**Tổng quan về Tomcat**

**Người tìm hiểu: Nguyễn Bá Thành**

**Mục lục**

[Phần 1. Tổng quan về Tomcat 7](#_Toc494408782)

[1.1. Giới thiệu 7](#_Toc494408783)

[1.2. Các thuật ngữ cần nhớ: 7](#_Toc494408784)

[1.3. Tổ chức thư mục tomcat: 7](#_Toc494408785)

[Phần 2. Cài đặt Tomcat 8](#_Toc494408786)

[2.1. Cài đặt Tomcat trên Linux: 8](#_Toc494408787)

[2.2. Cài đặt Tomcat 8.5 8](#_Toc494408788)

[2.3. Để tắt service Tomcat 9](#_Toc494408789)

[2.4. Cấu hình user truy cập nội bộ Apache Manager 10](#_Toc494408790)

[Phần 3. Triển khai Web Application trên Tomcat. 13](#_Toc494408791)

[3.1. Giới thiệu: 13](#_Toc494408792)

[3.2. Vấn đề cơ bản khi triển khai trong Tomcat 13](#_Toc494408793)

[3.2.1. Triển khai web app có thể thực hiện bằng nhiều cách: 13](#_Toc494408794)

[3.2.2. Tomcat Manager 13](#_Toc494408795)

[3.2.3. Thuật ngữ Contexts 13](#_Toc494408796)

[3.2.4. Context Descriptor được lưu tại: 13](#_Toc494408797)

[3.3. Deployment khi Tomcat được khởi động 13](#_Toc494408798)

[3.3.1. Quá trình triển khai Web App khi Tomcat khởi động: 14](#_Toc494408799)

[3.4. Deploying trển Tomcat server đã được khởi động 14](#_Toc494408800)

[3.5. Triển khai Web App sử dụng Tomcat Manager 15](#_Toc494408801)

[3.6. Triển khai trong Tomcat 16](#_Toc494408802)

[3.6.1. Kiến trúc của WebArchive 16](#_Toc494408803)

[3.6.2. Tổ chức dạng thư mục: 16](#_Toc494408804)

[3.6.3. Archive Files 17](#_Toc494408805)

[3.6.4. Các loại file sử dụng cho việc triển khai Web App: 17](#_Toc494408806)

[3.6.5. Exploded archive directories (Project web app) 17](#_Toc494408807)

[3.6.6. Các hoạt động trong triển khai 17](#_Toc494408808)

[3.6.7. Các trạng thái sử dụng khi triển khai 18](#_Toc494408809)

[3.6.8. Các dạng có thể sử dụng khi muốn triển khai Web App trên Tomcat. 19](#_Toc494408810)

[3.6.9. Mô hình các cách triển khai 20](#_Toc494408811)

[Phần 4. Quản trị Tomcat 21](#_Toc494408812)

[4.1. Giới thiệu 21](#_Toc494408813)

[4.2. Cấu hình truy cập Manager Application 21](#_Toc494408814)

[4.3. Các role cơ bản có sẵn: 22](#_Toc494408815)

[4.4. Realm là gì? 22](#_Toc494408816)

[Phần 5. Giám sát Tomcat (Monitoring) 23](#_Toc494408817)

[5.1. Tổng quan 23](#_Toc494408818)

[5.2. Các cách khác nhau để giám sát 23](#_Toc494408819)

[5.3. Việc giám sát có thể thực hiện xong bằng 3 cách trên hệ thống: 23](#_Toc494408820)

[5.4. Cấu hình Monitor cho Web app, Web server, Database 24](#_Toc494408821)

[5.5. Tomcat Manager trong Tomcat 24](#_Toc494408822)

[5.5.1. Các bước để truy cập Tomcat Manager: 25](#_Toc494408823)

[5.6. Giám sát trong Tomcat 26](#_Toc494408824)

[5.6.1. Tông quan về Server Status trên Tomcat 26](#_Toc494408825)

[5.6.2. Complete Server Status trên Tomcat 27](#_Toc494408826)

[5.6.3. Application List 27](#_Toc494408827)

[5.6.4. JVM 28](#_Toc494408828)

[5.6.5. Connections on the HTTP port (8080) 28](#_Toc494408829)

[5.6.6. Connections on the AJP 28](#_Toc494408830)

[Phần 6. Logging trong Tomcat 30](#_Toc494408831)

[6.1. Tổng quan: 30](#_Toc494408832)

[6.2. JULI: 30](#_Toc494408833)

[6.2.1. Giới thiệu 30](#_Toc494408834)

[6.2.2. Loggers, appenders, và layouts 30](#_Toc494408835)

[6.2.3. Các thành phần: 30](#_Toc494408836)

[6.3. Các loại Logging trong Tomcat 31](#_Toc494408837)

[6.3.1. Application log 31](#_Toc494408838)

[6.3.2. Server log 32](#_Toc494408839)

[6.3.3. Console log 32](#_Toc494408840)

[6.3.4. Access log 32](#_Toc494408841)

[6.3.5. Host manager 34](#_Toc494408842)

[6.3.6. Các mức độ log trong Tomcat 34](#_Toc494408843)

[6.4. Log4j 35](#_Toc494408844)

[6.4.1. Giới thiệu 35](#_Toc494408845)

[6.4.2. Log level trong log4j 35](#_Toc494408846)

[Phần 7. Security Manager trong Tomcat 37](#_Toc494408847)

[7.1. Tổng quan 37](#_Toc494408848)

[7.2. WARNING: 37](#_Toc494408849)

[7.3. Permissions 37](#_Toc494408850)

[7.4. Standard Permissions 37](#_Toc494408851)

[7.5. Cấu hình Tomcat với SecurityManager 38](#_Toc494408852)

[7.6. Chính sách mặc định sử dụng cho Tomcat 38](#_Toc494408853)

[7.7. Khởi động Tomcat với SecurityManager 38](#_Toc494408854)

[7.8. Quyền cho gói WAR files 38](#_Toc494408855)

[7.9. Cấu hình Package Protection trong Tomcat. 39](#_Toc494408856)

[Phần 8. Phân tích, giải quyết lỗi trong Tomcat 40](#_Toc494408857)

[8.1. Các vấn đề phổ biến: 40](#_Toc494408858)

[8.2. 1 số vấn đề sẽ gặp phải trong real-time production support: 40](#_Toc494408859)

[8.3. Cách giải quyết lỗi khi gặp phải: 41](#_Toc494408860)

[8.3.1. Vấn đề 41](#_Toc494408861)

[8.3.2. Làm sao để tìm ra và sửa lỗi: 41](#_Toc494408862)

[8.4. Phân tích khi gặp lỗi trên Web App – App chạy chậm 41](#_Toc494408863)

[8.4.1. Phân tích vấn đề: 41](#_Toc494408864)

[8.4.2. Cách giải quyết vần đề: 42](#_Toc494408865)

[8.4.3. Sửa lỗi tại góc nhìn User 42](#_Toc494408866)

[8.4.4. Sửa lỗi tại Webserver 42](#_Toc494408867)

[8.4.5. Sửa lỗi tại Java process 43](#_Toc494408868)

[8.4.6. Sửa lỗi tại mức Database 45](#_Toc494408869)

[8.5. Cách thu thập thread dump trong Tomcat 45](#_Toc494408870)

[8.5.1. Giới thiệu 45](#_Toc494408871)

[8.5.2. Lấy Thread dump bằng Kill command 45](#_Toc494408872)

[8.5.3. Sử dụng jstack 46](#_Toc494408873)

[8.5.4. Phân tích thread dump 47](#_Toc494408874)

[8.6. Công cụ đánh giá Web server 48](#_Toc494408875)

[8.6.1. ApacheBench 48](#_Toc494408876)

[8.6.2. Jmeter 48](#_Toc494408877)

[Phần 9. Cơ chế Class Loader trong Tombat 49](#_Toc494408878)

[9.1. Tổng quan 49](#_Toc494408879)

[9.2. Định nghĩa Class Loader: 49](#_Toc494408880)

[9.2.1. Bootstrap: 49](#_Toc494408881)

[9.2.2. System: 49](#_Toc494408882)

[9.2.3. Common: 50](#_Toc494408883)

[9.2.4. WebappX 51](#_Toc494408884)

[9.2.5. Theo góc nhìn của web app, class or resource loading theo: 51](#_Toc494408885)

[Phần 10. Proxy trong Tomcat 52](#_Toc494408886)

[10.1. Tích hợp Tomcat với Apache Web server 52](#_Toc494408887)

[10.2. User request flow (web/application level) 52](#_Toc494408888)

[10.3. Tại sao sử dụng Apache HTTP server 52](#_Toc494408889)

[10.4. Cài đặt Apache HTTP 53](#_Toc494408890)

[10.4.1. Cài đặt apache HTTP trên Linux. 53](#_Toc494408891)

[10.5. Apache Jserv protocol (AJP) 54](#_Toc494408892)

[10.6. Cài đặt và cấu hình mod\_jk 55](#_Toc494408893)

[10.6.1. Cài đặt Installation of mod\_jk 55](#_Toc494408894)

[10.6.2. Cấu hình mod\_jk trong Apache 56](#_Toc494408895)

[10.7. mod\_proxy configuration 58](#_Toc494408896)

[10.7.1. Cài đặt module mod\_proxy 59](#_Toc494408897)

[10.7.2. Các bước cấu hình mod\_proxy 59](#_Toc494408898)

[10.8. So sánh giữa mod\_jk và mod\_proxy 60](#_Toc494408899)

[Phần 11. Clustering và Session Replication trong Tomcat 61](#_Toc494408900)

[11.1. Tổng quan 61](#_Toc494408901)

[11.2. Lợi ích sử dụng Cluster: 61](#_Toc494408902)

[11.3. Bất lợi khi sử dụng clustering: 62](#_Toc494408903)

[11.4. Kiến trúc Cluster 62](#_Toc494408904)

[11.4.1. Vertical clustering 62](#_Toc494408905)

[11.4.2. Horizontal clustering 63](#_Toc494408906)

[11.5. Sử dụng cluster đơn giản: 64](#_Toc494408907)

[11.6. Security 65](#_Toc494408908)

[11.7. Cài đặt Vertical clustering in Apache Tomcat 65](#_Toc494408909)

[11.7.1. Yêu cầu 65](#_Toc494408910)

[11.7.2. Cài đặt 65](#_Toc494408911)

[11.8. Horizontal clustering in Apache Tomcat 71](#_Toc494408912)

[11.8.1. Yều cầu 71](#_Toc494408913)

[11.8.2. Cài đặt 71](#_Toc494408914)

[11.9. Testing of the clustered instance 78](#_Toc494408915)

[11.10. Tài liệu mở rộng Cluster 79](#_Toc494408916)

[11.10.1. Cluster Basics 79](#_Toc494408917)

[11.10.2. Giới thiệu 79](#_Toc494408918)

[11.10.3. Mô hình sử dụng 80](#_Toc494408919)

[11.10.4. Thông tin về Cluster 80](#_Toc494408920)

[11.10.5. Liên kết session sau khi gặp lỗi, session chuyển tới node chịu lỗi 80](#_Toc494408921)

[11.10.6. Cấu hình đơn đơn giản: 81](#_Toc494408922)

[11.10.7. Kiến trúc Cluster: 84](#_Toc494408923)

[11.10.8. Kịch bản Cluster Tomcat: 84](#_Toc494408924)

[11.10.9. The Cluster object 86](#_Toc494408925)

[11.10.10. The ClusterManager object 88](#_Toc494408926)

[11.10.11. The Cluster Channel object 88](#_Toc494408927)

[11.10.12. The Cluster Valve object (Van Cluster object) 89](#_Toc494408928)

[11.10.13. The Cluster Deployer object 90](#_Toc494408929)

[11.11. Apache Tribes 92](#_Toc494408930)

[11.11.1. Giới thiệu 92](#_Toc494408931)

[11.11.2. Định nghĩa Tribes 93](#_Toc494408932)

[11.11.3. Giới thiệu tính năng 93](#_Toc494408933)

# Tổng quan về Tomcat

## Giới thiệu

- Apache Tomcat là một Java Servlet được phát triển bởi Apache Software Foundation. Tomcat thi hành các ứng dụng Java Servlet và JavaServer Pages (JSP). Tomcat cung cấp một máy chủ HTTP cho ngôn ngữ Java thuần túy.

- Apache Tomcat là một phần mềm mã nguồn mở để chạy các ứng dụng web có nền tảng là ngôn ngữ Java. Tomcat được hỗ trợ và tăng cường bởi một nhóm tình nguyện viên từ cộng đồng mã nguồn mở của Java.

- Apache Tomcat rất ổn định và có tất cả các tính năng của một ứng dụng web thương mại nhưng đi kèm theo giấy phép mã nguồn mở của Apache. Tomcat cũng cung cấp một số chức năng bổ sung như tomcat manager application, speciallized realm imlementation và tomcat valves.

- Các phiên bản của apache tomcat trùng với phiên bản và đặc điểm kỹ thuật của servlet java hoặc java servlet API. Tomcat 5.5X hỗ trợ Servlet API 2.3, tomcat 6.0X hỗ trợ servlet API 2.4 và tomcat 7.0 hỗ trợ servlet API 3.0. Ngoài Servlet versions API, phiên bản tomcat hỗ trợ phiên bản JSP API tương ứng.

- Apache Tomcat có các tính năng như chứng thực người dùng, virtual hosting, CGI, Java Servlet, SSI, hỗ trợ Ipv6, …

- Apache Tomcat hỗ trợ các hệ điều hành như windows, linux, OS, BSD, solaris,…

## Các thuật ngữ cần nhớ:

- “$CATALINA\_HOME” đại diện cho thư mục root cài đặt Tomcat.

VD: “$CATALINA\_HOME/README.txt” file = “README.txt” trong thư mục root Tomcat.

- “$CATALINA\_BASE” = ““$CATALINA\_HOME”

## Tổ chức thư mục tomcat:

“/bin” = chưa các script thực thi tomcat (\*.sh cho unix) (\*.bat cho window).

“/conf” = Cấu hình file. File cấu hình quan trọng nhất “server.xml”. Nó là file main config.

“/logs” = Log file mặc định xuất hiện tại đây.

“/webapp” = Nơi các web được host.

# Cài đặt Tomcat

## Cài đặt Tomcat trên Linux:

**Yêu cầu trước khi cài đặt:**

- Sử dụng CentOS 7.

- Đã cấu hình IP tĩnh cho Tomcat 8.5.

- Đã update hệ điều hành.

## Cài đặt Tomcat 8.5

**Bước 1:**

- Download package dạng \*.zip về Server

# wget

**Bước 2:**

- Cài đặt môi trường chạy Tomcat

# yum install java -y

**Bước 3:**

- Copy thư mục vừa giải nén

# cp -R ~/apache-tomcat/ /opt/apache-tomcat

**Bước 4:**

- Truy câp thư mục thực thi Tomcat

# cd /opt/apache-tomcat/bin

**Bước 5:**

- Cấp quyền thực thi cho Script file

# chmod +x \*.sh

**Bước 6:**

- Khởi động Tomcat

- Kiểm tra version Tomcat

# ./version.sh

Using CATALINA\_BASE: /opt/apache-tomcat-test

Using CATALINA\_HOME: /opt/apache-tomcat-test

Using CATALINA\_TMPDIR: /opt/apache-tomcat-test/temp

Using JRE\_HOME: /usr

Using CLASSPATH: /opt/apache-tomcat-test/bin/bootstrap.jar:/opt/apache-tomcat-test/bin/tomcat-juli.jar

Server version: Apache Tomcat/8.5.20

Server built: Aug 2 2017 21:35:49 UTC

Server number: 8.5.20.0

OS Name: Linux

OS Version: 3.10.0-514.el7.x86\_64

Architecture: amd64

JVM Version: 1.8.0\_141-b16

JVM Vendor: Oracle Corporation

- Kiểm tra cấu hình mặc định

# ./configtest.sh

Using CATALINA\_BASE: /opt/apache-tomcat-test

Using CATALINA\_HOME: /opt/apache-tomcat-test

Using CATALINA\_TMPDIR: /opt/apache-tomcat-test/temp

Using JRE\_HOME: /usr

Using CLASSPATH: /opt/apache-tomcat-test/bin/bootstrap.jar:/opt/apache-tomcat-test/bin/tomcat-juli.jar

Sep 24, 2017 2:04:09 PM org.apache.catalina.startup.VersionLoggerListener log

INFO: Server version: Apache Tomcat/8.5.20

Sep 24, 2017 2:04:09 PM org.apache.catalina.startup.VersionLoggerListener log

INFO: Server built: Aug 2 2017 21:35:49 UTC

Sep 24, 2017 2:04:09 PM org.apache.catalina.startup.VersionLoggerListener log

INFO: Server number: 8.5.20.0

Sep 24, 2017 2:04:09 PM org.apache.catalina.startup.VersionLoggerListener log

INFO: OS Name: Linux

Sep 24, 2017 2:04:09 PM org.apache.catalina.startup.VersionLoggerListener log

INFO: OS Version: 3.10.0-514.el7.x86\_64

Sep 24, 2017 2:04:09 PM org.apache.catalina.startup.VersionLoggerListener log

…

INFO: Initialization processed in 494 ms

- Chạy service Apache Tomcat

# ./startup.sh

Using CATALINA\_BASE: /opt/apache-tomcat-test

Using CATALINA\_HOME: /opt/apache-tomcat-test

Using CATALINA\_TMPDIR: /opt/apache-tomcat-test/temp

Using JRE\_HOME: /usr

Using CLASSPATH: /opt/apache-tomcat-test/bin/bootstrap.jar:/opt/apache-tomcat-test/bin/tomcat-juli.jar

Tomcat started.

- Kiểm tra service Tomcat với

# ps aux | grep tomcat

## Để tắt service Tomcat

# ./shutdown.sh

Using CATALINA\_BASE: /opt/apache-tomcat-test

Using CATALINA\_HOME: /opt/apache-tomcat-test

Using CATALINA\_TMPDIR: /opt/apache-tomcat-test/temp

Using JRE\_HOME: /usr

Using CLASSPATH: /opt/apache-tomcat-test/bin/bootstrap.jar:/opt/apache-tomcat-test/bin/tomcat-juli.jar

**Truy cập giao diện quản trị Tomcat**

http://localhost:8080

## Cấu hình user truy cập nội bộ Apache Manager

**Bước 1:**

- Cấu hình file tomcat-user.xml

- Mặc định các user sẽ không tồn tại, cần được cấu hình

# vim ../tomcat../conf/tomcat-users.xml

###

<role rolename="admin-gui"/>

<user username="tomcat" password="123456" roles="admin-gui"/>

<role rolename="manager-gui"/>

<user username="tomcat" password="123456" roles="manager-gui"/>

**Bước 2:**

- Thêm quyền truy cập Tomcat Manager từ xa

- Thêm mới file nếu chưa tồn tại

# vim conf/Catalina/localhost/manager.xml

## thêm

<Context privileged="true" antiResourceLocking="false"

docBase="${catalina.home}/webapps/manager">

<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="^.\*$" />

</Context>

**Bước 3:**

- Khởi động lại Tomcat

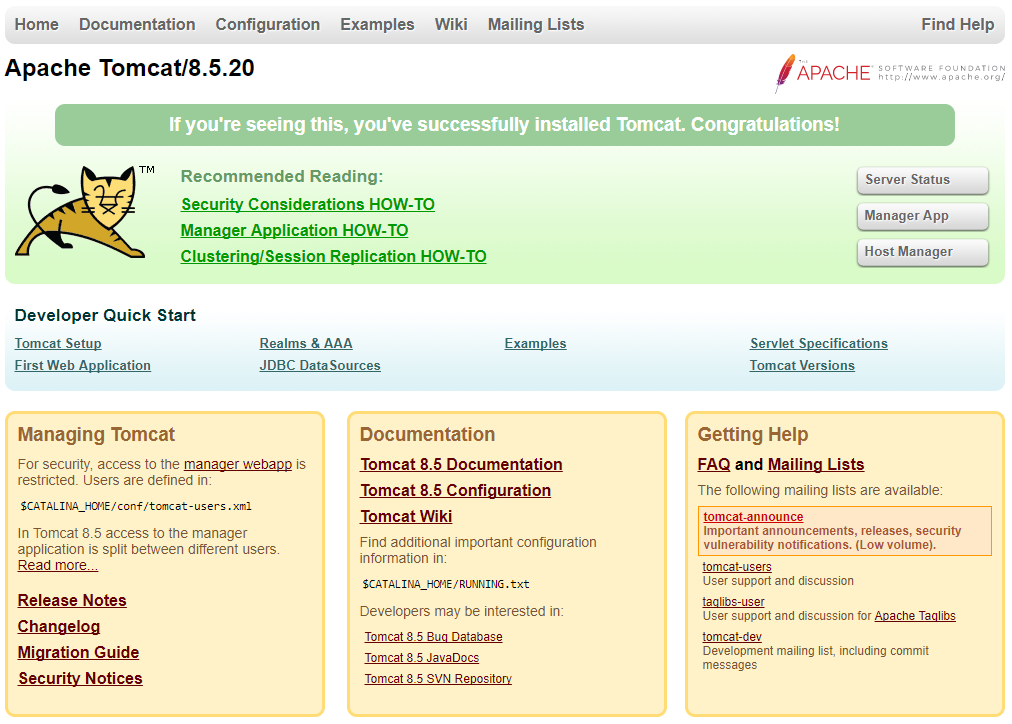
# ./shutdown

# ./startup

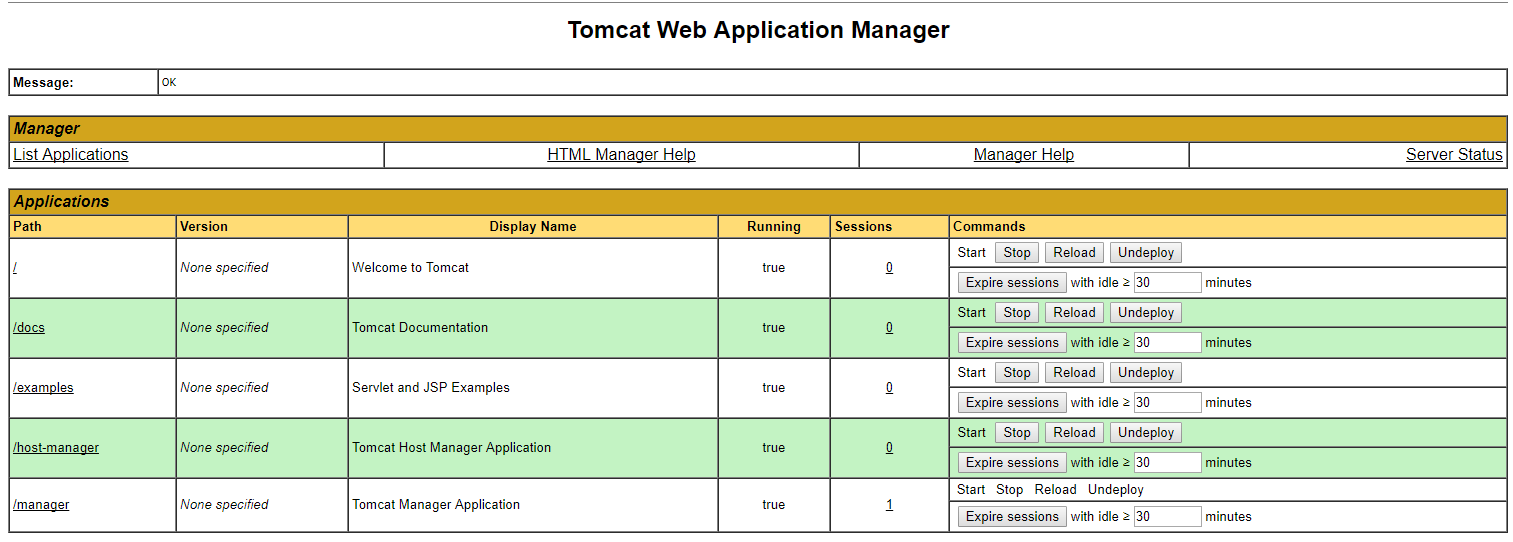
Note:

- Mặc định tại Tomcat 8.5, Tomcat Manager chỉ có thể được truy cập từ Localhost, vì thế, để có thể truy cập thông qua remote, ta cần thêm quyền cho Tomcat.

