JBS-102

JBOSS 6.x Adminstration

Agenda

- 1.Introduction
  - J2EE introduction
  - JBoss directory structure
- 2. Basic deployments
- 3. Customizing JBossWeb (Tomcat)
- 4. Class-loading in JBoss
- 5.Managing datasources
- 6. Managing security
- 7. JMS in JBoss
- 8.Managing Enterprise Beans
- 9. JBoss WS frameworks
- 10. Monitoring JVM and JBoss
- 11. Clustering

1. Introduction into J2EE

Objectives

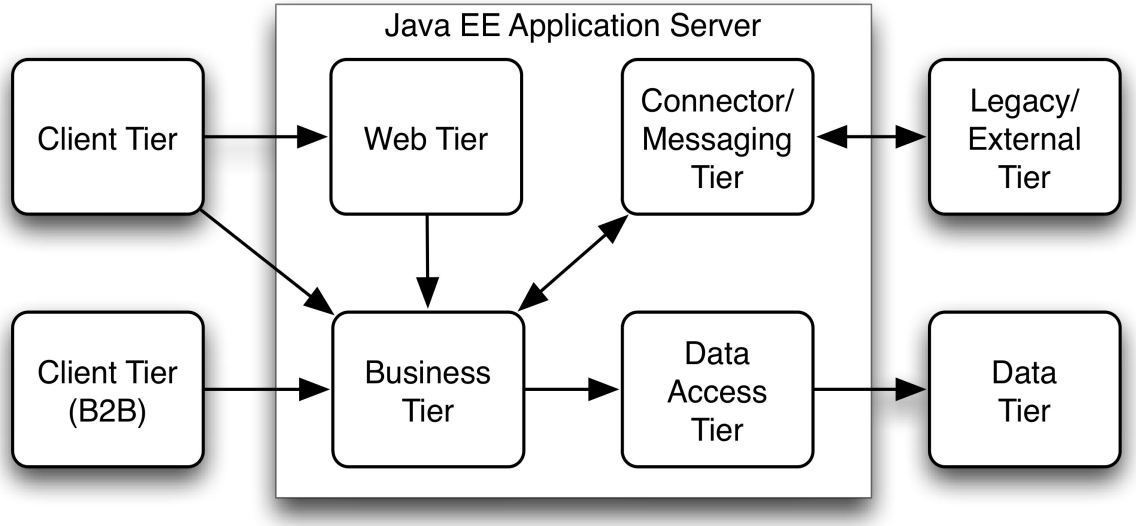
-
  - J2EE architecture elements
  - Related standards
  - Defining J2EE application server
  - JBoss AS
    - Flavours
    - Directory layout
    - Main configuration entries

What is Java EE?

- Java EE is an open and standard-based platform for developing, deploying, and managing multi-tier, web-enabled, server-centric, and component-based enterprise applications
- Super-set of Java SE, Java EE adds additional specifications (JSR), libraries, documentation, and tools.

J2EE overview

- OO framework for integrating together:
  - Clients (e.g. web browsers)
  - Access and presentation components
  - Business components (e.g. ecommerce business logic)
  - Data-stores (e.g. RDBMS)
  - Legacy systems (e.g. order fulfillment)



J2EE overview

- Client Tier (B2B) is a set of external applications that makes requests into the Business Tier through Web Services over SOAP or directly through Java's RMI.
- Web Tier is usually implemented with Servlets, JSPs, and simple JavaBeans, based on the Model-View-Controller design pattern.
- Business Tier is composed of EJBs and/or plain old Java objects (POJOs).

J2EE overview

- Data Access Tier is either managed by the applications (e.g. BMP, DAO), O/R mapping tools (e.g. Hibernate), or through Container Managed Persistence (CMP).
- Connector/Messaging Tier allows asynchronous access to legacy systems or other external systems.
- Legacy/External Tier consists of enterprise information systems.
- Data Tier is usually composed of RDBMS and/or LDAP Directory Server

J2EE Components (not complete set)

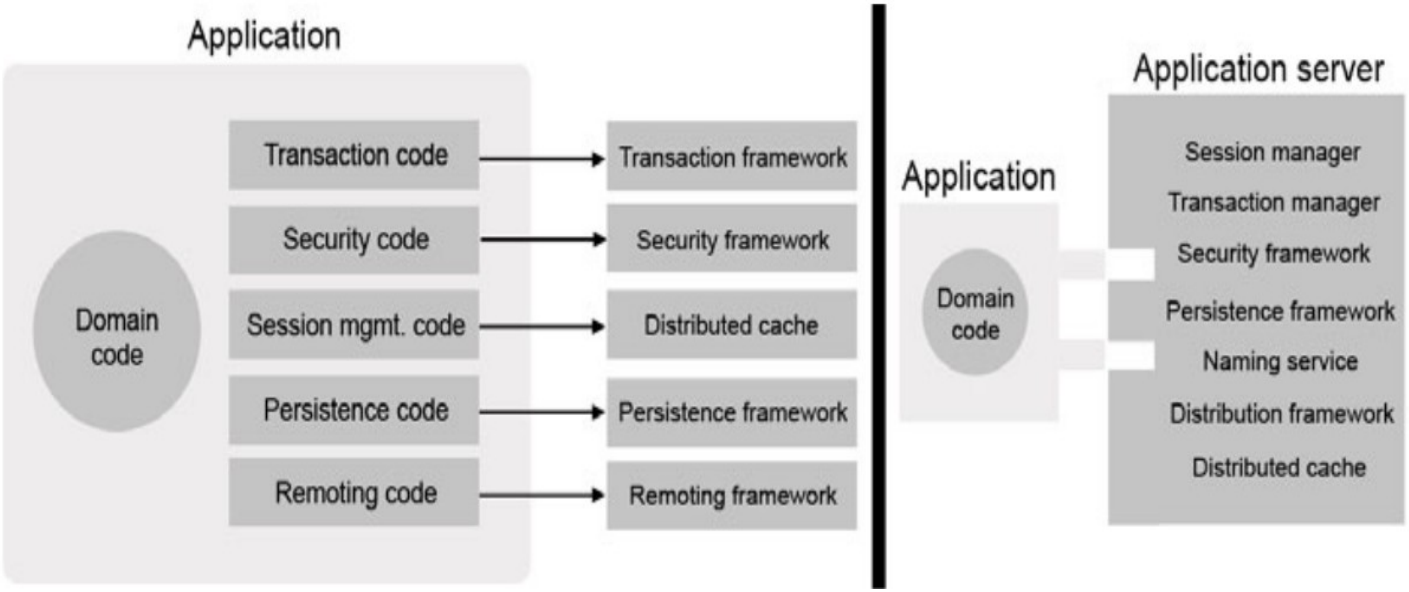
- Java SE 5
- Presentation tier
  - Web Services 1.1, JSR-109
  - Java API for XML-Based RPC (JAX-RPC) 1.1, JSR-101
  - JavaServer Pages (JSP) 2.1, JSR-245
  - JavaServer Faces (JSF) 1.0, JSR-127
  - JavaServer Pages Standard Tag Library (JSTL) 1.1, JSR-52
  - Java Servlet 2.5, JSR-154
- Container specific
  - Management API, JSR-77
  - Deployment API, JSR-88
  - Java Management Extensions (JMX) 1.1, JSR-3
  - Java Naming and Directory Interface (JNDI) 1.2.1

J2EE components

- Persistence/session mgt
  - J2EE Connector Architecture (JCA) 1.5, JSR-112
  - Java Transaction API (JTA) 1.0.1, JSR-907
  - Enterprise Java Beans (EJB) 3.0, JSR-220
- Messaging
  - Java Messaging Service (JMS) 1.1, JSR-914
  - Java Mail 1.3. JSR-919
- Security
  - Java Authentication and Authorization Service (JAAS) 1.0
  - Java Authorization Contract for Containers (JACC) 1.0, JSR-115
  - Authorization SPI, JSR-115

Defining J2EE application server

- an application server is a framework that extends your software with various features that often emerge as nonfunctional requirements (security, logging, measurements, transaction handling etc.)
- a container (wraps your code)
- supports + defines standardized component models



Application servers

- Commercial
  - BEA WebLogic
  - IBM Websphere
  - Oracle Application Server 11g
- Free/Open source
  - JBoss AS
  - Glassfish
- Container only
  - Apache Tomcat
  - Mort Bay Consulting Jetty

JBoss AS

- flavors?
  - JBoss AS
    - open source, free, buildable/costumizable
  - JEMS (JBoss Enterprise Middleware Suite)
    - Free but pay for support
    - JBoss AS + Tomcat
    - O/R Mapping and Persistence (Hibernate)
    - Portal Platform (JBoss Portal)
    - Business Process Management and Rules (JBoss jBPM, JBoss Rules)
    - Object/Data Cache (JBoss Cache)
    - Distributed Transaction Management (JBoss Transactions)

JBoss AS Background

- Established in 1999 as an open-source EJB container
- Version 2.x becomes a full J2EE application server
- Version 3.x is the production series - based on JMX microkernel and service-oriented architecture (May 2002)
- Version 4.x explores the aspect-oriented middleware services, adds support for EJB 3 (September 2004)
- Version 5.x comes with JBoss Microcontainer (also a stand-alone project) and replaces JMX Microkernel of the 3.x and 4.x JBoss series (December 2008)
- Version 6.x to add support for Java EE 6 APIs and Profiles. Replaces JBoss Messaging with HornetQ (6.0.0.M3 released in April 2010)

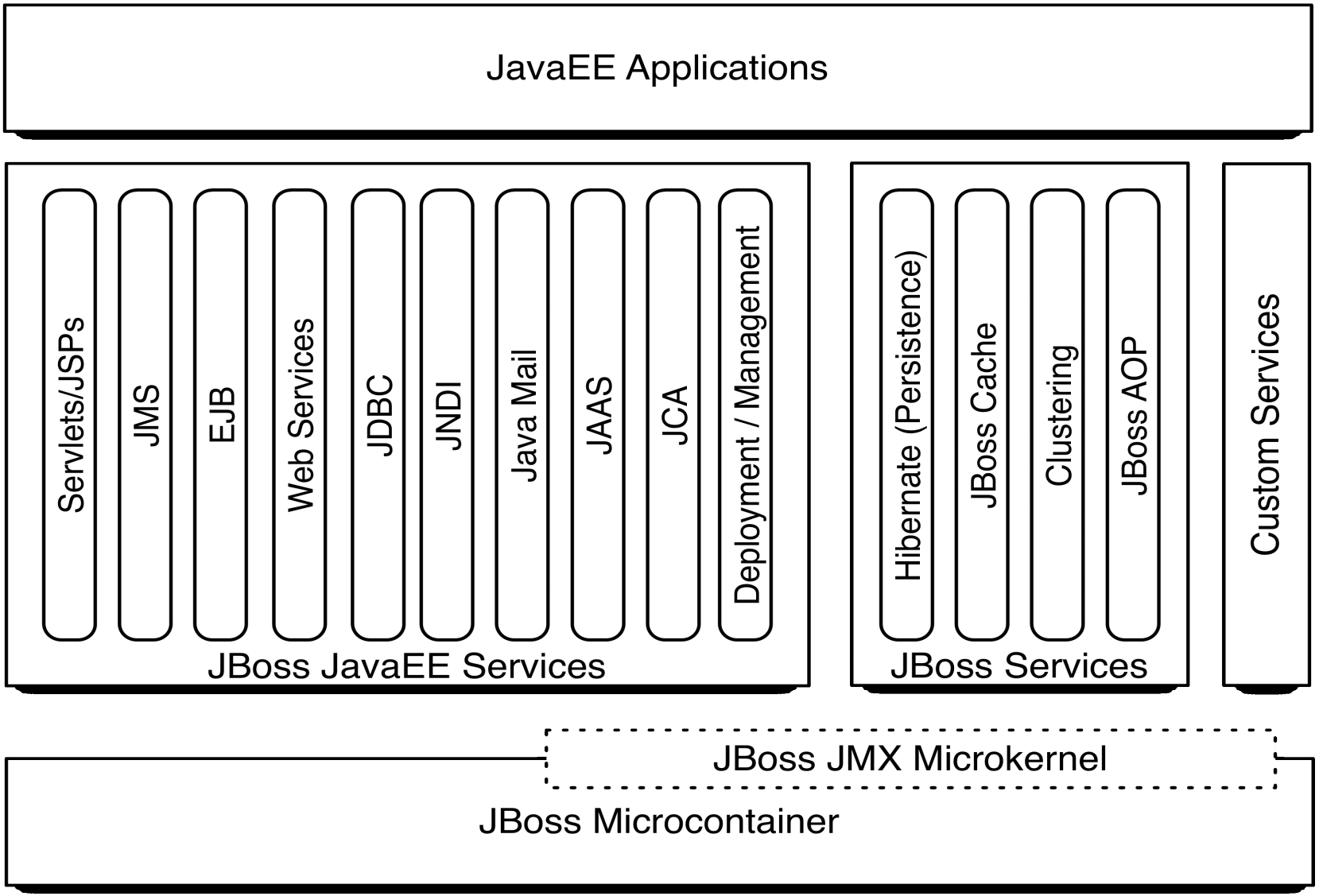
What is new in JBoss AS 5?

- New kernel ⇒ JBoss Microcontainer
  - is a refactoring of old JMX Microkernel (JBoss AS 3.x and 4.x)
  - the core of JBoss AS 5
- New messaging provider ⇒ JBoss Messaging
  - Replaces old JBossMQ (shipped with JBoss AS 4.x series)
- Support for EJB 3.0 specification
  - (dating back to 4.x series)
- Reliable transaction manager: JBoss TS
- JBoss Web based on Apache Tomcat 6.0
- JBoss WS 3.0 (support for JAX-WS/JAX-RPC)
  - can be replaced by Sun Metro or Apache CXF for example
    - in JBoss7 CXF is the main supported

What is new in JBoss AS 5?

- Two new configurations:
  - standard: Java EE compliant configuration.
  - web: provides support for JTA/JCA and JPA in addition to the Servlet/JSP container. The server can only be accessed through the http port.

JBoss 5 architecture



JBoss Microcontainer

- Replaces the JMX microkernel (but supports all the old features)
- Provides IoC framework
- POJO based kernel
  - no need for XMBean/MBeanProxy
- Simplified lifecycle management, additional control over depencencies
  - http://docs.jboss.org/ejb3/app-server/tutorial/service/service.html
- Transparent AOP
  - AOP provides deploy time support for the so called crosscutting concerns (logging, profiling, intercepting methods,etc.)

