

---

# SISTEMAS WEB

## CURSO 2023/2024

Tomcat – ShareInfo II  
Aplicación web para compartir mensajes



Web Sistemak by [Oskar Casquero](#) & [María Luz Álvarez](#) is licensed under a [Creative Commons Reconocimiento 4.0 Internacional License](#).

# APLICACIÓN WEB DINÁMICA - SHAREINFO

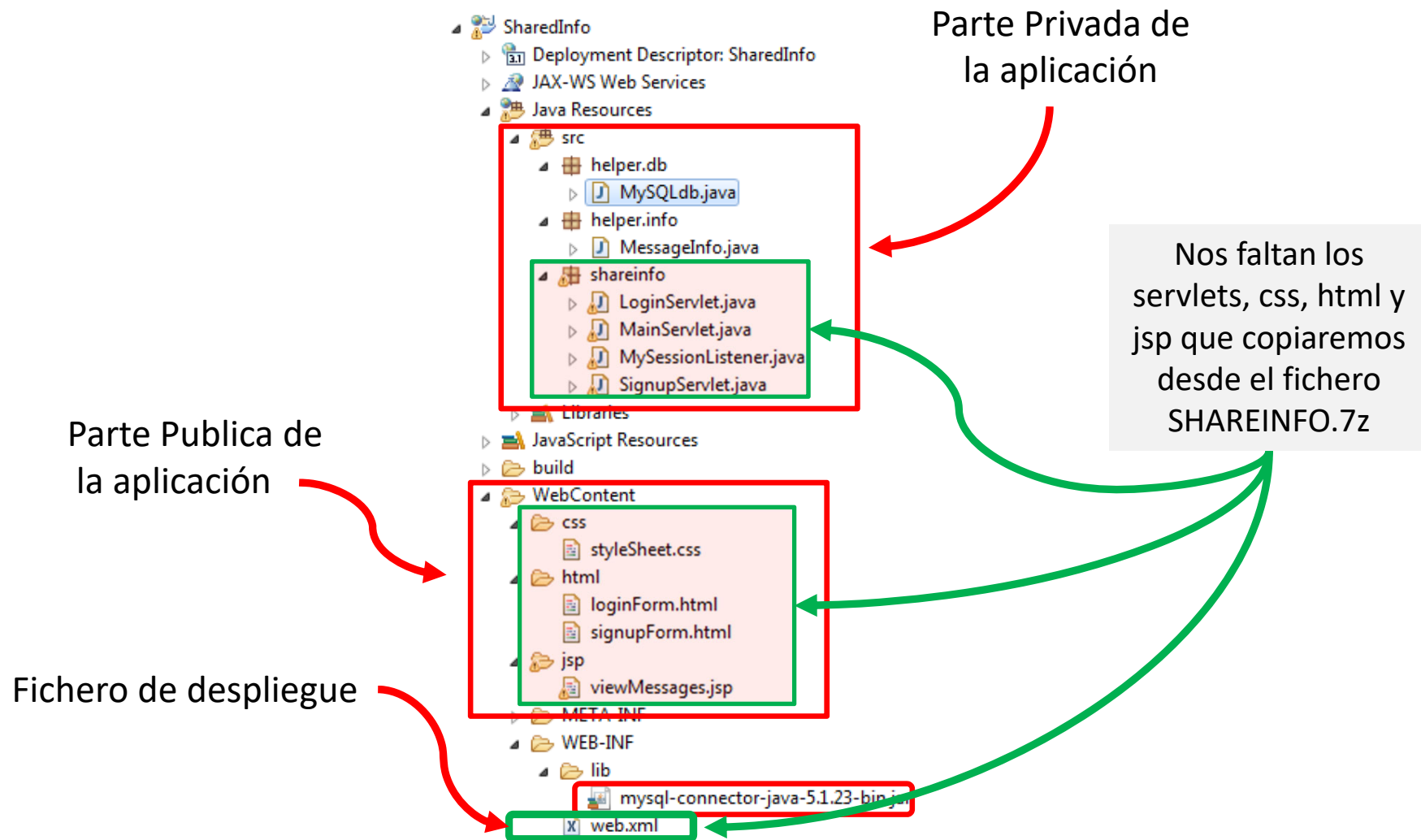
## INTRODUCCIÓN

---

- **ShareInfo**, es una aplicación de ejemplo para compartir mensajes cortos.
- Utilizando **ShareInfo** se van a analizar, introducir y profundizar en diversos aspectos de las aplicaciones Web en un servidor:
  - Uso de bases de datos en una aplicación Web
  - Ciclo de vida de un servlet
  - Conceptos de sesión (un visitante)
  - Concepto de contexto de la aplicación (varios visitantes)
  - Compartir información entre servlets, en una sesiones y en diferentes sesiones
  - Control de eventos (listeners)
  - JSP
  - AJAX - Javascript y XML Asíncrono