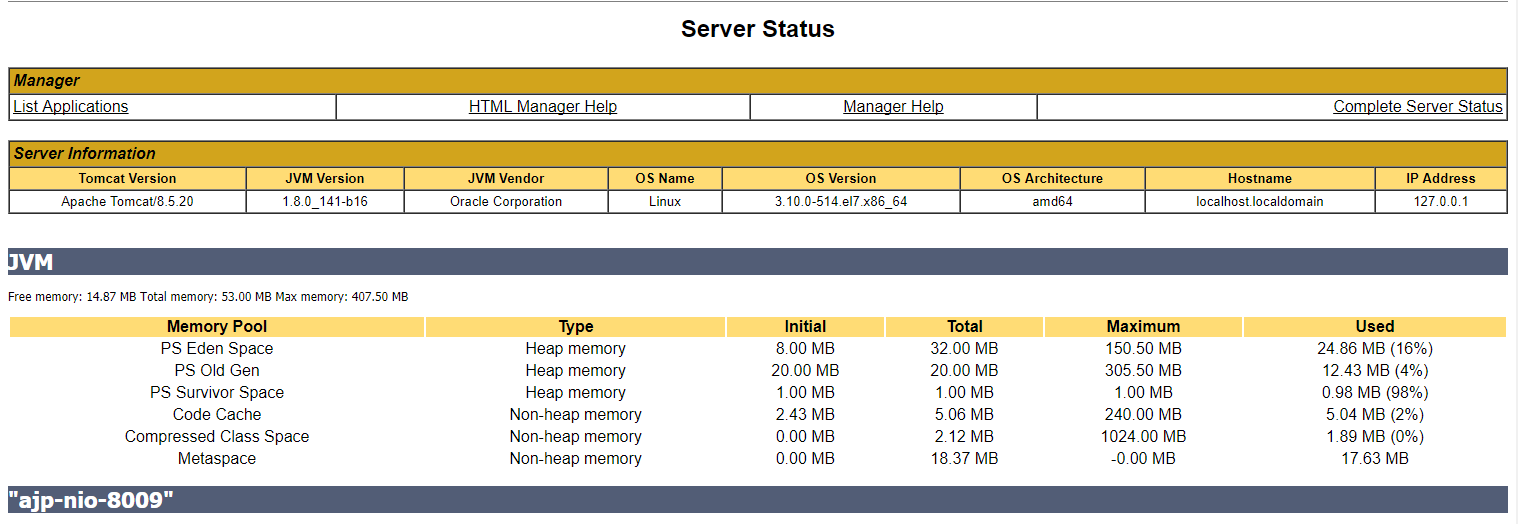
**Giao diện chính Tomcat**



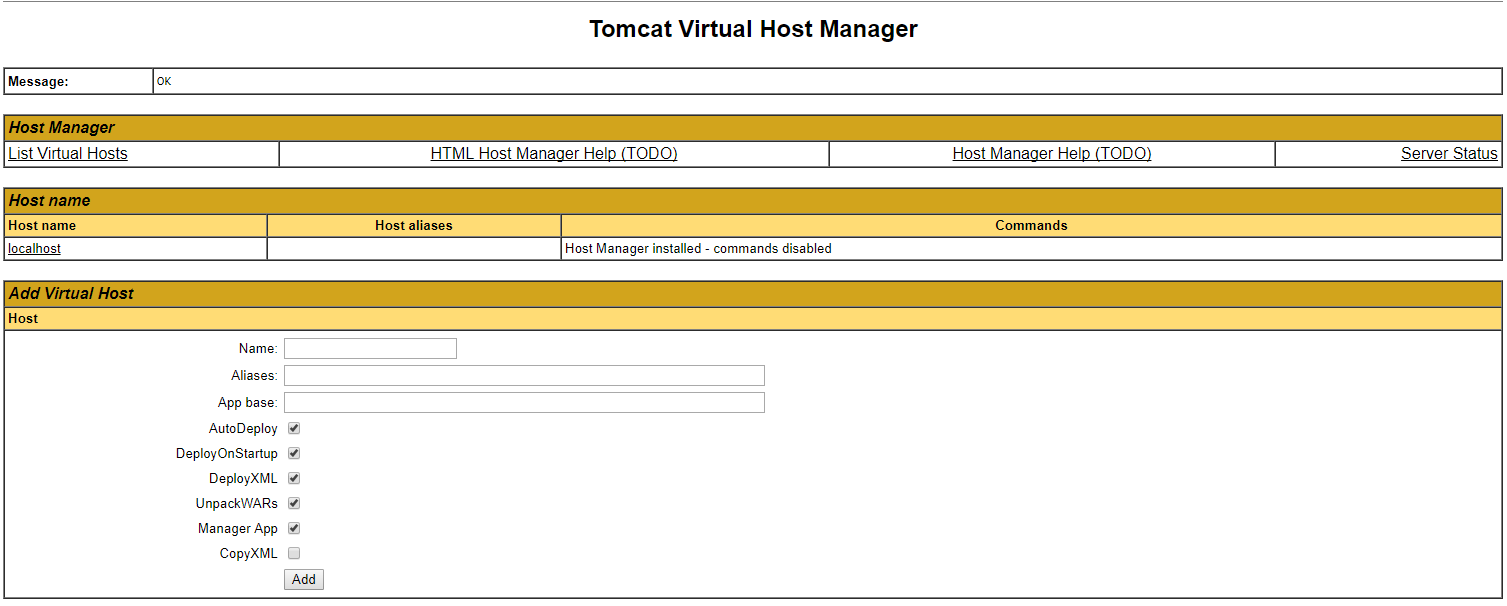
**Tomcat Web App Manager**



**Tomcat Server Status**



**Tomcat Virtual Host Manager**



# Triển khai Web Application trên Tomcat.

## Giới thiệu:

- Deployment là thuật ngữ sử dụng cho công việc cài đặt Web application (có thể là 3rd party war hoặc cusweb app) tới Tomcat server.

## Vấn đề cơ bản khi triển khai trong Tomcat

### Triển khai web app có thể thực hiện bằng nhiều cách:

+ Tĩnh (Statically): web app được cài trước khi Tomcat start

+ Động: Trực tiếp triển khai web app (real time) (tính năng auto-deployment feature) hoặc triển khai thông qua Tomcat Manager web app.

### Tomcat Manager

- Web interface, sử dụng để quản trị Tomcat web server thông qua giao diện trực quan (HTML GUI) hoặc lập trình thông qua API.

### Thuật ngữ Contexts

- Khi nói về triển khai web app, khái niệm về Context cần phải được hiểu rõ. Context là định nghĩa web app trong Tomcat.

- Để cấu hình Context, yêu cầu sử dụng Context Descriptor. Context Descriptor đơn giản là XML file chứa các thông tin Context. Ở các phiên bản trước, các cấu hình Context Descriptor thường được lưu trực tiếp trong “server.xml” nhưng tại các phiên bản hiện tại đã được tách ra.

- Ngoài ra, Tomcat Manager, TCD cũng sử dụng Context Descriptor để thực hiện các công việc giống nhau.

### Context Descriptor được lưu tại:

+ $CATALINA\_BASE/conf/[enginename]/[hostname]/[webappname].xml

+ $CATALINA\_BASE/webapps/[webappname]/META-INF/context.xml

- (1) được đặt tên theo webapp ([webappname.xml]), (2) sử dụng cho mặc định. Nếu Context Descriptor không cung cấp cho Context, Tomcat config sẽ sử dụng config mặc định.

## Deployment khi Tomcat được khởi động

- Nếu bạn không sử dụng Tomcat Manager or TCP. Ta sẽ deploy web app tĩnh.

- Vị trí deploy web app được gọi là “appBase”, nó sẽ có giá trị đặc biệt cho mỗi host.

- Ta có thể copy “exploded web application resource” hoặc “compressed web application resource” tới thư mục này.

- Web app được đặt trong 1 vị trí rõ ràng tại (mặc định localhost) “appBase” host attribute (mặc định “appBase” tại "$CATALINA\_BASE/webapps"). Chúng sẽ được deploy khi Tomcat khởi động khi trường “deployOnStartup = true”

### Quá trình triển khai Web App khi Tomcat khởi động:

+ Bất kỳ Context Descriptor được tìm thấy sẽ được triển khai trước.

+ Exployed web app không được tham chiếu bới Context Descriptor sẽ được deploy. Nếu chúng liên kết với .WAR file trong “appBase” và war file mới hơn thì Tomcat sẽ chọn .WAR để deploy.

+ Các file WAR được deploy.

## Deploying trển Tomcat server đã được khởi động

- Có thể deploy app trong khi Tomcat đang chạy

- Nếu trường “autoDeploy = true”, host sẽ có khả năng deploy và update web app động.

VD: copy war file vào “appBase”.

- Để thực hiện công việc này, host cần tiến trình ngầm

- Khi set “autoDeploy = true”, tomcat cho phép:

+ Triển khai .WAR file khi copy tới Host appBase

+ Triển khai “exploded web applications” khi đã copy tới Host appBase.

+ Triển khai lại web app đã được deployer từ .WAR file (Được update). Trong trường hợp “exploded web application” và .WAR được thêm lại.

+ Load lại Web app nếu “/WEB-INF/web.xml” hoặc bắt kỳ tài nguyên nào được định nghĩa như “WatchedResource” được updated.

+ Load lại Web app nếu Context Descriptor file của nó (đã được deployed) được updated.  
+ Triển khai lại các web app phụ thuộc nếu Global hoặc per-host Context Descriptor file được updated.

+ Triển khai lại Web app nếu Context Desscriptor file (với tên file tương ứng với Context path của web app đã triển khai trước đây) được thêm vào tại thư mục “$CATALINA\_BASE/conf/[enginename]/[hostname]/”.

+ Bỏ triển khai web app nếu “document base (docBase)” bị xóa.

Note:

- Mở gói sẽ không xảy ra nếu Host được cấu hình trước. Vì thế .WAR sẽ không được exploded nếu trường “unpackWARs attribute” được set = false, trong trường hợp đó Web app sẽ đơn giản redeploy như compressed archive.

Note:

- Web application cũng sẽ reloading nếu cấu hình lại trong loader thay đổi (trong trường hợp classes bị theo dõi thay đổi).

## Triển khai Web App sử dụng Tomcat Manager

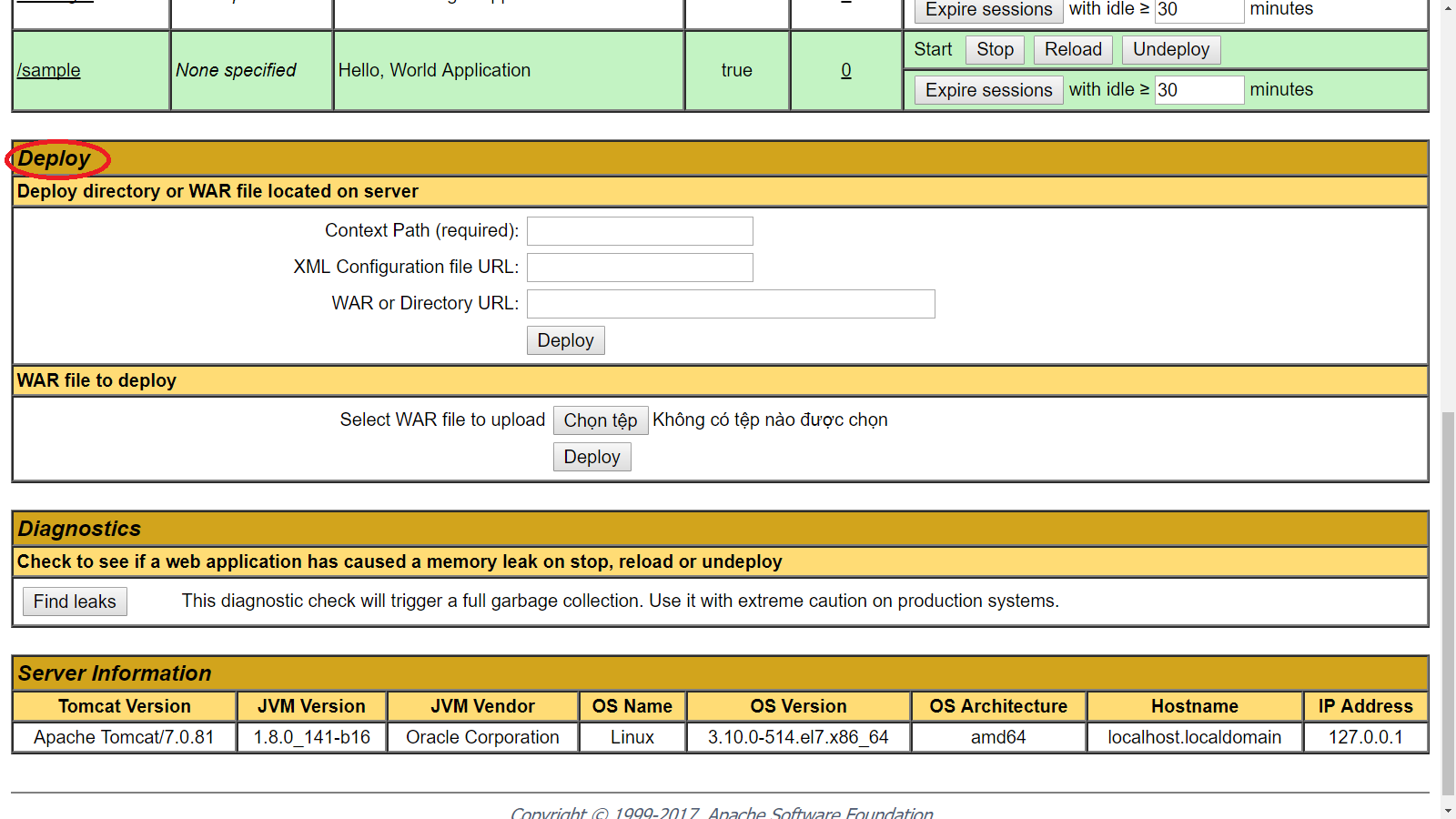
**Bước 1:**

- Truy cập <http://localhost:8080>

**Bước 2:**

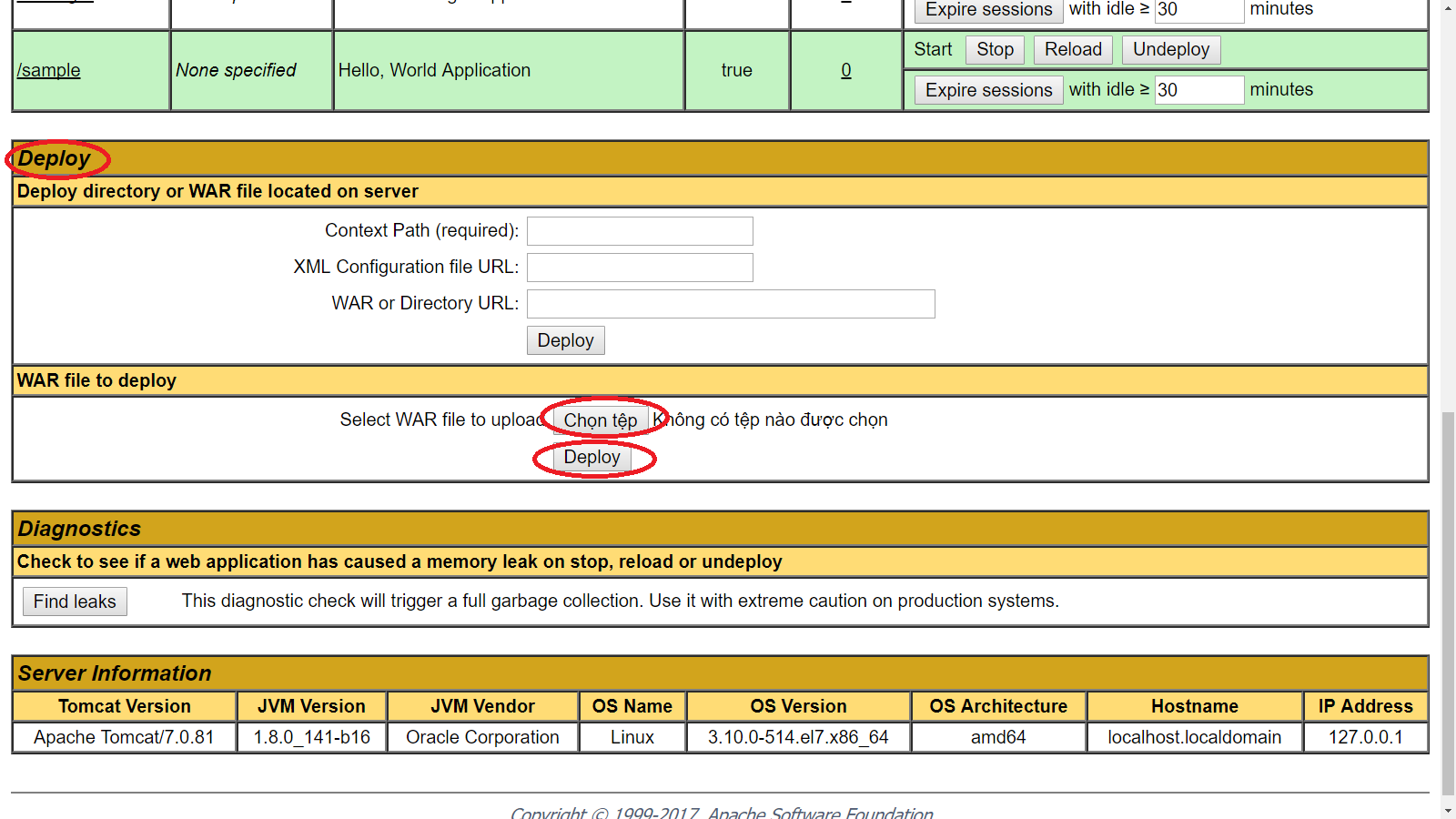
- Click “Manager App”





**Bước 3:**

- Tại Deploy path, chọn Deploy tab, chọn Browse lựa chọn WAR file. Lựa chọn Deploy.



## Triển khai trong Tomcat

- Về cơ bản, Tomcat cho phép ta sử dụng War file (file nén) và cả dạng project không nén.

### Kiến trúc của WebArchive

- Project web java được tổ chức theo qui tắc nhất định, vì thế, ta có thể tổ chức và triển khai trên tomcat

- Tất cả servlets, classes, static files, and other resources thuộc về web application được tổ chức dưới dạng cây thư mục.

- Root của cấu trúc này định nghĩa là document root trong web app.

- Tất cả file thuộc cấu trúc root direc có thể sử dụng để phục vụ client, trừ file thuộc thư mục đặc biệt “WEB-INF”, đặt trong root directory. Tên web app được sử dụng cho việc quyết định request cho mỗi thành phần app.

Note:

Always place private files (files which are not required to serve to the client) in the WEB-INF directory, under the root directory. All files under WEB-INF are private, and are not served to the client.

### Tổ chức dạng thư mục:

+ WebApplicationName/: chứa tất cả static file như html, jsp files được lưu trữ. Thư mục này là document root trong web app.

+ /WEB-INF/web.xml: chưa các deployment descriptor cho web app. Nhưng nguồn đặc biệt của web sẽ được lưu tại đây.

+ /WEB-INF/classes: chứa tất cả server-side class hoặc các app thuộc phần mềm thứ 3

+ /WEB-INF/lib: thư mục chưa JAR files sử dụng để hoàn thành JSP.

+ web.xml: Chứa thông tin về các dynamic file và các cấu hình khác như thông tin về session time or, các nguồn data source (DB)

### Archive Files

- File sử dụng để triên khai Web App, nhận được từ Web developer. File này được nén, bao gồm tất cả thông tin về app, module class, static file, direc, deployment descriptor.

- Archive file thường được tạo bằng cách sử dụng JAR utility hoặc Ant JAR tool.

### Các loại file sử dụng cho việc triển khai Web App:

+ EJBs are packaged as .jar files

+ Web applications are packaged as .war files

+ Resource adapters are packaged as .rar files

+ Enterprise applications are packaged as .ear files, and can contain any combination of EJBs, web applications, and resource adapters

+ Web services can be packaged either as .ear files or as .war files

### Exploded archive directories (Project web app)

- Thư mục giải nén chứa tất cả các file, direc giống như JAR archive. Tuy nhiên các file và direc được giải nén ra thay vì đóng gói thành 1 archive duy nhất thông qua JAR utility.

- Đơn vị triển khai này được sử dụng khi:

+ Bạn muốn update 1 phần trong web app mà ko phải redeploy cả web app

+ Sử dụng Tomcat manager dynamically edit và lựa chọn 1 số giá trị deployment descriptor values trong gói triển khai

+ deploy 1 web app mà trong nó chứa các static file sẽ phải update theo chu kỳ.

### Các hoạt động trong triển khai

- Các phương thức được hỗ trở bởi Tomcat Manager

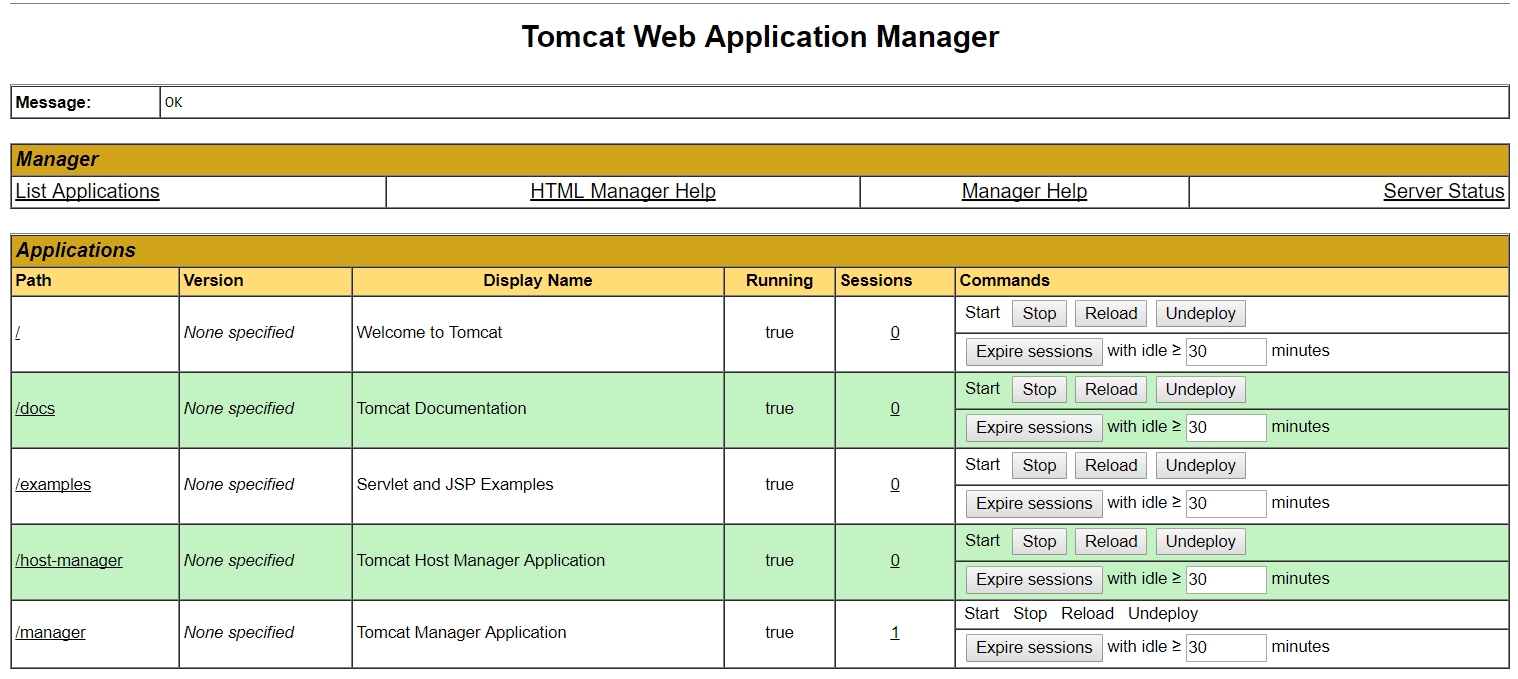
+ Deploy: Khiến source được target bởi server, loading các class vào class loaders, web app sẵn sàng phục vụ client.

+ Redeploy: sử dụng khi có update thành phần trong đơn vị triển khai, mà đơn vị nãy đã được triển khai trước đó, đang phục vụ client. Trong thời gian redeploy, web app sẽ không sẵn sàng để phục vụ client. (trong trạng thái stop)

+ Stop: unload các class, các file, khiến web app không sẵn sàng để phục vụ client. Nhưng web app này vẫn được target bởi server.

+ Start: nó sẽ reload lại các class, khiến web app sẵn sàng phục vụ client. Để chạy đòi hỏi web app phải đã được target bởi server, kết quả của việc deploy trước đó.

+ Undeploy: stop đơn vị triển khai, và xóa bỏ khỏi target server.



### Các trạng thái sử dụng khi triển khai

- Các trạng thái này được định nghĩa khi các đơn vị triển khai được load, target bởi server.

- Tomcat hỗ trợ 3 lựa chọn trạng thái:

+ Stage mode

+ Nostage mode

+ External\_stage mode

- Ý nghĩa các trạng thái:

|  |  |  |
| --- | --- | --- |
| Trạng thái | Các xử lý | Sử dụng khi |
| Stage | Tomcat admin đầu tiên sẽ copy tài nguyên của đơn vị triển khai vào staging directory, sau đó target servers sẽ deploy phiên bản copy. | Triển khai app cỡ nhỏ và vừa tới multi tomcat 7 server  Triển khai app nhỏ và vừa tới cluster |
| Nostage | Tomcat admin không copy file thuộc đơn vị triển khai. Thay vì đó, tất cả server deploy sẽ sử dụng chung 1 physical copy file thuộc đơn vị triển khai, nó sẽ được truy cập trực tiếp bởi tomcat admin và target server  Nostage deployment không khuyến kích sử dụng source dạng exployed archive. | Triển khai trên single server  Triển khai tới cluster trên multi-homed machine  Triển khai app lớn tới multi target hoặc cluster, ở đó deployment file được lưu trên server |
| External\_stage | Người quản trị Tomcat sẽ không copy file thuộc đơn vị triển khai. Thay vì thế, admin phải chắc chắn rằng deployment file được được phân bố tới staging direc localtion trước khi deployment  Với external\_stage deployment, tomcat admin yêu cầu copy deployment file với mục đích chứng thực (validation). | Triển khai khi bạn muốn kiểm soát việc phân phối các file deployment tới target servers.  Triển khai tới server thực thi nơi phần mềm thứ 3 hoặc script quản lý việc sao chep các copy của deployment file tới staging directory. |

### Các dạng có thể sử dụng khi muốn triển khai Web App trên Tomcat.

- Có rất nhiều cách để triển khai app trên tomcat. Nhưng có thể tổng kết lại thành 5 loại chính:

+ War deployment: Có thể triển khai bằng cách đưa War file vào CATALINA\_BASE direc và reset tomcat để view app. Đây là cách được sử dụng phổ biến trong môi trường triển khai

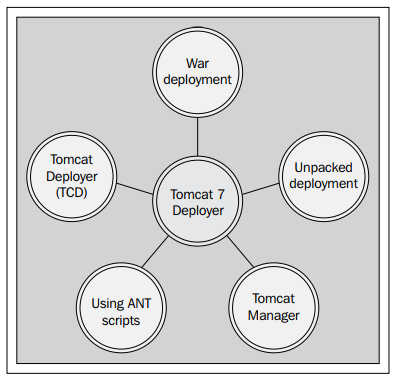
+ Unpacked deployment: Tại phương pháp này, War file được giải nén trong CATALINA\_BASE direc cho việc thực thi. Phương pháp này thường sử dụng khi phát triển server.

