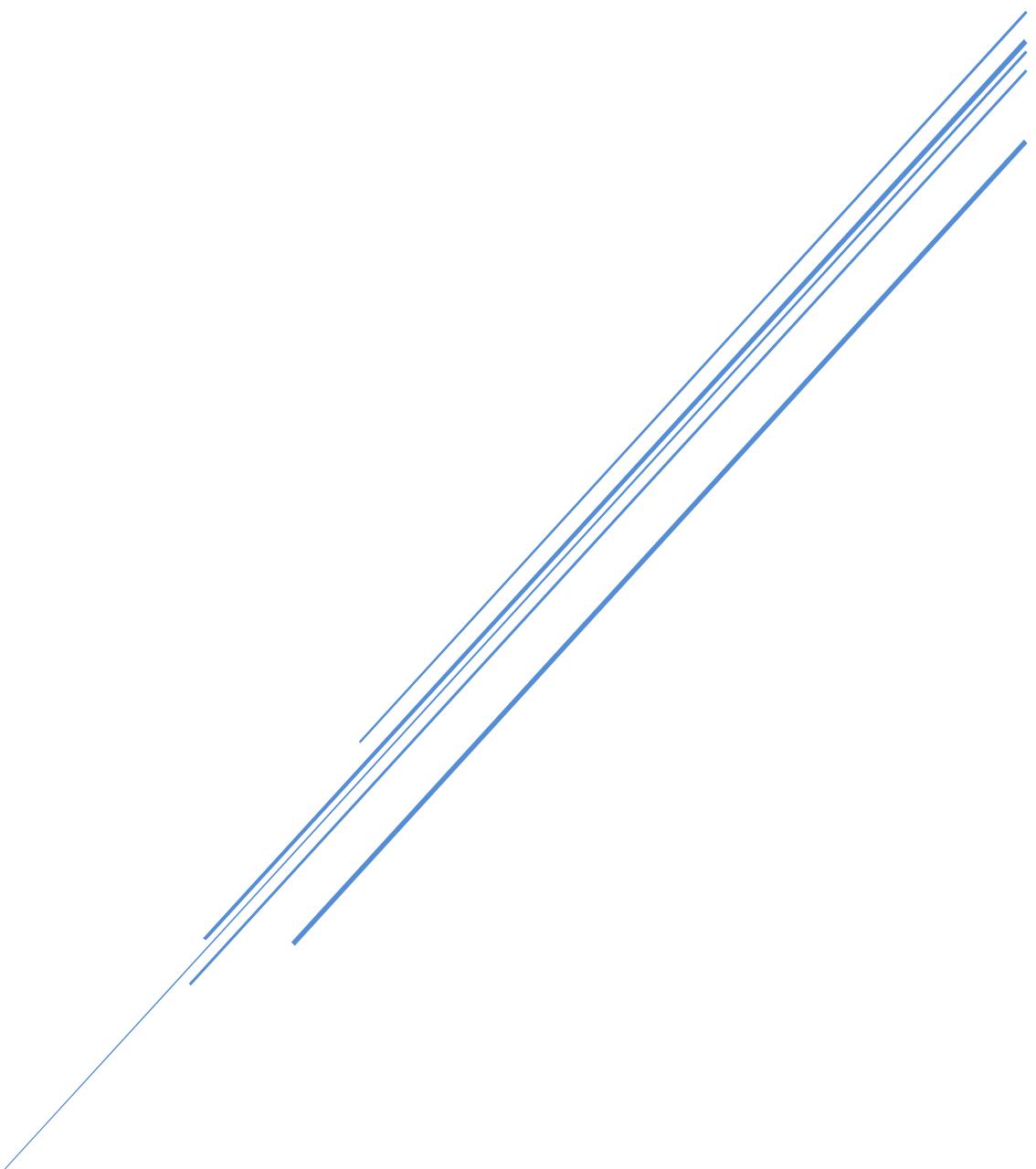


UD3. Administración avanzada de servidores de aplicaciones



Microsoft

Configuración de Tomcat

Táboa de contidos

1.	Introducción a Tomcat.....	4
1.1.	Arquitectura básica dos servidores de aplicáns	7
1.2.	Outros servidores de aplicación.....	7
1.3.	Estrutura de ficheiros dunha aplicación web baixo J2EE	8
1.4.	Descriptor de despregamento	9
1.5.	Aspectos de seguridade nos servidores de apliacións.....	10
1.6.	Resumo dos pasos de despregamento e cuestiós a ter en conta.....	11
2.	Instalación e configuración básica de Tomcat	12
2.1.	Instalación en Windows	13
2.2.	Configuración no Xampp.....	14
3.	Instalación de exemplos e documentos de axuda.....	15
4.	Ficheiros de configuración de Tomcat	16
5.	Tomcat Web Manager	18
6.	Despreglo manual dunha aplicación web	21
6.1.	Despregamento manual.....	21
6.2.	Despregamento automático.....	22
6.3.	Despreglo de aplicacións no ROOT context	22
6.4.	Despregamento dunha aplicación dende o manager.....	23
7.	Listado de cartafoles	26
8.	Autenticación e autorización (Memory Realm)	27
9.	Logs en Tomcat	29
9.2.	Logs específicos para unha aplicación web.....	30
10.	Hosts virtuais.....	31
11.	Tomcat host manager	34
12.	Integración de Apache e Tomcat empregando mod_proxy	36
13.	Seguridade SSL	38

14.	Tomcat no IDE Eclipse	41
14.1.	O arquivo web.xml	45
15.	Configurar o acceso a datasources	46

1. Introducción a Tomcat

Un **servidor de aplicaciones** é un paquete software que proporciona servicios ás aplicacións como poden ser seguridade, servizos de datos, soporte para transaccións, balanceo de carga e xestión de sistemas distribuidos.

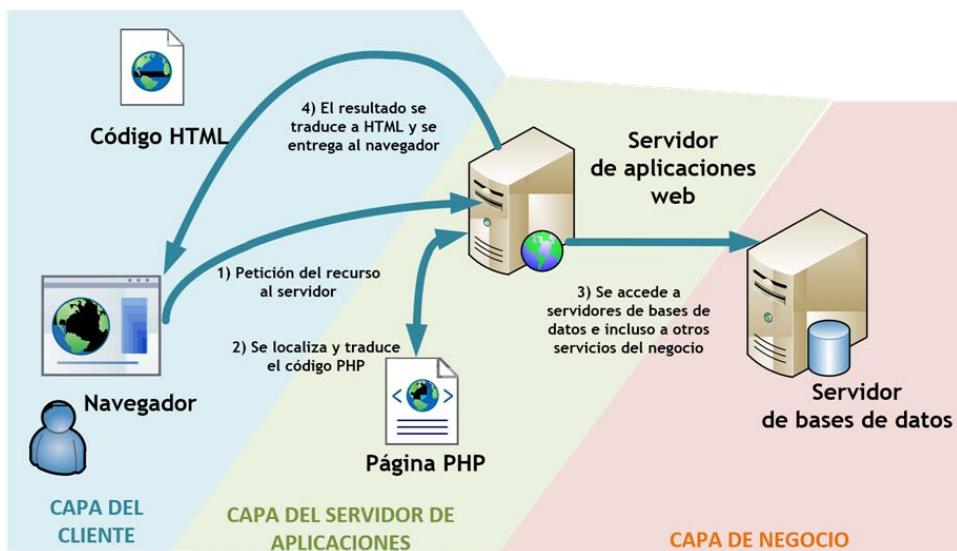
Este tipo de servidores non só serven para atender peticións http, senon que ademais son capaces de entender instruccións de linguaxes avanzados da web e traducirlas ou ben son capaces de acceder a recursos doutros servidores. Ese proceso faise de forma transparente ao usuario, é dicir o usuario pide o servicio a través, normalmente, do seu navegador e o servidor de aplicaciones atende a petición, e interpreta o código da aplicación a fin de traducirlo e mostrar ao usuario o resultado de forma entendible polo seu navegador.

Á forma de traballar dun servidor de aplicacións, coñécese normalmente como arquitectura de tres capas (a veces fálase de máis capas). Unha primeira capa é a do navegador que é capaz de traducir código do lado do cliente (HTML, JavaScript, CSS, Flash,...). Para elo esa capa debe dispoñer de todos os componentes necesarios para facer esa labor no ordenador do usuario.

A segunda capa está formada polo servidor de aplicacións na súa labor de traducir código no lado do servidor (JSP, PHP, Ruby on Rails, Cold Fussion...) e convertilo ó formato entendible polo navegador.

A terceira capa son todos os servicios aos que accede o servidor de aplicacións para poder realizar a tarefa encomendada á aplicación.

A imaxe amosa o funcionamento dun sevidor de aplicacións PHP:



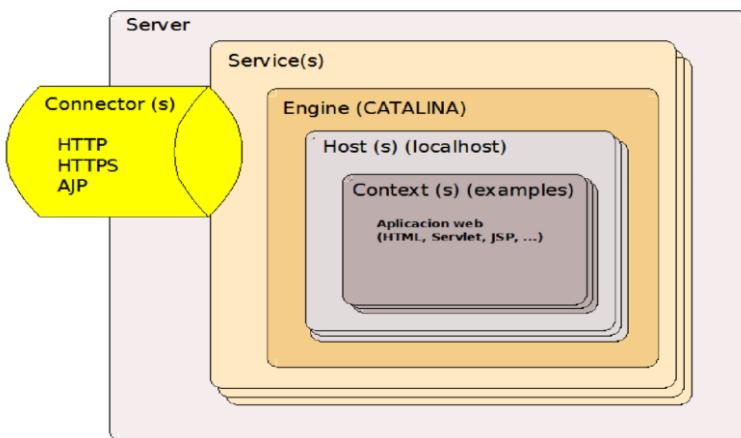
Apache Tomcat é un servidor de aplicacíons Java (OpenSource) desarrollado baixo o proxecto Jakarta na Apache Software Foundation para aloxar servlets e Java Server Pages(JSP).

Tomcat pode empregarse como servidor web independiente ou ben coma unha extensión do servidor web xa instalado (no noso caso empregaremos esta opción).

Entre as súas características destacan;

- É un servidor multiplataforma (Linux, Unix, Windows...)
- Estable e lixeiro
- Máis fácil de administrar econfigurar que “servidores de aplicacíons completos”

Tomcat está formado por varios **compoñentes** que se organizan de forma xerárquica:



- <Server> Pode conter un ou varios Services.
- <Service> Asocia un ou máis Connectors cun único Engine
- <Connector> É unha asociación cun porto IP para manexar as peticións e as respuestas cos clientes. Por defecto hay un conector HTTP creado
- <Engine> Recibe as peticións dos conectores e trasladaas ao Host correspondente.
- <Host> Define un servidor virtual, e pode conter unha ou máis aplicacións web (webapp).
- <Context> Cada webapp executándose dentro dun Host, está representada por un Context.

Por defecto existe un host configurado, chamado localhost. As aplicacións do host colócanse en `$CATALINA_BASE\webapps`

A xerarquía de directorios de Apache Tomcat é a seguinte:

- **bin:** contén os scripts de arranque, parada.
- **common:** clases comuns que poden empregar Catalina e as aplicacións web.
- **conf:** ficheros XML e a correspondente DTD para a configuración de Apache Tomcat.
- **logs:** logs do contenedor de servlets e das aplicacións.
- **server:** clases empregadas polo contenedor de servlets
- **shared:** clases compartidas por todas as aplicacións web.
- **webapps:** directorio que contén as aplicacións web.
- **work:** almacenamento temporal de ficheiros e directorios.

A partir da versión 4.x Tomcat lanzouse co contenedor de servlets "Catalina", co contenedor HTTP "Coyote" e cun motor para JSP chamado "Jasper".

As principais características destes tres compoñentes son:

Catalina

Implementa as especificacións de servlets e JSP. Para Apache Tomcat o elemento principal é unha base de datos de nomes de usuarios, password e roles. Catalina integrase onde xa existe información de autenticación como describe a especificación de servlets.

Coyote

Compoñente conector que admite o protocolo HTTP 1.1 para o servidor web e que escoita nun porto TCP especificado polo servidor e envía a solicitude ó motor Tomcat para que éste a procese e envie unha resposta o cliente.

Jasper

Analiza arquivos JSP para compilar o código Java e, si se producen cambios, éste volveos a compilar.

1.1. Arquitectura básica dos servidores de aplicacións

A arquitectura que se adoita empregar neste tipo de aplicacións é un **patrón software que separa a lóxica de negocio e os datos da parte representativa que observa o usuario**, os eventos e las comunicacións entre os distintos compoñentes.

Este modelo denominase **Modelo-Vista-Controlador** (MVC) e basease na separación de módulos para facilitar o seu posterior mantemento e a reutilización de código, faiéndoo ademáis máis sinxela a detección de errores en caso de fallos; porén é importante **separar a representación dos datos (Vista) da parte interna ou Modelo de datos**, coma por exemplo a base de datos.

Os tres grandes compoñentes nos que se divide o modelo son:

1. **Modelo**: encárgase de representar a información coa que a aplicación traballa; realiza as consultas e modificacións con base aos privilexios definidos previamente na análise de requisitos da aplicación. A petición chega por parte do controlador e este compoñente executa a acción e preséntalla á Vista.
2. **Vista**: visualiza o modelo cun tipo de representación para interactuar co usuario. Normalmente é a interfaz da aplicación que pode ser unha aplicación web, aplicación de escritorio ou calquera aplicación en calquiera entorno ou sistema operativo.
3. **Controlador**: é o módulo máis importante que controla os outros dous compoñentes, servindo de mediador entre o Modelo e a Vista. Responde a peticionadas polo usuario (pulsar un botón da interfaz ou calquera outra acción) comunicando co Modelo para manipulalo facendo cambios na Vista.

1.2. Outros servidores de aplicación

Un servidor de aplicacións é un software global que permite dar servizo a as aplicacións que se publican no mesmo dando soporte para a seguridade, as transaccións, a administración de sesiones, o rexistro de logs, etc.

Existe un abanico de servidores de aplicacións que son usados:

- [WildFly](#) (JBoss Application Server): é un software implementado por JBoss e desenvolto en Java. É software libre disponible para todos os sistemas operativos do mercado. A estructura interna conta con dous núcleos principais:
 - JBoss Modules: controla a carga dos recursos da clase contenedora.
 - Modular Service Container: administra os servizos usados polo contedor. Por exemplo, instala e desinstala os diferentes servizos.
- [GlassFish](#): software de código aberto desenvolto por Sun Microsystems para a plataforma Java EE. Proporciona un proxecto estruturado de desenvolvemento que permite ter disponibles as npvas funcionalidades de Java.
- [Apache Tomcat](#): servidor de aplicacións libre implementando Java Servlet, JavaServer Pages, Java Expression Language e Java WebSocket. Estes módulos son desenvoltos polo Java Community Process. Oracle certifica coa versión de Java correspondente.
- [Oracle WebLogic Server](#) é unha plataforma unificada e extensible para desenvolver, implementar e executar aplicacións empresariais (coma aplicacións Java) *on-premises* e

na nube. Ofrece unha implementación sólida, madura e escalable de Java Enterprise Edition (EE) e Jakarta EE.