Services Layer

- POJO or JMX Managed Bean
  - JMX kernel is still available but provided by the microcontainer
- Services are hot-pluggable
  - New services are deployed as SAR or JAR
  - Undeploying / disabling on service shrink the overall footprint
    - eg. “web” service config/layout
    - Slimming JBoss discussed later

Aspect Layer

- Increase the modularity of an application by allowing the separation of cross-cutting concerns
  - (e.g logging is often required in many parts of your application).
  - Based on aspect-oriented programming model (AOP)
- Defines cross-cutting simple-to-use services
  - Makes it possible to add object persistence, caching, replication, security, etc. late in the development cycle by annotating existing POJOs
    - Annotate a POJO to act as an MBean
    - Annotate EJB containers to set default pool sizes
    - Annotate an EJB/POJO method for
      - profiling, logging
      - additional security restrictions

JBoss AS Services

- JBoss Microcontainer - POJOs services container
- JMX MBean server
  - One of the primary POJOs created by JBoss Microcontainer
- Web Application Services
  - based on Tomcat (Servlet, JSP, JSF)
- Enterprise Services: EJB, ORB, JNDI, JTA
- Web Services - based on SOAP, WSDL, UDDI, and XML
- Messaging Services: JMS, JDBC, JCA
- Persistence Services
  - Hibernate O/R mapping and transparent persistence
- HA Services
  - clustering, fail-over, load-balancing, distributed deployments
- Security Services - based on JAAS
- Console Services
  - monitoring, configuration, deployment, management, lifecycle

Installing JBOSS 5.x

- Get JAVA
  - JRE is enough, JSP can be compiled by an internal compiler
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - Set JAVA_HOME
- Get JBoss
  - http://www.jboss.org/jbossas/downloads
- Install
  - Console: simply unpack
  - Graphical: jemsinstaller
- Start it
  - JBOSS_HOME/bin/run.sh (run.bat)

JBoss directory structure

- ${jboss.home.dir}/bin
  - classpath.sh:
    - A tool to determine JBoss classpaths (both client and server)
  - jboss_init_XXX.sh
    - JBoss system control scripts for RedHat and SuSE systems
  - probe.{sh,bat}
    - used for discovering JBoss AS clusters
  - service.bat
    - Script to manage JBoss as a Windows service.
  - run{sh,bat,jar}
    - Scripts for starting JBoss AS
  - shutdown.{sh,bat,jar}
    - Scripts for shutting down JBoss AS (including remote instances) + bootstrap code

JBoss directory structure

- ${jboss.home.dir}/bin (cont)
  - twiddle.{sh,bat,jar}
    - JMX client script + bootstrap
  - wsconsume, wsprovide, wsrunclient and wstools
    - utilities for Web Services. We will see that later on Web Services chapter.
- ${jboss.home.dir}/client
  - Contains the Java libraries (JARs) required for clients that run outside the JBoss AS containers, such as
    - WebService clients
    - EJB clients
    - JMX clients
  - JBoss 5: client/jbossall-client.jar
    - contains references to all other JARs
    - referring this jar file is enough

JBoss directory structure

- ${jboss.home.dir}/common/lib
  - aka.(jboss.common.lib.url)
  - Contains jar files used by many configrations
    - like hsqldb.jar
  - Final directory for classloaders
- ${jboss.home.dir}/lib
  - Contains JBoss bootstrap libraries (core libraries)
    - like jboss-main.jar, jboss-jmx.jar, etc
  - Do not place your own files there
  - Sometimes repackaging run.jar is needed (hack)

JBoss directory structure

- ${jboss.home.dir}/server
    - aka. jboss.server.base.dir
  - location of configuration sets
    - minimal
      - just JNDI and Log4j: starting point for your own config set
    - default
      - Default J2EE 5 config
    - all
      - default + JAXR, IIOP, clustering
    - standard
      - The certified J2EE 5
      - Call-by-value and deployment isolation are enabled by default
    - web
      - J2EE 6 web profile
      - Standard Servlet/JSP container + JCA/JPA

Configuration set contents

- Variables
  - jboss.server.home.url: eg. “server/default/”
  - jboss.server.name: eg. “default”
- server/<confset>/conf
  - aka jboss.server.config.url
  - jboss-service.xml
    - core JMX services, like JAAS, Logging, MainDeployer, etc
  - bootstrap/*, bootstrap.xml
    - Bootstrap descriptors for core microcontainer services defined in bootstrap.xml
  - jndi.properties:
    - Specifies a set of properties that are passed to JNDI when new InitialDirContext() is called within JBoss
  - jboss-log4j.xml:
    - Configuration file for the logging service (Log4J) defining log filters, priorities, and destinations

Configsets

- server/<confset>/conf (cont.)
  - login-config.xml:
    - Defines security realms used for authentication and authorization (JAAS)
  - props/*.properties
    - Java property files (usually used for JAAS realms). A change might need a restart
  - java.policy
    - Placeholder for security permissions (Java Security Manager). Grant-All by default
  - standardjboss.xml
    - Configuration file for the standard EJB container
  - standardjbosscmp-jdbc.xml
    - Configuration file for the standard JBossCMP engine
  - xmdesc/*-mbean.xml
    - XMBean descriptors for services configured in the jboss-service.xml file

Configsets

- server/<configset>/data
  - backstore for HyperSonic DB. XA transactions, etc. Might be empty
  - http session store
- server/<configset>/deploy
  - default deployment directory
  - subject for hot-deployment (HDScanner)
  - compressed archives are exploded to server/.../tmp
  - mixup for jboss specific and local applications
    - and additional deploy directory might be useful
  - Most important components
    - Console applications:
      - ROOT.war, jmx-console.war, admin-console.war
    - Tomcat
      - jbossweb.sar

Configsets

JBIA:51

- server/<configset>/deployers
  - Set of deployers services specialized to one archive type
  - Some examples:
    - jbossweb.deployer
      - for WAR files
    - ejb-deployer-jboss-beans.xml
      - Service responsible for deploying EJB JAR files
    - ear-deployer-jboss-beans.xml
      - Service responsible for deploying EAR files

- server/<configset>/lib
    - jboss.server.lib.url
    - jar files shares among different archives (such as JDBC driver)
    - use common/lib, if jar is shared with jboss services as well

Configsets

- server/<config>/log
    - jboss.server.log.dir
  - Different log files:
    - boot log
      - console log until logging service starts
    - server.log
      - configured by ${jboss.server.config.url}/jboss-log4j.xml
  - default priority is DEBUG (to be changed for production)
  - LOG4J allows redirection to SNMP,Syslog,JMS, whatever
    - See: http://logging.apache.org/log4j
- server/<config>/tmp
  - Unpack destination for non-exploded archives
- server/<config>/work
  - JSP compile area, useful for debuging

Alternate config sets

- When JBoss goes on diet
  - Remove unneeded services
    - Frees up memory/threads for other services
    - Faster startup
    - Minimization for security
    - But may break J2EE compliance
  - Possible delegates for removal
    - JMX-console/Web-console/ROOT.war
    - Bean Shell Deployer/ UUID key generator/JMX invoker adapter
    - JMS
    - Cluster stuff: Cache Invalidaton Service, HA/Remote JNDI
  - Reference

http://goo.gl/EfbpT

Starting JBoss

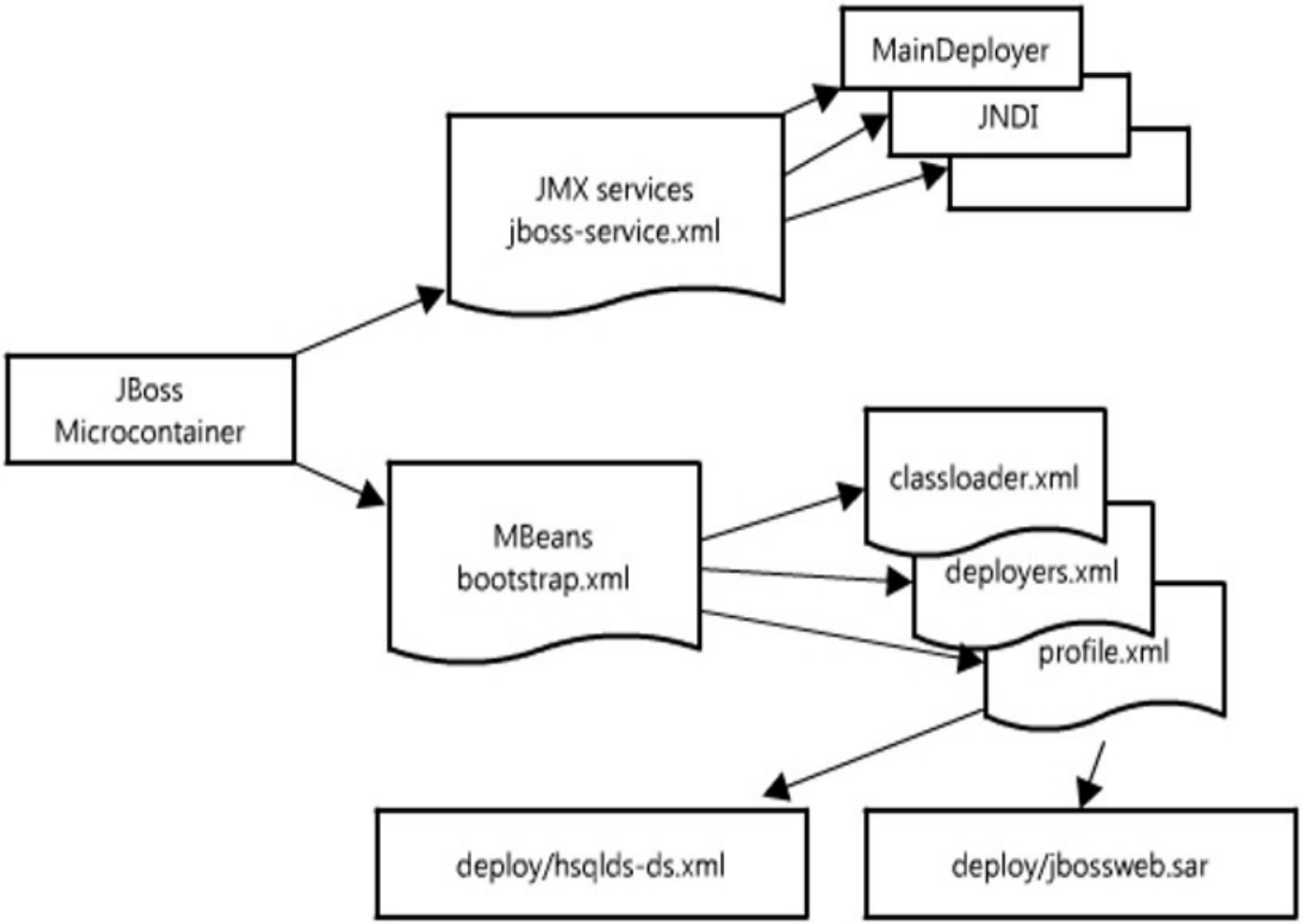
- bin/run.sh params
  - c: configset to be used (default)
  - b: bind address (0.0.0.0)
  - g: partition name for clustering (DefaultDomain)
  - m: multicast port (for JGroups, advised to be different per cluster)
  - p: patchdir
    - (puts a lib on first place in cp, redeploy might be better)
  - D: override some properties
    - -D jboss.bind.address=<IP> is eqivalent of -b <IP>
- stopping JBoss
  - kill -TERM is also fine
  - shutdown.sh (or bat)
    - JMX client, also for remote shutdown

      shutdown.sh -s jnp://<remote IP>:1099 -S

Configure as a service

- Install Stop/start script
  - {JBOSS_HOME}/bin/jboss_init_redhat.sh
    - or service.bat for Windows
    - #chkconfig and #description has to be added to be used with chkconfig
  - run.conf
    - run.sh parameters
    - JAVA params
- Configure parameters (in run.conf)
    - JBOSS_CONF
      - the “layout” to be started: default
    - JBOSS_BIND_ADDR
      - default is 0.0.0.0
    - JBOSS_USER
      - consider not using root

JBoss bootstrap



Lab 1: Install JBOSS

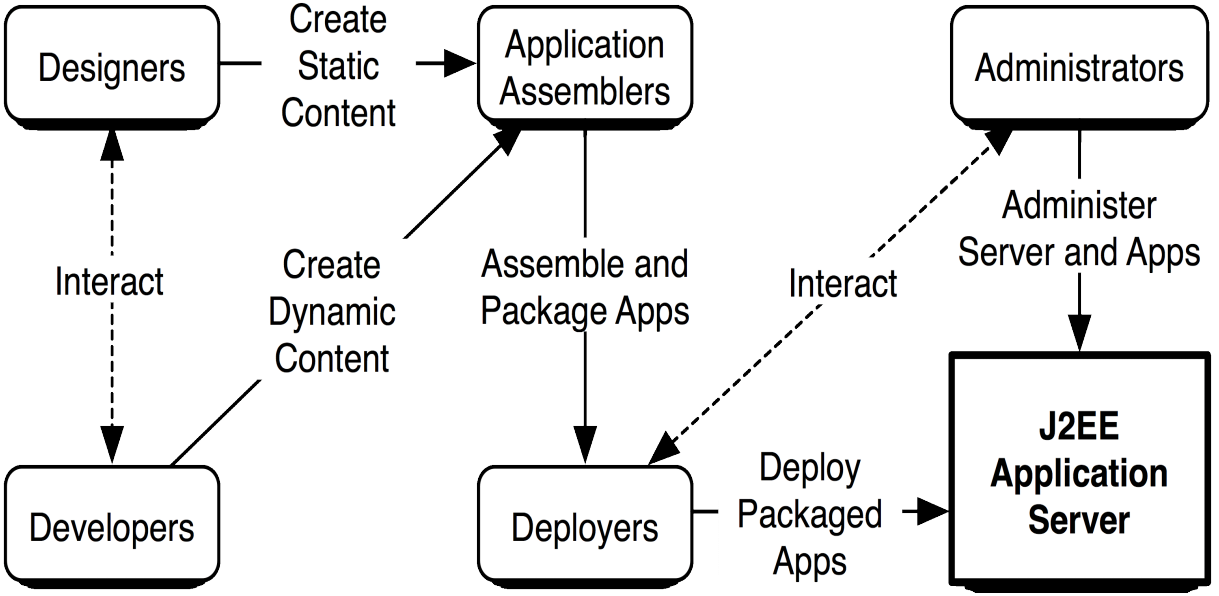
- Install HotSpot Jdk
  - find it in /download
  - use command alternatives to override openjdk binarys
    - alternatives –install
- Install JBoss5
  - unzip from /downloads
  -

2. Deployments on JBoss

- Objectives
-
  - J2EE application life-cycle
  - Archive types
  - Deployment methods
  - Deployment related beans
  - Directory structure

