Tecnologías en el Servidor
Java - JSP

Curso 2020/21

Javier Albert Segui

Universidad de Alcalá

Índice

- Introduccion a JSP
- Ciclo de vida de las páginas JSP
- Elementos JSP
- Manejo de sesiones desde JSP

JSP

- JSP nos permite mezclar, en una misma página, código HTML para la generación de la parte estática de la misma, con contenido dinámico generado a partir de unas marcas especiales. **<% %>**
- El contenido dinámico se obtiene gracias a la posibilidad que tenemos de incrustar dentro de la página de código en JAVA
- Nos permite acceso a Bases de datos remotas para las operaciones CRUD
- Permite la gestión de sesiones en el navegador.

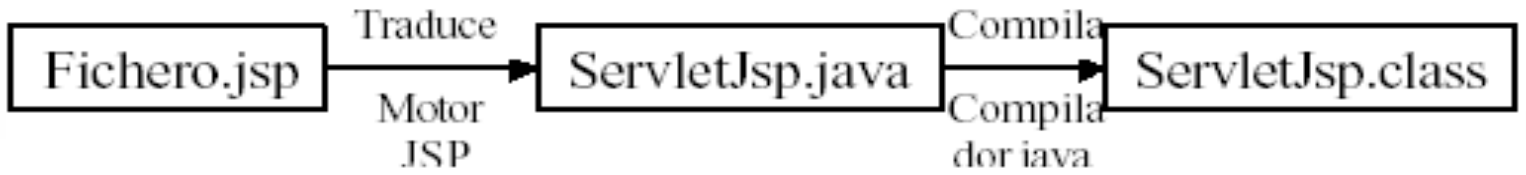
JSP

- El primer borrador de la especificación, por parte de *Sun Microsystems*, vio la luz en el año 1998, apareciendo la especificación 1.0 en el año 1999
- La versión 1.1 apareció a finales del mismo año
- En la actualidad estamos con la versión 2.3.x en producción y con la próxima publicación de la versión 3.0 ya como parte del proyecto Jakarta.
- Toda la información podemos encontrarla en la web del proyecto de la fundación Eclipse:

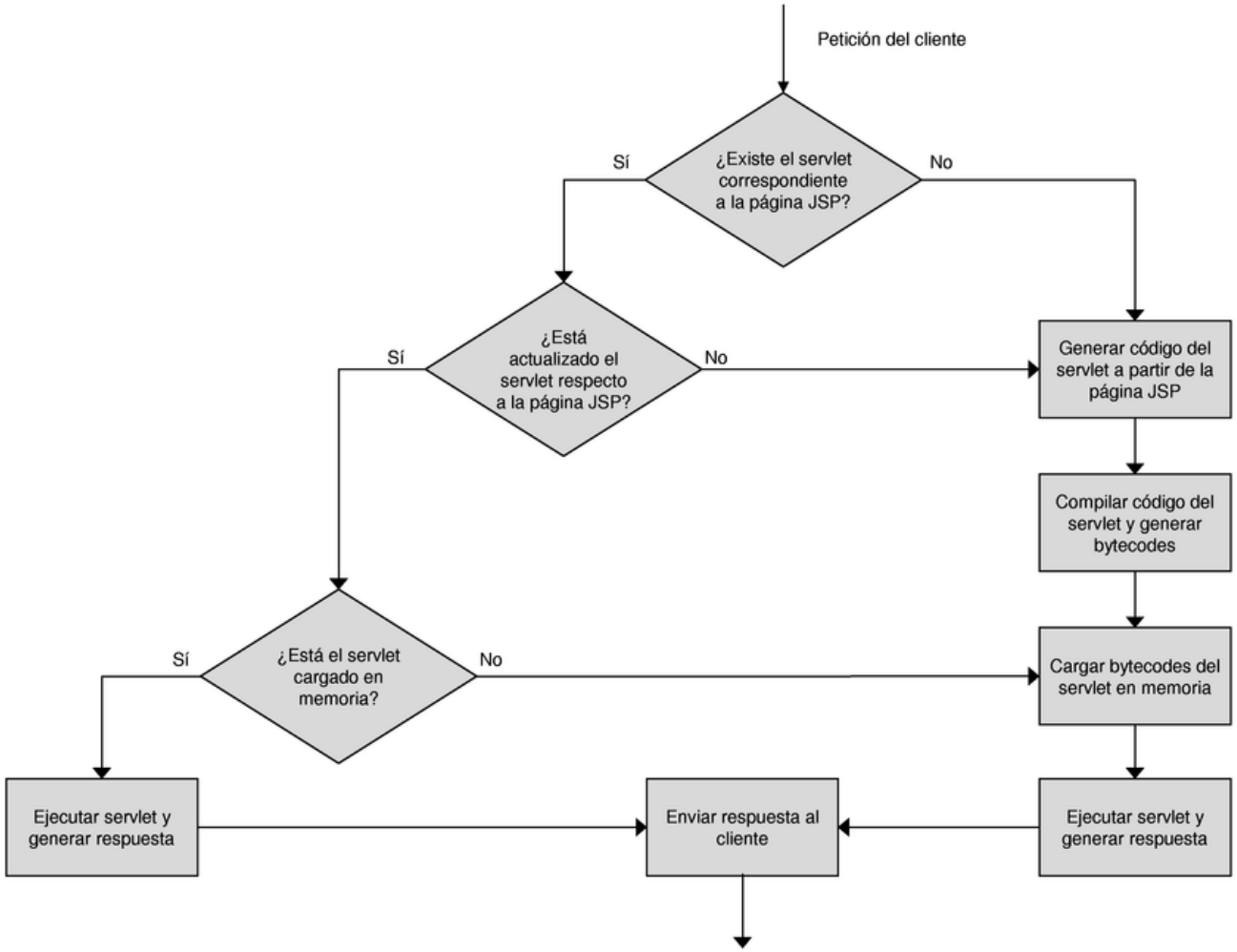
  https://projects.eclipse.org/projects/ee4j.jsp

