Tech Mahindra

JEE – Servlets

TLS

Rewards and Recognition

Objectives

At the end of this session, you will be able to,

- Describe Web Architecture
- Describe structure of Web Application
- Define Servlets
- Describe Life Cycle of Servlets
- Describe Servlet API
- Design, Deploy and execute Servlet Program on application server
- Describe and Demonstrate Servlet Config and Servlet Context
- Describe and Demonstrate InterServlet Communication
- Describe State Management and its ways
- Describe and Demonstrate Connection Pooling using JNDI

Agenda

Following points to be covered in the session,

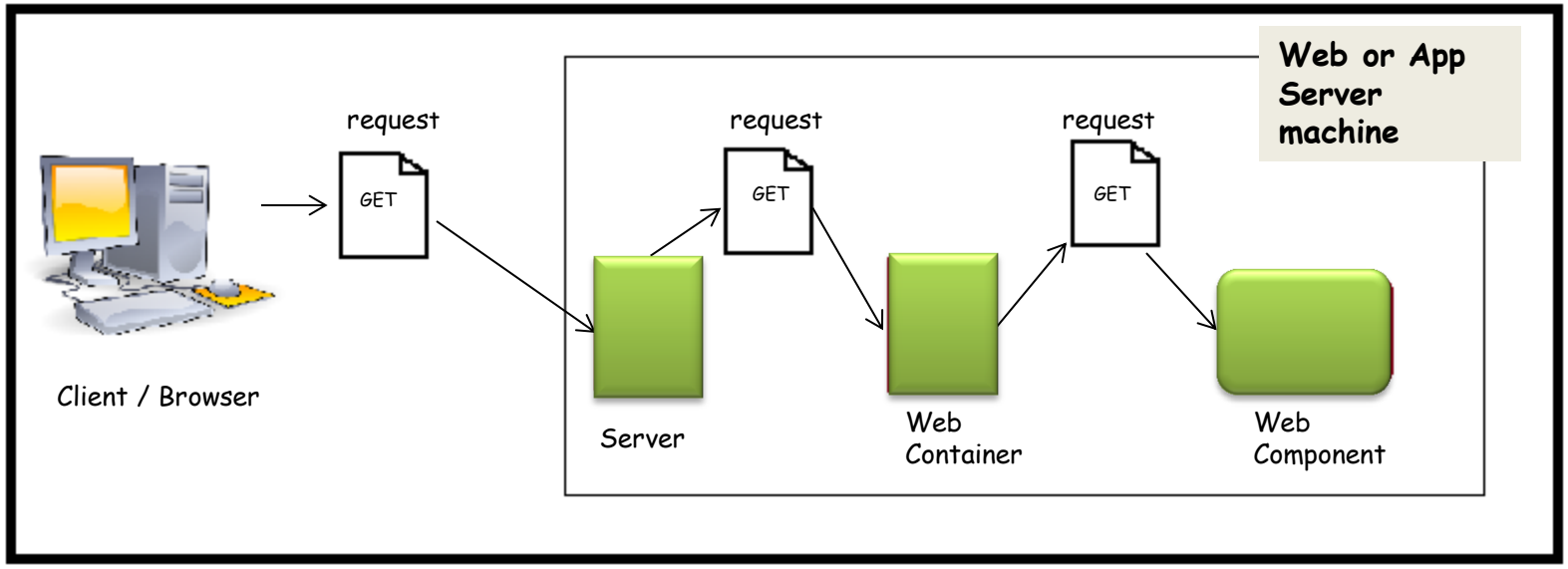
- Web Component
- Servlets
- Life Cycle of Servlets
- Servlet API
- Servlet Configuration with annotations
- Servlet Config and Servlet Context
- InterServlet Communication
- JNDI and DataSource Configuration on Application Server
- Filters and Listners
- State Management
- State Management techniques

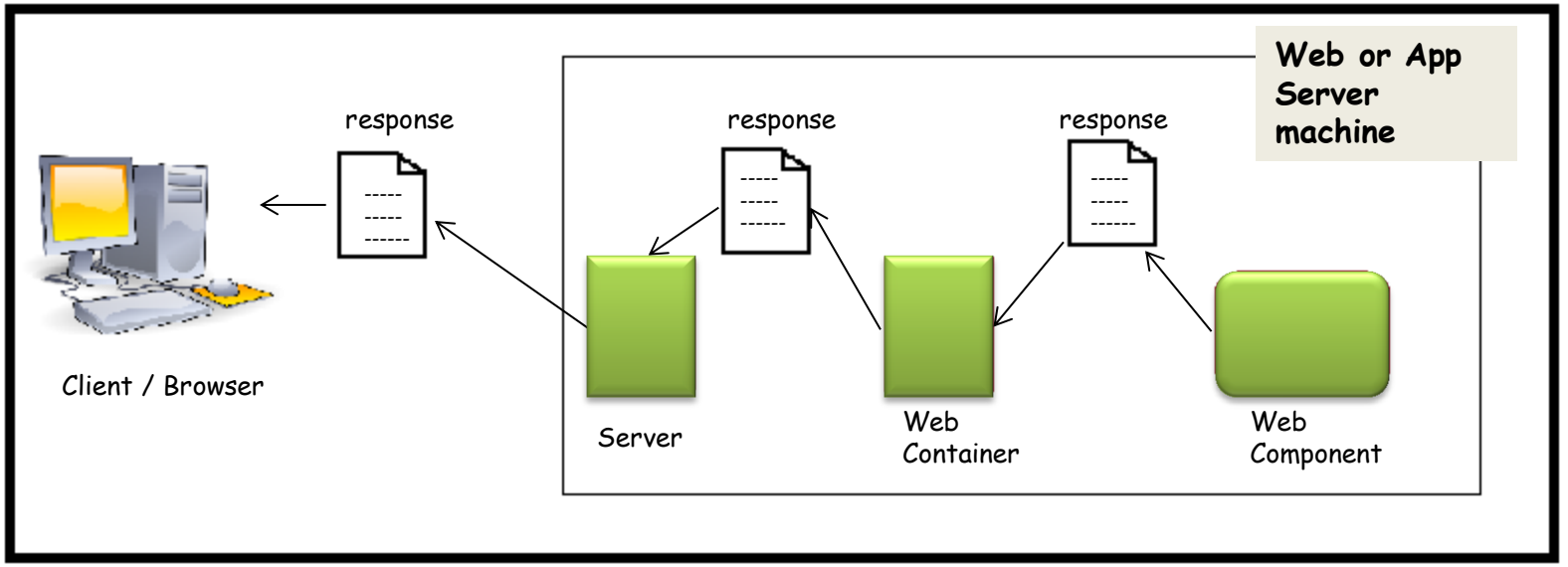
Web Component

TLS

Rewards and Recognition

Web Architecture





Web Application

- A Web application is a dynamic extension of a Web server.
- A Web application consists of Web components, static resource files such as images, and helper classes and libraries
- Web components provide the dynamic extension capabilities for a Web server.
- Web components are either Java Servlets or JSP pages.
- Web components are supported by the services of a runtime platform called a Web container.
  - The Web container provides services such as request dispatching, security, concurrency and life cycle management.
  - It also gives Web components access to APIs such as naming, transactions, and e-mail

