Enterprise 환경에서 Tomcat 운영하기

Part 1 : Tomcat 잘 알고 사용하기

Victor Lee
sleepred@gmail.com

2015.12.29

Contents

- **Tomcat 개요**
- **Tomcat Architecture**
- **Basic Features**
  - Configuration, Connector, Tomcat JDBC Connection Pool 등
- **Advance Features**
  - 웹서버 연동, Session Clustering 등
- **Management & Monitoring**
  - Tomcat Manager, JK Manager, VisualVM, Scouter 등
- **Tomcat Troubleshooting & 운영 노하우**
- **Tomcat Community 소개**
  - Build, Tomcat 확장하기, Contribution

Tomcat 개요

Why OSS?

WebSphere®

JEUS® Web Application Server

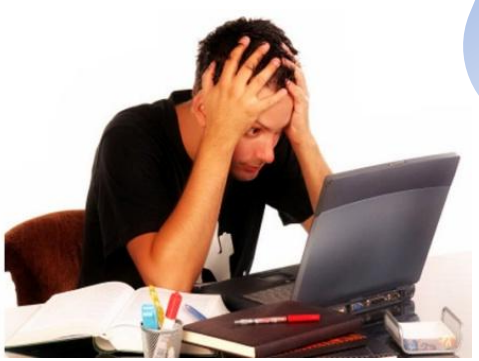
ORACLE WebLogic

JBoss® by Red Hat

Complexity

Vendor Dependency

Cost Overrun



Solution?

open source

Why OSS?

WildFly

GlassFish

Apache Tomcat

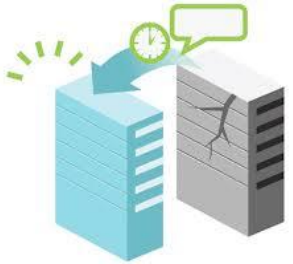
Apache

jetty://

Solution?

Stability

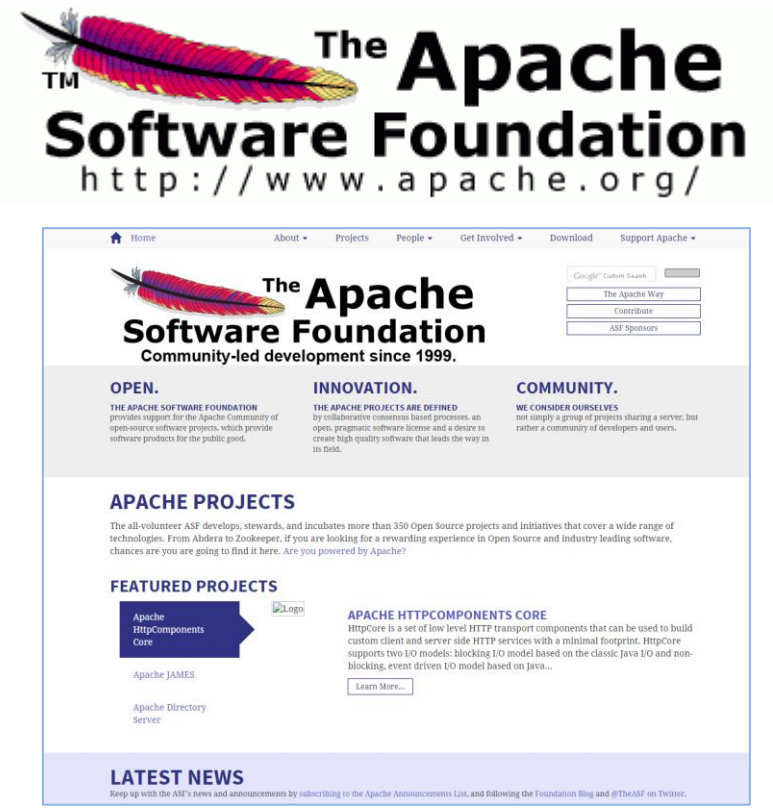
Security



Availability

Why Tomcat?

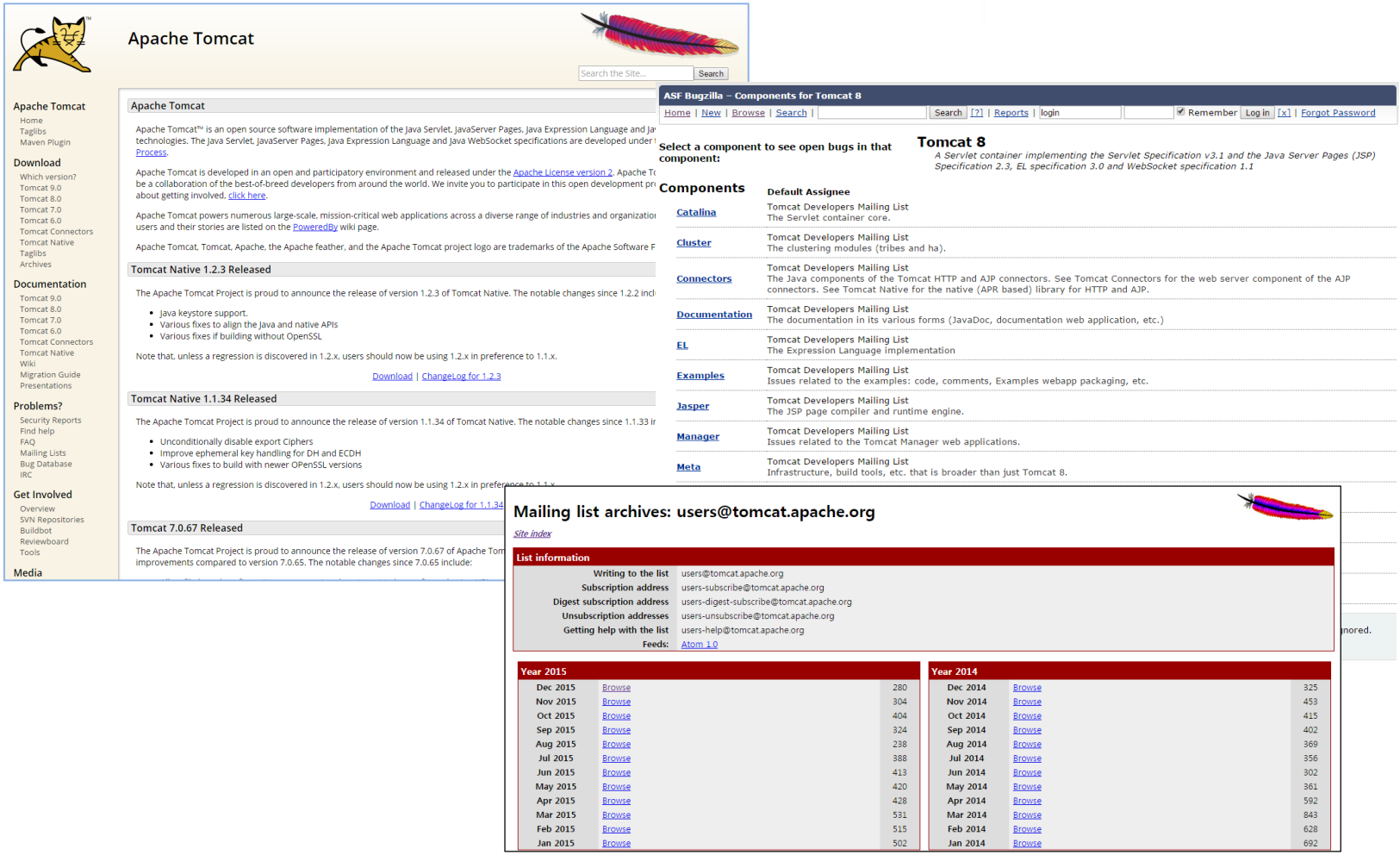
Top Open Source Communities



	PROJECTS	COMMITTERS
APACHE	323	1445
ECLIPSE	171	868
MOZILLA	77	1491
OPEN MRS	39	94
OUTERCURVE	31	191
OPENSTACK	13	723
LINUX KERNEL	1	3,100

※ ASF : http://www.apache.org/

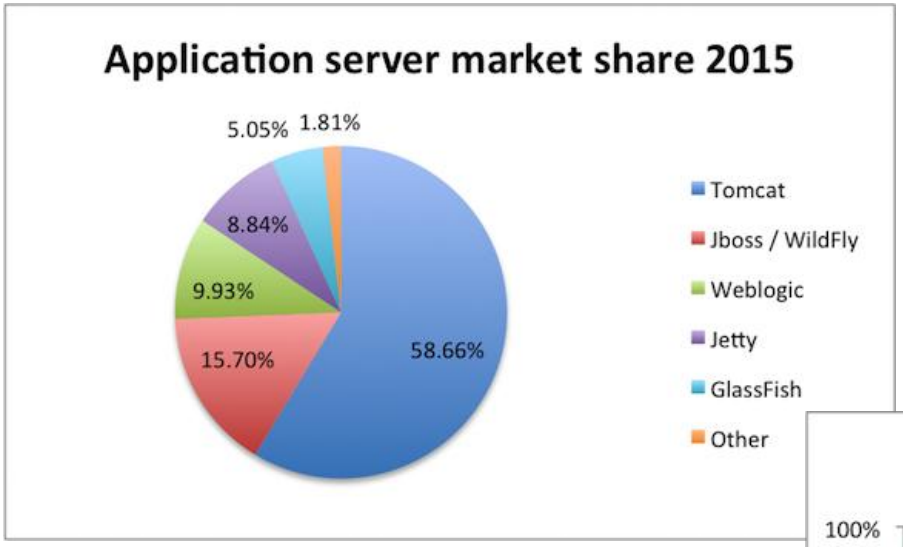
Why Tomcat?

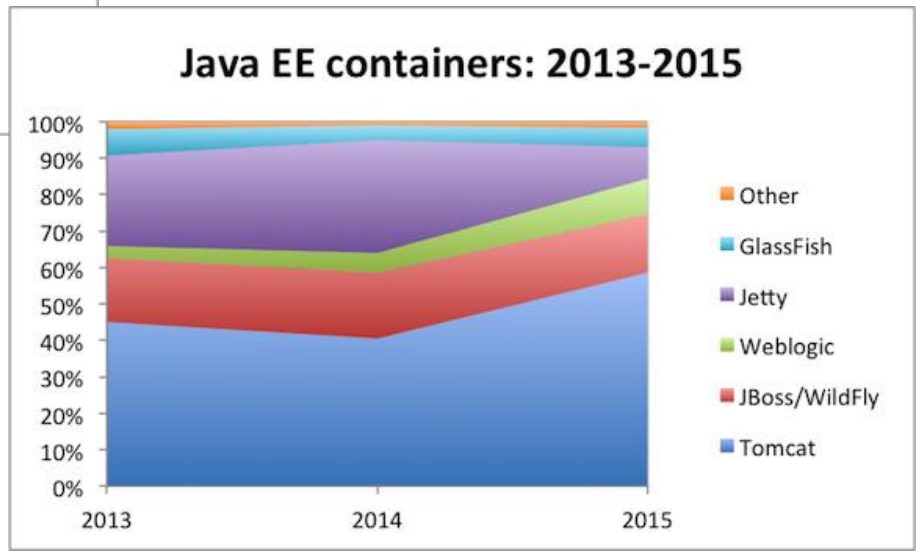


※ Apache Tomcat : http://tomcat.apache.org/

※ **User Mailing list :** http://mail-archives.apache.org/mod_mbox/tomcat-users/

Why Tomcat?





※ https://plumbr.eu/blog/java/most-popular-java-ee-containers-2015-edition

적용 사례?

해외

#	WEBSITE	DETECTIONS
1	linkedin.com	176
2	de.lovense.com	97
3	ehportal.southernnevadahealthdistrict.org	36
4	on281.infusionsoft.com	35
5	dsm-1.sec.cloudpack.jp	33
6	h21021.www2.hp.com	31
7	elearning.uky.edu	29
8	ekstrabladet.dk	27
9	as278.infusionsoft.com	22
10	ebay.com	21

WEBSITE LISTS AVAILABLE

Create a list of **158,040** websites using Apache Tomcat.

※ https://wappalyzer.com/applications/apache-tomcat

국내

What is Tomcat?

- 1996년 6월 제 4회 JavaOne (당시 Sun Microsystems 주최) 가 개발하고 있던, **JSWDK(JavaServer Web Development Kit)**을 Apache Software Foundation 에 기증하여 시작 되었음.

- **Tomcat**은 **Apache Software Foundation(ASF) 하위의 Apache Tomcat 프로젝트**에서 가 개발 되고 있는 **Sevlet/JSP Container**가 포함된 **웹 어플리케이션 서버(WAS)**

- **Tomcat History**
  - Tomcat 3.x (1999년) : 최초 릴리즈
  - Tomcat 4.x (2003년) : Catalina 아키텍처 도입
  - Tomcat 5.x (2004년) : Session Replication 기능 구현
  - Tomcat 6.0 (2007년) : Primary-Secondary 방식 Sessioin Replication 구현
  - Tomcat 7.0 (2011년) : Security 강화, Connector 리팩토링
  - Tomcat 8.0 (2014년) : Resouce 리팩토링, WebSocket 기능 추가
  - Tomcat 9.0 (2015년) : 9.0.0.M1(alpha) Release (2015-11-19)

Apache Tomcat Versions

- Tomcat은 Serlvet Spec 별 Major 버전으로 릴리즈

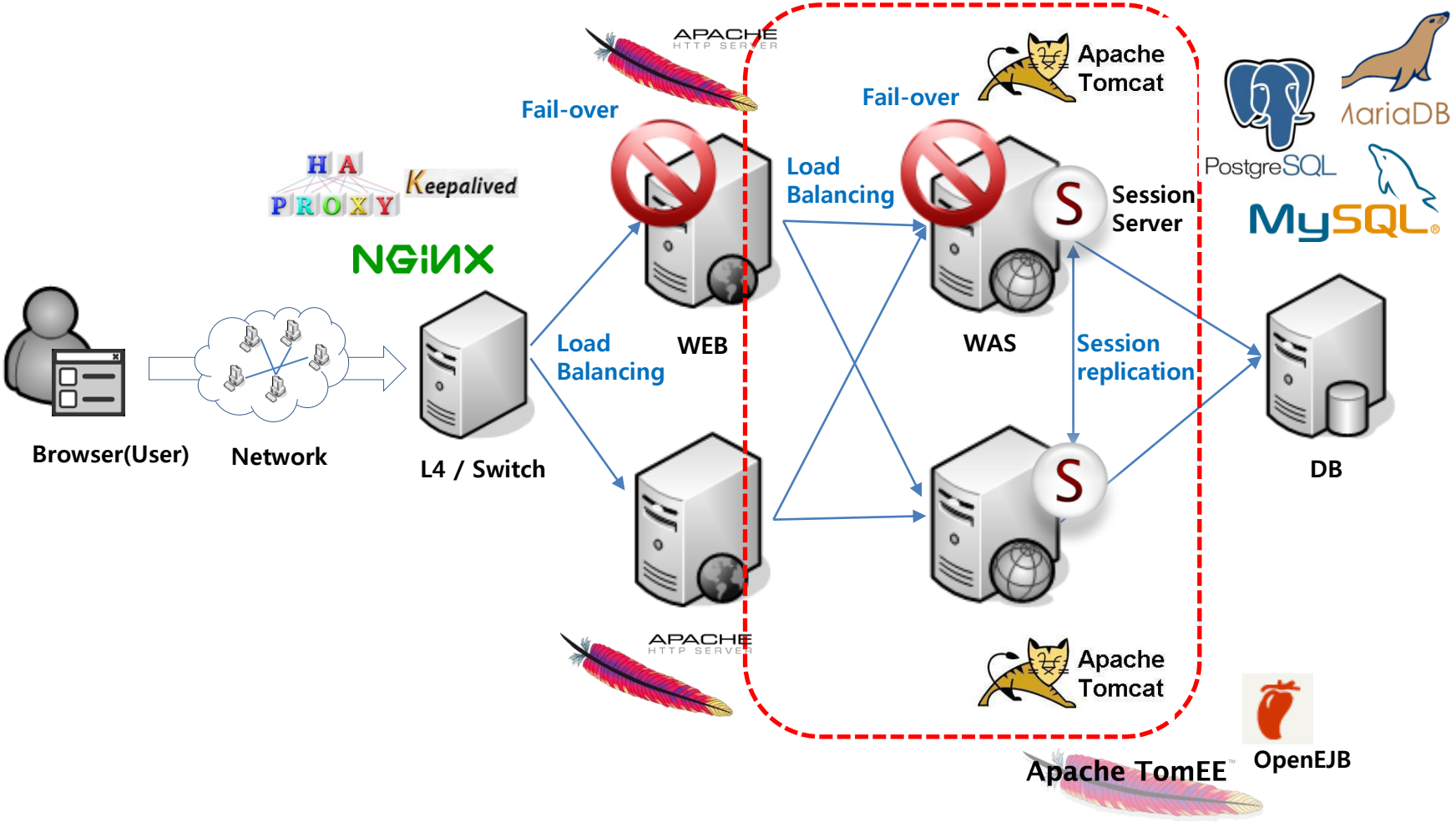
- Servlet 버전과 Tomcat 사양(Spec)

Servlet Spec	JSP Spec	EL Spec	WebSocket Spec	Tomcat version	Release revision	Support Java Versions
4.0	TBD (2.4?)	TBD (3.1?)	TBD (1.2?)	9.0.x	9.0.0.M1 (alpha)	8 and later
3.1	**2.3**	**3.0**	**1.1**	**8.0.x**	**8.0.28**	**7 and later**
3.0	**2.2**	**2.2**	**1.1**	**7.0.x**	**7.0.65**	**6 and later (WebSocket 1.1 requires 7 or later)**
2.5	2.1	2.1	N/A	6.0.x	6.0.44	5 and later
2.4	2.0	N/A	N/A	5.5.x (archived)	5.5.36 (archived)	1.4 and later
2.3	1.2	N/A	N/A	4.1.x (archived)	4.1.40 (archived)	1.3 and later
2.2	1.1	N/A	N/A	3.3.x (archived)	3.3.2 (archived)	1.1 and later