+ Tomcat Manager: Tool hữu ích cho việc triển khai, thường được sử dụng khi triển khai các app trong môi trường chính và thường nhất khi triển khai từ xa.

+ Using ANT scripts: Có thể triển khai bằng ANT scripts. Script này chứa thông tin về các source, nguồn, target file.

+ TCD (Tomcat Deployer): It is a tool which is used for application deployment. ANT should be installed for the TCD to be working and the Tomcat instance should be running. There is no need to install he TCD on the Tomcat instance.

### Mô hình các cách triển khai



e

# Quản trị Tomcat

## Giới thiệu

- Trong môi trường triển khai, sẽ rất hữu ích nếu có thể deploy 1 web app mới hoặc undeploy 1 web đang tồn tại mà không phải restart toàn bộ service. Đông thời có khả năng yêu cầu app reload lại chính nó thậm chí khi không khai báo reloadable trong Tomcat server config.

- Để hỗ trợ khả năng này, Tomcat thêm vào web app (install mặc đinh theo context path /manager) cho phép:

+ Triển khai 1 Web app mới bằng cách upload lên nội dung 1 WAR file.

+ Triển khai 1 Web app, với đường dẫn context cụ thể, từ server file system.

+ Liệt kê các Web app đã deployed, cũng như các session đang hoạt động triển các Web app.

+ Khởi động lại 1 Web app đã tồn tại, phản ánh các thay đổi trong nội dung “/WEB-INF/classes” hoặc “/WEB-INF/lib”.

+ Liệt kê giá trị OS và JVM property.

+ Liệt kê các tài nguyên global JNDI có sẵn, sử dụng cho tool triển khai bằng các chuẩn bị <ResourceLink> element bên trong <Context> deployment description.

+ Chạy app đang dừng (Khiến nó sẵn sàng trở lại)

+ Dừng lại app đã tồn tại (Khiến nó không sẵn sàng) nhưng không undeploy nó.

+ Loại bỏ triển khai Web app và xóa nó khỏi “document base directory” (Trừ khi nó được deployed từ file system).

- Mặc định Tomcat sẽ bao gồm Manager. Để thêm thực thi Manager web app Context tới 1 host mới, cài đặt manager.xml context config file trong “$CATALINA\_BASE/conf/[enginemane]/[hostname]”.

VD:

<Context privileged="true" antiResourceLocking="false"

docBase="${catalina.home}/webapps/manager">

<Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.0\.0\.1" />

</Context>

## Cấu hình truy cập Manager Application

- Giá trị “$CATALINA\_BASE” được tham chiếu đến thư mục base directory. Nếu không cấu hình Tomcat cho nhiều thực thi bằng cách setting “CATALINA\_BASE” thì “$CATALINA\_BASE” sẽ được set giá trị của “$CATALINA\_HOME” (thư mục cài đặt Tomcat).

- Sẽ không an toàn nếu để các cấu hình mặc định cho phép bất kỳ ai trong Internet truy cập vào Manager app trên server. Vì thế, Manager app sẽ yêu cầu bất kỳ ai truy cập vào để phải xác thực, sử dụng user name, passwd và quyền được gán (manager-xxx).

- Mặc định, sẽ không có bất kỳ user trong users file ($CATALINA\_BASE/conf/tomcat-user.xml)

## Các role cơ bản có sẵn:

+ manager-gui — Truy cập vào HTML interface.

+ manager-status — Chỉ truy cập vào “Server Status”

+ manager-script — Truy cập vào công cụ plain text interface  
Access to the tools-friendly plain text interface that is described in this document, and to the "Server Status" page.

+ manager-jmx — Truy cập vào JMX proxy interface và “Server Status page”.

- Giao diện HTML được bảo vệ khỏi CSRF (Cross-Site Request Forgery) nhưng text và JMX interface thì không. Vì thế, ta cần cẩn thận khi cho phép text và JMX interface.

- Nếu bạn sử dụn web brower để acces Manager app, user truy cập cần có quyền manager-script or manager-jmx role.

- Không bao giờ cấp manager-script or manager-jmx role cho user có quyền manager-gui.

Note:

Tìm hiểu thêm tại: <http://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html>

## Realm là gì?

- Là “database” username và passwd cho phép xác thực user của web app (hoặc tập các app), cùng với đó là mỗi quyền được gán tới mỗi user. Ta có thể coi roles như là 1 group tring Unix/Linux system, vì truy cập vào Web app, người dùng được cấp quyền cụ thể. 1 người dùng sẽ có 1 số quyền liên kết với chính user đó.

Tìm hiểu thêm: <http://tomcat.apache.org/tomcat-8.5-doc/realm-howto.html>

# Giám sát Tomcat (Monitoring)

## Tổng quan

- Giám sát là 1 phần quan trọng trong công việc IT administrator. Khiến việc dự đoán và đánh giá, phát hiện lỗi khi xảy ra.

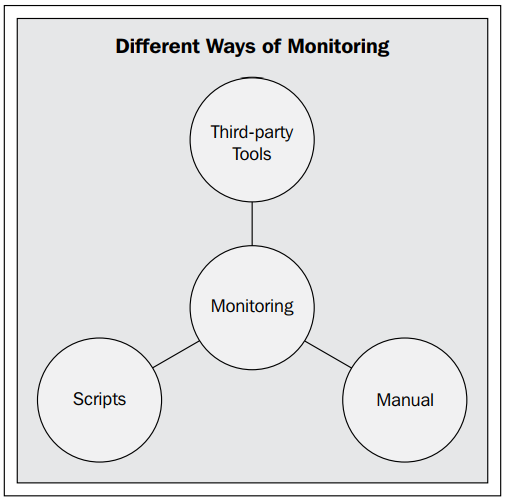
- Trước khi tìm hiểu vấn đề giám sát tomcat 7, ta sẽ giải thích vai trò quan trọng của giám sát trong thực tế. Trong môi trường thực tế, hệ thống có thể gặp lỗi vì rất nhiều lý do như sự cố (glitch) network, cpu tăng đột ngột, JVM crash, .. Và có một số hệ thống rất quan trọng, vì nó thu thâp, thực hiện các giao dịch như hệ thống ngân hàng, các sysad chỉ biết được vấn đề xảy ra khi nhận được phàn nàn từ khác hàng. Nó sẽ gây ảnh hướng tới vấn đề kinh doanh. Nếu hệ thống giám sát được set up trên server, web admin có thể nhận được thông báo về trạng thái hệ thống khi gặp lỗi, và ngay lập tức thực hiện các hoạt động sửa lỗi, từ đó giảm được thời gian downtime hệ thống.

**Note:**

IT administrators support thousand of servers, it's practically impossible to validate the system every day. Hence, monitoring is very helpful.

## Các cách khác nhau để giám sát

- Hiện nay, hạ tầng ngày càng tăng cao, nó sẽ khó cho admin quản lý các server. Để xác định vấn đề, giảm thời gian downtime, giám sát config hệ thông. Ta có thể cấu hình multi-level monitoring trên hệ thống, dự trên yêu cầu hạ tầng như OS, Web, App, Database, trên mỗi mức ứng dụng. Có nhiều các khác nhau để cấu hình giám sát multi-level.



## Việc giám sát có thể thực hiện xong bằng 3 cách trên hệ thống:

- Công cụ Third-party

+ Cài đặt các phần mềm thứ 3 như Wily, SiteScope, Nagios, ..

+ Các loại công cụ giám sát được sử dụng cho môi trường doanh nghiệp, khi có thể có hàng trăm server với kiến trúc khác nhau như web, app, db, filesystem server, ..

- Scripts + Script được sử dụng cho việc giám sát, trong các trường hợp đặc biệt như lấy kết quá số lượng user được log vào nhưng khoảng thời gian đặc biệt và cho nhưng quyền user đặc biệt.

+ Sử dụng trên mọi tổ chức

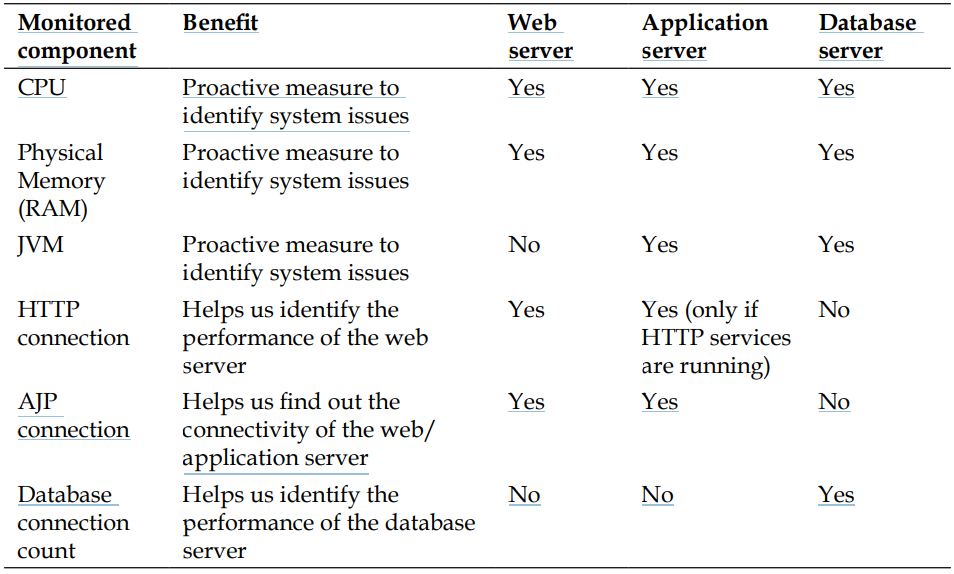
- Manual

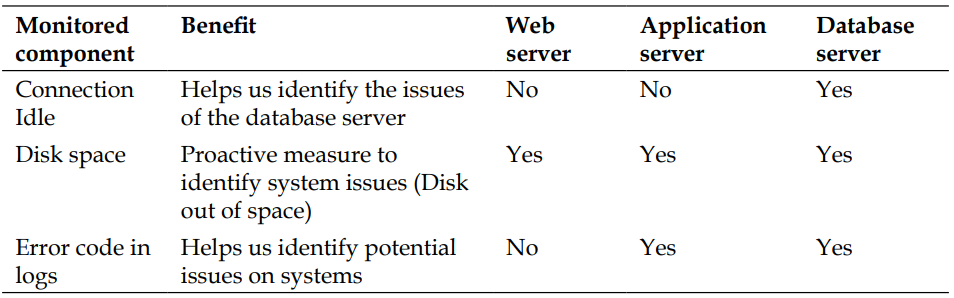
+ Process được sử dụng khi hiệu năng của app xuồng thấp vì nhưng nhưng module đặc biệt

+ Hầu như sử dụng khi số hệ thống nhỏ

## Cấu hình Monitor cho Web app, Web server, Database

- Các thành phần sẽ được giám sát





## Tomcat Manager trong Tomcat

- Tomcat manager là công cụ mặc định để quản lý hoạt động Apache Tomcat.

- Nó chó phép các IT admin quản lý từ xa app và giám sát hệ thống.

- Các điểm mạnh của Tomcat Manager:

+ Allow remote deploy, rollback, start, and stop features for the administrator

+ Cung cấp thông tin giám sát về trạng thái app và server.

+ Admin không cần tại office 24x7. Trong trường hợp các vấn đề, hẹ có thể log vào Tomcat Manager để giải quyết các vấn đề.

### Các bước để truy cập Tomcat Manager:

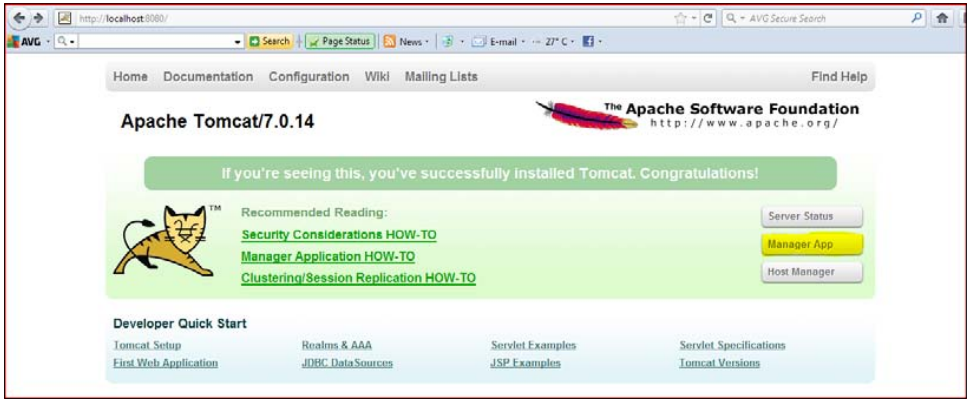
**Bước 1:**

- Truy cập Tomcat Manager sử dụng URL http://localhost:8080/

**Bước 2:**

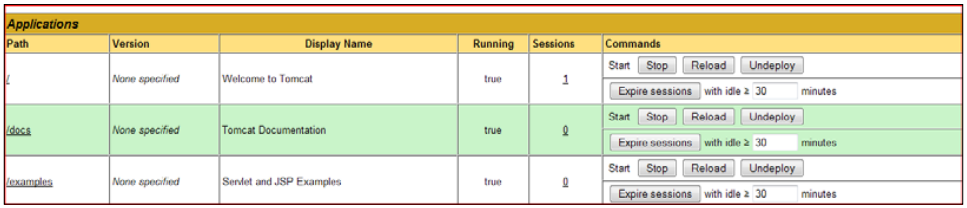
- Chọn Manager App. Nó sẽ cần mật khẩu

- Mật khẩu được set trong “tomcat\_user.xml”, khi “tomcat\_user.xml” có thể tìm thấy tại “TOMCAT\_HOME/CONF”.



**Bước 3:**

- Tại màn hình Tomcat Manager, nó sẽ đưa ra bức tranh tổng quan về app deploy, trạng thái server, chuẩn đoán (diagnostics), thông tin server, ..

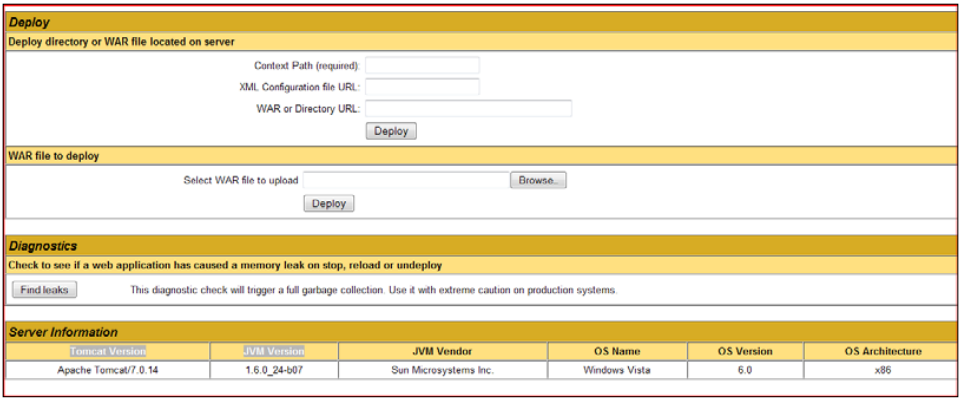


**Các tính năng trong Tomcat 7 Manager:**

+ Deployment of a new application

+ Diagnostic (memory or connection leak)

+ Server information



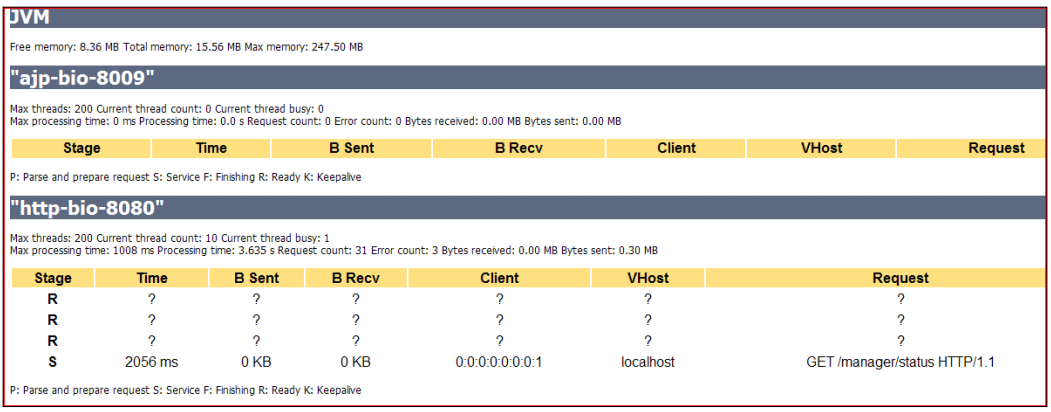
## Giám sát trong Tomcat

- Có thể sử dụng Tomcat Manager để giám sát Tomcat. Mặc định, Tomcat Manager cung cấp trạng thái server với các thông tin mô tả trạng thái hệ thống. Thông tin này sẽ rất hữu ích cho admin khi chữa lỗi. Bên canh đó, admin không cần log và máy để thu thập thông tin, tomcat manager sẽ thực hiện vấn đề đó.

### Tông quan về Server Status trên Tomcat

- bảo gồm thông tin về JVM, HTTP và HTTPS connection.

- Truy cập: <http://localhost:8080/manager/status>



- Thông tin trạng thái:

|  |  |
| --- | --- |
| - JVM  + Free memory  + Used memory  + Total memory | Connections on the HTTP port  + Max threads  + Current thread count  + Current thread busy  + Max processing time (ms)  + Processing time (s)  + Request count  + Error count  + Bytes received (MB)  + Bytes sent (MB) |
| Connections on the AJP  + Max threads  + Current thread count  + Current thread busy  + Max processing time (ms)  + Processing time (s)  + Request count  + Error count  + Bytes received (MB)  + Bytes sent (MB) |  |

### Complete Server Status trên Tomcat

- Xem toàn bộ trạng thái trên Tomcat.

http://192.168.2.131:8080/manager/status/all

### Application List

- Liệt kê các host trong Tomcat, URL mapping truy cập app



**- Các thông tin về mỗi host:**

+ Khi click vào 1 host trong list, nó sẽ chuyển ta đến thông tin về host đó

+ Their statuses such as the status of the application response, servlet response,

and the JSP responses



- Application response: Thông tin trả lại của ứng dụng và trạng thái hiện tại của app.

- Servlet details: Hiện thị respone time của servlet deployed trên app

- JSP: In this section, the dashboard displays the response time of the JSP deployed for a sample application with the following parameters:

### JVM

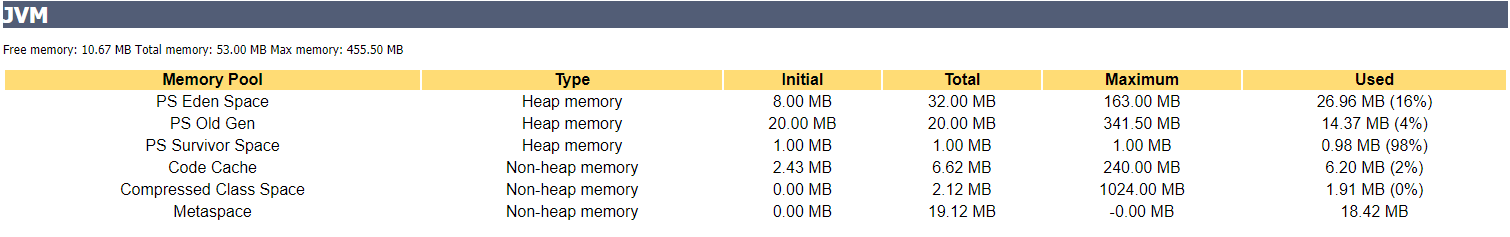
- Hiện thị JVM memory sử dụng trong thực thi Tomcat.

Bao gồm:

+ Free memory

+ Used memory

+ Total memory

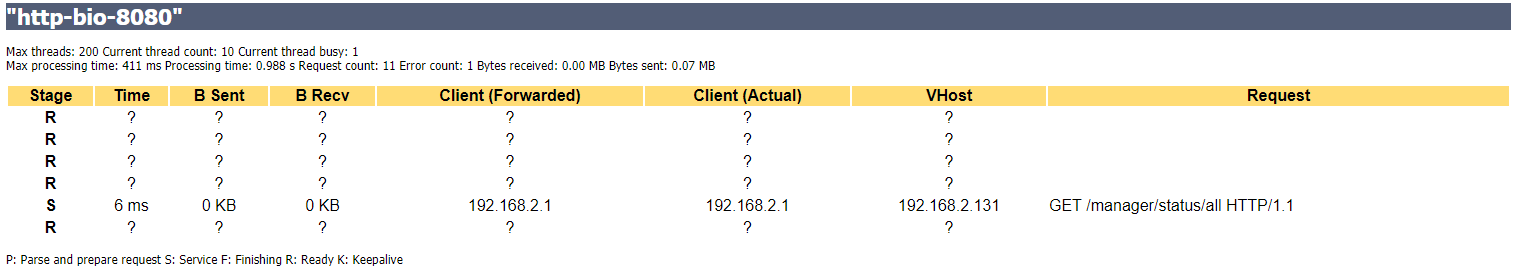


### Connections on the HTTP port (8080)

- Phần này hiển thị các HTTP connection status cho thực thi tomcat

- Bao gồm:

Max threads • Current thread count • Current thread busy • Max processing time (ms) • Processing time (s) • Request count • Error count • Bytes received (MB) • Bytes sent (MB)

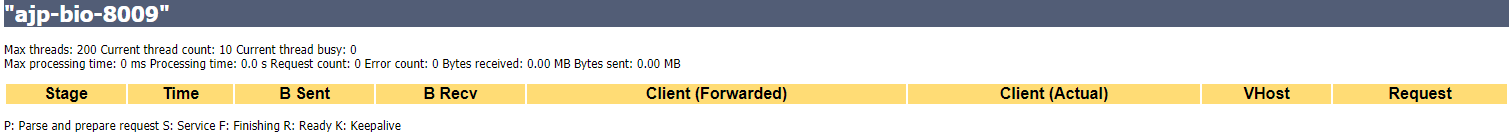


### Connections on the AJP

- Hiện thị trạng thái cái kết nối AJP connection status.

- Bao gồm:

Max threads • Current thread count • Current thread busy • Max processing time (ms) • Processing time (s) • Request count • Error count • Bytes received (MB) • Bytes sent (MB)



e

# Logging trong Tomcat

## Tổng quan:

- Logging service là 1 phần quan trọng trong công việc hàng ngày của admin và dev. Nó giúp cho việc quan trị app từ giai đoạn phát triển đến giai đoạn thực thi chính. Logging services giúp bạn tìm các vấn đề thực xảy ra trong web app. Đồng thời, nó cũng là 1 phần thiết yếu trong việc nâng cao hiệu năng cho bất kỳ app nào.

## JULI:

### Giới thiệu

- Ở phiên bản trước, Tomcat sử dụng thành phần cơ bản trong Apace logging service. Đây là điểm yếu trong kỹ thuật logging vì nó chỉ xử lý 1 JVM config duy nhất và khiến việc thiết lập logging cho mỗi class loader trở nên khó khăn.

- Để giải quyết vấn đề này, Tomcat dev đã giới thiệu 1 API riêng biệt trong Tomcat 6, nó cho phép nắm băt từng sự hoạt động class loader trong Tomcat log. Nó sử dụng java.util.logg framework.

- Mặc đinh, Tomcat 7 sử dụng Java logging API để thực thi logging services. Có cũng có thể gọi là JULI. API này có thể được tìm thấy tại “TOMCAT\_HOME/bin”, tên “tomcat-juli.jar”.

- JULI cung cấp tính năng custom logging cho mỗi web app, và nó cũng hỗ trợ private logging config cho mỗi app. Với tính năng nâng cao của việc tách riêng class loader logging, nó giúp việc phát hiện vấn đề về memory khi unloading class trong môi trường runtime.

**Tìm hiểu thêm:**

Link: http://tomcat.apache.org/tomcat-8.5-doc/logging.html

### Loggers, appenders, và layouts

- Có 1 số thành phần quan trọng trong logging, ta sẽ sử dụng các thành phần thực hiện cơ chế logging cho app. Mỗi thuật ngữ sau đây rất quan trọng trong việc theo dõi các event của app.

### Các thành phần:

+ Loggers: Nó có thể định nghĩa tên logical cho log file. Logical name được viết cho app code. Ta có thể cấu hình logger riêng biệt cho mỗi app.

+ Appenders: Quá trình tạo các log được xử lý bởi appenders. Có rất nhiều loại appender như FileAppender, ConsoleAppender, SocketAppender .. Nó có sẵn trong log4j.

+ Layout: Nó định nghĩa format log được hiện thị trong logfie. Appender sử dụng layout format log files (gọi là patterns).

