



Universidad
Andrés Bello®
Conectar • Innovar • Liderar

DESARROLLO DE APLICACIONES WEB DINÁMICAS JAVA

Personalizando el descriptor de despliegue web.xml

Las aplicaciones web de Java usan un archivo descriptor de implementación para determinar cómo se asignan las URL a los servlets, qué URL requieren autenticación y más información. Este archivo se llama web.xml y se encuentra en el WAR de la aplicación dentro del directorio WEB-INF/. web.xml es parte del estándar del servlet para aplicaciones web.

Acerca de los descriptores de implementación

Un descriptor de implementación de una aplicación web describe las clases, los recursos y la configuración de la aplicación y cómo los usa el servidor web para entregar solicitudes web. Cuando el servidor web recibe una solicitud para la aplicación, usa el descriptor de implementación con el fin de asignar la URL de la solicitud al código que debe controlarla.

El descriptor de implementación es un archivo llamado web.xml. Se encuentra en el WAR de la app en el directorio WEB-INF/. Es un archivo XML cuyo elemento raíz es <web-app>.

A continuación, se muestra un ejemplo simple de web.xml que asigna todas las rutas de URL (/*) a la clase de servlet mysite.server.ComingSoonServlet:

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1">
  <servlet>
    <servlet-name>comingsoon</servlet-name>
    <servlet-class>mysite.server.ComingSoonServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>comingsoon</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Servlets y rutas de URL

El archivo web.xml define las asignaciones entre las rutas de URL y los servlets que controlan las solicitudes con esas rutas. El servidor web usa esta configuración para identificar el servlet que controlará una solicitud determinada y llamar al método de clase que corresponde al método de la solicitud (p. ej., el método doGet() para solicitudes HTTP GET).

Para asignar una URL a un servlet, decláralo con el elemento <servlet> y define una asignación desde una ruta de URL a una declaración de servlet con el elemento <servlet-mapping>.

El elemento <servlet> declara el servlet, incluido un nombre usado por otros elementos en el archivo para referirse al servlet, la clase que se usará en este y los parámetros de inicialización. Puedes declarar varios servlets con la misma clase, pero con diferentes parámetros de inicialización. El nombre de cada servlet debe ser único en todo el descriptor de implementación.

```
<servlet>
  <servlet-name>redteam</servlet-name>
  <servlet-class>mysite.server.TeamServlet</servlet-class>
  <init-param>
    <param-name>teamColor</param-name>
    <param-value>red</param-value>
  </init-param>
  <init-param>
    <param-name>bgColor</param-name>
    <param-value>#CC0000</param-value>
  </init-param>
</servlet>

<servlet>
  <servlet-name>blueteam</servlet-name>
  <servlet-class>mysite.server.TeamServlet</servlet-class>
  <init-param>
    <param-name>teamColor</param-name>
    <param-value>blue</param-value>
  </init-param>
  <init-param>
    <param-name>bgColor</param-name>
    <param-value>#0000CC</param-value>
  </init-param>
</servlet>
```

El elemento <servlet-mapping> especifica un patrón de URL y el nombre de un servlet declarado para usar en solicitudes cuyas URL coincidan con el patrón. El patrón de URL puede usar un asterisco (*) al principio o al final del patrón para indicar cero o más de cualquier carácter. (El estándar no admite comodines en medio de una string y no permite múltiples comodines en un patrón). El patrón coincide con la ruta completa de la URL, incluida la barra inicial (/) que sigue al nombre del dominio.

JSP

Una aplicación puede usar JavaServer Pages (JSP) para implementar páginas web. Los JSP son servlets definidos mediante contenido estático (como HTML) mezclado con código Java.

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

App Engine admite la compilación automática y la asignación de URL para JSP. Un archivo JSP en el WAR de la aplicación (fuera de WEB-INF/) cuyo nombre de archivo termina en .jsp se compila en una clase de servlet de manera automática y se asigna a la ruta de URL equivalente a la ruta del archivo JSP de la raíz del WAR. Por ejemplo, si una aplicación tiene en su WAR un archivo JSP con el nombre start.jsp en un subdirectorío llamado register/, App Engine lo compila y lo asigna a la ruta de URL /register/start.jsp.

```
<servlet-mapping>
  <servlet-name>redteam</servlet-name>
  <url-pattern>/red/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>blueteam</servlet-name>
  <url-pattern>/blue/*</url-pattern>
</servlet-mapping>
```

Para tener más control sobre cómo se asigna el JSP a una URL, puedes especificar la asignación de forma explícita si la declaras con un elemento `<servlet>` en el descriptor de implementación. En lugar de un elemento `<servlet-class>`, especifica un elemento `<jsp-file>` con la ruta del archivo JSP desde la raíz del WAR. El elemento `<servlet>` para el JSP puede contener parámetros de inicialización.

```
<servlet>
  <servlet-name>register</servlet-name>
  <jsp-file>/register/start.jsp</jsp-file>
</servlet>

<servlet-mapping>
  <servlet-name>register</servlet-name>
  <url-pattern>/register/*</url-pattern>
</servlet-mapping>
```

Lista de archivos de bienvenida

Cuando las URL de tu sitio representan rutas a archivos estáticos o JSP en tu WAR, a menudo es una buena idea que las rutas a los directorios también hagan algo útil. Un usuario que visita la ruta de URL `/help/accounts/password.jsp` para obtener información sobre las contraseñas de la cuenta puede intentar visitar `/help/accounts/` con el fin de encontrar una página que presente la documentación del sistema sobre la cuenta.

El descriptor de implementación puede especificar una lista de nombres de archivo que el servidor debe probar cuando el usuario accede a una ruta que representa un subdirectorio del WAR (que no esté asignado de manera explícita a un servlet). El estándar de servlet llama a esto la "lista de archivos de bienvenida".

Por ejemplo, si el usuario accede a la ruta de URL `/help/accounts/`, el siguiente elemento `<welcome-file-list>` del descriptor de implementación le indica al servidor que debe verificar `help/accounts/index.jsp` y `help/accounts/index.html` antes de informar que la URL no existe.

Los filtros te permiten crear tareas de procesamiento de solicitudes desde el descriptor de implementación.

