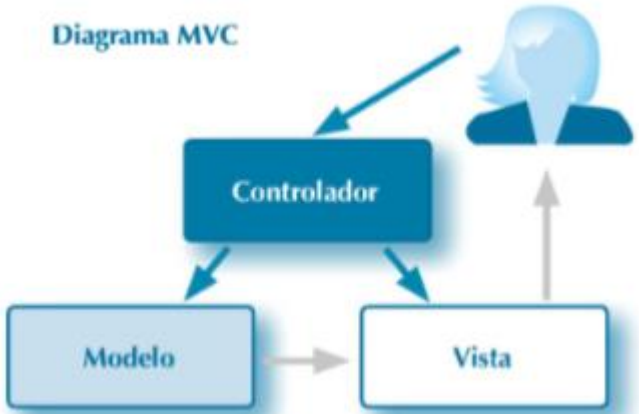
Administración de servidores de aplicaciones

Despliegue de aplicaciones web

DAW – Despliegue de aplicaciones web (2º - 0614)

Arquitectura y configuración básica

- Un servidor de aplicaciones es un software global que permite dar servicio a las aplicaciones que se publican en el mismo, dando soporte para la seguridad, para las transacciones, administración de sesiones, registro de logs, etc.
- Arquitectura de las aplicaciones web: Modelo-Vista-Controlador (MVC)
  - Patrón de software que separa la lógica de negocio y los datos de la parte representativa que observa el usuario, los eventos y las comunicaciones entre los distintos componentes.
  - Se basa en la separación de módulos para su posterior mantenimiento y reutilización de código, además de la facilidad de detectar errores en caso de fallos.
  - Se distinguen los siguientes componentes:



- **Modelo**: es el componente que se encarga de representar la información con la que la aplicación trabaja. Realiza las consultas y modificaciones con base en los privilegios definidos previamente en el análisis de requisitos de la aplicación. La petición llega por parte del controlador y este componente ejecuta la acción y la presenta a la Vista.
- **Vista**: visualiza el modelo con un tipo de representación para interactuar con el usuario. Normalmente es la interfaz de la aplicación, ya que puede ser una aplicación web, aplicación de escritorio, o cualquier aplicación en cualquier entorno o sistema operativo.
- **Controlador**: controla los otros dos componentes, respondiendo a peticiones realizadas por el usuario (pulsar un botón de la interfaz), y a partir de ahí se comunica con el modelo para manipularlo haciendo cambios en la vista. Es el mediador entre el Modelo y la Vista.

Arquitectura y configuración básica

- Las principales ventajas de este tipo de tecnología es la centralización y disminución de la complejidad en el desarrollo de aplicaciones.
- La principal desventaja es que las aplicaciones web están más expuestas a ataques:
  - **Ataques a la computadora del usuario (cliente)**. Para evitarlos se requiere navegadores y plataformas seguras, libres de virus.
  - **Ataques al servidor**. Para evitarlos hay que garantizar que los datos no sean modificados sin autorización (integridad) y que sólo sea distribuida a las personas autorizadas (control de acceso)
  - **Ataques al flujo de información que se transmite entre cliente y servidor**. Para evitarlos la información no debe ser leída (confidencialidad), modificada o destruida por terceros, al mismo tiempo que hay que garantizar un canal de comunicación fiable que no se interrumpa con relativa facilidad.

Arquitectura y configuración básica

- Para conseguir aplicaciones web seguras hay que establecer mecanismos que garanticen:
  - **Autenticación**: permite identificar, en todo momento, quién es el usuario que está accediendo, mediante los siguientes métodos:
    - Autenticación básica: solicitud de usuario y clave.
    - Autenticación con certificados.
      - HTTP DIGEST AUTH (HTTP Autenticación de texto implícita).
      - HTTP NTLM AUTH (HTTP Autentication Microsoft NT LAN Manager).
  - **Autorización**: permite, una vez autenticado, determinar a qué datos y módulos de la aplicación puede acceder el usuario.
  - **Validación de entradas**, ya que se puede manipular el código de validación del lado del cliente.
  - **Inyección de comandos SQL**: técnica para explotar aplicaciones web que no validan la información suministrada por el cliente para generar consultas SQL peligrosas.
- Para conseguir aplicaciones web seguras hay que utilizar una serie de mecanismos y herramientas:
  - Deshabilitación de servicios y cuentas no utilizadas.
  - Actualización del sistema operativo y aplicaciones (parches).
  - Fortaleza en las contraseñas.
  - Utilización de Firewalls.
  - Back-ups periódicas.
  - Análisis periódico de logs.
  - Verificación periódica de servicios activos.
  - Cifrado del tráfico.
  - Establecimiento de políticas de seguridad.