Deployment on JBoss

- J2EE application deployment
  - Provide resources
    - DB,JMS,Mail,WS
  - Customize the deployment descriptors (eg.web.xml, ejb-jar.xml)
    - Security
    - Transactions
    - Persistence



Application types

- JBoss provides a hiearchical architecture for deployment/classloding
  - Every single app type has a separate deployer
    - ${jboss.server.home.url}/deployers/
  - Standart J2EE types
    - WAR,EAR,EJB-JAR,RAR,WSR
  - Custom types
    - SAR: Service Archive
      - Provides a JBoss Microcontainer service
      - jbossweb.sar, jbossws.sar
    - HAR: Hibernate Archive
    - XML: standalone xml file describe a MBean service
      - *-ds.xml: converted to datasource
      - *-aop.xml: aspect archives
      - *-jboss-service.xml, *-jboss-bean:xml: standalone MBean service
    - .zip, .last: depends on the internal deployment descriptor

Deployment config

http://goo.gl/GkwMl

- Deployers
  - Deployers config
    - ${jboss.server.home.url}/conf/bootstrap/deployers.xml (see JBIA/83)
  - Alternate deploy dir:
    - 6.x, 5.1: /conf/bootstrap/profile.xml:
      BootstrapProfileFactory → applicationURIs
    - 5.0: /conf/bootstrap/profile-service.xml
      VFSDeploymentScanner → URIList
  - Deployment order
    - AOPs, SARs, POJOs, RARs, Data Sources, HARs, EJBs, ZIPs, WARs, Web Services, EARs, Bean Shell Scripts
    - .*last guarantied to load last time
    - Dependecies are taken into account

```
uuid-key-generator.sar/META-INF/jboss-service.xml:
<mbean..>
<depends optional-attribute-name="DataSource">
jboss.jca:service=DataSourceBinding,name=DefaultDS </depends>
</mbean>
```

${jboss.server.home.url}/deploy/

- Copy there deploys the app
  - If noexploded, an copy is made under ${jboss.server.temp.dir}
  - On exploded app, use mv instead of cp -r (multiple deployment, deployment loop)
  - Dependecies are handled, classloading is made
  - Nested deployments are supported (eg. WAR in EAR)
  - Farming deployment also works
- Undeploy
  - When deleting from deploy
  - Cleanup is made on “tmp” (usually)
  - Depencencies are might also undeployed
- Redeploy
  - Performed every time when you make a change on the deployed stuff
    - Changing deployment descriptors, like web.xml
    - Redeploy dependencies

In production

- Cold deployment is better
  - Avoids class loading issues
    - ClasscastException on the same class loaded with different classloaders
  - Avoids accidental reloads, redeploy loop
    - Unwanted change in a xml, dependency loops
- Unexploded is better
  - More prone to errors
  - Errors are more reproducible
    - a temporal change might be overwritten on the next deploy
- Deploy/undeploy through JMX

```
twiddle.sh -s localhost invoke "jboss.system:service=MainDeployer" deploy
"file:///something.war"

twiddle.sh -s localhost invoke "jboss.system:service=MainDeployer" undeploy

"file:///something.war"
```

Lab 2

- Deploy in JBOSS
  - Create an alternate deploy directory
  - JBIA helloapp.war
  - Customizations
    - overwrite context-root
    - add virtual host
    - add dependency

3. Deploying web applications

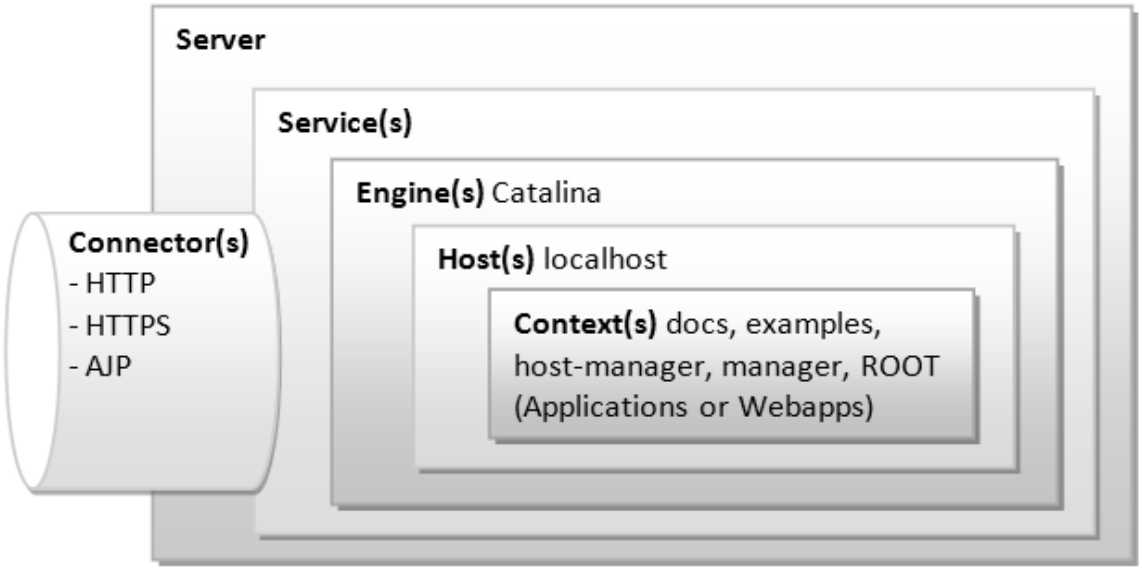
- Objectives
-
  - JBossWeb/Tomcat
  - server.xml
  - top-level/default web.xml
  - ROOT.war
  - Virtual hosts
  - JBossWeb + Apache

Simple webapps with Tomcat

- Tomcat 6.x
  - Open source web container (Servlet 2.5, JSP 2.1 (Jasper2))
  - Deployed as jbossweb.sar
    - was tomcat-xxx.sar in older releases
- Two main config
  - deploy/jbossweb.sar/server.xml
    - Tomcat's own config file
  - deployers/jbossweb.deployer/web.xml

    (6.x deploy/jbossweb.sar/web.xml)
    - DefaultServlet, Welcome/Error pages
    - Mime types
    - Filters
    - JSP compiler options
      - Eclispe JDT, no need for JDK (JRE is enough)

Structure of server.xml

- Containers
  - Server -> top level
  - Service -> the group of connectors
  - Engine -> handle all requests for a service
  - Host -> handle all requests for a virtual host
  - Context -> all requests for a given webapp
- Nested components
  - Some component can be nested into all other containers, such as Realm and Valve
  - 

server.xml elements

- Connectors
  - HTTP 1.1/HTTPS
    - redirectPort refers SSL port
    - URIEncoding=”UTF-8” is important
  - AJP
    - For sitting behind Apache/Lighttpd.
    - Socket handling effectivity/load balancing

```
 <Connector protocol="HTTP/1.1" port="8080" address="${jboss.bind.address}"
               connectionTimeout="20000" redirectPort="8443" />
 <Connector protocol="AJP/1.3" port="8009" address="${jboss.bind.address}"
         redirectPort="8443" />
<Connector protocol="HTTP/1.1" SSLEnabled="true" port="8443"
  address="${jboss.bind.address}" scheme="https" secure="true" clientAuth="false"
           keystoreFile="${jboss.server.home.dir}/conf/chap8.keystore"
           keystorePass="rmi+ssl" sslProtocol = "TLS" />
```

server.xml elements

- Engine
  - defines the default Host element, if more than one exists
- Host
  - Defines Host/IP specific params
  - Alias (adds a virtual hostname)
    - use WEB-INF/jboss-web.xml restrict on app to a virtualhost
  - Context
    - webapp context, specific params for one web application
    - place for JDBC references, access control, etc.
    - the same as WEB-INF/context.xml (optional)
- Realm
  - JAAS /user-role realm reference inherited from JBoss (discussed later in detail)

jboss.web:type=Host,host=.../addAlias

```
<jboss-web>
  <context-root>/myapp</context-root>
  <virtual-host>myhost.com</virtual-host>
</jboss-web>
```

server.xml elements

- Valve

jboss.web:type=Host,host=.../valveNames

  - Interceptors for incoming connections
    - AccessLogValve,FastCommonAccessLogValve: log for requests
    - RemoteAddrValve,RemoteHostValve: request blocking
    - RequestDumperValve
    - authenticator.SingleSignOn:
      cookie based SSO
    - service.sso.ClusteredSingleSignOn
      clustered SSO, needs shared/clustered cache service
    - cacheConnectionValve
      dropping DB connection on session disconnect (devel only, to detect connection leaks)
  - Docs at http://goo.gl/On3E7

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

       prefix="localhost_access." suffix=".log"

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

Tomcat's web.xml

- Default web descriptor for all web apps deployers/jbossweb.deployer/web.xml
- Configures
  - Common Filters
  - Servlets for handling static content (DefaultServlet), JSPs, SSI, CGI scripts, invokers, etc.
  - Default session timeout
  - MIME Type mappings
  - Welcome file list: index.html, index.jsp, etc.

```
<servlet>
  <servlet-name>default</servlet-name>
  <servlet-class>org.apache.catalina.servlets.DefaultServlet
</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

Servlet mappings

- WEB-INF/web.xml elements
  - servlet/ servlet-mapping
    - <servlet-name> - reference for later mappings
    - <servlet-class> - the HTTPServlet instance
    - <init-param>: passed tp ServletConfig
    - <load-on-startup>: load in order

```
<servlet>
 <servlet-name>MyServlet</servlet-name>
 <servlet-class>example.servlet.MyServlet</servlet-class>
  <init-param>
   <param-name>file</param-name><param-value>/WEB-INF/ip.log</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
 <servlet-name>IPLoggerServlet</servlet-name>
 <url-pattern>/ip</url-pattern>
</servlet-mapping>
```

Servlet mappings

- filter
  - do arbitrary mangling on the request (stackable)
  - implements javax.servlet.http.Filter interface
  - supports optional init-param

```
<filter>
  <filter-name>Logger</filter-name>
  <filter-class>example.servlet.AccessLogFilter</filter-class>
</filter>

<!ELEMENT filter-mapping (filter-name, (url-pattern | servlet-name))>
<filter-mapping>
  <filter-name>Logger</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Servlet mappings

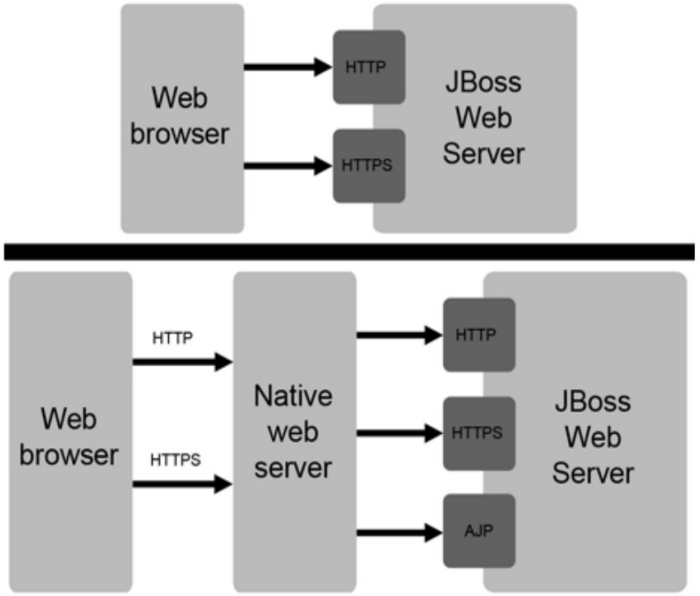
- session-config
  - session timeout
    - how long the servlet container will maintain an idle session
    - has significant impact on server memory usage
- welcome-file-list
  - welcome-file
    - used if request made for a directory
    - multiple entry allowed, the first found served
    - HTTP 404 if not defined or disabled
- error-page
  - error-code
    - HTTP error code
  - exception-type
    - exception caught on top level
  - location
    - the customized error page location
    - starts with /, relative to webapp

jboss.web:type=Manager,host=...,path=.../listSessionIds

```
<error-page>
  <exception-type>
    java.lang.Throwable
  </exception-type>
  <location>
    /error.html
  </location>
</error-page>
```

JBoss/Tomcat behind Apache

- Common in production
  - Using AJP connection
    - mod_jk: older, sometimes buggy
    - mod_proxy_ajp: new,fast, allows load balancing
    - mod_cluster: new from JBoss, advanced balancing, more feedback from node
- Reasons
  - Socket/connection handling
    - Apache (written in C) is more efficient on that
      - (using MPM is even faster)
    - Static content could be served easier
    - SSL setup is more straightforward
    - Using additional apache features
      - eq. URL Rewrite



Example with mod_proxy_ajp

```
LoadModule proxy_module libexec/apache2/mod_proxy.so
LoadModule proxy_ajp_module libexec/apache2/mod_proxy_ajp.so
LoadModule proxy_balancer_module libexec/apache2/mod_proxy_balancer.so

ProxyRequests Off
<Proxy balancer://jboss-cluster>
    BalancerMember ajp://jboss-1:8009  #multiple of that can be defined
    Order deny,allow
    Allow from all
    ProxySet lbmethod=bytraffic  # or byrequests
    ProxySet stickysession=JSESSIONID
</Proxy>
ProxyPass       / balancer://jboss-cluster/
ProxyPassReverse    / balancer://jboss-cluster/
```

Lab 3

- Configuring Tomcat
  - Create and serve static content
    - either via ROOT.war or apache
  - Add a mapping to helloapp.war
  - Customize tomcat's welcome and error pages
  - Catch exceptions generated by buggy.war
  - Create an access log for context of helloapp.war only
  - Restrict access to helloapp.war from IP

4. Core components

- Objectives
  - Class loading
    - J2EE specifications
    - Class loader implementations in JBOSS
    - WAR/EAR class loading
  - JNDI
    - JNDI service
    - JNDIView in JMX

Class Loading in JBoss

- J2EE mandates a namespace isolation
  - Classes loaded/initialized by different applications must not conflict
  - Class loaders provide a runtime identity for a class
    - fully qualified name + the classloader
    - JVM defines class loading domains
    - the same class loader by 2 different loader cannot be casted (ClassCastException)
    - Fall back to serialization -> slow
    - But hot deployment is possible
  - Sharing classes
    - Fast as call by reference
    - Change the class force a restart of all the referring applications
  - Solution
    - isolate shared classes, and the rest

