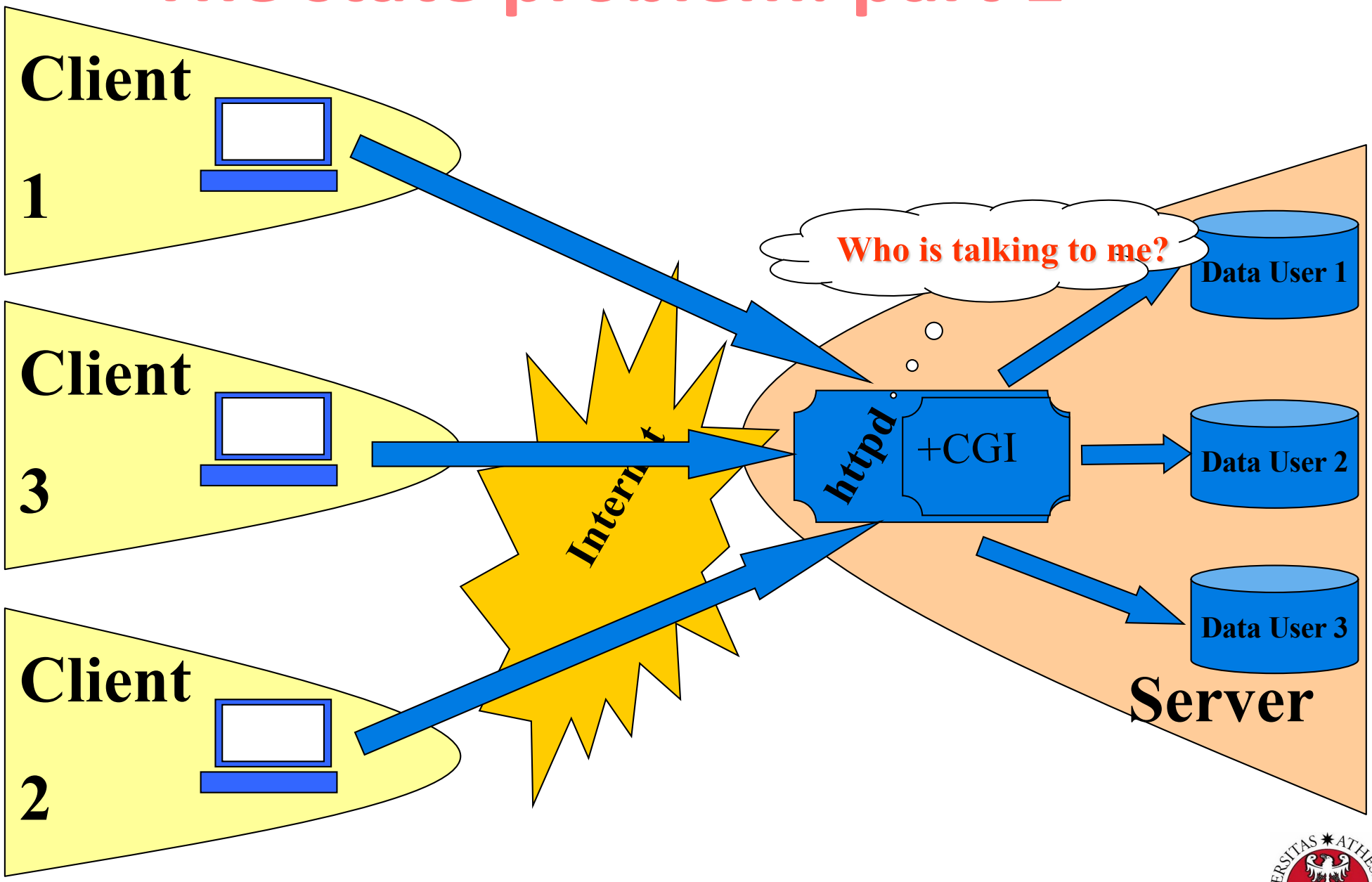
Cookies

HTTP is stateless

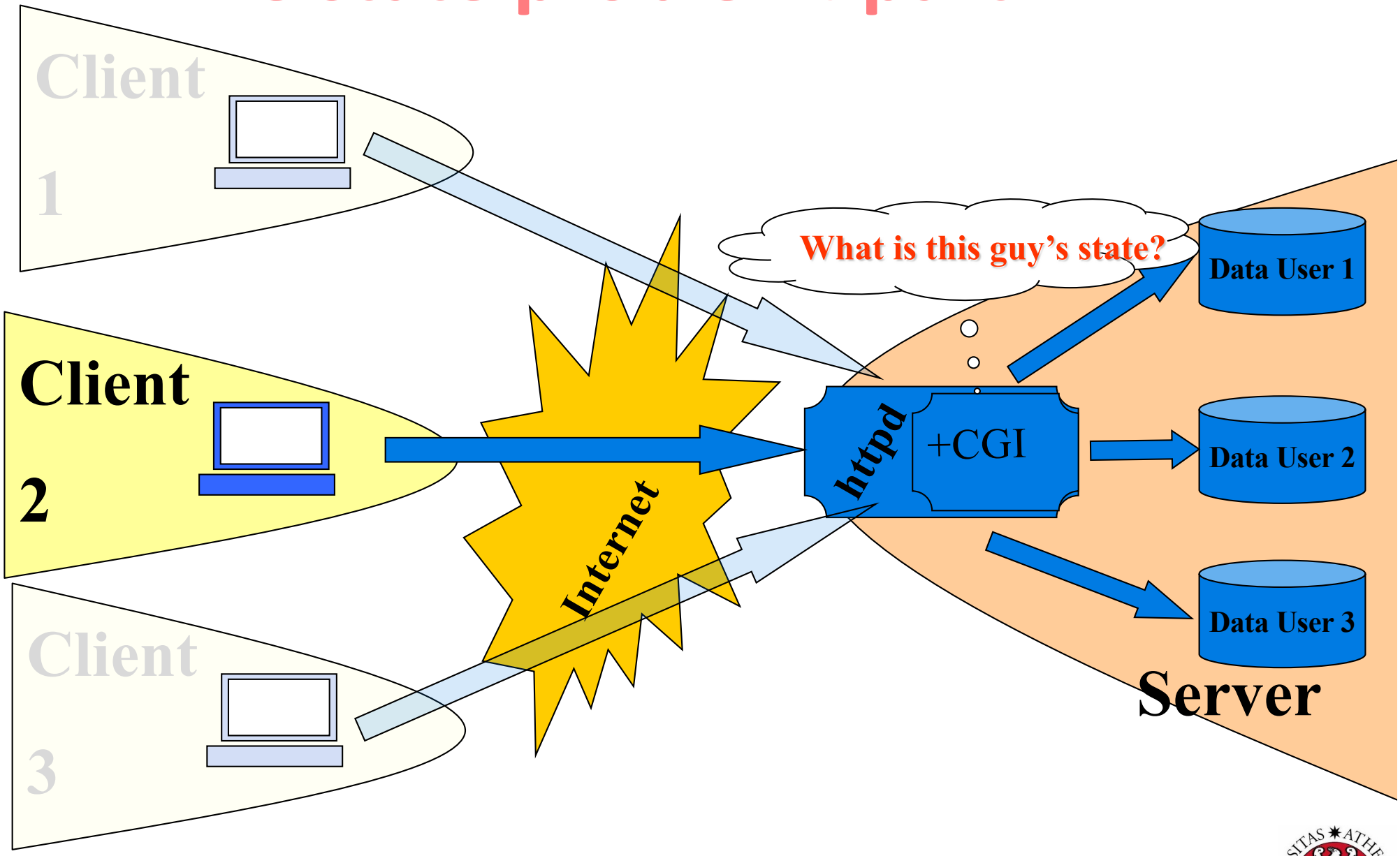
How can we keep track of who is who, and which state of the process s/he is at?