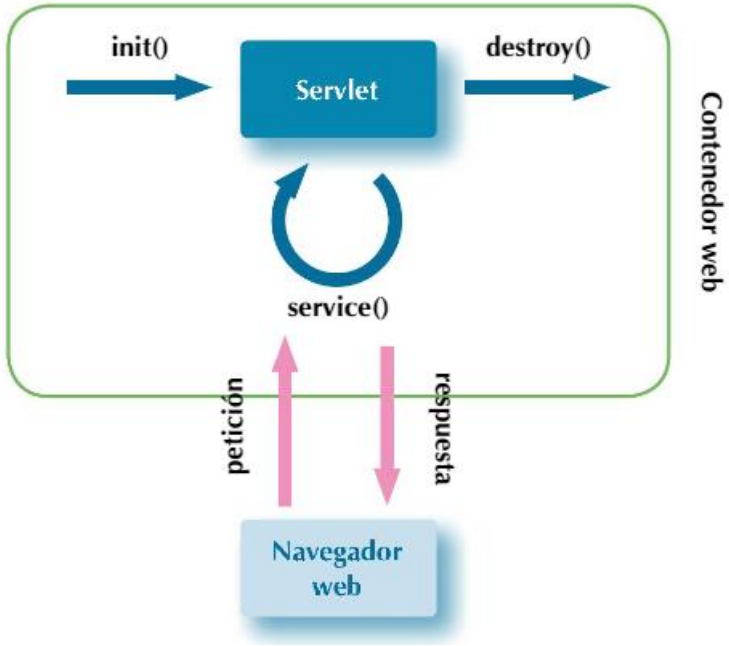
Arquitectura y configuración básica

- Servidores de aplicaciones más usados:
  - **Apache-Tomcat**: es un servidor de aplicaciones open source implementando Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. Estos módulos son desarrollados por Java Community Process. Oracle certifica con la versión de Java correspondiente.
  - WildFly (JBoss Application Server): es un software implementado por JBoss y está desarrollado en Java. Está disponible para todos los sistemas operativos del mercado y está basado en los proyectos punteros de software open source. Si se analiza la estructura interna, está basado en dos núcleos principales:
    - JBoss Module: controla la carga de los recursos de la clase contenedor.
    - Modular Service Container: administra los servicios usados por el contenedor, por ejemplo, instala y desinstala los diferentes servicios.
  - GlassFish: es un software de código abierto desarrollado por Sun Microsystems para la plataforma Java EE. Se encuentra disponible la versión GlassFish 5.0.1 con Java EE 8. Este proyecto open source proporciona un proyecto estructurado de desarrollo que permite tener disponibles las nuevas funcionalidades de Java.
  - JOnAs: es un software libre indicado para servidores de aplicaciones desarrollado por el consorcio ObjectWeb.

