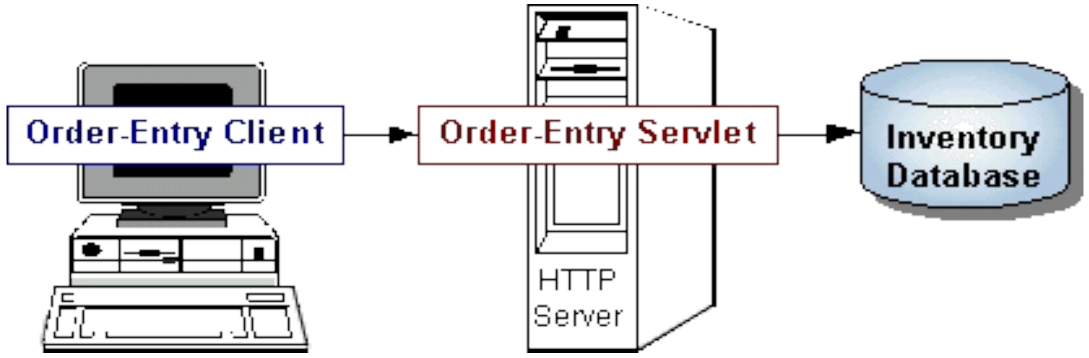
Servlets

Servlets

Servlets are modules that extend Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database.

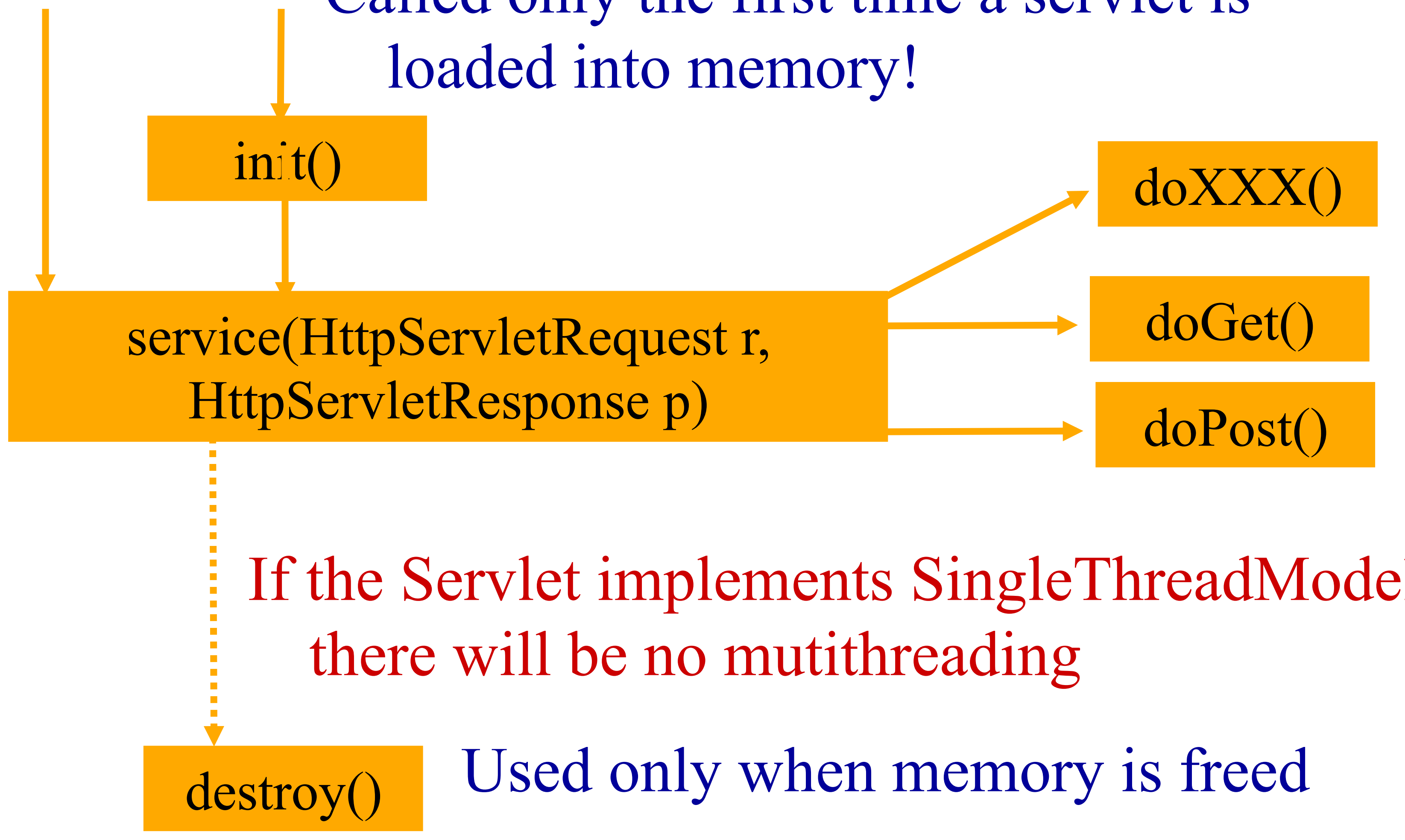


Servlets are to servers what applets are to browsers. Unlike applets, however, servlets have no graphical user interface.

For a full tutorial, see

http://docs.oracle.com/javaee/7/tutorial/doc/servlets.htm

Servlet Lifecycle



Called only the first time a servlet is loaded into memory!
init()
doXXX()
service(HttpServletRequest r, HttpServletResponse p)
doGet()
doPost()
If the Servlet implements SingleThreadModel there will be no mutithreading
destroy()
Used only when memory is freed

Get vs Post

What are "Get" and "Post"?

Get and Post are methods used to send data to the server:
With the **Get** method, the browser appends the data onto the URL.
With the **Post** method, the data is sent as "standard input.“

Why Do I Care?