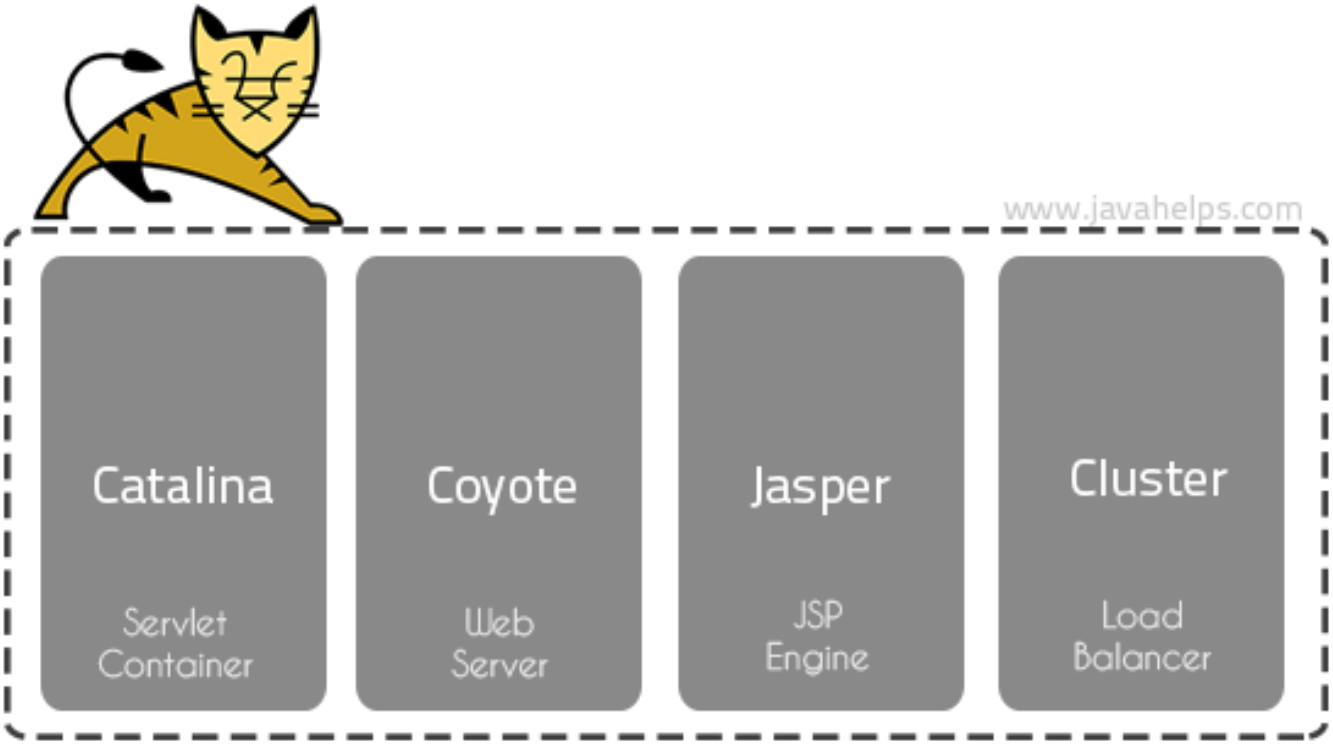
※ **TBD :** To Be Determined
※ **Community support version :** Tomcat6, Tomcat7, Tomcat8, Tomcat9
End of life for Apache Tomcat 6.0.x : 31 December 2016.

Tomcat Architecture

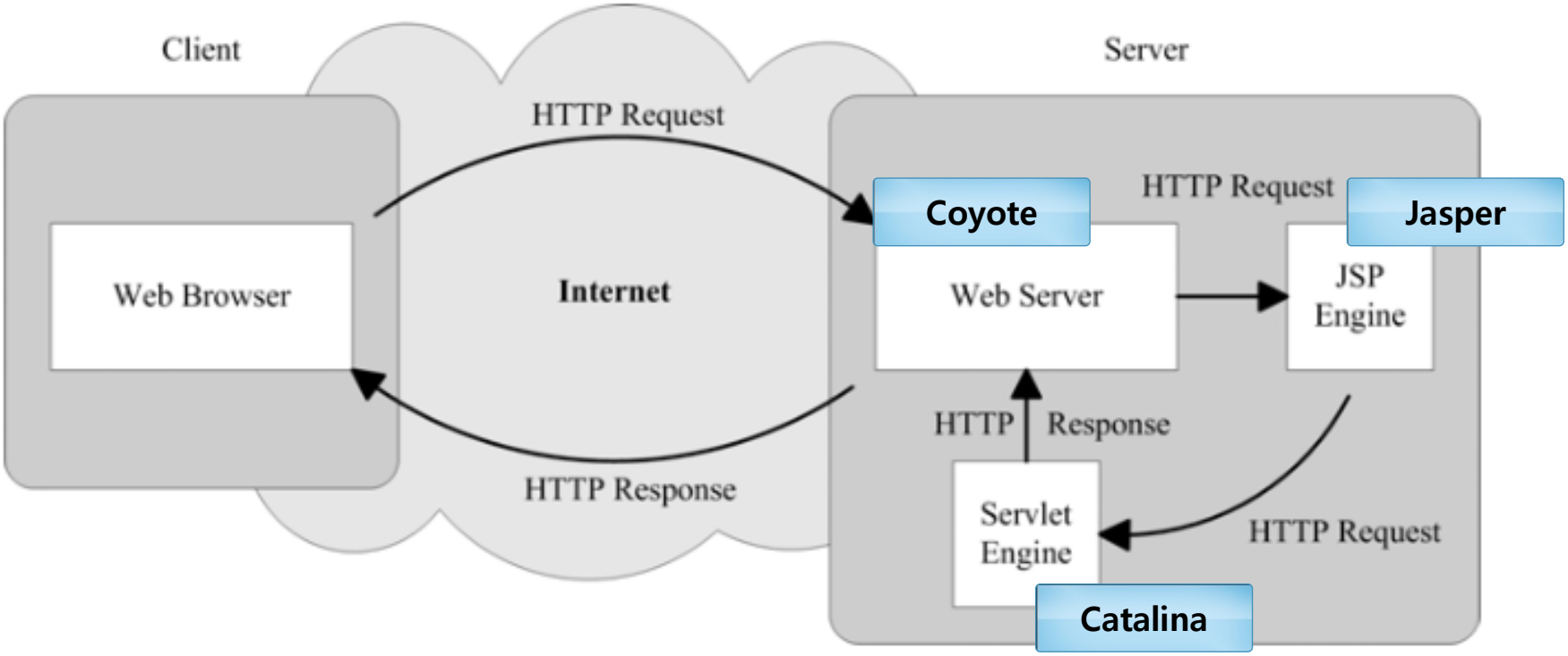
Enterprise System Architecture(with OSS)



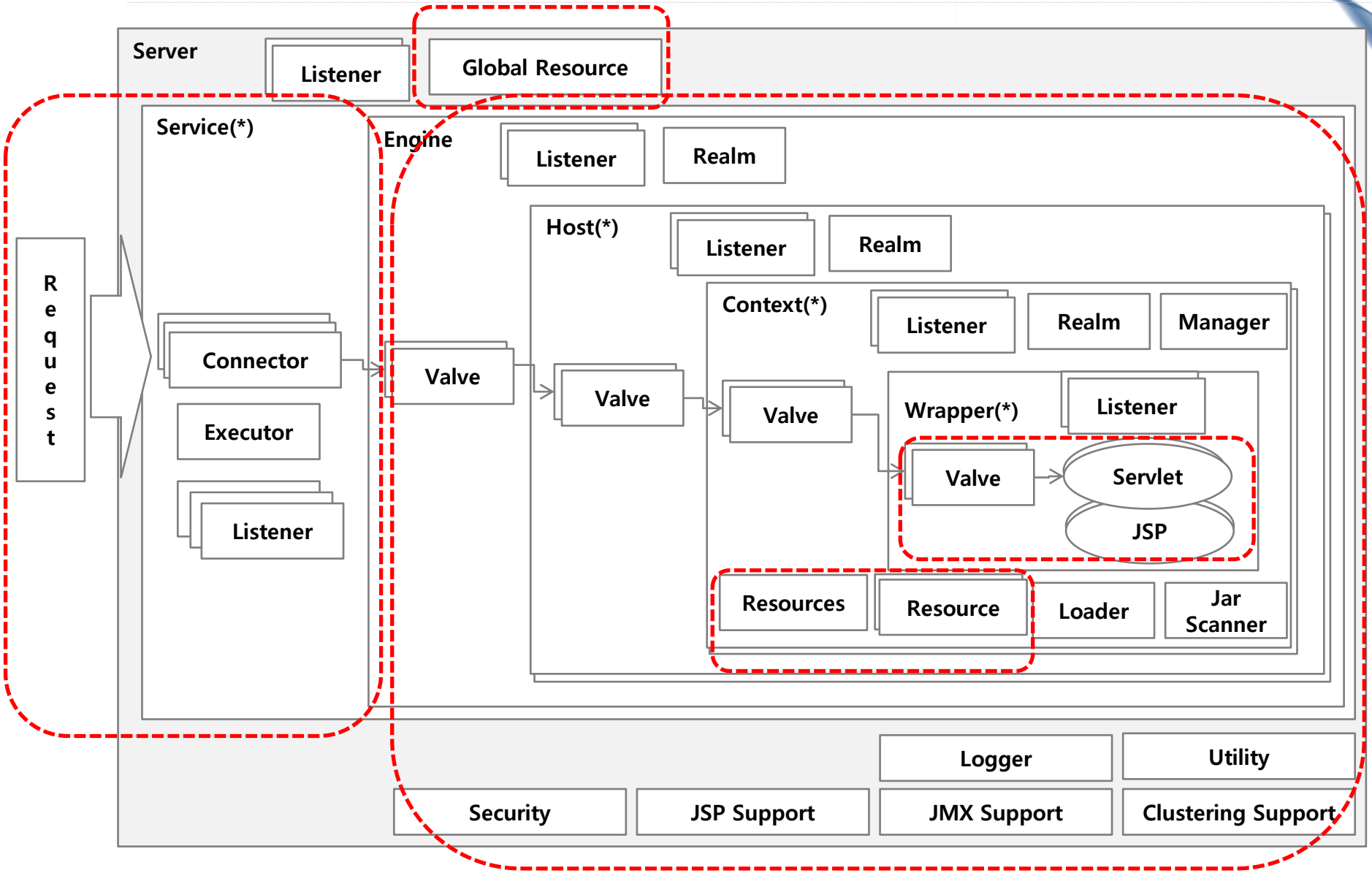
Tomcat Architecture(Core Component)



Tomcat Architecture

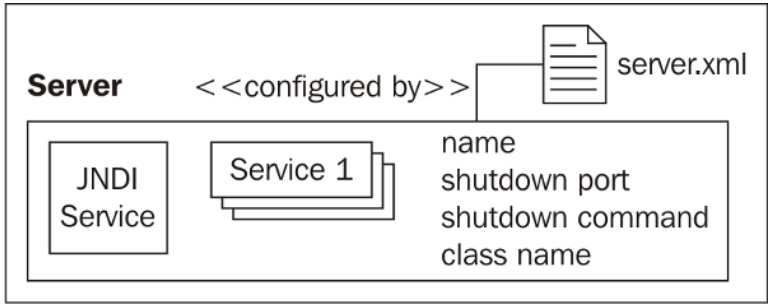


Tomcat Architecture

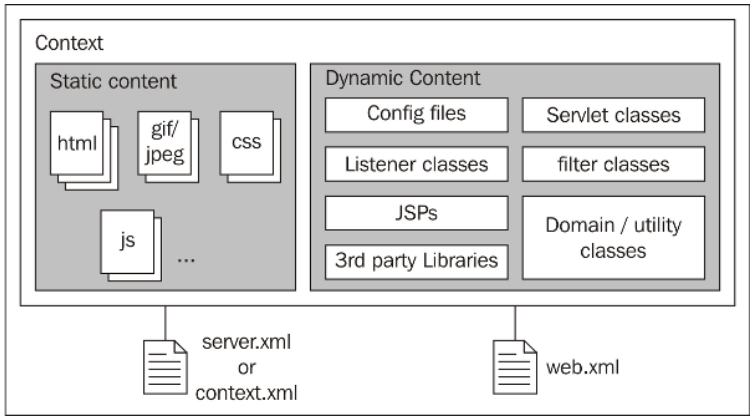


Tomcat Component

- **Server** : **Server** = Catalina servlet container = **Tomcat Instance**
  JVM 안에서 Singleton 으로 존재, Server내에 포함되는 Service들 Life Cycle관리 담당



- **Context** : **web application을 나타내는 구성요소,** Spec 상 ServletContext와 맵핑.
  WAR/EAR 파일이나, exploded된 디렉토리



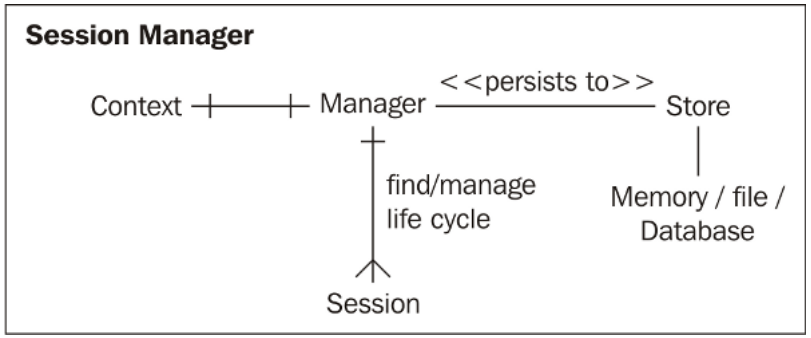
※ http://tomcat.apache.org/tomcat-8.0-doc/config/server.html 를 참고

Tomcat Component

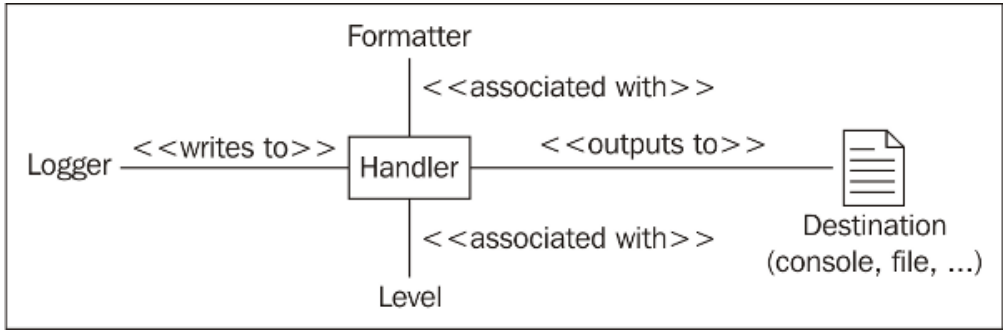
- **Listeners** : Tomcat Lifecycle 이벤트 발생시 action을 수행하는 컴포넌트
  Tomcat의 lifecycle 을 커스터마이징하고 싶은 경우 활용 가능

  ex) APR Lifecycle Listener, Global Resources Lifecycle Listener 등

- **Manager** : HTTP 세션을 생성 관리해주는 session manager

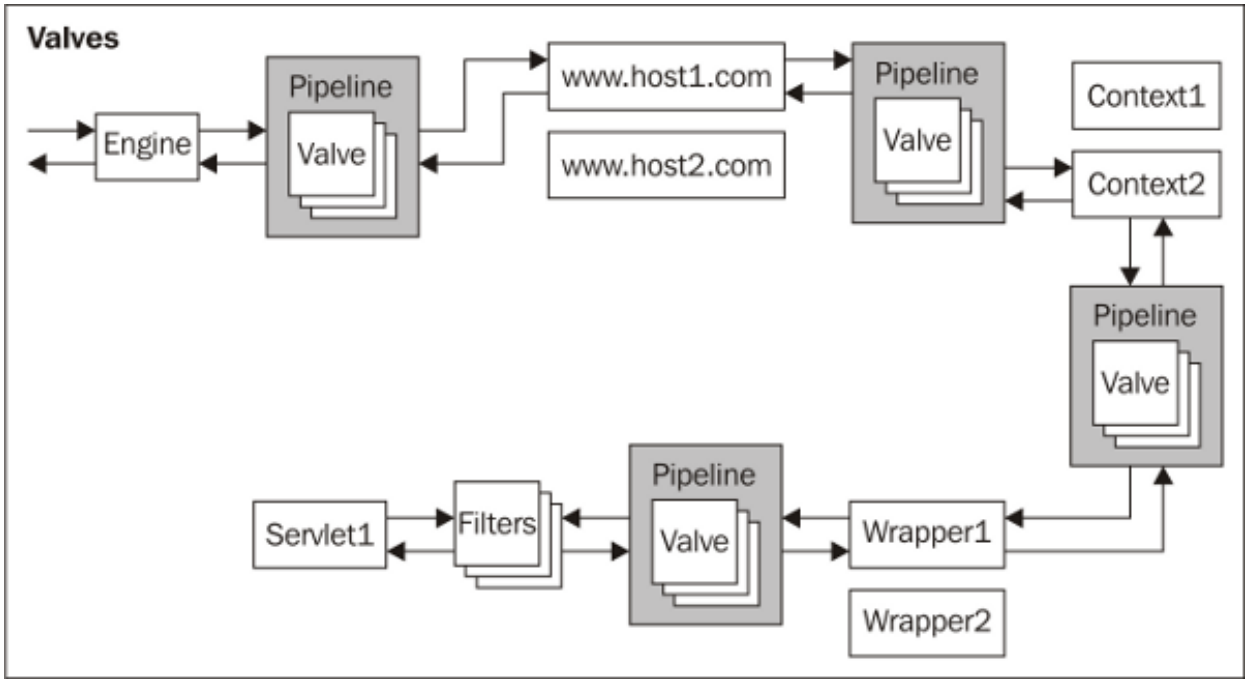


- **Logger** : 안에 JULI(Java Logging implimentation)라는 자체 구현체를 제공



Tomcat Component

- **Valve** : 컨테이너(Engine, Host, Context)와 관련된 각각의 request 처리 pipeline(valve chain 객체)안에 들어가 있는 컴포넌트