- [Jetty](#) é un servidor HTTP e un contenedor de Servlets escrito en Java. Públícase como un proxecto de software libre baixo a licenza Apache 2.0. É utilizado por outros proyectos (por exemplo os servidores de aplicacións JBoss e Apache Geronimo; e polo *plugin* Google Web Toolkit para Eclipse. Jetty enfócase en crear un servidor web sinxelo, eficiente, empotrábel e *pluggable*. O tamaño tan pequeño de Jetty faiño axeitado para ofrecer servicios Web nunha aplicación Java empotrada.
- [IBM WebSphere Application Server](#) é un entorno de execución de servidor Java flexible e altamente seguro para aplicacións empresariais
- [Apache TomEE](#) é a Java Enterprise Edition de Apache Tomcat (Tomcat + Jakarta EE = TomEE) que combina outros proxectos coma Apache [OpenEJB](#), Apache [OpenWebBeans](#), Apache [OpenJPA](#), Apache [MyFaces](#) e outros]
- [Apache Geronimo](#) é un servidor de aplicacións de código abierto desenvolto pola Apache Software Foundation e distribuído baixo a licencia Apache]

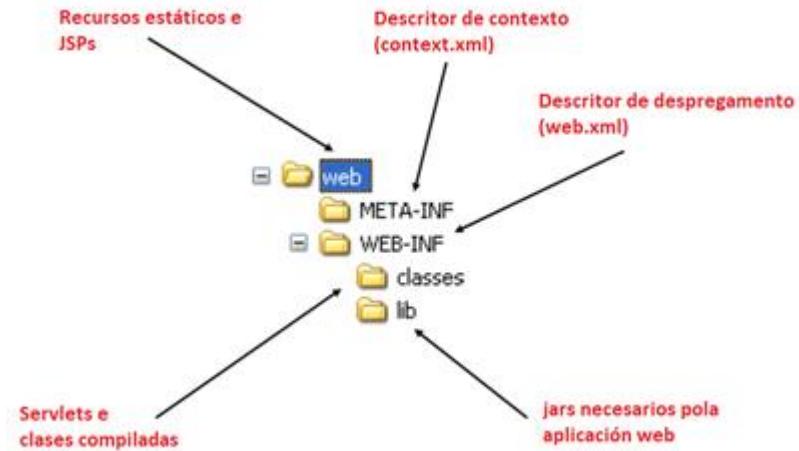
Nalgúns casos os produtos son resultado da integración de varias tecnoloxías e pode resultar confusa a súa distinción.

Por exemplo, entre Tomcat, TomEE (e as súas variantes) e TomEE+. Podes atopar un [resumo explicatorio](#) pero é preciso coñecer moitas das tecnoloxías para acadar unha boa comprensión de cales son as súas semallanzas, diferenzas, ou cal é máis axeitado para un proxecto.

1.3. Estrutura de ficheiros dunha aplicación web baixo J2EE

Unha aplicación web JEE debe ter unha estrutura de ficheiros e directorios determinada:

- /: Na carpeta raíz do proyecto almacénanse elementos empregados nos sitios web coma documentos html, CSS e os elementos JSP (*.html *.jsp *.css). Podense subdividir tamén en directorios.
- /META-INF/: é un directorio privado da aplicación que contén habitualmente un ficheiro denominado context.xml que permite definir o contexto da aplicación.
- /WEB-INF/: Atópanse os elementos de configuración do arquivo .WAR coma poden ser: a páxina de inicio, a ubicación dos servlets, parámetros adicionais para outros compoñentes. Como veremos máis adiante o módulo relevante é o ficheiro web.xml ou descriptor de despregue da aplicación.
- /WEB-INF/classes/: Contén as clases Java empregadas no ficheiro .WAR; normalmente nesta carpeta atópanse os servlets.
- /WEB-INF/lib/: Contén os ficheiros JAR utilizados pola aplicación: son librerías, entre outras as empregadas para conectarse coa base de datos ou as empregadas polas librerías de JSP.
- O resto de carpetas son para os elementos estáticos da aplicación (imáxenes, etc) que podemos estruturar o noso antoxo.



1.4. Descriptor de despregamento

Un **descriptor de despregue** (*Deployment Descriptor* ou *DD*) é un compoñente de aplicacións J2EE que **describe cómo se debe despregar** (ou implantar) **unha aplicación web**. Esto dirixe a unha ferramenta de despregue (ou publicación) para despregar un módulo ou aplicación con opcións de contedor e requisitos de configuración específicos.

XML úsase para a sintaxe do ficheiro descriptor de despregue en aplicacións J2EE. Chámase *web.xml* e debe ser colocado nun subdirectorio chamado *WEB-INF*, directamente debaixo da raíz da aplicación web.

AMPLIACIÓN.

Podes atopar información do [descriptor de despregue para o servidor de aplicacións IBM Web Application Server](#)

Pódese modificar:

- manualmente editando o ficheiro
- utilizando a consola da ferramenta
- usando un editor de descriptor de despregue da ferramenta de ensamblaxe

As dúas últimas son más recomendables porque permiten asegurarnos de que o descriptor ten as propiedades válidas e de que as referencias conteñen os valores correctos.

Exemplo sinxelo de descriptor de despregue

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <servlet>

```

```
<servlet-name>HolaMundoServlet</servlet-name>
<servlet-class>org.prueba.servlet.HolaMundoServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>HolaMundoServlet</servlet-name>
    <url-pattern>/HolaMundoServlet</url-pattern>
</servlet-mapping>
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
</web-app>
```

1.5. Aspectos de seguridade nos servidores de aplicacións

Débese protexer á aplicación web e o servidor de aplicacións de accesos malintencionados, de que non interceptan a información.

Seguridade e autenticación

Para protexer ante as anteriores ameazas un sistema de seguridade basease en tres conceptos clave:

1. **Autenticación** : proceso para identificar quén entra na aplicación é quén di ser, y validar que pode acceder ó recurso requerido.
2. **Confidencialidade** : os extremos da comunicación deben coñecer a información que se intercambia sen que poida ser accedita por terceiros.
3. **Integridade** : a información que se transmite de extremo a extremo non é alterada por axentes externos.

É importante que o servidor de aplicacións controle de xeito transparente ó usuario as comunicacións entre os distintos elementos que intercambian información no fluxo da aplicación. O ficheiro web.xml encárgase desta función coas marcas que xa se explicaron.

Pódese tamén controlar a seguridade mediante a programación dos servlets e ficheiros .jsp que permitan a seguridade da aplicación.

Autenticación

Existen distintos tipos de autenticación que se poden implementar nunha aplicación web:

- **Autenticación basic**: basease en solicitar datos o usuario, como un nome de usuario e un contrasinal. Esta información non vai codificada polo que é perigoso usar este tipo de autenticación.
- **Autenticación digest**: variante da *basic* onde o contrasinal viaxa pola rede encriptada mediante unha función hash. Non todos os servidores soportan este tipo de autentificación.
- **Autenticación baseada en formularios**: basease en solicitarlle datos ó usuario mediante un formulario no que introduzca o usuario e contrasinal. Este mecanismo é débil de cara a los hackers xa que pódese obter esta información de forma fácil.

- **Certificados dixitais e SSL:** o ideal é usar o protocolo HTTPS que funciona no porto 443 permitindo garantir a confidencialidade e integridade da información, e asegurando a autenticación. O funcionamento basease na criptografía de clave pública.
- Configurar o servidor de aplicacións con soporte SSL/TLS

1.6. Resumo dos pasos de despregamento e cuestións a ter en conta

- a arquitectura de modelo-vista-controlador é básica en cualquier aplicación del mercado, xa que permite separar ao programador as partes ben diferenciadas das que consta una aplicación.
 - o modelo fai de ponte entre o controlador y la vista e encárgase de realizar as peticións e modificacións en base aos privilexios.
 - a vista é a parte que observa o usuario e que posee unha capa de abstracción que permite que se oculte toda a lóxica que existe detrás.
 - o controlador é o cerebro da aplicación e permite que todo funcione correctamente.
- Existen no mercado moitos servidores de aplicacións
 - Apache Tomcat é un dos más populares
 - a instalación e posta en funcionamiento basease no uso do JDK para compilar as clases de Java, na instalación do seu software e na configuración das variables de entorno.
 - para poder administrar as aplicacións no servidor de aplicaciones permite configurar usuarios, contrasinais e roles no ficheiro de configuración tomcat-users.xml
- Na administración de aplicacións é fundamental
 - coñecer cómo se despliega unha aplicación a partir dun ficheiro .war ou de forma manual.
 - a estrutura fixa de directorios que debe poseer unha aplicación para poder desplegarse.
- A autenticación de usuarios realiza en Tomcat a partir dos dominios de seguridade (*Realm*)
 - engloba o conxunto de usuarios, contrasinais e roles que se van definir nos correspondentes ficheiros .xml.
- Para a administración e configuración das sesións das aplicacións é necesario coñecer cómo se configura unha sesión persistente.
 - Podemos axudar examinar á configuración dos rexistros de acceso e os filtros de solicitudes.
- La instalación e configuración en sistemas Windows é parecida á de Linux
 - as diferencias estriban na estrutura de directorios e algúns ficheiros coma os de execución
- Unha vez programada unha aplicación web debemos documentar cómo se desprega indicando a súa estructura e a configuración dos ficheiros clave no despreglo:
 - web.xml
 - context.xml
 - o ficheiro de inicio index.jsp, index.html
 - calquera outro que sexa compatible co servidor de aplicacións.
- Débese securizar ó servidor de aplicacións mediante SSL para impedir accesos malintencionados
 - para iso é necesario modificar o ficheiro server.xml del servidor.

2. Instalación e configuración básica de Tomcat

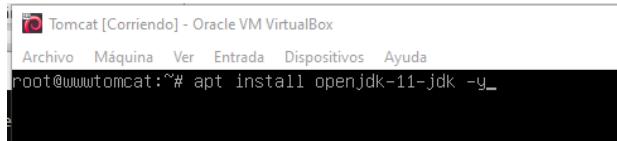
A pesar de que na unidade 1 fixemos unha instalación de Tomcat manual/automática imos poder repetir os pasos da instalación para unha contorna Debian.

Como sempre facemos unha actualización da paquetería

```
apt-get update
```

Logo efectuamos a instalación da versión actual do JDK

```
apt-get install openjdk-11-jdk -y
```



Se xa tiveramos instalado o jdk, podemos actualizar a paquetería existente ás versións máis actuáis mediante a execución do comando

```
apt-get dist-upgrade
```

O anterior comando tentará actualizar as versións das aplicacións do sistema e mailas dependencias existentes. Nunha contorna de producción a execución deste comando debe tomarse con certa cautela, xa que habería que examinar se os aplicativos actuáis funcionarán coas novas versións. No noso caso en principio, non habería problemas.

Agora xa podemos instalar o noso tomcat versión 10, se xa non o está.

```
apt install tomcat9 -y
```

Engadimos un novo grupo para facilitar as tarefas de administración

```
groupadd tomcat9
```

Creamos un usuario chamado tomcat9

```
useradd -s /bin/false -g tomcat9 -d /etc/tomcat9 tomcat9
```

Unha vez instalado, e co obxectivo de deixar a contorna lista para o **desenvolvemento** e servizo produtivo de aplicacións, imos configurar as variables de contorna más importantes.

As variables de contorna a configurar son:

- JAVA_HOME
- CATALINA_HOME
- CATALINA_BASE

Estas variables son útiles para indicarlle a certas aplicacións cal é o cartafol que contén a instalación da máquina virtual de Java, o tomcat, ...

Imos darrle valor mediante a edición do arquivo **/etc/environment**

```

GNU nano 5.4                               /etc/environment
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
CATALINA_HOME=/usr/share/tomcat9
CATALINA_BASE=/var/lib/tomcat9

```

Podes comprobar que as rutas que se marcan nas variables en cuestión **existen**.

Para que as variables de sesión tomen valor é aconsellable reiniciar a sesión de usuario.

Lembrar que podemos iniciar e para o servizo de tomcat mediante

```

systemctl start tomcat9
systemctl stop tomcat9

```

Comprobamos tamén que o noso tomcat está instalado e servindo no porto 8080.

```

ss -ltn

```

Vemos o valor das nosas variables de sesión.

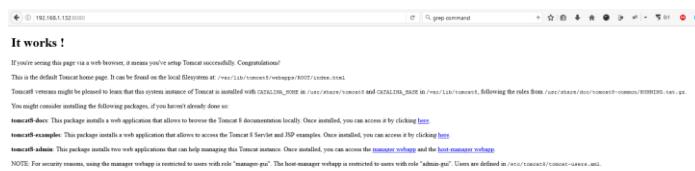
```

env | egrep "_HOME|CATA"

```

Consulta da páxina web de exemplo.

Agora tan só resta probar a páxina de inicio de Tomcat para verificar o seu funcionamiento. Para iso escribimos a ip do servidor seguido do porto 8080.



Debemos ter presente que esta páxina web que se nos amosa está ubicada en **/var/lib/tomcat9/webapps/ROOT** e o seu nome é **index.html**.

2.1. Instalación en Windows

Descargamos os binarios dende <https://tomcat.apache.org/download-10.cgi> escollendo a versión que precisemos. Debemos ter en consideración que dende a versión 10 Tomcat non soporta **javax.Servlet** substituíndose por **jakarta.Servlet**, o que implica a actualización das aplicacións a despregar.

Debemos descomprimir o arquivo en cuestión, por exemplo en c:\apache-tomcat-10

O script de inicio temólo en c:\apache-tomcat-10\bin\startup.bat

O script de parada témolo en c:\apache-tomcat-10\bin\shutdown.bat

É imprescindible dispoñer da variable de contorno **JAVA_HOME** apuntando a unha instalación de jdk.

Variables del sistema	
Variable	Valor
java_home	C:\java\jdk-11.0.12

2.2. Configuración no Xampp

O propio xampp dispón dun servidor tomcat incorporado no cartafol **xampp/tomcat**. Funcionaría de forma similar ao Tomcat do punto anterior, pero para que poida ser iniciado dende o Panel de Control de Xampp directamente debemos tomar conta de que precisa dunha entrada no rexistro de Windows para funcionar. O Xampp chama ao script **xampp/catalina_start.bat** para á súa vez invocar o startup.bat de tomcat.

No caso de efectuar unha instalación manual do JDK debemos ingresar de forma manual tamén a clave de rexistro:

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Development Kit insertando a variable **JavaHome**

Outra opción é crear un arquivo de entrada de rexistro de Windows **.reg** co seguinte contido, e posteriormente executalo.

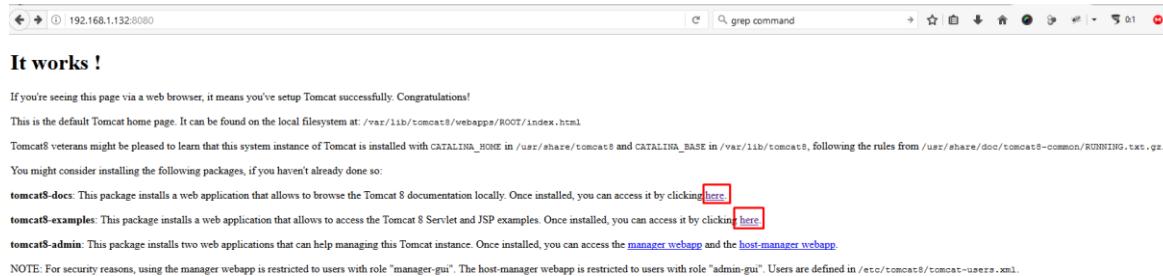
```
[HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Development Kit]
"JavaHome"="C:\\java\\jdk-17.0.4.1"
```

3. Instalación de exemplos e documentos de axuda.

Para poder facer máis probaturas imos instalar os exemplos e tamén os documentos de axuda do Tomcat. Para isto executamos o comando

