

Cookies

Demo: Cookies in action

Output

Chrome

Hi! What is your name? Marco Error

Hi Marco, nice to meet you!
Delete Cookies? Yes, Delete No, Do not Error

Hi Marco, welcome back! (2)
Delete Cookies? Yes, Delete No, Do not Error

All cookies have been deleted
Go to the initial page. Error

Safari

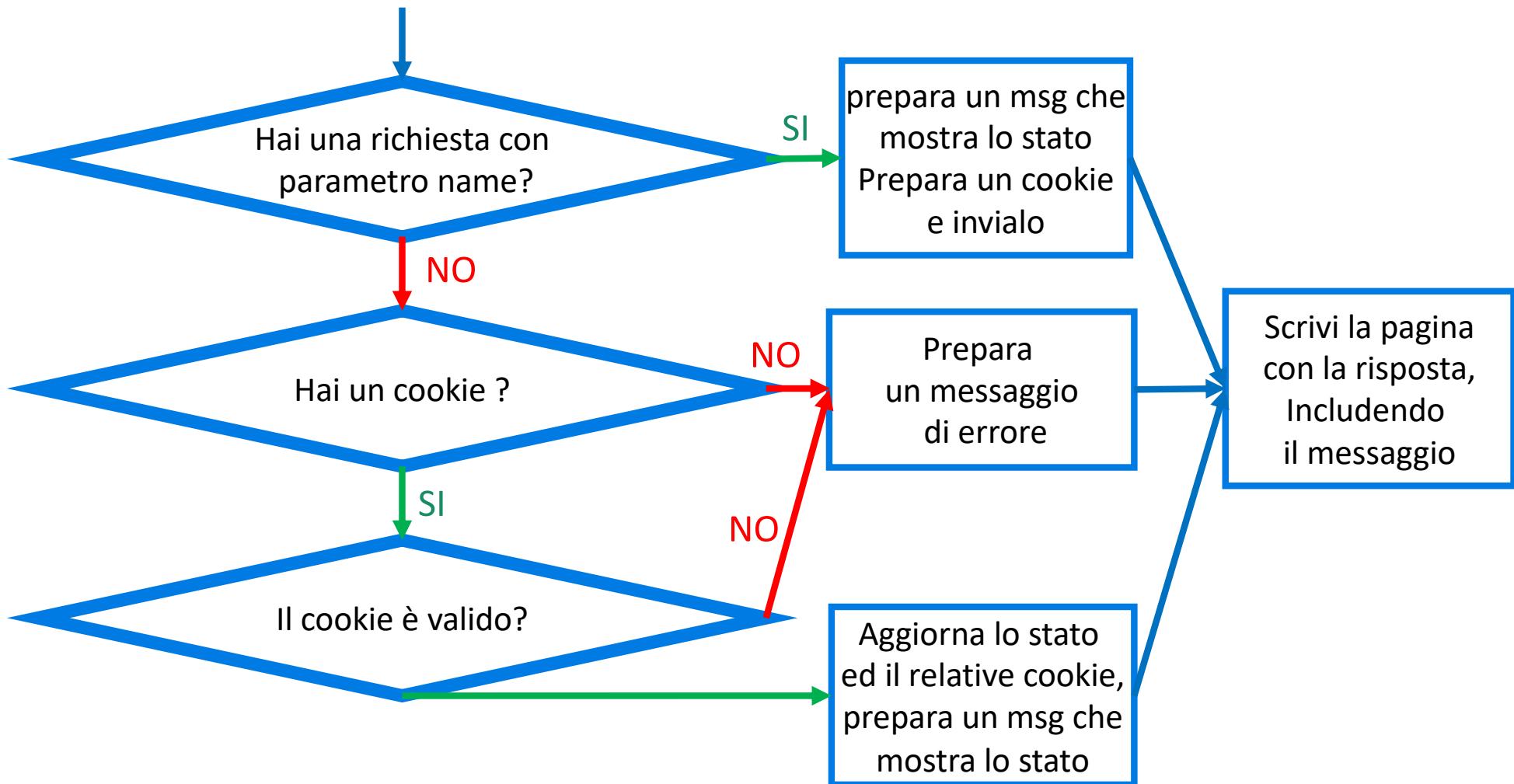
Hi! What is your name? Pietro

Hi Pietro, nice to meet you!
Delete Cookies? Yes, Delete No, Do not Error

Hi Pietro, welcome back! (5)
Delete Cookies? Yes, Delete No, Do not Error

Sorry, we do not know each other...
Please introduce yourself.
What is your name? Error