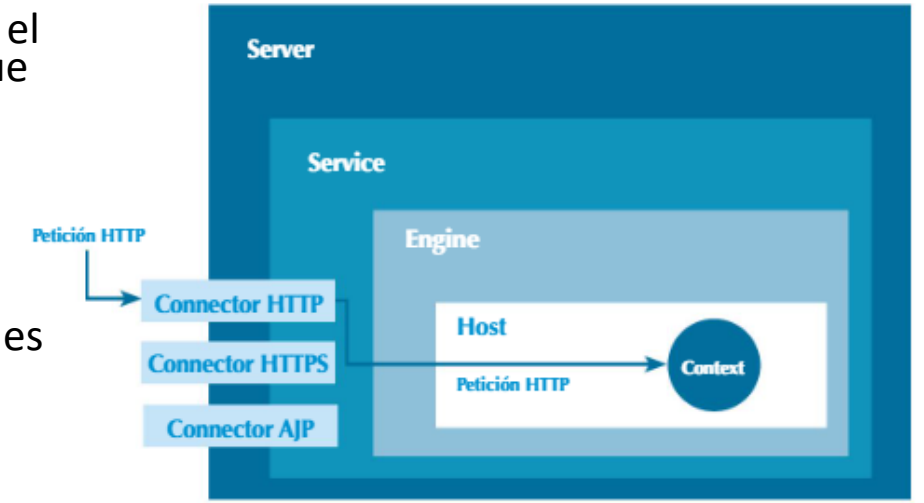
Servidor de aplicaciones Tomcat

- Funcionamiento de un contenedor de servlets (como Apache-Tomcat):
  - El navegador web del cliente solicita una página al servidor HTTP.
  - El contenedor de servlets procesa la petición y la asigna al servlet apropiado.
  - El servlet elegido es el encargado de generar el texto de la página web y entregarla al contenedor de servlets.
  - El contenedor devuelve la página web al navegador del cliente.
- La arquitectura de Tomcat permite de forma lógica conectar sus diferentes componentes que cada uno realice una función determinada:
  - **Server**: es el componente que representa el contenedor al completo y dentro de él se ejecutan los demás componentes. Posee una interfaz gráfica que prácticamente no se customiza.
  - **Service**: es un componente intermedio que se comunica y que está dentro del componente server, que posee varias conexiones.
  - **Engine**: es el servicio que permite procesar las peticiones que provienen desde el exterior y responder a tales solicitudes. Por ello, posee múltiples conectores que hacen posible resolver las peticiones.
  - **Host**: es un nombre de red que puede poseer varios hosts o alias, como pueden ser www.daw.com o www.daw.es.
  - **Connector**: es el encargado de controlar las comunicaciones con el cliente. Tomcat incluye varios conectores, como AJP Connector o HTTP Connector.
  - **Context**: representa una aplicación web. Un host puede poseer varios o múltiples contextos pero con una única ruta.

Funcionamiento de un contenedor de servlets



Arquitectura Tomcat



Servidor de aplicaciones Tomcat

- Directorios/Ficheros (a partir de /opt/tomcat):
  - **bin**: es el directorio que contiene los ficheros binarios y los scripts de inicio, startup.sh (inicio) y shutdown.sh (parada).
  - **conf**: configuración global del servidor de aplicaciones:
    - catalina.policy: define la política de seguridad.
    - catalina.properties: es el fichero en el que se relacionan los ficheros .jar de Java para la clase Catalina. La información está relacionada con los paquetes de seguridad y las rutas de las clases cargadas. También contiene algunas configuraciones de las cadenas más usadas en la caché.
    - context.xml: contiene la información de contexto común a todas las aplicaciones web que se ejecuten en Tomcat. También permite localizar el archivo web.xml de cada una de las aplicaciones.
    - jaspic-propierties.xml: es un fichero que permite definir un módulo de autenticación diferente a JAAS.
    - jaspic-propierties.xsd: es el esquema XSD que define si es válido el fichero anterior xml.
    - logging.propierties: este fichero permite configurar las funciones de logging de todas las aplicaciones y de la actividad del servidor.
    - **server.xml**: es el fichero principal de configuración de Tomcat.
    - tomcat-users.xml: este fichero contiene los usuarios, contraseñas y roles que serán usados para acceder al servidor Apache-Tomcat.
    - tomcat-users.xsd: es el esquema XSD que define si es válido el fichero anterior xml.
    - web. xml: es un fichero estándar para las aplicaciones web, común a todas las aplicaciones web, ya que posee la configuración global a todas ellas.
  - **lib**: este directorio contiene todos los ficheros jar usados en el servidor que corresponden a Tomcat, a JSP, etc.
  - **logs**: el directorio donde se almacenarán los logs.
  - **webapps**: contiene todas las aplicaciones web.
- Variables de entorno
  - CATALINA_BASE: es el directorio que representa la configuración de una instancia de Tomcat, normalmente coincide con el valor de la variable CATALINA_HOME (solamente se diferencia en caso de varias instancias).
  - CATALINA_HOME: indica el directorio raíz de la instalación del servido (/opt/tomcat).
  - CATALINA_TMPDIR : indica el directorio temporal ($CATALINA_BASE/temp).
  - JRE_HOME: indica el directorio de JRE.
  - CLASSPATH: rutas de búsqueda de fichero .jar.

