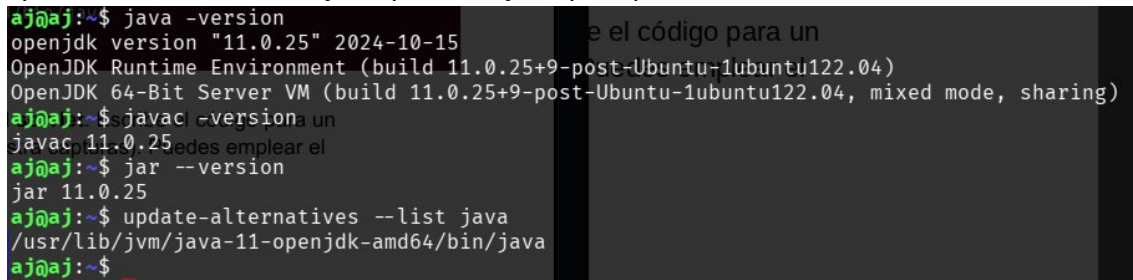


## Práctica 8. Tomcat - Servlet

Lee detenidamente cada uno de los puntos antes de realizar las tareas solicitadas.  
Revisa los recursos incluidos.

1. El objetivo de este ejercicio es aprender a compilar, desplegar y ejecutar un servlet sencillo. Para desarrollar este ejercicio es necesario disponer del JDK de Java. **Confirma las versiones** instaladas de java, javac y la ruta donde está instalado. (Capturas de pantalla de `java -version`, `javac -version`, `jar --version` y `update-alternatives --list java` para ver java path.)



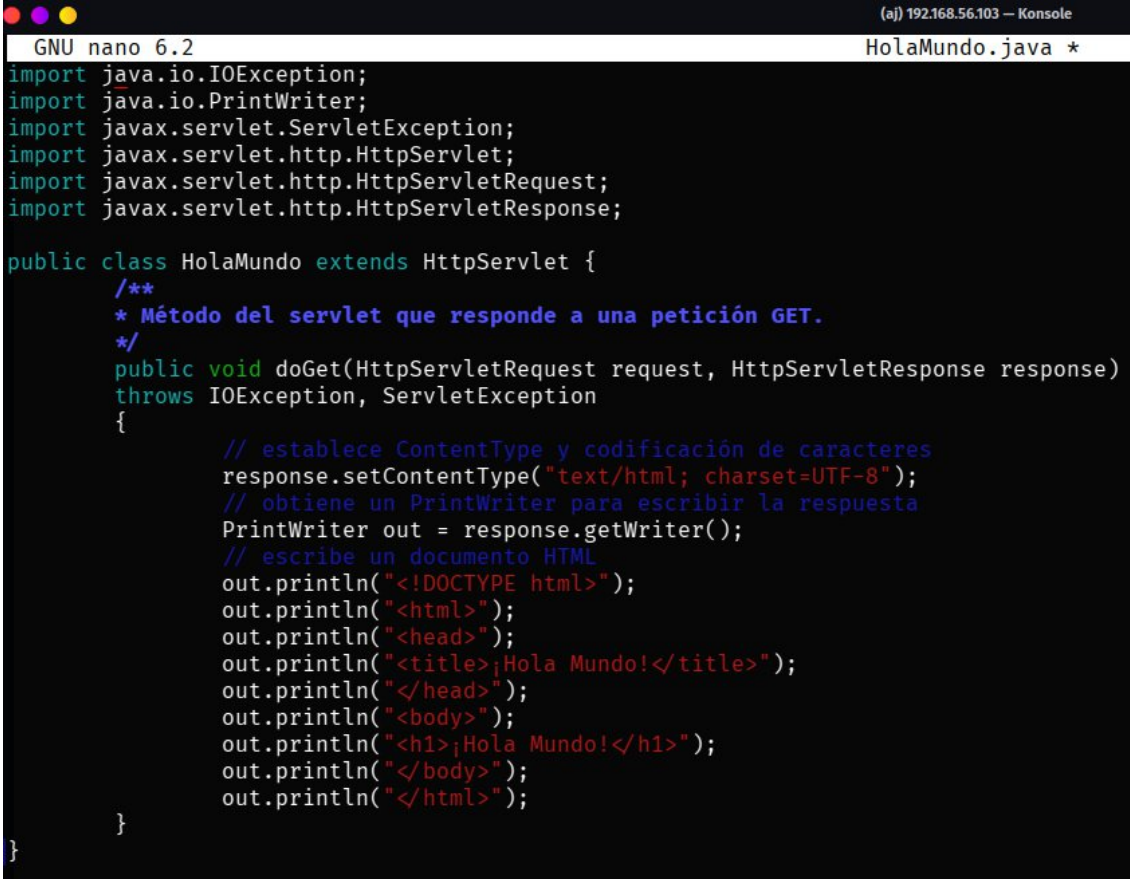
```
aj@aj:~$ java -version
openjdk version "11.0.25" 2024-10-15
OpenJDK Runtime Environment (build 11.0.25+9-post-Ubuntu-1ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.25+9-post-Ubuntu-1ubuntu122.04, mixed mode, sharing)
aj@aj:~$ javac -version
javac 11.0.25
aj@aj:~$ jar --version
jar 11.0.25
aj@aj:~$ update-alternatives --list java
/usr/lib/jvm/java-11-openjdk-amd64/bin/java
aj@aj:~$
```