```
apt install tomcat9-docs tomcat9-examples
```

Unha vez instalados poderemos acceder aos exemplos e documentación, ou ben dende a páxina principal de Tomcat no noso servidor



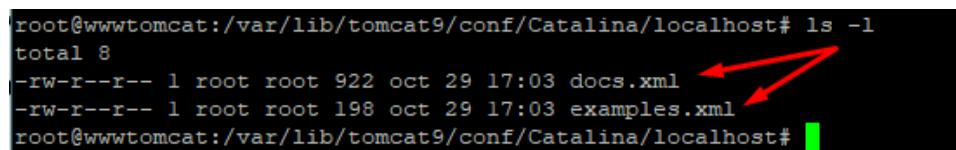
Ou ben indicando na barra de localización do noso navegador a URL http://<<ip_servidor>>:8080/docs e http://<<ip_servidor>>:8080/examples

Proba os programas de HolaMundo, JSP de aritmética, ...

Onde están os ficheiros/scripts destas aplicacións web?

Consulta o cartafol \$CATALINA_BASE/conf/Catalina/localhost

```
root@wwwtomcat:/var/lib/tomcat9/conf/Catalina/localhost# ls -l
total 8
-rw-r--r-- 1 root root 922 oct 29 17:03 docs.xml
-rw-r--r-- 1 root root 198 oct 29 17:03 examples.xml
root@wwwtomcat:/var/lib/tomcat9/conf/Catalina/localhost#
```



Se consultas o contido de cada un dos ficheiros xml que se detallan verás que, por exemplo: **examples.xml** contén a seguinte información:

```
GNU nano 5.4                               examples.xml
<Context path="/examples"
          docBase="/usr/share/tomcat9-examples/examples">
    <!-- Enable symlinks for the jars linked from /usr/share/java -->
    <Resources allowLinking="true"/>
</Context>
```

facendo alusión a que os ficheiros de exemplo que se servirán en /examples (http://<<ip_servidor>>/examples) están almacenados en /usr/share/tomcat9-examples/examples.

4. Ficheiros de configuración de Tomcat

A instalación de Tomcat deixounos a seguinte estrutura de cartafoles en CATALINA_HOME:

- **bin**: binarios cos scripts de inicio/parada/reinicio de Tomcat.
- **lib**: contén os JAR dispoñibles para todas as aplicacións. É habitual incluír aquí drivers JDBC e outros elementos de interacción con entidades externas.

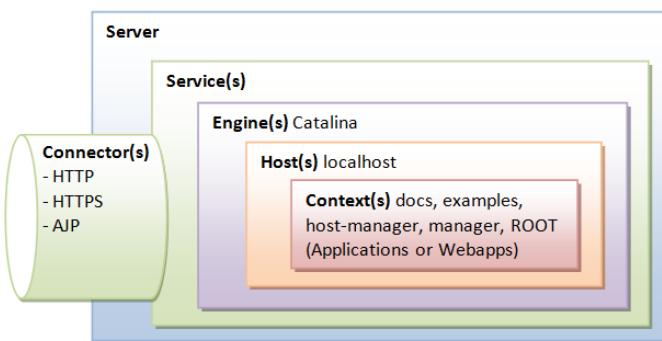
A instalación tamén nos dexa a seguinte estrutura de cartafoles en CATALINA_BASE:

- **conf**: configuración global para todas as aplicacións. A instalación por defecto prové:
 - un ficheiro de políticas: catalina.policy para especificar políticas de seguridade.
 - Dous ficheiros de propiedades: catalina.properties e logging.properties,
 - Catro ficheiros de configuración xml:
 - **server.xml** (ficheiro principal de configuración)
 - **web.xml** (descriptor de despregamento global de aplicacións)
 - **context.xml** (opción de configuración global de tomcat)
 - **tomcat-users.xml** (base de datos de usuarios, contrasinais e roles para autenticación e control de acceso).

Este cartafol tamén contén un subcartafol por cada motor, p.e., Catalina, que á súa vez contén un sub-sub-cartafol por cada un dos seus hosts, p.e: localhost. Poderemos situar información de contextos específicos similares a context.xml para cada aplicación web.

- **logs**: conteñen ficheiros de log do motor Catalina, do host, manager, host-manager,
- **webapps**: é o cartafol base por defecto para as aplicacións web de localhost.
- **work**: conteñen as clases JSP/JSF, e os servlets traducidos de Java e que se empregaron nalgún momento polo servidor Tomcat.

A seguinte imaxe reflicte a arquitectura de Tomcat 8.



Tomcat é un servidor HTTP, aínda que tamén é un contenedor de servlets que pode executalos e tamén é quen de converter JSP (Java Server Pages) e JSF (Java Server Faces) as servlets de Java. Como reflicte a antedita arquitectura emprega unha arquitectura modular e xerárquica.

O principal ficheiro de configuración é **server.xml**, que se atopa en <CATALINA_BASE>/conf.

Debemos fixarnos que o cartafol <CATALINA_BASE>/conf é unha ligazón simbólica a /etc/tomcat9.

Imos analizar un chisco o ficheiro **server.xml**, xunto coas declaracíons de certo interese.

```
<?xml version='1.0' encoding='utf-8'?>
<Server port="8005" shutdown="SHUTDOWN">
    <Listener className="org.apache.catalina.startup.VersionLoggerListener" />
    <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
    <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
    <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

    <GlobalNamingResources>
        <Resource name="UserDatabase" auth="Container"
            type="org.apache.catalina.UserDatabase"
            description="User database that can be updated and saved"
            factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
            pathname="conf/tomcat-users.xml" />
    </GlobalNamingResources>
    <Service name="Catalina">
        <Connector port="8080" protocol="HTTP/1.1"
            connectionTimeout="20000"
            URIEncoding="UTF-8"
            redirectPort="8443" />
        <Engine name="Catalina" defaultHost="localhost">
            <Realm className="org.apache.catalina.realm.LockOutRealm">
                <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
                    resourceName="UserDatabase"/>
            </Realm>
            <Host name="localhost" appBase="webapps"
                unpackWARs="true" autoDeploy="true">
                <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
                    prefix="localhost_access_log" suffix=".txt"
                    pattern="%h %l %u %t \"%r\" %s %b" />
            </Host>
        </Engine>
    </Service>
</Server>
```

- **Server** representa unha instancia de Tomcat (poderiamos ter varios tomcat no mesmo servidor)
- **Listener** é un elemento que escoita e resposto a determinados eventos
- **Service** asocia un ou varios conectores cunha máquina (motor). A configuración por defecto define un servizo denominado Catalina cun conector HTTP.
- **Connector** asocia un porto TCP para manipular as conexión entre cliente e servidor. Por defecto vemos que é o 8080.
- **Engine** (motor) é unha agrupación de hosts que comparten configuracións comúns.
- **Host** é coma un virtual host de apache.
- **Valve** permite interceptar peticións HTTP e facerlle un preproceso antes de reenviarllas ás aplicacións do noso Tomcat. Unha valve pode ser definida a nivel de host, engine e context.

Debemos tomar en consideración que os aspectos de configuración de Tomcat a nivel sistema imos efectualos en /etc/tomcat9.

A nivel runtime os binarios de tomcat9 están ubicados en /usr/share/tomcat9 (CATALINA_HOME) e tamén as aplicacións e exemplos atópanse situados en /usr/share/tomcat9-xxxxxx.

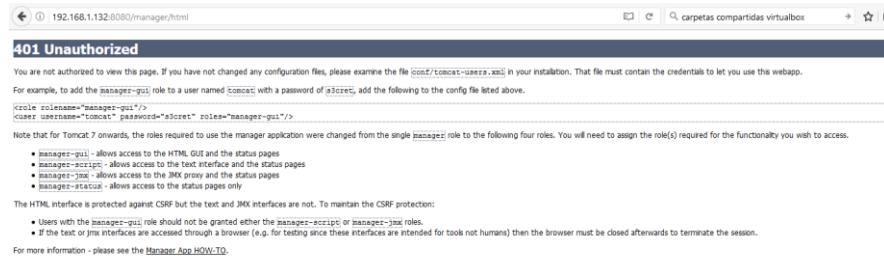
5. Tomcat Web Manager

Instalamos o Tomcat manager, que nos permitirá configurar o noso servidor dun xeito amigable e a través dunha interface web.

```
sudo apt update
sudo apt install tomcat9-admin
```

En */var/lib/tomcat9/conf/Catalina/localhost* debeu crearse o ficheiro **manager.xml**.

Tenta acceder dende o navegador web a http://<>ip_servidor>:8080/manager. Pedirache autenticación, e ao non certar amosarache a seguinte páxina con instrucións sobre como salvar o problema.



Para configurar un acceso autorizado debemos editar o ficheiro */var/lib/tomcat9/conf/tomcat-users.xml* para engadir usuarios e roles. Aquí só engadimos un usuario. Unha vez feito reiniciamos o tomcat.

```
GNU nano 5.4                               tomcat-users.xml
<role rolename="admin"/>
<role rolename="admin-gui"/>
<role rolename="manager"/>
<role rolename="manager-gui"/>
<user username="mosquera" password="abc123." roles="admin, admin-gui, manager, manager-gui"/>
<!-- <user username="both" password=<must-be-changed> roles="tomcat,role1"/>
<user username="role1" password=<must-be-changed> roles="role1"/>-->

