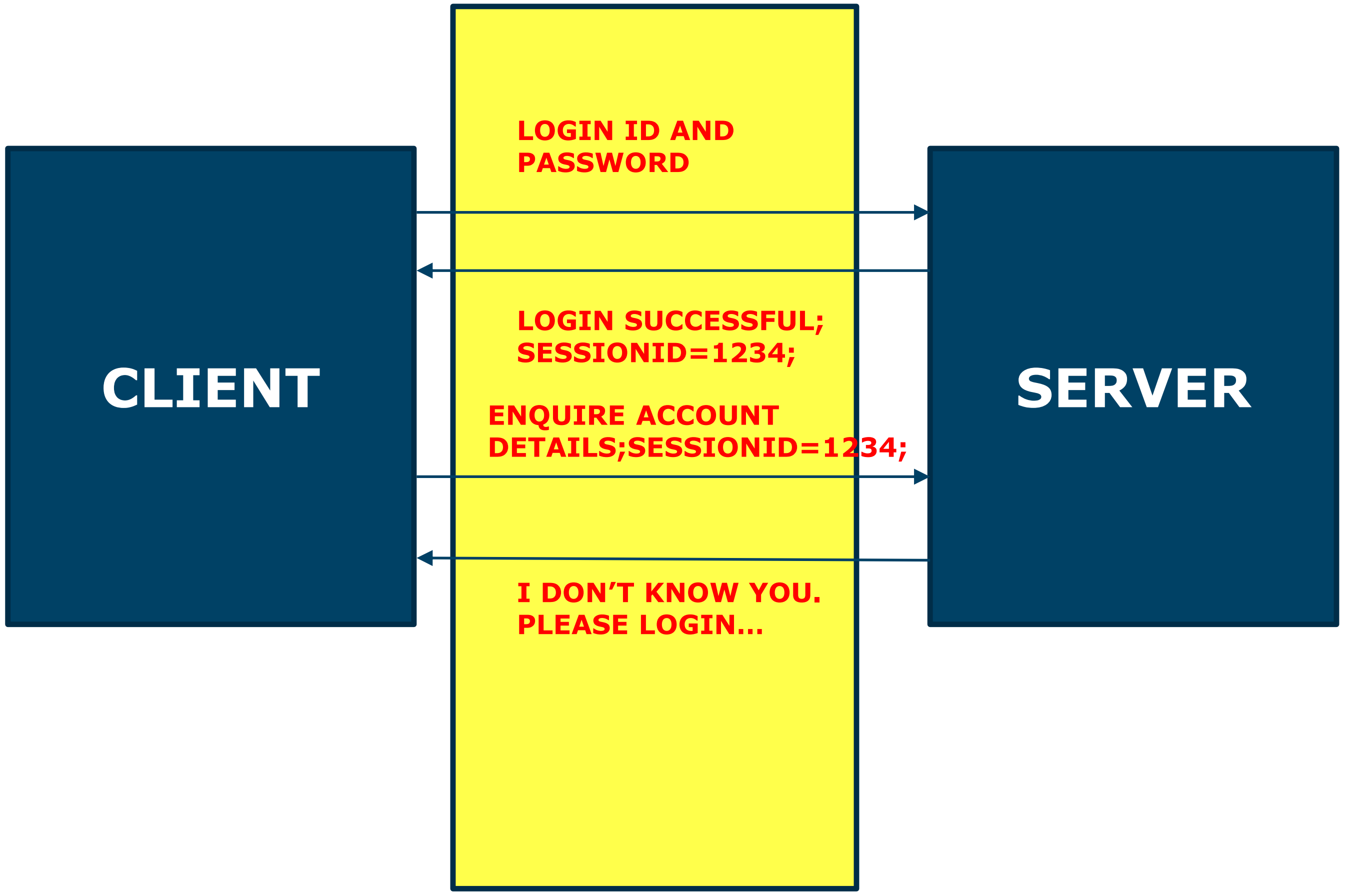
HTTP IS STATELESS



SESSION
- COOKIES…

HOW DO I MAINTAIN SESSIONS?

HTTP IS STATELESS


LOGIN ID AND PASSWORD
CLIENT
SERVER
LOGIN SUCCESSFUL; SESSIONID=1234;
ENQUIRE ACCOUNT DETAILS;SESSIONID=1234;
I DON'T KNOW YOU. PLEASE LOGIN…

SESSION
- COOKIES

HOW DO I MAINTAIN SESSIONS?

Parameters vs Attributes

- Parameters may come into our application **from the client request, or may be configured through deployment descriptor (web.xml)** elements or their corresponding annotations. When you submit a form, form values are sent as request parameters to a web application. In case of a GET request, these parameters are exposed in the URL as name value pairs and in case of POST, parameters are sent within the body of the request.
- Servlet init parameters and context init parameters are set through the deployment descriptor (web.xml) or their corresponding annotations. All parameters are **read-only** from the application code. We have methods in the Servlet API to retrieve various parameters.
- **Attributes** are objects that are attached to various scopes and can be modified, retrieved or removed. Attributes can be read, created, updated and deleted by the web container as well as our application code. We have methods in the Servlet API to add, modify, retrieve and remove attributes. When an object is added to an attribute in any scope, it is called **binding** as the object is bound into a scoped attribute with a given name.

Parameters vs Attributes

- Parameters are read only, attributes are read/write objects.
- Parameters are String objects, attributes can be objects of any type.

