Tìm hiểu Apache tomcat

- Apache Tomcat là 1 opensource được viết dựa trên Java-based web và servlet container.

- Sử dụng để host ứng dụng Java based

- Được khởi tạo bởi Jakarta Tomcat nhưng sau khi project dần lớn, nó được thiết kế, nâng cấp lại bởi James Duncan Davidson, kiến trúc sư phần mềm thuộc Sun Microsystems. Apache tomcat được thiết kế lại, trở thành 1 opensource và được ủng hộ cho The Apache Software Foundation.

- Tom cat sử dụng Java Servlet và JavaServer Pages (JSP) được cung cấp bởi Sun

Microsystems, cung cấp môi trường Pure Java HTTP cho việc chạy Java code.

History of TomcatTomcat support matrix

- Apache tomcat có thể phân loại thành nhiều thành phần khác nhau như:

+ JDK version, enhancement, stability, …

- Ví dụ: Giả sử dụng ứng dụng sử dụng Servlet 2.4 và JSP 2.0 => ta phải lựa chọn version 5.x để có thể chạy service tốt nhất.

- Trong thực tể để lựa chọn phiên bản tốt nhất cả Tomcat thì phải viết framework sử dụng cho ứng dụng Java.

Xem thêm: <http://tomcat.apache.org/whichversion.html>

Features and enhancements of Apache Tomcat 7

- Apache Tom cat 7.x được phát hành, giải quyết 1 số vấn đề của phiên bản trước.

Web application memory leak detection and prevention

- Tomcat had a chronological problem of memory leaks in 4.x/5.x versions. While

reloading the applications in the entire life cycle of Tomcat, OutOfMemoryError

exceptions were generated. Tomcat has put an exceptional effort in tracking down

the bugs and issues related to memory, in order to avoid memory leaks.

Servlet 3.0

- Asynchronous Support: Servlet 3.0's asynchronous support has been

fully integrated into Tomcat 7. The biggest advantage of asynchronous

programming is that the server doesn't have to wait for the response from

the resources. For example, if you have 2000 concurrent users using an

application, then we cannot allocate 2000 connections to the database and

make the connection idle untill we get the response. By using asynchronous

programming, your application can handle other user requests while this

particular user is waiting for the response from the resources, such as DB,

NAS, and so on.

- Dynamic Configuration: It is again, a very vital feature of Servlet 3.0. Due

to Tomcat 7's support for web fragments, developers now don't need to

implement the specific library configurations for their application web.xml.

This means, you can integrate the library reference in web.xml

- Annotation-based Configuration: With the additional support for Servlet

3.0 in Tomcat 7, developers can include decorative programming styles.

The biggest advantage of implementing decorator is you can configure

rewrite rules in the application servlet classes instead of web servers. Hence,

you reduce the dependency on web servers. It also eliminates the need for

deployment descriptors.

Improved logging

Tomcat 7 includes two new features for logging, in order to provide a good

understanding to the users for log analysis:

• Asynchronous file handler: The asynchronous handler allows Tomcat to

write logs to the disk by a dedicated thread, so that logging operations do not

cause any delay in processing threads.

• Single line log formatter: The single line formatter writes logs in a single

line, which is a better feature for administrators.

Aliases

This is the best feature for an administrator. It provides the administrator with

the freedom to eliminate the dependency of another web server to host multiple

websites. In addition to this, you can host the entire static content (image/JavaScript

in a single package).

Cài đặt Tomcat 7

- Triển khai trên CentOS 7.

Download package tomcat từ Apache tomcat.

Bước 1: Truy cập <http://tomcat.apache.org/download-70.cgi>

# wget <http://mirror.downloadvn.com/apache/tomcat/tomcat-7/v7.0.81/bin/apache-tomcat-7.0.81.tar.gz>

# wget <http://mirror.downloadvn.com/apache/tomcat/tomcat-7/v7.0.81/bin/apache-tomcat-7.0.81.zip>

Bước 2: Cài đặt nền tảng Java

- Vì Tomcat được chạy trên nền java nên cần cài đặt môi trường cho Tom Cat chạy

# yum install java -y

Bước 3: Unzip gói tin

# unzip apache-tomcat\*

Bước 4: Truy cập thư mục vừa giải nén

# cd apache-tomcat/bin

Bước 5: cấp quyền thực thi script file

# chmod 0755 \*.sh

Bước 6: Cài đặt, chạy service Tomcat

- Check version gói tin thông qua script

# ./version.sh

Using CATALINA\_BASE: /opt/apache-tomcat-7.0.12

Using CATALINA\_HOME: /opt/apache-tomcat-7.0.12

Using CATALINA\_TMPDIR: /opt/apache-tomcat-7.0.12/temp

Using JRE\_HOME: /opt/jdk1.6.0\_24

Using CLASSPATH: /opt/apache-tomcat-7.0.12/bin/bootstrap.jar:/opt/

apache-tomcat-7.0.12/bin/tomcat-juli.jar

Server version: Apache Tomcat/7.0.12

Server built: Apr 1 2011 06:13:02

Server number: 7.0.12.0

OS Name: Linux

OS Version: 2.6.18-8.el5

Architecture: i386

JVM Version: 1.6.0\_24-b07

JVM Vendor: Sun Microsystems Inc.

- Kiểm tra các cấu hình trước khi chạy, sử dụng để phát hiện lỗi khi config

# ./configtest.sh

Using CATALINA\_BASE: /opt/apache-tomcat-7.0.12

Using CATALINA\_HOME: /opt/apache-tomcat-7.0.12

Using CATALINA\_TMPDIR: /opt/apache-tomcat-7.0.12/temp

Using JRE\_HOME: /opt/jdk1.6.0\_24

Using CLASSPATH: /opt/apache-tomcat-7.0.12/bin/bootstrap.jar:/opt/

apache-tomcat-7.0.12/bin/tomcat-juli.jar

May 22, 2011 4:06:16 PM org.apache.coyote.AbstractProtocolHandler init

INFO: Initializing ProtocolHandler ["http-bio-8080"]

May 22, 2011 4:06:16 PM org.apache.coyote.AbstractProtocolHandler init

INFO: Initializing ProtocolHandler ["ajp-bio-8009"]

May 22, 2011 4:06:16 PM org.apache.catalina.startup.Catalina load

INFO: Initialization processed in 1401 ms

- Chạy service apache tomcat

# ./startup.sh

Using CATALINA\_BASE: /opt/apache-tomcat-7.0.12

Using CATALINA\_HOME: /opt/apache-tomcat-7.0.12

Using CATALINA\_TMPDIR: /opt/apache-tomcat-7.0.12/temp

Using JRE\_HOME: /opt/jdk1.6.0\_24

Using CLASSPATH: /opt/apache-tomcat-

7.0.12/bin/bootstrap.jar:/opt/apache-tomcat-7.0.12/bin/

tomcat-juli.jar

- Cách tắt service tomcat

# ./shutdown.sh

Using CATALINA\_BASE: /opt/apache-tomcat-7.0.12

Using CATALINA\_HOME: /opt/apache-tomcat-7.0.12

Using CATALINA\_TMPDIR: /opt/apache-tomcat-7.0.12/temp

Using JRE\_HOME: /opt/jdk1.6.0\_24

Using CLASSPATH: /opt/apache-tomcat-

7.0.12/bin/bootstrap.jar:/opt/apache-tomcat-7.0.12/bin/tomcat-juli.jar

- Truy cập giao diện Tomcat

<http://localhost:8080>

- Cấu hình user Apache tomcat

Truy cập đường dẫn

# 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"/>

e

Configuration and Deployment

- Tomcat chạy với setup mặc định, ta có thể custom lại tomcat theo các thành phần của nó, như:

+ Services, Servers, Engine, Connectors, Realm, and Valve can be configured

- Tất cả config của tomcat nằm trong “conf” directory

- Cấu trúc thư mục config

drwxr-xr-x. 3 root root 23 Sep 1 01:03 Catalina

-rw-r--r--. 1 root root 13451 Aug 11 13:23 catalina.policy

-rw-r--r--. 1 root root 6633 Aug 11 13:23 catalina.properties

-rw-r--r--. 1 root root 1428 Aug 11 13:23 context.xml

-rw-r--r--. 1 root root 3352 Aug 11 13:23 logging.properties

-rw-r--r--. 1 root root 6757 Aug 11 13:23 server.xml

-rw-r--r--. 1 root root 2181 Sep 1 01:57 tomcat-users.xml

-rw-r--r--. 1 root root 172452 Aug 11 13:23 web.xml

- Ý nghĩa các file cấu hình

+ catalina.policy: file mô tả chính sách bảo mật, quyền truy cập trên Tomcat 7. Nó thi hành chính sách bảo mật cho phép JVM chạy web app

+ catalina.properties: chứa các định nghĩa chia sẻ trên các share, shared loader, JARs sẽ được scant khi server startup.

+ server.xml: 1 trong nhưng file cấu hình quan trọng nhất của tomcat, nó chứa các thông tin quan trọng như ip, port, virtual host, các đường dẫn, v.v.

+ tomcat-users.xml: file này sử dụng cho việc xác thực, ủy quyền và định nghĩa các luật. Nó sử dụng để thực thi 1 database về user/passwd/role cho việc xác thực, quản lý các container-managed security. Để thêm, xóa user, gán quyền / bỏ quyền user, edit this file.

+ logging.properties: như tên, file định nghĩa các thuộc tính log của thực thi tomcat.

+ web.xml: định nghĩa các giá trị mặc định cho tất cả web app khi thực thi tomcat được khởi chạy. Nếu giá trị của web app đã được set chi tiết thì các giá trị đó sẽ override lại giá trị mặc định.

+ context.xml: Nội dung của file sẽ được load cho tất cả app. Cấu hình các tham số như session, comet connection tracking .. sẽ được thực hiện tại đây.

Configuration of Tomcat 7

- Cơ chế hoạt động web app:

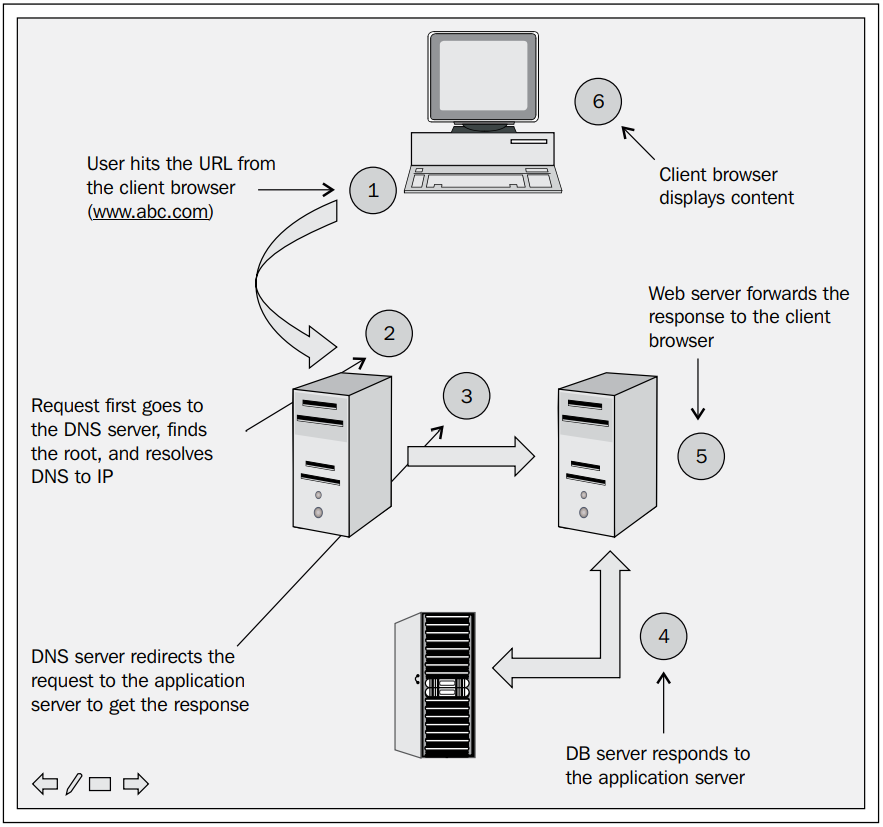
Bước 1: user nhập URL tại web browser, browser sẽ contact DNS server

Bước 2: DNS server sẽ kết nối đến ISP, yêu cầu thông tin về URL => map tới web server

Bước 3: Khi web server sẽ chập nhận request từ client browser, nó sẽ redirect tới database server

Bước 4: Database server sẽ trả lại giá trị query cho webserver.

Bước 5: Web server sẽ trả lại nội dụng tới response tới client browser và cuối cùng browser sẽ hiện thị lại nội dung cho user.



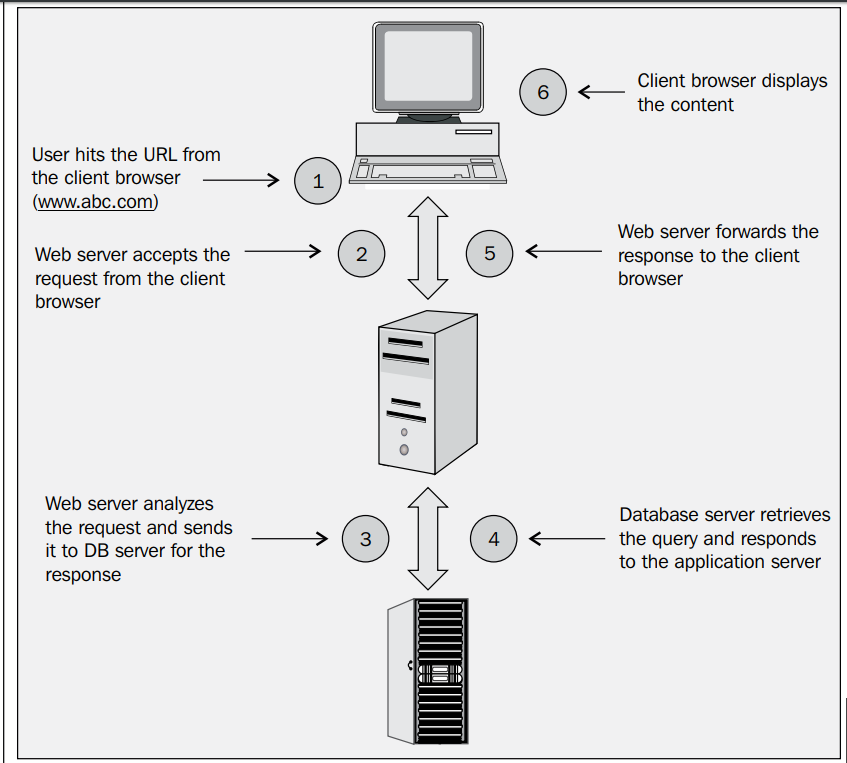
DataSource configuration

- Trong bất kỳ web app nào, csdl đòng vai trò rất quan trọng trong web, nó được coi là backbone trong các ứng dụng doanh nghiệp. Để app hoạt động chính xác, việc cấu hình data source chính xác trở nên cần thiết.

- Mô tả quả trình web app phân tích request và trả lại cho client

Bước 1: khi user hit URL, request sẽ được gửi tới web server.

Bước 2: khi server chấp nhận request từ client browser, nó sẽ phân tích request dự trên truy vấn, nó sẽ gửi query tới db và nhận lại kết quả.



Cấu hình thực trong Tomcat.

Một số thuận ngữ về database trong Tomcat.

- JDBC: Java Database Connectivity (JDBC) là công nghệ truy cập dựa trên Java, sử thông qua API, cho phép client truy cập vào server database. Nó định hướng cơ sở dữ liệu quan hệ, cung câp cách để query và update database.

- JNDI: Java Naming and Directory Interface (JNDI) service là API cho nền tảng Java, cung cấp việc đặt tên và các chức năng về directory cho việc viết app sử dụng ngôn ngữ Java.

- DataSource: là đối tượng java sử dụng cho việc truy cập cơ sở dữ liệu quan hệ thông qua JDBC API. Nó làm việc tốt khi tích hợp JNDI và sau khi datasource object được đăng ký với JNDI naming service. Đối tượng có thể truy cập bởi chính ứng dụng và kết nối tới database.

- Các giá trị sau đây sẽ cần thiết cho bất kỳ database server connect tới Tomcat 7 với database và cũng như các điều kiện tiên quyết cho datasource config:

+ IP address

+ Port number

+ JNDI name

+ Database user ID/password

Note:

Database servers in production The applications which are hosted on the Internet, their web servers are always configured in the Demilitarized Zone (DMZ). For more information on the DMZ zone, please refer to http:// en.wikipedia.org/wiki/DMZ\_(computing). Database servers are placed in an internal network. In this situation, the firewall port needs to be open between the web servers and the database server for communication.

- Database Connection Pool (DBCP) config được lưu tại TOMCAT\_HOME hoặc CATALINA\_HOME/lib/tomcat-dbcp.jar. file JAR sẽ chịu trách nhiệm cho connection pooling.

- ví trí lưu trữ “tomcat-dbcp.jar” tại “../tom-cat../lib/\*”

Note:

+ Database Connection pool

+ Common DBCP properties

+ Configuration of the database server details in server.xml

+ The database specific JAR or JDBC driver needs to be placed in the lib directory

+ The JNDI should be defined in the application web.xml file

+ Application code should have proper JNDI configuration defined

**Tomcat Manager configuration**

**-** Tomcat manager là công cụ mạnh mẽ cho người quản trị Tomcat.

- Trong môi trường thực tế, không thể luôn luôn làm việc tại data center all time. Trong 1 số thời điểm, ta có thể truy cập Tomcat từ xa để giải quyết lỗi, tại thời điểm đó, tomcat manager trở nên rất hữu ích cho việc giải quyết lỗi.

- Các tính năng của Tomcat manager:

+ Triển khai 1 ứng dụng từ xa.

+ Loại bỏ các session không làm việc

+ Loại bỏ 1 ứng dụng mà không phải restart tomcat.

+ Phân tích bộ nhớ

+ Trạng thái JVM

+ Trạng thái server status

Truy cập tomcat manager

- Mặc định Tomcat manager không được thiết lập, để sử dụng nó, cấu hình “tomcat-users.xml” trong thư mục “conf”

- Cấu hình “tomcat-user”