</tomcat-users>
```

Probamos a acceder agora.



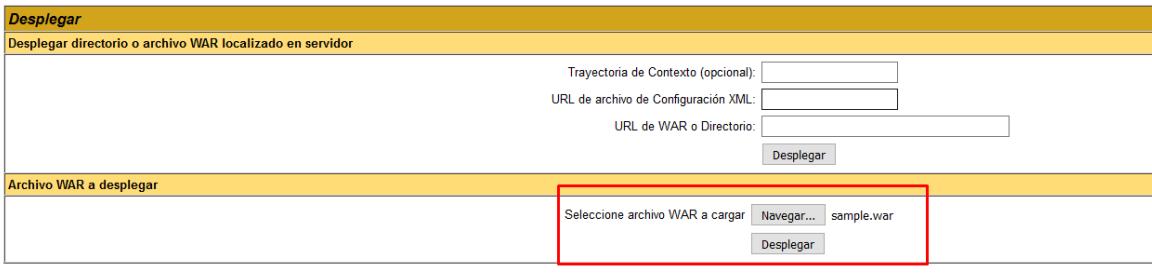
The screenshot shows the Apache Tomcat Manager interface. At the top, there's a header with the Apache Software Foundation logo and a search bar. Below that is a navigation bar with tabs: 'Listar Aplicaciones' (selected), 'Ayuda HTML de Gestor', 'Ayuda de Gestor', and 'Estado de Servidor'. The main content area is titled 'Gestor de Aplicaciones Web de Tomcat' and contains a table titled 'Aplicaciones'. The table has columns for 'Trayectoria', 'Versión', 'Nombre a Mostrar', 'Ejecutándose', 'Sesiones', and 'Comandos'. It lists several applications: 'docs' (Tomcat Documentation), 'examples' (Servlet and JSP Examples), 'host-manager' (Tomcat Host Manager Application), and 'manager' (Tomcat Manager Application). Each application row includes buttons for 'Arrancar', 'Parar', 'Recargar', 'Replegar', and session expiration settings.

Con este administrador podemos realizar tarefas diversas como:

- Listar as aplicacións despregadas
- Iniciar /parar aplicacións
- Desplegar aplicacións
- Listar as sesións activas e consultar estatísticas
- Consultar o estado do servidor
- ...

Proba por exemplo a parar algunha das aplicacións configuradas e comproba que non podes acceder a elas.

Agora imos desplegar unha aplicación web de exemplo, descargada de Tomcat e denominada *sample.war*. Selecciono o arquivo, e logo premo en Desplegar. Aquí hai que ter en conta o tamaño do arquivo a subir no servidor. Normalmente hai un límite de 10 MB/200MB.



The screenshot shows the 'Desplegar' (Deploy) page. It has two main sections: 'Desplegar directorio o archivo WAR localizado en servidor' (Deploy directory or local WAR file) and 'Archivo WAR a desplegar' (WAR file to deploy). In the second section, there's an input field labeled 'Seleccione archivo WAR a cargar' (Select WAR file to load) with the value 'sample.war' and a 'Desplegar' (Deploy) button. A red box highlights the 'sample.war' entry in the input field.

Unha vez despregado xa o podemos ver na nosa lista de aplicacións:

Aplicaciones						
Trayectoria	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos	
/	Ninguno especificado	Mosquera APP	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
/examples	Ninguno especificado	Servlet and JSP Examples	true	0	Arrancar	Parar
					Recargar	Replegar
					Expirar sesiones	sin trabajar ≥ 30 minutos
/host-manager	Ninguno especificado	Tomcat Host Manager Application	true	0	Arrancar	Parar
					Recargar	Replegar
					Expirar sesiones	sin trabajar ≥ 30 minutos
/manager	Ninguno especificado	Tomcat Manager Application	true	1	Arrancar	Parar
					Recargar	Replegar
					Expirar sesiones	sin trabajar ≥ 30 minutos
/sample	Ninguno especificado	Hello, World Application	true	0	Arrancar	Parar
					Recargar	Replegar
					Expirar sesiones	sin trabajar ≥ 30 minutos

Desplegar

Na seguinte captura obtemos o estado global do servidor, coas versións de Tomcat JVM, estado de memoria, listado de aplicacións, ...

The Apache Software Foundation


Complete Server Status

Manager		HTML Manager Help		Manager Help		Server Status																																					
List Applications																																											
Server Information																																											
Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address																																				
Apache Tomcat/8.0.32 (Ubuntu)	1.8.0_151-b15-b12-ubuntu0.16.04.2-b12	Oracle Corporation	Linux	4.4.0-98-generic	i386	uadmin	127.0.1.1																																				
JVM																																											
Free memory: 15.51 MB Total memory: 39.56 MB Max memory: 123.75 MB																																											
<table border="1"> <thead> <tr> <th>Memory Pool</th> <th>Type</th> <th>Initial</th> <th>Total</th> <th>Maximum</th> <th>Used</th> </tr> </thead> <tbody> <tr> <td>CMS Old Gen</td> <td>Heap memory</td> <td>21.37 MB</td> <td>30.00 MB</td> <td>85.37 MB</td> <td>15.42 MB (18%)</td> </tr> <tr> <td>Par Eden Space</td> <td>Heap memory</td> <td>8.50 MB</td> <td>8.50 MB</td> <td>34.12 MB</td> <td>7.55 MB (22%)</td> </tr> <tr> <td>Par Survivor Space</td> <td>Heap memory</td> <td>1.06 MB</td> <td>1.06 MB</td> <td>4.25 MB</td> <td>1.06 MB (25%)</td> </tr> <tr> <td>Code Cache</td> <td>Non-heap memory</td> <td>2.25 MB</td> <td>6.78 MB</td> <td>240.00 MB</td> <td>6.71 MB (2%)</td> </tr> <tr> <td>Metaspace</td> <td>Non-heap memory</td> <td>0.00 MB</td> <td>12.39 MB</td> <td>-0.00 MB</td> <td>12.01 MB</td> </tr> </tbody> </table>								Memory Pool	Type	Initial	Total	Maximum	Used	CMS Old Gen	Heap memory	21.37 MB	30.00 MB	85.37 MB	15.42 MB (18%)	Par Eden Space	Heap memory	8.50 MB	8.50 MB	34.12 MB	7.55 MB (22%)	Par Survivor Space	Heap memory	1.06 MB	1.06 MB	4.25 MB	1.06 MB (25%)	Code Cache	Non-heap memory	2.25 MB	6.78 MB	240.00 MB	6.71 MB (2%)	Metaspace	Non-heap memory	0.00 MB	12.39 MB	-0.00 MB	12.01 MB
Memory Pool	Type	Initial	Total	Maximum	Used																																						
CMS Old Gen	Heap memory	21.37 MB	30.00 MB	85.37 MB	15.42 MB (18%)																																						
Par Eden Space	Heap memory	8.50 MB	8.50 MB	34.12 MB	7.55 MB (22%)																																						
Par Survivor Space	Heap memory	1.06 MB	1.06 MB	4.25 MB	1.06 MB (25%)																																						
Code Cache	Non-heap memory	2.25 MB	6.78 MB	240.00 MB	6.71 MB (2%)																																						
Metaspace	Non-heap memory	0.00 MB	12.39 MB	-0.00 MB	12.01 MB																																						
"http-nio-8080"																																											
Max threads: 200 Current thread count: 10 Current thread busy: 1 Kept alive sockets count: 1 Max processing time: 371 ms Processing time: 3.926 s Request count: 35 Error count: 2 Bytes received: 0.00 MB Bytes sent: 0.39 MB																																											
Stage	Time	B Sent	B Recv	Client (Forwarded)	Client (Actual)	VHost	Request																																				
S	1 ms	0 KB	0 KB	192.168.1.128	192.168.1.128	192.168.1.132	GET /manager/status/all HTTP/1.1																																				
R	?	?	?	?	?	?																																					

P: Parse and prepare request S: Service F: Finishing R: Ready K: Keepalive

Application list

localhost/docs
localhost/
localhost/examples
localhost/host-manager
localhost/host-manager
localhost/manager

Tamén disponemos dun host manager que nos permite xestionar distintos servidores de aplicacións a modo de máquina virtual nun mesmo host físico e coa mesma instalación de Tomcat.

Esta utilidade é accesible dende a url http://<>ip_servidor>:8080/host-manager

6. Despreglo manual dunha aplicación web

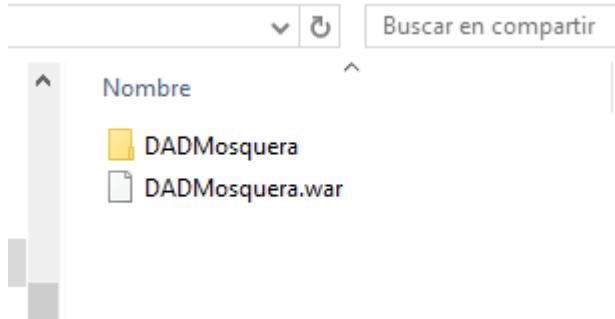
O despreglo manual dunha aplicación web feita con tecnoloxía Java consiste, no sentido máis básico, en situar o contido dun arquivo **WAR (web archive)** dentro do cartafol **webapps** descrito nos apartados anteriores.

6.1. Despreglo manual.

O primeiro que debemos ter en conta é que nos deben dar un ficheiro war. No noso caso **DADMosquera.war**.

En segundo lugar debemos trasladar este arquivo á computadora que contén o servidor Tomcat.

Volvendo ao despreglo manual, no noso Windows descomprimo manualmente (co 7zip ou Winrar) o arquivo **DADMosquera.war** no cartafol **DADMosquera**.



Copiamos o cartafol **DADMosquera** en **\$/CATALINA_BASE/webapps**.

```
uadmin@uadmin:/media/sf_compartir$ uadmin@uadmin:/media/sf_compartir$ sudo cp -R DADMosquera /var/lib/tomcat8/webapps/
uadmin@uadmin:/media/sf_compartir$
```

Agora debemos cambiar o dono e grupo do cartafol en cuestión para evitar problemas de permisos.

```
uadmin@uadmin:/var/lib/tomcat8/webapps$ uadmin@uadmin:/var/lib/tomcat8/webapps$ sudo chown tomcat8:tomcat8 -R DADMosquera
uadmin@uadmin:/var/lib/tomcat8/webapps$
```

Agora accedemos á nosa aplicación dende o navegador. Respectade as maiúsculas e minúsculas.



6.2. Despregamento automático.

Antes que nada eliminamos o cartafol en webapps feito no paso anterior.

```
rm -R /var/lib/tomcat9/webapps/DADMosquera
```

Copiamos o arquivo *DADMosquera.war* directamente a *DADMosquera* en *\$CATALINA_BASE/webapps*.

```
cp /media/sf_compartir/DADMosquera.war /var/lib/tomcat9/webapps
```

Cambiamos o dono e grupo do arquivo.

```
chown tomcat9:tomcat9 /var/lib/tomcat9/webapps/DADMosquera.war
```

```
admin@uadmin:~$ cp /media/sf_compartir/DADMosquera.war /var/lib/tomcat8/webapps/
admin@uadmin:~$ sudo chown tomcat8:tomcat8 /var/lib/tomcat8/webapps/DADMosquera.war
admin@uadmin:~$
```

Feito esto reiniciamos o servidor tomcat.

```
service tomcat9 restart
```

Vemos que agora se creou automáticamente o cartafol *DADMosquera* coa aplicación xa despregada.

6.3. Despugue de aplicacións no ROOT context

Nos puntos anteriores vimos que é posible despregar unha aplicación en webapps e o acceso á mesma será a través dunha URL tipo http://<>ip_servidor>:8080/cartafoldespugue.

E se queremos que unha aplicación sexa accesible dende http://<>ip_servidor>:8080, sen ter que indicar o cartafol de despugue?

Debemos despugala entón no ROOT Context. Temos tres opcións:

1. Copiar o contido de *DADMosquera* directamente en */var/lib/tomcat9/webapps/ROOT*, machacando o contido existente.
2. Cambiar o nome de *DADMosquera.war* a *ROOT.war* para que o despugue automático se realízase no cartafol *ROOT*
3. Crear outro *ROOT* context e eliminar o anterior

Imos coa terceira opción.

Primeiro eliminamos o ROOT context actual e reiniciamos tomcat.

```
rm -R /var/lib/tomcat9/webapps/ROOT
```

```
uadmin@uadmin:~$ sudo rm -R $CATALINA_BASE/webapps/ROOT
uadmin@uadmn:~$ sudo service tomcat8 restart
```

Movemos o cartafol *DADMosquera* a outra ubicación (p.e: /usr/share). Eliminamos por se acaso *DADMosquera.war*.

```
mv /var/lib/tomcat9/webapps/DADMosquera /usr/share
```

```
uadmin@uadmin:~$ sudo mv /var/lib/tomcat8/webapps/DADMosquera /usr/share
```

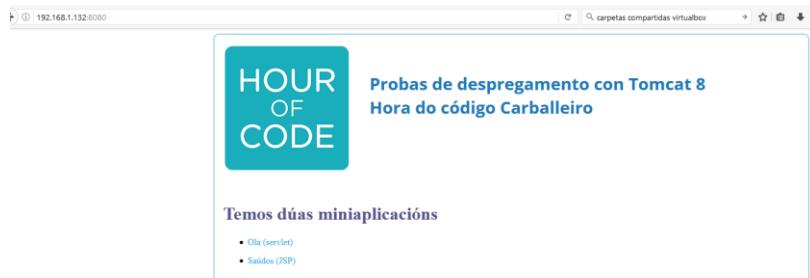
```
rm /var/lib/tomcat9/webapps/DADMosquera.war
```

```
uadmin@uadmin:~$ sudo rm /var/lib/tomcat8/webapps/DADMosquera.war
```

Creamos o ficheiro */var/lib/tomcat9/conf/Catalina/localhost/ROOT.xml* co atributo docBase como se detalla a continuación.

```
uadmin@uadmin: /var/lib/tomcat8/conf/Catalina/localhost
GNU nano 2.5.3 Ficheiro: ROOT.xml
<Context docBase="/usr/share/DADMosquera"/>
```

Reiniciamos Tomcat e probamos a acceder dende o navegador.



6.4. Despregamento dunha aplicación dende o manager.

Poderemos despregar aplicacións empaquetadas con war de forma automática co manager de Tomcat. Ademáis de ver as aplicacións que actualmente están despregadas, poderemos paralas, eliminarlas do servidor, recargarlas, ...

Tamén dispoñemos dun formulario mediante o cal poderemos elixir o .war a despregar.



Gestor de Aplicaciones Web de Tomcat

Mensaje:

Gestor					
Listar Aplicaciones		Ayuda HTML de Gestor		Ayuda de Gestor	
Aplicaciones					
Ruta	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado		true	0	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>
/0_simple	Ninguno especificado	Primeira App feita con Eclipse	true	0	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>
/1_cartas	1.0.10	exercicioCartas	true	0	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>
/2_propiedadesweb	2.0.10(TOMCAT_9)	Propiedades do web.xml e acceso a sesión	true	0	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>
/docs	Ninguno especificado	Tomcat Documentation	true	0	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>
/examples	Ninguno especificado	Servlet and JSP Examples	true	0	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>
/host-manager	Ninguno especificado	Tomcat Host Manager Application	true	0	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>
/manager	Ninguno especificado	Tomcat Manager Application	true	1	<input type="button" value="Arrancar"/> <input type="button" value="Parar"/> <input type="button" value="Expirar sesión"/>

Desplegar					
Desplegar directorio o archivo WAR localizado en servidor					
Trayectoria de Contexto (opcional): <input type="text"/> Version (for parallel deployment): <input type="text"/> URL de archivo de Configuración XML: <input type="text"/> URL de WAR o Directorio: <input type="text"/>	<input type="button" value="Desplegar"/>				

Archivo WAR a desplegar					
Seleccione archivo WAR a cargar <input type="button" value="Examinar..."/> No se ha seleccionado ningún archivo.					
<input type="button" value="Desplegar"/>					

Cando dispoñemos dun war a aplicación despregarase nunha ruta de contexto que se chama igual que o nome do arquivo war. Se o arquivo se nomea a *xestion.war*, a aplicación será finalmente accesible mediante http://<<ip_servidor>>:8080/xestion/

Cando falamos de **contexto** en Tomcat, adoita significar **aplicación**.

Tamén podemos efectuar algunha configuración máis afinada. Se o arquivo se denomina *xestion##10.5.45.war*, significa que a aplicación se despregarán en http://<<ip_servidor>>:8080/xestion/ pero tendo en conta que a súa versión de despregamento é a 10.5.45.

Ruta	Versión
/	Ninguno especificado
/0_simple	Ninguno especificado
/1_cartas	1.0.10
/2_propiedadesweb	2.0.10(TOMCAT_9)

Por outra banda, existe un formulario más fino no que poderemos incluso indicar a ruta de contexto a despregar.

Se tivesemos un war na home de dadmin poderíamos proceder deste xeito:

Desplegar
Desplegar directorio o archivo WAR localizado en servidor
Trayectoria de Contexto (opcional): <input type="text" value="/appSimple"/> Version (for parallel deployment): <input type="text" value="v2.5.72"/> URL de archivo de Configuración XML: <input type="text"/> URL de WAR o Directorio: <input type="text" value="/home/dadmin/0_simple.war"/> <input type="button" value="Desplegar"/>

Mensaje:	OK - Desplegada aplicación en trayectoria de contexto [/appSimple##v2.5.72]
----------	-----------------------------------------------------------------------------

Gestor		
Listar Aplicaciones	Ayuda HTML de Gestor	Ayuda en línea

Aplicaciones					
Ruta	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	
/	Ninguno especificado				true
/1_cartas	1.0.10	exercicioCartas			true
/2_propiedadesweb	2.0.10(TOMCAT_9)	Propiedades do web.xml e acceso a sesión			true
/appSimple	v2.5.72	Primeira App feita con Eclipse			true

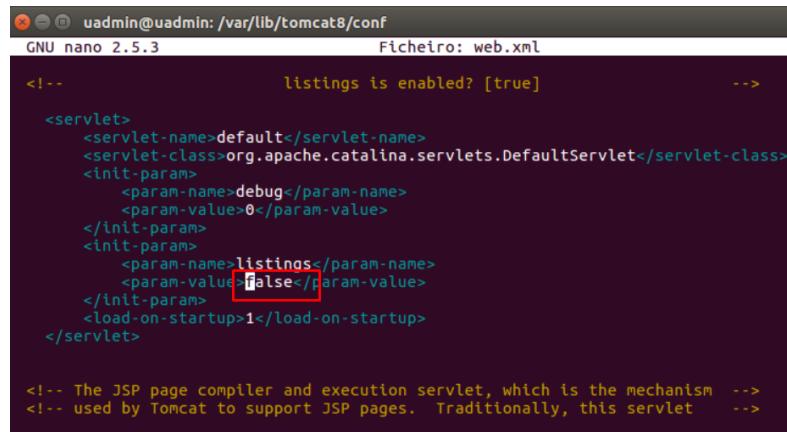
Do mesmo xeito se quixeramos despregar outra aplicación no root, neste formulario poderíamos indicar que a traxectoria do contexto fose / únicamente.

7. Listado de cartafoles

Tal e como facemos no Apache para permitir ou non a listaxe dos arquivos dun cartafol (Indexes) tamén o podemos facer en Tomcat.

De forma xeral podemos configuralo en `$CATALINA_BASE/conf/web.xml`, de forma transversal a todos os cartafoles.

Trataríase de cambiar o valor da propiedade **listings** a **true**.



