

En alguno de los ejercicios de esta tarea es necesario realizar una instalación de software. Para dichas instalaciones es necesario disponer de un sistema (preferiblemente una máquina virtual), que cuente con una instalación limpia del sistema operativo Ubuntu en la versión 14.04 LTS, en la que el entorno de red este apropiadamente configurado. Dicho sistema debe tener conexión a Internet y es necesario trabajar con permisos de root (a través de sudo).

Para la realización de los ejercicios donde sea necesario realizar una instalación o configuración, es necesario indicar cada uno de los pasos realizados, los comandos implicados y/o los archivos editados, cuando proceda. Y siempre que sea posible, y ajustándose al tamaño máximo de archivo para la tarea, realizar una captura de pantalla que demuestre la realización de los diferentes pasos, y donde, también dentro de lo posible, sea visible vuestra foto de perfil en la plataforma de enseñanza a distancia.

1. Una aplicación web desarrollada conforme a la especificación Java Servlets 2.2 puede ser desplegada en diferentes contenedores de servlets y servidores de aplicaciones manteniendo su funcionalidad y ningún tipo de modificación, ¿cuál es la estructura de directorios que debe tener? ¿Qué podemos poner en dichos directorios? ¿Cuál es el objetivo del archivo web.xml y donde se sitúa?

La estructura básica de un Servlet sería esta:

```
- /webapps/
  |
  |-- /Nombre_aplicacion/
  |   |
  |   |-- /WEB-INF/
  |   |   |
  |   |   |-- /classes/ (carpeta que contiene los ficheros compilados de las clases de la aplicación)
  |   |   |-- /lib/ (Archivos .jar, clases adicionales comprimidas usadas en la aplicación web)
  |   |   |-- /web.xml (Archivo de configuración de la aplicación, seguridad valores de inicio en variables, etc)
  |   |   |
  |   |   |-- /index.html (Página estática)
```

Como vemos nuestra aplicación debe ir alojada dentro de la carpeta /webapps, dentro de una carpeta que normalmente es el nombre de nuestro proyecto, en este caso "Nombre_aplicacion". Dentro de esta aplicación tiene que estar al menos la carpeta *WEB-INF* donde estarán los recursos Web de la aplicación web.

Dentro de *WEB-INF* debe estar el archivo web.xml que describe tanto parámetros de configuración, de inicialización, restricciones de seguridad, etc.

Al mismo nivel del archivo web.xml deben estar la carpeta *classes* y la carpeta *lib*, en *classes* estarán todos los servlets y cualquier otra clase de utilidad o complementaria que necesite la aplicación para ejecutarse. Suele tener solamente archivos .class. La carpeta *lib* contiene archivos comprimidos Java (.jar) de los que depende la aplicación.

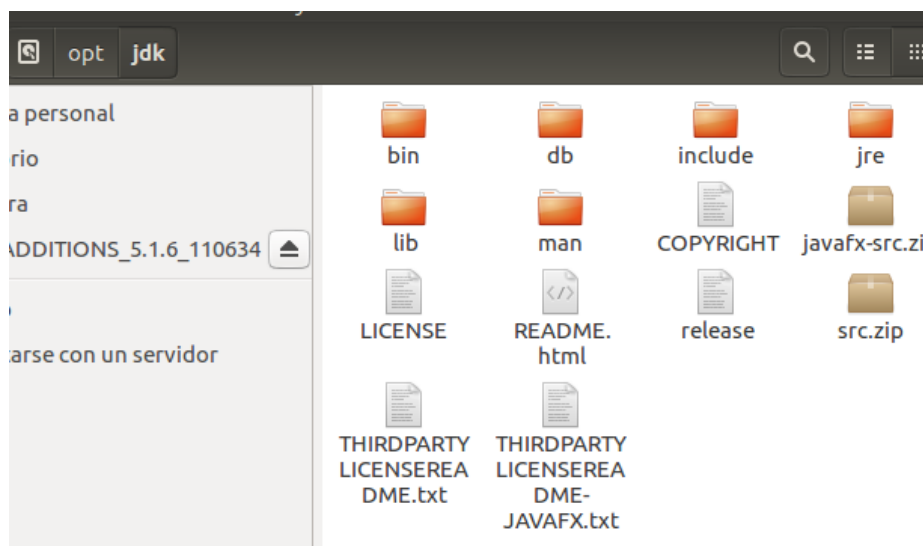
Al mismo nivel que la carpeta *WEB-INF* o creando subdirectorios para organizarlos deben estar los archivos .html ó .jsp de la aplicación web.

2. Instalar el JDK 8 de Oracle.

He tomado la decisión de hacerlo a mano, una vez descargado el jdk8 de Oracle, nos vamos a la carpeta `opt` y creamos dentro la carpeta `jdk`:

```
[sudo] password for virtual:
root@virtual-VirtualBox:/home/virtual# cd /opt
root@virtual-VirtualBox:/opt# mkdir tomcat
root@virtual-VirtualBox:/opt# ls
lamp  tomcat  VBoxGuestAdditions-5.1.6
root@virtual-VirtualBox:/opt#
```

Dentro de la carpeta `jdk` descomprimos los archivos que descargamos de manera que la carpeta `bin` quede debajo de `jdk` (`/opt/jdk/bin`):