It's important for you to know which method you are using. The Get method is the default, so if you do not specify a method, the Get method will be used automatically.

The Get method has several disadvantages:

- There is a limit on the number of characters which can be sent to the server, generally around 100 - 150 characters.
- Your user will see the "messy codes" when the data is sent.

service()

This code is part of the class HttpServlet

```
protected void service (HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
    {
     String method = req.getMethod ();
     if (method.equals ("GET")) {
        long       ifModifiedSince;  long       lastModified;   long     now;
        ifModifiedSince = req.getDateHeader ("If-Modified-Since");
        lastModified = getLastModified (req);
        maybeSetLastModified (resp, lastModified);
        if (ifModifiedSince == -1 || lastModified == -1) doGet (req, resp);
        else {
            now = System.currentTimeMillis ();
            if (now < ifModifiedSince || ifModifiedSince < lastModified)
                doGet (req, resp);
           else
               resp.sendError (HttpServletResponse.SC_NOT_MODIFIED);
        }
```

service()

This code is part of the class HttpServlet

```
protected void service (HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
    {
     String method = req.getMethod ();
     if (method.equals ("GET")) {
        long       ifModifiedSince;  long       lastModified;   long     now;
        ifModifiedSince = req.getDateHeader ("If-Modified-Since");
        lastModified = getLastModified (req);
        maybeSetLastModified (resp, lastModified);
        if (ifModifiedSince == -1 || lastModified == -1) doGet (req, resp);
        else {
            now = System.currentTimeMillis ();
            if (now < ifModifiedSince || ifModifiedSince < lastModified)
                doGet (req, resp);
           else
               resp.sendError (HttpServletResponse.SC_NOT_MODIFIED);
        }
```

service()

```
    } else if (method.equals ("HEAD")) {
        long        lastModified;
        lastModified = getLastModified (req);
        maybeSetLastModified (resp, lastModified);
        doHead (req, resp);
    } else if (method.equals ("POST")) {
        doPost (req, resp);
    } else if (method.equals ("PUT")) {
        doPut(req, resp);
    } else if (method.equals ("DELETE")) {
        doDelete(req, resp);
    } else if (method.equals ("OPTIONS")) {
        doOptions(req,resp);
    } else if (method.equals ("TRACE")) {
        doTrace(req,resp);
    } else {
        resp.sendError (HttpServletResponse.SC_NOT_IMPLEMENTED,
              "Method '" + method + "' is not defined in RFC 2068");
    }
}
```

A taste of servlet programming-1

```
public class SimpleServlet extends HttpServlet {
/** Handle the HTTP GET method by building a simple web page.
   */
   public void doGet (HttpServletRequest request,
                HttpServletResponse response)        throws
                     ServletException, IOException {

       PrintWriter out;
       String title = "Simple Servlet Output";
```

A taste of servlet programming-2

```
        // set content type and other response header fields first

        response.setContentType("text/html");
        // then write the data of the response
        out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>");
        out.println(title);
        out.println("</TITLE></HEAD><BODY>");
        out.println("<H1>" + title + "</H1>");
        out.println("<P>This is output from
            SimpleServlet.");
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

Forms (a quick overview)

See also:

- http://www.cs.tut.fi/~jkorpela/forms/
- http://www.w3schools.com/html/

Forms

Give to the user the possibility to di send information to the Web server

The FORM tag defines a form and has the following attributes:

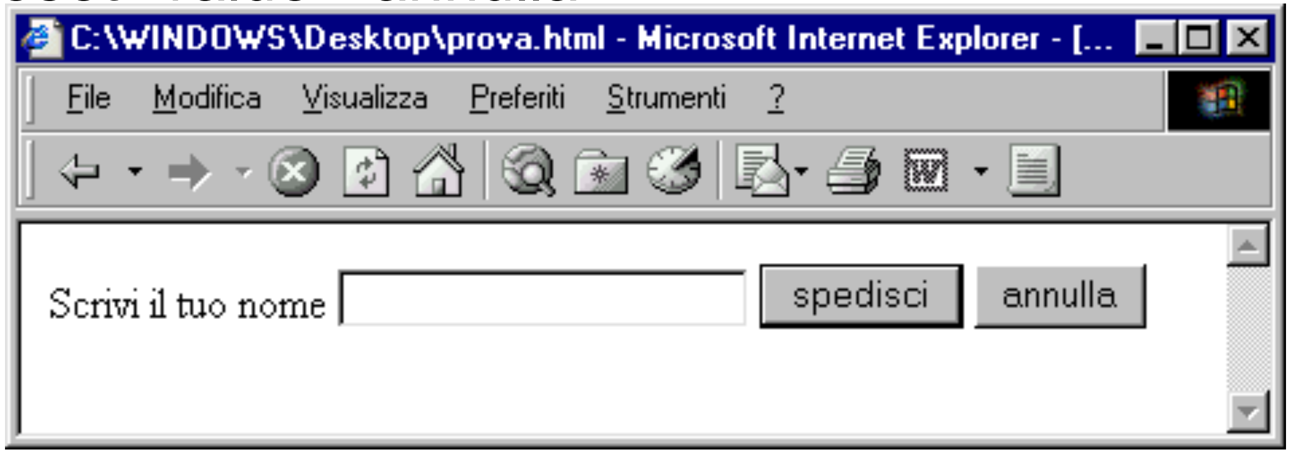
- ACTION identifies the processing engine
- ENCTYPE specificies the MIME type used to pass data to the server (Es. Text/html)

FORM contains the sub-tag:

- several tags for collecting data
- An INPUT tag must be of type SUBMIT for sending the data
- An INPUT can be of tye RESET to cancel all the gathered data

Form - input

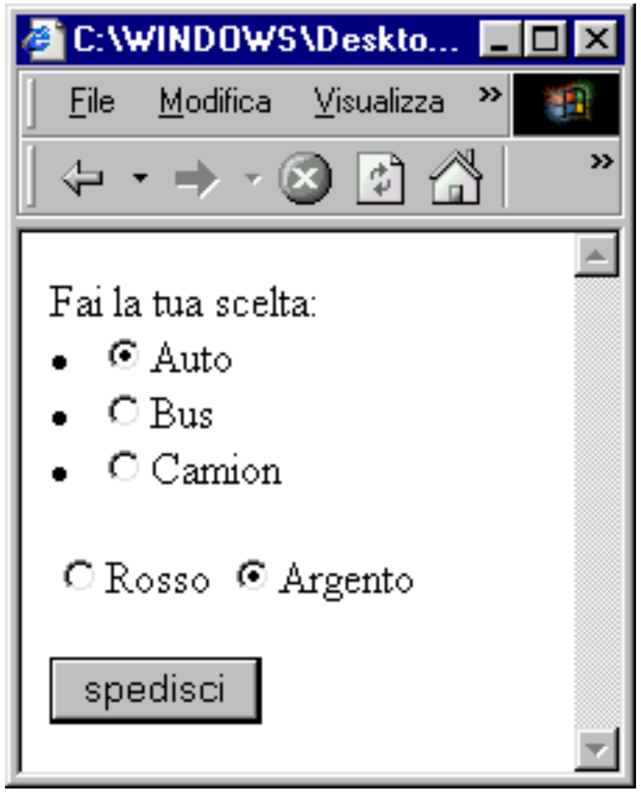
```
<FORM method="POST" action="/cgi-bin/elabora">
  Scrivi il tuo nome
  <Input type="text" size“=25" maxlength="15“ name=“a”>
  <Input type="submit" value="spedisci">
  <Input type="reset" value="annulla">
</FORM>
```


C:\WINDOWS\Desktop\prova.html - Microsoft Internet Explorer - [...
File Modifica Visualizza Preferiti Strumenti ?
Scrivi il tuo nome
spedisci
annulla

Sends a url of type
http://…/cgi-bin/elabora?a=MarcoRonchetti&b=…

Form – input type="radio"