```

uadmin@uadmin:/var/lib/tomcat8/conf
GNU nano 2.5.3                               Ficheiro: web.xml

<!--
      listings is enabled? [true] -->

<servlet>
  <servlet-name>default</servlet-name>
  <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>false</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<!-- The JSP page compiler and execution servlet, which is the mechanism -->
<!-- used by Tomcat to support JSP pages. Traditionally, this servlet -->

```

Logo a nivel particular podemos configurar cada App mediante o seu descriptor de despregamento `web.xml` particular. Por exemplo:

```

<servlet>
  <servlet-name>DefaultServletOverride</servlet-name>
  <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>DefaultServletOverride</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

```

8. Autenticación e autorización (Memory Realm)

A autenticación e autorización, de modo similar ao que facíamos en Apache podemos levala a cabo en Tomcat empregando como almacén de usuarios **\$CATALINA_BASE/conf/tomcat-users.xml**.

Entón editamos o antedito ficheiro e engadimos, por exemplo dous usuarios con rol *aplicacionMosquera*, para pedirlle autorización de entrada na nosa aplicación principal.

```
<role rolename="aplicacionMosquera"/>
<user username="alumno" password="abc123." roles="aplicacionMosquera"/>
<user username="profesor" password="abc123." roles="aplicacionMosquera"/>
```

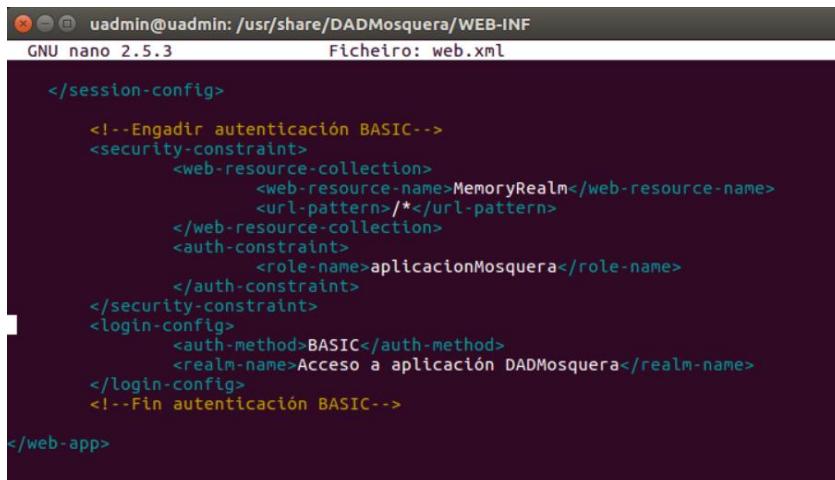
O seguinte paso sería **crear** un ficheiro **context.xml** para a aplicación na que queremos que solicite autenticación. O ficheiro context.xml estará dentro do cartafol META-INF da aplicación en cuestión. No noso caso podería estar en **/usr/share/DADMosquera/META-INF**

O contido podería ser o seguinte:



```
<Context>
    <Realm className="org.apache.catalina.realm.MemoryRealm"/>
</Context>
```

O seguinte paso sería editar o descriptor de despregue (**web.xml**) da aplicación DADMosquera engadindo os seguintes elementos. Lembra que este arquivo está dentro do cartafol WEB-INF da aplicación en cuestión:



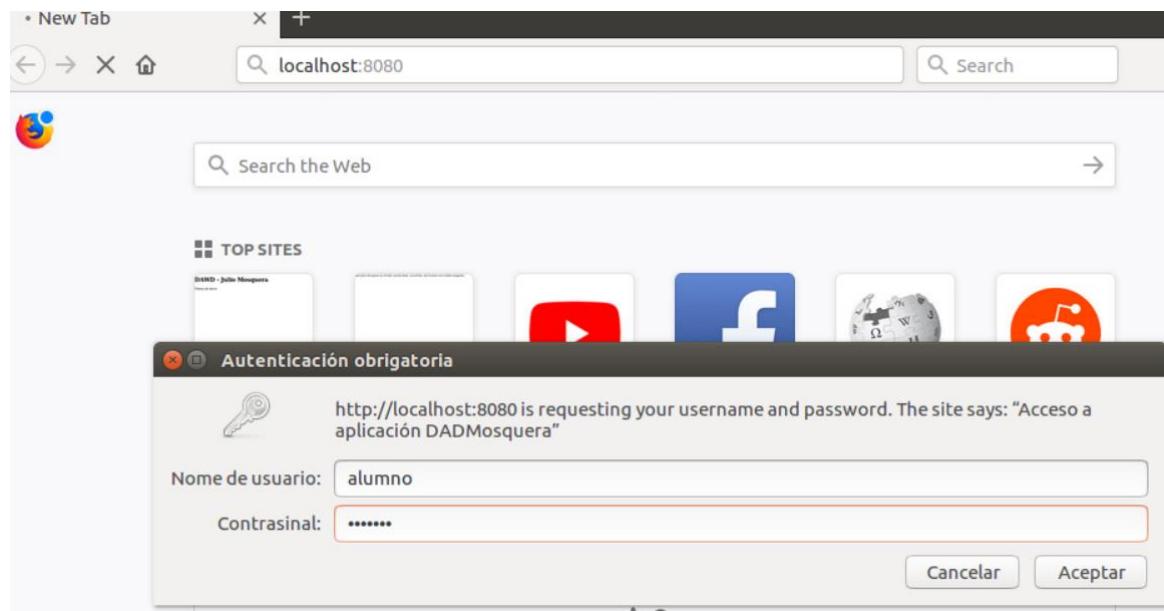
```
</session-config>

<!--Engadir autenticación BASIC-->
<security-constraint>
    <web-resource-collection>
        <web-resource-name>MemoryRealm</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>aplicacionMosquera</role-name>
    </auth-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Acceso a aplicación DADMosquera</realm-name>
</login-config>
<!--Fin autenticación BASIC-->

</web-app>
```

OLLO!!! O ideal sería modificar o web.xml antes de despregar a aplicación e na contorna de desenvolvemento (NetBeans, Eclipse, ...), xa que nunha futura versión de despregue estes cambios non se manterían.

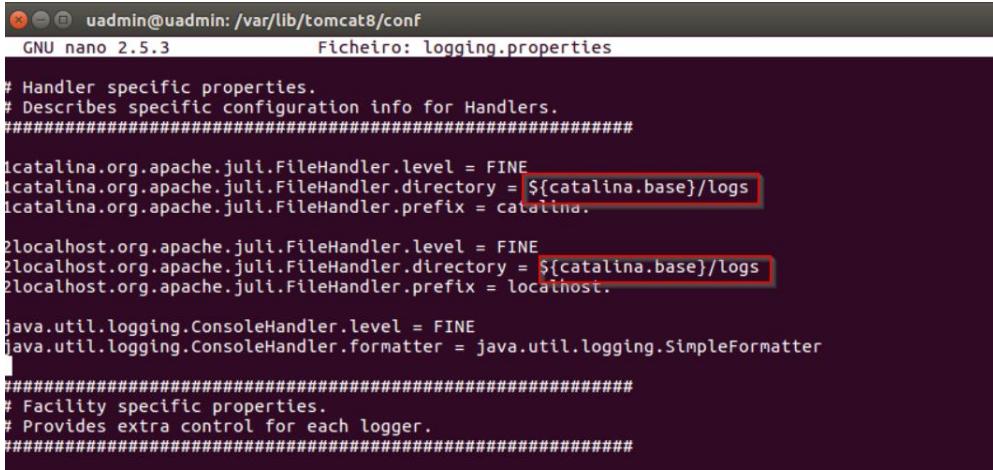
Vemos que xa nos solicita autenticación para entrar na aplicación.



Existen outras formas más elaboradas, escalables e cómodas de protexer o acceso a aplicáisons, p.e: mediante JDBC cunha Base de Datos en MySQL por exemplo. Isto anterior coñécese como JDBCRealm.

9. Logs en Tomcat

No ficheiro `/var/lib/tomcat9/conf/logging.properties` vemos a configuración do manipulador de logs e en onde os vai almacenar.



```

uadmin@uadmin:/var/lib/tomcat8/conf
GNU nano 2.5.3          Ficheiro: logging.properties

# Handler specific properties.
# Describes specific configuration info for Handlers.
#####
catalina.org.apache.juli.FileHandler.level = FINE
catalina.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
catalina.org.apache.juli.FileHandler.prefix = catalina.

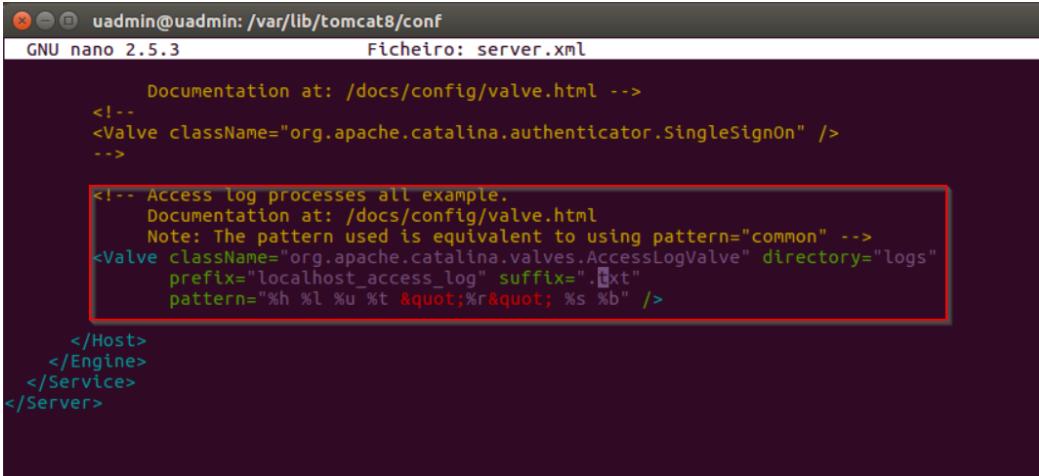
localhost.org.apache.juli.FileHandler.level = FINE
localhost.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
localhost.org.apache.juli.FileHandler.prefix = localhost.

java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
#####

# Facility specific properties.
# Provides extra control for each logger.
#####

```

Se consultamos, ao mesmo tempo `/var/lib/tomcat9/conf/server.xml` vemos más información sobre o nomeamento dos logs.



```

uadmin@uadmin:/var/lib/tomcat8/conf
GNU nano 2.5.3          Ficheiro: server.xml

      Documentation at: /docs/config/valve.html -->
<!--
<Valve className="org.apache.catalina.authenticator.SingleSignOn" />
-->

<!-- Access log processes all example.
     Documentation at: /docs/config/valve.html
     Note: The pattern used is equivalent to using pattern="common" -->
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
      prefix="localhost_access_log" suffix=".txt"
      pattern="%h %l %u %t "%r" %s %b" />

</Host>
</Engine>
</Service>
</Server>

```

En xeral os logs estarán situados en `/var/lib/tomcat9/logs` ou no subcartafol `logs` en Windows. Os logs por defecto son:

- catalina.log: un log con entradas que describen a actividade a nivel de servidor
- localhost.log: un log que recolle a actividade a nivel de web
- localhost_access.log: un log que recolle as peticións dirixidas e procesadas polo servidor
- host_manager.log: un log específico para a app web host-manager
- manager.log: un log específico para a app manager

Todos os arquivos de log teñen o sufijo coa data na que se producen os eventos en distintos formatos DD-MM-YYYY ou similar.

9.2. Logs específicos para unha aplicación web

Ao crear un host virtual podemos establecer logs específicos, pero tamén poderemos configurar logs específicos para unha aplicación web aloxada no mesmo servidor.

Para crear un log específico que amose mensaxes de log a nivel Severe, warning, info, config, ... debemos crear un arquivo denominado **logging.properties** en **WEB-INF\classes** co seguinte contido mímino:

```

1  handlers = org.apache.juli.FileHandler, java.util.logging.ConsoleHandler
2
3  org.apache.juli.FileHandler.level = FINEST
4  org.apache.juli.FileHandler.directory = ${catalina.base}/logs/app_cartas/
5  #Aquí poderíamos poñer o nome do arquivo do log ou ben que o colla automaticamente
6  org.apache.juli.FileHandler.prefix = ${classloader.webappName}.
7
8  java.util.logging.ConsoleHandler.level = FINEST
9  java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
10

```

Crearíase un log denominado **1_cartas##1.0.10.2023-01-17.log** dentro do cartafol **CATALINA_BASE/logs/app_cartas**

Se ademais quixeramos obter un log de accesos coas peticións HTTP procesadas, deberíamos editar o arquivo **/META-INF/context.xml** e incluír o seguinte contido:

```

<?xml version="1.0" encoding="UTF-8"?>
<Context>
    <Valve className="org.apache.catalina.valves.AccessLogValve"
           rotatable="true"
           directory="logs/app_cartas"
           prefix="access_log_cartas_"
           suffix=".txt"
           fileDateFormat="yyyyMMdd_HH"
           pattern="combined"
           buffered="false"
           resolveHosts="false"
    />
</Context>

```

Podemos ver máis opcións do patrón de información a almacenar no log de accesos dende https://tomcat.apache.org/tomcat-9.0-doc/config/valve.html#Access_Logging.

10. Hosts virtuais

Tal e como facíamos en Apache, en Tomcat tamén podemos definir hosts virtuais que atendan peticións a diferentes IPs, portos ou direccións, aínda que é un pouco máis laborioso á vez que configurable.

Imos despregar a aplicación **DADChancas**, pero esta vez será accesible dende a URL www.chancastic.gal. Debemos lembrar configurar o arquivo /etc/hosts ou %windir%/system32/drivers/etc/hosts para mapear a IP do servidor co nome (*esto facémolo porque non temos ningún servidor DNS na rede*).

O cartafol en onde almacenaremos a aplicación web será **/var/lib/tomcat9/webapps-chancas**

O ficheiro de logs de acceso estará en **/var/log/tomcat9/chancas_access_log.data.txt**

Creamos o cartafol no servidor e outorgamos os permisos pertinentes.

