**JSP Application Primer**

**Visión de conjunto**

Si usted no tiene experiencia en el desarrollo de aplicaciones web, este documento está pensada para ayudarle a visualizar - *en términos muy sencillos* - ¿Cómo se construye una aplicación genérica JSP, y lo que sucede en tiempo de ejecución. También identifica algunos vocabularios / conceptos clave utilizados en la Guía del desarrollador de OA Framework y en los Tutoriales de ToolBox.

**Contenido**

* [Componentes clave de la aplicación JSP](https://translate.googleusercontent.com/translate_f#key)
* [¿Qué sucede en tiempo de ejecución?](https://translate.googleusercontent.com/translate_f#runtime)
* [Manejo de eventos: interfaz de usuario web frente a interfaz de usuario clásica del cliente](https://translate.googleusercontent.com/translate_f#event)
* [Navegación de página](https://translate.googleusercontent.com/translate_f#nav)
* [¿Qué es una Cookie?](https://translate.googleusercontent.com/translate_f#cookie)
* [Más sobre sesiones de servlets](https://translate.googleusercontent.com/translate_f#sessions)

**Lectura sugerida**

Para obtener información adicional acerca de las aplicaciones y servlets JSP (ninguno de los cuales es una lectura obligatoria para trabajar con OA Framework), es posible que desee revisar lo siguiente para ver tutoriales, documentación y sugerencias de libros:

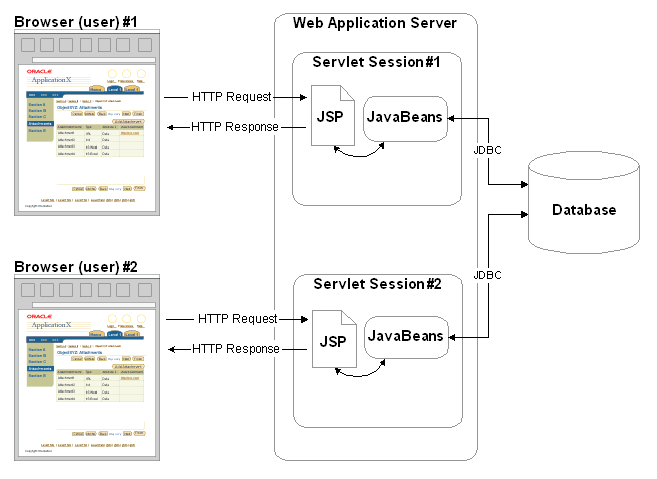
* [Red de tecnología de Oracle](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://otn.oracle.com/tech/java/content.html)
* [JavaSoft Java Developer Connection](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://developer.java.sun.com/developer/onlineTraining/)

**Componentes clave de la aplicación JSP**

Una aplicación JSP típica implica los siguientes componentes: un navegador para acceso de cliente, una base de datos para datos empresariales y un servidor de aplicaciones web ("nivel medio") donde viven los objetos de la aplicación.

* El navegador se comunica con el de nivel medio mediante **HTTP** (Hyper Text Transfer Protocol), que implica el envío de una **solicitud** mensaje al que el nivel medio responde con un mensaje de **respuesta.**
* Un **JSP** es un archivo con código HTML y Java que se ejecuta de arriba abajo. En tiempo de ejecución, que se compila en una clase Java, que es en realidad un **servlet.**
* Un **servlet** es un programa de extensión de servidor de aplicaciones web basado en Java que implementa una API estándar.
* Una **sesión de servlet** es un mecanismo para mantener el estado entre las solicitudes HTTP durante un período de interacción continua entre un navegador y una aplicación web. Una sesión puede ser iniciada en cualquier momento por la aplicación y terminada por la aplicación, por el usuario que cierra el navegador, o por un período de inactividad del usuario. Una sesión generalmente corresponde a un ciclo de inicio / cierre de sesión de la aplicación
* Un **JavaBean** (o "bean" para abreviar) es simplemente un componente reutilizable que implementa patrones de diseño específicos para facilitar que los programadores y las herramientas de desarrollo descubran las propiedades y el comportamiento del objeto.
* Todos los objetos de la clase media que se comunican con la base de datos utilizan un **JDBC** (Conectividad de la base de datos de Java).