```
<FORM method="POST" action="/cgi-bin/elabora">
  Fai la tua scelta:
  <LI><Input type="radio" name="tipo"
  value="auto" checked>Auto
  <LI><Input type="radio" name="tipo"
  value="bus">Bus
  <LI><Input type="radio" name="tipo"
  value="camion">Camion
  <P><Input type="radio" name="colore"
  value="rosso">Rosso
  <Input type="radio" name="colore"
   value="argento" checked>Argento</P>
  <Input type="submit" value="spedisci">
</FORM>
```



Fai la tua scelta:

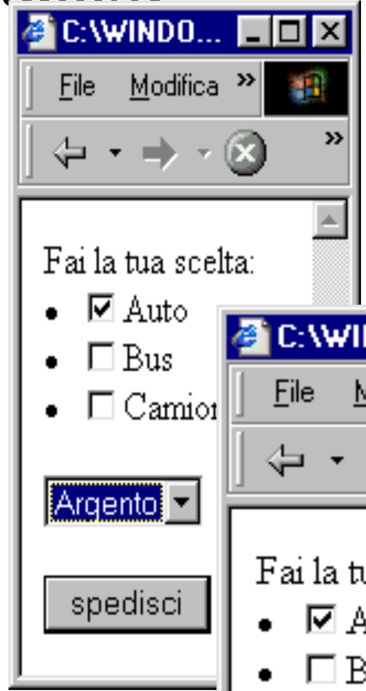
- [✓ Auto]
- Bus
- Camion

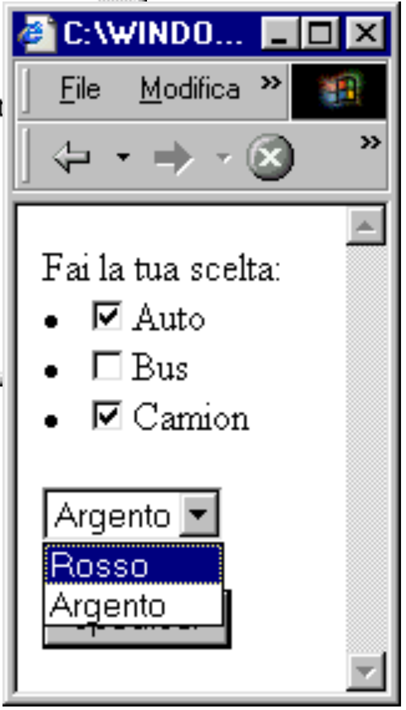
Rosso [✓ Argento]

spedisci

Form – input type="checkbox" - select

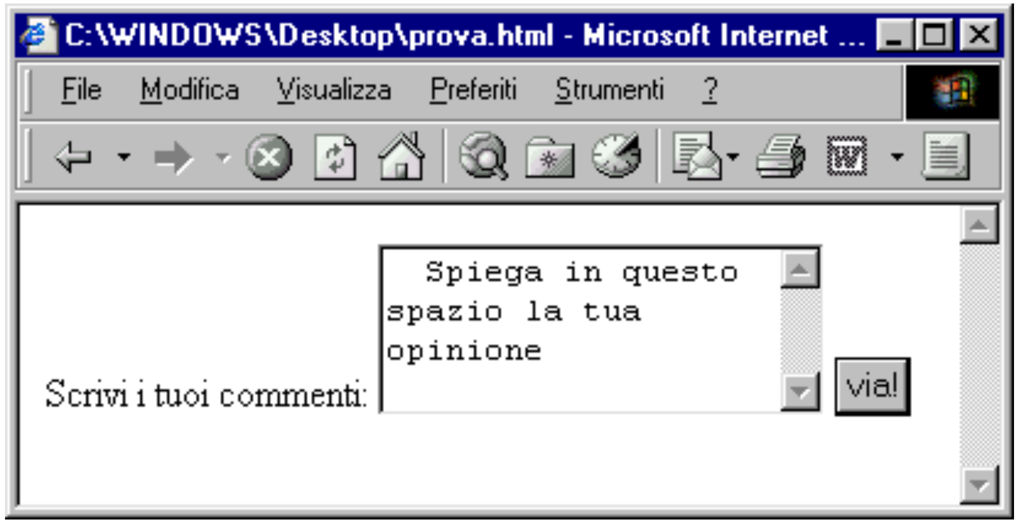
```
<FORM method="POST" action="/cgi-bin/elabora">
  Fai la tua scelta:
 <LI><Input type="checkbox"
 name="tipo" value="auto" checked>Auto
 <LI><Input type="checkbox"
 name="tipo" value="bus">Bus
 <LI><Input type="checkbox"
 name="tipo" value="camion">Camion
 <P><Select name="colore">
 <option>Rosso
 <option selected>Argento
 </select></P>
 <Input type="submit" value="spedisci">
</FORM>
```





Form – textarea

