Servlets Session Management

Lecture 2 - Servlets Session Management

- Servlet Filters
- Servlet Context & Config
- Session object
- Session API
- Session attributes
- Session Tracking

Servlets - Writing Filters

- They are used to
  - intercept requests from a client before they reach a servlet
  - manipulate responses from server before they are sent to the client
- Types of filters:
  - Authentication
  - Data compression
  - Encryption
  - Image conversion
  - Logging & auditing
  - XSL/T Filters to transform XML content, etc.

Filters in Web.xml

- Deployed in web.xml where they map to servlet URL patterns

```
<filter>
  <filter-name>LogFilter</filter-name>
  <filter-class>com.ece.jee.LogFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>LogFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

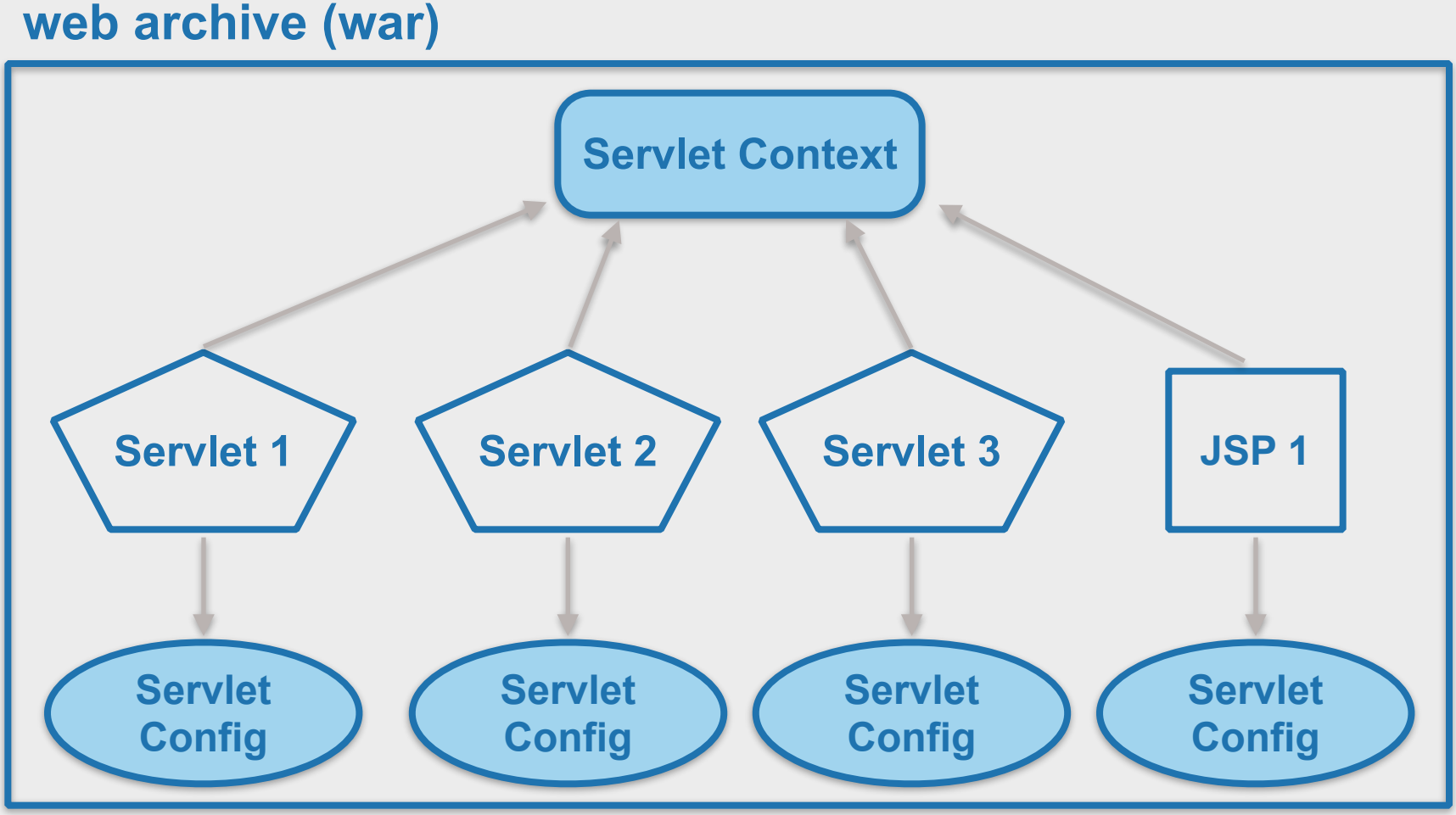
Servlet Filters

- An instance of each filter declared in the deployment descriptor is created when the web container starts.
- Filters execute in the order that they are declared in the deployment descriptor.
- Methods to implement
  - doFilter() - called each time a request/response is passed through the chain
  - init() - initializing and placing the filter in service
  - destroy() - taking the filter out of service

Filter Example

