



---

# TOMCAT

Aziz GRIMEH  
Ingénieur d'études

# Sommaire

- **Qu'est-ce que Apache Tomcat**
- **Architecture du serveur Tomcat**
- **installer Apache Tomcat sur Ubuntu**
- **Création d'un utilisateur Tomcat**
- **Création de service systemd pour Tomcat**
- **CONFIGURATION DE TOMCAT**
- **Déployer une application dans Tomcat**
- **Les fichiers journaux de Tomcat**
- **modifier le paramètre de JVM**
- **Configuration des ressources**

# Qu'est-ce que Apache Tomcat

- **Tomcat**, souvent appelé Apache Tomcat, est l'une des applications les plus populaires conçues pour exécuter un servlet Java et rendre des serveurs web avec un codage de page Java. Cette application à code source libre est publiée par Apache Software Foundation,

# Avantages apportés par Apache Tomcat

- Tomcat est un moyen rapide et facile d'exécuter vos applications dans Ubuntu. Il permet un chargement rapide et aide à faire fonctionner un serveur plus efficacement
- Tomcat contient une série de choix de personnalisation complets et intégrés qui permettent à ses utilisateurs de travailler de manière flexible
- Tomcat est une application gratuite et à code source ouvert (open-source). Cette application offre une grande personnalisation grâce à l'accès au code
- Tomcat offre à ses utilisateurs un niveau de sécurité supplémentaire

# Architecture du serveur Tomcat

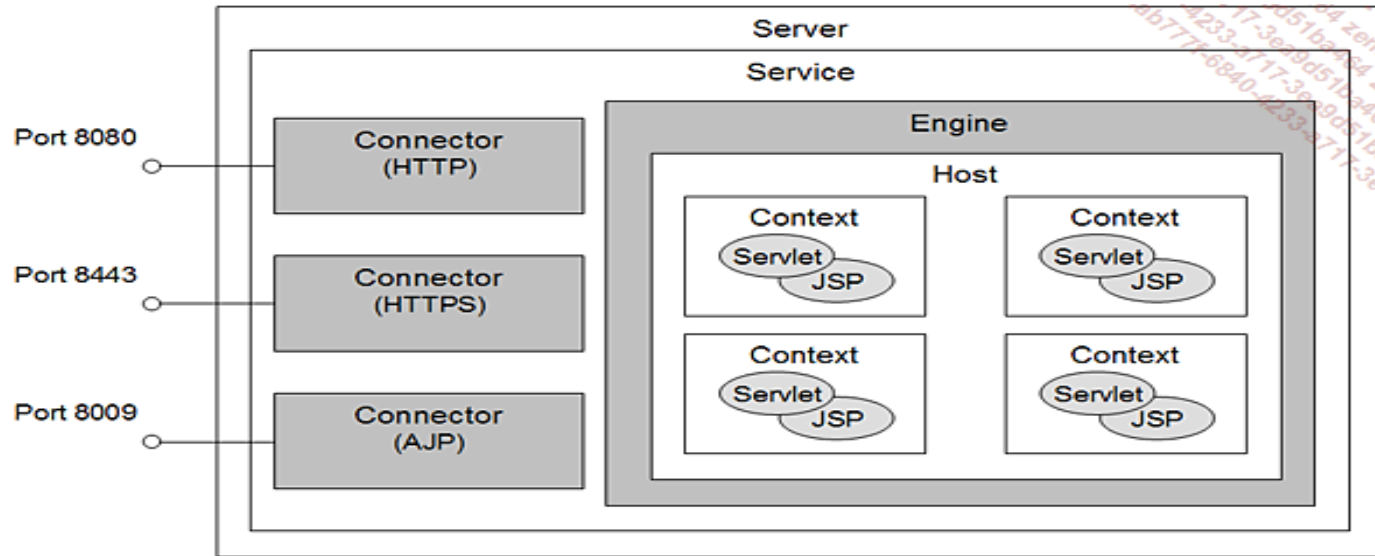
1. Les différents composants de Tomcat
2. Arborescence de l'installation

# 1. Les différents composants de Tomcat

- L'architecture de Tomcat consiste en un ensemble de composants dédiés au traitement et à la satisfaction des demandes émises par les clients via le protocole HTTP, du traitement et analyse de la requête, jusqu'à l'exécution de la ressource demandée par le client.
- Les composants principaux de Tomcat sont appelés conteneurs tout simplement parce qu'ils contiennent d'autres composants plus spécifiques.

Le schéma qui suit fait apparaître les éléments fondamentaux de Tomcat, les conteneurs sont tous représentés, ce sont les éléments:

- Server
- Service
- Engine
- Host
- Context



- Cette hiérarchie de conteneurs et de composants est représentée, d'un point de vue de la configuration, par le fichier `server.xml` qui est le principal fichier de configuration de `Tomcat`.
- L'utilisation du format `XML` pour ce fichier permet de bien comprendre l'imbrication des conteneurs les uns dans les autres, ainsi que la position, et, de ce fait, l'impact, des autres éléments du serveur.
- Ainsi, le schéma précédent met en évidence les conteneurs `Context`, qui représentent les applications, dans un conteneur `Host`, d'un point de vue de l'organisation du fichier `server.xml`, les conteneurs `Context` apparaîtront comme éléments `XML` enfants du conteneur `Host`.

Exemple :

```
<Host ... >  
  <Context ... >  
  </Context>  
</Host>
```



## 2. Arborescence de l'installation

L'arborescence d'installation d'un serveur Tomcat se présente comme ceci:

```
vagrant@tomcat:/opt/tomcat$ ls -lt
total 152
drwxr-x--- 2 tomcat tomcat 4096 May 23 12:24 bin
drwxr-x--- 2 tomcat tomcat 4096 May 23 12:24 lib
drwxr-x--- 2 tomcat tomcat 4096 May 23 12:24 temp
-rw-r----- 1 tomcat tomcat 19004 Mar 31 14:24 BUILDING.txt
-rw-r----- 1 tomcat tomcat 6210 Mar 31 14:24 CONTRIBUTING.md
-rw-r----- 1 tomcat tomcat 60269 Mar 31 14:24 LICENSE
-rw-r----- 1 tomcat tomcat 2333 Mar 31 14:24 NOTICE
-rw-r----- 1 tomcat tomcat 3378 Mar 31 14:24 README.md
-rw-r----- 1 tomcat tomcat 6905 Mar 31 14:24 RELEASE-NOTES
-rw-r----- 1 tomcat tomcat 16507 Mar 31 14:24 RUNNING.txt
drwx----- 2 tomcat tomcat 4096 Mar 31 14:24 conf
drwxr-x--- 2 tomcat tomcat 4096 Mar 31 14:24 logs
drwxr-x--- 7 tomcat tomcat 4096 Mar 31 14:24 webapps
drwxr-x--- 2 tomcat tomcat 4096 Mar 31 14:24 work
```

### Répertoire bin

Le répertoire bin contient tous les fichiers qui permettent de lancer Tomcat, que ce soit sous Linux (fichiers .sh ) ou Windows (fichiers .bat).

### Répertoire conf

Ce répertoire contient les fichiers de configuration de Tomcat Nous reverrons la signification des fichiers qui se trouvent ici en détails Les fichiers de ce répertoire permettent de configurer tous les aspects du fonctionnement de Tomcat, de sa sécurité, et aussi du chargement des applications web.

