

Table of Contents – JBoss AS 5 Administration

<i>Administering the JBoss® 5 Application Server</i>	<i>1</i>
Workshop Overview	2
Workshop Objectives	3
Release Level	4
Working with Windows and/or Unix	5
Typographic Conventions	6
Labs	7
Session 1: Introduction and Installation	8
Lesson Objectives	9
Java EE Overview	10
Java Has Multiple Platforms	11
What Is Java EE?	12
Some Common Java EE APIs	13
What Are Java EE Application Servers?	14
Commercial Java EE Products - Examples	15
Multi-tiered Architecture	16
Example: Web-Based Architecture	17
Web-Based Architecture with EJB	18
Rich Client Architecture with EJB	19
Supporting Both Web Clients and Rich Clients	20
Playing Server Games	21
SOA – Service Oriented Architecture	22
SOA Architecture	23
JBoss AS Overview	24
JBoss Application Server (JBoss AS)	25
JBoss AS History and JBoss the Company	26
JBoss the Company	27
JBoss 5 Application Server	28
Server Architecture - the Microcontainer	29
Legacy Server Architecture - JMX Microkernel	30
Services Provided by the MC and MK	31
JBoss AS 5.x - Architecture	32
JBoss AS 5.x - Common Technologies	33
Installing and Running JBoss AS	34
System Requirements	35
Choosing Your JVM	36
Operating System Considerations	37
Environment Variables	38
Installing JBoss AS	39
Server Configurations	40
Zip Installation	41
Graphical Installer	42
Starting JBoss AS	43
Starting JBoss AS - the Microcontainer	44
Starting JBoss AS - the Microkernel	45
The Server Is UP !	46
Stopping JBoss	47
Lab 1.1 – Setting Up JBoss AS	48
More about Starting and Stopping JBoss AS	60

The <jboss>\bin Directory and Its Files	61
The run Script	62
JBoss System Properties and run -D	63
run -D	64
System Properties and Directory Structure	65
Other run Options	66
Heap Allocation	67
The shutdown Script	68
Using the shutdown Script	69
Windows: Running JBoss AS as a Service	70
Lab 1.2 – Starting and Stopping JBoss AS	71
Documentation	75
Documentation Overview	76
The <jboss>\docs Directory	77
Lab 1.3 – Review the Documentation	78
Session 2: JBoss AS Structure and Architecture	82
Lesson Objectives	83
Server Directory and File Structure	84
Server Configurations	85
Server Configuration Structure	86
Configuration Directory Structure	87
The conf Directory	88
bootstrap.xml and bootstrap*.xml Files	89
Sample MC Configuration File	90
The Legacy jboss-service.xml file	91
jboss-service.xml Example	92
Services Deployed in jboss-service.xml	93
Microcontainer (MC) and Microkernel (MK)	94
The lib Directories	95
The Deploy Directory	96
The Deploy Directory	97
Summary of Directories	98
Server Logging	99
The log Directories and Logging Configuration	100
log4j Overview	101
Appenders and Priority	102
JBoss Boot Logging	103
JBoss Server Logging	104
The jboss-log4j.xml File	105
Controlling Application Logging	106
Creating New Appenders	107
Lab 2.1 – Working with JBoss	108
The JBoss Microcontainer	120
Overview of the Microcontainer (MC)	121
About Dependency Injection (DI)	122
MC Configuration File Structure (XML)	123
<bean>'s Attributes & Sub-elements	124
How JBoss AS Boots - Low Level Details	125
Core Services Started in the Bootstrap	126
A Complete Service Definition	127
Bean Definition - jboss-beans.xml	128
MC Bean Deployments	129

JMX and the Microkernel	130
Overview of JMX	131
Overview of JMX	132
JMX Architecture	133
JMX Object Names	134
About the JBoss Microkernel (MK)	135
Relation between JBoss Microkernel and JMX	136
About MK Based JBoss Services	137
Service Definition - <i>jboss-service.xml</i>	138
MK Bean Deployments	139
Providing JMX Access to MC Beans	140
The Admin Consoles	141
The JMX Console	142
The JMX Console – Home Page	143
The JMX Console	144
Looking at MyDependentService	145
Things You Can Do with the JMX-Console	146
JMX Console - Logging Control	147
twiddle – Command line JMX	148
The Web Console	149
The Web Console	150
The Admin Console (Embedded Jopr)	151
The Admin Console	152
Jopr Overview	153
Jopr / JBoss Operations Network (JON)	154
Jopr Structure	155
Jopr Structure	156
Jopr Agent Structure	157
GUI Console - The Dashboard	158
Console - A Server Detail Page	159
Console - A Service Detail Page	160
Lab 2.2 – Working with the Consoles	161
Session 3: Deployment	170
Lesson Objectives	171
Deployment Archives	172
Java EE and JBoss Archives	173
Java EE Archive Structure	174
WAR / EAR Structure	175
Deployers and Deployment	176
JBoss 5 AS Deployment Architecture	177
Deploying Into JBoss AS	178
Configuring/Controlling Deployment Scanning	179
Deployment Options	180
Deployment Options	181
Working With exploded Deployments	182
Deployment Ordering	183
Nested Deployments	184
Classloaders	185
Classloading	186
Classloading in Java EE Servers	187
Classloader Hierarchy - Normal Java EE	188
JBoss Class Loader Repository	189