**Note:**

- 4 dòng dưới định nghĩa DailyRollingFileAppender trong log4j, các log này sẽ được đưa tới “catalina.out”

VD:

+ log4j.appender.CATALINA=org.apache.log4j.DailyRollingFileAppender

+ log4j.appender.CATALINA.File=${catalina.base}/logs/catalina.out

+ log4j.appender.CATALINA.Append=true

+ log4j.appender.CATALINA.Encoding=UTF-8

# Roll-over the log once per day

+ log4j.appender.CATALINA.DatePattern='.'dd-MM-yyyy'.log'

+ log4j.appender.CATALINA.layout = org.apache.log4j.PatternLayout

+ log4j.appender.CATALINA.layout.ConversionPattern = %d [%t] %-5p

%c- %m%n

VD:

<Valve className="org.apache.catalina.valves.AccessLogValve"

directory="logs" prefix="localhost\_access\_log." suffix=".txt"

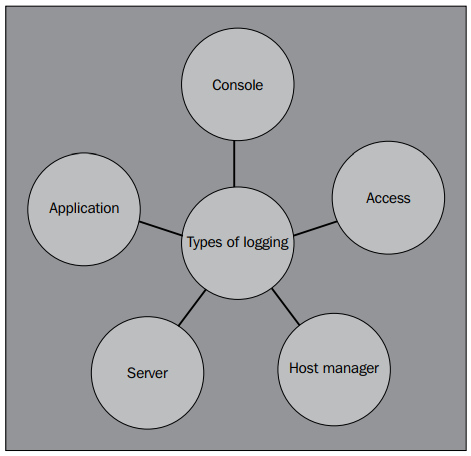
pattern="%h %l %u %t &quot;%r&quot; %s %b" resolveHosts="false"/>

## Các loại Logging trong Tomcat

- Có thể tạo Logging dự trên các yêu cầu khác nhau

- Có tổng cộng 5 loại log sẽ được sử dụng để theo dõi server, Web app.

- Các thành phần này sẽ được sử dụng kết hợp để theo dỗi Tomcat



### Application log

- Log này sử dụng để bắt các sự kiện app trong khi đang chạy app transaction. Log này rất hữu tích cho việc xác thực mức độ vấn đề app.

- VD: Giả sử app chạy chậm khi thực hiện 1 transition nào đó, thông tin về transition có thể traced trong app log.

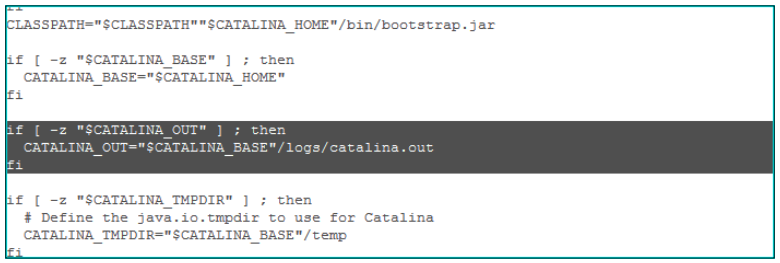
- Điểm mạnh lớn nhất của app log là ta có thể cấu hình riêng biệt log level và log file cho mỗi app. Khiến việc sửa chữa lỗi của admin trở nên dễ dàng hơn.

### Server log

- Server log giống với console log. Điểm mạnh duy nhất của server là có thể lấy ra bất kỳ thời điểm nào. (consolo log không có sẵn sau khi ta log out từ console)

### Console log

- Log này cung cấp các thông tin đầy đủ về quá trình startup và loader sequence tomcat 7. Log file đưa ra “catalina.out” có thể tìm thấy trong “TOMCAT\_HOME/logs”. Log fire này rất hữu dụng cho việc kiểm tra app deployment và server startup testing cho bất kỳ môi trường nào. Log có thể được cấu hình trong Tomcat file “catalina.sh” tại “TOMCAT\_HOME/bin”.



**Note:**

By default, the console logs are configured as INFO mode.

### Access log

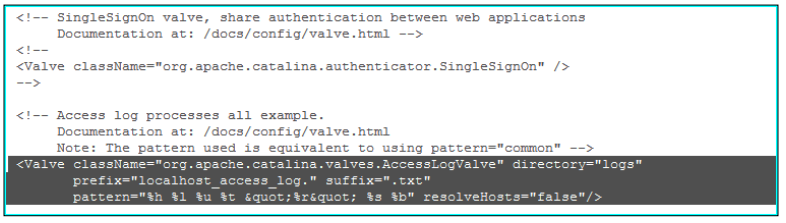
- Access log được customized logs, sử dụng để:

+ Đưa ra thông tin gười truy cập

+ Các thành phần app được truy cập

+ Ip nguồn ..

- Log này là thành phần quan trọng cho việc phân tích traffic cho app, phân tích băng thông cần thiết và fix vấn đề khi app tải nặng. Các cấu hình nằm trong “server.xml” tại “TOMCAT\_HOME/conf”



- Phân tích cú pháp:

<Valve className="org.apache.catalina.valves.AccessLogValve"

directory="logs" prefix="localhost\_access\_log." suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" resolveHosts="false"/>

Note:

+ Class Name: This parameter defines the class name used for the generation

of logs. By default, Apache Tomcat 7 uses the org.apache.catalina.

valves.AccessLogValve class for the access logs.

+ Directory: This parameter defines the directory location for the log file. All

the log files are generated in the log directory—TOMCAT\_HOME/logs—but

we can customize the log location based on our environment setup and then

update the directory path in the definition of the access logs.

+ Prefix: This parameter defines the prefix of the access log filename, that is, by

default, the access log files are generated by the name localhost\_access\_

log.yy-mm-dd.txt.

+ Suffix: This parameter defines the file extension of the log file. Currently it is

in .txt format.

+ Pattern: This parameter defines the format of the log file. The pattern is a

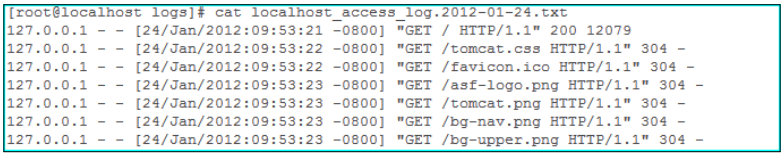
combination of values defined by the administrator, for example, %h = remote

host address. The following screenshot shows the default log format for

Tomcat 7. The access logs show the remote host address, date/time of the

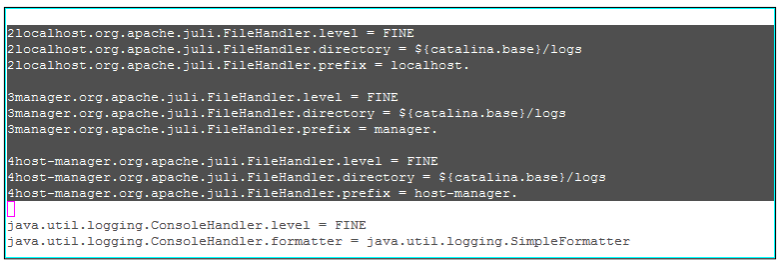
request, the method used for the response, URI mapping, and HTTP

status code.



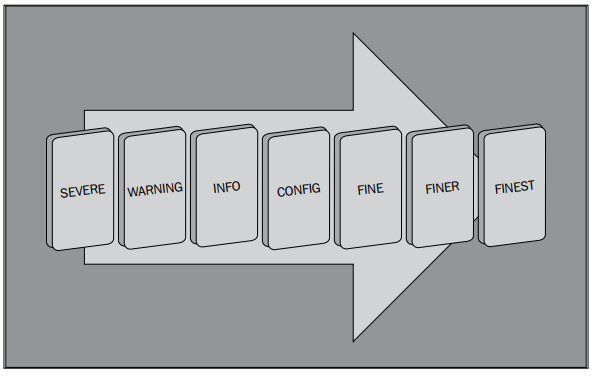
### Host manager

- Log định nghĩa các sự kiện sử dụng dụng Tomcat Manager, như thực hiện 1 số task, trạng thái app, deploy app, vòng đời tomcat. Các conf được đặt tại “logging.properties”, có thể tìm thấy tại “TOMCAT\_HOME/conf”.



### Các mức độ log trong Tomcat

- Có 7 loại lvl định nghĩa trạng thái Tomcat logging server (JULI). Nó được set dựa trên yêu cầu app.



- Mỗi log level trong JULI đều có tinh năng riêng

|  |  |
| --- | --- |
| Log level | Description |
| SEVERE(highest) | Bắt exception và Error |
| WARNING | Warning messages – Cảnh báo |
| INFO | Thông tin message, liên quan đến các hoạt động của server |
| CONFIG | Thông báo cấu hình |
| FINE | Thông tin về các hoạt động server transaction (như debug) |
| FINER | Thông tin chi tiết hơn FINE |
| FINEST(least) | Toàn bộ luồng event (Như trace) |

VD:

- Xem xét 1 appender trong logging.properties và log leved sử dụng cho appender đó. Logging đầu tiên cho localhost, sử dụng FINE như log level:

localhost.org.apache.juli.FileHandler.level = FINE

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

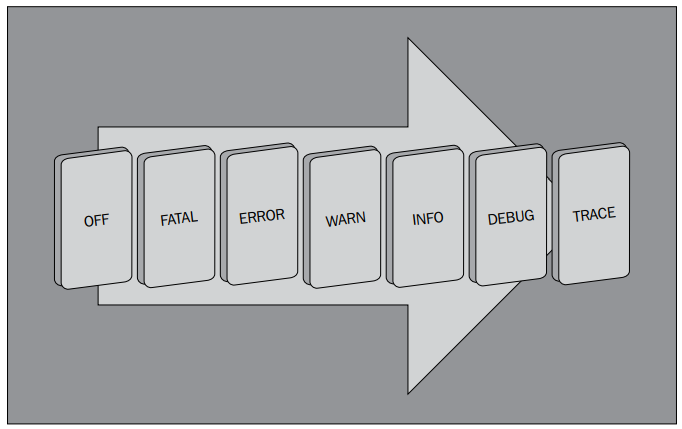
localhost.org.apache.juli.FileHandler.prefix = localhost.

## Log4j

### Giới thiệu

- Log4j là project cung cấp bởi The Apache Software Foundation. Project này giúp việc cho phép log level khác nhau tại server và app.

- Điểm mạnh của log4j là khả năng quản lý. Nó cung cấp cho dev khả năng thay đổi log level trong config file level. Mặc dù bạn có thể enable/disable log tại config level, nó không cần thiết việc thay đổi code. Ta có thể customize log pattern dựa trên các app riêng biệt. Log4j có 6 level.



### Log level trong log4j

- Mỗi log level trong log4j có tính năng riêng. Xem bảng chức năng trong mỗi log level trong log4j:

|  |  |
| --- | --- |
| Log level | Description |
| OFF | Mức độ này được set khi bạn muốn tắt chức năng logging (Stopped logging). |
| FATAL | Log level này sẽ in các lỗi nghiêm trọng mà nguyên nhân do kết thúc sớm |
| ERROR | Log level sử dụng để bắt các runtime errors hoặc các trường hợp không mong muốn. Mong muốn các log này sẽ hiện thị ngay tại status console. |
| WARN | Mức này sử dụng trong nhưng phiên bản trước.  Đưa ra các cảnh báo lỗi có thể xảy ra, không mong muốn trong mỗi trường runtime nhưng không nhất thiết xảy ra lỗi  Mong muốn các log này sẽ hiện thị ngay tại status console |
| INFO | Mức log level sẽ đinh nghĩa các event quan tâm trong môi trường runtime (khởi tạo/ tắt).  It is best practice to put the logs at the INFO level |
| DEBUG | Thông tin chi tiết về luồng của hệ thống |
| TRACE | Log level sẽ bắt tất cả event trong sys và app. |

**Cách sử dụng Log4j**

Xem thêm tại trang chủ: https://logging.apache.org/log4j/2.x/download.html

e

# Security Manager trong Tomcat

## Tổng quan

- Java SecurityManager là chính sách bảo mật của Tomcat, cho phép một trình duyệt chạy applet trong sanbox để ngăn chặn các mã không tin cậy truy cập vào tệp tin cục bộ. Giống như các SecurityManager bảo vệ bạn khởi các untrusted applet chạy trên brower, sử dụng SecurityManager trong khi chạy Tomcat có thể bảo vệ server khỏi trojan servlets, JSPs, JSP beans, tag libraries hoặc các lỗi ngoài ý muốn

VD:

- 1 người dùng thêm 1 đoạn JSPs trên website và vô tình sử dụng 1 đoạn code trong their JSP:

<% System.exit(1); %>

- Mỗi khi JSP này được thực thi bởi Tomcat, Tomcat sẽ thoát, Sử dụng Java SecurityManager chỉ cần thêm 1 dòng để phòng chống và giữ server bảo mật và đảm bảo.

## WARNING:

- Kiểm tra bảo mật được tiến hành bởi Tomcat dựa trên code. Hầu hết các package quan trọng đều được bảo vệ. Vì thế, đảm bảo bạn hài lòng với chính sách SecurityManager config trước khi cho phép user đẩy web app lên, JSPs, servlets, beans,. Cuối cùng, việc chạy SecurityManager được đĩnh nghĩa rõ ràng tốt hơn việc không sử dụng nó.

## Permissions

- Permissions class sử dụng để định nghĩa các permission cho các class loaded bởi Tomcat. Có 1 số Permission class tiêu chuẩn trong JDK, ta cũng có thể tạo mới Permission Class cho web app riêng.

## Standard Permissions

Tài liệu chuẩn: <http://docs.oracle.com/javase/7/docs/technotes/guides/security/>

Tóm tắt các quyền thường dùng:

+ java.util.PropertyPermission - Controls read/write access to JVM properties such as java.home.

+ java.lang.RuntimePermission - Controls use of some System/Runtime functions like exit() and exec(). Also control the package access/definition.

+ java.io.FilePermission - Controls read/write/execute access to files and directories.

+ java.net.SocketPermission - Controls use of network sockets.

+ java.net.NetPermission - Controls use of multicast network connections.

+ java.lang.reflect.ReflectPermission - Controls use of reflection to do class introspection.

+ java.security.SecurityPermission - Controls access to Security methods.

+ java.security.AllPermission - Allows access to all permissions, just as if you were running Tomcat without a SecurityManager.

## Cấu hình Tomcat với SecurityManager

- Chính sách polices được thực thi bởi Java SecurityManager được config trong “$CATALINA\_BASE/conf/catalina.policy”. This file hoàn toàn thay thế java.policy file trong JDK system directories. “catalina.policy” file có thể chỉnh sửa bằng tay hoặc bạn có thể sử dụng policytool app.

- Đối tượng trong catalina.policy sử dụng tiêu chuẩn chung “java.policy” file format.

// Example policy file entry

grant [signedBy <signer>,] [codeBase <code source>] {

permission <class> [<name> [, <action list>]];

};

Note:

- Đối tượng “signedBy” và “codeBase” là các optional khi cung cấp quyền.

- “codeBase” trong form URL, hoặc file URL có thể sử dụng tại ${java.home} và ${catalina.home} properties (được mở rộng ra các đường dẫn định nghĩa cho chúng bởi các biến môi trường JAVA\_HOME, CATALINA\_HOME và CATALINA\_BASE)

## Chính sách mặc định sử dụng cho Tomcat

- Mặc định “$CATALINA\_BASE/conf/catalina.policy”.

## Khởi động Tomcat với SecurityManager

- Chạy SecurityManager với option "-security".

$CATALINA\_HOME/bin/catalina.sh start -security (Unix)

%CATALINA\_HOME%\bin\catalina start -security (Windows)

## Quyền cho gói WAR files

- Khi sử dụng war đã đóng gói, cần sử dụng Tomcat custom WAR URL protocol để gán quyền vào code web app.

VD: Để gán quyền cho toàn bộ web app, chính sách sẽ như sau

// Example policy file entry

grant codeBase "war:file:${catalina.base}/webapps/examples.war\*/-" {

...

};

VD: Đế gán quyền cho 1 JAR duy nhất bên trong Web app, cấu hình sẽ như sau:

// Example policy file entry

grant codeBase "war:file:${catalina.base}/webapps/examples.war\*/WEB-INF/lib/foo.jar" {

...

};

## Cấu hình Package Protection trong Tomcat.

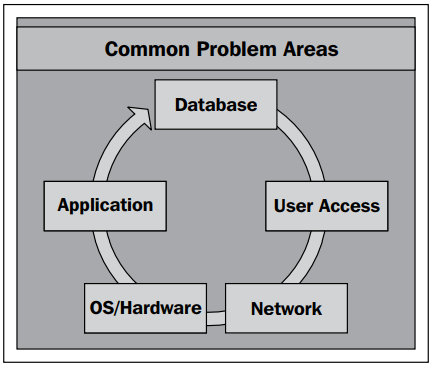
- Tomcat cho phép cấu hình bảo vệ các internal package khởi các tác động từ các package definition và access.

Link: <http://www.oracle.com/technetwork/java/seccodeguide-139067.html>

# Phân tích, giải quyết lỗi trong Tomcat

## Các vấn đề phổ biến:

- Lỗi phát sinh khi triển khai Web App có rất nhiều loại, để có thể phát hiện ra lỗi đòi hỏi phải hiểu rõ về cách Tomcat Web Server hoạt động



## 1 số vấn đề sẽ gặp phải trong real-time production support:

+ Application: Vấn để xảy ra khi app không chạy chinh xác, nguyên nhân do 1 số class loader xung đột, app deployment xung đột, thiếu tham số config, …

+ Database: Hiệu năng Database ảnh hưởng đến web App. Sẽ rất khó để tìm ra vấn đề liên quan tới DB. Một số do JNDI không tìm thấy, broken pipe error, ..

+ User Access: Vấn đề truy cập có thể xảy ra do database hoặc app thiếu config.

VD: User không thê truy cập vào app, không thể đang nhập, access denied..

+ Network: Là phần quan trọng trong hạ tầng IT. Nếu kết nối giữa các server gặp vấn đề, thì hiệu năng giữa các thành phần sẽ giảm, từ đó sinh ra lỗi services.

+ OS/Hardware: OS/hardware tạo ở tầng dưới so với app layer. Nếu gặp vấn đề tại OS/Hardware, nó sẽ ảnh hưởng đến services của Tomcat server.

## Cách giải quyết lỗi khi gặp phải:

### Vấn đề

- Không thể giải quyết lỗi dựa trên đánh giá của user hay trạng thái của lỗi. Để xác định chính xác vấn đề, ta phải thu hẹp phạm vi, phân tích các thành phần để tìm ra nguyên nhân dẫn đến lỗi, sau đó sửa.

### Làm sao để tìm ra và sửa lỗi:

- Lỗi sẽ tìm được khi ta phân tích đúng cách, đào sâu đúng vấn đề.

- Tìm ra do kinh nghiệm làm việc lâu dài với nền tảng, sử dụng kinh nghiệm để sửa.

## Phân tích khi gặp lỗi trên Web App – App chạy chậm

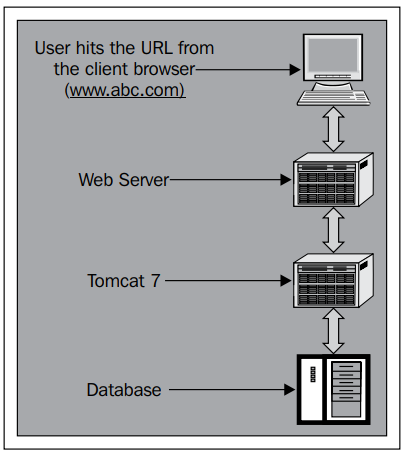
- Lỗi phổ biến khi host web trên server.

- App khi triển khai được tùy chỉnh theo yêu cầu, kết hợp với Apache HTTP server như frontend, Tomcat sử dụng servlet container và oracle database chạy phía dưới làm backend database server.

### Phân tích vấn đề:

- Vẫn đề thường xảy ra và nó sẽ mất thời gian để admin có thể giải quyết vấn đề. Ta có thể đặt tên cho nó là “slowness of application”. Vấn đề này thường xuất hiện khi có user report lại hiệu năng của app. Khi Sysad phát hiện được vấn đề, sẽ mất thời gian để tìm ra lỗi vì vấn đề có thể bắt nguồn từ bất cứ thành phần nào trong Web (OS, DB, web server, network, …) Và chỉ đến khi ta phát hiện, fix lỗi, hệ thống mới khôi phục lại sự ổn định.

Để bắt đầu, ta cần hiểu được luồng dữ liệu trong Web app



### Cách giải quyết vần đề:

- Để giải quyết vấn đề gây chậm cho App, cách tốt nhất là bắt đầu từ điểm nhìn User.

### Sửa lỗi tại góc nhìn User

- Thực hiện các bước sau:

+ Truy cập app thông qua User browser, kiểm tra thời gian load page

+ Ping tới server và xem kết quả lại lại. Nếu thấy ping response bình thường, ta có thể loại bỏ khả năng lỗi bắt nguồn từ kết nối App Server và User. Và ngược lại.

### Sửa lỗi tại Webserver

- Khi loại bỏ được lỗi không xảy ra tại kết nối user vs app, ta sẽ kiểm tra tại mức Web Server.

- Vấn đề với Webserver thường liên quan đến load server, sử dụng thread hoặc các vấn đề về liên kết các thành phần trong Web App.

**Các bước kiểm tra:**

**Bước 1:**

- Kiểm tra tiến trình Web Server (Còn chạy và trạng thái).

# ps –aef | grep httpd

- Nếu có quá nhiều process id hoặc tiến trình sử dụng quá nhiều CPU thì có thể, đã xảy ra vấn đề về lượng người dùng sử dụng đồng thời, vấn đề đọc ghi ổ đĩa, ..

**Bước 2:**

- Kiểm tra load CPU, trạng thái memory của hệ thống, kiểm tra lại tiến trình Apache sử dụng bao nhiêu CPU.

- Nếu thấy quá nhiều CPU và Memory sử dụng cho Apache, tức nguyên nhân có thể bắt nguồn từ Apache Proxy.

**Bước 3:**

- Kiểm tra Apache log, tìm kiểm các error trong error log và access log.

# tail -f log file | grep ERROR

# grep " 500 " access\_log

**Bước 4:**

- Kiểm tra hard drive, xem dung lượng ổ đĩa đang sử dụng, có bị quá tải, full, vấn đề đọc ghi ổ đia ..

Note:

Nếu qua các bước trên, ta vấn không tìm thấy sự bất thường, ta có thể loại bỏ khả năng lỗi phát sinh tại Web Server

### Sửa lỗi tại Java process

- App viết trên Java có rất nhiều vấn đề. Có thể do JVM memory, cách deploy, cấu hình db,…

**Các bước kiểm tra:**

Bước 1:

- Kiểm tra tiến trình Java, và load trung bình thực thi

- Load trung bình có thể đưa ra 1 số đầu mối về lỗi. Trong trường hợp load trung bình cao, sau đó ta kiểm tra load cho app, nếu vẫn cao thì có thể nguyên nhân xuất phát từ tiến trình Java process. Bên cạnh đó lỗi cũng có thể bắt nguồn từ Ram sử dụng và Swap Ram sử dụng.

# ps –ef | grep java

**Bước 2:**

- Kiểm tra log trong Tomcat “TOMCAT\_HOME/logs”, tìm kiểm các exception. Các exeception có thể tìm thấy trong các file: “catalina.out”, “localhost.yyyy-mm-dd.log”.

# grep INFO catalina.out

# grep ERROR catalina.out

**Note:**

Nếu không tìm thấy điều bất thường, ta có thể đến với phần tiếp theo, kiểm tra tại mực Database

**Phân tích thực thi JVM trong Tomcat**

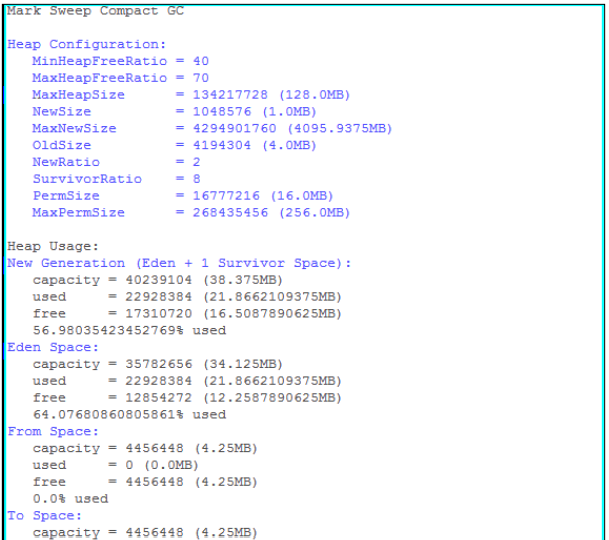
- Kiểm tra memory sử dụng cho JVM, ta có thể sử dụng “jmap” cmd.

**Câu lệnh:**

# jmap -heap "TOMCAT INSTANCE PID "

# jmap -heap 10638

- “jmap” cmd được đặt trong "JAVA\_HOME/bin" và nếu set "JAVA\_HOME/bin" tại script thực thi linux, có thể gọi câu lệnh trực tiếp

e

**Các thành phần trong JVM memory:**

+ Heap configuration

+ Heap usage

+ From space

+ To space

+ Tenured generation

+ Perm generation

+ Eden space

- Quá tải bộ nhớ như perm generation và max heap thường xảy ra trong môi trường thực thi. Kiểm tra memory, khi thấy bất kỳ thành phần nào đặt 95% thì nên cấu hình nâng cao bộ nhớ cho thành phần đó.

- Ta sẽ xác định thành phần JVM nào tạo ra lỗi khi thực thi Tomcat. Nếu memory hoạt động tốt, nó có thể đã sinh ra thread dump tại app. (xem tại phần sau)

### Sửa lỗi tại mức Database

- Đối với Sysad, ta không cần phải truy cập vào database web. Nhưng ta có thể kiểm tra kết nối connection string tới DB, service DB.

Note:

- Nên sử dụng các tool giám sát để phát hiện lỗi nhanh nhất

## Cách thu thập thread dump trong Tomcat

### Giới thiệu

- Thread dump là cách ta xác định trạng thái thread của tiến trình Java tại mức App. Có rất nhiều cách thu thập thread, ta sẻ sử dụng 2 cách thông thường.

### Lấy Thread dump bằng Kill command

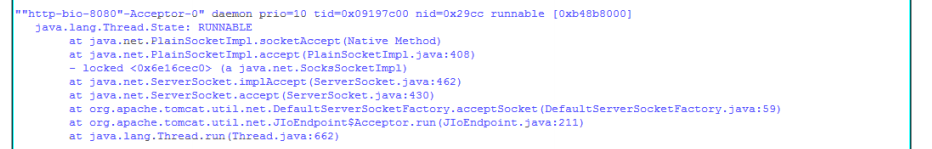
- Cmd sẽ chuyển hướng thread dump tới catalina.out log.