Server.xml

- Server.xml 예제

```
<?xml version='1.0' encoding='utf-8'?>
<Server port="8005" shutdown="SHUTDOWN">
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <GlobalNamingResources>
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>
  <Service name="Catalina">
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <Engine name="Catalina" defaultHost="localhost">
      <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
               resourceName="UserDatabase"/>
      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true"
            xmlValidation="false" xmlNamespaceAware="false">
        <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" />
      </Host>
    </Engine>
  </Service>
</Server>
```

※ **It is NOT recommended to place <Context> elements directly in the server.xml file**
(https://tomcat.apache.org/tomcat-8.0-doc/config/context.html)

Basic Features

- Tomcat Connector

Tomcat Connector 개요

- Client 로부터 요청을 받아 Container에 전달하기 위한 Component
- Connector 는 Tomcat의 웹서버 기능으로 Coyote Architecture 로 구현됨
- Protocal과 처리 방식에 따른 구현체를 가짐

- Connector Protocal
  - HTTP/HTTPS (HTTP/1.1)
  - AJP (AJP/1.3)

- Connector 처리방식
  - Java Blocking I/O (BIO) : All I/O operations are blocking in processor thread (Read, Write, etc.)
    → Simple, stable, mature
  - Java Non-blocking I/O (NIO) : Allows huge number of parallel requests
  - Java NIO.2 (NIO2) : like the NIO connector but uses the NIO2 framework
  - Native / Apache Portable Runtime (APR) : Native code (C), Use JNI

※ http://www.tecbar.net/optimize-tomcat-performance/

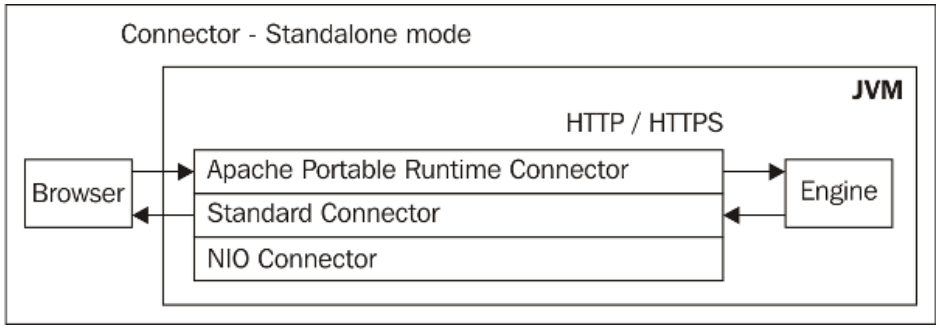
Tomcat Connector 성능

- Connector 성능
  - 이론적으로는 APR > NIO > BIO 이나, JAVA 버전, OS 설정, 사용자 부하 유형에 따라 상이함
  - Stability : BIO > NIO or APR
  - SSL : APR > NIO > BIO
  - Low concurrency : BIO > APR > NIO
  - High concurrency No Keep-Alive : BIO > APR > NIO
  - High concurrency Keep-Alive : APR > NIO > BIO

※ http://www.tecbar.net/optimize-tomcat-performance/

Tomcat Connector - HTTP Protocal

- HTTP Connector는 HTTP/1.1 protocol을 지원하는 컴포넌트



- 속성 :

```
<Service name="Catalina">
    <Connector port="8080" protocol="HTTP/1.1"  connectionTimeout="20000" redirectPort="8443" />
    ...
</Service>
```

항목	설명	기본값
protocol	HTTP, AJP 등 Connector의 Protocal을 설정하는 항목으로 구현체를 직접 설정도 가능함 - BIO(java connector) : org.apache.coyote. http11.Http11Protocol - NIO(java connector) : org.apache.coyote. http11.Http11NioProtocol - NIO2(java connector) : org.apache.coyote. http11.Http11Nio2Protocol - APR(native connector) : org.apache.coyote. http11.Http11AprProtocol	HTTP/1.1

Tomcat Connector - HTTP Protocal

- HTTP Connector는 HTTP/1.1 protocol을 지원하는 컴포넌트

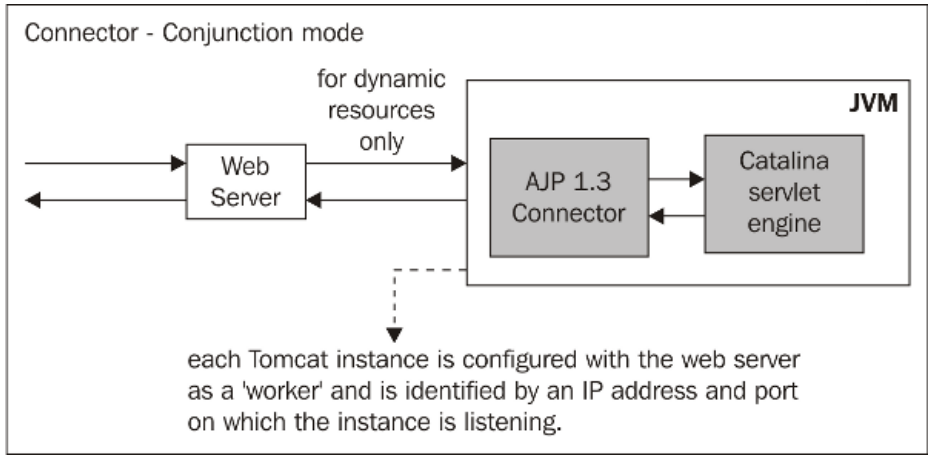
	Java Blocking Connector BIO	Java Nio Blocking Connector NIO	Java Nio2 Blocking Connector NIO2	APR/native Connector APR
Classname	Http11Protocol	Http11NioProtocol	Http11Nio2Protocol	Http11AprProtocol
Tomcat Version	3.x onwards	6.x onwards	8.x onwards	5.5.x onwards
Support Polling	NO	YES	YES	YES
Polling Size	N/A	maxConnections	maxConnections	maxConnections
Read HTTP Request	Blocking	Non Blocking	Non Blocking	Blocking
Read HTTP Body	Blocking	Sim Blocking	Blocking	Blocking
Write HTTP Response	Blocking	Sim Blocking	Blocking	Blocking
Wait for next Request	Blocking	Non Blocking	Non Blocking	Non Blocking
SSL Support	Java SSL	Java SSL	Java SSL	OpenSSL
SSL Handshake	Blocking	Non blocking	Non blocking	Blocking
Max Connections	maxConnections	maxConnections	maxConnections	maxConnections

※ http://tomcat.apache.org/tomcat-8.0-doc/config/http.html

※ http://tomcat.apache.org/tomcat-8.0-doc/config/http.html#Connector_Comparison

Tomcat Connector - AJP Protocal

- AJP Connector는 AJP protocol을 사용하여 Tomcat과 web server와 통신할 때 사용되는 컴포넌트



- 속성 :

```
<Service name="Catalina">
    <Connector port="8089" protocol="AJP/1.3" redirectPort="8443" />
    ...
</Service>
```

항목	설명	기본값
protocol	HTTP, AJP 등 Connector의 Protocal을 설정하는 항목으로 구현체를 직접 설정도 가능함 - BIO : org.apache.coyote.ajp.AjpProtocol - NIO : org.apache.coyote.ajp.AjpNioProtocol - NIO2 : org.apache.coyote.ajp.AjpNio2Protocol - APR : org.apache.coyote.ajp.AjpAprProtocol	AJP/1.3

Tomcat Connector - AJP Protocal

- AJP Connector는 AJP protocol을 사용하여 Tomcat과 web server와 통신할 때 사용되는 컴포넌트

	Java Blocking Connector BIO	Java Nio Blocking Connector NIO	Java Nio2 Blocking Connector NIO2	APR/native Connector APR
Classname	AjpProtocol	AjpNioProtocol	AjpNio2Protocol	AjpAprProtocol
Tomcat Version	3.x onwards	7.x onwards	8.x onwards	5.5.x onwards
Support Polling	NO	YES	YES	YES
Polling Size	N/A	maxConnections	maxConnections	maxConnections
Read Request Headers	Blocking	Sim Blocking	Blocking	Blocking
Read Request Body	Blocking	Sim Blocking	Blocking	Blocking
Write Response	Blocking	Sim Blocking	Blocking	Blocking
Wait for next Request	Blocking	Non Blocking	Non Blocking	Non Blocking
Max Connections	maxConnections	maxConnections	maxConnections	maxConnections

※ http://tomcat.apache.org/tomcat-8.0-doc/config/ajp.html

※ http://tomcat.apache.org/tomcat-8.0-doc/config/http.html#Connector_Comparison

※ http://www.tecbar.net/optimize-tomcat-performance/

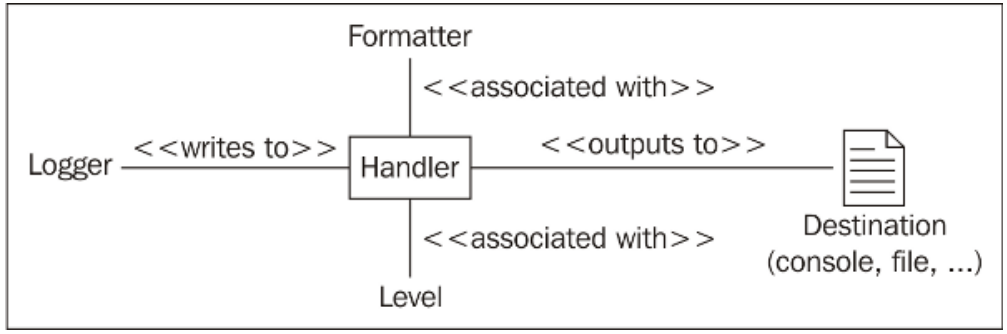
Basic Features

- Logging

Tomcat Logging

- **JULI Logging Framework 사용**
  Apache Commons Logging 기반으로 구현
  → java.util.logging 사용 (default)
  extra 패키지를 통해 Log4j 등 Logging Framework 변경 가능

- **JULI Logging Architecture**



Tomcat Logging

- **ClassLoaderLogManager**

  java.util.logging.LogManager를 확장

  - $CATALINA_BASE/conf/logging.properties 파일 Loading 하여, Tomcat에 적용
  - ClassLoader의 path에서 logging.properties를 찾아, ClassLoader별 Log 구성을 적용
    -> 웹어플리케이션 별 log Configuration 구현

```
# ------------------------------------------------------------------------
#   LOGGING_MANAGER (Optional) Override Tomcat's logging manager
#                   Example (all one line)
#                   LOGGING_MANAGER="-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager"
# ------------------------------------------------------------------------

if [ -z "$LOGGING_MANAGER" ]; then
  LOGGING_MANAGER="-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager"
fi
```

Tomcat Logging

- **logging.properties**

```
handlers = 1catalina.org.apache.juli.AsyncFileHandler, 2localhost.org.apache.juli.AsyncFileHandler,
java.util.logging.ConsoleHandler

.handlers = 1catalina.org.apache.juli.AsyncFileHandler, java.util.logging.ConsoleHandler

############################################################
# Handler specific properties.
# Describes specific configuration info for Handlers.
############################################################

1catalina.org.apache.juli.AsyncFileHandler.level = FINE
1catalina.org.apache.juli.AsyncFileHandler.directory = ${catalina.base}/logs
1catalina.org.apache.juli.AsyncFileHandler.prefix = catalina.

2localhost.org.apache.juli.AsyncFileHandler.level = FINE
2localhost.org.apache.juli.AsyncFileHandler.directory = ${catalina.base}/logs
2localhost.org.apache.juli.AsyncFileHandler.prefix = localhost.

############################################################
# Facility specific properties.
# Provides extra control for each logger.
############################################################

java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter = org.apache.juli.OneLineFormatter

org.apache.catalina.core.ContainerBase.[Catalina].[localhost].level = INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].handlers = 2localhost.org.apache.juli.AsyncFileHandler
```

Tomcat Logging

- **Tomcat Logging 종류**
  java.util.logging, javax.servlet.ServletContext.log, Console, Access Logging 이 제공됨

구분	설명	파일명
Java Logging API	Tomcat에서는 Application당 로깅설정을 제어하기 위해 java.util.logging 기반의 구현체 JULI를 사용 logging.properties 파일의 설정에 따름	ex) localhost.yyyy-MM-dd.log
Servlets logging API	javax.servlet.ServletContext.log(...)를 통해 메시지를 출력할 때 사용됨 logging.properites 파일의 설정 중 org.apache.catalina.core.ContainerBase.[${engine}].[${host}].[${context}] 의 설정에 따름	ex) manager.yyyy-MM-dd.log
Console	Tomcat운영 시 출력되는 STDERR/STDOUT	ex) catalina.out
Access logging	AccessLogValve 또는 ExtendedAccessLogValve에서 생성하는 로그 Access Log Valve의 설정에 따름	ex) access_log.yyyy-MM-dd

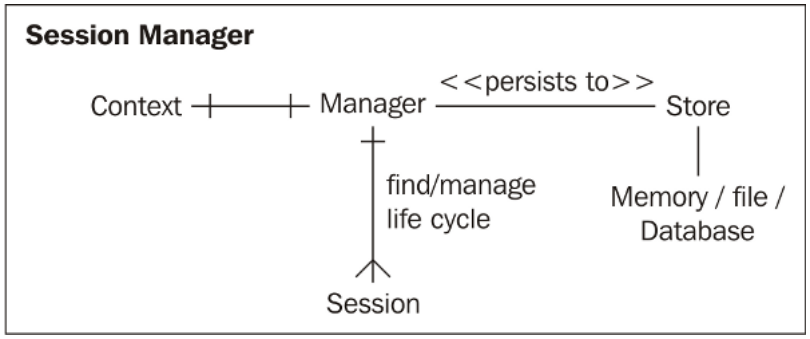
Basic Features

- **Session Manager**

Tomcat Session Manager

- **Tomcat Session Manager**
  HTTP 세션을 생성 관리해주는 session manager로 Context안에 설정하며, 설정이 안되어 있는 경우 자동으로 기본 설정값에 의해 생성됨 (default : org.apache.catalina.session.StandardManager)

```
<Manager classname="org.apache.catalina.session.StandardManager" />
```



- Session Manager 종류

구분	설명	파일명
StandardManager	기본 설정. 한개의 instance를 사용하는 경우만 적용가능	org.apache.catalina.session.StandardManager
PersistentManager	디스크 또는 DB에 세션을 Persist. 세션 스와핑과 장애 대처(fault tolerance)	org.apache.catalina.session.PersistentManager
DeltaManager	All-to-All 방식 Session Replication 기능 구현	org.apache.catalina.session.DeltaManager
BackupManager	Primary-Secondary Session Replication 기능을 구현	org.apache.catalina.session.BackupManager

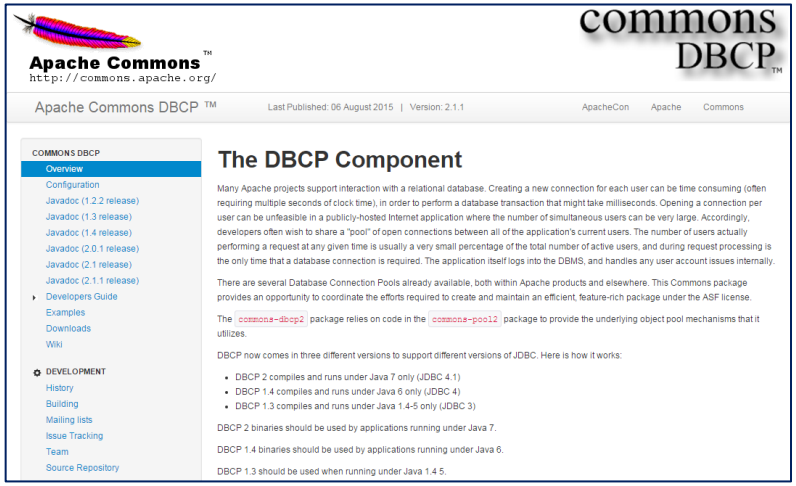
※ http://tomcat.apache.org/tomcat-8.0-doc/config/manager.html 를 참고

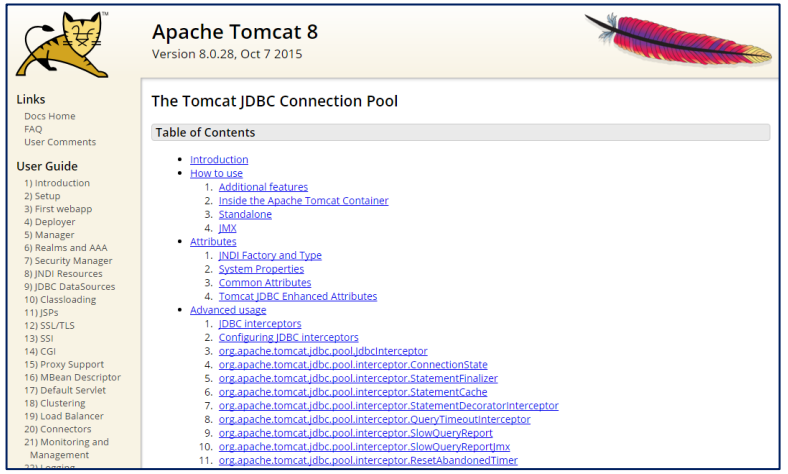
Basic Features

- Tomcat JDBC Connectivity

Tomcat Connnection Pool

- Tomcat은 DataBase Connection Pool 로 **Commons DBCP**와 **Tomcat JDBC Pool**의 2개지 Pool을 제공





Commons DBCP (http://commons.apache.org/proper/commons-dbcp/)

The Tomcat JDBC Connection Pool (https://tomcat.apache.org/tomcat-8.0-doc/jdbc-pool.html)

※ Tomcat 7.0.18 – extra package로 포함
Tomcat 7.0.19 – 정식 Release

Tomcat JDBC Connnection Pool

- Tomcat은 DataBase Connection Pool 로 Commons DBCP와 Tomcat JDBC Pool의 2개지 Pool을 제공
  - **Commons DBCP** : **org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory (default)**
    - org.apache.commons -> org.apache.tomcat. 패키지 변경
  - **Tomcat JDBC** : **org.apache.tomcat.jdbc.pool.DataSourceFactory**

- DBCP 는 Apache Commons 프로젝트가 제공하는, DataBase Connection Pool로 오픈 소스 DataBase Connection Pool 를 가장 많이 알려져 있고, 많이 사용하고 있다.

