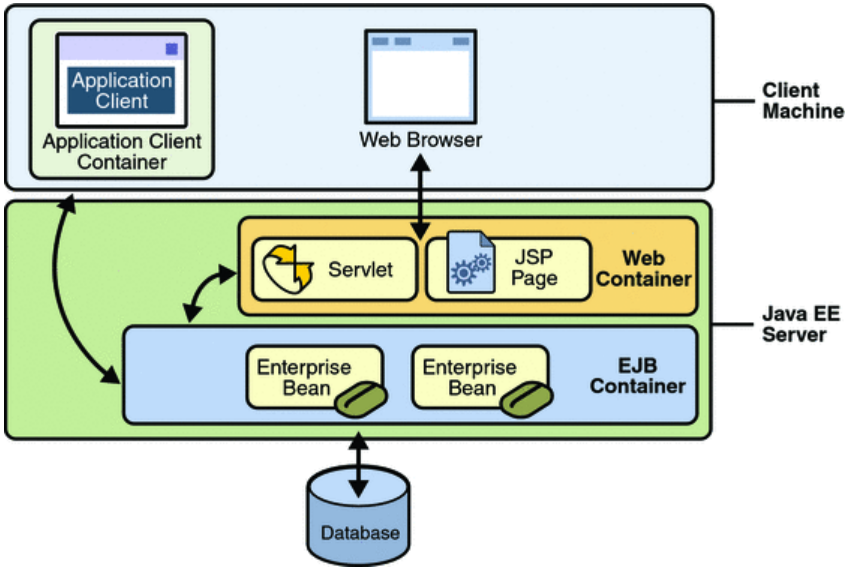


Servlets+JSP

- Separando lógica(procesado) y presentación -

ÍNDICE DE CONTENIDOS

SERVLET + JSP

Servlets

Adecuados para el procesamiento (la lógica de la aplicación)

JSP

Adecuados para la presentación.

SERVLET + JSP

Servlets

- Procesado de petición
- Acceso a BD
- Lógica de Aplicación.