There is no way of doing it server-side…

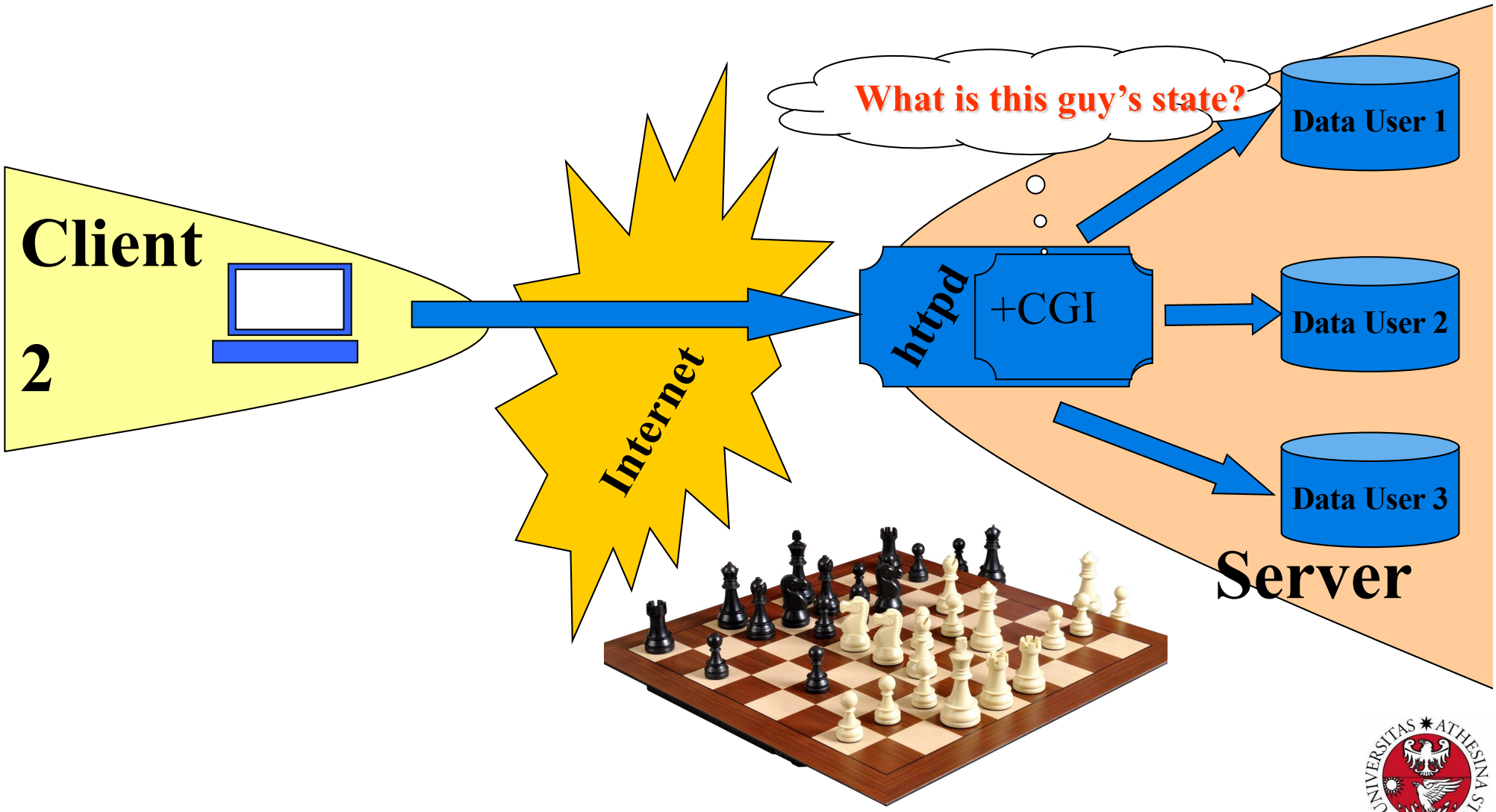
The state problem: part 1



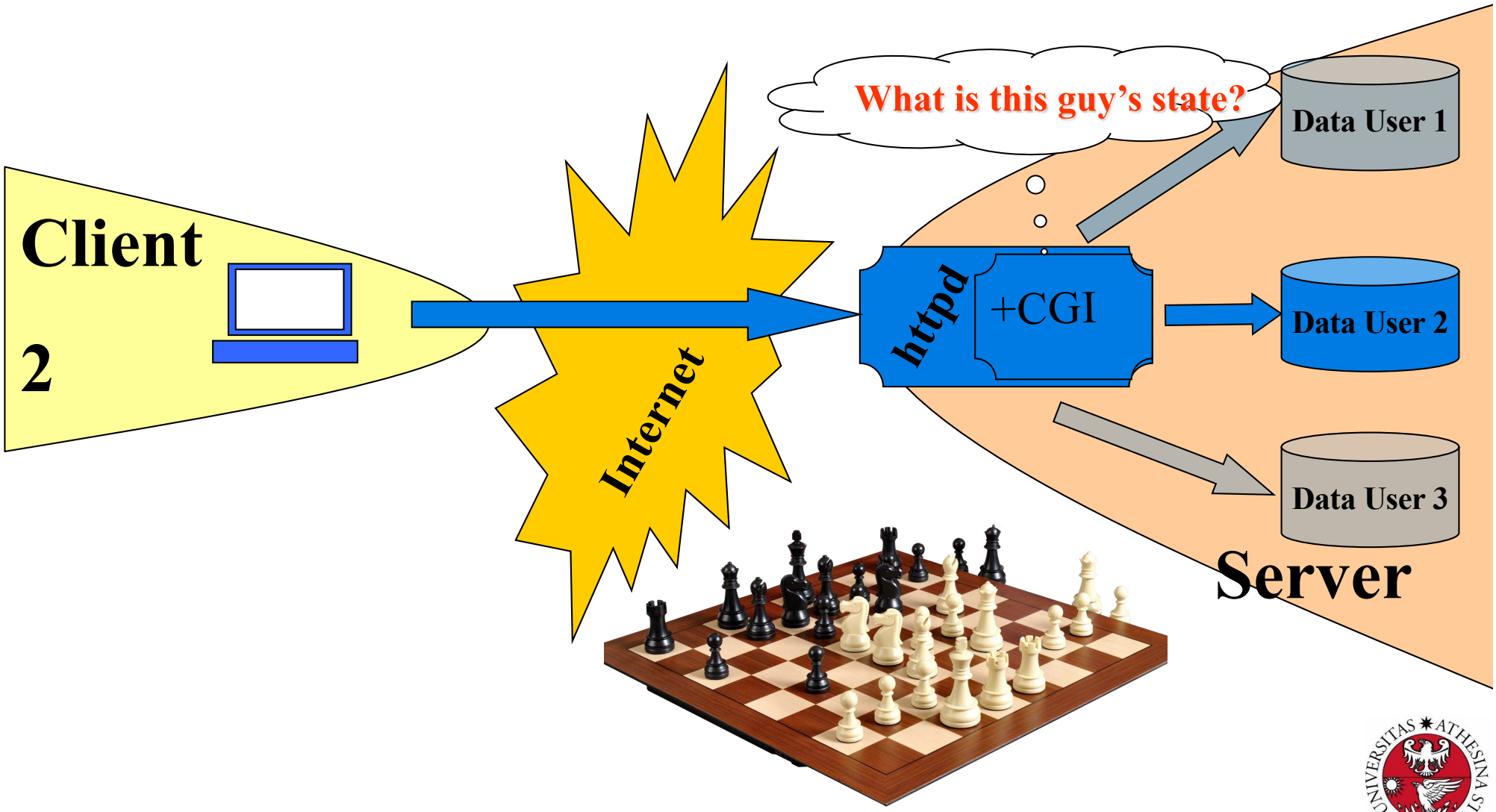
The state problem: part 2



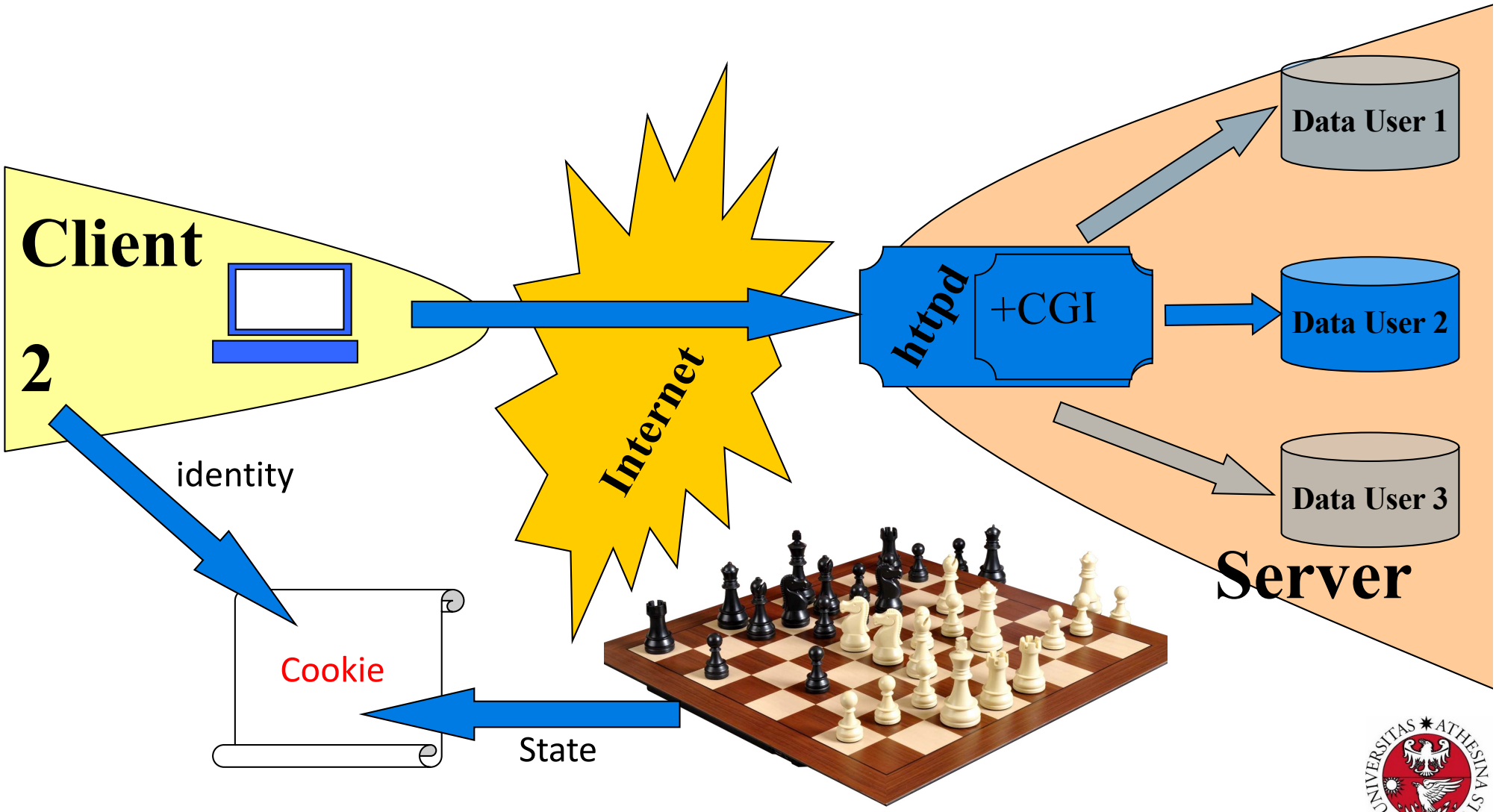
The state problem: part 2 - example