Kill -3 java process id

Kill -3 10638

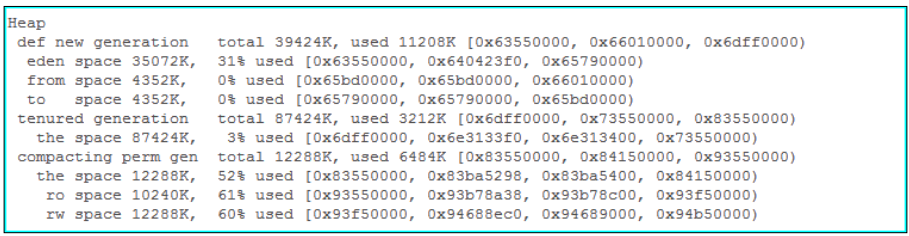
VD:

- Các thread java đang hoạt động



Note:

- Tại screenshot hiển thị đầu ra thread dump command tại catalina.out log. Ta thấy phần in đậm hiện thị “httpbio-8080-Acceptor” thread status với trạng thái “runnable” tức thread này vẫn chạy bình thường



Screenshot trên:

- Hiện thị trang thái sử dụng memory sử dụng tại thời điểm thread dump.

- Once the thread generation is complete, it then collects the memory dump for the Java processes. The previous screenshot shows the memory status at the time of the thread dump. This memory dump gives us the complete footprint of the memory used

### Sử dụng jstack

- Cách khác để sinh ra thread dump, nó sử dụng Java command-line utility gọi là jstack, thuộc gói JDK 1.5 hoặc các phiên bản mới nhất. jstack in Java stack thread cho java process. Công cụ này rất hữu tích trong môi trương thực thi, khi ta không thể trực tiếp đưa output thread tới server logs. 1 số option thương sử dụng trong jstack cmd:

|  |  |
| --- | --- |
| Options | Description |
| -f | Generates a Java stack forcefully. Majorly used when the process is in the hang state |
| -l | Long listing (displays the additional information on locks) |
| -m | Mixed mode Java stack generation |

- Sử dụng cmd:

# jstack –f Pid > threaddump.txt

# jstack -f 10638 > threaddump.txt

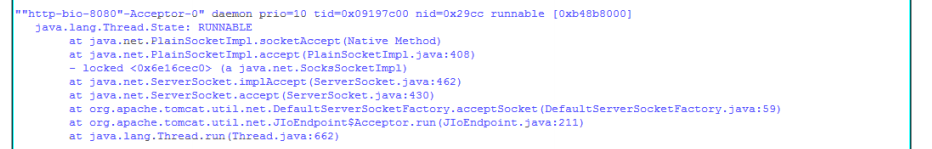
### Phân tích thread dump

- Phân tích thread-dump khá khó hiểu vì nó đưa các thông tin chi tiết nằm sâu trong app. Các phương pháp sau sử dụng để phân tích thread dump:

**Bước 1:**

- Obtain the thread dumps six times for the Java process ID with an interval of 10 seconds, using the command kill -3 or jstack.

- Các thread java đang hoạt động



**Bước 2:**

- Đối chiếu tất cả 6 thread dumps để tìm kiếm long running thread.

**Bước 3:**

- Tìm tất cả thread trong trạng thái stuck và tìm nguyên nhân dẫn đến stuck thread app và server-level threads.

Note:

- Nếu stuck thread tại app level, thì vấn đề liên quan đến app code.

- Nếu stuck thread tại server level, nó có thể do vấn đề server hoặc app level

**Errors and their solutions**- Có nhiều vấn đề sẽ xảy ra trong môi trường productionm và web admin cần phải đào sây vào logs. Sẽ luôn khó cho sysad hiểu được ý nghĩa các exception khi nó sinh ra trong app. Các tốt nhất là tìm hiểu exception đầu tiên, nó sẽ mô tả tổng quan nhất vấn đề.

- Lỗi có thể được chia thành 3 loại dựa trên các thành phần:

+ Application

+ JVM (memory)

+ Database

## Công cụ đánh giá Web server

- Công cụ đánh giá web server. Nó sẽ hỗ trợ việc test web server

- Nó sẽ giả lập việc tải nặng, và ước tính hiệu suất của web. Nó hữu ích khi ta muốn đo khả năng làm việc web server.

- Các công cụ:

+ ApacheBench (ab)

+ Jmeter

+ LoadRunner

+ OpenSTA

+ vv

### ApacheBench

ApacheBench is a command-line tool for web server benchmarking. It comes under the Apache HTTP server and is very useful when we want to generate only HTTP threads. It's a single thread process.

### Jmeter

JMeter is one of the widely used open source tools used for load testing. This tool is developed under the Apache Jakarta project. It is capable of generating traffic for JDBC, web services, HTTP, HTTPS, and JMS services. It's a desktop software, which does not support all features of browsers. Following are the advantages of JMeter:

• Portable (can be run on any platform)

• Supports multitasking that allows the administrator to test multiple processes

# Cơ chế Class Loader trong Tombat

## Tổng quan

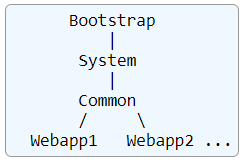
- Như các server app, Tomcat cài đặt chứa nhiều class loader (các class thực thi bởi java.lang.ClassLoader) để cho phép các phần khác nhau trên Tomcat webserver, và cũng như các web app chạy, các truy cập vào các kho chứa khác nhau có sẵn trong class và resource.

Note:

This mechanism is used to provide the functionality defined in the Servlet Specification, version 2.4 — in particular, Sections 9.4 and 9.6.

- Trong môi trường Java, class loader được bố trí theo hình cây. Cơ bản, khi class loader yêu cầu load 1 class đặc biệt hoặc 1 source, nó sẽ nạp các parent class loader của class hoặc source. Nếu không parent class loader, nó sẽ tìm kiếm trong các kho chứa.

- Khi tomcat khởi động, nó sẽ tạo tập các class loader được tổ chức, load theo quan hệ cha con:



## Định nghĩa Class Loader:

- Mô tả các giá trị trong bảng:

### Bootstrap:

- Class loader chứa các basic runtime class, cung cấp bởi Java Virtual Machine, thêm vào bất ký các JAR file tồn tại trong System Extensions directory ($JAVA\_HOME/jre/lib/ext)

Note:

some JVMs may implement this as more than one class loader, or it may not be visible (as a class loader) at all.

### System:

- Class loader này cơ bản khởi tạo các nội dung chứa tại biến môi trường “CLASSPATH”. Tuy nhiên, Script khởi động trong Tomcat ($CATALINA\_HOME/bin/catalina.sh) sẽ hoàn toàn bỏ qua các content trong biến môi trường “CLASSPATH”, thay thể nó bằng việc xây dựng System class loader từ kho chứa riêng.

+ “$CATALINA\_HOME/bin/bootstrap.jar” — Chứa main() methods, nó được sử dụng để khởi tạo Tomcat server, và các thực thi class loader phụ thuộc.

+ “$CATALINA\_BASE/bin/tomcat-juli.jar” hoặc “$CATALINA\_HOME/bin/tomcat-juli.jar” — Thực thi logging class. Bao gồm các class tăng cường cho java.util.logging API, được biết với tên Tomcat JULI, và package-renamed copy từ Apache Commons Logging library sử dụng trong nội bộ Tomcat.

++ Nếu tomcat-juli.jar tồn tại trong “$CATALINA\_BASE/bin”, nó sẽ sử dụng thay thế cho “$CATALINA\_HOME/bin”. Nó sẽ hữu ích cho 1 số logging config đặc biệt

+ “$CATALINA\_HOME/bin/commons-daemon.jar” — Classes thuộc Apache Commons Daemon project. JAR file này không tồn tại trong “CLASSPATH” built bởi catalina.bat.sh script, nhưng được tham chiều từ manifest file thuộc bootstrap.jar.

Note:

Xem thêm: http://tomcat.apache.org/tomcat-8.5-doc/logging.html

### Common:

- Class loader này chứa các class bổ sung được thêm hỗ trợ cho các class nội bộ Tomcat và cho tất cả các web app.

- Cơ bản, applocation class không được lưu tại đây, vị trí tìm kiếm các class loader sẽ được định nghĩa bởi common.loader property “$CATALINA\_BASE/conf/catalina.properties”.

- Cấu hình mặc đinh sẽ tìm kiếm các vị trí:

+ unpacked classes and resources in $CATALINA\_BASE/lib

+ JAR files in $CATALINA\_BASE/lib

+ unpacked classes and resources in $CATALINA\_HOME/lib

+ JAR files in $CATALINA\_HOME/lib

- Mặc định các class sẽ được cung cấp:

annotations-api.jar — JavaEE annotations classes.

catalina.jar — Implementation of the Catalina servlet container portion of Tomcat.

catalina-ant.jar — Tomcat Catalina Ant tasks.

catalina-ha.jar — High availability package.

catalina-storeconfig.jar — Generation of XML configuration files from current state

catalina-tribes.jar — Group communication package.

ecj-\*.jar — Eclipse JDT Java compiler.

el-api.jar — EL 3.0 API.

jasper.jar — Tomcat Jasper JSP Compiler and Runtime.

jasper-el.jar — Tomcat Jasper EL implementation.

jsp-api.jar — JSP 2.3 API.

servlet-api.jar — Servlet 3.1 API.

tomcat-api.jar — Several interfaces defined by Tomcat.

tomcat-coyote.jar — Tomcat connectors and utility classes.

tomcat-dbcp.jar — Database connection pool implementation based on package-renamed copy of Apache Commons Pool and Apache Commons DBCP.

tomcat-i18n-\*\*.jar — Optional JARs containing resource bundles for other languages. As default bundles are also included in each individual JAR, they can be safely removed if no internationalization of messages is needed.

tomcat-jdbc.jar — An alternative database connection pool implementation, known as Tomcat JDBC pool. See documentation for more details.

tomcat-util.jar — Common classes used by various components of Apache Tomcat.

tomcat-websocket.jar — WebSocket 1.1 implementation

websocket-api.jar — WebSocket 1.1 API

### WebappX

- Các class loader được tạo riêng cho từng Web app (Đã triển khai trên Tomcat). Tất cả unpacked class và resource được lưu trong “/WEB-INF/classes”. Các class, resource được thêm vào dạng JAR tại “/WEB-INF/lib”. Các class này sẽ chỉ được sử dụng bởi Web App.

### Theo góc nhìn của web app, class or resource loading theo:

- Bootstrap classes of your JVM

+ /WEB-INF/classes of your web application

+ /WEB-INF/lib/\*.jar of your web application

+ System class loader classes (described above)

+ Common class loader classes (described above)

- Nếu web app loader được cấu hình với <Loader delegate="true"/>, quá trình load sẽ là:

+ Bootstrap classes of your JVM

+ System class loader classes (described above)

+ Common class loader classes (described above)

+ /WEB-INF/classes of your web application

+ /WEB-INF/lib/\*.jar of your web application

# Proxy trong Tomcat

## Tích hợp Tomcat với Apache Web server

- Apache HTTP server là 1 trong nhưng máy chủ web frontend được sử dụng nhiều nhất. Project này được bắt đầu năm 1995 và được phát hành bởi The Apache Software Foundation.

## User request flow (web/application level)

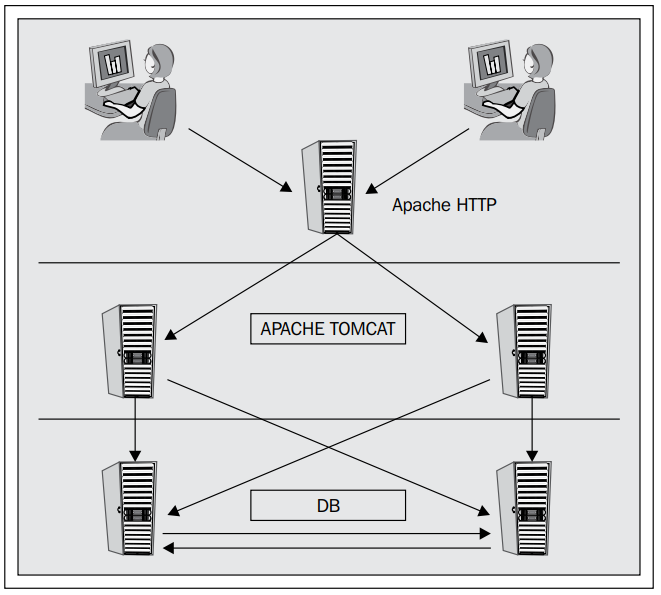
- Trước khi bàn luận về cài đặt Apache, ta sẽ nói về vấn đề tổng quát về các luồng request từ web và app server.

- Đây là luồng xử lý user request trong web app.

Bước 1: User nhập URL trên browser và request tới HTTP server thay vì tomcat.

Bước 2: HTTP service chập nhận request và chuyển hướng nó tới Tomcat cho xử lý business logic

Bước 3: Tomcat bên trong kết nối tới database server, get dữ liệu và gửi trả lời tới user thông qua kênh nhận request.



## Tại sao sử dụng Apache HTTP server

- Apache HTTP server là 1 trong nhưng web server thành công nhất trong nền công nghiệp IT. Nó thành công vì được phát triển bởi cộng đồng sớm.

- Nó được sử dụng rỗng rãi vì:

+ Efficiently serves static content (Phân phối nội dung 1 cách hiệu quả): Các nội dung tĩnh như hình ảnh, Js, css và Html file được phục vụ tốt bởi HTTP server trong môi trường rất nhiều người dùng. Tomcat cũng có khả năng như vậy nhưng thời gian đáp ứng sẽ tăng lên

+ Increase the speed by 10 percent: Như so sánh với tomcat, Apache phục vụ các static content với hiệu năng tốt hơn 10%. Tích hợp Apache rất hữu ích trong kịch bản có nhiều người cùng sử dụng, nâng cao khả năng chịu tải.

+ Clustering: Apache là 1 trong nhưng giải pháp tiết kiệm và ổn định nhất trong việc kết nối các thực thi Tomcat. Điểm mạnh nhất của tính năng này là app sẽ luôn online dù trong nhưng server thực thi gặp sự cố. Cũng như vậy, trong khi triển khai, ta có thể triển khai code trên 1 server thực thi mà trong các server thực thi khác vẫn online, phục vụ request tới từ user, nói bằng thuật ngữ là không có downtime trong app.

+ Security: Apache nâng cao an ninh cho user và host-based. Nó cũng có thể làm bằng Tomcat. Chúng ta phải quyết định sử dụng loại security nào giữa Tomcat or Apache dựa trên yêu cầu của app.

+ Multiple website hosting: 1 trong nhưng tính năng tốt nhất của Apache HTTP server là khả năng host nhiều web site. Tính năng này cũng được hỗ trợ tại Tomcat 7.

+ Modules: Apace rất linh động với nhiều module để tham chiếu. Ta có thể compile và decompile bất kỳ module nào dựa trên yêu cầu của app. Đây là tính này rất hữu ích trong khả năng mở rộng và tích hợp với công cụ thứ 3.

+ Decorator: Có thể định nghĩa như app URL (Redirect and Rewrites) quy tắc thiết kể cho việc phục vụ user request dựa trên app content. Apache web server hỗ trợ cả 2, Redirects và Rewrites rules với hiệu năng cao.

Note:

We can create 32 virtual hosts using httpd.conf. In case you want to configure more than 32 virtual hosts, then we have to create a separate virtual.conf file and include it in httpd.conf (httpd.conf and virtual.conf are the configuration files of the Apache HTTP server).

Note:

We can create Redirect and Rewrites in application code also. These rules are in the form of servlet classes.

## Cài đặt Apache HTTP

### Cài đặt apache HTTP trên Linux.

**Bước 1: Giải nén thư thục source apache**

# tar -zxvf httpd-2.2.34.tar.gz

**Bước 2: Cài đặt gói công cụ cho nhà phát triển**

# yum groupinstall "Development Tools"

**Bước 3: Chạy và cài đặt thư viện apr và apr-util**

- Cài đặt apr

# cd httpd-\*/srclib/apr

# ./configure --prefix=/usr/local/apr

# make clean (sử dụng khi make install báo lỗi)

# make

# make install

- Cài đặt apr-util

# cd httpd-\*/srclib/apr-util

# ./configure --with-apr=/usr/local/apr/

# make clean (sử dụng khi make install báo lỗi)

# make

# make install

**Bước 4: Cài đặt Apache**

# cd httpd-\*/

# ./configure --with-included-apr --prefix=/opt/apache

# make clean (sử dụng khi make install báo lỗi)

# make

# make install

**Bước 5: Kiếm tra việc cài đặt apache**

- Các thư mục xuất hiện sau khi cài đặt thành công

# ls /opt/apache

bin build cgi-bin conf error htdocs icons include lib logs man manual modules

- Kiểm tra config

# cd /opt/apache/bin

# ./apachectl configtest

Syntax OK

**Bước 6: Chạy service Apache**

# ./apachectl start

- Kiếm tra service:

ps -ef | grep httpd

## Apache Jserv protocol (AJP)

- Giao thức này chủ yếu sử dụng cho Proxy. Webserver sẽ nhận request sau đó chuyển request đó tới App server nằm sau nó và ngược lại. Request Data truyền qua network dạng binary thay vì dạng text. Nó sử dụng giao thức TCP và packet-based, tăng hiệu năng hoạt động web server. Ngoài ra, việc giải mã request được thực hiện trên web server nên app server sẽ không tốn tài nguyên cho vấn đề này.

- mod\_jk và mod\_proxy được thiết kế dựa trên AJP protocol. Nó giúp ích cho việc truyền high content respone qua web browser.

Note:

Nếu sử dụng AJP, network traffic sẽ tăng lên, vì nó chạy trên giao thức TCP.

Note:

If we use the latest version of mod\_jk for integration of Apache and Tomcat, then we can store the response header of 64k in the web browsers. This process is very useful in the case of SSO enabled applications or storing Java session values in the browser

## Cài đặt và cấu hình mod\_jk

- mod\_jk là AJP connector, nó được sử dụng để tích hợp vào web server như Apache hoặc IIS tới tomcat 7. Trong trường hợp ta ko install mod\_jk, ta không thể sử dụng frontend web server cho tomcat. Module này rất hứu ích cho việc giấu đi Tomcat bên dưới frontend webserver và cũng loại bỏ port khi truy cập thông qua URL. Nó sẽ bao gồm rất nhiều bước cho việc cài đặt và cấu hình mod\_jk.

### Cài đặt Installation of mod\_jk

- Truy cập vào trang chủ, down Tomcat Connectors JK 1.2

Link: http://tomcat.apache.org/download-connectors.cgi

**Bước 1:**

- Down source từ trên mạng:

# wget <http://mirrors.viethosting.com/apache/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.42-src.tar.gz>

- Giải nén gói:

# tar –zxvf tomcat-connectors-\*

**Bước 2:**

- Truy cập vào source, chạy việc cài đặt

# cd /opt/tomcat-connectors\*/native/

# ./configure –with-apxs=/opt/apache/bin/apxs

# make

# make install

Note:

mod\_jk is specific to the Apache version and the Apache Extension Tool (APXS) should be used for the current version of Apache, which we will use in the environment.

Once mod\_jk is compiled on one server, there is no need to generate for another Apache instance. It can be directly copied to the other instance. This trick is tested on Linux only.

**Bước 3:**

* Kiểm tra thư mục module tại /opt/apache

# ls /opt/apache/modules

httpd.exp mod\_jk.so

### Cấu hình mod\_jk trong Apache

- Cấu hình mod\_jk khá phức tạp trong Apache. Có nhiều cách để thực hiện việc cấu hình, nhưng cách thường sử dụng nhất là sử dụng option “workers.properties” và “mod\_jk.conf”.

**Bước 1:**

- Copy mod\_jk.so trong thư mục apache 2.0 (cùng thư mục tomcat-connector) sau khi cài đặt xong mod\_jk tới thư mục chứa apache.

# cp tomcat-connectors\*/native/apache-2.0/mod\_jk.so /opt/apache/modules/

**Bước 2:**

- Truy cập thư mục conf của Apache tại “$APACHE\_HOME/conf”

# cd /opt/apache/conf

# vim mod\_jk.conf

### content

LoadModule jk\_module modules/mod\_jk.so

JkWorkersFile conf/workers.properties

JkLogFile logs/mod\_jk.log

JkLogLevel info

JkMount /sample/\* node1

JkMount /\* node1

Note:

+ Module path: định nghĩa vị trí lưu trữ module để apache load khi thực thi. Tại đây ta load module mod\_jk.so

+ Worker file path: định nghĩa vị trí worker file, file này chứa thông tin chi tiết về thực thi Tomcat như IP, port, load balancing …

+ Log file: Sinh log trong quá trình thực thi, tích hợp Apache Tomcat. Nó cũng ghi lại trạng thái kết nối giưa apache và tomcat. (JkLogFile logs/mod\_jk.log).

+ URL mapping: Định nghĩa context path cho Apache cũng như set rule như chuyển hướng request nếu nhận được request tới URL đc định nghĩa.

VD: JkMount /sample/\* node1 => có nghĩa khi nhận bất kỳ request nào tới URL <http://localhost/sample> sẽ chuyển hướng tới Tomcat node 1.

+ Log level: Định nghĩa các mức độ các sự kiện khác nhau sẽ được đưa ra log.

**Bước 3:**

- Tạo file workers.properties trong thư mục config apache:

“$APACHE\_HOME/conf”

# vim workers.properties

worker.list=node1

worker.node1.port=8009

worker.node1.host=192.168.2.131 # ip server kết nối tới

worker.node1.type=ajp13

worker.node1.lbfactor=1

Note:

- Các trường giá trị workers.properties:

+ Node name: (Tên định nghĩa cho host)

+ AJP port: port định nghĩa bởi tomcat, port này sẽ chấp nhận giao thức AJP

+ Host IP: ip của node tomcat

+ Protocol sử dụng (trong ví dụ là AJP)

+ Load balancing methods, phương thức (Round robin, persistence, and so on)

**Bước 4:**

- Bước cuối, thêm giá trị cầu hình vừa tạo vào file cấu hình chính của apache. (thêm tại dòng cuối)

- Cấu hình apache tại file: ../apache/conf/http.conf

# vim httpd.conf

## thêm tại dòng cuối:

## Include mod\_jk

Include conf/mod\_jk.conf

**Bước 5:**

- Sau khi đã làm các bước trên, restart apache service

# cd ../apache/bin/

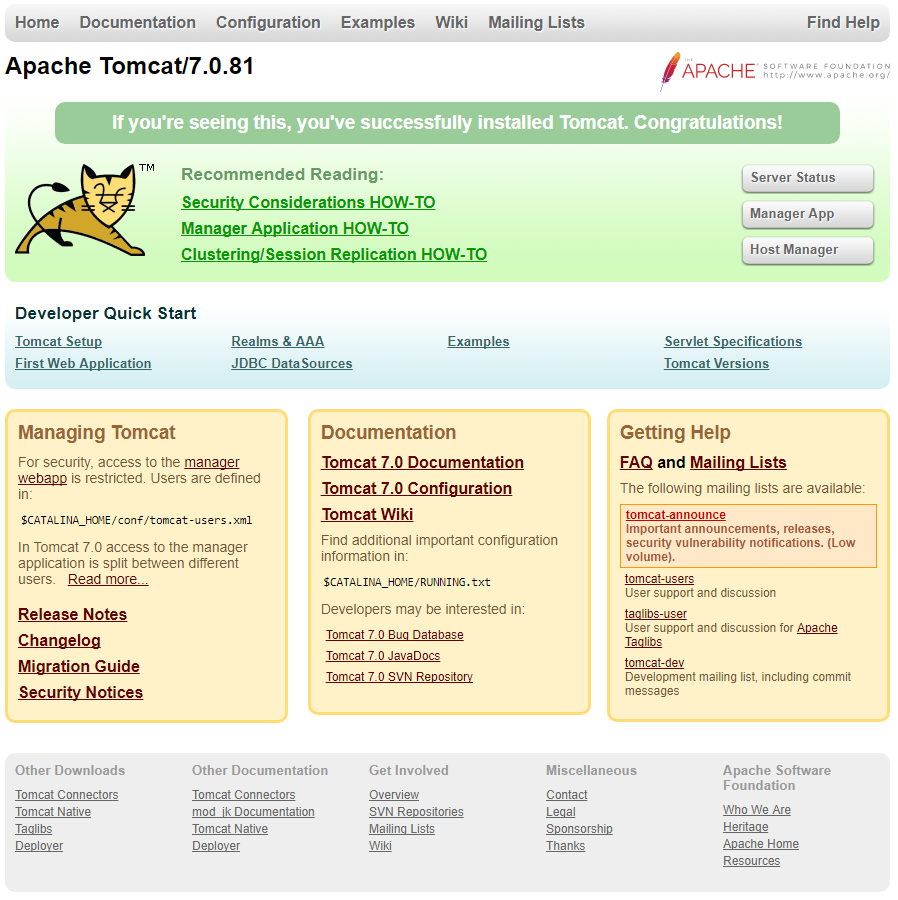
# ./apachectl stop

# ./apachectl start

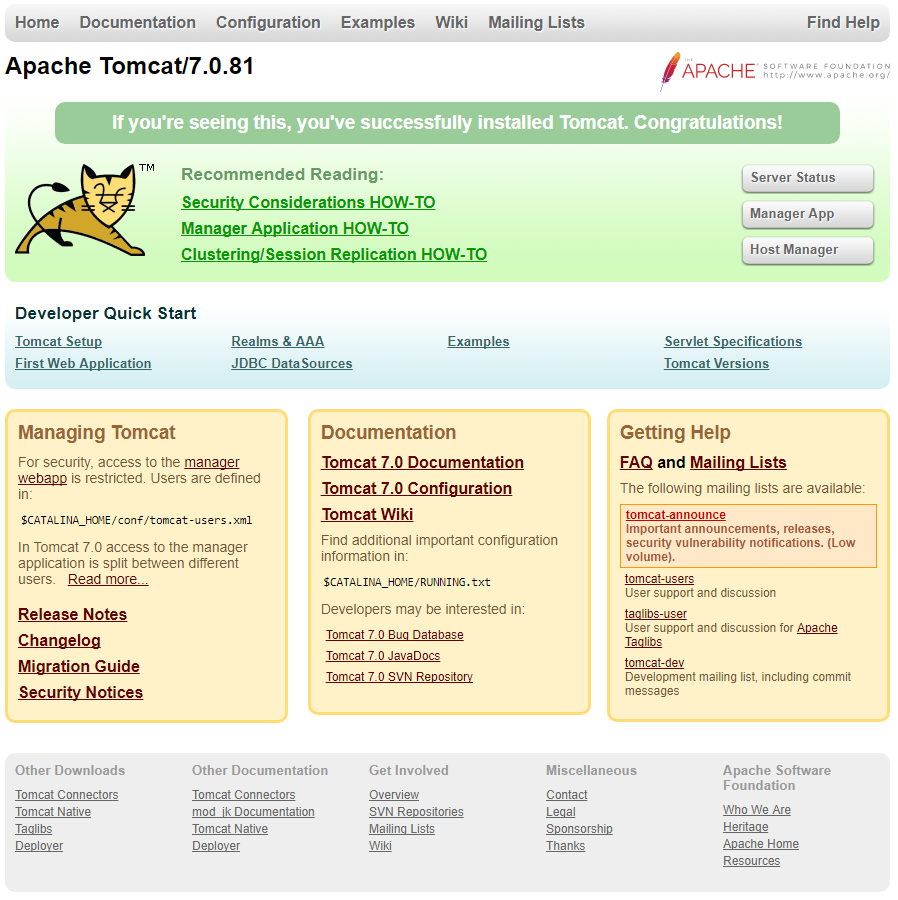
**Bước 6:**

- Kiểm tra lại kết quả:

- Truy cập <http://localhost:8080/>



* Truy cập trực tiếp qua service apache: truy cập <http://localhost/>



## mod\_proxy configuration

- Mod\_proxy config đơn giản hơn so với mod\_jk.