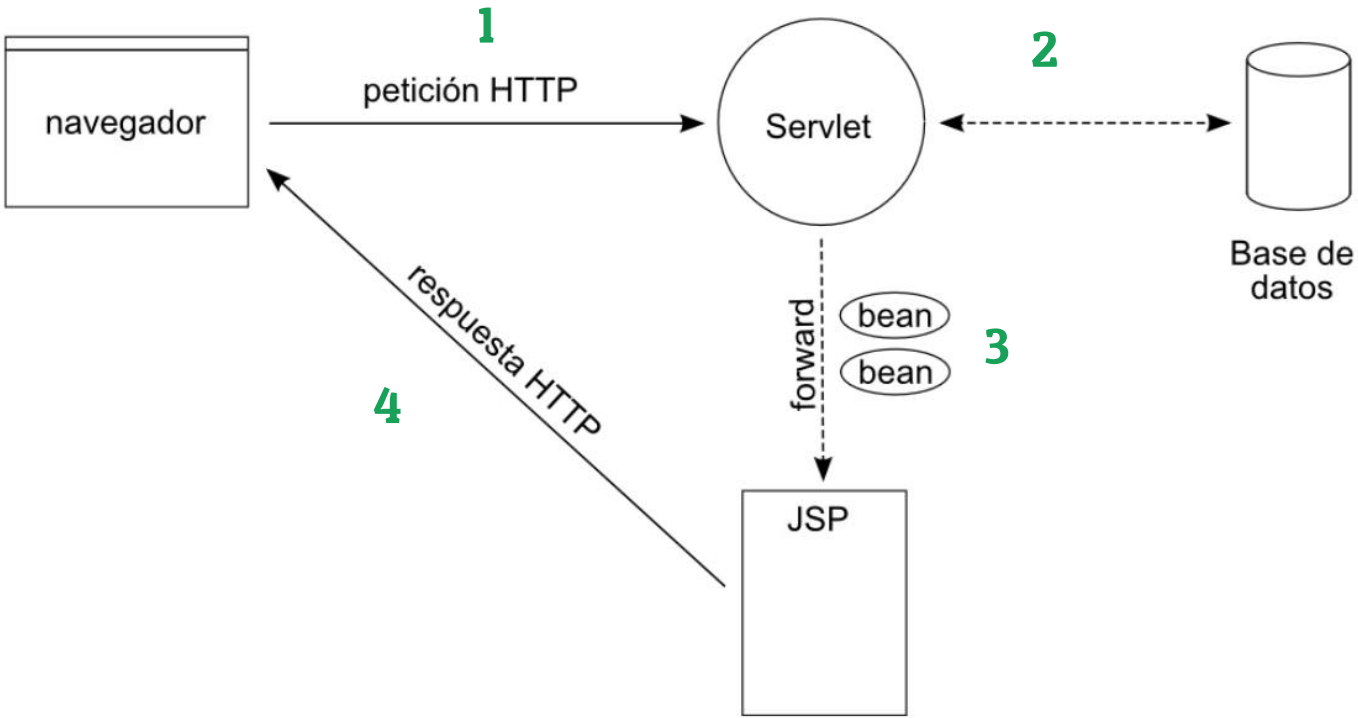
JSP

- Presentación (vistas)

FUNCIONAMIENTO

1. El cliente envía la petición al Servlet
2. El servlet procesa la petición, si es necesario accede a BD.
3. El servlet redirige la petición a un JSP. Si es necesario pasa Beans como parámetros.
4. El JSP lee los parámetros y devuelve la respuesta formateada (HTML) al usuario.

FUNCIONAMIENTO



MECANISMOS DE REDIRECCIÓN

REDIRECT	FORWARD
.sendRedirect() El servidor envía al cliente un código 3XX y la nueva URL a la que enviar una nueva petición GET.	• Redirecciones internas del servidor. • Se redirige petición a otro elemento de la misma aplicación que el que elabora la respuesta HTTP. • Transparente a cliente (no ve el cambio)

MECANISMOS DE REDIRECCIÓN

REDIRECT

```
// Redirección URI absoluta
response.sendRedirect("http://www.ejemplo.es/");

//Redirección con URL relativa
response.sendRedirect("otro.jsp");
```

MECANISMOS DE REDIRECCIÓN

FORWARD

```
// Forward desde un Servlet
RequestDispatcher rd = request.getRequestDispatcher("vista.jsp")
rd.forward(request,response);

//Forward desde JSP
<jsp:forward page="vista.jsp"/>
```

MECANISMOS DE REDIRECCIÓN

FORWARD CON PARÁMETROS

```
// Añado los parámetros en el servlet
Noticia nuevaNoticia = .....
request.setAttribute("noticia",nuevaNoticia);
RequestDispatcher rd = request.getRequestDispatcher("vista.jsp")
rd.forward(request,response);
```

MECANISMOS DE REDIRECCIÓN

FORWARD CON PARÁMETROS

// Recogida del parámetros desde un Servlet

Noticia nuevaNoticia = (Noticia) request.getAttribute(“noticia);