- Tomcat JDBC는 Apache Tomcat 프로젝트가 독립적으로 개발한 DataBase Connection Pool 이다.

```
<Resource type="javax.sql.DataSource"
    name="jdbc/TestDB"
    factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/mysql"
    username="mysql_user"
    password="mypassword123" />
```

Tomcat JDBC Connnection Pool 기본설정

- 공통 DB 접속 설정

구분	설명	비고
username	DB 연결을 위해 JDBC 드라이버에 전달되는 사용자 이름	
password	DB 연결을 위해 JDBC 드라이버에 전달되는 사용자 패스워드	
url	DB 연결을 위해 JDBC 드라이버에 전달되는 연결 URL	
driverClassName	JDBC 드라이버의 FQCN （ Fully Qualified Class Name)	
connectionProperties	DB 연결을 위해 JDBC 드라이버에 전달되는 연결 Properties	

항목	설명	기본값
defaultAutoCommit	생성된 Connection의 자동 Commit Mode	JDBC Dirver 기본값 - Oracle : true - MySQL/MariaDB : true
defaultReadOnly	생성된 Connection의 ReadOnly	JDBC Dirver 기본값
defaultTransactionIsolation	생성된 Connection의 TransactionIsolation	JDBC Dirver 기본값
defaultCatalog	생성된 Connection의 defaultCatalog	JDBC Dirver 기본값

※ **Commons DBCP :** defaultAutoCommit 의 기본값은 true

Tomcat JDBC Connnection Pool 기본설정

- 공통 DB 접속 설정

항목	설명	기본값
initialSize	Pool 기동 시 생성되는 Connection의 초기 사이즈	10
maxActive	Pool에서 동시에 유지할수 있는 최대 연결 수	100
maxIdle	Pool에서 유지하는 최대 Connection 수	100
minIdle	Pool에서 유지하는 최소 Connection 수	initialSize
maxWait	사용 가능한 Connection이 존재하지 않는 경우 대기하는 최대시간	30,000 (30초)
maxAge	Connection을 보존하는 시간. Connection 반환 시에 maxAge 이상 경과했으면, 해당 Connection을 Close 시킴	0 (msc)
fairQueue	getConnection 호출이 Connection이 FIFO방식으로 공정하게 제공하려면 true로 설정. org.apache.tomcat.jdbc.pool.FairBlockingQueue 로 구현되어 있으며, 쓰레드가 도착한 순서대로 Connection을 획득하는 것이 보장됨	true

Tomcat JDBC Connnection Pool 기본설정

- Connection Validation 관련 설정

항목	설명	기본값
validationQuery	Connection 유효성 확인을 위한 SQL Query	MySQL/MariaDB : SELECT 1 Oracle : SELECT 1 FROM DUAL
testOnBorrow	Pool에서 끄낸 Connection을 전달하기 전에 유효성을 검증 검증에 실패했을 경우, 해당 연결을 삭제하고, 새롭게 Pool에서 Connection을 끄내려고 시도	False, true 설정시 validationQuery 필요
testOnReturn	Pool에 Connection을 반환하기 전에 Connection의 유효성을 검증	false, true 설정시 validationQuery 필요
testWileIdle	Idle 상태의 Connection 에 대한 유효성을 검증 검증에 실패할 경우 해당 Connection을 파기함 이기능은 Evictor 를 이용하기 떄문에 Evictor를 활성화 해야됨 (timeBetweenEvictionRunsMillis 설정 참조)	false, true 설정시 validationQuery 필요
initSQL	Connection을 만들떄 실행되는 SQL Query	
validationClassName	Validation Class명을 지정. 이 속성을 지정 시 validationQuery에 의한 검증 대신, Validator Class에 의한 연결의 검증이 이루어짐 (org.apache.tomcat.jdbc.pool.Validator 인터페이스 구현 필요)	
validationInterval	Validation의 실행 간격 지정. 마지막 검증으로부터 이 값 이상 경과하지 않은 경우 연결 검증을 거치지 않음	30000 (30초)
logValidationErrors	Validation중 오류 발생시, 로그 출력할지에 대한 설정	false

Idle 상태의 Connection 에 관한 파라미터(Eviction)

- PoolCleaner 관련 설정 (Evictor)

항목	설명	기본값
timeBetweenEvictionRunsMilles	Idle 상태의 Connection 을 check 하는 간격,	5,000 (5초)
minEvictableIdleTimeMillis	Idle Connection의 생존 기간	60,000(60초)

* Commons DBCP의 속성 중 numTestsPerEvictionRun 속성 삭제

- Connection Leak 검출 설정 (removeAbandoned)
  - PoolCleaner 관련 설정 (removeAbandoned)

항목	설명	기본값
removeAbandoned	Connection 누수 감지 설정 사용 여부	false
removeAbandonedTimeout	Connection 누수로 판단하기까지의 시간	60 초
logAbandoned	Connection 누수를 감지했을 때, Connection을 Close하지 않은 어플리케이션의 Stuck Thread를 로그로 출력 여부의 설정	false
suspectTimeout	Connection 누수의 의심에 대한 판단 할때까지의 시간 suspectTimeout 시간을 초과한 Connection에 대해 경고 로그 및 JMX에 통지를 수행	0
abandonWhenPercentageFull	누수된 Connection 을 파기할 비율을 설정 0~100 사이로 설정하며, 이 설정을 초과한 누수 Connection을 파기, removeAbandonedTimeout가 초과한 Connection은 모두 파기.	0

Tomcat JDBC Connnection Pool 설정 예제

- DataSouce Resource 설정

```
<Resource
    name="${jdbc.resource.name}"
    factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
    auth="Container"
    type="javax.sql.DataSource"
    username="${jdbc.username}"
    password="${jdbc.password}"
    driverClassName="${jdbc.driverClassName}"
    url="${jdbc.url}"
    testWhileIdle="true"
    testOnBorrow="true"
    testOnReturn="false"
    validationQuery="SELECT 1"
    validationInterval="30000"
    timeBetweenEvictionRunsMillis="5000"
    maxActive="100"
    minIdle="10"
    maxWait="10000"
    initialSize="10"
    removeAbandonedTimeout="60"
    removeAbandoned="true"
    logAbandoned="true"
    minEvictableIdleTimeMillis="30000"
    jmxEnabled="true"
    jdbcInterceptors="ConnectionState;StatementFinalizer;SlowQueryReportJmx(threshold=10000)"/>
```

Tomcat DataSource 설정

- Global Resource Level
  → DataSource Pool Global level 공유
  - Global Resource(server.xml) + Resouce Link(application-context.xml)
    → 해당 Application 에서만 사용 가능
  - Global Resource(server.xml) + Resouce Link(Global context.xml)
    → 모든 Application에서 사용 가능

- Context Level
  → DataSource Pool 해당 Context 별 생성됨
  - Global Context (context.xml)
    → 모든 Application에서 사용 가능
  - Application Context (application-context.xml)
    → 해당 Application에서만 사용 가능

Global Resource Level (+ Resouce Link)

- Global Resource Level (+ Resouce Link)
  - Server : ${CATALINA_HOME}/conf/server.xml

```
<Server port="${port.shutdown}" shutdown="ARGO_INSTANCE_SHUTDOWN">
  <GlobalNamingResources>
  <Resource  name="jdbc/argo"
   driverClassName="com.mysql.jdbc.Driver"
   url="jdbc:mysql://10.0.1.88:3306/oom"
    username="root“
    password="uCO5FkZ3NfgDo/LWB0l+0A=="
    ……
  />
  </GlobalNamingResources>
</Server>
```

  - Context(1) : ${CATALINA_HOME}/conf/context.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<Context>
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <ResourceLink global="jdbc/argo" name="jdbc/argo" type="javax.sql.DataSource"/>
</Context>
```

  - Context(2) : ${CATALINA_HOME}/conf/Catalina/localhost/${CONTEXT_NAME}.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <Context docBase="/sorc001/argo/application/sample-simpleweb" path="/" reloadable="false">
    <ResourceLink global="jdbc/argo" name="jdbc/argo" type="javax.sql.DataSource"/>
</Context>
```

Context Level

- Context Level
  - Context(1) : ${CATALINA_HOME}/conf/context.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Context>

  <WatchedResource>WEB-INF/web.xml</WatchedResource>

 <Resource  name="jdbc/argo"
   driverClassName="com.mysql.jdbc.Driver"
   url="jdbc:mysql://10.0.1.88:3306/oom"
    username="root“
    password="uCO5FkZ3NfgDo/LWB0l+0A=="
    ……
  />

</Context>
```

  - Context(2) : ${CATALINA_HOME}/conf/Catalina/localhost/${CONTEXT_NAME}.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <Resource  name="jdbc/argo"
   driverClassName="com.mysql.jdbc.Driver"
   url="jdbc:mysql://10.0.1.88:3306/oom"
    username="root“
    password="uCO5FkZ3NfgDo/LWB0l+0A=="
    ……
  />
</Context>
```

Tomcat JDBC 로 변경해야 하는 이유.

- **Commons DBCP** 는 Single Thread를 사용한다. 그래서 Thread가 안전할 수 있도록 전체 풀(pool) 에 대해서 lock을 건다. 멀티 코어/CPU 환경에서 **성능이 떨어진다. (느리다)** → **(not DBCP 2.x)**

- **Commons DBCP** 는 복잡하다. Commons DBCP 는 관리해야 할 클래스가 60여개, Tomcat 은 8개의 핵심클래스만 사용한다.

- **Tomcat JDBC는 Common DBCP 에서 크게 변경 없이 전환이 가능**하며, 추가적으로 JMX 지원 확장 등으로 외부에서 Connection Validation, JdbcInterceptor를 통한 DB처리의 독립적인 처리 추가 기능 등 다양한 기능이 추가되어 있음

- 소스 코드도 Common DBCP에 비해, 상당히 간단하며, 릴리즈도 Tomcat 과 동시에 되기 때문에, 버그 수정 등 **유지 보수에 대한 안정성/신뢰성**을 제공함.

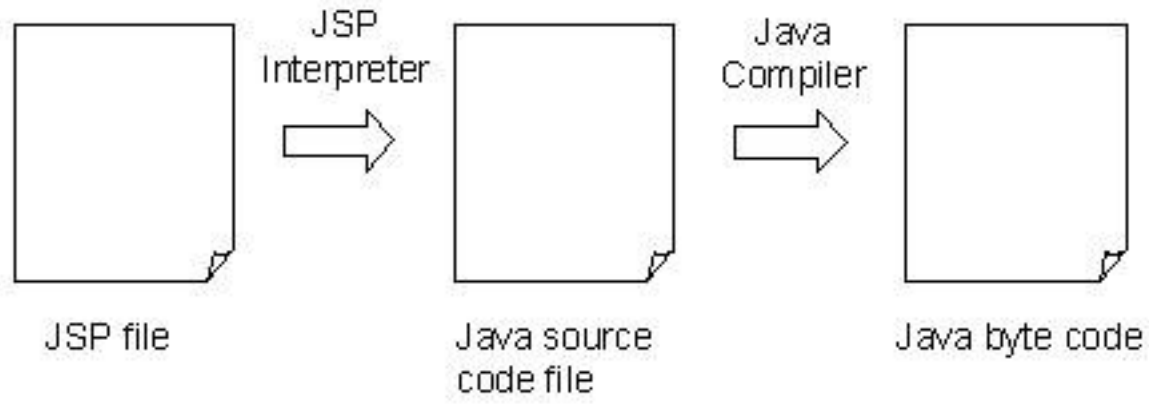
※ **Commons DBCP vs Tomcat JDBC :**
http://vigilbose.blogspot.kr/2009/03/apache-commons-dbcp-and-tomcat-jdbc.html
http://people.apache.org/~fhanik/tomcat/jdbc-pool.html

Basic Features

- Jasper

Jasper 개요

- Jasper란 Tomcat의 JSP엔진으로, JSP를 컴파일하기 위한 컴포넌트
- JSPC(JSP to Servlet), JavaC(Servlet to Class) 수행



Jasper Compile Opetion

- 모든 JSP 파일은 org.apache.jasper.servlet.JspServlet 과 Servlet 매핑되어 있음.

```
 <servlet>
        <servlet-name>jsp</servlet-name>
        <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
        <init-param>
            <param-name>fork</param-name>
            <param-value>false</param-value>
        </init-param>
        <init-param>
            <param-name>xpoweredBy</param-name>
            <param-value>false</param-value>
        </init-param>
<!--
        <init-param>
            <param-name>checkInterval</param-name>
            <param-value>0</param-value>
        </init-param>
        <init-param>
            <param-name>development</param-name>
            <param-value>true</param-value>
        </init-param>
-->
        <load-on-startup>3</load-on-startup>
    </servlet>

<!-- The mappings for the JSP servlet -->
    <servlet-mapping>
        <servlet-name>jsp</servlet-name>
        <url-pattern>*.jsp</url-pattern>
        <url-pattern>*.jspx</url-pattern>
    </servlet-mapping>
```

Advance Features

- 웹서버 연동 (with Apache HTTP Server)

웹서버와 연동해야되는 이유

- 정적 파일의 처리
  - Static Contents에 대한 성능
  - Apache HTTP Server의 다양한 옵션 및 기능 활용
    (Access Control, Contents Caching, Filter 등)
- 부하 분산
  - 사용자 요청에 대한 부하 분산
  - WEB + WAS에 대한 부하 분산
- 장애 대응
  - Fail Over
  - Fail Back
- 기타
  - 보안 (Security)

웹서버와 연동 방법(with Apache HTTP Server)

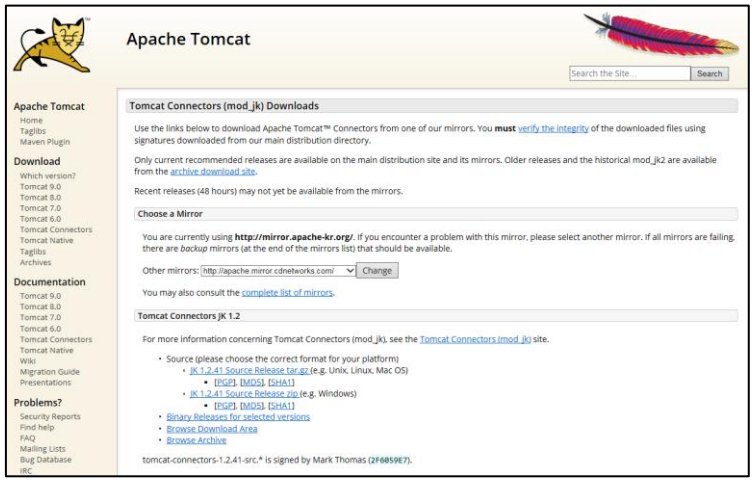
- 웹서버 연동 방법
  - **mod_proxy** - http://httpd.apache.org/docs/2.4/mod/mod_proxy.html
  - **mod_jk** - http://tomcat.apache.org/connectors-doc/
  - **mod_cluster** - http://mod-cluster.jboss.org/

- 연동방법 비교

항목	특징	비고
mod_proxy	• Protocal(http, https, ajp)에 관계없이 연동 가능 • Apache HTTP Server 에 모듈로 포함되어 있어, 별도 설치 및 관리가 필요 없음 • 기본적인 Load Balancer 기능 제공 • Apache HTTP Server 프로젝트에서 제공	
mod_jk	• 진보된 Load Balancer 기능 제공 • 진보된 장애 감지 기능 제공 • Apache Tomcat 프로젝트에서 제공 • 적용 사례 많음	
mod_cluster	• 동적으로 worker(Tomcat) 추가가 가능 • worker 상태에 따른 Load Balancing 가능 • JBOSS 연동을 위한 모듈 (Tomcat도 연동 가능) • 적용 사례가 많지 않음	

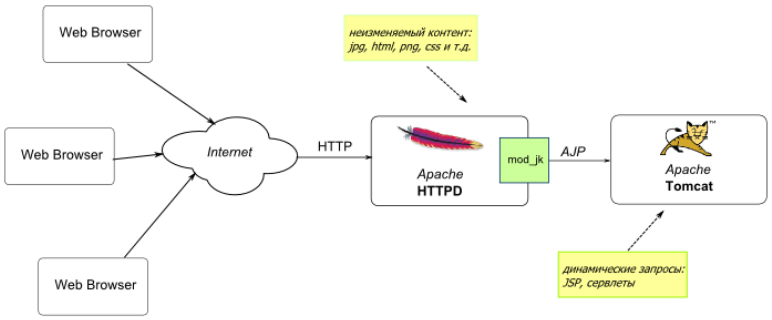
mod_jk(Tomcat Connectors)