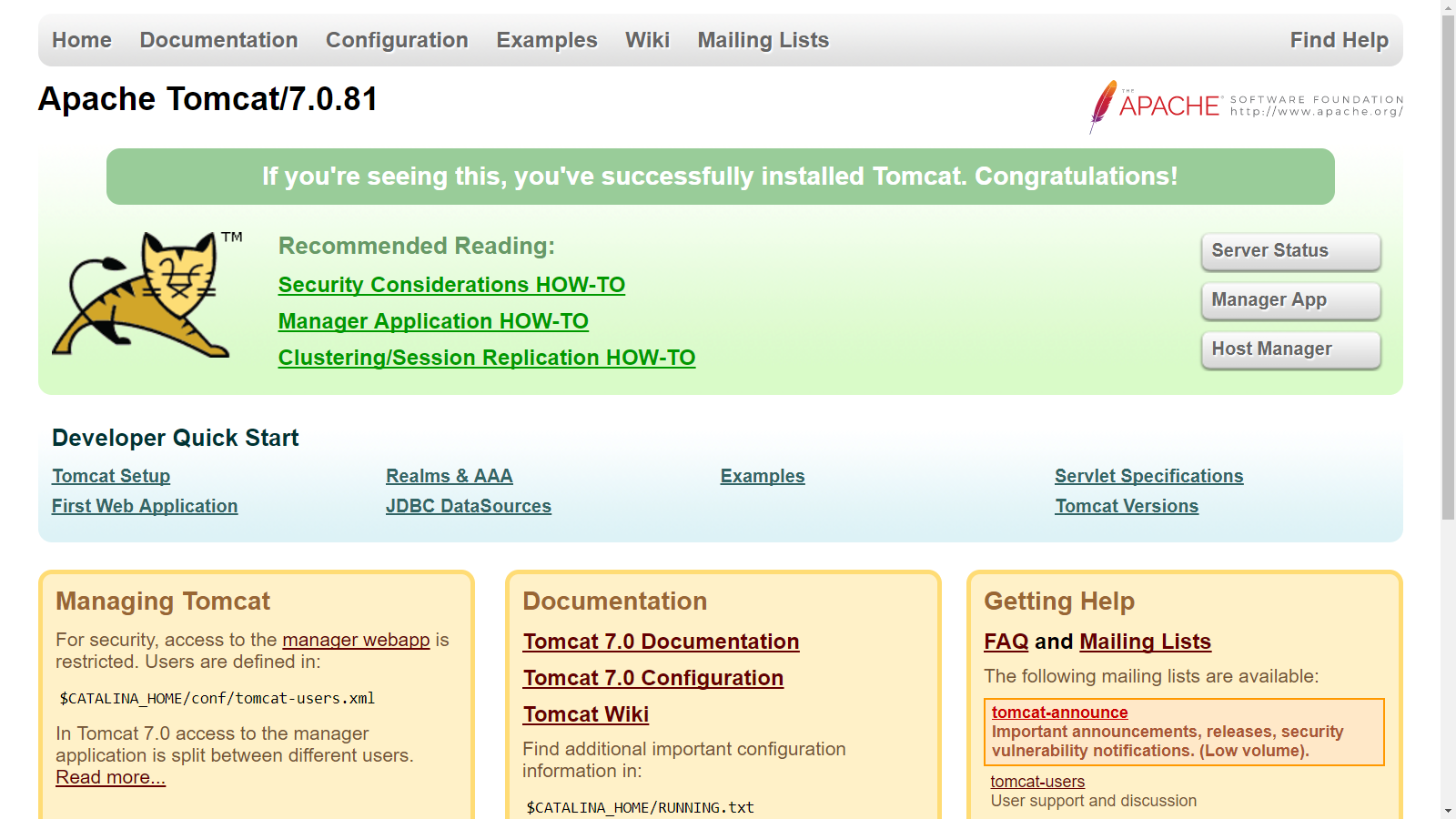
# vim tom-cat../conf/tomcat-users

<role rolename="admin-gui"/>

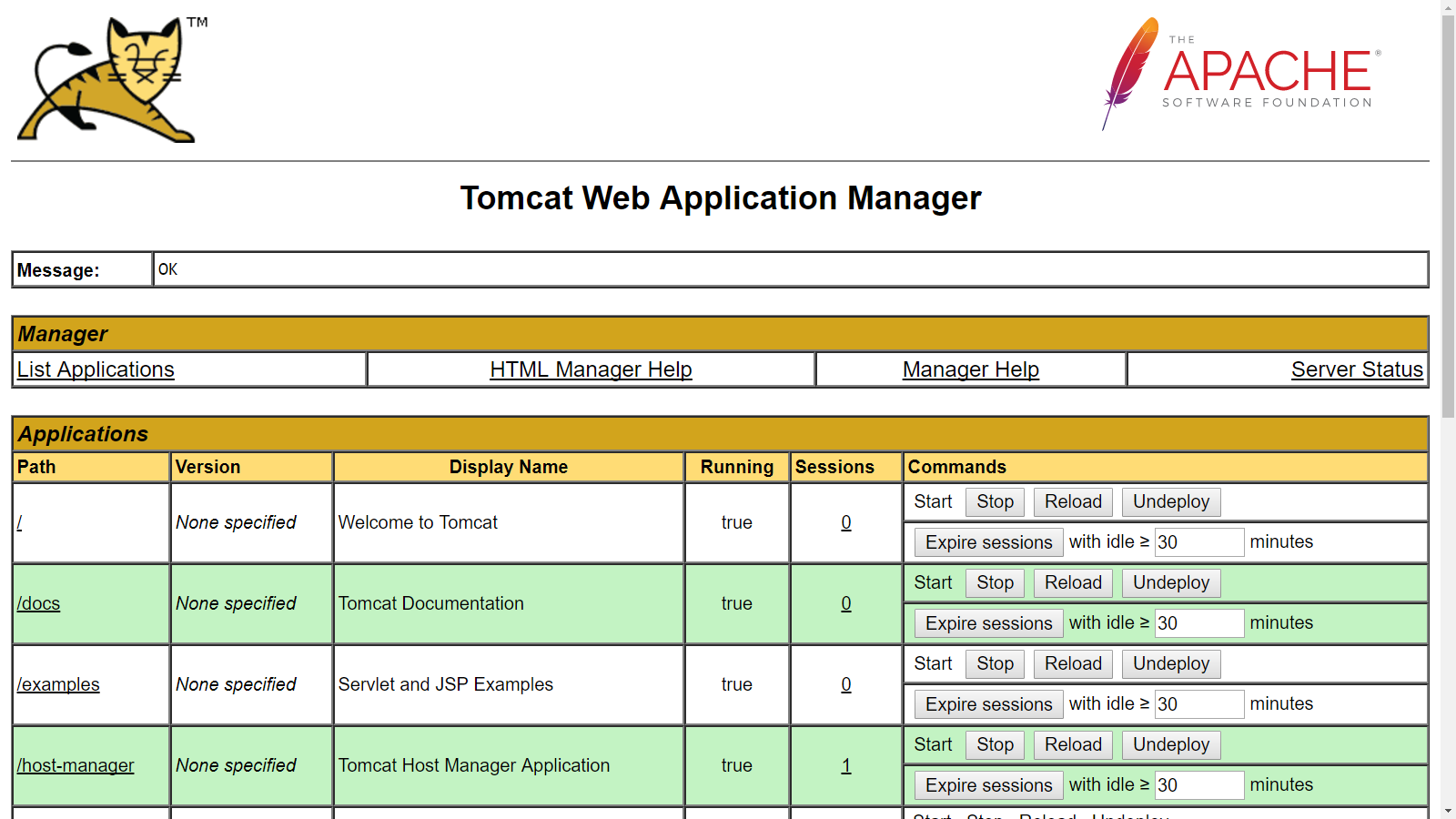
<role rolename="manager-gui"/>

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

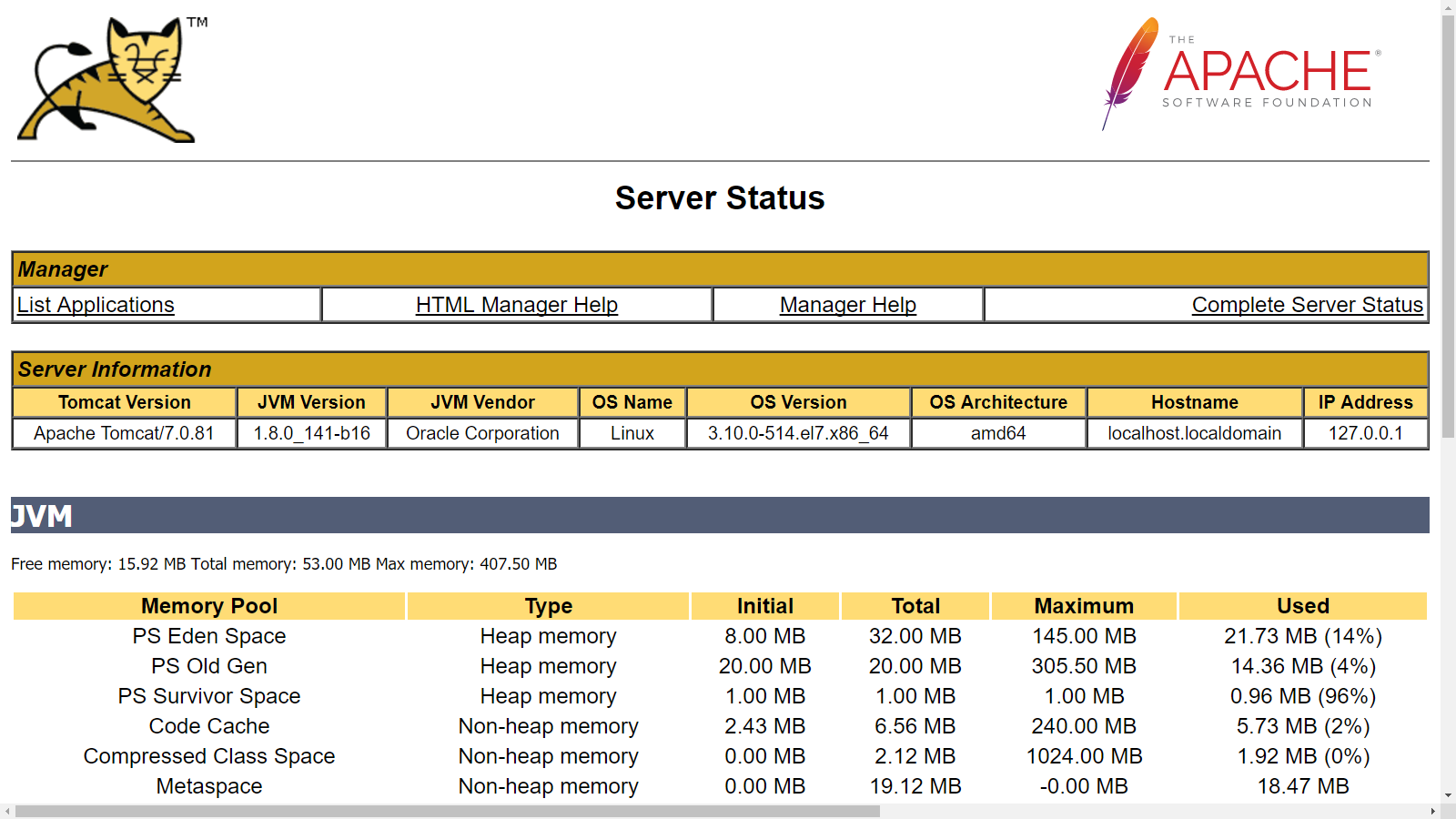
Giao diện chính Tomcat



- Quản trị web



- Server status:



- Server status bao gồm:

- JVM status

+ Max memory

+ Total memory

+ Free memory

- Connection of AJP port 8009

+ Connection state

+ Data sent

+ Data received

+ Client

+ Virtual host

- Connection on HTTP port 8080

+ Connection state

+ Data sent

+ Data received

+ Client

+ Virtual host

- Server information

+ Tomcat version

+ OS version

+ JVM version

+ System architecture

Context path

- context path là 1 thành phần quan trọng trong web app. Nó cũng được sử dụng cho virtual host. Virtual host có thể được định như 1 phương thức mà thông qua đó, bạn có thể host nhiều domain names trên cùng 1 webserver hoặc 1 ip duy nhất.

- context path cũng được sử dụng để định nghĩa URL mapping cho .war files.

- Bằng cách định nghĩa context path, ta có thể giảm tải việc load trên server. Khi server nhận được request với URL. Nó sẽ check server.xml hoặc context path được định nghĩa từ URL. Nếu nó tìm thấy, URL sẽ được sử dụng hoặc nó sẽ tìm kiếm tất cả war file đã được deploy. Do vậy, làm giảm tải CPU.

- Bên cạnh đó, nó còn cho phép tư do trong việc tùy chỉnh logging, appBase, DB connec ..

- Giả sử 1 công ty cần deploy 100 Tomcat server với 1 single app. Có thể deploy chung 1 source thông qua NAS share.

Sử dụng context path:

- Context path trong Tomcat có thể được sử dụng thông qua 2 cách:

+ GUI sử dụng Tomcat Web Application Manager

+ Cli conf trong server.xml

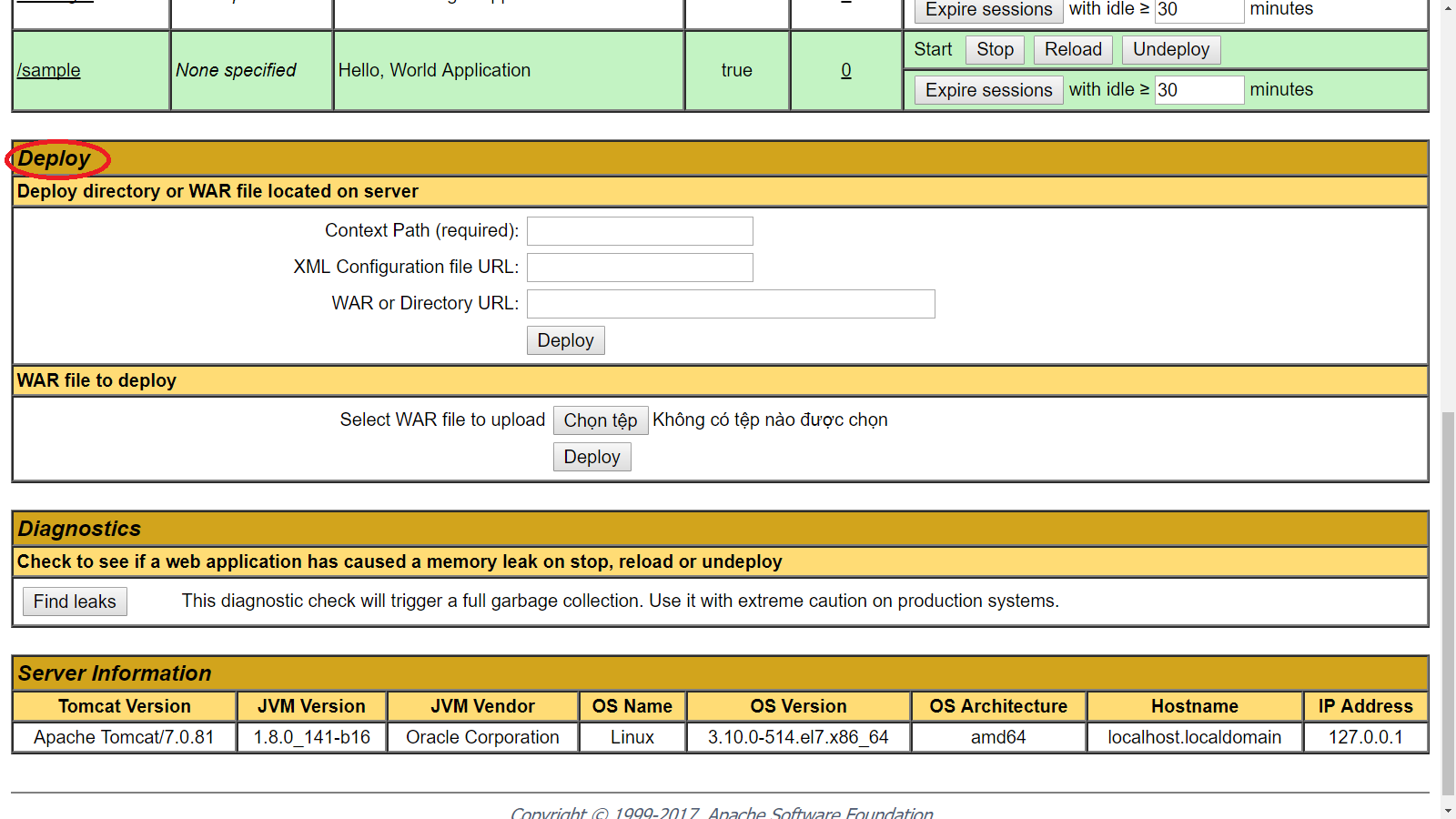
Sử dụng GUI Tomcat Web Application Manager

- Có thể sử dụng context path trên Tomcat Manager

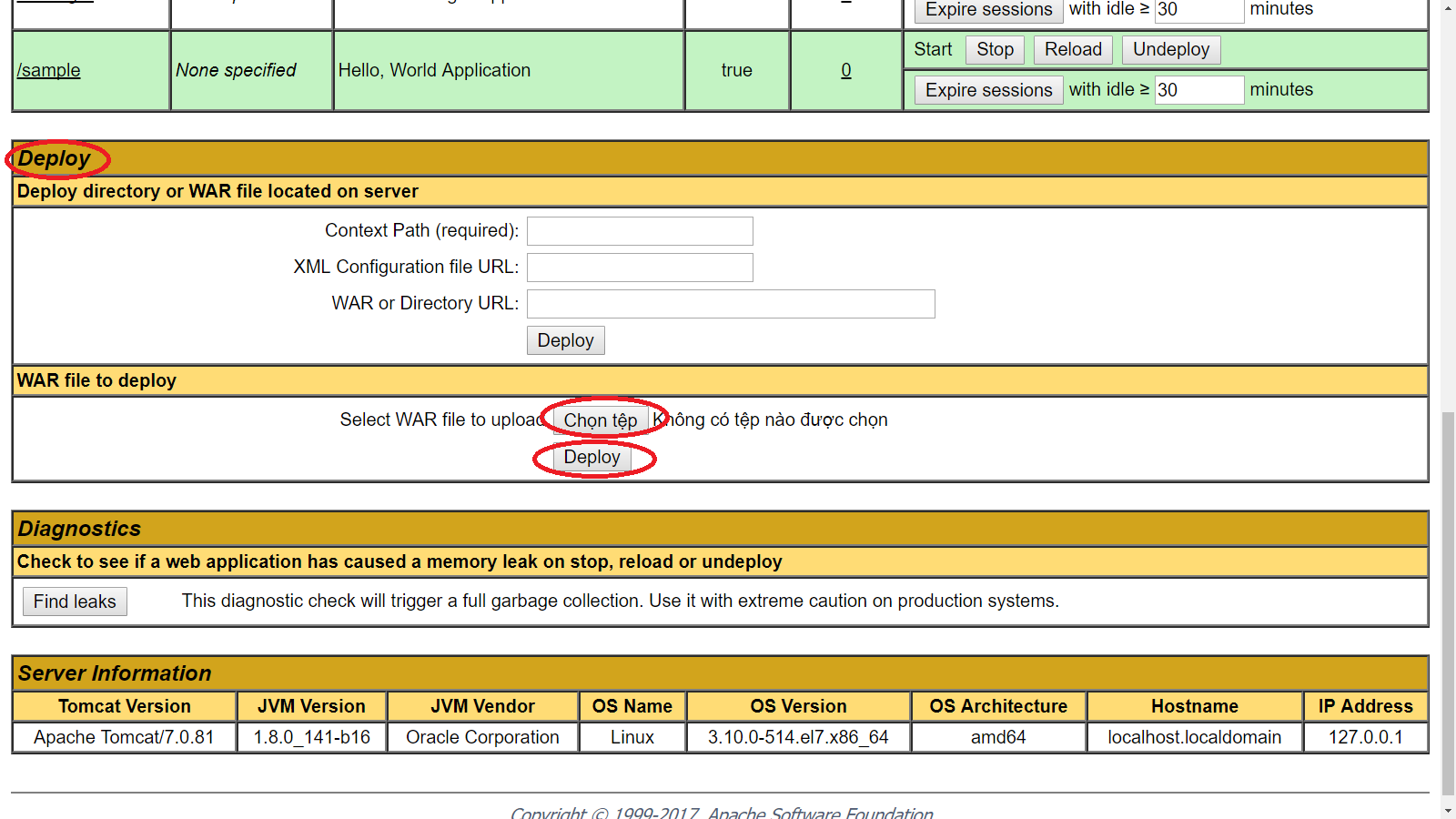
Bước 1: Truy cập <http://localhost:8080>

Bước 2: Click “Manager App”





Bước 3: Tại Deploy path, chọn Deploy tab, chọn Browse lựa chọn WAR file. Lựa chọn Deploy.



Deploy thông qua cli tại server.xml

Deployment in Tomcat 7

* Cơ bản, triển khai là sử dụng WAR files để khởi tạo web. Nói 1 cách khác, chúng ta có thể giải nén war file trong Tomcat wepapps directory

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 7
* 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.

+ 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)

VD:

<servlet>

<servlet-name>classB</servlet-name>

<servlet-class>class.classB</servlet-class>

</servlet>

Archive Files

- Trong môi trường triển khai, bạn thường nhận được các đơn vị triển khai từ dever. Các archi file có thể là 1 file chứa tất cả thông tin về app, module class, static file, direc, deployment descriptor. Archive file thường được tảo bằng cách sử dụng JAR utility or Ant JAR tool.

- Các đơn vị triển khai là nhưng package sử dụng JAR utility có 1 số đuôi mở rộng đặc biệt để phân biệt từng loại:

+ 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

- Thư mục giải nén chứa tất cả các file, direc giống như JAR archive. Tuy nhiên các file và direc được giải nén ra và không được đóng gói thành 1 singel archive với JAR utility.

- Đơn vị triển khải được sử dụng như 1 thư mục giải nén 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.

Note

It's not possible to edit deployment descriptor values in the console for deployments from the archive files or .war files.

Deployment operations

- Các phương thức được hỗ trợ bởi toot Tomcat.

+ 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, phục vụ client. Trong thời gian redeploy, web app sẽ ko có sẵn để phục vụ client. (trong trạng thái stop)

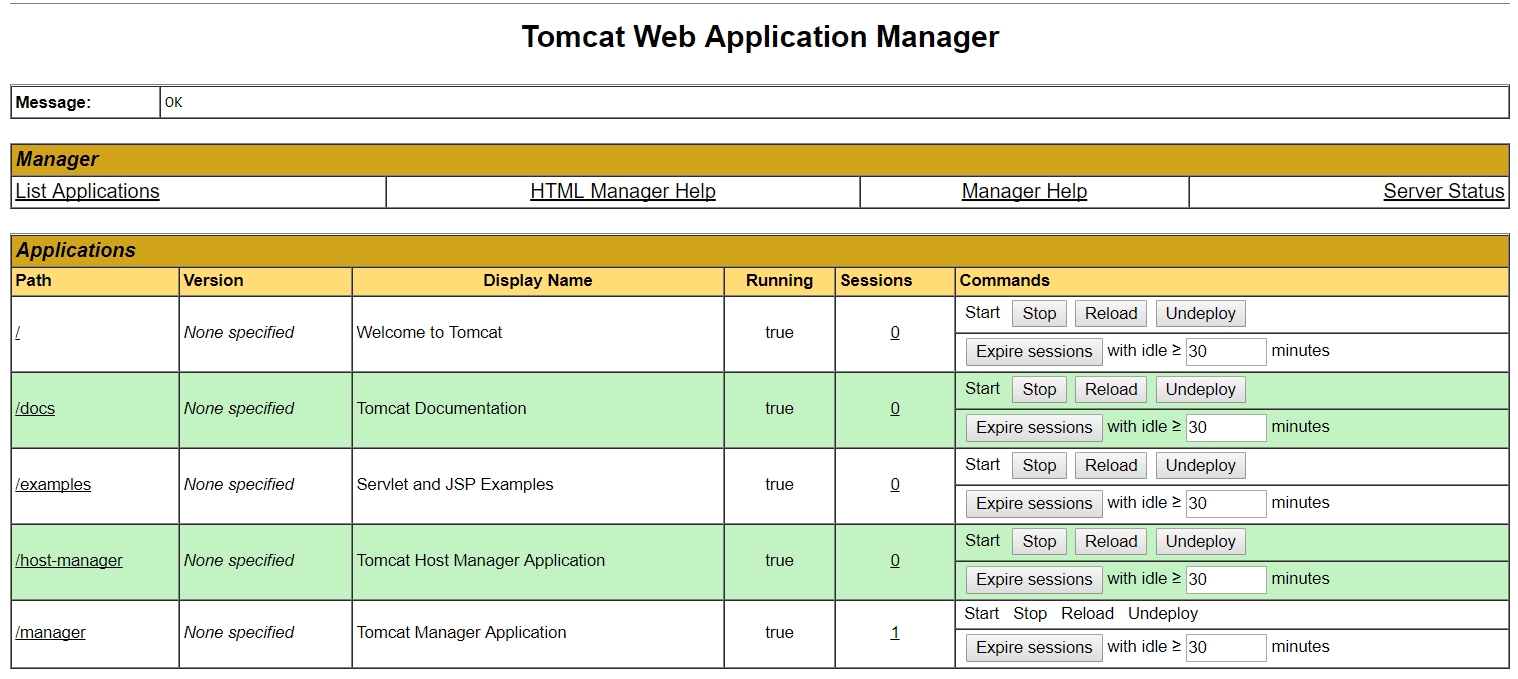
Note:

An application becomes unavailable to clients during redeployment. The Tomcat 7 server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time. For this reason, redeployment is not recommended for use in a production environment.