Cookies in action - outline



Cookies in action - 1

```
... Page index.html Welcome.java WhatsYourNameFragment.html DeleteCookiesFragment.html DeleteCookies.java...
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Let's meet...</title>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   </head>
8   <body>
9     <form method="GET" action="welcome">
10       <label for="name">Hi! What is your name?</label>
11       <input type="text" name="name">
12       <input type="submit" >
13     </form>
14   </body>
15 </html>
```

```
... Page index.html Welcome.java WhatsYourNameFragment.html DeleteCookiesFragment.html
1 <form method="GET" action="welcome">
2   <label for="name">What is your name?</label>
3   <input type="text" name="name" value="">
4   <input type="submit" >
5 </form>
```



Cookies in action - 2



The screenshot shows a code editor with several tabs at the top: "... Page", "index.html", "Welcome.java", "WhatsYourNameFragment.html", "DeleteCookiesFragment.html", and "DeleteCookies.java...". The "index.html" tab is active. The code in the editor is:

```
<form method="GET" action="deleteCookies">
    Delete Cookies?
    <input type="submit" value="Yes, Delete">
    <input type="submit" value="No, Do not" formaction="welcome" >
</form>
```



The screenshot shows a code editor with several tabs at the top: "...java", "WhatsYourNameFragment.html", "DeleteCookiesFragment.html", "DeleteCookies.java", and "CookiesHaveBeenDeleted.html". The "CookiesHaveBeenDeleted.html" tab is active. The code in the editor is:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cookies have been deleted</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        All cookies have been deleted <br>
        Go to the <a href="/WebAppWithCookies/welcome">initial page</a>.
    </body>
</html>
```

The left sidebar shows the project structure of "WebAppWithCookies" with files like "index.html", "DeleteCookiesF", "WhatsYourNameFragment.html", and "Welcome.java".

Cookies in action - 3

```
package it.unitn.disi.ronchet.myservlets;

import ...

@WebServlet(urlPatterns = {"welcome"})
public class Welcome extends HttpServlet {

    String msg;
    boolean isInitialIteration ;

    private void dealWithInvalidCookie() {
        msg = "Sorry, we do not know each other...<br>" +
              + "Please introduce yourself.<br>";
        isInitialIteration = true;
    }
}
```

Cookies in action - 4

```
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
    isInitialIteration=false;
    // manage params and cookies
    String name = request.getParameter("name");
    if (name != null && ! name.equals("")) {
        // there is the right parameter,
        // no need to read cookie, but we set them
        log("name != null && ! name.equals(\"\"')");
        Cookie cookie = new Cookie("name", name);
        msg = "Hi " + name + ", nice to meet you!";
        response.addCookie(cookie); // identity
        Cookie cookie1 = new Cookie("counter", "0");
        response.addCookie(cookie1); // state
        // finished! Go to end
    } else {
```

Cookies in action - 5

```
    } else {
        // no parameter, let's try with cookies
        Cookie cookies[] = request.getCookies();
        if (cookies==null || cookies.length == 0) {
            // no cookies
            log("no cookies found");
            dealWithInvalidCookie();
        } else {
            Cookie n_Cookie=null; // cookie con il nome
            Cookie c_Cookie=null; // cookie con il contatore
            for (Cookie c:cookies) {
                String cookieName = c.getName();
                if (cookieName.equals("name")) {
                    n_Cookie=c;
                } else if (cookieName.equals("counter")) {
                    c_Cookie=c;
                }
            }
            if (n_Cookie==null) {
                log ("valid cookies not found");
                // invalid cookie
                dealWithInvalidCookie();
            } else
```

Cookies in action - 6

