Giới thiệu về Tomcat

Tổng quan Apache Tomcat

* Apache Tomcat là một Java Servlet được phát triển bởi Apache Software Foundation. Tomcat thi hành các ứng dụng Java Servlet và JavaServer Pages (JSP). Tomcat cung cấp một máy chủ HTTP cho ngôn ngữ Java thuần túy. Apache Tomcat là một phần mềm mã nguồn mở để chạy các ứng dụng web có nền tảng là ngôn ngữ Java. Tomcat được hỗ trợ và tăng cường bởi một nhóm tình nguyện viên từ cộng đồng mã nguồn mở của Java.
* Apache Tomcat rất ổn định và có tất cả các tính năng của một ứng dụng web thương mại nhưng đi kèm theo giấy phép mã nguồn mở của Apache. Tomcat cũng cung cấp một số chức năng bổ sung như tomcat manager application, speciallized realm imlementation và tomcat valves. Các phiên bản của apache tomcat trùng với phiên bản và đặc điểm kỹ thuật của servlet java hoặc java servlet API. Tomcat 5.5X hỗ trợ Servlet API 2.3, tomcat 6.0X hỗ trợ servlet API 2.4 và tomcat 7.0 hỗ trợ servlet API 3.0. Ngoài Servlet versions API, phiên bản tomcat hỗ trợ phiên bản JSP API tương ứng.
* Apache Tomcat có các tính năng như chứng thực người dùng, virtual hosting, CGI, Java Servlet, SSI, hỗ trợ Ipv6, …
* Apache Tomcat hỗ trợ các hệ điều hành như windows, linux, OS, BSD, solaris,…

Các thuật ngữ cần nhớ:

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

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

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

Tổ chức thư mục tomcat:

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

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

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

- “/webapp” = Nơi các web được host.

Cài đặt Tomcat:

Yêu cầu trước khi cài đặt:

Trên Linux:

- Trước khi cài đặt Tomcat, ta cần cài đặt môi trường Java cho việc chạy thực thi Tomcat.

Tomcat Web Application Deployment

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

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

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

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

Tomcat Manager

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

A word on Contexts

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

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

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

Context Descriptor được lưu tại:

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

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

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

Deployment on Tomcat startup

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

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

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

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

Quá trình triển khai khi Tomcat khởi động:

+ Bất kỳ Context Descriptor sẽ được triển khai trước

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

+ Các file WAR được deploy.

Deploying on a running Tomcat server

- Có thể deploy app trong khi Tomcat đang chạy

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

VD: copy war file vào “appBase”.

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

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

+ Deployment of .WAR files copied into the Host appBase.

+ Deployment of exploded web applications which are copied into the Host appBase.

+ Re-deployment of a web application which has already been deployed from a .WAR when the new .WAR is provided. In this case the exploded web application is removed, and the .WAR is expanded again. Note that the explosion will not occur if the Host is configured so that .WARs are not exploded with a unpackWARs attribute set to "false", in which case the web application will be simply redeployed as a compressed archive.

+ Re-loading of a web application if the /WEB-INF/web.xml file (or any other resource defined as a WatchedResource) is updated.

+ Re-deployment of a web application if the Context Descriptor file from which the web application has been deployed is updated.

+ Re-deployment of dependent web applications if the global or per-host Context Descriptor file used by the web application is updated.

+ Re-deployment of a web application if a Context Descriptor file (with a filename corresponding to the Context path of the previously deployed web application) is added to the $CATALINA\_BASE/conf/[enginename]/[hostname]/ directory.

+ Undeployment of a web application if its document base (docBase) is deleted. Note that on Windows, this assumes that anti-locking features (see Context configuration) are enabled, otherwise it is not possible to delete the resources of a running web application.

Note:

- web application reloading cũng có thể được cấu hình lại trong loader, trong trường hợp classes cần được thay đổi.

Deploying using the Tomcat Manager

Manager App HOW-TO

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

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

+ Deploy a new web application from the uploaded contents of a WAR file.

+ Deploy a new web application, on a specified context path, from the server file system.