+ Stop: uload các class, các file, khiến web app ko có sẵn để phục vụ client. Nhưng web app này vẫn được target bởi server.

+ Start: nó sẽ reload lại các class, khiến web app có sẵn để 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 target server.



Các trạng thái triển khai:

- Các trạng thái triển khai định nghĩa rõ ràng chác mà các file deploy được load, target bởi server. Tomcat cung cấp 3 lựa chọn cho vấn đề này:

+ Stage mode

+ Nostage mode

+ External\_stage mode

- Bằng giải thích 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. |

Ways of application deployment in Tomcat 7- Có rất nhiều cách để triển khai app trên tomcat 7. Nhưng có thể tổng kết lại thành 5 loại chính:

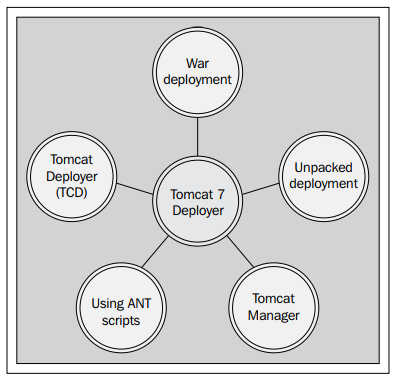
+ War deployment: Có thể triển khai bằng cách đưa War file vào CATALINA\_BASE direc và reset tomcat để view app. Đây là cách được sử dụng phổ biến trong môi trường triển khai

+ 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.



Performance Tuning

* Sử dụng cho việc nâng cao hiệu năng Tomcat, tránh downtime
* Các phần chính:

+ Memory related issues

+ JVM parameter optimization

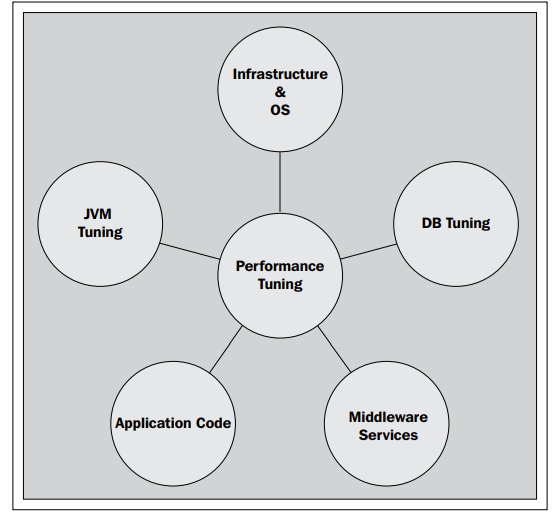
+ OS level optimization to improve the performance

Performance tuning for Tomcat 7

* Hiệu chỉnh hiệu suất có vai trò quan trọng để thực thi app, tránh hiện tượng downtime
* Việc hiệu chỉnh cũng nâng cao hiệu năng Tomcat trong khai đang chạy app. Tùy chình tomcat có sự khác nhau giữa các app vì mỗi app đều có nhưng yêu cầu riêng

Why we need performance tuning?

* Lý do phải tùy chỉnh, mỗi web app đều có nhưng yêu cầu riêng. Như có web app yêu cầu sử dụng nhiều memory trong khi app khác lại yêu cầu ít ram hơn nhưng lại đòi hỏi high GC pause. Chính vì vậy, tomcat admin cần hiệu chỉnh tomcat dựa trên yêu cầu app.
* Việc tùy chình JVM vẫn không đủ. Để nâng cao hiệu năng cần quan tâm đến rất nhiều vấn đề như database config, os setting, phần cứng.



* Có nhiều khía cạnh khác nhau để nâng cao hiệu năng:

+ Application Code: Nếu app code được thiết kế không chính xác, nó có thể dẫn đến vấn đề về hiệu năng. VD tạo nhiều db connect nhưng không kết thúc phiên, nó sẽ khiến app chậm lại

+ Database Tuning: đấy là 1 phần trong quan trọng trong việc tùy chỉnh tomcat. Có rất nhiều vấn đề có thể khiến database chạy chậm => dẫn đến ảnh hưởng hiệu năng tomcat.

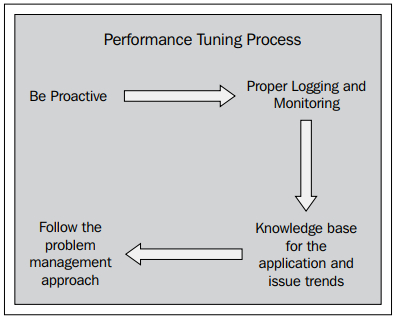
+ JVM Tuning: Mỗi app có yêu cầu về memory riêng, nếu app đòi hỏi nhiều memory mà lại cấp thiếu, nó sẽ dẫn đến vấn đề OOM (out of memory), gây ra vấn đề về hiệu năng.

+ Middleware Services: Vấn đề trong việc sử dụng giao diện mở rộng (API). Vd, app mobi yêu được kết nối tới server mà ta chỉ cho phép server làm việc với web app => app mobi không thể kết nối tới server.

+ Infrastructure and OS: Cơ sở hạ tầng cũng là 1 lý do chính trong vấn đề về hiệu năng. Vd: băng thông không đủ, băng thông internet, ….

How to start performance tuning:

* Việc hiệu chỉnh các app không có các bước rõ ràng nhưng có những quy tắc chính để nâng cao hiệu năng.
* Hình mô tả việc nâng cao hiệu năng:



+ Be Proactive (chủ động): nghĩ về các vấn đề có thể xảy ra với app trong tương lại, sau đó chuẩn bị các giải pháp để tránh gặp vấn đề đó. Sử dụng các tool có sẵn để tìm ra các vấn đề có thể gặp phải.

Note:

If you are proactively checking the application and avoid bottlenecks for the application, then around 70 percent of the issues are solved in the production environment. This approach will give your customers great satisfaction when using the application.

+ Proper Logging and Monitoring: Đọc log và giám sát hiệu năng. Việc bật tính năng sinh log sẽ rất hữu ích trong việc lần vết các vần đề tiềm tàng mà nó có thể xảy ra cho app. Đồng thời luôn giám sát tình trạng hệ thống (ram, cpu,.. ).

+ Knowledge base for the application and issue trends: có hiểu biết cơ bản về app và các vấn đề thường gặp phải để giúp app luôn chạy 24/7. Đồng thời cần có tài liệu chi tiết để giúp người dùng tránh lỗi thường gặp.

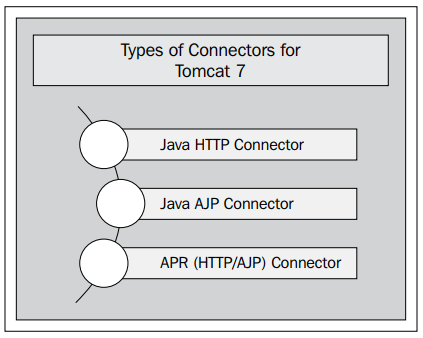
+ Follow the problem management approach: Sử dụng cơ chế Root Cause Analysis (RCA) đối với tất cả các vấn đề. Nó sẽ giúp cho việc phòng chống các lỗi giống nhau xảy ra trong tương lai.

Tomcat components tuning:

* Trong tomcat, có nhiều các cấu hình để nâng cao hiệu năng server, thread tuning, port customization, JVM tuning.

Types of connectors for Tomcat 7:

* Kết nối có thể được định nghĩa như các điểm giao, nó sẽ quyết định chập nhận request và đáp ứng, trả lại kết quả. Có 3 loại connector được sử dụng trong tomcat 7. Các connectors này được sử dụng tùy theo yêu cầu của app.



Java HTTP Connector:

* Java HTTP Connecter dựa trên HTTP Protocal, nó chỉ hỗ trợ HTTP/1.1 protocol. Nó cho phép Tomcat server hoạt động như 1 standalone web server và cũng cho phép các tình năng host JSP/Servlet.

Note:

For more information on the HTTP Connector, please visit   
http:// tomcat.apache.org/tomcat-7.0-doc/config/http.html.

Java AJP Connector

- Java AJP Connector dựa trên giao thức AJP (Apache JServ Protocol) và giao tiếp giữa web server thông qua AJP. Connector này chủ yếu sử dụng khi bạn không muốn để lộ your Java servlet container tới Internet (Sử dụng khác frontend server) và cũng hỗ trợ SSL khi tomcat không hỗ trợ. Một số ví dụ như AJP mod\_jk, mod\_proxy, …

Note:

For more information on AJP Connectors, please visit the URL

http://tomcat.apache.org/tomcat-7.0-doc/config/ajp.html.

APR (AJP/HTTP) Connector:

- Apache Portable Runtime (APR) rất hữu ích cho sự mở rộng, nâng cao hiệu năng, và tăng khả năng tương tác giữ các webserver khác nhau.

- Nó cung cấp các tính năng mở rộng như Open SSL, Shared memory, Unix socket, .. Nó cũng cho phép nhúng Java như công nghệ web server hơn là 1 công nghệ phụ trợ.

Thread optimization for Tomcat 7

- Hiệu chỉnh Thread là 1 trong những vấn đế trong hiệu chỉnh Tomcat. Trong nhiều trường hợp, có nhưng app đặc biệt có thể chạy tốt trên nhiều nền tảng khác nhau, nhưng khi chúng ta thực thi trên cùng 1 ứng dụng thì hiệu năng lại giảm. Lý do có thể bắt đầu khi chúng ta hiệu chỉnh thread không hợp lý, dẫn đến hiệu năng máy chủ giảm xuống. Các thành phần khác nhau trong thread tuning thread pools.

- Thread pool có thể định nghĩa như khả năng của web server, nó quyết đinh khả năng chấp nhận số lượng connection và request cho việc xử lý.

- Có thể config 2 loại thread pool, Shared pool và Dedicated pool. Các config cần thực hiện tại “server.xml” tại “TOMCAT\_HOME/conf/server.xml”.

Shared thread pool (shared executor)

- Nó có thể định nghĩa thread pool như việc chia sẻ giữ các kết nối.

VD:

- khi bạn có 4 conf connector, sau đó bạn share các thread pool cho tất cả connector

Bước 1: Edit server.xml, thêm định nghĩa share thread pool trong services section.

<Executor name="tomcatThreadPool"

namePrefix="catalina-exec-"

maxThreads="150"

minSpareThreads="4"/>

Bước 2: Khi bạn đã định nghĩa shared thread pool, gọi tham chiếu thread setting trong Connector definition cho services section của server.xml theo:

<Connector executor="tomcatThreadPool"

port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

**Dedicated thread pool**- Có thể định nghĩa thread pool mà sẽ chỉ sử dụng cho 1 connecter definition. VD, nếu app cần high load, thì việc sử dụng 1 luồng trong thread pool tốt hơn cho connector, và cũng kiểm thực thi Tomcat chạy tốt hơn.

VD: cấu hình 1 dedicated thread pool. Giá trị đặc biệt định nghĩa giá trị cho dedicated connection pool:

Bước 1: Edit server.xml, định nghĩa cấu hình dedicated thread pool trong Connector section.

<Connector port="8443" protocol="HTTP/1.1"

SSLEnabled="true"

maxThreads="150"

scheme="https"

secure="true"

clientAuth="false" sslProtocol="TLS" />

Shared thread pool versus dedicated thread pool- So sánh thread pool và dedicated thread pool

|  |  |  |
| --- | --- | --- |
| Features | Shared thread pool | Dedicated thread pool |
| Number of users | Less | High |
| Environment | Development | Production |
| Performance | Low | Good |

maxThreads- maxThreads có thể định nghĩa như số lượng nhiều nhất các request sẽ được server accept. Mặc định, Tomcat 7 sử dụng maxThreads=150. Trong mỗi trường thực thi, ta cần hiệu chỉnh maxThreads dựa trên hiệu năng của server.

- Trong môi trường thực tế cần maxThreads=300 cho app, nếu giá trị maxThreads cấu hình không chính xác, nó sẽ giảm hiệu năng của server.

- Cách tốt nhất để giảm sát cấu hình maxThread có được cấu hình chính xác không là theo dõi hiệu năng CPU trên server. Nếu CPU chạy ở mức cao, giảm giá trị thread xuống. Nó có nghĩa là tại tình huống này, thread tomcat đã giảm hiệu năng server. Nếu giá trị CPU chạy ở mức bình thường, có thể tăng giá trị thread để tăng hiệu năng phục vụ người dùng.

Note:

While setting the maxThreads, you have to consider other resources as well, such as the database connection, network bandwidth, and so on.

maxKeepAlive- Đinh nghĩa số lượng connection TCP xảy ra đồng thời trong Tomcat 7. Mặc định, maxKeepAlive set = 1, điều đó có nghĩa nó được tắt

- Nếu maxKeepAlive = 1

+ SSL sẽ không chấm dứt trong Tomcat

+ Kỹ thuật cân bằng tải được sử dụng

+ Có thể nhiều người cùng sử dụng

- Nếu maxKeepAlive > 1 thì:

+ SSL bị kết thúc Tomcat.

+ Số người dùng đồng thời ít hơn.

JVM tuning

- Trước khi bắt đầu với JVM tunning, ta cần note lại một số nhà cung cấp hàng đầu cho JVM. Dựa tển yêu cầu app, ta sẽ lựa chọn JDK theo các nhà cung cấp đó.

Note:

Sun JDK is widely used in the IT industries.

Why do we need to tune the JDK for Tomcat?- Tomcat 7 bắt đầu với heap size = 256 MB, app ngày nay sẽ cần lượng lớn memory để chạy. Để chạy app, ta cần hiệu chỉnh giá trị JVM cho tomcat 7.

- Hiển thị giá trị tomcat và memory sử dụng

# ps -ef | grep java

root 2205 1 80 23:23 pts/0 00:00:02 /usr/bin/java -Djava.util.logging.config.file=/root/apache-tomcat-7.0.81/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djdk.tls.ephemeralDHKeySize=2048 -Djava.endorsed.dirs=/root/apache-tomcat-7.0.81/endorsed -classpath /root/apache-tomcat-7.0.81/bin/bootstrap.jar:/root/apache-tomcat-7.0.81/bin/tomcat-juli.jar -Dcatalina.base=/root/apache-tomcat-7.0.81 -Dcatalina.home=/root/apache-tomcat-7.0.81 -Djava.io.tmpdir=/root/apache-tomcat-7.0.81/temp org.apache.catalina.startup.Bootstrap start

- Từ kết quả trên ta có, ta để ý trường pid

JMAP (Memory Map)

- JMAP hiện ra toàn bộ các thành phần trong Java Virtual Memory. Nó là công cụ tốt cho người quản trị kiểm tra trạng thái về shared memory.

- Bẳng mô tả các giá trị option

|  |  |
| --- | --- |
| Options | Description |
| -dump | Dumps the Java heap in hprof binary format |
| -finalizer info | Prints information on objects awaiting finalization |
| -heap | Prints a heap summary |
| -histo | Prints a histogram of the heap |
| -permstat | Prints class loader-wise statistics of permanent generation of the Java heap |

Syntax for jmap

- Cú phản để sử dụng jmap command là ./jmap -heap <process id> khi process id chính là tiến trình java ta muốn kiểm tra memory

- Như phần trước, ta đã biết PID, sử dụng cho cmd

# ./bin/jmap -heap 2205

Attaching to process ID 2205, please wait...

Debugger attached successfully.

Server compiler detected.

JVM version is 25.141-b16

using thread-local object allocation.

Parallel GC with 2 thread(s)

Heap Configuration:

MinHeapFreeRatio = 0

MaxHeapFreeRatio = 100

MaxHeapSize = 480247808 (458.0MB)

NewSize = 10485760 (10.0MB)

MaxNewSize = 159907840 (152.5MB)

OldSize = 20971520 (20.0MB)

NewRatio = 2

SurvivorRatio = 8

MetaspaceSize = 21807104 (20.796875MB)

CompressedClassSpaceSize = 1073741824 (1024.0MB)

MaxMetaspaceSize = 17592186044415 MB

G1HeapRegionSize = 0 (0.0MB)

Heap Usage:

Exception in thread "main" java.lang.reflect.InvocationTargetException

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at sun.tools.jmap.JMap.runTool(JMap.java:201)

at sun.tools.jmap.JMap.main(JMap.java:130)

Caused by: java.lang.RuntimeException: unknown CollectedHeap type : class sun.jvm.hotspot.gc\_interface.CollectedHeap

at sun.jvm.hotspot.tools.HeapSummary.run(HeapSummary.java:144)

at sun.jvm.hotspot.tools.Tool.startInternal(Tool.java:260)

at sun.jvm.hotspot.tools.Tool.start(Tool.java:223)

at sun.jvm.hotspot.tools.Tool.execute(Tool.java:118)

at sun.jvm.hotspot.tools.HeapSummary.main(HeapSummary.java:49)

Kết quả trả lại bao gồm

+ The heap configuration for the application (the highlighted code describes

the heap configuration)

+ The heap utilization for each JVM component

+ The algorithm used for the garbage collection

How to increase the heap size in Tomcat 7

- Để tăng heap size trong Tomcat 7, ta cần thêm JAVA\_OPTS parameter trong catalina.sh trong TOMCAT\_HOME/bin.

- Tăng max heapsize từ 256 tới 512mb thêm vào dòng. Nếu chưa thấy dòng định nghĩa giống thì thêm tại dòng đầu tiên của script.

# vim TOMCAT\_HOME/bin/catalina.sh

….

JAVA\_OPTS="-Xms128m -Xmx512m -XX:MaxPermSize=256m"

Note:

Every change in configuration for the JVM parameter will be in effect after restarting the Tomcat server. You can verify the change done in the JVM parameter by running the jmap command.

- Ngoài ra ta có thể định nghĩa các giá trị trong catalina.sh:

+ JRE\_HOME, JAVA\_OPTS, JAVA\_ENDORSED\_DIRS, JPDA\_TRANSPORT, JPDA\_ADDRESS, JPDA\_SUSPEND, JPDA\_OPTS, LOGGING\_CONFIG, LOGGING\_MANAGER

Garbage collection

- Garbage nghĩa là rác sinh ra trong quá trình thực thi. Nó chính là nhưng object lưu tại JVM memory nhưng không thuộc về bất kỳ program nào.

- Garbage collector là 1 thuật toán chạy định kỳ, nó sẽ thu thập trạng thái của các object đang hoạt động và không hoạt động trong memory, xóa các object không hoạt động để giải phóng memory.

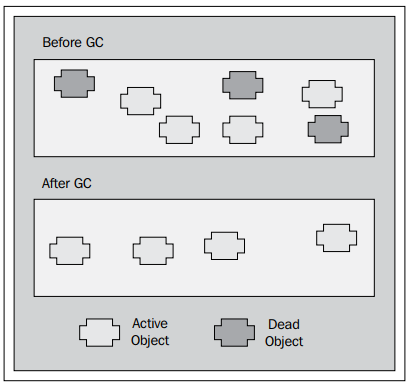
- Các sự kiện garbage collection:

+ Garbage collection sẽ chạy không chính xác đối với app cần bộ nhớ lớn

+ Garbage collection sẽ không cân nhắc 1 số đối tượng có vòng đời ngắn, một số khác lại có vòng đời dài và không thường xuyên được sử dụng => hiểu lầm đối tượng.

How garbage collection works

- Khi thuật toán GC được gọi, nó sẽ tập hợp lại tập hợp lại các đối tượng không được sử dụng trong bộ nhớ, sau đấy sẽ dọn dẹp bộ nhớ. Nó có thể được giải thích đối ngược lại việc quản lý bộ nhớ bằng tay.



- Nó sẽ xóa toàn bộ các đối tượng inactive từ memory và tất cả đối đượng active sẽ được giữa lại. Có 3 loại GC collector được sử dụng trong môi trường real time:

+ Serial collector

+ Parallel collector

+ Concurrent low pause collector

- Bảng mô tả các tính năng về Serial collector:

|  |  |
| --- | --- |
| Features | Serial collector |
| Process | Single thread is used for GC |
| GC pause | High |
| Threading | Single threaded |
| Application | Small application (data less than 100 MB) |
| Advantage | There is single thread communication |

- Bẳng mô tả các tính năng của parallel collector

|  |  |
| --- | --- |
| Features | Parallel collector |
| Process | Parallel thread does minor GC |
| GC pause | Less than Serial |
| Threading | Multithreaded |
| Application | Mid-large |
| Advantage | Used in applications when peak performance is needed |

- Bảng mô tả các tính năng của concurrent collector

|  |  |
| --- | --- |
| Features | Concurrent collector |
| Process | GC is done concurrently |
| GC pause | Short pause |
| Threading | Multithreaded |
| Application | Mid-large |
| Advantage | Used in applications when a response is needed |

Note:

Parallel and concurrent collector algorithms should not be used together

JVM options- Tùy chọn Java HotSpot option được phân loại thành standard và non-standard options.

