JAVA - Web

1. Tecnología Web
2. Modelo Cliente- Servidor
3. Acceso a Datos
4. Servlets
5. Seguimiento de cliente con Servlets
6. Introducción a servlets

Introducción a la tecnología WEB con J2EE

La Web dinámica se ha desarrollado desde un sistema de información distribuido (HTML) basado en red que ofrecía información estática hasta un conjunto de portales y aplicaciones en Internet que ofrecen un conjunto variado de servicios.

Las soluciones de *primera generación* incluyeron **CGI**, que es un mecanismo para ejecutar programas externos en un servidor web. El problema con los scripts CGI es la escalabilidad; se crea un nuevo proceso para cada petición.

Las soluciones de *segunda generación* incluyeron vendedores de servidores Web que proporcionaban plug-ins y APIs para sus servidores. El problema es que sus soluciones eran específicas a sus productos servidores. Microsoft proporcionó las páginas activas del servidor (ASP) que hicieron más fácil crear el contenido dinámico. Sin embargo, su solución sólo trabajaba con Microsoft IIS o Personal Web Server. Otra tecnología de segunda generación son los **Servlets**. Los Servlets hacen más fácil escribir aplicaciones del lado del servidor usando la tecnología Java. El problema con los CGI o los Servlets, sin embargo, es que tenemos que seguir el ciclo de vida de escribir, compilar y desplegar .

Las páginas **JSP** son una solución de *tercera generación* que se pueden combinar fácilmente con algunas soluciones de la segunda generación, creando el contenido dinámico, y haciendo más fácil y más rápido construir las aplicaciones basadas en Web que trabajan con una variedad de otras tecnologías: servidores Web, navegadores Web, servidores de aplicación y otras herramientas de desarrollo.

La tecnología Java Server Pages™ (JSP) nos permite poner segmentos de código servlet directamente dentro de una página HTML estática. Cuando el navegador carga una página JSP, se ejecuta el código del servlet y el servidor de aplicaciones crea, compila, carga y ejecuta un servlet en segundo plano para ejecutar los segmentos de código servlet y devolver una página HTML o imprimir un informe XML.

Modelo cliente-servidor

Cuando se utiliza un servicio en Internet, como consultar una base de datos, transferir un fichero o participar en un foro de discusión, se establece un proceso en el que entran en juego dos partes. Por un lado, el usuario, quien ejecuta una aplicación en el ordenador local: el denominado *programa cliente*. Este programa cliente se encarga de ponerse en contacto con el ordenador remoto para solicitar el servicio deseado. El ordenador remoto por su parte responderá a lo solicitado mediante un programa que esta ejecutando. Este último se denomina *programa servidor*. Los términos *cliente* y *servidor* se utilizan tanto para referirse a los programas que cumplen estas funciones, como a los ordenadores donde son ejecutados esos programas.

El programa o los programas cliente que el usuario utiliza para acceder a los servicios de Internet realizan dos funciones distintas. Por una parte, se encargan de gestionar la comunicación con el ordenador servidor, de solicitar un servicio concreto y de recibir los datos enviados por éste; y por otra, es la herramienta que presenta al usuario los datos en pantalla y que le ofrece los comandos necesarios para utilizar las prestaciones que ofrece el servidor

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden.