```
<FORM method="POST" action="/cgi-bin/elabora">
   Scrivi i tuoi commenti:
  <Textarea
  name="commenti" rows="4" columns="14">
 Spiega in questo spazio la tua opinione
  </TEXTAREA>
  <Input type="submit" value="via!">
</FORM>
```



Scrivi i tuoi commenti:

```
  Spiega in questo
spazio la tua
opinione
```

Example

Esempio: ShowParameters

```
package coreservlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class ShowParameters extends HttpServlet {
  public void doGet(HttpServletRequest request HttpServletResponse
     response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Reading All Request Parameters";
    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>Parameter Name<TH>Parameter Value(s)");
```

Esempio: ShowParameters

```
  Enumeration paramNames = request.getParameterNames();
  while(paramNames.hasMoreElements()) {
    String paramName = (String)paramNames.nextElement();
    out.print("<TR><TD>" + paramName + "\n<TD>");
    String[] paramValues = request.getParameterValues(paramName);
    if (paramValues.length == 1) {
      String paramValue = paramValues[0];
      if (paramValue.length() == 0) out.println("<I>No Value</I>");
      else out.println(paramValue);
    } else {
      out.println("<UL>");
      for(int i=0; i<paramValues.length; i++) {out.println("<LI>"
   +paramValues[i]);   }
      out.println("</UL>");
    }
  }
  out.println("</TABLE>\n</BODY></HTML>");
}
```

Esempio: ShowParameters

```
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
    doGet(request, response);
  }
}
```

Esempio: ShowParameters

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
 <TITLE>A Sample FORM using POST </TITLE>
</HEAD>
<BODY BGCOLOR="#FDF5E6">
<H1 ALIGN="CENTER">A Sample FORM using POST</H1>

<FORM ACTION="/servlet/coreservlets.ShowParameters“
    METHOD="POST”>
 Item Number: <INPUT TYPE="TEXT" NAME="itemNum"><BR>
 Quantity: <INPUT TYPE="TEXT" NAME="quantity"><BR>
 Price Each: <INPUT TYPE="TEXT" NAME="price" VALUE="$"><BR>
 <HR>
 First Name: <INPUT TYPE="TEXT" NAME="firstName"><BR>
 Last Name: <INPUT TYPE="TEXT" NAME="lastName"><BR>
 Middle Initial: <INPUT TYPE="TEXT" NAME="initial"><BR>
 Shipping Address:
 <TEXTAREA NAME="address" ROWS=3 COLS=40></TEXTAREA><BR>
```

Esempio: ShowParameters

```
  Credit Card:<BR>
  &nbsp;&nbsp;<INPUT TYPE="RADIO" NAME="cardType“
    VALUE="Visa">Visa<BR>
  &nbsp;&nbsp;<INPUT TYPE="RADIO" NAME="cardType"
                VALUE="Master Card">Master Card<BR>
  &nbsp;&nbsp;<INPUT TYPE="RADIO" NAME="cardType"
                VALUE="Amex">American Express<BR>
  &nbsp;&nbsp;<INPUT TYPE="RADIO" NAME="cardType“
    VALUE="Discover">Discover<BR>
  &nbsp;&nbsp;<INPUT TYPE="RADIO" NAME="cardType"
                VALUE="Java SmartCard">Java SmartCard<BR>
  Credit Card Number:
  <INPUT TYPE="PASSWORD" NAME="cardNum"><BR>
  Repeat Credit Card Number:
  <INPUT TYPE="PASSWORD" NAME="cardNum"><BR><BR>
  <CENTER><INPUT TYPE="SUBMIT" VALUE="Submit Order"></
    CENTER>
</FORM>
</BODY>
</HTML>
```

WebApps
(Tomcat configuration)

Static pages

To let Tomcat serve static pages, we must define a “Web Application”.

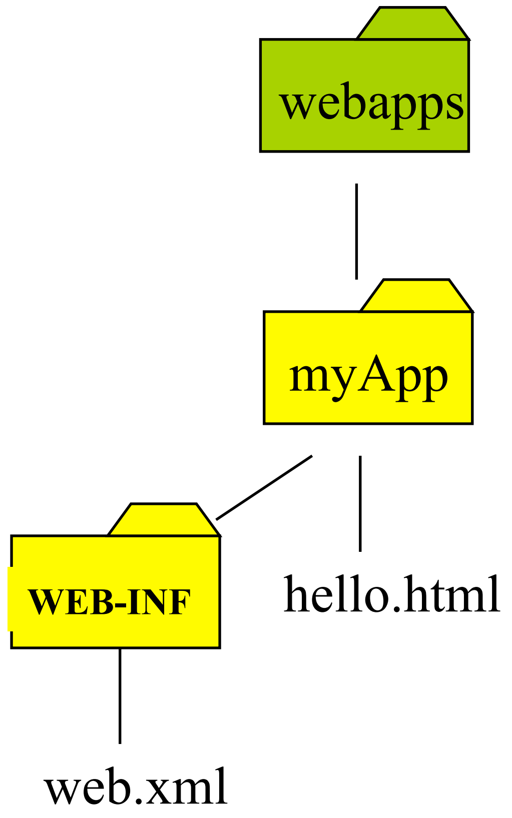
That is, in the Tomcat Document Root (by default *$CATALINA_HOME/webapps/*) we must create a folder named after our Web Application (e.g. myApp).

