Servlets, JavaServer Pages y Enterprise JavaBeans

Julia León Alonso

Índice

Servlets

Sus especificaciones fueron creadas por Sun Microsystems en 1997 y son módulos de código Java, como las applets, pero mientras que estos se ejecutan en el cliente los servlets lo hacen en el servidor.
Pueden usarse desde para recoger o mostrar información en una página web hasta para conectarse con una base de datos, pero su uso más común consiste en la generación de páginas web de forma dinámica a partir de los parámetros que envíe el navegador web.

Características:

Se consideran la siguiente generación de los CGI (Common Gateway Interfaces) ya que tienen las mismas ventajas que estos (se encuentra en el servidor y depende de este para dar servicio), pero añaden algunas características y ventajas más:

- Programación en Java, con lo que obtiene:
  · Características multiplataforma.
  · Puede acceder a APIs como JDBC o RMI.
  · Portabilidad, que evita la necesidad de compilar los programas para el sistema operativo del servidor.
  · Seguridad y control de errores mejorados, además de disponer de varias técnicas de comprobación de errores en tiempo de ejecución.

- Aumento de rapidez de rendimiento gracias a los servlets, que una vez usados quedan almacenados en la memoria del servidor y no es necesario cargarlos una vez por cada petición del cliente, como sí ocurre en los CGI.

- Control de sesiones con lo que puede hacerse un seguimiento de usuarios.

- Manejo sencillo de cookies.

- Necesitan contenedores de servlets para su funcionamiento. Algunos contenedores conocidos son Tomcat, Jboss, WebSphere, WebLogic, etc. El primero es de Apache y el más usado, la mayoría suelen estar basados en él.

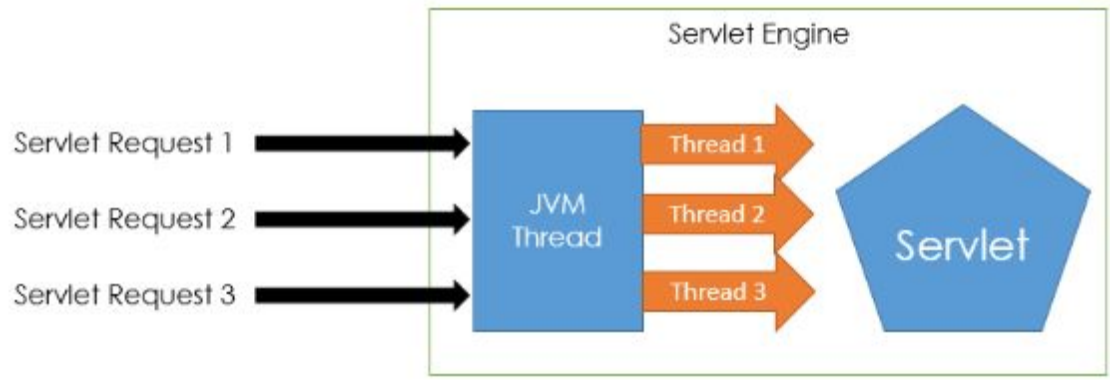
Utilización y funcionamiento:

Al ser clases Java los servlets se crean de la misma forma que estas, pero con la diferencia de que extienden de “HttpServlet” y heredan su método “doGet” que recibe los parámetros de entrada a través de la clase “HttpServletRequest” y empaqueta la respuesta para el cliente a través de la clase “HttpServletResponse”.

```java
import javax.servlet.*;
import javax.servlet.http.*;

public class MiPrimerServlet extends HttpServlet {
   public void doGet (HttpServletRequest req, HttpServletResponse res) throws ServletException, IOExc
     PrintWriter out; out = res.getWriter();
     res.setContentType("text/html");
     out.println("<html>");
     out.println("<head><title>Mi Primer Servlet</title></head>");
     out.println("<body>);
     out.println("Este es mi Primer Servlet");
     out.println("</body></html>");
   }
}

```

Para la utilización de servlets es necesario un contenedor de servlets. A la hora de colocar los servlets en el contenedor se debe utilizar el directorio adecuado para ellos (en el caso de Tomcat *webapps/WEB-INF/classes) y para gestionarlos hay que crear un archivo "web.xml" que describa sus servicios en el directorio pertinente (webapps/WEB-INF en Tomcat).*

Una vez configurado el contenedor y con los servlets y la web terminados, el proceso de funcionamiento sería este:



· El cliente solicita una página al servidor que es un contenedor de servlets.