```
    } else {
        // ok, the cookie is good!
        String userName=n_Cookie.getValue();
        String counterAsString=c_Cookie.getValue();
        log ("name == "+userName);
        // let's update the counter, and the cookie
        int counter=Integer.valueOf(counterAsString)+1;
        c_Cookie.setValue(""+counter);
        response.addCookie(c_Cookie);
        msg = "Hi " + userName + ", welcome back! (" +
              +counter+"); }
```

}

}

Cookies in action - 7

```
// prepare response and send it
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html><body>");
        out.println(msg);
        if (isInitialIteration) {
            request.getRequestDispatcher(
                "WhatsYourNameFragment.html")
                .include(request, response);
        } else {
            request.getRequestDispatcher(
                "DeleteCookiesFragment.html")
                .include(request, response);
        }
        out.println("</body></html>");
    }
}
```



Cookies in action – 8 - deleteCookies

```
@WebServlet(name = "DeleteCookies",
    urlPatterns = {" /deleteCookies"})
public class DeleteCookies extends HttpServlet {

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        Cookie cookies[] = request.getCookies();
        if (cookies != null) {
            for (Cookie c : cookies) {
                c.setMaxAge(0);
                response.addCookie(c);
            }
        }
        response.setContentType("text/html; charset=UTF-8");
        request.getRequestDispatcher(
            "CookiesHaveBeenDeleted.html")
            .include(request, response);
    }
}
```



Cookies in PHP

https://www.w3schools.com/php/php_cookies.asp



Cookies: legal aspects

Italian regulations