VFS classloader

- Introduced in 5.x
  - Simplifies the handling of classes inside jar files
  - Class loading semantics are the same as for JBoss 3 and 4.
  - JBoss has rules for
    - class loader / class
    - visibility of the class (sharable or not)
- Default class load order
  - $CLASSPATH + ${jboss.home.dir}/bin
  - Classes deployed by other apps, a classes found in the JAR files under ${jboss.server.home.url}/lib
  - Classes found in a WAR file
    - classes can be cached

Class loading for WARs

- Servlet spec requests the total isolation of classes
  - System classes
  - WEB-INF/classes , WEB-INF/lib comes first
  - Root repository cache
  - <confset>/lib
- Jboss specific class scopes
  - <loader-repository> element
  - Scoped deployments maintain own loader repository

Scoped Class Search Order

- java2ParentDelegation=false (default)
  - $JBOSS_HOME/server/<config>/deploy/<app>/
  - System Bootstrap
  - Default Repository Cache
  - $JBOSS_HOME/server/<config>/lib/
- java2ParentDelegation=true
  - Default Repository Cache
  - System Bootstrap ($JAVA_HOME/lib/ext)
  - $JBOSS_HOME/server/<config>/deploy/<app>/
  - $JBOSS_HOME/server/<config>/lib/

JBoss class loading references

- https://community.jboss.org/wiki/classloadingconfiguration
- https://community.jboss.org/wiki/JBossClassLoadingUseCases
- https://community.jboss.org/wiki/JBossClassLoaderHistory
- http://www.javacodegeeks.com/2012/09/jboss-as-7-classloading-explained.html
- http://community.jboss.org/wiki/ClassCastExceptions
- http://community.jboss.org/wiki/GetClassNotFoundExceptionOrNoClassDefFoundError
- JBIA/86

Java Naming and Directory Interface (JNDI)

Java Naming and Directory Interface (JNDI)

- A DNS like core infrastructure of Java, to locate object/services
  - Defined in J2SE, but has fundamental role in J2EE
  - Important in clustering
    - hides the actual location of the object/service
- JNDI in J2EE
  - Maps JDBC connections, EJB, properties files
  - 3 main namespaces
    - Global for remote calls (EJB remoting, JMS)
      - queue/DLQ
    - JVM local (starts with java:)
      - java:Mail
    - App private (ENC)
      - java:comp/env

JNDI ENC conventions

- java:comp/env/var
  - Environmental variables
- java:comp/env/mail
  - JavaMail sessions
- java:comp/env/jms
  - JMS connection factories and destinations
- java:comp/env/ejb
  - EJB home interfaces
- java:comp/env/jdbc
  - JDBC Datasources

JNDI in JBoss

- Local and remote access (port 1099)
  - call by value or reference (configurable, depends on configset)
- JNDI operations can be invoked over JMX
  - like list, create an alias
- HA-JNDI
  - support for clustering (discussed in the clustering section)
- Defined in jboss-service.xml

```
<mbean code="org.jboss.naming.NamingService"
       name="jboss:service=Naming"
       xmbean-dd="resource:xmdesc/NamingService-xmbean.xml">
  ...
<mbean code="org.jboss.naming.JNDIView"
       name="jboss:service=JNDIView"
       xmbean-dd="resource:xmdesc/JNDIView-xmbean.xml">
</mbean>
```

Remote JNDI

```
# /jboss5/bin/twiddle.sh -s jnp://10.201.0.50:1099  invoke "jboss:service=JNDIView" listXML
<jndi>
<other-encs><context>
<name>java:comp</name>
<attribute name='bean'>jboss.j2ee:jar=profileservice-
secured.jar,name=SecureDeploymentManager,service=EJB3</attribute>
<leaf>
<name>EJBContext</name>
<attribute name='class'>javax.ejb.EJBContext</attribute>
</leaf>
<link-ref>
<name>TransactionSynchronizationRegistry</name>
<link>java:TransactionSynchronizationRegistry</link>
<attribute name='class'>javax.naming.LinkRef</attribute>
</link-ref>
<leaf>
...
```

5. Configuring Datasources

- Objectives
  - Datasource by definition
  - JDBC/JCA
  - Datasource hot deployment
  - JMX bindings

Configuring Datasources

- Application needs access to relational databases
- DB connection managed by app
  - Deployments are harder
  - Reinvent the wheel
    - Connection pooling has to be manually implemented
  - On connection pool per app
    - Maybe out a connections soon
- Connection managed by container
  - Shared pools
    - better utilization
  - Single point of configuration/tuning

Configuration steps

- Install JDBC driver
  - .jar may be sitting in app's WEB-INF/lib but shared is better
    - Copy under ${jboss.server.lib.url} or ${jboss.common.lib.url}
- Defined a datasource
  - separate xml file (usually xxx-ds.xml, use templates)
    - JNDI name (under java:/)
    - jdbc connection params
      - url, user,password
    - JCA params
      - min-pool-size/max-pool-size (maybe the same)
      - idle-timeout-minutes/blocking timeout
      - new-connection-sql/check-valid-connection-sql
- Map resources
  - glue the application requested datasource name with yours

Example 1: my-ds.xml

```
<datasources>
  <local-tx-datasource>
    <jndi-name>MyDB</jndi-name>
        <connection-url>jdbc:mysql://localhost:3306/mydb?autoReconnect=true
         </connection-url>
        <driver-class>com.mysql.jdbc.Driver</driver-class>
        <user-name>myuser</user-name>
        <password>secret</password>
        <exception-sorter-class-
name>org.jboss.resource.adapter.jdbc.vendor.MySQLExceptionSorter</exception-sorter-
class-name>
        <new-connection-sql>SELECT 1</new-connection-sql>
        <check-valid-connection-sql>SELECT 1</check-valid-connection-sql>
  </local-tx-datasource>
</datasources>
```

Example 2: Resource mapping

- map java:/comp/env/OtherDB to java:/MyDB

```
#web.xml
<web-app ...>
  ...
  <resource-ref>
    <description>DB Connection</description>
        <res-ref-name>jdbc/OtherDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
  </resource-ref>
  ...
</web-app>
```

```
#jboss-web.xml
<jboss-web>
  <resource-ref>
    <res-ref-name>jdbc/OtherDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <jndi-name>java:/MyDB</jndi-name>
  </resource-ref>
</jboss-web>
```

Connection leaks

- If a servlet not correctly closes connections
  - close() return connection to pool
  - DB Object(Statement,Resultset) kept in memory as long as connection exists
- jbossjca-service.xml
  - defines CacheConnectionManager
    - triggered by Tomcat's CacheConnectionValve
    - Tomcat depends on this service
  - Enabled by default
    - advised to be used during development
    - can be turned off in production (dependency can also be removed)

Lab 4: Creating datasources

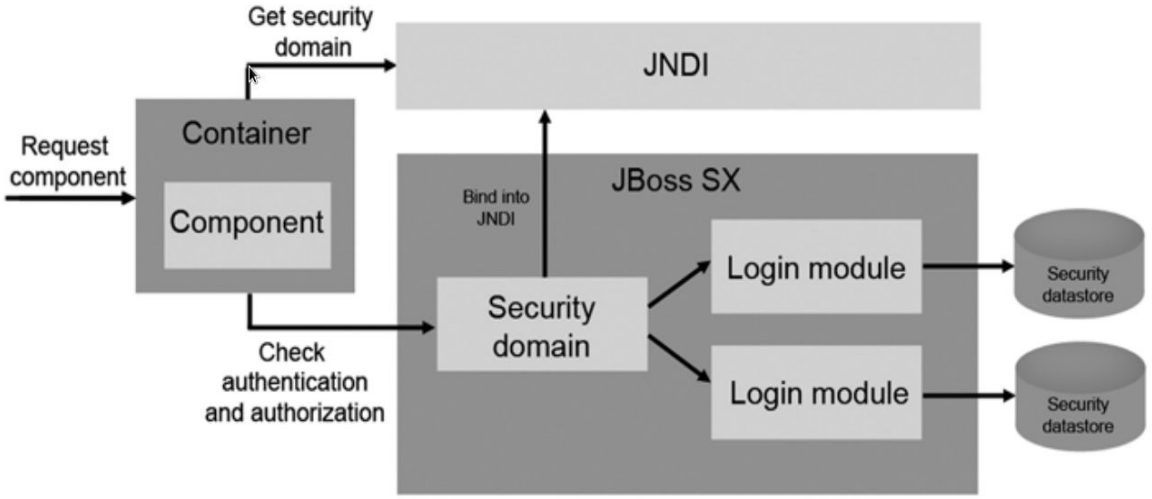
- Create a db/user in mysql
- Check mysql driver
- Define mysql datasource
- Make mapping for hellodb.war
- Detect leaks for hellodb.war

6. JBoss Security

- Objectives
-
  - JAAS and JBossSX
  - Login module configuration
    - UserRolesLoginModule
    - DatabaseLoginModules
    - LDAP based modules
  - Enforcing authentication in webapps

JAAS and JBossSX

- JAAS (Java Authorization and Authentication Service)
  - Authentication
    - Security principals (like username) and credentials (like password)
    - JBoss SX has several login-modules providing that service.
    - Principals has a role assigned
  - Authorization
    - J2EE defines a role based authorization
      - Servlets/JSPs can be accessed by one role members
      - EJB methods access can be restricted to roles



Declaring security constrains

- standard files

```
<web-app ...>
  ...
  <security-constraint>
    <web-resource-collection>
          <web-resource-name>All Resources
             </web-resource-name>
          <description>...</description>
          <url-pattern>/*</url-pattern>
          <http-method>GET</http-method>
        </web-resource-collection>
        <auth-constraint>
          <role-name>MyRole</role-name>
        </auth-constraint>
  </security-constraint>
  ...
```

```
  ...
  <security-constraint>
  ...
  </security-constraint>
  <login-config>
    <auth-method>BASIC</auth-method>
        <realm-name>MyRealm</realm-name>
  </login-config>
   <security-role>
    <description>MyRole</description>
        <role-name>MyRole</role-name>
  </security-role>
</web-app>
```

Declaring security constrains

- AS specific files

```
#jboss-web.xml
<jboss-web>
   <security-domain>
          java:jaas/MyAuth
   </security-domain>
   <context-root>...</context-root>
</jboss-web>
```

```
#login-config.xml
...
<application-policy name="MyAuth">   <!-- registered as java:/jaas/MyAuth -->
<authentication>
    <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
                   flag="required">
    <module-option name="usersProperties">my-users.properties</module-option>
    <module-option name="rolesProperties">my-roles.properties</module-option>
    </login-module>
</authentication>
</application-policy>
```

Login modules in JBoss

- can be stacked
  - defined in conf/login-config.xml
  - Unix pam like stacks, same keyword
    - sufficient, required, requisite,optional
  - password-stacking
    - useFirstPass: only one prompt for password
- large set of modules out of box
  - See: http://docs.jboss.org/jbosssecurity/docs/6.0/security_guide/
  - UsersRolesLoginModule
    - user.properties, roles.properties
      - first found in you classpath
    - hashAlgorithm, hashEncoding
      - to avoid cleantext
    - login-config.xml
      - has a “last change” login module as a default one

Login modules in JBoss

- DataBaseLoginModule
  - dsJndiName
    - the datasource query from
  - principalsQuery: prepared statement like:

    SELECT Password from Users WHERE Username=?
  - rolesQuery:

    SELECT Rolename, "Roles" FROM Roles WHERE Username=?
- LdapLoginModule
  - Use auth
    - simple BIND to the LDAP service with dn + password
    - principalDnPrefix: such a “uid=”
    - principalDnSuffix: like “,ou=People,dc=example,dc=com”
  - Role matching:
    - rolesCtxDn: baseDN to search for roles
    - uidAttributeID: the attribute of the roles object that lists users
    - matchOnUserDN: whether search for 'user1' or 'uid=user1,....'

Login modules in JBoss

- LdapExtLoginModule
  - still tests with BIND, but users/roles are found recursively in the LDAP tree
    - password hashing is the choice of LDAP server, not of JAAS
  - bindDN/bindCredential:
    - user/password to BIND just for searching
  - baseFilter/roleFilter:
    - LDAP filter expression to find user/roles dn's
  - searchTimeLimit/searchScope
    - LDAP search attributes

Declaring security constrains

Optionally, you can create login config locally

```
#jboss-service.xml  inside some war/ear
<server>
<mbean
code="org.jboss.security.auth.login.DynamicLoginConfig"
name="jboss:service=DynamicLoginConfig-xxx">
   <attribute name="AuthConfig">
      dynamic-login-config.xml  <!just a file>
   </attribute>
 ...
</mbean>
</server>
```

```
#dynamic-login-config.xml
<policy>
<application-policy name="MyAuth">
<authentication>
    <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
                  flag="required">
    </login-module>
</authentication>
</application-policy>
</policy>
```

Lab 5: Security

- Secure helloapp.war
  - HTTP Basic with UsersRolesLoginModule
    - With plaintext passwords
    - With hashed passwords
  - HTTP Digest with DatabaseServerLoginModule
  - HTTP Basic with LDAPLoginModule
  - HTTP FORM auth with UsersRolesLoginModule

7. JMS in JBoss

- Objectives
  - Messaging model elements
    - Topic
    - Queue
    - DLQ/Expiry
  - JMS implementations
    - JBossMQ
    - JBoss Messaging
    - HornetQ

JMS

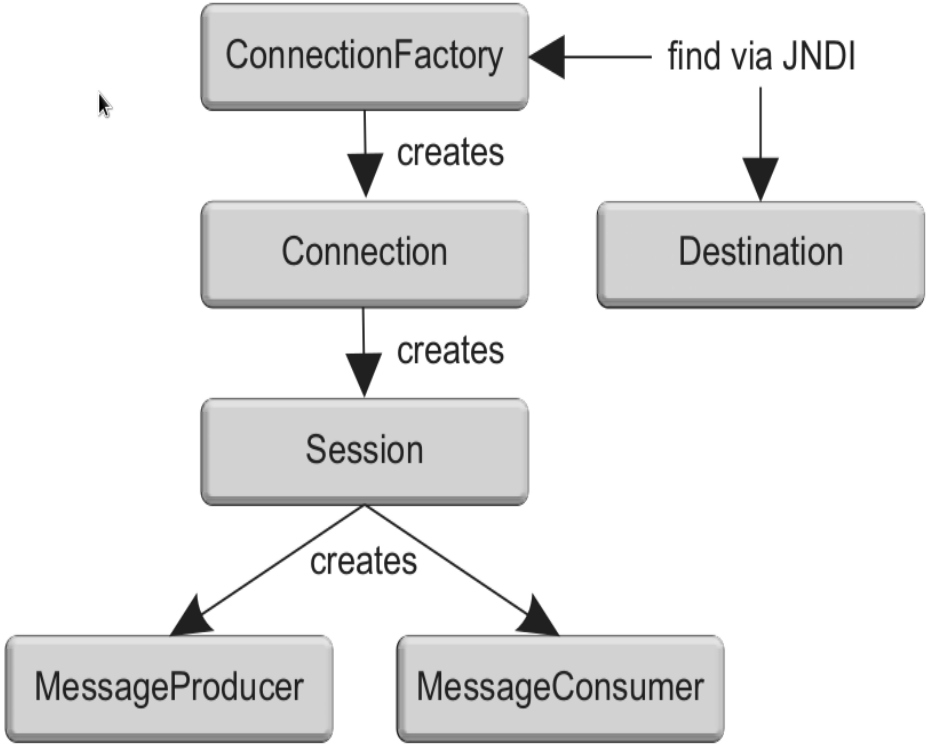
- About Java Messaging Service
  - Framework for communication between distributed components
    - In or cross JVM
    - sync or async
  - Loosely coupled
    - Unlike RMI (eg. EJB remoting)
    - Client has to now only the destination and the message format
  - Reliable
    - message delivered/processed only once
  - Transaction support
    - Message send/receive operations can participate in a distributed transaction