//Recogida de parámetros desde un JSP

‹jsp:useBean id=”noticia” class=”beans.Noticia” scope=”request” /›

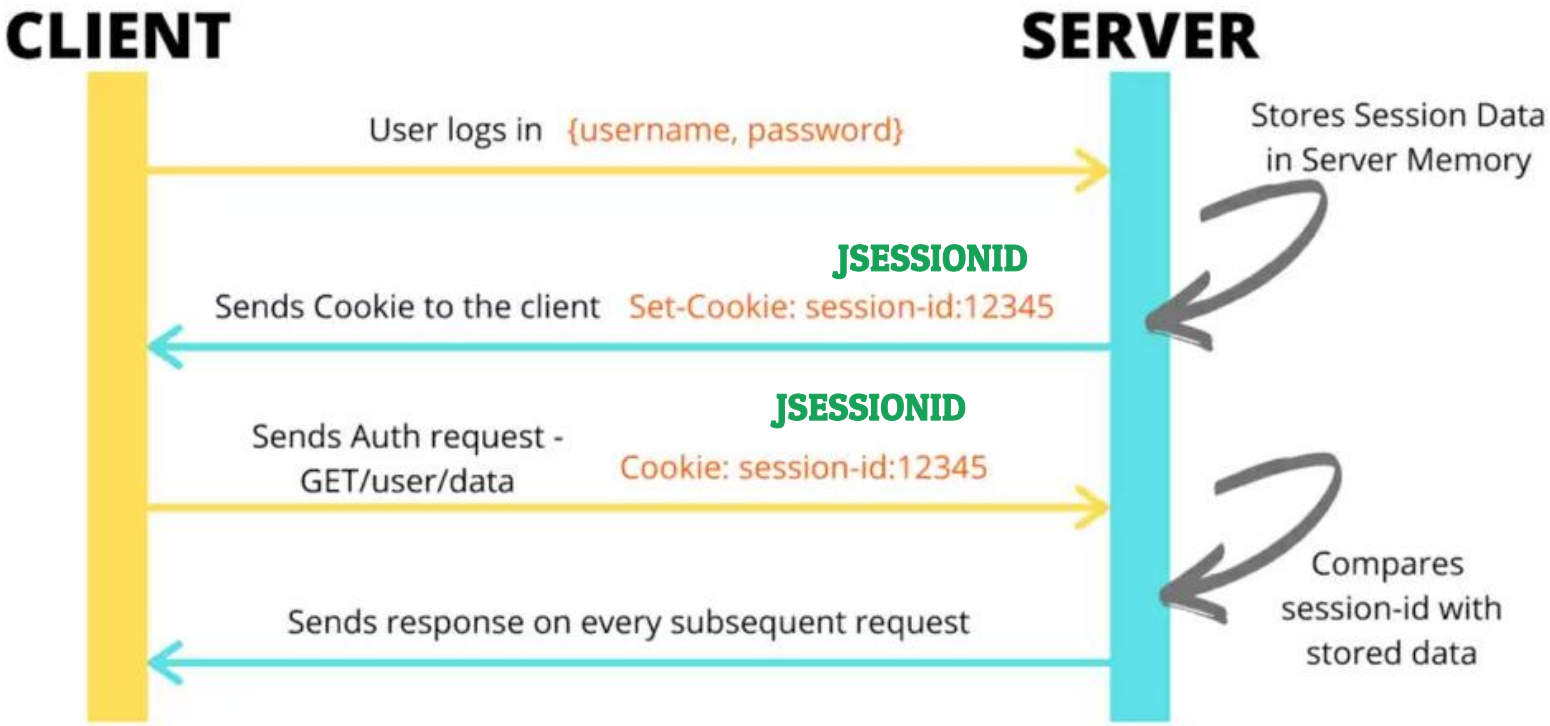
SESSIONS

> Mecanismo para agrupar distintas peticiones por pertenecer (por ejemplo) a un mismo usuario.

SESSIONS

> Hay distintos métodos para ello, nosotros nos vamos a centrar en el método basado en cookies que asocia un ID de sesión con unos datos en el servidor. Lo usaremos para autentificación.

