
SISTEMAS WEB

CURSO 2023/2024

Tomcat – ShareInfo III
Aplicación web para compartir mensajes



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

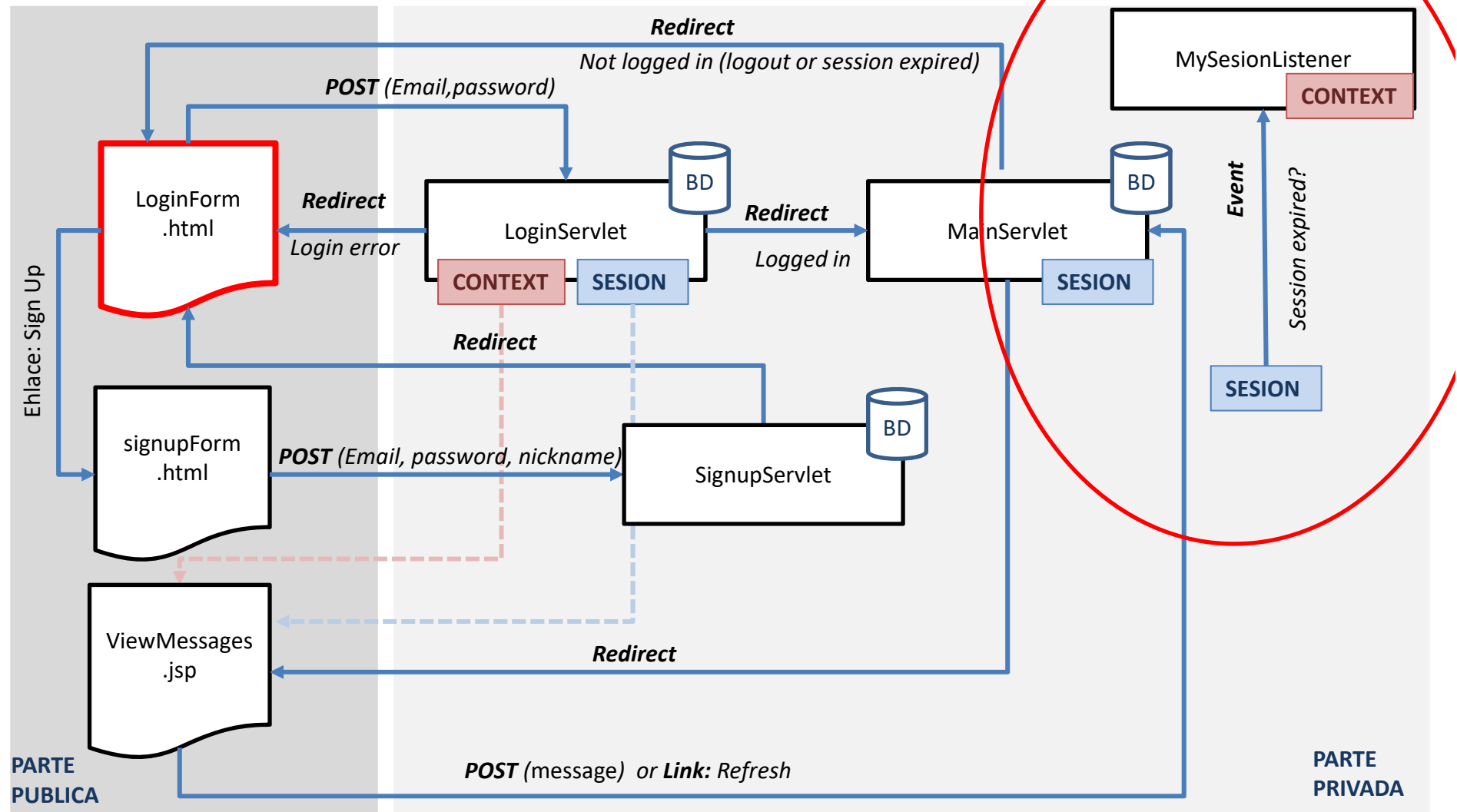
APLICACIÓN WEB DINÁMICA - SHAREINFO

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
 - Redireccionamiento
 - **Control de eventos (listeners)**
 - JSP
 - AJAX - Javascript y XML Asíncrono