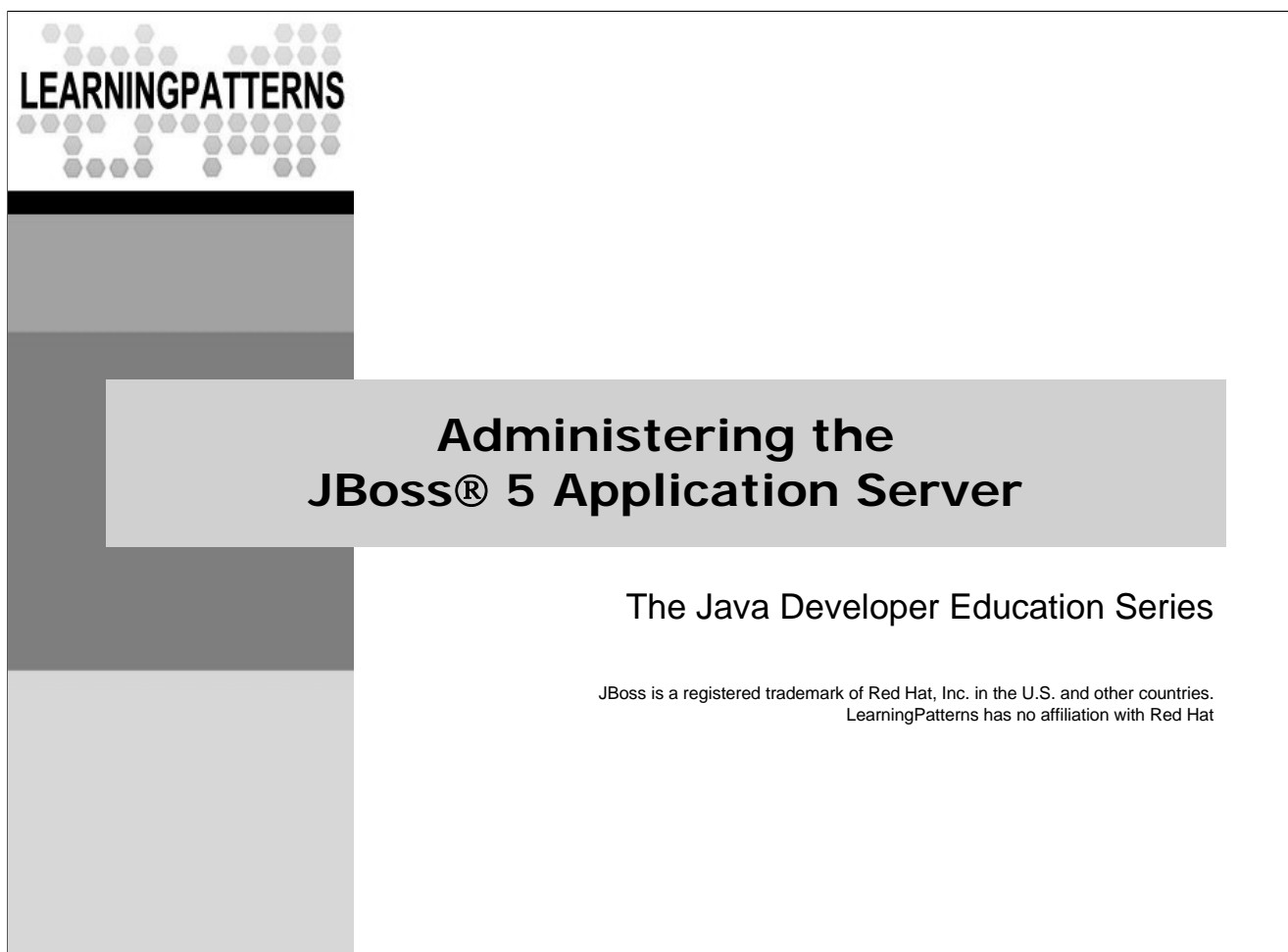
JBoss Class Loader Repository	190
Creating Isolated Deployments	191
Lab 3.1 – Working with the Deployments	192
Session 4: Web Container, Port Configuration, JNDI, and DataSources	200
Lesson Objectives	201
The Web Container	202
Web Container Overview	203
JBossWeb Additional Capabilities	204
Configuration Files	205
server.xml Structure	206
Configuring the HTTP Connector - server.xml	207
Other Connectors - server.xml	208
Access Logging - server.xml	209
ROOT.war - The Root Web Application	210
The Default <i>web.xml</i> File	211
Monitoring the Web Container	212
The Tomcat Status Servlet	213
Lab 4.1 – Working with the Web Container	214
Port Configuration / ServiceBindingManager	219
Overview	220
ServiceBindingManager Definition	221
ServiceBindingManager Definition	222
Using ServiceBindingManager	223
Accessing Services on Different Ports	224
Summary of Default Ports	225
Lab 4.2 – ServiceBindingManager	226
JNDI	230
Review - JNDI Overview	231
JNDI Name Tree	232
JNDI Name Tree Diagram	233
Review - JNDI in the Java EE Platform	234
Configuring JNDI Clients	235
How JNDI Works	236
Configuring Naming on JBoss AS	237
Tunneling JNDI Through HTTP	238
Lab 4.3 – JNDI	239
Datasources	243
Java and Database Connectivity	244
Java EE and Datasources	245
Connection Pooling	246
Non-Pooled vs. Pooled	247
Getting a Pooled Connection - Illustrated	248
Closing a Pooled Connection - Illustrated	249
Datasources in JBoss AS	250
Configuring Datasources in JBoss AS	251
Datasource Configuration File	252
Hypersonic Datasource Configuration	253
Sample Datasource Configuration	254
Other Configuration Elements	255
Configuring non-tx and XA datasources	256
The Hypersonic Database and DefaultDS	257
Changing the Hypersonic Database	258
Lab 4.4 – Datasources	259

Session 5: Other Services	273
Lesson Objectives	274
Remote Object Invokers	275
UnifiedInvoker – JBoss Remoting	276
JBoss Remoting Details	277
UnifiedInvoker Configuration	278
Connector Configuration	279
EJB3 Invoker Configuration	280
Older JBoss AS RMI Invokers	281
HTTP Invoker	282
HTTP Invoker - http-invoker.sar	283
How the HTTP Invoker Works	284
Configuring the HTTP Invoker SAR	285
Using the HTTP Invoker	286
JMS (Java Message Service)	287
Note on JMS	288
Messaging Overview	289
Publish/Subscribe - Illustrated	290
Point-to-Point - Illustrated	291
What Is JMS - Java Message Service?	292
JBoss Messaging	293
JBoss Messaging High Level	294
ServerPeer Configuration	295
Persistence: hsqldb-persistence-service.xml	296
Persistence Manager Configuration	297
Post Office / User Manager	298
ConnectionFactory Configuration	299
ConnectionFactory Configuration	300
messaging-jboss-beans.xml / Security	301
Remoting Connector Configuration	302
Configuring Destinations	303
Configuring Destinations	304
JBossMessaging and the JMX Console	305
Destinations and the JMX Console	306
Lab 5.1 – Working with JMS	307
Web Services	320
SOA / Web Services	321
SOA / Web Services	322
SOAP	323
WSDL	324
JBossWS - Web Services	325
JBossWS Management Pages	326
JBossWS and JMX Console	327
Lab 5.2 – Web Services	328
Other Services	335
- EJB: Enterprise JavaBeans -	336
- JMX Invoker Adapter -	337
- RMI Class Downloading -	338
Session 6: Security in JBoss AS	339
Lesson Objectives	340
Java EE Security Overview	341

Security Requirements	342
Java EE Security in JBoss AS	343
Transport Level Security with HTTPS/SSL	344
Java EE Security Overview	345
Java EE Security Example	346
Java EE Declarative Security	347
JMX Console - Security Role Declaration	348
Specifying Security Constraints	349
Security Constraints - Deployment Descriptor	350
Security Constraints - Deployment Descriptor	351
JBoss AS Security	352
Overview of Security in JBoss AS	353
JBoss Security Mechanism Overview	354
Security Domains and Login Modules	355
JBoss Login Modules	356
UsersRolesLoginModule - File Based	357
UsersRolesLoginModule Example	358
Specifying Security Domains	359
Encrypting User Passwords	360
Configuring Password Hashing	361
Generating Hashed Passwords	362
Lab 6.1 – Securing JMX Console	363
DatabaseServerLoginModule - DB Based	370
Security Domain - DatabaseServerLoginModule	371
Tables and Queries for Messaging Example	372
LdapLoginModule - LDAP Based	373
Using the LdapLoginModule	374
JBoss Security MBeans	375
The JaasSecurityManager MBean	376
Lab 6.2 –DatabaseServerLoginModule	377
Encrypting Datasource Passwords	384
Encrypting Database Login Passwords	385
Using SecureIdentityLoginModule	386
[Optional] Lab 6.3 – Encrypting Datasource Passwords	387
TLS/SSL and HTTPS	391
TLS/SSL & HTTPS - Transport Level Security	392
TLS/SSL Meets Some Security Needs	393
TLS / SSL Requires Server Setup	394
The keytool Program	395
Using the keytool Program	396
keytool Examples	397
Enabling Tomcat HTTPS	398
Testing if HTTPS Works	399
Requiring HTTPS on a Web App	400
[Optional] Lab 6.4 – Enabling HTTPS	401
Securing JBoss Services	406
Securing JBoss - Overview	407
Secure Access to Admin Console/HTTP Invoker	408
Secure Access to JMX RMI Invoker Adapter	409
Remove/Restrict Web Class Loading Service	410
Secure Transport - EJB 3 Over SSL	411
Setting up SSL for EJB3	412
Secure Transport - EJB 2 / RMI Over SSL	413

Secure Access to JMS	414
Secure Transport - JMS and SSL	415
Secure Transport - RMI/JRMP and SSL	416
Secure Transport - RMI/JRMP and SSL	417
Secure Access to Datasource	418
Debugging Security	419
Session 7: Tuning JBoss AS	420
Lesson Objectives	421
Tuning Memory Usage	422
Tuning Overview	423
Tuning Overview	424
Memory, Memory, Memory	425
JVM - Heap Size	426
JVM - Perm Space	427
JVM - Garbage Collection	428
JVM - Generational Garbage Collection	429
JVM - Minor vs Full Collections	430
JVM - Optimizing Generational GC	431
JVM - Parallel Garbage Collectors	432
JVM - Choosing Garbage Collectors	433
OS - Memory and Threading	434
OS and Hardware - Other Considerations	435
Resource Tuning	436
Database Tuning	437
Datasource Tuning	438
Tomcat Tuning - Connectors	439
Tomcat Tuning - JSP	440
Other Tomcat Tuning	441
Logging	442
Other Services	443
Removing Services	444
Architecture Issues	445
Playing Server Games	446
Topology Performance Ramifications	447
Lab 7.1 –Tuning	448
Session 8: Clustering	454
Lesson Objectives	455
Overview	456
Clustering Overview	457
Cluster Partitions	458
Cluster Partitions	459
JGroups	460
Partition Configuration	461
JGroups Troubleshooting	462
JGroups Troubleshooting	463
Clustered Services	464
Architecture - Client Side Interceptors	465
Architecture - Load Balancer	466
Load Balancing Policies - Client Interceptor	467
HA-JNDI - High Availability JNDI	468
Clustered JNDI Architecture	469
HA-JNDI Server Side Configuration	470