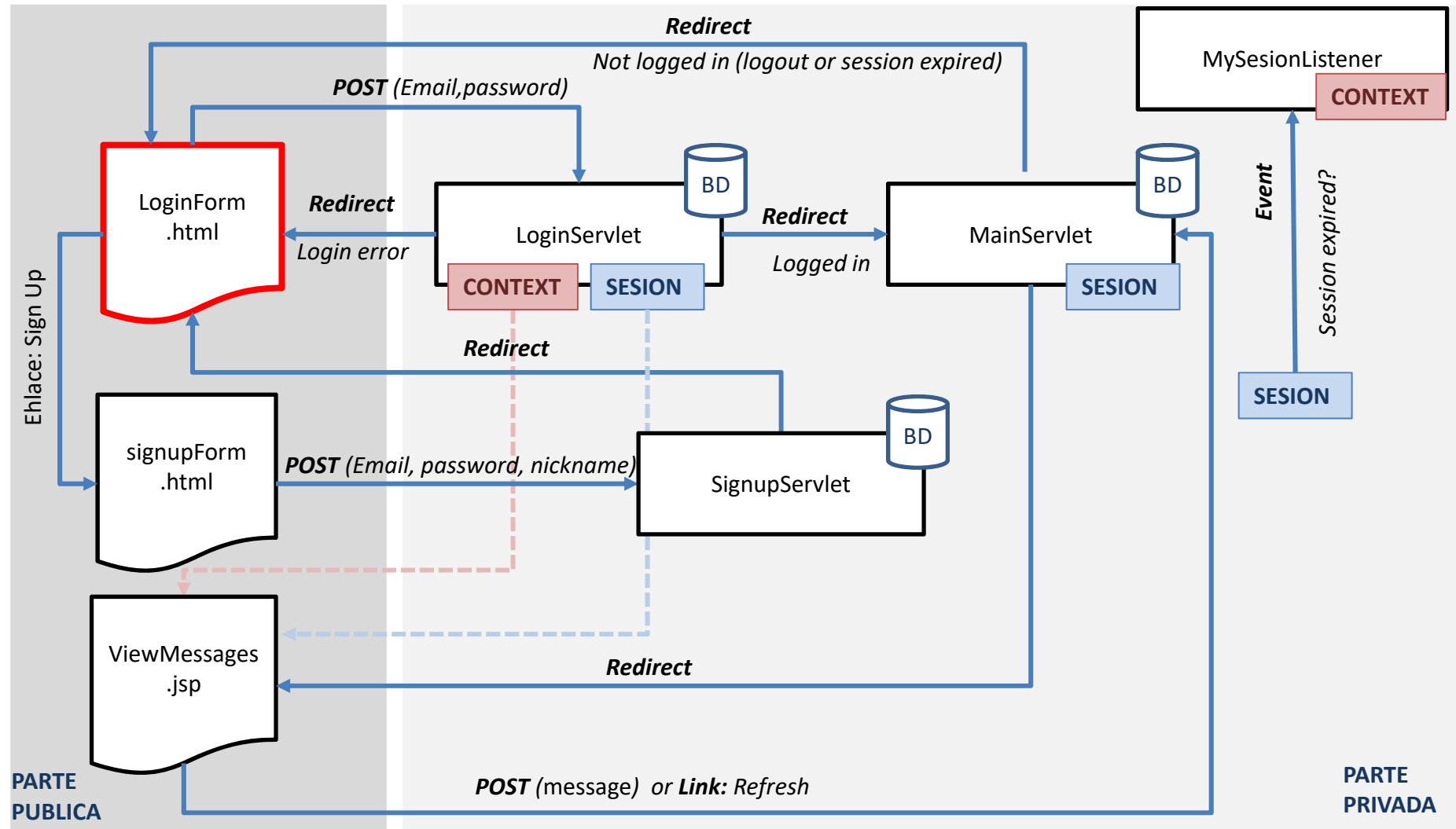
# APLICACIÓN WEB DINÁMICA - SHAREINFO

## INSTALAR PROYECTO JAVA



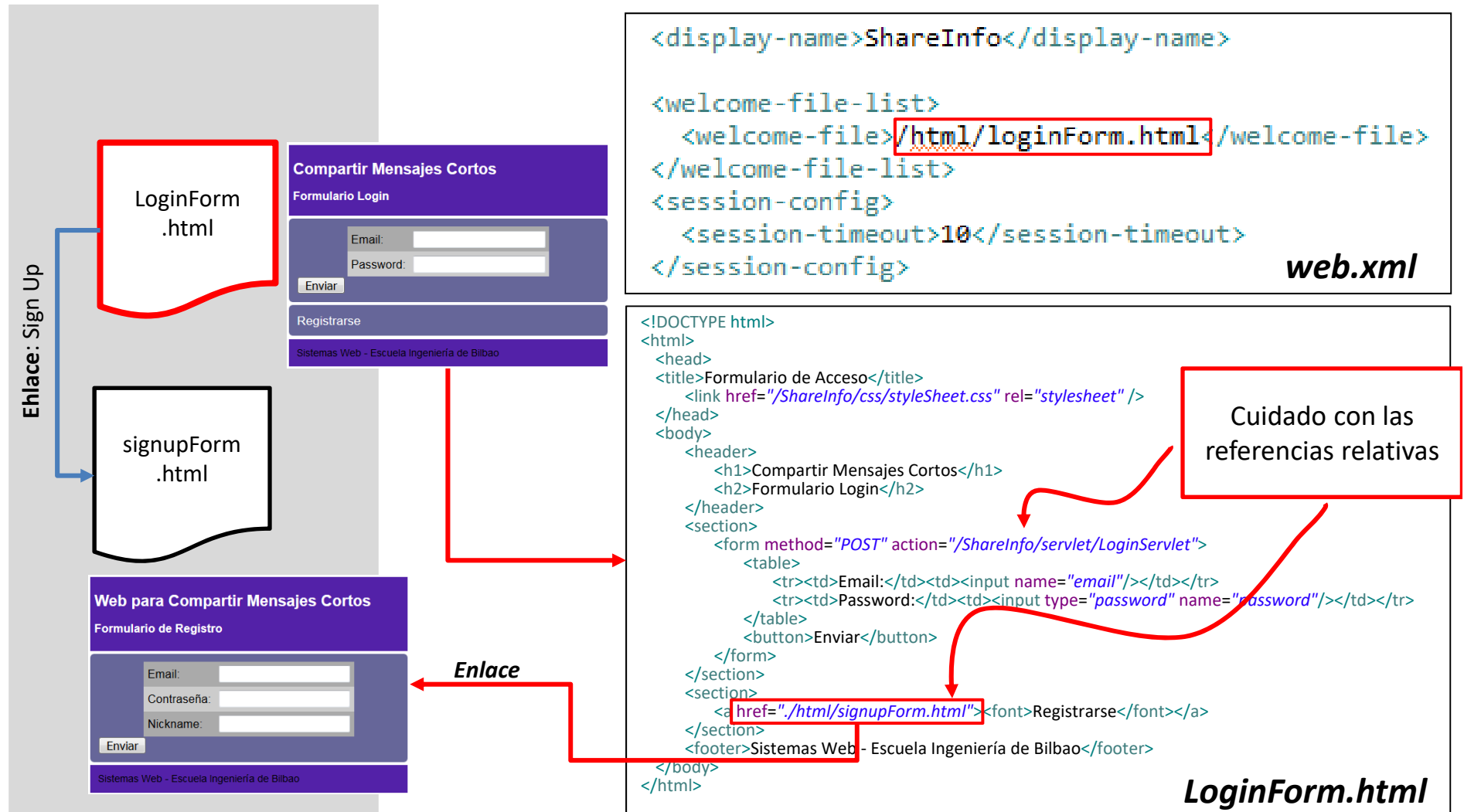
# APLICACIÓN WEB DINÁMICA – SHAREINFO

## DIAGRAMA DE FLUJO DE LA APLICACIÓN



# APLICACIÓN WEB DINÁMICA – SHAREINFO

## PAGINA DE INICIO DE LA APLICACIÓN



# APLICACIÓN WEB DINÁMICA – SHAREINFO

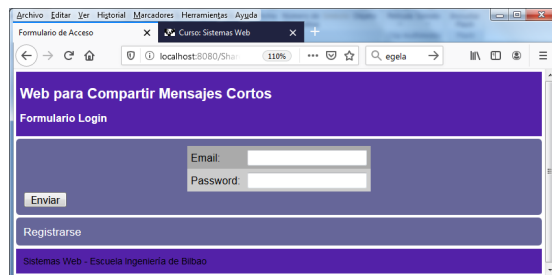
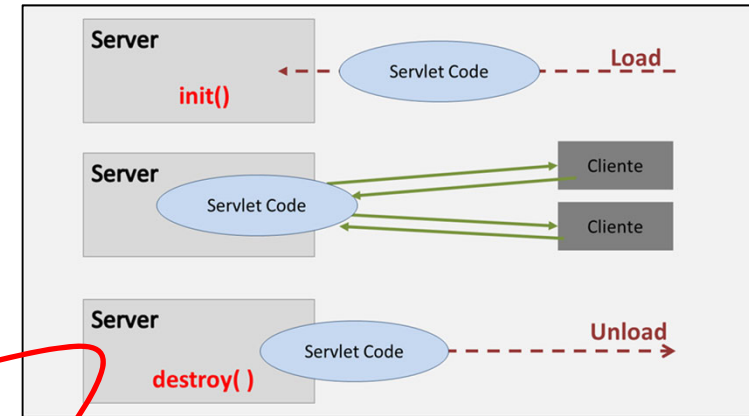
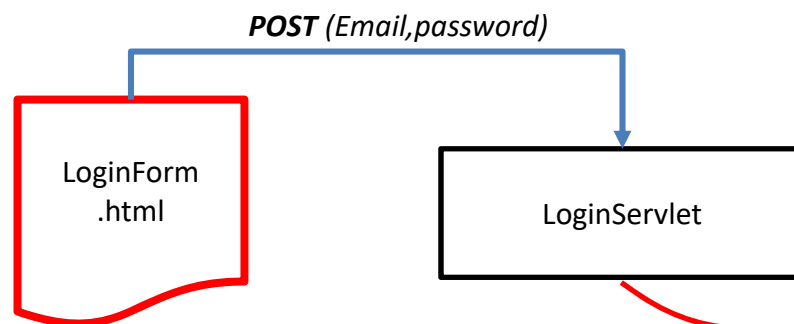
## INTRODUCCIÓN

---

- **ShareInfo**, es una aplicación de ejemplo para compartir mensajes cortos.
- Utilizando **ShareInfo** se van a analizar, introducir y profundizar en diversos aspectos de las aplicaciones Web en un servidor:
- Utilizando **ShareInfo** se van a analizar, introducir y profundizar en diversos aspectos de las aplicaciones Web en un servidor:
  - Uso de bases de datos en una aplicación Web
  - **Ciclo de vida de un servlet**
  - Conceptos de sesión (un visitante)
  - Concepto de contexto de la aplicación (varios visitantes)
  - Compartir información entre servlets, en una sesiones y en diferentes sesiones
  - Control de eventos (listeners)
  - JSP
  - AJAX - Javascript y XML Asíncrono