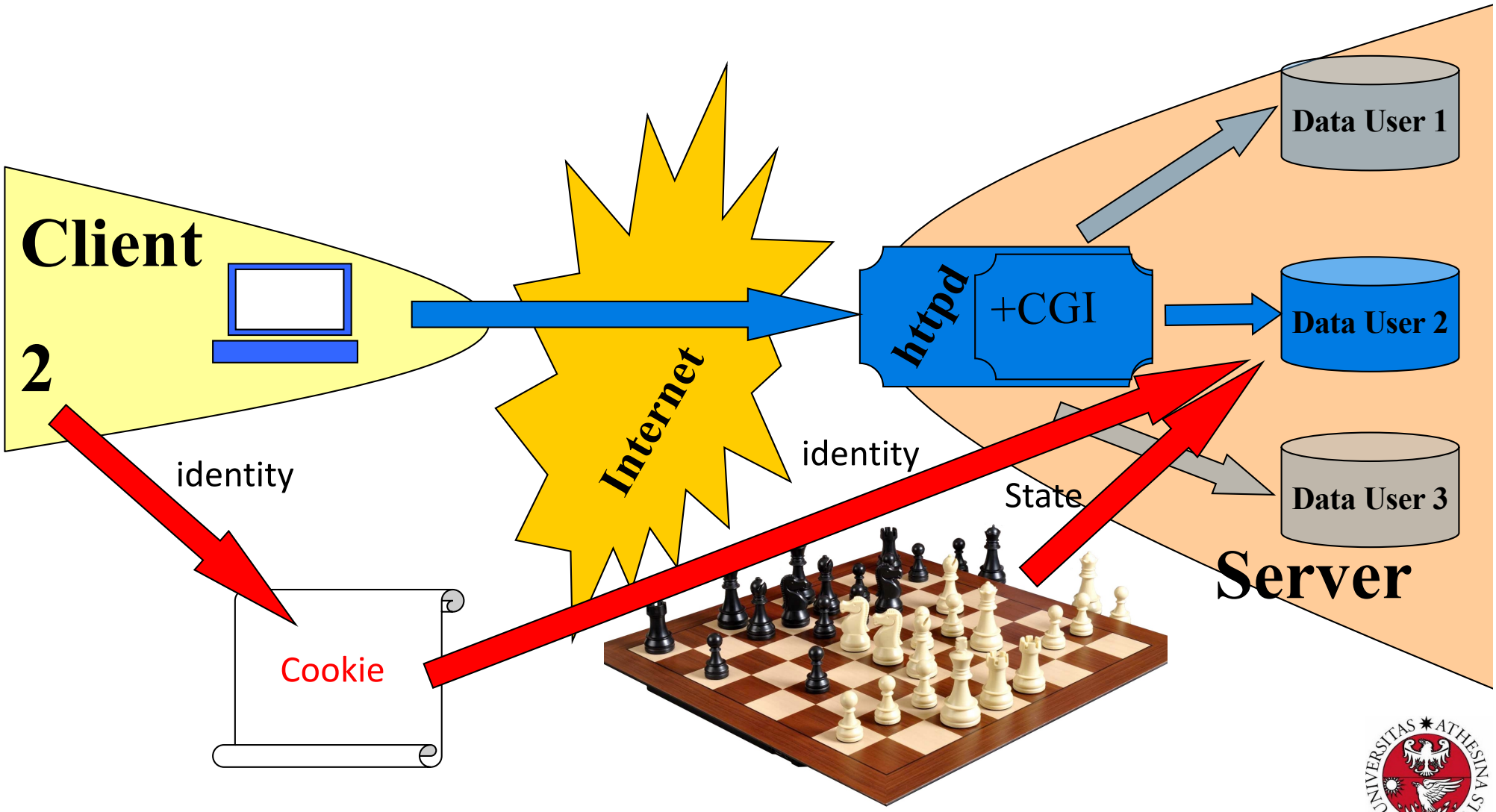
The state problem: part 2 - example



The state problem: solution 1



The state problem: solution 2



Cookies: what are they

A Cookie is a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server.