JMS in J2EE

- Application clients, EJBs, and web components can send and receive JMS messages
- SLSB can be “upgraded” to Message-driven beans
  - enable the asynchronous (possibly concurrent) consumption of messages,
  - easy to plug in new business event handlers into an existing deployment
- Dependences
  - Compile time dependency (IoC/DI)
  - Run-time dependency (lookup through JNDI)
- JMS 2.0 in 2013 Q2
  - prepare for J2EE7
  - removes the need for some boilerplate code
  - See: http://java.net/downloads/jms-spec/JMS2.0Devoxx2011.pdf

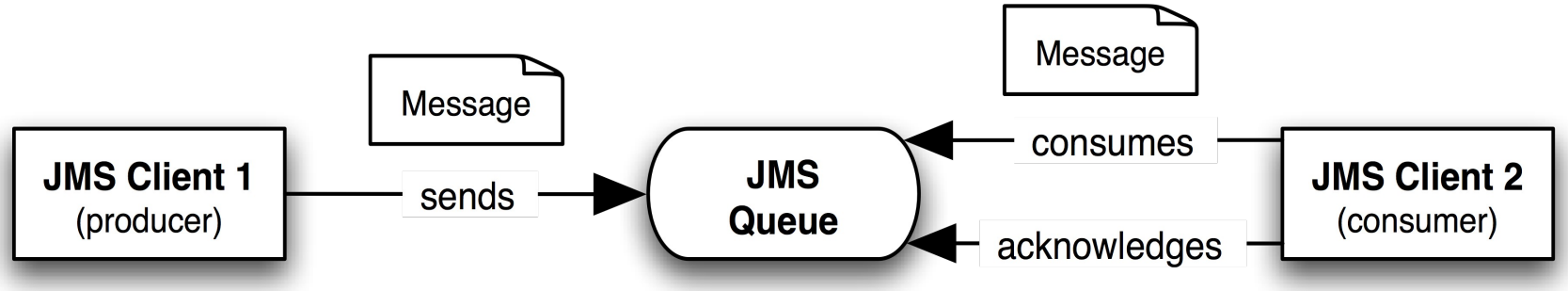
JMS components

- Connection Factory
  - to create JMS connection
- Destination
  - logical entry to proxy messages
- JMS client
  - Producer/consumer of messages
  - MDB for J2EE integration
  - receive(sync), onMessage(async)
- JMS provider
  - Implements JMS interfaces(such as CF)
  - such as JBoss MQ, JBoss Messaging
-



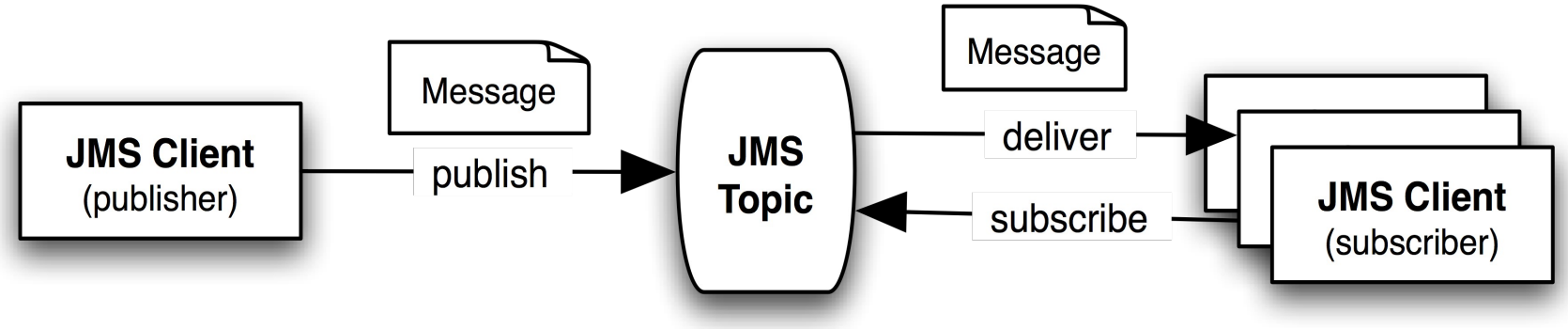
Messaging models

- Producer/Customer
  - 1:1
  - Each message has exactly on consumer
  - Queue store the messages until DeliveryTime exceeds
    - file or db store)
  - No timing dependency
    - consumer can receive a message that was sent before his login



Messaging models

- Publish/Subscribe
  - Typically 1:N, so a message can have multiple consumers
  - The topics hold the messages as long as it takes to deliver them to all currently subscribed clients
    - short term typically
  - Timing
    - delivery for messages the latest messages only
    - no delivery guarantee if subscription is not durable



JBoss messaging versions

- JBossMQ
  - pre 5.x default messaging, first JMS implementation
- JBoss Messaging 1.x
  - works with 4.x, default in 5.x
  - fully JMS 1.1 compatible
  - code written for MQ is still working, but deployment descriptors are not
    - however very similar
- HornetQ (aka JBoss Messaging 2.x)
  - improved performance
  - could be used as standalone message service (outside the scope of JBoss)

JMS config in JBoss 5.x

- Under **deploy/messaging**
  - in detail: http://docs.jboss.org/jbossmessaging (Chapter 6)
  - messaging-service.xml
    - Defines
      - jboss.messaging:sevice=ServerPeer
    - Top level definitions
      - JNDI context (topic, queue)
      - Timeouts, counters
  - connection-factories-service.xml
    - Defines
      - jboss.messaging.connectionfactory:service=ConnectionFactory
    - connection factories
      - clustered/non clustered (load balancing/failover)
      - simple or distributed
      - JNDI bindings

JMS config in JBoss

- destination-service.xml
  - Defines default example destinations
    - jboss.messaging.destination:service=Queue,name=DLQ
    - jboss.messaging.destination:service=Queue,name=ExpiryQueue
- hsqldb-persistence-service.xml
    - persistent storage for messages
    - examples at docs/examples/jms/ for different dbs
      - hsqldb is generally not safe for production
    - also configures the PostOfficeService
      - acts as message router
      - also has persistence setting for mapping addresses (Topic current subscribers)
- jms-ds.xml
    - defines JMSProvider
- messaging-jboss-beans.xml
    - main focus on JMS security

Messaging config in JBoss 6.x

- deploy/hornetq
  - hornetq-configuration.xml
    - configuration values (ports, IPs)
    - security settings
  - hornetq-jboss-beans.xml
    - JMSProvider/Factory beans
  - hornetq-jms.xml
    - connection factories
    - default queue/topic objects
  - jms-ds.xml
    - backend for persistence

Create custom queue

- Just use destination-service.xml/hornetq-jms.xml as template
- Additional attributes
  - JNDIName
    - if not derived from name (or outside the DefaultQueueJNDIContext)
  - DLQ
    - overriding the JBoss default
    - special queue for failed delivery messages
    - usually for monitoring (you can write you own MBD)
      - write a consumer that logs message, send warning to admin, etc.
  - ExpiryQueue
    - used it non-persistent delivery message's TTL expires
  - RedeliveryDelay, MaxDeliveryAttempts
  - FullSize
    - control memory consumption of the queue

Example queue 5.x

```
<server>
  <mbean code="org.jboss.jms.server.destination.TopicService"
    name="jboss.messaging.destination:service=Topic,name=exampleTopic"
    xmbean-dd="xmdesc/Topic-xmbean.xml">
    <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
    <depends>jboss.messaging:service=PostOffice</depends>
  </mbean>
</server>
```

```
<server>
  <mbean code="org.jboss.jms.server.destination.QueueService"
    name="jboss.messaging.destination:service=Queue,name=exampleQueue"
    xmbean-dd="xmdesc/Queue-xmbean.xml">
    <depends optional-attribute-name="ServerPeer">
jboss.messaging:service=ServerPeer</depends>
  </mbean>
</server>
```

Example queue 6.x

```
## hornetq-jms.xml ##
<queue name="exampleQueue">
      <entry name="/queue/exampleQueue"/>
</queue>

<topic name="exampleTopic">
       <entry name="/topic/exampleTopic"/>
</topic>
```

```
## hornetq-configuration.xml ##
 <address-settings>
      <!--default for catch all-->
      <address-setting match="#">
      <dead-letter-address>jms.queue.DLQ</dead-letter-address>
      <expiry-address>jms.queue.ExpiryQueue</expiry-address>
      <redelivery-delay>0</redelivery-delay>
      <max-size-bytes>10485760</max-size-bytes>
</address-settings>
```

Configuring persistence

- For guaranteed/persistent delivery, there is need for storage
  - Create a JCA datasource first
  - Create service=PersistenceManager
    - MaxParams: for prepared statements
    - UsingBatchUpdates: improves performance
    - SQLProperties: DB specific DDL
  - The same datasource to be used for
    - service=PostOffice
    - service=JMSUserManager
  - Use templates from docs/examples/jms

Lab 6: JMS

- Deploy the chat application

8. Managing Enterprise JAVA beans

- Bean types
  - Entity
  - Stateful/Stateless Session Bean
  - Message Driven Beans
- Bean lifecycles
- EJB deployments
- Pool config

Managing Enterprise JAVA beans

- Bean: Java->Coffee->Bean
    - just a piece of Java code
- Beans live in EJB Containers
  - Just focus on business logic, the container does the rest
    - Transactions
    - Load balancing, pooling
    - Persistence (now JPA)
    - Security
- EJB Advantages
  - Scalability
    - transparent distribution across multiple server instances
    - distributed transactions
  - Easy access to bean provided business logic
    - JMS, Web services, Servlet/JSP

Managing Enterprise JAVA beans

- EJB Disadvantages
  - Heavyweight technology
    - till EJB3 development was far to complex
      - had to deal with a complex object/interface set in order to implement a simple bean (Remote, RemoteHome, ...)
    - Unit testing was a challenge
    - Slower development-> deployment-> test cycle
  - EJB3 improvement
    - has the same complexity in background (except for Entity Beans)
    - but everything is controlled/described by annotations
      - much smaller number of xml descriptors a needed
    - many features of Hibernate are incorporated into the JPA standard, replacing Entity Beans.

Bean types

- Session bean
  - Reusable component that contains business logic
  - Provides public methods for local/remote clients
  - Stateless Session Bean
    - **Does not** maintain the state of the client
      - client data has to be passed for each method invocation
    - Easier maintain, can be pooled
      - In general, performs better
  - Stateful Session Bean
    - **Does** maintain the client state
      - e.g. shopping cart
    - Bigger maintenance overhead
      - Cannot be pooled, each client has an instance
      - Handle timeouts for Passivate/activate and expiration has overhead
    - Bad programming could lead to OOM errors
      - If miss to call remove() when finished

Bean types

- Entity bean
  - Represents a relation database table/row
    - till EJB3 standard is open to other backstores as well
    - Properties are accessed through getter/setter methods
  - Change in EJB3
    - In EJB2, object an abstract class that implements EntityBean interface
      - Choice of CMP, BMP
      - A lot has to be done in xml
        - declaring roles,permissions, field mappings, etc.
    - In EJB3, object is a POJO, annotated with javax.persistence.Entity
      - Annotation for roles, etc.
      - Lazy loading and other container tricks are done by the given JPA implementation (such as Hibernate)

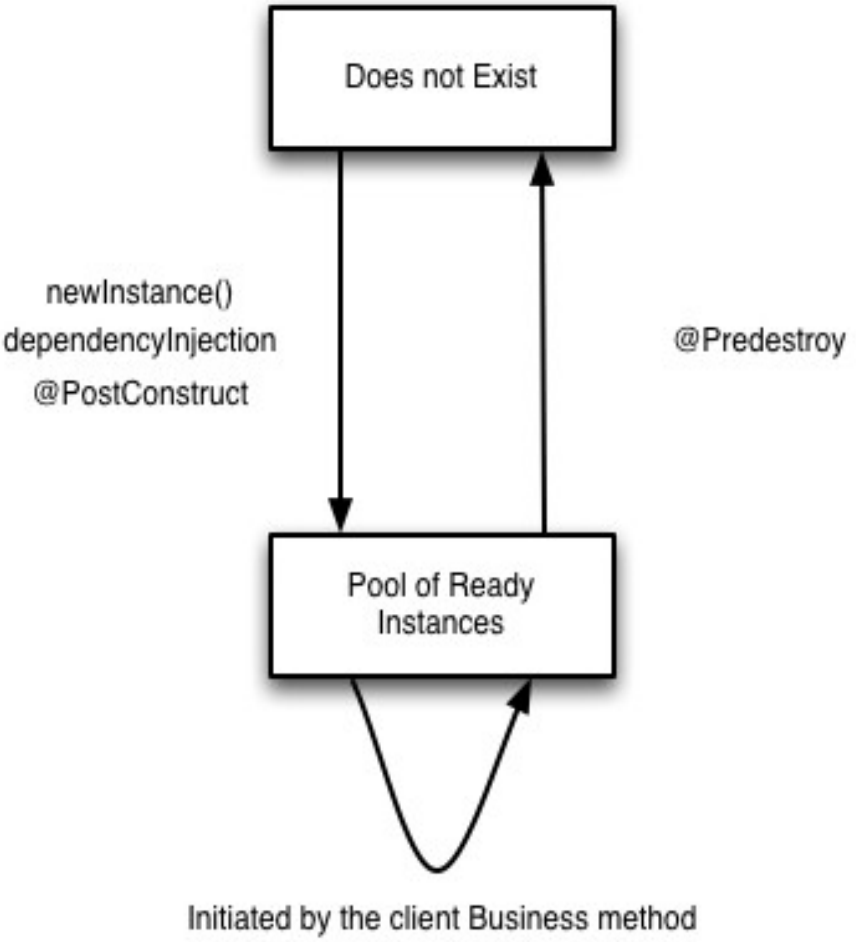
Message Driven Beans

- Similar to SLSB,
  - stateless
  - can be pooled
- but one major difference
  - SLSB has clearly defined interface, with public method could be called by local/remote/ws clients
  - MDB implemented as a JMS (Topic/Queue) listener.
    - not called directly (you can send a message to the JMS, to interact with an MDB
    - has an onMessage() method