# APLICACIÓN WEB DINÁMICA – SHAREINFO

## CICLO DE VIDA DE UN SERVLET – init()



**init()** es el método llamado por el contenedor para indicar a un servlet que se está iniciando el servicio y realice las tareas necesarias antes de iniciar.

```
package shareinfo;
import java.io.*;

public class LoginServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;
    private MySQLdb mySQLdb;

    public void init(ServletConfig config) {
        System.out.println("---> Entrando en init() de LoginServlet");
        mySQLdb = new MySQLdb();
        System.out.println("---> Saliendo de init() de LoginServlet");
    }

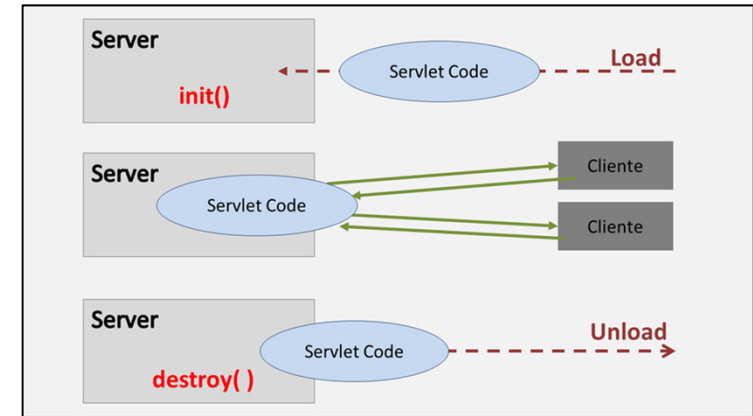
    public void doPost(HttpServletRequest request, HttpServletResponse response){
```

*Iniciar la conexión con la base de datos la primera vez que se llama al servlet.*

# APLICACIÓN WEB DINÁMICA – SHAREINFO

## CICLO DE VIDA DE UN SERVLET – `destroy()`

- **`destroy()`** es el método llamado por el contenedor para indicar a un servlet que se está cerrando el servicio y realice las tareas necesarias antes de cerrar.



- **Modificar** para conocer el número de veces que los usuarios se han conectado con este servlet y lo imprima en pantalla cuando se cierre el servicio.
- **Modificar** para que se muestre el número de veces que un usuario además se ha autenticado correctamente en la aplicación.