APLICACIÓN WEB DINÁMICA – SHAREINFO

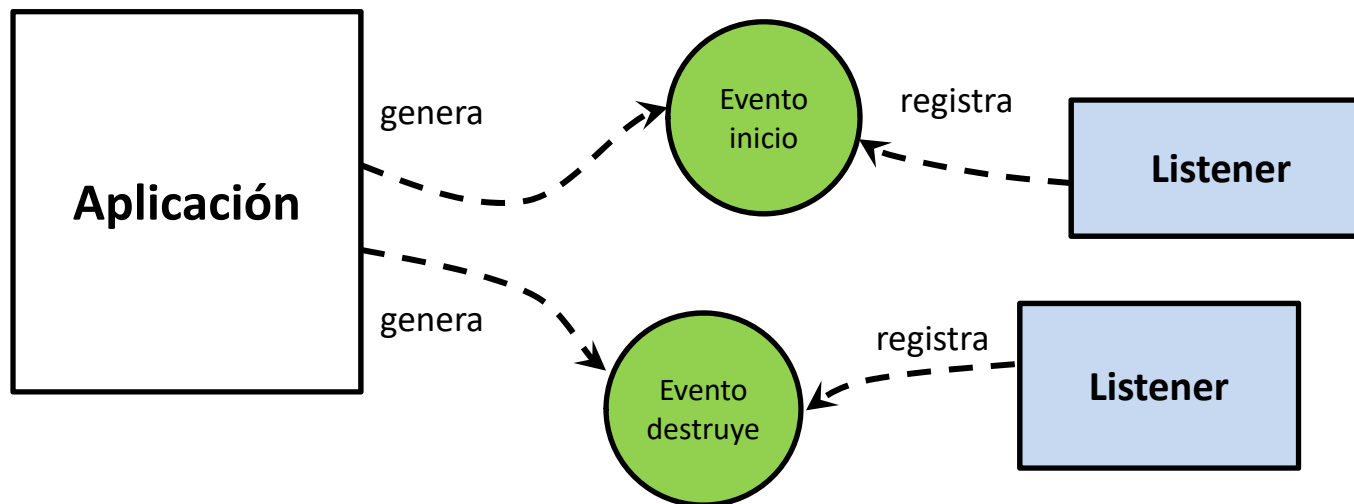
DIAGRAMA DE FLUJO DE LA APLICACIÓN



APLICACIÓN WEB DINÁMICA - SHAREINFO

EVENTS - LISTENERS

- En una aplicación web, se producen **eventos**:
 - Las **aplicaciones, sesiones y peticiones** se crean y destruyen.
 - Los **atributos** de la aplicación, sesión y petición se agregan, eliminan o modifican.
- La API de Servlet proporciona interfaces denominados **Listeners**. Estos están diseñados para **escuchar los diferentes eventos que se producen en el ciclo de vida de la aplicación web y reaccionar ante ellos**.



APLICACIÓN WEB DINÁMICA - SHAREINFO

EVENTOS - LISTENERS

**Servlet
ContextListener**

Listener que se encarga de gestionar los eventos de creación y destrucción de una aplicación (CONTEXT)

**HttpSession
Listener**

Listener que se encarga de gestionar los eventos de creación, invalidación y destrucción de sesiones.

**Servlet Request
Listener**

Listener que se encarga de los eventos de creación y destrucción de **peticiones**.

Creación y destrucción

**ServletContext
AttributeListener**

Listeners que se encargan de los eventos que se crean al añadir, eliminar o modificar un atributos de contexto, sesión o petición

**HttpSession
AttributeListener**

**ServletRequest
AttributeListener**

Atributos

EJEMPLO - SHAREINFO

HTTPSESSIONLISTENER

- Para ejecutar acciones al crear o cerrar una sesión, es necesario escribir la clase que implemente el interface **HttpSessionListener**, en la cual se definen los métodos **sessionCreated()** y **sessionDestroyed()**.
- En *ShareInfo*, se ha definido *MySessionListener*. Este se encarga de controlar un evento de sesión. Cuando un usuario abandona la sesión se encarga de eliminar el usuario del atributo de contexto "loggedin_users".
- Necesario configurar esta clase en web.xml

```
<listener>  
  <listener-class>shareinfo.MySessionListener</listener-class>  
</listener>
```

https://docs.oracle.com/cd/E29582_01/df.131/df_api_ref/com/endeca/portal/session/SessionListener.html