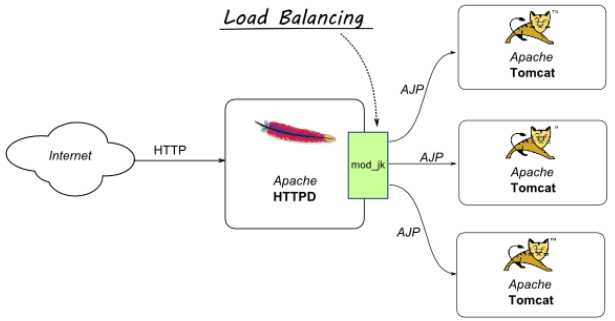
- Tomcat 프로젝트에서 제공하는 웹서버 연동 모듈
  - Document : http://tomcat.apache.org/connectors-doc/
  - Download : http://tomcat.apache.org/download-connectors.cgi



- mod_jk 기능
  - worker 감시 기능 (CPing/CPong)
  - Connection Pooling
  - Sticky Session
  - Retry (Recovery)
  - Load banance
  - worker Status 관리

- 적용 아키텍처





Apache HTTP Server와 Tomcat 연동

- mod_jk 적용 방안
  - mod_jk 설치 및 설정 (httpd.conf)
  - worker 정의 (worker.properties)
  - WEB-WAS 요청 매핑 정의 (uriworkermap.properties)

- mod_jk 설정

```
LoadModule jk_module    modules/mod_jk.so

<IfModule mod_jk.c>
    JkWorkersFile ${INSTALL_PATH}/conf/extra/workers.properties
    JkLogFile "|${ENGN_HOME}/bin/rotatelogs ${INSTALL_PATH}/logs/jk.log_%Y%m%d 86400 +540"
    JkLogLevel error
    JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
    JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
    JkShmFile ${INSTALL_PATH}/logs/jk.shm
    JkMountFile ${INSTALL_PATH}/conf/extra/uriworkermap.properties
</IfModule>

<Location /jk-status/>
    JkMount jkstatus
    Order deny,allow
    Deny from all
</Location>
```

Apache HTTP Server와 Tomcat 연동

- worker 정의

```
worker.list=${SERVER_LB_ID},jkstatus

worker.template.type=ajp13
worker.template.lbfactor=1
worker.template.socket_timeout=300
worker.template.socket_keepalive=True
worker.template.connect_timeout=30000
worker.template.connection_pool_size=128
worker.template.connection_pool_minsize=32
worker.template.connection_pool_timeout=20

##########################################
## Worker for Tomcat
##########################################

worker.argow_8180_hostname.reference=worker.template
worker.argow_8180_hostname.host=127.0.0.1
worker.argow_8180_hostname.port=8009

##########################################
## Load Balancer
##########################################

worker.${SERVER_LB_ID}.type=lb
worker.${SERVER_LB_ID}.balance_workers=argow_8180_hostname
worker.${SERVER_LB_ID}.sticky_session=1

worker.jkstatus.type=status
```

Apache HTTP Server와 Tomcat 연동

- 요청 매핑 정의

```
/*.jsp|/=${SERVER_LB_ID}
/*.do|/=${SERVER_LB_ID}
```

Status Worker (JK Status Manager)

- status type의 Worker 설정시 Status Worker로 동작함
- Status Worker는 worker의 상태 모니터링 및 mod_jk의 설정 정보를 관리하는 기능 제공

JK Status Manager for 188.0.1.190:8089

Server Version: Apache/2.2.3 (Red Hat) Server Time: Fri, 18 Jul 2014 16:08:32 CST
JK Version: mod_jk/1.2.40 Unix Seconds: 1405670912

Start auto refresh (every 10 seconds) | Change format XML
[Read Only] [Dump] [S=Show only this worker, E=Edit worker, R=Reset worker state, T=Try worker recovery]

Listing Load Balancing Worker (1 Worker) [Hide]

[S|E|R] Worker Status for wlb

Type	Sticky Sessions	Force Sticky Sessions	Retries	LB Method	Locking	Recover Wait Time	Error Escalation Time	Max Reply Timeouts [Hide]
lb	True	False	2	Request	Optimistic	60	30	0

Good	Degraded	Bad/Stopped	Busy	Max Busy	Next Maintenance	Last Reset [Hide]
1	0	0	0	1	58/120	287

Balancer Members [Hide]

Name	Type	Hostname	Address:Port	Connection Pool Timeout	Connect Timeout	Prepost Timeout	Reply Timeout	Retries	Recovery Options	Max Packet Size [Hide]
ajp13w	ajp13	localhost	127.0.0.1:8009	0	0	0	0	2	0	8192

	Name	Act	State	D	F	M	V	Acc	Sess	Err	CE	RE	Wr	Rd	Busy	Max	Con	Route	RR	Cd	Rs	LR	LE
[S|E|R]	ajp13w	ACT	OK	0	1	1	0	2 (0/sec)	0 (0/sec)	0	0	0	1.8K (6 /sec)	240 (0 /sec)	0	1	2	ajp13w		0/0	287		

Edit this attribute for all members: Activation Go

URI Mappings for wlb (7 maps) [Hide]

Server	URI	Match Type	Source	Reply Timeout	Sticky	Ignore	Stateless	Fail on Status	Active	Disabled	Stopped	Use Server Errors
localhost.localdomain	/servlets-examples/*.jpeg	Unmount	Wildchar	uriworkermap	-1	0	0	-	-	-	-	0
localhost.localdomain	/servlets-examples/*	Wildchar		uriworkermap	-1	0	0	-	-	-	-	0
localhost.localdomain	/jsp-examples/*	Wildchar		uriworkermap	-1	0	0	-	-	-	-	0
localhost.localdomain	/examples/*	Wildchar		uriworkermap	-1	0	0	-	-	-	-	0
localhost.localdomain	/manager/*	Wildchar		uriworkermap	-1	0	0	-	-	-	-	0
localhost.localdomain	/admin/*	Wildchar		uriworkermap	-1	0	0	-	-	-	-	0
localhost.localdomain	/*.jsp	Wildchar		uriworkermap	-1	0	0	-	-	-	-	0

Advance Features

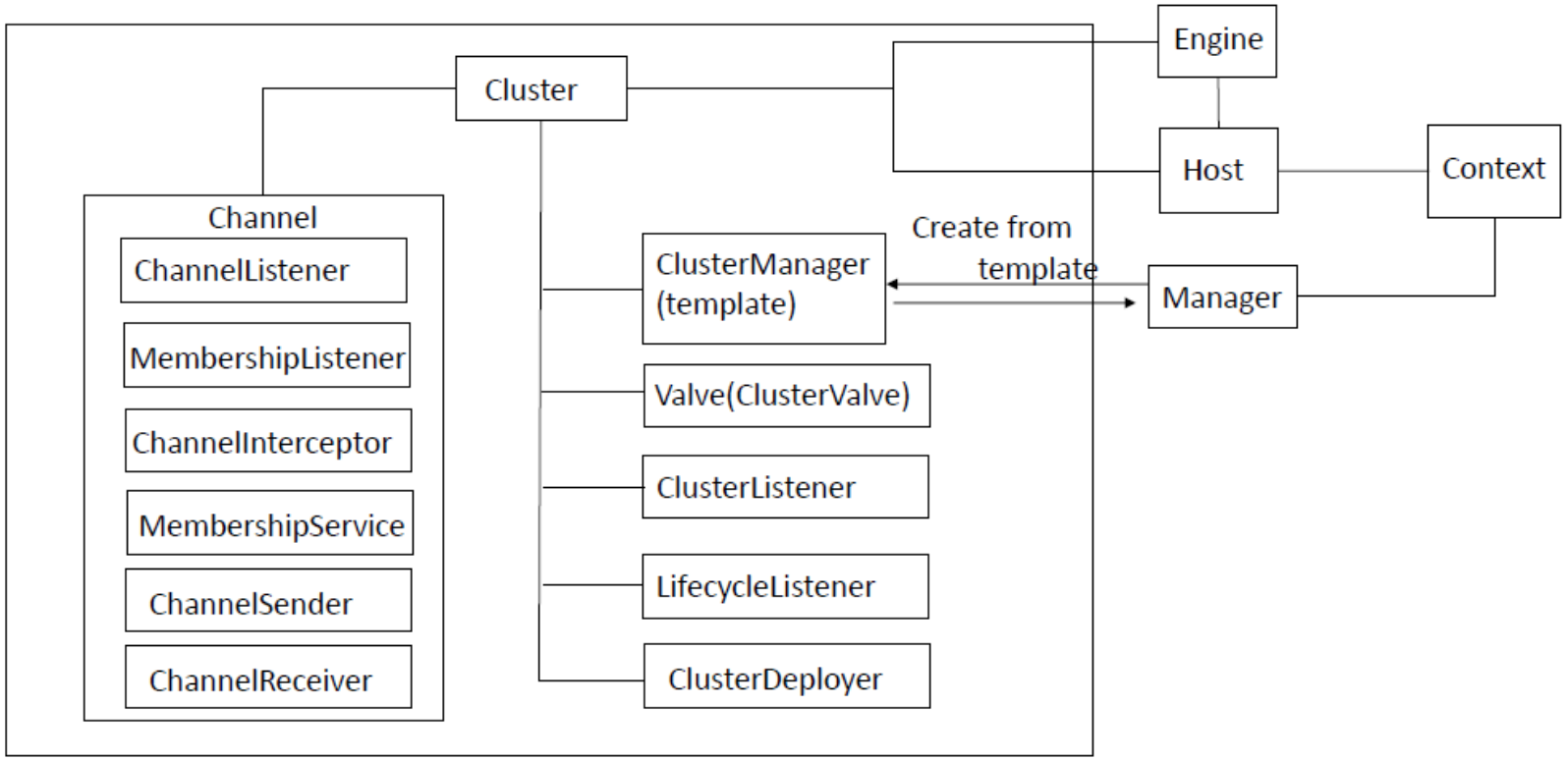
- Session Clustering

Clustering

- Clustering
  - 다수의 서버를 연결하여, 하나의 서버로 보이게 하는 기술
  - Clustering 목적 : 부하 분산, **성능 향상**, **가용성 향상**, Fail-over
- Tomcat Clustering
  - Cluster Load Balance (with Apache HTTP Server)
  - Cluster Membership
  - Session Replication

Cluster Architecture

- Clustering Arcihitecture



Tomcat Cluster Architecture

- Cluster Architecture

Cluster

Manager - Cluster 환경에서의 Session Manager를 설정.

Channel - cluster group의 communication 관리를 담당.

Membership - Multicast(default) 방식으로 cluster group의 노드 추가/삭제에 대한 공지를 책임짐.

Receiver - Member가 보내오는 Message receive함.

Sender- 외부로 나가는 Message를 관리.

Interceptor - Message send할 때 interceptor stack을 거침.
(Send 전 Message를 변경 할 수 있음)

Deployer - Application Deploy를 cluster 단위로 가능하게 해줌.

Valve - Request가 Servlet Container 출입하는 시점에 Track.(Tomcat Valve와 동일)

ClusterListener - Cluster Group의 Message send/receive 시의 Track.

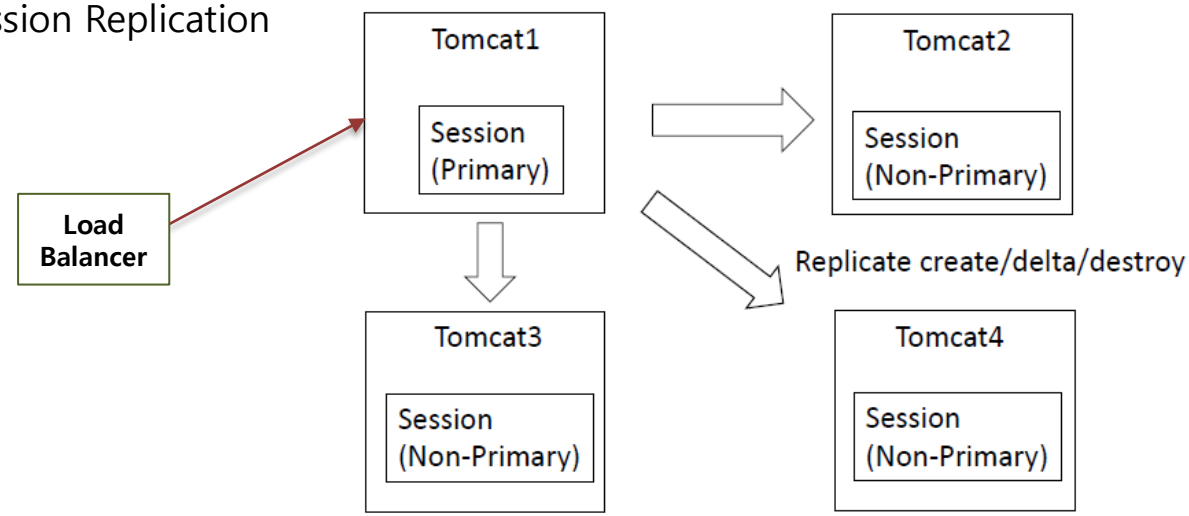
Tomcat Session Replication Manager

- Cluster Session Manager 종류

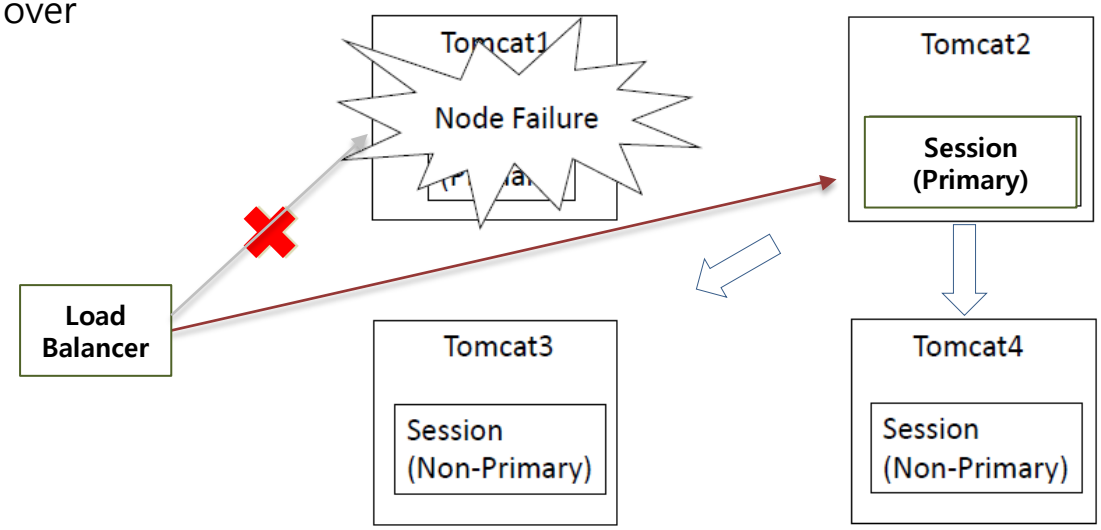
구분	설명	파일명
DeltaManager	• All-to-All session replication • Default Session Manager in Cluster environment • For small cluster	org.apache.catalina.session.DeltaManager
BackupManager	• Primary-Secondary session replication • For large cluster	org.apache.catalina.session.BackupManager