### Répertoire lib

Ce répertoire contient les librairies Java dont Tomcat a besoin pour fonctionner.

### Répertoire log

Ce répertoire est vide C'est ici que Tomcat écrit ses fichiers de journalisation dans sa configuration par défaut.

### Répertoire temp

Répertoire temporaire contenant des fichiers temporaires,

## Répertoire webapp

Le répertoire `webapp` contient les applications web gérées par `Tomcat`. Cet endroit peut bien sûr être redéfini.

Par défaut, il contient cinq applications, très utiles lorsque l'on débute :

- `docs` : contient les pages de documentation de `Tomcat`, également accessibles en ligne
- `exemples` : contient des exemples simples de servlets et de pages `JSP`
- `host manager` et `manager` : contiennent l'application de gestion des applications web de `Tomcat`. Cette application permet de charger des applications web à chaud, et de les visualiser dans une interface web.
- `ROOT` : racine des applications web chargées par défaut.

## Répertoire work

Répertoire de travail de `Tomcat`, dans lequel, entre autres, les classes Java correspondant aux pages `JSP` sont créées et compilées.

# installer Apache Tomcat sur Ubuntu

- Étape 1 - Installer Java
- Étape 2 - Créer un utilisateur Tomcat
- Étape 3 - Installez Tomcat 10

# Étape 1 : Installer Java

- Avant d'installer Tomcat sur Ubuntu, nous devons installer Java pour exécuter le code de l'application web Java. OpenJDK est la version Java par défaut dans Ubuntu 20.04.

L'installation de Java est simple et rapide. Il suffit de suivre les commandes ci-dessous :

```
sudo apt update
```

- Installez le paquet OpenJDK en exécutant la commande :

```
sudo apt install default-jdk
```

- Vérifiez la version Java active actuelle :

```
java -version
```

## Étape 2 - Créer un utilisateur Tomcat

Nous vous recommandons d'exécuter le serveur Tomcat avec un compte utilisateur dédié. Créez un nouvel utilisateur, ce qui est recommandé pour des raisons de sécurité, principalement pour les déploiements de production.

Pour créer un nouveau compte, tapez:

```
sudo useradd -m -d /opt/tomcat -U -s /bin/false tomcat
```

La commande ci-dessus créera un utilisateur et un groupe avec le nom " tomcat" dans votre système.

# Étape 3 - Installez Tomcat 10

L'équipe de développement d'Apache Tomcat publie de temps à autre la dernière version de Tomcat.

Il sera donc bon de vérifier le téléchargement de la dernière version de Tomcat à partir du [serveur de téléchargement officiel](#) .

Utilisez la commande ci-dessous pour télécharger Tomcat 10.

```
wget https://downloads.apache.org/tomcat/tomcat-10/v10.0.20/bin/apache-tomcat-10.0.20.exe
```

Une fois le téléchargement terminé, extrayez l'archive téléchargée et copiez tout le contenu dans le répertoire de base de Tomcat.

```
sudo tar xzvf apache-tomcat-10*.tar.gz -C /opt/tomcat --strip-components=1
```

Ensuite, définissez les autorisations de fichier appropriées.

```
sudo chown -R tomcat:tomcat /opt/tomcat/  
sudo chmod -R u+x /opt/tomcat/bin
```

**Vous avez maintenant la dernière application Tomcat  
sur votre système.**



# Création d'un utilisateur Tomcat

**Étape 4 -Créer un utilisateur Tomcat**

**Étape 5 -Activer l'accès à distance à Tomcat**

# Étape 4 -Créer un utilisateur Tomcat

Maintenant, configurez votre tomcat avec des comptes d'utilisateurs pour sécuriser l'accès aux pages d'administration/gestionnaire. Pour ce faire, éditez le fichier `conf/tomcat-users.xml` dans votre éditeur et collez le code suivant dans les balises `<tomcat-users>` `</tomcat-users>`.

Nous vous recommandons de changer le mot de passe dans la configuration ci-dessous avec un mot de passe hautement sécurisé.

```
sudo vim /opt/tomcat/conf/tomcat-users.xml
```

Ajoutez les valeurs suivantes. Assurez-vous de changer le mot de passe pour l'accès administrateur et gestionnaire.

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager-gui" />
  <role rolename="admin-gui" />
  <user username="admin" password="secret" roles="manager-gui, admin-gui" />
</tomcat-users>
```

Enregistrez le fichier et fermez.

# Étape 5 -Activer l'accès à distance à Tomcat

Les applications Tomcat manager et host-manager par défaut sont accessibles uniquement pour `localhost`. Pour autoriser l'accès à ces pages depuis le système distant, vous devez modifier les fichiers de configuration suivants. Vous pouvez soit autoriser un système distant spécifique, soit tout autoriser. Modifiez le context.xml fichier pour le manager et l'application de host manager:

```
sudo vim /opt/tomcat/webapps/manager/META-INF/context.xml
```

Commentez la section ajoutée pour la restriction d'adresse IP afin d'autoriser les connexions de n'importe où.

```
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
    sameSiteCookies="strict" />
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|
filters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
```

De la même manière, modifiez context.xml pour l'application Host Manager dans l'éditeur de texte :

```
sudo vim /opt/tomcat/webapps/host-manager/META-INF/context.xml
```

Commentez la même section pour autoriser les connexions de n'importe où.

```
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
    sameSiteCookies="strict" />
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|
filters\.\CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
```

# Création de service systemd pour Tomcat

## Étape 6 -Créez un fichier d'unité Tomcat Systemd

Tomcat fournit des scripts `bash` pour démarrer, arrêter le service. Mais, pour simplifier, créez un script de démarrage pour gérer Tomcat en tant que service `systemd`. Créons un fichier `tomcat.service` avec le contenu suivant:

```
sudo vim /etc/systemd/system/tomcat.service
```

```
[Unit]
```

```
Description=Tomcat 10 servlet container
```

```
After=network.target
```

```
[Service]
```

```
Type=forking
```

```
User=tomcat
```

```
Group=tomcat
```

```
Environment="JAVA_HOME=/usr/lib/jvm/default-java"
```

```
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"
```

```
Environment="CATALINA_BASE=/opt/tomcat"
```

```
Environment="CATALINA_HOME=/opt/tomcat"
```

```
Environment="CATALINA_PID=/opt/tomcat/temp/tomcat.pid"
```

```
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"
```

```
ExecStart=/opt/tomcat/bin/startup.sh
```

```
ExecStop=/opt/tomcat/bin/shutdown.sh
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Rechargez le service démon systemd pour charger les fichiers nouvellement créés.

```
sudo systemctl daemon-reload
```

Maintenant, démarrez l'application Tomcat pour la première fois.

```
sudo systemctl start tomcat.service
```

Ensuite, activez le démarrage automatique du service Tomcat pour les démarrages ultérieurs du système. Ceci est plus important pour les déploiements de production.

```
sudo systemctl enable tomcat.service
```

À partir de maintenant, l'application Tomcat est en cours d'exécution sur votre système. Vous pouvez vérifier l'état du service en exécutant la commande ci-dessous. Assurez-vous que l'état affiche " active (running)".

```
sudo systemctl status tomcat.service
```

# Accédez à l'interface Web Tomcat

**Étape 7 -Accédez à l'interface Web Tomcat**



# Étape 7 - Accédez à l'interface Web Tomcat

Le serveur Tomcat par défaut s'exécute sur le port **8080**. Comme vous avez configuré Tomcat sur votre système, vous pouvez accéder à l'interface Web à partir de votre système.