# APLICACIÓN WEB DINÁMICA – SHAREINFO

## CICLO DE VIDA DE UN SERVLET – destroy()

---

```
public class LoginServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;
    private MySQLdb mySQLdb;
    private int numConexiones=0;

    public void init(ServletConfig config) {
        System.out.println("---> Entrando en init()de LoginServlet");
        mySQLdb = new MySQLdb();
        System.out.println("---> Saliendo de init()de LoginServlet");
    }

    public void destroy() {
        System.out.println("--->El número de conexiones ha sido:" + numConexiones);
    }

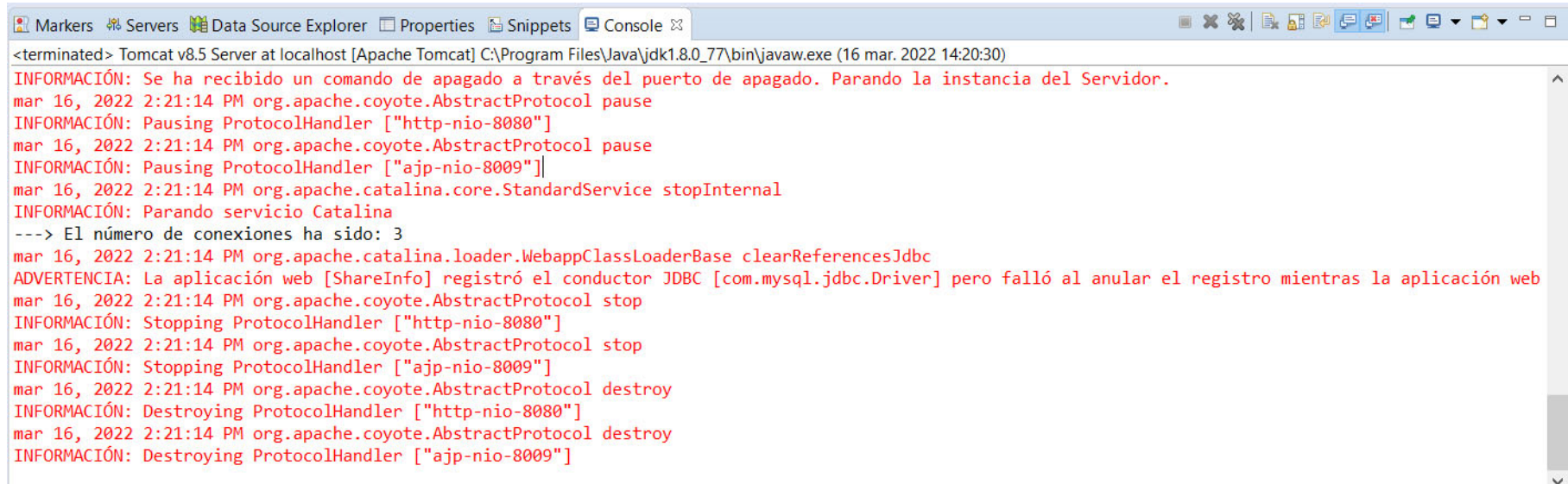
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        System.out.println("---> Entrando en doPost() de LoginServlet");
        numConexiones=numConexiones+1;

        String email = request.getParameter("email");
        String password = request.getParameter("password");
```

# APLICACIÓN WEB DINÁMICA – SHAREINFO

## CICLO DE VIDA DE UN SERVLET – destroy()



```
<terminated> Tomcat v8.5 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk1.8.0_77\bin\javaw.exe (16 mar. 2022 14:20:30)
INFORMACIÓN: Se ha recibido un comando de apagado a través del puerto de apagado. Parando la instancia del Servidor.
mar 16, 2022 2:21:14 PM org.apache.coyote.AbstractProtocol pause
INFORMACIÓN: Pausing ProtocolHandler ["http-nio-8080"]
mar 16, 2022 2:21:14 PM org.apache.coyote.AbstractProtocol pause
INFORMACIÓN: Pausing ProtocolHandler ["ajp-nio-8009"]
mar 16, 2022 2:21:14 PM org.apache.catalina.core.StandardService stopInternal
INFORMACIÓN: Parando servicio Catalina
---> El número de conexiones ha sido: 3
mar 16, 2022 2:21:14 PM org.apache.catalina.loader.WebappClassLoaderBase clearReferencesJdbc
ADVERTENCIA: La aplicación web [ShareInfo] registró el conductor JDBC [com.mysql.jdbc.Driver] pero falló al anular el registro mientras la aplicación web
mar 16, 2022 2:21:14 PM org.apache.coyote.AbstractProtocol stop
INFORMACIÓN: Stopping ProtocolHandler ["http-nio-8080"]
mar 16, 2022 2:21:14 PM org.apache.coyote.AbstractProtocol stop
INFORMACIÓN: Stopping ProtocolHandler ["ajp-nio-8009"]
mar 16, 2022 2:21:14 PM org.apache.coyote.AbstractProtocol destroy
INFORMACIÓN: Destroying ProtocolHandler ["http-nio-8080"]
mar 16, 2022 2:21:14 PM org.apache.coyote.AbstractProtocol destroy
INFORMACIÓN: Destroying ProtocolHandler ["ajp-nio-8009"]
```

**Modificar** para que se muestre el número de veces que un usuario además se ha autenticado correctamente en la aplicación.