JSP

- La página JSP se transforma automaticamente en un servlet.
- Esta conversión se realiza en el servidor donde esta desplegada la página, SOLO se realiza la primera vez que se hace la petición de la pagina, quedando ya compilado para futuros usos.
- El servlet generado será el responsable de procesar cualquier petición a la pagina JSP.
- En caso de realizar cambios en la pagina, se regenera y compila de nuevo el servlet.



JSP


Petición del cliente
¿Existe el servlet correspondiente a la página JSP?
Sí
No
¿Está actualizado el servlet respecto a la página JSP?
Sí
No
Generar código del servlet a partir de la página JSP
Compilar código del servlet y generar bytecodes
¿Está el servlet cargado en memoria?
Sí
No
Cargar bytecodes del servlet en memoria
Ejecutar servlet y generar respuesta
Enviar respuesta al cliente
Ejecutar servlet y generar respuesta

JSP

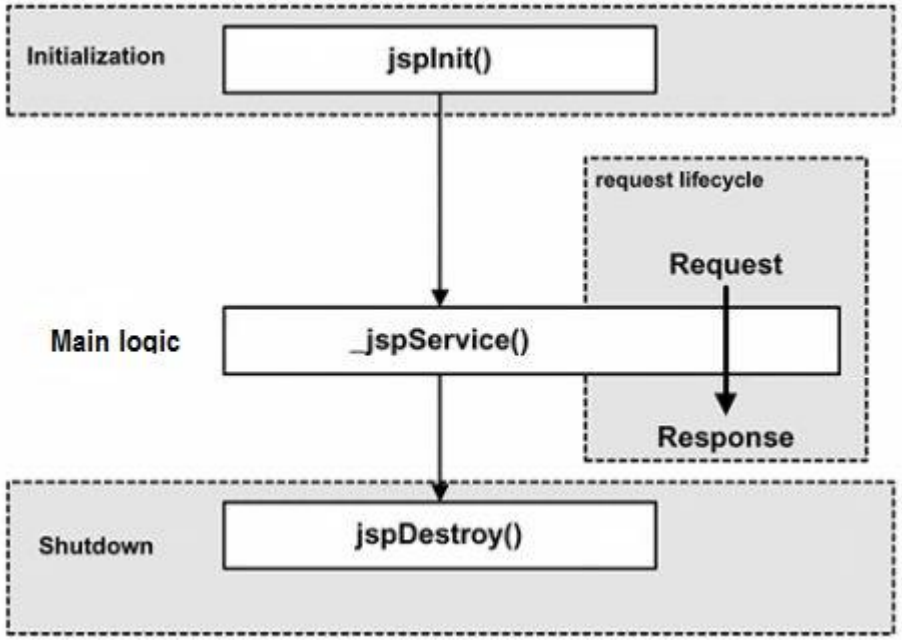
```
<%@ page info="Un ejemplo Hola Mundo" import="java.util.Date"
%>
      <HTML>
      <head> <title> Hola, Mundo </title> </head>
      <body> <h1> ¡Hola, Mundo! </h1>
      La fecha de hoy es: <%= new Date().toString() %>
      </body>
      </HTML>
```

