**Ejercicio 1: Servlets: creación y despliegue de servlets, de forma manual, en apache-tomcat.**

La primera aplicación web va a contener unos servlets sencillos que tendremos que crear, compilar y desplegar a mano. Todo este proceso se simplificará al utilizar un entorno de desarrollo como **Eclipse** o **NetBeans.**

**Pasos:**

* **Se creará la estructura de directorios de la aplicación, la aplicación va a contener 3 servlets y un descriptor de despliegue (web.xml) que le indicará al contenedor cómo localizar cada servlet y cómo se mapeará en la URL de acceso.**
* **Se creará cada servlet (archivo.java), se compilará y se moverá el archivo compilado: archivo.class al directorio: WEB-INF\classes\**
* **Se creará el descriptor de despliegue de la aplicación: archivo web.xml, en: el directorio: WEB-INF\**
* **Por último se desplegará la aplicación moviendo la carpeta de la aplicación al directorio de despliegue de tomcat: C:\Program Files\ApacheSoftware Foundation\Tomcat v.0\webapps\**
* **Se empaquetará la aplicación en un archivo .war (con la utilidad jar), para que pueda ser desplegado en cualquier otro contenedor que cumpla las especificaciones Java para servlets.**

1. Primero habrá que definir las siguientes variables del entorno: PATH, CLASSPATH, CATALINA\_HOME y JAVA\_HOME.

Desde una consola del sistema, se puede utilizar el comando:

**SET variable=valor**

(Se está suponiendo que tenemos instaladas las versiones: **jdk1.7.0\_06**  y **Tomcat 7.0** (sustituir por las versiones que se hayan instalado realmente)

**SET JAVA\_HOME=C:\Program Files\Java\jdk1.7.0\_06 (**directorio donde tenemos instalado **JDK)**

**SET CATALINA\_HOME=C:\ProgramFiles\ApacheSoftware Foundation\Tomcat 7.0** (directorio donde tenemos instalado **tomcat**).

Se añadirá, a la variable PATH, la ruta donde se encuentran los ejecutables de la máquina virtual **JDK**  y de **tomcat** (directorios **\bin**  de ambos)**: las rutas se separan con dos puntos “:”**

Podemos utilizar las variables definidas previamente:

**SET PATH=%PATH%;%JAVA\_HOME%\bin;%CATALINA\_HOME%\bin**

**SET CLASSPATH=.;%CATALINA\_HOME%\lib\servlet-api.jar** (el primer “.” representa el directorio activo; y la ruta hasta la api de los servlets (las librerías que los gestionan).

1. **Aplicación: ejemplos.** Prepara la estructura de directorios de la app: crea un directorio con el nombre de la aplicación**: ejemplos.** Crea, dentro, la estructura de directorios de la aplicación con los directorios básicos requeridos:

**ejemplos\WEB-INF**

**ejemplos\ WEB-INF\classes**

**ejemplos\src** (esta carpeta la creamos para dejar los archivos fuente  **.java**, se puede desplegar, o no, en el servidor, aunque nosotros lo haremos por comodidad).

1. Cambia el directorio activo a: **ejemplos\src**, crea con **NotePad++**  el primer servlet: **Saludo.java** , que generará la cadena: “**Primer servlet. Genera texto en formato HTML**”.

Nota: En la implementación de los métodos **doGet();**  y **doPost();** se suelen utilizar los objetos implícitos: **request, response** y **out** que representan, respectivamente, la petición que realiza el cliente, la respuesta que devuelve el servlet al cliente y el objeto salida donde se escribe dicha respuesta. En estos primeros ejemplos utilizaremos para mayor claridad, los nombres lógicos: **petición, respuesta**  y **salida**.

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.\*;

public class **Saludo** extends **HttpServlet**  {

public void **doGet** (**HttpServletReques**t **peticion**, **HttpServletResponse respuesta**) throws ServletException,IOException {

//Establece el tipo MIME del mensaje de respuesta en este caso HTML

**respuesta.setContentType("text/html");**

//Crea un flujo de salida para escribir la respuesta a la petición del cliente. //Pertenece a java.io

**PrintWriter salida** = **respuesta.getWriter();**

//Escribe el mensaje de respuesta en una página HTML

salida.println("<html>");

salida.println("<body>");

salida.println("<h1>Primer servlet. Genera salida en format HTML </h1>");

salida.println("</body>");

salida.println("</html>");

}

}

1. Desde la línea de comandos, compila el archivo: **javac Saludo.java**. Comprueba que se ha generado correctamente el archivo: **Saludo.class.**
2. El archivo **Saludo.class** debe quedar en el directorio: **ejemplos\WEB-INF\classes** Muévelo.
3. Crea el descriptor de despliegue de la aplicación: **web.xml** en el directorio**: ejemplos\ WEB-INF\** (utiliza NotePad++ y guarda el archivo en formato **.xml**)

<web-app version="3.0"

xmlns="http://java.sun.com/xml/ns/j2ee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation=”[http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_3\_0.xsd](http://java.sun.com/xml/ns/j2ee%20http://java.sun.com/xml/ns/j2ee/web-app_3_0.xsd)”>

<servlet>

<servlet-name>Saludo</servlet-name>

<servlet-class>**Saludo**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Saludo</servlet-name>