Standard options

- Standard options được thừa nhận bởi Java HotSpot VM mentioned trong java app launcher launcher page cho mỗi OS.

VD:

# java -showversion

openjdk version "1.8.0\_141"

OpenJDK Runtime Environment (build 1.8.0\_141-b16)

OpenJDK 64-Bit Server VM (build 25.141-b16, mixed mode)

Usage: java [-options] class [args...]

(to execute a class)

or java [-options] -jar jarfile [args...]

(to execute a jar file)

where options include:

-d32 use a 32-bit data model if available

-d64 use a 64-bit data model if available

-server to select the "server" VM

The default VM is server,

because you are running on a server-class machine.

-cp <class search path of directories and zip/jar files>

-classpath <class search path of directories and zip/jar files>

A : separated list of directories, JAR archives,

and ZIP archives to search for class files.

-D<name>=<value>

set a system property

-verbose:[class|gc|jni]

enable verbose output

-version print product version and exit

-version:<value>

Warning: this feature is deprecated and will be removed

in a future release.

require the specified version to run

-showversion print product version and continue

-jre-restrict-search | -no-jre-restrict-search

Warning: this feature is deprecated and will be removed

in a future release.

include/exclude user private JREs in the version search

-? -help print this help message

-X print help on non-standard options

-ea[:<packagename>...|:<classname>]

-enableassertions[:<packagename>...|:<classname>]

enable assertions with specified granularity

-da[:<packagename>...|:<classname>]

-disableassertions[:<packagename>...|:<classname>]

disable assertions with specified granularity

-esa | -enablesystemassertions

enable system assertions

-dsa | -disablesystemassertions

disable system assertions

-agentlib:<libname>[=<options>]

load native agent library <libname>, e.g. -agentlib:hprof

see also, -agentlib:jdwp=help and -agentlib:hprof=help

-agentpath:<pathname>[=<options>]

load native agent library by full pathname

-javaagent:<jarpath>[=<options>]

load Java programming language agent, see java.lang.instrument

-splash:<imagepath>

show splash screen with specified image

See http://www.oracle.com/technetwork/java/javase/documentation/index.html for more details.

Non-standard options- Non-standard option được định nghĩa với -x hoặc -xx option trong JVM.

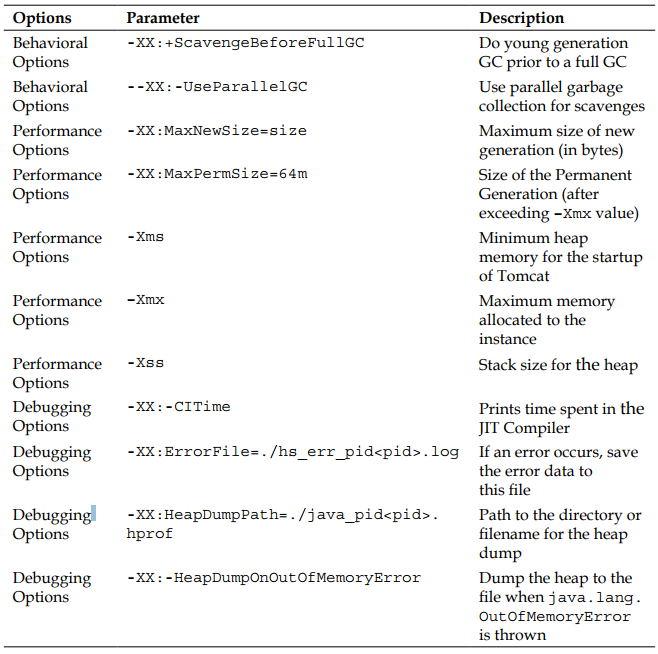
- Các option được chia ra làm 3 loại:

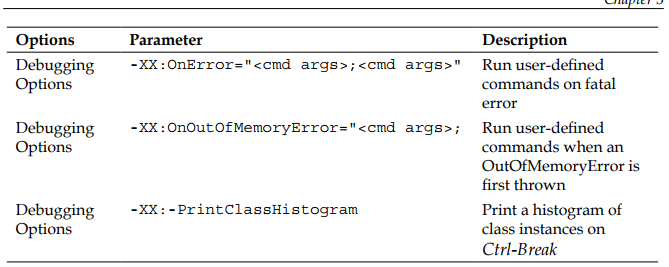
+ Behavioral options: Thay đổi hành vị của VM

+ Performance tuning options (Tùy chọn điều chỉnh hiệu suất): Nó kích hoạt hoạt động tối ưu hiệu năng của VM. Có 1 số option rất hưu dụng cho tunning server.

+ Debugging options: Hiển thị đầu ra và các thông tin về VM. Bên cạnh đó, nó cho phép lần vết logs (option rất hữu ích khi xử lý các sự cố)

- Bảng mô tả các tuy chọn thường sử dụng cho JVM





Parameters displayed in the logs for GC- Các giá trị hiển thi trong log GC.

- GC hiện thị output của garbage collection thông qua chuẩn stdout. Mỗi garbage collection sẽ hiện thị 5 trường

[%T %B->%A(%C), %D]

+ %T: This is "GC" when the garbage collection is a scavenge, and "Full GC:" is

performed, then scavenge collects live objects from the new generation only,

whereas a full garbage collection collects objects from all spaces in the

Java heap.

+ %B: It is the size of the Java heap used before the garbage collection, in KB.

+ %A: It is the size of the Java heap after the garbage collection, in KB.

+ %C: It is the current capacity of the entire Java heap, in KB.

+ %D: It is the duration of the collection in seconds.

SurvivorRatio- Định nghĩa tỷ lệ của eden tới survivor space size.

The default value is 8, meaning that eden is 8 times bigger than from and to, each. The syntax for the SurvivorRatio is -XX:SurvivorRatio=<size>.

VD:

+ Xmn / (SurvivorRatio + 2) = size of from and to, each ( Xmn / (SurvivorRatio + 2) ) \* SurvivorRatio = eden size

OS tuning

- Mỗi OS đều có những yêu cầu tiên quản để chạy tomcat 7 và hệ thống cần được hiệu chỉnh dựa trên yêu cầu app, nhưng mỗi OS lại có nhưng điểm giống nhau.

- OS là phần quan trọng trong việc nâng cao hiệu năng. Dựa trên phần cứng, hiệu năng sẽ được tăng lên hoặc giảm xuống.

VD:

+ Performance characteristics of the 64 bit versus 32 bit VM: (Điểm khác biệt giừa phiên bản 64 bit và 32 bit. Lợi ích của việc sử dụng phiên bản 64 bit VM là có thể sử dụng được nhiều memory hơn so vơi phiên bản 32 bit.

+ File size: Dựa trên yêu cầu app, dung lượng file size được set trong OS. Nếu app sử dụng nhiều nhiều giao dịch, file limit cần được mở rộng.

+ Ulimits: Dựa trên các phiên, user có thể tăng giới hạn.

+ Huge page size: Many applications send a huge page size causing the application to run slow. In this case, you can increase the page size based on the application needs.

VD: check the page size by running the following command:

[root@localhost bin]# cat /proc/meminfo

Tích hợp Tomcat vơi 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.

- Các phần sẽ bàn luận:

+ The Apache HTTP installation

+ The various modules of Apache

+ Integration of Apache with Tomcat 7

+ How IT industry environments are set up

User request flow (web/application level)

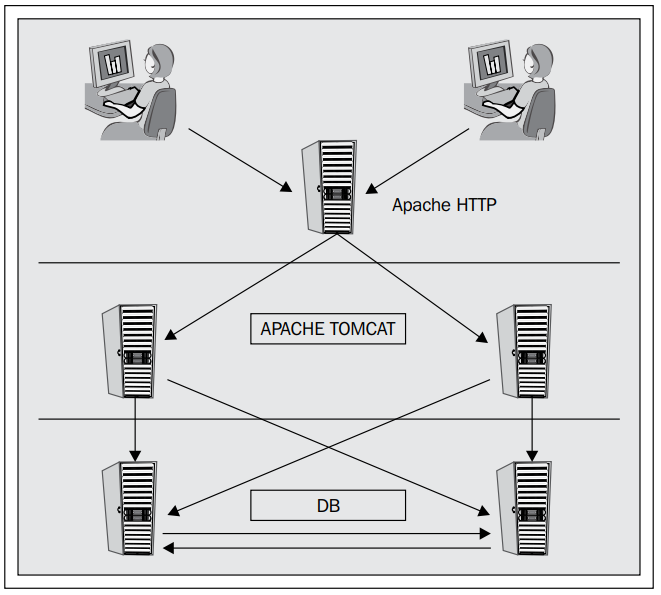
- Trước khi bàn luận về cài đặt Apache, ta sẽ nói về vấn đề tổng quát về các luồng request từ web và app server.

- 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 lại đáp ứng tới user thông quang kênh request.



Why the 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.

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).

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

Installation of the 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 để triển data qua network thống qua dạng binary thay vì dạng text. Nó sử dụng TCP và packet-based protocol, sau đó tăng hiệu năng web server. Một số thông tin khác, việc giải mã request được làm xong trên web server nên app server sẽ không tốn tài nguyên cho vấn đề này.

Note:

Nếu sử dụng AJP, network traffic sẽ hạ xuống, ví giao thức chạy trên giao thức TCP.

* 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:

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

Installation and configuration of 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

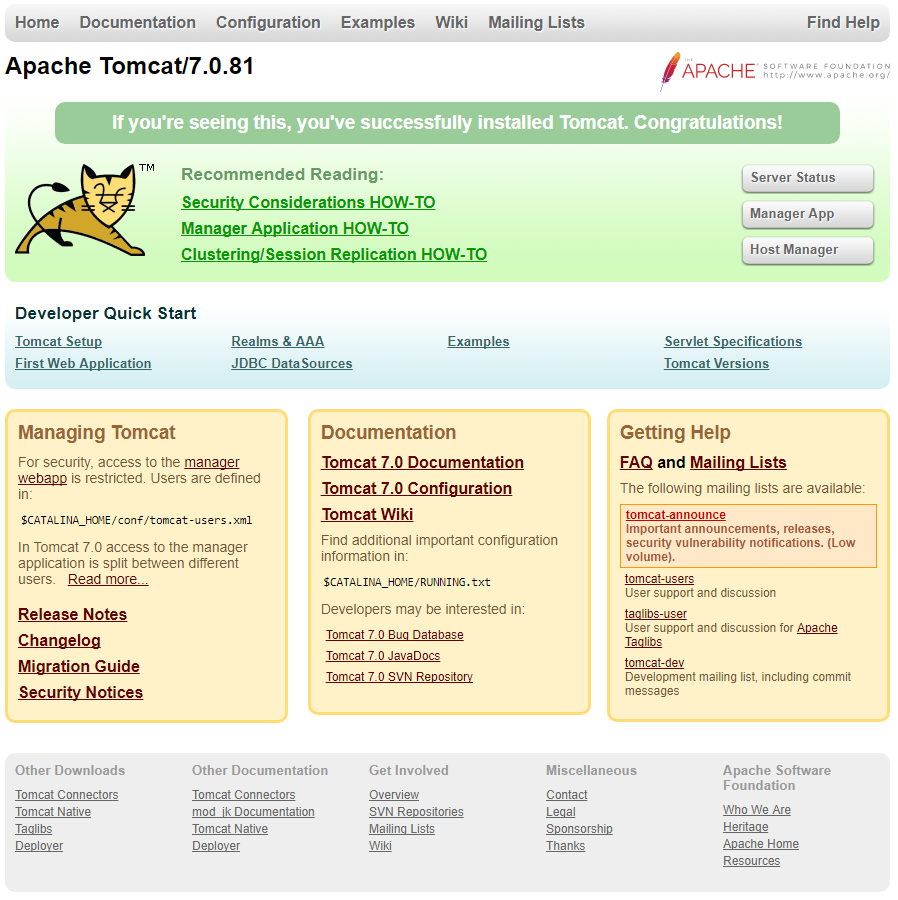
# cd ../apache/bin/

# ./apachectl stop

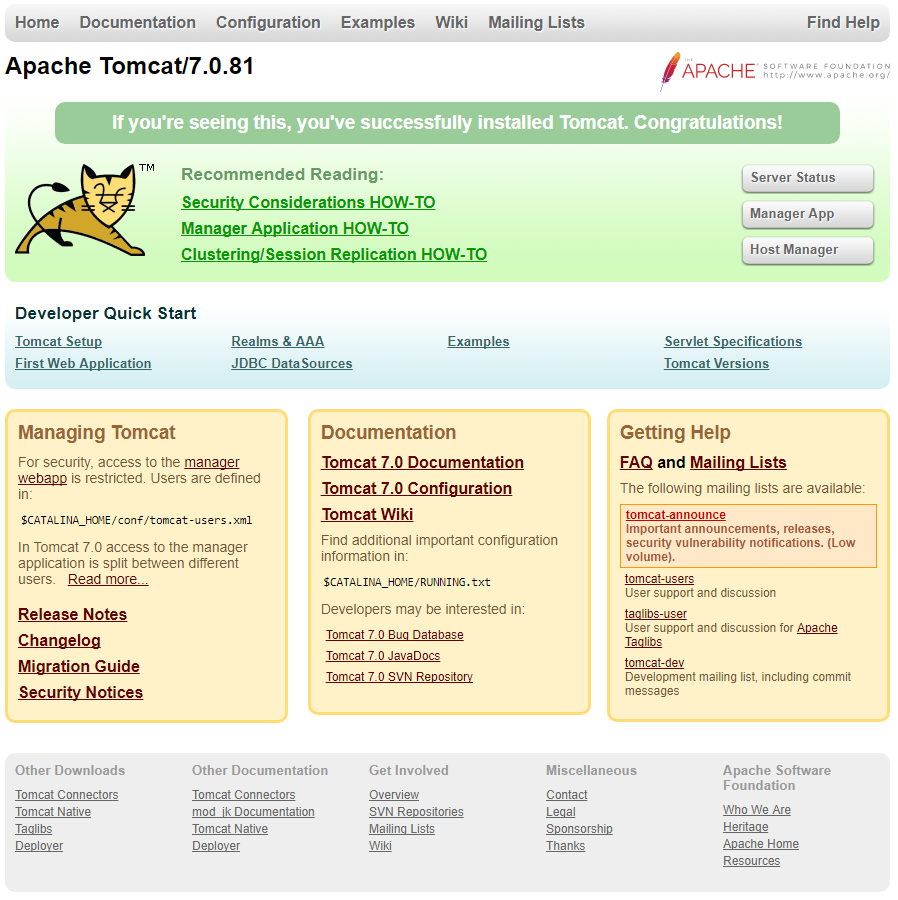
# ./apachectl start

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.

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

Comparison between mod\_jk and 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 |

Securing Tomcat 7

Tomcat Manager

* Tính bảo mật trở thành 1 vấn đề quan trọng trong các công ty IT, vì thế sinh ra 1 bộ phân riêng cho vấn đề bảo mật. Công việc của họ là chịu trách nhiệm cho việc đảm bảo không có lỗ hổng trong hệ thống network, web và hạ tầng OS.
* Nên down các package từ nhưng nguồn đảm bảo, tránh sử dụng các nguồn ko rõ danh tính trên mạng.

Tomcat security permissions

* Tomcat được thiết lập với good security-enabled options, nhưng mỗi môi trường đều có nhưng yếu cầu riêng cho việc bảo mật, nó dựa trên mục đích của các app.
* VD: ngân hàng cần mức độ bảo mật cao, trong khi user app cần mức độ bảo mật thấp hơn.
* Thiết lập quyền cơ bản trong “TOMCAT\_HOME/Conf”, bao gồm các file bên dưới.

catalina.properties

- File này chứa thông tin liên quan đến việc truy cập package, định nghĩa package, common loader, shared loader, tập JAR files. Nó sẽ không cần thiết khi quét tại thời điểm khởi tạo Tomcat. Nó giúp cho việc nâng cao hiệu năng, như nếu thêm nhiều JAR tới skip list sẽ cải thiện việc tiêu thụ bộ nhớ. Nếu bạn muốn thêm bất kỳ common JAR, bạn phải định nghĩa trong “catalina.properties”.

- Trong môi trường thực thi, 1 số thư viện JARs được shared cho nhiều thực thi Tomcat, và trong trường hợp đó, ta sử dụng shared loader parameter. Mặc định, chính sác tomcat sử dụng các package để nâng cao tính bảo mật. Ta có thể customize lại chính sách dựa trên các yêu cầu của app, loại sử dụng.

VD: Cấu trúc sử dụng trong catalina.properties:

package.definition=sun.,java.,org.apache.catalina.,org.apache.coyote.,\

org.apache.jasper.,org.apache.naming.,org.apache.tomcat.

…..

tomcat.util.scan.DefaultJarScanner.jarsToSkip=\

bootstrap.jar,commons-daemon.jar,tomcat-juli.jar,\

annotations-api.jar,el-api.jar,jsp-api.jar,servlet-api.jar,websocket-api.jar,\

catalina.jar,catalina-ant.jar,catalina-ha.jar,catalina-tribes.jar,\

jasper.jar,jasper-el.jar,ecj-\*.jar,\

tomcat-api.jar,tomcat-util.jar,tomcat-coyote.jar,tomcat-dbcp.jar,\

tomcat-jni.jar,tomcat-spdy.jar,\

tomcat-i18n-en.jar,tomcat-i18n-es.jar,tomcat-i18n-fr.jar,tomcat-i18n-ja.jar,\

tomcat-juli-adapters.jar,catalina-jmx-remote.jar,catalina-ws.jar,\

tomcat-jdbc.jar,\

tools.jar,\

commons-beanutils\*.jar,commons-codec\*.jar,commons-collections\*.jar,\

commons-dbcp\*.jar,commons-digester\*.jar,commons-fileupload\*.jar,\

commons-httpclient\*.jar,commons-io\*.jar,commons-lang\*.jar,commons-logging\*.jar,\

commons-math\*.jar,commons-pool\*.jar,\

jstl.jar,taglibs-standard-spec-\*.jar,\

geronimo-spec-jaxrpc\*.jar,wsdl4j\*.jar,\

ant.jar,ant-junit\*.jar,aspectj\*.jar,jmx.jar,h2\*.jar,hibernate\*.jar,httpclient\*.jar,\

jmx-tools.jar,jta\*.jar,log4j.jar,log4j-1\*.jar,mail\*.jar,slf4j\*.jar,\

xercesImpl.jar,xmlParserAPIs.jar,xml-apis.jar,\

junit.jar,junit-\*.jar,hamcrest\*.jar,org.hamcrest\*.jar,ant-launcher.jar,\

cobertura-\*.jar,asm-\*.jar,dom4j-\*.jar,icu4j-\*.jar,jaxen-\*.jar,jdom-\*.jar,\

jetty-\*.jar,oro-\*.jar,servlet-api-\*.jar,tagsoup-\*.jar,xmlParserAPIs-\*.jar,\

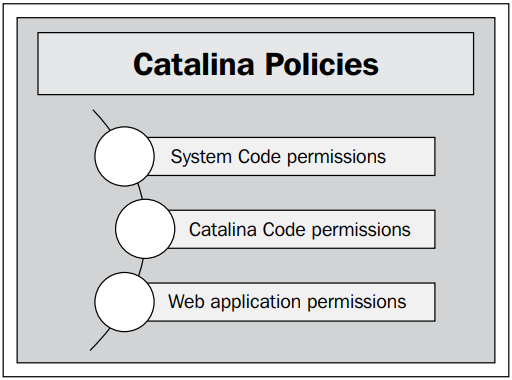
xom-\*.jar

….

catalina.policy

- File này chứa thông tin về Tomcat permission và các app nó đã triên khai, thời gian chạy. Nếu bạn muốn truy cấp bất kỳ giá trị hệ thống nào, như OS details, Tomcat intercode, web app code từ nhưng thư mục khác nhau, bạn có thể định nghĩa permisstion tại đây.

- Có 3 loại permission có thể sử dụng trong Tomcat.



System Code permissions

- Chính sách định nghĩa quyền truy cập vào Java library, nó sẽ cần thiết để kiểm tra tại runtime bởi thực thi Tomcat.

- Code sau đây cho phép full access permissions được thư viện Java:

grant codeBase "file:${java.home}/lib/-" {permission java.security.AllPermission;

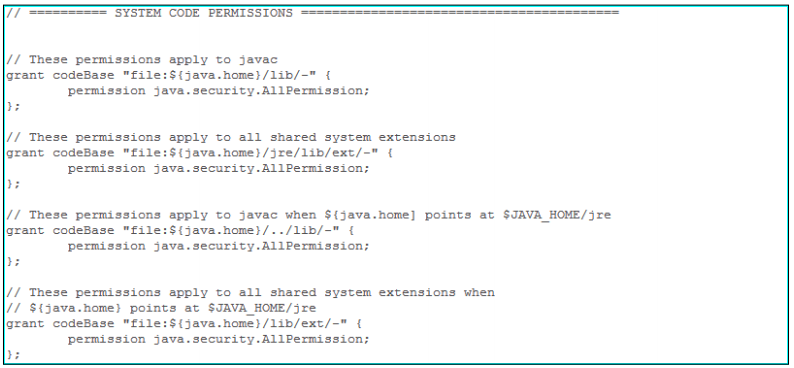
Note:

There are a few more policy options you can implement in Catalina, such as System Code, Catalina Code, and Web application permissions

- Các điểm sau đây mô tả các custom policy mà ta có thể thêm vào Tomcat 7:

+ Read/write access(R/W) to the document root of the web application

+ Read, write, and delete access to the user for the web application directory. The following screenshot shows the different options



Catalina Code permissions (Tomcat core permission)

- Phần này chứa quyền truy cập các file nội bộ Tomcat qua code. Nó giúp việc kiểm soát các tính năng nội bộ Tomcat.