Una clase de filtro implementa la interfaz `javax.servlet.Filter`, incluido el método `doFilter()`. A continuación, se muestra una implementación de filtro simple que registra un mensaje y pasa el control a lo largo de la cadena, que puede incluir otros filtros o un servlet, como lo indica el descriptor de implementación:

```
package mysite.server;

import java.io.IOException;
import java.util.logging.Logger;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class LogFilterImpl implements Filter {

    private FilterConfig filterConfig;
    private static final Logger log
        = Logger.getLogger(LogFilterImpl.class.getName());

    public void doFilter(ServletRequest request,
        ServletResponse response,
        FilterChain filterChain)
        throws IOException, ServletException {
        log.warning("Log filter processed a "
            + getFilterConfig().getInitParameter("logType")
            + " request");

        filterChain.doFilter(request, response);
    }

    public FilterConfig getFilterConfig() {
        return filterConfig;
    }

    public void init(FilterConfig filterConfig) {
        this.filterConfig = filterConfig;
    }

    public void destroy() {
    }
}
```

Controladores de errores

Por medio del descriptor de implementación, puedes personalizar lo que el servidor envía al usuario cuando se produce un error. El servidor puede mostrar una ubicación de página alternativa cuando está a punto de enviar un código de estado HTTP particular o cuando un servlet genera una excepción de Java específica.

El elemento `<error-page>` contiene un elemento `<error-code>` con un valor de código de error HTTP (como 500) o un elemento `<exception-type>` con el nombre de la clase de excepción esperada (como `java.io.IOException`). También contiene un elemento `<location>` que incluye la ruta de URL del recurso que se muestra cuando se produce el error.

```
<error-page>
  <error-code>500</error-code>
  <location>/errors/servererror.jsp</location>
</error-page>
```

Conociendo el servidor Tomcat

¿Qué es tomcat ?

Apache Tomcat es un contenedor y gestor de servlet, tiene soporte para páginas JSPs.

Mientras que los servlets se encargan de procesar las peticiones que recibe del cliente, por ejemplo, validar el email de un usuario, las páginas JSP permiten la presentación de información en el navegador.

Cuenta con todas las especificaciones técnicas para estas dos tecnologías (Servlet's y JSP), también es conocido como servidor de aplicaciones. Tomcat permite desplegar aplicaciones construidas con tecnología Java EE, básicamente se encarga gestionar temas como seguridad, gestión de ciclo de vida de un servlet, concurrencia, procesamiento de transacciones entre otros servicios.

Configurando un entorno de desarrollo web con Java


Entorno Java Web = Jdk + IDE + Servidor de Aplicaciones/Contenedor de Servlet.
Dicho esto tú puedes configurar cualquier servidor de los mencionados anteriormente con cualquier IDE con el que tú programes, bien sea Eclipse o Netbeans.

Aclarar que los servidores mencionados son lo más comunes y gratuitos, aunque existen otros de pago, pero aprendiendo estos tranquilamente puedes montar tu entorno de desarrollo web, y no sólo en desarrollo, sino también en entornos de producción.

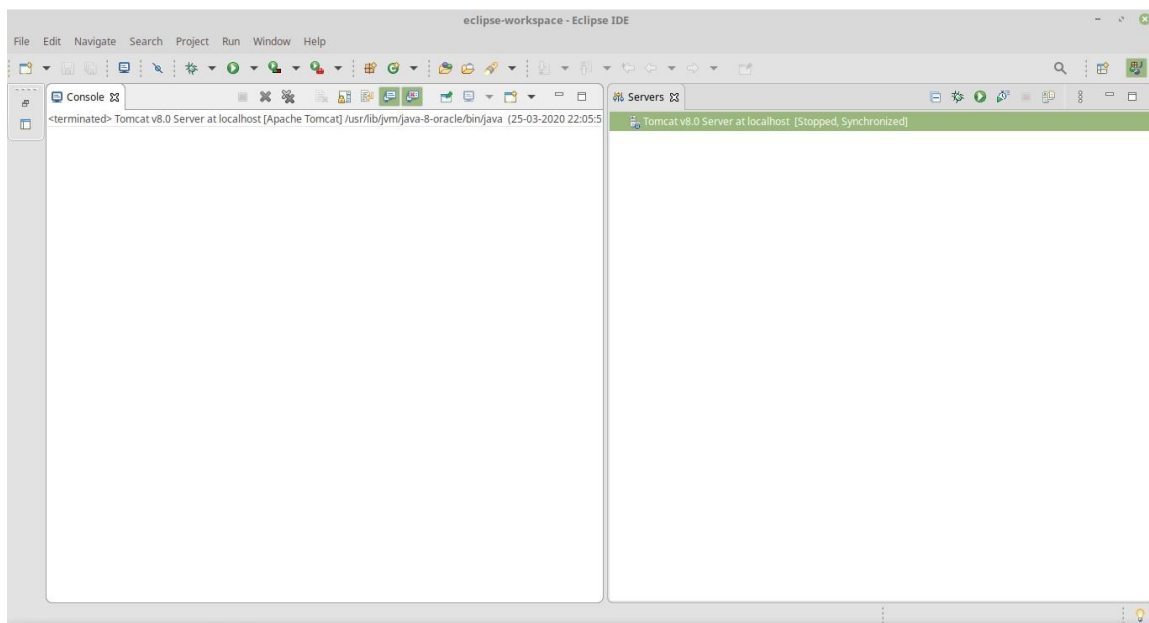
Iniciar Tomcat

Para iniciar el servicio de Tomcat solo tienes que presionar el botón con la flecha verde, que aparece en la ventana de servlet.

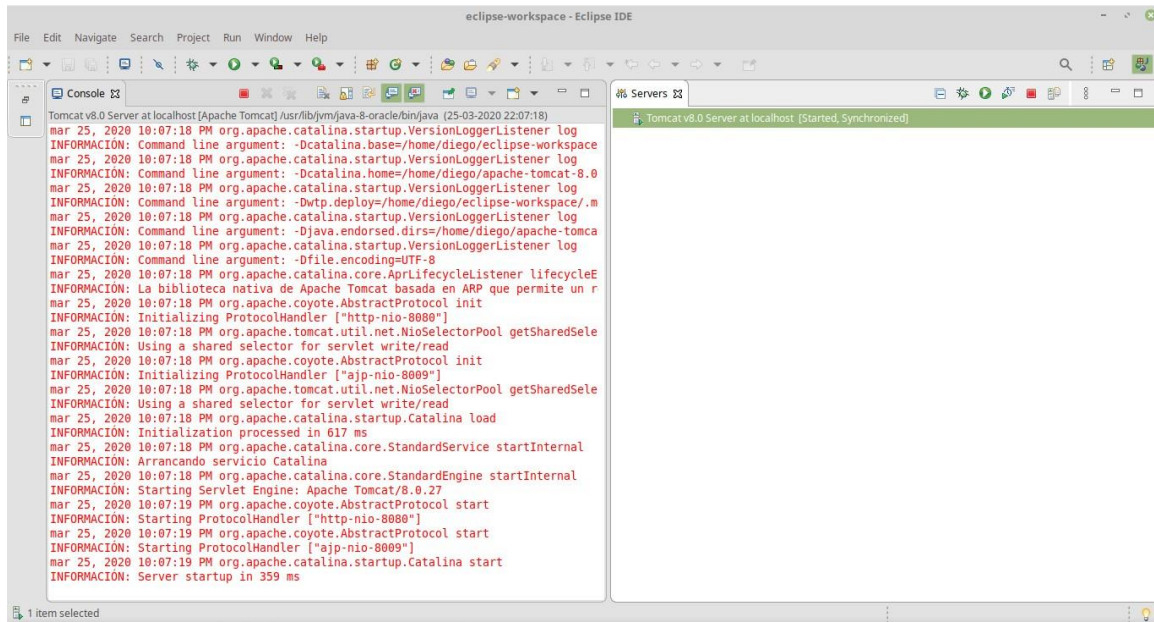
Nota:

Windows  Show view  

Windows  Show view  



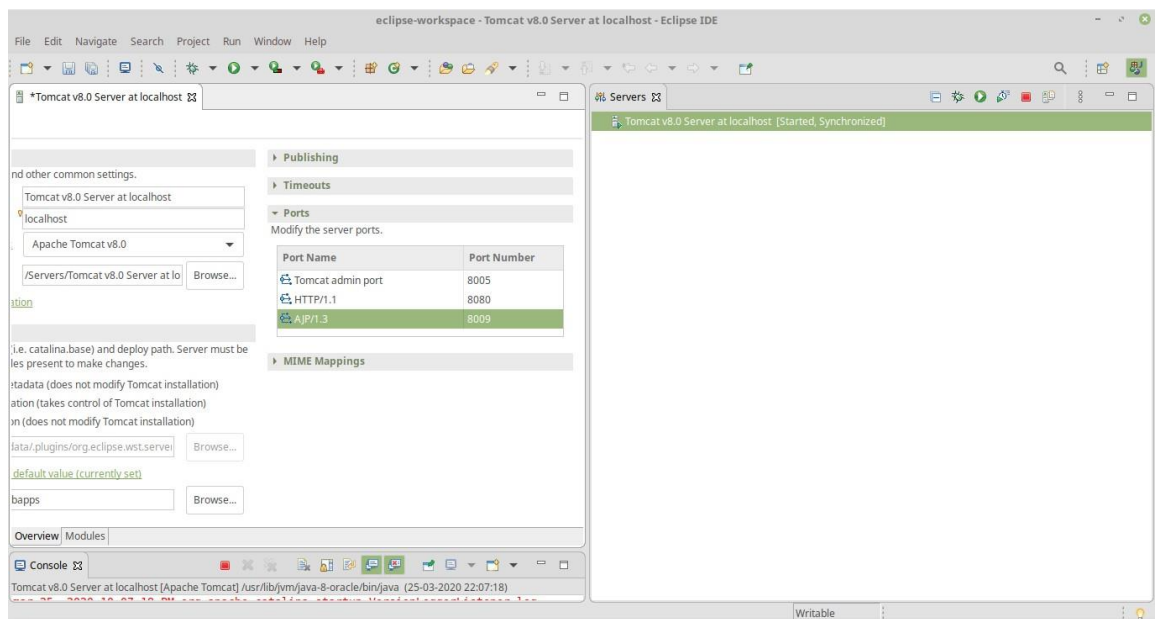
Una vez iniciado:



Cambiar de puerto

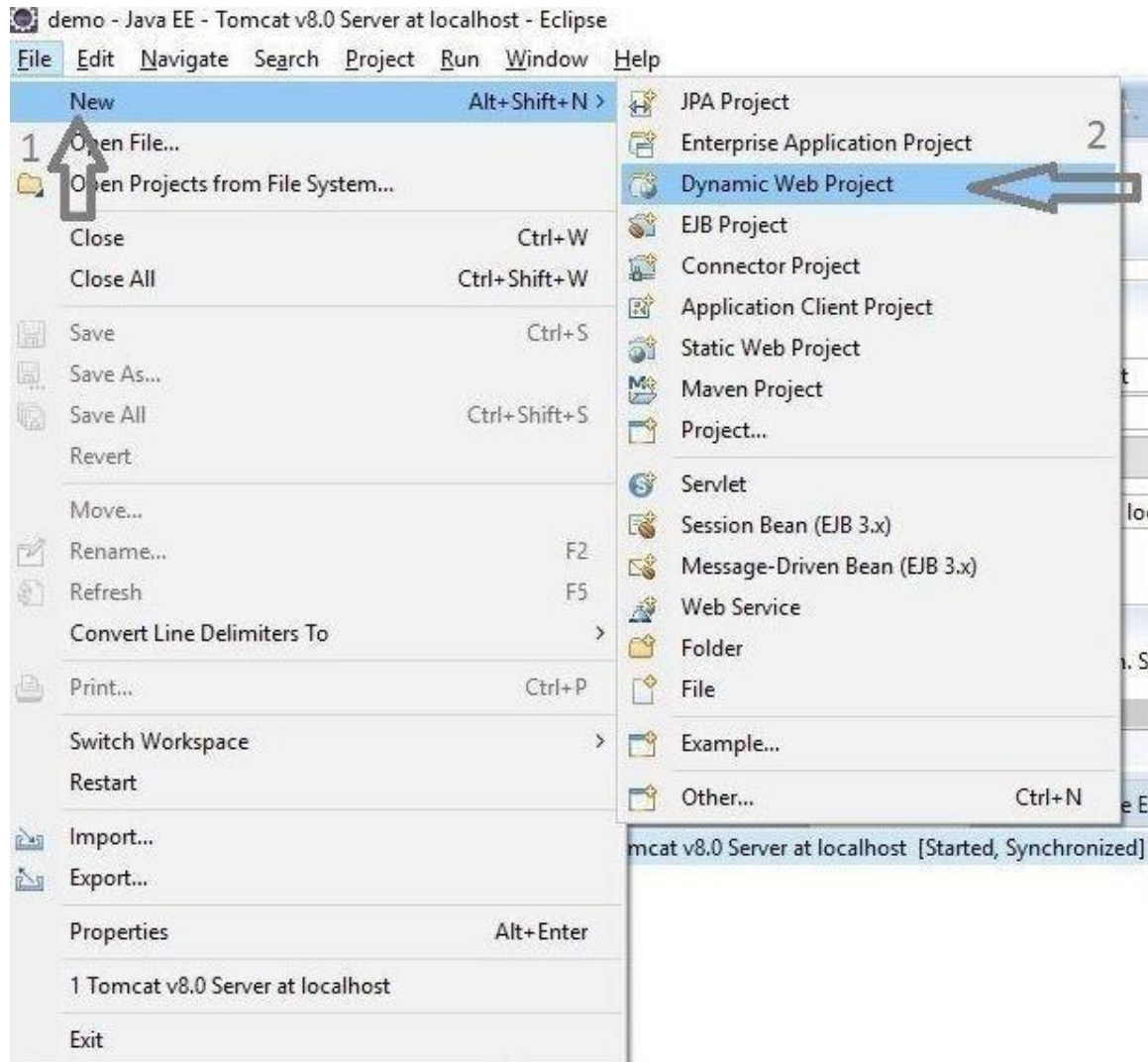
Apache Tomcat utiliza el puerto 8080 por defecto, pero hay veces que este puerto ya está ocupado por alguna otra aplicación, si fuera ese el caso, debes cambiar el puerto en la pestaña Ports como se ve en la imagen, cambias y luego guardas los cambios con la tecla CTRL + S.