DeltaManager 메카니즘

- Session Replication

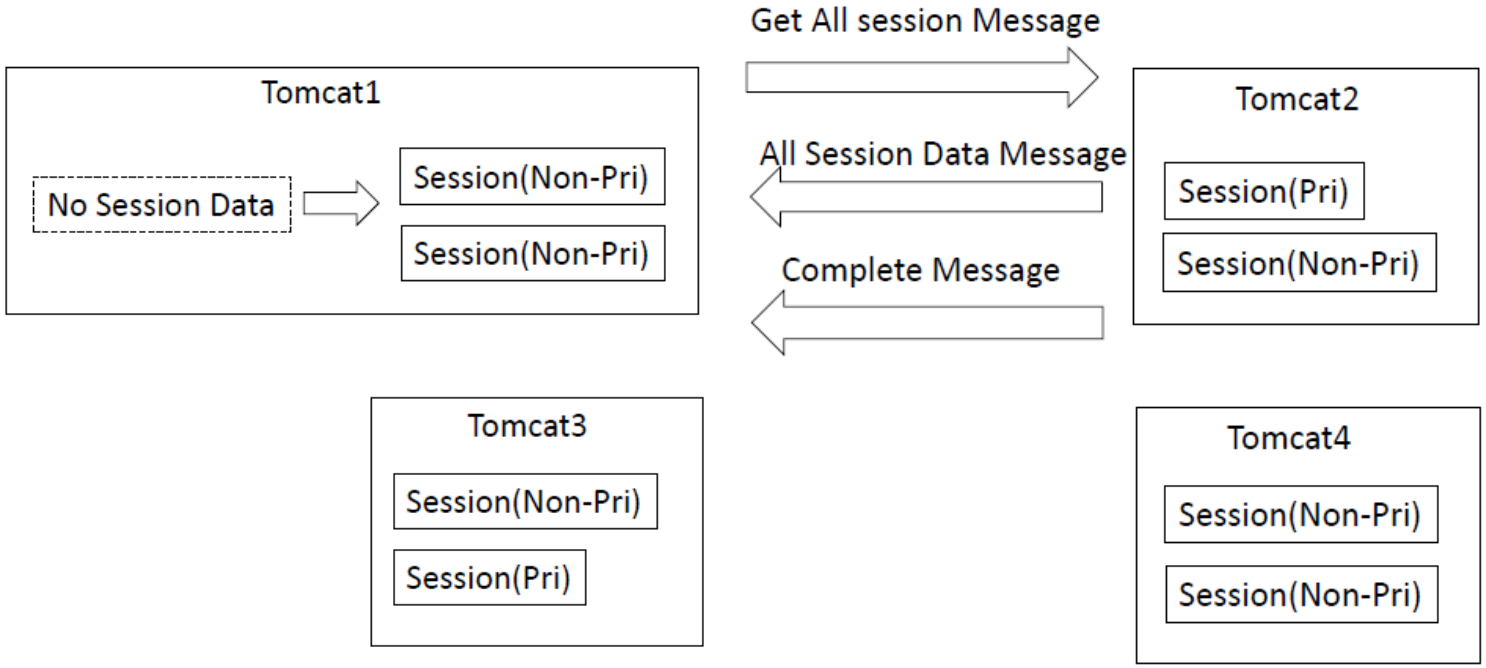


- Failover



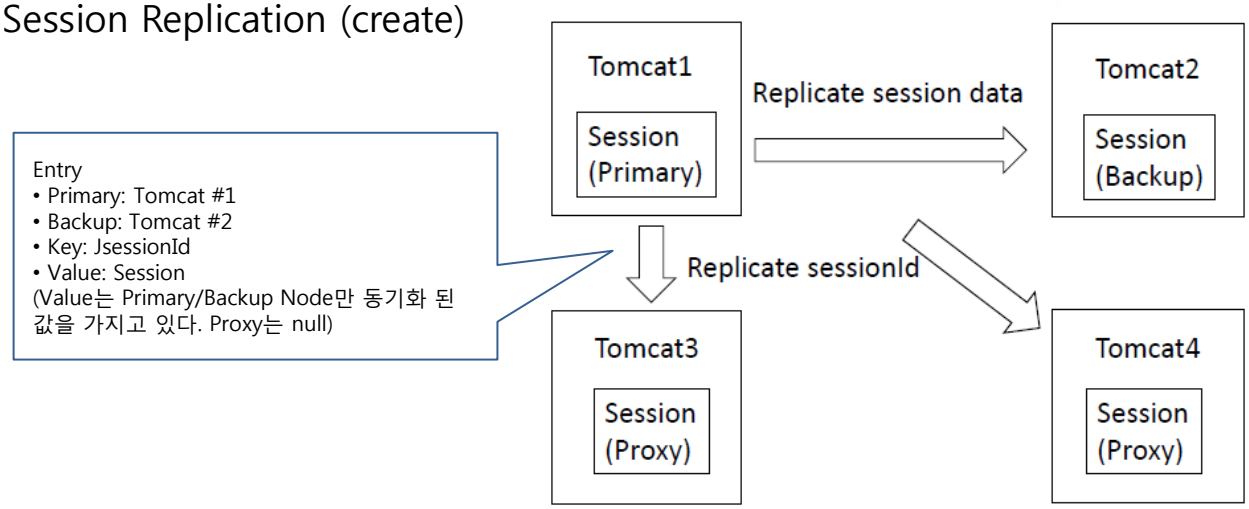
DeltaManager 메카니즘

- Node Recovery

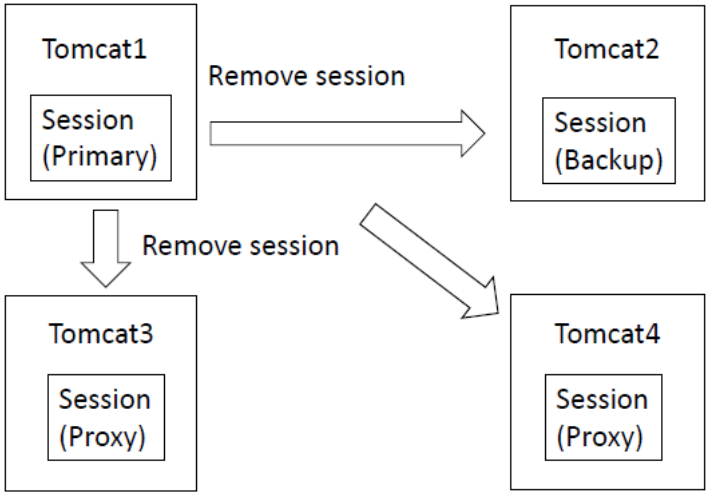


BackupManager 메카니즘

- Session Replication (create)



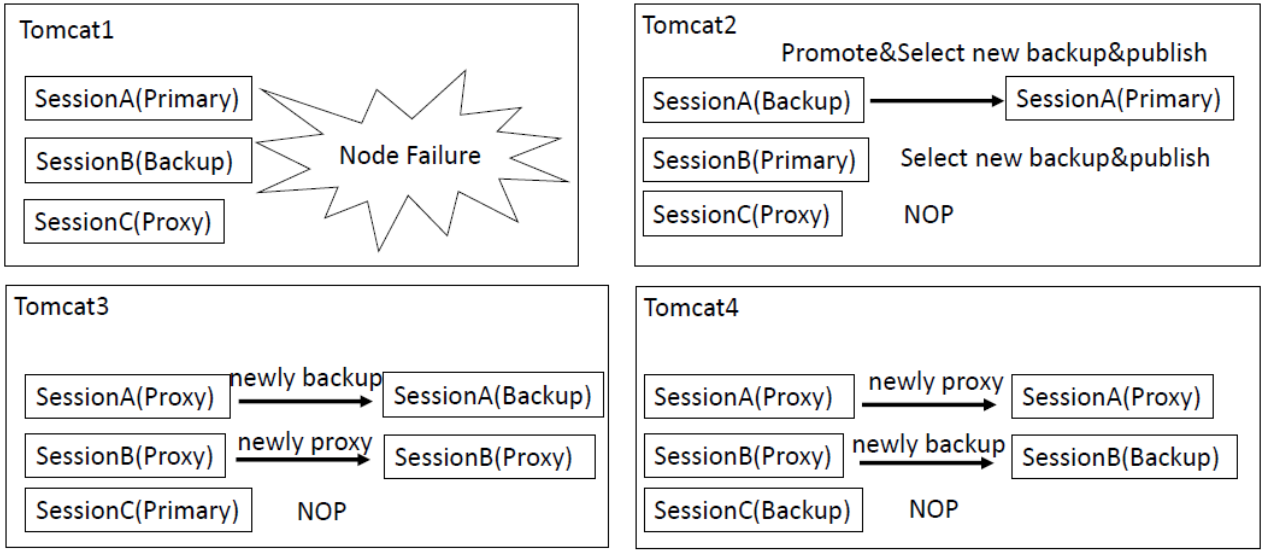
- Session Replication (destroy)



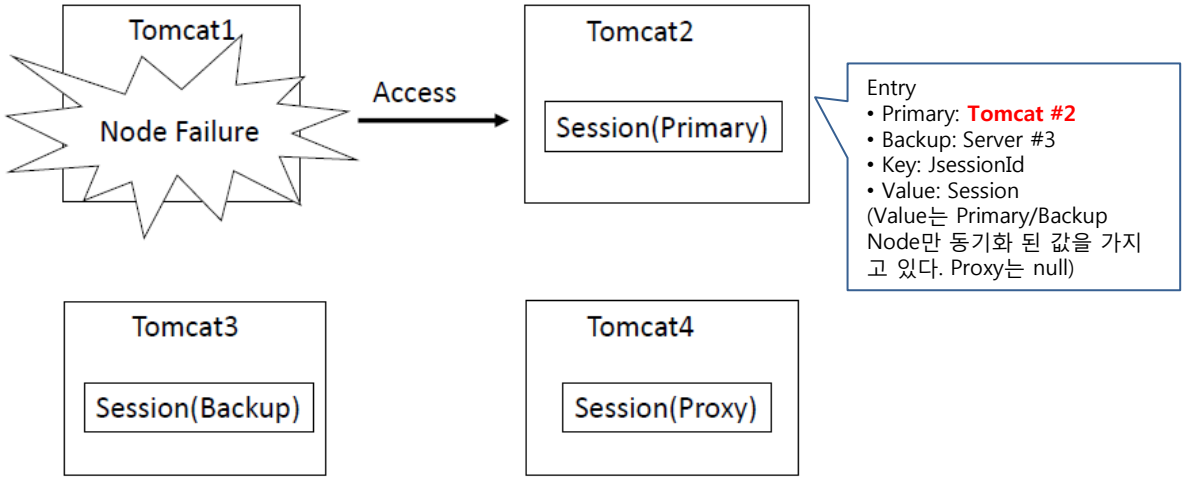
BackupManager 메카니즘

- Failover

  Node Failure
  → Tomcat1

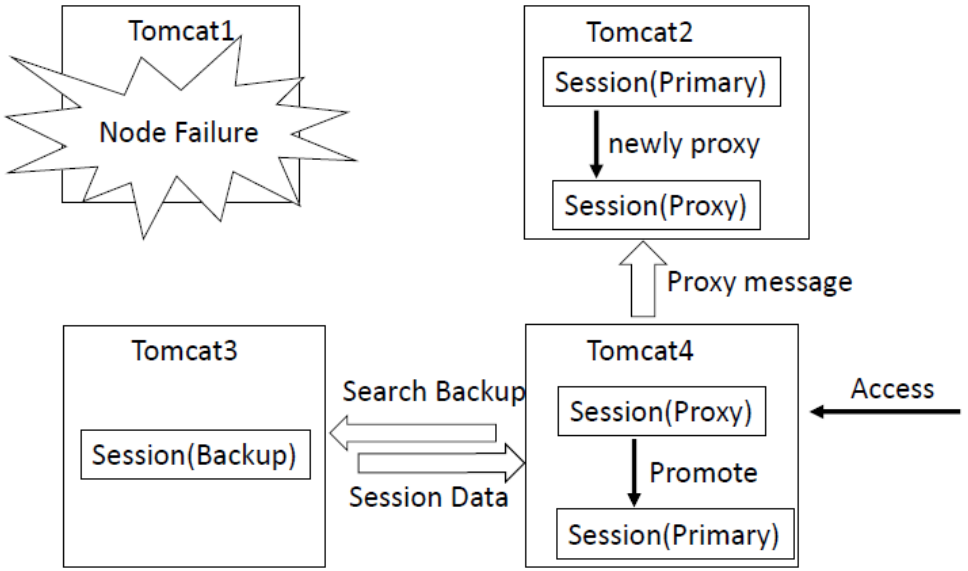


- Access to Primary node

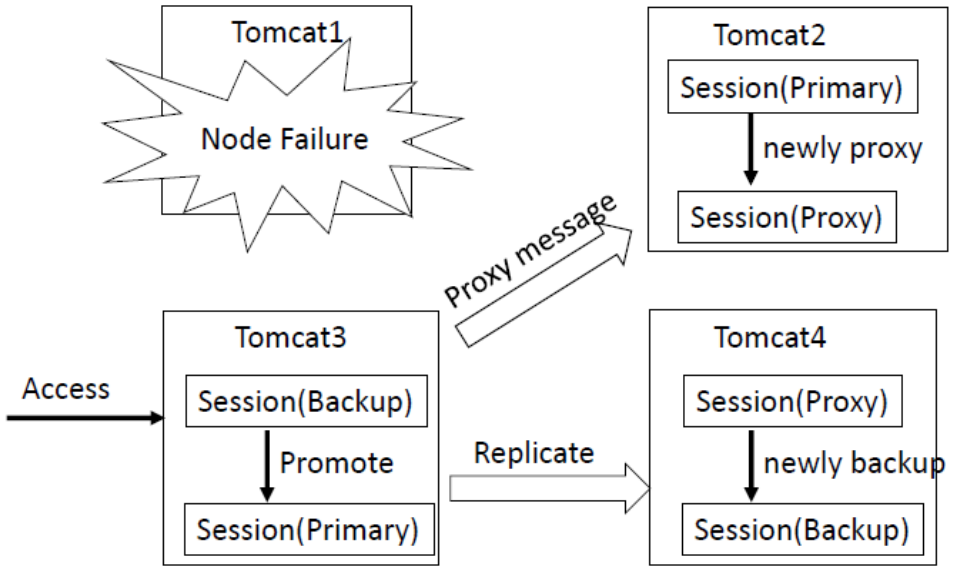


BackupManager 메카니즘

- Access to Backup node



- Accessto Proxy node



Tomcat Cluster Configuration

- Configuration

Tomcat Clustering을 위해서는 conf/server.xml 의 <Engine> 또는 <Host> 에 설정한다.