In that “myApp” folder, we MUST create a WEB-INF folder (that can be empy).

In the myApp folder we can then depost the static html files.

On our Tomcat server, the URL for the hello.html file becomes: http://*machine/port*/myApp/hello.html

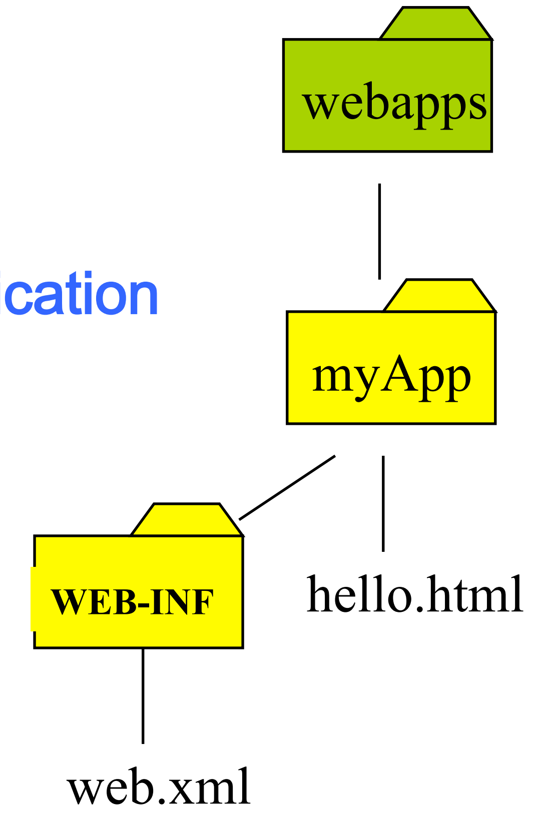
To actually see the webapp, we might have to restart Tomcat



Static pages

A web.xml file **MUST** be provided:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN“
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
</web-app>
```



Servlets

To let Tomcat serve servlet, we need add some info. The compiled servlets (.class) must be stored in a “classes” directory in WEB-INF.

Moreover, the web.xml file MUST contain at least:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <servlet-mapping>
        <servlet-name>invoker</servlet-name>
        <url-pattern>/magic/*</url-pattern>
    </servlet-mapping>
</web-app>
```

The “magic” word is the servlet activation keyword (you can of course customize this word). To execute the servlet called MyServlet.class, the URL will be:

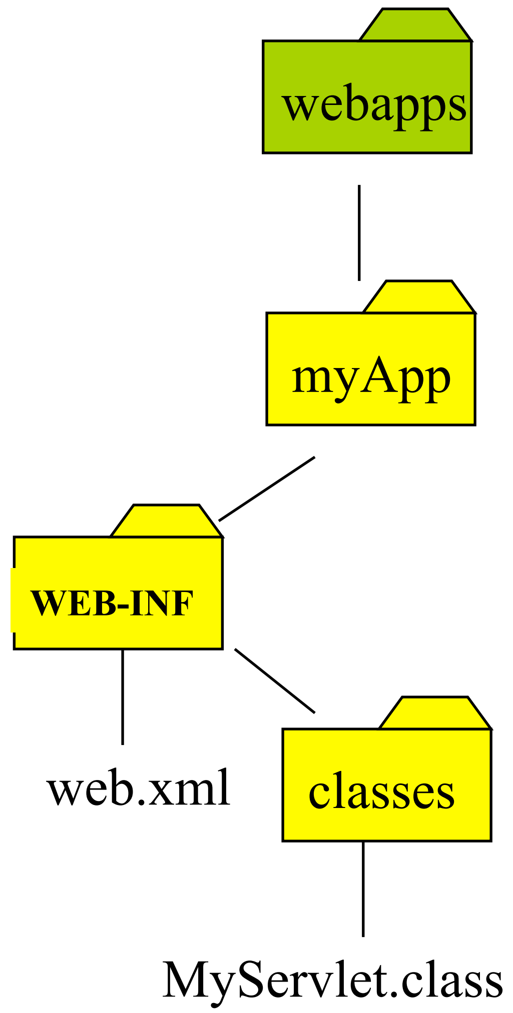
http://*machine/port*/myApp/magic/MyServlet

Servlets

The web.xml file CAN contain many additional info.
For instance, it can contain a section defining an alias name for the servlet:

```
…
    <servlet>
        <servlet-name>pippo</servlet-name>
        <servlet-class>myServlet</servlet-class>
    </servlet>
…
```

In such case, the servlet called MyServlet.class
Can be activated ALSO by the URL:
http://*machine/port/*myApp/magic/pippo