Para que te aparezca la pantalla como se ve en la imagen, primero en la pestaña Servers, das doble click sobre Tomcat.

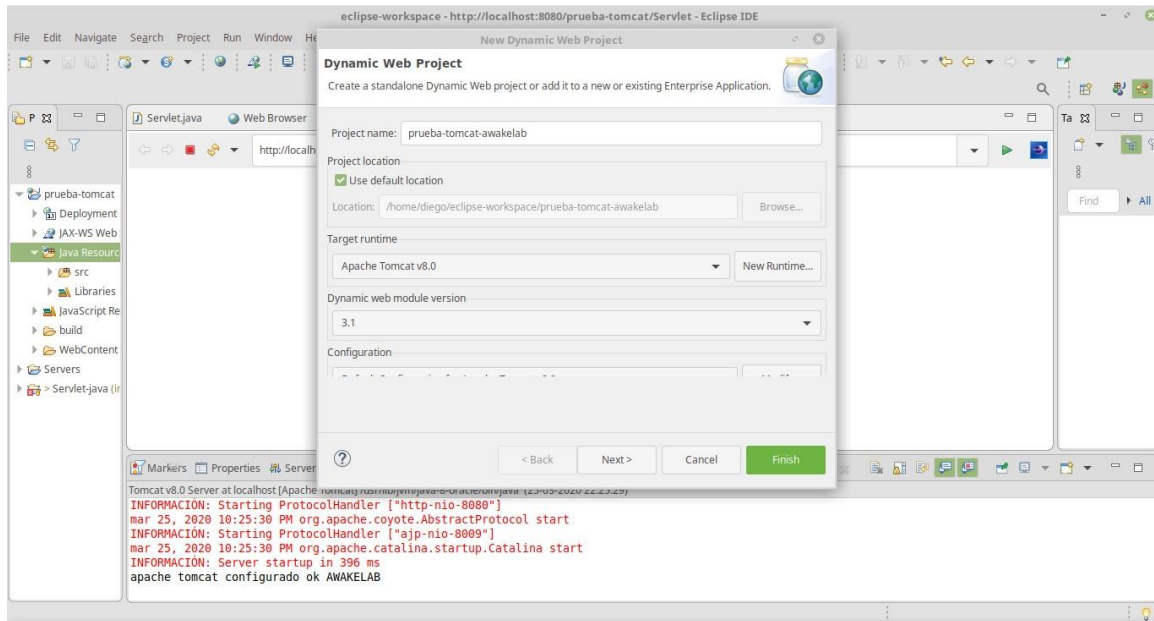


Despliegue de una aplicación web con Tomcat Manager

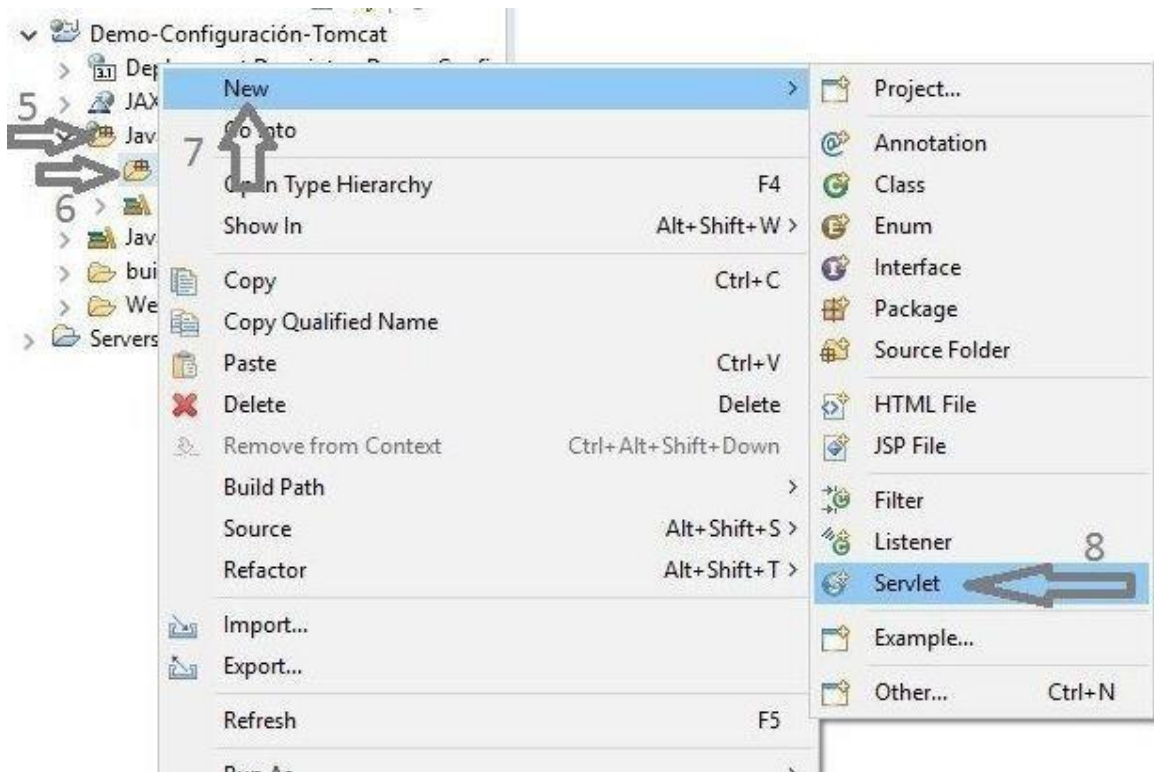
Paso 1: En Eclipse creamos un proyecto nuevo, del tipo Dynamic Web Project.