A cookie's value can uniquely identify a client, so cookies are commonly used for session management.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

Where is the name coming from…

Cookies: why?

- To maintain status across a “user session”

- To maintan infos across multiple interactions:
- Customer identification
- Targeted advertisement
- Elimination of username e password

Cookies

- The servlet sends cookies to the browser by using the HttpServletResponse.addCookie(javax.servlet.http.Cookie) method, which adds fields to HTTP response headers to send cookies to the browser, one at a time. The browser is expected to support 20 cookies for each Web server, 300 cookies total, and may limit cookie size to 4 KB each.

- The browser returns cookies to the servlet by adding fields to HTTP request headers. Cookies can be retrieved from a request by using the HttpServletRequest.getCookies() method.

Cookies and caching

- Cookies affect the caching of the Web pages that use them. HTTP 1.0 does not cache pages that use cookies created with this class.

- The Java class “Cookie” does not support the cache control defined with HTTP 1.1. This class supports both the Version 0 (by Netscape) and Version 1 (by RFC 2109) cookie specifications. By default, cookies are created using Version 0 to ensure the best interoperability

Attribute summary

- **String getComment() / void setComment(String s)**
  - **Gets/sets a comment associated with this cookie.**

- **String getDomain() / setDomain(String s)**
  - **Gets/sets the domain to which cookie applies. Normally, cookies are returned only to the exact hostname that sent them. You can use this method to instruct the browser to return them to other hosts within the same domain. Note that the domain should start with a dot (e.g. .prenhall.com), and must contain two dots for non-country domains like .com, .edu, and .gov, and three dots for country domains like .co.uk and .edu.es.**

Attribute summary

- **int getMaxAge() / void setMaxAge(int i)**

  - **Gets/sets how much time (in seconds) should elapse before the cookie expires. If you don't set this, the cookie will last only for the current session (i.e. until the user quits the browser), and will not be stored on disk. See the LongLivedCookie class below, which defines a subclass of Cookie with a maximum age automatically set one year in the future.**

- **String getName() / void setName(String s)**

  - **Gets/sets the name of the cookie. The name and the value are the two pieces you virtually always care about. Since the getCookies method of HttpServletRequest returns an array of Cookie objects, it is common to loop down this array until you have a particular name, then check the value with getValue. See the getCookieValue method shown below.**

Attribute summary

- **String getPath() / void setPath(String s)**
  - **Gets/sets the path to which this cookie applies. If you don't specify a path, the cookie is returned for all URLs in the same directory as the current page as well as all subdirectories. This method can be used to specify something more general. For example, someCookie.setPath("/") specifies that all pages on the server should receive the cookie. Note that the path specified must include the current directory.**

- **boolean getSecure / setSecure(boolean b)**
  - **Gets/sets the boolean value indicating whether the cookie should only be sent over encrypted (i.e. SSL) connections.**

Attribute summary

- String getValue() / void setValue(String s)
- Gets/sets the value associated with the cookie. Again, the name and the value are the two parts of a cookie that you almost always care about, although in a few cases a name is used as a boolean flag, and its value is ignored (i.e the existence of the name means true).

- int getVersion() / void setVersion(int i)
- Gets/sets the cookie protocol version this cookie complies with. Version 0, the default, adheres to the original Netscape specification. Version 1, not yet widely supported, adheres to RFC 2109.

Placing Cookies in the Response Headers

The cookie is added to the Set-Cookie response header by means of the addCookie method of HttpServletResponse. Here's an example:

```
  Cookie userCookie = new Cookie("user",
"uid1234");
  response.addCookie(userCookie);
// before opening the body of response!
// i.e. before any out.print
```

Reading Cookies from the Client

- To read the cookies that come back from the client, you call getCookies on the HttpServletRequest. This returns an array of Cookie objects corresponding to the values that came in on the Cookie HTTP request header.

- Once you have this array, you typically loop down it, calling getName on each Cookie until you find one matching the name you have in mind. You then call getValue on the matching Cookie, doing some processing specific to the resultant value.

Cookies: examples

- Cookie userCookie = new Cookie(“user”,”uid1234”);
- userCookie.setMaxAge(60*60*24*365);
- response.addCookie(userCookie);

Cookies

Demo: setCookies - showCookies

SetCookies - 1

Sets six cookies:
- three that apply only to the current session (regardless of how long that session lasts)
- three that persist for five minutes (regardless of whether the browser is restarted).

```
import …

public class SetCookies extends HttpServlet {

  public void doGet(HttpServletRequest request,
  HttpServletResponse response)throws ServletException,
  IOException {

    for(int i=0; i<3; i++) {

      Cookie cookie = new Cookie("Session-Cookie-" + i,
                                 "Cookie-Value-S" + i);

      response.addCookie(cookie);

      cookie = new Cookie("Persistent-Cookie-" + i,
                                 "Cookie-Value-P" + i);

      cookie.setMaxAge(3600);

      response.addCookie(cookie);

    }
```

Default maxAge is -1, indicating cookie applies only to current browsing session

Cookie is valid for an hour, regardless of whether user quits browser, reboots computer, or whatever.

SetCookies - 2

```
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Setting Cookies";
    out.println ("<HTML><HEAD><TITLE>" +title+ "</TITLE></HEAD>"
       +"<BODY BGCOLOR=\"#FDF5E6\">" +"<H1 ALIGN=\"CENTER\">"
       + title + "</H1>"
       +"There are six cookies associated with this page.\n"
       + "</BODY></HTML>");
  }
}
```

ShowCookies - 1

Creates a table of the cookies associated with the current page

```
import …;
public class ShowCookies extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
                    response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Active Cookies";
    out.println("<HTML><HEAD><TITLE>" +title+ "</TITLE></HEAD>" +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
                "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" +
                "  <TH>Cookie Name\n" + "  <TH>Cookie Value");
```