Figura 1: componentes clave de la aplicación web y comunicación entre el navegador y el servidor



**¿Qué sucede en tiempo de ejecución?**

**Paso 1**

Cuando el usuario selecciona un enlace, un botón o una imagen activa, el navegador envía una solicitud HTTP al servidor de aplicaciones web para su procesamiento. A los efectos de esta introducción, nos centraremos en los dos métodos de solicitud HTTP POST y primaria (GET) que son relevantes para una aplicación Marco de la OA.

**HTTP GET**

Cada vez que el usuario hace clic en un vínculo o una imagen con una URL asociada (como http://www.yahoo.com) el navegador envía una solicitud GET.

Puedes pensar en un Recibe como una postal: tanto la dirección (URL) y cualquier información que el emisor quiere transmitir (parámetros de URL) se escriben en la tarjeta en sí (que es inherentemente con límite de espacio, ¿cuánto se puede escribir en una tarjeta postal?). Esto significa que toda la información para la comunicación es visible externamente (y en un HTTP GET, todos los datos enviados al servidor es visible como parámetros en la URL).

**HTTP POST**

Cada vez que el usuario hace clic en un botón, imagen o vínculo que realiza un envío de formulario en una aplicación marco OA (ver [¿Qué es un formulario?](https://translate.googleusercontent.com/translate_f#an13) Abajo), el navegador envía una solicitud POST al servidor (técnicamente, un formulario puede ser presentado con un GET , pero a los efectos de trabajar con el Marco de la OA, se puede asumir una forma presentar es un POST).

Se puede pensar en un puesto como un sobre: la dirección (URL) que está escrito en el exterior, pero el contenido tiene dentro de la información que el emisor quiere transmitir. No hay límite para la cantidad de información que se puede almacenar dentro del sobre.Además, los datos enviados no son visibles en la URL, del mismo modo que los contenidos de una envolvente no son visibles externamente (aunque la metáfora no es absolutamente precisa: un desarrollador también podría agregar algunos parámetros a la URL al enviar un formulario) .

**¿Qué es una "Forma"?**

En términos simples, un "formulario" le permite recopilar datos ingresados ​​por los usuarios en "campos de formulario" y enviarlos al servidor para su procesamiento.

Un formulario es un constructo HTML que agrupa los controles de entrada de datos, como los campos (tanto ocultos como visibles), las listas emergentes, etc., con controles de acción (como botones) que son capaces de "enviar el formulario". Cuando el usuario selecciona un botón de enviar, por ejemplo, el navegador envía una solicitud POST, que envía los datos del formulario al servidor.

**Consejo:** Las personas a menudo usan los términos POST y "Enviar formulario" indistintamente cuando se habla sobre el Marco de OA.

**Paso 2**

El oyente HTTP en el servidor de aplicaciones web enruta la solicitud entrante al JSP. El código del desarrollador no sabe ni le importa si el navegador emitió un POST o GET. Todo lo que hace es leer los valores de solicitud para determinar qué hacer. Así, por ejemplo, uno de los valores de petición puede decir la JSP que un botón Ir había sido presionado, lo que significa que debe ejecutar una consulta.

**Paso 3**

Como se muestra en [la figura 1](https://translate.googleusercontent.com/translate_f#figure01) anteriormente, los delegados JSP a uno o más JavaBeans, que implementan diversos comportamientos incluyendo la interacción de base de datos. Una vez que han completado su trabajo, el JSP prepara el contenido HTML apropiado para enviar de vuelta al navegador en la respuesta.

**Nota:** Se incluyeron los JavaBeans en este ejemplo sólo para hacer el punto de que en una aplicación de cualquier complejidad - y modularidad - la JSP no hace el trabajo de aplicación en su propia ya que no debe combinar el modelo, la vista y el código del controlador de el mismo archivo. Sin embargo, no existe un requisito técnico absoluto para que el JSP funcione con otras clases de Java, y si lo hace, no hay ningún requisito de que estas clases sean JavaBeans.