Vous pouvez accéder aux interfaces Tomcat en saisissant l'adresse IP de votre serveur ou un nom de domaine pointant vers ce serveur, suivi du port 8080 dans votre navigateur:

```
http://IP_serveur:8080/
```

Vous verrez la page comme ci-dessous:

# Apache Tomcat/10.0.20



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](#)[First Web Application](#)[Realms & AAA](#)[JDBC DataSources](#)[Examples](#)[Servlet Specifications](#)[Tomcat Versions](#)

## Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

`$CATALINA_HOME/conf/tomcat-users.xml`

In Tomcat 10.0 access to the manager application is split between different users.

[Read more...](#)

### [Release Notes](#)

### [Changelog](#)

### [Migration Guide](#)

### [Security Notices](#)

## Documentation

### [Tomcat 10.0 Documentation](#)

### [Tomcat 10.0 Configuration](#)

### [Tomcat Wiki](#)

Find additional important configuration information in:

`$CATALINA_HOME/RUNNING.txt`

Developers may be interested in:

[Tomcat 10.0 Bug Database](#)

[Tomcat 10.0 JavaDocs](#)

[Tomcat 10.0 Git Repository at GitHub](#)

## Getting Help

### [FAQ](#) and [Mailing Lists](#)

The following mailing lists are available:

#### [tomcat-announce](#)

Important announcements, releases, security vulnerability notifications. (Low volume).

#### [tomcat-users](#)

User support and discussion

#### [taglibs-user](#)

User support and discussion for [Apache Taglibs](#)

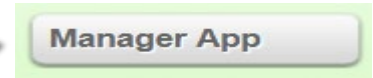
#### [tomcat-dev](#)

Development mailing list, including commit messages

L'application Tomcat Manager est une application Web fournie avec le serveur d'application Tomcat.

L'interface Manager nous fournit les fonctionnalités de base dont nous avons besoin pour gérer nos applications Web déployées.

Cliquez sur le **Manager App** bouton de la page d'accueil ou saisissez directement l'/manager URL du navigateur du serveur Tomcat principal pour y accéder.



**`http://IP_serveur:8080/manager`**



## Gestionnaire d'applications WEB Tomcat

Message : OK

### Gestionnaire

<a href="#">Lister les applications</a>	<a href="#">Aide HTML Gestionnaire</a>	<a href="#">Aide Gestionnaire</a>	<a href="#">Etat du serveur</a>
-----------------------------------------	----------------------------------------	-----------------------------------	---------------------------------

### Applications

Chemin	Version	Nom d'affichage	Fonctionnelle	Sessions	Commandes
/	Aucun spécifié	Welcome to Tomcat	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/docs	Aucun spécifié	Tomcat Documentation	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/examples	Aucun spécifié	Servlet and JSP Examples	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/host-manager	Aucun spécifié	Tomcat Host Manager Application	true	1	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/manager	Aucun spécifié	Tomcat Manager Application	true	1	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes

### Deployer

Emplacement du répertoire ou fichier WAR de déploiement sur le serveur

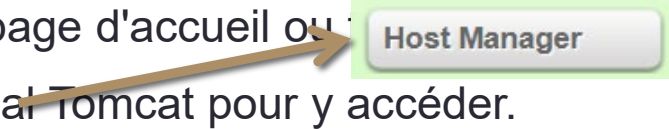
Chemin de contexte (requis) :

Version (pour les déploiements en parallèle) :

L'application Tomcat Host Manager est une autre application Web fournie avec le serveur d'application Tomcat.

Elle est utilisée pour créer/supprimer des hôtes virtuels dans le service Tomcat. Un hôte virtuel vous permet de définir plusieurs noms d'hôtes sur un seul serveur.

Cliquez sur le **Host Manager** bouton de la page d'accueil ou sur l'/host-manager URL dans le serveur principal Tomcat pour y accéder.



[http://IP\\_serveur:8080/host-manager](http://IP_serveur:8080/host-manager)



## Gestionnaire d'Hôtes Virtuels de Tomcat

Message :	OK
-----------	----

### Gestionnaire d'hôte ("Host Manager")

<a href="#">Liste des Hôtes Virtuels</a>	<a href="#">Aide HTML du manager d'hôte</a>	<a href="#">Aide du Gestionnaire d'Hôtes</a>	<a href="#">Etat du serveur</a>
------------------------------------------	---------------------------------------------	----------------------------------------------	---------------------------------

### Nom d'hôte

Nom d'hôte	Alias de l'hôte	Commandes
<a href="#">localhost</a>		Gestionnaire d'Hôtes installé, commandes désactivées

### Ajouter un hôte virtuel (Virtual Host)

#### Hôte

Nom :

Alias :

Répertoire de base :

Déploiement automatique ☒

Déploiement au démarrage ☒

Déployer le XML ☒

Décompresser les WARs ☒

App gestionnaire ☒

Copier le XML ☐

# CONFIGURATION DE TOMCAT

1. Le fichier **server.xml**
2. Le fichier tomcat-users.xml

## Le fichier server.xml

Le principal fichier de configuration de Tomcat s'appelle `server.xml` et se trouve dans le répertoire `CATALINA_HOME/conf`. Comme son nom l'indique, le contenu de ce fichier de configuration s'écrit en XML.

Cependant, lors de son démarrage, Tomcat vérifie la syntaxe des éléments déclarés dans ce fichier, aussi il est important de bien respecter la syntaxe d'écriture et la distinction majuscule/minuscule.

En fait, les erreurs commises dans ce fichier peuvent avoir deux conséquences:

**Le serveur ne démarre pas** . Un élément est correctement positionné dans le fichier mais sa syntaxe n'est pas correcte, il faut vérifier le nom de l'élément et de ses attributs.

**Le serveur démarre**, mais la nouvelle configuration n'a pas été appliquée. L'élément n'est peut être pas positionné au bon endroit dans le fichier, ou bien les valeurs avec lequel il est configuré ne sont pas correctes.



Le fichier **server.xml**, fourni par défaut avec toute nouvelle installation de serveur Tomcat, est très bien commenté et des exemples de configuration sont même donnés en commentaire, de sorte qu'il suffit simplement de décommenter ces exemples pour activer tel ou tel autre élément de configuration.

Il est recommandé de faire une copie de sauvegarde de ce fichier immédiatement après une installation fonctionnelle du serveur, mais aussi avant chaque modification du contenu de ce fichier.

## Les éléments de configuration:

- Chacun des éléments de configuration du fichier **server.xml** est lié à une classe Java particulière du serveur Tomcat.
- Certains de ces éléments sont indispensables, et d'autres non, l'objectif de cette partie est de présenter précisément chacun de ces éléments.
- L'imbrication des éléments de configuration les uns aux autres est toutefois quelque chose d'assez complexe à comprendre au premier abord.

## Élément Server:

- L'unique élément racine Server modélise un serveur Catalina dans sa totalité. Catalina est le nom du serveur proprement dit, alors que Tomcat est le nom du projet complet.
- Tout ce qui est défini dans cet élément est global au serveur, et sera donc appliqué, entre autres, à toutes les applications web qu'il supporte. Techniquement, un server est un élément XML, et une interface Java: `org.apache.catalina.Server`.

