OBJECT™
TECHNOLOGIES

Servlet IV

What will be covered

OBJECT TECHNOLOGIES

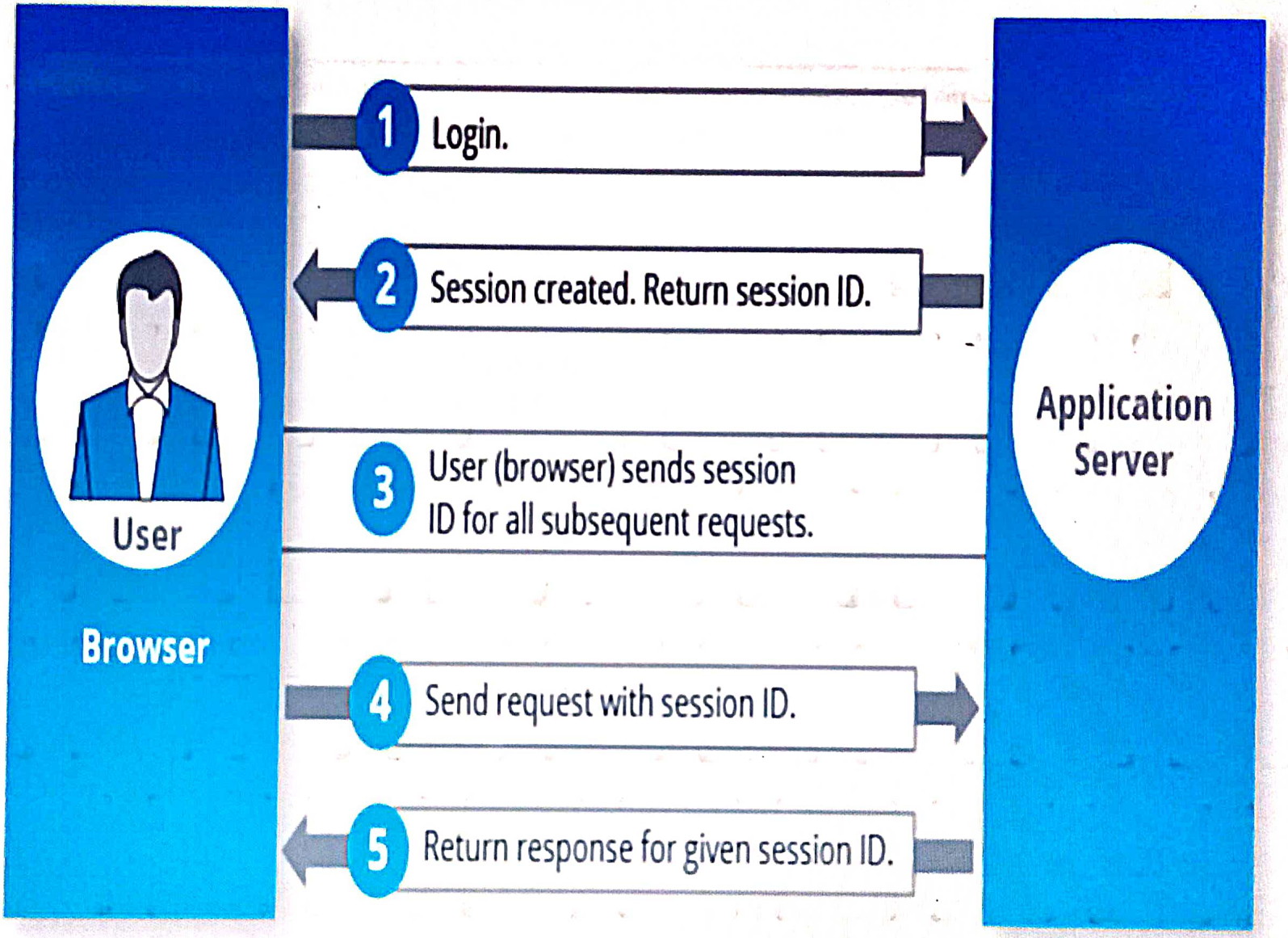
- What is http session
- Working of sessions
- Key differences in cookies and sessions
- Session management
- Session functionality
- Introduction to filters
- Filter API
- Example of filters

What is http session

OBJECT

- Session simply means a particular interval of time.
- Session is a conversional state between client and server and it can consists of multiple request and response between client and server.
- Session is a way to maintain state (data) of an user.
- Session can be considered as some block of memory residing on server side which is used to store information about a single client in the form of key - value pair and can span across multiple requests.
- A web session is a series of contiguous actions by a visitor on an individual website within a given time frame. This could include your search engine searches, filling out a form to receive content, scrolling on a website page, adding items to a shopping cart, researching airfare, or which pages you viewed on a single website. Any interaction that you have with a single website is recorded as a web session to that website property.

