J2EE-SERVLETS

OUTLINE

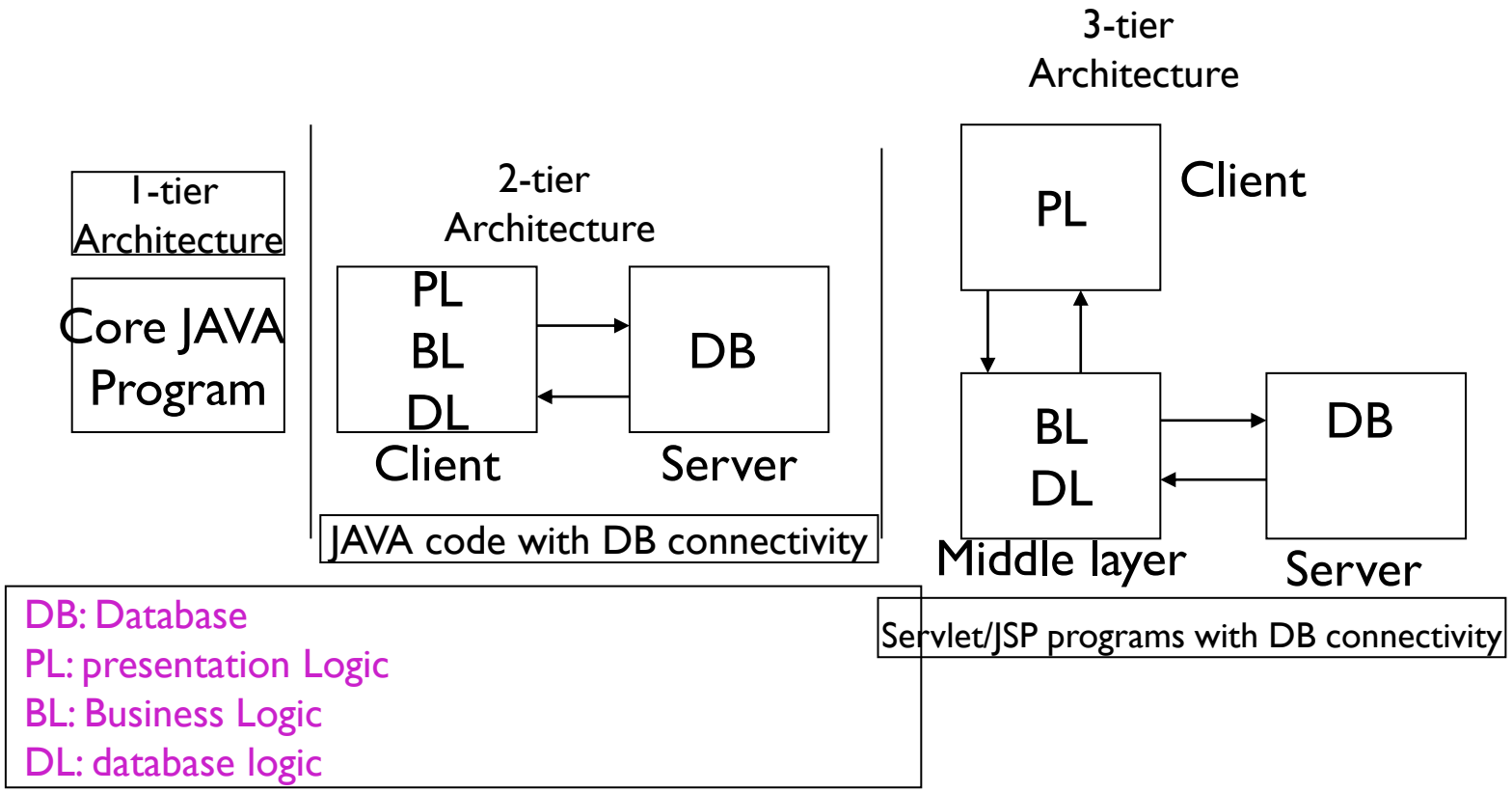
- Overview of JEE Architecture
- Web Based Communication using HTTP
- Web Server
- Introduction to Servlets
- Deployment Descriptor File
- Thread Safety in Servlets
- Retrieving Parameter Data
- Servlet Chaining
- ServletContext and ServletConfig
- Session Management
- Servlet Filters
- Servlet LifeCycle Events
- Asynchronous Servlets

OVERVIEW OF J2EE ARCHITECTURE

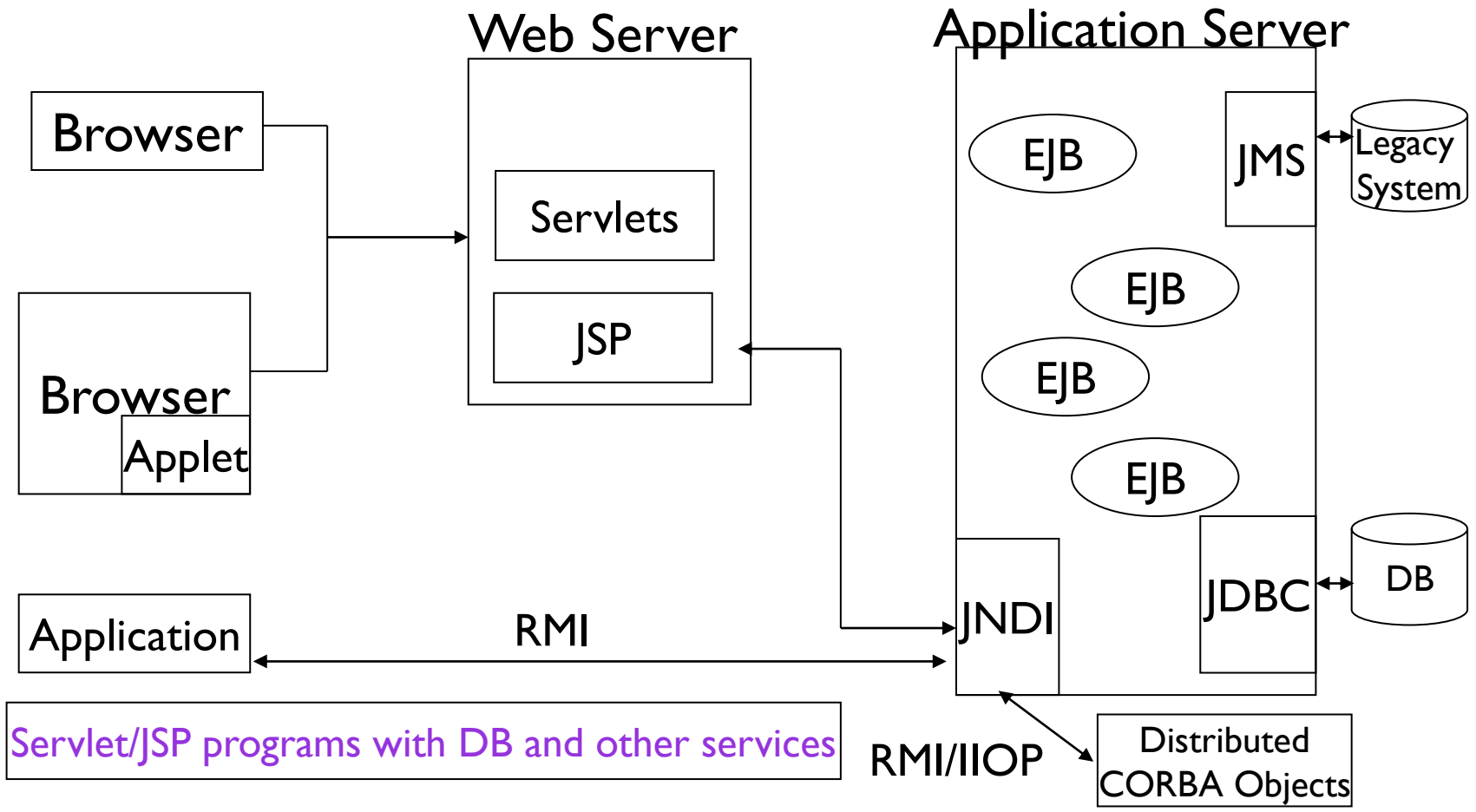
J2EE TIERED ARCHITECTURE

- 1 tier (No Server required),
- 2 tier (Client and DB Server)
- 3 tier (Client, Middle ware, DB Server)
- N tier

JQUERY IMPORTANT FEATURES



N TIER ARCHITECTURE



N TIER ARCHITECTURE CONT…

- EJB: Enterprises Java Beans
- JMS: Java Messaging Service
- JNDI: Java Naming and Directory Interface
- JDBC: Java DataBase Connectivity
- RMI: Remote Method Invocation
- IIOP: Internet Inter ORB Protocol
- CORBA: Common Object Request Broker Architecture
- JSP: JAVA Server Pages

Web Based Communication Using HTTP

HOW THE INTERNET WORKS

- Internet uses Client/Server technology for its working
- The world wide web uses the Browser as the client software and Web Server as the server software
- The user types the required URL in the browser
- The IP Address of the Server is found from the URL
- The Web Server will be listening in that Server at Port No 80
- The browser connects to Port No 80 of the specified Server
- Based on the request, the Web Server will deliver the appropriate page to the browser

HTTP MESSAGES

- HTTP is a request-response protocol. Given below are the requests
  - GET
    - To ask the server to get a resource and send it back e.g. clicking on a hyperlink gets a new html page
  - POST
    - Request something from the server and at the same time send form data to the server

-> GET and POST are the most frequently used messages

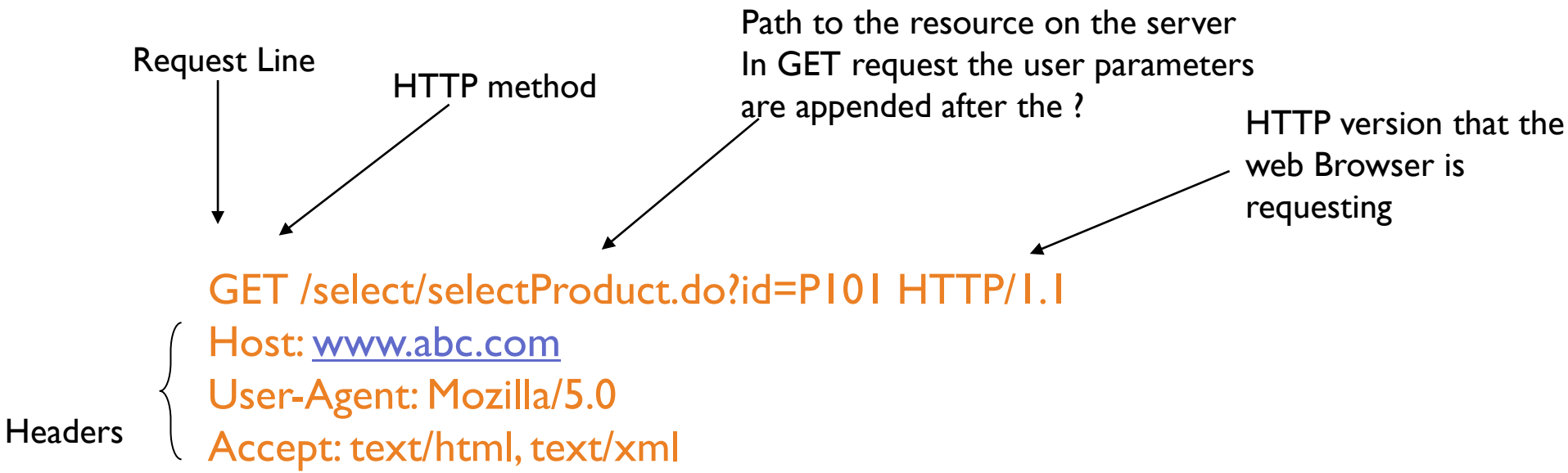
GET VS. POST

- Parameters can also be sent through GET request, but:
  - Total amount of characters in a GET is limited e.g. a big search text may not fit in the GET request. In the GET request the parameters are appended at the end of the URL e.g. http://www.google.com?search="servlet tutorial"
  - As the data sent is part of the URL, it is exposed so sensitive data should not be part of the GET message
- User cannot bookmark a form submission when using POST
- GET Examples
  - User is requesting a new page via a hyperlink, user hits the next button to see the next page etc.
- GET and POST are the most frequently used messages

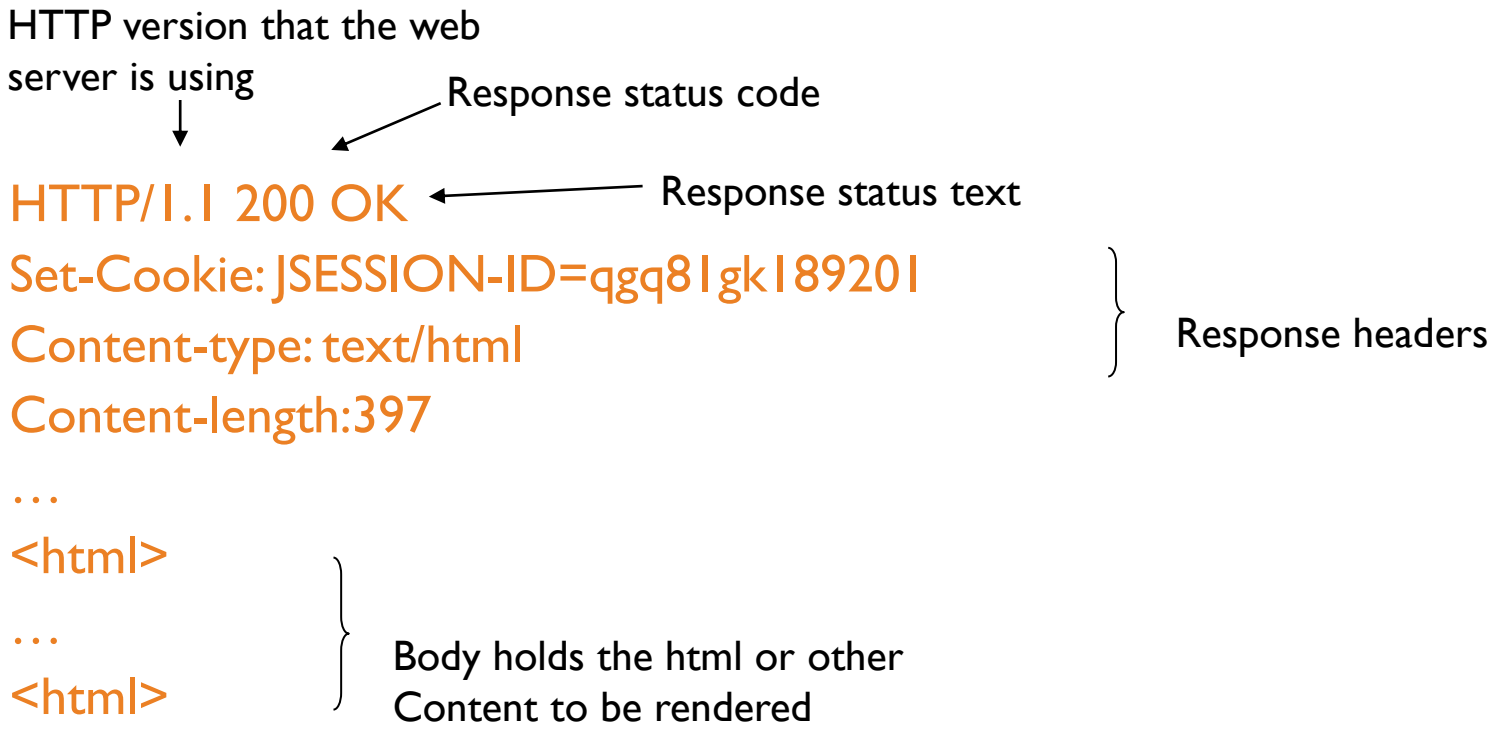
HTTP MESSAGES

- HEAD
  - Only the headers of the response comes back. An http response contains headers and a body. In case of HEAD request, the response does not carry the body part. Often used for testing hypertext links for validity, accessibility, and recent modification
- TRACE
  - Loopback of the request message, so that the client can see what's being received on the other end, for testing or troubleshooting
- PUT
  - Puts the body information at the requested URL
- DELETE
  - Delete a resource at the requested URL
- OPTIONS
  - Asks for a list of HTTP methods to which the application at the requested URL can respond
- CONNECT
  - Connect for the purposes of tunnelling