VD: Chính sách truy cập “Catalina/lib” với đầy đủ các quyền

grant codeBase "file:${catalina.home}/lib/-" {permission java.security.AllPermission;

Note:

The previous permission is used for the servlet API and their class loader, which are shared among the different codes.

The following screenshot shows the Catalina Code permissions:

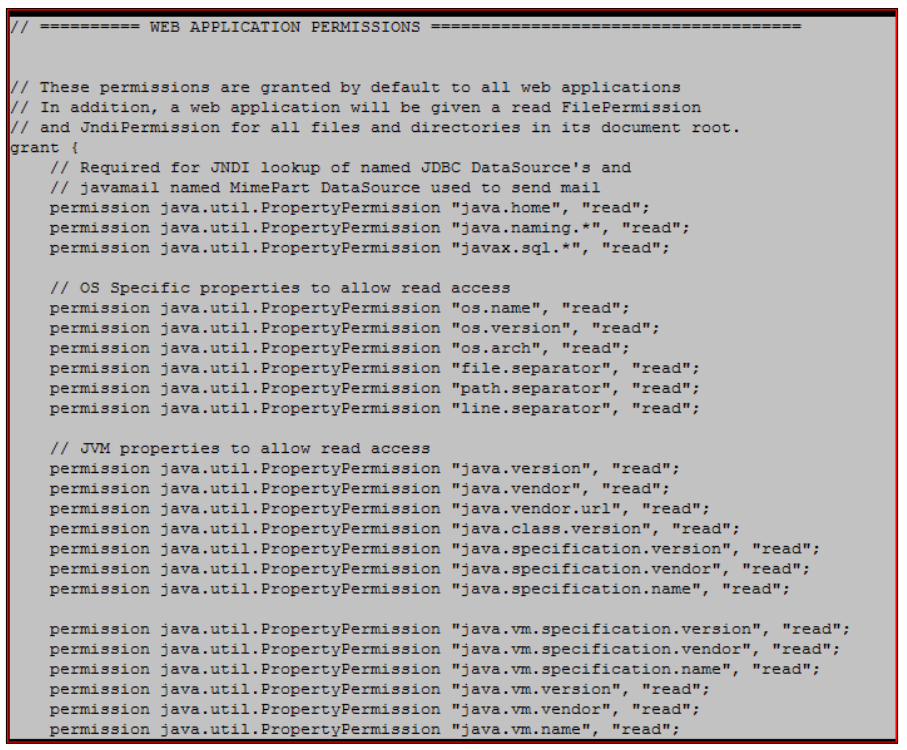
Web application permissions

- Phần này chứa các chính sách liên quan đến đến việc sử dụng tài nguyên của app như JVM, JNDI và .. .

VD: Nếu bạn sử dụng đoạn code sau, Tomcat classes có thể truy cập từ root directory:

// grant codeBase "file:${catalina.base}/webapps/examples/

WEB-INF/classes/-" {// };

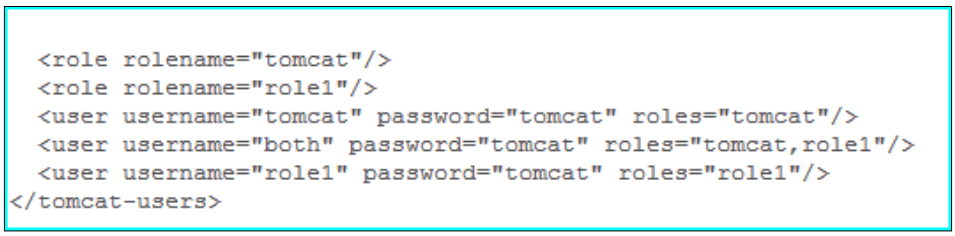


Node:

This policy will run from the root directory of the web application. If you want to access it from outside, then you need to customize the application.

tomcat-users.xml

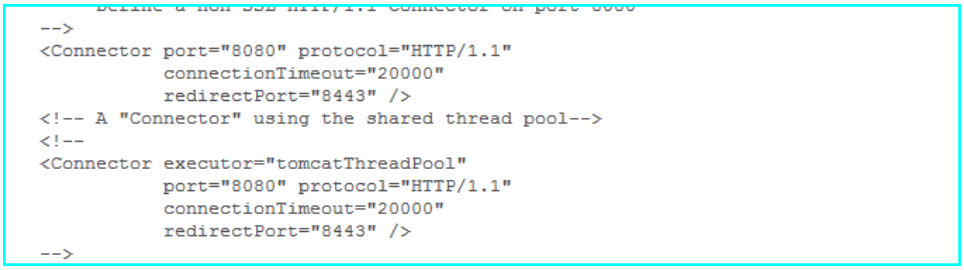
- File này chứa role và security passwd cho tomcat.

VD: 

server.xml

- Đây là file cấu hình chính của tomcat, nó chứa các cấu hình Connector port config.

VD:



Enabling Tomcat Manager

- Mặc đinh, Tomcat Manager không được khởi tạo trong Tomcat. Nó là 1 tool rất mạnh, nhưng đi cùng các vấn đề, nó có thể tạo ra các lỗi giữa sys ad và app ad.

- Vì vậy, phải kích hoạt Tomcat Manager với các bảo mật chính xác.

How to enable the Tomcat Manager

- Cho việc enable Manager, thay đổi file tomcat-users.xml, nó tồn tại trong “TOMCAT\_HOME/conf”.

VD: Thay đội, hoặc sửa nội dung tomcat-user.xml

<role rolename="admin-gui"/>

<role rolename="manager-gui"/>

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

Note:

- Khi sử dụng Tomcat Manager trong môi trường thực thi, đảm bảo nó chỉ có thể truy cập trong mạng nội bộ, không được truy cập thì môi trường khác, đặc biệt DMZ.

Securing Tomcat 7 for production

- Trong phần này, ta sẽ bàn luận các phương pháp tốt nhất để bảo mật Tomcat 7. Bảo mật Tomcat không có nghĩa chỉ Tomcat, nó còn bảo gồm cả Tomcat conf và các cấu hình hạ tầng khác.

Tomcat settings

- Có các cách khác nhau đển bảo đảm Tomcat 7 và nó đều xuất phát từ yêu cầu các app và chỉnh sách bảo mật được sử dụng các tổ chức It.

Note:

Mỗi tổ chức đều có chính xác bảo mật riêng và IT admin sẽ tuân theo quy tắc bảo mật để cấu hình Tomcat.

- Trong Tomcat 7, có nhưng cấu hình khác nhau, nó sẽ cần được thay đổi hoặc kích hoạt để bảo đảm Tomcat trong mỗi trường bên ngoài.

Connector Port

- Mặc định, Tomcat 7 sử dụng port 8080 thông qua HTTP protocol. Tất cả mọi người đều biết port mặc định này, nó dẫn đến dễ dàng bị hacker tân công vào this port và trap server. Vì thế khuyến kích thay đổi port mặc định này cũng như AJP port chạy trên 8009 để bảo đảm Tomcat.

Note:

Connectors are configured in server.xml in the conf directory

VD:

<Connector executor="tomcatThreadPool" port="8080" protocol="HTTP/1.1"

connectionTimeout="20000" redirectPort="8443" />

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

- Ta có thể kiểm tra các port sử dụng bởi các service khác nhau thông qua services file trong Window và Linux. Thông tin này sẽ rất hữu tích trong việc tránh port conflict giữa 2 services.

- Kiểm tra tại linux:

# cat /etc/services

- Network admin sẽ chịu trách nhiệm phân bố các port mới và update việc gán các port với code.

Slimming of Tomcat application

- Tomcat 7 tới với nhiều app và example built-in packages.

- Nó cũng yêu cầu việc xóa các app package không sử dụng.

- Lợi ích của việc xóa package.

+ Reduction in the JVM memory utilization

+ Chances of any vulnerability will be less, as unwanted applications

(libraries/JAR) are not available

+ Easier maintenance of applications

Note:

You can remove as many applications as you want to, the Tomcat Manager can also

be removed if it is not in use.

Disable hot deployment

- Hot deployment hoặc autodeployment là việc xử lý tự động code triển khai tới application mà không cần recycling services.

- Để disable hot deployment, edit server.xml

<Host name="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="true">

Change the autoDeploy to false.

<Host name="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="false">

- Sau khi thay đổi, app chỉ có thể deploy tới webserver sau khi restart.

Non-Tomcat settings

- Trong phần trước, Tomcat settings, ta đã bàn luôn về Tomcat-level conf để thực thi chính sách bảo mật cho Tomcat 7. Trong mỗi trường real-time, với cả công nghệ mới, các setting không đủ để đảm bảo bảo mật. Để làm hệ thống thêm bảo mật, ta cần phải bảo đảm cả kiến trúc hạ tầng.

Service as a separate user

- Ta không cần chạy Tomcat với quyền root user. Thay vì thế, tạo mới user và cấp quyền user chạy Tomcat server. Các config file cần có quyền root và thuộc user group, các thư mục như log cần quyền read/write cho user/group.

Firewall

- Tomcat cần cấu hình trong zone nội bộ nếu nó không sử dụng như frontend app, chỉ chạy connector port – AJP port – và cần mở external DMZ server. Trong trường hợp sử dụng frontend, Tomcat config trong frontend app, sau đó DB server cần đươc lưu trong zone nội bộ với các cấu hình firewall hạn chế. Nếu tạo 1 strong firewall, ta cần enable hoặc disable port tại mức OS lvl sử dụng system firewall. Ta cũng phải kiểm tra lại các firewall rule đã chinh xác chưa.

- Kiểm tra lại iptables hoặc firewalld trong Linux

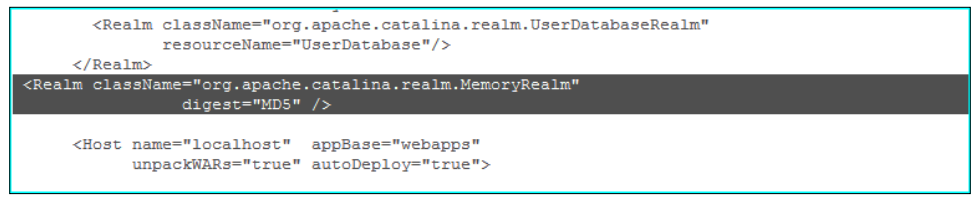
iptables –L

Password- Ta không nền sử dụng text passwd cho app. Khi ở mức độ bảo mật cao, ta cần mã hóa MD5 passwd bằng các thuật toán.

Mã MD5 passwd

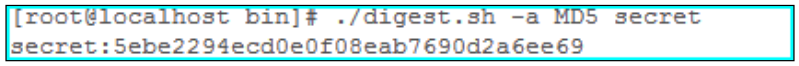
Bước 1: Định nghĩa password encryption trong Realm section server.xml.

<Realm className= "org.apache.catalina.realm.MemoryRealm"digest="MD5" />



Bước 2: Tới thư mục “tomcat\_home/bin” chạy cmd, nó sẽ sử dụng thuật toán mã hóa để sinh mã.

# ./digest.sh -a MD5 secret



- Ở phần trước mô tả ./digest.sh = script, nó sẽ sinh ra passwd cho Tomcal realm và -a = thuật toán sử dụng, ta thường sử dụng thuật toán MD5.

Bước 3:

- Copy MD5 string vừa nhận được vào vị trí pass dạng text trong tomcat\_user.xml theo:

<user name="admin" password="5ebe2294ecd0e0f08eab7690d2a6ee69 "

roles="manager-gui" />

Bước 4:

- Reload Tomcat services là log vào Tomcat Manager bằng passwd đã mã hóa

SSL configuration on Tomcat 7

- Secure Socket Layer (SSL) là cách khác để bảo mật dữ liệu kết nối. Nó là giao thức mã hóa, luồng dữ liệu sẽ được truyền qua kênh đảm bảo. Server sẽ gửi secure key cho client brower, client browser sẽ decrypt dữ liệu, sử dụng cơ chế bắt tay giữa server và client hoặc cũng có thể nói là cơ chế bắt tay 2 chiều trên kênh đảm bảo.

- Sử dụng ssl cho tomcat khi: SSL sẽ có sử dụng nếu bạn sử dụng tomcat như frontend server. Trong trường hợp bạn sẽ dụng Apache or IIS, khuyến khích sử dụng SSL trên Apache hoặc trên IIS server.

Types of SSL certificates

- Trước khi bắt đầu và cài đặt SSL, ta sẽ làm rõ về các chứng chỉ SSL hiện có:

+ Self-signed certificate: Chứng chỉ này sử dụng cho mục đích test app khi host được chạy trong mỗi trường nội bộ, khi không cần sự xác chỉ, chỉ bảo đảm truyền dữ liệu.

+ Signed certificate: Chứng chỉ này sử dụng trong môi trường real-time external environment, khi việc yêu cầu xác thực cũng như bảo đảm kênh truyền. Với chứng chỉ này, ta thường sử dụng các phần mềm thứ 3 để sinh ra chứng chỉ.

Process of installing SSL

- Việc xử lý, khởi tạo chứng chỉ SSL trên mỗi server, nhưng có 1 số thành phần nhất định, nó là Certificate Signing Request (CSR) (Yêu cầu chứng chỉ). Phương thức này sinh ra CSR có thể thay đổi nhưng 1 số thông tin cần thiết cho việc sinh CSR cần được giữa nguyên.

- Tuân theo bảng sau về CSR template:

|  |  |
| --- | --- |
| CSR Attributes | Domain information as to which CSR needs to be generated |
| Common | Name Define the domain name |
| Organization | Organization name |
| Department | Department name of the organization |
| City | City where this organization is located |
| State | State where this organization is located |
| Country | Country where this organization is located |
| Key size | 2048 (encryption bit) |

Note:

Common name: It is the hostname for which CSR needs to be generated.

Key size: It is the size of the encryption keys.

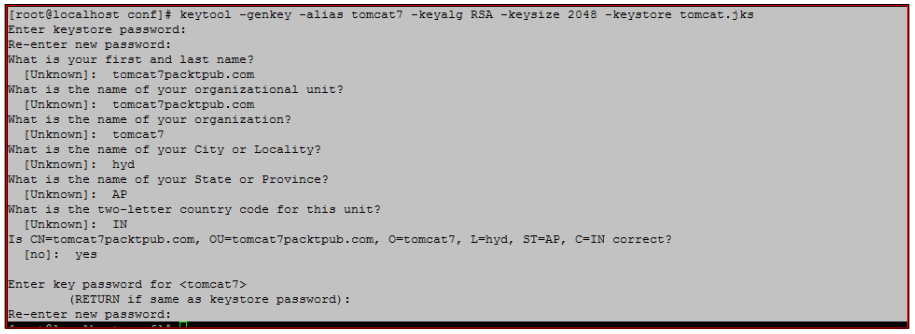
VD: Thực hiện real-time implementation cho việc khởi tạo SSL certificate trên Tomcat 7.

Bước 1: Tạo CSR template cho tomcat7packtpub.com

|  |  |
| --- | --- |
| CSR Attributes | Domain information as to which CSR needs to be generated |
| Common Name | tomcat7packtpub.com |
| Organization | tomcat7packtpub.com |
| Department | Tomcat |
| City | Hyd |
| State | AP |
| Country | IN |
| Key size | 2048 |

Bước 2: Tạo CSR cho host tomcat7.packtpub.com. Cho việc tạo CSR, ta cần chạy key tool trong “JAVA\_HOME/bin”.

# [root@localhost conf] # keytool -genkey -alias tomcat7 -keyalg RSA -keysize 2048 -keystore tomcat.jks



Bước 3: Sinh chứng chỉ CSR format sử dụng cmd. Nó sẽ hỏi passwd và gửi cho người cung cấp chứng chỉ đã có.

[root@localhost conf]# keytool -certreq -alias tomcat7 -file csr.txt - keystore tomcat.jks

Enter keystore password:

Bước 4: Thêm chứng chỉ vào Tomcat key store. Copy tomcat7.jks vào

“TOMCAT\_HOME/conf”.

[root@localhost conf] # keytool -import -trustcacerts -alias tomcat7 -file tomcat7packtpub.com.pb7 –tomcat7.jks

* Khi bạn đã ký chứng chỉ đã tạo, cấu hình thay đổi giá trị Tomcat conf

Bước 1: Mở Server.xml, thay đổi cấu hình

<Connector port="443" maxHttpHeaderSize="8192" maxThreads="150"

minSpareThreads="25" maxSpareThreads="75" enableLookups="false"

disableUploadTimeout="true" acceptCount="100" scheme="https"

secure="true" SSLEnabled="true" clientAuth="false"

sslProtocol="TLS" keyAlias="server" keystoreFile="tomcat7.jks"

keypass="changeit" />

Bước 2: restart lại service Tomcat

Logging in Tomcat 7

* 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.
* Các phần sẽ nói đến:

+ Logging services in Tomcat 7

+ JULI

+

+ Log level

+ Valve component

+ Analysis of logs

JULI

* 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:

<http://tomcat.apache.org/tomcat-7.0-doc/logging.html>

* Nội dung logging

[root@test bin]# ls -lt

total 804

-rwxr-xr-x. 1 root root 21667 Sep 9 00:14 catalina.sh

-rwxr-xr-x. 1 root root 1908 Aug 11 13:23 version.sh

-rwxr-xr-x. 1 root root 1922 Aug 11 13:23 configtest.sh

-rwxr-xr-x. 1 root root 7888 Aug 11 13:23 daemon.sh

-rwxr-xr-x. 1 root root 1965 Aug 11 13:23 digest.sh

-rwxr-xr-x. 1 root root 3547 Aug 11 13:23 setclasspath.sh

-rwxr-xr-x. 1 root root 1902 Aug 11 13:23 shutdown.sh

-rwxr-xr-x. 1 root root 1904 Aug 11 13:23 startup.sh

-rwxr-xr-x. 1 root root 5024 Aug 11 13:23 tool-wrapper.sh

-rw-r--r--. 1 root root 2026 Aug 11 13:23 version.bat

-rw-r--r--. 1 root root 28503 Aug 11 13:23 bootstrap.jar

-rw-r--r--. 1 root root 14028 Aug 11 13:23 catalina.bat

-rw-r--r--. 1 root root 1686 Aug 11 13:23 catalina-tasks.xml

-rw-r--r--. 1 root root 24283 Aug 11 13:23 commons-daemon.jar

-rw-r--r--. 1 root root 204944 Aug 11 13:23 commons-daemon-native.tar.gz

-rw-r--r--. 1 root root 2040 Aug 11 13:23 configtest.bat

-rw-r--r--. 1 root root 2091 Aug 11 13:23 digest.bat

-rw-r--r--. 1 root root 3430 Aug 11 13:23 setclasspath.bat

-rw-r--r--. 1 root root 2020 Aug 11 13:23 shutdown.bat

-rw-r--r--. 1 root root 2022 Aug 11 13:23 startup.bat

-rw-r--r--. 1 root root 44511 Aug 11 13:23 tomcat-juli.jar

-rw-r--r--. 1 root root 399836 Aug 11 13:23 tomcat-native.tar.gz

-rw-r--r--. 1 root root 4021 Aug 11 13:23 tool-wrapper.bat

Loggers, appenders, and layouts

* Có 1 số thành phần quan trọng cho việc logging, ta sẽ sử dụng để implement lại cơ chế logging cho app. Mỗi thuật ngữ sau đây là có tầm quan trọng riêng trong việc tracking event của app.

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

Note:

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

+ 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

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

VD:

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

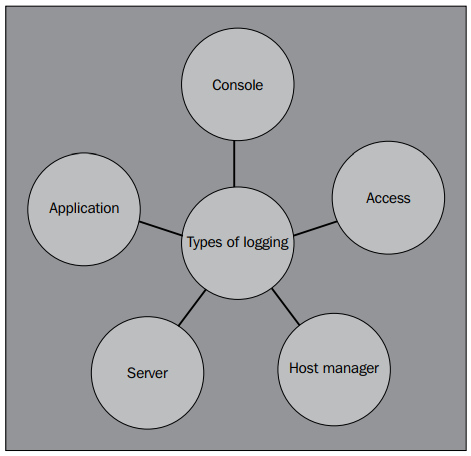
directory="logs" prefix="localhost\_access\_log." suffix=".txt"

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

Types of logging in Tomcat 7- Ta có thể enable logging trong Tomcat 7 bằng nhiều cách dựa trên các yều cầu

* Có tổng cộng 5 loại logging mà ta có thể cấu hình trong Tomcat như app, server, console ..
* Các phương phép này được kết hợp với nhau dựa trên từng yêu cầu các môi trường.

VD: Nếu server ko hiển thị, console logs rất hữu ích cho việc để xác định vấn đề này.