Individuazione delle modalità semplificate per l'informativa e l'acquisizione del consenso per l'uso dei cookie - 8 maggio 2014

- <http://garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/3118884>
- <https://www.garanteprivacy.it/home/doveri>
- https://it.wikipedia.org/wiki/Regolamento GENERALE SULLA protezione_dei_dati



Italian regulations

- a. **Cookie tecnici.**

- I cookie tecnici sono quelli utilizzati al solo fine di "effettuare la trasmissione di una comunicazione su una rete di comunicazione elettronica, o nella misura strettamente necessaria al fornitore di un servizio della società dell'informazione esplicitamente richiesto dall'abbonato o dall'utente a erogare tale servizio" (cfr. art. 122, comma 1, del Codice).



Italian regulations

- Essi non vengono utilizzati per scopi ulteriori e sono normalmente installati direttamente dal titolare o gestore del sito web. Possono essere suddivisi in
- **cookie di navigazione o di sessione**, che garantiscono la normale navigazione e fruizione del sito;
- **cookie analytics**, assimilati ai cookie tecnici laddove utilizzati direttamente dal gestore del sito per raccogliere informazioni, in forma aggregata, sul numero degli utenti e su come questi visitano il sito stesso;



Italian regulations

- **cookie di funzionalità**, che permettono all'utente la navigazione in funzione di una serie di criteri selezionati (ad esempio, la lingua, i prodotti selezionati per l'acquisto) al fine di migliorare il servizio reso allo stesso.
- Per l'installazione di tali cookie non è richiesto il preventivo consenso degli utenti, mentre resta fermo l'obbligo di dare l'informativa ai sensi dell'art. 13 del Codice, che il gestore del sito, qualora utilizzi soltanto tali dispositivi, potrà fornire con le modalità che ritiene più idonee.



Italian regulations

- b. **Cookie di profilazione.**