```
@WebFilter(description = "basic logging", urlPatterns = { "/*" })
public class LogFilter implements Filter {

    public void init(FilterConfig fConfig) throws ServletException {
        // Called when filter is placed in service
    }

    public void destroy() {
        // Called when filter is removed from service
    }

    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {
        // Get the IP address of client machine.
        String ipAddress = request.getRemoteAddr();

        // Log the IP address and current timestamp.
        System.out.println("IP " + ipAddress + ", Time "
                + new Date().toString());

        // pass the request along the filter chain
        chain.doFilter(request, response);
    }
}
```

Web Container - Servlet Context & Config



Initialization Parameters

	Context Init Parameters	Servlet Init Parameters
Scope	Scope is Web Container	Specific to Servlet or JSP
Servlet code	getServletContext()	getServletConfig()
Deployment Descriptor	Within the <web-app> element but not within a specific <servlet> element	Within the <servlet> element for each specific servlet

Servlet Init parameter - web.xml

```
<servlet>
   <servlet-name>simpleServlet</servlet-name>
   <servlet-class>com.ece.jee.SimpleServlet</servlet-class>

      <init-param>
      <param-name>myParam</param-name>
      <param-value>paramValue</param-value>
      </init-param>
</servlet>
```

Init parameter - Servlet

```
public class SimpleServlet extends GenericServlet {

  protected String myParam = null;

  public void init(ServletConfig servletConfig) throws ServletException{
    this.myParam = servletConfig.getInitParameter("myParam");
  }

  public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
   response.getWriter().write("<html><body>myParam = " +
           this.myParam + "</body></html>");
  }
}
```

Load-on-startup

- By setting a <load-on-startup> element, you can tell the servlet container to load the servlet as soon as the servlet container starts

```
<servlet>
    <servlet-name>controlServlet</servlet-name>
    <servlet-class>com.ece.jee.SimpleServlet</servlet-class>
    <init-param><param-name>container.script.static</param-name>
            <param-value>/WEB-INF/container.script</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

Context Init parameters

- **Web.xml**

```
<context-param>
   <param-name>myParam</param-name>
   <param-value>the value</param-value>
</context-param>
```

- **Inside HTTPServlet subclass**

```
String myContextParam =
request.getSession().getServletContext().getInitParameter("myParam");
```

Request Object

- HTTP - Stateless Protocol

- How to make our web app remember us?
- Used for login screens, shopping carts etc.

- How to share values between different Servlets?
- How to share values between different users?

Session

- Server will create session object
  - assign unique ID
  - track sessions by some method such as cookies or URL rewriting
- Servlet can store anything in session context using session attributes
- Getting the session object

```
HTTPSession session = req.getSession(true);
```

Session API

- Getting the ID

  String getId();

- Getting the creation date

  long getCreationTime();

- Getting last access time

  long getLastAccessTime();

- Terminating a session

  void invalidate();

Session API - Timeout

- Get the maximum inactive time (seconds) . After this the session closes automatically.