Application log

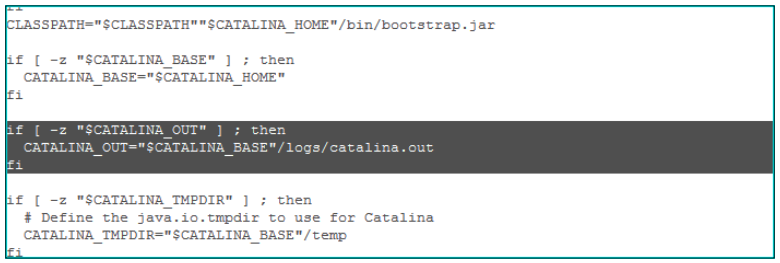
* Log này sử dụng để bắt các sự kiện app trong khi đang chạy app transaction. Log này rất hữu tích cho việc xác thực mức độ vấn đề app.
* VD: Giả sử app chạy chậm khi thực hiện 1 transition nào đóc, thông tin về transition có thể traced trong app log.
* Điểm mạnh lớn nhất của app log là ta có thể cấu 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

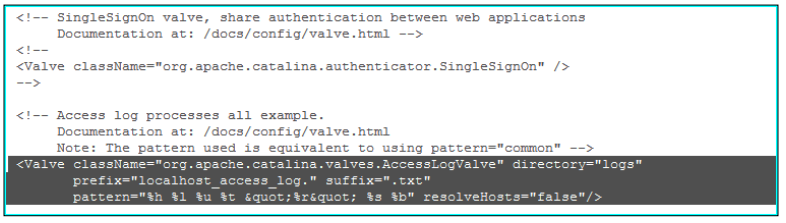
* Access log được customized logs, sử dụng để:

+ 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

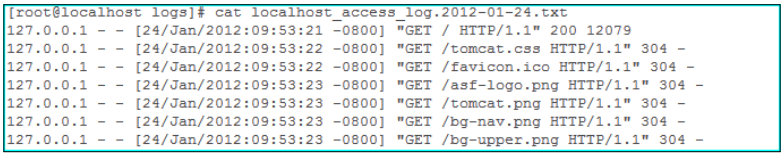
host address. The following screenshot shows the default log format for

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

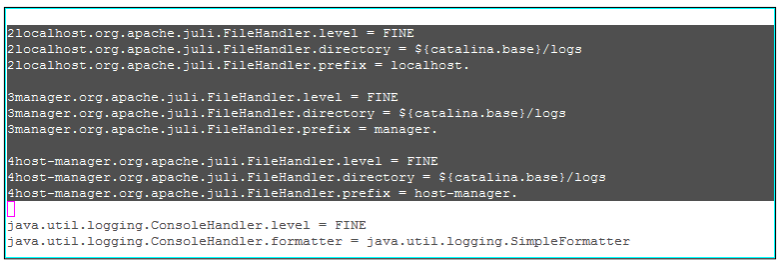
status code.

Note:



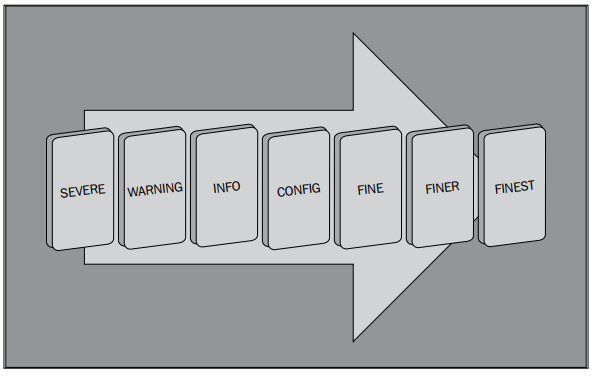
Host manager

* 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”.



Types of log levels in Tomcat 7

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



* Mỗi log level trong JULU 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

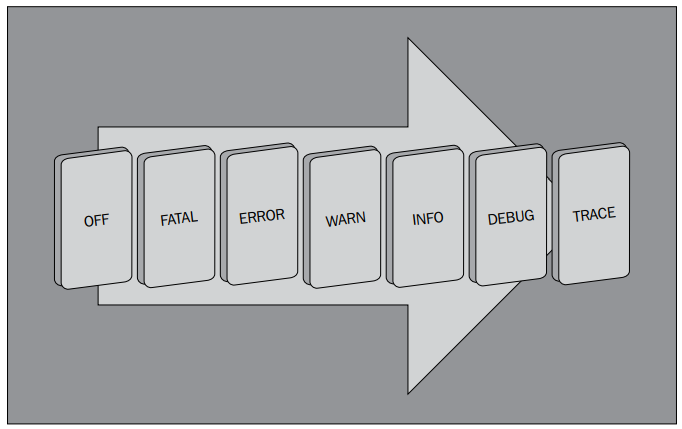
localhost.org.apache.juli.FileHandler.directory = ${catalina.base}/logs

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

Log4j

- 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 for 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. |

How to use log4j

- Các bước sau để thực hiện log4j:

Bước 1: Download log4j từ trang chủ

Link: <https://logging.apache.org/log4j/2.x/download.html>

Bước 2:

- Giải nén log4j.jar vào lib trong “TOMCAT\_HOME/lib”, xóa tất cả “juli\*.jar” từ lib.

Bước 3:

- Xóa logging.properties trong “TOMCAT\_HOME/CONF”.

Bước 4:

- Tạo log4j.properties trong “TOMCAT\_HOME/CONF”. Định nghĩa log appender trong thực thi Tomcat.



Note:

You can customize the log rotation based on size, day, hour, and so on, using the previous log4j appenders marked in bold.

Bước 5: Restart tomcat services.

Log level mapping

- So sáng các mức log level giữa JULI và log4j.

|  |  |
| --- | --- |
| Log level in JULI | Log level in log4j |
| SEVERE | FATAL, ERROR |
| WARNING | WARN |
| INFO | INFO |
| CONFIG | NA |
| FINE | DEBUG |
| FINER | DEBUG |
| FINEST | TRACE |

Values for Tomcat 7

Values are defined as identifiers which change the pattern of the string in the log.

Suppose you want to know the IP address of a remote host, which has accessed the

website, then you add the combination of the following values mentioned in the log

appenders. For example, let's customize the access logs for Tomcat 7

Log analysis

- Phân tích log là 1 phần quan trọng và khó khăn khi xử lý các vấn đề trong app.

- Một số lời khuyên khi phân tích log:

+ Kiểm trả log từ 1h trở lại khi xảy ra lỗi

+ Luôn luôn xem xét exception đầu tiên trong log khi error bắt đầu

+ Luôn nghĩ tới trường hợp các lỗi có thể không bắt nguồn từ tomcat mà xuất phát từ 1 tài nguyên khác.

- Các công cụ rất hữu ích khi sử dụng cho phân tích log là grep và awk.

+ grep: This utility prints the lines which match the string searched

+ awk: This command is used for pattern scanning

Troubleshooting in Tomcat

- Trong phần này sẽ mô tả:

+ Các vấn đề chung

+ Các công cụ phân tích thread dump

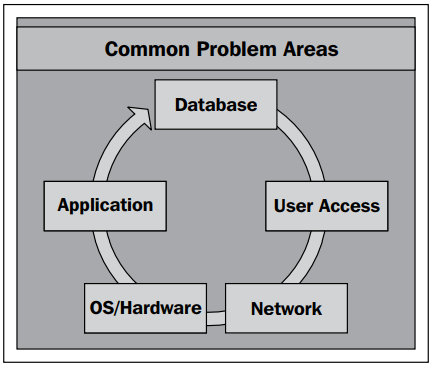
+ 1 số vấn đề đặc biệt trong Tomcat liên quan đến OS, JVM và database.

+ Cách sử chưa các vấn đền

+ Các lời khuyên trong môi trường chạy thực tế.

Các vấn đề phổ bến cho web admin.

- Web admin thường tìm các vấn đề trên app mà quên mất rằng 1 số vấn đề bắt đầu từ Tomcat server hoặc cũng có thể do 1 số thành phần khác.



- 1 số vấn đề web sysad 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 conflicts, app deployment conflict, config parameter missing, …

+ Database: Vấn đề về Database rất quan trọng đối với web ad. Nó rất khi để tìm ra vấn đề liên quan tơi 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 server gặp vấn đề, thì các giao tiếp với server cũng gặp vấn đề, từ đó sinh ra lỗi services.

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

How to troubleshoot a problem

- Ta không thể giải quyết các vấn đề thông qua user comment hoặc problem statement. Để có thể giải quết được vấn đề, ta phải thu hẹp phạm vi lỗi tới root level và fix vấn đề đó.

- Câu hỏi đặt ra: Làm sao có thể biết hệ thống có nhưng lỗi đặc biệt nào?

+ Giải pháp cho các vấn đề sẽ được tìm thấy nếu bạn đào sâu lỗi đúng cách.

+ Nếu bạn trải qua nhiều lỗi trong quá trình làm việc, bạn có thể sử được theo kinh nghiệm làm việc trước đó.

+ Không thể chỉ ra các lỗi đặc biệt hoặc dạy việc sửa lỗi, nó chỉ có thể biết được thông qua kinh nghiệm xử lý.

Slowness issue in applications

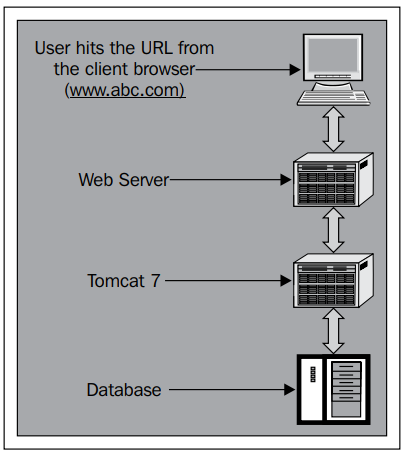
- Lỗi phổ biến trong môi trường thực thi khi user phàn nàn về hiệu năng của app. App bao gồm enterprise setup, nó sẽ kết hợp Apache HTTP server as frontend, Tomcat 7 sử dụng như servlet container và oracle database chạy phí backend database server.

Vấn đề:

- Các vấn đề thường thấy tại middleware app, nó sẽ khó để admin có thể giải quyết. Nó gọi là “slowness of application”, khi user sẽ phần nàn về app chạy chậm. Nó là critical problem từ điểm nhìn admin, vì sự chậm chễ có thể bắt nguồn từ bất từ thành phần nào của web như OS, DB, web server, network, …

- Cho đến khi ta tìm thấy thành phần nào gây nên lỗi, hệ thống mới có thể khôi phục lại sự ổn đinh (từ góc nhìn user).

- Sơ đồ kiến trúc web flow cho web app.



Cách giải quyết vấn đề chậm chạp trong Tomcat 7.

- Sự chậm chạp có thể do nguyên nhân của bất kỳ thành phần nào, các tốt nhất để tìm kiếm vấn đề là bắt đầu từ sự điểm nhìn của user.

User end troubleshooting

- Thực hiện các bước sau:

+ Thử truy cập vào app thông qua user browser và kiếm tra thời gian load app page

+ Kiểm tra ping response của server từ phía user. Sử dụng “ping” cmd. Nếu bạn thấy sự trả lời bình thường, điều đó có nghĩa kết nối giữa app server và user không xảy ra vấn đề. Và ngược lại.

Web server troubleshooting

- Khi biết lỗi không xảy ra do kết nối user vs app, ta sẽ kiểm tra tại mức app, đó là web server.

- Vấn đề về web server thường liên quan tới load server, user thread, hoặc vấn đề liên kết.

Các bước kiểm tra:

Bước 1:

- Kiểm tra web server process có còn chạy không. Nếu chạy, kiểm tra các tiến trình đang chạy thông qua cmd:

# ps –aef | grep httpd

- Nếu có quá nhiều process id, nó có nghĩa có thể đã xảy ra vấn đề về sử dụng CPU, high user traffic, high disk I/O,..

Bước 2:

- Kiểm tra load CPU, trạng thái memory của hệ thống để kiểm tra các tiền trình Apache đang sử dụng bao nhiều CPU

- Nếu thấy CPU và memory sử dụng nhiều cho apache, nó có thể là nguyên nhân dẫn đến web chạy chậm.

# top | head

Bước 3:

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

# tail -f log file | grep ERROR

# grep " 500 " access\_log

Bước 4:

- Lý do này xuất hiện do hard drive đang sử dụng đã quá tải, cần mơ rộng thêm bộ nhớ hoặc xóa bớt …

- Sử dụng cmd “df” để kiểm tra lại mount space.

Note:

- Nếu thông qua các bước trên, ta không tìm thấy lỗi trên các thành phần, ta có thể khẳng định lỗi không xảy ra tại web server.

Tomcat 7 troubleshooting

- Java-based app có rất nhiều vấn đề. Có thể do JVM memory, triển khai không đúng, cấu hình DB sai, ..

- Các bước kiểm tra:

Bước 1:

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

- Load trung có thể cho ta 1 số đầu mối. Trong trường hợp load trung bình rất cao, sau đó kiểm tra cpu sử dụng cho app, nếu cao hãy tìm nguyên nhân dẫn đến vấn đề đó. Nó cũng có thể do Ram và Swap usage.

# ps –ef | grep java

Bước 2:

- Kiểm tra log Tomcat tại “TOMCAT\_HOME/logs” và tìm kiếm các exception. Kiểm tra các file: “catalina.out”, “localhost.yyyy-mm-dd.log”, sử dụng cmd.

# grep INFO catalina.out

# grep ERROR catalina.out

Troubleshooting at the database level- Đối với web admin, bạn không cần truy cập vào database web. Nhưng web admin có thể kết nối tới bên ngoài DB, service DB, kiểm tra connection string.

VD: có thể telnet tới port DB server đang chạy, kiểm tra khi nào service đang chạy hoặc không.

Telnet DB server IP port:

- If the telnet is successful, then you can verify the following processes:

+ Number of database connections: We can always ask our DBA to check the number of connections on the database. If the connections count is high, then we can work with the DBA to reduce the connections on the server.

+ SQL query optimization: We can check with the DBA to see which queries consume more time to execute in the database and ask our developers to optimize the query. This really helps in improving the performance of the application.

+ Load balancing database across multiple servers: Another important point which may cause slowness in the application is the load balancing of the database across multiple servers. If the load balancing is not configured correctly, then it may cause slowness in the application. If there is a delay in the network between the two database servers, then sync may not happen appropriately.

JVM analysis in the Tomcat instance

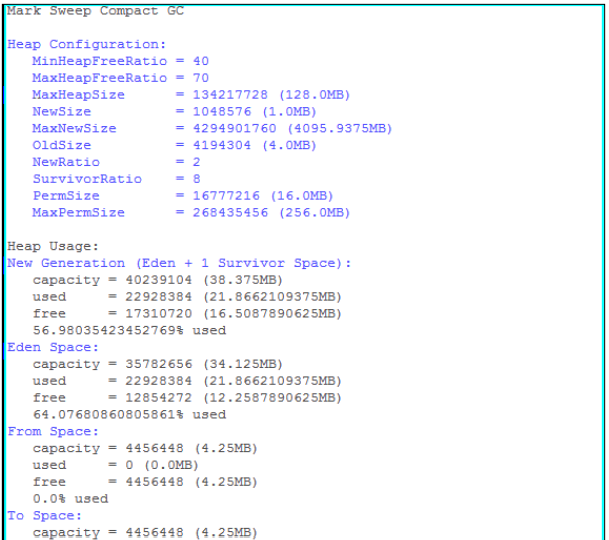
- Để xem số memory sử dụng cho JVM, ta có thể sử dung cmd “jmap”

VD:

# jmap -heap "TOMCAT INSTANCE PID "

# jmap -heap 10638

- jmap cmd được đặt trong “JAVA\_HOME/bin” và nếu set “JAVA\_HOME/bin” tại thực thi linux, ta có thể thực hiện cmd ở bất kỳ đâu.



- JVM memory bao gồm các thành phần:

+ 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.

How to obtain a thread dump in Tomcat 7

- Tìm kiếm, thu thập thread dump trong tomcat.

- Thread dump là cách ta xác định các thread status của Java-process tại mức app-level. Có rất nhiều cách để thu thập thread dump trong Tomcat, ta sẽ sử dụng 2 cách thường dùng nhất.

Thread dump using Kill command

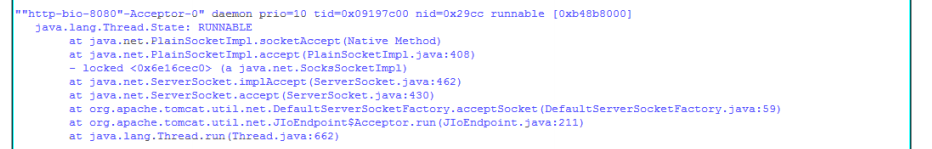
- cmd sinh ra và chuyển hướng thread dump tới catalina.out log. Sử dụng trên linux

Kill -3 java process id

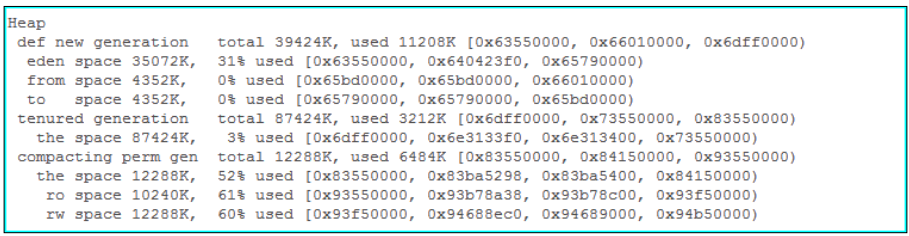
Kill -3 10638

VD:

- Các thread java đang hoạt động



- The previous screenshot shows the output of the thread dump command in catalina.out logs. We can see that the highlighted section shows the httpbio-8080-Acceptor thread status, which is currently in a runnable state, which means that the thread is alive and performing its functionality for the application.



- 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

Thread dump using 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

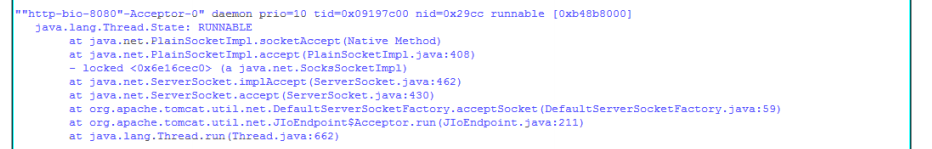
How to analyze the thread dump for Tomcat instance

- 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

JVM (memory) issues

- Ngày nay, app cần rất nhiều tài nguyên, thưc thi tomcat có thể full memory vì lý do đó, ad cần phải tùy chỉnh Tomcat 7 phù hợp với các môi trường khác nhau và theo yêu cầu app để tránh lỗi sinh ra từ internet.

Out of Memory exception

In an enterprise environment, out of memory issues are encountered on a regular

basis due to a high memory requirement of applications and the administrator has

to tune the JVM. Failure of this causes an Out of Memory exception for the

Tomcat instance.

Exception:

SEVERE: Servlet.service() for servlet jsp threw exception

java.lang.OutOfMemoryError: Java heap space

Reason:

This error may often occur while running an application, which requires high

memory-intensive resources. Hence, it causes the Out of Memory exception on the

server and leads to an interruption of services.

Solution:

You have to increase the maximum heap size for the Tomcat system. It's important to

note that you can only allocate 70 percent of physical memory as JVM memory and

30 percent is reserved for the OS. Check the JVM configuration using the command

jmap and then increase it in the configuration.

You have to add the following Java parameters in the startup script of Tomcat, which

can be found in TOMCAT\_HOME/bin, to increase the JVM allocation based on the

memory requirement and recycle the Tomcat instance.

JAVA\_OPTS="-Xms512m –Xmx1048m

OutOfMemoryError: PermGen space

Tomcat administrators often face the problem with the application's permanent

object generation, as every application has different requirements of object

generation. Hence, application slowness also results in the generation of

OutOfMemoryError: PermGen space exception in catalina.out.

Exception:

MemoryError: PermGen space

java.lang.OutOfMemoryError: PermGen space

Reason:

The permanent generation is unique because it holds metadata describing user

classes. Applications with a large code base can quickly fill up this segment of the

heap which causes java.lang.OutOfMemoryError: PermGen, no matter how high

your -Xmx and how much memory you have on the machine.

Solution:

The following parameter should be added to the startup script of Tomcat 7. The

parameter will increase the permanent generation space at the time of startup

of Tomcat 7.

-XX:MaxPermSize=(MemoryValue)m

For example:

-XX:MaxPermSize=128m

…..

Web server benchmarking

- 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

Monitoring and Management of Tomcat 7

- 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.

- Trong phần này sẽ bàn luận:

+ How to monitor Tomcat 7

+ Management of applications using the Tomcat Manager

+ A third-party utility used for monitoring Tomcat 7

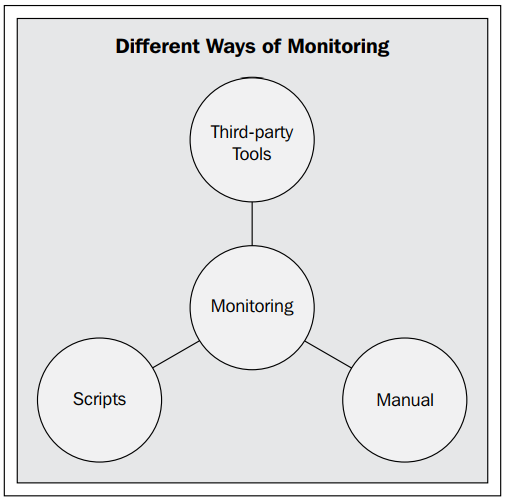
- Trước khi tìm hiểu vấn đề giám sát tomcat 7, ta sẽ giải thích vai trò quan trọng của giám sát trong thực tế. Trong môi trường thực tế, hệ thống có thể gặp lỗi vì rất nhiều lý do như sự cố (glitch) network, cpu tăng đột ngột, JVM crash, .. Và có một số hệ thống rất quan trọng, vì nó thu thâp, thực hiện 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.

Different ways of monitoring

- 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.