- I cookie di profilazione sono volti a creare profili relativi all'utente e vengono utilizzati al fine di inviare messaggi pubblicitari in linea con le preferenze manifestate dallo stesso nell'ambito della navigazione in rete. In ragione della particolare invasività che tali dispositivi possono avere nell'ambito della sfera privata degli utenti, la normativa europea e italiana prevede che l'utente debba essere adeguatamente informato sull'uso degli stessi ed esprimere così il proprio valido consenso.



Italian regulations

- 2. **Soggetti coinvolti: editori e "terze parti".**
- Un ulteriore elemento da considerare, ai fini della corretta definizione della materia in esame, è quello soggettivo. Occorre, cioè, tenere conto del differente soggetto che installa i cookie sul terminale dell'utente, a seconda che si tratti dello stesso gestore del sito che l'utente sta visitando (che può essere sinteticamente indicato come "editore") o di un sito diverso che installa cookie per il tramite del primo (c.d. "terze parti").



Italian regulations

- Nel momento in cui l'utente accede a un sito web, **deve essergli presentata una prima informativa "breve"**, contenuta in un banner a comparsa immediata sulla home page (o altra pagina tramite la quale l'utente può accedere al sito), **integrata da un'informativa "estesa"**, alla quale si accede attraverso un link cliccabile dall'utente.



Session

Introduction

Session tracking using cookies

The idea:



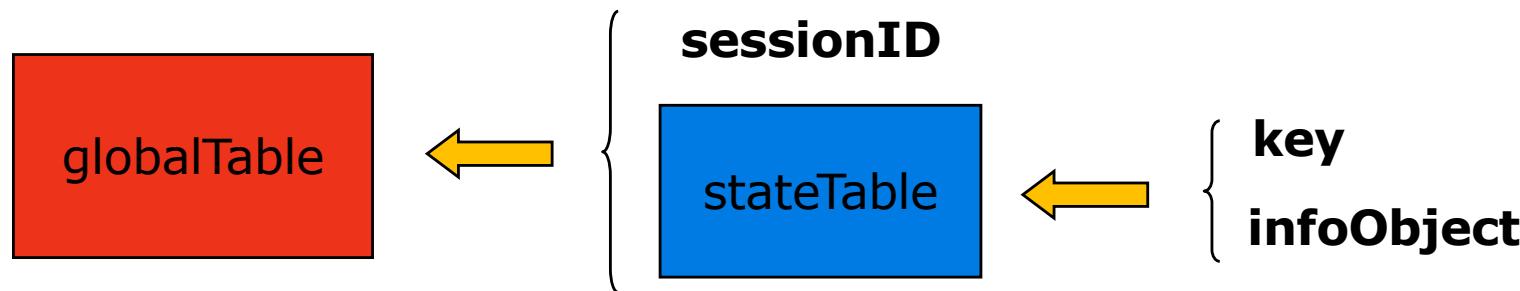
User ID	State (Map)	
	Key	Value
User 1	name	Object 1
	address	Object 2
	cart	Object 3
User 2	name	Object 1
	address	Object 2
	cart	Object 3
User 3	name	Object 1
	address	Object 2
	cart	Object 3

MEMORY



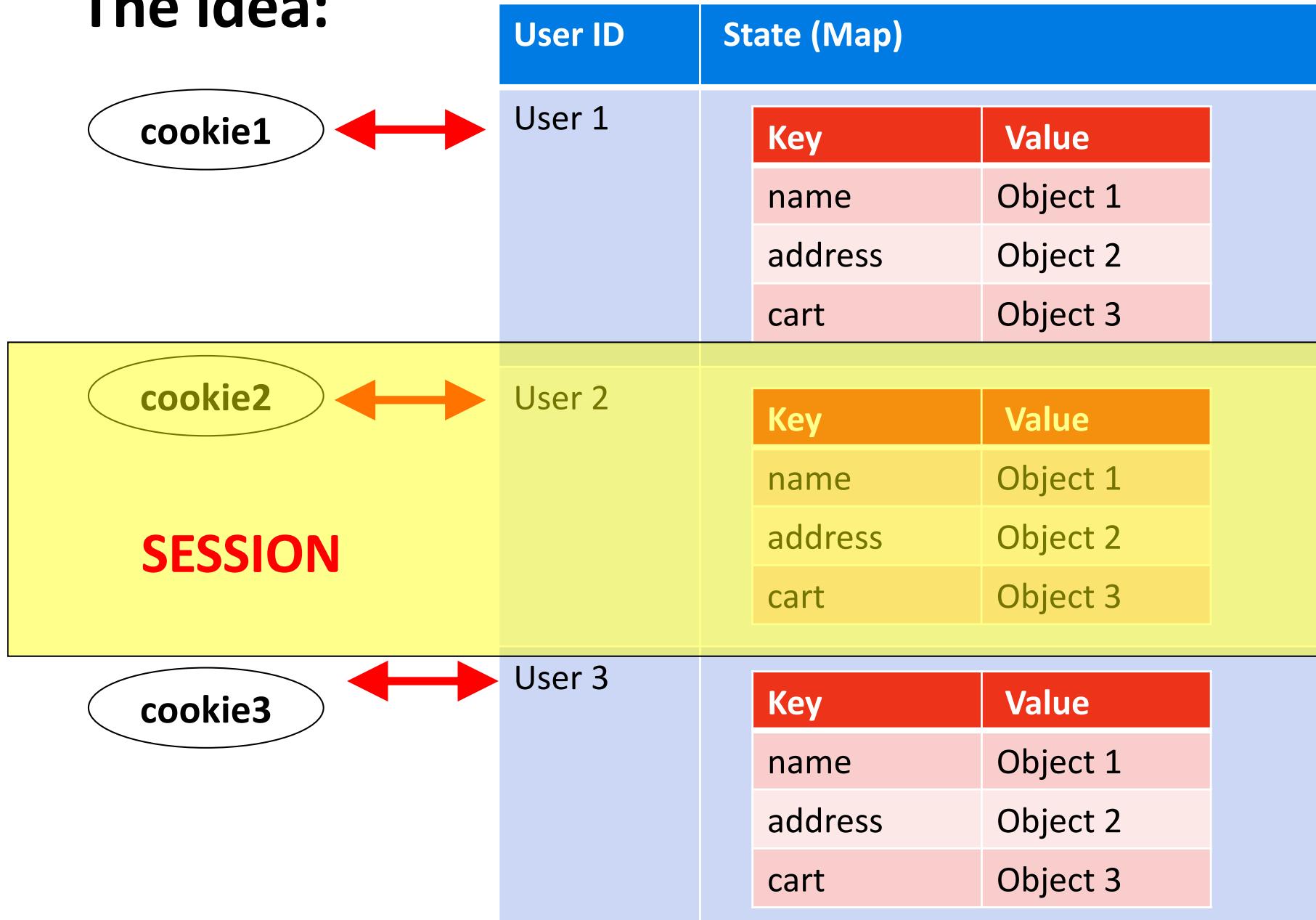
A possible implementation (pseudocode)