```
uadmin@uadmin:~$ sudo mkdir /var/lib/tomcat8/webapps-chancas
[sudo] password for uadmin:
uadmin@uadmin:~$ sudo chown tomcat8:tomcat8 /var/lib/tomcat8/webapps-chancas/
uadmin@uadmin:~$ sudo chmod 755 /var/lib/tomcat8/webapps-chancas/
uadmin@uadmin:~$
```

Agora editamos o arquivo **/var/lib/tomcat9/conf/server.xml** para habilitar o host virtual. Observa a etiqueta **Alias** que permite nomear doutros xeitos ao servidor.

```
uadmin@uadmin:/var/lib/tomcat8/conf
GNU nano 2.5.3                               Ficheiro: server.xml
</Host>
<Host name="www.chancastic.gal" appBase="webapps-chancas"
      unpackWARs="true" autoDeploy="true">
<Alias>chancastic.gal</Alias>
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
      prefix="chancas_access_log" suffix=".txt"
      pattern="%h %l %u %t "%r" %s %b" />
</Host>

</Engine>
</Service>
</Server>
```

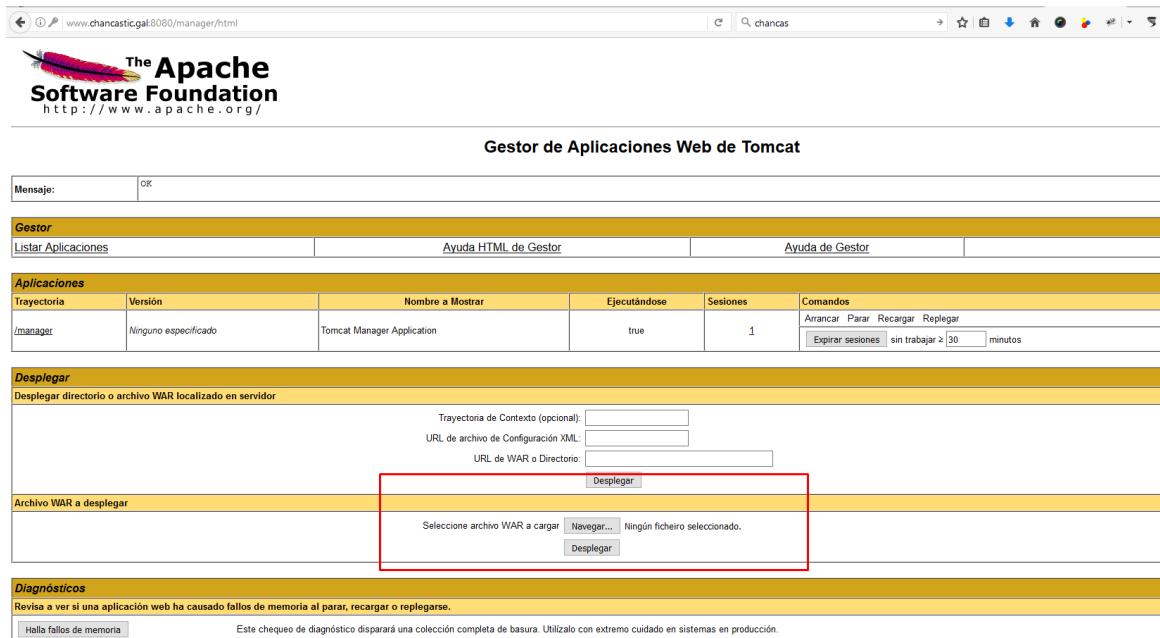
Reiniciamos Tomcat e vemos que se crea de forma automática o cartafol **/var/lib/tomcat9/conf/Catalina/www.chancastic.gal**

```
uadmin@uadmin: /var/lib/tomcat8/conf/Catalina
uadmin@uadmin:~/var/lib/tomcat8/conf/Catalina$ ls -l
total 8
drwxrwxr-x 2 root      tomcat8 4096 Nov 19 20:34 localhost
drwxr-xr-x 2 tomcat8  tomcat8 4096 Nov 20 23:04 www.chancastic.gal
uadmin@uadmin:~/var/lib/tomcat8/conf/Catalina$
```

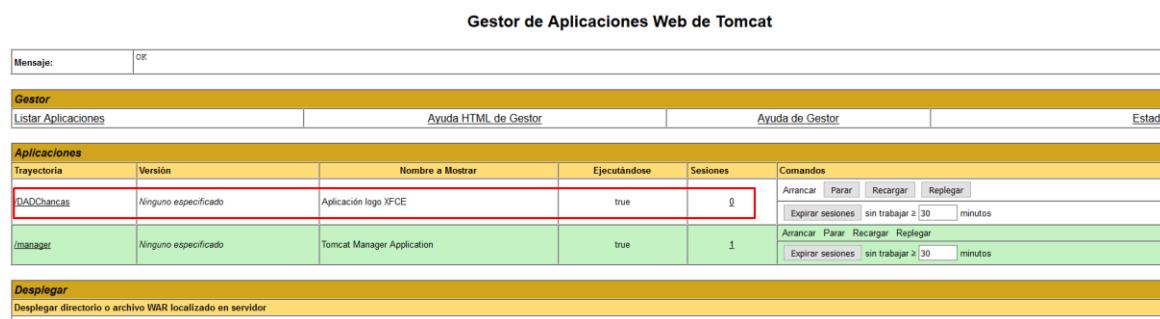
Copiamos `/var/lib/tomcat9/conf/Catalina/localhost/manager.xml` a `/var/lib/tomcat9/conf/Catalina/www.chancastic.gal` para habilitar o Web Manager de Tomcat tamén para este host virtual.

```
uadmin@uadmin: ~
uadmin@uadmin:~$ sudo cp /var/lib/tomcat8/conf/Catalina/localhost/manager.xml /var/lib/tomcat8/conf/Catalina/www.chancastic.gal/
uadmin@uadmin:~$
```

Desde o cliente en onde teñas debidamente configurado o arquivo hosts xa podes acceder a www.chancastic.gal:8080/manager e dende aí despregar o novo aplicativo.

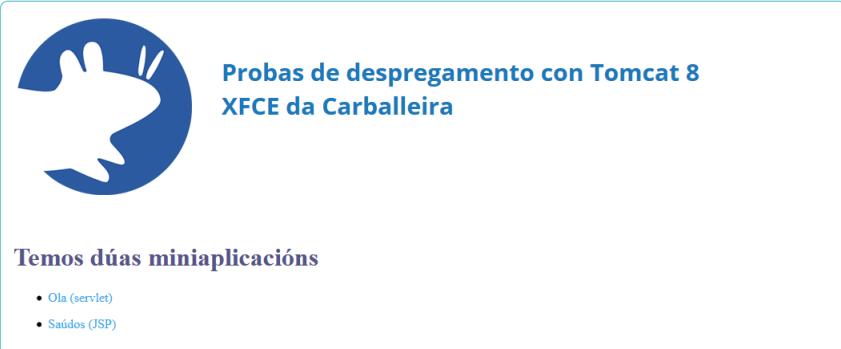


The screenshot shows the Apache Tomcat Manager interface at <http://www.chancastic.gal:8080/manager/html>. The 'Desplegar' (Deploy) section is highlighted with a red box. It contains fields for 'Trayectoria de Contexto (opcional)', 'URL de archivo de Configuración XML:', and 'URL de WAR o Directorio:'. Below these fields is a 'Desplegar' button, which is also highlighted with a red box. The 'Archivo WAR a desplegar' (WAR file to deploy) section shows a message: 'Seleccione archivo WAR a cargar' (Select WAR file to load) and a 'Desplegar' (Deploy) button.



The screenshot shows the Apache Tomcat Manager interface at <http://www.chancastic.gal:8080/manager/html>. The 'Aplicaciones' (Applications) table is shown with two rows. The first row, for 'DADChancas', has its entire row highlighted with a red box. The second row, for 'manager', is green. Both rows have columns for 'Trayectoria' (Context Path), 'Versión' (Version), 'Nombre a Mostrar' (Name to Show), 'Ejecutándose' (Running), 'Sesiones' (Sessions), and 'Comandos' (Commands). The 'Comandos' column for 'DADChancas' includes 'Arrancar', 'Parar', 'Recargar', and 'Replegar'. The 'Comandos' column for 'manager' includes 'Expirar sesiones' (Session timeout), 'Arrancar', 'Parar', 'Recargar', and 'Replegar'. The 'Sesiones' column for 'DADChancas' shows '0', while for 'manager' it shows '1'.

www.chancastic.gal:8080/DADChancas/



The screenshot shows a web browser window with the URL www.chancastic.gal:8080/DADChancas/. The page content is as follows:



Probas de despregamento con Tomcat 8
XFCE da Carballeira

Temos dúas miniaplicacións

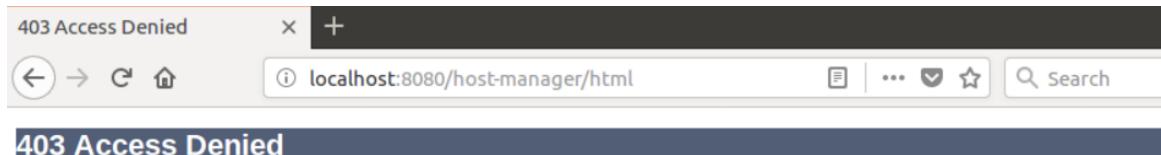
- Ola (servlet)
- Saúdos (JSP)

11. Tomcat host manager

En puntos anteriores instalamos a utilidade tomcat-admin, que entre outras cousas nos permitía acceder ao manager de aplicativos, configuración do servidor, ...

Ademáis do manager temos instalado un xestor de hosts máis visual e más cómodo a través de interface web.

Para acceder debemos acudir a http://<>ip_servidor>:8080/host-manager



403 Access Denied

You are not authorized to view this page.

If you have already configured the Host Manager application to allow access and you have used your browser's back button, used a saved book-mark or similar cross-site request forgery (CSRF) protection that has been enabled for the HTML interface of the Host Manager application. You will need to reset this protection. Once you return to this page, you will be able to continue using the Host Manager application's HTML interface normally. If you continue to see this check that you have the necessary permissions to access this application.

If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation. That file must contain the credentials:

For example, to add the `admin-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

```
<role rolename="admin-gui"/>
<user username="tomcat" password="s3cret" roles="admin-gui"/>
```

Note that for Tomcat 7 onwards, the roles required to use the host manager application were changed from the single `admin` role to the following two roles. You will require for the functionality you wish to access.

- `admin-gui` - allows access to the HTML GUI
- `admin-script` - allows access to the text interface

The HTML interface is protected against CSRF but the text interface is not. To maintain the CSRF protection:

- Users with the `admin-gui` role should not be granted the `admin-script` role.
- If the text interface is accessed through a browser (e.g. for testing since this interface is intended for tools not humans) then the browser must be closed before starting a new session.

Daranos un erro porque non temos configurado o usuario en `tomcat-users.xml` co rol axeitado.

Para iso debemos editar o ficheiro e a algún usuario asignarlle o rol `admin-gui`.

```
<role rolename="aplicacionMosquera"/>
<user username="alumno" password="abc123." roles="aplicacionMosquera"/>
<user username="profesor" password="abc123." roles="aplicacionMosquera"/>
<role rolename="manager-gui"/>
<role rolename="admin-gui"/>
<user username="mosquera" password="abc123." roles="manager-gui, admin-gui"/>
```

Feito isto reiniciamos o servidor, e tamén o navegador web por se tivera algúna sesión iniciada.

Agora veremos o noso host-manager xunto coas máquinas creadas, coa posibilidade de iniciala, parala, ... e tamén podermos crealas novas.

The Apache Software Foundation 



Gestor de Máquina Virtual de Tomcat

Mensaje:

Gestor de Máquina		Ayuda de Gestor de Máquina HTML (En breve)	Ayuda de Gestor de Máquina	Estado de Servicio
Lista de Máquinas Virtuales				
Nombre de Máquina	Aliases de Máquina	Comandos		
localhost	chancastic.gal	Host Manager installed - commands disabled		
www.chancastic.gal		<input type="button" value="Parar"/>	<input type="button" value="Quitar"/>	
Añadir Máquina Virtual				
Máquina Nombre: <input type="text"/> Aliases: <input type="text"/> App base: <input type="text"/> <input checked="" type="checkbox"/> AutoDeploy <input checked="" type="checkbox"/> DeployOnStartup <input checked="" type="checkbox"/> DeployXML <input checked="" type="checkbox"/> UnpackWARs <input checked="" type="checkbox"/> App de Gestor <input type="checkbox"/> CopyXML <input type="button" value="Añadir"/>				

12. Integración de Apache e Tomcat empregando mod_proxy

O que imos tentar facer é que Apache capture as peticións que se dirixan á aplicación DADMosquera.

Debemos lembrar que no propio Ubuntu tiñamos configurado en localhost o ROOT.xml para que lanzase a aplicación DADMosquera. Aínda así poderíamos facer a redirección para o cartafol e o punto que desexemos.



Ao teclear <http://www.mosquera.gal> que a petición a dirixa a Tomcat no porto 8080. Esta configuración podémola facer sobre o sitio por defecto de Apache, aínda que nos, para este exemplo ímolo facer sobre un novo sitio virtual denominado *www_chancastic_gal.conf*.

Os pasos a dar serían os seguintes:

- Habilitamos os módulos proxy e proxy_http de Apache

```
a2enmod proxy
a2enmod proxy_http
```

- Editamos o ficheiro **/etc/apache2/mods-enabled/proxy.conf** co seguinte contido

```
uadmin@uadmin: /etc/apache2
GNU nano 2.5.3          Ficheiro: mods-enabled/proxy.conf

<IfModule mod_proxy.c>

# If you want to use apache2 as a forward proxy, uncomment the
# 'ProxyRequests On' line and the <Proxy *> block below.
# WARNING: Be careful to restrict access inside the <Proxy *> block.
# Open proxy servers are dangerous both to your network and to the
# Internet at large.
#
# If you only want to use apache2 as a reverse proxy/gateway in
# front of some web application server, you DON'T need
# 'ProxyRequests On'.

#ProxyRequests On
<Proxy *>
    AddDefaultCharset off
    Require all granted
    #
    #Require local
</Proxy>
```

- Reiniciamos apache

```
service apache2 restart
```

4. Creamos un novo host virtual, denominado [mosquera_gal.conf](#) coa seguinte configuración.