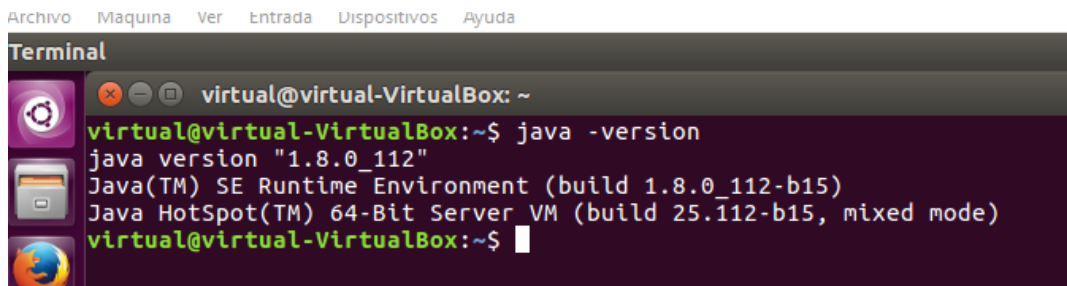
Configuramos Ubuntu para que el `PATH` y el `JAVA_HOME` se actualicen al arrancar, para ello creamos el archivo `/etc/profile.d/jdk.sh` y lo editamos introduciendo el siguiente código:

```
JDK_DIR="/opt/jdk"
if [ -d "$JDK_DIR" -a -d "$JDK_DIR/bin" ]; then
    PATH="$PATH:$JDK_DIR/bin"
    JAVA_HOME="$JDK_DIR"
    export PATH
    export JAVA_HOME
fi
```

Reiniciamos el PC y no debe salirnos ningún error, además podemos comprobar que las variables las ha cogido bien introduciendo estos comandos `sudo su -` y después `echo $JAVA_HOME` que nos debe mostrar la ruta `/opt/jdk` y después el comando `echo $PATH` donde entre las diferentes rutas que nos muestra la `opt/jdk/bin`:

```
root@virtual-VirtualBox:~# sudo su -
root@virtual-VirtualBox:~# echo $JAVA_HOME
/opt/jdk
root@virtual-VirtualBox:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/opt/jdk/bin
root@virtual-VirtualBox:~#
```

Si hacemos en consola el comando “java -version” debe aparecernos esto:



Archivo Maquina Ver Entrada Dispositivos Ayuda

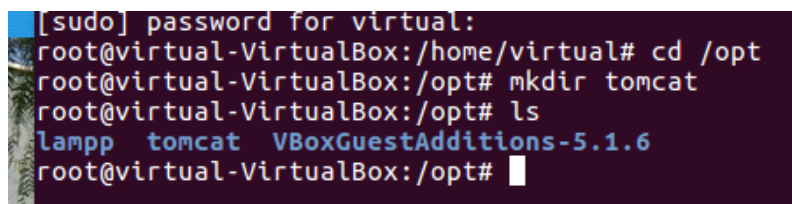
Terminal

virtual@virtual-VirtualBox: ~

```
virtual@virtual-VirtualBox:~$ java -version
java version "1.8.0_112"
Java(TM) SE Runtime Environment (build 1.8.0_112-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.112-b15, mixed mode)
virtual@virtual-VirtualBox:~$
```

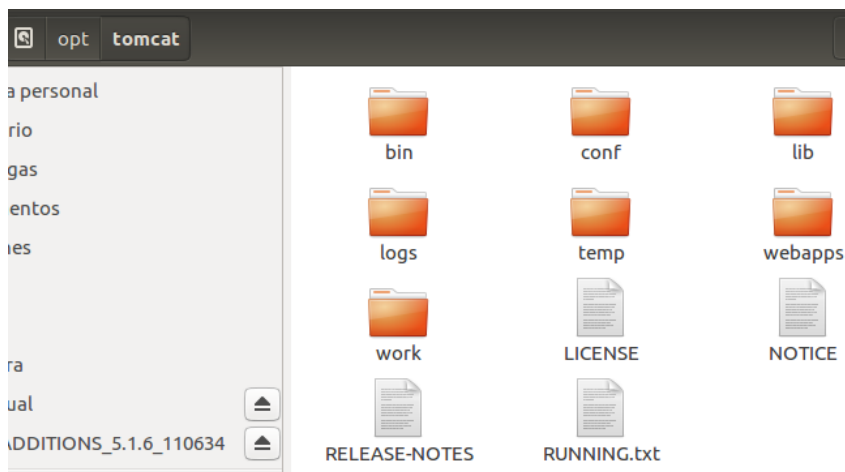
3. Instalación de Tomcat 8.5

He tomado la decisión de hacerlo a mano, una vez descargado el tomcat 8.5.11 de apache, nos vamos a la carpeta opt y creamos dentro la carpeta tomcat:



```
[sudo] password for virtual:
root@virtual-VirtualBox:/home/virtual# cd /opt
root@virtual-VirtualBox:/opt# mkdir tomcat
root@virtual-VirtualBox:/opt# ls
lampp tomcat VBoxGuestAdditions-5.1.6
root@virtual-VirtualBox:/opt#
```

Dentro de la carpeta tomcat descomprimos los archivos que descargamos:



Pasamos a crear las variables de entorno TOMCAT_HOME y CATALINA_HOME, modificamos la variable de entorno PATH, esto lo haremos creando un nuevo archivo en la ruta “/etc/profile.d/tomcat.sh” que editaremos insertando el siguiente código:

```
virtualBox:~$ sudo su
virtual:
virtualBox:/home/virtual# gedit /etc/profile.d/tomcat.sh

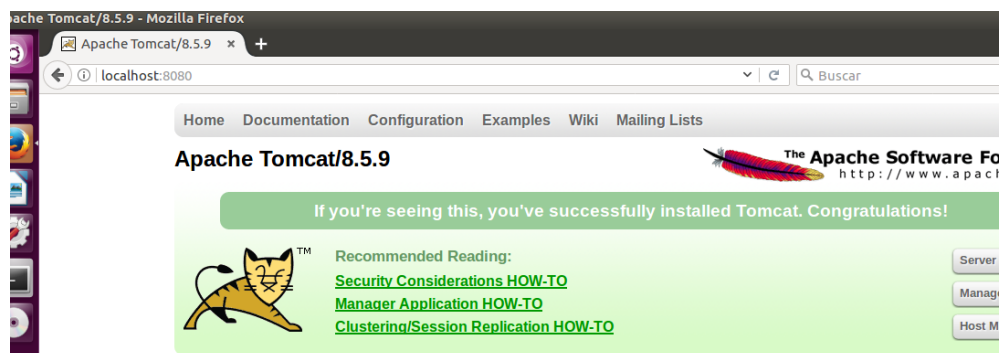
*tomcat.sh
/etc/profile.d

T_DIR="/opt/tomcat"
if [ -d "$T_DIR" -a -d "$T_DIR/bin" ]; then
    PATH="$PATH:$T_DIR/bin"
    export PATH
    CATALINA_HOME="$T_DIR"
    export CATALINA_HOME
    TOMCAT_HOME="$T_DIR"
    export CATALINA_HOME
fi
```

Reiniciamos el equipo y arrancamos el tomcat con los siguientes comandos (“-i” para usar las variables de entorno), “`sudo -i catalina.sh start`”:

```
virtual@virtual-VirtualBox: ~
virtual@virtual-VirtualBox:~$ sudo -i catalina.sh start
[sudo] password for virtual:
Using CATALINA_BASE: /opt/tomcat
Using CATALINA_HOME: /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME: /opt/jdk
Using CLASSPATH: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Tomcat started.
virtual@virtual-VirtualBox:~$
```

Ya podemos hacer uso de la URL <http://localhost:8080/> en nuestro equipo para acceder al servidor Tomcat:



Para parar Tomcat debemos hacer uso del comando “`sudo -i catalina.sh stop`”

4. Creación de aplicación web.

Para esta tarea primero haremos la estructura de archivos para la aplicación web. Por ello nos dirigimos a la ubicación “`/opt/tomcat/webapps/`” y crearemos una carpeta llamada `ut2_app1`, que es el nombre de nuestra aplicación/proyecto:

```
lualBox:~$ sudo su
lBox:/home/virtual# cd /opt/tomcat/webapps/
lBox:/opt/tomcat/webapps# mkdir ut2_app1
lBox:/opt/tomcat/webapps# cd ut2_app1/
lBox:/opt/tomcat/webapps/ut2_app1# mkdir WEB-INF
lBox:/opt/tomcat/webapps/ut2_app1# gedit index.html
```

Como vemos dentro de la carpeta de nuestro proyecto hemos seguido creando nuestro árbol de directorios predefinido para aplicaciones Java servlets 2.2. También he creado y editado el archivo index.html:

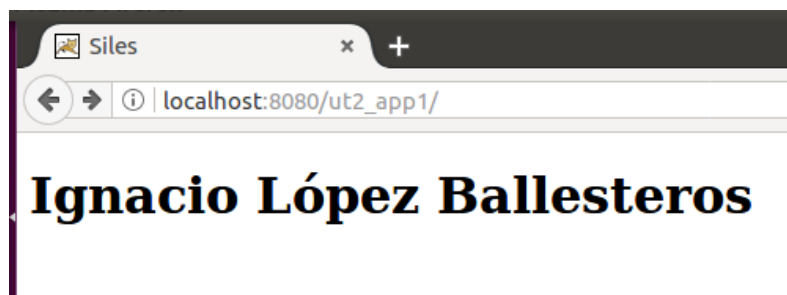
```
<!DOCTYPE html>
<html>
  <head>
    <title>Siles</title>
    <meta charset="UTF-8" />
  </head>
  <body>
    <h1>Ignacio López Ballesteros</h1>
  </body>
</html>
```

Sigo creando la estructura de directorios y quedaría así:

```
├── web.xml
└── ut2_app1
    ├── index.html
    └── WEB-INF
        ├── classes
        ├── lib
        └── web.xml
```

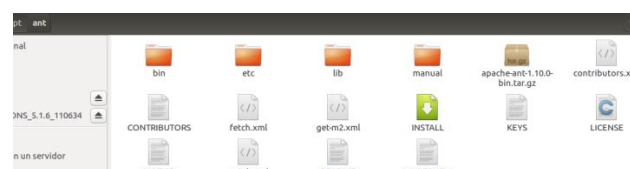
105 directories 578 files

Probamos que funcione con el servidor Tomcat. Lo arrancamos y desde el navegador con la URL http://localhost:8080/ut2_app1/ debe ejecutarse nuestra aplicación web:



5. Instalación de Ant

Una vez descargado Ant desde la página de Apache, nos vamos a la carpeta opt y creamos dentro la carpeta ant:



Una vez hecho esto pasamos a crear la variable de entorno ANT_HOME y configurar la variable PATH. Para ello creamos y editamos el archivo `/etc/profile.d/ant.sh` de tal manera que quede así:

```
contributors.xml      INSTALL      manual      WHATSNEW
root@virtual-VirtualBox:/opt/ant# gedit /etc/profile.d/ant.sh
```

```
ant.sh
/etc/profile.d
A_DIR="/opt/ant"
if [ -d "$A_DIR" -a -d "$A_DIR/bin" ]; then
    PATH="$PATH:$A_DIR/bin"
    ANT_HOME="$A_DIR"
    export PATH
    export ANT_HOME
fi
```