HTTP REQUEST



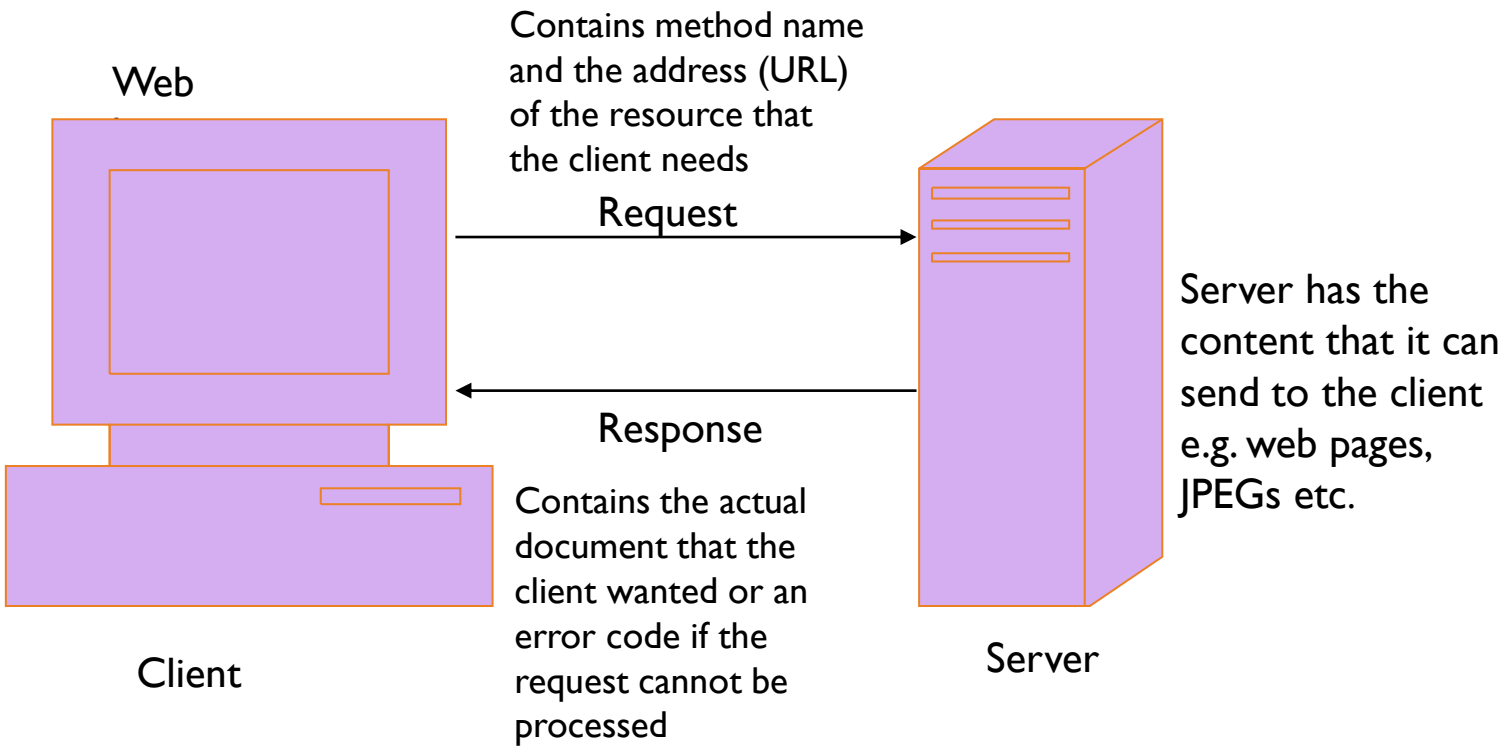
HTTP RESPONSE



HTTP RESPONSE

- The first line of the Response is called the status line
- The status line should contain the HTTP version, Status Code and a Status Description
- The following are some examples of status code used by HTTP
  - 200 - Ok
  - 400 – Bad Request
  - 401 – Unauthorized
  - 403 – Forbidden
  - 404 – Not Found
  - 500 – Internal Server Error

WEB SERVER


Web
Contains method name and the address (URL) of the resource that the client needs
Request
Response
Contains the actual document that the client wanted or an error code if the request cannot be processed
Server has the content that it can send to the client e.g. web pages, JPEGs etc.
Client
Server

WEB SERVER

- Functions of a Web Server:
  - Read any data sent by the user.
  - Look up any other information about the request that is embedded in the HTTP request.
  - Process the request
  - Generate the result.
  - Format the result inside a document.
  - Set the appropriate HTTP response parameters.
  - Send the document back to the client as HTML file

CAPABILITIES OF A WEB SERVER

- Alone, a web server cannot do the following:
  - Return a dynamic web page in the response

Static web page:

```
<html>
<body>
The current time is always 4:20 PM on the server
</body>
</html>
```

Dynamic web page:

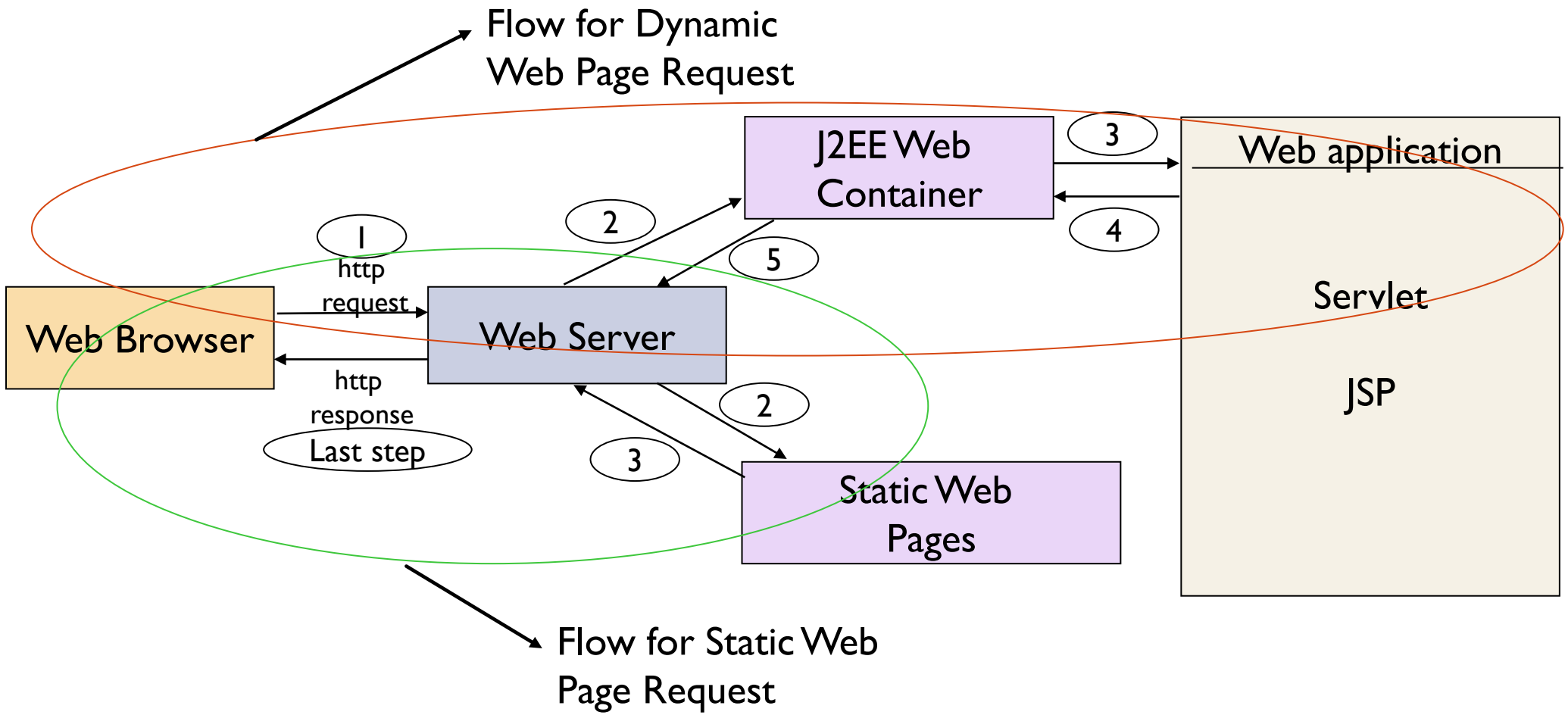
```
<html>
<body>
The current time is [insertTimeOnServer]
</body>
</html>
```

CAPABILITIES OF A WEB SERVER

- Thus a helper application is required for the web server to process user queries and return back dynamic pages

- The client sends the request to the web server and is unaware how the request is being processed. The web server may in turn pass the request to the helper application and after getting the response from the application, send it back to the client

REQUEST AND RESPONSE PROCESSING



Here Comes Servlets

WHAT IS A SERVLET

- Servlet is a server side script technology which can process client request and give back the response.
- Servlets can receive and respond to requests from web clients(browsers) working on any client-server protocol, but Http is predominantly used.
- A web application will be made of one or more servlets

Servlets runs inside the web server and process the request and sends response

SERVLET CONTAINER

- A Servlet Container is a container for servlets which is responsible for managing the servlets.
- The main functions are load ,initialize and execute servlets to serve client requests.
- When a request is received ,the container decides what servlet to call on a configuration file.
- Servlet container comes installed with the web servers.
- Examples:
  - Tomcat
  - GlassFish
  - JBoss

CONTAINER PROVIDES

- Communications Support
- Lifecycle Management
- Multithreading support
- Declarative Security
- JSP Support

SERVLET ADVANTAGES

- Servlets run within the context of the server.
- The server creates a single instance of the Servlet, which handles all client requests
- Each request begins a new thread to servlet than a process.
  - Saves time
  - Response time increases
  - It is scalable

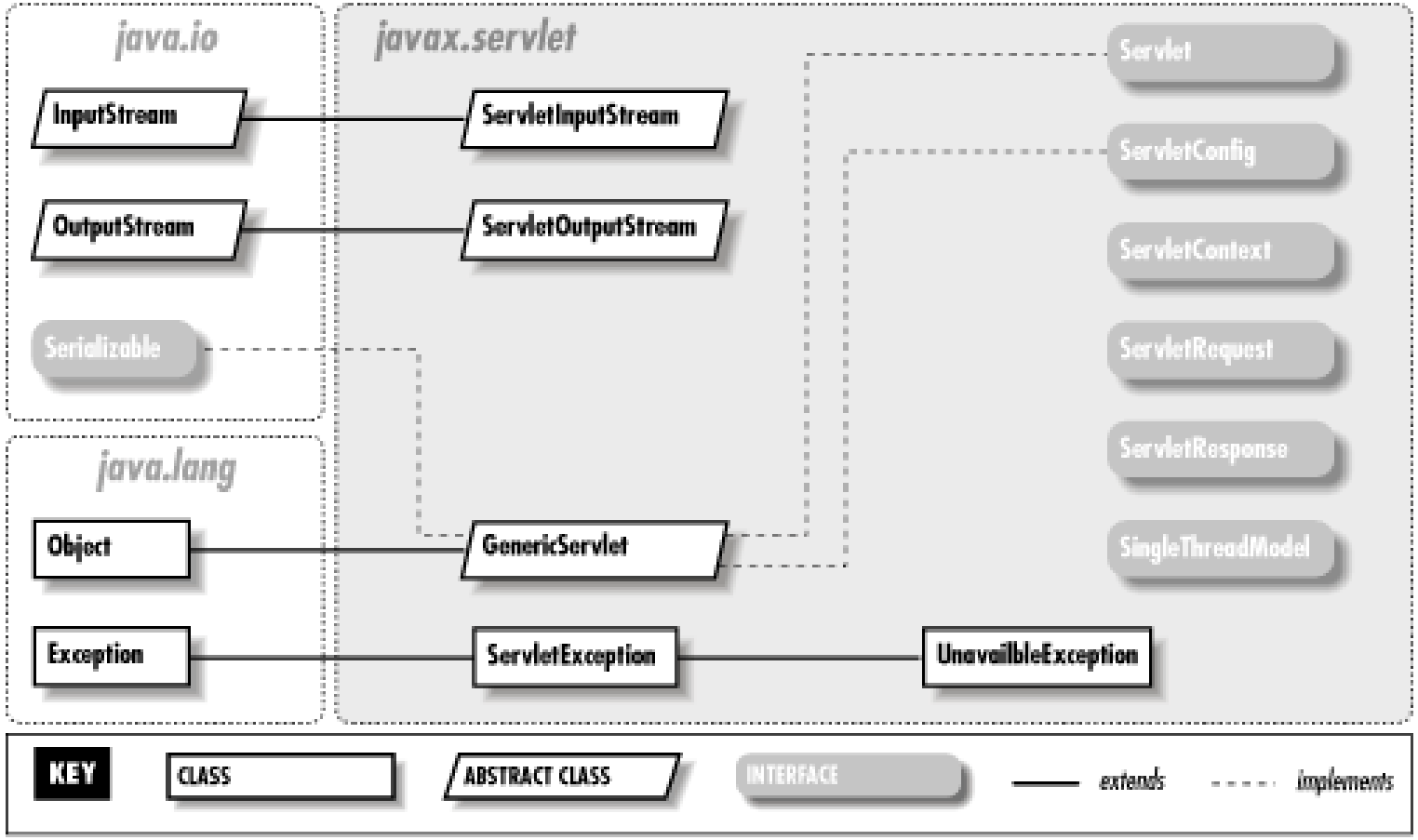
SERVLET API

- Servlet API's are available in two packages:

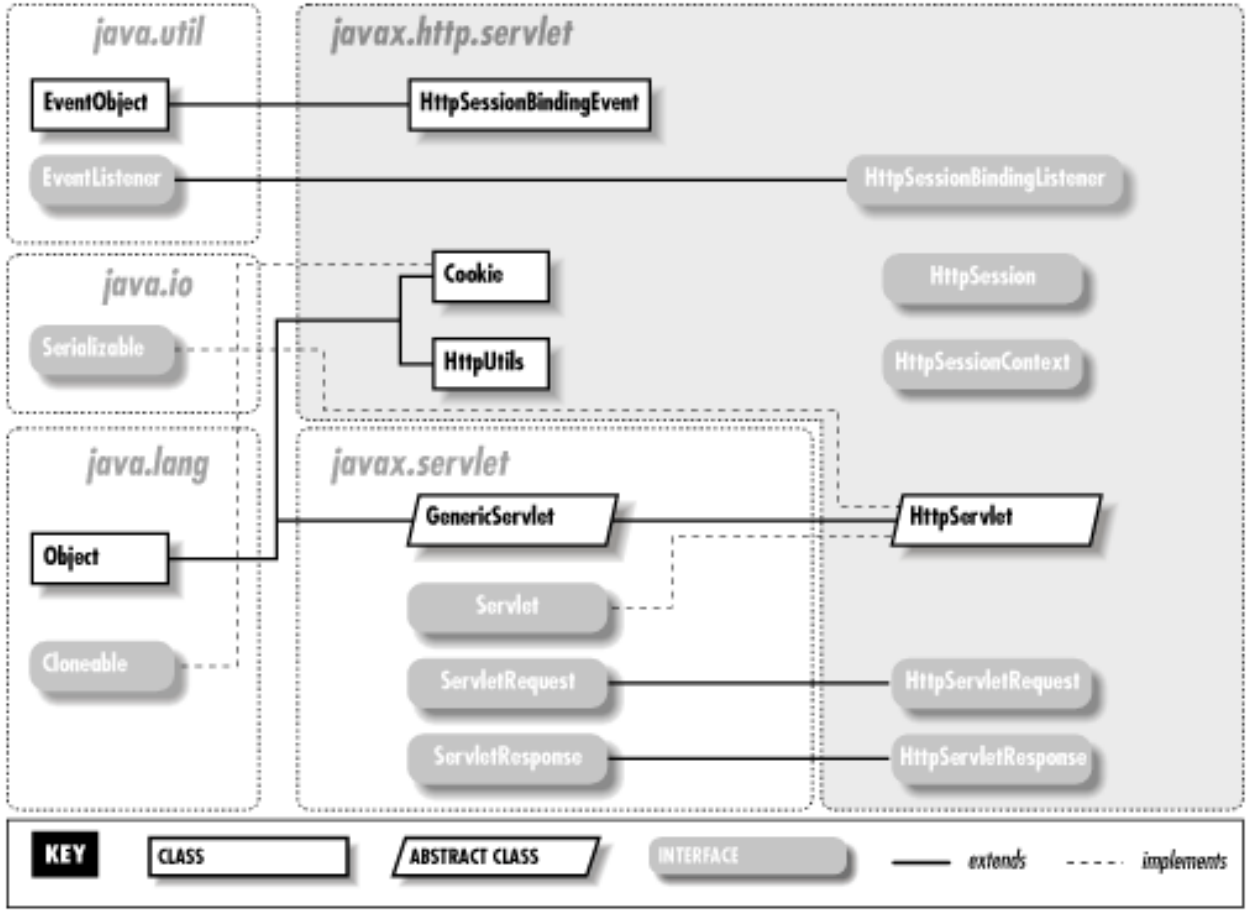
1. javax.servlet:
   - The classes and interfaces in javax.servlet are protocol independent.

2. javax.servlet.http:
   - The classes and interface in this package are specific for requests using HTTP protocol.

JAVAX.SERVLET PACKAGE



JAVAX.SERVLET.HTTP PACKAGE



SERVLET INTERFACE

- Every servlet must implement the javax.servlet.Servlet interface.

- It can be implemented in 2 ways
  - Directly implementing interface
  - Extending any of the 2 classes:
    - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet

DEPLOYMENT DESCRIPTOR

- Deployment Descriptor specifies how to configure servlets with a URL
- Mapping can be done in two ways
  - Web.xml
    - upto Servlet 2.x
  - Annotations
    - Since Servlet 3.0

WEB.XML

- Web.xml is the deployment descriptor file for a Web application.
- The deployment descriptor describes the classes, resources, and configuration of the application and how the web server uses them to serve web requests.
- Web.xml resides in application's WAR under the WEB-INF/ directory.
- When the web server receives a request for the application, it uses the deployment descriptor to map the URL of the request to the code that ought to handle the request.