HA-JNDI Client Side Configuration	471
HA-JNDI Client Side Autodiscovery	472
Clustered EJB	473
Clustered HTTP	474
The JBoss AS Farm Service	475
Clustered Singleton Services	476
Clustered JBoss Messaging	477
Clustered JBoss Messaging	478
Sucker Password	479
JBoss EAP – The Production Configuration	480
Lab 8.1 – Clustering Setup, EJB and JMS	481
Load Balancers	492
Load Balancers - Clustered HTTP	493
Load Balancers	494
Sticky Session Configuration in JBossWeb	495
Load Balancer and Sticky Sessions	496
mod_cluster	497
mod_cluster Overview	498
mod_cluster Overview	499
mod_cluster Advantages	500
Module configuration in Apache httpd	501
Configuration in httpd.conf (1 of 2)	502
Configuration in httpd.conf (2 of 2)	503
Manager Display	504
Installation / Configuration in httpd	505
Configuration in JBoss	506
Lab 8.2 – HTTP Load Balancing – mod_cluster	507
mod_jk	517
Configuration in Apache httpd	518
Configuration of Workers (Tomcat Nodes)	519
Configuration for a Load Balancer	520
Load Balancer and Sticky Sessions	521
Mounting Applications in an External File	522
Configuration Considerations	523
Lab 8.3 – HTTP Load Balancing with mod_jk	524
Recap	535
Recap of what we've done	536
Resources	537
End of Session	538



Notes

- ◆ JBoss is a registered trademark of Red Hat, Inc. in the U.S. and other countries. LearningPatterns has no affiliation with Red Hat

Workshop Overview

- ◆ This is an in-depth course covering the Administration of the JBoss 5.x Application Server (JBoss AS)
- ◆ It includes the following sessions
 - Session 1: **Introduction and Installation of JBoss AS**
 - Session 2: **JBoss AS Structure and Architecture**
 - Session 3: **Deployment**
 - Session 4: **The Web Container, Port Configuration, JNDI, and DataSources**
 - Session 5: **Other Services**
 - Session 6: **Security in JBoss AS**
 - Session 7: **Tuning JBoss AS**
 - Session 8: **Clustering**

Notes

- ◆ Though this course is based on JBoss AS 5.1.0, it is fairly applicable to most 5.x versions with minor changes
- ◆ It is not, however, suitable for JBoss AS 4.x
 - JBoss AS 5.x, introduces many changes, including Java EE 5 compatibility, and a completely new server kernel
 - These are not available in JBoss 4

Workshop Objectives

- ◆ At completion you should have a good understanding of JBoss 5 AS, and be able to manage and use its capabilities in the following areas:
- ◆ **JBoss Architecture:** Understand what a Java EE server is, and be familiar with the JBoss architecture (the Microcontainer and Microkernel)
- ◆ **Installation and Basic Configuration:** Understand how to install JBoss AS, understand the structure of a JBoss AS installation, know how to monitor the server, and configure the server for production use
- ◆ **Deployment:** Understand the different types of deployment that are supported, their packaging, and be able to deploy applications
- ◆ **Web Container:** Understand the structure of the (Tomcat based) JBoss Web container, and be able to configure it
- ◆ **Services:** Be familiar with the available services and how to configure them (including DataSources, JNDI, EJB/RMI, JMS, etc.)
- ◆ **Security:** Understand the JBoss AS security architecture, and configure the various security capabilities
- ◆ **Tuning:** Tune JBoss AS, as well as the various services
- ◆ **Clustering:** Understand and use JBoss clustering

Notes

Release Level

- ◆ This course contains material for running and managing JBoss AS using the following platforms:
 - **JBoss® 5.1.0 GA** (suitable for other 5.x versions also)
 - **Java 5 or 6** (Java Development Kit 1.5.0_xx)
 - Java 5 or later is required since some of the technologies supported depend on Java 5 capabilities – Java 6 should work also (see notes)
 - **Apache Ant 1.7.x** (any recent ant version should work)
 - **A basic editor of your choice** (see note)
- ◆ All labs tested on **Unix**, and **Microsoft Windows**
 - Lab instructions are given for both environments
 - Paths are generally shown using a backslash (\) (e.g. server\default\deploy), unless we are giving instructions specific Unix, in which case we'll use a forward slash (/)

Notes

- ◆ JBoss 5 AS works with Java 6, but is only certified with Java 5
 - However, you should be fine doing the course with Java 6 – however, read the below from the JBossAS 5.0.0.GA release notes (relevant for 5.1 also)
 - "JBossAS 5.0.0.GA can be compiled with both Java5 & Java6. The Java5 compiled binary is our primary/recommended binary distribution. It has undergone rigorous testing and can run under both a Java 5 and a Java 6 runtime. When running under Java 6 you need to manually copy the following libraries from the JBOSS_HOME/client directory to the JBOSS_HOME/lib/endorsed directory, so that the JAX-WS 2.0 apis supported by JBossWS are used:
 - jbossws-native-saaj.jar, jbossws-native-jaxrpc.jar, jbossws-native-jaxws.jar and jbossws-native-jaxws-ext.jar
 - The other option is to download the jdk6 distribution (jboss-5.0.0.GA-jdk6.zip) in which case no configuration changes are required.
- ◆ Ant is used to build the lab applications, and all recent versions of ant should work fine
 - The ant build files are straightforward, and you don't normally need to build the labs – if you don't rebuild the labs, ant is not generally needed
- ◆ If you don't have an editor, we recommend Notepad++ which is downloadable at:
<http://notepad-plus.sourceforge.net/uk/download.php>

Working with Windows and/or Unix

- ◆ JBoss AS runs under multiple environments
 - We support MS Windows and Unix variants in the course
 - Since there are so many flavors of Unix, e.g. many varieties of Linux, Mac OS X, etc, we refer to it as **nix* in the course
- ◆ In the slides, we usually give examples of both Windows and **nix* variants where there are differences
 - Sometimes **nix* versions will appear in the notes
- ◆ If there are no major differences, we only show one example
 - We will generally show the Windows variant
 - e.g. we might show ***run -c minimal*** (Windows) as opposed to ***./run.sh -c minimal*** (Unix)
 - We usually use the Windows path separator (backslash \) in these
 - You'll need to execute ***./run.sh*** on Unix – even if you see ***run***
 - You should be able to follow it easily

Notes

Typographic Conventions

- ◆ Code that is inline in the text will appear in a fixed-width code font, such at this:

JavaTeacher teacher = new JavaTeacher()

- Any class names, such as *JavaTeacher*, method names, or other code fragments will also appear in the same font
- If we want to emphasize a particular piece of code, we'll also bold it (and in the slide, change its color) such as ***BeanFactory***
- Filenames will be in italics, such as *JavaInstructor.java*
- We sometimes denote more info in the notes with a star *
- Lastly, longer code examples will appear in a separate code box as shown below

```
package com.javatunes.teach;  
public class JavaInstructor implements Teacher {  
    public void teach() {  
        System.out.println("BeanFactories are way cool");  
    }  
}
```

Notes

- ◆ There really isn't anything we need to tell you here - but the star * means it would be useful to take a look