# APLICACIÓN WEB DINÁMICA – SHAREINFO

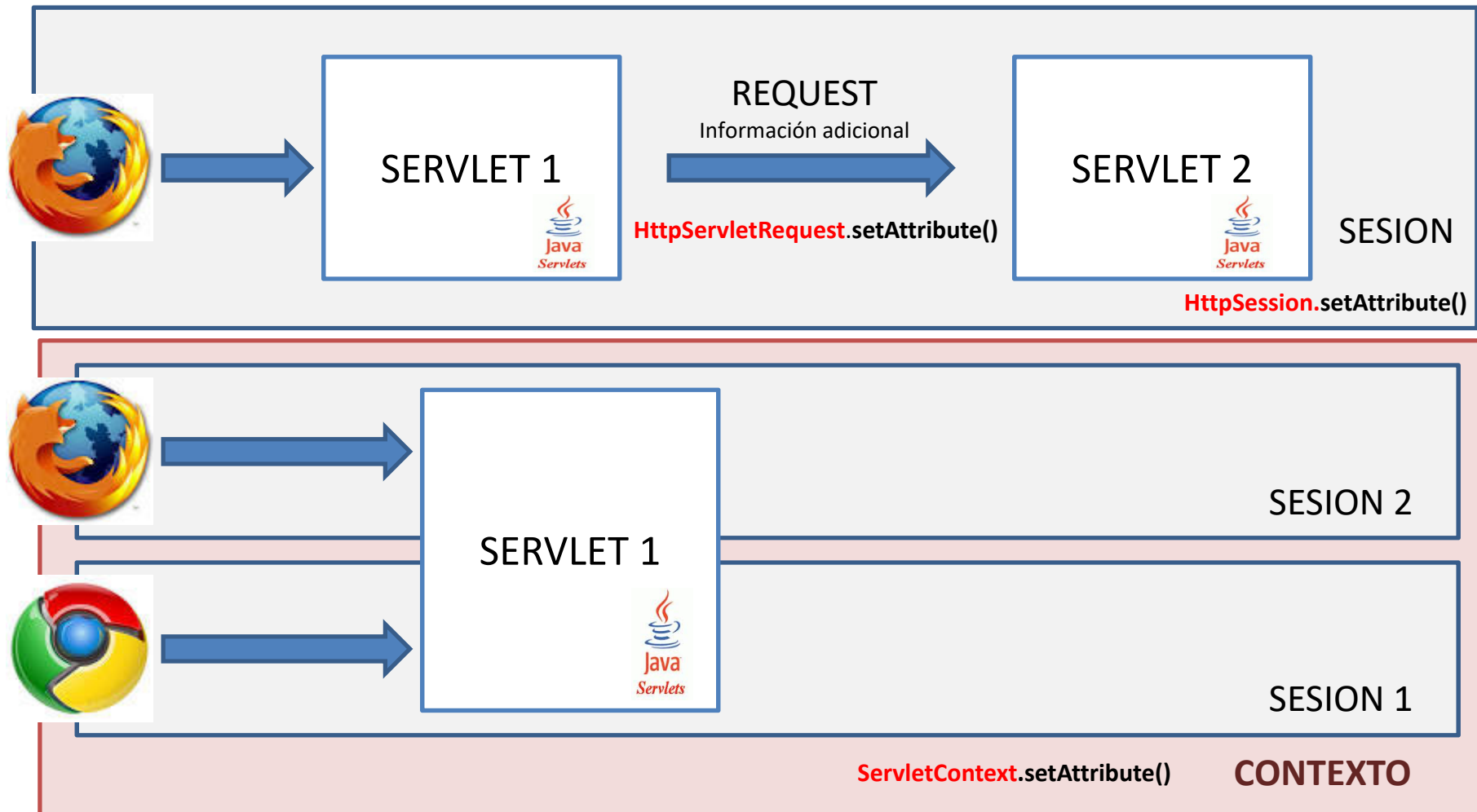
## INTRODUCCIÓN

---

- **ShareInfo**, es una aplicación de ejemplo para compartir mensajes cortos.
- Utilizando **ShareInfo** se van a analizar, introducir y profundizar en diversos aspectos de las aplicaciones Web en un servidor:
  - Uso de bases de datos en una aplicación Web
  - Ciclo de vida de un servlet
  - Conceptos de sesión (un visitante)
  - Concepto de contexto de la aplicación (varios visitantes)
  - Compartir información entre servlets, en una sesiones y en diferentes sesiones
  - Control de eventos (listeners)
  - JSP
  - AJAX - Javascript y XML Asíncrono

# SERVLETS

## COMPARTIR INFORMACIÓN



# APLICACIÓN WEB DINÁMICA - SHAREINFO

## SESIÓN: HTTPSESSION

---

- HTTP es un protocolo sin estado. En Java la clase **HttpSession** que sirve para almacenar información entre diferentes peticiones HTTP de un usuario. Podemos asociar a cada visitante de nuestra página web una sesión. Siempre que recibamos una petición de dicho visitante podremos acceder a su sesión, y sólo las peticiones de ese visitante podrán acceder a ella.
- En una sesión se pueden almacenar y recuperar atributos en una estructura de **HashMap (clave -valor)**, donde las claves son cadenas de caracteres y los valores pueden ser cualquier tipo de objeto java.
- Para obtener y crear el objeto sesión desde un objeto *HttpServletRequest*:
  - Devuelve la sesión activa, si no existe, crea una nueva:
    - **HttpSession session = request.getSession()**
    - **HttpSession session = request.getSession(true)**
  - Devuelve la sesión activa, si no existe, no crea una nueva y devuelve null:
    - **HttpSession session = request.getSession(false)**

<http://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpSession.html>

# APLICACIÓN WEB DINÁMICA - SHAREINFO

## SESIÓN: HTTPSESSION

---

- La sesión en sí es un objeto Java que ocupa memoria. Lo mismo sucede con todos los objetos que se almacenen dentro de ella. El servidor no puede crear sesiones de modo indefinido, y almacenarlas para siempre, ya que agotaría su memoria. En algún momento debe destruir esas sesiones. El tiempo de duración de la sesión se define en el fichero **web.xml**

```
<session-config>
  <session-timeout>2</session-timeout>
</session-config>
```

- Observa:
  - ¿Donde se crea la sesión en la aplicación *ShareInfo*?
  - En el monitor de red del navegador observa las cabeceras de las peticiones y en la consola de la aplicación el identificador de las sesiones. ¿Que observas?
  - Para que se utiliza en *MainServlet.java* el método *request.getSession()*

```
if(request.getSession(false) == null) {
    System.out.println("User is not logged in");

    System.out.println("Redirecting the user to LoginForm.html");
    RequestDispatcher rd = request.getRequestDispatcher("/html/loginForm.html");
    rd.forward(request, response);
} else {
    System.out.println("User is logged in");
    ...
}
```

# APLICACIÓN WEB DINÁMICA - SHAREINFO

## SESIÓN: HTTPSESSION

---

- **Métodos de HttpSession:**
  - void **setAttribute**(String name, Object value): añade a la sesión un objeto.
  - Object **getAttribute** (String name): devuelve un objeto java almacenado en la sesión del usuario, o null.
  - long **getCreationTime**(): devuelve el instante en el que fue creada la sesión.
  - String **getId**(): devuelve un identificador único para la sesión.
  - long **getLastAccessedTime**(): devuelve instante en el cual se realizó la última petición asociada con esta sesión;
  - Int **getMaxInactiveInterval**(): devuelve el máximo tiempo en de inactividad permitido a la sesión en segundos.
  - ServletContext **getServletContext**(): devuelve el ServletContext al cual pertenece esta sesión.
  - void **invalidate**(): invalida la sesión.
  - void **removeAttribute**(String name): elimina de la sesión el atributo asociado al nombre.

# APLICACIÓN WEB DINÁMICA - SHAREINFO

## CONTEXTO: SERVLETCONTEXT

---

- **ServletContext** define un conjunto de métodos que un servlet utiliza para comunicarse con su contenedor.
- Existe un contexto por aplicación web .
- El objeto contexto de una aplicación se puede conseguir tanto desde una petición como desde una sesión:
  - ServletContext context = **request/session**.**getServletContext()**;
- Al igual que en sesiones, en el **contexto** se pueden almacenar pares de **valores (clave, valor)**, donde las claves son cadenas de caracteres y los valores pueden ser cualquier objeto Java.