ANNOTATION

- @WebServlet
  - Used at Servlet Class level
  - Example

```
@WebServlet(urlPatterns={"/first.do"})
```

THE SERVLET LIFE CYCLE

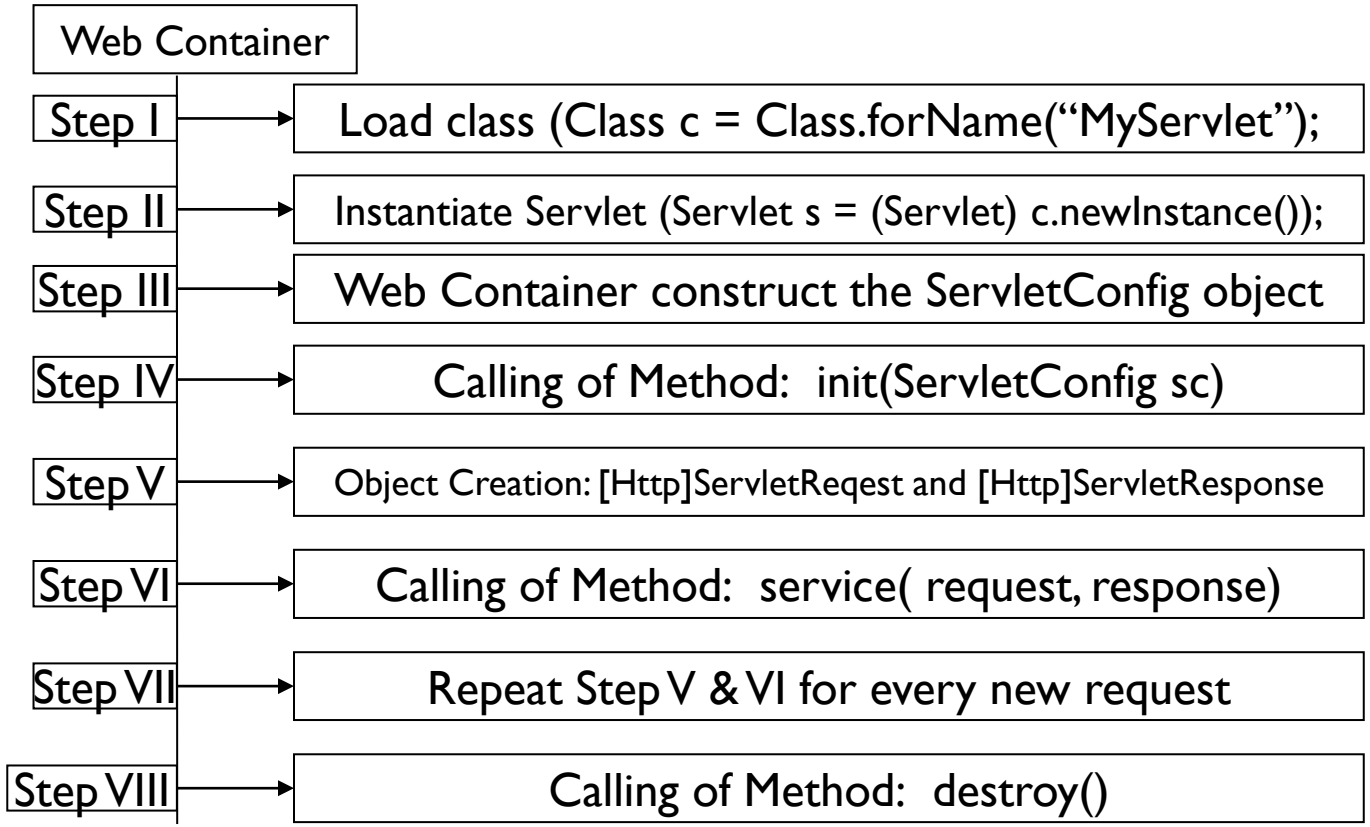
- The lifecycle of a servlet is initiated and terminated by the container

- The following methods are the lifecycle methods of a servlet:
  - init (ServletConfig sc)
  - service (ServletRequest req, ServletResponse res)
  - destroy ()

THE SERVLET LIFE CYCLE

- The following methods of HTTPServlet class are called from the service () method based on the request type of the HTTP request received from the client:
  - doGet (HttpServletRequest req, HttpServletResponse res),
  - doPost (HttpServletRequest req, HttpServletResponse res)

SERVLET EXECUTION STEPS



REQUEST AND RESPONSE PROCESSING

- User clicks on a link that has the URL of a servlet
- Container receives the request and creates two objects – HttpServletRequest and HttpServletResponse
- Container creates/allocates a thread for that request and passes the request and response objects to the servlet thread
- Container calls the servlet's service method and depending on the HTTP method, the service() method calls doGet() or doPost(). Lets assume doGet() for this scenario

REQUEST AND RESPONSE PROCESSING

- The doGet() method contains the logic to create the response page (may be with the help of a JSP) and send it back to the client
- The container deletes the request and response objects.
- Note that for the next request the same servlet object will be used by the container

LIFECYCLE OF A SERVLET

- Why do we need the init() method when we can put the initialization functionality in the constructor itself?
  - The constructor only makes an object, not a servlet. For becoming a servlet the object is given the servlet privileges like reference to the ServletConfig and ServletContext. In the init() method the object has these privileges and thus the servlet features can be utilized e.g. Creating a database connection in the init method and setting the database name as an attribute

THE SERVLET METHODS

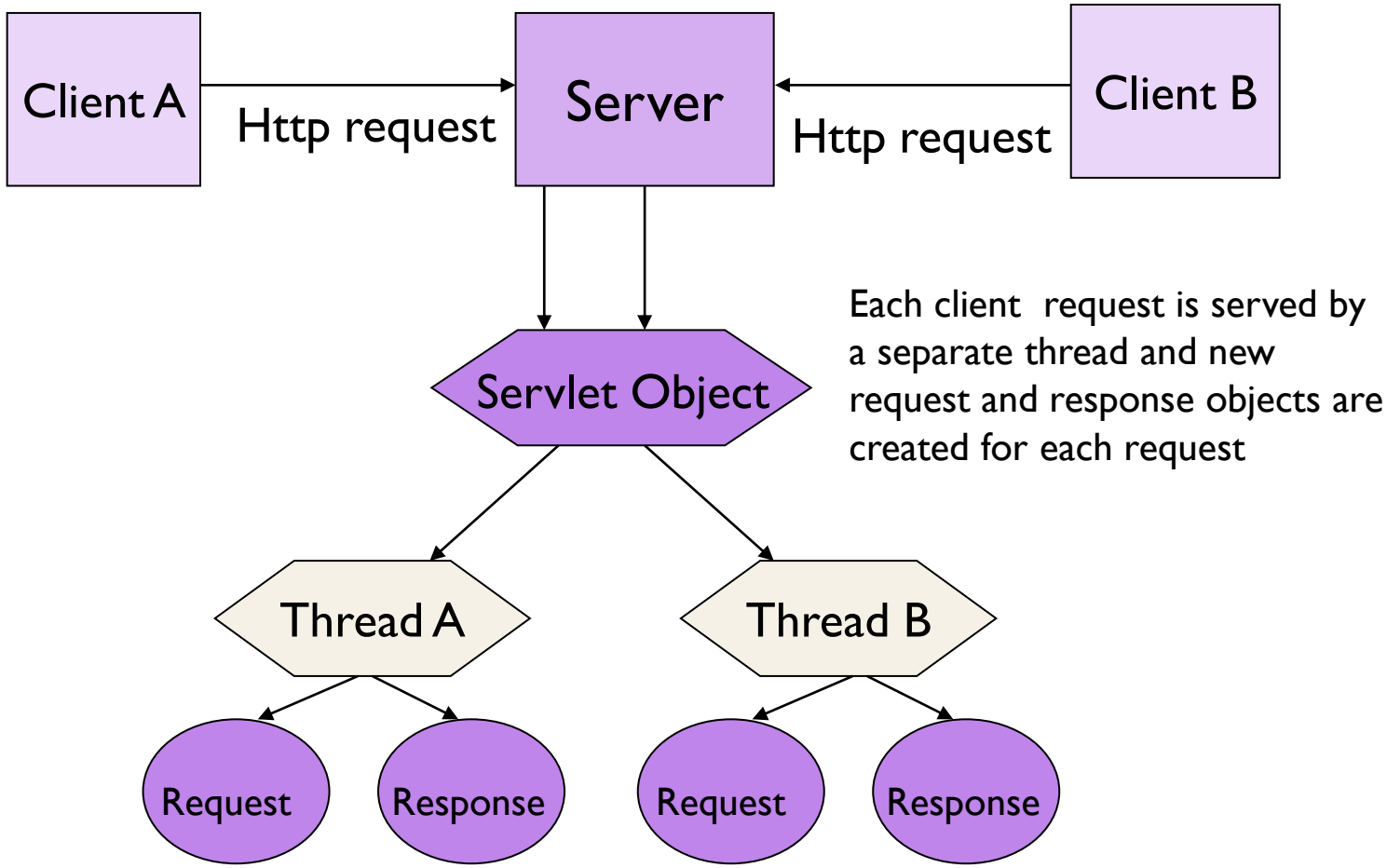
Method	What it is for	Do you override it?
init()	Gives a chance to initialize your servlet before handling client requests	Possibly e.g. establishing a database connection
service()	Looks at the request and based on the method calls doGet() or doPost()	No. Override doGet() or doPost() and let the service() implementation in HttpServlet call the right one
doGet() or doPost()	For processing the client request	Yes. At least one of them
destroy()	Gives a chance to clean up before the servlet object is destroyed	Possibly e.g. closing a database connection

THE SERVLET METHODS

- When is the destroy() method called?
  - This method is only called once all threads within the servlet's service method have exited or after a timeout period has passed.

- As a single servlet object is shared by multiple threads, does the container takes care of synchronization issues?
  - The container cannot take care of synchronization issues because where to apply the synchronization is specific to each application's logic. Thus, taking care of synchronization issues is the responsibility of the programmer

EACH REQUEST RUNS IN A SEPARATE THREAD



Thread Safety In Servlets

THREAD SAFETY

- Variables which are thread-safe:
  - Request Variables
  - Local Variables

- Variables which are not thread-safe:
  - Instance Variables
  - Session and application variables

HOW TO MAKE A SERVLET THREAD-SAFE?

- Avoidance
  - Avoid using the instance variables.
- Partial synchronization
  - If the variable must be shared between servlets, you require synchronization.
  - Synchronize the block in doGet() or doPost() method which is using the shared variables.
- Whole Synchronization
  - Implement SingleThreadModel interface. By implementing the interface, no two threads will be able to simultaneously execute the servlet's service() method.

RETRIEVING PARAMETER DATA

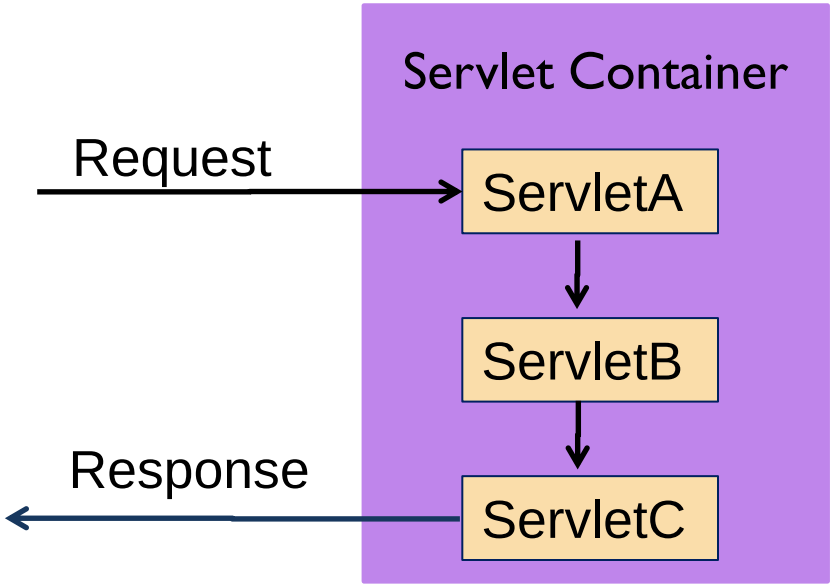
REQUEST HEADERS

ERROR HANDLING

SERVLET CHAINING

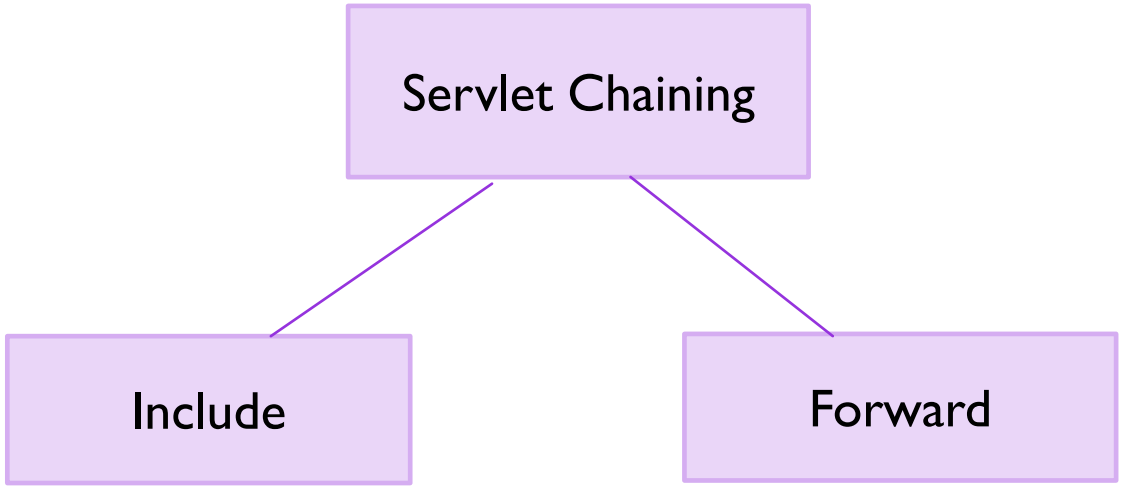
WHAT IS SERVLET CHAINING ?

- Servlet chaining is a technique in which two or more servlets take part in servicing a single request. In servlet chaining, one servlet's response is piped to the next servlet's input. This process continues until the last servlet is reached. Its output is then sent back to the client. The same request and response object are available across all the servlets in the process.



The request is processed by three servlets before giving back the response to the client

TWO WAYS OF SERVLET CHAINING



This refers to the process of including the response of one or more servlets with the response of the servlet being requested and the collated response sent to the client by the servlet being requested.

The servlet being requested by the client forwards the request to one or more servlet and finally the response is sent to the client by last servlet invoked in the chain.

REQUESTDISPATCHER INTERFACE

- Request Dispatcher is an object which accepts the request from the client and redirects them to any resource(Servlets, jsp, html etc).
- The servlet container creates the RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or identified by a particular name.
- How to create Dispatcher object ?
  - RequestDispatcher dispatcher= request.getRequestDispatcher("FooterServlet");
    - Where, FooterServlet is the servlet to which the request needs to be dispatched.

METHODS IN REQUESTDISPATCHER

- forward() : Used to forward request to another resource.
  - Syntax:
    - dispatcher.forward(request, response);
- include() : Includes the content of a resource (servlet, JSP page, HTML file) in the response.
  - Syntax:
    - dispatcher.include(request,response);

HOW TO SHARE VALUE IN SERVLET CHAIN?

- Values can be shared among the included/forwarded servlets by setting attributes, we can set values as attributes in the request object and share between the servlets.
- Request attribute is available only in the request scope which means the attribute is lost once the response is send to the user.
- Syntax
  - request.setAttribute(attributeName,value);
- Example :
  - request.setAttribute(“userName”,”Krishay”);

INCLUDE VS FORWARD

- Where to use include ?
  - Include is used for reusing common code. Like a servlet to print the user login information in all pages, the common servlet can be included in all the pages.
  - Use include if you want to present a collated response from a set of components(Servlets, jsp etc).
  - Include can also be used for including static content to a page such as the page footer can be reused across all the pages in the application.

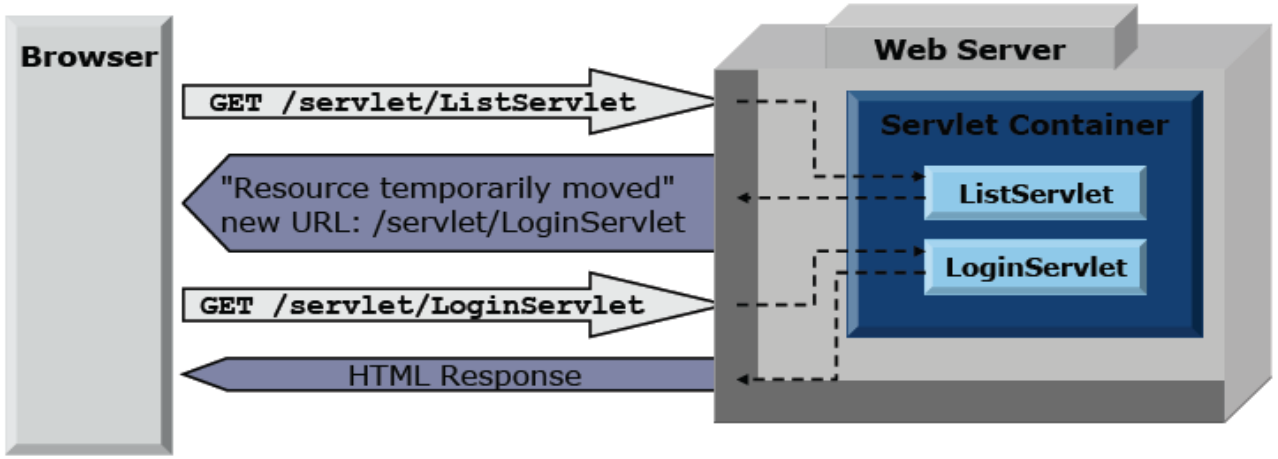
- Where to use Forward ?
  - Forward is also used again for code reusability.
  - Forward are typically used to forward a request to a success or error page after some processing. Example: After login credentials validated if success the control will be forwarded to home page else sent to error page.