JSP – JSP to Servlet

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse  response)
                        throws java.io.IOException, ServletException {
   JspWriter out = null;
   response.setContentType("text/html;ISO-8859-1");
   out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0
               Transitional//EN\">);
   out.println("<html>");
   out.println("<head>");
   out.println("<title>Hola, Mundo</title>");
   out.println("</head>");
   out.println("<body>");
   out.print("La fecha de hoy es: ");
   out.println(new java.util.Date());
   out.println("</body>");
   out.println("</html>");
}
```

Ciclo de vida JSP

- Cuando llamamos por primera vez y se produce el análisis y compilación de nuestra pagina JSP, se genera un servlet, interno al contenedor de servlets, en el que se implementan las siguientes operaciones:
  - jspInit()
    - Inicializa el servlet generado
    - SOLO se llama en la primera ejecución o cuando se cambia el código de la pagina
  - _jspService()
    - Maneja las peticiones.
    - Se invoca en cada petición.
  - jspDestroy()
    - Se invoca por el motor para destruir el servlet



Estructura de un JSP

- Desde la especificación JSP 2.0, las páginas JSP pueden ser plantillas de texto sin formato específico o documentos XML, aunque esta nueva notación está aún poco extendida y es algo farragosa.
- La extensión habitual de una página JSP es **.jsp**, aunque se usa **.jspx** para las páginas que son documentos XML y **.jspf** para los fragmentos JSP.
- En la notación habitual (sin usar ningún tipo de scriptlets), una página JSP está compuesta por:
  - Contenido estático: HTML, XML, WML, texto libre, etc.
  - Comentarios JSP: `<%--… --%>`
  - Directivas JSP: `<%@ …%>`
  - Código EL (Expression Language): `${ …}`
  - Elementos JSP estándar: `<jsp:…>…</jsp:…>`
  - Elementos custom tags: JSTL, Struts, etc.

Estructura de un JSP

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<%@ page language="java" %>
<%@ page contentType="text/html" %>
<%@ page pageEncoding="ISO-8859-1" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type"
content="text/html;charset=ISO-8859-1" />
<link rel="stylesheet" type="text/css" href="styles/estilo.css" />
<title>Página JSP de ejemplo</title>
</head>
<%-- sigue en la siguiente transparencia --%>
```

Estructura de un JSP

```
<%-- viene de la transparencia anterior --%>
<body>
        <h1>Ejemplo de página JSP 2.1</h1>
        <jsp:useBean id="errores" scope="session” class="es.us.lsi.daaw.beans.ListaMensajes"
/>
        <div id="div_errores" class="errores">
            <ul>
            <c:forEach var="mensaje" items="${errores.mensajes}">
                <li>${mensaje}</li>
            </c:forEach>
            </ul>
        </div>
</body>
</html>
```