**Etapa 4**

El navegador muestra el HTML que recibió en la respuesta.

**Manejo de eventos: interfaz de usuario web frente a interfaz de usuario clásica del cliente**

En las aplicaciones cliente / servidor tradicionales, tiene la opción de manejar eventos que varían en granularidad desde movimientos de mouse de muy bajo nivel hasta campo, región y, finalmente, ámbito de ventana. Además, cuando se comunica desde el cliente al servidor, puede enviar un único valor para ser validado de vuelta al servidor mientras espera un único resultado de validación. A continuación, puede modificar la interfaz de usuario en consecuencia, lo que permite una experiencia altamente interactiva para el usuario.

En una aplicación web, esencialmente maneja eventos de "nivel de página" (a menos que esté usando Javascript extensamente para crear una experiencia más interactiva, y dado que las Pautas de la interfaz de usuario de OA Framework Coding y Oracle Browser Look and Feel (BLAF) lo prohíben, no lo consideraremos aquí). En este caso, a medida que los usuarios navegan de un campo a otro e ingresan datos, no hay eventos para que usted los maneje como desarrolladores.

**Consejo:** OA Framework proporciona la representación de página parcial (PPR ) soporte para algunas acciones, lo que permite una experiencia de usuario más interactiva. Esto se describe completamente en el Capítulo 4.

Cuando el navegador envía una solicitud por último, como se describe más arriba, *todos* los datos de la página se envía en esa sola comunicación - incluyendo todos los valores introducidos por el usuario y la información sobre las acciones que el usuario desea realizar. El desarrollador lee los valores de solicitud para determinar qué sucedió (si el usuario presionó un botón, ¿cuál era?), Hace el trabajo requerido por la acción seleccionada y luego transmite una nueva página HTML al navegador.

**Navegación de página**

Hasta ahora, hemos revisado lo que sucede (en términos generales) cuando el navegador se comunica con el servidor web y viceversa, pero no hemos discutido los mecanismos principales para navegar de una página a otra.

**Nota:** En las siguientes descripciones genéricas, no importa si la solicitud enviada por el navegador es un POST o GET.

**Solicitud estándar**

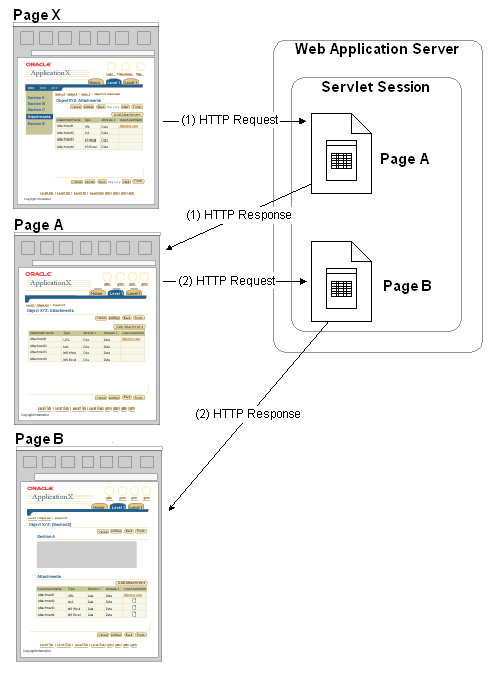
**Guión**

Un usuario selecciona un enlace en la Página X para navegar a la Página A. Mientras está en la Página A, selecciona un enlace para navegar a la Página B.

**Implementación**

El navegador envía una solicitud a la Página A, que hace su trabajo y envía una respuesta al navegador, incluido el HTML para mostrar. Cuando el usuario indica que quiere ver la página B, el navegador envía una nueva solicitud a la página B, que hace su trabajo y envía una respuesta para que la página B pueda mostrarse.