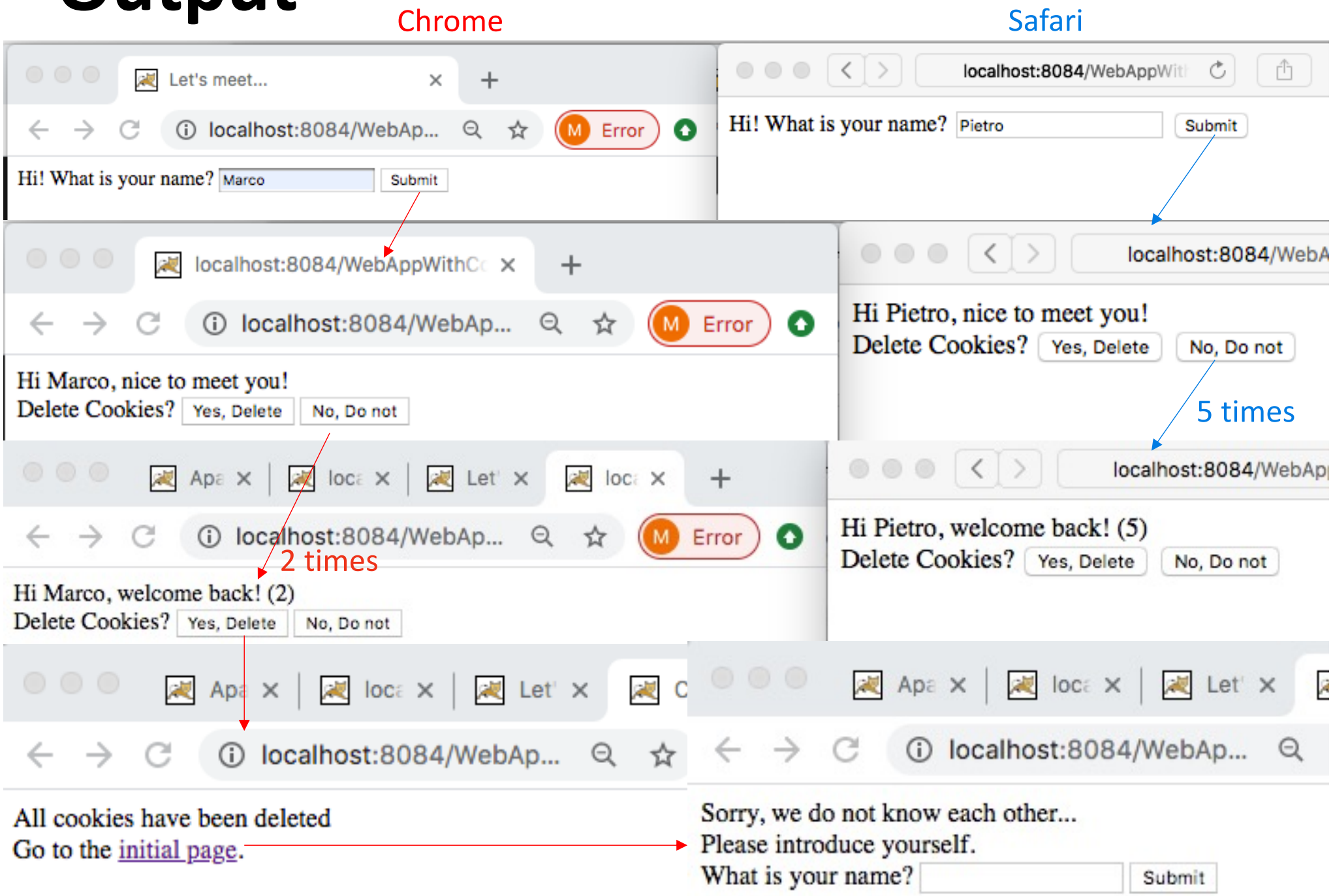
ShowCookies - 2

```
      Cookie[] cookies = request.getCookies();
      Cookie cookie;
      for(int i=0; i<cookies.length; i++) {
        cookie = cookies[i];
        out.println("<TR>\n" +
                    "  <TD>" + cookie.getName() + "\n" +
                    "  <TD>" + cookie.getValue());
      }
      out.println("</TABLE></BODY></HTML>");
    }
}
```

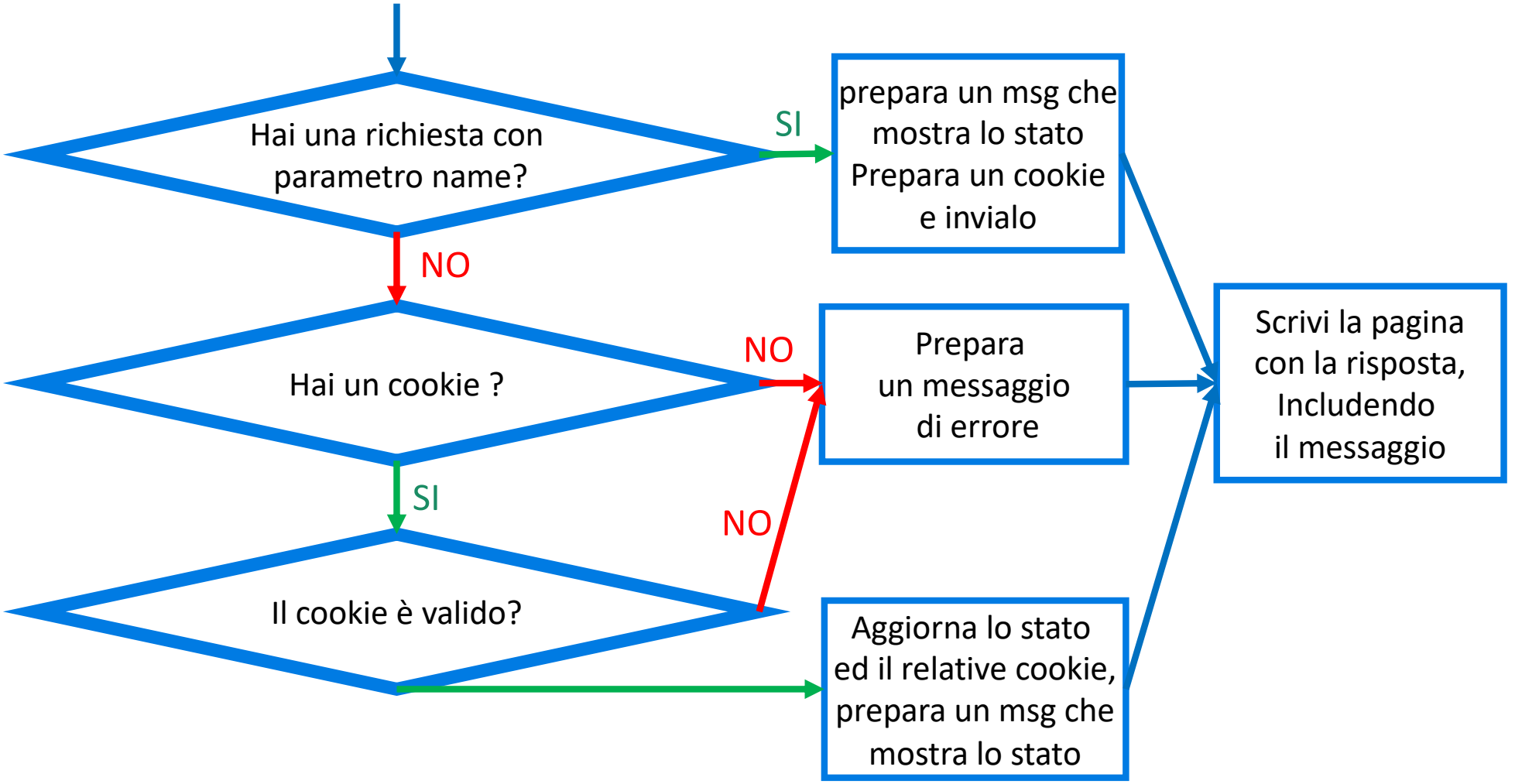
Cookies

Demo: Cookies in action

Output


Chrome
Safari
Let's meet...
localhost:8084/WebAp...
Error
Hi! What is your name? Marco
Submit
localhost:8084/WebAppWit
Hi! What is your name? Pietro
Submit
localhost:8084/WebAppWithC
Hi Marco, nice to meet you!
Delete Cookies? Yes, Delete
No, Do not
Hi Pietro, nice to meet you!
Delete Cookies? Yes, Delete
No, Do not
5 times
2 times
Hi Marco, welcome back! (2)
Delete Cookies? Yes, Delete
No, Do not
Hi Pietro, welcome back! (5)
Delete Cookies? Yes, Delete
No, Do not
All cookies have been deleted
Go to the initial page.
Sorry, we do not know each other...
Please introduce yourself.
What is your name?
Submit