+ List the currently deployed web applications, as well as the sessions that are currently active for those web apps.

+Reload an existing web application, to reflect changes in the contents of /WEB-INF/classes or /WEB-INF/lib.

+ List the OS and JVM property values.

+ List the available global JNDI resources, for use in deployment tools that are preparing <ResourceLink> elements nested in a <Context> deployment description.

+ Start a stopped application (thus making it available again).

+ Stop an existing application (so that it becomes unavailable), but do not undeploy it.

+ Undeploy a deployed web application and delete its document base directory (unless it was deployed from file system).

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

VD:

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

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

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

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

</Context>

Configuring Manager Application Access

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

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

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

You can find the role names in the web.xml file of the Manager web application. The available roles are:

+ manager-gui — Access to the HTML interface.

+ manager-status — Access to the "Server Status" page only.

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

+ manager-jmx — Access to JMX proxy interface and to the "Server Status" page.

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

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

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

Realm

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

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

Security Manager HOW-TO

Background

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

VD: 1 người dùng được chúng thực xuất bản 1 JSPs trên site và vô tình thêm vào 1 đoạn trong their JSP:

<% System.exit(1); %>

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

“WARNING”

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

Permissions

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

Standard Permissions

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

Tóm tắt các quyền thường dùng:

* **java.util.PropertyPermission** - Controls read/write access to JVM properties such as java.home.
* **java.lang.RuntimePermission** - Controls use of some System/Runtime functions like exit() and exec(). Also control the package access/definition.
* **java.io.FilePermission** - Controls read/write/execute access to files and directories.
* **java.net.SocketPermission** - Controls use of network sockets.
* **java.net.NetPermission** - Controls use of multicast network connections.
* **java.lang.reflect.ReflectPermission** - Controls use of reflection to do class introspection.
* **java.security.SecurityPermission** - Controls access to Security methods.
* **java.security.AllPermission** - Allows access to all permissions, just as if you were running Tomcat without a SecurityManager.

Configuring Tomcat With A SecurityManager

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

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

// Example policy file entry

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

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

};

Note:

- The signedBy and codeBase entries are optional when granting permissions

- The codeBase is in the form of a URL, and for a file URL can use the ${java.home} and ${catalina.home} properties (which are expanded out to the directory paths defined for them by the JAVA\_HOME, CATALINA\_HOME and CATALINA\_BASE environment variables).

The Default Policy File

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

Starting Tomcat With A SecurityManager

- Chạy SecurityManager với option "-security".

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

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

Permissions for packed WAR files

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

VD: To assign permissions to the entire web application the entry in the policy file would look like this:

// Example policy file entry

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

...

};

VD: To assign permissions to a single JAR within the web application the entry in the policy file would look like this:

// Example policy file entry

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

...

};

Configuring Package Protection in Tomcat

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

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

Class Loader HOW-TO

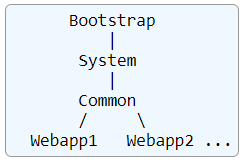
- Như các server app, Tomcat cài đặt chứa nhiều class loader (các class thực thi bởi java.lang.ClassLoader) để cho phép các phần khác nhau trên Tomcat webserver, và cũng như các web app chạy, các truy cập vào các kho chứa khác nhau có sẵn trong class và resource.

Note:

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

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

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



Class Loader Definitions:

- Bảng trên mô tả quá trình load classer:

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

Note:

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

+ System – Class loader cơ bản khởi tạo các content của CLASSPATH env variable. Tuy nhiên, Tomcat startup scripts ($CATALINA\_HOME/bin/catalina.sh) hoàn toàn bỏ qua các contents trong “CLASSPATH” env variable, thay thể xây dựng System class loader từ kho chứa riêng

$CATALINA\_HOME/bin/bootstrap.jar — Contains the main() method that is used to initialize the Tomcat server, and the class loader implementation classes it depends on.

$CATALINA\_BASE/bin/tomcat-juli.jar or $CATALINA\_HOME/bin/tomcat-juli.jar — Logging implementation classes. These include enhancement classes to java.util.logging API, known as Tomcat JULI, and a package-renamed copy of Apache Commons Logging library used internally by Tomcat. See logging documentation for more details.