```
Hashtable globalTable = getGlobalTable();  
String sessionID = makeUniqueString();  
Hashtable stateTable = new Hashtable();  
globalTable.put(sessionID, stateTable);  
Cookie sessionCookie=  
    new Cookie("SessionID", sessionID);  
response.addCookie(sessionCookie);  
stateTable.put("key", infoObject);
```



Session tracking using cookies

The idea:



MEMORY



Session concept

- To support applications that need to maintain state, Java Servlet technology provides an API for managing sessions and allows several mechanisms for implementing sessions.
- Sessions are represented by an **HttpSession** object.

Session tracking

- To associate a session with a user, a web container can use several methods, all of which involve passing an identifier between the client and the server. The identifier can be
- maintained on the client as a cookie, or
- the web component can include the identifier in every URL that is returned to the client. See URL rewriting later)

The screenshot shows a web browser window. The address bar contains the URL: `esse3.unitn.it/auth/docente/RegistroDocente/ChangeStatus.do;jsessionid=D2C0C7DBB9070ACD9E94742084...`. A yellow oval highlights the part of the URL after `ChangeStatus.do;`, specifically `jsessionid=D2C0C7DBB9070ACD9E94742084...`, which represents the session identifier.

Below the browser window, the University of Trento logo and the **ESSE³** branding are visible. The **ESSE³** logo features a blue mouse cursor icon next to the text "servizi *online* per la didattica". To the right of the logo is a graphic of a laptop surrounded by various colorful icons representing different online services and applications.

The page content includes a navigation menu with links like "Elenco Registri" and "Dati Registro". The main title "Dettaglio Registro" is displayed prominently. At the bottom, there is a footer bar with the text "Attività: Introduzione alla Programmazione per il web [145325]" and the University of Trento seal.

Disallowing cookies

← → ⌂ Chrome | chrome://settings/content/cookies

Settings

Search settings

You and Google

Auto-fill

Privacy and security

Appearance

Search engine

Default browser

On start-up

Advanced

Extensions

About Chrome

Cookies and site data

Allow sites to save and read cookie data (recommended)

Clear cookies and site data when you quit Chrome

Block third-party cookies
When on, sites can't use your browsing activity across different sites to personalise ads. Some sites may not work properly.

See all cookies and site data

Add

Block

localhost

Current incognito session

⋮



Output

v. Source code

Chrome

Let's meet...

Hi! What is your name?

localhost:8084/WebAppWithCoo...

Hi Marco, nice to meet you!

Hi Marco, welcome back! (2)

Delete Cookies?

localhost:8084/WebAppWithCoo...

All cookies have been deleted

Go to the [initial page](#).

Safari

localhost:8084/WebAppWithCoo...

Hi! What is your name?

localhost:8084/WebAppWithCoo...

Hi Pietro, nice to meet you!

Hi Pietro, welcome back! (5)

Delete Cookies?

localhost:8084/WebAppWithCoo...

Sorry, we do not know each other...
Please introduce yourself.

What is your name?

Session

Java HttpSession

Accessing Session

- You access an HttpSession object by calling the **getSession** method of a request object.
- This method returns **the current session** associated with this request; or, **if the request does not have a session, this method creates one.**

Associating objects with Session

You can associate object-valued attributes with an HttpSession by name.

Such attributes are accessible by any web component that belongs to the same web context *and* is handling a request that is part of the same session.

HttpSession methods: attributes

- **public Enumeration `getAttributeNames()`**
 - Returns an Enumeration of String objects containing the names of all the objects bound to this session.
- **public Object `getAttribute(String name)`**
 - Returns the object bound with the specified name in this session, or null if no object is bound under the name.
- **public void `setAttribute(String name, Object value)`**
 - Binds an object to this session, using the name specified. If an object of the same name is already bound to the session, the object is replaced.
- **public void `removeAttribute(String name)`**
 - Removes the object bound with the specified name from this session. If the session does not have an object bound with the specified name, this method does nothing.

Session lifecycle

- A session consumes resources (memory), hence it has to be managed. Since http is stateless, there is no notion of “log out”.
- The way of solving the problem, is to decide an expiry time for sessions (timeout)

HttpSession methods: timing

- **public long getCreationTime()**
 - Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
- **public long getLastAccessedTime()**
 - Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container received the request.
- **public void setMaxInactiveInterval(int interval)**
 - Specifies the time, in seconds, between client requests before the servlet container will invalidate this session. A negative time indicates the session should never timeout.
- **public int getMaxInactiveInterval()**
 - Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses. After this interval, the servlet container will invalidate the session.



Setting Session global Timeout

To set the timeout period in the deployment descriptor using NetBeans IDE, follow these steps.

- Expand the node of your project in the **Projects** tab.
- Expand the **Web Pages** and **WEB-INF** nodes that are under the project node.
- If WebInf is empty, select it and right-click new->other->Standard Deployment Descriptor
- Double-click web.xml
- If not present, add

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

(30 is the number of minutes after which the session will expire)

web.xml

Java web applications use a deployment descriptor file named **web.xml** to determine many things, such as how URLs map to servlets, which URLs require authentication, etc..

web.xml resides in the app's WAR under the WEB-INF/ directory.

See

<https://cloud.google.com/appengine/docs/standard/java/config/webxml>

Why did we not use web.xml so far?

Some of the info expected in the web.xml can be provided via annotation. E.g.