Web Application

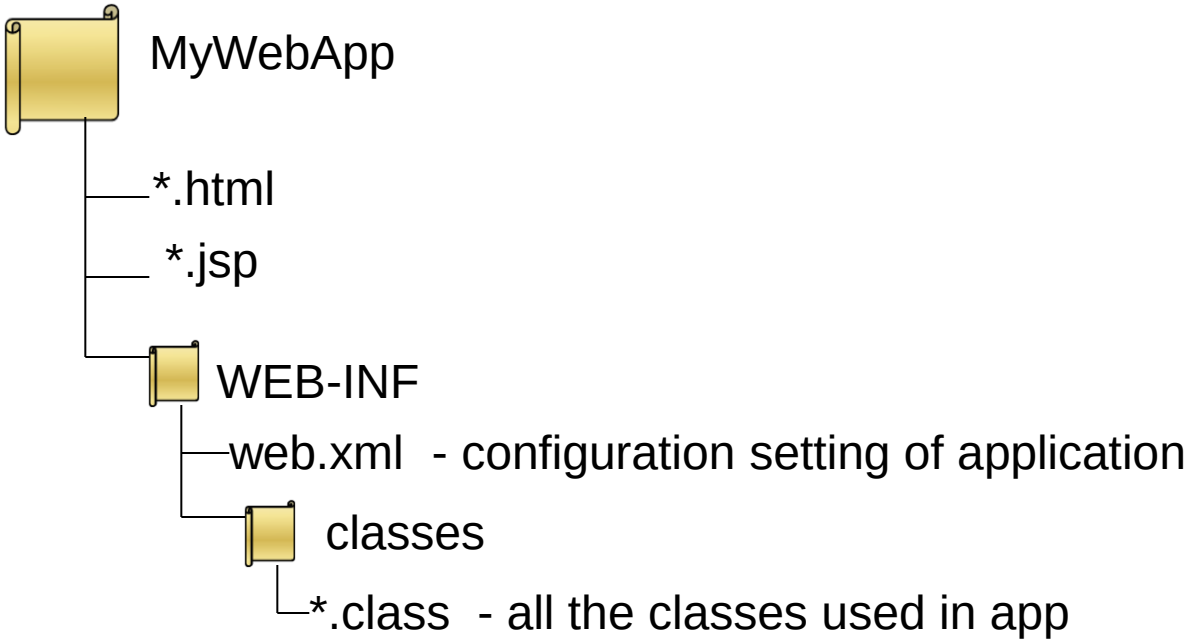
- Certain aspects of Web application behavior can be configured when the application is deployed to the Web container.
- The configuration information is maintained in XML format called a Web application **deployment descriptor(web.xml).**
- Web components and static Web content files such as images are called Web resources.
- A Web module is the smallest deployable and usable unit of Web resources in a J2EE application.

Web Application Creation …..Steps

- The process for creating, deploying, and executing a Web application can be summarized as follows:

1. Develop the Web component code (including deployment descriptor or annotations).
2. Build the Web application components along with any static resources (for example, images) and helper classes referenced by the component.
3. Install or deploy the application into an Application Server.
4. Access a URL that references the Web application.

Structure of a Web Application



Web Development

Web development has two aspects,

1. Client side technologies

- **HTML** – lightweight markup language which is a W3C std
- DHTML – Dynamic effects can be introduced using
- Applet - Sun
- Java script – Netscape
- Vb Script – Microsoft
- Style sheets
- Angular JS
- JQuery

Web Development (Contd.)

2. Server side technologies

- CGI – Common Gateway Interface
- ASP – Active Server pages, interpreted platform dependent technology from Microsoft
- **Servlet /JSP** – object oriented, compiled, platform independent technology from sun microsystem.
- ASP.Net – Object oriented ,platform independent web technology from Microsoft.

Need for Server Side Technology

- Processing client request
- Implementing Business logic
- Generating HTML response for client
- If resource is not available then generate appropriate error message
- Making data access from various databases

Tech Mahindra

Servlets

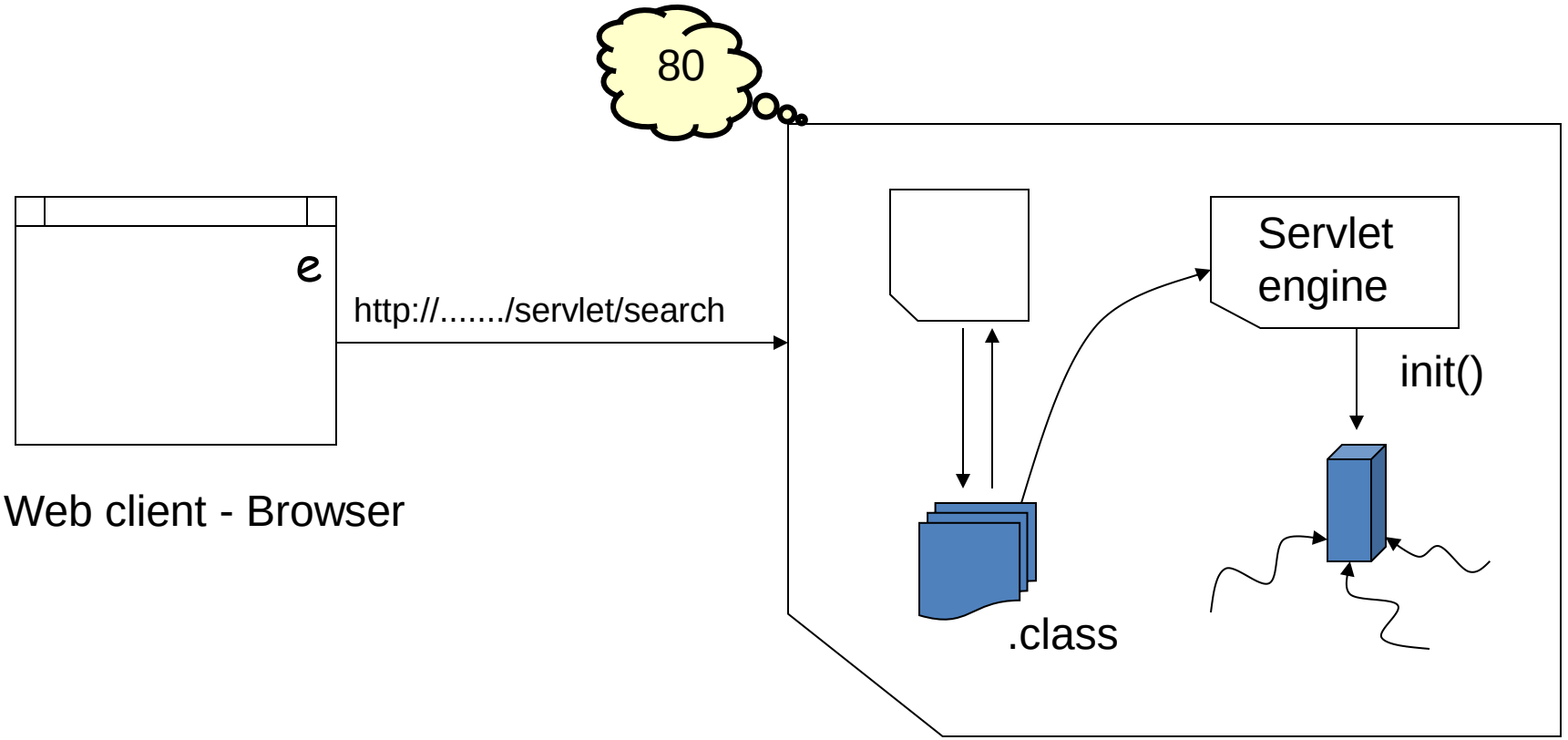
TLS

Rewards and Recognition

Servlets

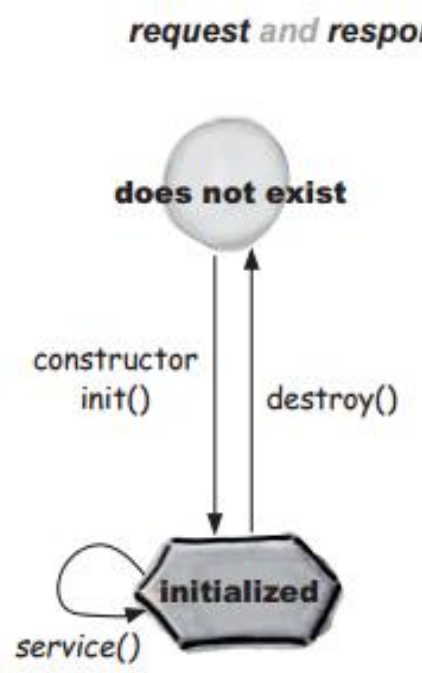
- A servlet is a Java programming language class used to extend the functionality of HTTP servers
- A servlet is accessed via a request-response programming model
- Features and Advantages :
  - Platform Independent
  - Access to Java APIs
  - Efficiency and endurance
  - Safety
  - Extensibility and flexibility

