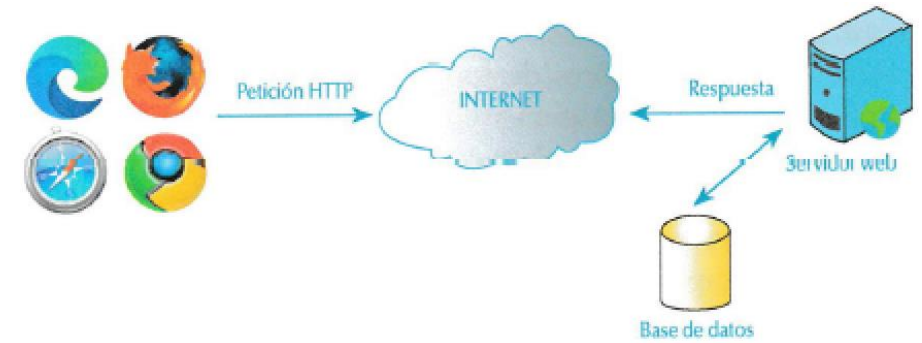


# Implantación de arquitecturas web

Despliegue de aplicaciones web

DAW – Despliegue de aplicaciones web (2º - 0614)

# Arquitecturas web



- Actualmente, para que una aplicación funcione correctamente tiene tres elementos principales que permiten la conexión y el acceso a datos por parte de cualquier petición de un cliente:
  - Servidor web (capa de negocio). Es el servidor o cerebro de la arquitectura, donde está escuchando las peticiones HTTP/HTTPS desde el navegador. También realiza consultas a la base de datos para responder a las peticiones.

Es la que recibe las peticiones del usuario y desde donde se le envían las respuestas; en esta capa se verifica que las reglas establecidas se cumplen
  - Base de datos (capa de acceso a datos). Es un conjunto de datos organizados jerárquicamente al servicio del servidor web.

Es la formada por determinados gestores de datos que se encargan de almacenar, estructurar y recuperar los datos solicitados por la capa de negocio
  - Cliente web (capa de presentación). Es el que realiza las peticiones al servidor web mediante un navegador y desde un sistema operativo concreto que es independiente de la arquitectura.

Es la encargada de la navegabilidad, validación de los datos de entrada, formateo de los datos de salida, presentación de la web, etc.; se trata de la capa que se presenta al usuario

# Evolución de la tecnología web

- Componentes de la web que han ido evolucionando:
  - Ancho de banda. HW de comunicaciones más económico y mayor velocidad.
  - Almacenamiento. HW de almacenamiento más económico y mayor capacidad.
  - Información. La información es más dinámica y existe más interacción con el usuario.
  - Computación. El HW (procesadores, memoria y disco) más rápido.
  - Tecnología. Amplio abanico de posibilidades para implementar tanto en el cliente como en el servidor.
  - Infraestructura. Alta disponibilidad, backup remoto, duplicidad de nodos en caso de fallo HW y SW.
- Evolución del concepto de web
  - Web 1.0: Finalidad divulgativa con páginas estáticas.
  - Web 2.0 (2004): El usuario crea y comparte información (Web social)
    - Aplicaciones ricas en Internet (RIA, rich Internet Applications). Presentaciones casi idénticas a las de escritorio.
    - Arquitectura orientada al servicio (SOA, Service Oriented Architecture). Los componentes se diseñan para usarse como un servicio en red.
    - Web social. En las aplicaciones WEB el usuario es el centro de las mismas.
  - Web 3.0 (Web semántica):
    - Acercar al usuario al lenguaje natural y adaptación de la navegación a sus preferencias.
    - Inteligencia artificial
    - Múltiples dispositivos.
    - Geoespacial
    - Ambiente 3D
  - Web 4.0
    - Dispositivos móviles
    - Lenguaje natural
    - Comunicación máquina-máquina (M2M y IoT)