Servidor de aplicaciones Tomcat

- Parámetros del fichero server.xml

Marca	Descripción	Ejemplo
<server> </server>	Es la etiqueta principal que engloba a toda la configuración del fichero. Engloba a uno o más servicios, y contiene al atributo port, que permite definir el puerto por el que escucha.	<Server port="8005" shutdown="SHUTDOWN">
<Listener ….>	Para definir las extensiones JMX ("Java Management Extensions") que serán usadas por Apache-Tomcat. Tienen un atributo principal llamado className.	<Listener className="org. apache.catalina.startup. VersionLoggerListener"/>
<GlobalNamingResources> …… </GlobalNamingResources>	Esta marca define los recursos tipo JNDI para usarlos globalmente en el servidor de aplicaciones. Usa la etiqueta Resource para especificar la localización.	<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" />

Marca	Descripción	Ejemplo
<Service> ….. </Service>	Esta marca permite agrupar uno o más conectores. Si se teclea Catalina se ejecutará el servidor como independiente.	<Service name="Catalina">
<Connector />	Representa las conexiones TCP desde el exterior que serán abiertas cuando arranca el servidor. Uno de los principales es el HTTPConnector.	<Connector port="8090" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8444" />
<Engine> …… </Engine>	Se usan dentro de la marca Service o de Host. Se procesan las peticiones que llegan a la marca Connector y que la cabecera disponga el Host por defecto.	<Engine name="Catalina" defaultHost="localhost">
<Logger/>	Esta marca indicará dónde serán envíados los registros de logs.	<Logger className="org.apache. catalina.logger.FileLogger" directory="logs" prefix="localhost_log." suffix=".txt" timestamp="true"/>
<Host> …. </Host>	A partir de esta etiqueta se pueden definir varios hosts virtuales para atender las peticiones.	<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
<Context> …. <Context>	Indicará la ruta a partir de la cual se encontrarán las aplicaciones ejecutadas en Tomcat. Normalmente se encuentran debajo del directorio webapps.	

Servidor de aplicaciones Tomcat

- Administrar aplicaciones web
  - Acceso:
    - http://<IP>:8080/manager/html
    - http://<IP>:8080 -> “Manager App”
  - Usuario/Password, fichero “$CATALINA_HOME/conf/tomcat-users.xml” y roles “manager-gui” y “admin-gui”

```
<user username="admin" password="ubuntu" roles="manager-gui,admin-gui"/>
```

  - Opciones de la aplicación
    - Path: es el nombre del directorio a partir del cual se encuentra la estructura e información de la aplicación desplegada.
    - Versión name: es el número de versión de la aplicación, que va incrementando a medida que se van realizando modificaciones a la aplicación.
    - Display Name: el nombre de la aplicación que se va a desplegar.
    - Running: si está funcionando en el momento que se accede al administrador de aplicaciones (true / false).
    - Sessions: número de sesiones abiertas.
    - Commands: comandos que nos permiten controlar la aplicación: Start, Stop, Reload y Undeploy (borrado).
    - Expire sessions. Elimina las sesiones con cierto tiempo de espera (30 minutos).

Path	Version	Display Name	Running	Sessions	Commands
/examples	*None specified*	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