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

위 설정은 아래와 동일하다.

```
<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.MessageDispatch15Interceptor"/>
    </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” temp
         watchDir="/tmp/war-listen/“ watchEnabled="false"/>

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

Session Manager 설정
•DeltaManager
•BackupManager

동일 Cluster Group으로 설정하기 위해서는 address와 port (Multicase 주소) 가 동일해야 함.

Cluster Message를 받는 Receiver 설정
Port는 충돌방지를 위해 자동 bind.
port <= **자동설정된port** < port + autoBind

Http Request 중 data 복제 여부를 설정. txt, html 등 일 땐 session이 변경되지 않으므로 filter에 설정하면 됨.

mod_jk Failover를 썼을 경우, sessionId 뒤의 jvmRoute 값을 바꿔줌.

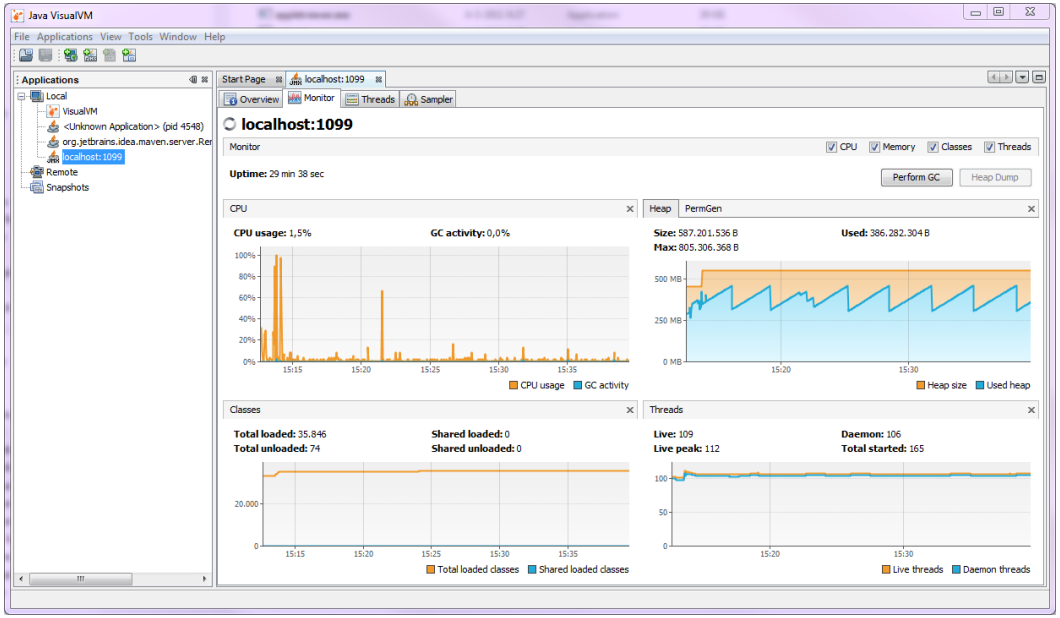
추가로, Application/WEB-INF/web.xml 에 <distributable/> 을 추가해준다.

Monitoring & Management

- Tomcat Manager
- JK Manager
- Visual VM
- Scouter

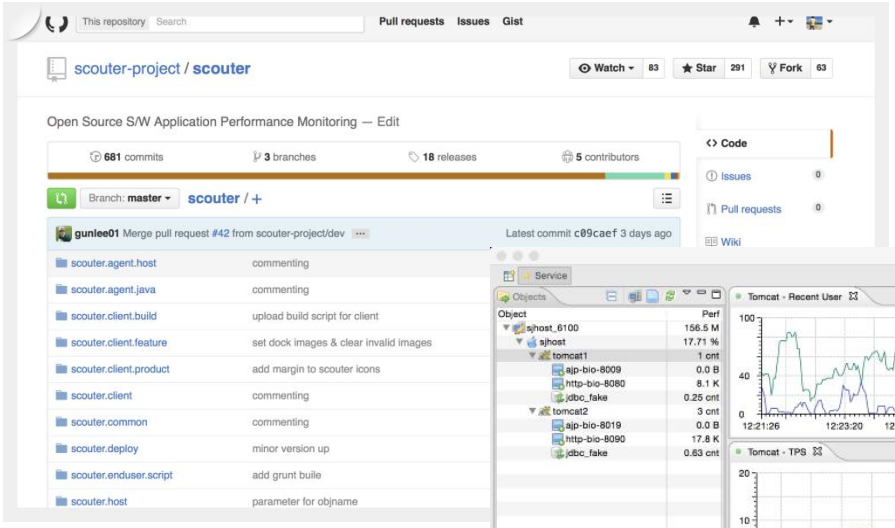
VisualVM

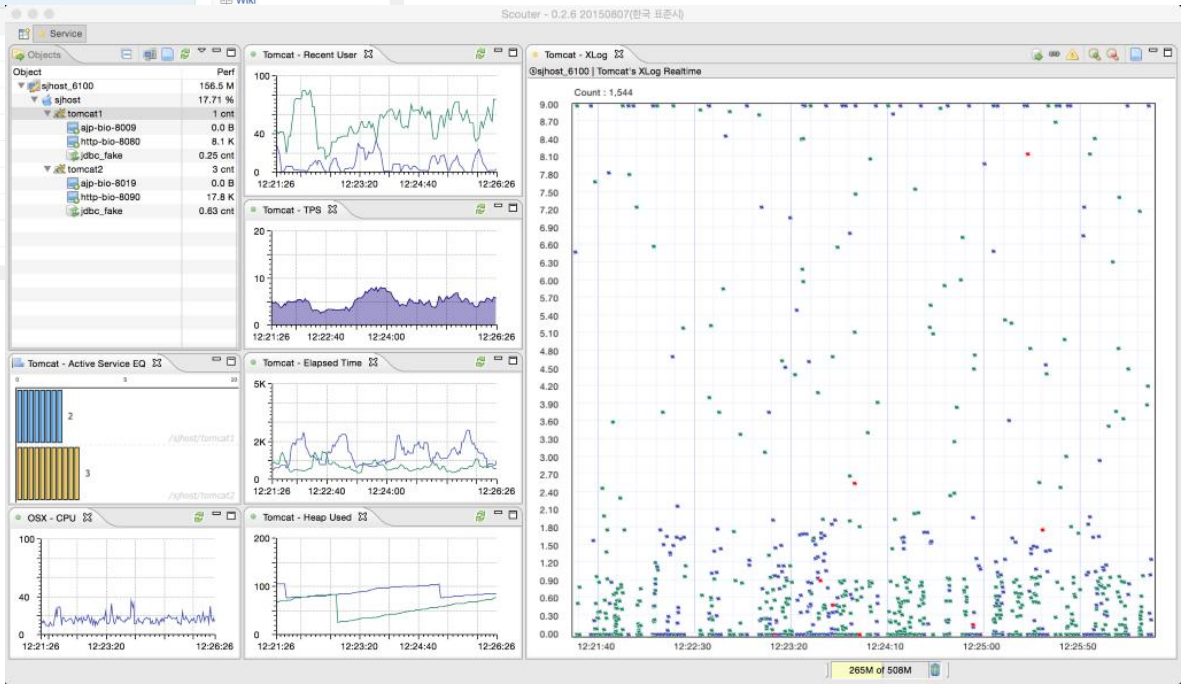
- JVM 전반에 대해 모니터링을 제공하는 툴
- JVM에 포함($JAVA_HOME/bin/jvisualvm)되어 있으며, 공식 홈페이지에서도 다운로드 가능 (http://visualvm.java.net/)
- 제공 기능
  - Display local and remote Java applications
  - Display application configuration and runtime environment
  - Monitor application threads, application performance and memory consumption
  - Take and display thread dumps
  - Take and browse heap dumps



Scouter

https://github.com/scouter-project/scouter

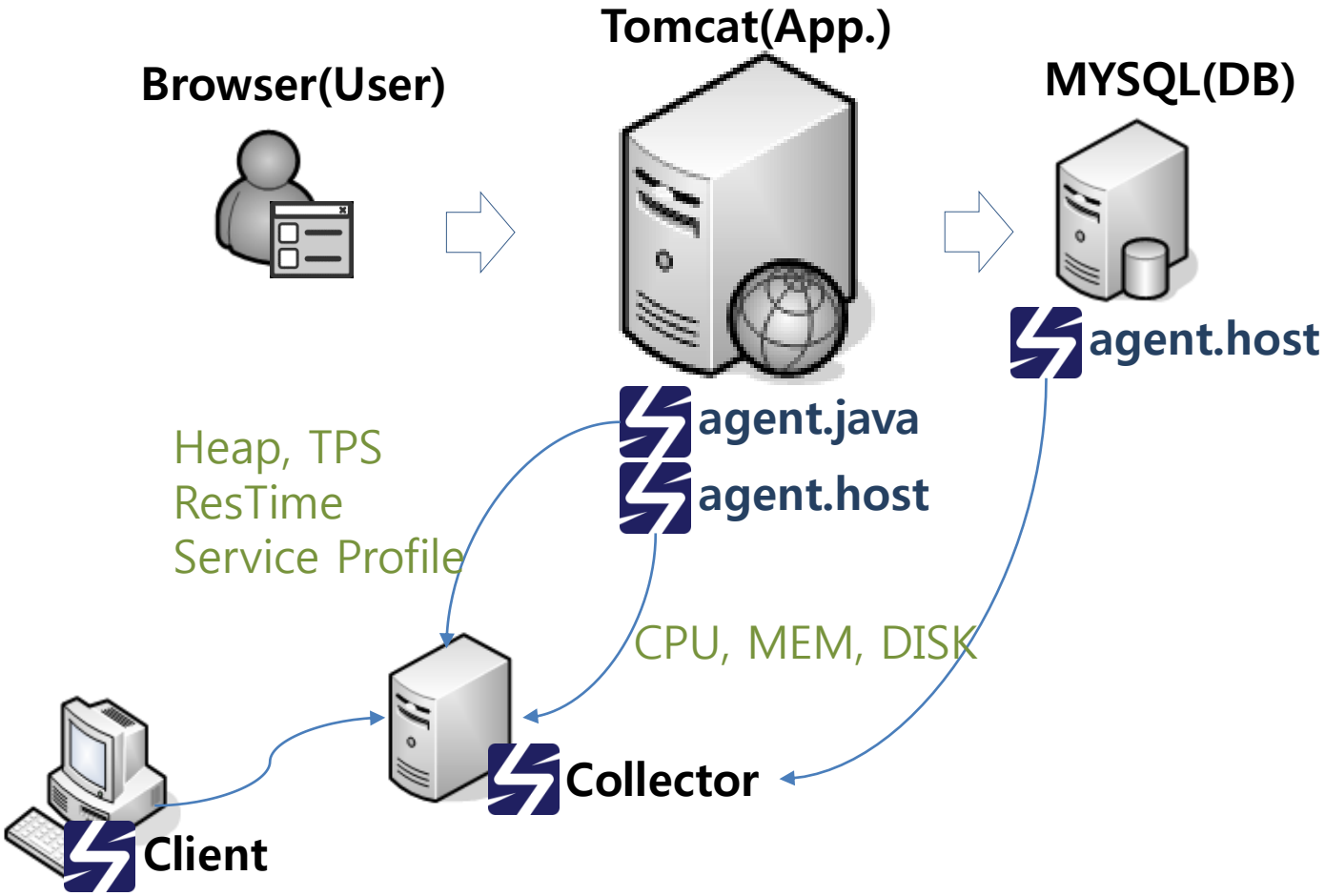




- 2015.7.1 initial commit
- Apache 2 License
- Java/Scala/Eclipse-RCP

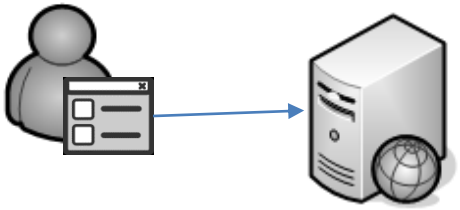
Scouter

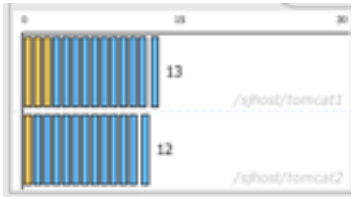
Web Application Monitoring(APM)

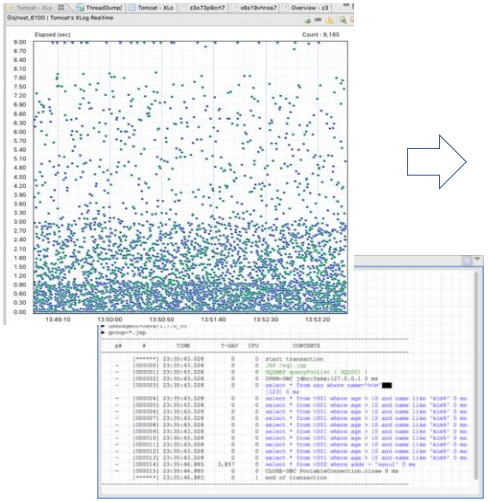


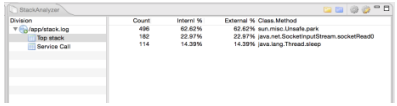
Scouter

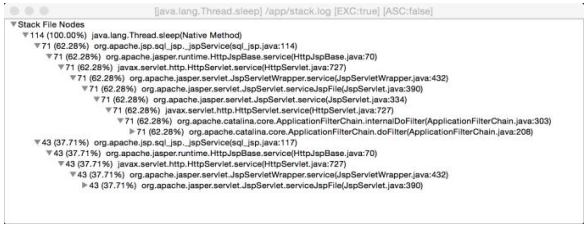
In a short period of time, the system can be stable.









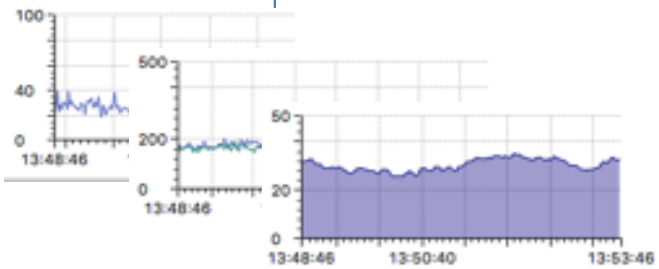


Active Service & Control

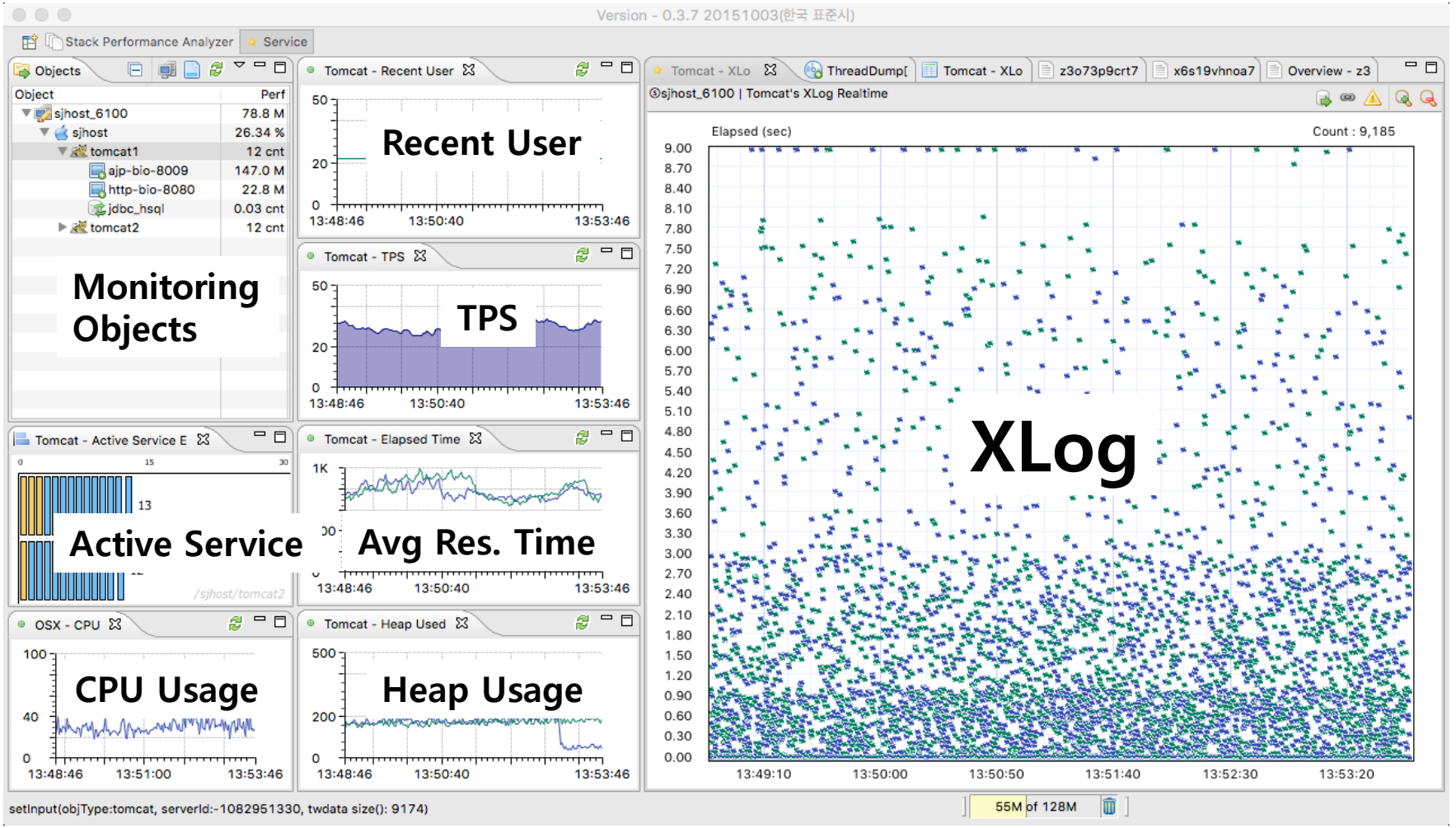
Indivisual Request Service Perf. & Profile

Top-Stack Frequecy

Resource Usage & Service Perf.



Scouter Dashboard



Tomcat 운영 노하우

- 상용 WAS 전환
- 웹서버 연동 Trobleshtting
- Multi Instance 관리 방안
- 기타 운영 TIP

U2L 전환에 따른 OSS WEB/WAS 전환

- L사는 온라인 비즈니스 매출 증대로 시스템 증설 수요가 있었으며, 기존 Unix 및 상용 WEB/WAS로는 많은 투자비가 발생하게 되어, OSS의 전환
  - 인터넷쇼핑몰 등 총 22개 업무를 CentOS/Apache/Tomcat으로 전환함

U2L 전환에 따른 OSS WEB/WAS 전환

- **Tomcat 전환 이슈 & 해결**

  - HTTPS 에서 생성된 세션을 HTTP 로 이동 시 인식하지 못하는 현상
    (반대의 경우, 즉 HTTP 에서 생성된 세션은 HTTPS 에서도 정상적으로 사용 가능)
    → Tomcat Valve 추가 개발 (HTTPS 세션 생성 시, HTTP 세션 생성 기능)

  - JSP 한글 깨짐
    → web.xml 의 jsp-property-group내 page-encoding 설정

  - DB Password 노출
    → Tomcat 기본적으로 DataSource 설정 내 DB Password 가 Plain Text 로 노출되어 있어 이를 암호화하여 관리하기 위한 별도의 클래스 배치

U2L 전환에 따른 OSS WEB/WAS 전환

- Apache 전환 이슈 & 해결

  - Tomcat Connector 의 Tomcat VM 다운 인식 지연
    → OS (VM) 이 다운되어 있는 경우 Tomcat Connector 가 이에 대한 인식까지 지연 현상 발생
    → ICA (IP Table Control for Apache) 적용하여 해결함

  - 간헐적 서비스 지연 현상 발생
    → 이미지 서버에 mod_expires 를 통한 이미지 브라우저 캐시 기능 적용
    → js, css 파일 대상으로 mod_deflate 적용을 통한 압축 기능 적용

- 응용 전환 이슈 & 해결

  - <url-pattern> 중복 오류
  - 태그 라이브러리 버전 호환 오류
  - JSP include tag 사용 오류
  - 상이한 contentType 에 의한 오류
  - JSP 내 공백이 없어 발생한 오류

U2L 전환에 따른 OSS WEB/WAS 전환

- L사 쇼핑몰 전환 사례 – 성능 분석
  - 인터넷 쇼핑몰 병행 운영 중에 Jeus와 Tomcat의 서비스 응답시간을 Scouter APM툴로 비교해본 결과 큰 차이가 없음

U2L 전환에 따른 OSS WEB/WAS 전환

- U2L 전환 사례

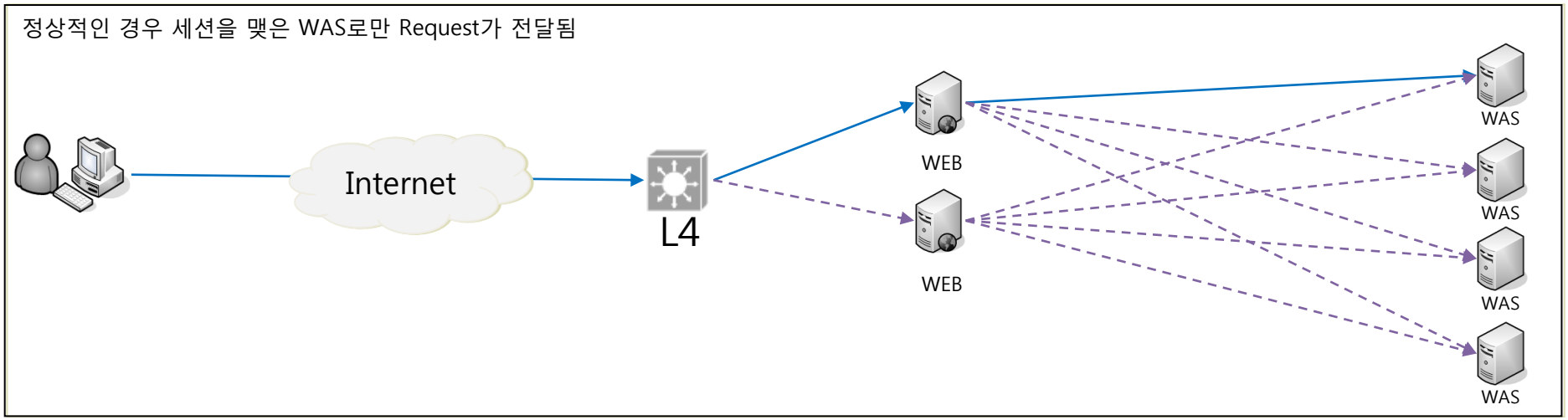
  - 기존에 소규모의 비핵심 업무에만 적용했던 OSS를 중요/핵심 업무에도 적용함으로써 대규모 Enterprise환경에도 안정적인 운영이 가능하다는 사례를 보여 줌

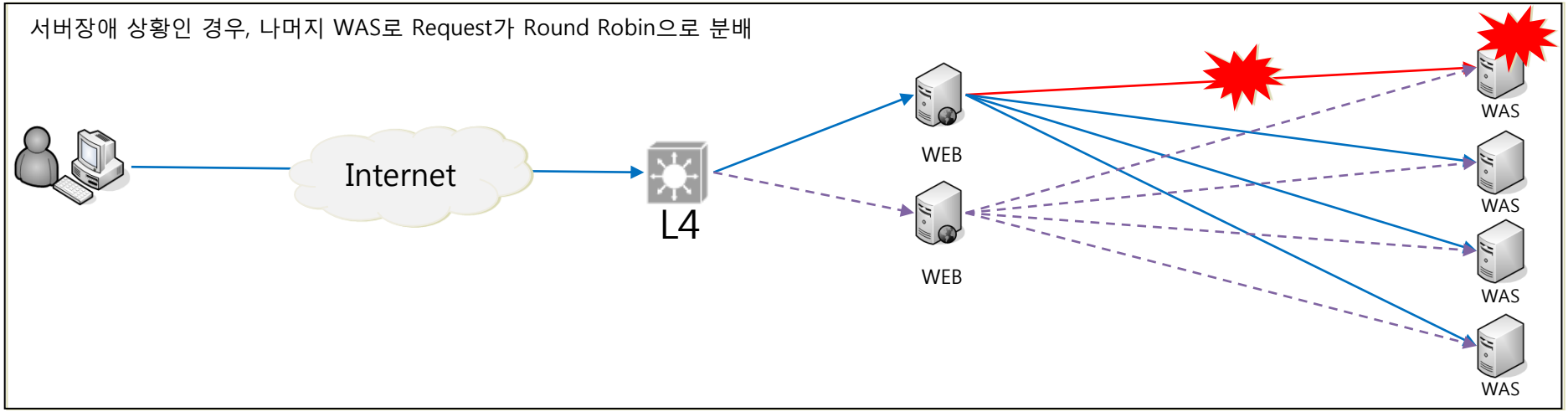
  - 기존 오픈소스에서는 부족한 기능을 보완할 수 있는 추가적인 솔루션을 개발하여 제공함으로써 추가적인 고객 value를 제공 가능하였음

  - Scale-out이 용이한 WEB/WAS 시스템은 서버 한대의 다운으로 인한 서비스 영향이 거의 없기 때문에 적극적인 OSS전환 전략 수립이 가능해 보임

웹서버 연동 Trobleshtting

- 웹서버 연동 이슈 (mod_jk Fail-over 메커니즘)
  - WEB서버에 sticky로 등록된 WAS를 재시작하거나 소스를 배포하는 경우 세션이 나머지 WAS로 Round Robin방식으로 분배되어 불필요하게 세션이 늘어나는 증상이 발생





웹서버 연동 Trobleshtting

- mod_jk worker lb(Load Banancer) : was와 직접적으로 통신하지는 않고, 멤버로 있는 실제 worker에 대한 관리를 수행

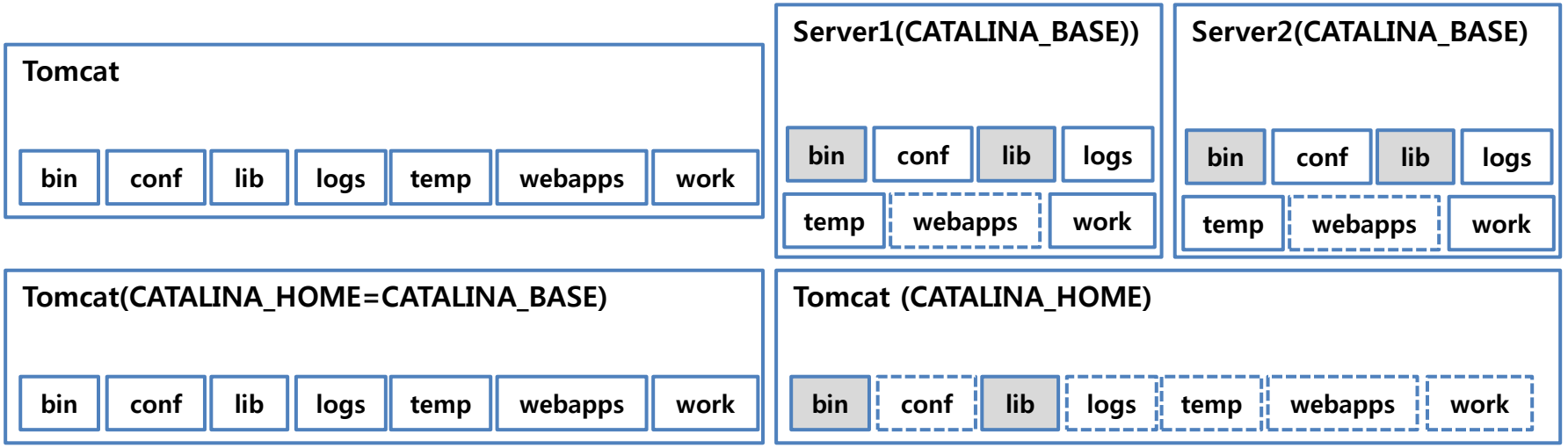
- lb(Load Banancer)
  - worker를 초기화
  - load-balancing factor를 이용하여 부하분산
  - sticky세션 처리
  - 특정 worker에 장애발생 시 다른 worker로 failover
  - load balancer와 멤버의 상태정보를 제공
  .