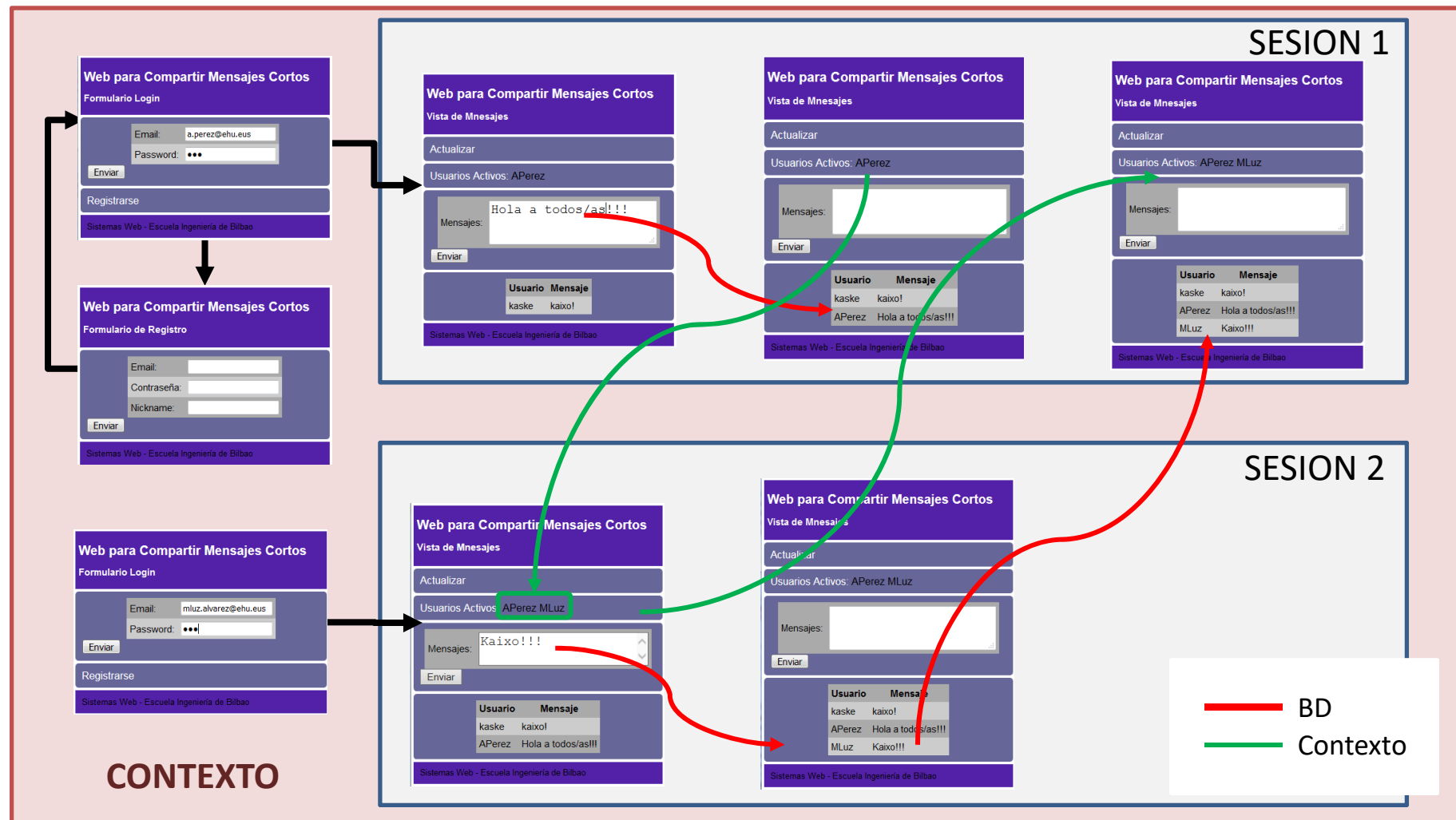
<http://docs.oracle.com/javaee/6/api/javax/servlet/ServletContext.html>

- **Metodos de ServletContext**
  - void **setAttribute** (String name, Object object): añade un objeto al contexto
  - Object **getAttribute** (String name): devuelve el objeto asociado con la cadena de caracteres que se le pasa como argumento..
  - void **removeAttribute**(String name): elimina el atributo asociado con la cadena de caracteres que se le pasa como argumento.