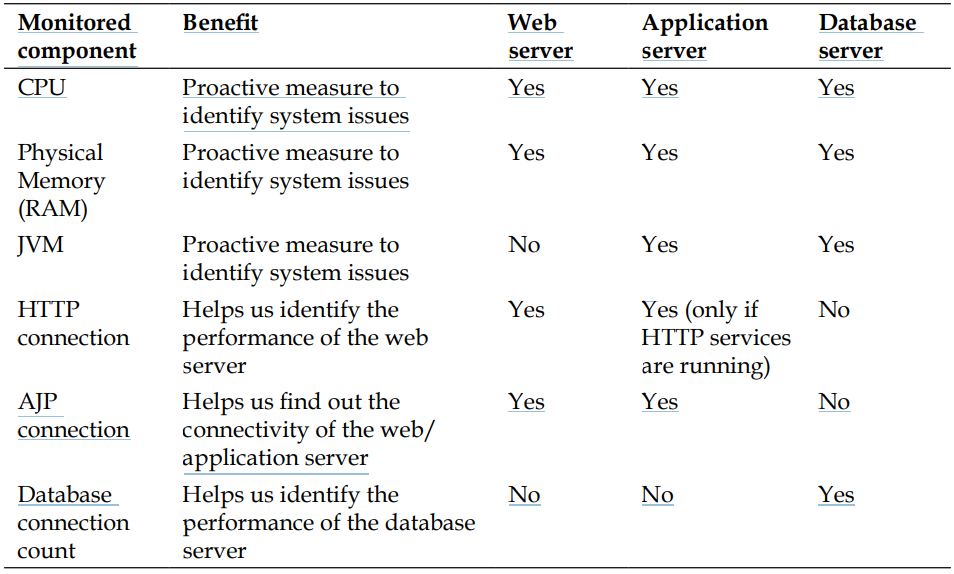
+ Sử dụng trên mọi tổ chức

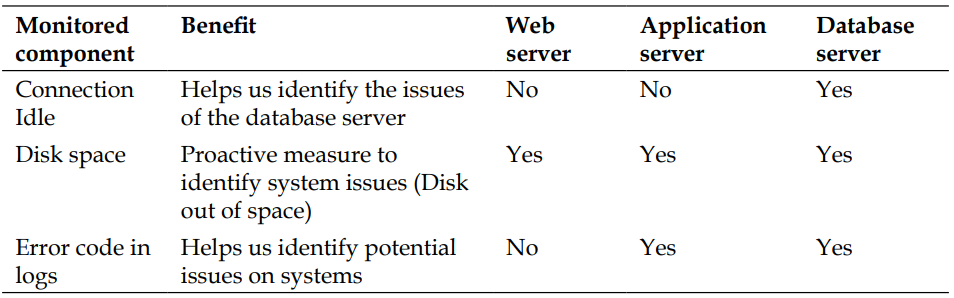
- Manual+ Process được sử dụng khi hiệu năng của app xuồng thấp vì nhưng nhưng module đặc biệt

+ Hầu như sử dụng khi số hệ thống nhỏ (<3)

Monitoring setup for a web application and database server

- Các thành phần sẽ được giám sát





Tomcat Manager in Tomcat 7

- Tomcat manager là công cụ mặc định để quản lý hoạt động Apache Tomcat 7.

- 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.

- Ngắn gọn, we can say remote administration of Tomcat becomes very easy for administrators

Note:

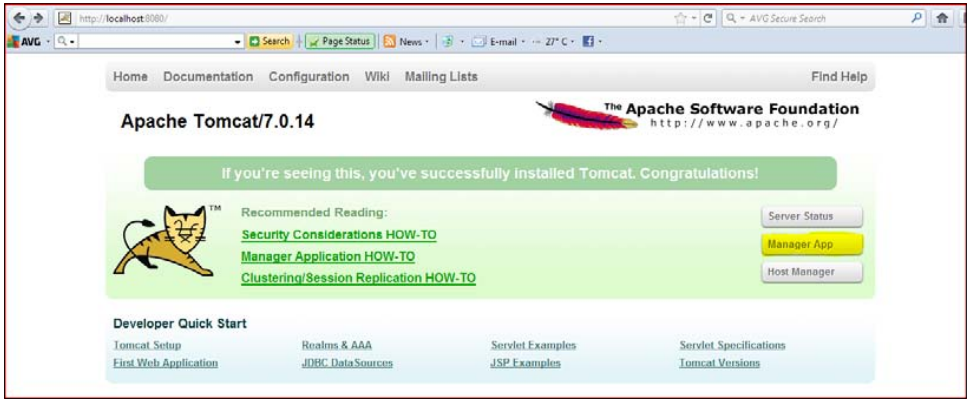
It's not recommended to open the Tomcat Manager from the Internet. In case you have to do so, then we have to enforce strong security policies on Tomcat 7 or we can configure the Virtual Private Network (VPN) for the administrators

Để thực hiện các bước để truy cập Tomcat Manager:

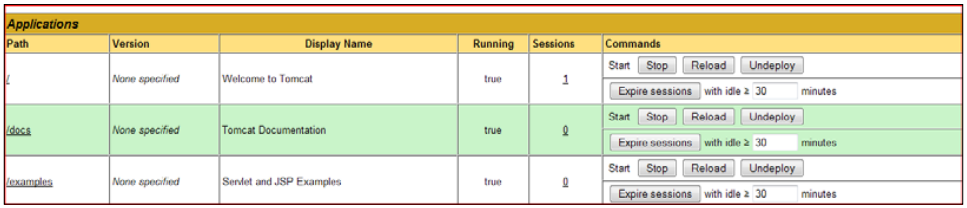
Bước 1: Truy cập Tomcat Manager sử dụng URL <http://localhost:8080/>

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, ..

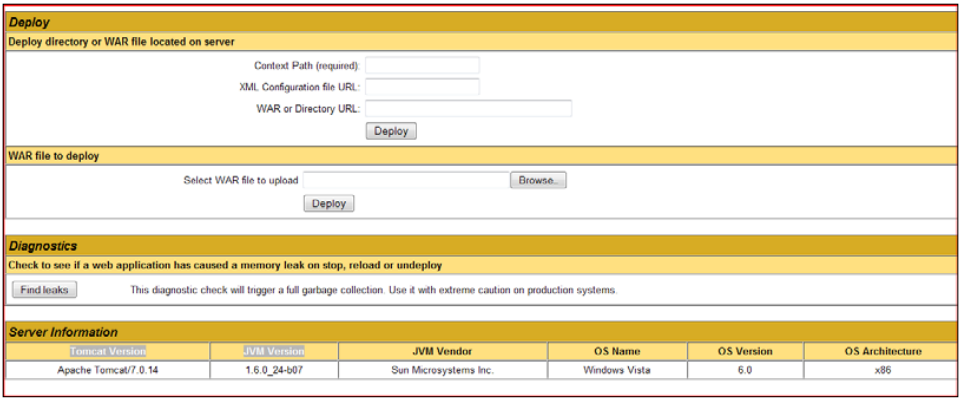


Các tính năng trong Tomcat 7 Manager:

+ Deployment of a new application

+ Diagnostic (memory or connection leak)

+ Server information



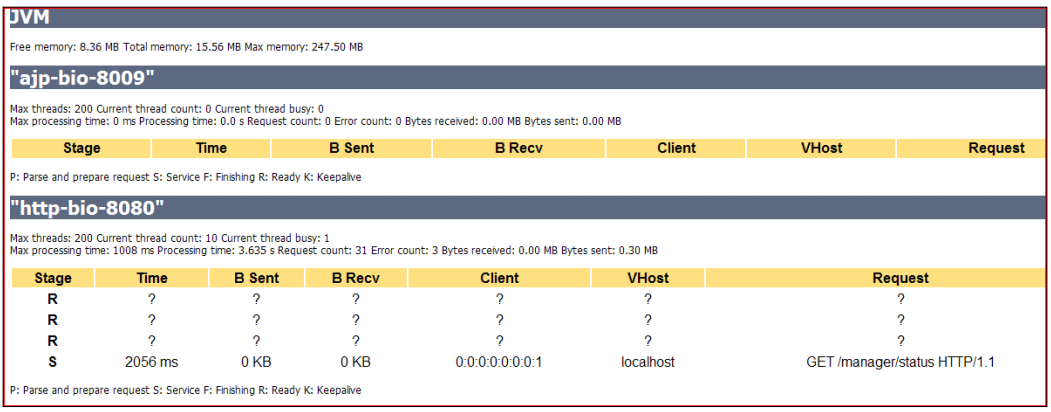
Monitoring in Tomcat 7

- Có thể sử dụng Tomcat Manager để giám sát Tomcat 7. 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.

Summary of the Server Status of Tomcat 7

- Tóm tắt Tomcat 7 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 of Tomcat 7

- Xem toàn bộ trạng thái trên Tomcat 7.

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

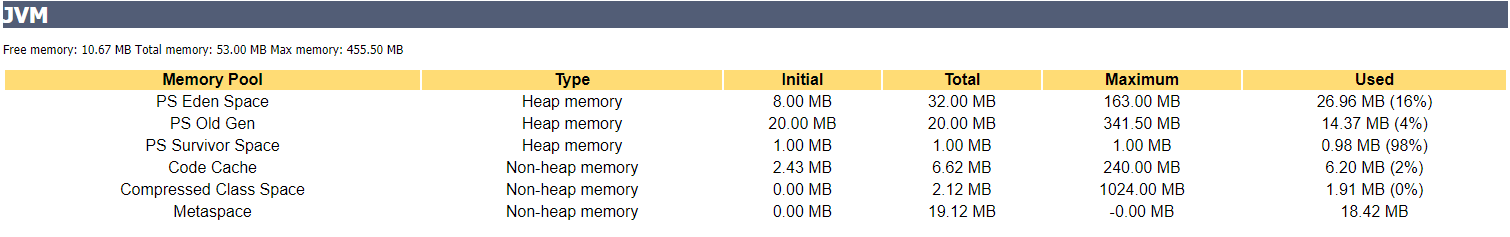
- Hiện thị JVM memory sử dụng trong thực thi Tomcat.

Bao gồm:

+ Free memory

+ Used memory

+ Total memory

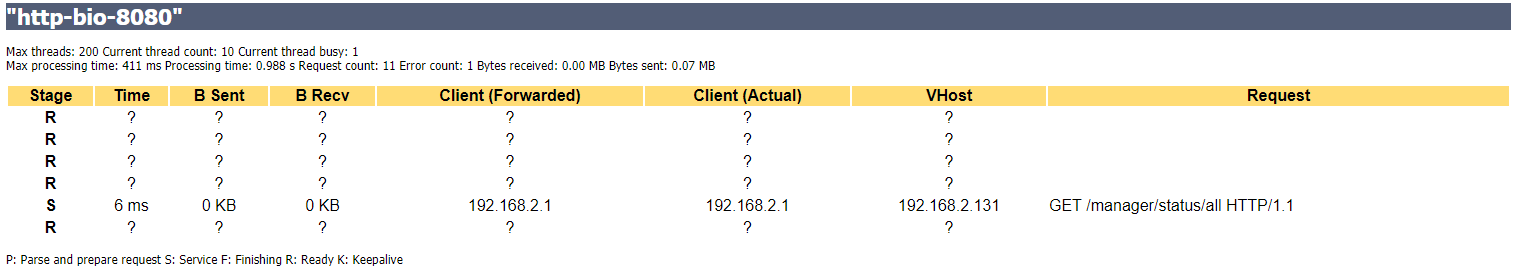


Connections on the HTTP port (8080)

- 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)

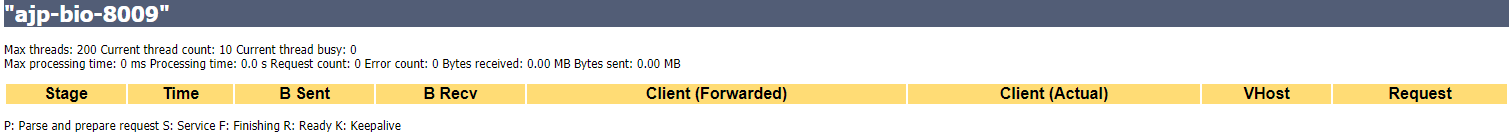


Connections on the AJP

- 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)



JConsole configuration on Tomcat 7

- Jconsole là 1 trong nhưng công cụ giám sát tốt nhất cho JDK 1.5 hoặc hơn.

- Tên đầy đủ Java Monitoring and Management Console.

- Nó là 1 graphical tool, với các thông tin về app, server performance.

Bao gồm:

+ Detect low memory

+ Enable or disable the GC and class loading verbose tracing

+ Detect deadlocks

+ Control the log level of any loggers in an application

+ Access the OS resources—Sun's platform extension

+ Manage an application's Managed Beans (MBeans)

Remote JMX enabling

- Để sử dụng Jconsole cho Tomcat 7 monitoring, ta cần enable Java Managerment Extension (JMX) trên Tomcat 7.

- Để enable, thêm tham số “CATALINA\_OPTS” trong “catalina.sh”. Mặc định thông tin về các giá trị được enable bằng:

CATALINA\_OPTS=-Dcom.sun.management.jmxremote \

-Dcom.sun.management.jmxremote.port=%my.jmx.port% \

-Dcom.sun.management.jmxremote.ssl=false \

-Dcom.sun.management.jmxremote.authenticate=false

VD:

CATALINA\_OPTS="-Djava.awt.headless=true -Xmx128M -server -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=8086 -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote=true"

- Các giá trị:

-Djava.awt.headless: It is a system configuration option that helps the graphics rendering program to accept the graphics console and redirects the program to work in the command-line mode. It is very useful while connecting to the remote server.

-Dcom.sun.management.jmxremote: This JMX allows the host to connect to the system.

-Dcom.sun.management.jmxremote.port: It defines the port where your Remote Method Invocation (RMI) is connected.

-Dcom.sun.management.jmxremote.authenticate: It defines the authentication mechanism for the connection.

-Dcom.sun.management.jmxremote.ssl: It defines the protocol used for communication. If it is set to false, then, by default, it uses the

HTTP protocol.

Note:

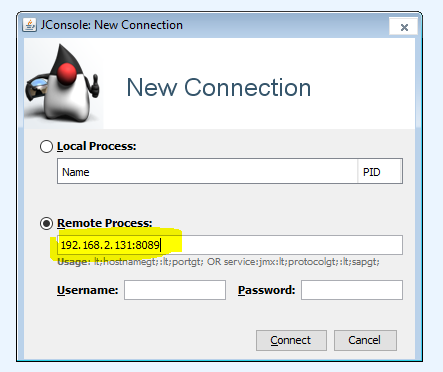
- Khi thay đổi catalina.sh nhớ reset lại tomcat

How to connect to the Jconsole

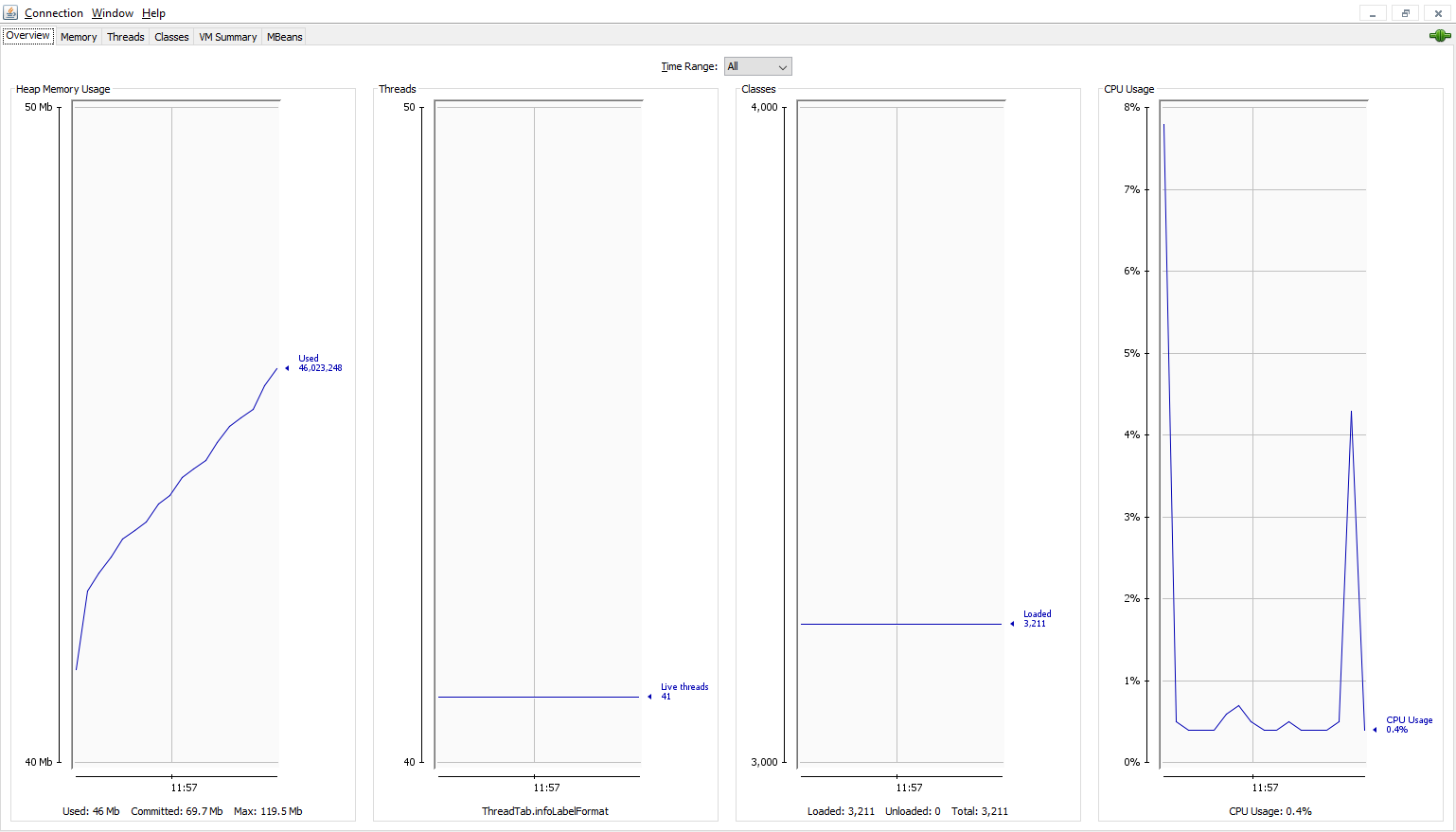
- Để chạy, sử dụng giao diện quản lý jconsole, thông qua chế độ remote connection.

Bước 1:

- Giao diện cơ bản



- Giao diện giám sát



- Các thông tin giám sát bao gồm:

+ Memory

+ Thread

+ Classes

+ Mbeans

- Các điểm mạnh của công cụ:

+ Online analysis of the application

+ Customized report for the analysis

+ Deadlock can be retrieved in the systems

Memory overview

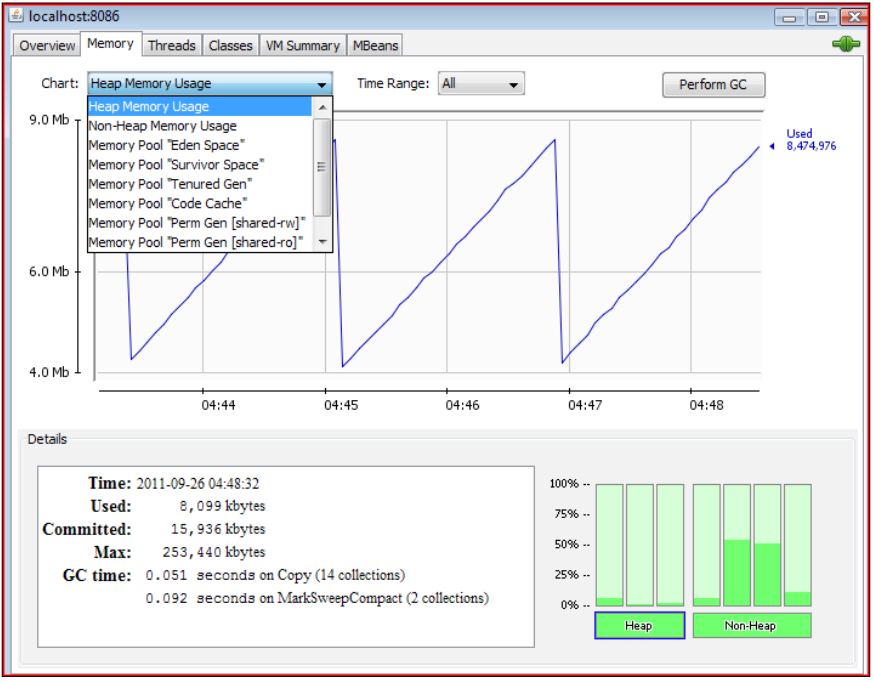
- Nó cần thiết cho web app phân tích tình trạng memory sử dụng, tránh gặp vấn đề với server.

- Các tính năng:

+ Graphical presentation of memory with their JVM footprints

+ Customization of the Memory chart based on the requirement analysis

+ Ability to perform the GC



Threads overview

- Tình trang các server thread và web app host tại Tomcat.

Note:

The following screenshot shows the live status of the threads utilization, and the Deadlock Detection button is highlighted. In real-time, this thread analysis tool is a very handy tool for the administrators.