```
uadmin@uadmin:/etc/apache2/sites-available
Ficheiro Editar Ver Buscar Terminal Axuda
GNU nano 2.5.3          Ficheiro: mosquera_gal.conf

<VirtualHost *:80>
    ServerName www.mosquera.gal
    ServerAlias mosquera.gal

    ProxyPreserveHost on
    ProxyPass / http://localhost:8080/
    ProxyPassReverse / http://localhost:8080/

</VirtualHost>
```

Se quixeramos redirixir só unha aplicación determinada (p.e: *sample*) poríamos *ProxyPass /sample http://localhost:8080/sample*

5. Habilitamos o sitio virtual

```
a2ensite mosquera.gal.conf
```

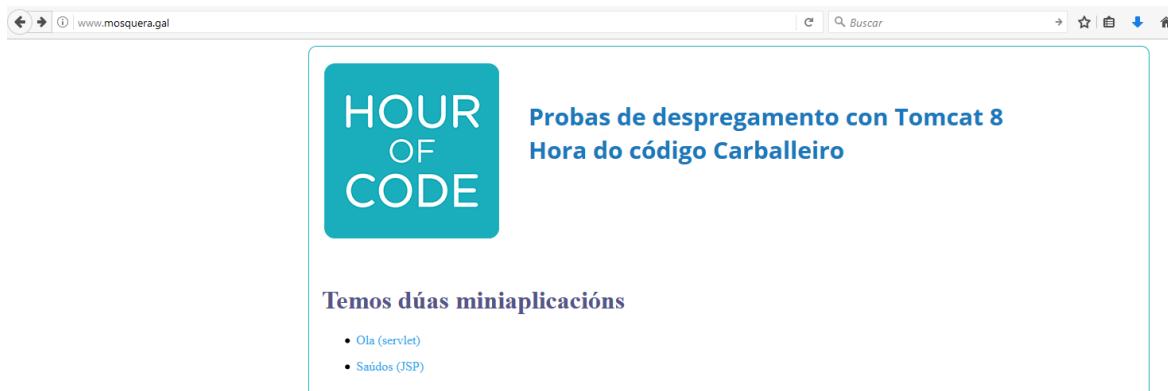
6. Reiniciamos apache

7. Editamos o ficheiro **/var/lib/tomcat9/conf/server.xml**. Neste caso o conector HTTP recibe as peticións de Apache por HTTP, e hai que indicar o porto do proxy empregado en Apache. Engadimos, polo tanto a propiedade *proxyPort="80"*

```
uadmin@uadmin:/var/lib/tomcat8/conf
GNU nano 2.5.3          Ficheiro: server.xml

    and responses are returned. Documentation at :
    Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
    Java AJP Connector: /docs/config/ajp.html
    APR (HTTP/AJP) Connector: /docs/apr.html
    Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        URIEncoding="UTF-8"
        redirectPort="8443"
        proxyPort="80"
    />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
        port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443" />
```

8. Reiniciamos Tomcat e xa debería de funcionar



www.mosquera.gal

HOUR OF CODE

Probas de despregamento con Tomcat 8
Hora do código Carballeiro

Temos dúas miniaplicacións

- Ola (servlet)
- Saídos (JSP)

13. Seguridade SSL

Podemos configurar un determinado host (p.e: localhost) para que soporte cifrado SSL, é dicir, que poidamos empregar HTTPS para poder acceder aos contidos da web.

Imos facer un pequeno exemplo de configuración, tendo en conta que se pode evolucionar moito máis.

Os pasos a dar serían os seguintes:

1. Crear os certificados coa clave pública e privada. Podemos empregar a ferramenta *keytool*.

```
keytool -genkey -alias tomcatmosquera -keyalg RSA -keystore /var/lib/tomcat9/depositochaves
```

Temos que inserir a información que creamos oportuna para o certificado. Debemos ter en conta que é autoasignado e a súa validez prácticamente inexistente.

```
uadmin@uadmin:~$ sudo keytool -genkey -alias tomcatmosquera -keyalg RSA -keystore /var/lib/tomcat8/depositochaves
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Julio Mosquera
What is the name of your organizational unit?
[Unknown]: Departamento de Informática
What is the name of your organization?
[Unknown]: CIFP A Carballeira
What is the name of your City or Locality?
[Unknown]: Ourense
What is the name of your State or Province?
[Unknown]: Ourense
What is the two-letter country code for this unit?
[Unknown]: ES
Is CN=Julio Mosquera, OU=Departamento de Informática, O=CIFP A Carballeira, L=Ourense, ST=Ourense, C=ES correct?
[no]: yes

Enter key password for <tomcatmosquera>
      (RETURN if same as keystore password):

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore /var/lib/tomcat8/depositochaves -destkeystore /var/lib/tomcat8/depositochaves -deststoretype pkcs12".
uadmin@uadmin:~$
```

2. Agora modificamos o arquivo */var/lib/tomcat9/conf/server.xml* incluíndo un novo conector indicando os valores asociados ao certificado creado no paso anterior.

```
Ficheiro Editar Ver Buscar Terminal Axuda
GNU nano 2.5.3                                     Ficheiro: conf/server.xml

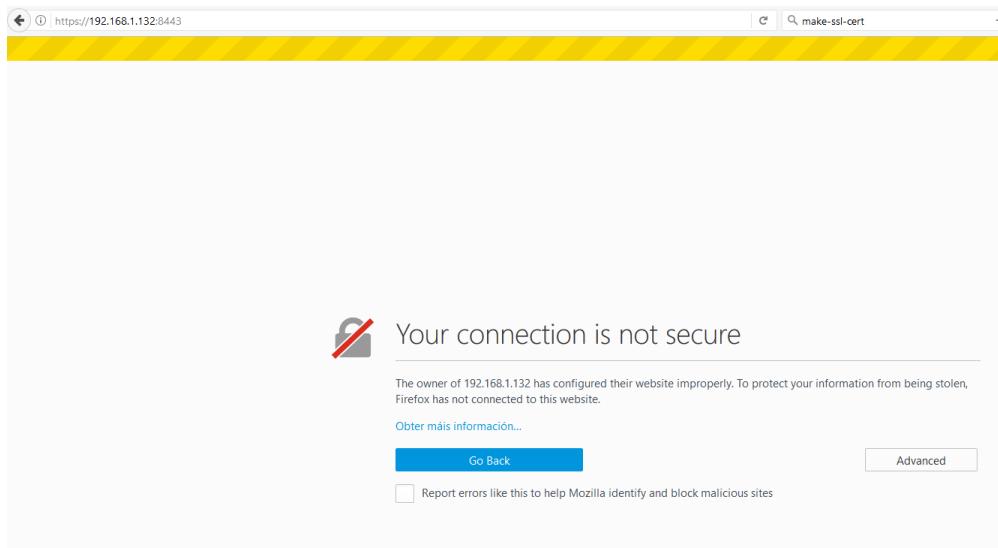
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
    port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
-->
<!-- Define a SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation that requires the JSSE
style configuration. When using the APR/native implementation, the
OpenSSL style configuration is required as described in the APR/native
documentation -->
<Connector port="8443"
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150"
    SSLEnabled="true"
    scheme="https"
    secure="true"
    clientAuth="false"
    sslProtocol="TLS"
    keystoreFile="/var/lib/tomcat8/depositochaves"
    keystorePass="abc123."
    keyAlias="tomcatmosquera"
    keyPass="abc123."
/>
```

3. Reiniciamos tomcat e asegurámonos que estamos a escutar nos porto 8443 (HTTPS no noso caso)

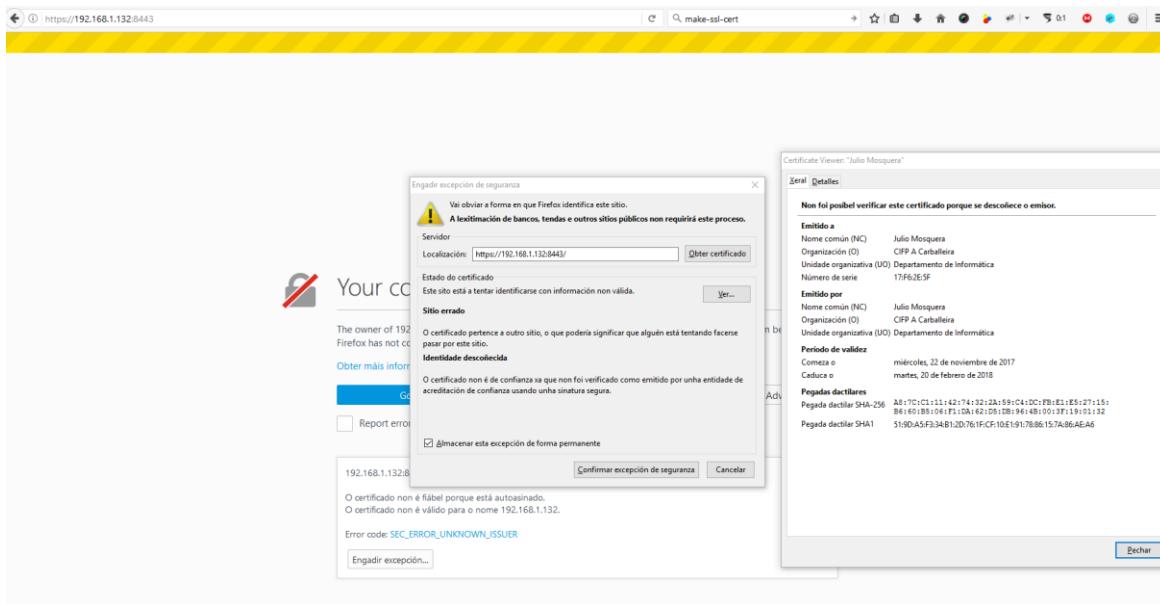
```
ss -ltn
```