Cookies in action - outline



Cookies in action - 1

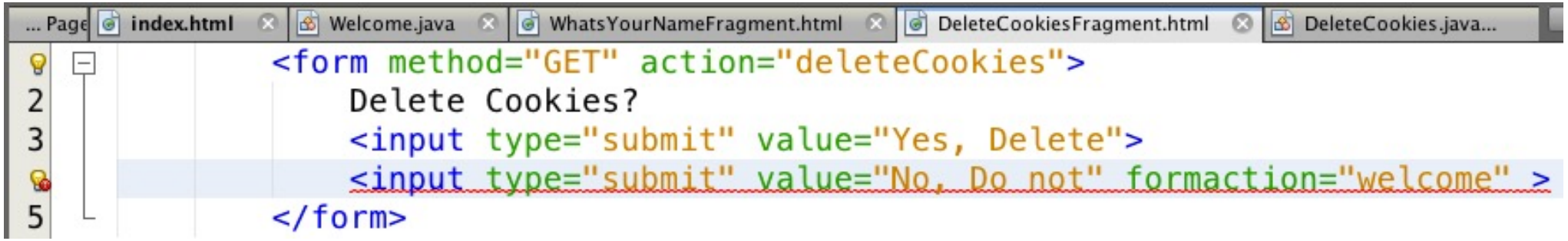
index.html | Welcome.java | WhatsYourNameFragment.html | DeleteCookiesFragment.html | DeleteCookies.java...

```
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">
           <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>
```

index.html | Welcome.java | WhatsYourNameFragment.html | DeleteCookiesFragment.html

```
<form method="GET" action="welcome">
    <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



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

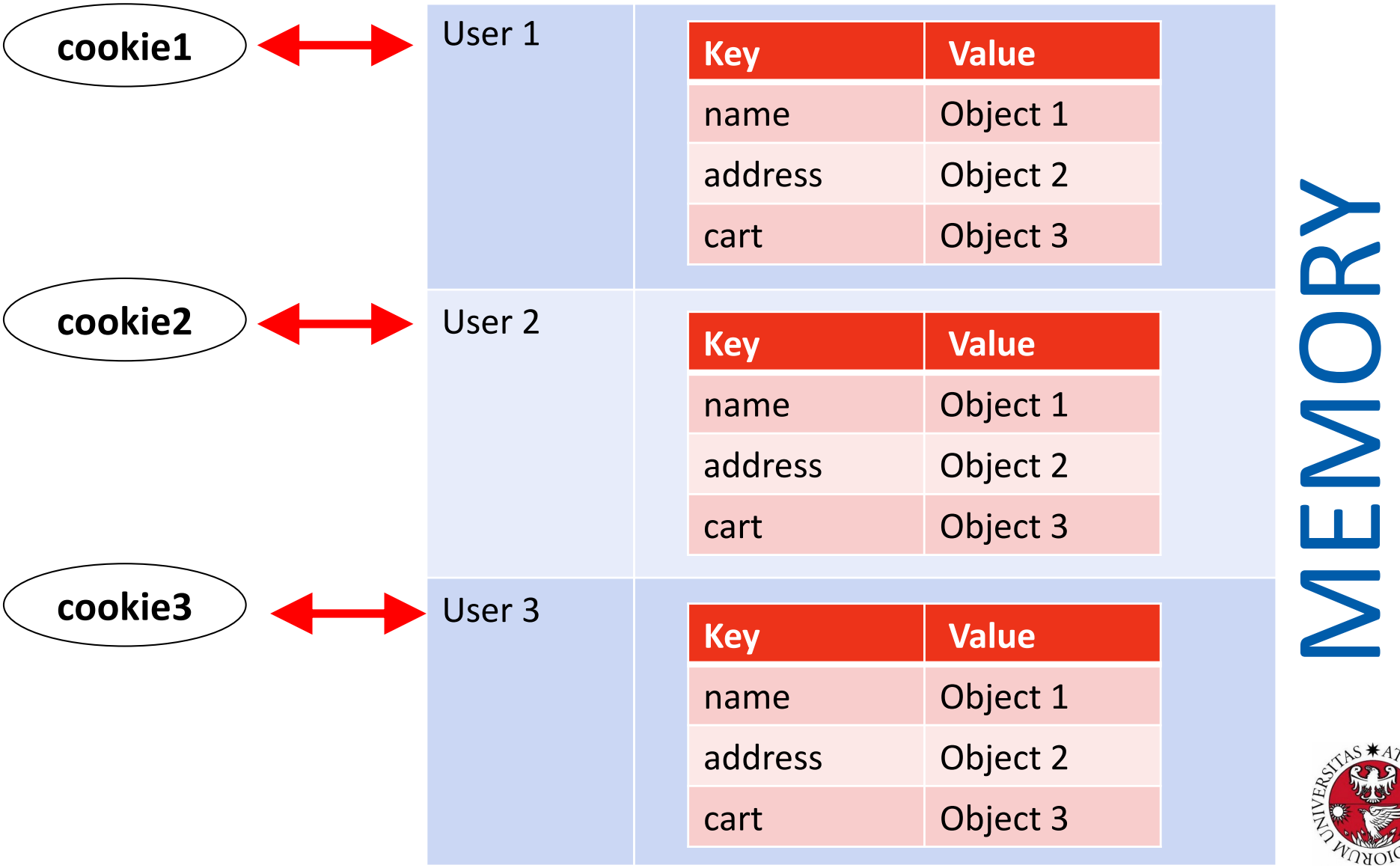
https://www.w3schools.com/php/php_cookies.asp

Session

Introduction

Session tracking using cookies

The idea:

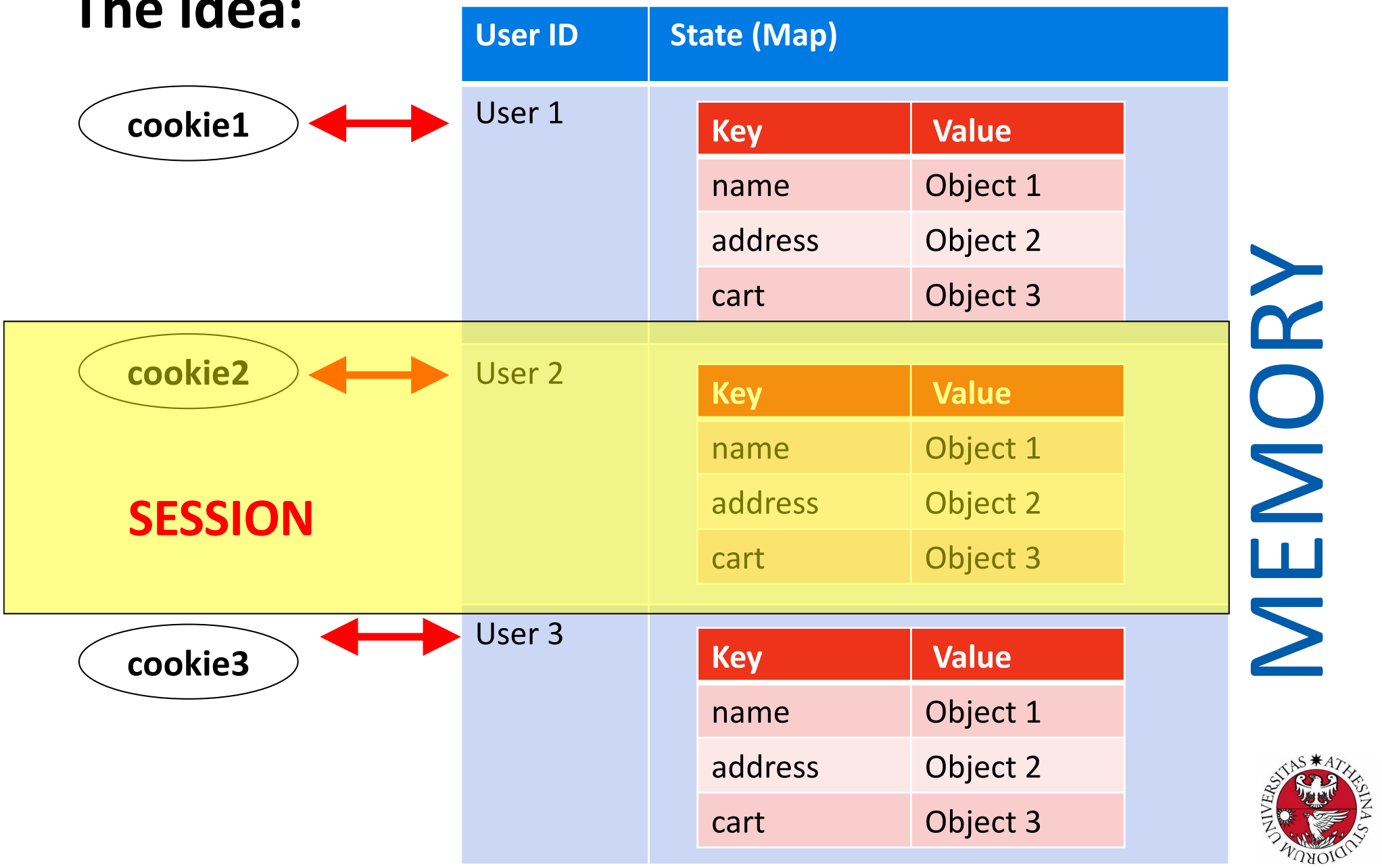


A possible implementation (pseudocode)