<url-pattern>**/hola**</url-pattern>

</servlet-mapping>

</web-app>

1. **Despliega**  la aplicación en el contenedor **tomcat**: Mueve la carpeta de la aplicación: **ejemplos,** al directorio de despliegue de tomcat: **CATALINA\_HOME\webapps** ,
2. Prueba el servlet escribiendo la URL con el Alias que hemos especificado en el descriptor de despliegue: [**http://server:8080/ejemplos/hola**](http://server:8080/ejemplos/hola) **Captura la pantalla.** 
3. Cambia el mapeo o Alias de la URL, modificando en el descriptor de despliegue: **web.xml** , la línea: **<url-pattern>**  indicando: **/prueba** Para y arranca tomcat desde el icono de la barra de herramientas.
4. Accede de nuevo a la aplicación, utilizando: **/prueba**  en la URL de acceso.
5. **Segundo servlet: Fecha.** Crea el servlet: **Fecha.java**, en: **\ejemplos\src\Fecha.java** . Es como el anterior añadiendo:

**import java.util.Date;**

salida.println("<h1>Segundo servlet </h1>");

salida.println(**"hoy es " + new Date()**);

1. Repite los pasos realizados con el primer servlet:Compila el archivo:**Fecha.java.** Mueve el archivo **Fecha.class** al directorio: **ejemplos\ WEB-INF\classes**

Modifica el descriptor: **web.xml** de la aplicación, añadiendo las etiquetas correspondientes al nuevo servlet: **Fecha**, con el patrón **<url-pattern>/dia</url-pattern>**

<servlet>

<servlet-name>Fecha</servlet-name>

<servlet-class>**Fecha**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Fecha</servlet-name>

<url-pattern>**/dia**</url-pattern>

</servlet-mapping>

1. **Despliega de nuevo** la aplicación en el contenedor **tomcat**: Mueve otra vez, la carpeta de la aplicación: **ejemplos,** al directorio de tomcat:**webapps\** . Para y arranca tomcat desde el icono de la barra de herramientas.
2. Prueba el 2º servlet: [**http://server:8080/ejemplos/dia**](http://server:8080/ejemplos/dia) **Captura la pantalla.**



1. **Tercer servlet: ejem3** es habitual que las aplicaciones java se organicen en paquetes o directorios para gestionarlos más fácilmente. Basta indicar en el código fuente la sentencia: **package nombre.del.paquete.** Tendremos que crear a mano el directorio que representa el paquete, dentro del directorio : **classes** Crea el directorio: **paquete**  en: **ejemplos\ WEB-INF\classes\paquete**
2. Crea el servlet: **Datos.java** en: **ejemplos\src\Datos.java**. Este servlet en la salida HTML al cliente enviará una serie de datos que se obtendrán en la petición realizada desde el navegador, como por ejemplo**:** la URL, el protocolo, la dirección IP del equipo y que son extraídos del objeto: **HttpServletRequest**  con los métodos apropiados. Añade, al comienzo del archivo **Datos.java**, la línea:

**package paquete;**

salida.println("<html>");

salida.println("<body>");

salida.println("<h1>Tercer servlet. Envía Datos de la petición del cliente </h1>");

salida.println("<p>URL: " + peticion.**getRequestURI()** + " </p>");

salida.println("<p> Protocolo: "+ peticion.**getProtocol()** + “"</p>");

salida.println("<p> Dirección IP remota: " + peticion.**getRemoteAddr()** +"</p>");

salida.println("</body>");

salida.println("</html>");

1. Compila el archivo:**Datos.java.** Mueve el archivo **Datos.class** al directorio: **ejemplos\ WEB-INF\classes\paquete**
2. Añade al descriptor **web.xml** de la aplicación, las etiquetas correspondientes al nuevo servlet: **Datos**  añadiendo la ruta dentro del paquete en el que se encuentra el servlet: **Datos.class**

<servlet>

<servlet-name>Datos</servlet-name>

<servlet-class>**paquete.Datos**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Datos</servlet-name>

<url-pattern>**/fin**</url-pattern>

</servlet-mapping>

1. **Despliega de nuevo** la aplicación en el contenedor **tomcat**: Mueve otra vez, la carpeta de la aplicación: **ejemplos,** al directorio de tomcat:**webapps\** , Para y arranca tomcat desde el icono de la barra de herramientas.
2. Prueba el tercer servlet. **Captura la pantalla.**



1. **Creación del archivo .war que servirá para poder desplegar la aplicación en cualquier servidor compatible J2EE.** Se empaqueta con el comando: **jar.** Borra primero la aplicación: **CATALINA\_HOME\webapps\ejemplos**
2. **Empaqueta la aplicación creando su archivo .war.**  Ejecuta desde una consola ms-dos: cambia el directorio activo, al de la aplicación: **ejemplos\**. Ejecuta: **jar cvf ../ejemplos.war .** (el último “.” representa el directorio activo se empaqueta el directorio activo pero el .war lo dejamos en el directorio padre, no en el propio directorio que estamos empaquetando).
3. **Despliegue**: mueve el archivo **ejemplos.war** a: **CATALINA\_HOME\webapps** , la aplicación se desplegará automáticamente. Arrancar tomcat.
4. Prueba los 3 servlets. Comprueba que la aplicación **ejemplos** se ha desplegado dentro del directorio **webapps\** visualizando el contenido de este directorio.