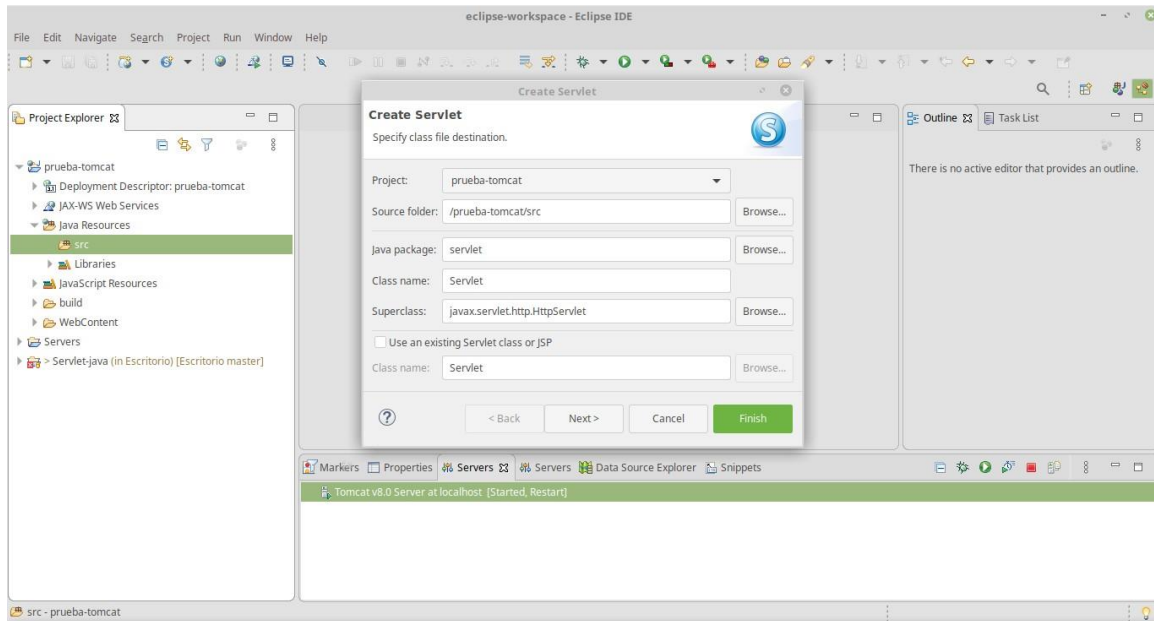
Paso 2: Le das un nombre al proyecto como se ve en la imagen y le das click al botón Finish.



Paso 3: Vas a la opción Java Source del proyecto, luego a la opción src y das click derecho, New y eliges la opción Servlet, como se muestra en la imagen siguiente.



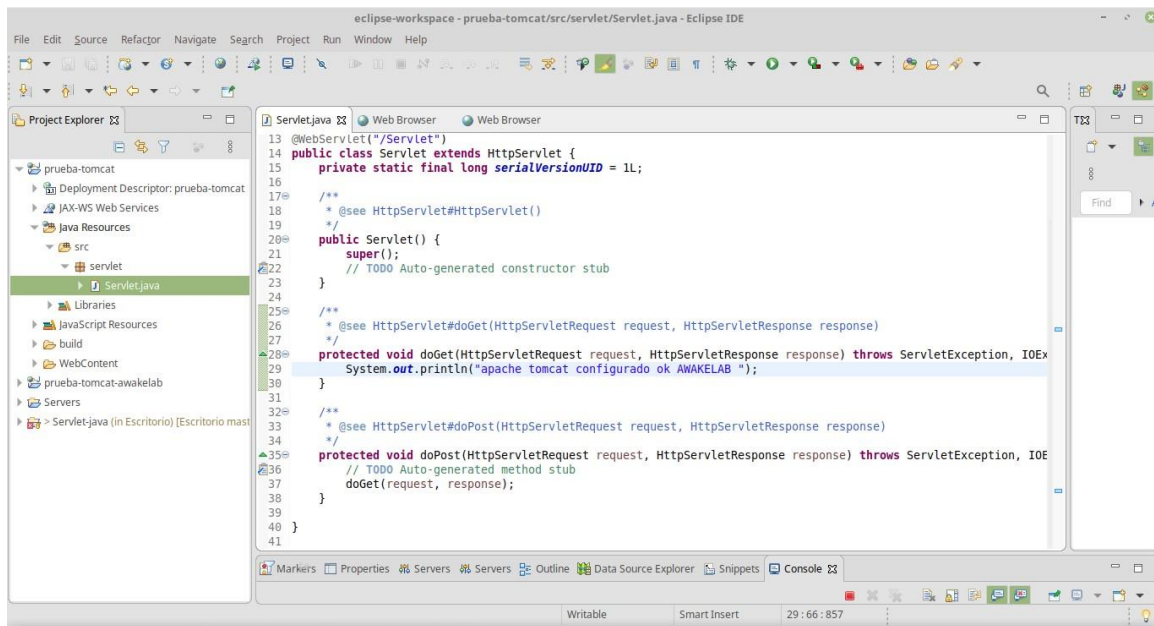
Paso 4: En la siguiente pantalla te va pedir que llenes datos como es: el nombre del paquete, y el nombre del Servlet, debe quedar como está en la imagen.



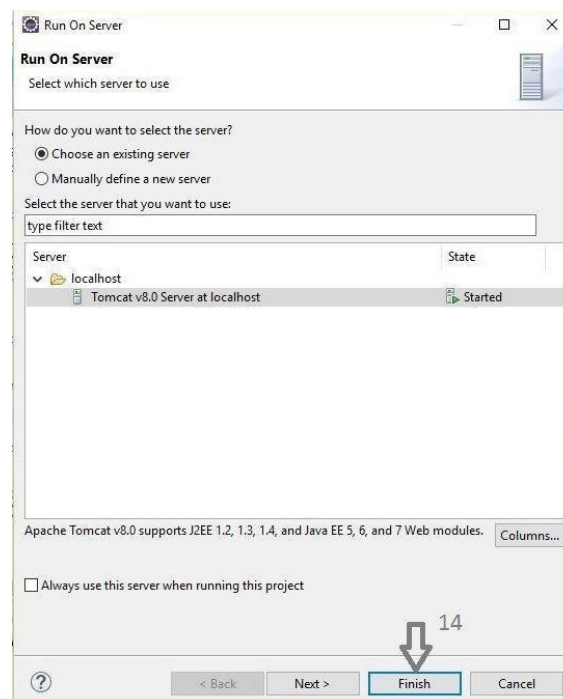
Paso 5: Te mostrará el Servlet creado. Lo que vas hacer es eliminar la parte seleccionada como se ve en la imagen.