```
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:



Session concept

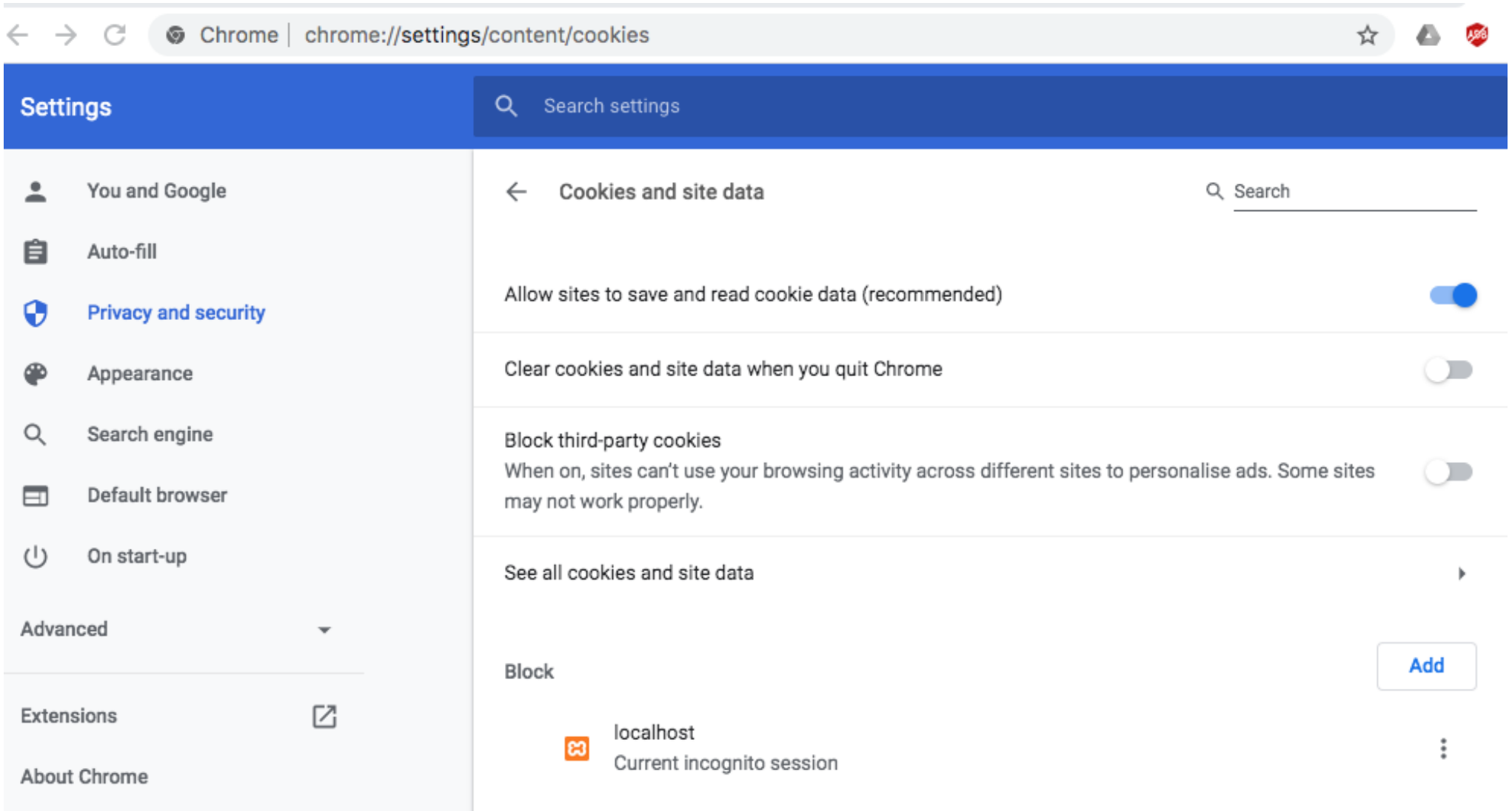
- To support applications that need to maintain state, Java Servlet technology provides an API for managing sessions and allows several mechanisms for implementing sessions.

- 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)

Disallowing cookies

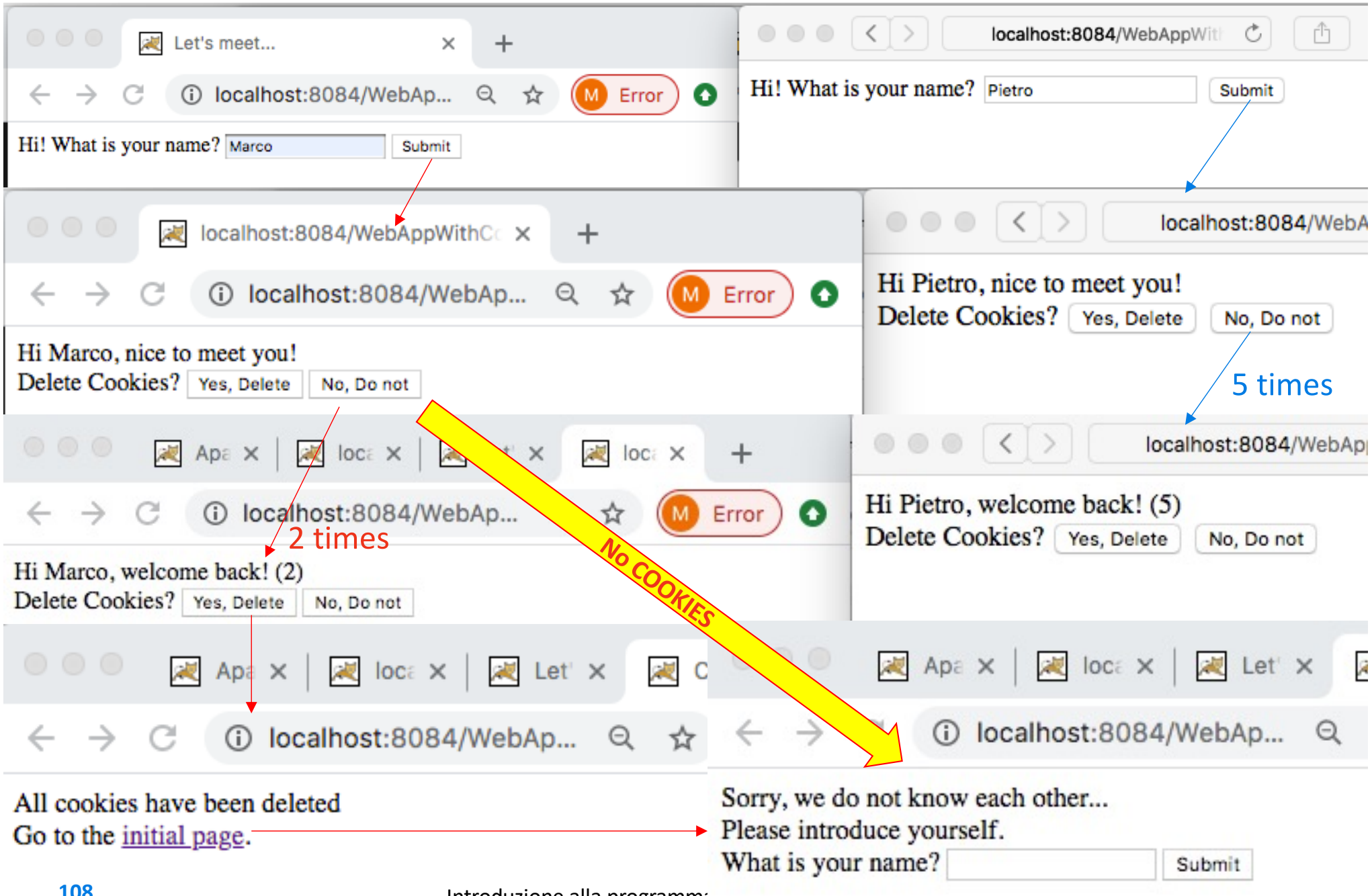


Output

v. Source code

Chrome

Safari



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 minuts 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.

