



**AMPLIACIÓN DE BASES DE DATOS**

---

**INSTALACIÓN Y  
CONFIGURACIÓN DE  
TOMCAT**

---

Soporte de Oracle a Aplicaciones

---

Orlando Alemán Ortiz

4º Ing. Informática

Curso 2005/06



# Licencia

---



Esta obra ha sido publicada bajo licencia "Reconocimiento-NoComercial-CompartirIgual 2.5 Spain" de Creative Commons, la cual implica que:

## Usted es libre de:

- copiar, distribuir y comunicar públicamente la obra
- hacer obras derivadas

## Bajo las condiciones siguientes:



**Reconocimiento.** Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador.



**No comercial.** No puede utilizar esta obra para fines comerciales.



**Compartir bajo la misma licencia.** Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

## Y además:

- Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
- Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor
- Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.

Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc-sa/2.5/es/> o envíe una carta a Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

## Índice de contenido

Objetivos.....	4
Introducción.....	4
Requisitos del entorno de instalación.....	4
Software utilizado.....	5
Instalación de Tomcat.....	5
Comprobación: Puesta en funcionamiento.....	6
Configuración básica de Tomcat.....	6
Comprobación: Ejecución de una aplicación JSP.....	8
Instalación del driver JDBC.....	8
Anexo I.....	10
Anexo II.....	12

# Objetivos

---

- ★ Instalación y configuración de *Apache Tomcat* como servidor Web
- ★ Instalación del driver Oracle para JDBC

# Introducción

---

El presente documento pretende abordar de manera sencilla la instalación de un servidor de aplicaciones básico que dé soporte para acceder a un sistema de gestión de bases de datos. En este contexto, *Apache Tomcat* aparece como una opción más que interesante, ya que soporta *JavaServer Pages* y el estándar de acceso *JDBC*.

*Apache Tomcat*, o simplemente *Tomcat*, es un contenedor de *servlets* o programas que ofrecen funciones suplementarias a un servidor. Incluye el compilador *Jasper*, que compila las *JSPs* convirtiéndolas en *servlets*. A menudo se presenta como motor de *servlets* en combinación con un servidor Web (generalmente *Apache*), aunque también puede funcionar como servidor Web por sí mismo, eso sí, con peor rendimiento. Dado que *Tomcat* fue escrito en *Java*, funciona en cualquier sistema operativo que disponga de la máquina virtual.

Las *JavaServer Pages* son los elementos básicos de una tecnología desarrollada por *Sun Microsystems*, basada en el procesamiento de scripts *Java* y que permite a los desarrolladores generar dinámicamente páginas web (*HTML*, *XML* u de otro tipo).

Para posibilitar el acceso a una base de datos desde una página *JSP* haciendo uso de *JDBC* se necesita de un *driver* o controlador, que de no estar ya incluido en las librerías de la máquina virtual, pudiera hacer necesaria su adquisición.

# Requisitos del entorno de instalación

---

Las características del entorno donde se va a llevar a cabo la instalación son:

- Máquina: Computador Personal (x86)
- Sistema Operativo: Microsoft Windows XP
- Software Instalado: J2SE 1.5 **[Imprescindible]**
- Software de compresión/descompresión: 7zip [sólo en el caso de bajar binarios comprimidos]

El hecho de ejecutar *Tomcat* sobre la máquina virtual de *Java* posibilita su instalación en la mayoría de plataformas existentes hoy día, pese a que en nuestro caso lo hagamos sobre el sistema operativo de *Microsoft*.

## Software utilizado

---

- Apache Tomcat 4.1  
(<http://tomcat.apache.org/download-41.cgi>)
- JDBC Driver para Oracle 10g  
([http://www.oracle.com/technology/software/tech/java/sqlj\\_jdbc/index.html](http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html))

## Instalación de Tomcat

---

Pongamos por caso que hemos descargado *Tomcat* en el formato comprimido TAR.GZ. Haciendo uso de una herramienta de descompresión, extraemos el contenido hacia el directorio donde se va a montar el servidor, denominado en *Tomcat* con la denominación CATALINA\_HOME. En nuestro caso su ruta será "C:\tomcat".