Directivas JSP <%@ page...%>

Atributo	Significado	Ejemplo
import	el equivalente a una sentencia import de Java	<%@ page import="java.util.Date" %>
contentType	genera una cabecera HTTP Content-Type	<%@ page contentType="text/plain" %>
isThreadSafe	si es false, el servlet generado implementará el interface SingleThreadModel (ún único hilo para todas las peticiones). Por defecto, el valor es true.	
session	Si es false, no se creará un objeto session de manera automática. Por defecto, es true.	
buffer	Define el tamaño del buffer para la salida (en kb), o none si no se desea buffer. Su existencia permite generar cabeceras HTTP o códigos de estado aunque ya se haya comenzado a escribir la salida.	<%@ page buffer="64kb" %>
autoflush	Si es true (valor por defecto), el buffer se envía automáticamente a la salida al llenarse. Si es false, al llenarse el buffer se genera una excepción.	
extends	Permite especificar de qué clase debe descender el servlet generado a partir de la página JSP. No es habitual cambiarlo.	
info	define una cadena que puede obtenerse a través del método getServletInfo	<%@ page info="carro de la compra" %>
errorPage	especifica la página JSP que debe procesar los errores generados y no capturados en la actual.	<%@ page errorPage="error.jsp" %>
isErrorPage	Si es true, indica que la página actúa como página de error para otro JSP. El valor por defecto es false.	
language	Permite especificar el lenguaje de programación usado en el JSP. En la práctica, el lenguaje siempre es Java, por lo que esta directiva no se usa.	
pageEncoding	define el juego de caracteres que usa la página. El valor por defecto es ISO-8859-1.	

Directivas JSP <%@ include...%>

- Incluye un archivo (normalmente un fragmento JSP) antes de que el contenedor transforme la página en un servlet.* Su único atributo es:
  - file = "URL relativa"
  - URL relativa del archivo a incluir. Si empieza por /, se considera relativo al contexto de la aplicación web.

Directivas JSP <%@ taglib...%>

- Permite usar una biblioteca de custom tags. Sus atributos son los siguientes:
  - uri = "URI de la la biblioteca"
    - URI que identifica de manera única a una biblioteca de custom tags, p.e. http://java.sun.com/jsp/jstl/core.
- prefix = "prefijoBiblioteca"
  - Prefijo para las etiquetas de los elementos definidos en la biblioteca de custom tags. No se admiten prefijos vacíos ni ninguno de los siguientes: jsp, jspx, java, javax, servlet, sun, y sunw.

Directivas JSP <%@ taglib…%>

- La Java Server Pages Standard Tag Library (JSTL), es un conjunto de 5 bibliotecas de *custom tags* estandarizadas:
  - Core (prefijo c): gestión de variables, control de flujo, gestión de URLs.
  - XML (prefijo x): similar a core, pero orientado a XML incluyendo transformación con XSL.
  - i18n (prefijo fmt): internacionalización, incluyendo formateo de fechas y números.
  - SQL (prefijo sql): acceso a bases de datos con SQL, sólo para prototipos (la capa de presentación no debe acceder directamente a los datos).
  - Funciones (prefijo fn): funciones de longitud de colecciones y manipulación de cadenas.

AREA	URI	PREFIJO
Core	http://java.sun.com/jstl/ea/core	c
XML	http://java.sun.com/jstl/ea/xml	x
Internacionalización (I18N)	http://java.sun.com/jstl/ea/fmt	fmt
SQL	http://java.sun.com/jstl/ea/sql	sql

Directivas JSP Lenguaje de Expresiones (EL)

- El Expression Language permite especificar expresiones de forma sencilla en las páginas JSP.
- Las expresiones EL son analizadas y procesadas por el contenedor de servlets al transformar la página JSP en un servlet.
- Una expresión EL tiene la forma ${expresión}, y puede incluir números, cadenas y propiedades de JavaBeans, incluyendo entre otros, los siguientes objetos implícitos* que son tratados como mapas:
  - pageScope: contexto de la página que permite acceder a su mapa y a las propiedades de los objetos servletContext, session, request y response.
  - requestScope: mapa con los atributos de request.
  - sessionScope: mapa con los atributos de sesión.
  - applicationScope: mapa con los atributos de aplicación.
  - param y paramValues: valores enviados por un formulario.

Directivas JSP Lenguaje de Expresiones (EL)

Ejemplos de uso de expresiones EL

- ${param.nombre}
  - El valor del parámetro "nombre" o null si no se ha enviado o es una cadena vacía. Equivale a ${param["nombre"]} y a ${param['nombre']}.
- ${!empty param.direccion}
  - Cierto (true) si el parámetro "dirección" se ha enviado y no es una cadena vacía. Equivalente a ${!empty param["direccion"]}.
- ${sessionScope.carrito.size}
  - El valor de la propiedad "size" de la variable de sesión "carrito". Equivale a ${sessionScope["carrito"].size}.
- ${applicationScope["numUsuarios"]}
  - El valor de la variable de aplicación "numUsuarios". Equivale a ${applicationScope.numUsuarios}.
- – ${carrito.precioTotal}
  - El valor de la propiedad "precioTotal" de la variable "carrito" si la encuentra en los ámbitos de página, request, sesión o aplicación (en ese orden). Si no laencuentra, devuelve null. Equivale a ${carrito["precioTotal"]}.