Servlet Architecture



Lifecycle of Servlet

- Lifecycle methods from javax.servlet.Servlet interface
  - **init()**
    called only once, used for initialization
  - **service()**
    called for all the request, used for logic implementation
  - **destroy()**
    Called before servlet engine destroys servlet instance

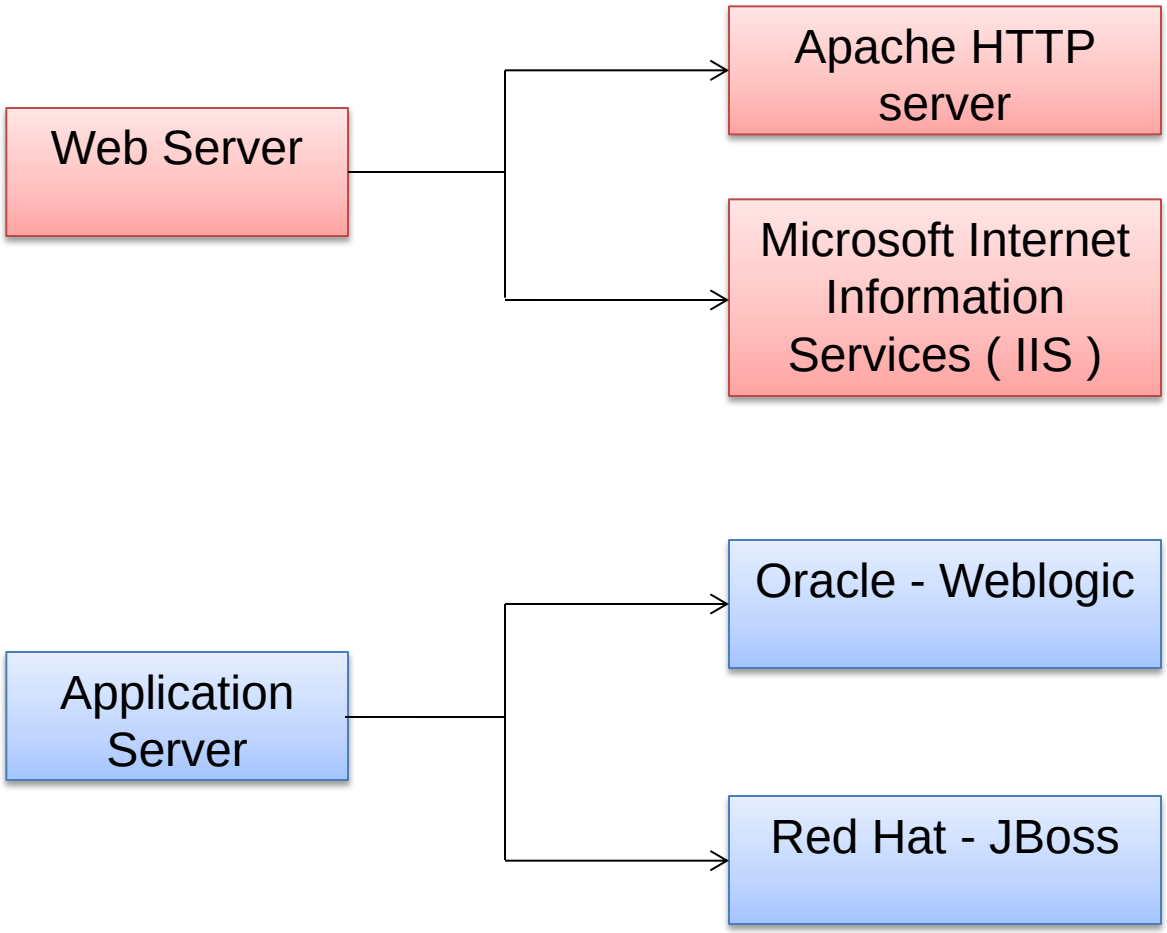


Request / Response Model

- service() method of Servlet Interface receives two parameters. (ServletRequest and ServletResponse objects)
- Request response object will be managed by servlet engine
- Every user call is associated with Request and Response object
- Request object will carry all the client details to server
- Server will send information to client through Response object
- Request object is loaded with Http headers which it will carry to the server

Web and Application Server

- Few of the popular servers available in the market are :