```
package it.unitn.disi.ronchet.myservlets;  
  
@WebServlet(name="myServlet",  
            urlPatterns = {"/welcome"})  
  
public class Welcome extends HttpServlet
```

Is equivalent to

```
</web-app>  
  
    <servlet>  
        <servlet-name>myServlet</servlet-name>  
        <servlet-class>it.unitn.disi.ronchet.myservlets.Welcome  
        </servlet-class>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>myServlet</servlet-name>  
        <url-pattern>/welcome</url-pattern>  
    </servlet-mapping>  
</web-app>
```



web.xml and annotations together

**Whatever is defined in web.xml
overwrites annotations.**

Try it!

Redefine the URL via web.xml, and see who wins between annotation and configuration.

HttpSession: other methods

- **public java.lang.String getId()**
 - Returns a string containing the unique identifier assigned to this session. The identifier is assigned by the servlet container and is implementation dependent.
- **public boolean isNew()**
 - Returns true if the client does not yet know about the session or if the client chooses not to join the session (e.g., if client had disabled the use of cookies).
- **public void invalidate()**
 - Invalidates this session then unbinds any objects bound to it.

Session tracking

- If your application uses session objects, **you must ensure that session tracking is enabled by having the application rewrite URLs whenever the client turns off cookies.**
- You do this by calling the response's **encodeURL(URL)** method on **all URLs** returned by a servlet.
- This method includes the session ID in the URL only if cookies are disabled; otherwise, the method returns the URL unchanged.

Session

Demo

Session is new? true

You accessed this site 0 times in this session.

- Your session ID is B9438711FE9D0054CFAB92C7C566C791
- Session creation time is Thu Mar 26 16:18:10 CET 2020
- Session last access time is Thu Mar 26 16:18:10 CET 2020
- Session max inactive interval is 1800 seconds)

[Refresh](#)

[Refresh with URL rewriting](#)

[End Session](#)

Session is new? false

You accessed this site 1 times in this session.

- Your session ID is B9438711FE9D0054CFAB92C7C566C791
- Session creation time is Thu Mar 26 16:18:10 CET 2020
- Session last access time is Thu Mar 26 16:18:10 CET 2020
- Session max inactive interval is 1800 seconds)

[Refresh](#)

[Refresh with URL rewriting](#)

[End Session](#)



All cookies have been deleted
Go to the [initial page](#).



Session is new? true

You accessed this site 0 times in this session.

- Your session ID is DBEEBD4BA61625E08A3670773EB9650A
- Session creation time is Thu Mar 26 16:20:25 CET 2020
- Session last access time is Thu Mar 26 16:20:25 CET 2020
- Session max inactive interval is 1800 seconds)

[Refresh](#)

[Refresh with URL rewriting](#)

111

[End Session](#)

Output

Session in action - 1

```
package it.unitn.disi.ronchet.myservlets;

import ...

@WebServlet(urlPatterns = {"DemoSession"})
public class DemoSession extends HttpServlet {

    PrintWriter out=null;
    private void p(String s) {
        out.println(s);
    }
}
```

Session in action - 2

```
@Override
public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException {
    out = response.getWriter();

    // Return the existing session if there is one.
    // Create a new session otherwise.
    HttpSession session = request.getSession();
    Integer accessCount;
    synchronized(session) {
        accessCount =
            (Integer) session.getAttribute("accessCount");
        if (accessCount == null) {
            accessCount = 0;    // autobox int to Integer
        } else {
            accessCount = new Integer(accessCount + 1);
        }
        session.setAttribute("accessCount", accessCount);
    }
}
```

Session in action - 3

```
try {
    response.setContentType("text/html;charset=UTF-8");
    p("<!DOCTYPE html>
        +<html>
            +<head><title>Session Test Servlet</title></head><body>")
    p("Session is new? "+session.isNew());
    p("<h2>You accessed this site " + accessCount
        + " times in this session.</h2>");
    p("<ul><li>Your session ID is " + session.getId() + "</li>");
    p("<li>Session creation time is " +
        new Date(session.getCreationTime()) + "</li>");
    p("<li>Session last access time is " +
        new Date(session.getLastAccessedTime()) + "</li>");
    p("<li>Session max inactive interval is " +
        session.getMaxInactiveInterval() + " seconds)</li></ul>");
```

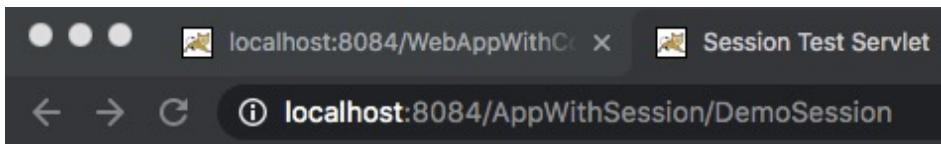


Session in action - 4

```
p("<p><a href=' " + request.getRequestURI()
    + "'>Refresh</a>") ;
p("<p><a href='"
    + response.encodeURL(request.getRequestURI())
    + "'>Refresh with URL rewriting</a>\n")
    p("<form method=\"GET\" action=\"endSession\">\n"
        +"<input type=\"submit\" value=\"End Session\">\n"
        +"</form>");
    p("</body></html>") ;
} finally {
    out.close(); // Always close the output writer
}
} // end DoGet
```



Without cookies...



You have access this site 0 times in this session.

(Session ID is ECB0CF247DD8C156A0EFE73FF9A3AA4A)

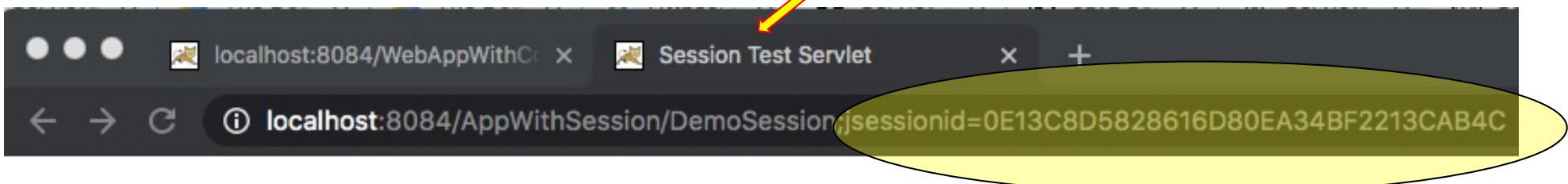
(Session creation time is Thu Mar 26 12:08:11 CET 2020)

(Session last access time is Thu Mar 26 12:08:11 CET 2020)

(Session max inactive interval is 1800 seconds)

[Refresh](#)

[Refresh with URL rewriting](#)



You have access this site 1 times in this session.

(Session ID is 0E13C8D5828616D80EA34BF2213CAB4C)

(Session creation time is Thu Mar 26 12:09:34 CET 2020)

(Session last access time is Thu Mar 26 12:09:34 CET 2020)

(Session max inactive interval is 1800 seconds)

[Refresh](#)

[Refresh with URL rewriting](#)

Session in action – 5 - endSession

```
package it.unitn.disi.ronchet.webProg;

import ...

@WebServlet(name = "endSession", urlPatterns = {" /endSession"})
public class DeleteSession extends HttpServlet {

    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {
        HttpSession s= request.getSession();
        s.invalidate();
        response.setContentType("text/html; charset=UTF-8");
        request.getRequestDispatcher("SessionHasBeenDeleted.html")
            .include(request, response);
    }
}
```



Session in action – 6 -

SessionHasBeenDeleted.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Session has been deleted</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
            initial-scale=1.0">
    </head>
    <body>
        All cookies have been deleted <br>
        Go to the <a href="/WebAppWithSession/DemoSession">
            initial page</a>.
    </body>
</html>
```

Session in action – 7 – web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
    <welcome-file-list>
        <welcome-file>DemoSession</welcome-file>
    </welcome-file-list>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```



Session

Final notes



Advanced: associating events with session objects

- Your application can notify web context and session listener objects of servlet lifecycle events ([Handling Servlet Lifecycle Events](#)). You can also notify objects of certain events related to their association with a session, such as the following:
 - When the object is added to or removed from a session. To receive this notification, your object must implement the `javax.servlet.http.HttpSessionBindingListener` interface.
 - When the session to which the object is attached will be **passivated** or **activated**. A session will be passivated or activated when it is moved between virtual machines or saved to and restored from persistent storage. To receive this notification, your object must implement the `javax.servlet.http.HttpSessionActivationListener` interface.

Sessions in PHP

https://www.w3schools.com/PHP/php_sessions.asp