Labs



- ◆ The workshop consists of approximately 50% discussion, 50% hands-on lab exercises, including a series of brief labs
 - The labs are contained directly in the course book, and have detailed instructions on what needs to be done
- ◆ The course includes setup zip files that contain skeleton code for the labs
 - Students just need to add code for the particular capabilities that they are working with
 - There is also a solution zip file that contains completed lab code
- ◆ Lab slides contain an icon like the one in the upper right corner of this slide
 - The end of each lab is clearly marked with a stop like this one to the right



Notes



Session 1: Introduction and Installation

Java EE Overview
JBoss AS Overview
Installing and Running JBoss AS
More about Starting and Stopping JBoss AS
Documentation

Notes

Lesson Objectives

- ◆ Understand what Java EE, and a Java EE application server are
- ◆ Learn about the JBoss Application Server (JBoss AS), including its background, capabilities, and current status
- ◆ Be able to install, run and stop JBoss AS

Notes



Java EE Overview

Java EE Overview

JBoss AS Overview

Installing and Running JBoss AS

More about Starting and Stopping JBoss AS

Documentation

Notes

Java Has Multiple Platforms

- ◆ **Java Standard Edition (Java SE)** (previously J2SE)
 - Platform for desktop computers and high-end small devices
 - Contains some APIs that are essential to enterprise applications (e.g., JDBC and JNDI)
- ◆ **Java Enterprise Edition (Java EE)** (previously J2EE)
 - Platform for multitier enterprise applications
 - Depends upon Java SE
 - Adds enterprise-specific APIs such as Servlets/JSP and EJB
 - JBoss AS supports Java EE
- ◆ **Java Micro Edition (Java ME)** (previously J2ME)
 - Platform for small devices such as PDAs and cell phones
 - Subset of Java SE - provides a small-footprint Java environment
- ◆ See note about Java EE 5 and Platform Names

Notes

- ◆ The naming of the Java platform has changed a number of times
 - As of the latest release, Java 5, and Java EE 5, the Java platform names are the ones we use in the slides
- ◆ In the previous release the names were slightly different
 - Java SE was called J2SE (Java 2 Standard Edition)
 - Java EE was called J2EE (Java 2 Enterprise Edition)
 - Java ME was called J2ME (Java 2 Micro Edition)

What Is Java EE?

- ◆ **Java EE** provides technologies to more easily develop **multi-tier, enterprise services**
- ◆ Java EE consists of a number of different parts
 - **Java EE specification** : Defines the APIs and how they interact
 - Sun writes the specification, and third parties (such as JBoss) create implementations that conform to the specification
 - **Java EE Reference Implementation** : Sun's free implementation of Java EE (called Glassfish), which is meant to be a full production level server
 - **Java EE Compatibility Test Suite** : Set of standard tests to determine if a particular product conforms to the specification
- ◆ **JBoss AS 5.x** is a server (commonly known as an application server) that supports the **Java EE 5** level of the specification (and passes the compatibility test suite)

Notes

- ◆ Although Sun is the official owner of all Java specifications, it does not write them alone. It works with vendors and industry experts via the *Java Community Process* (JCP) in order to develop them collaboratively.
- ◆ JBoss AS was one of the first servers, and the first open source server, to be certified for the J2EE 1.4 specification
- ◆ One benefit of the Java EE Reference Implementation is that it serves as a basis for comparing the behavior of a particular vendor's product versus what the specification requires.
 - For example, if you think you are seeing incorrect behavior in a vendor's product, you can test your code on the RI – if things work properly on the RI, you have likely run into a bug in the vendor's product.

Some Common Java EE APIs

- ◆ **Servlet/JSP** Java based HTTP services (Web Container)
- ◆ **EJB** Distributed Transactional Java objects
- ◆ **JMS** Java Message Service - Messaging
- ◆ **JavaMail** An email API
- ◆ **JTA** Java Transaction API
- ◆ **JMX** Java Management eXtensions
- ◆ **Web Services APIs**
 - **JAX-WS** Java API for XML Web Services (supersedes JAX-RPC)
 - **SAAJ** SOAP with Attachments API for Java
 - **JAXR** Java API for XML Registries
- ◆ **JDBC** SQL Based Data Access to Relational DB
- ◆ **J2EE Connector Architecture** Connections to other resources

Notes

- ◆ Recall that Java EE relies upon Java SE. Some of the important APIs from Java SE that are employed in Java EE include:
 - **RMI** Remote Method Invocation
 - **RMI-IIOP** RMI over Internet Inter-ORB Protocol
 - **JDBC** Java Database Connectivity
 - **JNDI** Java Naming and Directory Interface
 - **JAXP** Java API for XML Processing

What Are Java EE Application Servers?

- ◆ **App servers:** Programs that host Java EE technologies
 - Provide integration and management of the various technologies
 - Allow the different technologies to work together efficiently (for example Servlets/JSP and EJB components)
- ◆ Many companies sell commercial Java EE App servers
 - Vendors compete on the quality of their implementations of the open Java EE standard
 - Price, performance, ease of use, support, reliability, scalability
 - Additional functionality, e.g., clustering
 - Vendor reputation and stability
- ◆ The products on the market may support different versions of the specification (most are now at Java EE 5 level)

Notes

- ◆ Java EE 5 has been defined for quite some time now
 - However, development and migration to this new specification was slow
 - As of when this course was written, most of the production servers supported Java EE 5
- ◆ Many vendors offer a suite of Java EE products
 - Base application server platform
 - Products with functionality layered on top of the base server
 - Scaled-down, low-cost alternative to the base server
 - Tools to aid in development and deployment

Commercial Java EE Products - Examples

Vendor	Product
IBM	WebSphere App Server
JBoss	JBoss App Server (Open Source)
Oracle	Oracle App Server Weblogic App Server (formerly BEA)
Sun / Oracle	Glassfish App Server (Open Source)
Sun / Oracle	Sun Glassfish Enterprise Server (supported version of Glassfish)
Apache	Geronimo App Server (Open Source)

◆ This list is not exhaustive!

15

Notes

- ◆ For some projects, an open source server such as JBoss might be a reasonable choice. For example, the Apache Web Server is open source and is one of the most popular Web servers. (In fact, some vendors in the list above use the Apache HTTP server in their Java EE products, adding functionality and support.)

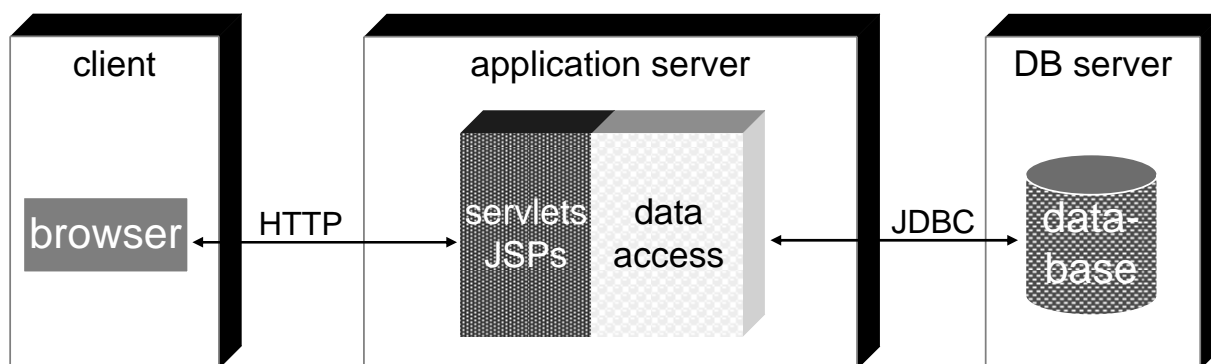
Multi-tiered Architecture

- ◆ Presentation logic, business logic, and persistent storage are separated into different **tiers**
 - The tiers may or may not be on physically separate machines
 - Though the database server is typically on its own machine
- ◆ Many enterprise applications will thus involve multiple physical servers and possibly multiple databases
- ◆ In the next slides, we show a number of possible configurations for using the various technologies
 - This is not an exhaustive presentation, but shows some of the common ways that the technologies are used, and the communications between the different technologies

Notes

Example: Web-Based Architecture

- ◆ This is a popular Java EE architecture for Web applications
- ◆ It is relatively simple, yet suitable for many types of applications