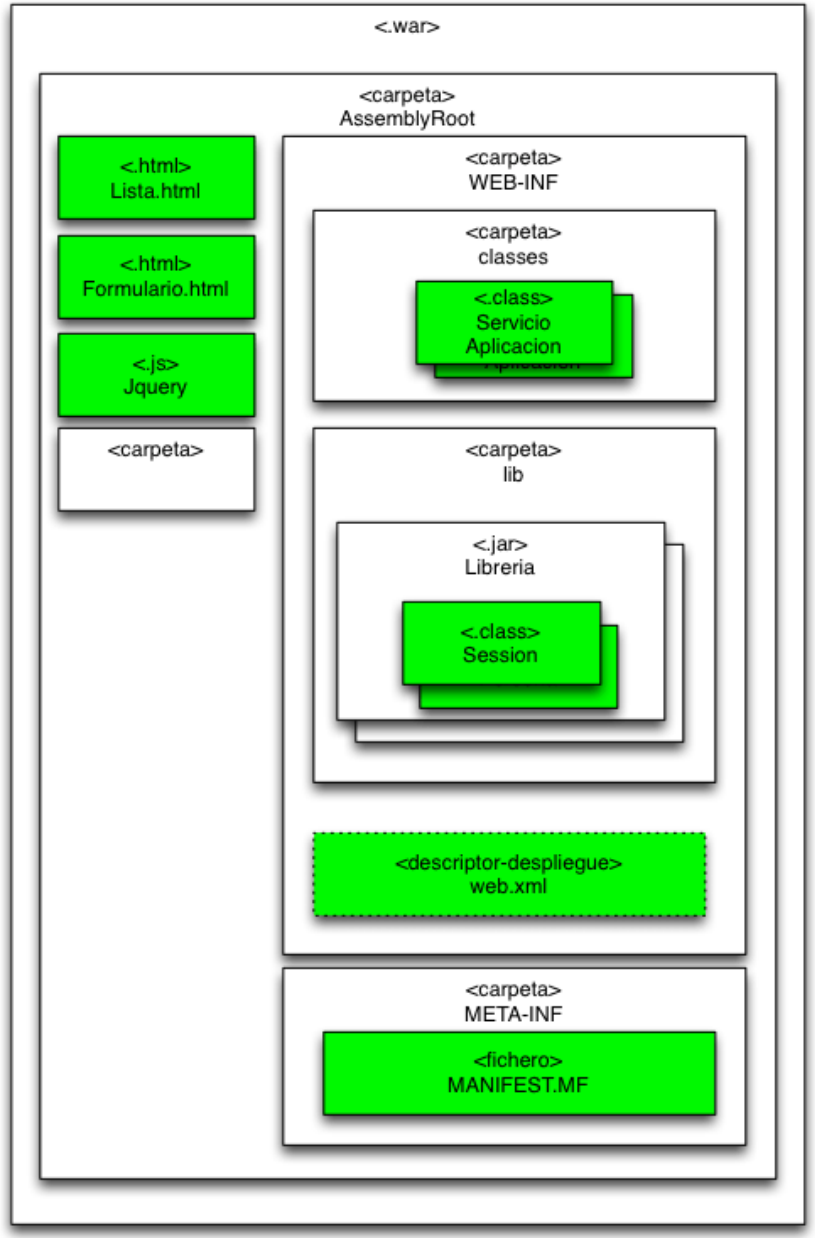
Servidor de aplicaciones Tomcat

- Generación de fichero .war (app.war, y desde directorio de la aplicación)

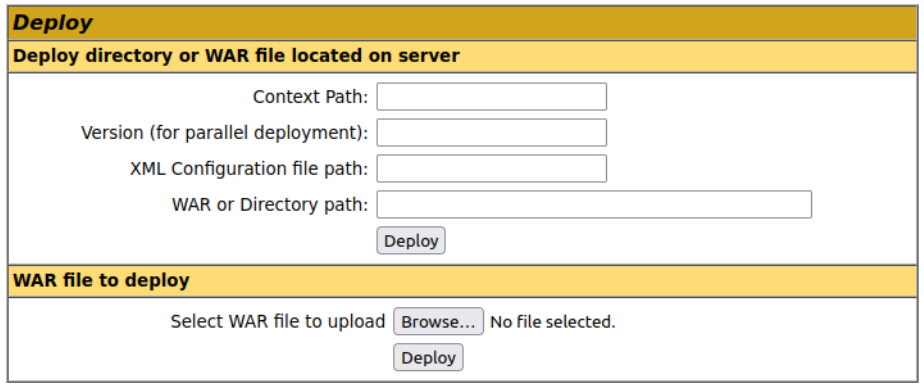
```
#jar -cvf app.war *
```

  - Opciones del comando:
    - c: crea un fichero sobre la salida estándar
    - v: visualiza la información sobre el fichero creado, tamaño, modificación, etc.
    - f: este argumento permite especificar el fichero .war que se va a crear
- Despliegue de una aplicación de forma automática (a partir de un fichero .war)
  - Opción 1: Indicando la ruta del contexto de la aplicación, la versión de la misma, la ruta del fichero .xml de la configuración y la ruta del directorio de la aplicación o del fichero .war.
  - Opción 2: Subiendo al administrador de aplicaciones el fichero .war.