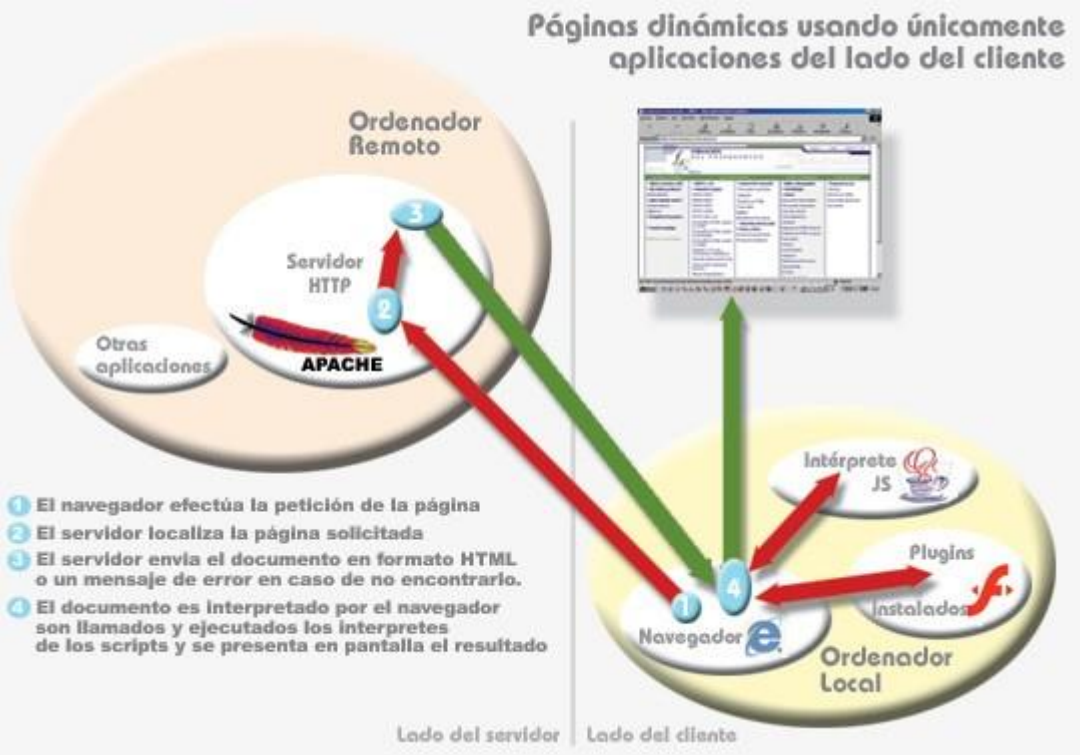


Fig. Lenguajes del lado cliente y del lado servidor

Cuando pinchamos sobre un enlace hipertexto, en realidad lo que pasa es que establecemos una petición de un archivo HTML residente en el servidor (un ordenador que se encuentra continuamente conectado a la red) el cual es enviado e interpretado por nuestro navegador (el cliente).

Sin embargo, si la página que pedimos no es un archivo HTML, el navegador es incapaz de interpretarla y lo único que es capaz de hacer es salvarla en forma de archivo. Es por ello que, si queremos emplear lenguajes accesorios para realizar un sitio web, es absolutamente necesario que sea el propio servidor quien los ejecute e interprete para luego enviarlos al cliente (navegador) en forma de archivo HTML totalmente legible por él.

De modo que, cuando pinchamos sobre un enlace a una pagina que contiene un script en un lenguaje comprensible únicamente por el servidor, lo que ocurre en realidad es que dicho script es ejecutado por el servidor y el resultado de esa ejecución da lugar a la generación de un archivo HTML que es enviado al cliente.

Así pues, podemos hablar de lenguajes de lado servidor que son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Por otro lado, los lenguajes de lado cliente (entre los cuales no sólo se encuentra el HTML sino también JavaScript) son aquellos que pueden ser directamente "ejecutados" por el navegador y no necesitan un pre-procesamiento.