  `int getMAxInactiveInterval();`

- Setting the maximum inactive time. A negative value means infinite time.

  `void setMaxInactiveInterval(int seconds);`

Web container - attributes

	Attributes	Parameters
Types	Context Request Session	Context Init Request Servlet Init
Method to set	setAttribute(String, Object)	We cannot set Init parameters.
Return type	Object	String
Method to get	getAttribute(String)	getInitParameter (String)

Request Attributes

- One per request
- Not available across multiple requests
  - When you get redirected to another servlet, your request dies
- Normally used for passing data from jsp to your servlet when you submit the form.

Session Attributes

- One per user/machine
- Object available across multiple requests
- Every request object has a handle to the session object
- Used for
  - Storing user credentials once user is authenticated.
  - Checking if the user has right access to do operations like add/ delete/edit on some database.
- Once the session goes idle for x amount minutes, the session dies and all info in it will be gone

Session Attributes

- Placing attributes in session

  `session.setAttribute("age", "25");`

- Getting an attribute

  `Integer value = (Integer) session.getAttribute("age");`

- Getting all attribute names

  `Enumeration aNames = request.getAttributeNames();`

- Deleting an attribute

  `session.removeAttribute("age");`

Context attributes

- Across entire application
- Shared across multiple servlets and users
- Initialization code

Example - Servlet Attributes

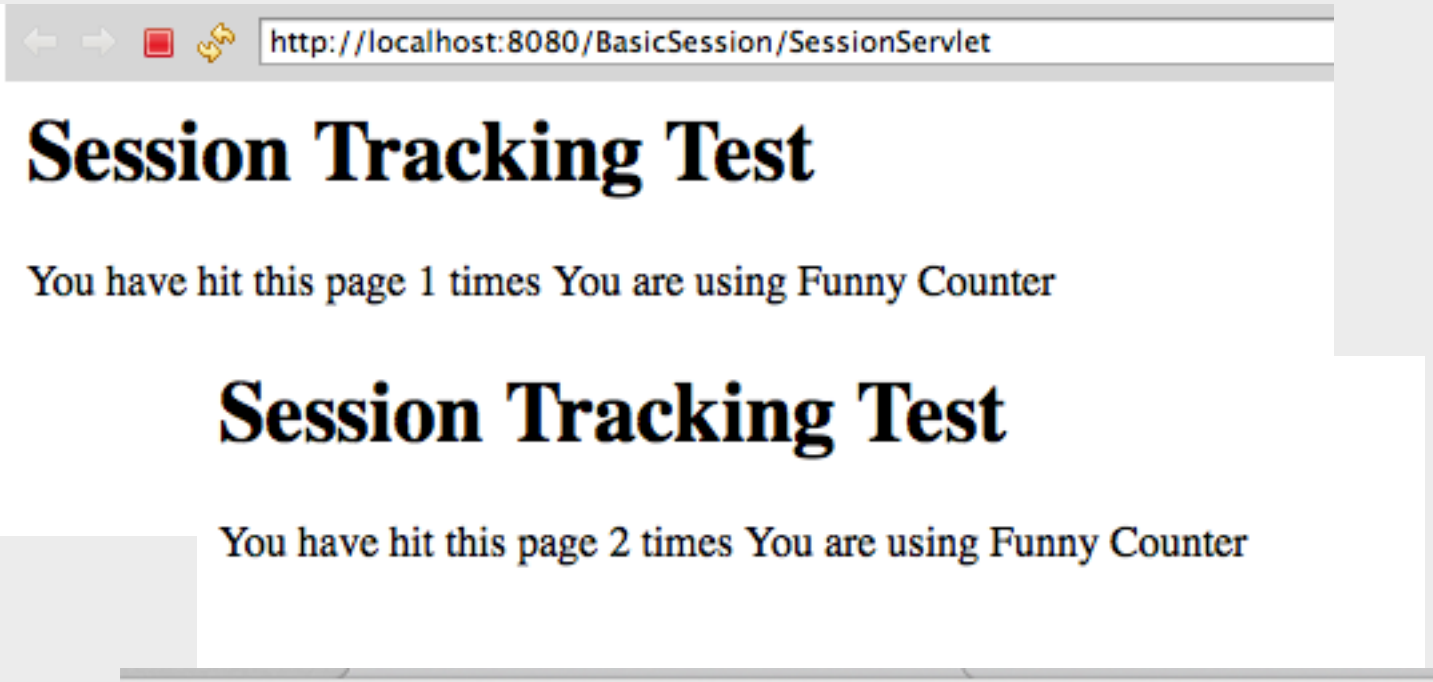
```
@WebServlet(description = "testing the session", urlPatterns = { "/
SessionServlet" })
public class SessionServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        // Part 1: Get the Session & Context object
        HttpSession session = request.getSession(true);
        ServletContext context = request.getServletContext();

        // Part 2: Get the session data value
        Integer ival = (Integer) session.getAttribute("counter");
        if (ival == null)
            ival = new Integer(1);
        else
            ival = new Integer(ival.intValue() + 1);
        session.setAttribute("counter", ival);
```