## Élément Server:

Cet élément supporte trois attributs :

- `className`: la classe d'implémentation de l'interface `org.apache.catalina.Server`. En principe on n'en change pas, et l'on peut omettre cet attribut, qui prendra alors sa valeur par défaut.
- `port`: le port d'arrêt (*shutdown*) de Tomcat. Ce port ne correspond pas au port HTTP écouté par Tomcat, mais à un port sur lequel Tomcat reçoit la commande de s'arrêter. Lorsque l'on tape la commande `shutdown` sur l'invite de commande, cet ordre est émis, et Tomcat s'éteint alors.
- `shutdown`: la commande envoyée sur ce port.

## Exemple2. Exemple d'élément Server

```
<Server port="8005" shutdown="SHUTDOWN">  
  <!-- Contenu de l'élément -->  
</Server>
```

## Élément Service:

Un service est un container dans lequel on peut trouver autant de sous-éléments `Connector` que l'on veut, et un unique sous-élément `Engine`. On peut définir autant de services que l'on veut au sein d'un serveur donné, à condition qu'ils diffèrent tous par leurs noms.

Cet élément possède deux attributs :

- `className`: nom de la classe Java qui implémente l'interface `org.apache.catalina.Service`. Peut être omis, dans ce cas l'implémentation par défaut est choisie.
- `name`: nom logique de ce service, doit être unique au sein de tous les services définis dans un serveur

## Exemple3. Exemple d'élémentService

```
<Service name="Catalina">  
    <!-- Contenu de cet élément -->  
</Service>
```

## Élément Connector:

Un connecteur est un objet Java capable d'écouter un port précis et comprenant un protocole précis.

À chaque protocole supporté par Tomcat est associé une classe Java de connecteur.

Chaque connecteur dirige ensuite les requêtes qu'il reçoit au moteur de servlets défini dans ce service.

Donc un moteur de servlet donné peut répondre à des requêtes en provenance de plusieurs ports, et suivant des protocoles différents.



Quel que soit le connecteur choisi, un certain nombre d'attributs sont toujours disponibles :

- `port`: le port que ce connecteur écoute.
- `enableLookups`: autorise ou non les requêtes DNS lorsqu'une servlet invoque la méthode `request.getRemoteHost()`.

Une requête DNS est un processus éventuellement coûteux. Si `enableLookups` est à `false`, alors `request.getRemoteHost()` retourne l'adresse IP plutôt que le nom de domaine associé.

- `maxPostSize`: la taille maximale des requêtes POST supportée par ce serveur.

La valeur par défaut est fixée à 2Mo. Une valeur de -1 signifie qu'il n'y a pas de limite.

## Connecteur HTTP

Le connecteur `HTTP` supporte le standard `HTTP/1.1`, et permet à Tomcat de fonctionner comme un serveur web à part entière.

- `maxThreads`: indique le nombre maximal de requêtes que ce connecteur peut traiter à la fois (une requête est traitée dans son propre thread par le connecteur `HTTP`).

Au-delà de cette limite, les requêtes sont placées dans une file d'attente.

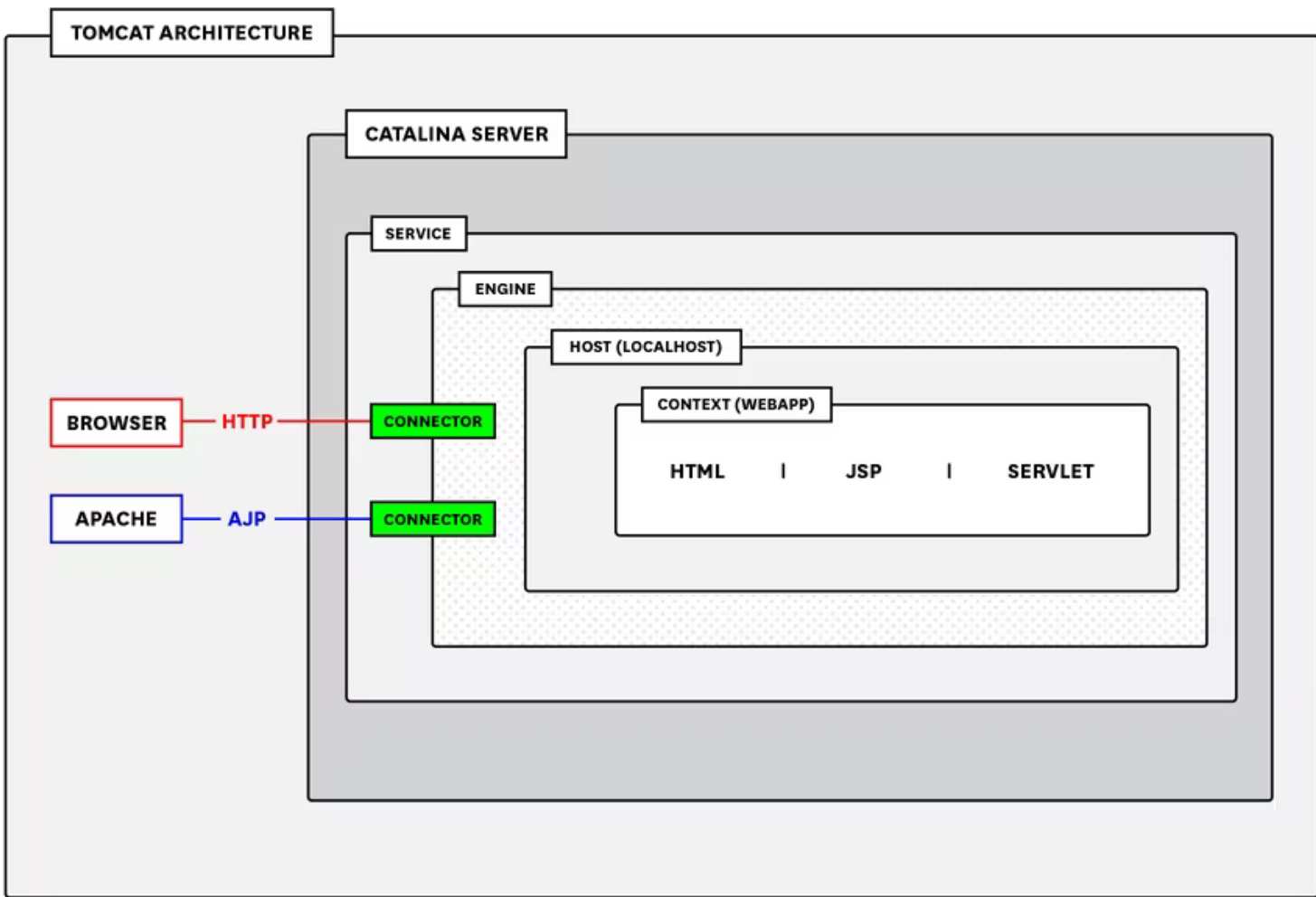
- `acceptCount`: taille maximale de la file d'attente des requêtes. Si des requêtes supplémentaires arrivent, alors elles sont rejetées.

- `protocol`: cet attribut peut prendre deux types de valeur, un nom de protocole, tel que `HTTP/1.1`

## Connecteur AJP

Le connecteur AJP communique avec un autre serveur web, en utilisant le protocole AJP. Il existe deux connecteurs de ce type :

- **JK 1.2.\*** : permet de communiquer avec tous les serveurs web qui supportent le protocole JK ;
- **mod\_proxy**: permet de communiquer avec Apache 2.2.\*.