Bean lifecycles

Stateless session Bean

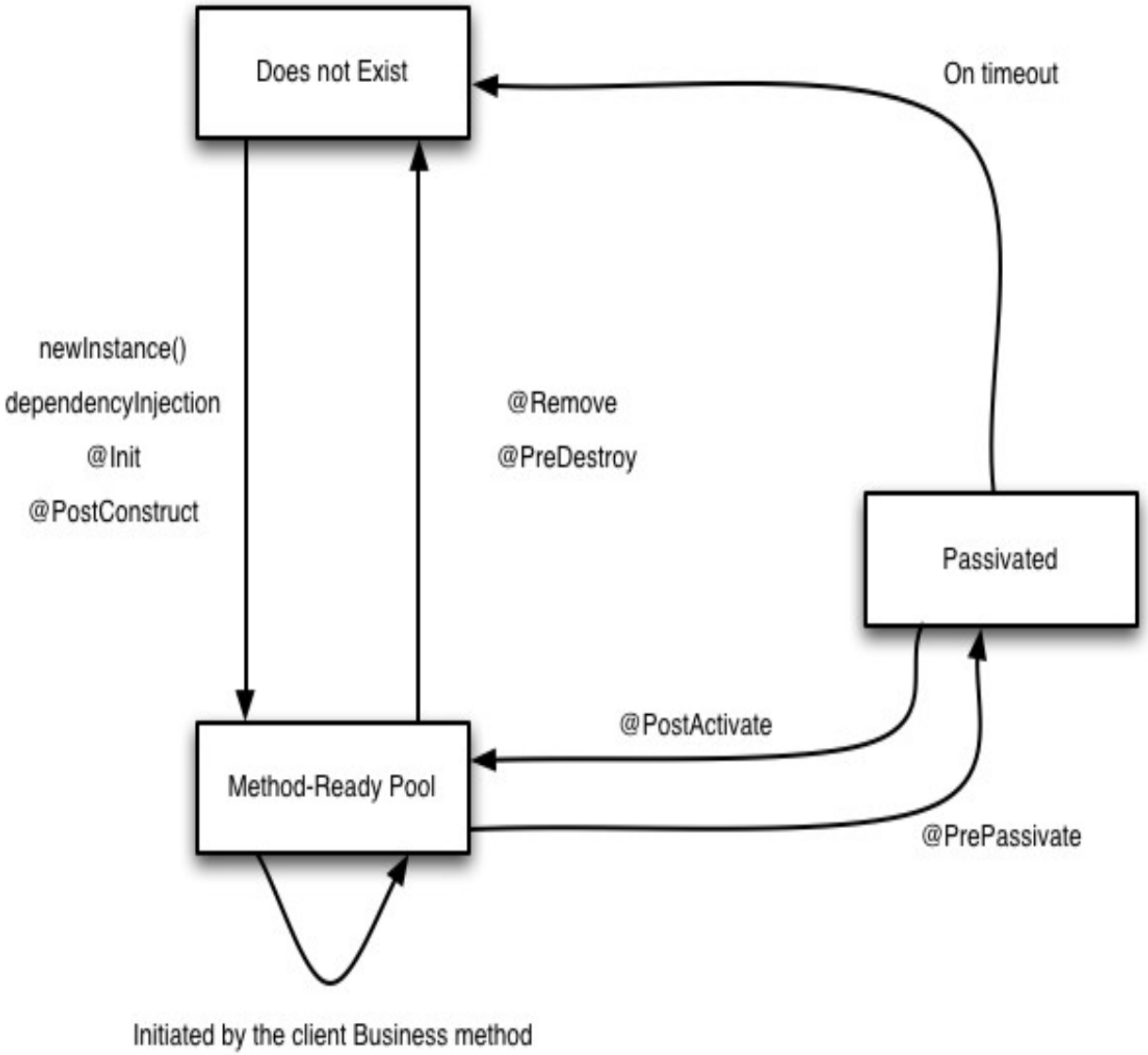
- Instances created by the EJB container
- These instances are kept into a pool of ready instances
- A method call is made by a client
- The container assign a ready instance from the pool to the client for the duration of the call
- The SLSB instance is then returned to the pool.



Bean lifecycles

Stateful session Bean

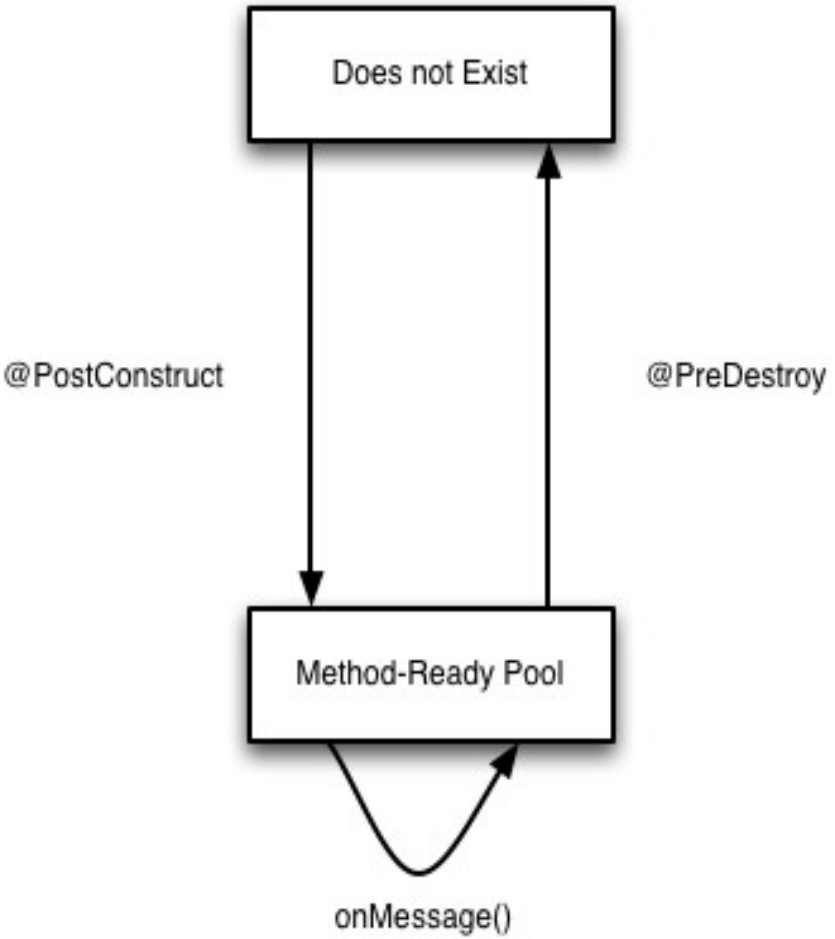
- A client session is started
- Default constructor is invoked
- Ressources are injected (if any)
- @PostContruct annotated method is called
- Now the SFSB is in cache and ready to execute any business method invocation from the client



Bean lifecycles

Message driven Bean

- A message is received
- The contained checks any Message Driven Bean instance available in the pool
- If available, JBoss use this MDB
- When onMessage() complete, the MDB instance is returned to the pool



Packaging EJBs

```
# jar -tf myapp.ear
./myapp.war
./myapp-ejb.jar
./META-INF/application.xml
./META-INF/jboss-app.xml
```

```
# jar -tf myapp-ejb.jar
./com/myapp/MyApp.class
./META-INF/ejb-jar.xml
./META-INF/jboss.xml
./META-INF/persistence.xml
```

```
#application.xml
<application>
    <display-name>MyApp</display-name>
    <module>
        <web>
            <web-uri>myapp.war</web-uri>
            <context-root>/MyApp</context-root>
        </web>
    </module>
    <module>
        <ejb>myapp-ejb.jar</ejb>
    </module>
</application>
```

Packaging EJBs

```
#ejb-jar.xml
<ejb-jar>
<description>MyApp</description>
<display-name>MyApp</display-name>
  <enterprise-beans>
     <session> ... </session>
       <message-driven> ... </message-driven>
   </enterprise-beans>
   <assembly-descriptor>
      <security-role> ... </security-role>
      <method-permission> ...
      </method-permission>
       <container-transaction> ...
     </container-transaction>
      <exclude-list> ... </exclude-list>
   </assembly-descriptor>
</ejb-jar>
```

```
#jboss.xml
<jboss ...>
<security-domain>some-domain</security-
domain>
<enterprise-beans>
<session>
    <ejb-name>ShoppingCart</ejb-name>
    <jndi-name>ShoppingCart</jndi-name>
    <clustered>false</clustered>
       <cluster-config>
       </cluster-config>
      <security-domain>
         overridden-domain
      </security-domain>
</session>
```

EJB2 pool configurations

- Essential for optimal performance
  - at conf/standardjboss.xml
    - SLSB/SFSB pool settings under <container-pool-conf> element
      - MinSize/MaxSize
      - strictMaxSize/strictTimeout
        - possible java.rmi. ServerException
    - SFSB passivation
      - <container-cache-conf>
        - LRUStatefulContextCachePolicy by default
        - can be NoPassivationCachePolicy
      - <persistence-manager>
        - File persistence by default
    - More in detail JBIA/211
  - gather pool stats
    - JMX/jar=<your-ejb-jar>.jar,name=<YourBean>,service=EJB<2,3>

EJB3 pool configurations

- Finer grained configuration with AOP
  - at deploy/ejb3-interceptors-aop.xml
    - separate domain for each type
    - domain contains a stack of actions (interceptors).
    - calls to the target method goes through the statck of actions

```
#ejb3-interceptors-aop.xml

<domain name="Stateless Bean" extends="Intercepted Bean" inheritBindings="true">

      <bind pointcut="execution(public * @org.jboss.ejb3.annotation.SecurityDomain->*(..))">

         <interceptor-ref name="Basic Authorization"/>

      </bind>

      <annotation expr="class(*) AND !class(@org.jboss.ejb3.annotation.Pool)">

         @org.jboss.ejb3.annotation.Pool (value="ThreadlocalPool", maxSize=30, timeout=10000)

      </annotation>

</domain>
```

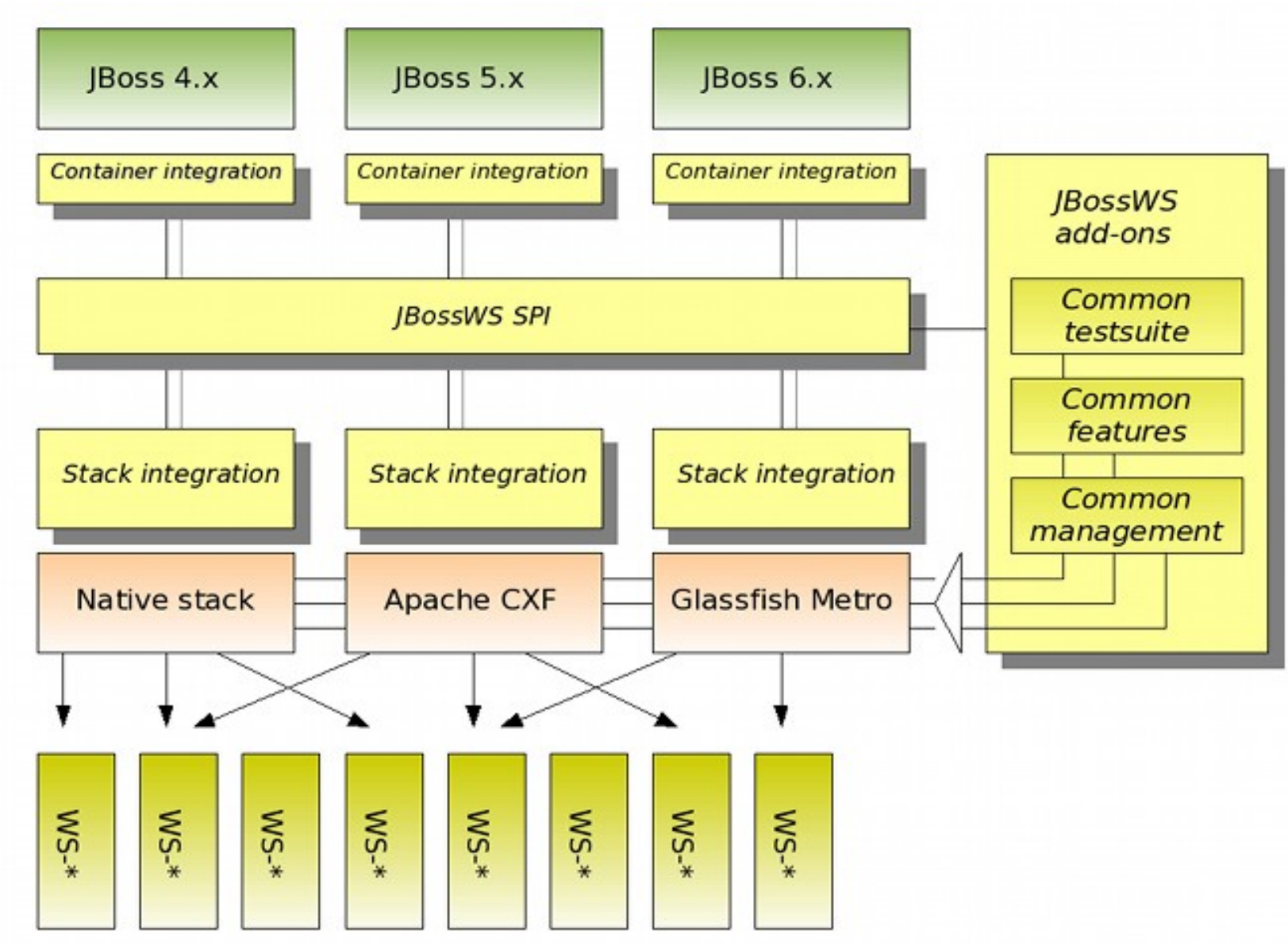
Lab 7: Enterprise Beans

- Deploy greeterapp.jar
  - Provide the neccessary datasource
  - Test with the Client class bundled.
  - See the resulting rows in the database
- Package/ deploy and test the GreeterApp.ear
  - Solve the problems in the JNDI lookup between greeterapp.war and greeterapp.jar
  - Restrict access to beans greet method, with a UsersRolesModule based access control
- Secure greeterapp.ear
- SFSB session timeouts.