Example - Servlet Attributes

```
        // Part 3: Set the SessionContext attribute
        context.setAttribute("counterName", "Funny Counter");
        String name = (String) context.getAttribute("counterName");

        // Part 4: Output the page
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Session Tracking Test</title></
head>");
        out.println("<body>");
        out.println("<h1>Session Tracking Test</h1>");
        out.println("You have hit this page " + ival + " times");
        out.println("You are using " + name);
        out.println("</body></html>");
    }
}
```

Example - Servlet Attributes

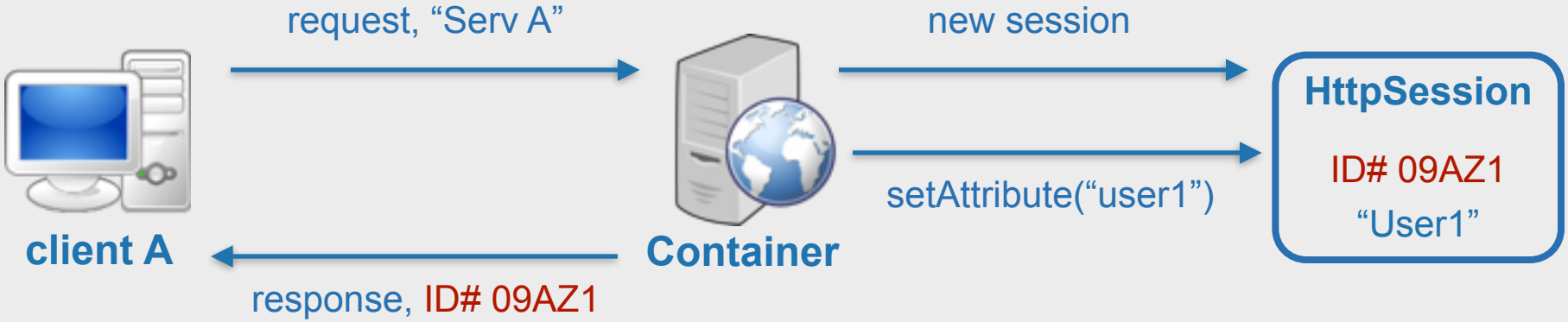
First Browser



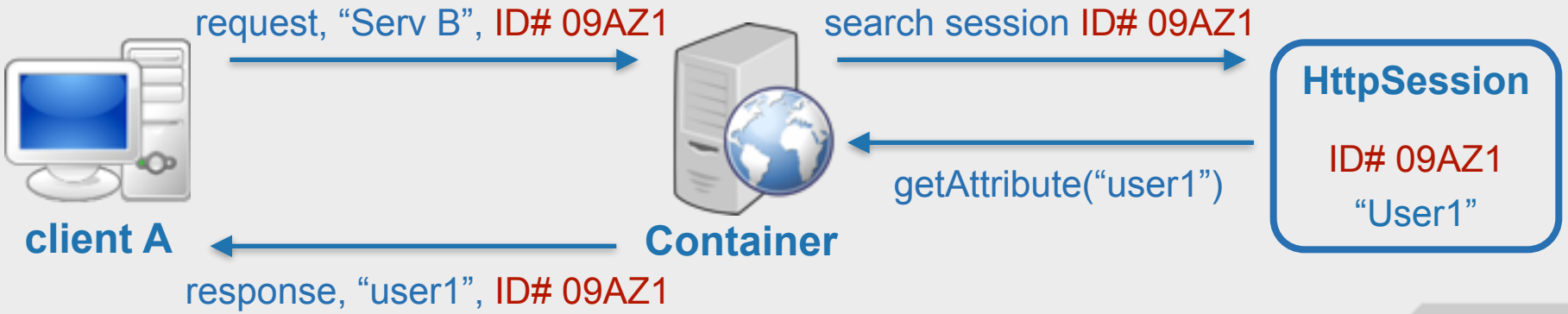
Second Browser

Session tracking

First Request



Subsequent Request



Session tracking - cookies

First Response

client A

Http Response

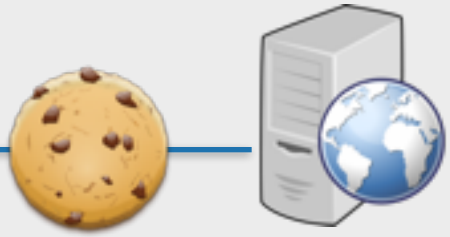
HTTP/1.1 200 OK

Location: http://www.abcd.com/login

Set-Cookie: JSESSIONID=09AZ1

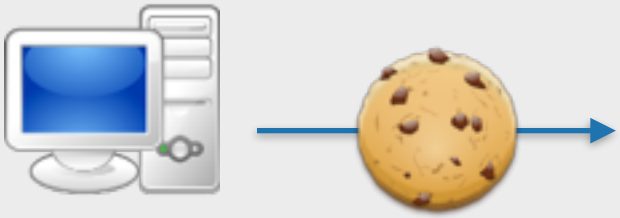
Domain=.abcd.com;path=/;HttpOnly

……



Container

Subsequent Requests



client A

Http Request

POST/login.do HTTP/1.1

Host: www.abcd.com

Cookie: JSESSIONID=09AZ1

……

Container

Session tracking - URL Rewriting

First Response

Subsequent Requests

Session tracking - URL Rewriting

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
…

// Get the current session ID
    String sessionid = req.getPathInfo();

…..
out.println("<FORM ACTION=\"/servlet/ShoppingCart/" + sessionid +
            "\" METHOD=POST>”);
…..
}
```

Session Tracking

- With first response server sends both cookies and url rewriting.
- If the cookies are not blocked by the client, the server chooses this method and manages the session through cookies
- Incase cookies are disabled, then the clients next request uses url rewriting. After this, server choosing the url rewriting for maintaining the session.

Problems with browser cache

- For the practical exercise on Login Servlets, the web browsers may show you older pages and images, store in cache.

- To avoid this, you can avoid cache in HTTP response

```
response.setHeader("Pragma", "No-cache");
response.setDateHeader("Expires", 0);
response.setHeader("Cache-Control", "no-cache");
```

Transferring Control - Forward

- Performed by servlet internally
  - client browser does not know about it
  - the client browser displays original request URL
- Browser Reload: means repeating the original request
- In the implementing method

```
RequestDispatcher rd =
   request.getRequestDispatcher("SuccessServlet");
rd.forward(request, response);
```

Transfering Control - Redirect

- Browser tells user web-browser to fetch a different URL
- The browser request is a second request not the the *original* request
- Slower than Forward
- Objects of *original* request are not available in the second request
- Reload means a second request not repeating original
- In doPost or DoGet method

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