Estructura fichero war



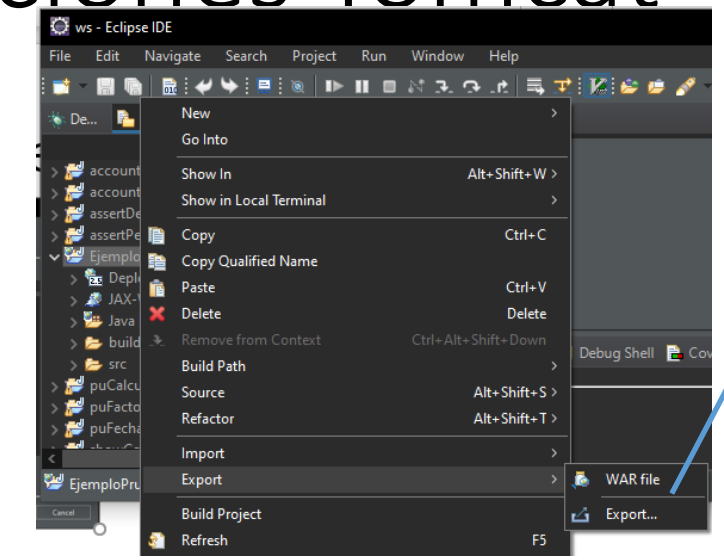
EjemploPruebaCarga.war
EjemploPruebaCarga.zip