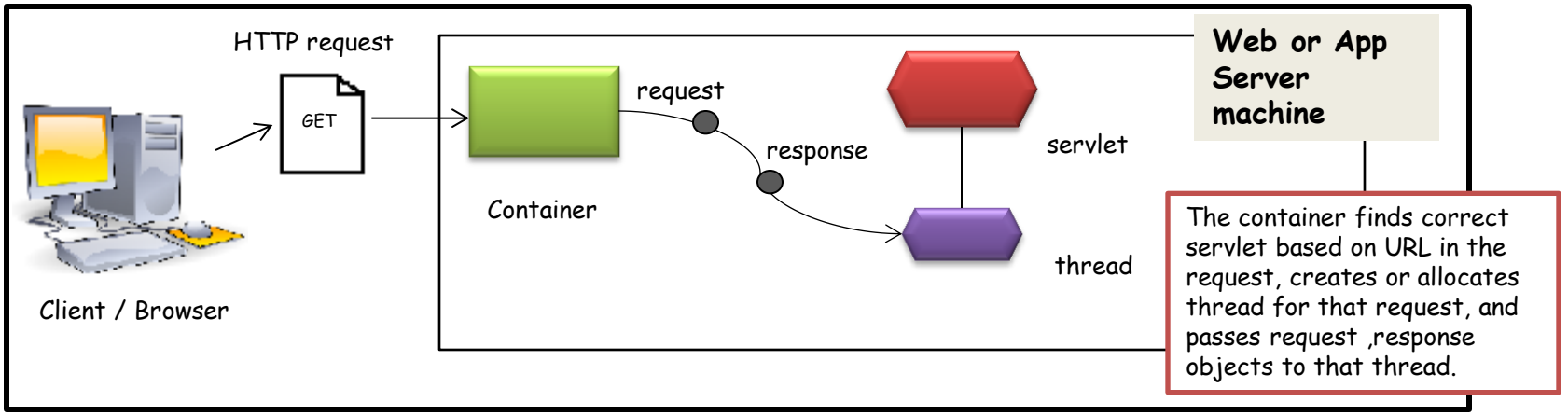


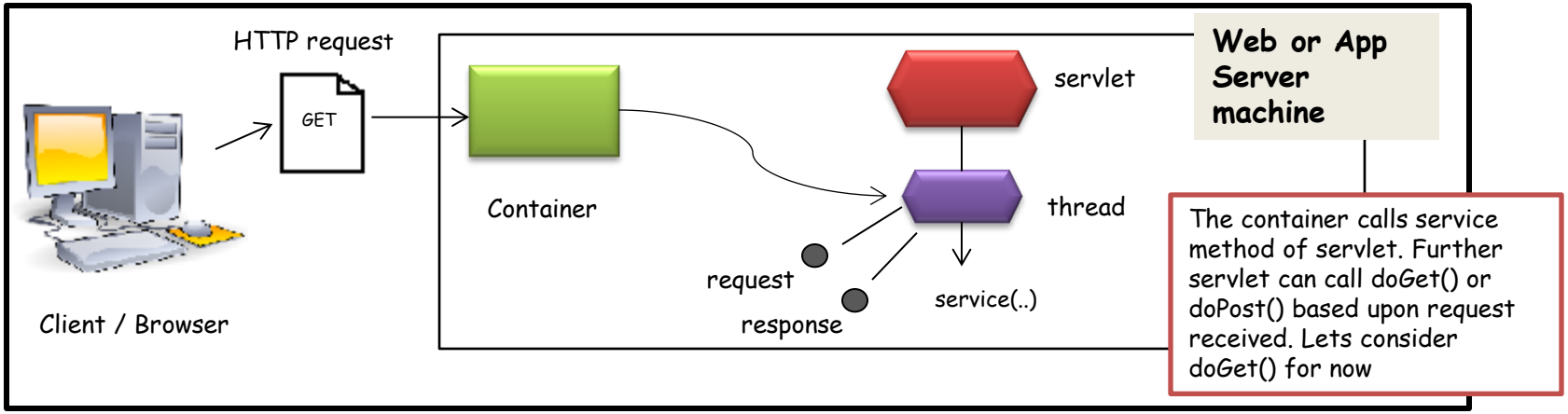
Web Container

- Along with server we need one more entity to function i.e. Web Container. Eg Apache server has Tomcat web container.
- Functionalities of Web Container
  - Communications Support for Servlets to interact with Server
  - Lifecycle Management for servlets
  - Multithreading Support
  - Declarative security using deployment descriptor
  - JSP support

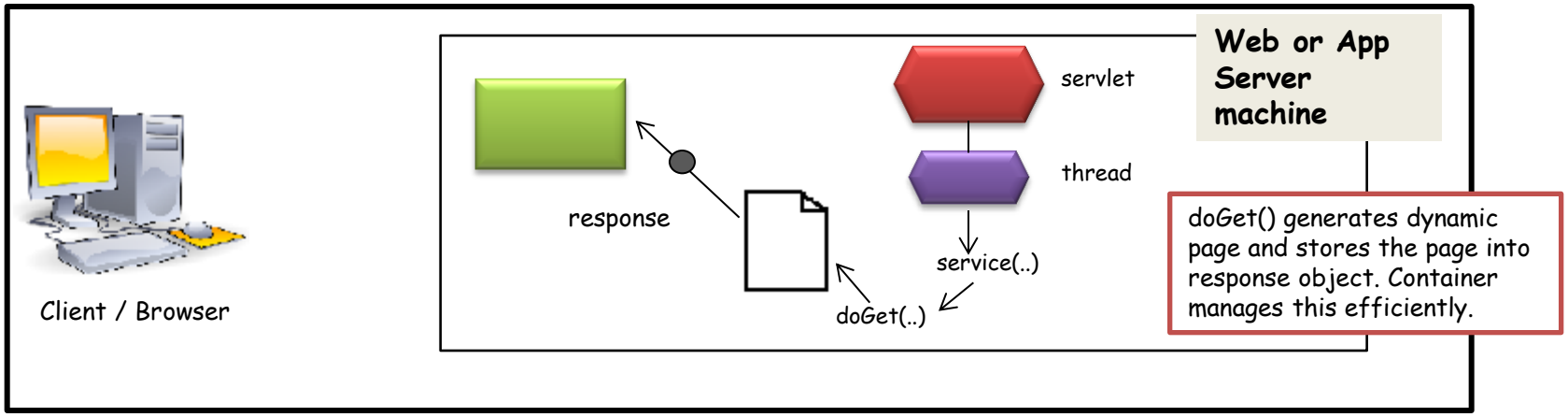
Web Container (Cont..)

- When the request comes on server side following steps are performed..





Web Container (Cont..)



- Once the response is sent back to Client, following objects created in memory are destroyed as they contain client's request data
  - thread
  - request
  - response
- Servlet instance will be there in memory to serve other client requests. It will only be removed if server shuts down or application is removed from server.

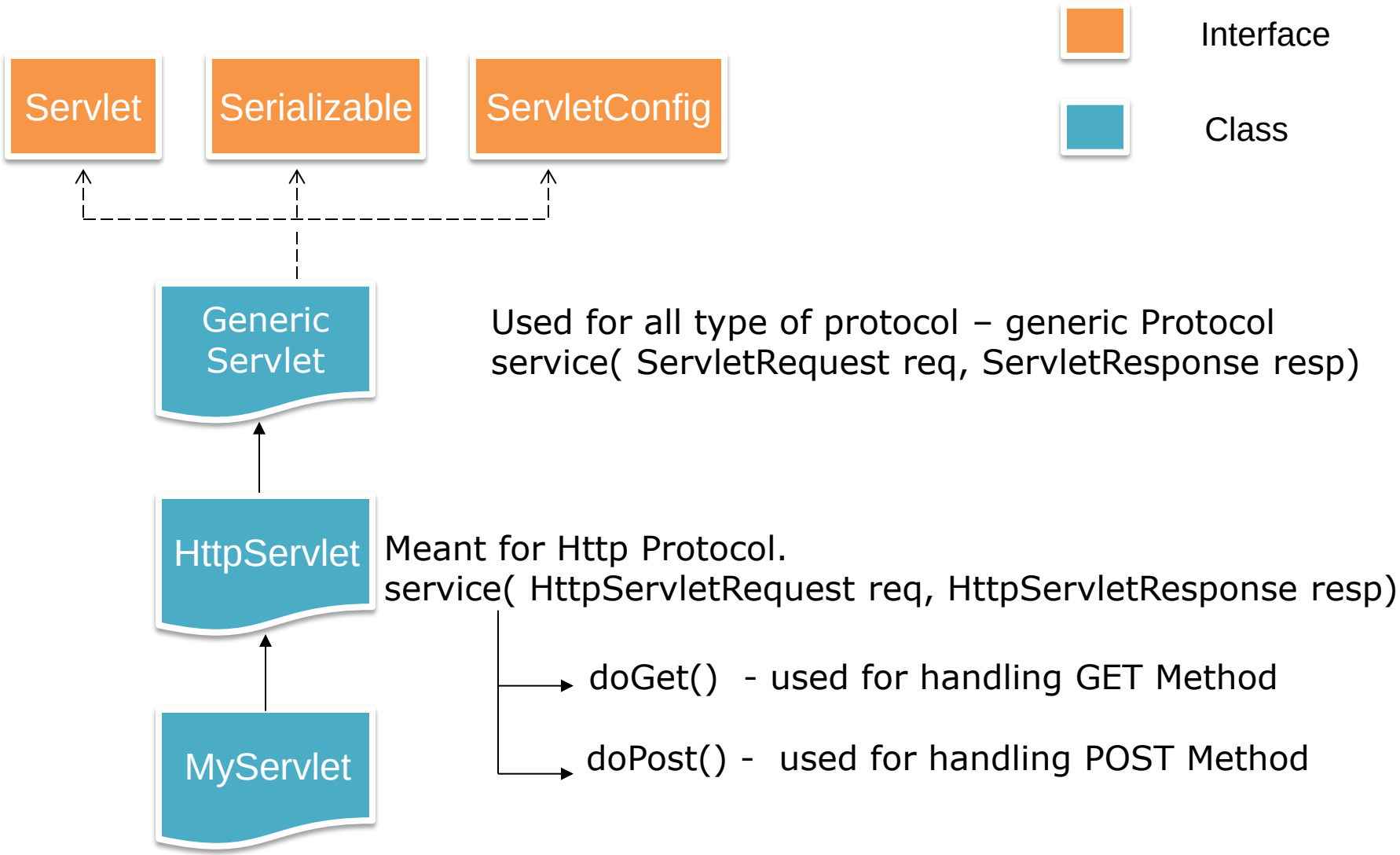
J2EE API

- Servlet
  - Must be implemented by all servlets
- ServletConfig
  - Provides servlet configuration information
- ServletContext
  - Provides information about the servlet container
- ServletRequest
  - Provides information about the client request
- ServletResponse
  - Used to send information to the client
- RequestDispatcher
  - Used to communicate with other servlets
- SingleThreadModel
  - Implemented by servlets to control concurrent access