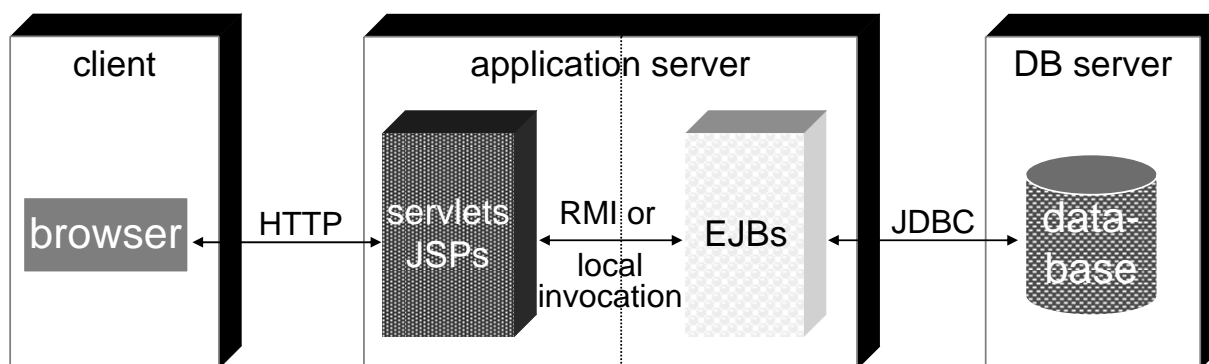


Notes

- ◆ The presentation code and business code reside in the same tier.
 - Database access can be implemented directly in the servlets/JSPs, but is more often encapsulated into a data access layer.
 - Many types of systems can be built with this relatively simple architecture.

Web-Based Architecture with EJB

- ◆ This extends the previous architecture by using EJBs for business logic and data access
- ◆ This is another fairly standard Java EE architecture used for larger systems

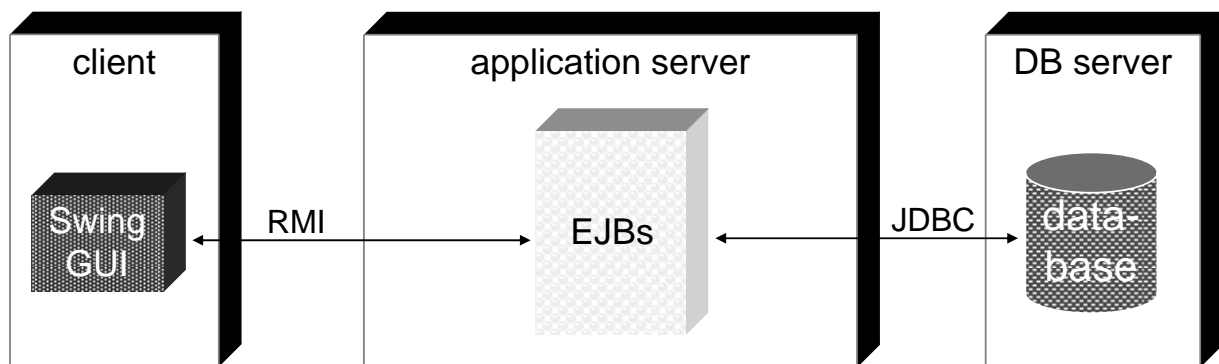


Notes

- ◆ You'll see this picture, or a variation of it, in almost every overview document about Java EE.
 - The difference between this and the previous solution is that the business code has been implemented in EJBs and exists in a different tier than the presentation code.
 - Larger, more scalable systems can be built with this architecture.
- ◆ Although EJBs are distributable and have remote method invocation (RMI) support built in, it is not required that they be invoked over a network via RMI.
 - The Web components and EJBs don't have to be on separate machines.
 - *Local* EJBs are resident in the same JVM as their client components and can be invoked via regular method calls. This reduces the overhead associated with RMI and allows for parameters and return values to be passed by reference instead of serialized and sent over a network.

Rich Client Architecture with EJB

- ◆ This architecture utilizes client processing power and allows for rich client GUIs

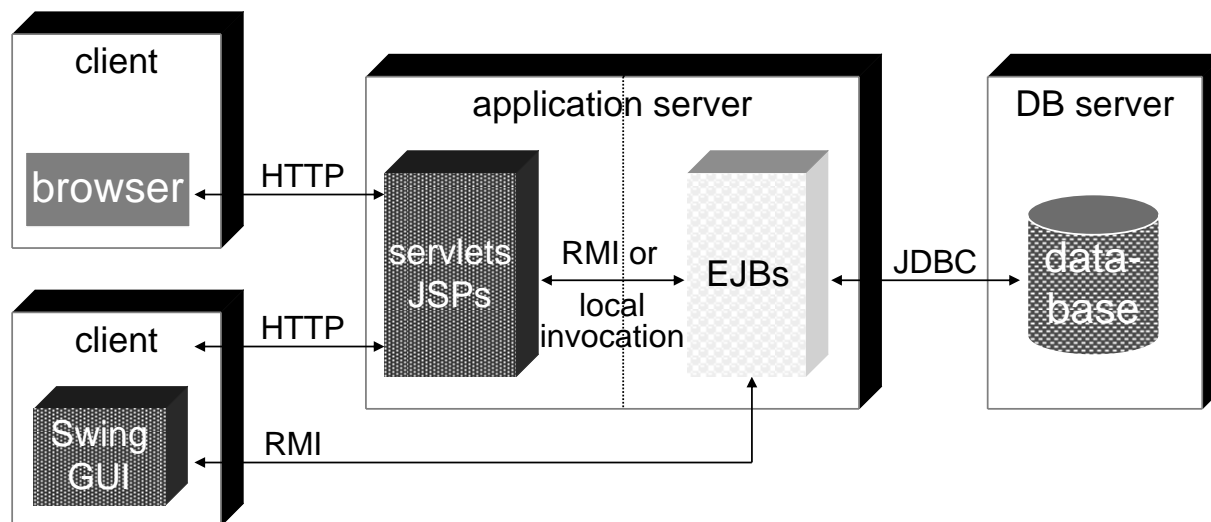


Notes

- ◆ Rich clients can interact directly with the business tier.

Supporting Both Web Clients and Rich Clients

- ◆ Web clients communicate via HTTP
- ◆ Rich clients can communicate via HTTP or RMI
 - This includes applets running in a browser

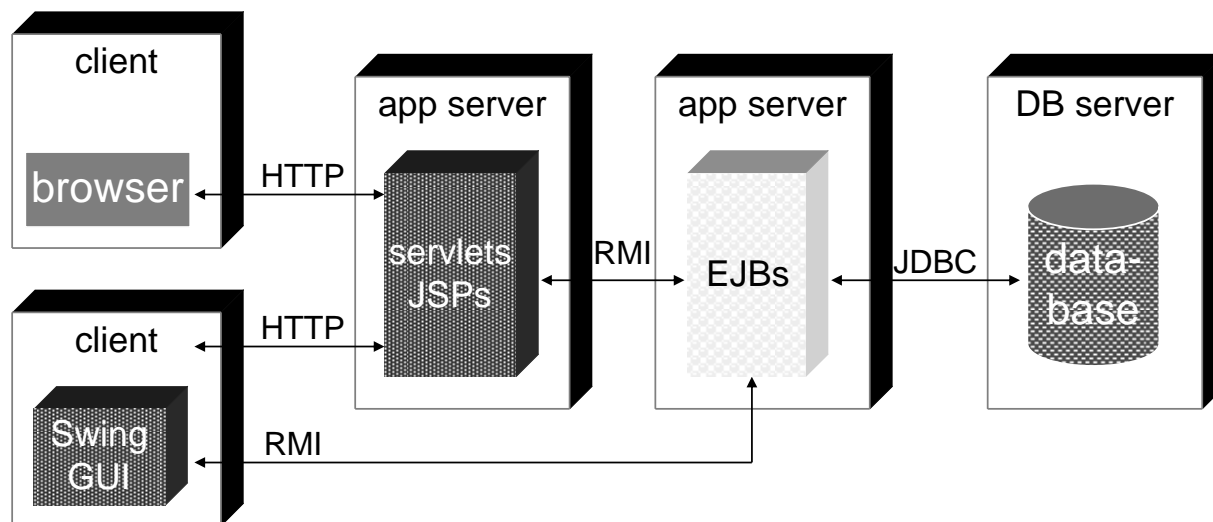


Notes

- ◆ This architecture may be attractive because you can support both Web browser clients and Swing clients, and do so in several different ways
- ◆ Web browser clients interact with the EJB business tier indirectly, via the servlet/JSP presentation tier
- ◆ Swing clients can be installed on a Java SE client machine or written as a downloadable applet that executes in the JVM of a Web browser
 - Swing clients can generate HTTP requests to the servlet/JSP presentation tier or can interact directly with the EJB business tier - You might want to use HTTP to reuse an existing servlet/JSP interface or to get through a firewall
 - Swing clients can also use HTTP as the network transport for RMI calls, which is another technique for tunneling through a firewall