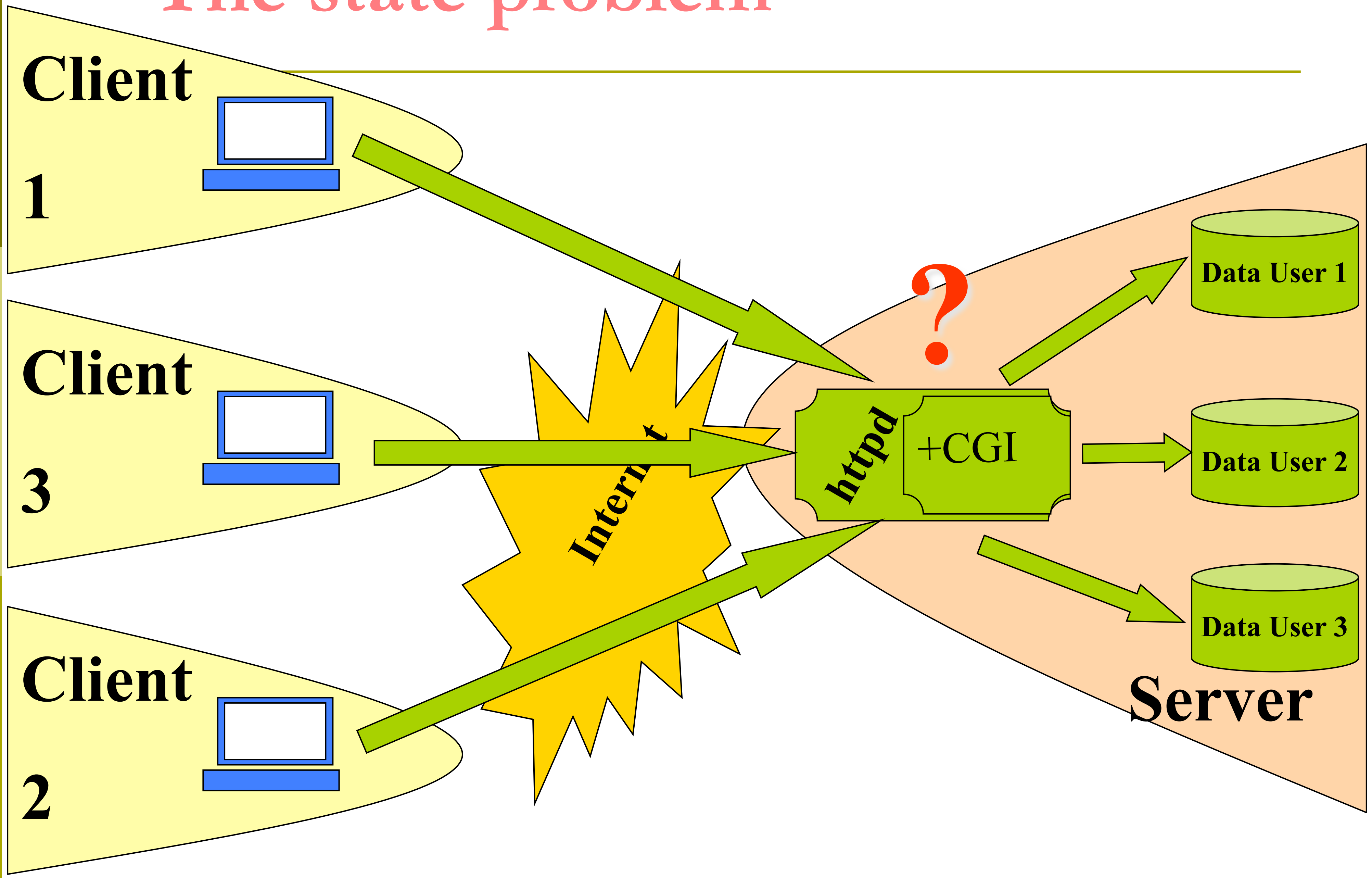


Servlets

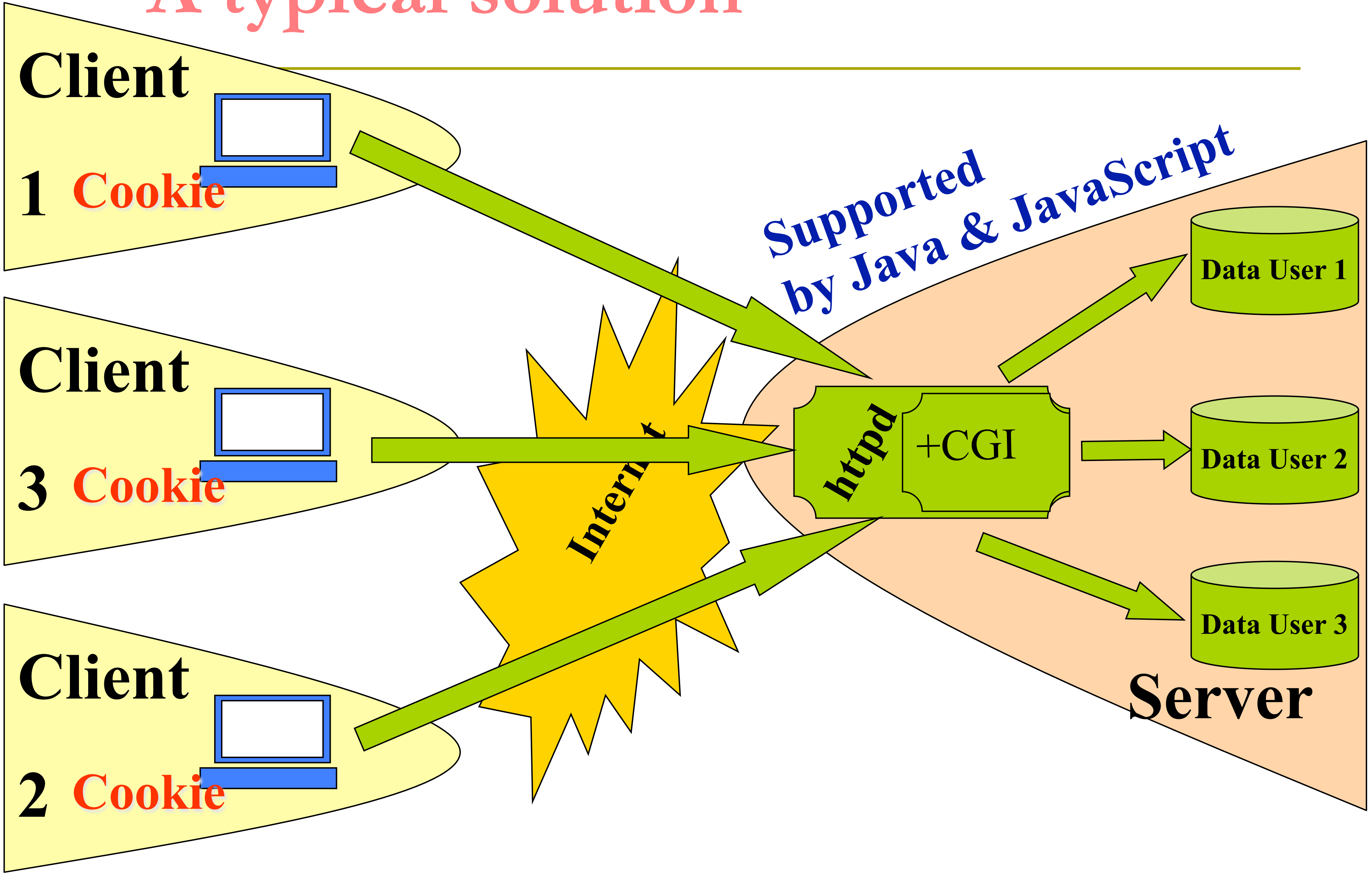
In JEE 7 it is possible to use annotation in place of the web.xml file

```
@WebServlet("/report")
public class MoodServlet extends HttpServlet {…}
```