## Exemple4. Exemple d'éléments Connector

```
<!-- Port standard d'écoute de requêtes HTTP sur le port 8080-->
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    maxthreads="400" acceptCount="100"
    redirectPort="8443" />

<!-- Port standar d'écoute de requêtes Apache AJP/1.3 -->
<Connector protocol="AJP/1.3"
    address="::1"
    port="8009"
    redirectPort="8443" />
```

## Élément Engine:

L'élément `Engine` modélise le moteur de servlet proprement dit. À chaque serveur `Catalina` est associé un unique moteur de servlet, auquel on accède via des connecteurs.

Un moteur peut comporter plusieurs sous-éléments `Host`, chacune représentant un hôte virtuel.

Toutes les implémentations par défaut de cet élément supportent les attributs suivants :

- `name`: le nom de ce moteur, notamment utilisé dans les fichiers journal ;
- `defaultHost`: l'hôte virtuel par défaut vers lequel diriger cette requête.

## Exemple5. Exemple d'élément Engine:

```
<Engine name="Catalina" defaultHost="localhost">  
    <!-- Contenu de l'élément Engine -->  
</Engine>
```

## Élément Host:

Cet élément modélise un hôte virtuel.

Un hôte virtuel doit être associé à l'adresse IP de ce serveur, via un DNS ou un fichier hosts.

Si l'attribut `defaultHost` du moteur dans lequel se trouve cet hôte est défini, alors au moins un des hôtes doit obligatoirement posséder ce nom.



Toutes les implémentations de Host doivent supporter les attributs suivants :

- `name`: le nom de cet hôte virtuel ;
- `appBase`: l'*application base* pour cet hôte. Il s'agit du répertoire contenant les applications web pour cet hôte. Il peut s'agir d'un chemin absolu, ou d'un chemin relation au répertoire `$CATALINA_BASE`.
- `autoDeploy`: indique à `Tomcat` s'il doit charger automatiquement les applications web qui sont copiées dans le répertoire `appBase`. Le cas échéant, Tomcat scrute le contenu de ce répertoire à intervalles de temps réguliers, afin de détecter de nouveaux répertoires, ou de nouveaux `fichiers.war`. La valeur par défaut de cet attribut est `true`.
- `deployOnStartup`: indique à Tomcat s'il doit charger les applications web se trouvant dans `appBase` lors de son démarrage. La valeur par défaut de cet attribut est `true`.

L'implémentation par défaut supporte en plus les attributs suivants:

- `workDir`: répertoire de travail propre à cet hôte. Par défaut, Catalina utilise un sous-répertoire de `$CATALINA_BASE/work` pour chaque hôte, mais il est possible d'en choisir un autre. On peut accéder à ce répertoire d'une servlet, via l'attribut d'application `javax.servlet.context.tempdir`.
- `unpackWARs`: indique à Catalina qu'il doit exécuter les applications web de cet hôte dans un répertoire de travail avant de les exécuter. La valeur par défaut de cet attribut est `true`.
- `deployXML`: indique à Catalina s'il peut ou non prendre en compte les fichiers `/META-INF/context.xml` des applications web. La valeur par défaut de cet attribut est `true`.

- **Exemple6. Exemple d'élément Host**

```
<Host name="www.website1.com" appBase="website1-webapps"  
      unpackWARs="true" autoDeploy="true">  
    <!-- Contenu de l'élément Host -->  
</Host>
```

# Déployer une application dans Tomcat

1. Déploiement automatique d'applications
2. Utiliser le répertoire webapps/
3. L'élément <Context>
4. Déploiement avec XML

Il existe plusieurs méthodes permettant d'installer (on utilise également le terme déployer) une application dans **Tomcat**, selon les cas d'utilisation du serveur.

En **développement**, il est appréciable de pouvoir installer une application en copiant simplement le fruit de son travail dans le répertoire adéquat de Tomcat, alors que sur un serveur de **production**, on préfère avoir un contrôle plus fin des différentes options d'installation, grâce à des d'outils d'administration.

# 1. Déploiement automatique d'applications

Le déploiement automatique d'applications **Web** permet d'installer et de rendre disponible de nouvelles applications, sans avoir besoin de redémarrer le serveur. Cette configuration est idéale lorsque le serveur **Tomcat** est utilisé en phase de développement, puisqu'elle permet aux développeurs de facilement tester leurs applications.

Le fichier `server.xml` de Tomcat contient l'élément de configuration `<Host>`. Cet élément, possède l'attribut `autoDeploy`, qui permet d'activer le déploiement automatique, ce qui est le cas par défaut.

```
<Host name="localhost"  appBase="webapps"  unpackWARs="true"  
      autoDeploy="true"  xmlValidation="false"  xmlNamespaceAware="false">  
    ...  
</Host>
```

## 2. Utiliser le répertoire webapps/

La méthode la plus simple pour déployer une application Web Java EE dans Tomcat, consiste simplement à déposer dans le répertoire `webapps/` de l'arborescence du serveur, le répertoire contenant l'application, ou bien l'archive d'application Web (fichier `.WAR`).

Si le déploiement automatique des applications est activé, le serveur déploie et rend l'application disponible sans qu'il soit nécessaire de le redémarrer. Au démarrage du serveur, toutes les applications présentes dans ce répertoire sont rendues disponibles par Tomcat par la création automatique d'un contexte d'application Web, mais uniquement si l'attribut `deployOnStartup` de l'élément possède la valeur `true`. À noter également que cet élément `<Host>` possède également l'attribut `unpackWARs` qui, positionné à la valeur `true` provoque la décompression des archives d'applications Web.

L'emplacement et le nom de ce répertoire peuvent être modifiés pour prendre en compte des contraintes d'organisation du système de fichiers et des applications, il suffit alors de modifier l'attribut `appBase` de l'élément de configuration `<Host>` en spécifiant le nouveau répertoire

### 3. L'élément <Context>

Pour rendre disponible une application Web, elle doit être associée à un contexte, ce contexte est automatiquement créé lorsque l'on installe une application en la copiant dans le répertoire `webapps/`.

Cependant, il est également possible de définir explicitement un contexte dans le fichier `server.xml`, cette opération permet d'installer une application Web en dehors du répertoire `webapps` mais également de ne pas tenir compte du contenu de ce répertoire, et de se fier uniquement à la configuration écrite du serveur, ce qui est un gage de sécurité.

L'élément <Context> suivant permet de créer le contexte `/demo` pour l'application installée sous `/home/vagrant/workspaces/tmp/demo/demoapp.war`

```
<Host name="www.website1.com" appBase="website1-webapps"
    unpackWARs="true" autoDeploy="true">
    <Context path="/demo" docBase="/home/vagrant/workspaces/tmp/demoApp"/>
    <!-- Contenu de l'élément Host -->
</Host>
```



## 4. Déploiement avec XML

La contrainte de la solution de déploiement précédente est qu'elle nécessite le redémarrage du serveur pour la prise en compte de la nouvelle application puisqu'il y a eu modification du fichier de configuration du serveur.

L'élément `Context` peut être défini dans un fichier XML séparé et utilisé pour faire un déploiement automatique de l'application. Comme précédemment, cette méthode requiert l'attribut `autoDeploy` de l'élément `<Host>` à `true` pour fonctionner. Une fois écrit, ce fichier doit être copié dans le répertoire

`CATALINA_BASE/conf/<engine>/<host>/`, où `<engine>` et `<host>` représentent respectivement le nom, indiqué par leurs attributs `name`, des éléments `<Engine>` et `<Host>` à utiliser pour déployer cette application.