Como consecuencia del proceso, en "C:\tomcat" podremos encontrar la siguiente estructura de directorios:

- bin - arranque, cierre, y otros scripts y ejecutables
- common - clases comunes que pueden utilizar *Catalina* y las aplicaciones web
- conf – ficheros *XML* y los correspondientes *DTDs* para la configuración de Tomcat
- logs - logs de *Catalina* y de las aplicaciones
- server - clases utilizadas solamente por *Catalina*
- shared - clases compartidas por todas las aplicaciones web
- temp – almacenamiento temporal para la máquina Java
- webapps - directorio que contiene las aplicaciones web
- work - almacenamiento temporal de ficheros y directorios

(fuente: Wikipedia)

Pero no basta simplemente con situar los archivos, también es necesario indicar de forma global su localización final. Ésto se hace mediante la definición de la variable global CATALINA\_HOME, que evidentemente contendrá el *path* del servidor de aplicaciones.

Para definir una variable global en *Windows* disponemos de varias posibilidades. La primera es hacer uso del comando "SET" en el terminal desde el que luego se arrancará el servidor. Véase:

```
C:\> set CATALINA_HOME="C:\tomcat"
```

La desventaja de este método es que su efecto dura lo que dure la sesión en el terminal donde se ejecutó la instrucción. Su utilización en un archivo de ejecución por lotes (.bat) podría ser la mejor salida

Una segunda forma, y posiblemente la mejor, es definir CATALINA\_HOME como una variable global de usuario o de sistema. Para ello, accedemos a "Inicio > Panel de Control > Mi PC > Propiedades > Opciones avanzadas > Variables de entorno" y añadimos una

entrada para la nueva definición.

Un detalle a comprobar antes de acabar con primera fase es si la variable `JAVA_HOME` se encuentra definida y apunta correctamente al directorio raíz de la instalación de la *J2SE Development Kit*. Para ello, ejecutamos "set" sin parámetros en una consola. Como resultado obtendremos la lista de variables definidas actualmente. En caso de que no se encuentre definida o no posea la ruta completa, procedemos como en la definición de `CATALINA_HOME`.

## Comprobación: Puesta en funcionamiento

El mejor método para conocer si el proceso marcha bien es ejecutar el *Tomcat*. Por defecto está configurado para ofrecer una página de prueba muy sencilla. Así que, si nos encontramos algún problema durante esta fase, seguro será de nuestra cosecha.

La orden para iniciar la ejecución es:

```
%CATALINA_HOME%\bin\startup
```

Como consecuencia, el servidor arrancará escuchando peticiones sobre uno o varios puertos. En este caso, observamos que uno de ellos es el 8080. Si tecleamos ahora en un navegador la dirección "http://localhost:8080" obtendremos la página de prueba que habíamos dicho.

Antes de pasar a la siguiente etapa conviene detener la ejecución del servidor. Ésto lo conseguiremos ejecutando

```
%CATALINA_HOME%\bin\shutdown
```

## Configuración básica de Tomcat

Los aspectos de configuración más importantes para nuestros propósitos actuales se encuentran recogidos en dos ficheros situados bajo "%CATALINA\_HOME%\conf". "web.xml" recoge los valores por defecto a utilizar por todas las aplicaciones web cargadas en la instancia de *Tomcat*, como pudiera ser por ejemplo la página a cargar por defecto. "server.xml", por contra, contiene la definición estructural del servidor: Nombre del *host*, servicios, conectores, etc.

En principio, nos conformamos con el contenido actual de "web.xml" y pasamos a explicar algunos detalles de "server.xml".

Componentes Del Fichero "server.xml"		
Etiqueta	Explicación	Ejemplo
<code>&lt;server&gt;</code> <code>&lt;/server&gt;</code>	(Único y engloba toda la configuración) Define el elemento de configuración básico del fichero server.xml Es único y contiene uno o más servicios ("Service"). El atributo "port" indica el puerto destinado a la escucha del comando de cierre, indicado por "shutdown" o cierre.	<code>&lt;Server port="8005"</code> <code>shutdown="SHUTDOWN"&gt;</code>
<code>&lt;Listener/&gt;</code>	(Único) Permiten definir las clases JMX que	<code>&lt;Listener</code> <code>className="org.apache.catalina.mbean</code>