Working of sessions



Working of sessions

OBJECT

- Sessions are created in response to some user request. So, when User 1 send Request 1 to server, the requested page creates a new session. As a result, a session is created at server side. A unique session ID is generated,.
- When a new session is created, it stores the session ID in HTTP response in Set-Cookie header. The cookie name used by java is jsessionid, so Set-Cookie:jsessionid=S1 is stored in HTTP response packet. So that it could be sent to browser.
- When web browser receive an HTTP response that contain Set-Cookie header, it creates a new cookie on user computer which is stored in browser cookies.
- When User 1 send Request 2 to some page, browser automatically add jsessionid cookie in HTTP request packet under Cookie header i.e. Cookie:jsessionid=S1
- When server receive jsessionid cookie with S1 value, it loads data stored against S1 ID HttpSession object. The web page may retrieve earlier stored data or add new data into session.

Key differences in cookies and sessions

OBJECT

- Cookies are client-side files that contain user information, whereas Sessions are server-side files that contain user information.
- Cookie is not dependent on session, but Session is dependent on Cookie.
- Cookie expires depending on the lifetime you set for it, while a Session ends when a user closes his/her browser.
- The maximum cookie size is 4KB whereas in session, you can store as much data as you like.
- You won't set all your important information in a cookie, because users can mess that information up. Data in your session is more secure.
- A cookie's data can be modified, as the data is stored locally (on the client), where as a session's data is stored on the server, and can not be modified (by the client).

Session management

OBJECT

- In Java, a HttpSession object represents the session of a particular user.
- Note that HttpSession is an interface defined in the javax.servlet package, whereas the actual implementation is injected to the HttpServletRequest by the servlet container (i.e. the server like Tomcat).
- You can store user-related information in a session in form of key and value pairs.
- The HttpSession interface defines the setAttribute(key, value) method to store a key-value entry and getAttribute(key) method to get value of a specified key.

Session functionality

OBJECT

- **Getting or Creating a Session**
- a session is automatically created when the user visits the website. To obtain the HttpSession object representing the user's session, invoke the getSession() method of the HttpServletRequest interface in doGet() or doPost() method of a Java Servlet.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    HttpSession session = request.getSession();
```

- This method returns the current session associated with the request, or create a new one if the request doesn't have a session. That means the returned HttpSession object is always not null.
- To get a session without creating a new one if not exist, you can use invoke getSession(false) on the HttpServletRequest:

```
HttpSession session = request.getSession(false);
if (session != null) {
    // a session exists

} else {
    // no session
}
```

Session functionality

OBJECT

- **Store data in session**
- To store a value in a session, use the method setAttribute(key, value) of the HttpSession object. For example, the following statement stores username of the user:

```
session.setAttribute("username", "Daniel Tran");
```

- Here, the key is username and value is Daniel Tran. Data stored in a session is managed by the server and will be deleted when the session ends.
- You can store any kind of object in the session. For example, the following code stores a List of Student objects in the session.

```
List<Student> students = studentDao.getStudents();
session.setAttribute("listStudent", students);
```

Session functionality

OBJECT

- **Read data from session**
- To get value from a session, use the getAttribute(key) method of the HttpSession object. For example, the following code gets value of the username attribute from the session.

```
String username = (String) session.getAttribute("username");
```

- We need a cast to String type because the getAttribute() method always returns a value of Object type.
- The following statement reads a List collection from the session.