EJEMPLO - SHAREINFO

HTTPSESSIONLISTENER

```
public class MySessionListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent event) {
        System.out.println("    A session is being created");
    }

    public void sessionDestroyed(HttpSessionEvent event) {
        System.out.println("    A session is being destroyed");

        HttpSession session = event.getSession();
        String sessionID = session.getId();
        ServletContext context = session.getServletContext();
        HashMap<String, String> loggedInUsers = (HashMap<String, String>)
        context.getAttribute("loggedin_users");

        for(Map.Entry<String, String> entry : loggedInUsers.entrySet())
        {
            if(entry.getValue().equals(sessionID)) {
                loggedInUsers.remove(entry.getKey());
                context.setAttribute("loggedin_users", loggedInUsers);
                break;
            }
        }
    }
}
```

Borra el usuario que ha salido de la sesión del atributo de contexto "loggedin_users",

MySessionListener.java

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
 - Redireccionamiento
 - Control de eventos (listeners)
 - **JSP**
 - AJAX - Javascript y XML Asíncrono

APLICACIÓN WEB DINÁMICA - SHAREINFO

GENERAR UN JSP – COMPARTIR DATOS ENTRE PETICIONES

Web para Compartir Mensajes Cortos

Formulario Login

Email:

Password:

Enviar

Registrarse

Sistemas Web - Escuela Ingeniería de Bilbao

Web para Compartir Mensajes Cortos

Formulario Login

Información: Usuario no registrado

Email:

Password:

Enviar

Registrarse

Sistemas Web - Escuela Ingeniería de Bilbao

Modificamos la aplicación

APLICACIÓN WEB DINÁMICA - SHAREINFO

GENERAR UN JSP – COMPARTIR DATOS ENTRE PETICIONES

- Necesitamos que la pagina LoginForm.html sea diferente dependiendo de si el usuario se ha logeado bien o no. Para ello convertir LoginForm.html en LoginForm.jsp y añadimos el código java.

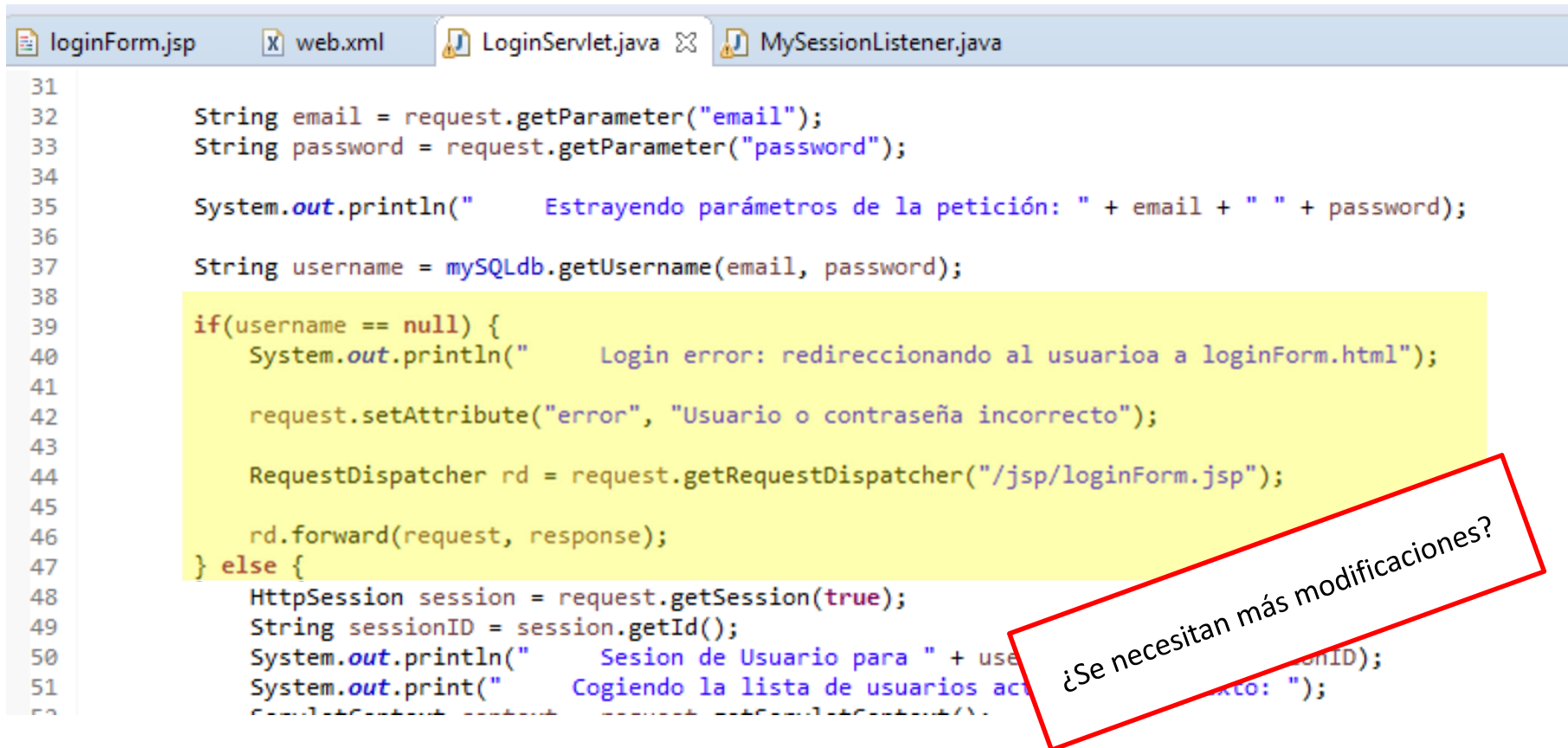


```
loginForm.jsp web.xml LoginServlet.java MySessionListener.java
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Formulario de Acceso</title>
5     <link href="/ShareInfo/css/styleSheet.css" rel="stylesheet" />
6   </head>
7   <body>
8     <header>
9       <h1>Compartir Mensajes Cortos</h1>
10      <h2>Formulario Login</h2>
11    </header>
12
13    <section>
14      <font>Información:
15        <%=request.getAttribute("error") %>
16      </font>
17    </section>
18
```