The state problem



A typical solution



Cookies

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.

Some Web browsers have bugs in how they handle the optional attributes, so use them sparingly to improve the interoperability of your servlets.

Cookies

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

Cookies: why?

To maintain status across a “user session”

To maintan infos across sessions

- Customer identification
- Targeted advertisement
- Elimination of username e password

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);
```

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. This is such a common process that the following section presents a simple getCookieValue method that, given the array of cookies, a name, and a default value, returns the value of the cookie matching the name, or, if there is no such cookie, the designated default value.

Cookies: examples

```
Cookie userCookie = new Cookie("user","uid1234");
userCookie.setMaxAge(60*60*24*365);
response.addCookie(userCookie);
```

Code to check if the client accepts cookies:

See http://www.purpletech.com/code/src/com/purpletech/servlets/CookieDetector.java

SetCookies

```
import java.io.*;   import javax.servlet.*; import javax.servlet.http.*;
/** Sets six cookies: three that apply only to the current session
 * (regardless of how long that session lasts) and three that persist for an hour
 * (regardless of whether the browser is restarted).
*/
public class SetCookies extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                 throws ServletException, IOException {
    for(int i=0; i<3; i++) {
      // Default maxAge is -1, indicating cookie
      // applies only to current browsing session.
      Cookie cookie = new Cookie("Session-Cookie-" + i,
                                 "Cookie-Value-S" + i);
      response.addCookie(cookie);
```

SetCookies

```
cookie = new Cookie("Persistent-Cookie-" + i,"Cookie-Value-P" + i);
      // Cookie is valid for an hour, regardless of whether
      // user quits browser, reboots computer, or whatever.
      cookie.setMaxAge(3600);
      response.addCookie(cookie);
    }
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Setting Cookies";
    out.println (("<HTML><HEAD><TITLE>" +title+ “</TITLE></HEAD>" +
       "<BODY BGCOLOR=\"#FDF5E6\">\n" +"<H1 ALIGN=\"CENTER\">"
     + title + "</H1>\n" +"There are six cookies associated with this page.\n" +
"</BODY></HTML>");
  }
}
```

ShowCookies

```
import java.io.*;  import javax.servlet.*;   import javax.servlet.http.*;
/** Creates a table of the cookies associated with the current page. */
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

```
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: legal aspects

DIRECTIVE 2009/136/EC (EUROPEAN PARLIAMENT) 25 November 2009

Third parties may wish to store information on the equipment of a user, or gain access to information already stored, for a number of purposes, ranging from the legitimate (such as certain types of cookies) to those involving unwarranted intrusion into the private sphere (such as spyware or viruses). It is therefore of paramount importance that users be provided with clear and comprehensive information when engaging in any activity which could result in such storage or gaining of access. The methods of providing information and offering the right to refuse should be as user-friendly as possible.

DIRECTIVE 2009/136/EC (EUROPEAN PARLIAMENT) 25 November 2009

Exceptions to the obligation to provide information and offer the right to refuse should be limited to those situations where the technical storage or access is strictly necessary for the legitimate purpose of enabling the use of a specific service explicitly requested by the subscriber or user. Where it is technically possible and effective, in accordance with the relevant provisions of Directive 95/46/EC, the user's consent to processing may be expressed by using the appropriate settings of a browser or other application. The enforcement of these requirements should be made more effective by way of enhanced powers granted to the relevant national authorities.

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

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 wew;

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").

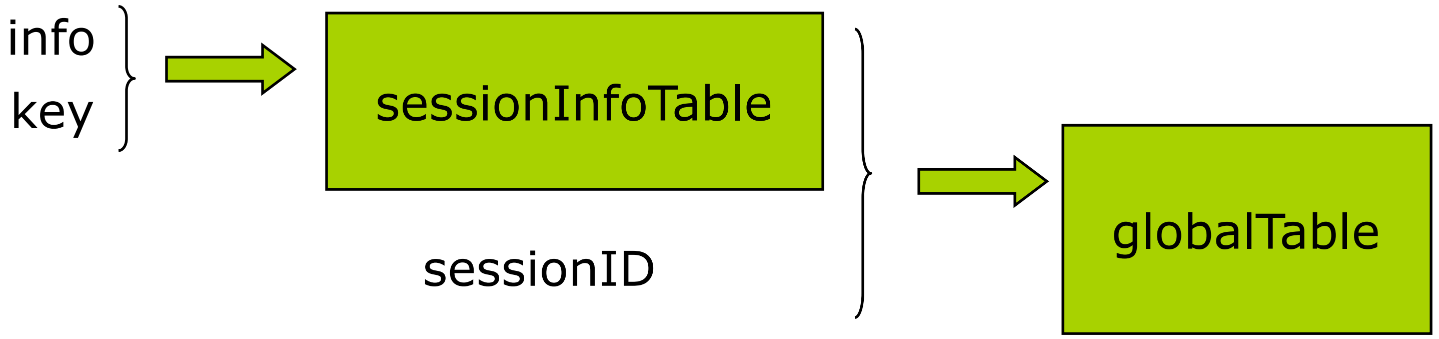
Italian regulations

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.

Sessions

Session tracking using cookies