2. El primer paso es escribir el **código del servlet**. Escribe el código para un servlet llamado "HolaMundo.java" (muestra capturas). Puedes emplear el siguiente código a modo de ejemplo:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HolaMundo extends HttpServlet {
    /**
     * Método del servlet que responde a una petición GET.
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        // establece ContentType y codificación de caracteres
        response.setContentType("text/html; charset=UTF-8");
        // obtiene un PrintWriter para escribir la respuesta
        PrintWriter out = response.getWriter();
        // escribe un documento HTML
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>¡Hola Mundo!</title>");
    }
}
```

```
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>¡Hola Mundo!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```



```
GNU nano 6.2                                     (aj) 192.168.56.103 — Konsole
HolaMundo.java *
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HolaMundo extends HttpServlet {
    /**
     * Método del servlet que responde a una petición GET.
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        // establece ContentType y codificación de caracteres
        response.setContentType("text/html; charset=UTF-8");
        // obtiene un PrintWriter para escribir la respuesta
        PrintWriter out = response.getWriter();
        // escribe un documento HTML
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>¡Hola Mundo!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>¡Hola Mundo!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

3. **Compila el servlet.** (Captura de `javac -cp ${CATALINA_HOME}/lib/servlet-api.jar: HolaMundo.java`). ¡Ojo! Servlet-api.jar se encuentra en las propias librerías donde está instalado Tomcat.

Desplegar un servlet consiste en incluir una serie de ficheros en un contenedor web (en nuestro caso, Tomcat) para que los clientes puedan acceder a su funcionalidad.

Normalmente, el desarrollo de un servlet forma parte de lo que se denomina una aplicación Web, que no es más que una colección de servlets, páginas HTML, JSP, clases y otros recursos que se pueden empaquetar y ejecutar en distintos contenedores web, de distintos proveedores, y que ofrecen una determinada funcionalidad a la que los clientes acceden típicamente a través de un navegador.

La especificación de servlet indica que estas aplicaciones web deben estructurarse según la siguiente jerarquía de subdirectorios:

- ✦ **Directorio raíz:** contiene los ficheros estáticos y públicos (HTML, imágenes, hojas de estilo, etc.) y JSPs.
  - Directorio **WEB-INF**: debe contener un fichero *web.xml*. Este fichero configura la aplicación. Por ejemplo, permite declarar servlets, asignarles la URL de acceso y parámetros de inicio, declarar alias y filtros, etc.
    - Directorio **classes**: debe contener los ficheros compilados (servlets) de las clases utilizadas por la aplicación web.
    - Directorio **lib**: contendrá las bibliotecas de clases adicionales (comprimidas con *.jar*) que utilice la aplicación.
  - Directorio **META-INF** (no obligatorio): Para definir contexto a nivel de aplicación.

Para desplegar, manualmente, una aplicación web que contenga el servlet creado:

```
aj@aj:~$ sudo javac -cp /opt/tomcat/lib/servlet-api.jar:. HolaMundo.java
```

4. Replica la estructura de carpetas. Crea un directorio raíz con el nombre de la aplicación (siguiendo el ejemplo, "holamundo"). Este directorio se debe crear en `#{CATALINA_HOME}/webapps/`. Dentro de este se debe crear la estructura de subdirectorios necesaria. (Capturas de las carpetas creadas). Al crear este directorio raíz se define el primer nivel de la ruta de acceso a la aplicación. Por tanto, en este caso la URL principal de la aplicación será: `http://[IP_SERVIDOR_TOMCAT]:8080/holamundo/`.