Figura 2: Ilustración de solicitud estándar



**JSP hacia adelante**

**Consejo:** Usted codificar muchos delanteros JSP en la aplicación de OA Framework. Debes entender este concepto.

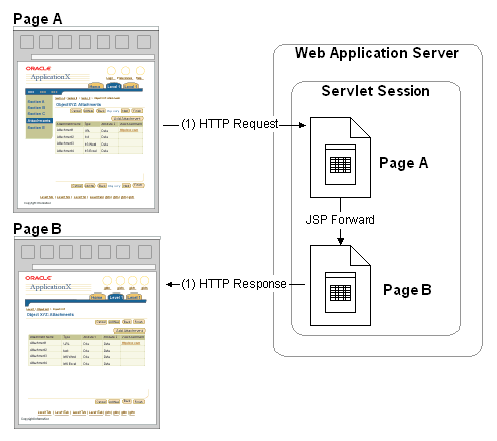
**Guión**

Mientras está en la Página A, un usuario selecciona una acción de una lista definida dinámicamente. El código en JSP A necesita manejar esa acción para determinar qué página mostrar en respuesta.

**Implementación**

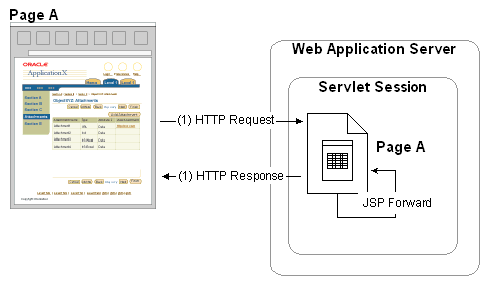
En este caso, mientras maneja una solicitud entrante como resultado de la selección de acción del usuario, JSP A "reenvía" a JSP B, que hace su trabajo y envía una respuesta que incluye el HTML para mostrarse. Dado que la acción de "adelanto" ocurre en el servidor, el navegador no sabe nada al respecto y las dos páginas comparten la misma solicitud.

Figura 3: avance de JSP de una página a la siguiente dentro de una sola solicitud



En otra variante, que es muy común en las páginas de OA Framework por razones, que se describirán más adelante en este capítulo y el siguiente, la Página A podría realizar un Reenvío JSP de nuevo a sí mismo como se muestra a continuación.

Figura 4: avance de JSP de una página a sí mismo dentro de una única solicitud



**Redirigir cliente**

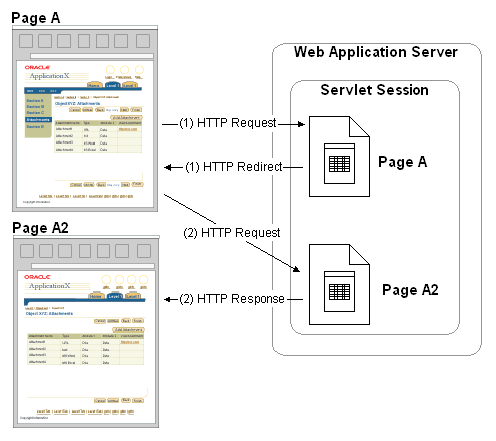
**Guión**

Un usuario selecciona un enlace en la Página X para navegar a la Página A, pero el enlace es antiguo, por lo que el desarrollador desea enviar automáticamente al usuario al nuevo reemplazo, Página A2.

**Implementación**

En este caso, mientras maneja una solicitud entrante, JSP A envía un mensaje especial de "redirección" al navegador indicándole que acceda inmediatamente a JSP A2. El navegador envía una *segunda* solicitud a JSP A2, que hace su trabajo y envía una respuesta incluyendo el código HTML para mostrar.

Figura 4: Un cliente redirige de una ubicación (página) a la siguiente (la misma página en una ubicación diferente)



**¿Qué es una Cookie?**

Para comprender completamente cómo el Marco de OA mantiene el contexto de la aplicación después de que un usuario inicie sesión, debe comprender qué es una "cookie" de navegador.

Una "cookie" es una pepita de información que una aplicación web puede dar a un navegador con la condición de que el navegador la envíe de vuelta al servidor con cada solicitud. En otras palabras, es un mecanismo para aferrarse a una pequeña cantidad de estado entre solicitudes.