```
String sessionID = makeUniqueString();
Hashtable sessionInfoTable = new Hashtable();
Hashtable globalTable = getTableStoringSession();
globalTable.put(sessionID, sessionInfoTable );
Cookie sessionCookie=new Cookie("SessionID",sessionID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```



HttpSession Class

Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

A session usually corresponds to one user, who may visit a site many times. The server can maintain a session in many ways such as using cookies or rewriting URLs.

HttpSession Class

This interface allows servlets to View and manipulate information about a session, such as the session identifier, creation time, and last accessed time Bind objects to sessions, allowing user information to persist across multiple user connections.

When an application stores an object in or removes an object from a session, the session checks whether the object implements HttpSessionBindingListener. If it does, the servlet notifies the object that it has been bound to or unbound from the session.

Session tracking API

```
HttpSession session = request.getSession(true);
ShoppingCart cart =
    (ShoppingCart)session.getValue(“carrello”); // 2.1
// 2.2   (ShoppingCart)session.getAttribute(“carrello”);
if (cart==null) {
    cart=new ShoppingCart();
    session.putValue(“carrello”,cart); //2.1
//2.2 session.putAttribute(“carrello”,cart);
}
doSomeThingWith(cart);
```

Session tracking API

```
public void putValue(String name, Object value); //2.1
public void setAttribute(String name, Object value);        //2.2

public void removeValue(String name);   //2.1
public void removeAttribute(String name);           //2.2

public String[] getValueNames()            //2.1
public Enumeration getAttributeNames() //2.2
```

Session tracking API

```
public long getCreationTime();
public long getLastAccessdTime();
   milliseconds since midnight,  1.1.1970

public int getMaxInactiveInterval();
public void setMaxInactiveInterval(int sec);

public void invalidate();
```

ShowSession

```
import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
import java.net.*; import java.util.*;
/** Simple example of session tracking. */
public class ShowSession extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                          throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Session Tracking Example";
    HttpSession session = request.getSession(true);
    String heading;
    // Use getAttribute instead of getValue in version 2.2.
    Integer accessCount = (Integer)session.getValue("accessCount");
```

ShowSession

```
if (accessCount == null) {
      accessCount = new Integer(0);
      heading = "Welcome Newcomer";
    } else {
      heading = "Welcome Back";
      accessCount = new Integer(accessCount.intValue() + 1);
    }
    // Use setAttribute instead of putValue in version 2.2.
    session.putValue("accessCount", accessCount);
```

ShowSession

```
out.println(("<HTML><HEAD><TITLE>" +title+ “</TITLE></HEAD>" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=\"CENTER\">" + heading + "</H1>\n" +
            "<H2>Information on Your Session:</H2>\n" +
            "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
            "<TR BGCOLOR=\"#FFAD00\">\n" +
            "  <TH>Info Type<TH>Value\n" +
            "<TR>\n" +"  <TD>ID\n" +"  <TD>" + session.getId() + "\n" +
            "<TR>\n" +"  <TD>Creation Time\n" +
            "  <TD>" + new Date(session.getCreationTime()) + "\n" +
            "<TR>\n" +"  <TD>Time of Last Access\n" +
            "  <TD>" +new Date(session.getLastAccessedTime()) + "\n" +
            "<TR>\n" +"  <TD>Number of Previous Accesses\n" +"  <TD>" +
            accessCount + "\n" + "</TABLE>\n" +"</BODY></HTML>");
}
```

ShowSession

```
  /** Handle GET and POST requests identically. */

  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
    throws ServletException, IOException {
   doGet(request, response);
  }
}
```

No cookies?

URL REWRITING

In forms:

```
response.encodeURL(myUrl);
```

Accessibility

Accessibility

What is Section 508?
The legislation referred to as "Section 508" is actually an amendment to the Workforce Rehabilitation Act of 1973. The amendment was signed into law by President Clinton on August 7, 1998. Section 508 requires that electronic and information technology that is developed or purchased by the Federal Government is accessible by people with disabilities.

See http://jimthatcher.com/webcourse8.htm for accessibility when using forms

http://jimthatcher.com/webcourse1.htm for accessibility in general.

http://www.innovazione.gov.it/ita/normativa/pubblicazioni/2004_rapporto_comm_acc.pdf

Accessibility in Italy

Legge Stanca 9 gennaio 2004, n. 4

Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici

Testo della legge:
- http://www.pubbliaccesso.gov.it/normative/legge_20040109_n4.htm

Vedi anche:
- http://www.cnipa.gov.it/site/it-IT/Attivit%C3%A0/Commissioni_e_Gruppi_di_Lavoro_interministeriali/Accessibilit%C3%A0/

Rapporto 2004 della commissioneCommissione interministeriale permanente per l'impiego delle ICT a favore delle categorie deboli o svantaggiate
- **http://www.innovazione.gov.it/ita/normativa/pubblicazioni/2004_rapporto_comm_acc.pdf**