Componentes Del Fichero "server.xml"		
	permitirá escuchar Tomcat.	s.ServerLifecycleListener" />
<GlobalNamingResources> </GlobalNamingResources>	(Único) Permite definir elementos JNDI para ser utilizados globalmente.	[VER ANEXO]
<Service> </Service>	Estas etiquetas permiten agrupar uno o más conectores de forma que compartan un único contenedor de aplicaciones. Poseen un único atributo, "name", que fija los identificadores individuales. Si "name" se fija como "Catalina" o "Tomcat-Standalone", se habilitará a Tomcat como servidor web independiente.	<Service name="Tomcat-Standalone"></Service>
<Connector />	(dentro de "Service") Conecta un contenedor de datos con el exterior, definiendo el elemento final a través del cual se realizarán las peticiones de usuario y se enviarán las respuestas. Entre sus parámetros de configuración están el puerto de escucha, "port", la clase encargada de su definición, "className" y el número máximo de conexiones simultáneas permitidas, "acceptCount".	<Connector className="org.apache.coyote.tomcat4.CoyoteConnector" port="8080" minProcessors="5" maxProcessors="75" enableLookups="true" acceptCount="100" connectionTimeout="20000" useURValidationHack="false" disableUploadTimeout="true" />
<Engine> </Engine>	(dentro de "Service") Punto donde se procesan las peticiones que llegan a los "Connector" que posean en la cabecera el valor de "defaultHost" como destino.	<Engine name="Standalone" defaultHost="localhost">
<Logger/>	(dentro de "Service" o de "Host") Permite establecer el nombre del fichero de logs. Como parámetros tiene la clase encargada de su definición, "className", el formato nombre del archivo, como la unión de un prefijo, "prefix", y un sufijo, "suffix".	<Logger className="org.apache.catalina.logger.FileLogger" prefix="catalina_log." suffix=".txt" timestamp="true"/>
<Host></Host>	Con estas etiquetas podemos definir uno o más elementos Host virtuales para atender a las peticiones.	<Host name="localhost" debug="0" appBase="webapps" unpackWARs="true" autoDeploy="true">
<Context> </Context>	(dentro de "Host") Se utiliza para indicar la ruta ("docBase") a partir de la cual se encuentran las aplicaciones a ser ejecutadas en Tomcat (a partir de "%CATALINA_HOME%\webapps" y el path url ("path") a partir del cual acceder a los servicios.	[VER ANEXO]

En base a estas explicaciones, tomamos por base el mismo fichero que nos proporciona el fabricante y sobre él realizamos las modificaciones necesarias para adaptarlo a nuestras necesidades. En el Anexo I, se muestra el fichero de ejemplo que hemos confeccionado para esta prueba.

## Comprobación: Ejecución de una aplicación JSP

---

Una vez hemos configurado *Tomcat* a nuestro gusto, pasamos a ejecutar una sencilla aplicación web en JSP, ya que es este tipo de aplicaciones la finalidad última de nuestra tarea. En el Anexo II (página ) disponemos del fichero correspondiente. Debemos situarlo bajo "%CATALINA\_HOME%\webapps\practica", que es la ruta que hemos establecido en "server.xml" como raíz de nuestras aplicaciones.