GUIDE LINES IN FORWARDING REQUEST

- Forward should be called before the response has been committed to the client
- If the response already has been committed, this method throws an IllegalStateException.
- Uncommitted output in the response buffer is automatically cleared before the forward.

Difference Between SendRedirect & Forward Methods

HOW SEND REDIRECT WORKS ?

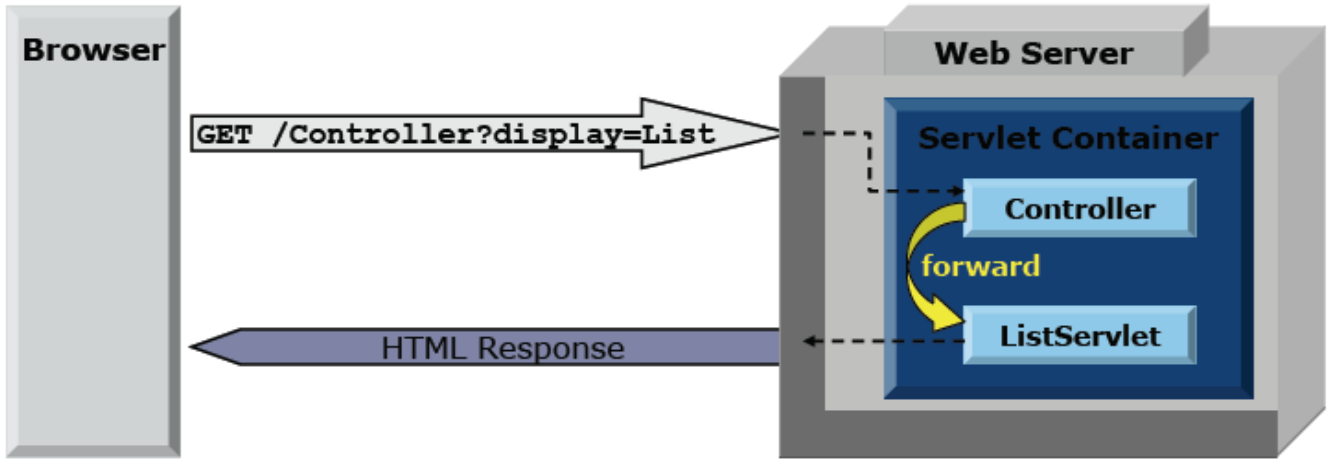


- User requests a URL to the server
- Server sends the browser a status to redirect to a new URL
- Browser locates the new URL and sends a new request.

Disadvantages:

- Requires a round trip process
- The original request parameters are not included in the new HTTP request

HOW FORWARD WORKS ?



- User request for a resource in server.
- Server redirects the request to a new resource.
- New resource sends the response to the client.

Advantages:
Internal to the servlet engine hence it is faster.
Forward is invisible to browser .
The original request received is preserved in the redirected resource.

PARAMETER DATA

- The request object provides information from the HTTP request to the servlet

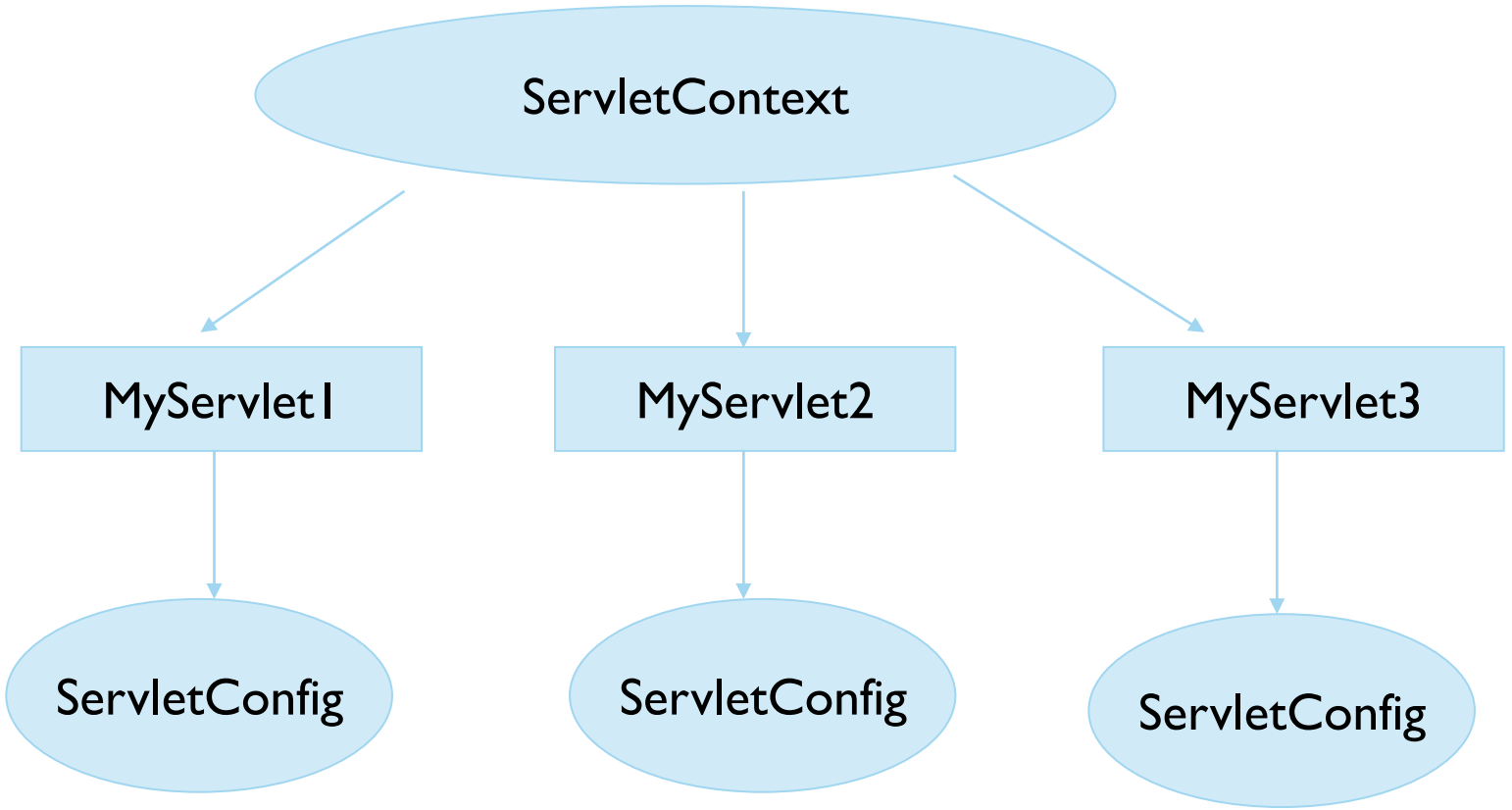
- One type of information is parameter data, which is information from the query string portion of the HTTP request.

```
http://www.example.com/servlet/PrintThis?arg=aString
```

Query string with one parameter

SERVLETCONTEXT & SERVLETCONFIG

SERVLETCONFIG AND SERVLETCONTEXT



SERVLETCONFIG INTERFACE

- Servlet Container is a object used for holding the configuration details of the servlets in a web application.

- Configuration Example:
  - Initialization Parameters, Servlet Name.

- How Servlet Config is loaded?
  - Step 1:When a request to a servlet comes the web container will initialize the servlet.
  - Step 2:The web container will parse the web.xml file and read the configurations of the servlet to be initialized.
  - Step 3:The web Container then will create a ServletConfig object and load the configuration details of the servlet in the object.

SOME MORE DETAILS ABOUT SERVLET CONFIG

- A single instance of ServletConfig is created for each Servlet.
- Example: EmployeeServlet and TaxServlet will have their own Servlet config objects.
- The ServletConfig is an interface with methods defined in the JEE specifications.
- All Web containers need to provide implementations for the specifications defined in the ServletConfig interface.
- The implementation of ServletConfig will be different for different web containers.

HOW TO USE SERVLETCONFIG?

- ServletConfig object is passed as an argument to the init() method of the servlet by the web container during servlet initialization.

- Hence to use the ServletConfig we need to either
  - override the init(ServletConfig config) method if we want to keep the config object stored as an instance variable
  - getServletConfig() method to get hold of ServletConfig object.

- During Servlet initialization, container passes the config object as an argument to the init method.

SET AND RETRIEVE INITIALIZATION PARAMETERS

- Initialization parameters are used by developers to set some parameter values for the servlet to use. This will be stored as key/value pairs.
- These parameters can be configured in web.xml or using Annotations.

Setting init parameter in web.xml

```
<init-param>
<param-name>name</param-name>
<param-value>value</param-value>
</init-param>
```

Setting init parameter through Annotations

```
@WebServlet(urlPatterns = {"/initParams"},
        initParams={ @WebInitParam(name="n1", value="v1"), @WebInitParam(name="n2", value="v2") })
```

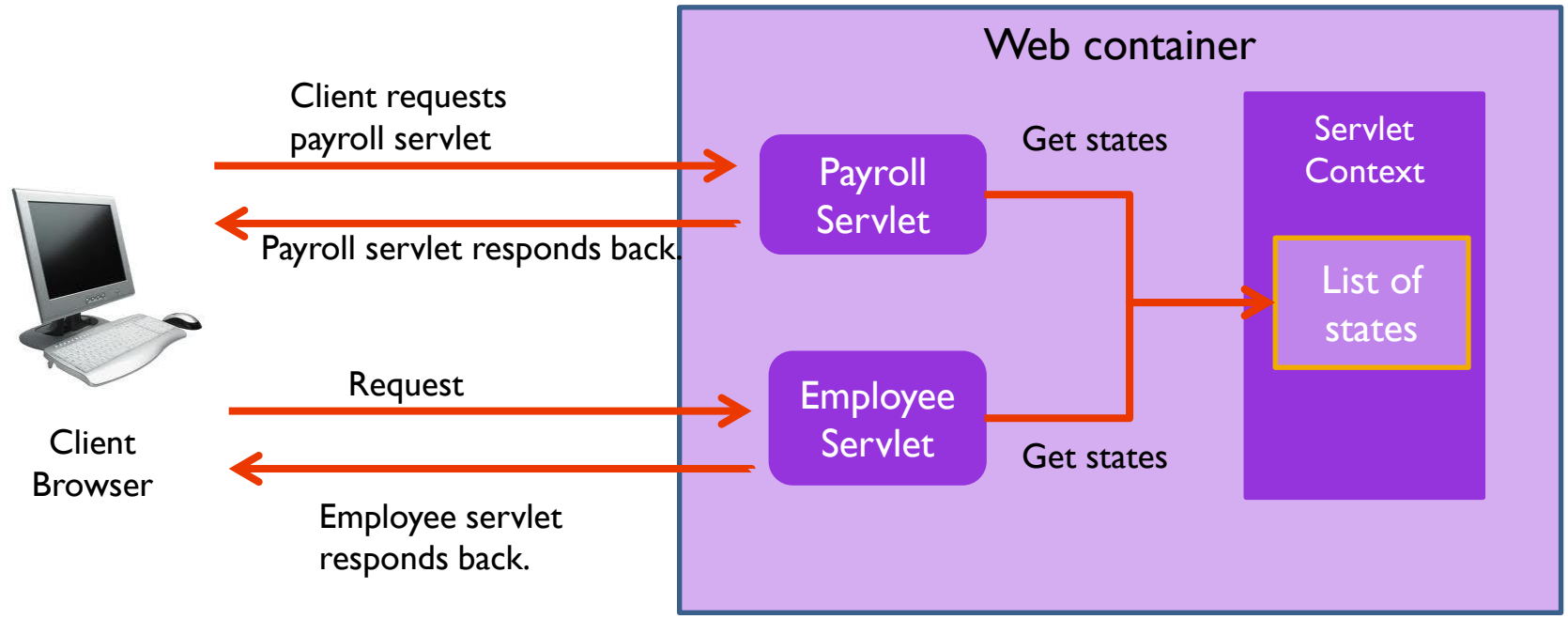
Read init parameter in servlets

- Two methods
  - String getInitParameter(paramname)
  - Enumeration getInitParameters();

WHAT IS A SERVLET CONTEXT?

- Servlet context refers to the context in which the servlet runs.
- It is a gateway between the web container and the servlets.
- There will be only one instance of servlet context per web application shared by all the servlets in the application.
- Used for storing values which can be shared across all the servlets in the web application.
- Context level Init parameters can be set which can be reused by all the servlets in the context
- A servlet can set values to the context as attributes which can be consumed by other servlets.

HOW SERVLET CONTEXT WORKS?



The servlet context handle is obtained using the ***getServletContext*** method of servlet config.

ServletContext context = config.getServletContext();

SETTING AND READING CONTEXT LEVEL PARAMETER

- Context parameters for an application can be declared by setting in the deployment descriptor.

```
<context-param>
<param-name>Title</param-name>
<param-value>My Application</param-value>
</context-param>
```

NOTE: Context parameters are declared outside all the servlet declaration since they are common to the entire application

- Reading the Context Init Parameters
  - String getInitParameter(paramname)
  - Enumeration getInitParameterNames()

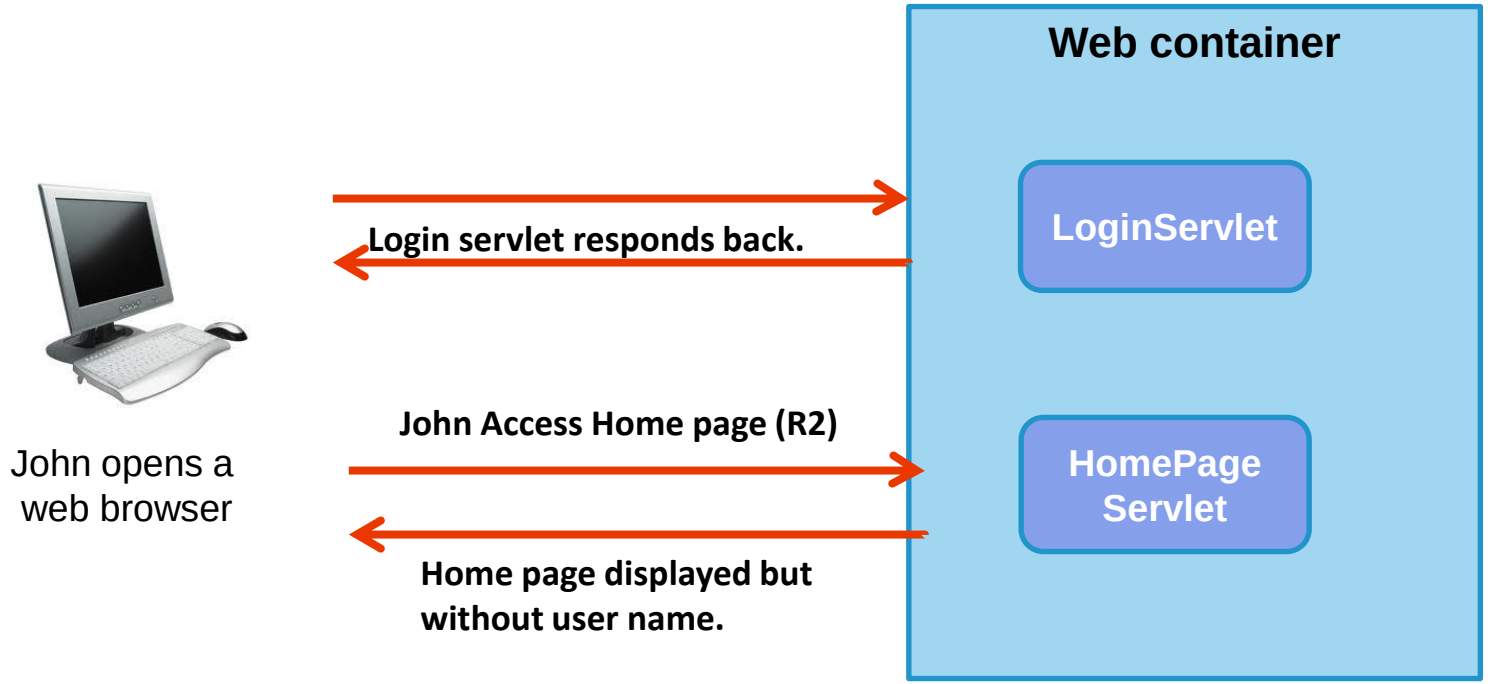
SESSION MANAGEMENT

SESSION TRACKING

- HTTP is stateless: When it gets a page request, it has *no memory* of any previous requests from the same client
  - This makes it difficult to hold a "conversation"
    - Typical example: Putting things one at a time into a shopping cart, then checking out--each page request must somehow be associated with previous requests
  - The server must be able to keep track of multiple conversations with multiple users

- Session tracking is keeping track of what has gone before in this particular conversation
  - Since HTTP is stateless, it does not do this for you
  - You have to do it yourself, in your servlets.

SERVLET WITHOUT SESSION TRACKING



The second request for Home page, the user is **NOT** passing user name nor the value from request **R1** is stored anywhere .
So home page cannot display the user name.