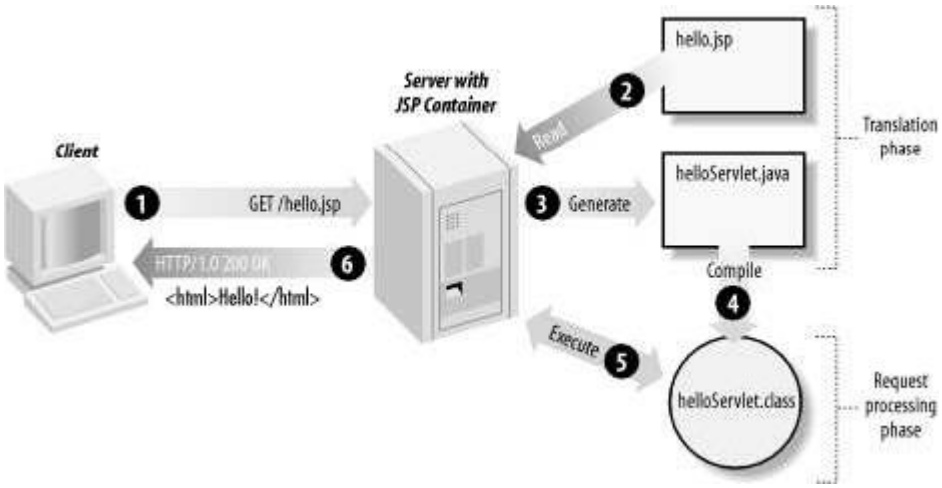


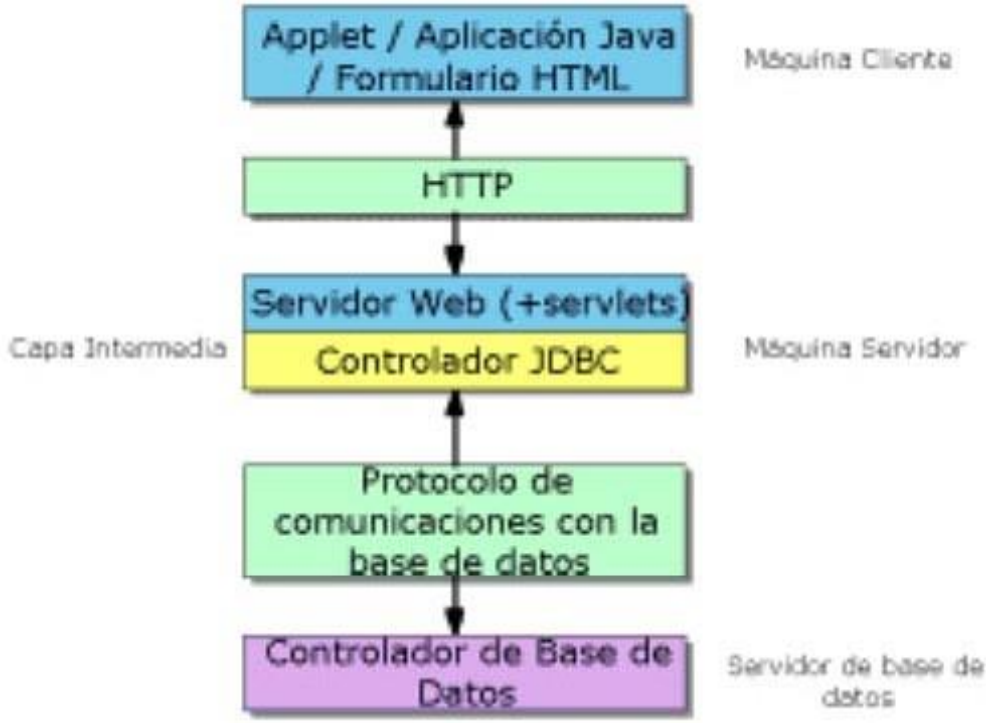
Fig. Ejecución en el modelo web cliente-servidor

En resumen, los programadores de aplicaciones JAVA mediante la especificación J2EE escriben componentes de aplicación J2EE. Un componente J2EE es una unidad de software funcional auto-contenida que se ensambla dentro de una aplicación J2EE y que se comunica con otros componentes de aplicación. La especificación J2EE define los siguientes componentes de aplicación:

- Componentes de Aplicación Cliente: Navegador y páginas *jsp* renderizadas
- Componentes JavaBeans Enterprise: *clases de negocio (EJBs)*
- Componentes Servlets y JavaServer Pages (también llamados componentes Web): *clases controlodaras*.
- Applets: *pequeñas aplicaciones que se ejecutan en el cliente*

Acceso a Datos

Una de las tareas más importantes y más frecuentemente realizadas por los ***servlets*** es la conexión a ***bases de datos*** mediante ***JDBC***. Esto es debido a que los ***servlets*** son un componente ideal para hacer las funciones de capa media en un sistema con una arquitectura de tres capas como la mostrada en la Figura:



Este modelo presenta la ventaja de que el nivel intermedio mantiene en todo momento el control del tipo de operaciones que se realizan contra la base de datos, y además, está la ventaja adicional de que los *drivers* ***JDBC*** no tienen que residir en la máquina cliente, lo cual libera al usuario de la instalación de cualquier tipo de *driver*. En cualquier caso, tanto el ***Servidor http*** como el ***Servidor de Base de Datos*** pueden estar en la misma máquina, aunque en sistemas empresariales de cierta importancia esto no suele ocurrir con frecuencia.

JDBC (***Java DataBase Connectivity***) es una parte del API de ***Java*** que proporciona clases para conectarse con bases de datos. Dichas clases forman parte del package ***java.sql***, disponible en el ***jdk 1.1.7*** y en ***jdk 1.2***. El nombre ***JDBC*** es fonéticamente similar a ***ODBC*** (***Open DataBase Connectivity***), que es el estándar más extendido para conectar PCs con bases de datos.

Formas de seguir la trayectoria de los usuarios (clientes)

Los ***servlets*** permiten seguir la trayectoria de un cliente, es decir, obtener y mantener una determinada información acerca del cliente. De esta forma se puede ***tener identificado a un cliente*** (usuario que está utilizando un browser) durante un determinado tiempo. Esto es muy importante si se quiere disponer de aplicaciones que impliquen la ejecución de varios ***servlets*** o la ejecución repetida de un mismo ***servlet***. Un claro ejemplo de aplicación de esta técnica es el de los ***comercios vía Internet*** que permiten llevar un ***carrito de la compra*** en el que se van guardando aquellos productos solicitados por el cliente. El cliente puede ir navegando por las distintas secciones del comercio virtual, es decir realizando distintas conexiones ***HTTP*** y ejecutando diversos ***servlets***, y a pesar de ello no se pierde la información contenida en el carrito de la compra y se sabe en todo momento que es un mismo cliente quien está haciendo esas conexiones diferentes.

El mantener información sobre un cliente a lo largo de un proceso que implica múltiples conexiones se puede realizar de tres formas distintas:

1.1 Mediante ***cookies***
1.2 Mediante ***seguimiento de sesiones*** (***Session Tracking***)
1.3 Mediante la ***reescritura*** de URLs y paso de parámetros en formulario (***Request***)

Introducción a JSP

La tecnología JSP (*Java Server Pages*) es una especificación abierta desarrollada por Sun Microsystems como un alternativa a Active Server Pages (ASP) de Microsoft, y son un componente dominante de la especificación de Java 2 Enterprise Edition (J2EE). Muchos de los servidores de aplicaciones comercialmente disponibles (como BEA WebLogic, IBM WebSphere, Live JRun, Orion, etcétera) ya utilizan tecnología JSP.

Esta tecnología permite desarrollar páginas web con contenido dinámico y supone una evolución frente a la tecnología CGI, y los Servlets. Un fichero JSP puede contener etiquetas HTML normales, y elementos especiales para generar el contenido dinámico.

Al mismo tiempo permite una separación en n capas de la arquitectura de la aplicación web y se integra perfectamente con todas las API's empresariales de Java: JDBC, RMI (y CORBA), JNDI, EJB, JMS, JTA, ...

Estructura de una página JSP

Una página JSP es básicamente una página Web con HTML tradicional y código Java. La extensión de fichero de una página JSP es ".jsp" en vez de ".html" o".htm", y eso le dice al servidor que esta página requiere un manejo especial que se conseguirá con una extensión del servidor o un plug-in.

Un ejemplo sencillo:

```
<%@ page language="java" contentType="text/html" %>

<html>
   <head>
         <title>Hola, mundo!!</title>
   </head>
   <body>
         <h1>Hola, mundo!</h1>
         Hoy es <%= new java.util.Date() %>.
   </body>
</html>
```

¿Qué es Spring?

Como definición podemos decir que Spring es un framework de código abierto para la creación de aplicaciones empresariales Java, con soporte para Groovy y Kotlin. Tiene una estructura modular y una gran flexibilidad para implementar diferentes tipos de arquitectura según las necesidades de la aplicación.

¿Qué es un framework?

Spring se considera un framework, pero este es un concepto que en ocasiones se confunde, así que vamos a ver qué es un framework.

Una librería es un conjunto de clases, de funciones y de utilidades que nos permiten realizar algunos procesos. Un buen ejemplo de librería sería una librería matemática, a la cual le podemos dar muchos datos y nos puede calcular, por ejemplo, la desviación típica, o le podríamos plantear una integral y la podría resolver.

A diferencia de una librería, un framework es:

- Un conjunto de artefactos software, es decir, que puede incluir una librería, de conceptos y de metodologías.
- Nos proveé de un mecanismo genérico para resolver uno o más problemas de un tipo determinado.
- Es extensible a través de código escrito por los usuarios.
- Ofrece facilidad para el desarrollo y despliegue.

Si tuviéramos que desarrollar una aplicación web, podríamos utilizar un framework que nos facilite la tarea, que nos aporte soluciones a ese desarrollo. Uno de ellos podría ser, por ejemplo Spring MVC, que nos permitiría crear fácilmente una aplicación web, ya que nos aislaría de determinados problemas, como el hecho de crear servlet o registrar las peticiones, así nos podríamos dedicar a lo que realmente importa.

Qué es una aplicación empresarial

Una aplicación empresarial normalmente es:

- Una gran aplicación, es decir, suele ser bastante amplia y orientada a un ámbito comercial o industrial.
- Compleja, es decir, con bastantes funcionalidades, con muchos requisitos y que será utilizada por muchas personas, por lo que debería ser escalable, que

pueda crecer con el tiempo, es decir, que sí hoy tenemos mil usuarios y dentro de un año tenemos un millón, que no haya que rehacer la aplicación de nuevo.
- Distribuida, es decir, que incluso la podamos tener deslocalizada en distintos servidores.
- Crítica, es decir, que no permita o no tolere fallos e incluso, si tuviéramos algún tipo de fallo, que sea capaz de reponerse.
- Orientada a desplegarse dentro de redes corporativas o, el esquema más usual a día de hoy, en internet a través de la nube.
- Centrada en los datos, por lo que todas las funcionalidades se suelen desarrollar en torno a los mismos.
- Intuitiva, de un uso fácil para evitar el rechazo de los usuarios.
- Suele tener, además, unos grandes requisitos de seguridad y de mantenibilidad.

Todas estas serían las características que definirían a una aplicación empresarial.

Spring funciona sobre JVM

Spring funciona con Java, aunque también tiene soporte para otras tecnologías.

Inicialmente se diseñó para trabajar con Java SE y algunas especificaciones o APIs de Java EE, pero a día de hoy trabaja con el JDK 8 y JDK 9. También podemos trabajar con Groovy y tiene soporte para Kotlin.

Spring tiene estructura modular

Spring ya no es solo un framework para la inyección de dependencias, sino que tiene toda una familia de proyectos que abarcan muchos ámbitos: el ámbito de desarrollo de aplicaciones web, aplicaciones web reactivas, seguridad, servicios web, microservicios, Android, etcétera.

Además, dentro de alguno de esos proyectos, podemos encontrar que tiene una estructura modular, es decir, que está orientada a poder tener distintos módulos que agrupan diversas funcionalidades, desde el contenedor de inversión de control, la programación orientada a aspectos, el acceso a datos, etcétera.

Spring es flexible

Spring nos permitir desarrollar todo tipo de aplicaciones diferentes:

Aplicaciones de escritorio, aplicaciones de línea de comando, aplicaciones web clásicas, web reactivas, microservicios…

Aplicaciones que acceden a base de datos vía SQL directamente, a través de algún tipo de ORM, bases de datos NoSQL…

Aplicaciones con esquemas de seguridad clásica donde almacenamos en nuestra base de datos los elementos de seguridad, clave o credenciales, pero también con otros sistemas.

Aplicaciones pequeñas, medianas y grandes, aplicaciones que tienden a ser escalables, aplicaciones que vayamos a gestionar a través de contenedores, aplicaciones que necesitemos desplegar en la nube.

Crear Nuestro Primer proyecto en Spring

Link video Crear proyecto con java Spring: https://youtu.be/qdPEZjhzjlU