If tomcat-juli.jar is present in $CATALINA\_BASE/bin, it is used instead of the one in $CATALINA\_HOME/bin. It is useful in certain logging configurations

$CATALINA\_HOME/bin/commons-daemon.jar — The classes from Apache Commons Daemon project. This JAR file is not present in the CLASSPATH built by catalina.bat|.sh scripts, but is referenced from the manifest file of bootstrap.jar.

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

++ Cơ bản, applocation class không được lưu tại đây, vị trí các class loader được định nghĩa bởi common.loader property “$CATALINA\_BASE/conf/catalina.properties”. Setting mặc đinh sẽ tìm kiểm các vị trí được cung cấp:

unpacked classes and resources in $CATALINA\_BASE/lib

JAR files in $CATALINA\_BASE/lib

unpacked classes and resources in $CATALINA\_HOME/lib

JAR files in $CATALINA\_HOME/lib

++ Mặc đinh bao gồm:

annotations-api.jar — JavaEE annotations classes.

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

catalina-ant.jar — Tomcat Catalina Ant tasks.

catalina-ha.jar — High availability package.

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

catalina-tribes.jar — Group communication package.

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

el-api.jar — EL 3.0 API.

jasper.jar — Tomcat Jasper JSP Compiler and Runtime.

jasper-el.jar — Tomcat Jasper EL implementation.

jsp-api.jar — JSP 2.3 API.

servlet-api.jar — Servlet 3.1 API.

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

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

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

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

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

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

tomcat-websocket.jar — WebSocket 1.1 implementation

websocket-api.jar — WebSocket 1.1 API

+ WebappX: Class loader được tạo cho mỗi web app (đã được deploy lên tomcat). Tất cả unpacked class và resource trong “/WEB-INF/classes”, các class và tài nguyền tâm vào dạng JAR file trong “/WEB-INF/lib”, khiến các class này có sẵn cho riêng web app đó.

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

Bootstrap classes of your JVM

+ /WEB-INF/classes of your web application

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

+ System class loader classes (described above)

+ Common class loader classes (described above)

Nếu web app load config <Loader delegate="true"/>, quá trình load sẽ là:

+ Bootstrap classes of your JVM

+ System class loader classes (described above)

+ Common class loader classes (described above)

+ /WEB-INF/classes of your web application

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

Jasper

- Jasper là công cụ JSP của Tomcat. Tomcat 5.x sử dụng Jasper 2, chính là một thực thi các trang JSP được chỉ rõ của Sun Microsystems. Jasper phân tích các file JSP để biên dịch chúng trong code Java như là các servlets (có thể được điều khiển bởi Catalina). Tại thời điểm thực thi, Jasper có khả năng tự động dò ra các file JSP và biên dịch chúng.

Jasper 2 JSP Engine How To

Introduction

- Tomcat 8.5 uses the Jasper 2 JSP Engine to implement the JavaServer Pages 2.3 specification.

Jasper 2

- Từ Jasper sang Jasper 2, có những đặc điểm quan trọng được thêm vào:

+ Cụm các thư viện JSP dùng chung - Mỗi cụm đặt trong các file JSP được điều khiển bởi một lớp điều khiển các tag. Các đối tượng lớp điều khiển các cụm có thể được dùng chung hoặc tái sử dụng lại trong toàn bộ JSP servlet.

+ Biên dịch JSP - Khi việc dịch lại sửa đổi Java code, các phiên bản cũ vẫn có hiệu lực cho yêu cầu máy chủ. Bản JSP servlet cũ nhất bị xóa đi một khi bản JSP servlet mới được biên dịch lại.

+ Dịch lại JSP khi các trang có sự thay đổi - Các trang cso thể được chèn và bao gồm trong một file JSP tại thời điểm biên dịch. JSP sẽ không chỉ được tự động dịch lại với các sự thay đổi các file JSP mà còn bao gồm sự thay đổi trang.