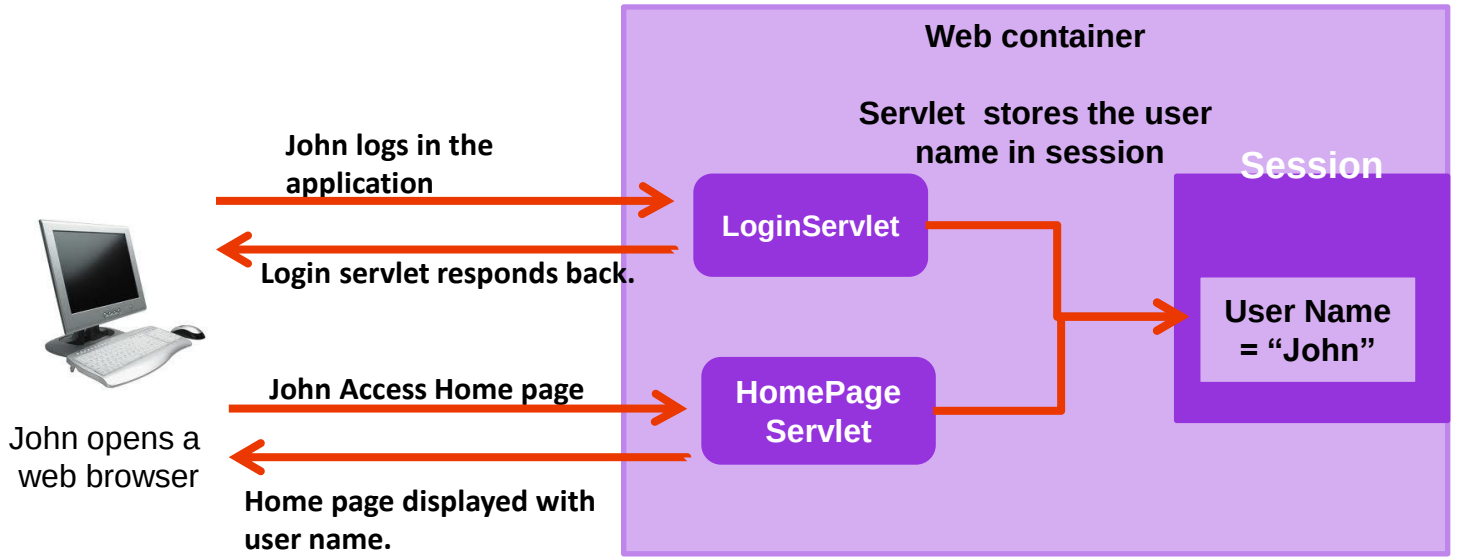
SESSION MANAGEMENT

- Session Management is a mechanism for maintaining state across multiple HTTP requests. This is managed by the Web container.

- In Simple words it is a technique to hold some values passed by user across multiple HTTP requests arising out from a single browser instance.

- Session life cycle is managed for each web browser instance opened and it exists till the browser is closed (or) till the session time outs (set in the server configurations).

SERVLET WITH SESSION TRACKING



Though the username is NOT in the request it can print the username by accessing the session

In the above example if John opens a new browser and access Home page servlet the user Name **cannot** be accessed since a new HTTP Session object is created

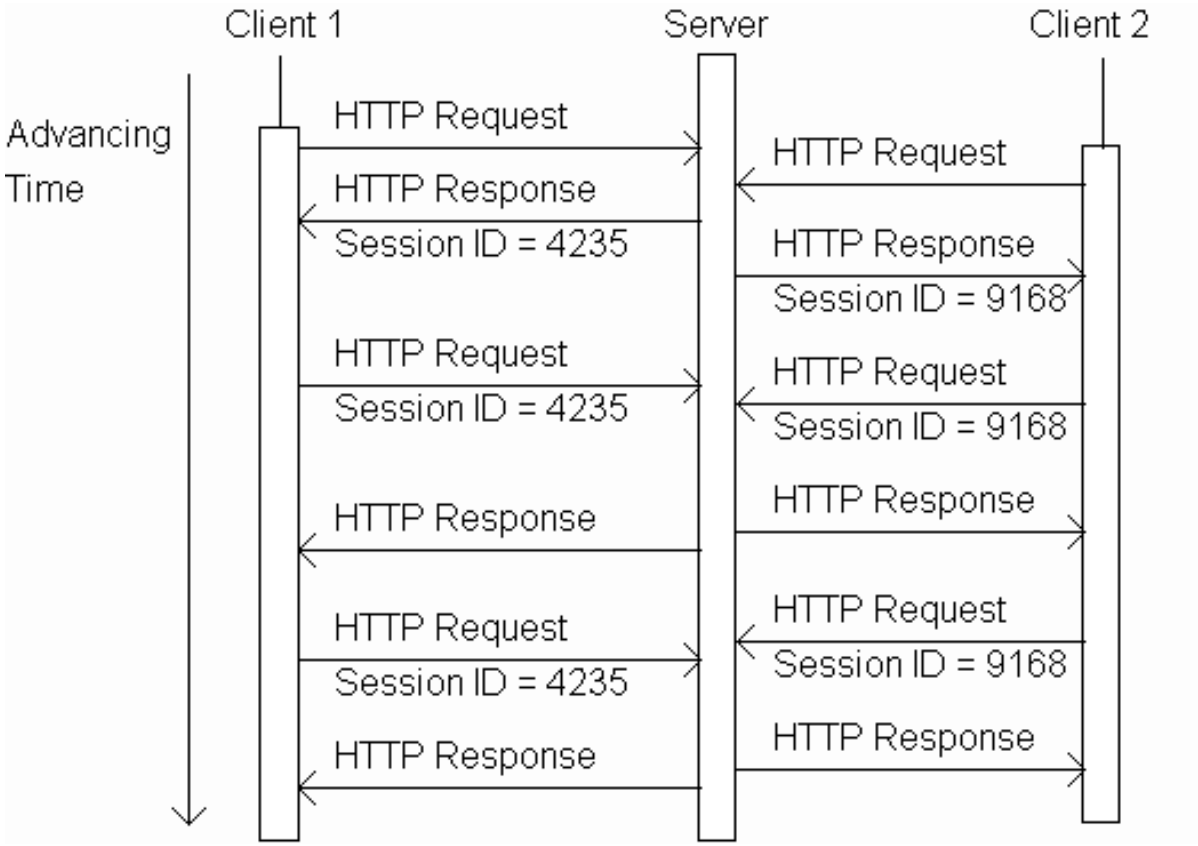
SESSION MANAGEMENT TECHNIQUES

- The following are the session management techniques. These are used for managing the data's between pages of the user in a session.
  - Client-Side
    - Hidden Fields
    - URL Rewriting
    - Cookies
    - User authentication – getRemoteUser()
      - Provided by HttpServletRequest
  - Server-Side
    - Server's built-in session tracking

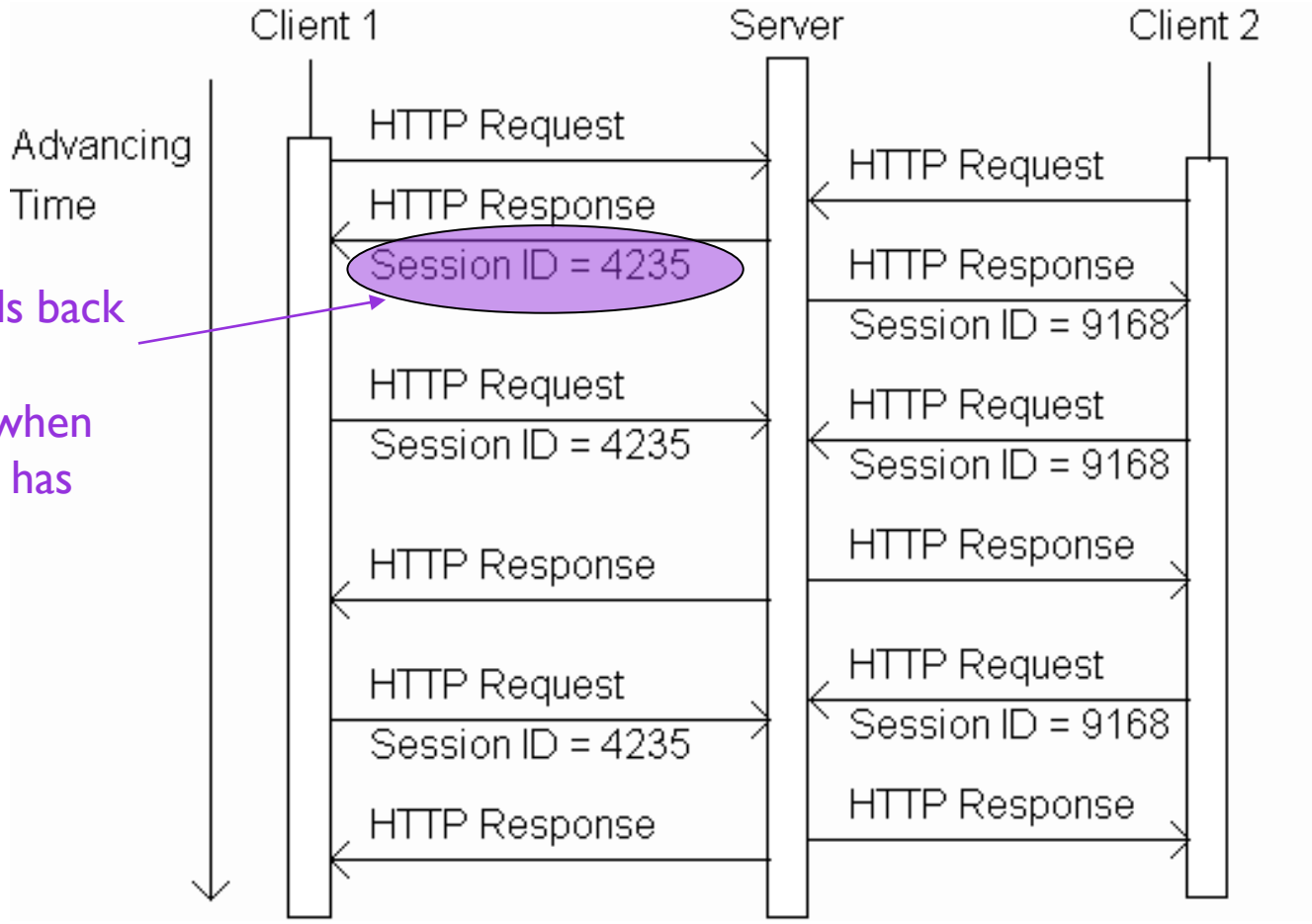
SERVER'S BUILT-IN SESSION TRACKING

- Session Tracking API
  - Devoted to servlet session tracking
- Most servers support session tracking
  - Through the use of Persistent Cookies (JSESSIONID)
  - Able to revert to URL rewriting when cookies fail (URLencode)
  - Servlet uses the tokens to fetch session state
- Session objects are maintained in memory
  - Some servers allow them to be written to file system or database as memory fills up or when server shuts down
    - Items in the session need to be serialized

SESSIONS

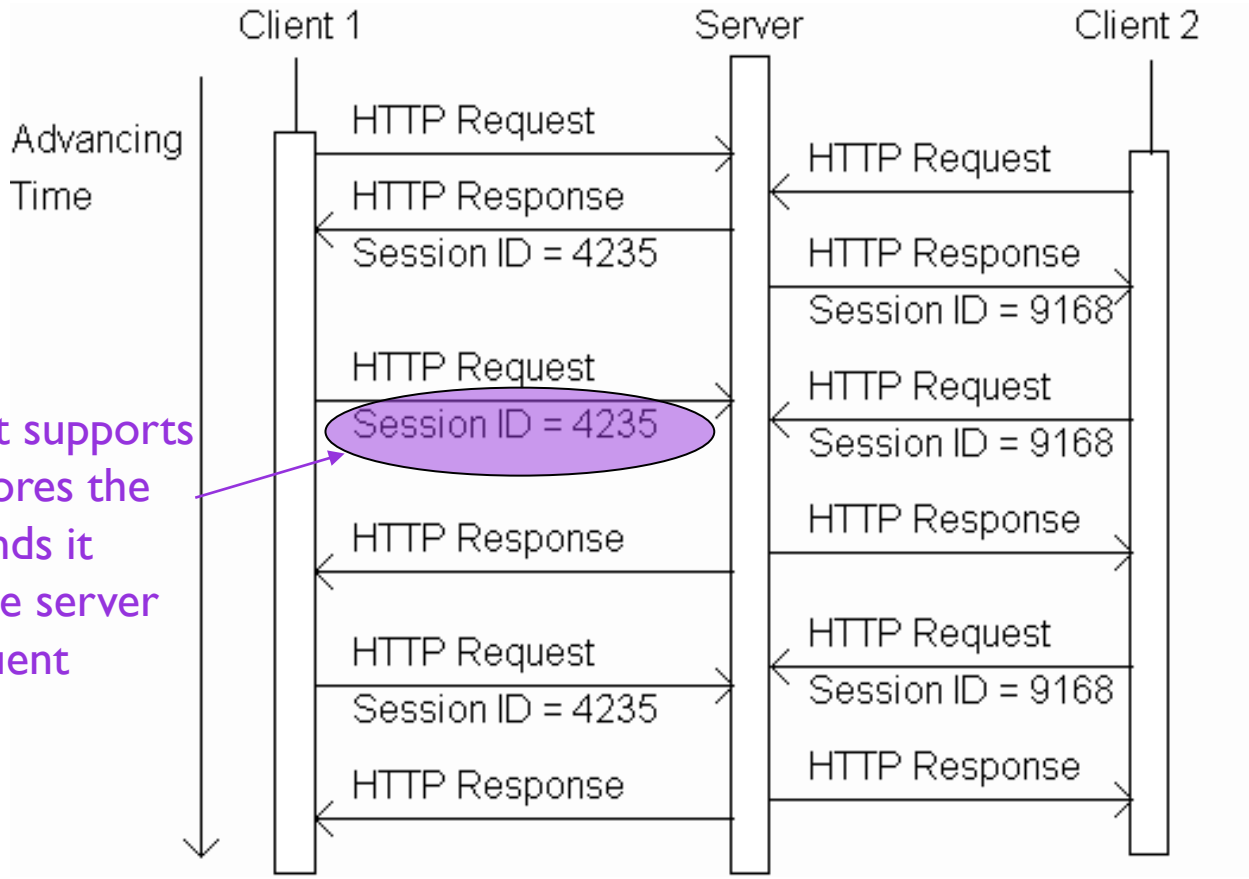

Client 1
Server
Client 2
Advancing
Time
HTTP Request
HTTP Response
Session ID = 4235
HTTP Request
HTTP Response
Session ID = 9168
HTTP Request
Session ID = 4235
HTTP Request
Session ID = 9168
HTTP Response
HTTP Response
HTTP Request
Session ID = 4235
HTTP Request
Session ID = 9168
HTTP Response
HTTP Response

SESSIONS



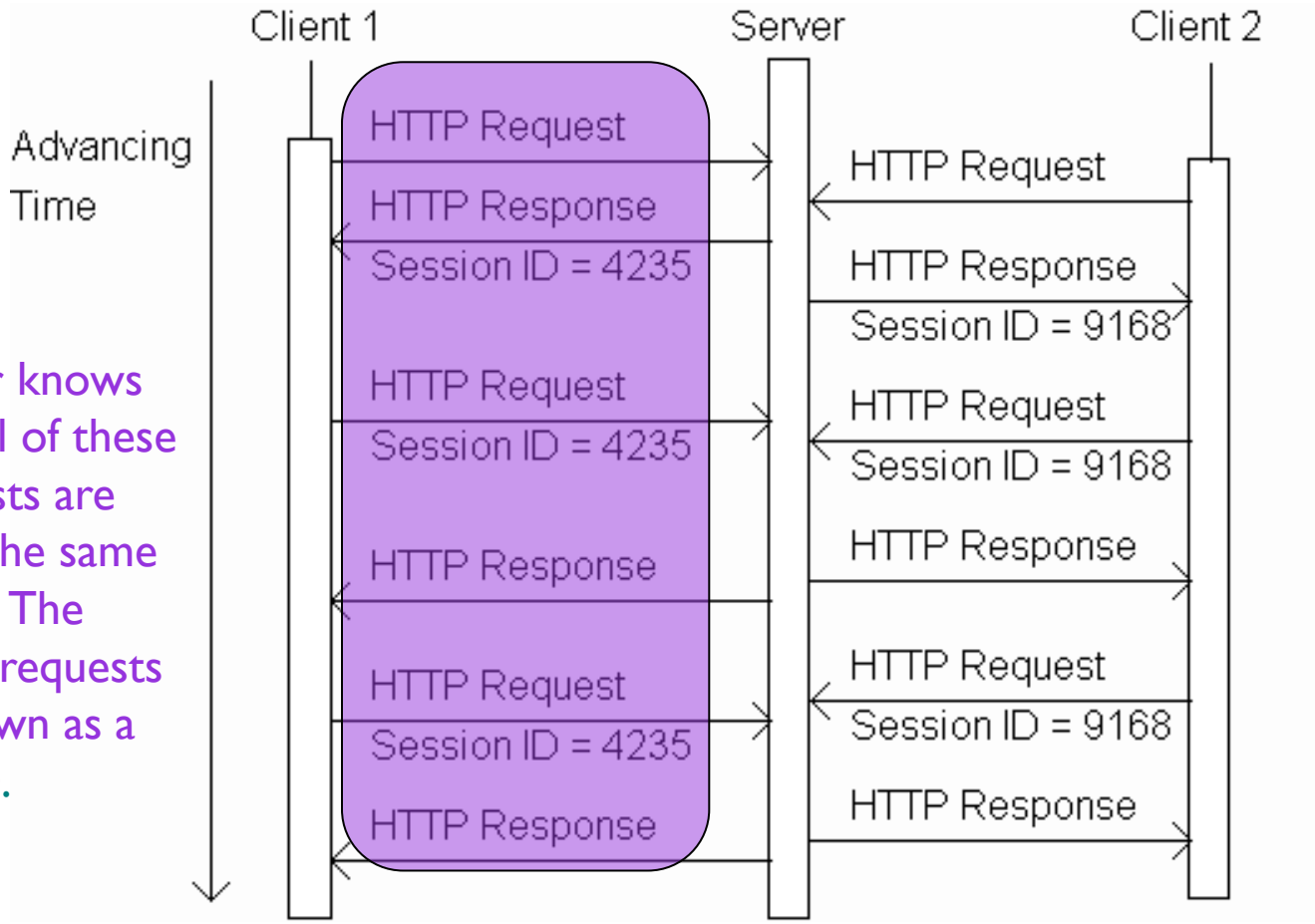
Server sends back new unique session ID when the request has none