- Các tính năng Threads tab:

+ Graphical presentation of threads and their picture

+ Individual thread analysis with their status

+ Deadlock detection



VM Summary and Overview

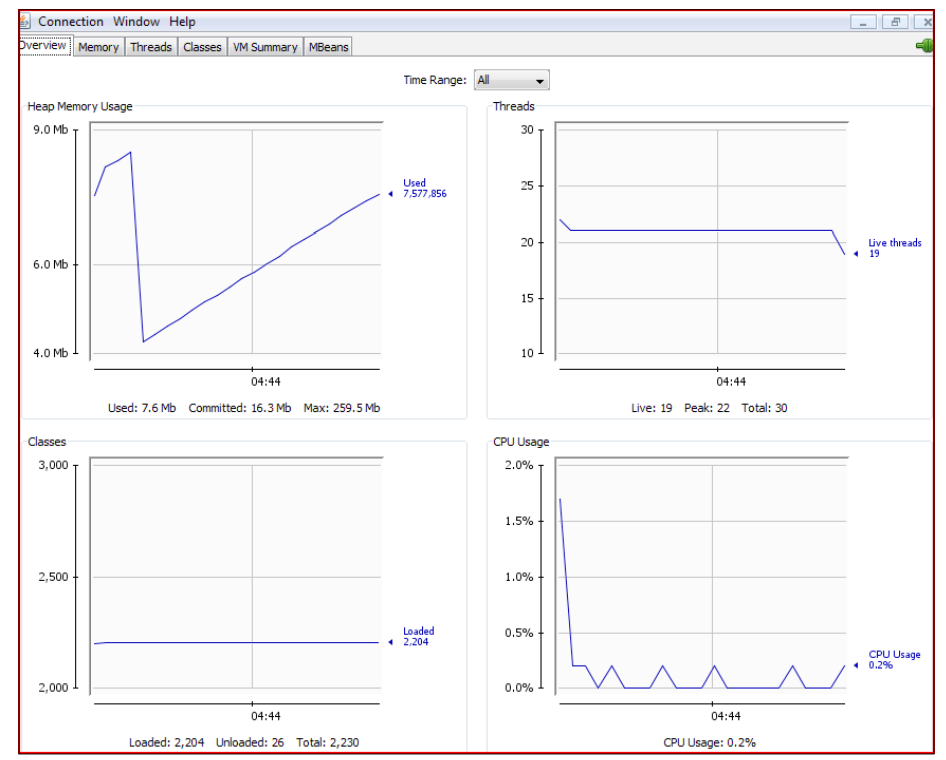
- Hiện thị hiệu năng tổng quan về các thành phần trong real time, tính năng rất hữu ích cho admin. Nếu có điều gì bất thường, có thể phát hiện nhanh và đi vào các thành phần để sửa chữa.

- Các tính năng:

+ Complete summary of the instances (Heap Memory Usage, Threads, CPU

Usage, Classes)

+ VM argument summary



MBeans-Tab này đưa ra Managed Beans (Mbeans) được triển khai trên Tomcat instance. Nó bao gồm Tomcat và app level Mbean.

- Các tính năng bao gồm:

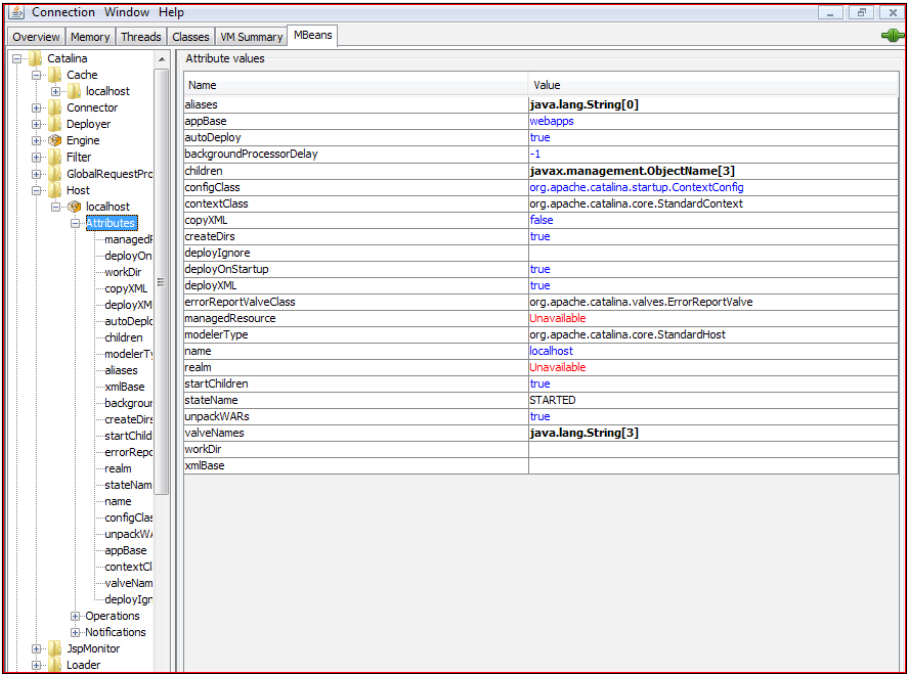
+ All parameters used in one tab

+ Easy-to-deploy, rollback, and invoke

+ We can create a user at the database level using MBeans

+ We can create notifications for events using MBeans

+ Configuration for resources can be done dynamically



Các loại Mbeans:

- Có 4 loại Mbeans.

• Standard MBeans: A standard MBean is a combination of an MBean interface and a class, where the interface defines the entire list of attributes and operations, while the class provides the functionalities for communication for a remote interface. It is one of the simplest MBeans.

• Dynamic MBeans: A dynamic MBean implements a separate interface (a specific method) and can be invoked at runtime.

• Open MBeans: It is a composition of Dynamic MBeans and the universal dataset used for manageability.

• Model MBeans: It is a composition of Dynamic MBeans with complete access to configurable parameters at runtime and self-described methods. This MBean requires classes.



Clustering in Tomcat 7

* Câu truyện mở đâu: có 2 team, A và B quản trị 2 hệ thống khác nhau, 1 ngày ô giám đốc bảo họ có 2 middleware env, và mỗi team phải quản trị 1 middleware, và họ làm theo các khác nhau. Đến thời gian tổng kết, team A duy trì thời gian up server khoảng 50% cho app và team B đến 99%, lý do team A dùng single server còn team B quyết định sử dụng kiến trúc HA cluster.
* Các phấn sẽ bàn luận:

+ High availability architecture and its advantages (HA và điểm mạnh)

+ Different types of high availability architectures including load balancing and clustering (Các loại kiến trúc HA bao gồm cân bằng tải và cluster)

+ Approaches used by IT industries while building a high availability architecture in an enterprise setup (Cách tiếp cận sử dụng khi xây dựng HA)

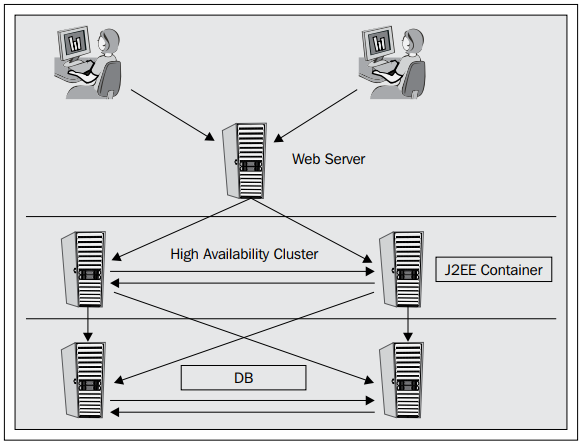
+ How to do Apache Tomcat clustering (HA trong tomcat)

+ Various clustering architectures (Các kiển trúc cluster)

+ How to solve common problems in clustering (vấn đề)

What is a cluster?

* 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)



Benefits of clustering

* 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.

Disadvantages of 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.

Clustering architecture

* 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:

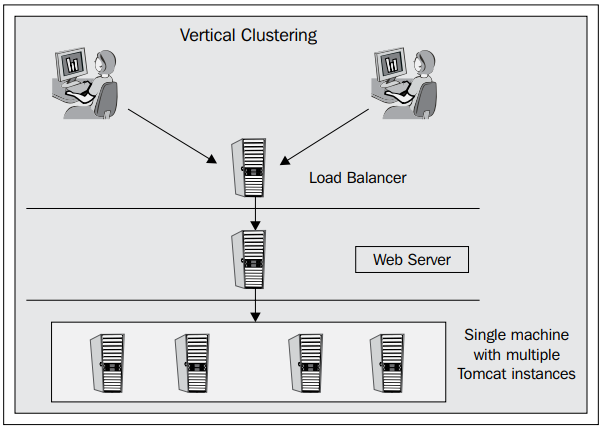
+ Vertical clustering (Thẳng đứng)

+ 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, ..



Advantages of vertical clustering

* Điểm mạnh của verical cluster:

+ 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.

Disadvantages of vertical clustering

+ không có khả năng chịu lỗi phần cứng

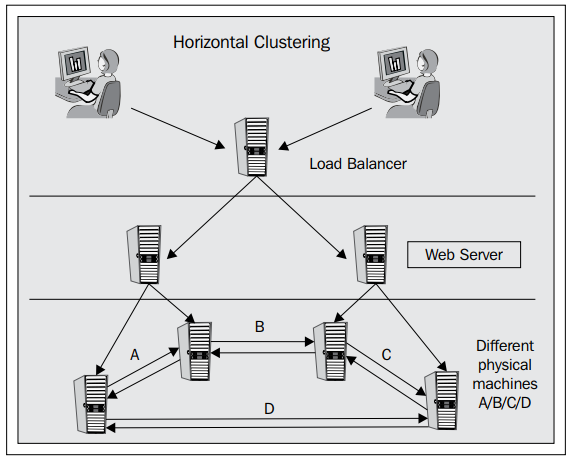
+ Nhiều vấn đề phải giải quyết

+ Phần cứng đủ cao để thực hiện

+ Giá thành cao.

Horizontal clustering

* 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.



Advantages of horizontal clustering

* Khả năng chịu lỗi cao
* Giá thành sẽ hạ xuống
* Vấn đề phát sinh ít hơn.

Disadvantages of 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.

Vertical clustering in Apache Tomcat 7

- Cho vertical cluster, cần 2 thực thi Apache Tomcat

- Các bước thực hiện:

Bước 1: Installation of the Tomcat instance.

Bước 2: Configuration of the cluster.

Bước 3: Apache HTTP web server configuration for the vertical cluster.

Installation of the Tomcat instance

Bước 1: down và unzip soft theo yêu cầu directory.

Bước 2: Cài đặt JDK, set Java\_home

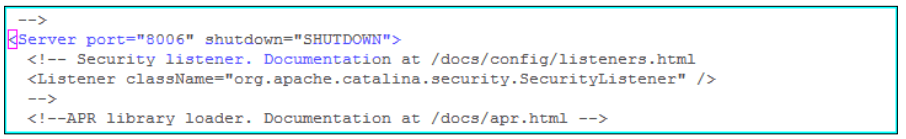
Bước 3: copy source code tomcat tới 2 thư mục khác nhau trong /opt/tomcat-X

Configuration of a vertical cluster

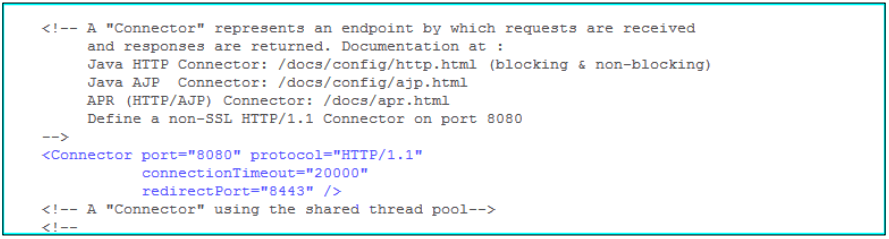
Configuration of instance 1

- Cấu hình node 1, ta sử dụng các cấu hình mặc định trong Connector, AJP, shutdown port trong server.xml.

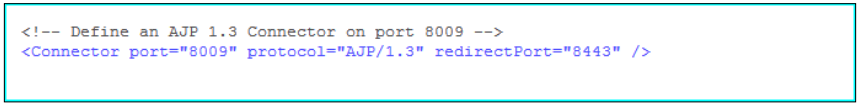
- Shutdown port: Nếu bỏ qua cấu hình nó, thực thi tomcat sẽ không thể start tomcat.



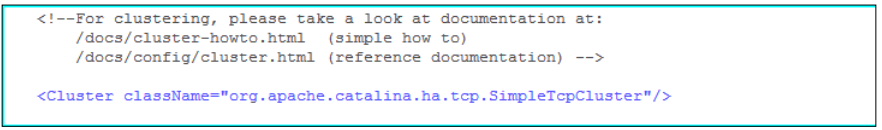
- Connector port: Port này sử dụng cho việc truy cập vào thực thi tomcat. VD: sử dụng localhost:8080, 8080 port được gọi là Connector port. Trong khi chạy nhiều thực thi, nếu bạn bỏ qua cấu hình port này, khi chạy tomcat sẽ xuất hiện lỗi Port đã sử dụng



- AJP port: Port này sử dụng cho kết nối AJP giữa Apache HTTP server với thực thi Tomcat. Khi chạy nhiều thực thi, bỏ qua cấu hình này sẽ sinh ra exception port đã sử dụng



- Cluster attributes: cho phép chạy dạng cluster



- Configuration test: chạy “configtest.sh” từ “TOMCAT\_HOME/bin” để kiểm tra lại cấu hình.

# ./configtest.sh



- Chạy thực thi 1 sử dụng script startup.sh

# ./startup.sh



- Kiểm tra thực thi

# ps -ef |grep java

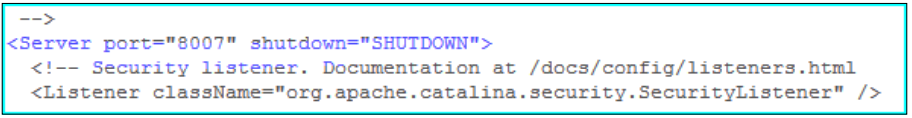


Configuration of instance 2

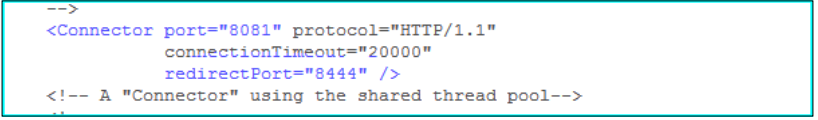
- Ta không thể sử dụng cấu hình mặc định tại node 2. Port sẽ bị xung đột

Cấu hình

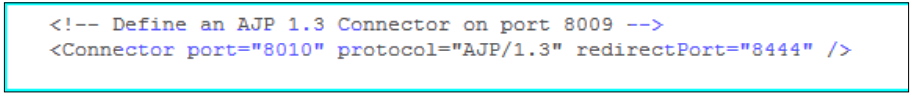
Bước 1: Thay đổi shutdown port cho thực thi 2, server.xml



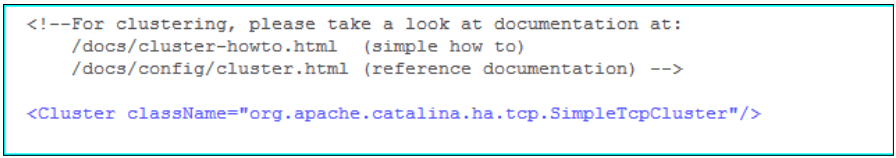
Bước 2: Thay đổi Connector và redirect port trong thực thi 2 (server.xml)



Bước 3: Thay đổi AJP và redirect port tại thực thi 2 server.xml.



Bước 4: Cho phép thuộc tính cluster để chạy cluster tại server.xml



Bước 5: Lưu cấu hình server.xml

Bước 6: Chạy “configtest.sh” script tại “TOMCAT\_HOME/bin”.

# ./configtest.sh

Bước 7: Chạy thực thi

# ./startup.sh

Bước 8: kiểm tra thực thi Tomcat

# ps -ef | grep java



Kiểm tra lại catalina.out:

- Kiểm tra lại log của cả 2 node:

+ 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:/127.0.0.1:4000

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

INFO: JvmRouteBinderValve started

Oct 11, 2011 5:00:37 PM org.apache.coyote.AbstractProtocolHandler

start

INFO: Starting ProtocolHandler ["http-bio-8080"]

Oct 11, 2011 5:00:37 PM org.apache.coyote.AbstractProtocolHandler

start

INFO: Starting ProtocolHandler ["ajp-bio-8009"]

Oct 11, 2011 5:00:37 PM org.apache.catalina.startup.Catalina start

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://{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 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.**

Apache web server configuration for vertical clustering

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

worker.list=tomcatnode1, tomcatnode2, loadbalancer

- Định nghĩa worker.list cho tất cả node trong cluster:

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

Note:

- tomcatnode1, tomcatnode2 sử dụng để định danh phân cụm

- Cấu hình 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 trong phần cấu hình chính httpd

# vi httpd.conf

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

Horizontal clustering in Apache Tomcat 7

- 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.

- 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)

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

Bước 1: Cài đặt thực thi Tomcat

Bước 2: Cấu hình cluster

Bước 3: Apache HTTP web configuration cho horizontal cluster.

Installation of the Tomcat instance

- Xem lại ở phần trước

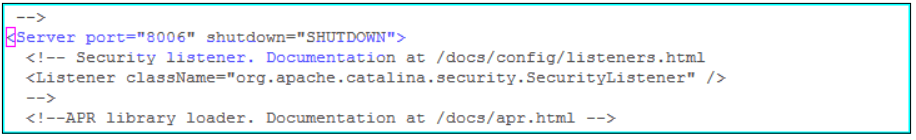
Configuration of the cluster

- 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.

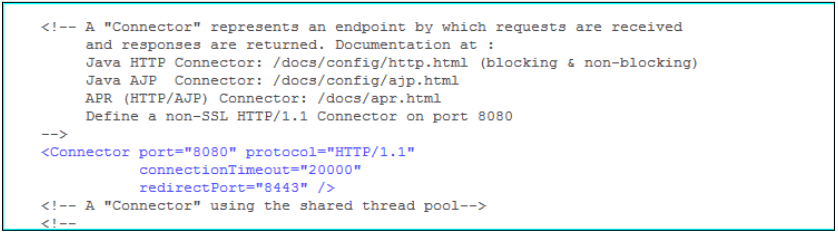
Configuration of instance 1

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

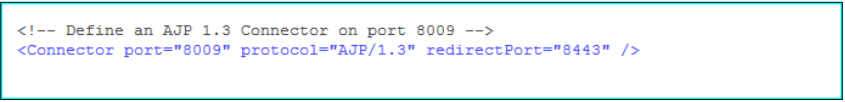
- Shutdown port: Trong khi chạy nhiều thực thi, nếu ta bỏ qua việc cấu hình port, thực thi Tomcat sẽ không thể khởi tạo.



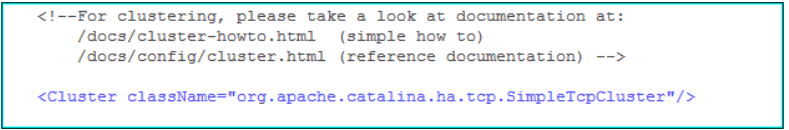
- Connector port: Port này sử dụng để truy cập vào thực thi tomcat, cơ bản ta truy cập vào thực thi Tomcat bằng cách sử dụng “<http://localhost:8080>”. Port 8080 gọi là connector port. Trong khi chạy nhiều thực thi, nếu bạn bỏ qua việc cấu hình port này, tomcat sẽ không thể chạy với thông bảo “Port đã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



- Cluster attributes: Cho phép cluster attributes trong server.xml.



- 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"/>

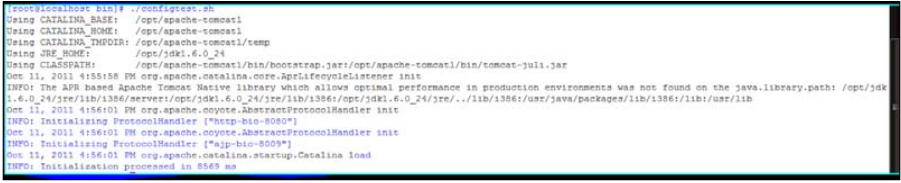
</Cluster>

- first highlighted code section hiện thị multicast IP. Multicast tạo sự thay đôi kết nối giữa 2 thực thi.

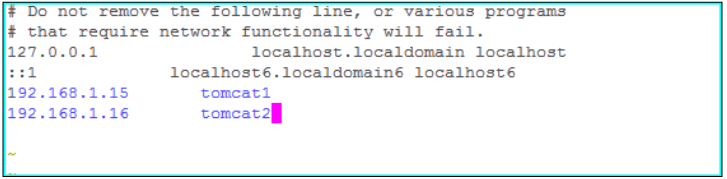
- The second highlighted section hiện deployment properties cho cluster instances.

Bước 5: Chạy “configtest.sh” trong “TOMCAT\_HOME/bin”.

# ./configtest.sh



Bước 6: Host entry: Thêm IP trong host files (/etc/hosts).



Bước 7: Khởi tạo thực thi tomcat. Chạy thực thi 1 tomcat.

# ./startup.sh



- Kiểm tra thực thi tomcat

# ps -ef | grep java



Configuration of instance 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

Apache web server configuration for horizontal clustering

- sau kho cấu hình xong cluster trên tomcat, ta tích hợp Apache server cho tomcat 7.

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 tomcatnode 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 thực thi 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 crashes

- 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 thực thi 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.

Monitoring of Tomcat clustering- Giám sát cluster tomcat thông qua:

+ Công cụ giám sát

+ Scripts

+ Manual

Giám sát cơ bản:

Bước 1: Kiểm tra tiền trình Tomcat sử dụng:

# ps -ef |grep java

Bước 2: kiểm tra log, xác nhận kết nối trên cluster.

Bước 3: Xác nhận các URL tới các node member.

e