J2EE API

- GenericServlet
  - Implements interfaces: Servlet, ServletConfig
  - Implements all functions except service()
- ServletInputStream
  - Allows reading raw bytes from client
- ServletOutputStream
  - Allows writing raw bytes to client
- ServletException
  - Is thrown when the Servlet reaches a problem which prevents it from completing a request
- UnavailableException
  - Extends ServletException
  - Used to indicate unavailability of service
  - Can be temporary or permanent

HttpServlet



HTTP Request Methods

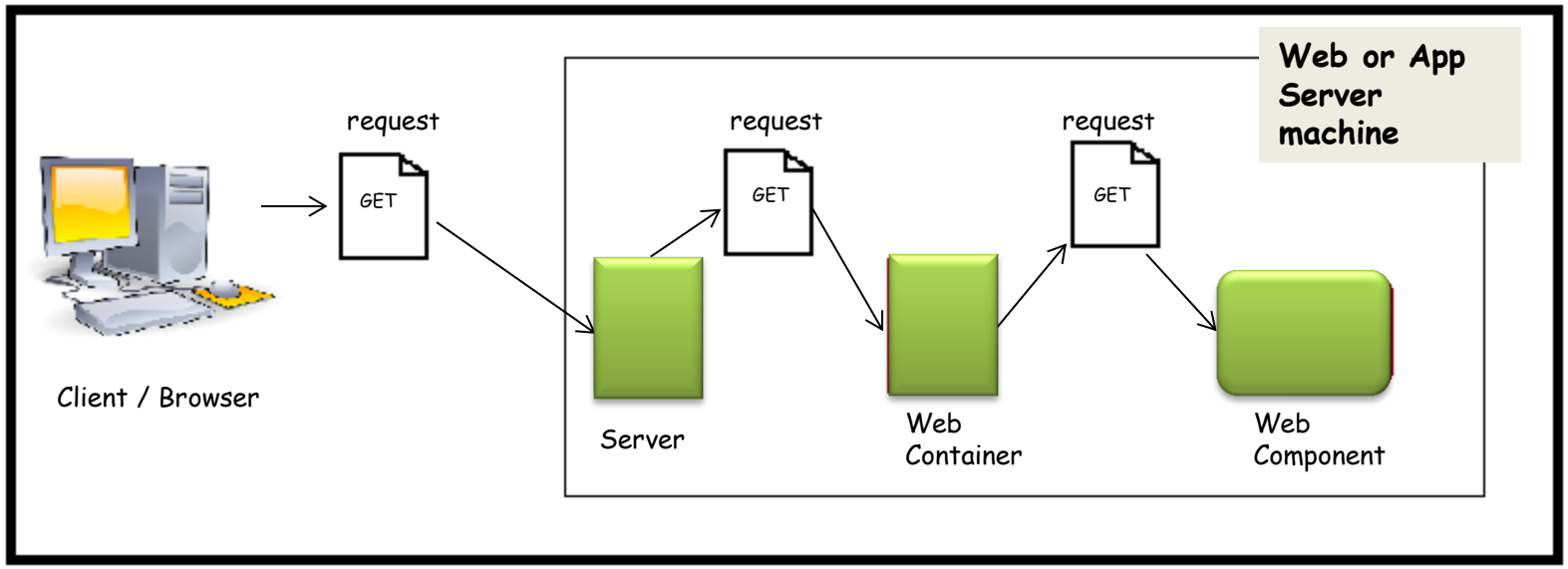
- **Get Request**
  - Is mainly used to get the data from the server, even though it can be used to send the data also to the server.
  - The data that is sent gets appended to the URL as a query string
  - The data gets exposed in the browser, so security is an issue here
  - There is a limitation to the size of the data that can be sent using this method
- **Post Request**
  - Is mainly used to post the data to the server, even though it can also be used to get the data from the server
  - The data is passed in the HTTP body
  - The data does not get exposed in the browser
  - Since the data is placed in the HTTP body, there is no limitation to the size of the data sent

HTTP Request Methods (Cont..)

- **Best Practices** : Use POST method for secure data transfer in an application. Eg . For credentials transfer / netbanking / sending credit card details.

- Follow java standards and naming conventions while coding. It avoids ambiguities and makes it easy to maintain the application.

Getting client data in Servlet

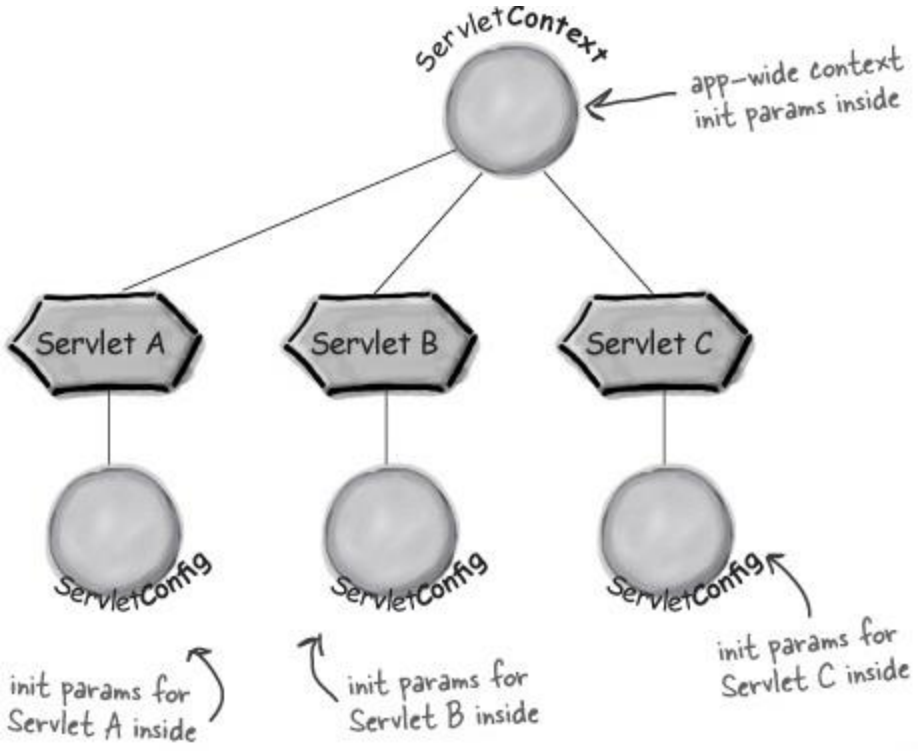


- When the control comes from front end to Server, web component Eg. Servlet handles the request.
- Client or request data is available in HTTPRequest object inside service().
- To retrieve the data following syntax can be used.

```
String data=request.getParameter("requetedInput");
```

Servlet Config and Servlet Context

- These are the two important interfaces which provide environmental support to the web application. When we want to pass any configuration information to a single servlet or multiple web components in web application we can use them.
  - ServletConfig
  - ServletContext



ServletConfig Interface

- ServletConfig allows Servlets to obtain Servlet specific configuration information like initialization parameters

```
public void init(ServletConfig config) throws ServletException
{
super.init(config);
String greeting = config.getInitParameter("greeting");
 }
```

```
web.xml
<servlet>
    <servlet-name>init</servlet-name>
    <servlet-class>InitCounter</servlet-class>
    <init-param>
      <param-name>greeting</param-name>
      <param-value>Welcome to learn Initialization Parameters</param-value>
    </init-param>
  </servlet>
```

ServletContext Interface

- ServletContext allows application specific initialization parameters and also provides the ability for application components (Servlets, JSP pages and Java beans) to interact with each other.