Reiniciamos el equipo y comprobamos que las variables de entorno están bien:

```
[sudo] password for virtual:
root@virtual-VirtualBox:~# echo $ANT_HOME
/opt/ant
root@virtual-VirtualBox:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/ant/bin:/snap/
bin:/opt/jdk/bin:/opt/tomcat/bin
root@virtual-VirtualBox:~#
```

Para comprobar que ant se ha instalado correctamente tenemos que en consola introducir el comando `"ant"`, nos debe de salir esto:

```
root@virtual-VirtualBox:~# ant
Buildfile: build.xml does not exist!
Build failed
root@virtual-VirtualBox:~#
```

6. Uso de Ant para generar un archivo WAR

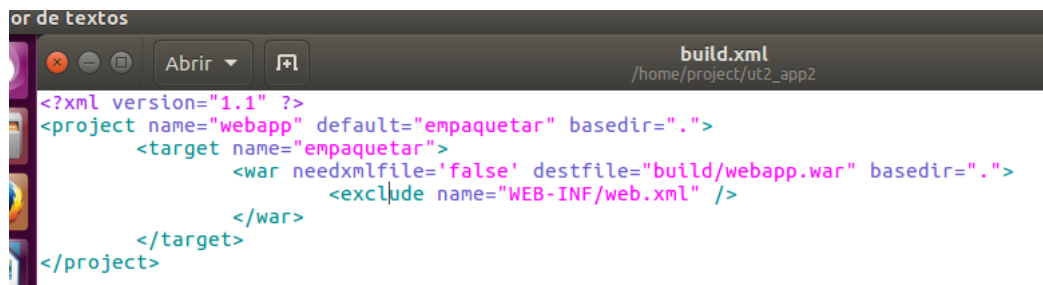
Para realizar esta tarea copiaremos la carpeta `ut2_app1` del ejercicio 4 ya la pegaremos dentro de una carpeta local (por ejemplo `/home/project`). Creamos un archivo llamado `build.xml` dentro de `Project`. Copiamos el proyecto `ut2_app1` y lo pegaremos dentro de la misma ubicación con el nombre `ut2_app2`. Debería quedar de esta manera:

```
root@virtual-VirtualBox:/home/project# tree
.
├── build.xml
├── ut2_app1
│   ├── conjsp.jsp
│   ├── index.html
│   └── WEB-INF
│       ├── classes
│       └── lib
└── ut2_app2
    ├── conjsp.jsp
    ├── index.html
    └── WEB-INF
        ├── classes
        └── lib

8 directories, 5 files
```

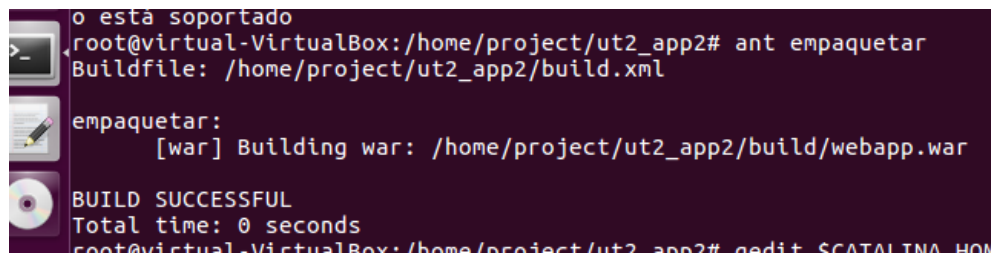
Ahora modificaremos el archivo index.html para que muestre nuestro DNI (sería modificar solamente el contenido de las etiquetas `<title>Nombre de mi pueblo</title>` por `<title>mi DNI</title>`).

El siguiente paso será editar el archivo build.xml para que funcione en mi proyecto y construya un archivo .war a través de Ant. Hemos excluido el archivo web.xml, ya que nos daría error si no lo tenemos. Quedaría así:



```
<?xml version="1.1" ?>
<project name="webapp" default="empaquetar" basedir=".">
  <target name="empaquetar">
    <war needxmlfile="false" destfile="build/webapp.war" basedir=".">
      <exclude name="WEB-INF/web.xml" />
    </war>
  </target>
</project>
```

Ahora, para construir el .war tenemos que desde en la línea de comandos en dentro de nuestra carpeta de proyecto /ut2_app2/ ejecutar el comando “ant empaquetar”, y veremos si se ha creado el archivo webapp.war correctamente:



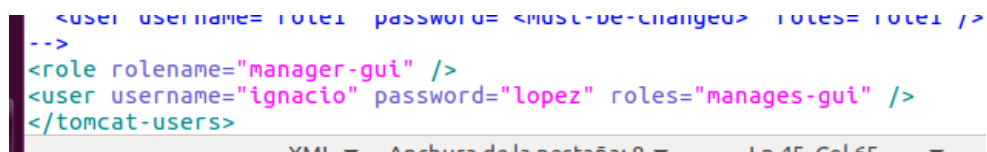
```
root@virtual-VirtualBox:/home/project/ut2_app2# ant empaquetar
Buildfile: /home/project/ut2_app2/build.xml

empaquetar:
[war] Building war: /home/project/ut2_app2/build/webapp.war

BUILD SUCCESSFUL
Total time: 0 seconds
root@virtual-VirtualBox:/home/project/ut2_app2#
```