# Tecnologías usadas en aplicaciones web

- En el lado del servidor:
  - CGI (Common Gateway Interfaz). El servidor web en su S.O. ejecuta programas (del tipo C, Perl y líneas de comando Powershell) cuyo resultado se trasmite al navegador.
  - ASP.NET (Active Server Pages). Framework de desarrollo libre de Windows orientado a objetos.
  - Java. Grupo de tecnologías (Sun Microsystems/Oracle) que se pueden ejecutar en el servidor, como JSF (JavaServer Faces), JSP (JavaServer Pages) y “servlets”.
  - Javascript. Lenguaje ligero, interpretado y orientado a objetos, tanto en el servidor como en el cliente.
  - Ruby. Lenguaje de programación interpretado de propósito general, libre y multiplataforma.
  - PHP, Perl, Python, R y VB Script.
- En el lado del cliente:
  - HTML y CSS.
  - Javascript. Lenguaje ligero, interpretado y orientado a objetos, tanto en el cliente como en el servidor.
  - Java. Los "applets" son pequeños programas interactivos que se encuentran incrustados en una página web y pueden funcionar con cualquier tipo de navegador.
  - VB Script. La respuesta de Microsoft a JavaScript, tanto en cliente como en servidor.
- En ambos:
  - DHTML (HTML dinámico). No es un lenguaje de programación como tal, sino el conjunto de técnicas que permiten crear sitios web.
  - XML. Tecnología relacionada con lenguajes de marca, y que permite compartir datos.

# Escalabilidad de un sistema web

- Escalabilidad de un sistema web:
  - Escalabilidad vertical. Interponiendo elementos conectores que actúen de middlewares es posible distribuir la aplicación de forma vertical (una máquina por cada capa del sistema), e incluso si esto no fuera suficiente, distribuyendo los elementos de una misma capa entre distintas máquinas servidoras.
  - Escalabilidad horizontal. Se trata de clonar el sistema en otra máquina de características similares y balancear la carga de trabajo mediante un dispositivo externo. El balanceador de carga puede ser:
    - Balanceador Software: Por ejemplo, habitualmente encontramos un servidor web apache junto con el módulo mod\_jk, que permite la redirección de las peticiones HTTP que a tal efecto sean configuradas entre las distintas máquinas que forman la granja de servidores. Este tipo de balanceadores examinan el paquete http e identifican la sesión del usuario, guardando registro de cuál de las máquinas de la granja se está encargado de servir a dicha sesión.
    - Balanceador hardware: Se trata de dispositivos que, respondiendo únicamente a algoritmos de reparto de carga (Round Robin, LRU, etc.), redireccionan una petición http del usuario a la máquina que, según dicho algoritmo, convenga que se haga cargo de la petición. Más rápidos que los software pero no mantiene las sesiones.
    - Balanceador hardware http: Se trata de dispositivos hardware pero que examinan el paquete http y mantienen la relación usuario-máquina servidora. Mucho más rápidos que los balanceadores software, pero algo menos que los hardware.
  - Cluster. Un cluster de servidores de aplicaciones permite el despliegue de una aplicación web corriente, de forma que su carga de trabajo vaya a ser distribuida entre la granja de servidores que forman el cluster, de modo transparente al usuario y al administrador. El cluster, mediante el mecanismo de replicación de sesión, garantiza que sea cual sea la máquina que sirva la petición http, tendrá acceso a la sesión del usuario (objeto HttpSession en java).

# Servidores y aplicaciones libres y propietarias

- Una plataforma web consta de cuatro componentes básicos:
  1. El **sistema operativo**, bajo el cual opera el equipo donde se hospedan las páginas web y que representa la base misma del funcionamiento del computador. En ocasiones limita la elección de otros componentes.
  2. El **servidor web** es el software que maneja las peticiones desde equipos remotos a través de la Internet. En el caso de páginas estáticas, el servidor web simplemente provee el archivo solicitado, el cual se muestra en el navegador. En el caso de sitios dinámicos, el servidor web se encarga de pasar las solicitudes a otros programas que puedan gestionarlas adecuadamente.
  3. El **gestor de bases de datos** se encarga de almacenar sistemáticamente un conjunto de registros de datos relacionados para ser usados posteriormente.
  4. Un **lenguaje de programación** interpretado que controla las aplicaciones de software que corren en el sitio web. Diferentes combinaciones de los cuatro componentes señalados, basadas en las distintas opciones de software disponibles en el mercado, dan lugar a numerosas plataformas web, aunque, sin duda, hay dos que sobresalen del resto por su popularidad y difusión: LAMP y WISA
- La plataforma LAMP trabaja enteramente con componentes de software libre y no está sujeta a restricciones propietarias:
  - Linux: Sistema operativo.
  - Apache: Servidor web.
  - MySQL: Gestor de bases de datos.
  - PHP: Lenguaje interpretado PHP, aunque a veces se sustituye por Perl o Python.
- La plataforma WISA está basada en tecnologías desarrolladas por la compañía Microsoft; se trata, por lo tanto, de software propietario:
  - Windows: Sistema operativo.
  - Internet Information Services: servidor web.
  - SQL Server: gestor de bases de datos.
  - ASP o ASP.NET como lenguaje para scripting del lado del servidor.
- Otras plataformas: WAMP (sólo para desarrollo local), WIMP y XAMPP.