SESSIONS



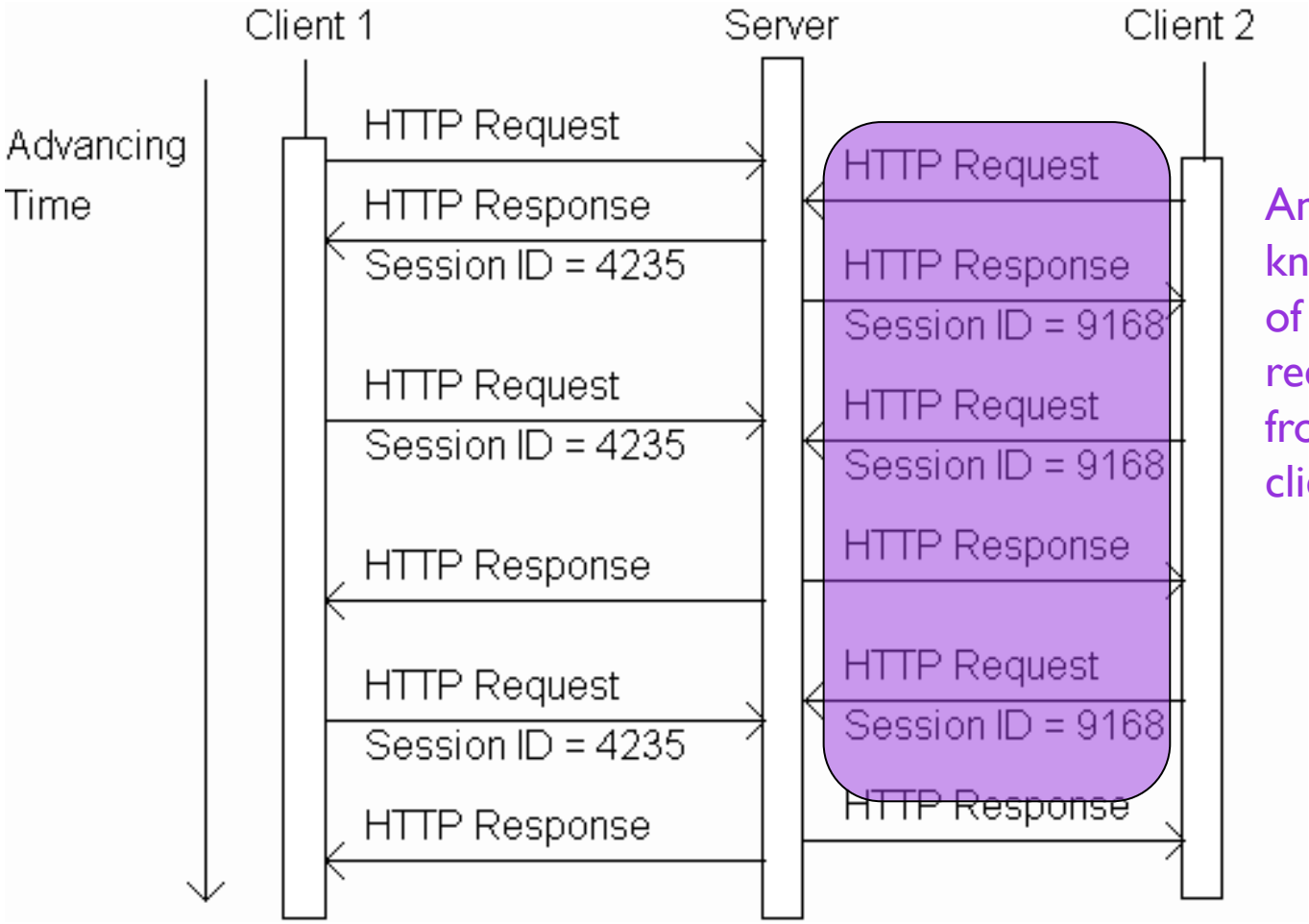
Client that supports session stores the ID and sends it back to the server in subsequent requests

SESSIONS



Server knows that all of these requests are from the same client. The set of requests is known as a *session*.

SESSIONS



And the server knows that all of these requests are from a different client.

HTTPSESSION OBJECT

- Every user of the site is associated with a javax.servlet.http.HttpSession object

- To retrieve the current Session object
  - HttpServletRequest interface methods
    - getSession()
      - Creates one if no valid session found, otherwise uses the exist
    - getSession(boolean flag)
      - If value passed in parameter is ‘false’, no new session will be created if no valid session found. It returns null.
      - If value passed in parameter is ‘true’, the method behaves same as getSession() method.

SESSION ATTRIBUTES

- To add data to session Object
  - setAttribute(String name, Object value)
- To retrieve an object from session
  - Object getAttribute(String name) – returns null if name is not found
- To get the names of all objects in session
  - Enumeration getAttributeNames()
- To remove an object from session
  - removeAttribute(String name)

SESSION LIFECYCLE

- Sessions do not last forever
  - Expires automatically due to inactivity (Default 30 min)
  - Explicitly invalidated by user

- When session expires or invalidated
  - The session object and its data values are removed

SETTING SESSION TIMEOUT

- Setting default Session timeout
  - In web.xml

    <session-config> <session-timeout> 30 </session-timeout> </session-config>

    - Time given is in minutes
    - Applies to all sessions created in a web application

  - Configured individually

    setMaxInactiveInterval(int sec)

- Terminating a session
  - session.invalidate()

TIPS ON USING SESSION OBJECT

- Session objects are stored in server and it utilizes memory. So store only the needed information in sessions. Avoid bulky objects in session this can result in "***out of Memory Error***" resulting in a application crash.

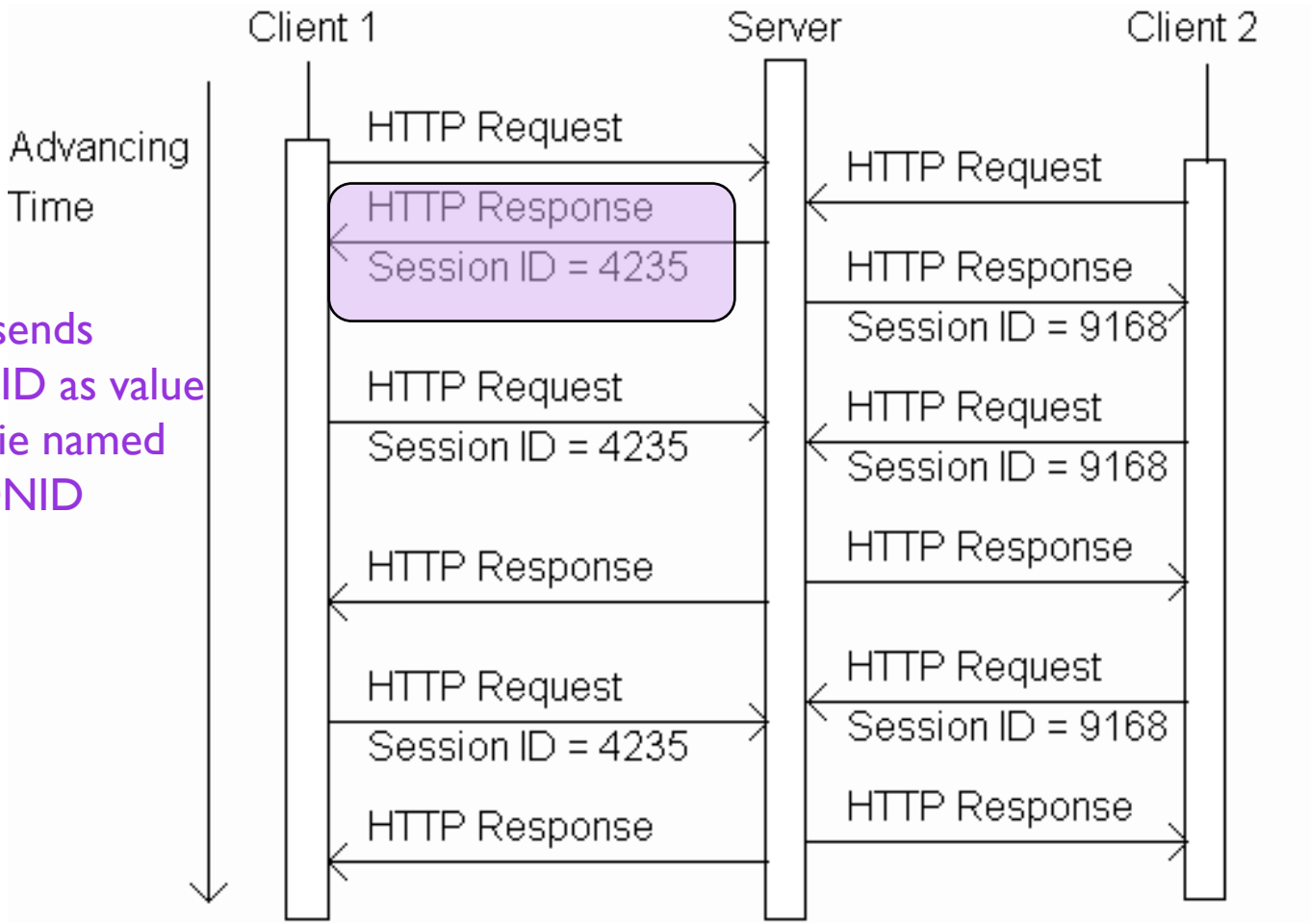
- Whenever a user closes the browser or logs out of the web application remove the session attributes set using the ***invalidate()*** and ***remove()*** API's.

COOKIE

- A cookie is a name/value pair in the Set-Cookie header field of an HTTP response

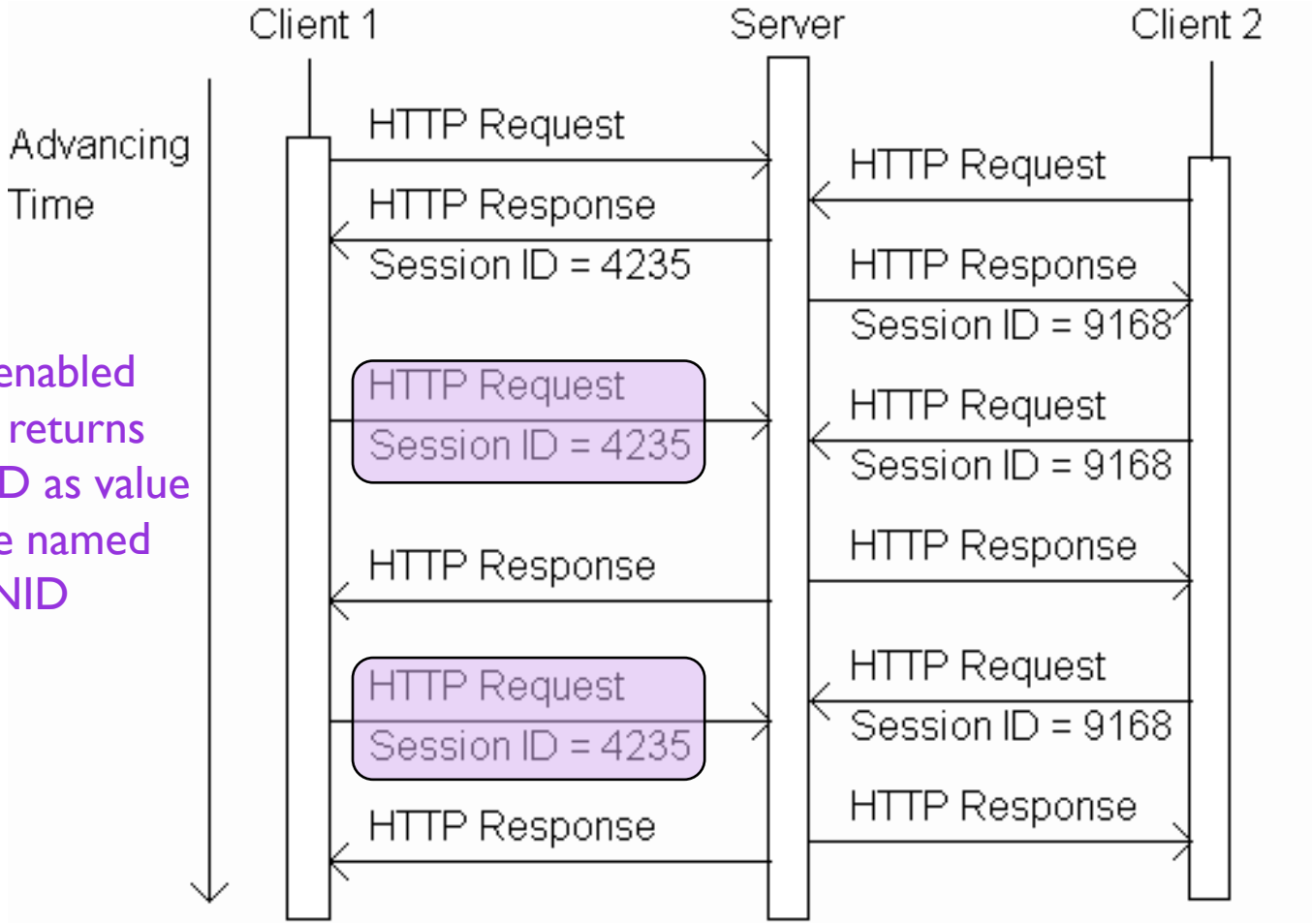
- Most (not all) clients will:
  - Store each cookie received in its file system
  - Send each cookie back to the server that sent it as part of the Cookie header field of subsequent HTTP requests

COOKIES



Server sends session ID as value of cookie named JSESSIONID

COOKIES



Cookie-enabled browser returns session ID as value of cookie named JSESSIONID

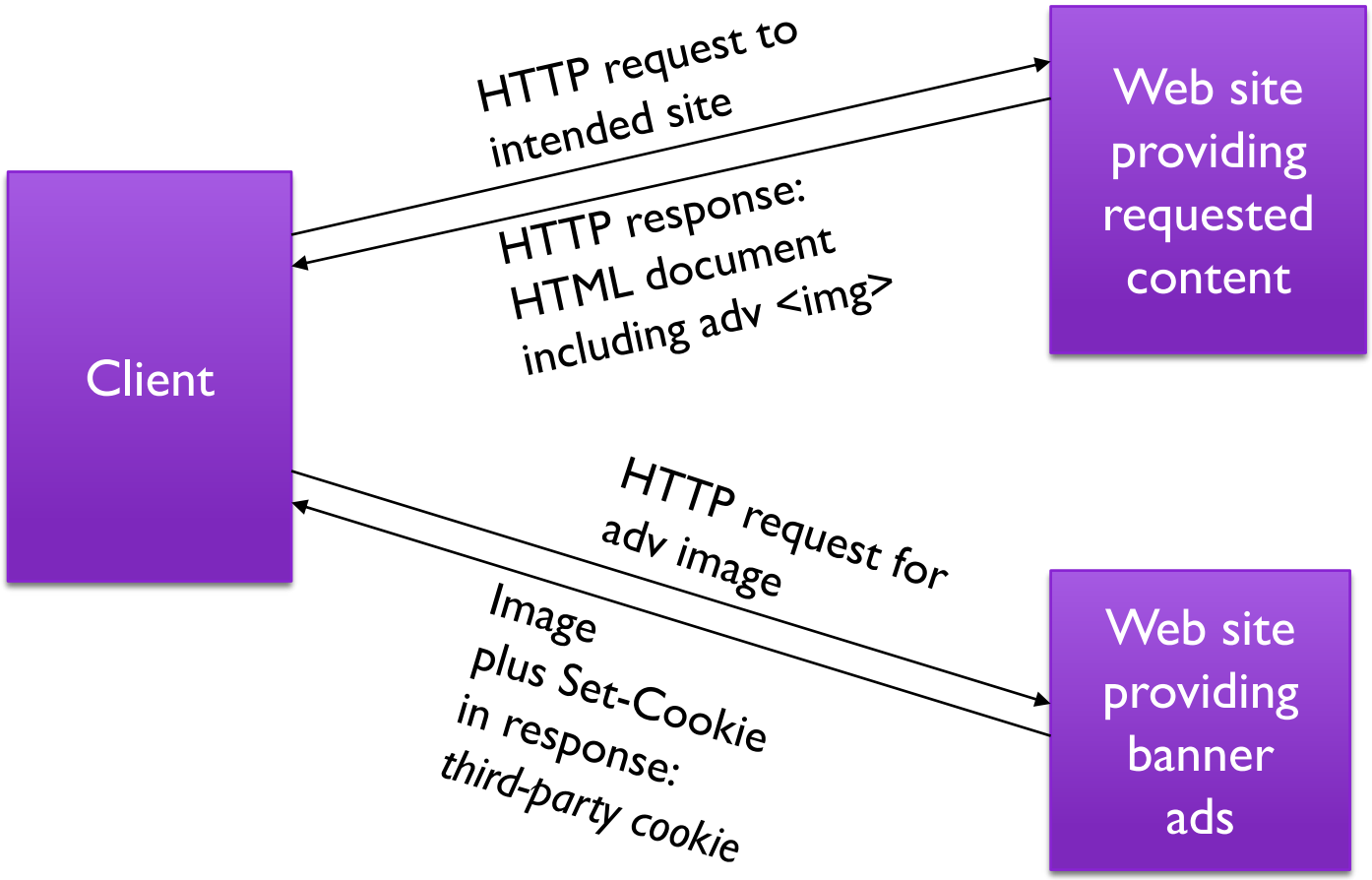
COOKIES

- Servlets can set cookies explicitly
  - Cookie class used to represent cookies
  - request.getCookies() returns an array of Cookie instances representing cookie data in HTTP request
  - response.addCookie(Cookie) adds a cookie to the HTTP response