```
aj@aj:~$ sudo su
root@aj:/home/aj# cd /opt/tomcat/webapps/
root@aj:/opt/tomcat/webapps# mkdir holamundo
root@aj:/opt/tomcat/webapps# ls
docs  examples  holamundo  host-manager  manager  ROOT
root@aj:/opt/tomcat/webapps# cd holamundo/
root@aj:/opt/tomcat/webapps/holamundo# mkdir WEB-INF
root@aj:/opt/tomcat/webapps/holamundo# LS
LS: command not found
root@aj:/opt/tomcat/webapps/holamundo# ls
WEB-INF
root@aj:/opt/tomcat/webapps/holamundo# cd WEB-INF/
root@aj:/opt/tomcat/webapps/holamundo/WEB-INF# mkdir classes
root@aj:/opt/tomcat/webapps/holamundo/WEB-INF# mkdir lib
root@aj:/opt/tomcat/webapps/holamundo/WEB-INF# ls
classes  lib
root@aj:/opt/tomcat/webapps/holamundo/WEB-INF#
```

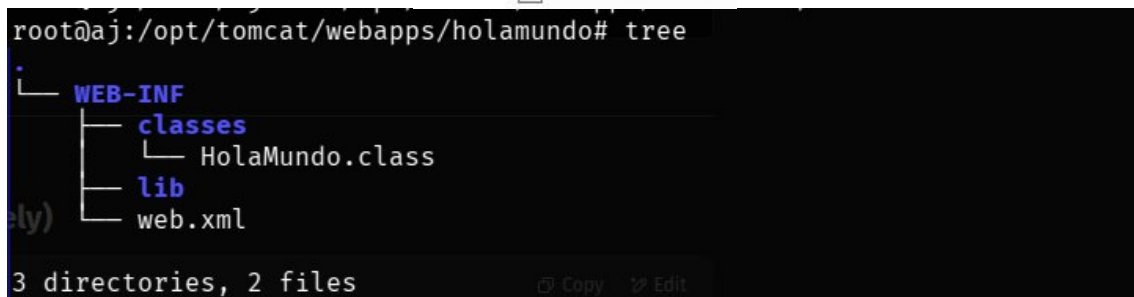
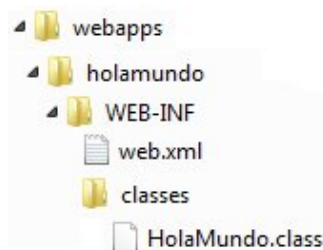
5. Crea el fichero de despliegue **web.xml** necesario (captura del contenido del fichero *web.xml*). Puedes emplear el siguiente código a modo de ejemplo:

```
<web-app>
  <servlet>
    <servlet-name>hola</servlet-name>
    <description>Servlet que presenta el mensaje "¡Hola
Mundo!".</description>
    <servlet-class>HolaMundo</servlet-class>
  </servlet>
```

```
<servlet-mapping>
  <servlet-name>hola</servlet-name>
  <url-pattern>/hola</url-pattern>
</servlet-mapping>
</web-app>
```



6. Asegúrate de que los ficheros creados están **ubicados correctamente** dentro de la estructura de carpetas (capturas de la ubicación de web.xml y de HolaMundo.class, el código java compilado).



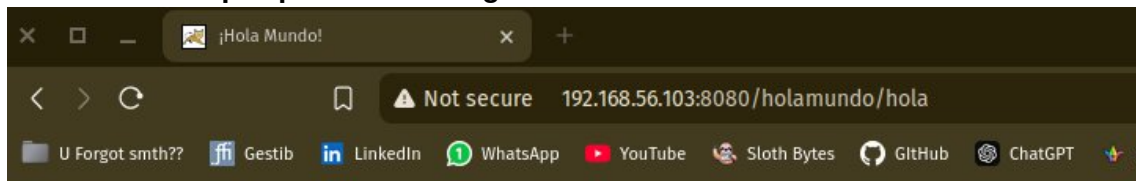
7. Detén el servidor y vuelve a arrancarlo para que Tomcat cargue la nueva aplicación web (Captura de los comandos de parada y reinicio).

```
aj@aj:~$ sudo systemctl stop tomcat.service
aj@aj:~$ sudo systemctl start tomcat.service
aj@aj:~$ sudo systemctl restart tomcat.service
aj@aj:~$ sudo systemctl status tomcat.service
● tomcat.service - Tomcat
   Loaded: loaded (/etc/systemd/system/tomcat.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2025-01-23 15:35:44 UTC; 16s ago
     Process: 3115 ExecStart=/opt/tomcat/bin/startup.sh (code=exited, status=0/SUCCESS)
    Main PID: 3122 (java)
      Tasks: 29 (limit: 2226)
     Memory: 131.2M
        CPU: 5.849s
    CGroup: /system.slice/tomcat.service
           └─3122 /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -Djava.util.logging.con