# Instalación y configuración básica del servidor apache

- En cuanto a su arquitectura podemos destacar lo siguiente:
  - Estructurado en módulos.
  - Cada módulo contiene un conjunto de funciones relativas a un aspecto concreto del servidor.
  - El archivo binario httpd contiene un conjunto de módulos que han sido compilados.
  - La funcionalidad de estos módulos puede ser activada o desactivada al arrancar el servidor.
  - Los módulos de Apache se pueden clasificar en tres categorías:
    - Módulos base: Se encargan de las funciones básicas.
    - Módulos multiproceso: Encargados de la unión de los puertos de la máquina, aceptando las peticiones y atendíéndolas.
    - Módulos adicionales: se encargan de añadir funcionalidad al servidor.
- El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation, gratuito y de código abierto, y multiplataforma.
- La Licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

# Instalación y configuración básica del servidor apache

- Instalación:

- `#apt-get update`

- `#apt-get install systemd apache2`

- Instalación de php y mysql

- `#apt-get install php libapache2-mod-php php-mysql`

- `#apt-get install mysql-server`

- Instalación de netstat

- `#apt-get install net-tools`

- Monitorización/arranque/parada:

- `#service apache2 [ start | stop | status | restart | force-reload ]`

- `#ps -ef | grep apache2`

- `#netstat -ltun | grep :80`

- Verificación (puerto por defecto 80): <http://localhost> o <http://127.0.0.1>

- Configuración: (directorio) </etc/apache2> (fichero) [apache2.conf](#)



# Instalación y configuración básica del servidor apache

- Estructura de configuración

```
/etc/apache2/  
├─ apache2.conf  
├─ conf-enabled  
├─ magic  
├─ mods-enabled  
├─ sites-available  
├─ conf-available  
├─ envvars  
├─ mods-available  
├─ ports.conf  
└─ sites-enabled
```

```
/etc/apache2/sites-available/  
├─ 000-default.conf  
└─ default-ssl.conf  
  
/var/www/html  
└─ index.html  
  
/etc/apache2/mods-available/mime.conf  
  
/etc/apache2/apache2.conf
```

Fichero/Directorio	Descripción
apache2.conf	Fichero de configuración principal del servidor apache2. Contiene las variables globales. Cualquier cambio implica reinicio.
conf-avaliable	Este directorio contiene configuraciones adicionales que están asociadas a un módulo en particular. Las configuraciones no están activas.
conf-enabled	Este directorio contiene enlaces al directorio anterior para poder activar las configuraciones que contiene.
magic	Fichero de configuración de tipo MIME, que permite configurar el tipo de medio del contenido que se va a visualizar en el servidor web
mods-available mods-enabled	Directorios similares a sus correspondientes “conf” pero para módulos que se pueden acoplar al servidor web
port.conf	Este fichero siempre está incluido en “apache2.conf” y permite configurar los puertos y las IP por las que escucha el servidor. El puerto por defecto es el 80. Apache2 puede escuchar de varios puertos, lo que supone varias entradas “Listen” en este fichero (y varios VirtualHosts en sites-enabled).
sites-available	Este directorio contiene los ficheros de los hosts virtuales definidos en el servidor
sites-enabled	Este directorio contiene las definiciones de los hosts virtuales que se están usando. Normalmente son enlaces al directorio anterior. Por defecto el directorio /var/www/html es donde se almacena la página principal.