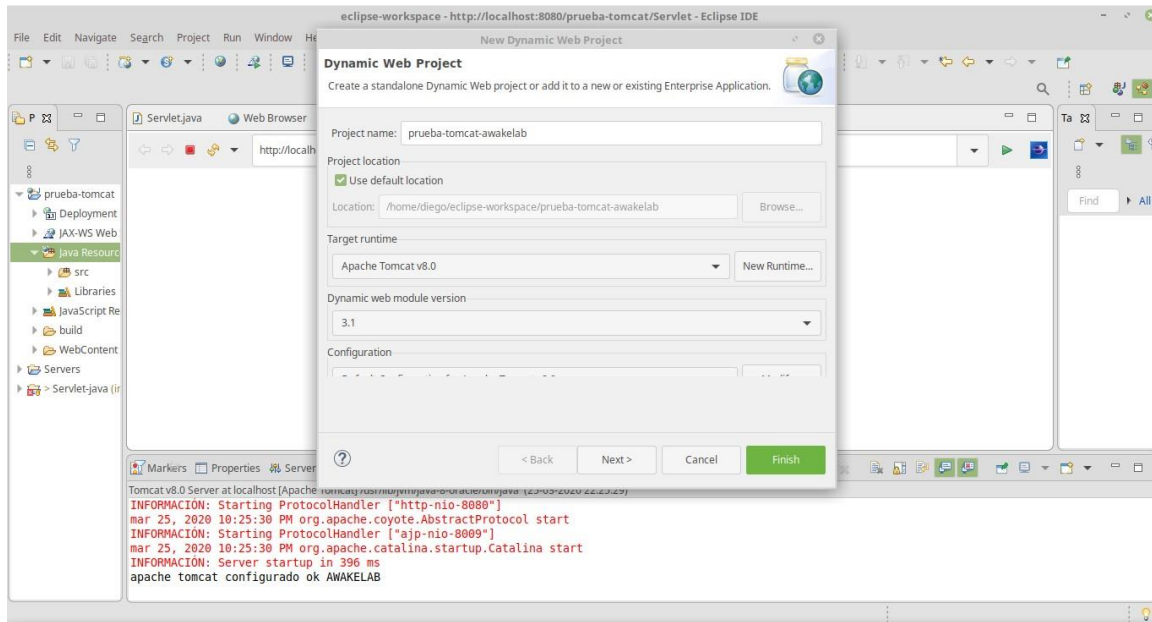
Paso 6: Debes poner la línea de código, la que imprimirá por consola un mensaje y validará la configuración de Apache Tomcat 8. Finalmente en la parte superior encuentra una flecha de color verde, la más grande, le vas a dar click para iniciar el proyecto.



Paso 7: En esta ventana te dice que debes escoger el servidor en el que se va ejecutar la aplicación das click en Finish.



Paso 8: Por último tendrás una pantalla como la que se muestra a continuación con el mensaje que mandamos a imprimir desde el Servlet, indicando que la configuración de Apache Tomcat es correcta.

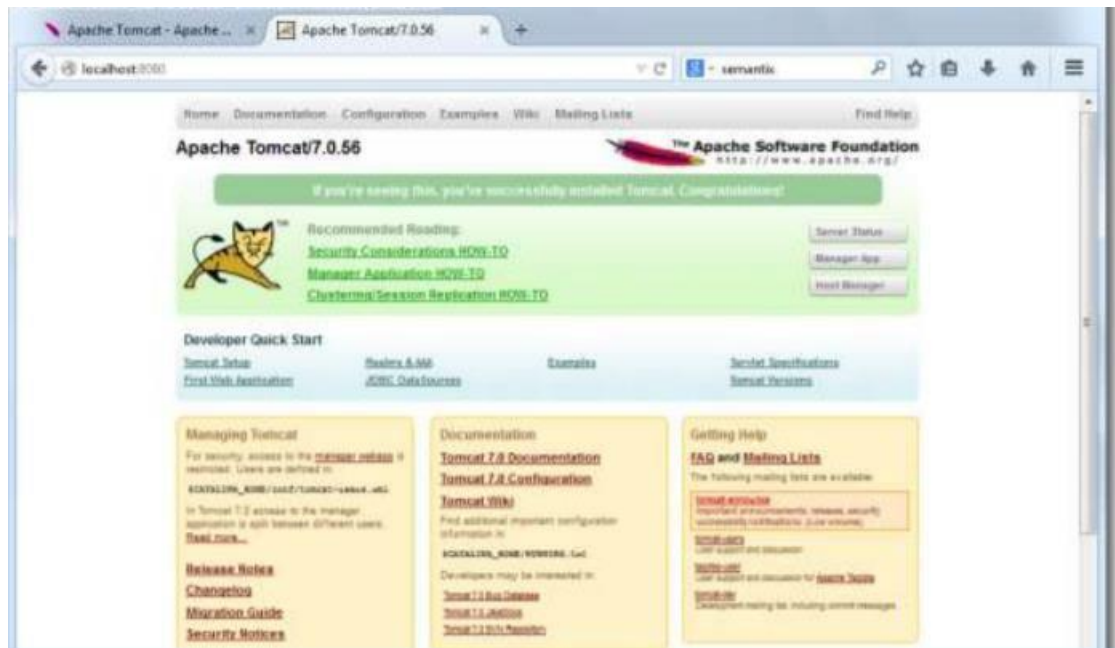


Administrando las aplicaciones instaladas

Interfaz de administración de Tomcat

Una vez que se ha instalado Tomcat en un equipo local, a fin de probar la instalación, ya sea en Windows o en Unix, la forma más directa es abrir una ventana del navegador y teclear la URL: <http://localhost:8080/>. El puerto por defecto es el 8080, sin embargo es probable que haya debido seleccionar otro por conflictos con otros programas anteriores; de ser así, usa el puerto indicado en la intalación.

Deberá aparecer la siguiente pantalla.