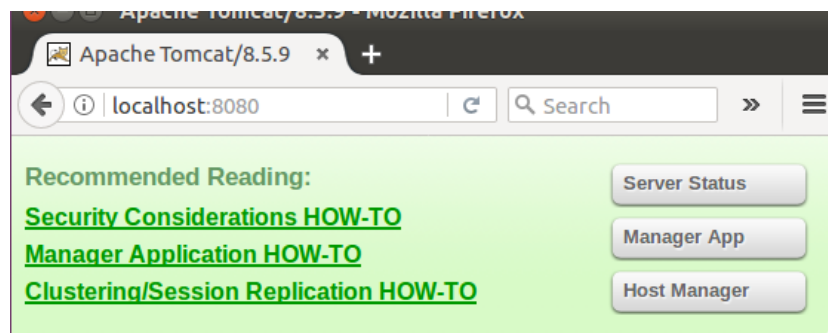
7. Utilización de gestor de aplicaciones de Tomcat

Para llevar a cabo esta tarea debemos editar el archivo de usuarios “\$CATALINA_HOME/conf/tomcat-user.xml”, introduciendo esto:



```
<user username="root" password="root" roles="root" />
-->
<role rolename="manager-gui" />
<user username="ignacio" password="lopez" roles="manager-gui" />
</tomcat-users>
```

Reiniciamos Tomcat y entonces accedemos a Tomcat a través del navegador con la dirección <http://localhost:8080>.



Pulsamos el botón Manager App, nos pedirá las credenciales donde introduciremos las que hemos creado y si todo es correcto debe aparecernos una página como esta:



Gestor de Aplicaciones Web de Tomcat

Mensaje:	OK
----------	----

Gestor			
Listar Aplicaciones	Ayuda HTML de Gestor	Ayuda de Gestor	Estado de Servidor

Aplicaciones					
Trayectoria	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado	Welcome to Tomcat	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/docs	Ninguno especificado	Tomcat Documentation	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos

El siguiente paso será desplegar la aplicación anterior con esta herramienta web, para ello bajamos en la página web hasta el final de la tabla y veremos un apartado llamado Desplegar, que consta de otros dos subapartados (Desplegar directorio o archivo WAR localizado en servidor, y otro, Archivo WAR a desplegar), utilizamos el segundo subapartado ya que nuestro archivo WAR aun no lo hemos llevado al servidor.

Pulsamos el botón Browse (1) y buscamos el archivo WAR que generamos en el ejercicio anterior ("/home/Project/ut2_app2/build/", (2)):

Seleccione archivo WAR a cargar

1

 No file selected.

home project ut2_app2 build

2

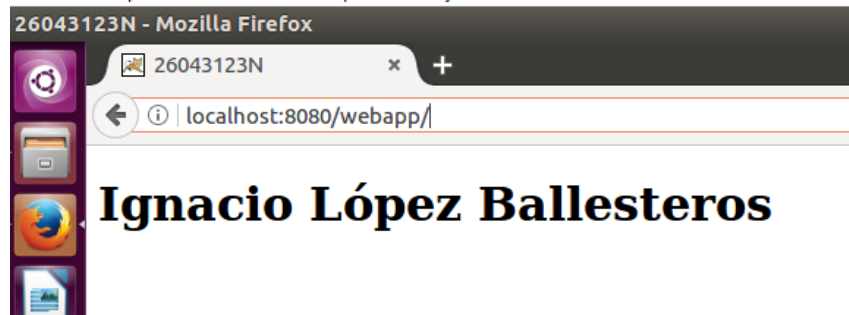
Nombre

webapp.war

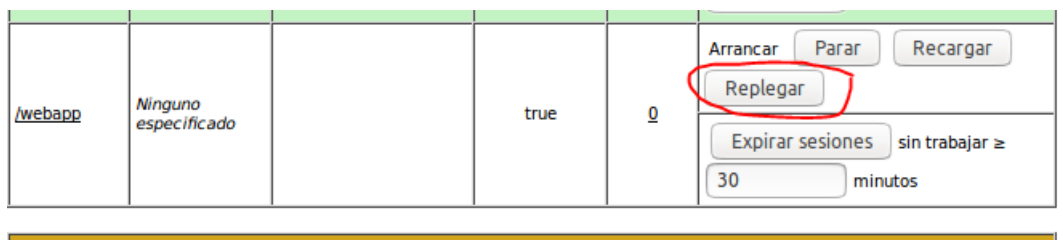
Le damos al botón desplegar y observamos que ya nos aparece en la tabla (webapp):

/ut2_app1	Ninguno especificado		true	0	Arran...
/webapp	Ninguno especificado		true	0	Arran...

Para comprobar que se ha desplegado ejecutamos en el navegador la ruta <http://localhost:8080/webapp> nos debe mostrar nuestra aplicación:



Tal y como nos pide el ejercicio replegaremos la aplicación para ello pulsaremos el botón replegar de la aplicación webapp:



8. Generar un log de acceso

Primero hacemos una copia de seguridad del archivo “/ \$CATALINA_HOME/conf/server.xml” y una vez guardada la copia en lugar seguro abrimos este mismo archivo para editarlo.