# Instalación y configuración básica del servidor apache

- Servidores virtuales (VirtualHosts). Servidor web en varios puertos simultáneamente: “sites-enabled” y “ports.conf”
  - Directivas de configuración de “apache2.conf”:
  - Otros
    - DocumentRoot
    - Listen
- *ServerRoot*: es el directorio raíz donde está instalado el servidor Apache. Por defecto es /etc/apache2.
  - *ServerName*: el nombre DNS que indica el nombre del servidor web, que en nuestro caso es webadmin. Sería necesario registrarlo en nuestro DNS o, en su defecto, en el fichero /etc/hosts. Se puede indicar también en el fichero de host virtual.
  - *ServerAdmin*: es el correo electrónico del administrador del servidor web, por si es necesario contactar con él.
  - *Timeout*: permite especificar el tiempo que espera Apache para cerrar las conexiones con un cliente determinado.
  - *KeepAlive*: si está activado permitirá las conexiones persistentes, es decir, si el servidor web deja abierta la conexión con un cliente para futuras conexiones. Tiene dos valores, on y off.
  - *MaxKeepAliveRequests*: delimita el número de peticiones permitidas por una conexión persistente. Por defecto son 100; si se quiere poner infinito se puede poner 0, pero no es recomendable por rendimiento y seguridad.
  - *KeepAliveTimeout*: el número de segundos que el servidor esperará para la próxima petición del mismo cliente.

# Aplicaciones web y servidores de aplicaciones

- Se define una aplicación web como una aplicación informática que se ejecuta en un entorno web, de forma que se trata de una aplicación cliente-servidor junto con un protocolo de comunicación previamente establecido:
  - Cliente: navegador.
  - Servidor: servidor web.
  - Comunicación: protocolo HTTP.
- Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general, a través de Internet y utilizando el protocolo HTTP
- Las principales ventajas de la tecnología de los servidores de aplicaciones son:
  - Centralización
  - Disminución de la complejidad en el desarrollo de las aplicaciones
  - Integridad de datos y código

# Aplicaciones web y servidores de aplicaciones

- El servidor de aplicaciones **Tomcat**

- Tomcat es el servidor web (incluye el servidor Apache) y de aplicaciones del proyecto Yakarta, con lo cual, gestiona las solicitudes y respuestas HTTP y, además, es servidor de aplicaciones o contenedor de Servlets y JSP (incluye el compilador Jasper, que compila JSP convirtiéndolas en servlets).
- Tomcat es un contenedor de servlets con un entorno JSP. Un contenedor de servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. Podemos dividir los contenedores de servlets en:
  - **Contenedores de servlets stand-alone** (independientes). Estos son una parte integral del servidor web. Este es el caso en el que se usa un servidor web basado en Java, por ejemplo, el contenedor de servlets es parte de JavaWebServer (actualmente sustituido por iPlanet). Por defecto Tomcat trabaja en este modo, sin embargo, la mayoría de los servidores no están basados en Java.
  - **Contenedores de servlets dentro-de-proceso**. El contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java. El plugin del servidor web abre una JVM (Máquina Virtual Java) dentro del espacio de direcciones del servidor web y permite que el contenedor Java se ejecute en él. En el caso de que una petición debiera ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando JNI). Un contenedor de este tipo es adecuado para servidores multi-thread de un sólo proceso y proporciona un buen rendimiento pero está limitado en escalabilidad.
  - **Contenedores de servlets fuera-de-proceso**. El contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java que se ejecuta en una JVM fuera del servidor web. El plugin del servidor web y el JVM del contenedor Java se comunican usando algún mecanismo IPC (normalmente sockets TCP/IP). Si una cierta petición tuviese que ejecutar un servlets, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando IPCs). El tiempo de respuesta en este tipo de contenedores no es tan bueno como el anterior, pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad, etc.).