Playing Server Games

- ◆ Logical tiers can be located on different physical machines



Notes

- ◆ Once you have split your application into logical tiers, it is possible to move them onto different physical machines.
 - This flexibility is one of the strengths of this architecture.
 - The need for this kind of architecture might also influence whether or not you choose to use EJB
- ◆ There is no single correct way to architect a system.
 - It all depends on requirements and available resources.

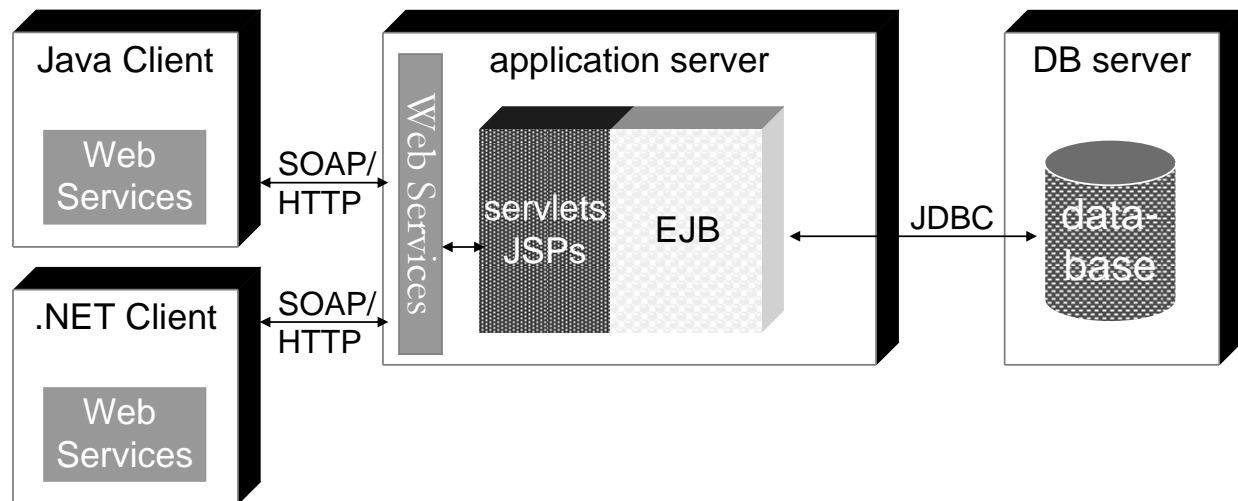
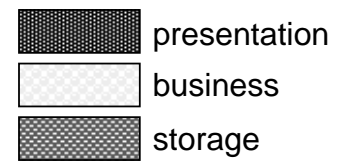
SOA – Service Oriented Architecture

- ◆ SOA is an important basis for enterprise architectures
- ◆ Based on providing resources on a network
 - As independent services independent of their implementation
 - Results in loosely coupled architecture
- ◆ Usually based on Web Services Technologies
 - Such as SOAP
 - Provide standard APIs and implementation independence
- ◆ When hosted on a Java EE server, these services are still implemented using standard Java EE technologies
 - Such as servlets or EJB

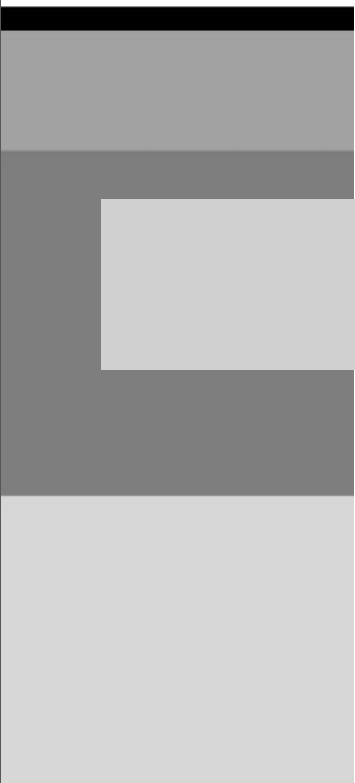

Notes

SOA Architecture

- ◆ Another popular Java EE architecture
- ◆ Provides loosely coupled access via standard protocols



Notes



JBoss AS Overview

- Java EE Overview
- JBoss AS Overview**
- Installing and Running JBoss AS
- More about Starting and Stopping JBoss AS
- Documentation

Notes

JBoss Application Server (JBoss AS)

- ◆ JBoss AS: Open source Java EE application server
 - Written in Java – so it's cross platform – just needs a VM
 - Version 6.x is a certified **Java EE 6** (Web profile only) server
 - Version 5.x is a certified **Java EE 5** server
 - Version 4.x is a certified **J2EE 1.4** server
 - 4.2+ also includes some Java EE 5 technologies – e.g. EJB3
- ◆ JBoss AS 5.x contains major changes from 4.x
 - Built on the **JBoss Microcontainer**
 - This is a **completely new kernel architecture** that replaces the previous JMX based microkernel architecture of JBoss AS 4.x
- ◆ JBoss **EAP** (Enterprise Application Platform) 5:
 - Bundles JBoss 5.1.0.GA with additional capabilities (see notes)
 - We'll cover relevant differences in the course

Notes

- ◆ JBoss AS runs on many platforms, including Windows, Linux and other *nix flavors, Mac OS, and many more
 - Basically wherever there is Java, you can run JBoss AS
- ◆ Though it is hard to be precise with app server rankings, JBoss AS is consistently in the top three - with IBM WebSphere and BEA WebLogic being the other two servers
 - And JBoss AS is free !
- ◆ JBoss has been very prompt on tracking new technology releases (though it lagged on the JBoss 5 AS release)
 - Large installed base in production usage, and generally considered reliable, and dependable
- ◆ The new JBoss Microcontainer was developed for over 3 years, and forms the basis for JBoss AS moving forward
 - It is also available as a standalone container for creating your own service platforms
- ◆ JBoss EAP bundles JBoss 5.1.0.GA with additional technologies, including:
 - JBoss Seam
 - RestEASY for Rest-based services
 - Enterprise Admin console

JBoss AS History and JBoss the Company

- ◆ JBoss AS started as an open source **EJB container**
 - Written mainly by a couple of developers
 - Eventually became a full fledged app server
 - Gained large market share due to its rapid adoption of new technologies and simplicity of use
- ◆ JBoss Inc, a for profit company, formed to support developers
 - Provides add-on services: **support, consulting, and training**
 - The app server remains open source
 - The company provides additional services in a professional, reliable way
 - Also sells some JBoss-based products that are not open source
- ◆ JBoss Inc was acquired by Red Hat, Inc. in 2005

Notes

- ◆ JBoss, Inc. called the model they used of a for-profit company coupled with an open source product "Professional Open Source"
 - It provides a high-quality open source platform, at no cost
 - It offers professional level services, such as support, at additional cost
 - This gives all the benefits of open source software (freely available source, active and open development community, etc) with the benefits of a professional company (quality support, viability on a long-term basis, etc)
- ◆ Here is JBoss' description of Professional Open Source:
 - We hire and pay experts in the open source community to write exceptional and innovative software full-time.
 - We only use open source licenses that are friendly to end-user IT shops, independent software vendors, and the community itself.
 - Directly and through our authorized partners, we deliver the best support services available; all of which are backed up by the real product experts.
 - Unlike first generation open source providers, we control the direction and source code for our projects. We can ensure that all bug fixes and patches are rolled into future versions of our products.