```
List<Student> listStudents = (List<Student>) session.getAttribute("listStudent");
```

- Note that the getAttribute(key) method will return null value if the given key is not found in the session.
- **Remove data from session**
- To delete a value associated with a key from the session, use the removeAttribute(key) method

```
session.removeAttribute("username");
```

Session functionality

OBJECT

- **Configure Session Timeout**
- If a user has been idle (has not made any requests) for a given amount of time, his session expires – which means all the data bound to his session is removed from the server – the session is destroyed.
- Session timeout for an individual web application can be specified by modifying its web deployment descriptor file (web.xml)

```
<web-app...>

  <session-config>
    <session-timeout>15</session-timeout>
  </session-config>

</web-app>
```

- A timeout value can be set for an individual session programmatically like following. Note: 300 seconds

```
session.setMaxInactiveInterval(300);
```

Session functionality

OBJECT

- **Invalidate a Session**
- By default, a session is destroyed only after the user has been idle for a timeout period. In case if it is needed to destroy immediately, call the invalidate() method

```
session.invalidate();
```

- This removes any objects bound to the session and destroy it.

Introduction to filters

OBJECT

- The Java Servlet specification has introduced a new component type, called a filter. A filter dynamically intercepts requests and responses to transform or use the information contained in the requests or responses.
- Filters typically do not themselves create responses, but instead provide universal functions that can be "attached" to any type of servlet or JSP page.
- They provide the ability to encapsulate recurring tasks in reusable units.
- Filters can perform many different types of functions
  - Authentication-Blocking requests based on user identity.
  - Logging and auditing-Tracking users of a web application.
  - Image conversion-Scaling maps, and so on.
  - Data compression-Making downloads smaller.
  - Localization-Targeting the request and response to a particular locale.

Filter API

OBJECT

- The filter API is defined by the Filter, FilterChain, and FilterConfig interfaces in the javax.servlet package.
- **Filter** - For creating any filter, you must implement the Filter interface. Filter interface provides the life cycle methods for a filter.
- The object of **FilterChain** is responsible to invoke the next filter or resource in the chain.This object is passed in the doFilter method of Filter interface. It has only one method.
- A **FilterConfig** contains initialization data.

Example of filter

OBJECT

- Basic methods for creating filters -

```
*/
public class SimpleServletFilter implements Filter {

    public void init(FilterConfig filterConfig) throws ServletException {
    }

    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain filterChain)
    throws IOException, ServletException {

    }

    public void destroy() {
    }
}
```

- doFilter() method can be used for implmenting filter's task

```
public void doFilter(ServletRequest request, ServletResponse response,
                     FilterChain filterChain)
throws IOException, ServletException {

    String myParam = request.getParameter("myParam");

    if(!"blockTheRequest".equals(myParam)){
        filterChain.doFilter(request, response);
        return;
    }
```

Example of filter

- Configuring the Servlet Filter in web.xml
- Servlet filter needs to be configured in the web.xml file of your web application, before it works.

```
<filter>
    <filter-name>myFilter</filter-name>
    <filter-class>servlets.SimpleServletFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>myFilter</filter-name>
    <url-pattern>*.simple</url-pattern>
</filter-mapping>
```

- With this configuration all requests with URL's ending in .simple will be intercepted by the servlet filter. All others will be left untouched.
- This configuration can be achieved using @WebFilter annotation also

Servlet IV

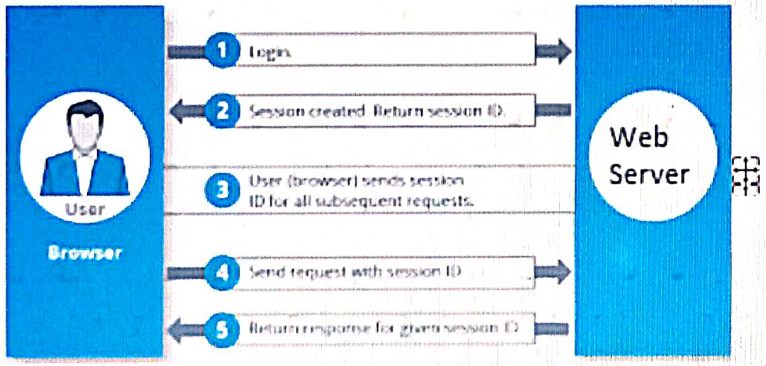
In this chapter we will get clear understanding of handling session for a client. We will even see in brief about use of filters in servlet API

Handling Sessions

Servlets provide an outstanding technical solution: the HttpSession API. This is a high-level interface that allows the server to "remember" a set of information relevant to a particular user's on-going transaction, so that it can retrieve this information to handle any future requests from the same user.

Whenever you need to remember some information about what the user does, you start a new HttpSession and attach (or store) the information to this session. Then behind the scene, your web server will assign a unique "session id" to the session and send this id to the browser, asking the browser to send back the id in any future requests. This way, the web server will be able to trace the requests coming from the same user, and let you retrieve whatever has been stored to handle the user's requests.

The servlet HttpSession uses one of two mechanisms to ask the browser to remember and send back the session id in future requests: Cookies or URL rewriting. If the user's browser supports cookies, the Tomcat server will ask the browser to store the session id in cookies. In case the browser does not support cookies or the user explicitly disabled it, the server reverts to URL-rewriting by appending the session id to the end of every URL.



Session Creation:

When a new user visits, the server creates a unique session ID and associates it with the user's browser.

The session ID is typically stored in a cookie on the client side.

Session can be obtained using :