· El servidor delega la petición al servlet adecuado.

· El servlet procesa la información según su código y genera los datos o la web de respuesta (este proceso se realizaría la primera vez que se ejecuta el servlet, ya que una vez compilado quedará almacenado en la memoria de servidor y el resto de solicitudes se delegarán al archivo compilado).

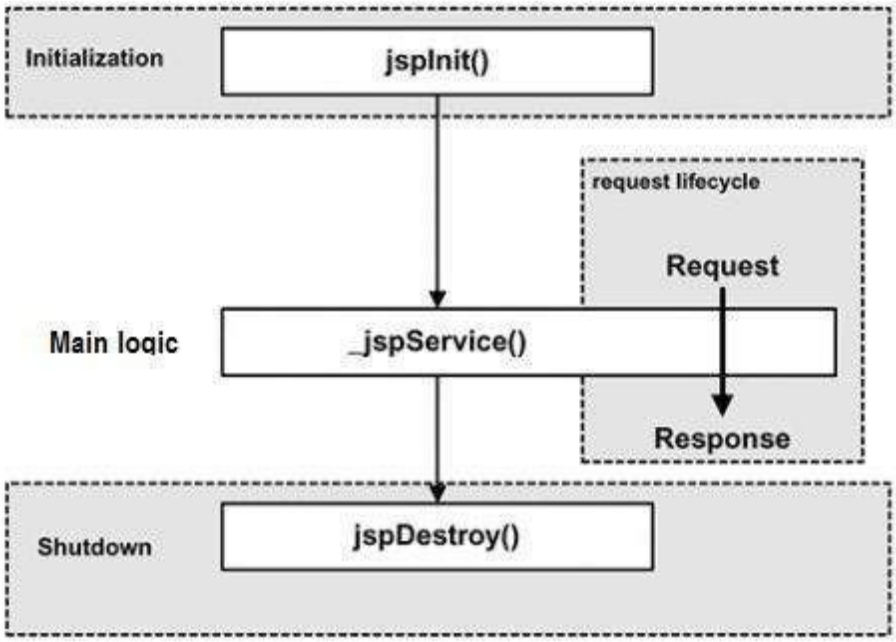
· Se envía esta respuesta al servidor-contenedor.

· Este la entrega al cliente que la solicitó.

JavaServer Pages

Es una tecnología de programación del lado del servidor orientada a crear sitios web dinámicos en Java. De la misma forma que PHP, los archivos se componen de código HTML, XML y JSP embebido y el código se ejecuta en el servidor.

Su motor está basado en los servlets de Java y pueden ser usados conjuntamente o por separado ya que los archivos JSP son convertidos en archivos servlet antes de compilarse. Por esto también los servlet son un poco más rápidos que los JSP, ya que mientras los primeros deben convertirse a servlet y luego compilarse, los segundos lo hacen directamente. Pese a esto el ciclo de vida de ambos es muy similar ya que una vez compilados se almacenan en la memoria del servidor y se ejecutan mediante diferentes métodos:



- Método de inicio (init() para servlets, jspInit() para JSP) se encarga de inicializar las conexiones a la base de datos, abrir archivos necesarios, etc.
- Método de servicio (service() para servlets, jspService() para JSP) interactúa con las solicitudes del cliente y envía las respuestas.
- Método de destrucción (destroy() para servlet, jspDestroy() para JSP) elimina el archivo del contenedor para, por ejemplo, liberar las conexiones de la base de datos, cerrar archivos o actualizar el código del servlet/JSP .

Características:

- Lenguaje Java, por lo que:
  · Es multiplataforma.
  · Tiene acceso a APIs (JDBC, JNDI, EJB, JAXP etc.).
  · Orientado a objetos.
  · Sus componentes son reutilizables gracias a una serie de etiquetas que permiten actuar sobre objetos Java del servidor.
  · Puede crear clases que manejen la lógica de negocio y el acceso a datos.

- Necesita un servidor web que soporte JSP y servlets, como Tomcat.

- Es escalable por su fácil integración con otros servicios de backend.