JBoss the Company

- ◆ Has become an important open source development group
 - Founded by Marc Fleury and Scott Stark in 2000
 - Brought developers on board from several major open source projects, including Hibernate, Tomcat, and Apache
 - Developers are paid - can devote quality time to development
 - Revenue generated from the services
 - Products remain open source
 - Allows "the best of both worlds" of open source and professional services
- ◆ Some people wonder if the products are still truly open source
 - They question the motivations and focus as the company has evolved into a standard "for profit" organization, driven by normal corporate profit needs

Notes

- ◆ An example of the "for profit" motivations can be seen in some of the services and products that are not open sourced, but only offered for sale
 - For example, some of the management products for the server (e.g. the "Operations Network" or ON) were only available as part of the support services
 - JBoss ON allows you to manage the server via a console, rather than through the XML configuration files that we will cover in this course
 - It's only recently that JBoss released an open source implementation of JON (Jopr)

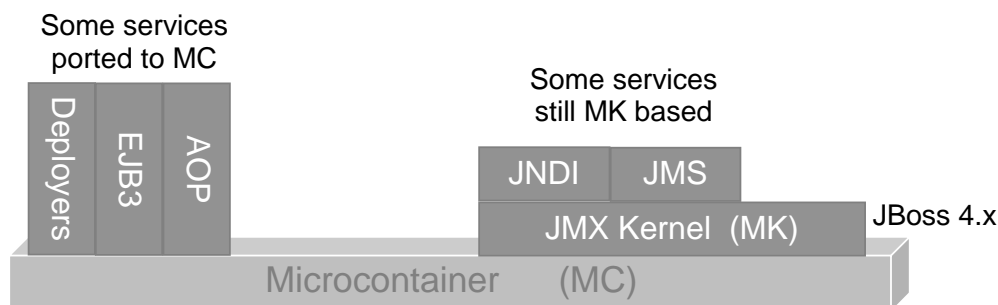
JBoss 5 Application Server

- ◆ **Major rewrite** of JBoss AS, with many core changes
 - Designed to be foundation for future JBoss AS versions
 - Lays groundwork for Java EE 6 features
 - New implementations of some key services - e.g. JMS
- ◆ **Java EE 5** - Full compliance
 - Complies with Java EE 5 conformance testing certification requirements
- ◆ **New JBoss Microcontainer** kernel architecture
 - Lightweight, POJO based, Dependency Injection (a la Spring)
 - Manages POJOs - their deployment, configuration, and lifecycle
 - Replaces JMX-based microkernel, supports JMX-based services
 - Can be used standalone to create your own server cores

Notes

Server Architecture - the Microcontainer

- ◆ JBoss is based on a small **microcontainer** (abbreviated **MC**)
 - Most services, including core server parts, are added into the MC
- ◆ **Microcontainer**: Dependency Injection (DI) based, providing:
 - Deployment and configuration of **POJO** based services
 - **Initialization** of objects via constructors and property setters
 - Specification of **dependencies** between objects
 - Uses **AOP** (Aspect Oriented Programming) heavily
 - Configuration usually specified via **XML** configuration files



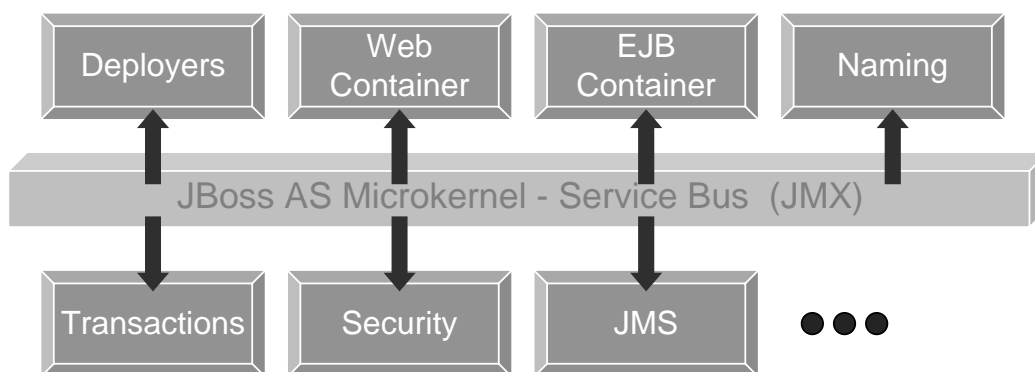
Notes

- ◆ The primary focus of the MC is as a framework for building server kernels. It replaces the legacy JMX based kernel found in JBoss AS 4.x and earlier with a POJO based kernel that has been generalized to better support extensibility along the primary aspects required for a server type of environment: AOP, metadata, class loading, deployments, state management, lifecycle/dependencies, configuration, and management. The use of the term "server" should not be correlated with large memory/CPU requirement environments of typical application/web servers. A "server" in the context of the MC is just a kernel for plugging POJO container models into. The MC can be configured for extremely lightweight application type setups like a Junit test case as well as full featured application servers.

[JBoss Microcontainer Reference Guide - Ch. 3]

Legacy Server Architecture - JMX Microkernel

- ◆ JBoss AS 4.x was built on a JMX-based microkernel
 - This includes a **bus** that other services can plug into (see below)
 - Not all services ported to MC, so JBoss 5 supports the legacy microkernel (abbreviated **MK**), that is run on top of the MC
- ◆ Managing JBoss AS 5.x requires knowing the MC **and** MK
 - Both contain services and associated configuration elements
 - Eventually all services will be migrated to the MC



Notes

- ◆ JMX (Java Management eXtensions) is a standard Java API for managing Java applications
 - JBoss extended the JMX concepts to use it as the core structure of the JBoss microkernel and app server
- ◆ The Microkernel is actually what is known as a JMX *MBeanServer*
 - We will cover JMX in a little more detail later on
- ◆ Migration of all services to the MC may not actually happen for a while – most likely in JBoss 6

Services Provided by the MC and MK

- ◆ **MC** is small, and capabilities include the following modules:
 - **kernel**: Bootstrap, POJO (bean) deployment, DI, naming
 - Services named by **bean names** configured in bean deployment
 - **dependency**: A controller managing service dependencies
- ◆ **MK** is also small, and capabilities include the following:
 - Principally **naming**, **lookup**, and **invocation** of **services**
 - Services named with special **JMX-based "object names"**
 - Service invocations done through service bus, and use the target's object name to specify what service is being invoked
- ◆ New services can be loaded dynamically (while server is up)
 - Services are decoupled, and support hot redeployment
 - Can easily add new services, or redeploy old ones
 - True of both the MC and MK architectures

Notes

- ◆ Theoretically, all services can be "hot deployed;" in practice, however, some services are so integrated into the server, that they can't be redeployed
 - For example, the Web container is actually integrated with JBoss AS as a deployable service
 - However, because it is so central to the operation of the server, you can't redeploy it without restarting the server

JBoss AS 5.x - Architecture



Notes

- ◆ The technologies listed above are based on JBoss AS 5.1

JBoss AS 5.x - Common Technologies

- ◆ **Web Container:** Based on Tomcat 6.0 - Supports Servlet 2.5/JSP 2.1 - integrates with Apache Web Server
 - Tomcat is very popular, administration familiar to Tomcat users
- ◆ **EJB:** Supports new EJB 3 specification, and Java Persistence API (based on Hibernate technology) (Java EE 5)
- ◆ **JAX-WS:** (Java API for XML based Web Services) - Full support for this new Web Services API (Java EE 5)
- ◆ **Flexible security** architecture - supports different security providers (Database based, LDAP, etc), and Java EE security
- ◆ **Clustering** for fail-over and scalability easily available
- ◆ Supports all standard Java EE services (JMS, JNDI, etc)