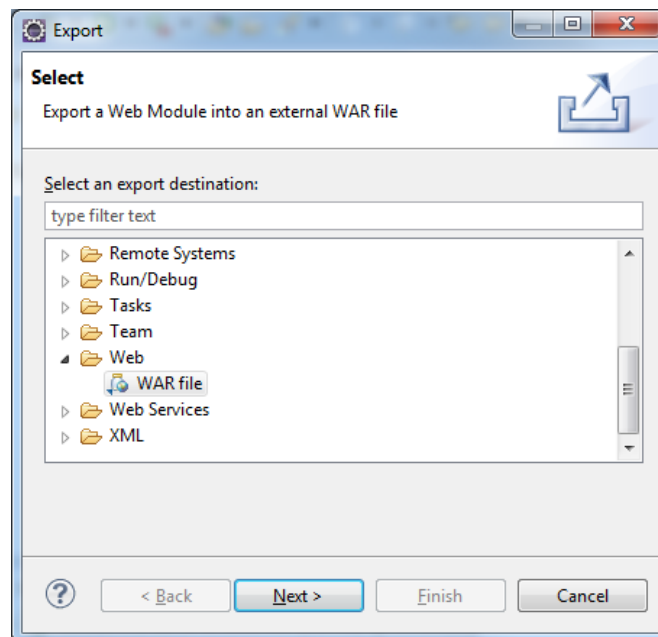
Presionando el botón “Manager App”, se podrá acceder al listado de aplicaciones en ejecución. Para ingresar a este apartado se le solicitará un usuario y clave, indicados al momento de instalar Tomcat. En este gestor es posible detener aplicaciones, eliminarlas, o bien agregar otras.

Además, accediendo a “Examples -> Servlets examples” y “Examples -> JSP examples” se puede comprobar que funciona correctamente como contenedor (sólo si se han instalado los ejemplos de servlets y JSP).

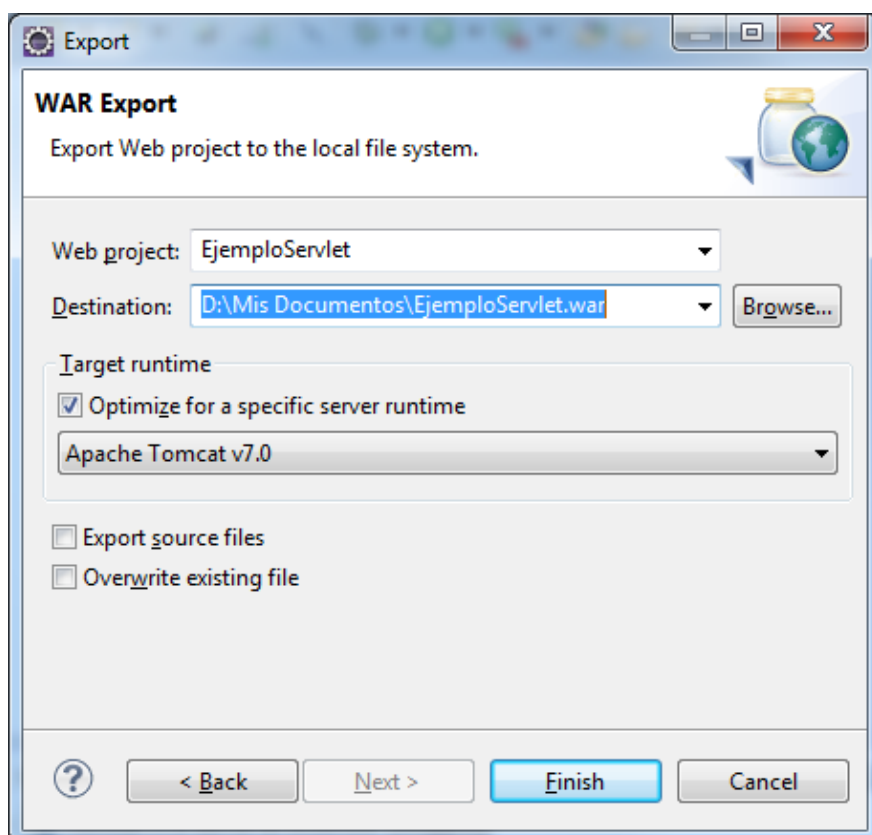
Generación de un archivo ejecutable

Si dispones de una aplicación web creada en Eclipse para el contenedor de Servlets Tomcat, debes seguir los siguientes pasos para llevar la aplicación al servidor que se encuentre en producción.

Seleccionando el proyecto que deseas desplegar, usa el menú File > Export > Web > WAR file.



Específica a continuación la ruta donde deseas guardar la aplicación en el cuadro Destination.



Finalmente debes copiar el archivo .WAR que se ha generado, en la carpeta webapps donde se encuentre instalado Apache Tomcat. Puede ser algo como: C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps.

Los archivos WAR (Web application ARchive) realmente son como los archivo JAR usado por Java, pero en este caso son utilizados para distribuir aplicaciones web que pueden contener páginas JSP, Servlets, clases Java, páginas web estáticas, y cualquier otro recurso utilizado por la aplicación web. Puedes ver su contenido con un gestor de archivos comprimidos.

Una vez situado el archivo en la carpeta webapps, puedes intentar abrir la aplicación desde el navegador web indicando la dirección y puerto de entrada al servidor, seguido del nombre asignado al proyecto de la aplicación y el nombre del archivo que contiene la página web, si no es index.html. Por ejemplo:


`http://localhost:8080/miAplicacion/pagina.html`

Despliegue de aplicaciones desde el gestor de aplicaciones

También es posible instalar las aplicaciones web con formato WAR desde el gestor de aplicaciones de Tomcat al que se puede acceder desde la página principal del servidor (`http://localhost:8080`), usando el botón Manage App.


[Home](#) [Documentation](#) [Configuration](#) [Examples](#) [Wiki](#) [Mailing Lists](#) [Find Help](#)

Apache Tomcat/9.0.36



APACHE® SOFTWARE FOUNDATION
<http://www.apache.org/>

If you're seeing this, you've successfully installed Tomcat. Congratulations!