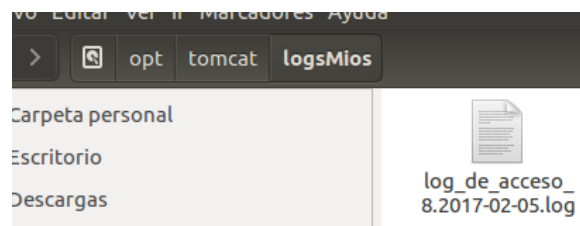
Tendremos que buscar el contenedor que nos exige el ejercicio el engine con nombre “Catalina” (<engine name=“Catalina” defaultHost=“localhost”> ...), y dentro de esta etiqueta y no de otras, para que abarque todos los accesos a este motor de Tomcat, añadiremos las siguientes líneas:

```
</Host>
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logsMios"
        prefix="log_de_acceso_8" suffix=".log"
        pattern="%h %l %u %t "%r" %s %b" />
</Engine>
```

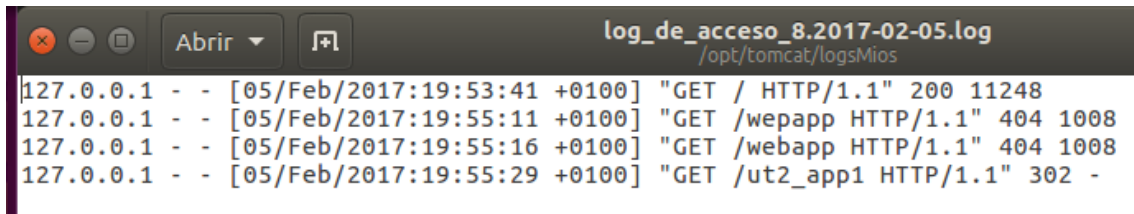
Creamos la carpeta logsMios dentro de Tomcat (“/opt/tomcat/logsMios”) y reiniciamos tomcat.

Accedemos a la carpeta donde indicamos que se guardaran los logs de acceso “/opt/tomcat/logsMios” y observamos si después de haber accedido a nuestras aplicaciones, se han generado ya esos registros:

Primer registro:



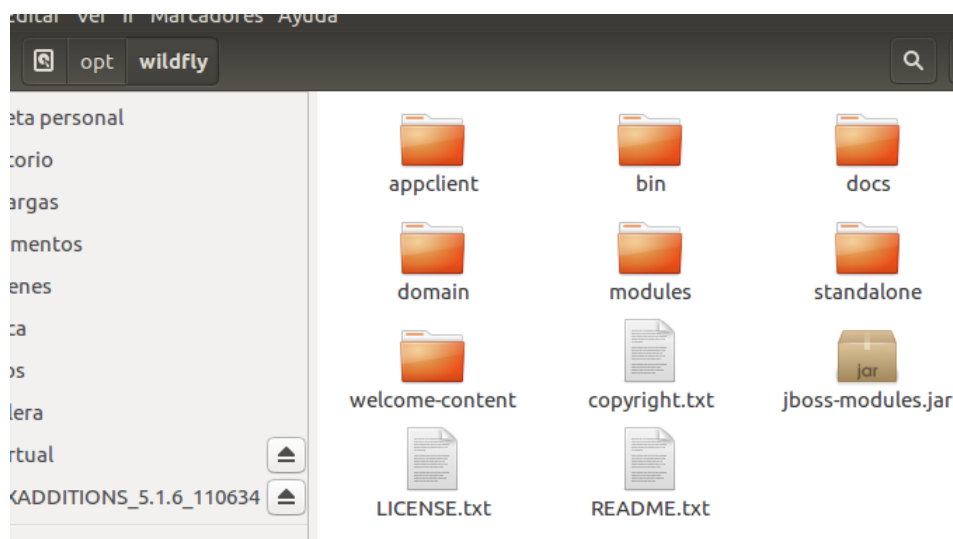
Contenido del registro:



```
log_de_acceso_8.2017-02-05.log
/opt/tomcat/logsMios
127.0.0.1 - - [05/Feb/2017:19:53:41 +0100] "GET / HTTP/1.1" 200 11248
127.0.0.1 - - [05/Feb/2017:19:55:11 +0100] "GET /wepapp HTTP/1.1" 404 1008
127.0.0.1 - - [05/Feb/2017:19:55:16 +0100] "GET /webapp HTTP/1.1" 404 1008
127.0.0.1 - - [05/Feb/2017:19:55:29 +0100] "GET /ut2_app1 HTTP/1.1" 302 -
```

9. Instalar y configurar WildFly

Nos vamos a la carpeta `/opt` y creamos dentro la carpeta `/wildfly` donde descomprimos la carpeta que hemos descargado de la página <http://wildfly.org/downloads/> (la versión 10.1.0 en nuestro caso). La carpeta en `/opt` quedaría así:



Una vez hecho esto nos dirigimos a la ruta `"/opt/wildfly/standalone/configuration/"`, y abrimos el archivo `standalone.xml` para editarlo con el objetivo de cambiar el puerto por el que trabaja para no entrar en conflicto con Tomcat, ya que por defecto el puerto con el que trabaja Wildfly es el mismo con el que trabaja Tomcat, el 8080, así que le asignaremos el puerto 8908 a Wildfly.

Para ello tenemos que localizar la línea:

```
<socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
<socket-binding name="http" port="${jboss.http.port:8908}"/>
<socket-binding name="https" port="${jboss.https.port:8443}"/>
```

Y aquí en esta línea hemos cambiado el 8080 por el 8908. Guardamos y salimos.

Ahora toca iniciar Wildfly y probar que funciona, para ello debemos crear un usuario de gestión, para ello hacemos lo siguiente:

- Primero cargamos las variables de entorno para que no tenga problemas para acceder a java wildfly, con el comando `"sudo su -"`.