Methods to manipulate parameters

- Request: (Request time constant)
  `ServletRequest.`**`getParameter`**`(paramname);`
  `ServletRequest.`**`getParameterNames`**`();`

- ServletContext init parameters: (Deployment time constant)
  `ServletContext.`**`getInitParameterNames`**`()`
  `ServletContext.`**`getInitParameter`**`(String paramName)`

- ServletConfig init parameters: (Deployment time constant)
  `ServletConfig.`**`getInitParameterNames`**`()`
  `ServletConfig.`**`getInitParameter`**`(String` paramName)

ServletConfig vs ServletContext

- **ServletConfig** **// Available for a specific servlet**
  - ServletConfig available in javax.servlet.*; package
  - ServletConfig object is one per servlet class
  - Object of ServletConfig will be created during initialization process of the servlet
  - This Config object is public to a particular servlet only
  - *Scope*: As long as a servlet is executing, ServletConfig object will be available, it will be destroyed once the servlet execution is completed.
  - We should give request explicitly, in order to create ServletConfig object for the first time
  - In web.xml – <*init-param*> tag will be appear under <*servlet-class*> tag

- **ServletContext** **// Available for whole application**
  - ServletContext available in javax.servlet.*; package
  - ServletContext object is global to entire web application
  - Object of ServletContext will be created at the time of web application deployment
  - *Scope*: As long as web application is executing, ServletContext object will be available, and it will be destroyed once the application is removed from the server.
  - ServletContext object will be available even before giving the first request
  - In web.xml – <*context-param*> tag will be appear under <*web-app*> tag

Example.. Identify the init parameters here..

```
<servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/config/springmvc-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>ApplicationName</param-name>
    <param-value>ApplicationName</param-value>
  </context-param>
</web-app>
```

How to get servletContext?

```
getServletConfig( ).getServletContext( );
request.getSession( ).getServletContext( );
getServletContext( );
request.getServletContext();
```

All are correct. Which one would you use?

Understanding scope of attributes

- Similar to scope and lifetime of variables in Java as you have seen in blocks and methods in java, parameters and attributes in a Java EE web application also have scope and lifetime in the context of the web application.
- The scope of a parameter/attribute denotes the availability of that parameter/attribute for use. A web application serves multiple requests from clients when it is up and running. These requests can be from same client or different clients. We have seen from the servlet life cycle that a servlet’s service() method is called every time a request comes.

Scoping in Servlets and JSP

- Request scope
- Session scope
- Application or context scope
- Page scope (only for JSP)

Request Scope

- Request scope start from the moment an HTTP request hits a servlet in our web container and end when the servlet is done with delivering the HTTP response.
- With respect to the servlet life cycle, the request scope begins on entry to a servlet’s service() method and ends on the exit from that method.
- A ‘request’ scope parameter/attribute can be accessed from any of servlets or jsps that are part of serving one request. For example, you call one servlet/jsp, it then calls another servlet/jsp and so on, and finally the response is sent back to the client.
- **Request** scope is denoted by javax.servlet.http.**HttpServletRequest** interface.
- Container passes the request object as an argument of type HttpServletRequest to Servlet's service method.
- Request object is available in a JSP page as an **implicit object** called **request**. You can set value for an attribute in request object from a servlet and get it from a JSP within the same request using the implicit request object.

Session scope

- A session scope starts when a client (e.g. browser window) establishes connection with our web application till the point where the browser window is closed.
- Session scope spans across multiple requests from the same client.
- A notable feature of tabbed browsing is that session is shared between the tabs and hence you can requests from other tabs too during a session without logging in again. For instance, you can load your Gmail inbox in another tab without logging in again. This also means browsing an unknown site and a secure site from different tabs from the same browser can expose your secure session ID to malicious applications. So always open a new browser window when you want to do secure transactions, especially financial transactions.
- **Session** scope is denoted by javax.servlet.http.**HttpSession** interface.
- Session object is available in a JSP page as an **implicit object called session**.
- In a servlet, you can get Session object by calling **request.getSession()**.

Application or context scope

- Context scope or application scope starts from the point where a web application is put into service (started) till it is removed from service (shutdown) or the web application is reloaded. Parameters/attributes within the application scope will be available to all requests and sessions.
- **Application** scope is denoted by javax.servlet.**ServletContext** interface.
- Application object is available in a JSP page as an **implicit object** called **application**.
- In a servlet, you can get application object by calling **getServletContext()** from within the servlets code directly (the servlet itself implements the ServletConfig interface that contains this method) or by explicitly calling **getServletConfig().getServletContext()**.
- The web container provides one ServletContext object per web application per JVM.

Page scope (Only for JSP, not for Servlets)

- The page scope restricts the scpoe and lifetime of attributes to the same page where it was created.
- **Page** scope is denoted by javax.servlet.jsp.**PageContext** abstract class.
- It is available in a JSP page as an **implicit object** called **pageScope**

Servlets – Scope at a glance

- Application scope

```
request.getServletContext();
request.getServletContext().setAttribute("attribute_name","value")
```

- Session scope

```
request.getSession(); //going to create the session if session is not
exist.
request.getSession(false); // Not going to create the session.
session.getAttribute("attribute_name");
```

- Request scope

```
request.setAttribute("attribute_name","value");
request.getAttribute("attribute_name"); // return the Object you have to
cast it
```