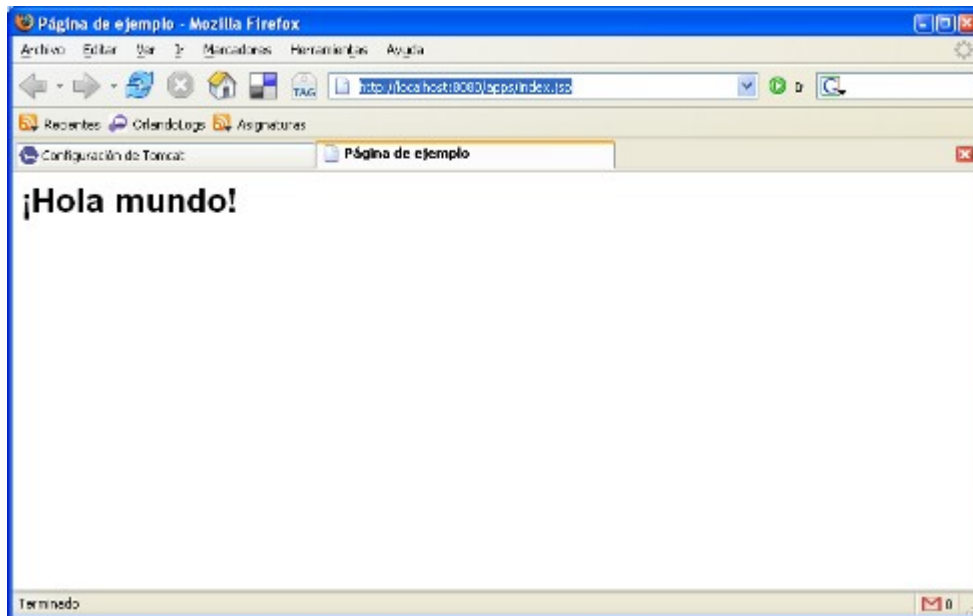


Ilustración 1: Ejecución de aplicación básica

## Instalación del driver JDBC

---

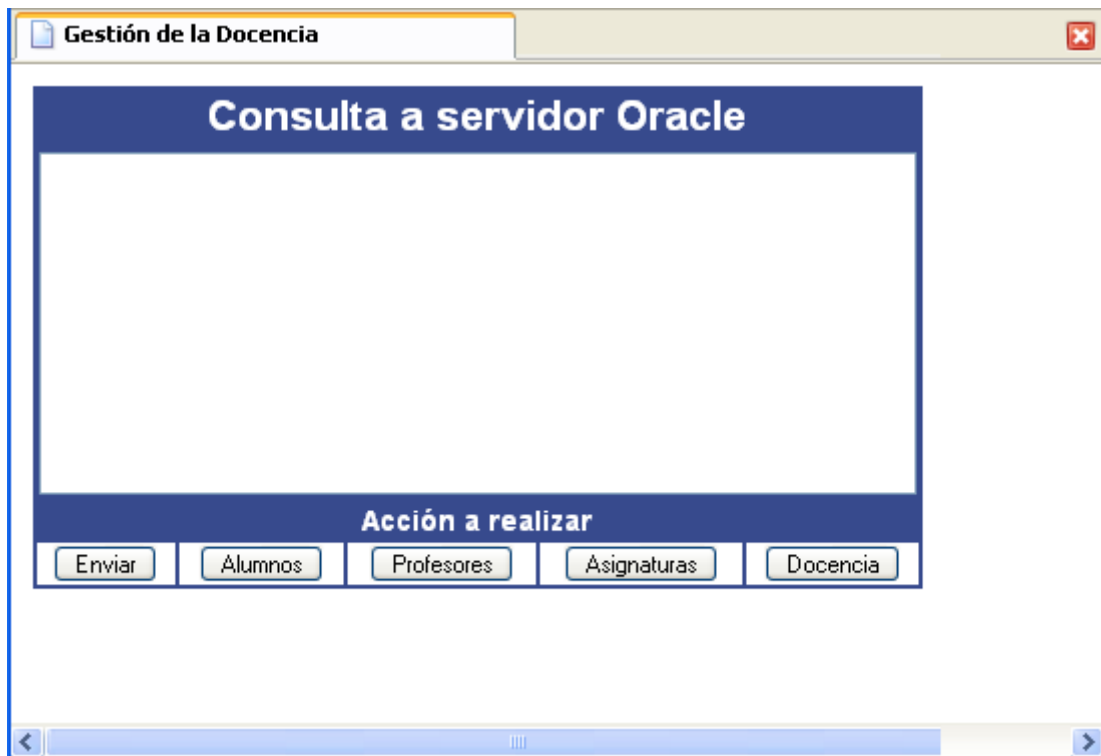
Para acabar, procedemos a instalar el driver *Oracle* para *JDBC*. Existen dos posibilidades, instalarlo como librería de la máquina virtual de *Java* o ponerlo como recurso único para el contexto que nos ocupa. Para nuestros propósitos basta con la segunda.

El primer paso, y único, es crear el árbol de directorios "WEB-INF/lib" bajo el directorio que fijamos como "docBase" en contexto del servidor y copiar el fichero \*.jar correspondiente al driver a esa nueva ubicación.