```
HttpSession session = request.getSession();
```

Session Creation:

When a new user visits, the server creates a unique session ID and associates it with the user's browser.

The session ID is typically stored in a cookie on the client side.

Session can be obtained using :

```
HttpSession session = request.getSession();
```

Above method will create a new session object if not already created for this user. If session already exists, it will be returned. In some cases if we need to access existing session without creating a new one, we can make use od following overloaded method.

```
HttpSession session = request.getSession(false);
```

'false' argument indicates that return session only if it exists. So if session is not already maintained, it returns null.

Storing Data in Session:

You can store objects in the session using

```
HttpSession session = request.getSession();
session.setAttribute(name, value);
```

Here name should be in the form of string where as value could be any object like string, array, or any type of collection etc.

Retrieving Data from Session:

Access stored objects using

```
HttpSession session = request.getSession();
session.getAttribute(name);
```

Above method returns Object which needs to be type casted in the required type.

Session Timeout:

Sessions expire after a certain period of inactivity, configurable in the web application's deployment descriptor (web.xml).

Session timeout can be set for an individual session as well using following method

```
session.setMaxInactiveInterval(30);
```

Remember this time is in minutes

Session Timeout:

Sessions expire after a certain period of inactivity, configurable in the web application's deployment descriptor (web.xml).

Session timeout can be set for an individual session as well using following method

```
session.setMaxInactiveInterval(30);
```

Remember this time is in minutes

Destroying a Session:

You can manually destroy a session using

```
session.invalidate();
```

This completely removes all the memory occupied by respective user's session.

Introduction to Servlet Filters

Filters are Java classes that intercept and process requests and responses before they reach servlets or static resources. They provide a way to implement cross-cutting concerns that apply to multiple servlets or resources.

Can be used for tasks like:

- Authentication and authorization
- Data compression
- Logging and auditing
- Image or data manipulation
- Character encoding
- Modification of request headers and parameters

Multiple filters can be configured in a chain, with each filter processing the request/response before passing it to the next.

In short : Filters are pluggable classes that stand between the client and a target component (like a servlet or JSP), within a web application. We can do pre or post processing of request/response data while it is coming from client to a servlet or from the servlet back to a client.

Filter lifecycle:

Introduction to Servlet Filters

Filters are Java classes that intercept and process requests and responses before they reach servlets or static resources. They provide a way to implement cross-cutting concerns that apply to multiple servlets or resources.

Can be used for tasks like:

- Authentication and authorization
- Data compression
- Logging and auditing
- Image or data manipulation
- Character encoding
- Modification of request headers and parameters

Multiple filters can be configured in a chain, with each filter processing the request/response before passing it to the next.

In short : Filters are pluggable classes that stand between the client and a target component (like a servlet or JSP), within a web application. We can do pre or post processing of request/response data while it is coming from client to a servlet or from the servlet back to a client.

Filter lifecycle:

Filter lifecycle is similar to servlet life cycle. Service method in servlet life cycle is similar to doFilter() method in filter life cycle which gets executed for every request.

1. init(FilterConfig config): Called once when the filter is initialized.
2. doFilter(ServletRequest request, ServletResponse response, FilterChain chain): Called for each request/response pair.
3. destroy(): Called when the filter is destroyed.

Creating Simple Filter