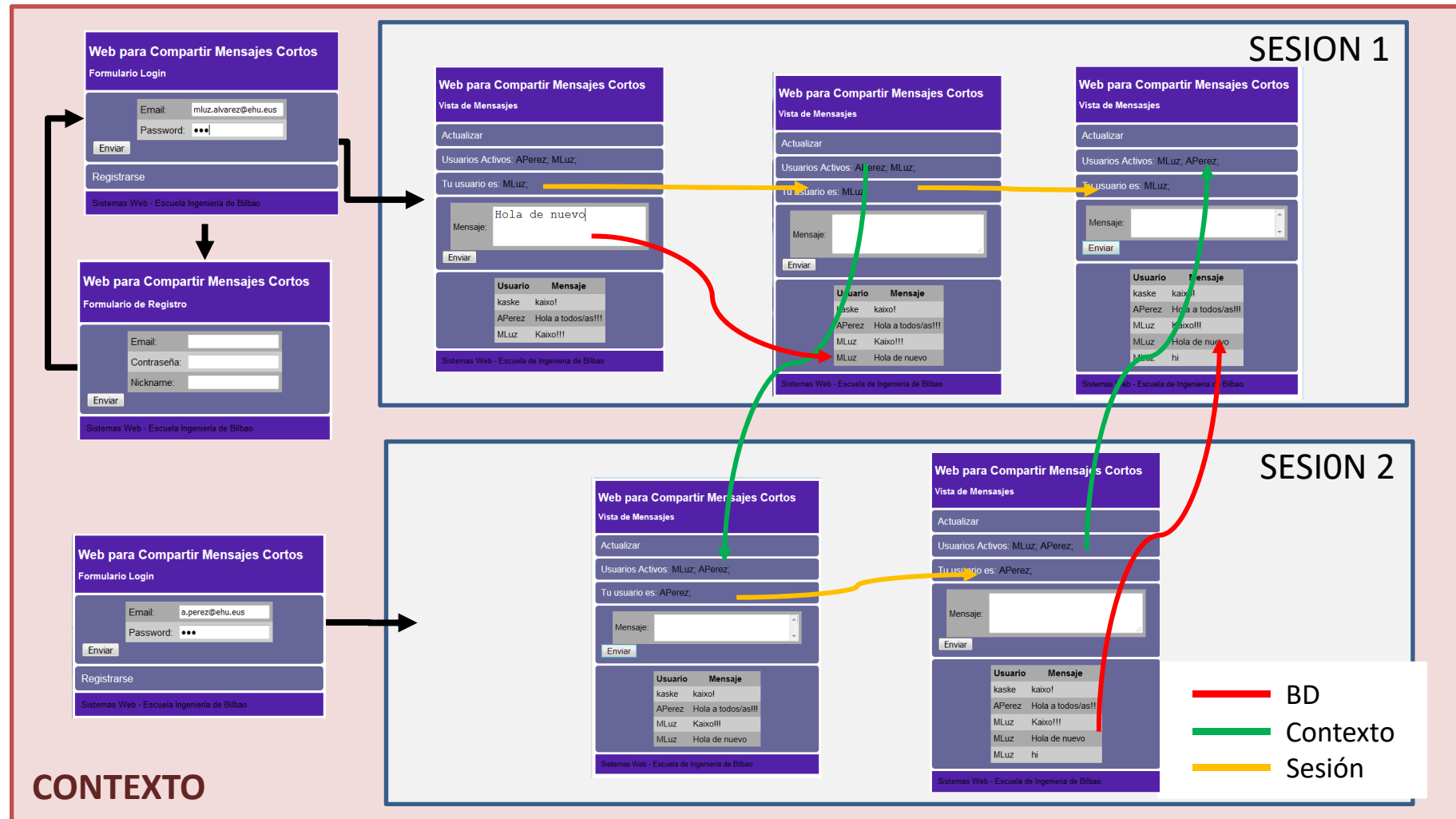
# APLICACIÓN WEB DINÁMICA – SHAREINFO

## COMPARTIR INFORMACIÓN: BD Y CONTEXTO



# APLICACIÓN WEB DINÁMICA – SHAREINFO (MODIFICACIÓN)

## COMPARTIR INFORMACIÓN: BD, **SESIÓN** Y CONTEXTO



# APLICACIÓN WEB DINÁMICA - SHAREINFO

## COMPARTIR INFORMACIÓN: BD, **SESIÓN** Y CONTEXTO

```
LoginServlet.java ✖
36     HttpSession session = request.getSession(true);
37     String sessionID = session.getId();
38
39     session.setAttribute("username", username);
40
41     System.out.println("    User session for " + username + ": " + sessionID);
```

```
viewMessages.jsp ✖
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2    pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html>
4
5  <%@ page import="java.util.*,helper.info.*"%>
6  <%
7    ArrayList<MessageInfo> messageList = (ArrayList<MessageInfo>) request.getAttribute("messageList");
8    ServletContext context = request.getServletContext();
9    HashMap<String, String> loggedInUsers = (HashMap) context.getAttribute("loggedin_users");
10   HttpSession session=request.getSession();
11 %>
```

```
viewMessages.jsp ✖
34     <%=entry.getKey()%>;
35     <% } %>
36 </section>
37
38 <section>
39     <font>Tu usuario es: </font>
40     <%=session.getAttribute("username")%>;
41 </section>
```

Nueva  
sección con  
usuario activo

# APLICACIÓN WEB DINÁMICA - SHAREINFO

## COMPARTIR INFORMACIÓN: BD, SESIÓN Y **CONTEXTO**

LoginServlet.java ×

```
4      String username = mySQLdb.getUsername(email, password);
5
6      if(username == null) {
7          System.out.println("    Login error: redireccionando al usuario a loginForm.html");
8          RequestDispatcher rd = request.getRequestDispatcher("/html/loginForm.html");
9          rd.forward(request, response);
10     } else {
11         numConexionesOK++;
12         HttpSession session = request.getSession(true);
13         String sessionID = session.getId();
14         session.setAttribute("username", username);
15         System.out.println("    Sesión de Usuario para " + username + ": " + sessionID);
16         System.out.print("    Cargando la lista de usuarios activos de contexto: ");
17         ServletContext context = request.getServletContext();
18         HashMap<String, String> loggedinUsers = (HashMap) context.getAttribute("loggedin_users");
19         if(loggedinUsers == null) {
20             System.out.println("Lista vacía");
21             loggedinUsers = new HashMap();
22             loggedinUsers.put(username, sessionID);
23         } else {
24             if(!loggedinUsers.containsKey(username)) {
25                 System.out.println(username + " no esta en la lista");
26                 loggedinUsers.put(username, sessionID);
27             } else {
28                 System.out.println(username + " ya esta en la lista");
29             }
30         }
31     }
32     context.setAttribute("loggedin_users", loggedinUsers);
33     System.out.println("    Loggedin users: " + loggedinUsers);
```

# APLICACIÓN WEB DINÁMICA - SHAREINFO

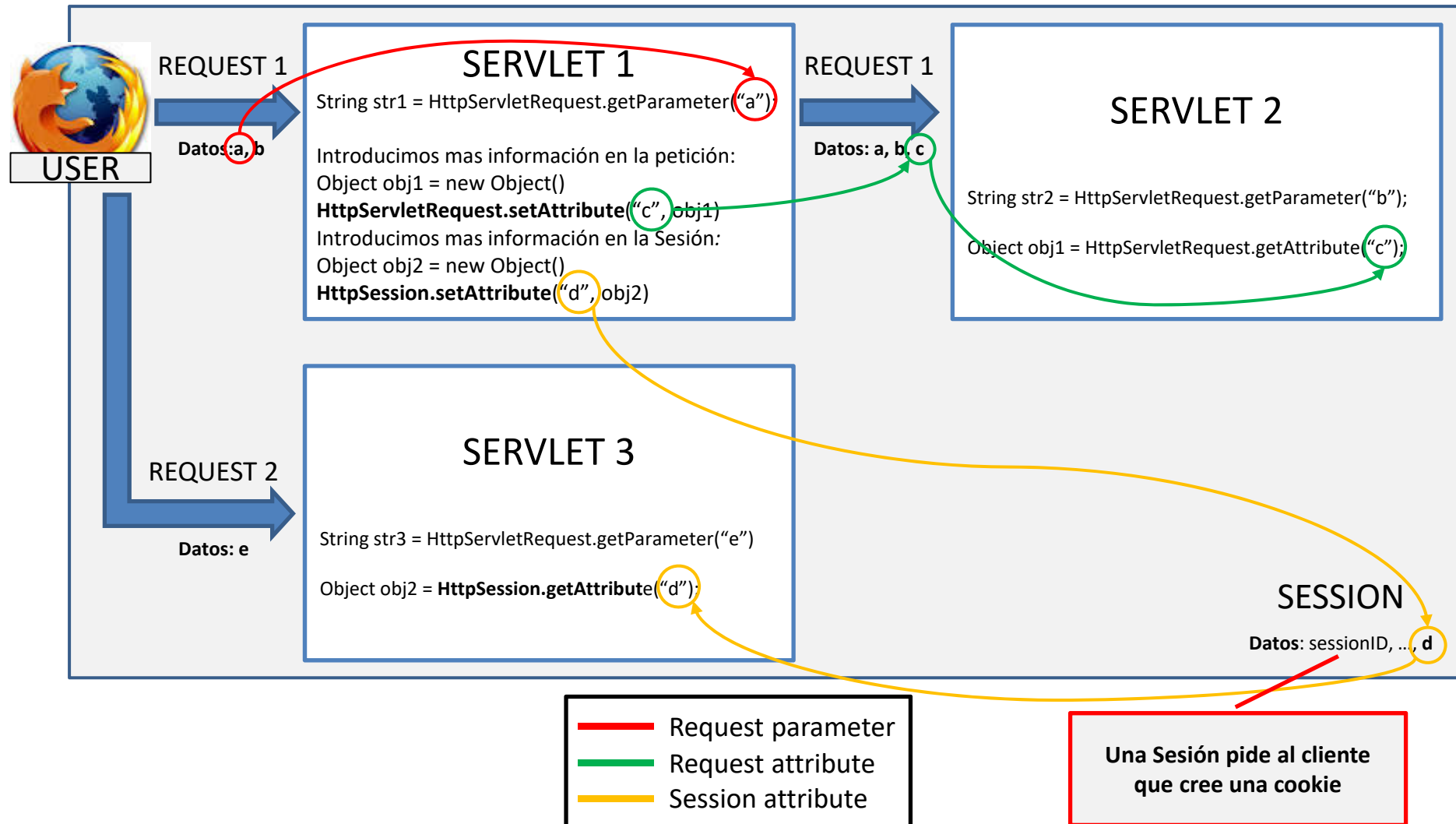
## COMPARTIR INFORMACIÓN: BD, **SESIÓN** Y CONTEXTO

```
MainServlet.java ×
35     System.out.println("---> Entrando en doPost() de MainServlet"+ request.getSession(false));
36
37     if(request.getSession(false) == null) {
38         System.out.println("    Usuario NO logeado: Redireccionar al usuario al loginForm.html");
39         response.sendRedirect("/ShareInfo/html/loginForm.html");
40
41         //RequestDispatcher rd = request.getRequestDispatcher("/html/loginForm.html");
42         //rd.forward(request, response);
43
44     } else {
45         System.out.println("    Usuario logeado");
46
47         String message = request.getParameter("message");
48         if(message != null) {
49
50             HttpSession session = request.getSession();
51             String sessionID = session.getId();
52             ServletContext context = request.getServletContext();
53
54             HashMap<String, String> loggedinUsers = (HashMap) context.getAttribute("loggedin_users");
55             System.out.println("    Usuarios Activos: " + loggedinUsers.toString());
56
57             for(Map.Entry<String, String> entry : loggedinUsers.entrySet()) {
58                 if(entry.getValue().equals(sessionID)) {
59                     String username = entry.getKey();
60                     mySQLdb.setMessageInfo(message, username);
61                     break;
62                 }
63             }
64         }
65     }
```

