

# ADVANCED JAVA PROGRAMMING

UNIT-4

## COOKIES SESSIONS TRACKING

# Why Session Tracking?



## **HTTP Basics**

: It's "stateless" – each page request is alone, no memory of past visits.  
Like calling a friend who forgets your name every time!

## **Need for Tracking**

: Save user info across pages, like login or cart items.  
Book Point: Sessions collect info from many browser-server chats.

**Real-Time Example** : Online store – add pen to cart, browse notes – cart remembers pen without re-ask.

## **Solutions**

: Cookies (client notes) and Sessions (server memory).

## **Key Fact**

: Both use Java Servlets for web apps.

# What are Cookies?



**Simple Definition:** Cookie is a small text file on user's browser – holds state info like name.

Encapsulates cookie; stored on client for tracking activities.

**Why Valuable?** : Remembers user without re-entry.

**Example :** Online store saves name/address – no typing each visit.

**How Created?** : Servlet uses addCookie() in HttpServletResponse – adds to HTTP response header

**Stored Info :**

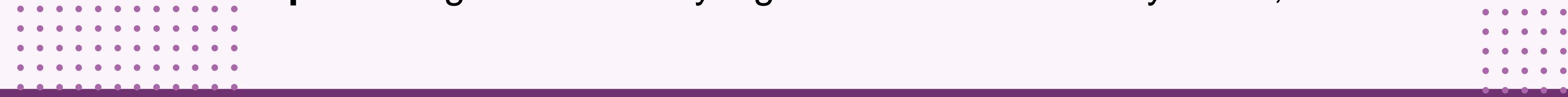
Name of cookie.

Value of cookie.

Expiration date (when deleted; default: end of browser session).

Domain and path (when cookie is sent back in requests).

**Real-Time Example:** Google saves "Stay signed in" – cookie with your ID, sent back on next visit.



# Cookie Types and Lifespan



## Types Based on Time:

**Session Cookies:** Short-lived – gone when browser closes (no expiration set).

Book: If no date, deleted at session end.

**Persistent Cookies:** Set expiration – stay till date (e.g., 1 month).

## Domain and Path Role:

Domain: Matches site (e.g., .amazon.com).

Path: Specific folder (e.g., /shop). Cookie sent only if URL matches.

**Real-Time Example:** Flipkart cookie for cart (session type during shop); address save (persistent for next buy).



# Cookie Class in Java (Constructor and Methods)



## Constructor:

- `Cookie(String name, String value)` – Sets name and value right away.
- **Example:** `Cookie user = new Cookie("userID", "12345");`

## Key Methods Table (Getters):

**Real-Time Tip:** Use `getName()` to check if it's your cookie, like finding your luggage tag.

Method	Description
<code>Object clone()</code>	Returns a copy of this object.
<code>String getComment()</code>	Returns the comment.
<code>String getDomain()</code>	Returns the domain.
<code>int getMaxAge()</code>	Returns max age in seconds.
<code>String getName()</code>	Returns the name.
<code>String getPath()</code>	Returns the path.
<code>boolean getSecure()</code>	True if secure, else false.
<code>String getValue()</code>	Returns the value.
<code>int getVersion()</code>	Returns the version.
<code>boolean isHttpOnly()</code>	True if HttpOnly attribute set.

# Cookie Class Setters (Subtopic: Changing Cookie Properties)



## Setters Table :

**Why Use?** Customize for safety, like `setSecure(true)` for bank sites.

**Real-Time Example:** Set `setMaxAge(3600)` for 1-hour login cookie on news site.

Method	Description
<code>void setComment(String c)</code>	Sets the comment to c.
<code>void setDomain(String d)</code>	Sets the domain to d.
<code>void setHttpOnly(boolean httpOnly)</code>	Adds HttpOnly if true (secure from JS); removes if false.
<code>void setMaxAge(int secs)</code>	Sets max age in seconds (after which cookie deleted).
<code>void setPath(String p)</code>	Sets the path to p.
<code>void setSecure(boolean secure)</code>	Sets security flag to secure.
<code>void setValue(String v)</code>	Sets the value to v.
<code>void setVersion(int v)</code>	Sets the version to v.

# Java Code – Creating Cookies

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

public class LoginCookieServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // Get user input
        String username = request.getParameter("username");

        // Create cookie using constructor
        Cookie userCookie = new Cookie("username", username); // From book constructor

        // Set properties (using setters)
        userCookie.setMaxAge(86400); // 1 day (book: seconds till delete)
        userCookie.setPath("/");    // All paths
        userCookie.setSecure(true); // Secure flag
        userCookie.setHttpOnly(true); // No JS access

        // Send to browser (book: addCookie in response)
        response.addCookie(userCookie);

        // Response
        PrintWriter pw =
        response.getWriter();
        pw.println("Welcome, " + username
        + "! Cookie set.");
    }
}
```

# Java Session Example

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

public class SessionExample extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        HttpSession session = request.getSession();
        session.setAttribute("username", "Tom");

        out.println("Session created for user Tom!");
    }
}
```

## Advantages of Cookies

- Easy to implement.
- Can store small data.
- Works even if server restarts.

## Limitations of Cookies

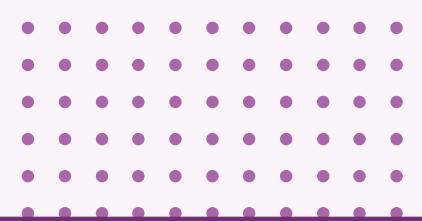
- Stored at client-side → less secure.
- Can store only small amount of data (4KB).
- Users can disable cookies in browser.

## Advantages of Sessions

- More secure (data stored on server).
- Can store large data.
- Easy to track logged-in users.

## Limitations of Sessions

- Consumes server memory.
- If too many users, performance may slow down.



# Difference Between Cookies and Sessions

Feature	Cookies	Sessions
<b>Storage Location</b>	Stored on <b>client-side</b> (browser).	Stored on <b>server-side</b> .
<b>Data Size</b>	Can store only <b>small data</b> (about 4KB).	Can store <b>large amount of data</b> (limited by server memory).
<b>Security</b>	Less secure (users can view/modify cookies).	More secure (data hidden in server).
<b>Dependency</b>	Works even without server (just client browser).	Needs server support (session is maintained by server).
<b>Lifespan</b>	Can be set with an expiry date (can live for days/months).	Ends when user logs out, browser closes, or session times out.
<b>Identification</b>	Stored as a simple <b>name-value pair</b> .	Identified using a <b>unique session ID</b> .
<b>Use Case Example</b>	Remembering theme, language, username.	Online banking, shopping cart, login sessions.
<b>Performance</b>	Does not use server memory → good for performance.	Uses server memory → heavy load if many sessions.