- Es una mejor opción que los servlets para generar las vistas ya que no utiliza HTML en forma de cadenas de caracteres si no en un entorno más similar al HTML y hace uso de otras construcciones como etiquetas que permiten a los desarrolladores front-end usar bucles y condiciones en una estructura más familiar.

Utilización y funcionamiento:

Para crear un archivo JSP se utiliza código HTML básico y se embebe el código Java entre etiquetas al igual que se haría con PHP por ejemplo.
El resultado es bastante distinto de los servlet, que tienen formato de clases Java.

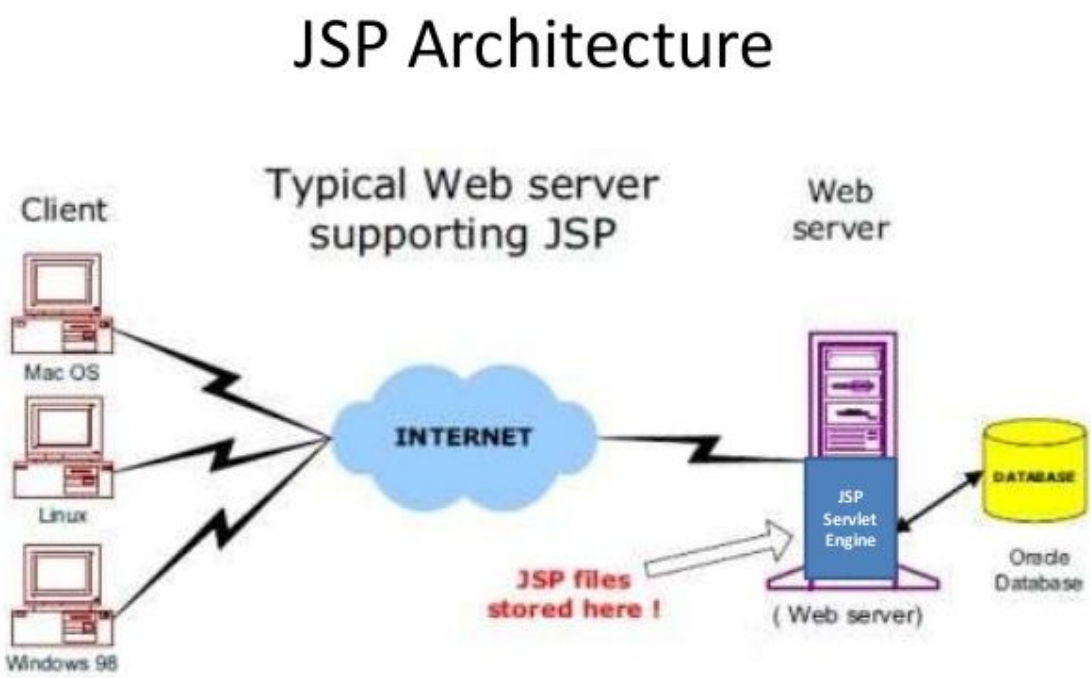
```jsp
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Ejemplos JSP</title>
</head>
<body>
    <H1>Ejemplos de expresiones</H1>
    <%-- Mostrar la fecha y hora actual --%>
    <%= new java.util.Date().toString() %><br>

    <%-- Convertir a mayúsculas un String --%>
    <%= "Pasar a mayúsculas".toUpperCase() %><br>

    <%-- Resultado de una expresión aritmética --%>
    <%= (5+2)/(float)3 %><br>

    <%-- Generar un número aleatorio --%>
    <%= new java.util.Random().nextInt(100) %>
</body>
</html>
```

Código JSP **Equivalente HTML recibido**

```html
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Ejemplos JSP</title>
</head>
<body>
    <H1>Ejemplos de expresiones</H1>
    Wed Jan 16 19:00:52 CET 2013<br>
    PASAR A MAYÚSCULAS<br>
    2.3333333<br>
    65
</body>
</html>
```

Al ser un tipo de servlet los JSP necesitan instalar y configurar un servidor web con soporte para servlets y JSPs como Tomcat. Para ello necesitará una versión del J2SDK, tener ubicada la JVM, un puerto de acceso para el servidor y agregar una variable de entorno, que será la ruta a la ubicación de los archivos jsp-api.jar y servlet-api.jar, en el archivo autoexec.bat.
Posteriormente se deberán alojar los JSP y servlets en el directorio pertinente (el mismo que en el tema anterior) y de nuevo, gestionarlos desde el archivo web.xml.