9. Web services

- Objectives
  - Web services definitions
  - Web service elements
  - Web service development tools
    - Annotations
    - wsxxx tools

WS architecture in JBoss



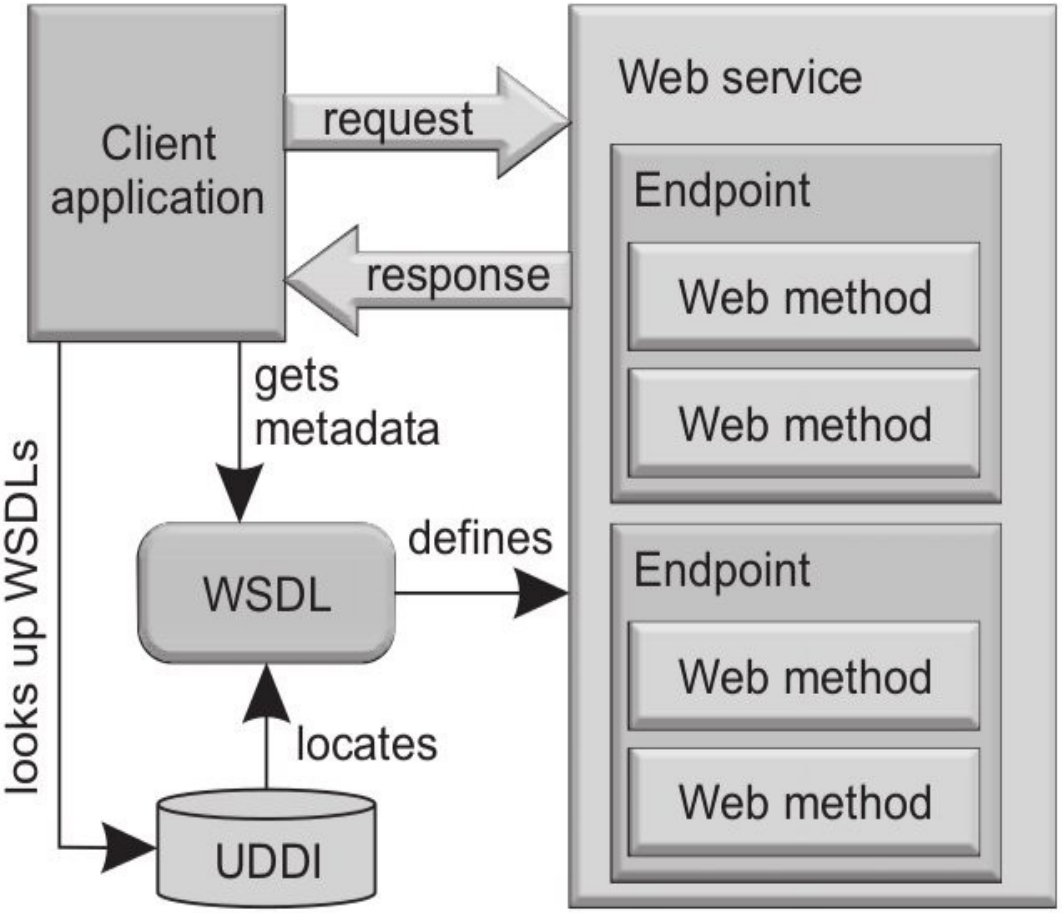
Web services

- “Web services are self-contained, self-describing, and modular applications that can be published, located, and invoked across the Internet”
  - Platform independent
    - client can be implemented in “any” language
  - Based on existing standards
    - like XML and HTTP
      - port 80 is generally more accessible then e.g. the RMI port
  - Less efficient than other RPC (RMI/IIOP, Corba)
    - but much simpler
  - Transactional behaviour is not granted by default
    - like in EJBs

Web service elements

- WDSL (Web Services Description Language)
  - describe service characteristics
- UUDI (Universal Description, Discovery, and Integration)
  - acts as a central repository for web services
- SOAP (Simple Object Access Protocol)
  - for remote method invocation
    - defines the message structure, encoding /representation rules
- JAX-WS
  - Java API for XML based services
    - RPC over XML
    - Messages with SOAP
    - HTTP as a transport protocol
  - Hides the complexity of SOAP from user
    - automatic conversion to/from SOAP messages on API calls

Web service elements



WS in JBoss

- On JBoss, the support for web services is provided by deploy/jbossws.sar.
  - Java EE applications can expose a web service from
    - The EJB tier using a stateless session bean
    - The web tier using a plain Java object
  - Both server-side components and Java EE client applications have a standard way of declaring references to external web services
  - The HTTP transport is handled by Tomcat
  - SOAP handling is provided by the embedded Apache's Axis engine.
- http://localhost:8080/jbossws/services
  - To view the list of all web services deployed on your server

WS development in JBoss

- Jboss provides annotations to turn a POJO to a WS endpoint (JBIA/277)
  - WebService
    - annotation on the class
    - optional portName and wsdlLocation attributes
  - WebMethod
    - annotation on a class method
  - WebContext
    - provides additional context information, such as
      - Virtualhost, contextRoot
      - Security: authMethod, transportGuarantee
- Classes are deployer as a simple .war
    - .wsr : same as .war, but deployed before .war's
    - a custom wdsl could be provided (referred in @WebService)
      - if missing, portName is derived from servlet name

WS development in JBoss

- Helper utilities
  - wsprovide
    - analyses a (properly annotated) class, and produce the corresponding WSDL
  - wsconsume
    - produce a WSDL file, and creates stub classes for development
  - wsrunclient
    - small wrapper around the simplified WS client, to load all the plumbing code needed to reach a web service. (used with wsconsume)

Lab 8: Install simple WS service

- Install web service example from JBoss tutorial
- Secure the web services with jboss.xml

10. Monitoring JBoss

- Objectives
-
  - Command line tools for JVM monitoring
  - Visualvm
  - Garbage collectors
    - Eden, survivor, tenured
    - GC tricks
  - Typical problems

Command line tools

- jps
  - JVM process status tool
  - little more than ps -ef |grep java
- jinfo
  - prints the given JVM startup params
- jmap
  - heap and shared object memory map printing
- jstack
  - print Java stacktrace to detect problems like deadlock
- jstat/jstatd
  - collects JVM statistics, such as GC.
- twiddle
  - general purpose JMX console

JPS

- give a list of the actual JVMs running
  - vmid/lvmid is input for other processes
  - vmid is usually a pid
- Can connected to jstatd over rmi
- Useful option
  - -l: full package name, jar file name
  - -v: all arguments passed

```
$> jps -l
32058 sun.tools.jps.Jps
31853 org.jboss.Main
30840 sun.tools.jstatd.Jstatd
7079 org.rhq.enterprise.agent.AgentMain
30830 sun.rmi.registry.RegistryImpl
```

Jinfo

- Gives you a summary of
  - JVM version
  - Complier options used
  - Classpath
  - Locale info
  - Command line options

```
$> jinfo -flags 2630

Attaching to process ID 2630, please wait...

Debugger attached successfully.

Server compiler detected.

JVM version is 23.5-b02

-Dprogram.name=run.sh -Xms128m -Xmx512m -XX:MaxPermSize=256m
-Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000 -Dcom.sun.management.jmxremote.port=12345
-Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote -XX:HeapDumpPath=/dump -Djava.net.preferIPv4Stack=true
-Djava.endorsed.dirs=/jboss5/lib/endorsed
```

Jstack

- Prints stacktrace from different sources
  - pid
  - core/executable
  - serverid/remote host
- Thread states
  - RUNNABLE
  - TIMED-WAITING/WAITING

```
$> jstack 2630|head -20
2013-01-17 14:28:43
Full thread dump Java HotSpot(TM) Server VM (23.5-b02 mixed mode):


"RMI TCP Connection(112)-10.201.0.1" daemon prio=10 tid=0x7fd8b000 nid=0xe38
runnable [0x773fe000]
   java.lang.Thread.State: RUNNABLE
at java.net.SocketInputStream.socketRead0(Native Method)
at java.net.SocketInputStream.read(SocketInputStream.java:150)
at java.net.SocketInputStream.read(SocketInputStream.java:121)
at java.io.BufferedInputStream.fill(BufferedInputStream.java:235)
at java.io.BufferedInputStream.read(BufferedInputStream.java:254)
- locked <0x8afd20b0> (a java.io.BufferedInputStream)
at java.io.FilterInputStream.read(FilterInputStream.java:83)
at sun.rmi.transport.tcp.TCPTransport.handleMessages(TCPTransport.java:535)
at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run0(TCPTransport.java:808)
at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(TCPTransport.java:667)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1110)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:603)
at java.lang.Thread.run(Thread.java:722)
```

Jstat

- Mainly garbage collection statistics
  - http://docs.oracle.com/javase/1.5.0/docs/tooldocs/share/jstat.html
  - Size and utilzation of heap area
    - Young/tenured, Eden/Suvivor, Perm space
    - Number of young/old collections.

```
$> jstat -gcutil 2630 1s 3
  S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
  9.76   0.00  51.53  63.31  99.67    162    3.563     9    5.516    9.079
  9.76   0.00  64.19  63.31  99.68    162    3.563     9    5.516    9.079
$> jstat -gcnew 2630 1s 10
 S0C    S1C    S0U    S1U   TT MTT  DSS      EC        EU     YGC     YGCT
13440.0 13440.0    0.0 1277.7 15  15 6720.0 108096.0  23450.7    163    3.580
13440.0 13440.0    0.0 1277.7 15  15 6720.0 108096.0  23713.5    163    3.580
13440.0 13440.0    0.0 1277.7 15  15 6720.0 108096.0  24246.2    163    3.580
```

Twiddle

- A general JMX tool but can be used a a getter/setter of some MBean attributes
- Useful targets
  - jboss.system:type=ServerInfo TotalMemory
  - jboss.j2ee:jar=some.jar,name=SomeBean,service=EJB3 InvokeStats
  - jboss.jca:name=DataSource,service=ManagedConnectionPool MaxConnectionsInUseCount

```
# twiddle.sh get jboss.system:type=ServerInfo TotalMemory
TotalMemory=400576512
# twiddle.sh get jboss.jca:name=MySqlDS,service=ManagedConnectionPool
MaxConnectionsInUseCount
MaxConnectionsInUseCount=1
#twiddle.sh invoke jboss.system:type=ServerInfo listThreadDump|elinks -dump
 Thread: RMI TCP Connection(437)-10.201.0.1 : priority:5, demon:true,
       threadId:413, threadState:RUNNABLE
         java.net.SocketInputStream.socketRead0(Native Method)
         java.net.SocketInputStream.read(SocketInputStream.java:150)
```

VisualVM/Jconsole

- Jstatd/JMX based data collection, graphical output
  - JMX
    - Either Jboss JMX remote or JVM HotMBeanService
    - http://docs.oracle.com/javase/6/docs/technotes/guides/management/agent.html
  - Jstatd
    - A fronted service for JMX, and jps, jinfo, etc.
  - VisualVM
    - Snapshots
    - Profiler for CPU/Memory
    - Load in core files, heap dump, and analyse thread/heap statistics offline
    - Wrapper for all JConsole plugins
    - VisualGC plugin

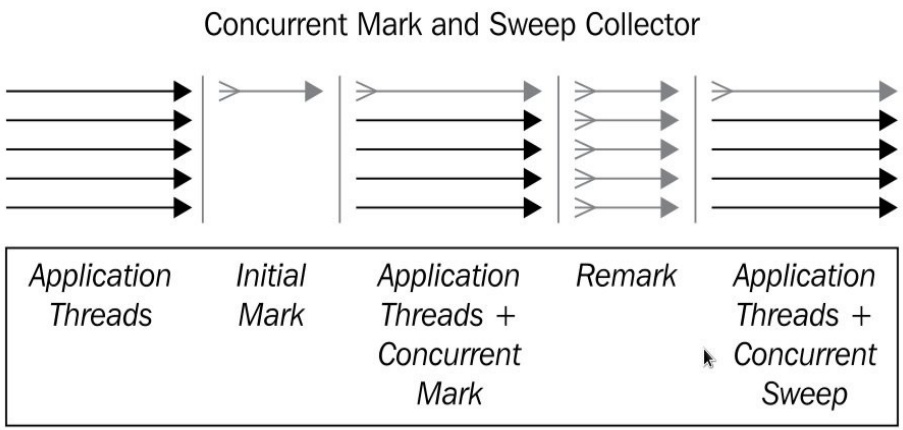
JVM tuning

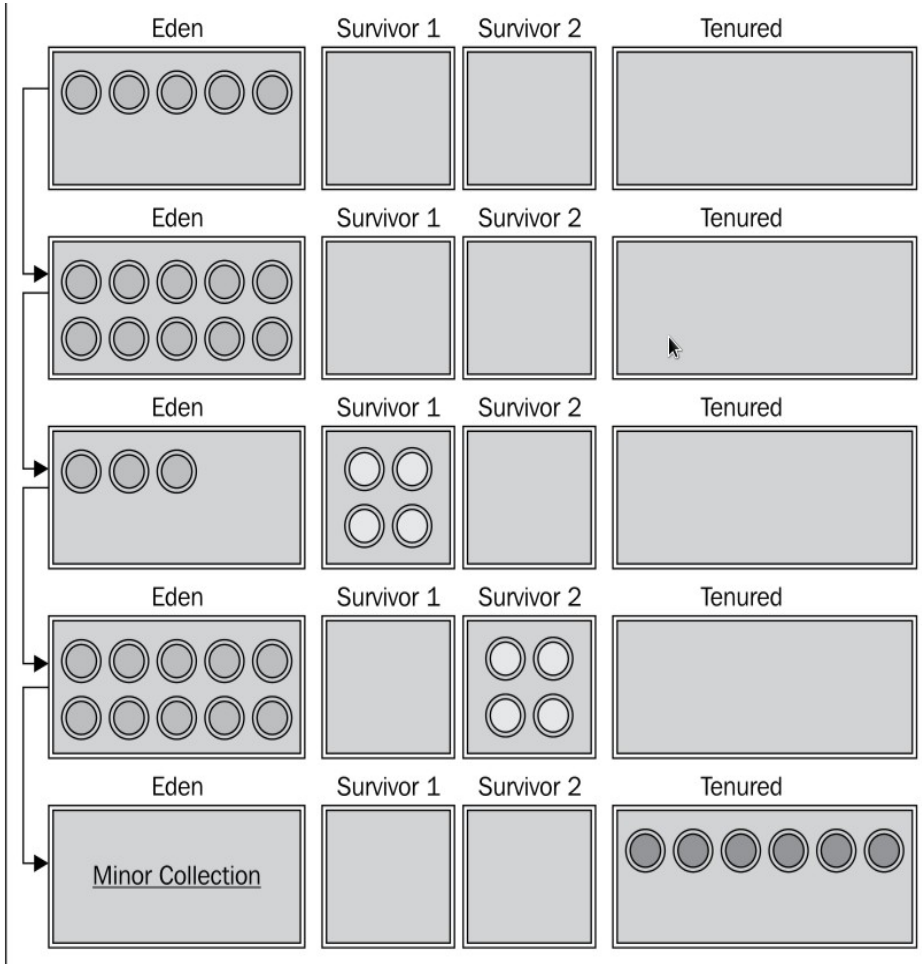
- Tuning garbage collection
  - Size of heap: -Xms,Xmx
  - -XX:Newsize/NewRatio: young/tenured size/ratio (33%)
  - -XX:SurvivorRatio: Eden/Survivor (1/8, 1/12)
  - XX: MaxPermSize
  - XX: MaxTenuringThreshold: how many minor collection to be survived to get tenured
  - GC logs
    - -verbose:gc
    - -XX:+PrintGCDetails
    - -Xloggc:<filename>

```
[GC      134841K->53479K(296960K),0.0743071 secs]
[GC      135399K->55534K(296960K),0.0906430 secs]
[GC      137454K->56650K(296960K),0.0767066 secs]
[Full GC 126196K->56872K(296960K),0.6024574 secs]
```