+ Trình biên dịch JDT Java - Jasper 2 có thể sử dụng trình biên dịch JDT Java là Eclipse thay cho Ant và javac.

SSL/TLS for Tomcat.

Proxy Support HOW-TO

- Apache proxy

Clustering/Session Replication HOW-TO

Simply add

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

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

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

1 số giá trị quan trọng mặc định:

- Multicast address is 228.0.0.4

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

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

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

- Listener is configured ClusterSessionListener

- Two interceptors are configured TcpFailureDetector and MessageDispatchInterceptor

The following is the default cluster configuration:

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

channelSendOptions="8">

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

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"/>

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

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

address="228.0.0.4"

port="45564"

frequency="500"

dropTime="3000"/>

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

address="auto"

port="4000"

autoBind="100"

selectorTimeout="5000"

maxThreads="6"/>

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

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

</Sender>

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

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

</Channel>

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

filter=""/>

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

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

tempDir="/tmp/war-temp/"

deployDir="/tmp/war-deploy/"

watchDir="/tmp/war-listen/"

watchEnabled="false"/>

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

</Cluster>

Security

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

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

private LAN

a Virtual Private Network (VPN)

IPSEC

Cluster Basics

- Để chạy việc nhân bản session trong Tomcat 8, các bước bao gồm:

+ All your session attributes must implement java.io.Serializable

+ Uncomment the Cluster element in server.xml

+ If you have defined custom cluster valves, make sure you have the ReplicationValve defined as well under the Cluster element in server.xml

+ If your Tomcat instances are running on the same machine, make sure the Receiver.port attribute is unique for each instance, in most cases Tomcat is smart enough to resolve this on it's own by autodetecting available ports in the range 4000-4100

+ Make sure your web.xml has the <distributable/> element

+ If you are using mod\_jk, make sure that jvmRoute attribute is set at your Engine <Engine name="Catalina" jvmRoute="node01" > and that the jvmRoute attribute value matches your worker name in workers.properties

+ Make sure that all nodes have the same time and sync with NTP service!

+ Make sure that your loadbalancer is configured for sticky session mode.

Note:

+ Remember that your session state is tracked by a cookie, so your URL must look the same from the out side otherwise, a new session will be created.

+ Clustering support currently requires the JDK version 1.5 or later.

Overview

- Để cho phép nhân rộng các session trong tomcat, 3 cách khác nhau:

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

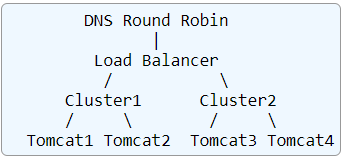
+ Sử dụng session persistence (kéo dài phiên), lưu session tới shared database (PersistenceManager + FileStore)

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

- Khi xuất hiện session replication, Tomcat có thể thực hiện nhân rộng các sesstion state all to all bằng các sử dụng “DeltaManager” hoặc thực hiện backup replication tới node sử dụng “BackupManager”. All to all replication là thuật toán chỉ hiệu quá khi cluster nhỏ. Đối với các larger cluster lớn, sử dụng primary-secondary session replication, ở đó session chỉ được lưu duy nhất tại 1 backup server cài đặt BackupManager.

- Hiện tại, ta có thể sử dụng domain worker attributre (mod\_jk > 1.2.8) để xây dựng phân vùng cluster với khả năng giải pháp mở rộng cluster với DeltaManager (Cần config domain interceptor cho nó). Để giữ, giảm lưu lượng mạng xuống trong môi trường all to all env, ta có thể chia tách cluster thành nhưng group nhỏ. Điều này có thể đạt được bằng cách sử dụng nhưng multicast address khác nhau cho mỗi group.

- Mô hình:



- Điều quan trọng cần đề cập tới là, session replication chỉ bắt đầu việc clustering. Một số khái niệm phố biến cho cluster như farming, .. Ta có thẻ deploy app trên 1 server, cluster sẽ phân phói deployment tới tất cả cluster. Đây là khả năng đến từ “FarmWarDeployer” (vd server.xml)

Cluster Information

- Membership được thiết lập bằng cách sử dụng multi heartbeats. Vì thế, nếu bạn mong muốn phân chia cluster, bạn cần thay đổi multicast IP address hoặc port trong thành phần <Membership>.