APLICACIÓN WEB DINÁMICA - SHAREINFO

GENERAR UN JSP – COMPARTIR DATOS ENTRE PETICIONES

- En *LoginServlet.java* definir atributo de petición “error” con información y redireccionar a *loginForm.jsp* si el usuario no esta registrado.



```
loginForm.jsp  web.xml  LoginServlet.java  MySessionListener.java

31
32  String email = request.getParameter("email");
33  String password = request.getParameter("password");
34
35  System.out.println("    Estrayendo parámetros de la petición: " + email + " " + password);
36
37  String username = mySQLdb.getUsername(email, password);
38
39  if(username == null) {
40      System.out.println("    Login error: redireccionando al usuario a loginForm.html");
41
42      request.setAttribute("error", "Usuario o contraseña incorrecto");
43
44      RequestDispatcher rd = request.getRequestDispatcher("/jsp/loginForm.jsp");
45
46      rd.forward(request, response);
47  } else {
48      HttpSession session = request.getSession(true);
49      String sessionId = session.getId();
50      System.out.println("    Sesion de Usuario para " + username + " con ID: " + sessionId);
51      System.out.print("    Cogiendo la lista de usuarios activos: ");
52      ServletContext context = request.getServletContext();
53      List<User> users = (List<User>) context.getAttribute("users");
54      for (User user : users) {
55          if (user.getUsername().equals(username) && user.getPassword().equals(password)) {
56              System.out.println("    Usuario encontrado: " + user.getUsername());
57              session.setAttribute("user", user);
58              response.sendRedirect("/jsp/home.jsp");
59              return;
60          }
61      }
62      System.out.println("    Usuario no encontrado");
63      request.setAttribute("error", "Usuario o contraseña incorrecto");
64      RequestDispatcher rd = request.getRequestDispatcher("/jsp/loginForm.jsp");
65      rd.forward(request, response);
66  }
67
68  }
```

¿Se necesitan más modificaciones?

EJERCICIOS

- Añadir en la vista de mensajes:
 - La posibilidad de **cerrar la sesión** al usuario.
 - Mostrar en el pie de página la fecha y la hora en el servidor y en el cliente.
- Mostrar en la pagina de login además de la información de error de usuario y contraseña incorrecto:
 - El usuario ha **cerrado la sesión**