- Después creamos el usuario con el comando `"/opt/wildfly/bin/add-user.sh"` que ejecutará ese mismo script, preguntándonos que tiene de usuario vamos a crear y sus credenciales, además de otras preguntas.
- Una vez hecho esto arrancamos wildfly con el comando `"/opt/wildfly/bin/standalone.sh"` y nos vamos al navegador y ponemos la URL <http://localhost:8908>, debe aparecer la siguiente página:



Ahora vamos a configurar WildFly para que funcione como un servicio de Linux, primero copiaremos el script init de Linux en el directorio `/etc/init.d`

```
root@virtual-VirtualBox:/home/virtual# cp /opt/wildfly/docs/contrib/scripts/init.d/wildfly-init-debian.sh /etc/init.d/wildfly
root@virtual-VirtualBox:/home/virtual#
```

Creamos un usuario del sistema solo para Wildfly ejecutando el siguiente comando:

```
[sudo] password for virtual:
root@virtual-VirtualBox:~# adduser --no-create-home --disabled-password --disabled-login wildfly
Añadiendo el usuario 'wildfly' ...
Añadiendo el nuevo grupo 'wildfly' (1001) ...
Añadiendo el nuevo usuario 'wildfly' (1001) con grupo 'wildfly' ...
No se crea el directorio personal '/home/wildfly'.
Cambiando la información de usuario para wildfly
Introduzca el nuevo valor, o presione INTRO para el predeterminado
Nombre completo []:
Número de habitación []:
Teléfono del trabajo []:
Teléfono de casa []:
Otro []:
¿Es correcta la información? [S/n] s
root@virtual-VirtualBox:~#
```

Nos vamos a la carpeta `/opt` para cambiar el propietario de wildfly al usuario del sistema wildfly:

```
root@virtual-VirtualBox:~# cd /opt
root@virtual-VirtualBox:/opt# chown -R wildfly.wildfly wildfly*
root@virtual-VirtualBox:/opt#
```

Ahora ejecutamos `update-rd` para asegurarnos de que wildfly se inicia al reiniciar y copiamos el archivo `"wildfly.conf"` en `"/etc/default/wildfly"`:

```
root@virtual-VirtualBox:/home/virtual# cp /opt/wildfly/docs/contrib/scripts/init.d/wildfly.conf /etc/default/wildfly
root@virtual-VirtualBox:/home/virtual#
```

Configuramos el archivo `"/etc/default/wildfly"` de esta manera (descomentarlo):

```
# not necessarily for JBoss as itself.
# default location: /etc/default/wildfly

## Location of JDK
#JAVA_HOME="/opt/jdk/bin/java"

## Location of WildFly
JBOSS_HOME="/opt/wildfly"

o ## The username who should own the process.
JBOSS_USER=wildfly

t ## The mode WildFly should start, standalone or
domain
JBOSS_MODE=standalone

t ## Configuration for standalone mode
JBOSS_CONFIG=standalone.xml

## Configuration for domain mode
# JBOSS_DOMAIN_CONFIG=domain.xml
#JBOSS_HOST_CONFIG=host-master.xml

## The amount of time to wait for startup
STARTUP_WAIT=120

## The amount of time to wait for shutdown
SHUTDOWN_WAIT=120

## Location to keep the console log
JBOSS_CONSOLE_LOG="/var/log/wildfly/console.log"
```

Guardamos y ejecutamos en consola el comando `"service wildfly start"`, con esto arrancaremos WildFly, nos vamos al navegador con URL <http://localhost:8908> y se cargara la web de wildfly.



10. Accediendo a la consola de administración

En la tarea anterior, sin saber que en esta tarea me lo pedirían creé un usuario y contraseña. De todos modos lo volveré a hacer con las características que pide esta tarea (usuario = "ignacio", y contraseña = "ignacio").

Para ello en consola ejecutamos el comando `"/opt/wildfly/bin/add-user.sh"`:

```
root@virtual-VirtualBox:~# /opt/wildfly/bin/add-user.sh
What type of user do you wish to add?
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
(a):
```

Elegimos la opción a y nos pedirá que introduzcamos el nombre de usuario (ignacio) y contraseña (ignacio), nos avisará que la contraseña introducida no debería ser la misma que el nombre de usuario, aun así insistimos y vuelve a pedirnos la contraseña. Nos pregunta una serie de cuestiones más.

Arrancamos wildfly con `"service wildfly start"` y nos dirigimos a la web donde accedemos al enlace *"Administration Console"*:

Welcome to WildFly 10

Your WildFly 10 is running.

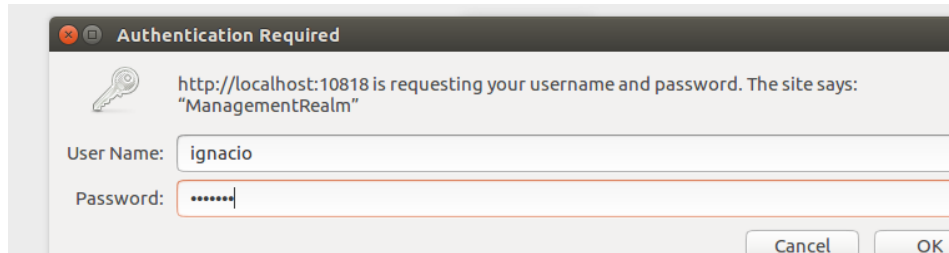
[Documentation](#) | [Quickstarts](#) | [Administration Console](#)

[WildFly Project](#) | [User Forum](#) | [Report an issue](#)