### Cài đặt module mod\_proxy

**Vấn đề:**

- Có thể module này không có sẵn, ta phải cài đặt từ source bên ngoài

**Bước 1:**

- Truy cập ../apache/bin/apxs –a –i–c [nguồn từ apache]/[tên module].c

# ../apache/bin/apxs –a –i–c [nguồn từ apache]/[tên module].c

Vd:

/usr/local/apache2/bin/apxs -a -i -c mod\_proxy.c proxy\_util.c

/usr/local/apache2/bin/apxs -a -i -c mod\_proxy\_ajp.c proxy\_util.c ajp\_utils.c ajp\_header.c ajp\_msg.c ajp\_link.c

/usr/local/apache2/bin/apxs -a -i -c mod\_proxy\_http.c proxy\_util.c

/usr/local/apache2/bin/apxs -a -i -c mod\_proxy\_balancer.c proxy\_util.c

/usr/local/apache2/bin/apxs -a -i -c mod\_proxy\_connect.c mod\_proxy\_ftp.c proxy\_util.c

/usr/local/apache2/bin/apxs -a -i -c mod\_proxy\_scgi.c proxy\_util.c

**Bước 2:**

* Kiểm tra lại thư mục chưa module trong apache

### Các bước cấu hình mod\_proxy

**Bước 1:**

- Mở httpd.conf, thêm cấu hình

# Thêm

LoadModule proxy\_module modules/mod\_proxy.so

LoadModule proxy\_http\_module modules/mod\_proxy\_http.so

# Thêm

NameVirtualHost \*

<VirtualHost \*>

ServerName abc.com

ProxyRequests Off

<Proxy \*>

Order deny,allow

Allow from all

</Proxy>

ProxyPass / http://localhost:8080/

ProxyPassReverse / http://localhost:8080/

<Location />

Order allow,deny

Allow from all

</Location>

</VirtualHost>

**Bước 2:**

- Restart service apache

# ./apachectl stop

# ./apachectl start

## So sánh giữa mod\_jk và mod\_proxy

* So sánh mod\_jk và mod\_proxy, từ đó lựa chọn cho môi trường thực tế

|  |  |  |
| --- | --- | --- |
| Feature | mod\_jk | mod\_proxy |
| Load balancing | High level | Basic |
| Management interface | Yes | No |
| Compilation | Separate process | Not required. By default comes with Apache |
| Configuration | Huge | Basic |
| Protocol | AJP | HTTP/HTTPS/AJP |
| Node failure | Advance | NA |

e

# Clustering và Session Replication trong Tomcat

## Tổng quan

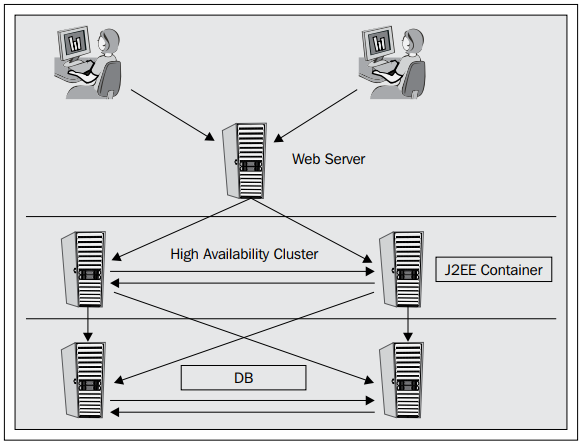
Cluster là gì?

- Cluster là 1 nhóm các server và máy tính được kết nối với nhau, nó có thể thực hiện các tính năng giống nhau. Hệ thống được kết nối thông qua high speed Ethernet.

- Cluster được sử dụng khi cần xử lý nhanh và tính HA hệ thống.

VD: Trong ngành tài chính, ngân hàng, bảo mật…

- Kiển trúc cluster (mô hình J2EE)



## Lợi ích sử dụng Cluster:

- Có rất nhiều điểm mạnh của cluster trong môi trường middleware. Nó dựa trên kỹ thuật mà ta sử dụng.

- Các điểm mạnh chính:

+ Tính mở rộng (Scalability): Cho phép system architect (kiến trúc sư hệ thống) cải tiến, nâng cao ứng dụng trong tương lại. VD: khi web mở rộng, lượng truy cập từ 100 -> 1000 user, cluster sẽ là giải pháp cho vấn đề đó.

+ Tính sẵn sàng (HA): Tính sẵn sàng hệ thống trong môi trường thực thi phải luôn đạt 99%, như ngân hàng, tài chính .. nơi mà các hoạt đông giao dịch diễn ra, và không được phép xuất hiện down time. => HA giải quyết vấn đề đó.

+ Nâng cao hiệu suất: 1 tính năng quan trọng của cluster là tăng tốc hệ thống tới n lần, trong đó n = số hệ thống sử dụng. VD, 1 server = 100 user kết nối => 5 server = 500 user.

+ Cloud computing: Cluster rất hữu ích trong môi trường tính toán đám mây. Nó sử dụng kỹ thuật tính toán lưới để nâng cao hiệu năng.

## Bất lợi khi sử dụng clustering:

+ Giá thành: là 1 vấn đề lớn khi sử dụng cluster, nếu sử dụng nó => cần đến nhiều hơn 1 server.

+ Giám sát: với việc tăng số lượng server, việc giám sát tăng lên => khó cho web admin quan trị.

## Kiến trúc Cluster

- Có rất nhiều kiến trúc cluster được sử dụng, và có nhiều cách để triển khai phụ thuộc theo vấn đề, yếu cầu.

- 2 kiên trúc cluster đơn giản thường sử dụng:

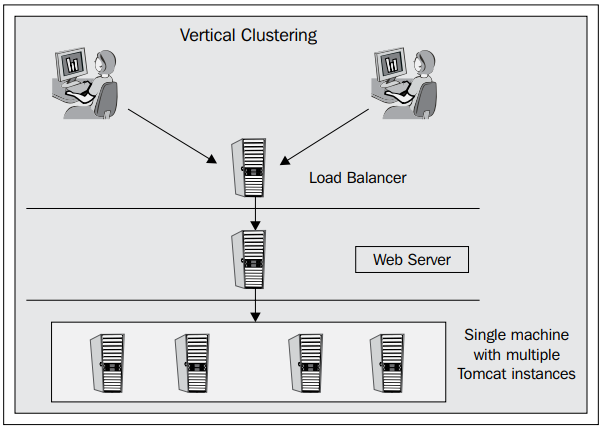
+ Vertical clustering (Thẳng đứng)

+ Horizontal clustering (nằm ngang)

- Mặc định Tomcat hỗ trợ cả 2 kiến trúc

### Vertical clustering

- Bao gồm 1 phần cứng với nhiều thực thi chạy cho phép shared resource trong hệ thống. Loại thiết lập này chủ yếu sử dụng trong môi trường thiết kế và kiêm tra chất lượng, test các tính năng app. Nó cũng có thể được thực hiện trong môi trường chính trong 1 số trường hợp, khi tài nguyên phần cứng có hạn. Sử dụng như share tài nguyên CPU, RAM, ..



**Điểm mạnh vertical clustering**

+ Không có vấn đề về băng thống mạng, chỉ có 1 thực thể web.

+ Phần cứng được chia sẻ bởi các thực thi tomcat khác nhau

+ Việc bổ sung phần cứng ko cần thiết

+ 1 thực thi JVM chia sẽ bởi nhiều thực thi con.

**Điểm yếu vertical clustering**

+ không có khả năng chịu lỗi phần cứng

+ Nhiều vấn đề phải giải quyết

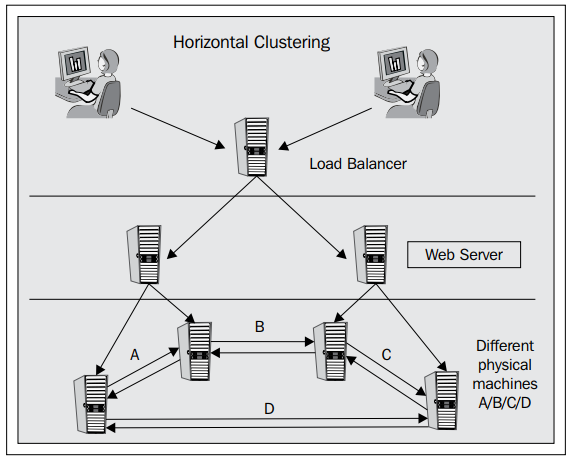
+ Phần cứng đủ cao để thực hiện

+ Giá thành cao.

### Horizontal clustering

**Giới thiệu**

- Tại đây, các thực thi được cấu hình riêng biệt trên mỗi máy vật lý, kết nối thông qua mạng tốc độ cao. Mô hình thường được sử dụng trong thực tế. Tài nguyên trên mỗi máy sẽ không được share. Có khả năng chịu lỗi khi phần cứng xảy ra vấn đề.



**Điểm mạnh horizontal clustering**

- Khả năng chịu lỗi cao

- Giá thành sẽ hạ xuống

- Vấn đề phát sinh ít hơn.

**Điểm yếu horizontal clustering**

- Vấn đề băng thông

- Vấn đề về các kết nối giữa các máy

- Mỗi thực thi yêu cầu 1 thành phần (node)

Note:

Horizontal clustering is the most preferred method in a production environment.

## Sử dụng cluster đơn giản:

- Thêm “<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>” vào <Engine> hoặc <Host> để cho phép chạy cluster.

- Sử dụng config sẽ cho phép nhân rộng session all to all bằng sử dụng DeltaManager thực hiện quá trình nhân bản session deltas. All to all nghĩa là session được nhân rộng tới tất cả các node khác trong cluster. Cấu hình này sẽ hoạt động tốt với các cluster nhỏ nhưng sẽ không khuyến khích sử dụng nó cho cluster lớn ( với rất nhiều nodes). Bên cạnh đó, khi sử dụng delta manager, nó sẽ nhận rộng tới tất cả node, thậm chí node không có app deployed.

- Để giải quyết vấn đề này, ta sẽ sử dụng BackupManager. Manager này sẽ chỉ sao chép session data tới 1 backup node và chỉ cho các node có app deployed.

1 số giá trị quan trọng mặc định:

+ Multicast address is 228.0.0.4

+ Multicast port is 45564 (the port and the address together determine cluster membership.

+ The IP broadcasted is java.net.InetAddress.getLocalHost().getHostAddress() (make sure you don't broadcast 127.0.0.1, this is a common error)

+ The TCP port listening for replication messages is the first available server socket in range 4000-4100

+ Listener is configured ClusterSessionListener

+ Two interceptors are configured TcpFailureDetector and MessageDispatchInterceptor

VD: Có cấu hình mặc định đầy đủ như sau:

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster" channelSendOptions="8">

<Manager className="org.apache.catalina.ha.session.DeltaManager" expireSessionsOnShutdown="false" notifyListenersOnReplication="true" />

<Channel className="org.apache.catalina.tribes.group.GroupChannel">

<Membership className="org.apache.catalina.tribes.membership.McastService" address="228.0.0.4" port="45564" frequency="500" dropTime="3000" />

<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver" address="auto" port="4000" autoBind="100" selectorTimeout="5000" maxThreads="6" />

<Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">

<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender" />

</Sender>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector" />

<Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatchInterceptor" />

</Channel>

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve" filter="" />

<Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve" />

<Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer" tempDir="/tmp/war-temp/" deployDir="/tmp/war-deploy/" watchDir="/tmp/war-listen/" watchEnabled="false" />

<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener" />

</Cluster>

## Security

- Thực hiện Cluster trên cơ sở bảo mật cơ bản, trusted network được sử dụng cho cluster liên quan trong network traffic.

- Có rất nhiều các để cung cấp sự bảm đảm, trusted network cho tomcat, bao gồm:

+ private LAN

+ a Virtual Private Network (VPN)

+ IPSEC

## Cài đặt Vertical clustering in Apache Tomcat

### Yêu cầu

- Để thực hiện Vertical clustering cần ít nhất 2 thực thi Tomcat (trên cùng 1 máy)

Các bước sẽ thực hiện:

Phần 1: Cài đặt thực thi Tomcat

Phần 2: Cấu hình Cluster

Phần 3: Cấu hình Apache HTTP web server as frontend cho veritical clustering.

### Cài đặt

**Phần 1:**

- Cài đặt thực thi Tomcat.

+ Thực hiện theo docs cài đặt

+ Copy 2 thư mục Tomcat tới “/opt/tomcat-node-X” (X là số thực thi)

**Phần 2:**

- Cấu hình Vertical cluster

**Cấu hình trên thực thi 1 (tomcat-node-1)**

**Yêu cầu:**

- Cấu hình node 1 các tham số trong Connector, AJP, shutdown port trong “server.xml”.

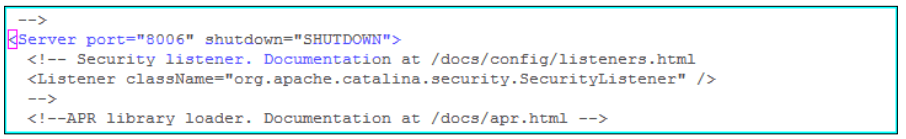
**Cấu hình:**

**Bước 1:**

- Cấu hình file “server.xml”

**Shutdown port:**

- Nếu bỏ qua cấu hình thuộc tính nào, thực thi tomcat sẽ không thể chạy

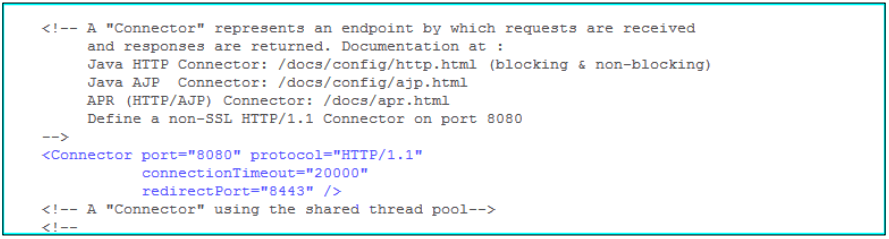


**Connector port:**

- Port này sử dụng để truy cập vào thực thi tomcat.

VD: sử dụng localhost:8080, 8080 port được gọi là Connector port.

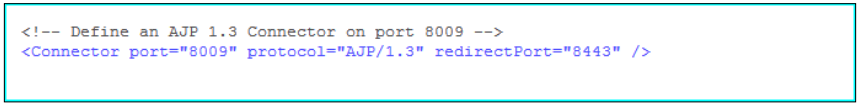
- Nếu chạy nhiều thực thi Tomcat, ta bỏ qua cấu hình port này, khi chạy tomcat sẽ xuất hiện lỗi Port đã sử dụng, hoặc đang sử dụng.



**AJP port:**

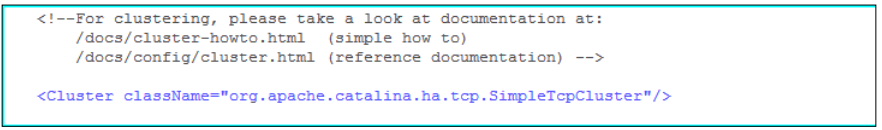
- Port này sử dụng để kết nối AJP giữa Apache HTTP server với thực thi Tomcat.

- Khi cấu hình nhiều thực thi, bỏ qua cấu thuộc tính này, nó sẽ sinh ra exception “port đã được sử dụng”.



**Cluster attributes:**

- Kích hoạt tính năng cluster trong Tomcat



**Bước 2: Chạy thực thi Tomcat 1**

- Test lại cấu hình Tomcat (Tránh lỗi cú pháp, đặt tham số sai sau khi cấu hình)

# ./configtest.sh



- Chạy thực thi 1 sử dụng script startup.sh

# ./startup.sh



- Kiểm tra thực thi

# ps -ef | grep java



**Cấu hình trên thực thi 2 (tomcat-node-2)**

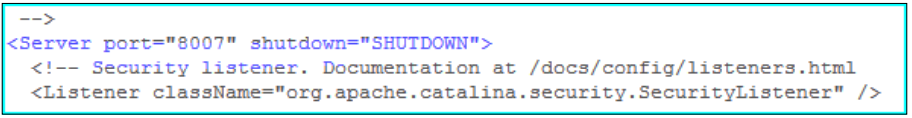
Lưu ý:

- Ta không thể sử dụng cấu hình mặc định tại node 2. Các tham số sẽ bị trùng với thực thi tomcat 1.

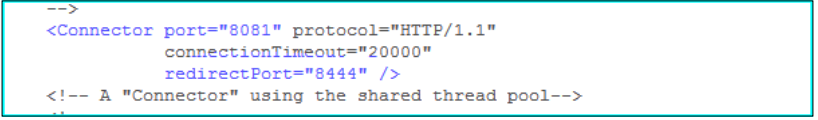
Cấu hình:

**Bước 1: Cấu hình file “server.xml”**

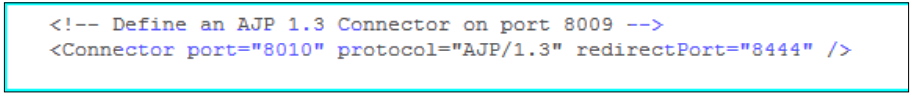
- Thay đổi “Shutdown port” mặc định



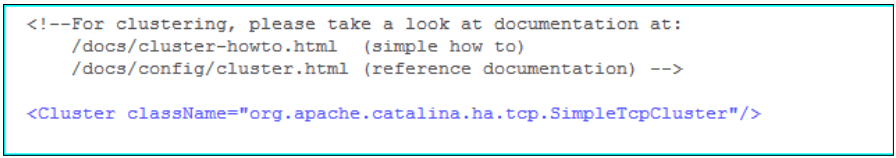
- Thay đổi “Connector” và “redirect port” mặc định



- Thay đổi “AJP” và “redirect port” mặc định



- Kích hoạt tính năng cluster chạy trên tomcat node 2.



**Bước 2:**

- Kiểm tra lại các cấu hình

# ./configtest.sh

- Chạy thực thi Tomcat node 2

# ./startup.sh

- Kiểm tra thực thi tomcat node 2

# ps -ef | grep java



**Kiểm tra log catalina.out:**

Log tại Tomcat node 1:

……

Oct 11, 2011 5:00:24 PM

org.apache.catalina.tribes.membership.McastServiceImpl setupSocket

**# Instance node 1 started on port 4000**

INFO: Setting cluster mcast soTimeout to 500

Oct 11, 2011 5:00:24 PM

INFO: Sleeping for 1000 milliseconds to establish cluster

membership,

start level:8

Oct 11, 2011 5:00:26 PM

org.apache.catalina.tribes.membership.McastServiceImpl

waitForMembers

**# waiting for other member to join the cluster**

org.apache.catalina.ha.session.JvmRouteBinderValve startInternal

……..

SimpleTcpCluster

memberAdded

INFO: Replication member

added:org.apache.catalina.tribes.membership.MemberImpl

[tcp://{127, 0, 0, 1}:4001,{127, 0, 0, 1},4001, alive=1043,

securePort=-1, UDP Port=-1, id={33 91 -59 78 -34 -52 73 -9 -99 124

-53 34 69 21 -40 -82 }, payload={}, command={}, domain={}, ]

**#Instance 2 joined the cluster node.**

Log tại Tomcat node 2:

INFO: Starting Servlet Engine: Apache Tomcat/7.0.12

Oct 11, 2011 5:23:41 PM org.apache.catalina.ha.tcp.

SimpleTcpCluster

startInternal

INFO: Cluster is about to start

Oct 11, 2011 5:23:42 PM org.apache.catalina.tribes.transport.

ReceiverBase bind

INFO: Receiver Server Socket bound to:/127.0.0.1:4001

Oct 11, 2011 5:23:42 PM

org.apache.catalina.tribes.membership.McastServiceImpl setupSocket

**# Instance node 2 started on port 4001**

INFO: Setting cluster mcast soTimeout to 500

Oct 11, 2011 5:23:42 PM

org.apache.catalina.tribes.membership.McastServiceImpl

waitForMembers

INFO: Sleeping for 1000 milliseconds to establish cluster

membership,

start level:4

Oct 11, 2011 5:23:43 PM org.apache.catalina.ha.tcp.

SimpleTcpCluster

memberAdded

INFO: Replication member

added:org.apache.catalina.tribes.membership.MemberImpl

[tcp://{127, 0, 0, 1}:4000,{127, 0, 0, 1},4000, alive=1398024,

securePort=-1, UDP Port=-1, id={28 42 60 -68 -99 126 64 -35 -118

-97 7 84 26 20 90 24 }, payload={}, command={}, domain={}, ]

**# Instance 1 joined the cluster node 2.**

**Phần 3:**

**Cấu hình Apache web server cho cluster**

- Sau khi thực hiện cấu hình Tomcat cho vertical cluster, ta sẽ tích hợp Apache web server cho Tomcat 7.

**Cấu hình:**

**Bước 1: Tạo 1 file mới “mod\_jk.conf” trong “conf” direc trong “APACHE\_HOME/conf”**

# cd /opt/apache/conf

# vi mod\_jk.conf

## nội dung

LoadModulejk\_module modules/mod\_jk.so

JkWorkersFile conf/workers.properties

JkLogFile logs/mod\_jk.log

JkLogLevel info

JkMount /sample/\* loadbalancer

JkMount /\* loadbalancer

**Bước 2: Tạo file mới tên “workers.properties” trong “conf”**

# vi workers.properties

**worker.list=tomcatnode1, tomcatnode2, loadbalancer**

**worker.tomcatnode1.port=8009**

**worker.tomcatnode1.host=localhost**

worker.tomcatnode1.type=ajp13

worker.tomcatnode1.lbfactor=1

**worker.tomcatnode2.port=8010**

**worker.tomcatnode2.host=localhost**

worker.tomcatnode2.type=ajp13

worker.tomcatnode2.lbfactor=1

**worker.loadbalancer.type=lb**

**worker.loadbalancer.balanced\_workers=tomcatnode1, tomcatnode2**

worker.loadbalancer.sticky\_session=1

Note:

- “worker.lists” tất các nodes trong Tomcat mà Apache sẽ kết nối tới giao thực AJP protocal.

- Định nghĩa các node bằng “worker.tomcatnodeX”.

- Định loadbalancer bằng “worker.loadbalancer”.

## Horizontal clustering in Apache Tomcat

- Để có thể triển khai horizontal cluster ta cần ít nhất Apache tomcat trên 2 máy vật lý hoặc máy ảo. Các máy physical có thể trên cùng 1 mạng, nó sẽ nâng cao băng thông của hệ thống.

Note:

If you want to configure clustering on different networks, then you have to open the firewall between the two networks for the AJP port and the clustering port.

### Yều cầu

- Trước khi cấu hình horizontal clustering cần:

+ Thời gian đồng bộ giữa 2 server

+ Kết nối mạng thích hợp giữa 2 server

+ Firewall port giữa 2 server (Nếu khác mạng)

### Cài đặt

- Để cấu hình horizontal cluster, ta cần thực hiện:

Phần 1: Cài đặt thực thi Tomcat

Phần 2: Cấu hình cluster

Phần 3: Apache HTTP web configuration cho horizontal cluster.

**Phần 1: Cài đặt thực thi Tomcat**

+ Làm theo hướng dẫn docs

**Phần 2: Cấu hình Cluster**

- Sau đây là các bước quan trọng để cấu hình horizontal cluster, tất cả cấu hình trong phần này cần chính xác, 1 lỗi đơn giản có thể khiến cluster không hoạt động.

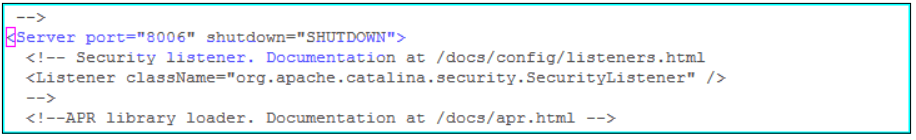
**Cấu hình thực thi 1:**

**Bước 1:**

- Cho thực thi đầu tiên, ta có thể sử dụng cấu hình mặc định như Connector, AJP, shutdown port trong “server.xml”.