JSP – Objetos implicitos

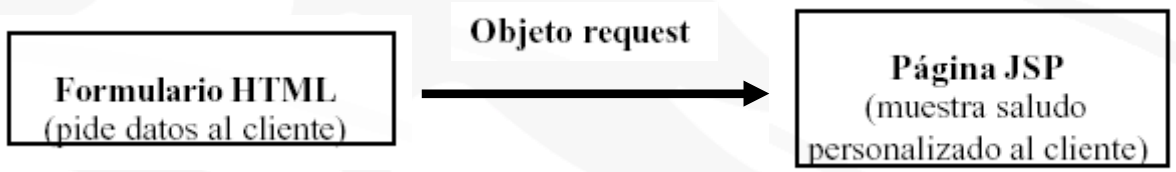
- Los objetos implícitos, son aquellos que están accesibles al motor JSP, por lo que el desarrollador de páginas JSP puede utilizarlos cuando los necesite sin más que invocarlos adecuadamente.
- Son objetos creados por el motor que no necesitan ser declarados para ser usados, sino que se pueden invocar directamente.
- Modelan mucha de la funcionalidad básica de cualquier aplicación web: tratamiento de sesiones, procesamiento de un formulario, etc.

JSP – Objetos implicitos

Objeto	Significado
request	el objeto HttpServletRequest asociado con la petición
response	el objeto HttpServletResponse asociado con la respuesta
out	el Writer empleado para enviar la salida al cliente. La salida de los JSP emplea un buffer que permite que se envíen cabeceras HTTP o códigos de estado, aunque ya se haya empezado a escribir en la salida (out no es un PrintWriter sino un objeto de la clase especial JspWriter).
session	el objeto HttpSession asociado con la petición actual. En JSP, las sesiones se crean automáticamente, de modo que este objeto está instanciado, aunque no se cree explícitamente una sesión.
application	el objeto ServletContext, común a todos los servlets de la aplicación web.
config	el objeto ServletConfig, empleado para leer parámetros de inicialización.
pageContext	permite acceder desde un único objeto a todos los demás objetos implícitos
page	referencia al propio servlet generado (tiene el mismo valor que this). Como tal, en Java no tiene demasiado sentido utilizarla, pero está pensada para el caso en que se utilizara un lenguaje de programación distinto.
exception	Representa un error producido en la aplicación. Solo es accesible si la página se ha designado como página de error (mediante la directiva page isErrorPage).

JSP – Objetos implicitos

Aplicación que pide el nombre al usuario y le devuelve un saludo. Se utiliza un fichero HTML con un formulario que pide los datos al cliente y se los pasa a una página JSP que muestra el saludo con estos datos. El paso de los datos del formulario al JSP se realiza a través de un objeto especial: **request**.




```
<HTML>
<head>
<title> Formulario de petición de nombre
</title>
</head>
<body>
<h1> Formulario de petición de nombre </h1>
<!-- Se envía el formulario al JSP “saludo.jsp” -->
<form method="post" action="saludo.jsp">
<p> Por favor, introduce tu nombre:
<input type="text" name="nombre">
</p>
<p> <input type="submit" value="enviar
información"> </form> </body>
</HTML>
```

```
<HTML>
<head>
<title> Formulario de petición de nombre</title>
</head>
<body>
<h1> Formulario de petición de nombre </h1>
<!-- Se envía el formulario al JSP “saludo.jsp” -->
<form method="post" action="saludo.jsp">
<p> Por favor, introduce tu nombre:
<input type="text" name="nombre">
</p>
<p> <input type="submit" value="enviar  información">
</form> </body>
</HTML>
```