# Aplicaciones web y servidores de aplicaciones

- Instalación y configuración básica de **Tomcat** (como usuario root)
  - Es necesario tener instalado JDK (Kit de desarrollo de Java), ya que el objetivo es que las peticiones a Apache se redirijan a Tomcat empleando un conector proporcionado por Java en este caso.  
`#apt update` `#apt install default-jdk` `#apt install curl`
  - Creación en el S.O. el grupo “tomcat” y usuario “tomcat”.  
`#groupadd tomcat` `#useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat`  
`#mkdir /opt/tomcat` `#chgrp -R tomcat /opt/tomcat`
  - Descargamos tomcat (p.e. en /tmp) y lo descrompimimos (en /opt/tomcat)  
`#cd /tmp` `#curl -O http://www.emambrilla.com/apache-tomcat-9.0.82.tar.gz`  
`#tar xzvf apache-tomcat-*tar.gz -C /opt/tomcat --strip-components=1`  
`#chgrp -R tomcat /opt/tomcat` `#chmod -R g+r /opt/tomcat/conf` `#chmod g+x /opt/tomcat/conf`  
`#chown -R tomcat /opt/tomcat/webapps /opt/tomcat/work /opt/tomcat/temp /opt/tomcat/logs`
  - Descargar el fichero tomcat.service en el directorio “/etc/systemd/system”  
`#cd /etc/systemd/system` `#curl -O http://www.emambrilla.com/tomcat.service`
- Monitorización/arranque/parada:
  - `#service tomcat [ start | stop | status | restart | force-reload ]`
  - `#ps -ef | grep tomcat`
  - `#netstat -ltun | grep :8080`
- Verificación (puerto por defecto 8080): `http://localhost:8080` o `http://127.0.0.1:8080`

# Aplicaciones web y servidores de aplicaciones

- Inicio de **Tomcat**


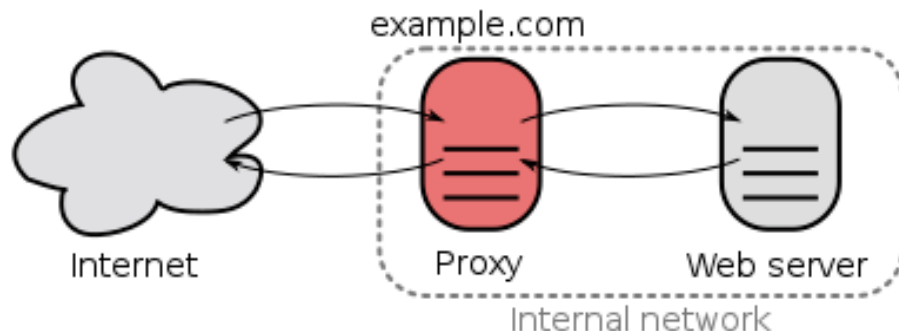
- Debemos añadir un usuario que pueda acceder a manager-gui y admin-gui (aplicaciones web que vienen con Tomcat). Editar el fichero `"/opt/tomcat/conf/tomcat-users.xml"` y añadir en la en la sección `<tomcat-users ...>` y fuera de los comentarios

```
<user username="admin" password="password" roles="manager-gui,admin-gui"/>
```

- Accesos (usuario: **admin** y password: **password**)

- Server status: <http://localhost:8080/manager/status>
- Manager app: <http://localhost:8080/manager/>
- Host manager: <http://localhost:8080/host-manager/>

- Proxy inverso



Recommended Reading:

- [Security Considerations How-To](#)
- [Manager Application How-To](#)
- [Clustering/Session Replication How-To](#)

Server Status

Manager App

Host Manager

# Estructura y despliegue de una aplicación web

- Una aplicación web está compuesta de una serie de servlets, páginas JSP, ficheros HTML, ficheros de imágenes, ficheros de sonidos, texto, clases, etc.; de forma que todos estos recursos se pueden empaquetar y ejecutar en varios contenedores distintos.
- Un servlet es una aplicación java encargada de realizar un servicio específico dentro de un servidor web.
- Ejemplo de estructura de directorios para los ficheros de una aplicación web (según especificación Servlet 2.2)

```
/index.jsp
/WebContent/jsp/welcome.jsp
/WebContent/css/estilo.css
/WebContent/js/utils.js
/WebContent/img/welcome.jpg
/WEB-INF/web.xml
/WEB-INF/struts-config.xml
/WEB-INF/lib/struts.jar
/WEB-INF/src/com/empresa/proyecto/action/welcomeAction.java
/WEB-INF/classes/com/empresa/proyecto/action/welcomeAction.class
```