Para utilizarlo desde una *JavaServer Page* basta con incluir en nuestro código script un fragmento como el siguiente:

```
try {
    // Cargamos el driver JDBC para Oracle
    Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch(ClassNotFoundException e) {
    System.out.println("Driver no encontrado.");
    System.out.println(e.toString());
    throw new UnavailableException(this, "Clase no encontrada.");
}
```

Y así podremos tener acceso a nuestra base de datos remota desde nuestro servicio web y diseñar aplicaciones como la siguiente:



*Ilustración 2: Aplicación que con la configuración actual, usa JDBC*

# Anexo I

---

Fichero de configuración "server.xml" creado para la demo. Su funcionamiento ha sido probado.

## SERVER.XML

```
<Server port="8005" shutdown="SHUTDOWN" debug="0">

  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"
    debug="0"/>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"
    debug="0"/>

  <GlobalNamingResources>

    <Environment name="simpleValue" type="java.lang.Integer" value="30"/>

    <Resource name="UserDatabase" auth="Container"
      type="org.apache.catalina.UserDatabase"
      description="User database that can be updated and saved">
    </Resource>
    <ResourceParams name="UserDatabase">
      <parameter>
        <name>factory</name>
        <value>org.apache.catalina.users.MemoryUserDatabaseFactory</value>
      </parameter>
      <parameter>
        <name>pathname</name>
        <value>conf/tomcat-users.xml</value>
      </parameter>
    </ResourceParams>

  </GlobalNamingResources>

  <Service name="Tomcat-Standalone">

    <!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8080" minProcessors="5" maxProcessors="75"
      enableLookups="true" redirectPort="8443"
      acceptCount="100" debug="0" connectionTimeout="20000"
      useURIVValidationHack="false" disableUploadTimeout="true" />

    <!-- Define the top level container in our container hierarchy -->
    <Engine name="Standalone" defaultHost="localhost" debug="0">

      <Logger className="org.apache.catalina.logger.FileLogger"
        prefix="catalina_log." suffix=".txt"
        timestamp="true"/>

      <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
        debug="0" resourceName="UserDatabase"/>

      <!-- Define the default virtual host -->
      <Host name="localhost" debug="0" appBase="webapps"
        unpackWARs="true" autoDeploy="true">

        <Logger className="org.apache.catalina.logger.FileLogger"
          directory="logs" prefix="localhost_log." suffix=".txt"
          timestamp="true"/>

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

```

<!-- Tomcat Examples Context -->
<Context path="/apps" docBase="practica" debug="0"
  reloadable="true" crossContext="true">
  <Logger className="org.apache.catalina.logger.FileLogger"
    prefix="apps_practica." suffix=".txt"
    timestamp="true"/>
  <Ejb name="ejb/EmplRecord" type="Entity"
    home="com.wombat.empl.EmployeeRecordHome"
    remote="com.wombat.empl.EmployeeRecord"/>

  <Environment name="maxExemptions" type="java.lang.Integer"
    value="15"/>
  <Parameter name="context.param.name" value="context.param.value"
    override="false"/>
  <Resource name="jdbc/EmployeeAppDb" auth="SERVLET"
    type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/EmployeeAppDb">
    <parameter><name>username</name><value>sa</value></parameter>
    <parameter><name>password</name><value></value></parameter>
    <parameter><name>driverClassName</name>
      <value>org.hsqldb.jdbcDriver</value></parameter>
    <parameter><name>url</name>
      <value>jdbc:HyperSQL:database</value></parameter>
  </ResourceParams>
  <Resource name="mail/Session" auth="Container"
    type="javax.mail.Session"/>
  <ResourceParams name="mail/Session">
    <parameter>
      <name>mail.smtp.host</name>
      <value>localhost</value>
    </parameter>
  </ResourceParams>
  <ResourceLink name="linkToGlobalResource"
    global="simpleValue"
    type="java.lang.Integer"/>
</Context>

</Host>

</Engine>

</Service>
</Server>

```

## Anexo II

---

Fichero de demo, "index.jsp". Su funcionamiento ha sido probado en apartados anteriores:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2 <html>
3 <head>
4   <meta content="text/html; charset=ISO-8859-1"
5     http-equiv="content-type">
6   <title>P&aacute;gina de ejemplo</title>
7 </head>
8 <body>
9 <% out.println("<hl>¡Hola mundo!</hl>"); %>
10 </ul>
11 </body>
12 </html>
```