JSP – Elementos Estándar

- Son 14 elementos que comienzan con el prefijo **jsp:**
- Algunos tienen que ver con la nueva sintaxis XML de JSP: jsp:root, jsp:body, jsp:attribute, jsp:text, ...
- Otros no se suelen usar mucho, como la inclusión dinámica (jsp:include), o el forwarding (jsp:forward, sin haber generado contenido previamente).
- Los más utilizados son:
  - ←jsp:useBean→: para usar o crear JavaBeans en cualquiera de los mapas de los posibles ámbitos (página, request, sesión o aplicación).
  - ←jsp:setProperty→: para dar valor a las propiedades de los JavaBeans declarados con ←jsp:useBean→.
- Otros como jsp:getProperty han quedado obsoletos con la aparición de las expresiones EL.

JSP – JavaBeans

- Los JavaBeans son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java.
- Se usan para encapsular varios objetos en un único objeto (Bean en inglés), para hacer uso de un solo objeto en lugar de varios más simples.
- La especificación de JavaBeans de Sun Microsystems los define como "componentes de software reutilizables que se puedan manipular visualmente en una herramienta de construcción".

JSP - JavaBeans

- Un bean no es más que una clase Java escrita siguiendo unas ciertas convenciones. Estas convenciones hacen posible que herramientas automáticas puedan acceder a sus propiedades y manipularlas sin necesidad de modificar el código. Esto puede servir en el caso de un IDE, por ejemplo, para realizar "programación visual". En JSP el uso principal de los beans es manipular componentes Java sin necesidad de incluir código en la página, accediendo a sus propiedades mediante etiquetas.
- El uso de beans en páginas JSP ofrece diversas ventajas con respecto al uso directo de código Java:
  - Se evita el uso de sintaxis Java, en su lugar se emplean etiquetas con sintaxis XML. Esto permite separar más fácilmente el trabajo de programadores y diseñadores web.
  - Se simplifica la creación y uso de objetos compartidos entre varias páginas.
  - Se simplifica la creación de objetos a partir de los parámetros de la petición

JSP - JavaBeans

- En lo que respecta a su uso con JSP, estas convenciones afectan al modo de definir constructores, métodos y variables miembro:
  - Un bean debe tener al menos un constructor sin argumentos. Este constructor será llamado cuando una página JSP cree una instancia del bean.
  - Un bean no debe tener variables miembros de acceso público. El acceso a las variables y su modificación se debe hacer a través de métodos.
  - El nombre de los métodos de acceso y modificación de variables miembro debe seguir una norma:
    - si la variable tiene el nombre nombreVar, entonces el método de acceso debe llamarse getNombreVar (obsérvese el cambio a mayúsculas de la "N", siguiendo las convenciones habituales de Java), y el método de cambio de valor (en caso de que exista) debe llamarse setNombreVar.
    - En el caso especial de variables booleanas, el método de acceso se debe denominar isNombreVar.

JSP - Gestión de sesiones

- ¡RECORDAR!