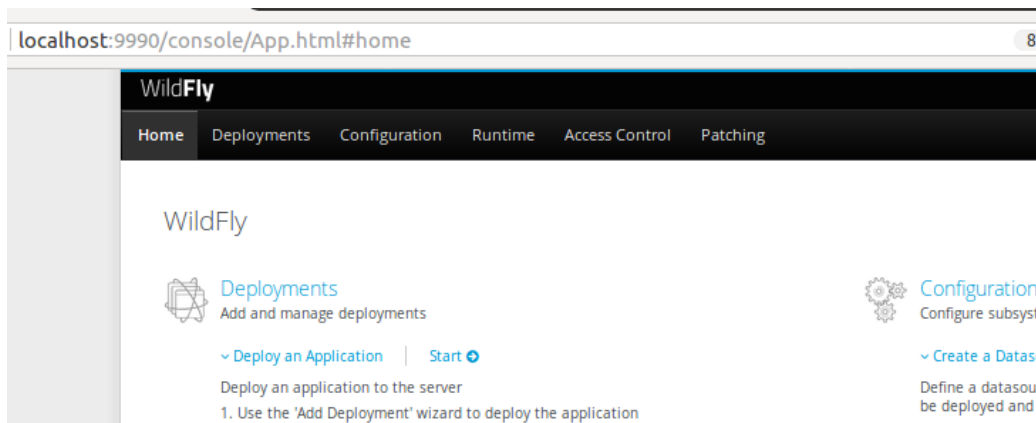
Join the WildFly Community

To place this page simply deploy your own war with / as its context path.

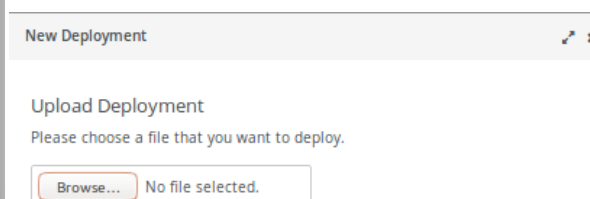
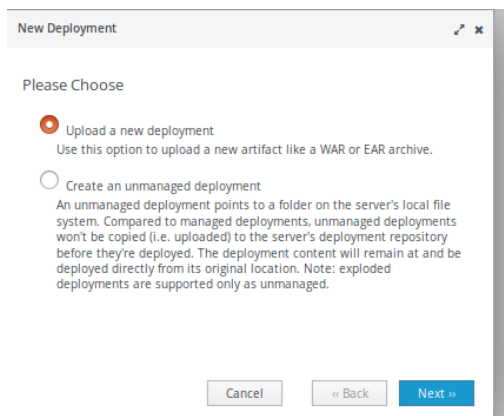
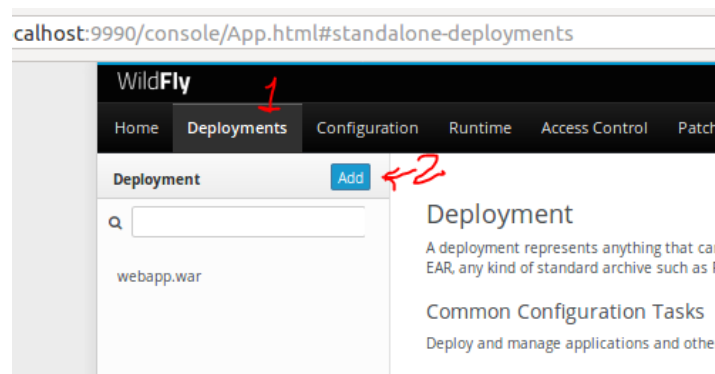
Nos pedirán las credenciales y metemos las del usuario que acabamos de crear para WildFly:



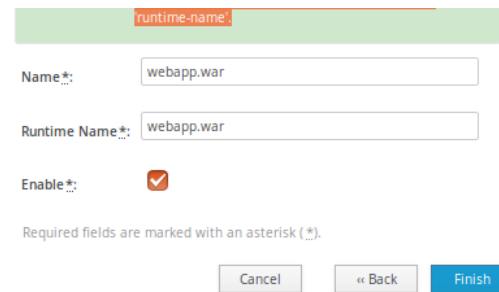
Y accederemos a la consola de administración.



Para desplegar nuestra aplicación comprimida en `.war` hacemos click en el enlace *Deployments* (1) y después en *Add* (2) para buscar nuestro `.WAR`:



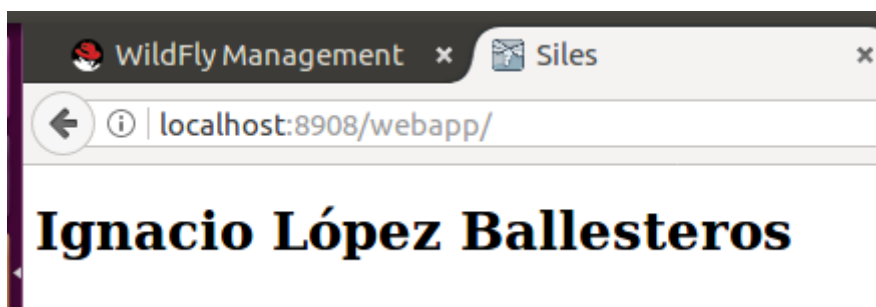
Le damos a la opción de *Upload a new deployment* y en el dialogo siguiente buscamos nuestro .WAR, dejamos por defecto los nombres y finalizamos:



The dialog shows the following fields and options:

- runtime-name:** (highlighted in green)
- Name*:** webapp.war
- Runtime Name*:** webapp.war
- Enable*:** ☒
- Required fields are marked with an asterisk (*).
- Buttons: Cancel, « Back, Finish

Nos saldrá un mensaje de éxito verde y podremos comprobar en el navegador que funciona con la URL <http://localhost:8908/webapp/>:



También pongo un pantallazo del estado de la aplicación en la consola de Wildfly:

