Corrección examen (Despliegues con TOMCAT):

Ejercicio 1: Despliegue básico de aplicación en tomcat

Se trabaja con proyecto1.war descargado del aula en una máquina debian a la que se accede usando putty:

Se pide despliegue en ROOT context.

Primero configuramos la red e instalamos tomcat :

```
apt install tomcat9
```

Para realizar este ejercicio en primer lugar desplegaremos la app de manera normal para comprobar que no hay errores, tras eso moveremos la app de lugar (al ROOT context).

Desplegaremos con el manager, para ellos necesitamos acceso, creamos un usuario:

```
Cd $CATALINA_BASE
```

Y entramos en : Conf > tomcat-users.xml : creamos user con roles: “admin-gui,manager,manager-gui”.

- Reiniciamos tomcat9: `Systemctl restart tomcat9`
- Ahora desplegamos la aplicación a través del asistente del manager.
- Podemos verlo con : ip:8080/default.jsp (es la única page disponible en el proyecto).

Ahora procedemos a desplegar en ROOT context:

Una forma es renombrar la app como ROOT.war y que se genere mediante el autodespliegue sin la carpeta ROOT (/var/lib/tomcat9/webapps/ROOT) la otra opción es trabajar con la carpeta ROOT.

Se puede desplegar directamente en el manager si lo hemos llamado ROOT.war.

Ahora debemos modificar el descriptor de despliegue (web.xml) para que muestre por defecto “default.jsp”.

En webapps, dentro de la app, buscamos la carpeta WEB-INF y editamos el web.xml:

```
<welcome-file-list>
        <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
```

Por último, con la app desplegada, debemos **redirigir el error 404 a una página personalizada**.

(disponemos de una página ya creada ErrorPage404.jsp)

Vamos a editar la página de error por defecto para el 404 a través del web.xml

```
<error-page>
        <error-code>404</error-code>
        <location>/WEB_INF/errorpages/ErrorPage404.jsp</location>
</error-page>
```

Para comprobar si lo hemos hecho de manera efectiva, solo debemos buscar una URL inexistente en la app, por ejemplo: ip:8080/lkhwnjeihn

Se debe mostrar el error personalizado.

Ahora configuramos el acceso por TLS en el puerto 8443

Primero debemos crear las claves con KeyTool y crear el connector en el puerto 8443:

```
Cd  /var/lib/tomcat9
keytool -genkey -alias tomcatexame -keyalg RSA -keystore ./deposito
```

(podemos almacenar cuantas claves queramos en un único depósito)

Pedirá contraseña dos veces e información irrelevante.

Ahora debemos crear el nuevo connector en server.xml (carpeta conf)

Y creamos el nuevo connector:

```
<connector port=”8443”
        protocol=”org.apache.coyote.httpll.HttpllNioProtocol”
        maxThreads=”150”
        SSLENabled=”true”
        Scheme =“https”
        Secure = “true”
        ClientAuth= “false”
        Sslprotocol = “TLS”
        keystoreFile = “val/lib/tomcat9/deposito”
        keystorePass= ”abc123.”
        keyalias = “tomcatexame”
        keypass “abc123.”
/>
```

Reiniciamos tomcat9

```
Systemctl restart tomcat9
```

Miramos logs en : cd /var/log/tomcat9/catalina.out (en caso de error)

Ejercicio 2: Despliegue con config de host

Pasos previos:

- Descargamos tomcat9 para Windows (lo guardamos en C)
- Se necesita realizar la descarga del JDK y la definición de la variable de entorno JAVA_HOME:
- Definimos la ruta de la carpeta descargada en config avanzada del sistema > variables de entorno
- Definimos de paso en el PATH la ruta de los binarios del JDK

Ya podemos utilizar tomcat a través de statup.bat.

Ahora, con una configuración correcta de tomcat, procedemos al despliegue de la aplicación solicitada:

- Creamos un usuario en tomcat-users.xml para acceder al manager: roles -> admin, manager-gui, admin-gui.

Ya podemos acceder al manager y desplegar la aplicación.war

Más adelante pedirán definir un nuevo rol : <role rolename=”appexame”/> que una vez definido podremos asignar a los usuarios.