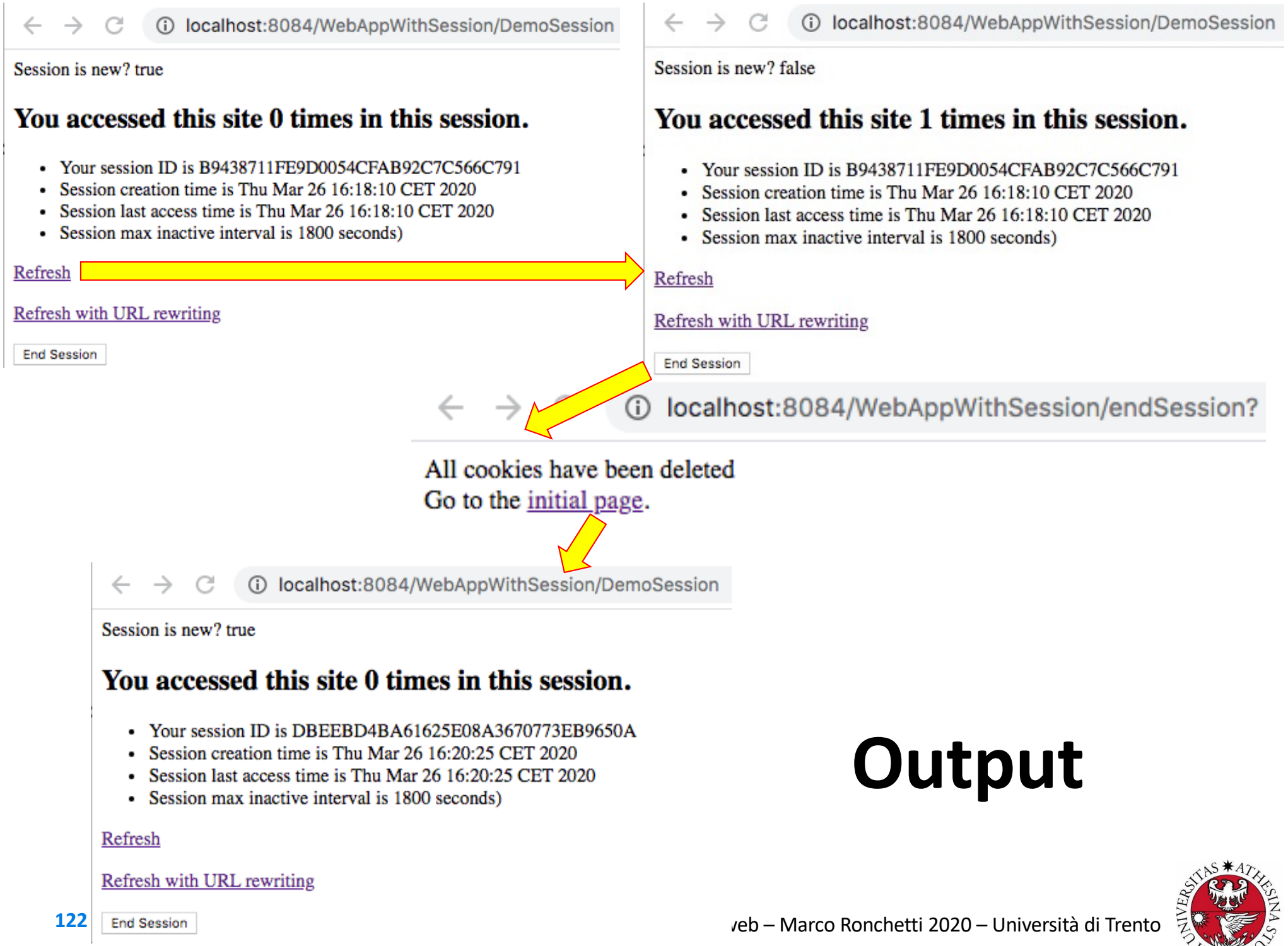
Session tracking

- 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


localhost:8084/WebAppWithSession/DemoSession
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
localhost:8084/WebAppWithSession/DemoSession
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
localhost:8084/WebAppWithSession/endSession?
All cookies have been deleted
Go to the initial page.
localhost:8084/WebAppWithSession/DemoSession
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
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);
      }
```

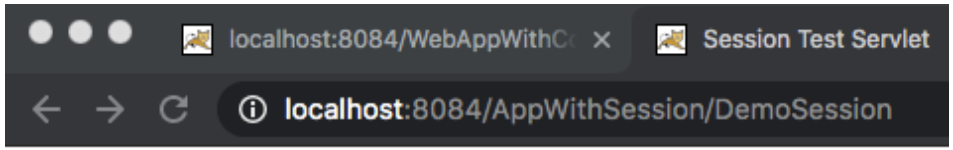

Session in action - 3

```
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)

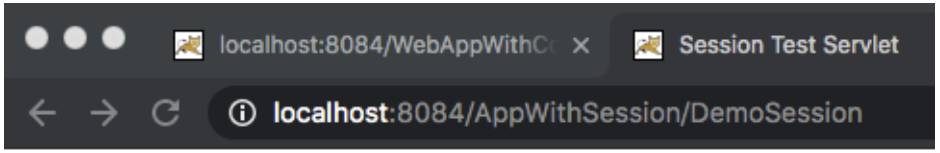
(Session creation time is Thu Mar 26 12:08:11 CET 2020)

(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 0 times in this session.

(Session ID is 0E13C8D5828616D80EA34BF2213CAB4C)

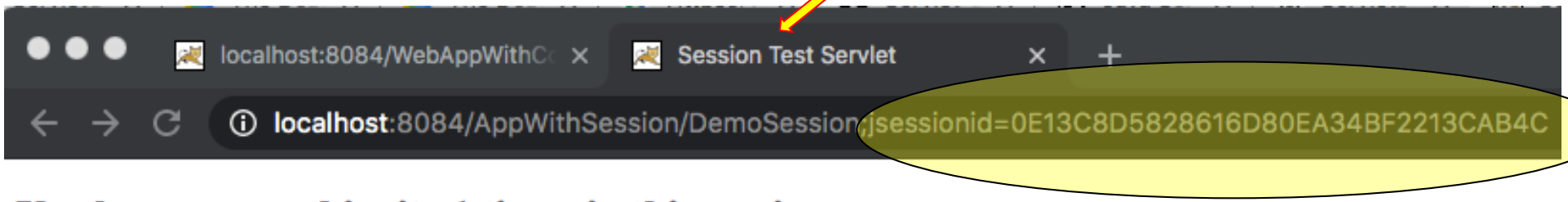
(Session creation time is Thu Mar 26 12:09:34 CET 2020)

(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



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