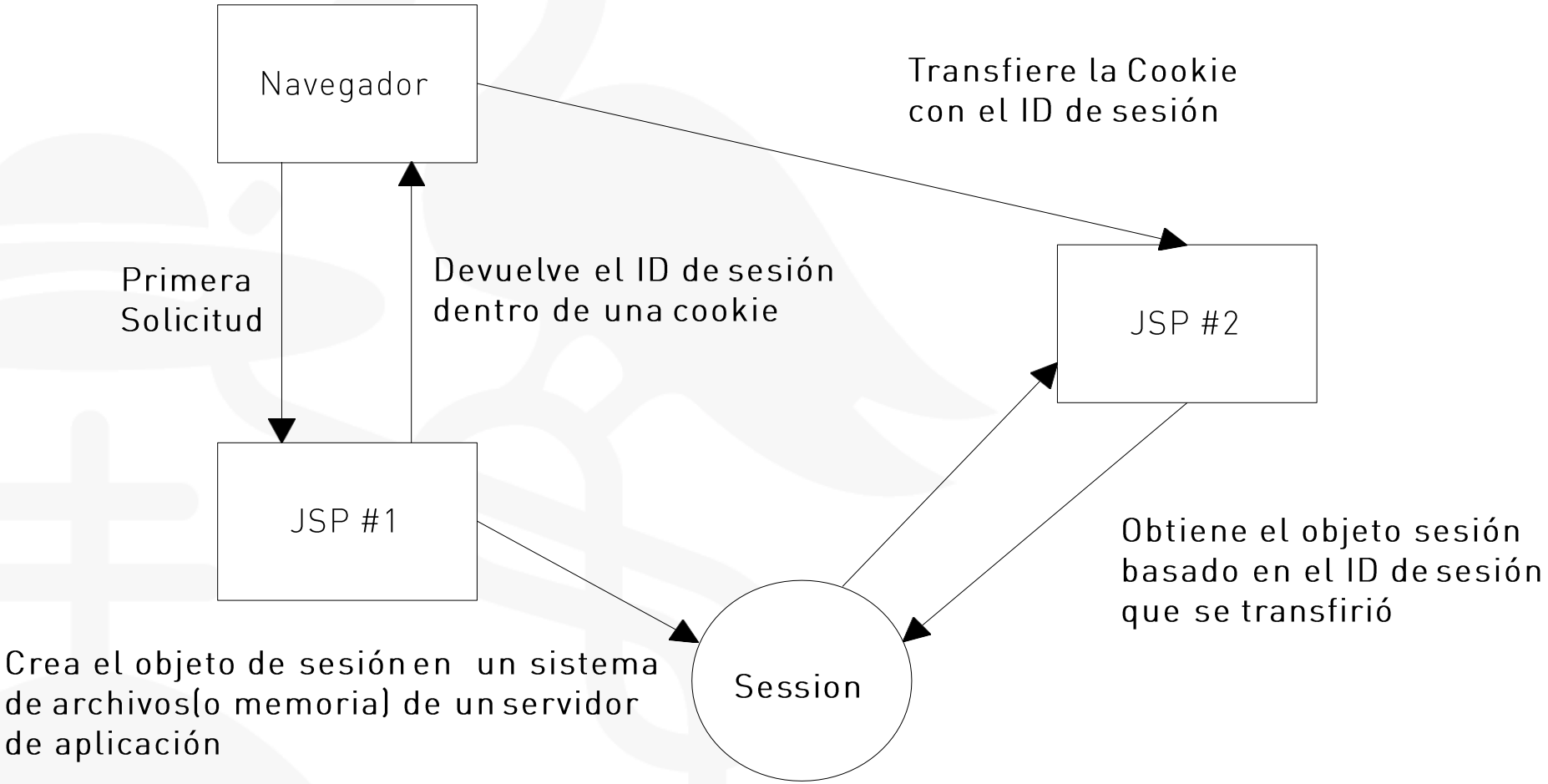
  El protocolo HTTP es un protocolo sin estado.

- No posee funcionalidad interna para seguir datos de una solicitud a otra.
- Aunque esto puede proporcionar algunos beneficios en cuanto al rendimiento, también supone un problema en algunas aplicaciones web. Por ejemplo, en aplicaciones tipo: tienda virtual, acceso a un banco, etc.

JSP – Gestión de sesiones

- La tecnología JSP proporciona acceso a un objeto implícito **session**, para almacenar y recuperar valores de una conexión lógica: **session**.
- El objeto **session** es único para cada usuario concreto. La primera vez que un usuario solicita una página jsp, se crea la sesión, esta sesión está identificada por un ID de sesión único y asociado a ella.

JSP – Gestión de sesiones



JSP – Gestión de sesiones

- El objeto session es análogo a la clase HttpSession de la tecnología servlets.
- Una vez obtenida la sesión, normalmente mediante el objeto implícito, es posible:
  - Escribir,
  - Obtener,
  - Eliminar atributos del objeto session.

JSP – Gestión de sesiones

- Escritura:
  - **<% session.setAttribute("nombre","Usuario pruebas"); %>**
- Lectura:
  - **<%String SNombre=(String) session.getAttribute("nombre"); %>**
  - Observar cómo se asigna el resultado de este método a String. Esto es necesario dado que la sesión, sólo almacena un objeto y devuelve un objeto, sin tener en cuenta el tipo de objeto que esté almacenado. Tenemos que realizar una conversión.
- Eliminar:
  - **<% session.removeAttribute("nombre"); %>**