- There is one context per "web application" per Java Virtual Machine.

```
public void init(ServletConfig config) throws ServletException
{
super.init(config);
String greeting = config.getServletContext().getInitParameter("greeting");
 }
```

web.xml

```
<context-param>
    <param-name>greeting</param-name>
    <param-value>Welcome to learn Initialization Parameters</param-value>
</context-param>
```

Servlet Configuration using Annotations

- In Servlet 3.0 , web.xml is optional.
- Configuration part is managed by annotation called **@WebServlet** and **@WebInitParam .** These are available in javax.servlet.annotation package.
- urlPattern attribute is mandatory.
- Annotations reduces complexity of managing web.xml.
- Both annotation should be in .java file before the class starts.
- E.g.

```
@WebServlet(urlPatterns = { “/MyServlet” })
public class MyServlet extends HttpServlet { ….}
```

```
@WebServlet
(
 urlPatterns = "/imageUpload",
 initParams = {
                    @WebInitParam(name = "saveDir", value = "D:/FileUpload"),
              @WebInitParam(name = "allowedTypes", value = "jpg,jpeg,gif,png")
                   }
 )
```

Servlet Configuration using Annotations (Cont..)

Attributes

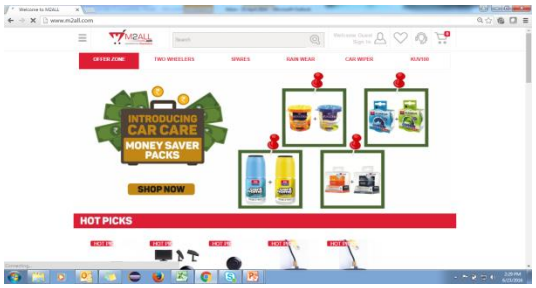
Name	Type	Required	Description
value or **urlPatterns**	*String[]*	Required	Specify one or more URL patterns of the servlet. Either of attribute can be used, but not both.
name	*String*	Optional	Name of the servlet
displayName	*String*	Optional	Display name of the servlet
description	*String*	Optional	Description of the servlet
asyncSupported	*boolean*	Optional	Specify whether the servlet supports asynchronous operation mode. Default is false.
initParams	*WebInitParam[]*	Optional	Specify one or more initialization parameters of the servlet. Each parameter is specified by `@WebInitParam` annotation type.
loadOnStartup	*int*	Optional	Specify load-on-startup order of the servlet.
smallIcon	*String*	Optional	Specify name of the small icon of the servlet.
largeIcon	*String*	Optional	Specify name of the large icon of the servlet.

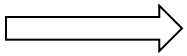
NOTE: the attributes *displayName*, *description*, *smallIcon* and *largeIcon* are primarily used by tools, IDEs or servlet containers, they do not affect operation of the servlet.

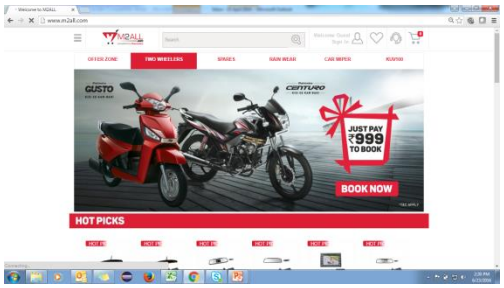
Brain Teaser..

- **Scenario 1 :** In Shopping website like M2ALL.com customer does shopping by visiting multiple pages. We have created multiple HTML pages and Servlets till now.

  How a control can go from HOME to Clothing section or any other section in same website ?







Solution - RequestDispatcher

- This is used to dispatch a request from one servlet to another (servlet / JSP/ HTML)
- RequestDispatcher can be used to transfer the control within the same application.
- Request data is also transferred along with the control. Thus it is accessible on next page.
- Syntax to create RequestDispatcher is :

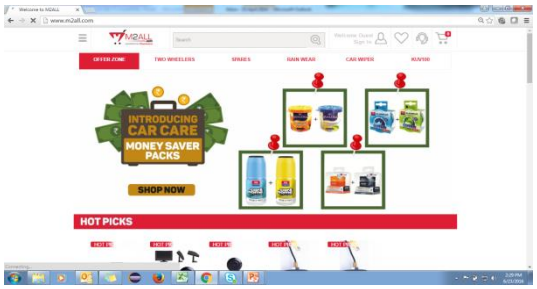
```
RequestDispatcher rd=request . getRequestDispatcher("URL");
```

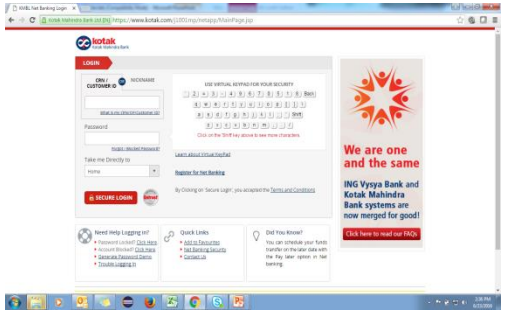
- There are two ways to transfer the control :

```
rd.forward(request,response);   //Transfers control to another servlet

            //OR

rd.include(request,response)  //Calls another servlet and embeds its output into
                                        // current servlet's output
```

Brain Teaser..

- **Section 2 :** After completing the shopping , user proceeds to pay. User chooses an option of internet banking or card payment. How a control can go from current website to another website ?





Solution - Redirection

- Redirection is nothing but redirecting the control from one location to another because of logical reasons / the website is not available / URL is changed / for load balancing between servers etc.
- Eg :

- It can transfer the control to same web application or another web application.
- Uses a new request object for new URL. Old is destroyed.
- Syntax for redirecting is

```
response.sendRedirect("URL");
```

InterServlet Communication

- Another way of Communicating between servlets is using the sendRedirect() method of the Response object

RequestDispatcher	Redirect
The servlet decides that the request should go to another part of the web application (It may be a JSP or a Servlet component).	The servlet decides that the request should go to a completely different URL. Then the servlet calls the **sendRedirect()** method of the response object and sends back the response to the browser along with the status code. Then the browser sees the status code and looks for that servlet which can now handle the request. Again the browser makes a new request, but with the name of that servlet which can now handle the request and the result will be displayed to you by the browser. The URL will have the address of the new servlet.
This is used when the request has to be sent another web component in the same web application.	This is used when the request has to be sent to another URL in different application also.
The RequestDispatch is done by the server itself.	The redirecting is done by the browser here

Brain Teaser..

- In a shopping website when customer does shopping from various sections across different pages, He gets the final products list at the end. How it can be possible?

As the pages are different and the code written in them are different, how the data is shared among different web components?

Solution - Scope objects available in servlet

- Data in web application can be shared by using different scopes. In servlet we have following scopes.
  1. request
  2. session
  3. context
- Syntax for the same is :

```
scope.setAttribute(key,value);
```

- E.g. : storing username in request object

```
request.setAttribute("username", "Shweta");
```

State Management

TLS

Rewards and Recognition

2012 ASTD BEST AWARD WINNER

Training TOP 125 Winner - 2013

Tata Institute of Social Sciences CLO Awards India Best Virtual Learning Award

State management need & introduction

- **Need for State management**
  - Http Protocol is stateless protocol. It will not maintain the client's state on server side.
  - Dynamic applications are built on the assumption that the same client access the application multiple times. But if you use HTTP client's data is lost with every request.
  - Thus Server needs Client details and performs authentications for every request.

  To avoid frequent authentication issues and give user friendly access to website we need **State management** also called as **Session Tracking.** So that client data can be maintained on the server side.

- **Definition** : *State Management is nothing but maintaining the Client state for future reference.*

Various ways

1. URL Rewriting
   - Appending state information to url.

2. Hidden Fields
   - HTML feature used for managing state.

3. Cookies
   - State information is kept in a form of text file at client side.

4. Session
   - Server side handling of state information.

5. Servlet Chaining
   - Forming chain of servlet which will carry state information across to each other.

1. URL Rewriting

- URLs can be rewritten or encoded to include session information.
- URL rewriting usually includes a session id.
- id can be sent as extra path information:
  - http://.../servlet/Rewritten/688
  - Works well if no need for extra path info.