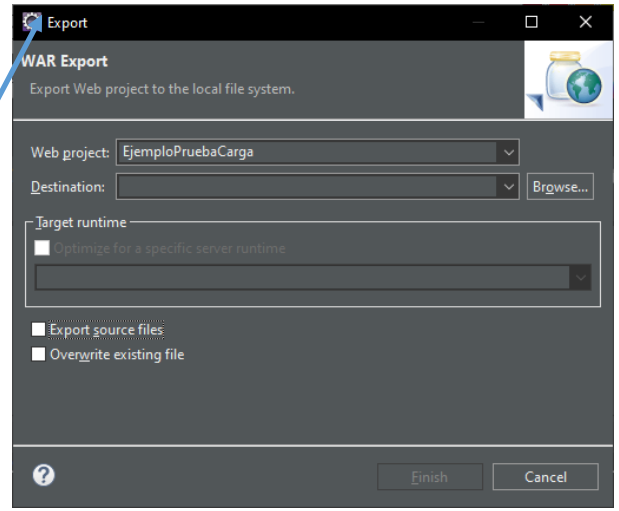


Despliegue de una aplicación

Servidor de aplicaciones Tomcat

- Creación de war desde eclipse



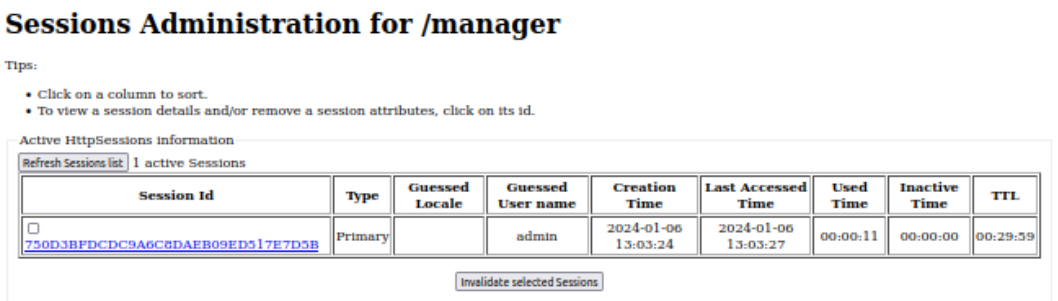


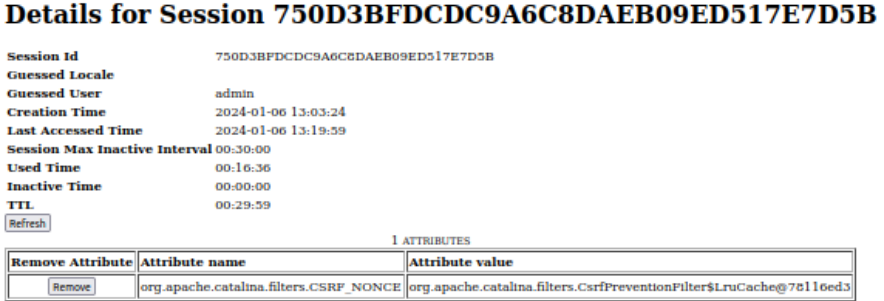
Aplic_Web.zip

- Despliegue de una aplicación de forma manual
  - Copiar la carpeta contenida en www (con el nombre del proyecto) en el directorio webapps de Tomcat.
  - Renombrar la nueva carpeta así creada en Tomcat con un nombre más sencillo. Esa será la carpeta de la aplicación en Tomcat.
  - Crear, dentro de dicha carpeta, otra nueva, y darle el nombre WEB-INF (respetando las mayúsculas).
  - Crear, dentro de WEB-INF, otros dos subdirectorios, llamados lib y classes.
  - Copiar en lib todas las librerías (.jar) que necesite la aplicación para su funcionamiento.
  - Copiar el contenido de la carpeta bin de la aplicación en el subdirectorio WEB-INF/classes del Tomcat.
  - Crear en WEB-INF un fichero de texto llamado web.xml, con las rutas de los servlets utilizados en la aplicación.

Servidor de aplicaciones Tomcat

- Sesiones
  - Por defecto las sesiones de las aplicaciones alojadas en Tomcat son **persistentes**, no se pierden en caso de fallo, pérdida de la conexión con el servidor o reinicio del mismo.
  - Información de las sesiones HTTP activas:
    - Session id: es una cadena de 32 caracteres (números y letras) en hexadecimal que identifica plenamente a una sesión que es creada por el administrador de aplicaciones de Tomcat. La longitud se crea usando el atributo sessionIdLength que define el número de bytes aleatorio, que, por defecto, es 16.
    - Type: es el tipo de sesión, dependiendo de si el acceso es primario o es con base en un acceso primario, que en ese caso sería secundario.
    - Guessed Locale: equipo desde el cual se ha generado la petición (configurable).
    - Guessed User name: el nombre del usuario que ha generado la sesión de la aplicación.
    - Creation time: es el momento de creación de la sesión.
    - Last Accessed time: es el atributo que indica el último acceso a la sesión.
    - Used time: el periodo de tiempo que está usando la sesión recursos del servidor de aplicaciones.
    - Inactive time: es el tiempo de inactividad de la sesión.
    - TTL: es el tiempo de vida que tiene la sesión, que si está inactiva va descontando de los 30 minutos (configurable) que posee antes de expirar.





Servidor de aplicaciones Tomcat

- Sesiones
  - Almacenamiento en disco de las sesiones (liberar recursos de memoria o persistencia)
    - Casuística:
      - Al parar Tomcat todas las sesiones activas se vuelcan a disco de manera que, al volver a arrancarlo, se podrán restaurar.
      - Sesiones inactivas, pero todavía no caducadas (configuración).
      - Las sesiones con un tiempo de vida que supere un límite (por seguridad y para evitar posibles bloqueos de sesión).
    - Configuración:
      - De forma global: fichero: <CATALINA_HOME>/conf/context.xml
      - A nivel local de una aplicación: fichero context.xml de dicha aplicación (<CATALINA_HOME>/webapps/<aplicacion>/META-INF/context.xml).

```
<Context>
    <!-- classname  especifica la clase del servidor que implementa el gestor, es recomendable utilizar el org.apache.catalina.session.PersistentManager -->
    <Manager className="org.apache.catalina.session.PersistentManager"
        <!--saveOnRestart=true para indicar que se guarden todas las sesiones al reiniciar el servidor -->
        saveOnRestart="true"
        <!--maxActiveSession cuando se supera el límite aquí establecido se comienzan a enviar a disco las nuevas sesiones. Se establece un valor -1 para indicar ilimitadas sesiones-->
        maxActiveSession="3"
        <!--minIdleSwap establece el número mínimo de segundos que transcurren antes de que una sesión pueda copiarse al disco duro -->
        minIdleSwap="0"
        <!--maxIdleSwap indica el número máximo de segundos que transcurren antes de que una sesión pueda copiarse al disco duro -->
        maxIdleSwap="60"
        <!--maxIdleBackup para indicar el número de segundos desde que una sesión estuvo activa por última vez hasta que se envíe al disco. La sesión no es eliminada de memoria. Permite restauración de la sesión en caso de caída del servidor. -->
        maxIdleBackup="5"
        <!--Store indica cómo y donde almacenar la sesión, están disponibles las siguientes implementaciones: org.apache.catalina.session.FileStore y org.apache.catalina.session.JDBCStore -->
        <Store className="org.apache.catalina.session.FileStore" />
    </Manager>
</Context>
```