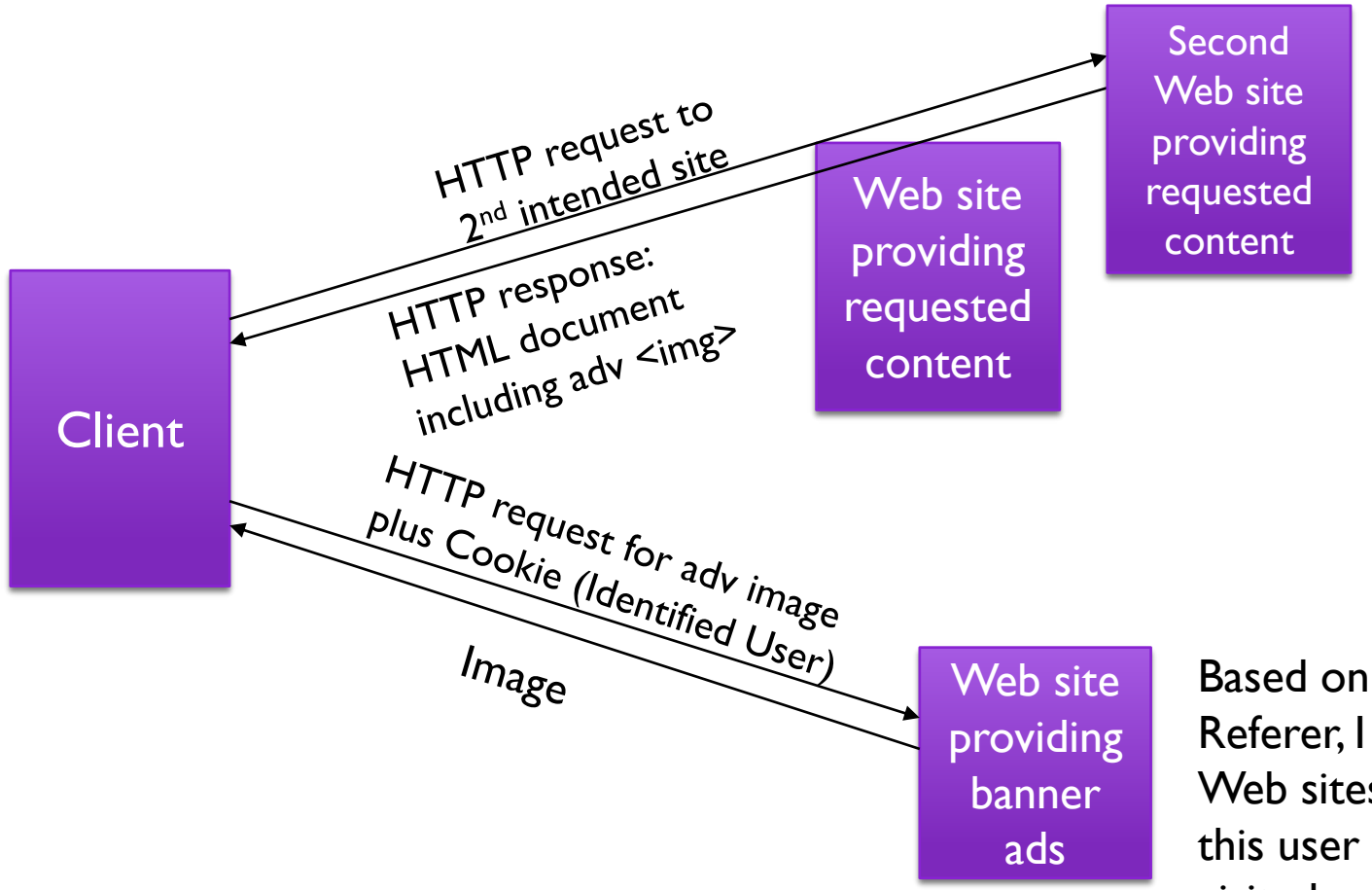
COOKIES

Method	Purpose
Cookie(String name, String value)	Constructor to create a cookie with the given name and value
String getName()	Return name of this cookie
String getValue()	Return value for this cookie
Void setMaxAge(int seconds)	Set delay until cookie expires. +ve value is delay in seconds. –ve value means that the cookie expires when the browser closes, and 0 (zero) means delete the cookie.

COOKIES PRIVACY ISSUES



COOKIE PRIVACY ISSUE



Based on Referer, I know two Web sites that this user has visited

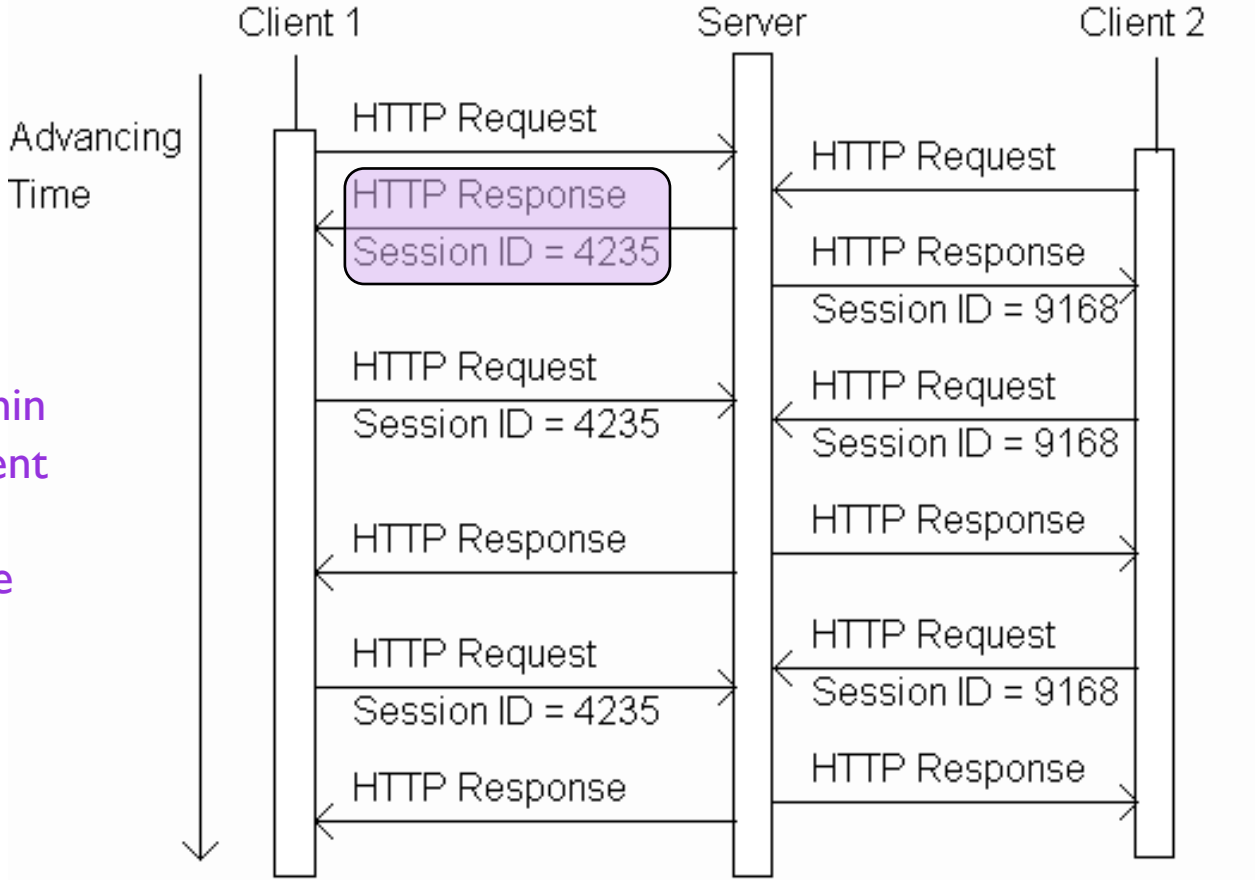
COOKIES PRIVACY ISSUE

- Due to privacy concerns, many users block cookies
  - Blocking may be fine-tuned. Ex: Mozilla allows
    - Blocking of third-party cookies
    - Blocking based on on-line privacy policy

- Alternative to cookies for maintaining session: URL rewriting

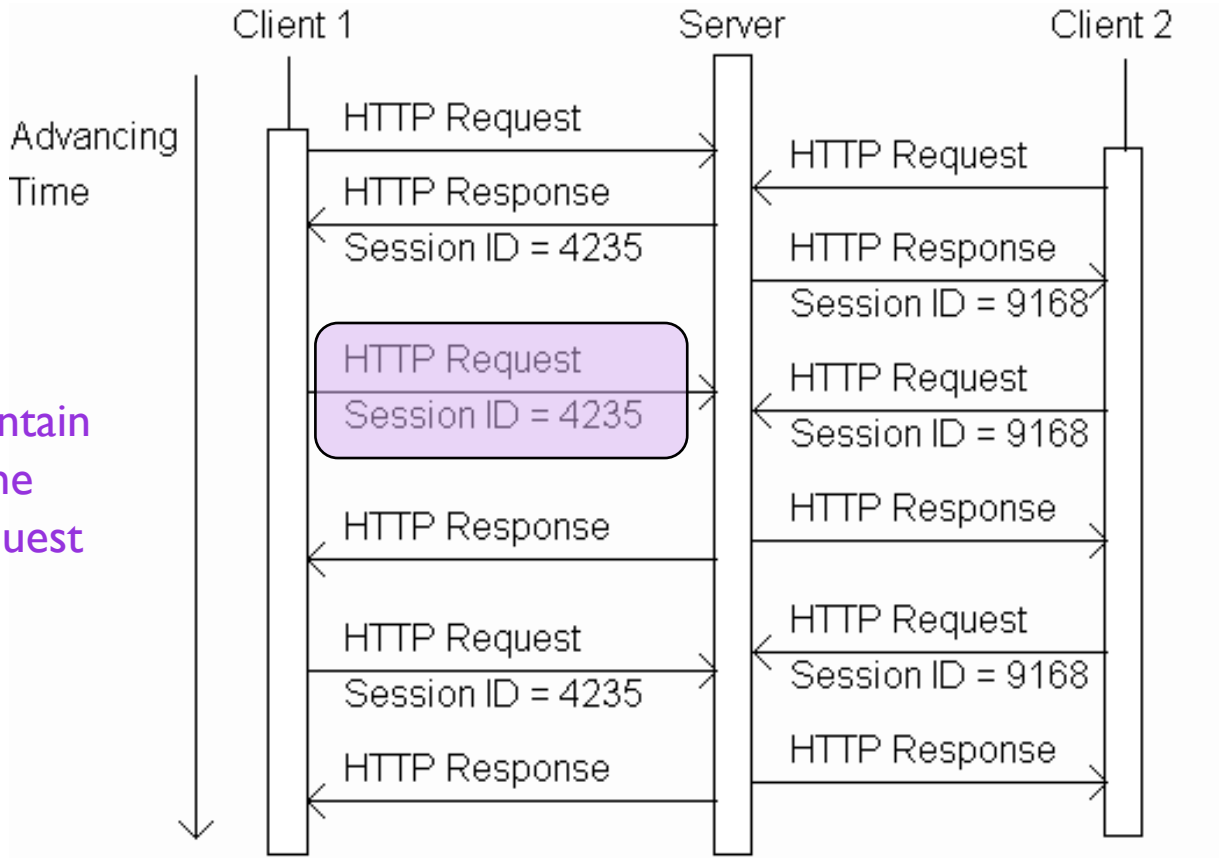
URL REWRITING

Server adds session ID within HTML document to all URL's referring to the servlet

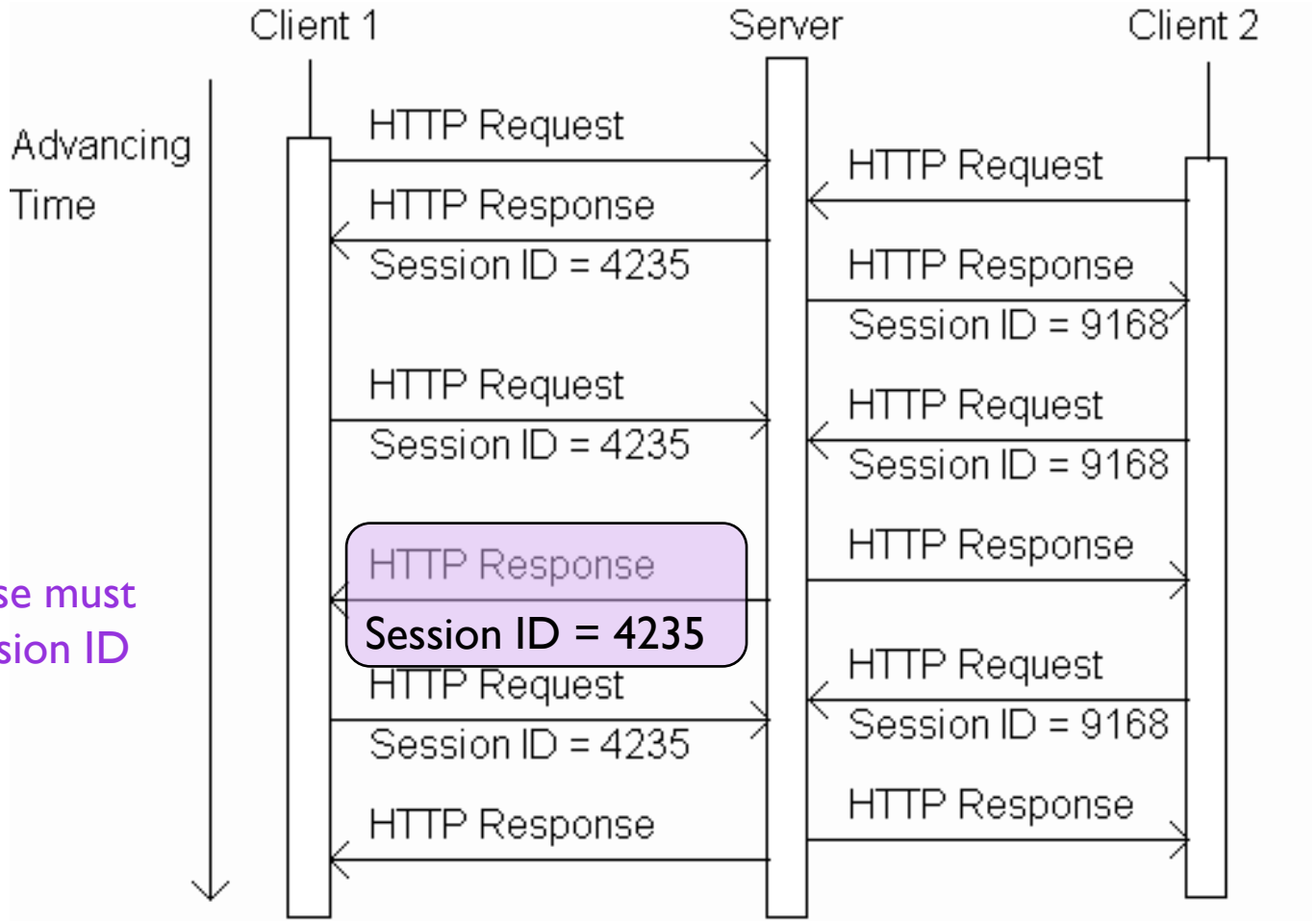


URL REWRITING

Subsequent request will contain session ID in the URL of the request



URL REWRITING



Next response must again add session ID to all URL's

URL REWRITING

- Original (relative) URL:

  href="URLEncodedGreeting"

- URL containing session ID:

  href="URLEncodedGreeting;jsessionid=0157B9E85"

  Path parameter

- Path parameter is treated differently than query string parameter
  - Ex: invisible to getParameter()

URL REWRITING

- HttpServletResponse method encodeURL() will add session id path parameter to argument URL