```
uadmin@uadmin:/var/lib/tomcat8$ sudo netstat -ltn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp     0      0 127.0.1.1:53              0.0.0.0:*              LISTEN
tcp     0      0 127.0.0.1:631             0.0.0.0:*              LISTEN
tcp6    0      0 127.0.0.1:8005            ::*:*                  LISTEN
tcp6    0      0 :::8080                ::*:*                  LISTEN
tcp6    0      0 :::80                 ::*:*                  LISTEN
tcp6    0      0 :::1:631               ::*:*                  LISTEN
tcp6    0      0 :::8443               ::*:*                  LISTEN
```

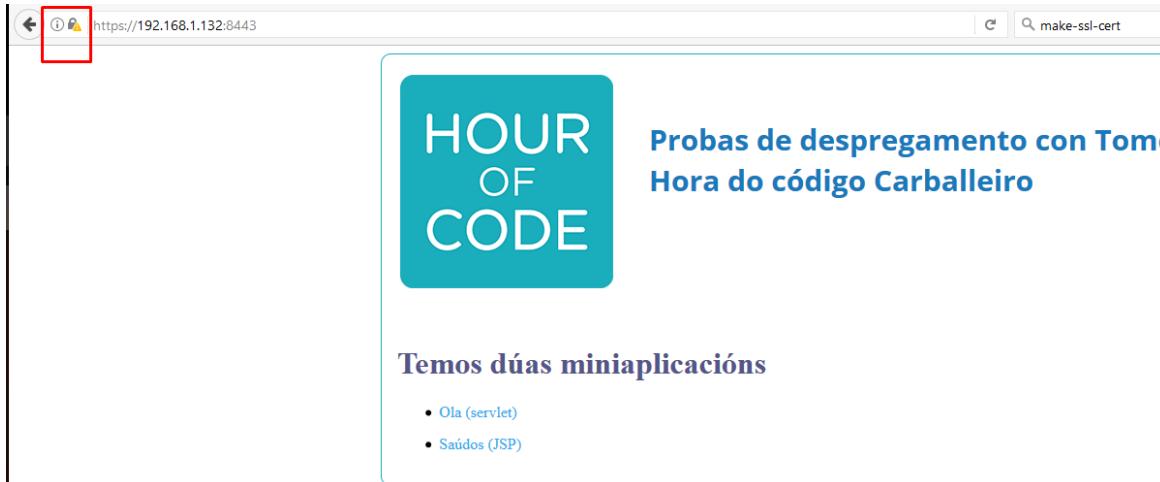
Ao tentar iniciar nun navegador unha conexión mediante <https://<>:8443> saíranos a seguinte advertencia indicando que a conexión non é segura xa que o certificado non é válido, está autoasasinado por nos mesmos.



Picamos en avanzado (navegador Mozilla Firefox), prememos en engadir excepción, e se queremos podemos ver detalles do antedito certificado.



Confirmamos a excepción de seguridade e listo.

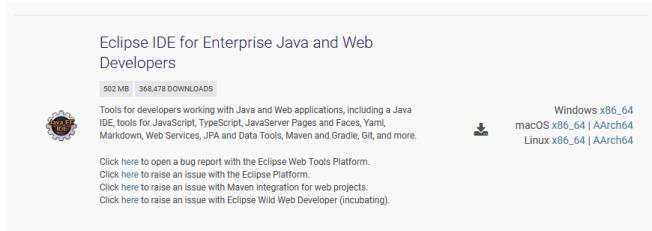


14. Tomcat no IDE Eclipse

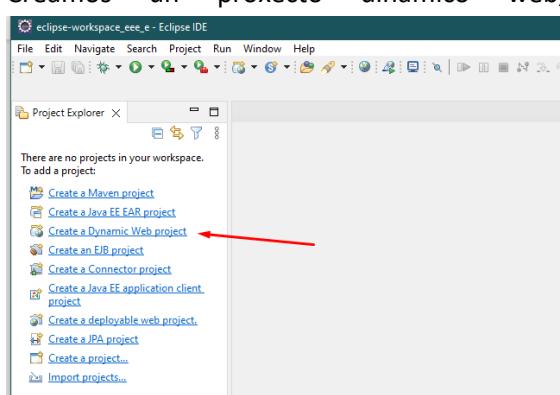
É moi habitual que teñamos que efectuar algún cambio ou efectuar consideracións nalgunha parte do código da nosa aplicación. Tampouco non está demais tentar reproducir na nosa contorna de desenvolvemento o escenario que nos podemos atopar en produción.

Para iso, neste caso, imos ver os pasos xenéricos á hora de preparar a contorna Eclipse.

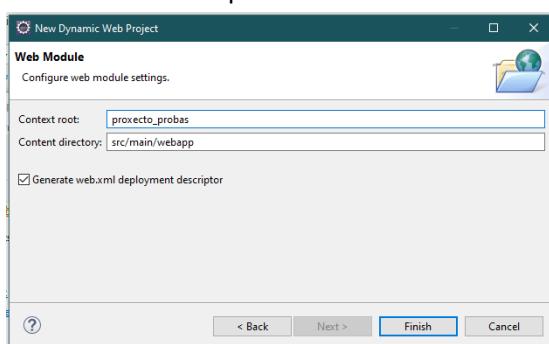
- Descargamos eclipse para desenvolver proxectos JavaEE



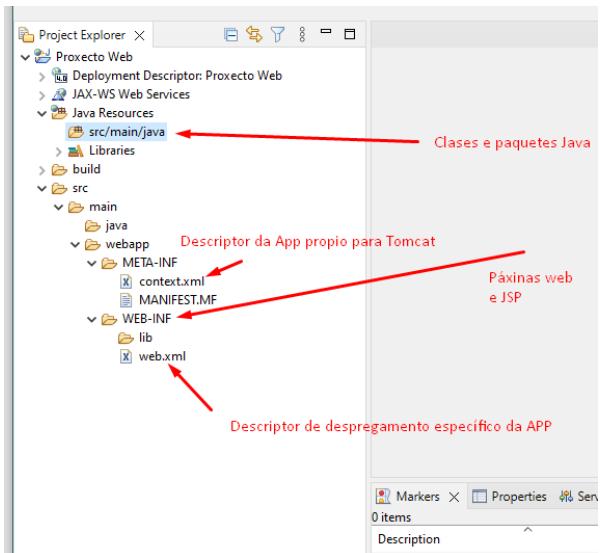
- Dispoñer dun Apache Tomcat debidamente configurado (en contornas windows, descomprimido nun cartafol accesible) e cun JDK axeitado igualmente.
- Creamos un proxecto dinámico web, ou ben abrimos un existente.



- Seguimos co asistente e finalizamos escollendo un contexto de despregamento axeitado. Indicamos tamén que xere un web.xml.

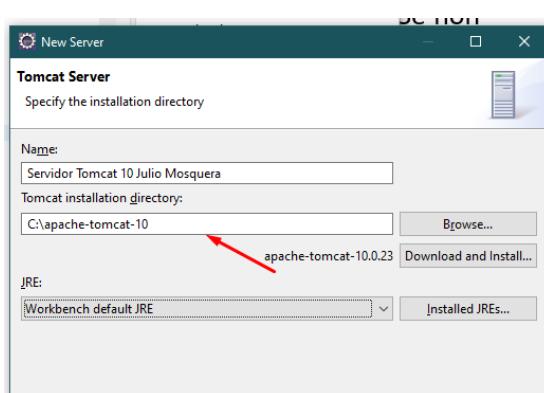
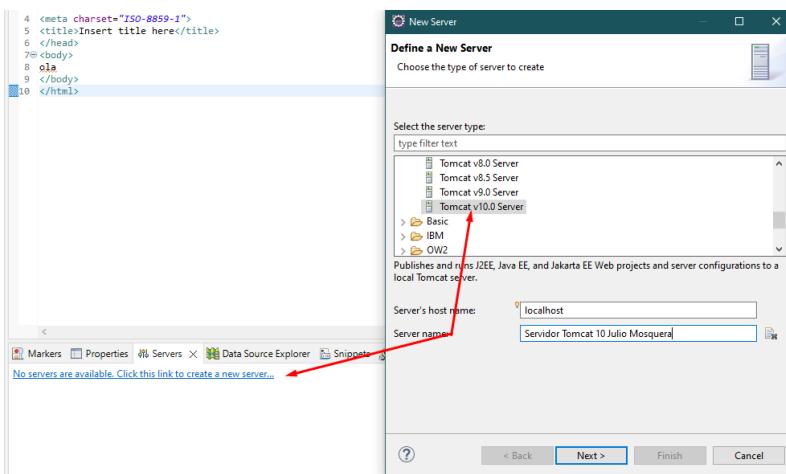


A estrutura de cartafoles é a seguinte

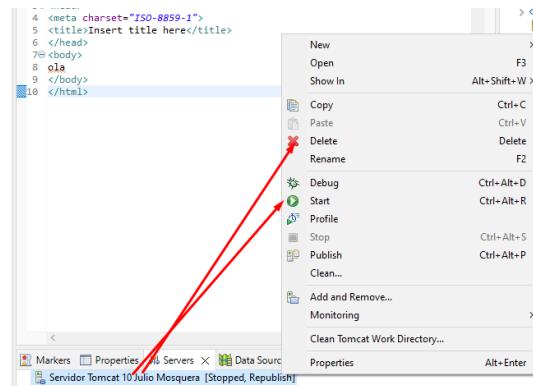


Poderemos crear un ficheiro index.html para probar que funciona o servidor.

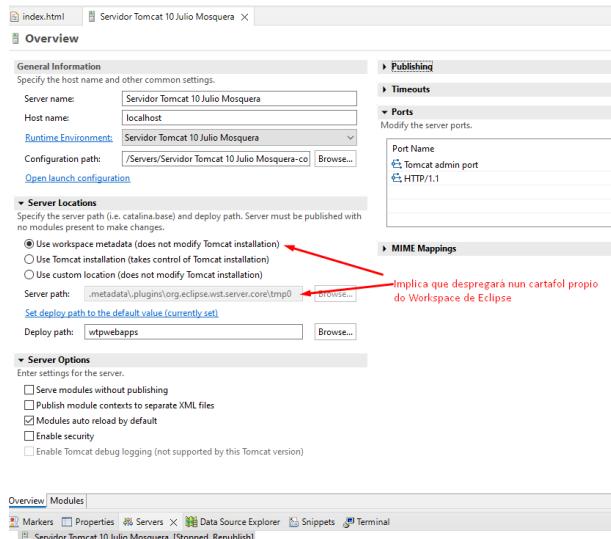
Agora teremos que configurar o servidor. Acudimos á lapela de **Servers** na parte inferior. Se non aparece acude a Window => Show View e procura Servers.



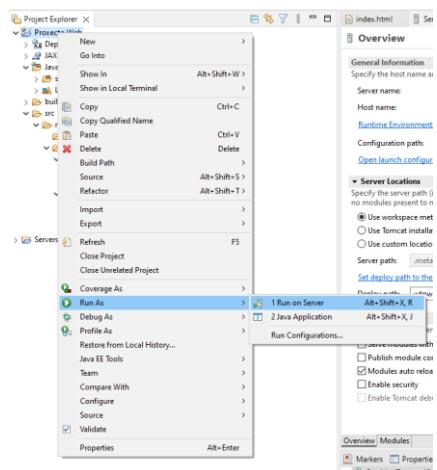
Xa temos o servidor listo e poderemos arrincalo, reinicialo, ...

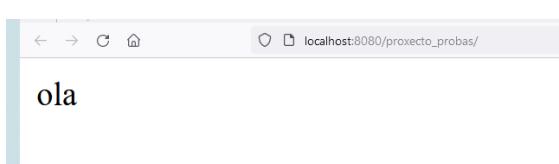
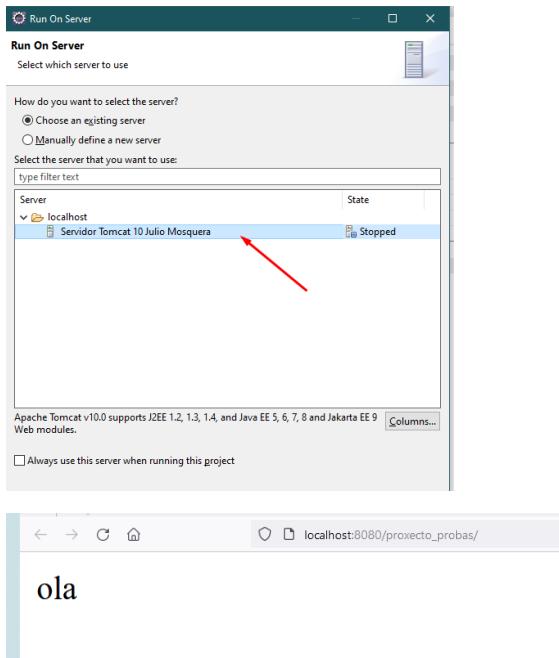


Con doble click podremos ver más información ao respecto.

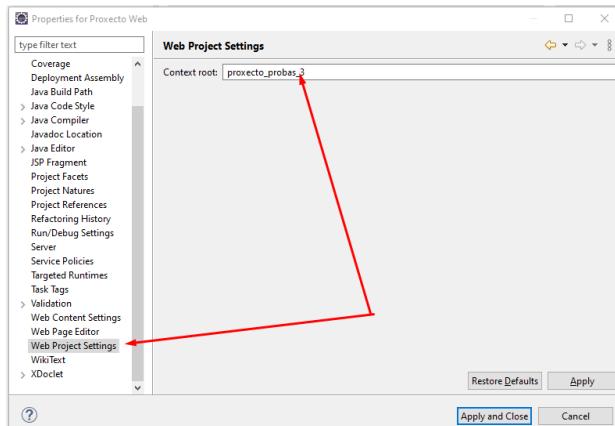


Agora configuraremos a App para que se despregue nese servidor.

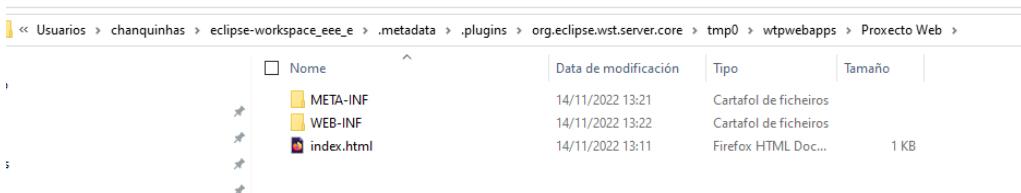




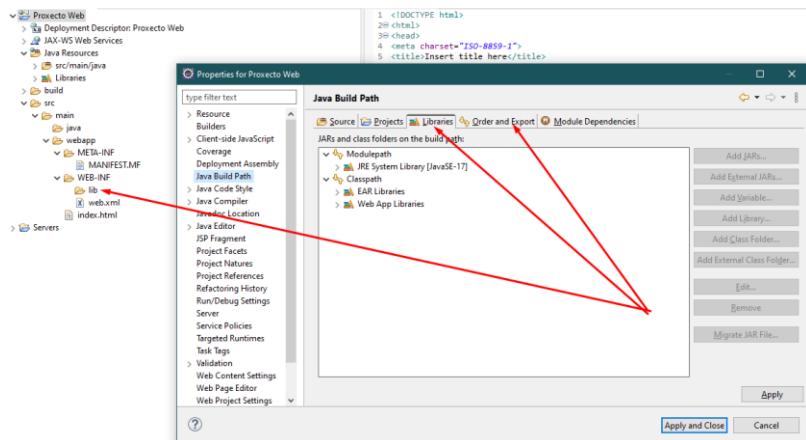
Poderemos modificar a ruta de despregamento en calquera momento.



No noso disco o despregamento farase, tal e como indicamos, no plugin de eclipse asociado.



Debemos ter en consideración que se precisamos de bibliotecas específicas para o proxecto deberemos incorporalas e incluso exportalas na creación do war para que se almacenen no cartafol lib de WEB-INF (por exemplo, os drivers de Mysql).



Outra opción é que no cartafol **lib** do tomcat teñamos os drivers en cuestión para todas as apps despregadas.

Debemos considerar tamén aspectos de codificación dende o Window=> Preferences => General => Workspace, para traballar con UTF-8, e incluso, se traballamos con outros proxectos doutras computadoras dende Window=> Preferences => General => Content Types => Text JSP, HTML, ... e indicar o **default encoding** máis axeitado.

14.1. O arquivo web.xml

Este arquivo pode conter unha serie de elementos que nos permiten orientar o comportamento da aplicación (https://docs.oracle.com/cd/E29542_01/web.1111/e13712/web_xml.htm). Podemos destacar a especificación dos ficheiros por defecto, o display name da aplicación, a duración da sesión, o mapeo dos servlets, a configuración de páxinas de erro específicas, ...

Este descriptor podería ser algo así coma un virtual host de Apache moi restrinxido, e coa vantaxe de poder definirse na propia aplicación.

Exemplo dun web.xml.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  id="WebApp_ID" version="4.0">
  <display-name>Propiedades do web.xml e acceso a sesión</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <session-config>
    <session-timeout>15</session-timeout>
  </session-config>
  | 
  <env-entry>
    <env-entry-name>nomeBaseDeDatos</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>almacenMosquera</env-entry-value>
  </env-entry>
  <env-entry>
    <env-entry-name>arquivoLog</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>f:/incidencias.log</env-entry-value>
  </env-entry>

  <error-page>
    <error-code>404</error-code>
    <location>/WEB-INF/errorpages/ErrorPage404.jsp</location>
  </error-page>
</web-app>

```

En CATALINA_BASE tamén existe dentro de conf un **arquivo web.xml común** a todas as aplicacións web despregadas. Todas as configuracións que repliquemos a nivel de aplicación terán prioridade sobre as xenéricas.

15. Configurar o acceso a datasources

É moi usual configurar o acceso ás bases de datos das aplicacións a través de JNDI dentro do contexto dunha aplicación no canto de emplegar as clásicas conexións persistentes por JDBC. Para iso Tomcat ofrécenos dúas alternativas:

- Un datasource só accesible a través da propia aplicación.
 - Configurando o arquivo META_INF/context.xml definindo o datasource
 - Configurando o arquivo WEB_INF/web.xml accedendo ao datasource definido
- Un datasource ou varios compartidos por todas as aplicacións
 - Configurando o arquivo de Tomcat, \$CATALINA_BASE/conf/server.xml definindo o datasource
 - Configurando o arquivo META_INF/context.xml enlazando o datasource en cuestión co definido en server.xml
 - Configurando o arquivo WEB_INF/web.xml accedendo ao datasource definido en context.xml

Exemplo de anaco de web.xml

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

Exemplo do context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource name="jdbc/dwcsprodutos" type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/dwcsprodutos?serverTimezone=UTC"
    username="root" password="" maxActive="20" maxIdle="10" maxWait="10000" />

</Context>
```

Se quixeramos definilos a nivel de servidor o web.xml quedaría como está, pero o server.xml tería que ter unha entrada similar a esta dentro de **GlobalNamingResources**

```
<GlobalNamingResources>
  <!-- Editable user database that can also be used by
      | UserDatabaseRealm to authenticate users
  -->

  <Resource name="jdbc/dwcsprodutos"
    global="jdbc/dwcsprodutos"
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/dwcsprodutos?serverTimezone=UTC"
    username="root"
    password=""
    maxActive="100"
    maxIdle="20"
    minIdle="5"
    maxWait="10000"/>
```

E o context.xml correspondente debería ter algo similar a isto:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <ResourceLink name="jdbc/dwcsprodutos"
    global="jdbc/dwcsprodutos" -->
    auth="Container"
    type="javax.sql.DataSource" />
</Context>
```



Dende unha clase Java o acceso a este recurso con Tomcat faríase:

```
try {
  Context ctx = new InitialContext();

  String tipoAlmacen = (String) ctx.lookup("java:comp/env/tipoAlmacen");
  dataSource = (DataSource) ctx.lookup("java:comp/env/jdbc/dwcsprodutos");
  Connection con = dataSource.getConnection();

} catch (NamingException | SQLException e) {
  System.out.println(e.getMessage());
}
```