SESSIONS


CLIENT
SERVER
User logs in {username, password}
Stores Session Data in Server Memory
JSESSIONID
Sends Cookie to the client Set-Cookie: session-id:12345
JSESSIONID
Sends Auth request - GET/user/data
Cookie: session-id:12345
Compares session-id with stored data
Sends response on every subsequent request

SESSIONS

- Usaremos *jakarta.servlet.http.HttpSession*
- Crear una sesión:
  - *HttpSession sesion = request.getSession()*
  - Si no existe la crea!!!!!!!!!
  - Si existe devuelve la sesión.
  - *HttpSession sesion = request.getSession(false)*
  - Si existe devuelve la sesión.
  - Si no existe devuelve null!!!!
  - **sesion.isNew()** -> True si la sesión es nueva, aun no es conocida por el cliente

SESSIONS

- Crear o modificar atributos de la sesión.
  - *session.setAttribute(nombre,valor)*
- Consulta de valores de atributos de la sesión.
  - Objecto obj = sesion.getAttribute(nombre)
- Consulta de todos los nombre de atributos de la sesión
  - *Enumeration<String> e = sesion.getAttributeNames()*

ELIMINANDO SESIONES

TimeOut

La sesión ha caducado ya que se ha sobrepasado el tiempo establecido

Método Propio

Forzamos la eliminación de la sesión llamando al método **invalidate()** de la misma.

¿DÓNDE?

Fallo o salida de la aplicación

Salimos del navegador o si la aplicación falla la sesión deja de ser válida. No es algo recomendable como método....

SESSIONS TIMEOUT

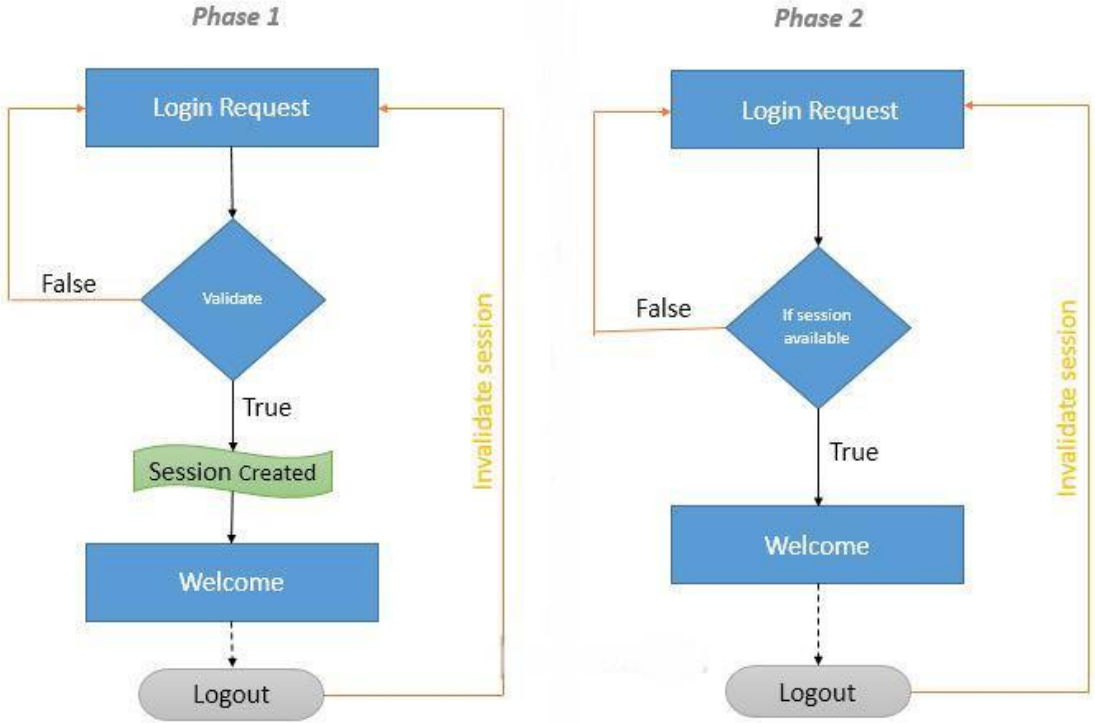
```
//De manera programática (Segundos)
session.setMaxInactiveInterval(xxxxxxx);

//En el descriptor de despliegue web.xml
<web-app...>
                ...
        <servlet>
                ...
        </servlet>
        <session-config>
                <session-timeout>XXX</session-timeout>
        </session-config>
</web-app>
```

OTROS MÉTODOS INTERESANTES

- .getCreationTime().
- .getLastAccesedTime().
- getMaxInactiveInterval(),.