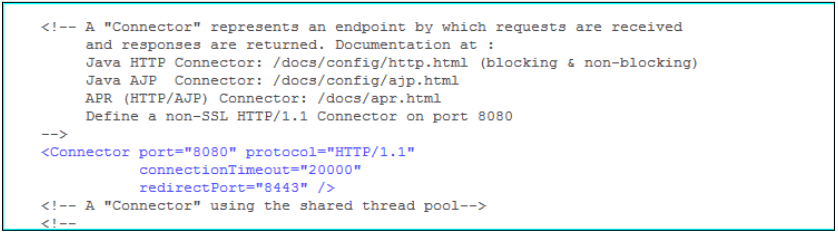
**Thay đổi cấu hình “Shutdown port”:**

- Trong khi chạy nhiều thực thi, nếu ta bỏ qua việc cấu hình port, thực thi Tomcat sẽ không thể khởi tạo.



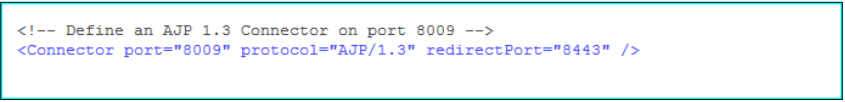
**Thay đổi cấu hình “Connector port”:**

- Port này sử dụng để truy cập vào thực thi tomcat, cơ bản ta truy cập vào thực thi Tomcat bằng cách sử dụng “http://localhost:8080”. Port 8080 gọi là connector port. Trong khi chạy nhiều thực thi, nếu bạn bỏ qua việc cấu hình port này, tomcat sẽ không thể chạy với thông bảo “Port đã sử dụng”.



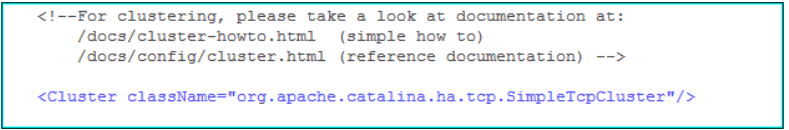
**AJP port:**

- Port này sử dụng cho kết nối AJP giữa Apache HTTP với thực thi Tomcat. Khi chạy nhiều thực thi, nếu bỏ qua cấu hình port, tomcat sẽ ko thể chạy với thông báo “port đã sử dụng”.



**Thuộc tính cluster:**

- Kích hoạt tính năng cluster trên thực thi



- Trong horizontal cluster, mỗi máy có 1 ip riêng, ta cần cấu hình broadcast addtess và port cho việc kết nối các note với cluster session. Thêm code tới server.xml, cho phép broadcast setting và replication (Nhân rộng):

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"

channelSendOptions="6">

<Manager className="org.apache.catalina.ha.session.

BackupManager" expireSessionsOnShutdown="false"

notifyListenersOnReplication="true" mapSendOptions="6"/>

<Channel className="org.apache.catalina.tribes.group.

GroupChannel">

<Membership className=

"org.apache.catalina.tribes.membership.McastService"

address="228.0.0.4" port="54446" frequency="500"

dropTime="3500"/>

<Receiver className=

"org.apache.catalina.tribes.transport.nio.NioReceiver"

address="auto" port="6000" selectorTimeout="100"

maxThreads="6"/>

**<Sender className=**

**"org.apache.catalina.tribes.transport.**

**ReplicationTransmitter">**

**<Transport className=**

**"org.apache.catalina.tribes.transport.nio.**

**PooledParallelSender"/>**

**</Sender>**

</Channel>

**<Deployer className="org.apache.catalina.ha.deploy.**

**FarmWarDeployer" tempDir="/opt/apachetomcat1/tomcat7-temp/"**

**deployDir="/opt/apachetomcat1/tomcat7-deploy/"**

**watchDir="/opt/apachetomcat1/tomcat7-listen/"**

**watchEnabled="false"/>**

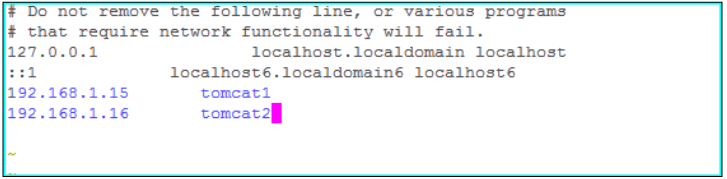
<ClusterListener className=

"org.apache.catalina.ha.session.ClusterSessionListener"/>

</Cluster>

**Bước 2:**

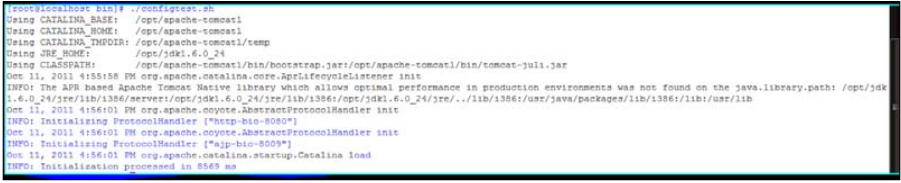
- Thêm đối tượng Host trong host file (/etc/hosts).



**Bước 3:**

- Chạy “configtest.sh” trong “TOMCAT\_HOME/bin”.

# ./configtest.sh



- Khởi tạo thực thi tomcat. Chạy thực thi 1 tomcat.

# ./startup.sh



- Kiểm tra thực thi tomcat

# ps -ef | grep java



**Cấu hình thực thi 2:**

- Để chạy cluster tại thực thi 2, cài đặt tomcat trên node và thực hiện các bước giống node 1.

**Kiểm tra**

- Check catalina.out trên cả 2 node. Xem giá trị kết nối hiển thị

- Tại node 1:

Oct 11, 2011 5:00:24 PM org.apache.catalina.ha.tcp.SimpleTcpCluster

startInternal

INFO: Cluster is about to start

Oct 11, 2011 5:00:24 PM org.apache.catalina.tribes.transport.

ReceiverBase

bind

**INFO: Receiver Server Socket bound to:/192.168.1.15:4000**

**Oct 11, 2011 5:00:24 PM**

**org.apache.catalina.tribes.membership.McastServiceImpl setupSocket**

# Instance node 1 started on port 4000

to establish cluster membership, start level:4

Oct 11, 2011 5:00:25 PM

org.apache.catalina.tribes.membership.McastServiceImpl waitForMembers

INFO: Done sleeping, membership established, start level:4

Oct 11, 2011 5:00:25 PM

org.apache.catalina.tribes.membership.McastServiceImpl waitForMembers

INFO: Sleeping for 1000 milliseconds to establish cluster membership,

start

level:8

Oct 11, 2011 5:00:26 PM

org.apache.catalina.tribes.membership.McastServiceImpl waitForMembers

# waiting for other member to join the cluster

INFO: Server startup in 13807 ms

Oct 11, 2011 5:23:42 PM org.apache.catalina.tribes.io.BufferPool

getBufferPool

INFO: Created a buffer pool with max size:104857600 bytes of

type:org.apache.catalina.tribes.io.BufferPool15Impl

Oct 11, 2011 5:23:43 PM org.apache.catalina.ha.tcp.SimpleTcpCluster

memberAdded

**INFO: Replication member**

**added:org.apache.catalina.tribes.membership.MemberImpl**

**[tcp://{192.168.1.16, 0, 0, 1}:4001,{192, 168, 1, 16},4001,**

**alive=1043, securePort=-1, UDP Port=-1, id={33 91 -59 78 -34 -52**

**73 -9 -99 124 -53 34 69 21 -40 -82 }, payload={}, command={},**

**domain={}, ]**

**#Instance 2 joined the cluster node**

- Tại Node 2:

**INFO: Starting Servlet Engine: Apache Tomcat/7.0.12**

**Oct 11, 2011 5:23:41 PM org.apache.catalina.ha.tcp.SimpleTcpCluster**

**startInternal**

**INFO: Cluster is about to start**

**Oct 11, 2011 5:23:42 PM**

**org.apache.catalina.tribes.transport.ReceiverBase bind**

**INFO: Receiver Server Socket bound to:/192.198.1.16:4001**

**Oct 11, 2011 5:23:42 PM**

**org.apache.catalina.tribes.membership.McastServiceImpl setupSocket**

# Instance node 1 started on port 4001

INFO: Setting cluster mcast soTimeout to 500

Oct 11, 2011 5:23:42 PM

org.apache.catalina.tribes.membership.McastServiceImpl waitForMembers

INFO: Sleeping for 1000 milliseconds to establish cluster membership,

start

level:4

**Oct 11, 2011 5:23:43 PM org.apache.catalina.ha.tcp.SimpleTcpCluster**

**memberAdded**

**INFO: Replication member**

**added:org.apache.catalina.tribes.membership.MemberImpl**

**[tcp://{192,168, 1, 15}:4000,{127, 0, 0, 1},4000, alive=1398024,**

**securePort=-1, UDP Port=-1, id={28 42 60 -68 -99 126 64 -35 -118 -**

**97 7 84 26 20 90 24 }, payload={}, command={}, domain={}, ]**

# Instance 1 joined the cluster node 2.

- 4 phần highlighted:

+ Phần đầu tiên, tomcatnode1 chạy và sẵn sàng nhận cluster message trên port 4000

INFO: Receiver Server Socket bound to:/192.168.1.15:4000

+ Phần thứ 2 hiện tomcatnode 2 đã joined vào cluster và node 1 hiện thị notification

added:org.apache.catalina.tribes.membership.MemberImpl[tcp://

{192.168.1.16, 0, 0, 1}:4001,{192, 168, 1, 16},4001, alive=1043

+ Phần thứ 3 hiện trên tomcatnode2 chạy và nhận cluster message port 4000.

INFO: Receiver Server Socket bound to:/192.198.1.16:4001

- Phần thứ 4 hiện tomcatnode1 đã join vào cluster, node 2 hiện thông báo

added:org.apache.catalina.tribes.membership.MemberImpl[tcp://

{192,168, 1, 15}:4000,{127, 0, 0, 1},4000, alive=1398024

**Cấu hình Apache cho horizontal clustering**

- Sau khi cấu hình xong cluster trên tomcat, ta tích hợp Apache server cho tomcat.

**Bước 1:**

- Tạo 1 file mới gọi mod\_jk.conf trong conf directory “APACHE\_HOME/conf”

# cd /opt/apache/conf

# vi mod-jk.conf

# Nội dung

LoadModule jk\_module modules/mod\_jk.so

JkWorkersFile conf/workers.properties

JkLogFile logs/mod\_jk.log

JkLogLevel info

JkMount /sample/\* loadbalancer

JkMount /\* loadbalancer

**Bước 2:**

- Tạo 1 file mới trong “workers.properties” trong conf direc

# vi workers.properties

worker.list=tomcatnode1, tomcatnode2, loadbalancer

- Định nghĩa worker.list cho node trong cluster

# Định nghĩa node 1 với port, ip

worker.tomcatnode1.port=8009

worker.tomcatnode1.host=192.168.1.15

worker.tomcatnode1.type=ajp13

worker.tomcatnode1.lbfactor=1

# Định nghĩa node 2 với ip, port

worker.tomcatnode2.port=8009

worker.tomcatnode2.host=192.168.1.16

worker.tomcatnode2.type=ajp13

worker.tomcatnode2.lbfactor=1

# Định nghĩa load balancing cho mod\_jk.

worker.loadbalancer.type=lb

worker.loadbalancer.balanced\_workers=tomcatnode1, tomcatnode2

worker.loadbalancer.sticky\_session=1

Bước 3: Thêm cấu hình mod\_jk.conf vào cấu hình chính “http.conf”.

# vi httpd.conf

Include conf/mod\_jk.conf # in the end of httpd.conf

## Testing of the clustered instance

- Để kiểm tra cluster, ta sẽ sử dụng các event.

1. Start tomcatnode1.

2. Start tomcatnode2 (wait for node 1 to start completely).

3. Node 1 crashes.

4. Node 2 takes over the user session of node 1 to node 2.

5. Start node 1 (wait for node 1 to start completely).

6. Node 2 and node 1 are in running state.

- Bắt đầu kịch bản:

**Bước 1: Chạy Node 1**

khởi động bằng quy trình tiêu chuẩn. Khi host được khởi tạo, cluster object sẽ liên kết với nó. Tomcat hỏi cluster class (trường hợp SimpleTcpCluster) để tạo manager for the cluster và cluster class chạy membership service.

Note:

The membership service is a mechanism in the cluster instance through the cluster domain, which adds the member node in the cluster. In simple terms, it is a service through which members are able to join the cluster.

**Bước 2: Chạy Note 2**

Khi thực thi tomcat 2 chạy, có sẽ tương tự tomcat 1 với 1 điểm khác biệt. Cluster đã khởi động và sẽ tạo ra liên kết giữa tomcatnode2 giữa (tomcatnode1, tomcatnode2). Tomcatnode2 tại thời điểm sẽ gửi request tới server đã tồn tại cluster.

Note:

In case the Tomcat instance does not respond within an interval of 60 seconds, then Tomcat instance 2 will update the cluster, and generate the entry in the logs.

**Bước 3: Node 1 lỗi**

Khi 1 thực thi tomcat crashes, cluster manager sẽ gửi thông báo tới tất cả member, trong trường hợp đó là tomcatnode2. Toàn bộ session của node 2 sẽ được tạo bản sao gửi tới node 2, nhưng user sẽ không thấy vấn đề gì khi sử dụng web.

**Bước 4: Node 2 sẽ nhận toàn bộ user session của node 1.**

Tomcatnode2 sẽ xử lý request bất kỳ reqest. User request sẽ được phục vụ tại node 2.

**Bước 5: Chạy lại Node 1:**

Khi chạy lại tomcatnode 1, đầu tiên nó sẽ join vào cluster, sau đó liên kết với tomcatnode 2 để nhận toàn bộ phiên làm việc hiện tại. Sau đấy, node 1 và node 2 sẽ phục vụ đồng thời các request.

**Bước 6: Node 2 và Node 1 cùng chạy**

Sau khi node 1 chạy, node 2 sẽ chia sẽ các session cho node 1, kết thúc nó tại node 2.

## Tài liệu mở rộng Cluster

### Cluster Basics

- Để chạy việc nhân bản session trong Tomcat 8, cấn chú ý:

+ Tất cả trường session cần phải thực thi “java.io.Serializable”

+ Mở comment “Cluster” element trong server.xml

+ Nếu bạn định nghĩa giá trị custer claster, chắc chắn rằng định trường ReplicationValve bên trong “Cluster” element.

+ Nếu các thực thi Tomcat chạy trên cùng machine, chắc chắc rằng trường “Receiver.port” khác nhau trên môi thực thi. Hầu như trong các trường hợp, tomcat sẽ tự giải quyết vấn đề này bằng cách tự động phát hiện port có sẵn trong khoảng 4000-4100.

+ Chắc chắn giá web.xml có “<distributable/>” element.

+ Nếu bạn sử dụng “mod\_jk”, chắc chắn giá trị “jvmRouter” được set trong <Engine>

(<Engine name="Catalina" jvmRoute="node01" >) và giá trị “jvmRouter” cần trùng với tên worker trong workers.properties.

+ Chắc chắn tất cả các node cấu hình NTP service.

+ Chắc chắn loadbalancer được cấu hình cho sticky session mode.

Note

+ Nhớ rằng, trạng thái session được theo dõi bởi cookie, vì thế dù URL có khác nhau nhưng phiên vẫn được giữ.

+ Cluster yêu cầu JDK version 1.5 hoặc mới hơn để chạy.

### Giới thiệu

- Để cho phép nhân bản session trong Tomcat, có 3 cách khác nhau:

+ Sử dụng session persistence (kéo dài phiên), lưu session tới shared file system (PersistenceManager + FileStore)

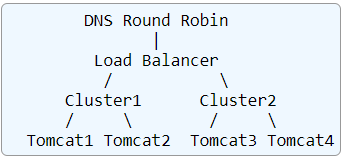
+ Sử dụng session persistence (kéo dài phiên), lưu session tới shared database (PersistenceManager + FileStore)

+ Sử dụng in-memory-replication (sao lưu bộ nhớ), sử dụng SimpleTcpCluster cùng với Tomcat (lib/catalina-tribes.jar + lib/catalina-ha.jar)

- Khi cho phép tính năng nhân bản session, tomcat sẽ thực hiện nhân bản all to all session state sử dụng “DeltaManager” hoặc thực hiện nhân bản backup tới 1 node duy nhất sử dụng “BackupManager”. Thuật toán nhân bản all to all chỉ hiệu quả đối với các cluster nhỏ. Đối với các cluster lớn, ta sẽ sử dụng phiên bản chính và phiên bản phụ cho nhân bản session, ở đó, session sẽ chỉ được lưu tại 1 backup server cài đặt “BackupManager”.

- Ở phiên bản hiện tại, ta có thể sử dụng thuộc tính domain worker (mod\_jk > 1.2.8) để xây dựng phân vùng cluster với khả năng mở rộng “DeltaManager” (cần cấu hình domain interceptor). Để giảm lưu lượng mạng trong môi trường nhân bản các session all to all, ta có thể tách riêng các cluster thành các nhóm nhỏ. Ta có thể làm điều này bằng cách sử dụng các multicast address khác nhau cho mỗi group.

### Mô hình sử dụng



- Vấn đề tiếp theo cần nói đến là, nhân bản các session sẽ bắt đầu khi chạy cluster. Một khái niệm quan trọng đi kèm theo cluster là farming. VD: Ta có thể deploy 1 app trên 1 server, cluster sẽ phân phối việc triển khai tới tất cả các node trong. Tinh năng này được cung cấp theo “FarrmWarDeployer” (Vd trong cluster tại server.xml).

### Thông tin về Cluster

- Thiết lập Membership sử dụng multicast heartbeats. Vì thế, nếu muốn chia tách Cluster thành nhiều phân, ta cần thay đổi giá trị Multicast IP address hoặc port trong thành phần <Membership>.

- Hearbeat bao gồm IP address của Tomcat node và TCP port được sử dụng cho Tomcat lắng nghe các traffic nhân bản. Tất cả kết nối data xảy ra thông qua TCP.

- “ReplicationValve” được sử dụng để xác nhận khi nào request hoàn thành và khởi động quá trình nhân bản. Data chỉ được nhân bản nếu session bị thây đổi (bằng cách gọi setAttribute hoặc removeAttribute trên session).

- 1 trong những phần quan trọng cần được cân nhắc về hiệu năng là sự đồng bộ - bất đồng bộ. Với trạng thái nhân bản đồng bộ, request sẽ không trả lại cho đến khi sự nhân bản các session được gửi tới toàn bộ Node cluster và được tái lưu trữ trên các node. Trạng thái bắt đồng bộ và động bị được cấu hình bằng cách set giá trị “channelSendOptions”, giá trị này là 1 số nguyên. Mặc định giá trị “SimpleTcpCluster/DeltaManager” sẽ bằng 8 tức trạng thái bất đồng bộ. Đối với trạng thái bất động bộ, request sẽ được trả lại ngay lập tức trước khi dữ liệu được nhân bản. Thời gian thực thi nhân bản bất đồng bộ sẽ ngắn hơn so với đồng bộ nhưng sẽ không thể bảo đảm việc truyền bằng nhân bản đồng bộ.

### Liên kết session sau khi gặp lỗi, session chuyển tới node chịu lỗi

- Nếu bạn sử dụng “mod\_jk” và không sử dụng sticky session vì 1 số lý do sticky session không hoạt động hoặc gặp lỗi, Session id sẽ bị thay đổi vì nó chức worker id của tiến trình tomcat lỗi (như định nghĩa “jvmRouter trong thành phần Engine”. Để giải quyết vấn đề này, ta sử dụng “JvmRouteBinderValve”.

- “JvmRouteBinderValve” tái ghi lại session id để chắc chắn request tiếp theo sẽ vẫn còn Sticky sau khi vượt qua lỗi(Và không báo lỗi khi chuyển session tới 1 node bất kỳ nào đó). valve ghi lại giá trị “JSESSIONID” trong cookie với cùng tên. Nếu không cấu hình valve chính xác, nó khó khăn khi chắc chắn stickness trong trường hợp lỗi vì mod\_jl module.

Note:

- Bảo đảm các giá trị thêm mới trong server.xml chính xác

Gợi ý:

- Với trường “sessionIdAttribute”, ta có thể thay đổi thuộc tính tên request được thêm vào trong session cũ. Mặc định thuộc tính tên là “org.apache.catalina.ha.session.JvmRouteOrignalSessionID”.

### Cấu hình đơn đơn giản:

**<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster" channelSendOptions="6">**

<Manager className="org.apache.catalina.ha.session.BackupManager" expireSessionsOnShutdown="false" notifyListenersOnReplication="true" mapSendOptions="6" />

<!--

<Manager className="org.apache.catalina.ha.session.DeltaManager"

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"/>

-->

<Channel className="org.apache.catalina.tribes.group.GroupChannel">

<Membership className="org.apache.catalina.tribes.membership.McastService" address="228.0.0.4" port="45564" frequency="500" dropTime="3000" />

<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver" address="auto" port="5000" selectorTimeout="100" maxThreads="6" />

<Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">

<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender" />

</Sender>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector" />

<Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatchInterceptor" />

<Interceptor className="org.apache.catalina.tribes.group.interceptors.ThroughputInterceptor" />

</Channel>

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve" filter=".\*\.gif|.\*\.js|.\*\.jpeg|.\*\.jpg|.\*\.png|.\*\.htm|.\*\.html|.\*\.css|.\*\.txt" />

<Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer" tempDir="/tmp/war-temp/" deployDir="/tmp/war-deploy/" watchDir="/tmp/war-listen/" watchEnabled="false" />

<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener" />

</Cluster>

- Để ý phần in đậm.

- Tại thành phần main, bên trong nó chứa tất cả thông tin vể cluster đã đượ cấu hình. Thuộc tính “channelSendOptions” sẽ được thêm vào mỗi message được gửi đi bởi lớp “SimpleTcpCluster” hoặc bất kỳ đối nào thêm phương thức “SimpleTcpCluster.send”.

Note:

Tìm hiểu thêm tại: <http://tomcat.apache.org/tomcat-8.5-doc/api/org/apache/catalina/tribes/Channel.html>

- Cấu hình mặc định cho manager confi sẽ được sử dụng nếu không định nghĩa manager trong thành phần “<Context>”. Trong Tomcat 5.x mỗi web app sẽ được đánh dấu phân bố chung 1 manager. Ở phiên bản hiện tại, có thể định nghĩa riêng manager class cho mỗi web app. Và đối với cluster, các manager được kết nối với nhau. Nếu không chỉ định manager cho web app, cầu hình mặc định sẽ tạo thực thi manager cho Tomcat.

Note:

<Channel className="org.apache.catalina.tribes.group.GroupChannel">

Xem thêm: <http://tomcat.apache.org/tomcat-8.5-doc/config/cluster-manager.html>

- Membership được thực hiện bởi multicasting. Bên cạnh đó Triber cũng hỗ trợ các static membership bằng tham số “StaticMembershipInterceptor”.

- Thuộc tính “address” và “port” là các trường được sử dụng để multicasting. 2 thuộc tính này cũng tạo ra sự tách biệt giữ 2 cluster.

- Thành phần membership quảng bá TCP address/port của chính nó tới các node khác vì thế, kết nối giữa các node mới có thể được thiết lập

VD:

<Membership className="org.apache.catalina.tribes.membership.McastService"

address="228.0.0.4"

port="45564"

frequency="500"

dropTime="3000"/>

- Quá trình gửi và nhận data thông qua 2 thành phần chính: receiver và sender.

- Receiver, như tên nó chịu trách nhiệm nhận các message. Có 1 thread pool trong thành phần này, định nghĩa thông qua “maxThreads” và “minThreads”.

- Thuộc tính “address” chính là host address được quản bá bởi thành phần membership.

<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"

address="auto"

port="5000"

selectorTimeout="100"

maxThreads="6"/>

- Sender là thành phần chịu trách nhiệm gửi thông tin tới các node. Sender có thành phần riêng đi kèm “ReplicationTransmitter” nhưng trong thực tế, vấn đề này được giải quyết bởi thành phần con khác “Transport”.

- Trisber hỗ trợ 1 pool cho sender, vì thế message có thể gửi song song và nếu sử dụng NIO, ta có thể gửi nhiều thông tin đồng thời.

VD:

<Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">

<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>

</Sender>

- Tribes sử dụng stack để gửi message. Mỗi thành phần trong stack được gọi là 1 “interceptor” (Chặn), nó giống như valve trong Tomcat servelt container. Sử dụng các “interceptor” , các logic có thể được phân nhỏ thành các code riêng biêt, dễ dàng hơn cho việc quản lý

**Note:**

- Vị trí khai báo các interceptor khá quan trọng, trong định nghĩa server.xml là cách chúng được sắp xếp trong stack quyết định quá trình xử lý.

VD: Cấu hình mặc định

+ TcpFailureDetector - verifies crashed members through TCP, if multicast packets get dropped, this interceptor protects against false positives, ie the node marked as crashed even though it still is alive and running.

+ MessageDispatchInterceptor - dispatches messages to a thread (thread pool) to send message asynchronously.

+ ThroughputInterceptor - prints out simple stats on message traffic.

- Cluster cũng cung cấp thành phần kiểm soát request tới web app. Thành phần Cluster không thuộc pipeline Tomgcat, vì thế, cluster được thêm valve cho “Parent container”. Nếu thành phần <Cluster> được thêm vào thành phần <Engiene>, valve sẽ được thêm vào Engine.

VD:

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve"

filter=".\*\.gif|.\*\.js|.\*\.jpeg|.\*\.jpg|.\*\.png|.\*\.htm|.\*\.html|.\*\.css|.\*\.txt"/>

- Mặc định, tomcat hỗ trợ farmed cho việc triển khai, nó cho phép kiểm soát việc deploy và undeploy tới nhiều node.