Ejemplo de configuración de almacenamiento en disco de sesiones (fichero: context.xml)

Servidor de aplicaciones Tomcat

- Configuración SSL (puerto 8443)
  - Creación del almacén de claves, en el directorio "/opt/tomcat/conf" y de nombre localhost-rsa.jks

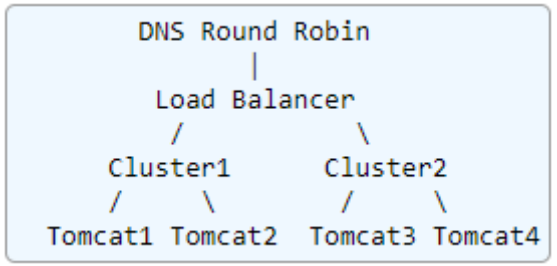
```
#keytool -genkey -alias tomcat -keyalg RSA -keystore localhost-rsa.jks -validity 365 -keysize 2048
```

  - Modificar fichero "/opt/tomcat/conf/server.xml"

```
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol
"
               maxThreads="150" scheme="https" secure="true" SSLEnabled="true"
keystoreFile="conf/localhost-rsa.jks" keystorePass="ubuntu" clientAuth="false"
sslProtocol="TLSv1.2" SSLVerifyClient="none" >
    </Connector>
```

  - Acceso https://localhost:8443 o https://<IP>:8443

Servidor de aplicaciones Tomcat



- Configurar Tomcat en clúster:
  - La implementación de clustering con Tomcat provee:
    - Escalabilidad: escalado horizontal (implica el incremento del número de servidores), escalado vertical (implica el incremento de los recursos del propio servidor).
    - Alta disponibilidad: Tomcat provee failover:
      - Request-level failover: Si un servidor cae, los siguientes requerimientos se redireccionarán a otros servidores activos.
      - Session-level failover: En el caso de que un servidor deje de dar servicio, otro servidor del clúster debería proporcionar la sesión a los clientes consiguiendo reducir al mínimo la pérdida de conexión.
    - Balanceo de carga: establecer un método de reparto de la carga de peticiones entre los servidores del clúster.
  - Las soluciones de clustering típicas se basan en ofrecer un sistema basado en ejecución distribuida:
    - El conector del servidor de clúster recibe la petición desde los clientes, y el procesador del servidor de clúster encapsula las peticiones en los objetos "RequestEntry" y los escribe en JavaSpace.
    - El conector del Worker del clúster toma dichas peticiones y el procesador del worker del clúster resuelve las peticiones
  - HOW-TO: https://tomcat.apache.org/tomcat-9.0-doc/clúster-howto.html
  - Pasos a seguir:
    - Todos los atributos de sesion deben implementar java.io.Serializable.
    - Descomentar el elemento Cluster en server.xml.
    - Descomentar Valve (ReplicationValve) en server.xml
    - Si las múltiples instancias de Tomcat están en la misma máquina el parámetro tcpListenPort tiene que ser único para cada una de las instancias.
    - Establecer en el archivo web.xml el elemento <distributable/> o bien definirlo de forma <Context distributable="true"/>.
    - El atributo jvmRoutes tiene que estar definido en el "Engine" <Engine name="Catalina“ jvmRoute="nodeX"> estableciendo su valor al nombre de la instancia en el clúster.
    - Sincronizar la hora de todos los nodos con un servicio NTP.
    - Configurar el parámetro loadbalancer en modo "sticky session".