Original servlet

```
printSignInForm(servletOut, "Greeting");
```

Servlet using URL rewriting

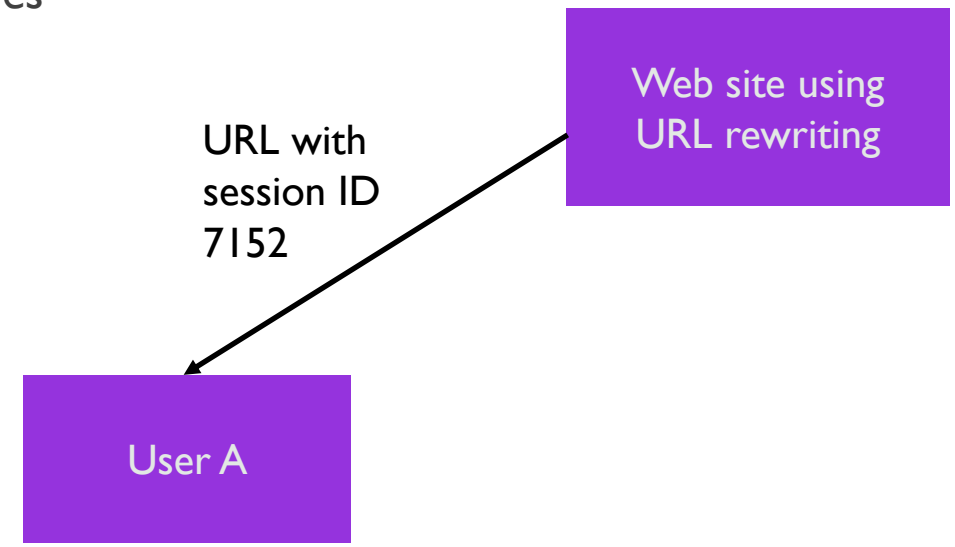
Relative URL of servlet

```
printSignInForm(servletOut,
                response.encodeURL("URLEncodedGreeting"));
```

URL REWRITING

- Must rewrite *every* servlet URL in *every* document

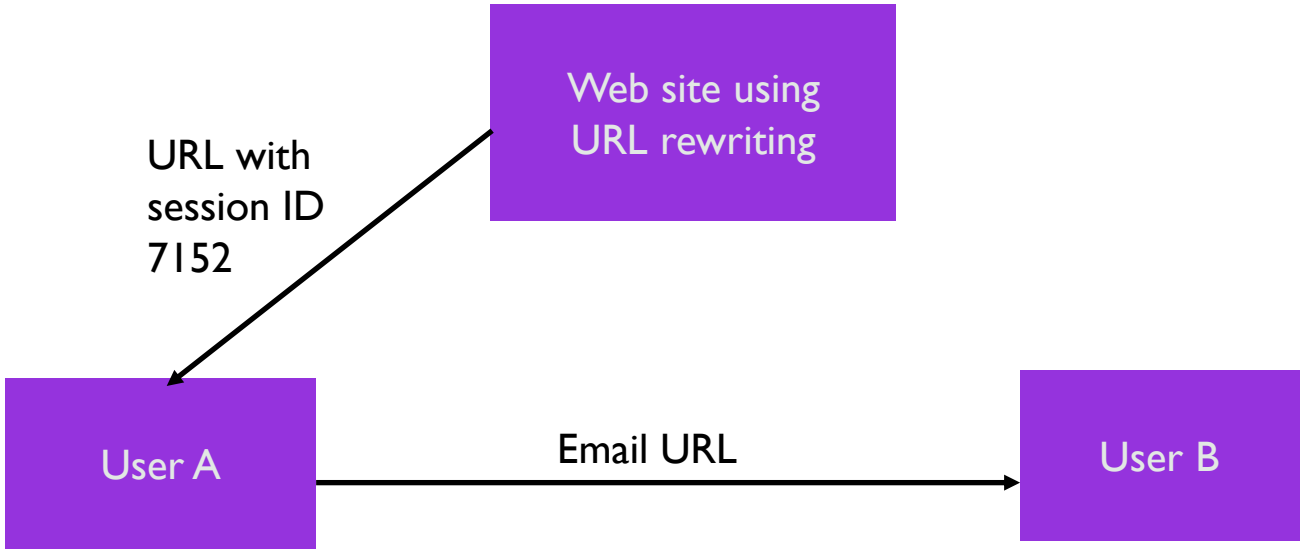
- Security issues



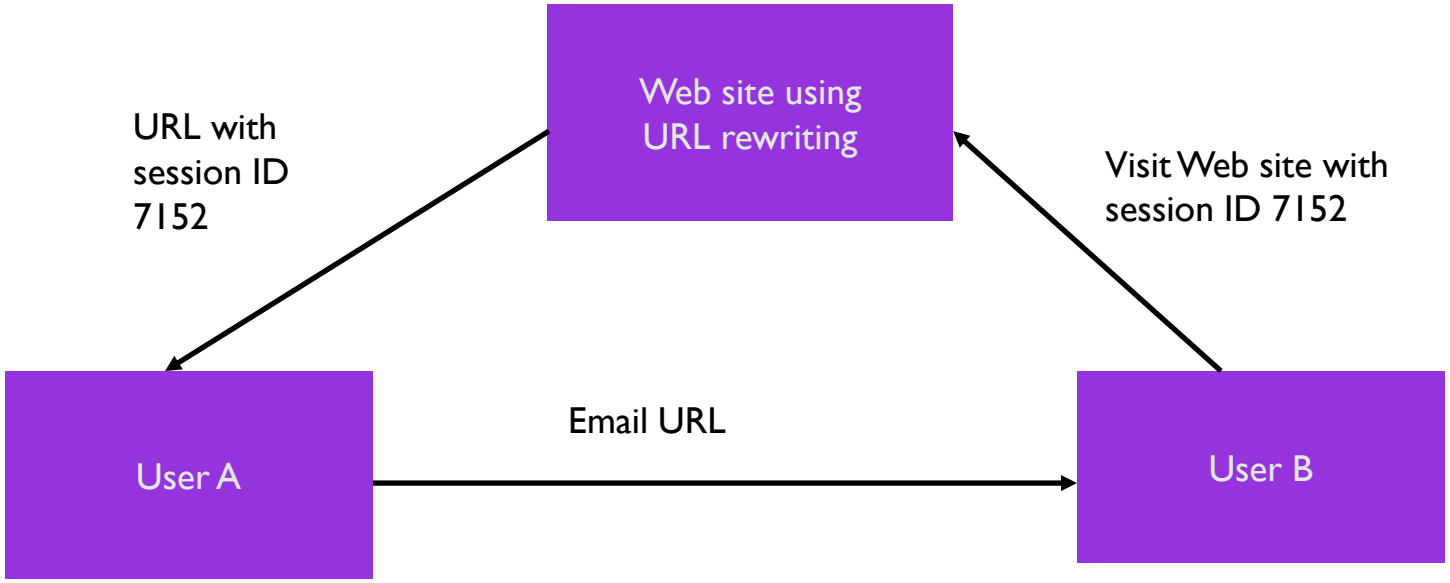
URL REWRITING

- Must rewrite *every* servlet URL in *every* document

- Security issues



URL REWRITING



GENERAL TIPS FOR USING SESSION MANAGEMENT

- Typically in Projects we use Session technique for holding the user attributes.
- Cookies are rarely used as there could be users accessing browsers without cookie support.
- If there is some state which is maintained in one or few pages, we use URL rewriting or hidden fields.
- Most Importantly never load session object with bulky objects. Load only the states which are used across the web application.

ATTRIBUTES SCOPES IN SERVLET

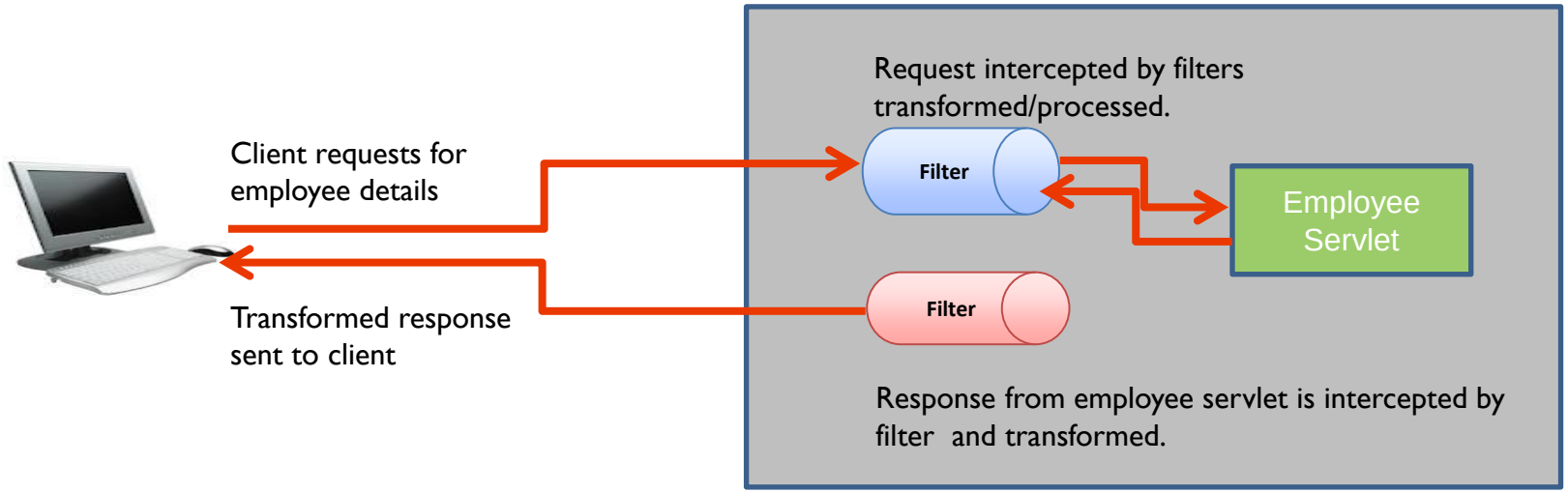
- Application
  - All Servlet/JSP in the web application have access.
  - Available for the life time of the application
  - Achieved by storing in the ServletContext object.

- Session
  - Available to all servlets/jsp that have access to the specific session.
  - Available for the life time of the session.
  - Achieved by Storing in the HttpSession object**.**

- Request
  - Available to servlets/jsp that have access to this specific request.
  - Available for the life of the request that is untill response send to client.
  - Achieved by storing in the HttpServletRequest object.

FILTERS

SERVLET FILTERS



- Servlet filter are programs (like the airport scanners) that runs on the server which intercepts HTTP request/response for transformations/processing/filtering.
- A filter program used for transforming/ filtering HTTP request or transforming HTTP response

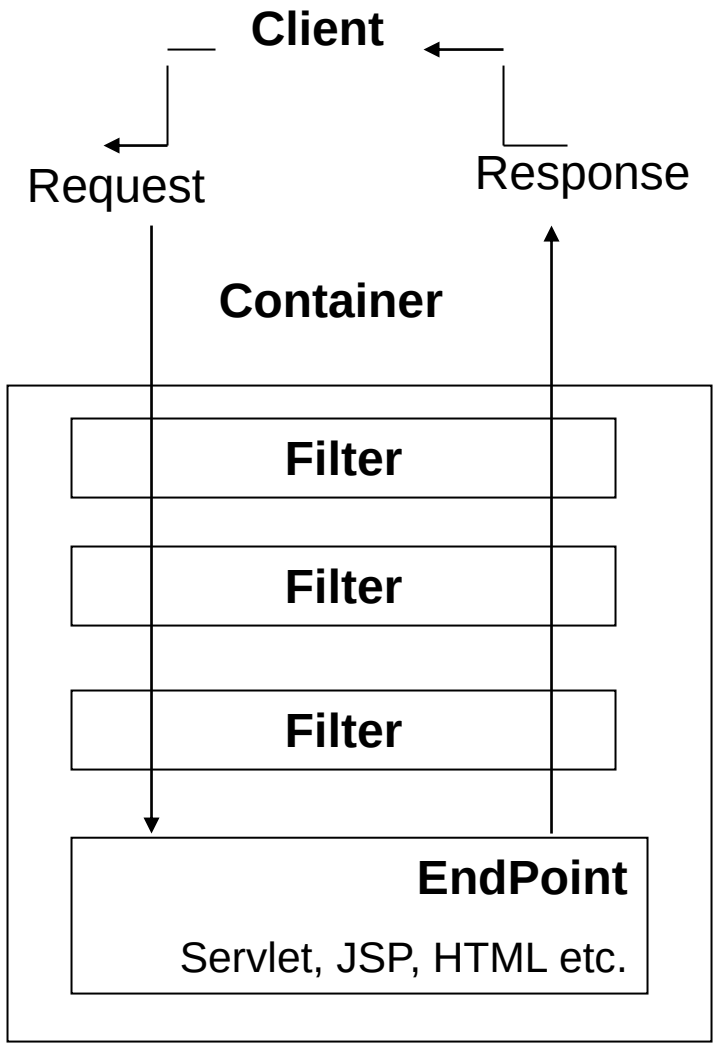
WHAT FILTERS CAN DO?

- Read the request data on the way to the endpoint.
- Wrap the request before passing it on.
- Wrap the response before passing it on.
- Manipulate the response data on the way back output.
- Return errors to the client.
- Request dispatch to another resource and ignore the original URL.
- Generate its own response before returning to the client.

ADVANTAGES OF FILTERS

- Identify type of request coming from the Web client, such as HTTP and FTP, and invoke the servlet that needs to process the request.
- Validate a client using servlet filters before the client accesses the servlet.
- Retrieve the user information from the request parameters to authenticate the user.
- Identify the information about the MIME types and other header contents of the request.
- Facilitate a servlet to communicate with the external resources.
- Intercept responses and compress it before sending the response to the client.

LISTING PATH TO SERVLET THROUGH FILTERS



FILTER CONFIGURATION

- Through Annotations
  - @WebFilter

```
@WebFilter(filterName = "AccessFilter",
            urlPatterns = { "/page_header.jsp", "/page_footer.jsp"}
)
```

- Through web.xml

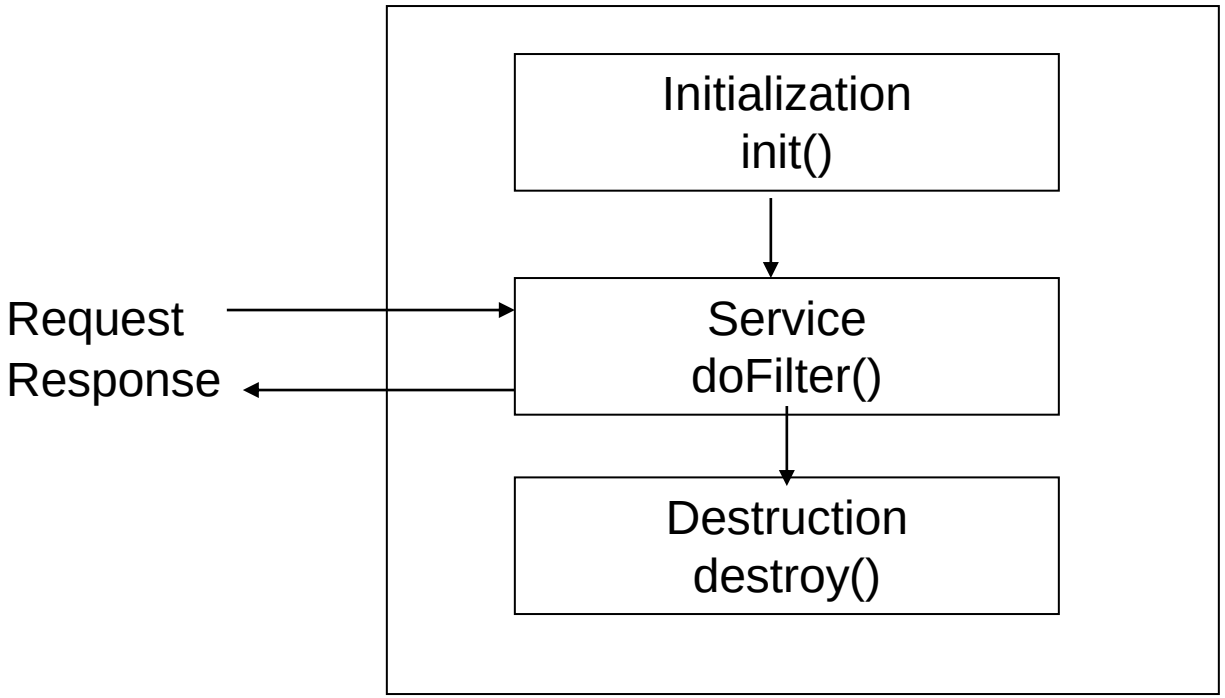
SAMPLE WEB.XML WITH FILTER CONFIGURED

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <filter>
        <filter-name>MyFilter</filter-name>
        <filter-class>myPackage.FilterClass</filter-class>
    </filter>
    <!-- ... -->
    <filter-mapping>
        <filter-name>MyFilter</filter-name>
        <url-pattern>/someDirectory/SomePage.jsp</url-pattern>
    </filter-mapping>
</web-app>
```

Name: Represents the name of the filter.
Class: Represents the fully classified filter class name.

Represents the URL patterns for which the filter will be invoked.

FILTER LIFECYCLE



FILTER CHAINING

- ***Filter chaining*** is the process of applying more than one filter to a Servlet.

- Assume there are two filters Filter A & B , they can be configured as below. The request goes through both the filters before it reaches the servlet.

FILTER CHAINING

- This is done by configuring more than one filter mapping for a servlet in the web.xml file

- The filters are invoked in the order they are declared in the web.xml file

```
<filter-mapping>
    <filter-name>Filter1</filter-name>
    <servlet-name>ChainingDemo</servlet-name>
</filter-mapping>
<filter-mapping>
    <filter-name>Filter2</filter-name>
    <servlet-name>ChainingDemo</servlet-name>
</filter-mapping>
```

Filter1 & Filter2 configured with the ChainingDemo servlet

POINTS ABOUT FILTERS

- A filter can be used for filtering more than one servlet, by configuring the same filter with multiple servlets.
- More than one filter can applied to a single servlet, applying two filters with a servlet called servlet chaining
- The following are some instances where filter are used on application development
  - Authentication/Authorization of all HTTP request to web applications to ensure that authorized user access we application.
  - To pre/post process request like checking message format, validating form data etc.