Jan 23 15:35:44 aj systemd[1]: Starting Tomcat ...
Jan 23 15:35:44 aj startup.sh[3115]: Tomcat started.
Jan 23 15:35:44 aj systemd[1]: Started Tomcat.
lines 1-14/14 (END)
```

8. Abre un navegador y conéctate a la URL

[http://\[IP\\_SERVIDOR\\_TOMCAT\]:8080/holamundo/hola](http://[IP_SERVIDOR_TOMCAT]:8080/holamundo/hola). Esto hará que se ejecute el servlet de ejemplo, que devolverá la página con el mensaje ¡Hola Mundo! (Captura del navegador). La parte final de la URL del servlet es */hola* porque se ha configurado así en el fichero *web.xml*.



## ¡Hola Mundo!

9. Crea un nuevo servlet “HolaMundo2” (puedes reutilizar el código anterior). Igual que en ejercicios anteriores, compila el código, crea el descriptor de despliegue (web.xml) y replica la estructura de carpetas. Finalmente, **empaqueta el servlet** en un fichero .war (**W**eb **a**pplication **A**Rchive). Estando en la carpeta holamundo2, emplea el comando `jar -cvf holamundo2.war *` (muestra capturas del código, de las carpetas y ficheros creados, y del comando jar y su resultado). -c sirve para crear el fichero, -v para mostrar detalles del resultado y -f para especificar el nombre del fichero. El asterisco (\*) es para incluir todos los ficheros, incluyendo subdirectorios.



```
root@aj:/opt/tomcat/holamundo2# jar -cvf WEB-INF/classes/HolaMundo2.war *
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/HolaMundo2.class(in = 925) (out= 548)(deflated 40%)
adding: WEB-INF/web.xml(in = 325) (out= 154)(deflated 52%)
adding: WEB-INF/lib/(in = 0) (out= 0)(stored 0%)
root@aj:/opt/tomcat/holamundo2# tree
.
├── WEB-INF
│   ├── classes
│   │   ├── HolaMundo2.class
│   │   └── HolaMundo2.war
│   └── lib
│       └── web.xml
└── 3 directories, 3 files
```

10. Ahora ves al **gestor de aplicaciones web de Tomcat** vía navegador (Manager App). Recuerda que deberás acceder con los credenciales introducidos en la practica anterior (también puedes crear nuevos usuarios modificando el fichero de configuración *tomcat-users.xml*). Desde esta página, despliega el fichero .war generado en el punto previo (saca capturas del proceso). ¿Puedes acceder a [http://\[IP\\_SERVIDOR\\_TOMCAT\]:8080/holamundo2/hola?](http://[IP_SERVIDOR_TOMCAT]:8080/holamundo2/hola?) Muestra captura.

```
root@aj:/opt/tomcat# cp holamundo2/WEB-INF/classes/HolaMundo2.war webapps/
root@aj:/opt/tomcat# cd webapps/
root@aj:/opt/tomcat/webapps# ls
docs  examples  holamundo  HolaMundo2  HolaMundo2.war  host-manager  manager  ROOT
```

### Recursos:

- Crear y arrancar un servlet:  
<https://studyglance.in/servlet/display.php?tno=6&topic=Servlet-Example>
- JAR – Java Archive Tool:  
<https://docs.oracle.com/javase/6/docs/technotes/tools/windows/jar.html>
- Java JDK y JRE:  
<https://www.oracle.com/es/java/technologies/downloads/>

### Condiciones de entrega:

- La práctica se **debe** entregar de forma **individual**, cada uno debe presentar sus propias respuestas. Sin embargo, se puede trabajar en equipo.
- Se debe entregar un documento de texto (.pdf, .docx, .odt, etc.) con los ejercicios correctamente ordenados, identificados y **numerados**.
- En cada página del documento debe aparecer el nombre completo del alumno.
- La nota comprenderá un valor numérico entre 0 y 10.
- **La fecha límite de entrega es la indicada en Google Classroom.**
- **Se podrá entregar hasta 72 horas más tarde de la fecha límite pero con una penalización sobre su puntuación (no será posible aspirar al 10).**



**Francesc  
de Borja Moll**  
Centre Integrat de  
Formació Professional

[www.cifpfbmoll.eu](http://www.cifpfbmoll.eu)  
C/ Caracas, 6 - 07007 - PALMA  
Tef. 971278150  
[cifpfrancescdeborjamoll@educaib.eu](mailto:cifpfrancescdeborjamoll@educaib.eu)

**FORMACIÓ**  
**PROFESSIONAL** **F**  
ILLES BALEARS