- heartbeat bao gồm IP address của Tomcat node và TCP port mà Tomcat lắng nghe cho sự nhân rộng traffic. Tất cả dữ liệu kết nối xảy ra thông qua TCP.

- “ReplicationValve” được sử dụng để tìm ra khi nào request đã hoàn thành và bắt đầu sự nhân rộng, nếu có. Data chỉ được sao chép nếu session được thay đổi (by calling setAttribute or removeAttribute on the session).

- 1 trong những phần quan trọng về hiệu năng cần được cân nhắc là sự động bộ, bất đồng bộ. Trạng thái động bộ replication, request sẽ không trả lại cho đến khi replicated session đã được gửi qua toàn bộ và tái lưu trữ trên tất cả node cluster. Synchronous vs. asynchronous cấu hình sử dụng “channelSendOptions” flage và nó là giá trị interger. Mặc định giá trị là “SimpleTcpCluster/DeltaManager” combo 8, nó là bắt đồng bộ. Trong suốt async replication, request được trả lại trước khi data được nhân rộng. async replication thời gian thực thi sẽ ngắn hơn và sys replication đảm bảo session được replicated trước khi request return.

Bind session after crash to failover node

- Nếu bạn sử dụng mod\_jk và không sử dụng sticky session hoặc vì 1 số lý do sticky session không hoạt động, hoặc đơn giản xảy ra lỗi, session id sẽ cần chỉnh sửa vì nó bao gồm worker id của tomcat trước (as defined by jvmRoute in the Engine element). Để giải quyết vấn đề này, ta sử dụng JvmRouteBinderValve.

- JvmRouteBinderValve ghi lại session id để chắc chắn request tiếp theo sẽ vẫn còn sticky (and not fall back to go to random nodes since the worker is no longer available) sau khi fail over. Giá trị valve rewrites JSESSIONID trong cookie có cùng tên. Không đặt valve chính xác, sẽ khiên nó khó trong việc chắc chắn stickness trong trường hợp thất bại mod\_jk module.

- Nhớ rằng, khi thêm giá trị mới vào server.xml thì các giá trị mặc định sẽ không còn chính xác nữa. Bảo đảm các giá trị custom thêm vào thích hợp.

Hint:

-Với trường “sessionIdAttribute”, ta có thể thay đổi tên request attribute mà được bao gồm trong old session id. Mặc định attribute name là:

org.apache.catalina.ha.session.JvmRouteOrignalSessionID.

Trick:

-Ta có thể cho phép mod\_jk turnover mode thông qua JMX trước khi ta drop node tới tất cả backup nodes. Set enable true trên tất cả “JvmRouteBinderValve” backups, disable worker tại mod\_jk và sau restart nó. Tiếp theo, enable mod\_jk worker và disable “JvmRouteBinderValves” lại. Sử dụng cách này để di chuyển phiên làm việc.

Configuration Example

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

channelSendOptions="6">

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

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"

mapSendOptions="6"/>

<!--

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

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"/>

-->

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

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

address="228.0.0.4"

port="45564"

frequency="500"

dropTime="3000"/>

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

address="auto"

port="5000"

selectorTimeout="100"

maxThreads="6"/>

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

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

</Sender>

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

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

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

</Channel>

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

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

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

tempDir="/tmp/war-temp/"

deployDir="/tmp/war-deploy/"

watchDir="/tmp/war-listen/"

watchEnabled="false"/>

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

</Cluster>

-Trong main element (<cluster>, tất cả giá trị của cluster được cài đặt. “channelSendOptions” là flag gắn liên với mỗi message gửi đi bởi SimpleTcpCluster class hoặc bất kỳ đối được nào được thêm từ SimpleTcpCluster.send methods. Mô tả của việc gửi flags có sẵn trong docs. DeltaManager gửi thông tin sử dụng SimpleTcpCluster method, trong khi backup manager gửi trực tiếp thông tin qua kênh.