Exemple: le fichier `demo.xml` qui permet de déployer sous le chemin de contexte `demo` l'application disponible dans `/home/vagrant/workspaces/tmp/demo/demoapp.war`

```
<Context path="/demo" docBase="/home/vagrant/workspaces/tmp/demoApp"/>
```

Sur une installation par défaut de Tomcat, l'élément `<Engine>` porte le nom **Catalina** et l'élément `<Host>` se nomme **localhost**. Il faudra dans ce cas copier le fichier **demo.xml** dans **CATALINA\_HOME/conf/Catalina/localhost**.

Un avantage de cette méthode est qu'il est possible de déclarer les ressources JNDI, ou les liens vers les ressources dont l'application a besoin, dans l'élément `<Context>`, par exemple:

```
<Context path="/demo" docBase="C:\applications\demo">
  <ResourceLink name="jdbc/demo"
    global="jdbc/GlobalDemo"
    type="javax.sql.DataSource" />
</Context>
```

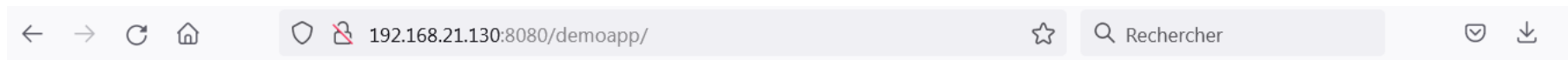
## Exemple de déploiement d'une archive .war sur tomcat

déployer un fichier **.WAR** = copier ce fichier dans le répertoire **webapps** de **tomcat**

```
root@tomcat:/opt/tomcat# tree -L 1 webapps/
webapps/
├── ROOT
├── demoapp
├── demoapp.war
├── docs
├── examples
├── host-manager
└── manager

6 directories, 1 file
root@tomcat:/opt/tomcat#
```

# Test déploiement



**Bonjour tout le monde,  
votre application c'est bien déployée..**

# Les fichiers journaux de Tomcat

Configuration des journaux de Tomcat

**a. Le système de journalisation de Tomcat**

**b. Structure du fichier logging.properties**

**c. Le fichier logging.properties par défaut**

## a. Le système de journalisation de Tomcat

La bibliothèque **Commons-logging** fait partie du projet Jakarta de la fondation Apache. Le projet **Jakarta Commons** contient un ensemble de bibliothèques de développement Java très variées. **Commons-logging** est une bibliothèque permettant la prise en charge de la journalisation des applications, il existe d'autres **API** de programmation Java pour la journalisation, comme **Log4j**, un autre projet Apache, et **java.util.logging**, une **API** standard incluse dans le **JDK**.

Le système de journalisation de Tomcat utilise le fichier de configuration **logging.properties** qui se trouve dans le répertoire **CATALINA\_HOME/conf**. C'est le fichier de configuration principal du serveur mais les applications déployées dans le serveur peuvent fournir leur propre fichier **logging.properties** dans leur répertoire **WEB-INF/classes**.

## b. Structure du fichier logging.properties

La première partie du fichier `logging.properties` permet de spécifier les gestionnaires de fichiers journaux (les `handlers`), la seconde, quant à elle, les éléments qui écrivent dans ces fichiers (les `loggers`).

### Les handlers

Les handlers peuvent écrire à destination d'un fichier texte ouvert la sortie standard. Les `handlers` qui écrivent dans un fichier utilisent la classe Java `org.apache.juli.AsyncFileHandler`, la sortie standard, quant à elle s'appelle `java.util.logging.ConsoleHandler`.

Pour déclarer un `handler`, il faut d'abord lui donner un nom, ce nom doit être composé d'un préfixe construit à partir d'un chiffre et de lettres, comme par exemple `1catalina`, auquel on ajoute le nom de la classe du handler, ce qui donnera par exemple `1catalina.org.apache.juli.AsyncFileHandler`. Le début du fichier `logging.properties` contient la propriété `handlers`, qui fait référence à tous les gestionnaires de journaux utilisés dans le fichier. La propriété `Handlers` permet de référencer le gestionnaire principal pour le serveur.

Une fois nommés, les `handlers` doivent être configurés grâce à des attributs. Certains de ces attributs sont communs aux deux types, et d'autres sont spécifiques à chacun.

Voici la liste des attributs de configuration communs à `org.apache.juli.FileHandler` et `java.util.logging.ConsoleHandler`:

- `level`: permet de spécifier le niveau de journalisation. Les valeurs possibles, des messages les plus graves aux plus anodins, sont: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST ou ALL. Le fait de positionner le niveau de journalisation à INFO affiche tous les messages de type INFO, WARNING, et SEVERE.
- `formatter`: permet d'indiquer une classe Java pour formater le contenu du fichier. Les valeurs possibles sont détaillées plus loin dans ce chapitre.





Voici la liste des attributs de configuration spécifiques à `org.apache.juli.AsyncFileHandler`:

- `prefix`: permet de spécifier le préfixe du nom du fichier. La valeur par défaut est `juli..`
- `suffix`: permet de spécifier le suffixe du nom du fichier. La valeur par défaut est `.log`.
- `directory`: permet d'indiquer le répertoire dans lequel sera stocké le fichier. L'emplacement par défaut est `CATALINA_HOME/bin/logs`.
- À noter que la rotation des fichiers journaux s'effectue toutes les nuits à minuit, et les noms des fichiers générés ont la syntaxe suivante:

`<prefix><date><suffix>` où `date` est exprimé selon le motif `AAAA-MM-JJ`.

## Les formatters

Les **formatters** sont des classes Java qui permettent de définir la manière dont sont écrits les messages par les **handlers**. Certaines de ces classes sont fournies en standard dans la plate-forme Java, d'autres sont spécifique à l'API JULI. Les **formatters** disponibles sont:

- **java.util.logging.SimpleFormatter**: c'est la valeur par défaut. Les entrées de messages sont formatées sur deux lignes.
- **java.util.logging.XMLFormatter**: permet de générer un journal au format XML pour exploitation avec un outil tel qu'une feuille de style XSL par exemple.
- **org.apache.juli.OneLineFormatter**: reprend le format du **SimpleFormatter** mais sur une seule ligne, ce qui facilite les extractions de lignes dans le fichier.
- **org.apache.juli.JdkLoggerFormatter**: formatage simplifié des messages, la date apparaît sous forme d'un **timestamp** UNIX.
- **org.apache.juli.VerbatimFormatter**: c'est le plus simple des **formatters** puisque seul le message apparaît.

```
java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter = java.util.logging.OneLineFormatter
java.util.logging.ConsoleHandler.encoding = UTF-8
```

## Les loggers

Les éléments de Tomcat qui peuvent utiliser le système de journalisation sont `<Engine>`, `<Host>` et `<Context>`.

Pour faire référence à un `<Engine>` ou à un `<Host>`, c'est la valeur de son attribut de configuration `name` dans le fichier `server.xml` qui va servir à l'identifier.