Las cookies pueden ser persistentes o estar basadas en sesiones:

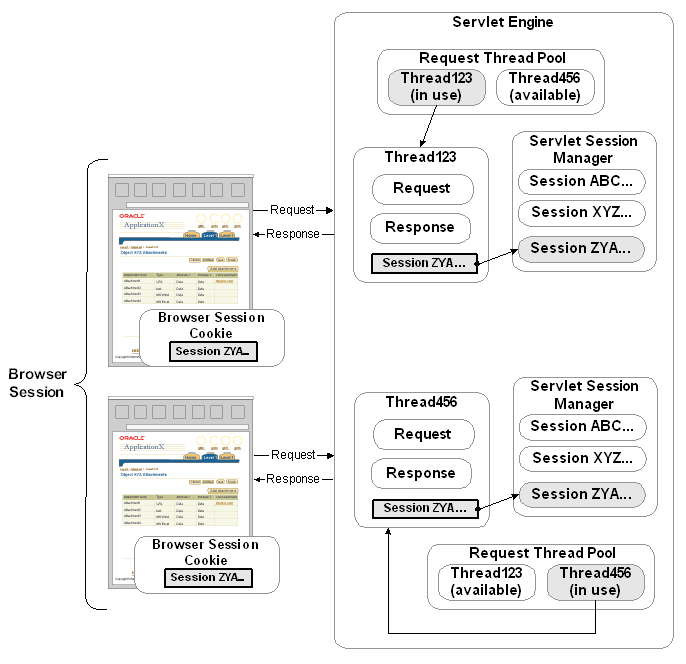
* El navegador guarda una cookie persistente en un archivo en la computadora del usuario, y la información perdura en todas las sesiones del navegador. ¿Alguna vez ha navegado a un sitio web que le saludó por su nombre antes de iniciar sesión? Si es así, esto se logró con una cookie persistente.
* Las cookies basadas en sesión se guardan en la memoria del navegador, y cuando el navegador se cierra, la cookie se destruye.

**Más sobre sesiones de servlets**

De la misma manera que AOL / J agrupa JDBC conexiones porque son un recurso precioso (aprenderá más sobre la agrupación de conexiones más adelante en este capítulo), el motor del servlet agrupa subprocesos de procesamiento de solicitudes. Como se ilustra en la Figura 5 a continuación, el motor de servlets asigna un hilo para procesar cada solicitud que recibe. Cuando la solicitud se completa, el motor de servlet devuelve el hilo a su grupo.

**Nota:** El diagrama siguiente se supone que un usuario realiza dos acciones resultantes en dos peticiones HTTP separados mientras se trabaja en la *misma ventana del navegador* (la misma sesión del navegador). No debe interpretarse en el sentido de que dos ventanas del navegador están abiertas.

Figura 5: Ilustración conceptual del procesamiento de solicitud de motor de servlet



Desde una sola sesión de navegador puede ser servido por numerosos subprocesos, (uno diferente para cada solicitud), la sesión de servlet proporciona un recurso para mantener el estado en todas las solicitudes.

* Si una aplicación web desea establecer una sesión de servlet, llama a un método en el objeto de solicitud para que se cree una sesión.
* El motor de servlets crea la sesión (en concreto, un objeto javax.servlet.http.HttpSession), junto con una cookie especial que devuelve al navegador con la respuesta. Esta cookie de sesión contiene la ID de la sesión servlet.
* Cuando se recibe una nueva solicitud con la cookie de ID de sesión, el motor de servlet usa esta ID para localizar el objeto de sesión de servlet de ese navegador en particular.
* El código de la aplicación web puede acceder a los datos almacenados en la sesión del servlet durante las solicitudes anteriores dentro de la misma sesión del navegador.

**Nota:** Puede hacer un seguimiento sesiones de dos maneras. La forma más común, que es lo que hace OA Framework, es usar una cookie de sesión. Alternativamente, puede codificar la cookie en las URL de solicitud. Si desea obtener más información acerca de este o cualquier otro conceptos presentados en este documento, consulte la [lectura sugerida](https://translate.googleusercontent.com/translate_f#reading) sección anterior.

[Copyright © 2000,2010, Oracle y / o sus afiliados.](https://translate.googleusercontent.com/copyright.htm) [Todos los derechos reservados.](https://translate.googleusercontent.com/copyright.htm)