- id can be sent as an added parameter:
  - http://.../servlet/Rewritten?sessionid=688
  - Doesn't work with POST, cause name clash.

- Id can be sent by a custom change technique:
  - http://.../servlet/Rewritten;$sessionid$688
  - May not work for all servers.

2. Hidden Form Fields

- Hidden form fields are another way to support session tracking.
- Hidden form fields do not display in the browser, but can be sent back to the server by submit.

```
<FORM ACTION="/servlet/ShowParameters" METHOD="POST">
   <INPUT TYPE="HIDDEN" NAME="OCCUPATION" VALUE="ENGINEER">
   <INPUT TYPE="HIDDEN" NAME="SESSIONID" VALUE="194043">
</FORM>
```

- Fields can have identification (session id) or just some thing to remember (occupation).
- Servlet reads the fields using req.getParameter().

3. Cookies

- Used to store client specific information on client side
- Introduced by Netscape, now a standard of HTTP protocol

- **Cookie is a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server.**

- The browser is expected to support 20 cookies for each Web server, 300 cookies total, and may limit cookie size to 4 KB each.

- Cookie is sent to client through Response object. The browser returns cookies to the servlet by adding fields to HTTP request headers.

- Cookies are of two types **Persistent** and **Non-persistent**.
  1. Persistent cookies are kept at client hard disk.
  2. Non–persistent cookies are maintained at browser memory.

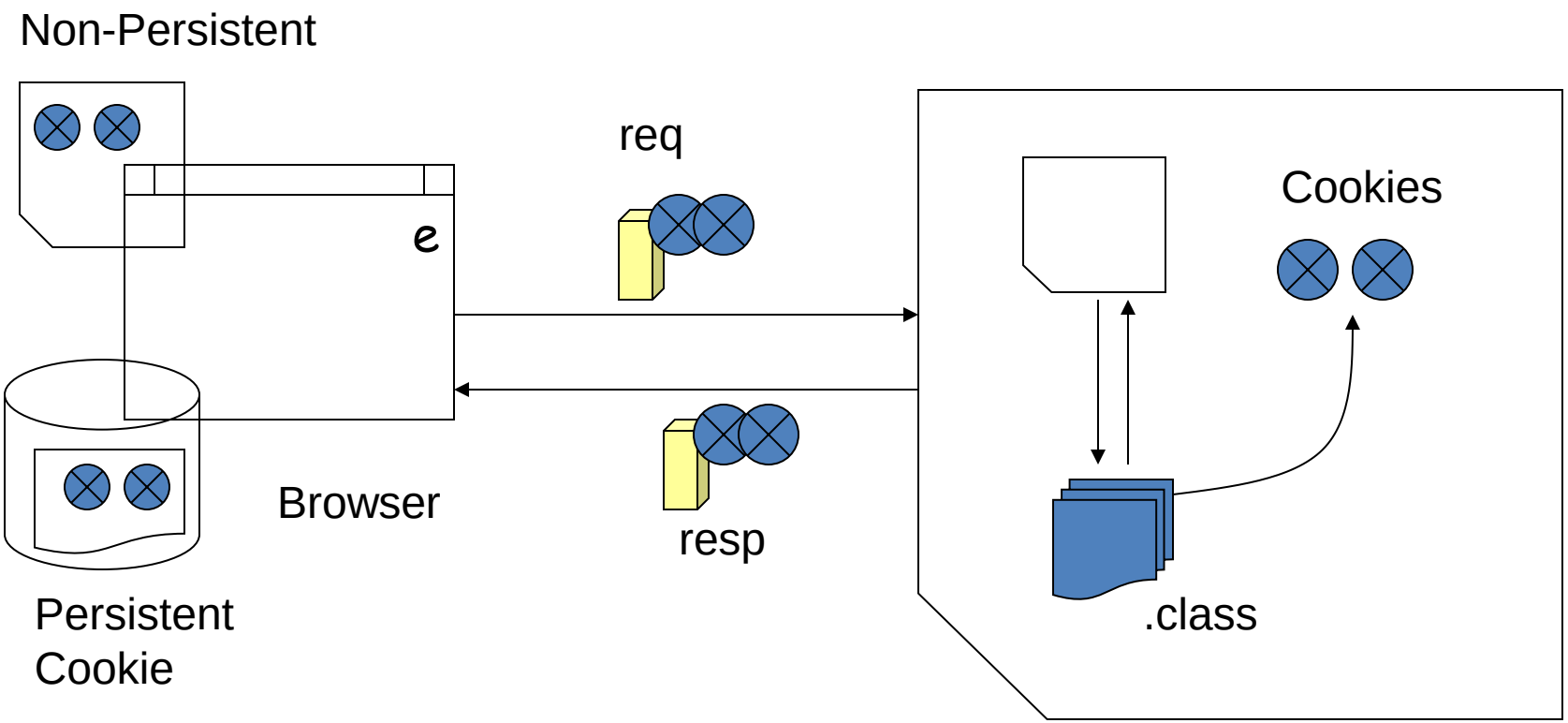
Servlet 3.0 updates

- In servlet 3.0 specification allows cookies to be marks as **HttpOnly cookies**. These cookies are not exposed to client side scripting code.

- JavaScript code written on the HTML file can not make use of cookies using **setHttpOnly(boolean isHttpOnly)** method.
- To identify whether the cookies if HttpOnly or not,use **boolean isHttpOnly()** method.

- This mechanism helps to prevent cross-side scripting attacks, meaning client side code can not use cookies to crack the server side code.

Cookie Execution



Programming Cookies

Creating a new Cookie

```
Cookie cookie = new Cookie("username", name);
```

The Cookie constructor takes a name/value String pair.

Setting how long a cookie will live on the client

```
cookie.setMaxAge(30*60);
```

setMaxAge is defined in SECONDS. This code says "stay alive on the client for 30*60 seconds" (30 minutes). Setting max age to -1 makes the cookie disappear when the browser exits. So, if you call getMaxAge() on the "JSESSIONID" cookie, what will you get back?