Finalmente la ejecución de los JSP, que es bastante típica:



· El cliente pide una página JSP.
· El servidor web con soporte para servlets y JSP busca el archivo pertinente.
· Cuando lo encuentra, si no ha sido usado antes, ejecuta el proceso de traducción, luego lo compila en un archivo que queda almacenado en la memoria y genera la respuesta para el cliente.
· La respuesta generada es transmitida al servidor.
· El servidor reenvía esta respuesta al cliente que la pidió.

Enterprise JavaBeans

Es un API de Java enfocada en el gestionado y construcción de aplicaciones empresariales robustas y escalables. Los EJB son componentes del lado del servidor que encapsulan la lógica de negocio y permiten al IDE hacer preguntas a los componentes y conocer las propiedades y eventos que generan.

Existen varios tipos de beans:

- Beans de sesión (Session Beans) que controlan la lógica de negocio que se ejecuta en el servidor y contienen información específica del cliente y se encargan de la interacción con él. Su tiempo de vida se limita a la sesión de tal cliente por lo que no tienen persistencia de datos.
  Dependiendo del tipo de conexión que necesitemos podremos elegir entre usar los bean de sesión con estado (Stateful), sin estado (Stateless) y singleton.
- Beans de entidad (Entity Beans) se encargan de almacenar los datos del negocio de forma persistente y permiten el acceso compartido de múltiples usuarios. Por ello se usan como una capa que simplifica el acceso y manipulación de datos.
- Beans dirigidos por mensajes (Message-Driven Beans) reciben mensajes asíncronamente. Estos beans son invocados por el contenedor cuando recibe un mensaje de un cliente en vez de mediante llamadas a métodos pero pese a esto el cliente no tiene contacto con el bean ni espera su respuesta una vez enviado el mensaje.

Características:

- Lenguaje Java.
- Portabilidad.
- Acceso a entornos operativos como sistemas de bases de datos.
- Componentes reusables gracias a la adaptabilidad de los beans a distintas aplicaciones.
- Separación de las capas de presentación y negocio gracias al encapsulamiento de los procesos o entidades.
- Separación de roles entre desarrollador de beans, ensamblador de aplicaciones, desplegador de aplicaciones, administrador de sistema, etc.
- Posibilidad de integración en sistemas no-Java gracias a la comunicación remota con CORBA, las especificaciones del Connector, JMS y beans manejados mediante mensajes.

Utilización y funcionamiento:

Los EJB deben componerse de dos interfaces Java (home y remota) y una clase que las implemente que deberá ser instanciada en el contenedor. Este contenedor generará las clases que actuarán como proxy en el cliente.

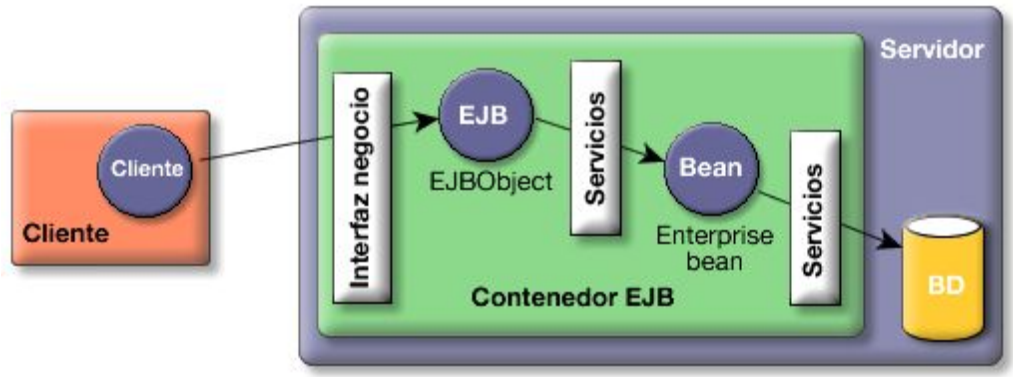
La interfaz home hereda de otra llamada EJBHome y se encarga de fabricar las referencias para los beans y distribuirlas entre los clientes.

La interfaz remota hereda de EJBObject y define los métodos que usará el cliente para llamar y comunicarse con los beans.