Con el nombre de usuario almacenado en la sesión, ¿podemos simplificar el MainServlet?"

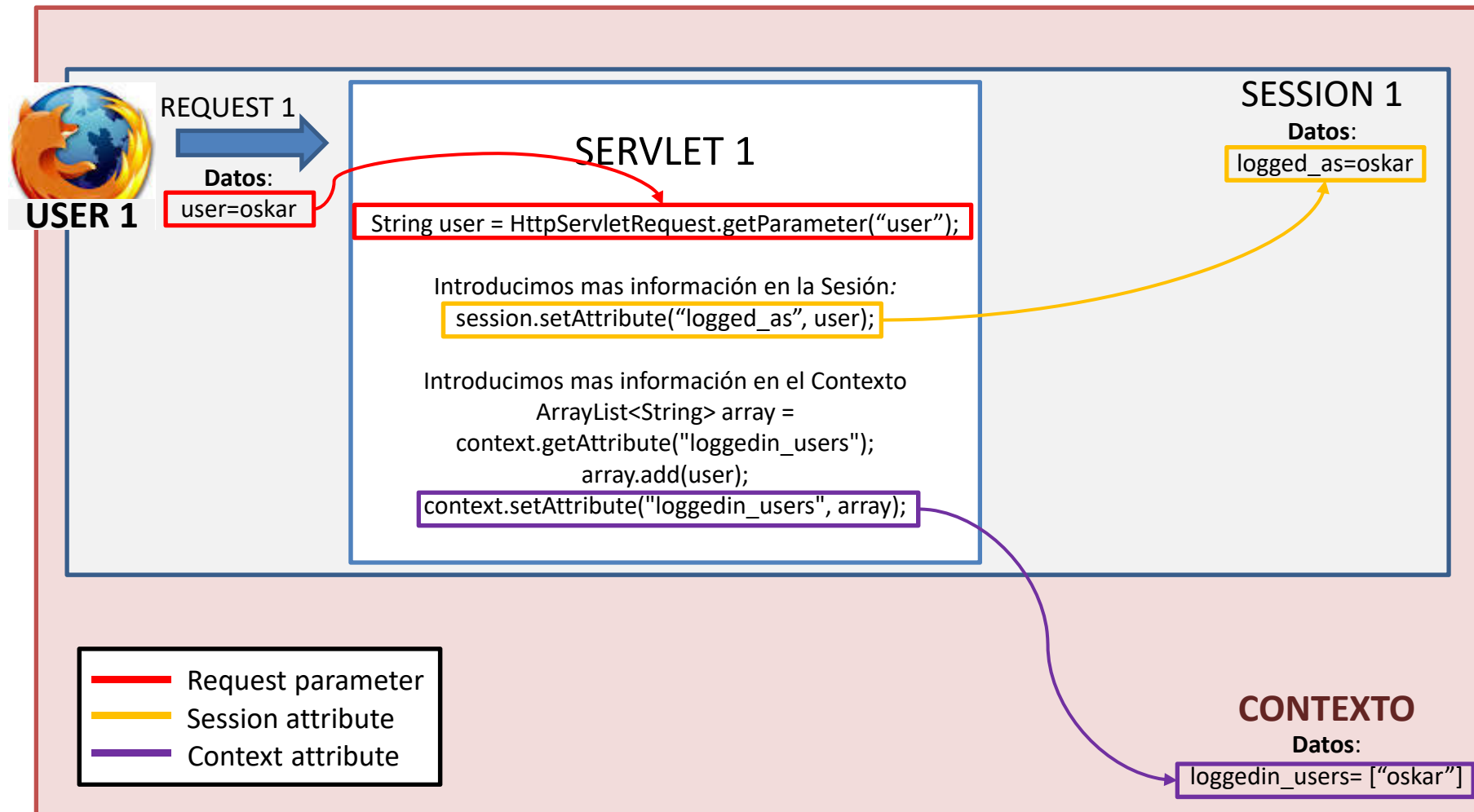
# APLICACIÓN WEB DINÁMICA – SHAREINFO

## COMPARTIR DATOS: PETICIÓN Y SESIÓN



# APLICACIÓN WEB DINÁMICA – SHAREINFO

## COMPARTIR DATOS: SESIÓN Y CONTEXTO



# APLICACIÓN WEB DINÁMICA – SHAREINFO

## COMPARTIR DATOS: SESIÓN Y CONTEXTO