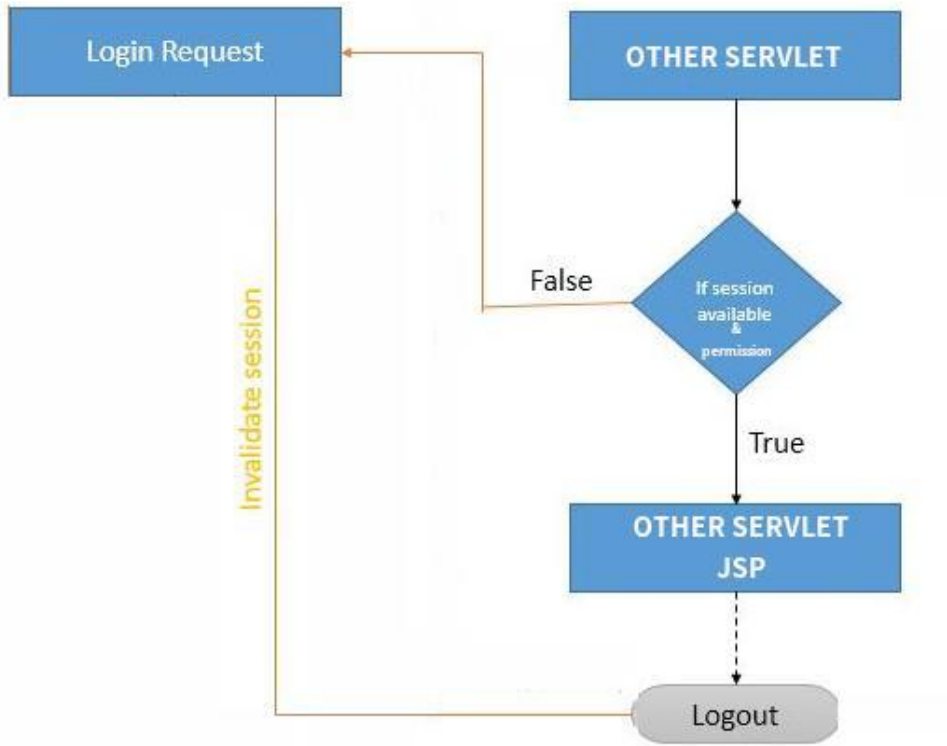
¿PROCESO DE LOGIN / LOGOUT USANDO SESIONES?

SESSIONS



En la página de Login

SESSIONS



En otras páginas que requieren gestión de sesiones

COOKIES

> Las sesiones “desaparecen” cuando caducan o cuando cerramos el navegador. Pero si usamos “cookies” para intercambiar valores entres cliente y servidor de forma casi permanente…..

COOKIES

> El proceso es similar a la sesión. Una vez se ha almacenado la cookie en el cliente se manda de vuelta al servidor en las cabeceras de las peticiones.

COOKIES

- Crear una cookie:
  - *Cookie cookie = new Cookie("nombre",valor);*
- Establecer la duración de una cookie (segundos)
  - *cookie.setMaxAge(XXXX);*
- Enviar una cookie al cliente.
  - response.addCookie(cookie).
- Obtener las cookies de la petición del cliente.
  - *Cookie[] cookies = request.getCookies();*
- Obtener el valor de una cookie.
  - *cookie.getValue();*

OTROS MÉTODOS DE COOKIES

- cookie.getDomain();
- cookie.getMaxAge();
- cookie.getName();
- cookie.getPath();
- .....

EJEMPLOS

END

jgardur081@g.educaand.es