```
import javax.servlet.*;
import java.io.IOException;
@WebFilter("/*")
public class LoggingFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        // Initialization code
    }
```

Creating Simple Filter

```
import javax.servlet.*;
import java.io.IOException;
@WebFilter("/*")
public class LoggingFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        // Initialization code
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException,
ServletException {
        // Log request information
        System.out.println("Request URI: " + request.getRequestURI());
        // Pass request to the next filter or servlet
        chain.doFilter(request, response);
        // Log response information
        System.out.println("Response status: " + response.getStatus());
    }
    @Override
    public void destroy() {
        // Cleanup code
    }
}
```

Example of a Simple Servlet:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;
@WebServlet("/simple")
public class SimpleServlet extends HttpServlet {
```

Example of a Simple Servlet:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;
@WebServlet("/simple")
public class SimpleServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
      response.setContentType("text/html");
      response.getWriter().println("<h1>Hello from SimpleServlet!</h1>");
    }
}
```

Deploy the servlet and filter:

Place the compiled classes in the appropriate directory in your web application. Now, the LoggingFilter will intercept all requests to your application, logging request and response information before passing them to the intended servlets or resources.

Assignments

1. Display session id on the web page. Provide refresh link on the page to verify whether every refresh should modify session id.(Check by enabling and disabling cookies)
2. In the above shopping application, display <welcome firstname lastname> message for the user when logged in successfully. Name should come from the database
3. Use bootstrap for login page
4. Generate log information for the user login and logout time

   when user succesfully logs in - make entry in the table logs having following fields

   logs - id - Pk, AI
   - userid - FK
   - login - current date time
   - logout - null

   logout time will be updated when user logs out successfully

Assignments

1. Display session id on the web page. Provide refresh link on the page to verify whether every refresh should modify session id.(Check by enabling and disabling cookies)
2. In the above shopping application, display <welcome firstname lastname> message for the user when logged in successfully. Name should come from the database
3. Use bootstrap for login page
4. Generate log information for the user login and logout time

   when user succesfully logs in - make entry in the table logs having following fields

   logs - id - Pk, AI
   - userid - FK
   - login - current date time
   - logout - null

   logout time will be updated when user logs out successfully
5. In the shoping application add the following features :
   a. Allow uesr to select the product of their choice and add the selected product in the cart.
   b. Allow user to view the selected products in the cart.
   c. When user confirms the cart, insert shopping summary in the database and display the message to the user. Display generated orderid to the user for tracking.
   d. Provide a logout link so that user can logs out from the application
   e. Allow the user to delete selected product from the cart.
   f. Update the logout time for the user for the entry in table which is made at the of login
6. Create a simple filter which logs the information about the request like URL, date and time of the request, user agent.
7. In the quiz application add the following features
   a. Show one question on one page
   b. Save the selected answer in session variables
   c. When no of questions are over, forward the request for showing the result
   d. Insert the result summary into the database
   e. Provide the option for logout
8. In the shoping application add the following features :
   a. Create admin user and create home page for admin user which shows the links for adding category and adding product.
   b. Implement add category feature by accepting all the required information

logs - id - Pk, AI

- userid - FK
- login - current date time
- logout - null

logout time will be updated when user logs out successfully

5. In the shoping application add the following features :
   a. Allow uesr to select the product of their choice and add the selected product in the cart.
   b. Allow user to view the selected products in the cart.
   c. When user confirms the cart, insert shopping summary in the database and display the message to the user. Display generated orderid to the user for tracking.
   d. Provide a logout link so that user can logs out from the application
   e. Allow the user to delete selected product from the cart.
   f. Update the logout time for the user for the entry in table which is made at the of login
6. Create a simple filter which logs the information about the request like URL, date and time of the request, user agent.
7. In the quiz application add the following features
   a. Show one question on one page
   b. Save the selected answer in session variables
   c. When no of questions are over, forward the request for showing the result
   d. Insert the result summary into the database
   e. Provide the option for logout
8. In the shoping application add the following features :
   a. Create admin user and create home page for admin user which shows the links for adding category and adding product.
   b. Implement add category feature by accepting all the required information
   c. Implement add product feature by accepting all the required information