Link <http://tomcat.apache.org/tomcat-8.5-doc/api/org/apache/catalina/tribes/Channel.html>

Note:

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

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"

mapSendOptions="6"/>

-Template cho manager configuration sẽ được sử dụng nếu không có manager định nghĩa trong <Context> element. Trong tomcat 5.x, mỗi webapp được đánh dấu, phân bố có cùng manager. Ở phiên bản hiện tại, có thể định nghĩa manager class cho mỗi webapp, từ đó, ta có thể trộn nhiều manager trong cluster. Tất nhiên, 1 manager trên 1 node app được kết nối tới cùng manager trên 1 app khác tại node khác.

- Nếu không chỉ định manager cho webapp, nó sẽ được đánh giấu <distributable/> và Tomcat sẽ tạo thực thi manager clonning cho config.

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

-Channel element sử dụng Apache Tribes module, module này phục vị cho kết nối Tomcat cluster.

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

- Membership được hoàn thành với multicasting. Đồng thời Tribes cũng hỗ trợ static membership bằng cách sử dụng “StaticMembershipInterceptor”.

- The address attribute is the multicast address used and the port is the multicast port

- membership broadbast TCP address/port của chính nó tới các node khác, vì thế, kết nối giữa các note có thể hoàn thành bằng các sử dụng TCP.

Thành phần gửi nhận:

- Để ý trường “Receiver.address”. Tribes gửi và nhận data thông qua 2 thành phần chính.

- Receiver, as the name suggests is responsible for receiving messages, there is a thread pool in this component that has a maxThreads and minThreads setting.

- Giá trị addres là host address sẽ được broadbast bởi thành phần membership tới các node khác.

- Sender là thành phần gửi thông tin tới các node khác. Sender có shell component, “ReplicationTransmitter” but the real stuff done is done in the sub component, Transport.

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

- Đồng thời nghĩ là 1 message được nhiều sender gửi và song song nghĩa là tất cả message cùng gửi đi tại cùng thời điểm

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

address="228.0.0.4"

port="45564"

frequency="500"

dropTime="3000"/>

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

address="auto"

port="5000"

selectorTimeout="100"

maxThreads="6"/>

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

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

</Sender>

-Tribes sử dụng stack gửi message. Mỗi element trong stack gọi là interceptor. Sử dụng interceptor, logic có thể được phân nhỏ thành code cho việc quản lý. Interceptor config bao gồm:

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

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

+ ThroughputInterceptor - prints out simple stats on message traffic.

Note:

- Cung cấp các interceptor khó quan trọng, cách định nghĩa server.xml là cách chúng được sắp xếp trong stack.

.

- Cluster sử dụng các giá trị dưới để kiểm sotats request tới web app. Clusster element không phải là 1 phần trong pipeline Tomcat, thay vì thế, cluster sẽ thêm giá trị tới parent container.

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

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

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

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

tempDir="/tmp/war-temp/"

deployDir="/tmp/war-deploy/"

watchDir="/tmp/war-listen/"

watchEnabled="false"/>

- Vì “SimpleTcpCluster” sử dụng sender và receiver thông qua Channel object, thành phần này có thể đăng ký chính nó như listener tại “SimpleTcpCluster”