Par exemple, le nom utilisé sur l'élément `<Engine>` du fichier `server.xml` par défaut est `Catalina`; pour faire référence à cet élément, il faut donc écrire:

```
org.apache.catalina.core.ContainerBase.[Catalina]
```

Pour ce qui est de l'élément `<Host>`, son nom est `localhost` ; ce qui donne:

```
org.apache.catalina.core.ContainerBase.[Catalina].[localhost]
```

Deux attributs de configuration sont disponibles, `level` et `handlers`.

L'attribut `level` permet de spécifier le niveau de journalisation, les valeurs possibles sont les mêmes que pour les `handlers`. Le niveau de journalisation sur les `handlers` permet de filtrer les types de messages qui seront écrits, alors que le niveau de journalisation sur les `loggers` détermine réellement les messages qui seront écrits. Ce double emploi est très utile car plusieurs `loggers` peuvent utiliser `handlers`.

L'attribut `handlers` permet de spécifier le composant ou une liste de composants `handler` à appliquer à ce composant.

### c. Le fichier logging.properties par défaut

Le fichier `CATALINA_HOME/conf/logging.properties` est par défaut configuré pour générer les journaux suivants:

- un journal global pour le serveur qui s'affiche simultanément dans la console MS-DOS du serveur sous Windows et dans le fichier `CATALINA_HOME/logs/catalina.AAAA-MM-JJ.log`.
- un journal pour l'hôte par défaut (**localhost**) dans le fichier `CATALINA_HOME/logs/localhost.AAAA-MM-JJ.log`.
- un journal pour chacune des applications d'administration du serveur, comme par exemple le manager et la console d'administration. Les fichiers générés portent le nom de l'application.

# Comment modifier le paramètre de JVM (-Xms-Xmx) de Tomcat

## Configurer le fichier setenv.sh

Il est très important pour nous de configurer tous les paramètres corrects lors de l'exécution de votre application dans un environnement de production ou même dans un environnement de développement.

nous allons passer en revue les étapes de configuration de la **-Xms**, **-Xmx** and **-XX:PermSize** valeur pour le serveur Tomcat.

### **-Xmx**

Spécifie la taille maximale, en **bytes**, du pool d'allocation de mémoire. Cette valeur doit être un multiple de 1024 supérieur à 2 Mo. Ajoutez la lettre **k or K** pour indiquer les kilo-octets ou **m or M** pour indiquer les mégaoctets. La valeur par défaut est 64 Mo. La limite supérieure de cette valeur sera d'environ 4 000 m sur les plates-formes Solaris 7 et Solaris 8 SPARC et 2000m sur les plates-formes Solaris 2.6 et x86. Donc, en termes simples, vous dites que Java utilise un maximum de 1024 Mo de mémoire disponible.

### **-Xms**

Spécifie la taille initiale de la **jvm** lors du démarrage de la machine

### **-XX:PermSize**

Il est utilisé pour définir la taille de la génération permanente. C'est là que sont conservés les fichiers de classe.

```
vi /opt/tomcat/bin/setenv.sh
```

```
export CATALINA_OPTS="$CATALINA_OPTS -Xms512m"  
export CATALINA_OPTS="$CATALINA_OPTS -Xmx3072m"  
export CATALINA_OPTS="$CATALINA_OPTS -XX:MaxPermSize=256m"
```

```
root@tomcat:/opt/tomcat# vi bin/setenv.sh  
export CATALINA_OPTS="$CATALINA_OPTS -Xms512m"  
export CATALINA_OPTS="$CATALINA_OPTS -Xmx3072m"  
export CATALINA_OPTS="$CATALINA_OPTS -XX:MaxPermSize=256m"
```

Execute command: `./catalina.sh run`

```
root@tomcat:/opt/tomcat# ./bin/catalina.sh run
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:        /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:    -Xms512m -Xmx8192m -XX:MaxPermSize=256m
NOTE: Picked up JDK_JAVA_OPTIONS:  --add-opens=java.base/java.lang=ALL-UNNAMED --add-ops
=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED
UNNAMED
```



# Comment générer un "heap dump" ?

Le "heap dump" ou "memory dump" contient une copie de la mémoire qui permet d'identifier un certain nombre de problèmes liés à celle-ci.

On le réalise généralement suite à une erreur de type ***OutOfMemoryError***

Attention: ce 'dump' peut être long à générer et fige l'application durant sa génération. Il faut par ailleurs disposer de suffisamment d'espace disque pour accueillir l'image.

## Méthode console:

L'utilitaire "jmap" se trouve dans le répertoire "/bin" de votre JDK.

```
jmap -dump:format=b,file=<FILENAME.hprof> <PID_DU_PROCESS_JAVA>
```

Notez que parfois sous Linux vous aurez besoin d'exécuter la commande avec le compte qui exécute tomcat avec une commande de ce type :

```
sudo su -l tomcat -s /bin/bash -c 'jmap-dump:format=b,file=<FILENAME.hprof> <PID_DU_PROCESS_JAVA>'
```

## Exemple:

```
sudo su -l tomcat -s /bin/bash -c 'jmap -dump:format=b,file=dump1.hprof 9200'
```

```
root@webserver:~# ss -ltnp
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port      Process
LISTEN     0            128         127.0.0.53:53            0.0.0.0:*                users:(( "systemd-resolve",pid=217,fd=13))
LISTEN     0            128         0.0.0.0:22               0.0.0.0:*                users:(( "sshd",pid=282,fd=3))
LISTEN     0            100         127.0.0.1:25             0.0.0.0:*                users:(( "master",pid=392,fd=13))
LISTEN     0            128         127.0.0.1:6010           0.0.0.0:*                users:(( "sshd",pid=9459,fd=10))
LISTEN     0            100         *:8080                   *:.*                     users:(( "java",pid=9200,fd=43))
LISTEN     0            128         [::]:22                  [::]:.*                 users:(( "sshd",pid=282,fd=4))
LISTEN     0            100         [::1]:25                 [::]:.*                 users:(( "master",pid=392,fd=14))
LISTEN     0            128         [::1]:6010               [::]:.*                 users:(( "sshd",pid=9459,fd=9))
LISTEN     0            1          [::ffff:127.0.0.1]:8005  [::]:.*                 users:(( "java",pid=9200,fd=53))
root@webserver:~# sudo su -l tomcat -s /bin/bash -c 'jmap -dump:format=b,file=dump1.hprof 9200'
sudo: setrlimit(RLIMIT_CORE): Operation not permitted
Dumping heap to /opt/tomcat/dump1.hprof ...
Heap dump file created [83335855 bytes in 0.483 secs]
```

The screenshot displays the Java VisualVM application window. On the left, the 'Applications' tree shows a local VM named 'VisualVM'. The main pane is titled '[heapdump] dump1.hprof' and shows a 'Heap Dump' analysis. The 'Summary' tab is selected, displaying various statistics and environment details.