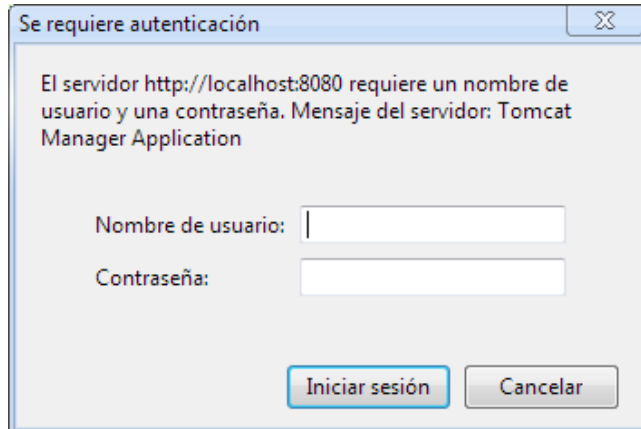
Recommended Reading:
[Security Considerations How-To](#)
[Manager Application How-To](#)
[Clustering/Session Replication How-To](#)

[Server Status](#)
[Manager App](#)
[Host Manager](#)

Developer Quick Start

[Tomcat Setup](#) [Building & JARs](#) [Examples](#) [Servlet Specifications](#)

Al intentar acceder te solicitará un nombre de usuario y contraseña que disponga de permisos para acceder al gestor.



Si no dispones de un usuario configurado para esto, observarás las instrucciones necesarias en la página que aparecerá cuando no hayas podido acceder.

401 Unauthorized

You are not authorized to view this page. 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 to let you use this webapp.

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


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

Te indica que debes registrar en el archivo de configuración `tomcat-users.xml`, un usuario con perfil de `manager-gui`. Para ello puedes usar un estructura similar a la que pone de ejemplo:


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

Tras guardar las modificaciones y reiniciar el servidor, debes poder acceder a la página del gestor de aplicaciones con el usuario que hayas declarado.

Obtendrás una página similar a la siguiente:



The Apache Software Foundation
http://www.apache.org/



Gestor de Aplicaciones Web de Tomcat

Mensaje: OK - Recargada aplicación en trayectoria de contexto /EjemploServlet

Gestor

[Listar Aplicaciones](#)
[Ayuda HTML de Gestor](#)
[Ayuda de Gestor](#)
[Estado de Servidor](#)

Aplicaciones

Trayectoria	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado	Welcome to Tomcat	true	0	<div> <div>Arrancar</div> <div>Parar</div> <div>Recargar</div> <div>Replegar</div> </div> <div> <div>Expirar sesiones</div> <div>sin trabajar ≥ 30 minutos</div> </div>
/EjemploServlet	Ninguno especificado		true	0	<div> <div>Arrancar</div> <div>Parar</div> <div>Recargar</div> <div>Replegar</div> </div> <div> <div>Expirar sesiones</div> <div>sin trabajar ≥ 30 minutos</div> </div>
/docs	Ninguno especificado	Tomcat Documentation	true	0	<div> <div>Arrancar</div> <div>Parar</div> <div>Recargar</div> <div>Replegar</div> </div> <div> <div>Expirar sesiones</div> <div>sin trabajar ≥ 30 minutos</div> </div>
/examples	Ninguno especificado	Servlet and JSP Examples	true	0	<div> <div>Arrancar</div> <div>Parar</div> <div>Recargar</div> <div>Replegar</div> </div> <div> <div>Expirar sesiones</div> <div>sin trabajar ≥ 30 minutos</div> </div>
/host-manager	Ninguno especificado	Tomcat Host Manager Application	true	0	<div> <div>Arrancar</div> <div>Parar</div> <div>Recargar</div> <div>Replegar</div> </div> <div> <div>Expirar sesiones</div> <div>sin trabajar ≥ 30 minutos</div> </div>
/manager	Ninguno especificado	Tomcat Manager Application	true	1	<div> <div>Arrancar</div> <div>Parar</div> <div>Recargar</div> <div>Replegar</div> </div> <div> <div>Expirar sesiones</div> <div>sin trabajar ≥ 30 minutos</div> </div>

Desplegar

Desplegar directorio o archivo WAR localizado en servidor

Trayectoria de Contexto (opcional):
URL de archivo de Configuración XML:
URL de WAR o Directorio:

Desplegar

Archivo WAR a desplegar

Selecione archivo WAR a cargar

Seleccionar archivo

No se ha seleccionado ningún archivo

Desplegar

Diagnósticos

Revisa a ver si una aplicación web ha causado fallos de memoria al parar, recargar o replegarse.

Halla fallos de memoria

Este chequeo de diagnóstico disparará una colección completa de basura. Utilízalo con extremo cuidado en sistemas en producción.

Información de Servidor

Versión de Tomcat	Versión JVM	Vendedor JVM	Nombre de SO	Versión de SO	Arquitectura de SO	NombreDeMáquina	Dirección IP
Apache Tomcat/7.0.34	1.7.0_11-b21	Oracle Corporation	Windows 7	6.1	amd64	ASUS-N61	192.168.56.1

Copyright © 1999-2012, Apache Software Foundation

Observa que aparece una lista de las aplicaciones que ya se encuentren instaladas. Puedes desinstalar cualquiera de ellas usando el botón Replegar, o instalar nuevas aplicaciones usando el botón Desplegar que puedes encontrar más abajo, seleccionando previamente el archivo WAR correspondiente con el botón Seleccionar archivo.

Anexo: Referencias

[1] Qué es Java Enterprise (J2ee, JEE)

Referencia: <https://www.fundesem.es/bt/publicacion-java-ee-y-el-desarrollo-web-un-enfoque-de-aprendizaje>

[2] Qué son los Servlets

Referencia: <https://users.dcc.uchile.cl/~jbarrios/servlets/general.html>

[3] Utilizando métodos post y get

Referencia: <https://www.ecodeup.com/enviar-parametros-servlet-desde-una-vista-utilizando-post-get/>

[4] Sesiones y cookies

Referencia: <https://ricardogeek.com/manejo-de-sesiones-y-cookies-en-servlets-java/>

[5] Etiquetas JSTL

Referencia: <http://www.itech.ua.es/ayto/ayto2008/jsp/sesion07-apuntes.html>

[6] MVC

Referencia: <https://www.ecodeup.com/patrones-de-diseno-en-java-mvc-dao-y-dto/>

[7] Patrón Singleton

Referencia: <https://reactiveprogramming.io/blog/es/patrones-de-diseno/singleton>

[8] Personalizando el descriptor de despliegue web.xml

Referencia: <https://cloud.google.com/appengine/docs/flexible/java/configuring-the-web-xml-deployment-descriptor?hl=es-419>

[9] Configuración de un servlet

Referencia: <https://www.ecodeup.com/integra-apache-tomcat-en-eclipse-neon-menos-5-minutos/>

[10] Despliegue de aplicaciones Web en Tomcat

Referencia: <https://javiergarciaescobedo.es/despliegue-de-aplicaciones-web/86-servidores-de-aplicaciones/312-despliegue-de-aplicaciones-web-en-tomcat>