- Definimos la versión de la aplicación : appnombre##numeroversión.war

Ejemplo: aplicación##2.23.12.war

Replegamos y desplegamos de nuevo la aplicación para comprobar que el manager refleja la versión.

Ahora trabajaremos con aspectos de la configuración de la aplicación, como la definición de un host virtual de acceso, el autoDeploy…etc

Necesitamos crear el host virtual en tomcat > conf > server.xml (por defecto solo está localhost):

```
<host name=”www.dawtomcat.gal” appBase=”dawtomcat_gal”
        unpackWARs=”true” autoDeploy=”true”>
        //Opcional <Alias>
        <Valve className=”org.apache.catalina.valves.AccessLogValve” directoy=”logs”
        prefix=”dawtomcat_exame_access_log” suffix=”.txt”
```

```
        pattern="%h %l %u %t &quot;%r&quot; %s %b" />
</host>
```

Ahora comprobamos el funcionamiento:

- Vemos si se crea el directorio dawtomcat_gal (al reiniciar el servidor) + fichero logs (valve)
- Enviamos la aplicación.war a dawtomcat_gal para que se despliegue ahí.

Ahora configuramos el archivo hosts con su dominio para poder acceder, el archivo se encuentra en: Windows>system32>drivers>etc>hosts

```
127.0.0.1 www.dawtomcat.gal
```

Accedemos a través de www.dawtomcat.gal:8080/proxecto2 (este despliegue no se realiza en ROOT context)

Permitir listar archivos: (a nivel servidor)

Configuramos el conf>web.xml

En la línea 124 debemos definir como "true" el param value de listings:->

```
 <param-name>listings</param-name>
<param-value>true</param-value>
```

//Si fuese para una app concreta, se haría en el web.xml de esa app

- Ahora queda configurar el acceso con autenticación:

Tres pasos: nuevo rol en tomcat-users.xml + web.xml + (en META-INF) context.xml

Vamos al descriptor de despliegue de la app: WEB-INF > web.xml

Debemos añadir lo siguiente:

```
<security-constraint>
        <web-resource-collection>
        <web-resource-name>MemoryRealm</web-resource-name>
        <url-pattern>/admin/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
        <role-name>appexame</role-name>
        </auth-constraint>
</security-constraint>
        <login-config>
```

```
        <auth-method>BASIC</auth-method>
        <realm-name>Acceso parte privada exame</realm-name>
        </login-config>
```

En context.xml:

```
<context>
        <Realm classname="org.apache.catalina.realm.MemoryRealm" />
</context>
```

Ahora copiamos el manager en la aplicación (desaconsejado) o creamos un xml en conf/catalina/dawtomcat_gal para poder acceder al manager existente:

Creamos: Manager.xml (asegurándonos de que el archivo tenga extensión .xml)

Editamos y añadimos:

```
<Context path="/manager" docBase="../webapps/manager" privileged="true" />
```

Ejercicio 3: Trabajo con dataSource (+ xampp)

Se trabaja con proyecto3.war descargado del aula:

Modificaremos : Web.xml + context.xml

- Primero desplegamos la aplicación con la que trabajaremos (manager o copia en webapps).

Comprobamos en localhost:8080/proxecto3 que está bien desplegado, ahí veremos que no tenemos conexión con la DB.

Para arreglar el problema necesitamos trabajar contra una base de datos, utilizamos para ello XAMPP e importamos en su phpmyadmin nuestro SQL después de crear la base de datos que queramos utilizar (a través del panel de administración).

- Ahora debemos configurar el acceso a la misma desde los documentos de configuración de la aplicación:

Ejecutamos el setup_xampp.bat para que se configuren las rutas.

Configuración del web.xml (de la aplicación):

```
<resource-ref>
        <res-ref-name>jdbc/publicación</res-ref-name>
        <res-type>java.sql.Datasource</res-type>
        <res-auth>Container</res-auth>
```

```
</resource-ref>
```

Configuración del context.xml :

```
<context>
        <resource name=”jdbc/publicacion” type=”javax.sql.DataSource”
        driverClassName=”com.mysql.jdbc.Driver”
url=”jdbc:mysql://localhost:3306/publicación?serverTimeZone=UTC”//definimos franja horaria
        username=”root” password=”” maxActive=”20” maxIdle=”10” maxWait=”10000” />
</context>
```