Un ejemplo de EJB:

```
package com.saludo;

import java.rmi.RemoteException;

import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

public class HolaMundoBean implements SessionBean{
        private static final long serialVersionUID = 1L;// Quita warning de serialización del objeto

        // Nuestro método "de negocio"
        public String saludo(String nombre){
                System.out.println("Un cliente me ha invocado");
                return "Hola, " + nombre;
        }
        // Métodos del ciclo de vida del Bean (obligatorios)
        public void ejbCreate(){}
        public void ejbActivate() throws EJBException, RemoteException {}
        public void ejbPassivate() throws EJBException, RemoteException {}
        public void ejbRemove() throws EJBException, RemoteException {}
        public void setSessionContext(SessionContext arg0) throws EJBException, RemoteException {}
}
```

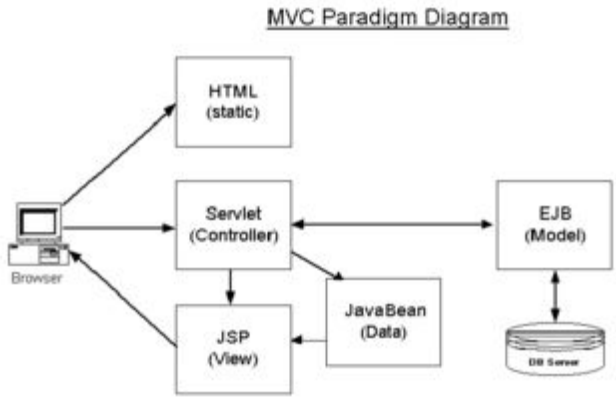
Al igual que los servlets y los JSP necesitan un servidor contenedor que sea compatible con ellos, como Tomcat o JBoss, tener instalados Java y el JDK.
Previamente al despliegue deberán ensamblarse varios ficheros EJB con bean en un fichero EAR al que pueden asignarse valores y constantes utilizadas por los bean.
A la hora de desplegar los beans cada servidor tiene unas características propias, pero la mayoría proporcionan una interfaz gráfica para gestionarlo.



Conclusión

Los servlets, JavaServer Pages y Enterprise JavaBeans trabajan de forma distribuida de igual forma que algunos patrones usados en ingeniería de software. Interactúan como lo harían las capas del un patrón Modelo-Vista-Controlador, que encaja muy bien en aplicaciones web ya que divide la aplicación en capas independientes haciéndolas más escalables, extensibles y robustas.

Al igual que haría la capa “vista” en el MVC, las JavaServer Pages son las que se comunican y reciben la información del cliente, pasándola posteriormente al servlet, la capa “controlador, que las examina y dirige a la capa “modelo” los Enterprise JavaBeans los cuales se encargan de realizar las tareas correspondientes y mandar los resultados o respuestas al controlador, que volverá redirigirlas a la vista.



Bibliografía

http://www.manualweb.net/java-ee/introduccion-a-los-servlets/
https://bannysolano.wordpress.com/2009/08/22/cgi-vs-servlets/
http://www.c4learn.com/java/servlet/servlet-vs-cgi/
https://users.dcc.uchile.cl/~jbarrios/servlets/general.html
https://es.wikipedia.org/wiki/Java_Servlet
http://gssi.det.uvigo.es/users/agil/public_html/LRO/tomcat.html
http://www.edu4java.com/es/servlet/servlet1.html

https://es.wikipedia.org/wiki/JavaServer_Pages
https://www.tutorialspoint.com/jsp/jsp_overview.htm
http://aplicaciones-web-lenguajes-programaci.blogspot.com.es/2011/12/jsp.html
https://blogdeaitor.wordpress.com/2008/10/20/servlet-frente-a-jsp/
http://desarrolloweb.com/articulos/831.php
http://html.rincondelvago.com/jsp.html
http://javabeat.net/difference-servlet-jsp/
http://www.javatpoint.com/MVC-in-jsp
https://www.quora.com/What-are-the-differences-between-JSP-and-Servlets
http://www.desarrolloweb.com/articulos/1613.php

https://en.wikipedia.org/wiki/Enterprise_JavaBeans
https://es.wikipedia.org/wiki/Enterprise_JavaBeans
https://www.tutorialspoint.com/ejb/index.htm
http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/javaBeans/fundamento.htm
http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01-apuntes.htm
http://www.devx.com/tips/Tip/26360