```
# The load balancer worker balance1 will distribute
# load to the members worker1 and worker2
worker.balance1.type=lb
worker.balance1.balance_workers=worker1, worker2
worker.worker1.type=ajp13
worker.worker1.host=myhost1
worker.worker1.port=8009
worker.worker2.type=ajp13
worker.worker1.host=myhost2
worker.worker1.port=8009
```

```
# The advanced router LB worker
worker.list=router
worker.router.type=lb
worker.router.balance_workers=worker1,worker2

# Define the first member worker
worker.worker1.type=ajp13
worker.worker1.host=myhost1
worker.worker1.port=8009
# Define preferred failover node for worker1
worker.worker1.redirect=worker2

# Define the second member worker
worker.worker2.type=ajp13
worker.worker2.host=myhost2
worker.worker2.port=8009
# Disable worker2 for all requests except failover
worker.worker2.activation=disabled
```

Multi Instance 관리 방안

- Tomcat 배포판(엔진) 하나 설치, ${CATALINA_BASE} 를 분리
  - CATALINA_HOME 공유, CATALINA_BASE 분리
  - 서버 library 공유, shell 재사용 가능, 설정 분리
  - 장점 : 서버 라이브러리 및 설정 파일 공유로, 관리 포인트가 줄어듬
  - 단점 : 서버 라이브러리 공유로 패치 및 버전 업그레이드시 전체 서버에 영향을 미침



[서버 N개 서리 방식]

[CATALINA_HOME 공유, CATALINA_BASE 분리 방식]

JAVA_OPTS vs CATALINA_OPTS

- Tomcat의 경우 JAVA 옵션 설정 시 JAVA_OPTS, CATALINA_OPTS 모두 사용할 수 있으며, 차이는 다음과 같음
  - **JAVA_OPTS → 기동/중지 시 모두 적용**
    → Catalina Java Options 등

ex)

```
## Catalina Java Options (don't modify them)
JAVA_OPTS=" ${JAVA_OPTS} -server"
JAVA_OPTS=" ${JAVA_OPTS} -DjvmRoute=${JVM_ROUTE}"
JAVA_OPTS=" ${JAVA_OPTS} -Dwas_name=${INST_NAME}"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.http=${SERVICE_PORT}"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.https=`expr ${SERVICE_PORT} + 363`"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.ajp=`expr ${SERVICE_PORT} - 71`"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.shutdown=`expr ${SERVICE_PORT} - 75`"
```

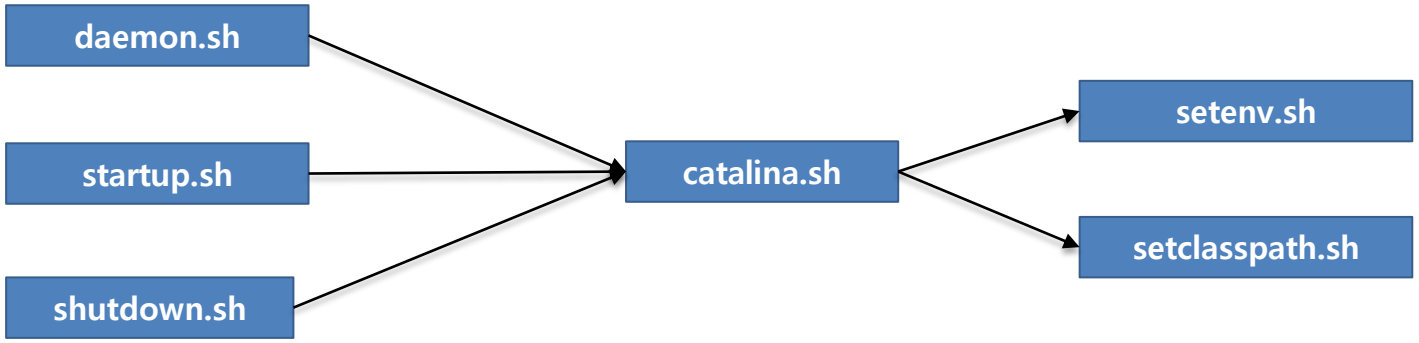
  - **CATALINA_OPTS → 기동 시에만 적용됨**
    → JVM Memory Options, Business System Java Options

ex)

```
## JVM Memory Options (tune them)
CATALINA_OPTS=" ${CATALINA_OPTS} -Xms2048m -Xmx2048m -XX:MaxPermSize=256m"
CATALINA_OPTS=" ${CATALINA_OPTS} -verbose:gc"
CATALINA_OPTS=" ${CATALINA_OPTS} -Xloggc:${LOG_HOME}/gc_${INST_NAME}.${DATE}.log"
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:+UseParallelGC"
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:+UseParallelOldGC"
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:-UseAdaptiveSizePolicy"
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:+PrintGCDetails"
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:+PrintGCDateStamps“
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:+ExplicitGCInvokesConcurrent"
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:+HeapDumpOnOutOfMemoryError"
CATALINA_OPTS=" ${CATALINA_OPTS} -XX:HeapDumpPath=${LOG_HOME}/hdump“

## Business System Java Options (for your application)
```

setenv script

- setenv 스크립트 활용
  - **Catalina 스크립트 내에서 setenv 스크립트 실행됨**
  - Tomcat 실행 스크립트 관계



  - Catalina 스크립트

```
# -----------------------------------------------------------------------------
# Control Script for the CATALINA Server
#
# Environment Variable Prerequisites
#
#   Do not set the variables in this script. Instead put them into a script
#   setenv.sh in CATALINA_BASE/bin to keep your customizations separate.
#
# -----------------------------------------------------------------------------

if [ -r "$CATALINA_BASE/bin/setenv.sh" ]; then
  . "$CATALINA_BASE/bin/setenv.sh"
elif [ -r "$CATALINA_HOME/bin/setenv.sh" ]; then
  . "$CATALINA_HOME/bin/setenv.sh"
fi
```

Configuration Template 적용

- Tomcat 설정 파일을 Template 방식으로 사용
  - 공통 include shell 예제

ex)

```
## Catalina Java Options (don't modify them)
JAVA_OPTS=" ${JAVA_OPTS} -server"
JAVA_OPTS=" ${JAVA_OPTS} -DjvmRoute=${JVM_ROUTE}"
JAVA_OPTS=" ${JAVA_OPTS} -Dwas_name=${INST_NAME}"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.http=${SERVICE_PORT}"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.https=`expr ${SERVICE_PORT} + 363`"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.ajp=`expr ${SERVICE_PORT} - 71`"
JAVA_OPTS=" ${JAVA_OPTS} -Dport.shutdown=`expr ${SERVICE_PORT} - 75`"
```

  - Tomcat 설정 파일 (Server.xml)

ex)

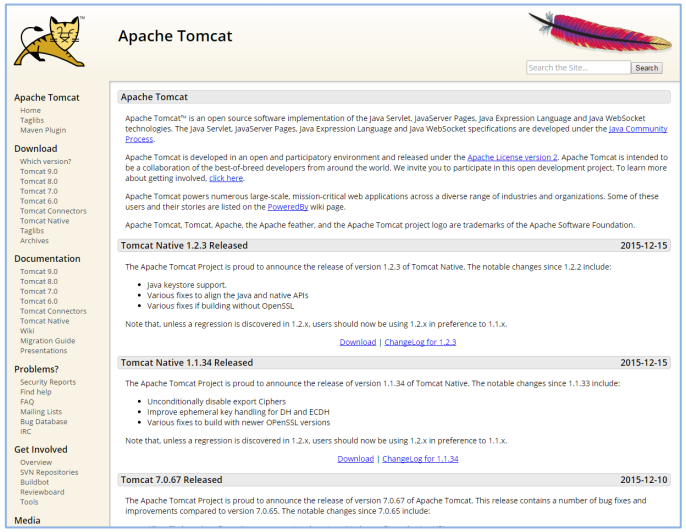
```
<?xml version='1.0' encoding='utf-8'?>
<Server port="${port.shutdown}" shutdown="ARGO_INSTANCE_SHUTDOWN">
  ......
  <Service name="Catalina">
    <Connector port="${port.http}" protocol="HTTP/1.1" server="Server"
               connectionTimeout="20000" URIEncoding="UTF-8"
               redirectPort="${port.https}" />

    <Connector port="${port.ajp}" protocol="AJP/1.3" server="Server"
redirectPort="${port.https}" maxPostSize="-1"
               maxThreads="256" connectionTimeout="60000" URIEncoding="UTF-8"
               acceptCount="10" enableLookups="false" tcpNoDelay="true" />
    <Engine name="Catalina" defaultHost="localhost">
    ......
      <Host name="localhost" unpackWARs="true" autoDeploy="false">
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="${log.home}"
               prefix="access_${was_cname}." suffix=".log"
               pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b %D" />
      </Host>
    </Engine>
  </Service>
</Server>
```

Tomcat Community 소개

- Tomcat 커뮤니티 소개
- Tomcat Build
- Tomcat 확장하기

Tomcat 공식 가이드 문서

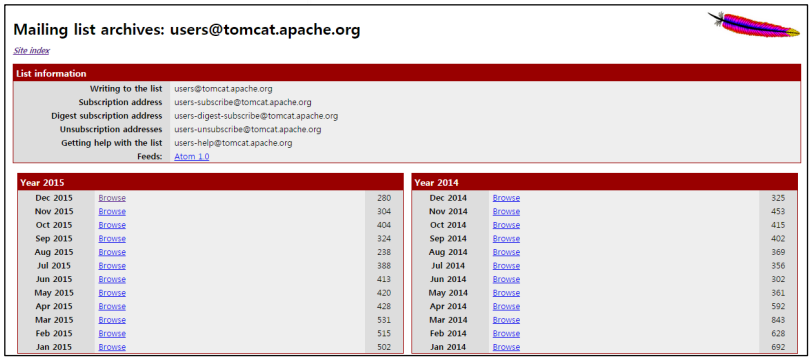
- Tomcat Major Version 별 Documentation 제공
  - Tomcat8 : http://tomcat.apache.org/tomcat-8.0-doc/



- Tomcat Connectors : mod_jk Documentation – http://tomcat.apache.org/connectors-doc/
- Tomcat Native - http://tomcat.apache.org/native-doc/
- Tomcat Wiki – http://wiki.apache.org/tomcat/FrontPage
- Migration Guide – http://tomcat.apache.org/migration.html
- Security Report – http://tomcat.apache.org.security.html

Tomcat Mailing Lists

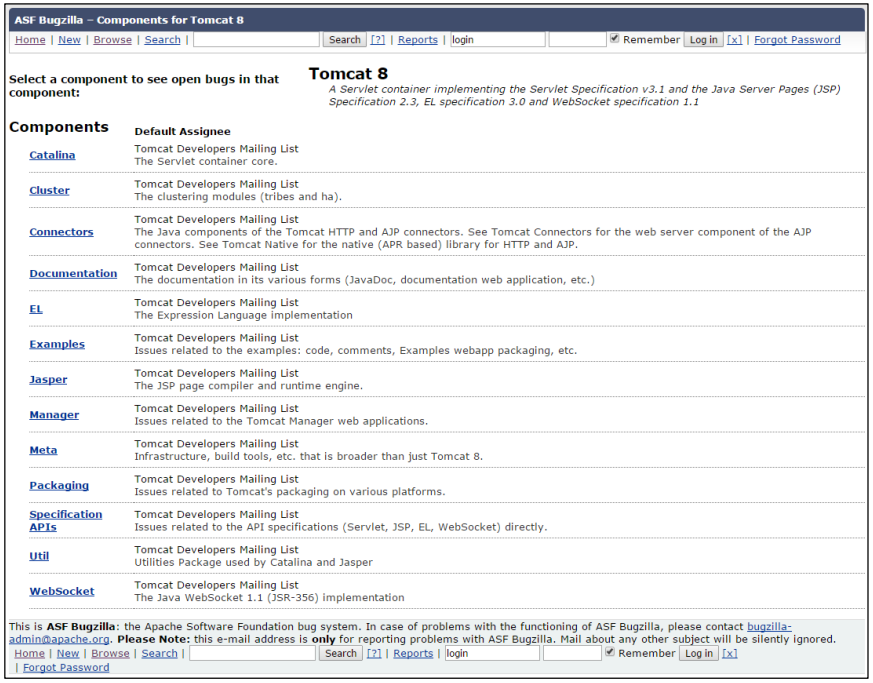
- Mailing List Guide : http://tomcat.apache.org/lists.html
  - Tomcat에 대한 공식 Mailing List 정보 제공



- Mailing List
  - tomcat-users (users@tomcat.apache.org)
    - subscribe: users-subscribe@tomcat.apache.org
    - Tomcat 사용자를 위한 Mailing List
  - tomcat-dev (dev@tomcat.apache.org)
    - subscribe: dev-subscribe@tomcat.apache.org
    - Tomcat의 개발,구현 및 세부기능에 대한 논의하기 위한 Mailing List
  - tomcat-announce (announce@tomcat.apache.org)
    - subscribe: announce-subscribe@tomcat.apache.org
    - Tomcat 릴리즈 정보 및 취약점 정보를 Announce 하기 위한 Mailing List
  - taglibs-user (taglibs-user@tomcat.apache.org)
    - subscribe: taglibs-user-subscribe@tomcat.apache.org
    - Apache Taglibs(http://tomcat.apache.org/taglibs/) 사용자에 대한 Mailing List

Tomcat Bug Database

- **Bug Report Guide - http://tomcat.apache.org/bugreport.html**
- Tomcat Community 는 버그 데이타베이스로 Bugzilla를 사용
- Bugzilla를 통하여, 버그 검색 및 버그 등록을 통한 수정 요청을 할 수 있음



END