- En la etapa de producción toda la estructura de la aplicación se empaqueta en un archivo WAR (.war)

# Estructura y despliegue de una aplicación web

- Archivos WAR (Web Application Archive)
  - Permiten empaquetar en una sola unidad aplicaciones web de Java completas, simplificando el despliegue. Su contenido es:
    - Servlets y JSP.
    - Contenido estático: HTML, imágenes, etc.
    - Otros recursos web.
  - Su estructura es la siguiente:
    - /: En la carpeta raíz del proyecto se almacenan elementos empleados en los sitios web: documentos html, hojas de estilo y los elementos JSP (\*.html, \*.jsp, \*.css).
    - /WEB-INF/: Aquí se encuentran los elementos de configuración del archivo .war como pueden ser: la página de inicio, la ubicación de los servlets, parámetros adicionales para otros componentes. El más importante de éstos es el archivo web.xml.
    - /WEB-INF/classes/: Contiene las clases Java empleadas en el archivo .war y, normalmente, en esta carpeta se encuentran los servlets.
    - /WEB-INF/lib/: Contiene los archivos JAR utilizados por la aplicación y que normalmente son las clases empleadas para conectarse con la base de datos o las empleadas por librerías de JSP.
- Para generar archivos .war se pueden emplear diversas herramientas: NetBeans, Eclipse, Jbuilder, Jdeveloper y Ant.



# Estructura y despliegue de una aplicación web

- Despliegue de aplicaciones con tomcat
  - Una aplicación web puede ser desplegada empleando uno de los siguientes métodos:
    - Por medio de archivos .war (Web Archive).
    - Editando los archivos web.xml y server.xml.
  - Descripción de despliegue editando los archivos web.xml y server.xml.
    - Los directorios que forman una aplicación compilada suelen ser : www, bin, src, tomcat y gwt-cache.
    - La carpeta www contiene, a su vez, una carpeta con el nombre y ruta del proyecto que contiene los ficheros que forman la interfaz (.html, .js, .css, etc.). La carpeta bin contiene las clases de java de la aplicación.
    - Pasos a seguir:
      1. Copiar la carpeta contenida en www (con el nombre del proyecto) en el directorio webapps de Tomcat.
      2. Renombrar la nueva carpeta así creada en Tomcat con un nombre más sencillo. Esa será la carpeta de la aplicación en Tomcat.
      3. Crear, dentro de dicha carpeta, otra nueva, y darle el nombre WEB-INF (respetando las mayúsculas).
      4. Crear, dentro de WEB-INF, otros dos subdirectorios, llamados lib y classes.
      5. Copiar en lib todas las librerías (.jar) que necesite la aplicación para su funcionamiento.
      6. Copiar el contenido de la carpeta bin de la aplicación en el subdirectorio WEB-INF/classes de Tomcat.
      7. Crear en WEB-INF un fichero de texto llamado web.xml, con las rutas de los servlets utilizados en la aplicación.
      8. A la aplicación ya puede accederse en el servidor, poniendo en el navegador la ruta del fichero .html de entrada, que estará ubicado en la carpeta de la aplicación en Tomcat.

# Estructura y despliegue de una aplicación web

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
    <!-- Tus definiciones van aquí -->
</web-app>
```

- Descriptor del despliegue

- Un Descriptor de Despliegue es un documento XML que describe las características de despliegue de una aplicación, un módulo o un componente.
- Cualquier aplicación web tiene que aportar un descriptor de despliegue situado en WEB-INF/web.xml (ejemplo en <TOMCAT\_HOME>/conf/web.xml)
- Situadas entre las etiquetas <web-app> y </web-app> estarían los descriptores de despliegue de servlets, los cuales deben contener las siguientes etiquetas en el siguiente orden:

```
<servlet>
    <servlet-name>nombre</servlet-name>
    <servlet-class>package.nombre.MiClass</servlet-class>
</servlet>
```

- El servlet se invoca desde el navegador como “http://{address}:port/{servletName}” (por ejemplo [http://localhost:8080/Servlet de prueba](http://localhost:8080/Servlet_de_prueba))