VD:

<Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"

tempDir="/tmp/war-temp/"

deployDir="/tmp/war-deploy/"

watchDir="/tmp/war-listen/"

watchEnabled="false"/>

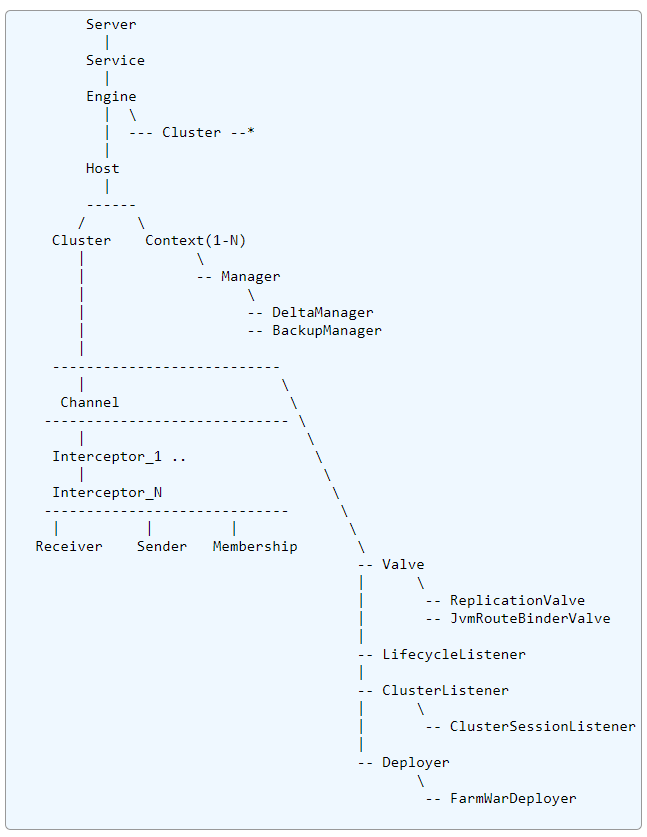
- Từ khi “SimpleTcpCluster” là thành phần gửi và nhận của đối tượng Channel, thành phần nào có thể đăng ký chính nó như listener cho “SimpleTcpCluster”.

<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>

</Cluster>

### Kiến trúc Cluster:

- Các mức thành phần:



### Kịch bản Cluster Tomcat:

TomcatA starts up

TomcatB starts up (Wait that TomcatA start is complete)

TomcatA receives a request, a session S1 is created.

TomcatA crashes

TomcatB receives a request for session S1

TomcatA starts up

TomcatA receives a request, invalidate is called on the session (S1)

TomcatB receives a request, for a new session (S2)

TomcatA The session S2 expires due to inactivity.

Bước 1: Tomcat A start up

- Tomcat khởi động tiêu chuẩn. Khi Host object được tạo, cluster object được liên kết với nót. Khi contexts được phân tích, nếu distribute element được lưu tại web.xml, Tomcat sẽ hỏi cluster class (trong SimpleTcpCluster) tạo manager cho việc tạo bản sao context. Vì clustering đã enable, distributable được set trong web.xml, Tomcat sẽ tạo “DeltaManager” cho context thay thì “StandaraManager”. Cluster class sẽ khởi động membership service (multicast) và tạo bản sao service (tcp unicast).

Bước 2: TomcatB starts up

- Khi TomcatB khởi động, nó sẽ tuân theo các bước khởi động TomcatA với 1 ngoại lệ. Cluster đã chạy và sẽ thiết lập quan hệ (TomcatA, TomcatB). TomcatB hiện tại sẽ yêu cầu trạng thái session từ server (đã tạo trong cluster tại TomcatA). TomcatA sẽ đáp trả request, và trước khi TomcatB bắt đầu lắng nghe HTTP request, trạng thái sẽ được chuyển từ TomcatA tới TomcatB. Trong trường hợp TomcatA ko trả lời, TomcatB sẽ chờ 60s, hiện log. Session state được truyền cho mỗi web app có thể được phân phố trong tệp web.xml.

Note:

Đế sử dụng session replication hiệu quả, tất cả thực thi Tomcat cần được config giống nhau.

Bước 3: TomcatA receives a request, a session S1 is created.

- Request tời TomcatA được xử lý nhưng ko nhân bản session. Hoạt động xảy ra khi request được hoàn thành, “ReplicationValve” sẽ chặng các request trước khi respone được trả lại cho user. Tại thời điểm nó tìm thấy session được sửa đổi, nó sử dụng TCP cho tại bản sao session tới TomcatB. Khi việc xếp hàng dữ liệu được bàn giào cho operating system TCP logic, request được trả lại user, trở lại valve pipeline. Cho mỗi request được tạo bản sao, nó cho phép sử dụng code chỉnh sửa thuộc tính mỗi session mà không phải gọi setAttribute or removeAttribute. “useDirtyFlag” configuration parameter có thể được sử dụng để tối đa số thời gian session được nhân bản.

Bước 4: TomcatA crashes

- Khi TomcatA crashes, TomcatB sẽ nhận thông báo TomcatA đã dropped khỏi cluster. Tomcat B xóa Tomcat khỏi membership list, và TomcatA sẽ không còn nhận được thông báo từ bất kỳ thày đổi xảy ra trong TomcatB. Load balancer sẽ redirect request từ Tomcat A tới TomcatB cùng với tất cả sessions hiện tại.

Bước 5: TomcatB receives a request for session S1

- TomcatB sẽ xử lý request như bất kỳ request tới.

Bước 6: TomcatA starts up

- Khi khởi động, trước khi TomcatA bắt đầu xử lý các request mới và khiến nó có sẵn. Nó sẽ join và cluster, liên lạc với TomcatB để biết các state hiện tại cho mỗi sessions. Khi nó nhận lại session state, nó sẽ kết thúc quá trình loading và mở port HTTP/mod\_jk của nó. Không có request sẽ tới TomcatA cho đến khi nó nhận lại được session state từ TomcatB.

Bước 7: TomcatA receives a request, invalidate is called on the session (S1)

- Các invalidate call sẽ được chặn cùng với session trong queued không hợp lệ. Khi request được hoàn thành, thay vì gửi session đã thay đổi, nó sẽ gửi message hết hạn đến TomcatB và Tomcat sẽ mất hiệu lức đối với phiên.

Bước 8: TomcatB receives a request, for a new session (S2)

- Quay lại bước 3

Bước 9: TomcatA The session S2 expires due to inactivity. (phiên s2 hết hạn do không hoạt động)

- Invalidate call bị chặn giống như khi sesion bị vô hiệu bởi người dùng, và session trong queue không hợp lệ. Tài thời điểm, các session ko hợp lệ sẽ không được nhân bản cho đến khi nó trở lại trạng thái hợp lệ.

### The Cluster object

**Giới thiệu**

- Thực thi Tomcat cluster cung cấp nhân bản session, nhân bản context atrribute và cluster wide WAR file deployment (Triển khai diện rộng). Trong khai Cluster config khá phức tạp, config mặc định sẽ làm việc cho hầu hết mọi người (không quan tâm nhiều đến tùy chỉnh bên trong).

- Thực thi Tomcat Cluster rất rộng, và vì thế ta cần hiểu các tùy chọn cơ bản, để làm được điều này, ta phải hiểu các vấn đề bên trọng.

**Security**

- Thực thi Cluster hoạt động dựa trên sự đảm bảo cơ bản, sử dụng trusted network cho tất cả cluster các network traffic liên quan đến cluster. Nó sẽ không an toàn nếu chạy cluster trong phân vùng không đảm bảo.

- Có rất nhiều option cung câp sử bảo đảm, trusted network cho Tomcat cluster. Bao gồm:

private LAN

a Virtual Private Network (VPN)

IPSEC

**Engine vs Host placement**

- Ta có thể lưu <Cluster> element trong <Engine> hoặc <Host> container.

- Lưu nó trong engine nghĩa là bạn sẽ hỗ trợ clustering cho tất cả virtual host of Tomcat và chia sẻ messaging component. Khi bạn lưu <cluster> bên trong <Engine> element, cluster sẽ thêm host name cho mỗi session manager để 2 context có cùng tên sẽ đặt bên trong 2 host khác nhau dễ phân biệt.

**Context Attribute Replication**

- Để cấu hình cấu hình context atribute replication, thực hiện đơn giản bằng cách tráo đổi thực thi context được sử dụng cho app context.

<Context className="org.apache.catalina.ha.context.ReplicatedContext"/>

**Thành phần con**

**Manager:**

- Session manager element là 1 loại session manager được sử dụng cho việc thực thi cluster. Cấu hình manager giống cấu hình sử dụng trong <Context> config thuông thường.

- Giá trị mặc định là “org.apache.catalina.ha.session.DeltaManager” được liên kết chắt chẽ với thực thi “SimpleTcpCluster”. Manager khác như “org.apache.catalina.ha.session.BackupManager” có thể không có quan hệ chặt chẽ với “SimpleTcpCluster” cho việc replication data.

**Channel:**

- Channel và sub component của nó là các phần trong IO layer trong cluster group, và ta đặt tên cho nó là “Tribes”

**Valve:**

- Thực thi Tomcat Cluster sử dụng “Tomcat Valves” để theo dõi request truy cập và đi ra khởi servlet container. Nó sử dụng valve để đưa ra các quyết định thông minh khi replication date mà luôn luôn ở cuối request.

**Deployer:**

- Thành phần Deployer là Tomcat Farim Deployer. Nó cho pháp bạn deploy hoặc undeploy app suốt cluster.

**ClusterListener:**

- ClusterListener được sử dụng để theo dõi message gửi và nhận sử dụng “SimpleTcpCluster”. Nếu bạn mong muốn theo dõi messsage, bạn có thể thêm listener tại đây, hoặc có thể thêm valve tới channel object.

Tìm hiểu thêm:

<https://tomcat.apache.org/tomcat-8.5-doc/config/cluster.html>

### The ClusterManager object

**Giới thiệu**

- Cluster manager là thành phần mở rộng của Tomcat’s session manager interface, “org.apache.catalina.Manager”. Cluster manage cần được thực thi “org.apache.catalina.ha.ClusterManager” và tự chịu trách nhiệm về cách session được nhân rộng.

- Có 2 manager khác nhau, “org.apache.catalina.ha.session.DeltaManager” sao chép delta của session data tới tất cả member trong cluster. Thực thi này đã được chứng minh hoạt động tốt, nhưng có hạn chế là các thành viên phải trong cụm đồng nhất, tất cả node cần được deploy same app và cùng phiên bản. “org.apache.catalina.ha.session.BackupManager” cũng nhân bản delta nhưng chỉ trên node backup. Ví trị backup được biết trên toàn bộ cluster. Nó cũng hỗ trợ deployment không đồng nhất, do đó manager biết được vị trí ứng dụng web được triển khai.

**The <Manager>**

- <Manager> element được định nghĩa bên trong <cluster> element, được định nghĩa theo mẫu tại web.xml. Tuy nhiên ta vẫn có thể override lại thực thi manager trên mỗi web app bằng cách đặt <manager> bên trong <context> element tại “context.xml” file hoặc “server.xml”.

Tìm hiểu thêm:

<https://tomcat.apache.org/tomcat-8.5-doc/config/cluster-manager.html>

### The Cluster Channel object

**Giới thiệu**

- Cluster channel là thành phần chính trong small framework ta đặt tên là Apache Tribes.

- Channel manager là tập các sub component và cùng vời nhau chúng tạo nên group communication framework.

- Framework này được sử dụng nội bộ bởi thành phần cần gửi message giữa các thực thi Tomcat khác nhau.

- ví dụ về thành phần là “SimpleTcpCluster” thực hiện các messaging cho DeltaManager, hoặc BackupManager sử dụng các replication stragy khác nhau. “ReplicatedContext object” cũng sử dụng channel object để kết nối context attribute changers.

**Thành phần con**

**Channel/Membership:**

- Membership component chịu trách nhiệm tự động tìm kiểm new node trong cluster và cùng như cung cấp các thông báo bất kỳ node không đáp trả với “heartbeat”. Mặc định thực thi sử dụng multicast. Trong thành phần membership, ta cấu hình các node, member sẽ tìm kiếm và phân chia chúng.

**Channel/Sender:**

- Thành phần Sender quản lý tất cả kết nối ra ngoài và data message được gửi thông qua mạng từ 1 node tới 1 node khác. Thành phần cho phép message được gửi song song. Mặc định thực thi sử dụng TCP client socket, socket tuning cho các message gửi đi sẽ được cấu hình tại đây.

**Channel/Sender/Transport:**

- Thành phần Transport ở lớp IO dưới cùng của thành phần gửi. Mặc đinh thực thi sử dụng non-blocking TCP client sockets.

**Channel/Receiver:**

- Thành phần Receiver lăng nghe các messger tới từ node khác. Ở đây ta cấu hình cluster thread pool. Nó sẽ gửi tin nhắn tới đến 1 thread nhanh hơn. Mặc định thực thi sử dụng non-blocking TCP server socket.

**Channel/Interceptor:**

- Chanel sẽ gửi messages thông qua interceptor stack. Vì thế, bạn có khả năng customize lại cách message được gửi và nhận và thậm chí cách xử lý membership.

Tìm hiểu thêm:

<https://tomcat.apache.org/tomcat-8.5-doc/config/cluster-channel.html>

### The Cluster Valve object (Van Cluster object)

**Giới thiệu**

- Cluster valve không khác biệt so với Tomcat Valve. Cluster valves là interceptor trong chuỗi request HTTP, và thực clustering thực thi sử dụng valve để đưa ra quyết định thông minh xung quanh data và khi data cần được nhân bản.

- Cluster valve cần thực thi “org.apache.catalina.ha.ClusterValve” interface. Nó đơn giản là interface mở rộng “org.apache.catalina.Valve” interface.

**org.apache.catalina.ha.tcp.ReplicationValve**

- The ReplicationValve will notify the cluster at the end of a HTTP request so that the cluster can make a decision whether there is data to be replicated or not.

Tìm hiểu thêm:

<https://tomcat.apache.org/tomcat-8.5-doc/config/cluster-valve.html>

### The Cluster Deployer object

**Giới thiệu**

- Farm War Deployer có thể deploy và undeploy web app trên trên node cluster khác.

**org.apache.catalina.ha.deploy.FarmWarDeployer**

Attributes

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **className** | The cluster deployer class, currently only one is available,org.apache.catalina.ha.deploy.FarmWarDeployer. |
| **deployDir** | Deployment directory. This is the pathname of a directory where deploy the web applications. You may specify an absolute pathname, or a pathname that is relative to the $CATALINA\_BASE directory. In the current implementation, this attribute must be the same value as the **Host's appBase**. |
| **tempDir** | The temporaryDirectory to store binary data when downloading a war from the cluster. You may specify an absolute pathname, or a pathname that is relative to the $CATALINA\_BASE directory. |
| watchDir | This is the pathname of a directory where watch for changes(add/modify/remove) of web applications. You may specify an absolute pathname, or a pathname that is relative to the $CATALINA\_BASE directory.**Note:**if **watchEnabled** is false, this attribute will have no effect. |
| watchEnabled | Set to true if you want to watch for changes of web applications. Only when this attribute set to true, you can trigger a deploy/undeploy of web applications. The flag's value defaults to false. |
| processDeployFrequency | Frequency of the Farm watchDir check. Cluster wide deployment will be done once for the specified amount of backgroundProcess calls (ie, the lower the amount, the most often the checks will occur). The minimum value is 1, and the default value is 2. **Note:**if **watchEnabled** is false, this attribute will have no effect. |
| maxValidTime | The maximum valid time(in seconds) of FileMessageFactory. FileMessageFactory will be removed immediately after receiving the complete WAR file but when failing to receive a FileMessage which was sent dividing, FileMessageFactory will leak without being removed. FileMessageFactory that is leaking will be automatically removed after maxValidTime. If a negative value specified, FileMessageFactory will never be removed. If the attribute is not provided, a default of 300 seconds (5 minutes) is used. |

Tìm hiểu thêm:

<https://tomcat.apache.org/tomcat-8.5-doc/config/cluster-deployer.html>

**The ClusterListener object**

**Giới thiệu**

- “org.apache.catalina.ha.ClusterListener” dựa trên class cho phép bạn lắng nghe message nhận được từ thành phần cluster.

**org.apache.catalina.ha.session.ClusterSessionListener**

- Khi sử dụng DeltaManager, messagse sẽ nhận bởi Cluster object và thông báo cho manager thông qua những gì nghe được

Tìm hiểu thêm:

<https://tomcat.apache.org/tomcat-8.5-doc/config/cluster-listener.html>

## Apache Tribes

### Giới thiệu

- Apache Tribes là group or peer-to-peer framework kết nối cho phép bạn dễ dàng kết nối các remote object tới các thành phần khác.

**Bao gồm:**

Import: org.apache.catalina.tribes.Channel

Import: org.apache.catalina.tribes.Member

Import: org.apache.catalina.tribes.MembershipListener

Import: org.apache.catalina.tribes.ChannelListener

Import: org.apache.catalina.tribes.group.GroupChannel

Create a class that implements: org.apache.catalina.tribes.ChannelListener

Create a class that implements: org.apache.catalina.tribes.MembershipListener

Simple class to demonstrate how to send a message:

Vd:

//create a channel

Channel myChannel = new GroupChannel();

//create my listeners

ChannelListener msgListener = new MyMessageListener();

MembershipListener mbrListener = new MyMemberListener();

//attach the listeners to the channel

myChannel.addMembershipListener(mbrListener);

myChannel.addChannelListener(msgListener);

//start the channel

myChannel.start(Channel.DEFAULT);

//create a message to be sent, message must implement java.io.Serializable

//for performance reasons you probably want them to implement java.io.Externalizable

Serializable myMsg = new MyMessage();

//retrieve my current members

Member[] group = myChannel.getMembers();

//send the message

myChannel.send(group,myMsg,Channel.SEND\_OPTIONS\_DEFAULT);

- Có rất nhiều ví dụ về Tribes mà ta có thể shown, xem thêm tại docs.

Note:

Tribes được xây dựng trên JDK1.5

### Định nghĩa Tribes

- Tribes là messaging framework với khả năng giao tiếp giữa các nhóm. Tribes cho phép bạn gửi và nhận message thông qua network, nó cũng cho phép tìm kiếm các node khác trong mạng.

- Tribes module bắt đầu năm 2006, là 1 phần nhỏ của code base tới từ clustering module tồn tại 2003 – 2004.

- Tribes nắm giữ phần phức tạp của messaging của replicatuon module và trở thành 1 thành phần độc lập, linh động trong group communication module.

- Trong Tomcat cữ “molules/cluster” trở thành “modules/groupcom”(Tribes) và “modules/ha” (replication). Nó sẽ cho phép các phát triển tập trung và các dev có thể tập trung vào các vấn đề mà đang làm việc mà không cần quan tâm đến các phần bên dưới.

- Tribes sẽ cho phép bảo điểm messaging và có thể tùy chỉnh theo nhiều các. Đây là 1 thành phần quan trọng vì người phát triển muông biết tin nhắn họ gửi đến được đích mong muốn, và hơn thế nữa, nếu message không tới được đích, app trên top Tribes sẽ thông bao message không được gửi và node nào lỗi.

### Giới thiệu tính năng

- Một số tính năng quan trọng trong Tribes

**Pluggable modules (Khả năng tích hợp)**

- Tribes được xây dựng sử dụng interfaces. Bất kỳ modules hoặc thành phần thuộc Tribes có thể tách ra customize thành phần đó.

**Guaranteed Messaging (Bảo đảm truyền tin)**

- Mặc định thực thi Tribes sử dụng TCP hoặc UDP cho tin nhắn. TCP đã có khả năng bảo đảm sự vận chuyển message và flow control built in. Và tin rằng hiệu năng của Java TCP sẽ hơn hẳn so với Java/UDP/flow-control/message. UDP messaging đã được thêm vào cho việc gửi message thông qua UDP thay vì TCP khi thiết kế. Có kịch bản bảo đảm được mô tả cho UDP nhưng vấn đề xảy ra kaf khi message mất đi, nó không được xem xét là lỗi.

- Tribes hỗ trợ cả non-blocking và blocking IP operations. Khuyến khích setting sử dụng non blocking vì nó sẽ hỗ trợ tính song song tốt hơn khi gửi là nhận messages. Blocking thực thi có sẵn trong NIO nhưng vẫn tồn tại lỗi.

**Các mức bảo đảm**

- Có 3 mức bảo đảm khác nhau khi gửi message.

**1. IO based send guarantee – nhanh nhất, ít tin cậy.**

- Có nghĩa tribes sẽ cần nhắc truyền message thành công nếu message đã gửi tới socket buffer và accepted.

- Trên blocking IO, nó sẽ là “socket.getOutputStream().write(msg)”

- Trên non-blocking IP, nó sẽ là “socketChannel.write()”, buffer byte buffer sẽ được làm mới cùng với “socketChannel.read()” để chắc chắn channel vẫn mở.

**2. ACK based – Khuyến khích, đảm bảo sự vận chuyển**

- Khi message đã được nhận trên remote node, ACK sẽ gửi lại sender, cho biết rằng message đã được nhận.

**3. SYNC\_ACK based. – đảm bản vận chuyển, bảo đảm xử lý, chậm nhất.**

- Khi message đã được nhận trên remote node, node sẽ xử lý message và nếu message được xử lý thành công, ACK sẽ gửi lại tới sender cho biết message đã được nhận và xử lý thành công. Nếu message được nhận nhưng xử lý lỗi, ACK\_FAIL sẽ gửi lại tới sender. Đây là 1 tính năng riêng hỗ trợ cho việc phát triển ứng dụng. Hầu như frameworks sẽ cho bạn biết message đã được gửi và application developer sẽ xây dựng logic cho message đã thực sự xử lý trên app tại remote node. Nếu config, Tribes sẽ gửi exception khi nhận được ACK\_FAIL cùng với thông điệp member không xử lý message.

**Per Message Delivery Attributes (Thuộc tính phân phối trên mỗi message)**

- Có thể tính năng kiến Tribes tách biệt khởi group commnication frameworks. Tribes cho phép bạn gửi và quyết định nhưng semantisc message truyền nên có trên mối message. Có nghĩa là, message sẽ không truyền dựa trên 1 số static config giữa nguyên khi message framework được bắt đầu.

VD: Gửi 10 message

Message\_1 - asynchronous and fast, no guarantee required, fire and forget

Message\_2 - all-or-nothing, either all receivers get it, or none.

Message\_3 - encrypted and SYNC\_ACK based

Message\_4 - asynchronous, SYNC\_ACK and call back when the message is processed on the remote nodes

Message\_5 - totally ordered, this message should be received in the same order on all nodes that have been

send totally ordered

Message\_6 - asynchronous and totally ordered

Message\_7 - RPC message, send a message, wait for all remote nodes to reply before returning

Message\_8 - RPC message, wait for the first reply

Message\_9 - RPC message, asynchronous, don't wait for a reply, collect them via a callback

Message\_10- sent to a member that is not part of this group

Note:

As you can imagine by now, these are just examples. The number of different semantics you can apply on a per-message-basis is almost limitless. Tribes allows you to set up to 28 different on a message and then configure Tribes to what flag results in what action on the message.

Imagine a shared transactional cache, probably >90% are reads, and the dirty reads should be completely unordered and delivered as fast as possible. But transactional writes on the other hand, have to be ordered so that no cache gets corrupted. With tribes you would send the write messages totally ordered, while the read messages you simple fire to achieve highest throughput.

There are probably better examples on how this powerful feature can be used, so use your imagination and your experience to think of how this could benefit you in your application.

**Interceptor based message processing (Xử lý dựa trên tin nhắn nhận được)**

- Tribes sử dụng interceptor stack được tùy chỉnh để xử lý các tin nhắn gửi và nhận.

- Tribes interceptor có thể phản ứng lại các message dựa trên mỗi “message-attributes” gửi runtime. Có nghĩa là, nếu ta thêm bào encryption interceptor mà sẽ encrypt toàn bộ tin nhắn hoặc nhưng message đc chỉ định.

- Số lượng các interceptor có sẵn đang tiếp tục tăng và có thể đóng góp.

**Threadless Interceptor stack (Giảm bớt thread Interceptor stack)**

- Interceptor không yêu cầu bất kỳ thread riêng biệt để thực hiện thao tác thông báo chung.

- Message được gửi sẽ piggy back trên thread mà đã gửi chúng tới tất cả way thông qua transmission. Exception là “MessageDispatchInterceptor” sẽ xếp trên tin nhắn và gửi nó thông qua 1 thread riêng biệt đổi với các bất đồng bộ message được truyền. Message nhận được quản lý bởi thread pool trong “receiver” component. Kênh object có thể gửi “heartbeat()” thông qua interceptor stack để cho phep timeout, cleanup và 1 số event khác.

Note:

The MessageDispatchInterceptor is the only interceptor that is configured by default.

**Parallel Delivery (truyền song song)**

- Tribes hỗ trợ truyền giao song song cho message. Có nghĩa là node\_A gửi 3 message tới node\_B song song. Tính năng này trở nên hữu ích khi gửi message khác delivery semantics. Bên cạnh đó, nếu Message\_1 được gửi là ordered, Message\_2 sẽ chờ cho đến khi message đó hoàn thành.

- Thông qua NIO (Network Input Output), Tribes cũng cho phép gửi message đến 1 số receiver cùng 1 thời điểm trên cùng 1 thread.

**Silent Member Messaging**

- Với Tribes, ta có thể gửi message tới member không nằm trong group. Mặc định, ta có thể gửi message thông qua WAN, mặc dù dynamic discover module (module phát hiện động) này được giới hạn trong mạng LAN bằng các sử dụng multicast để tìm các node động. Tính năng gửi mạng WAN sẽ được phát triển thêm trong tương lai. Nhưng nó sẽ rất hữu ích khi ta muốn dấu member khỏi phần còn lại của nhóm.

e