GC algorithms

- Stop-the-world issue
  - Pause time goal
  - Throughput goal
- Serial/Parallel
  - -XX: +UseParallelGC,
  - -XX: ParallelGCThreads
  - -XX: +UseParallelOldGC
  - -XX:+UseConcMarkSweepGC
  - -XX:+UseG1GC





Tipical problems

- To many tenured collections
  - To small size for young generation
  - To large survivor/eden
- Idle cores but slow application
  - Use parallel GC
- To long stop the world times
  - Use ConcMarkSweep or G1

Lab 9: Monitoring

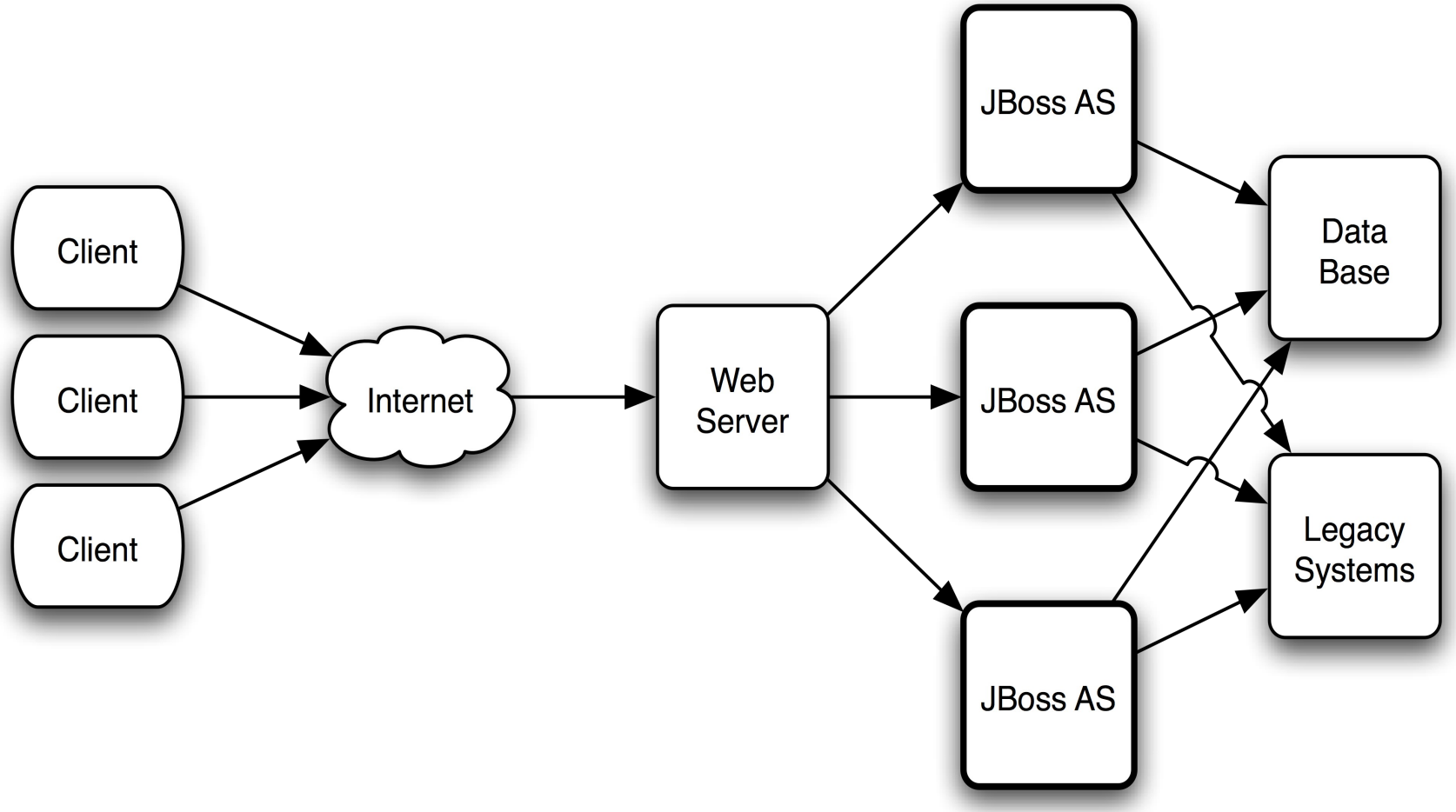
- Install and use monitoring tools
  - Take VisualVM
    - install plugin VisualGC
    - install jstatd locally
  - Start memhog.war
    - Examine gc activity with VisualGC, jstat
  - Install monitor bean
    - Start ConsoleAlertService for free memory smaller then 100Mb
    - Use webconsole to do the same

11. JBoss Clustering

- Objectives
  - Clustering terminology
  - Clustering components
    - JGroups
    - JBoss Cache/Infinispan
  - Load balancing for Apache
  - Replication configuration
  - Clustering related config/annotations

JBoss Clustering

- Cluster overview
  - http://docs.jboss.org/jbossclustering/cluster_guide/5.1



JBoss Clustering

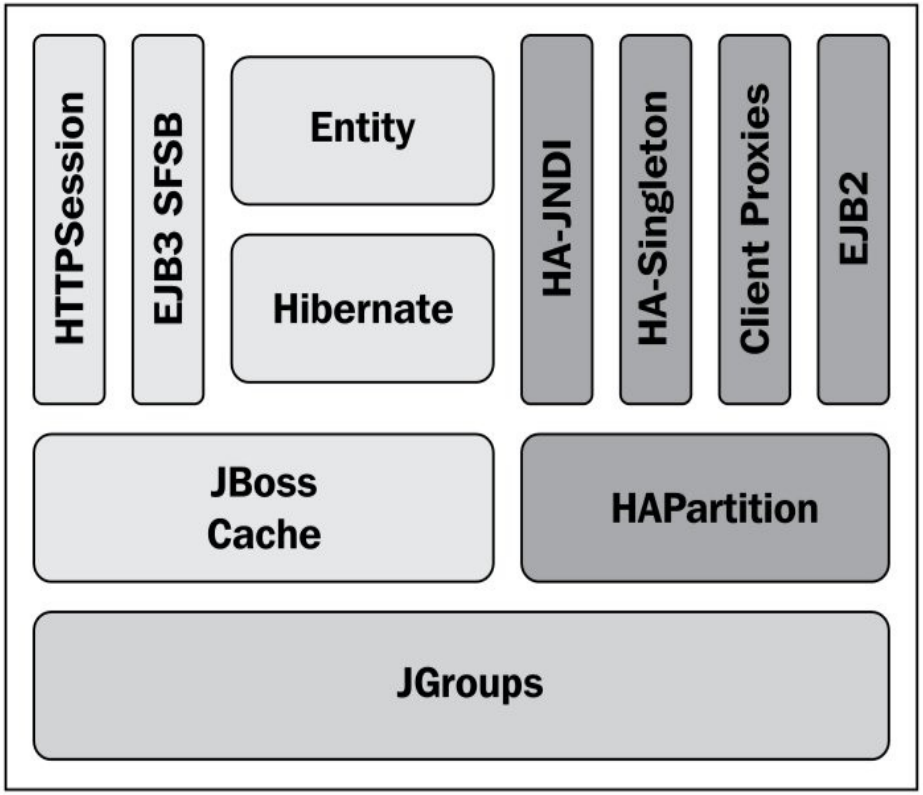
- Config parameters
  - run.sh
    - -c all : default cluster aware
    - -g MyPartition: partition name
    - -u: multicast address
    - -Djboss.messaging.ServerPeerID
      - if using JMS
    - -Djgroups.bind_addr
      - if you want to separate jgroups traffic from others

Setup LB for apache

- Possible configs
  - mod_jk
    - JKWorkersFile workers.properties
    - workers.properties
      - worker name must match jvmRoute prop in the Tomcats server.xml/Engine
      - sticky session
  - mod_proxy/mod_balancer
    - Proxy params
      - BalanceMember
      - ProxySet
        - lbmethod=byrequests|bytraffic
        - stickysession=JSESSIONID
        - nofailover

HA services

- Session replication
- HAJNDI
- Clustered SSO
- SFSB state replication
- Entity Bean clustering



Session replication

- cluster/jboss-cache-manager.sar
  - Cache Mode
    - REPL_SYNC, REPL_ASYNC
  - Caching Configuration: replication queue
  - Cluster Name and Configuration
- jboss-web.xml
  - Replication trigger
    - SET, SET_AND_GET, SET_AND_NON_PRIMITIVE_GET, ACCESS
  - Replication granularity
    - SESSION, ATTRIBUTE, FIELD

Session replication

```
<bean name="CacheConfigurationRegistry" ...>

...

        <bean name="StandardSessionCacheConfig"
class="org.jboss.cache.config.Configuration">

         ...

         <property name="clusterName">${jboss.partition.name:DefaultPartition}-
SessionCache</property>

         <property name="multiplexerStack">${jboss.default.jgroups.stack:udp}</property>

         <property name="fetchInMemoryState">true</property>

         <property name="nodeLockingScheme">PESSIMISTIC</property>

         <property name="isolationLevel">REPEATABLE_READ</property>

         <property name="useLockStriping">false</property>

         <property name="cacheMode">REPL_ASYNC</property>

         <property name="syncReplTimeout">17500</property>

         <property name="lockAcquisitionTimeout">15000</property>
```

Session replication

```
#jboss-web.xml
<jboss-web>
  <replication-config>
    <replication-trigger>SET_AND_NON_PRIMITIVE_GET</replication-trigger>
    <replication-granularity>SESSION</replication-granularity>
  </replication-config>
</jboss-web>
```

```
#web.xml
<web-app>
  <display-name>Session Test</display-name>
  <description>...</description>
  <distributable/> <!--  -->
  ...
</web-app>
```

Clustered SSO

- Enable by adding to Tomcat's all/deploy/jbossweb.sar/server.xml
  - (all or other all type configuration) file within a <Host> element
  - Be recognized on any web app associated with a same virtual host inside a cluster
  - Authentication replication handled by HTTP session replication service
  - Applications do not have to be explicitly enabled for session replication
- Replace Valve

  org.apache.catalina.authenticator.SingleSignOn

  to

  org.jboss.web.tomcat.service.sso.ClusteredSingleSignOn

Clustering with HA-JNDI

- Maintains a cluster-wide context tree
  - Available as long as there is at least one instance left in the cluster
  - Each instance also maintains its own local JNDI tree
  - Applications can bind to either one
  - HA-JNDI delegates to local when it cannot find the object with in the cluster-wide context
  - HA-JNDI not compatible with non-JNP JNDI implementations on local side (e.g. LDAP)
  - Objects bound to HA-JNDI are replicated to other

Clustering with HA-JNDI

- Server side:
  - deploy/cluster/hajndi-jboss-beans.xml
    - bindAddress,
    - port: naming proxy
    - rmiPort: dynamic proxy
    - autodiscovery (enabled by default)
      - autoDiscoveryAddress,autoDiscoveryGroup
        - multicast IP/port for autodisc.
      - autoDiscoveryBindAddress
        - the listen side
    - loadBalancePolicy:defaults to rr
- client side: conf/jndi.properties

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.provider.url=server1:1100,server2:1100,server3:1100
```

Clustering EJBs

- SLSB
  - No state, nothing to be replicated
  - But load balancing via dynamic-proxy is needed
- SFSB
  - Client state has to be replicated
    - JBoss Cache is involved, same as for HTTP Session caching
- Clustering is not automatic
  - Use @Clustered annotation
    - breaks compatibility
  - Use AS descriptors
    - deploy time clustering
    - not 100% true, that you can do it without the developer

Clustering EJBs

- @Clustered elements
  - loadBalancing policy
    - FirstAvailable
      - Each clients stick to a node (randomly selected). Switch to another node if the one used dies.
      - This is the **only** available for SFSB
    - FirstAvailableIdenticalAllProxies
      - all clients stick to the same node (randomly selected). Switch to another node if the one used dies
    - RandomRobin
      - Each request is addressed to a random node in the cluster
    - RoundRobin
      - Hit sequentially each node in the cluster
  - partition
    - by default use the default partition

Clustering SFSBs

- @CacheConfig
  - factory default params are
    - deploy/cluster/jboss-cache-manager.sar/META-INF/jboss-cache-manager-jboss-beans.xml
  - jboss.xml/cache-config
    - idleTimeoutSeconds:
      - Time in seconds a SFSB can be unused before being passivated (default 300)
    - maxSize: maximum number of beans that can be cached (default 10000)
    - name: sfsb-cache is the default for SFSB
    - removalTimeoutSeconds
      - Time in seconds a SFSB can be unused before being deleted by the cache (default 0)
    - replicationIsPassivation
      - Should a replication be considered as passivation? (default true)

Clustering references

- http://www.theserverside.com/news/1364410/Under-the-Hood-of-J2EE-Clustering
- http://docs.jboss.org/jbossclustering/cluster_guide/5.1/html/index.html

Lab 11: Clustering

- Setup a jboss on the second node
  - refer Lab1
- Start the “all” config
  - set a unique partion name (like cl1)
  - set a unique multicast address (224.11.1.1 for cl11)
- Setup mod_cluster
- Deploy apps
  - counter.war
  - counter.ear