Sending the cookie to the client

```
response.addCookie(cookie);
```

Getting the cookie(s) from the client request

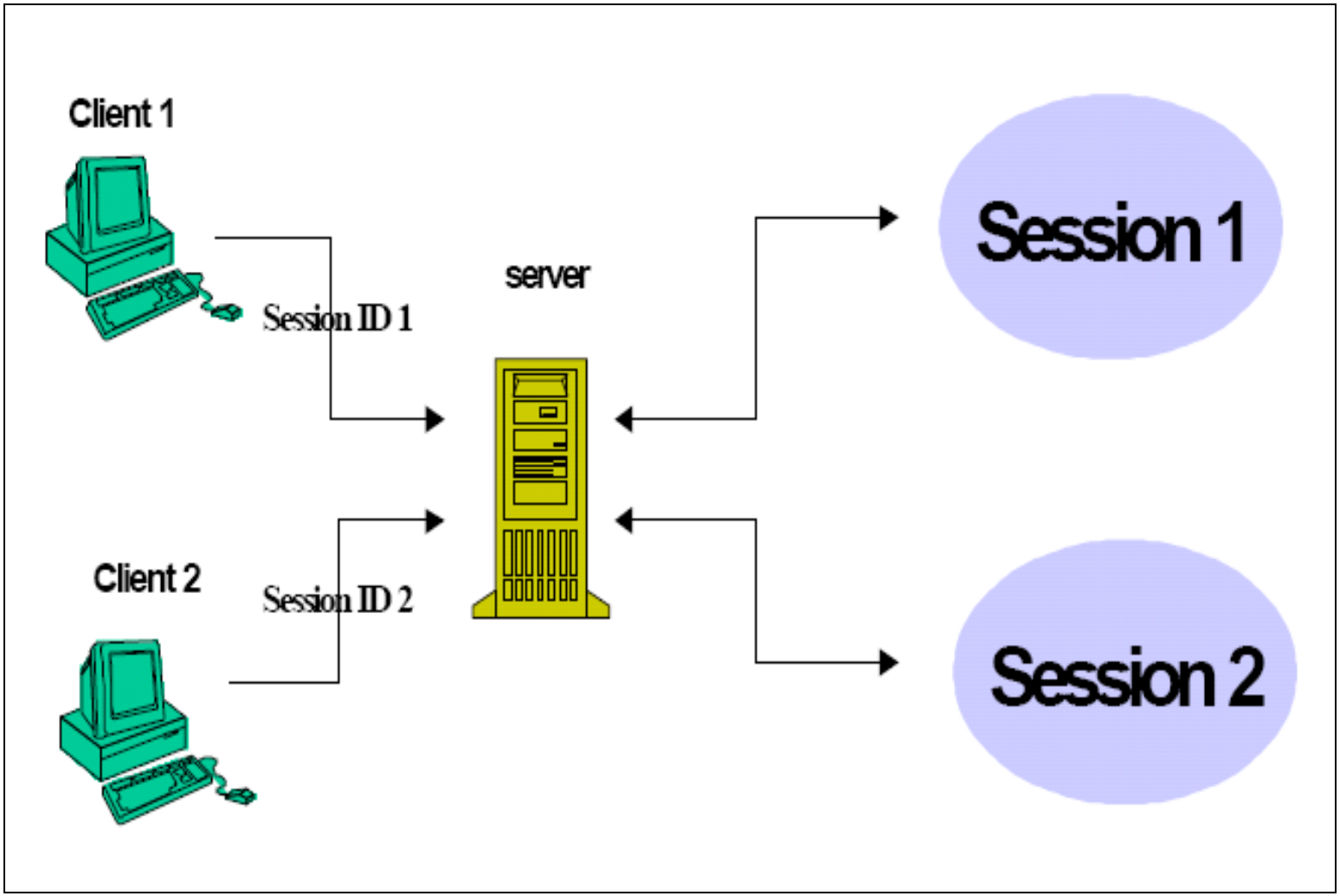
```
Cookie[] cookies = request.getCookies();
for (int i = 0; i < cookies.length; i++) {
    Cookie cookie = cookies[i];
    if (cookie.getName().equals("username")) {
        String userName = cookie.getValue();
        out.println("Hello " + userName);
        break;
    }
}
```

There's no getCookie(String) method... you can only get cookies in a Cookie array, and then you have to loop over the array to find the one you want.

4. Session Management using HTTPSession

- HttpSession interface is used to manage the sessions
- Provides a way to identify a user across more than one page request or visit a web site and to store information about that user
- Information about clients stored in server
- More secure
- Can store any Java Object

Session Execution



Session management in web application

- Provides a way to identify a user across more than one page request or visit a web site and store information about that user
- Session is an instance of HttpSession Interface managed by Servlet engine

- The session persists for a specified time period, across more than one connection or page request from the user

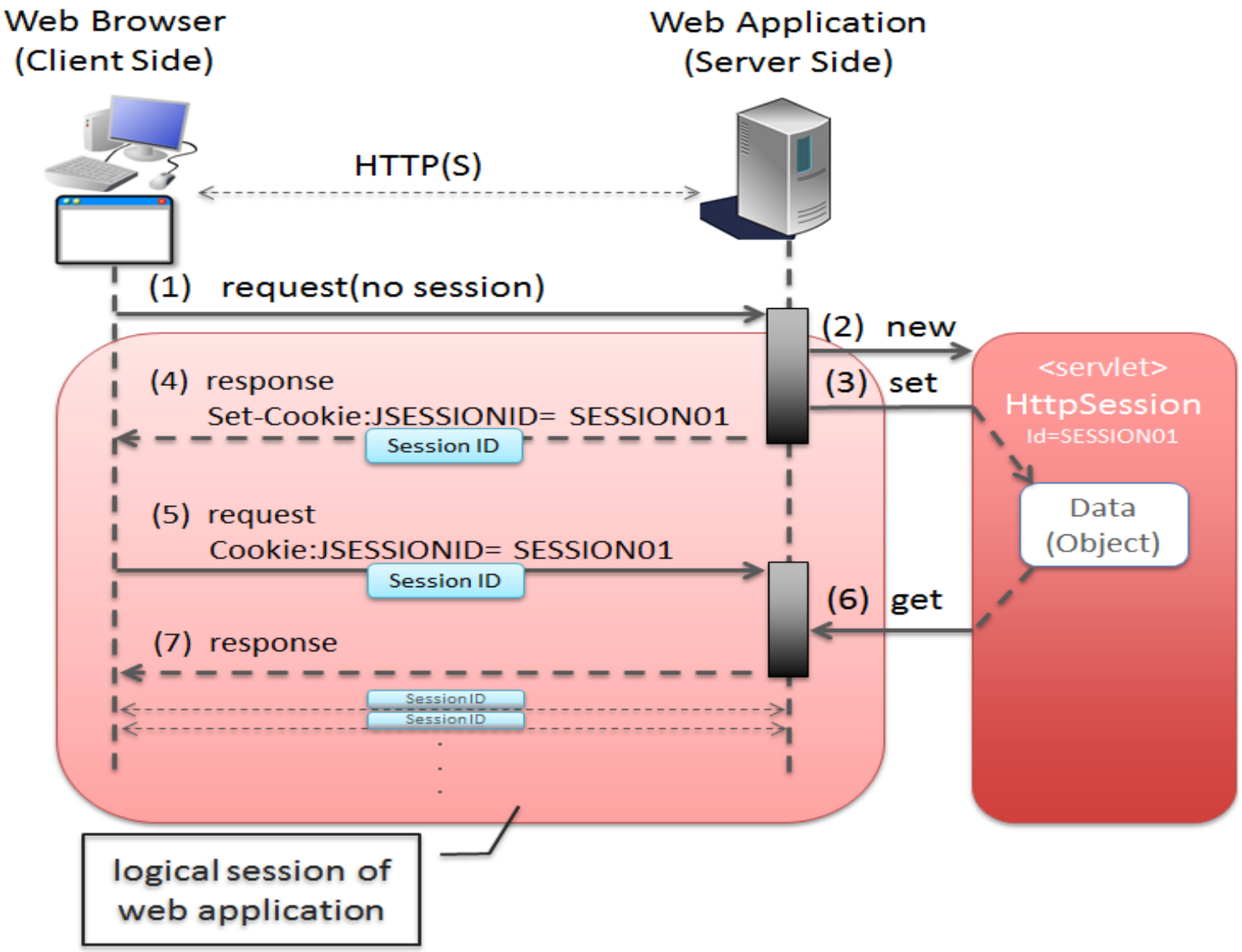
- Each session is identified by unique session id, which is sent to client using session cookie
- Session can be changed if any of these factors in current session are changed
  - User
  - Time
  - Browser

Obtaining a Session

Generates the session id if not existing and keeps its reference in the Hashtable

```
public class Servlet1 extends Servlet
{
        public void doGet(HttpServletRequest requset,
HttpServletResponse response) throws IOException,ServletException
        {
                HttpSession hs=request.getSession();
        }
}
```

Overall Session Tracking in web applications



State Management (Cont..)

- **Best Practices** : It is recommended to use combination of session tracking mechanisms based upon requirement.

  E.g. Cookies and HTTPSession as mentioned in earlier diagram.

Summary

In this session, we have covered,

- Web Component
- Servlets introduction
- Life Cycle of Servlets
- Servlet API
- Servlet Config and Servlet Context
- Servlet using annotations.
- Inter Servlet Communication
- Scopes in Servlet
- State Management and its techniques

Thank You

Disclaimer

Tech Mahindra Limited, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for informational purposes and private circulation only and do not constitute an offer to buy or sell any securities mentioned therein. They do not purport to be a complete description of the markets conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Unless specifically noted, TechM is not responsible for the content of these presentations and/or the opinions of the presenters. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.