**Basic info:**

- Date taken: Sat Mar 05 06:37:24 CET 2022
- File: C:\Users\Houcem\Desktop\dump1.hprof
- File size: 79,5 MB
- Total bytes: 74 310 341
- Total classes: 3 801
- Total instances: 1 179 004
- Classloaders: 64
- GC roots: 2 334
- Number of objects pending for finalization: 0

**Environment:**

- OS: Linux (5.3.18-2-pve)
- Architecture: amd64 64bit
- Java Home: /usr/lib/jvm/java-11-openjdk-amd64
- Java Version: 11.0.13
- JVM: OpenJDK 64-Bit Server VM (11.0.13+8-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)
- Java Vendor: Ubuntu

**System properties:**

[Show System Properties](#)

**Threads at the heap dump:**

[Show Threads](#)

Applications x

- Local
  - VisualVM
  - Remote
  - VM CoreDumps
  - Snapshots

Start Page x [heapdump] dump1.hprof x

## [heapdump] dump1.hprof

Heap Dump

[Summary](#) [Classes](#) [Instances](#) [OQL Console](#)

Classes

[Compare with another heap dump](#)

Class Name	Instances [%] v	Instances	Size
<b>byte[]</b>		303 339 (25,7 %)	26 095 141 (35,1 %)
java.lang.String		143 434 (12,2 %)	4 159 586 (5,6 %)
java.util.TreeMap\$Entry		109 782 (9,3 %)	6 257 574 (8,4 %)
java.io.File		89 701 (7,6 %)	3 946 844 (5,3 %)
java.lang.String[]		68 885 (5,8 %)	4 344 136 (5,8 %)
org.apache.catalina.LifecycleEvent		59 218 (5 %)	2 368 720 (3,2 %)
org.apache.catalina.Container[]		57 960 (4,9 %)	1 844 640 (2,5 %)
java.util.HashMap\$ValueIterator		57 959 (4,9 %)	3 245 704 (4,4 %)
java.util.concurrent.locks.AbstractQueuedSynchronizer\$Node		32 481 (2,8 %)	1 689 012 (2,3 %)
java.util.concurrent.ConcurrentHashMap\$Node		30 890 (2,6 %)	1 359 160 (1,8 %)
java.util.HashMap\$Node		21 592 (1,8 %)	950 048 (1,3 %)
java.util.concurrent.ConcurrentHashMap\$KeyIterator		12 582 (1,1 %)	1 006 560 (1,4 %)
org.apache.catalina.util.ContextName		10 080 (0,9 %)	483 840 (0,7 %)
java.util.TreeMap		8 846 (0,8 %)	707 680 (1 %)
org.apache.catalina.webresources.Cache\$EvictionOrder		8 820 (0,7 %)	141 120 (0,2 %)
java.util.TreeMap\$KeySet		8 820 (0,7 %)	211 680 (0,3 %)
java.util.concurrent.ConcurrentHashMap\$ValueIterator		8 820 (0,7 %)	705 600 (0,9 %)
java.util.HashMap\$KeyIterator		8 819 (0,7 %)	493 864 (0,7 %)
java.util.LinkedHashMap\$LinkedKeyIterator		8 818 (0,7 %)	458 536 (0,6 %)
java.lang.Object[]		6 923 (0,6 %)	628 208 (0,8 %)
java.lang.Object		5 987 (0,5 %)	95 792 (0,1 %)
int[]		5 746 (0,5 %)	5 231 664 (7 %)
java.util.ArrayList		5 246 (0,4 %)	167 872 (0,2 %)
java.lang.reflect.Method		4 225 (0,4 %)	616 850 (0,8 %)
java.lang.StringBuilder		3 792 (0,3 %)	109 968 (0,1 %)
java.lang.Class[]		3 037 (0,3 %)	109 664 (0,1 %)
char[]		2 732 (0,2 %)	567 090 (0,8 %)
java.util.Date		2 565 (0,2 %)	82 080 (0,1 %)
sun.util.calendar.ZoneInfo		2 544 (0,2 %)	208 608 (0,3 %)
java.text.DecimalFormatSymbols		2 541 (0,2 %)	246 477 (0,3 %)
java.util.ArrayList\$Itr		2 520 (0,2 %)	90 720 (0,1 %)
org.apache.catalina.startup.HostConfig\$DeployedApplication[]		2 519 (0,2 %)	131 016 (0,2 %)
java.lang.invoke.LambdaForm\$Name		2 099 (0,2 %)	104 950 (0,1 %)
java.util.HashMap		2 014 (0,2 %)	128 896 (0,2 %)
sun.util.locale.LocaleObjectCache\$CacheEntry		1 476 (0,1 %)	94 464 (0,1 %)
org.apache.catalina.Session[]		1 470 (0,1 %)	35 280 (0 %)

# Configuration des ressources

La plate-forme Java EE propose un ensemble d'API de services mis en œuvre par un serveur d'applications, tels que l'accès aux bases de données ou encore la connectivité à des serveurs de messagerie électronique. Les ressources rendues disponibles par le serveur sont ensuite directement exploitables par les applications déployées dans ce serveur.

# Configuration de la ressource JNDI pour le pool de connexions

Voyons maintenant comment configurer une ressource de base de données à partager par plusieurs applications Web. Maintenant que vous disposez du pilote JDBC pour MySQL, vous pouvez configurer le serveur Tomcat pour l'utiliser pour accéder au serveur MySQL. Pour rendre la base de données accessible, vous devez la configurer en tant que ressource d'interface de nommage et d'annuaire Java. Sur la base des Java Servlet 3.1 toutes les ressources JNDI doivent être spécifiés dans les fichiers de configuration dans le dossier CATALINA\_BASE\conf :

- context.xml, une entrée « Ressource » doit être ajoutée pour permettre à JNDI de localiser et à JDBC de configurer la source de données.
- Pour la configuration des ressources par application Web, un fichier CATALINA\_BASE\conf\Catalina\localhost\PROJECT\_DIR.xml devra être créé pour ajouter l'entrée « Ressource ».
- web.xml, une entrée « Resource Reference » peut être ajoutée pour fournir une référence à la source de données qui n'a pas d'informations spécifiques au serveur, ce qui permet une portabilité plus facile.

# Exemple avec MySQL

## 1-Pilote JDBC pour MySQL

Le pilote JDBC est un jar ([mysql-connector-java-8.0.17.jar](#)) qu'on peut copier directement dans le répertoire lib de tomcat dans le cas où plusieurs applications sont déployées dans la même instance du serveur et qui ont besoin au driver pour se connecter à la BDD.

Une deuxième solution est que chaque application soit déployé avec son propre driver



Copier le projet depuis GitHub <https://github.com/sindibad31/demojndi.git> dans CATALINA\_BASE/webapp

Exécuter le script <https://github.com/sindibad31/demojndi/blob/master/resources/dbscript.sql> dans MySQL

## Vi conf/context.xml

Ajouter le XML suivant entre les deux balise `<context> ...</context>`

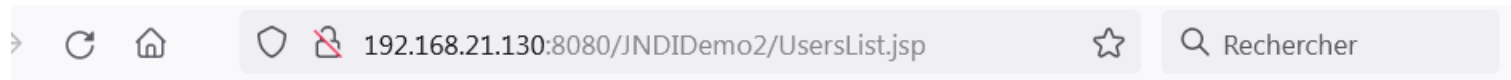
`<Resource`

```
    name="jdbc/UsersDB"
    auth="Container"
    type="javax.sql.DataSource"
    maxActive="100"
    maxIdle="30"
    maxWait="10000"
    driverClassName="com.mysql.cj.jdbc.Driver"
    url="jdbc:mysql://192.168.21.140:3306/usersdb"
    username="root"
    password="1234"
```

`</>`

`maxActive` est le nombre maximum de connexions dans le pool, donc utilisables en même temps

`maxIdle` est le nombre maximum de connexions libres (donc en attente d'être utilisée) dans le pool



## List of users

Name	Email
John	john@demo.com
Kerry	kerry@demoapp.com
toto	toto@gmail.com

**Merci pour votre attention**

**Questions**