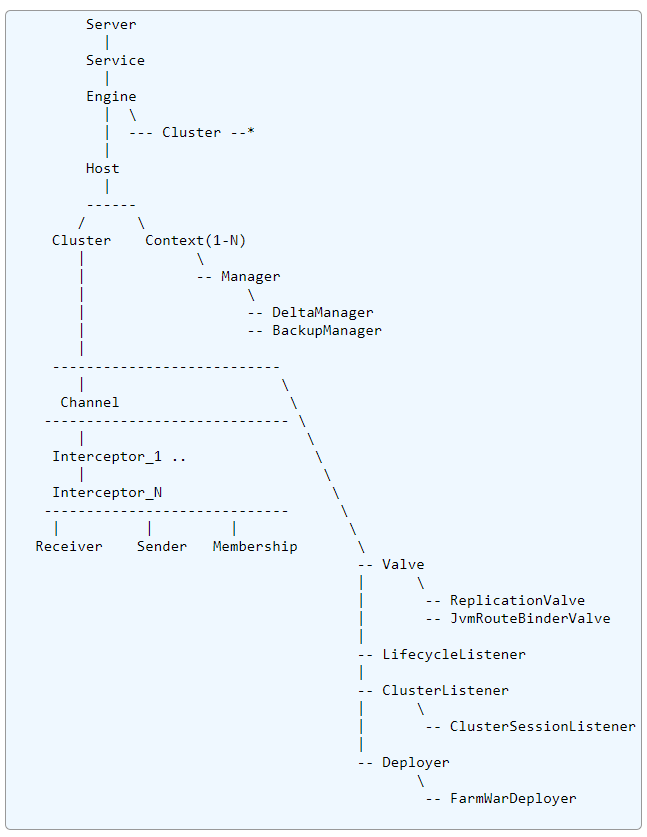
- The listener above “ClusterSessionListener” listens for DeltaManager replication messages and applies the deltas to the manager that in turn applies it to the session.

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

</Cluster>

Cluster Architecture

- Tổ chức trong cluster.



How it Works

- Kịch bản Tomcat:

TomcatA starts up

TomcatB starts up (Wait that TomcatA start is complete)

TomcatA receives a request, a session S1 is created.

TomcatA crashes

TomcatB receives a request for session S1

TomcatA starts up

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

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

TomcatA The session S2 expires due to inactivity.

Tomcat A start up

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

TomcatB starts up

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

Note:

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

TomcatA receives a request, a session S1 is created.

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

TomcatA crashes

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

TomcatB receives a request for session S1

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

TomcatA starts up

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

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

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

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

- Quay lại bước 3

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

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

Membership

- Clustering membership được thiết lập bằng cách sử dụng các simple multicast pings. Mỗi thực thi Tomcat sẽ định kì gửi multicast ping. Trong ping message, thực thi sẽ broadcast IP và TCP listen port cho sự nhân bản. Nếu thực thì không nhận được ping với khung thời gian, member sẽ được xem là đã chết. Đơn giản và hiệu quả.

TCP Replication

- Khi multicast ping được nhận, member sẽ được thêm vào cluster khi yêu cầu được nhân bản, gửi thực thi sẽ sử dụng host và port info cùng với thiết lập TCP socket. Sử dụng socket đó cho việc gửi over serialized data. Lý do chọn TCP socket vì nó được xây dựng với control folow đảm bảo việc gửi nhận.

Distributed locking and pages using frames

- Tomcat không giữa các thực thi session trong sync across cluster. Việc thực thi các logic sẽ được xử lý overhead và sẽ xảy ra 1 số lỗi. Nếu client truy cập vào cùng session cùng lúc sử dụng multiple request, request cuối cùng sẽ buh overrive tới cluster

List of Cluster Mbeans

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **MBean ObjectName - Engine** | **MBean ObjectName - Host** |
| Cluster | The complete cluster element | type=Cluster | type=Cluster,host=${HOST} |
| DeltaManager | This manager control the sessions and handle session replication | type=Manager,context=${APP.CONTEXT.PATH}, host=${HOST} | type=Manager,context=${APP.CONTEXT.PATH}, host=${HOST} |
| FarmWarDeployer | Manages the process of deploying an application to all nodes in the cluster | Not supported | type=Cluster, host=${HOST}, component=deployer |
| Member | Represents a node in the cluster | type=Cluster, component=member, name=${NODE\_NAME} | type=Cluster, host=${HOST}, component=member, name=${NODE\_NAME} |
| ReplicationValve | This valve control the replication to the backup nodes | type=Valve,name=ReplicationValve | type=Valve,name=ReplicationValve,host=${HOST} |
| JvmRouteBinderValve | This is a cluster fallback valve to change the Session ID to the current tomcat jvmroute. | type=Valve,name=JvmRouteBinderValve, context=${APP.CONTEXT.PATH} | type=Valve,name=JvmRouteBinderValve,host=${HOST}, context=${APP.CONTEXT.PATH} |

E