Notes

- ◆ The technologies listed above are based on JBoss AS 5.1



Installing and Running JBoss AS

Java EE Overview

JBoss Overview

Installing and Running JBoss AS

More about Starting and Stopping JBoss AS

Documentation

Notes

System Requirements

- ◆ JBoss AS 5.x requires **Java 5** or later
 - Java 5 is the recommended version (see notes here, next page)
 - Most operating systems that can run a JVM are supported - including Windows, Unix variants, MacOS, MVS, etc.
- ◆ JBoss runs on either 32 bit or 64 bit JVMs
 - Large applications (> 1.5GB RAM) can benefit from a 64-bit JVM
 - 64 bit JVMs support much larger memory/heap sizes, which improves the performance of large applications
- ◆ JBoss will also take advantage of multiple CPUs
- ◆ You must have adequate disk space to install JBoss
 - The initial installation may take 150+ MB of space
- ◆ The more memory (RAM) the better - we'll talk more about this

Notes

- ◆ JBoss AS is basically just a Java program, and it needs a working Java environment to run
 - JBoss AS can run under both Java 5 and Java 6
 - As of this writing, it is more thoroughly tested under Java 5
 - There are compiled binaries for both Java 5 and Java 6 - the Java 5 binary will run under both Java 5 and Java 6 (see *readme.html* in the JBoss distribution for info on using Java 6)
 -
- ◆ If 32 bit addressing supports the amount of memory your applications use, then it's very possible that they will run faster on a 32 bit JVM
 - Applications that run with more than 1.5GB of RAM (including free space for garbage collection optimization) should utilize the 64-bit JVM
 - 64-bit memory addressing gives virtually unlimited (1 exabyte) heap allocation.
 - However, large heaps affect garbage collection
 - 64 bit JVMs incur more overhead on each memory access
- ◆ The amount of disk space that JBoss takes to install can vary depending on how you install JBoss AS

Choosing Your JVM

- ◆ JBoss AS is very dependent on JVM performance
 - It uses the garbage collection and threading very heavily
 - In general, Sun VMs are most extensively tested with JBoss AS
 - Sun 1.5+ JVMs have very good performance - especially in the area of garbage collection
 - There are other JVMs (IBM and BEA have JVMs)
 - In general, there is no real reason to use these unless you have other reasons (e.g. you're running on IBM hardware/OS)
- ◆ JBoss 5.x **does NOT require** the full Java JDK
 - As of 4.2.0+, JBoss packages the Eclipse JDT compiler for compiling JSPs, so JAVA_HOME can point to a JRE
- ◆ If you have multiple CPUs, make sure your JVM supports this
 - Otherwise they will be no help

Notes

- ◆ From *readme.html* in the JBoss distribution : JBossAS 5.0.0.GA can be compiled with both Java5 & Java6. The Java5 compiled binary is our primary/recommended binary distribution. It has undergone rigorous testing and can run under both a Java 5 and a Java 6 runtime. When running under Java 6 you need to manually copy the following libraries from the JBOSS_HOME/client directory to the JBOSS_HOME/lib/endorsed directory, so that the JAX-WS 2.0 APIs supported by JBossWS are used:
 - jbossws-native-saaj.jar, jbossws-native-jaxrpc.jar, jbossws-native-jaxws.jar, jbossws-native-jaxws-ext.jar
- ◆ The other option is to download the jdk6 distribution (jboss-5.0.0.GA-jdk6.zip) in which case no configuration changes are required. If you still have problems using JBoss with a Sun Java 6 runtime, you may want to set -Dsun.lang.ClassLoader.allowArraySyntax=true, as described in JBAS-4491. Other potential problems under a Java 6 runtime include:
 - ORB getting prematurely destroyed when using Sun JDK 6 (see Sun Bug ID:6520484)
 - Unimplemented methods in Hibernate for JDK6 interfaces.
 - When JBossAS 5 is compiled with Java 6, support for the extended JDBC 4 API is included in the binary, however this can only be used under a Java 6 runtime. In this case no manual configuration steps are necessary. It should be noted however that the Java 6 compiled distribution of JBoss AS 5 is still in experimental stage.

Operating System Considerations

- ◆ Will run well on most modern versions of Windows and Unix variants
 - Your choice of JBoss AS will most likely not determine OS choice
 - Most groups choose an OS independently
- ◆ JBoss AS uses garbage collection and threading very heavily
 - How your OS handles threading and memory (heap) management will affect how well JBoss AS runs
 - Linux is also widely used because of its low cost, and modern versions have no trouble running JBoss AS
 - Windows is gaining wider acceptance in organizations that are already using it - it runs JBoss AS, but of course you are tied to the Windows platform on Intel hardware

Notes

- ◆ Older versions of Linux (pre 2.6) had problems with heavy threading usage
 - 2.6+ versions have a new Posix compliant threading implementation which is much better than older versions
 - In general, it's not a problem if you're using any fairly modern Linux distribution
- ◆ We will generally refer to all Unix or Unix-based variants as "Unix"
 - This is not strictly true, as you need to be certified as Unix compliant to actually use the name Unix
 - For our purposes though, the name Unix is clear enough

Environment Variables

- ◆ JBoss AS can be started from the command line, using scripts that come with the server
- ◆ The scripts use a number of environment variables, including:
 - **JAVA_HOME**: The location of your Java installation (JDK or JRE)
 - **JAVA_OPTS**: Options that will be passed to the JVM when the app server program is run - (these are optional)
- ◆ Unix variants use two additional environment variables
 - **JAVA**: The name of the JVM program (if different from *java*)
 - **MAX_FD**: The maximum number of file descriptors that will be made available by the OS
- ◆ In general, **JAVA_HOME should always be set**
 - This will ensure that you know what JVM is being used
 - You can also set the JBOSS_HOME environment variable *

Notes

- ◆ JBoss will attempt to start up even if JAVA_HOME is not set
 - It will simply use whatever JVM is on your path
 - It's safer to set JAVA_HOME to point to the JDK you want to use
- ◆ If you have more than one version of JVM installed in your machine, make sure you are using the JDK1.5 or JDK1.6 installation as the source for the java and javac executables

Installing JBoss AS

- ◆ Installing JBoss AS is very simple
 - Basically just setting up the needed files/directories
 - There are 3 methods you can use, as described below
- ◆ **Zip installation** - The standard download (a zip file)
 - Simply unzip the zip file to the desired location
 - This gives a default installation, with several server configurations
- ◆ **RPM installation** - Uses RPM Package Manager for automatic installation (Linux)
- ◆ **EAP Installers** – Enterprise Application Platform (EAP) is available with a fee-based support subscription
 - Provides more flexible/sophisticated installation management
 - Includes graphical installers, update/patch management, etc.

Notes

Server Configurations

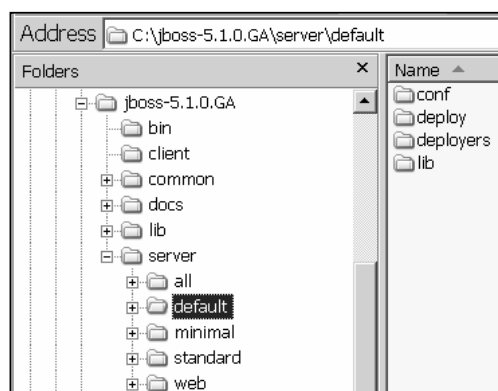
- ◆ JBoss AS includes a number of different server profiles
- ◆ The zip install creates **five** different server profiles
 - **all**, **default**, **minimal**, **standard**, and **web**
 - Located in subdirectories under the `<jboss>\server` directory
 - JBoss EAP provides an additional profile called **production**
- ◆ **all**: Starts all available services, including clustering and IIOP
- ◆ **default**: Base Java EE 5 config with default set of services
 - Most common configuration used - no JAXR, IIOP, or clustering
- ◆ **minimal**: Bare minimum of services needed to start JBoss
 - MC, MK, logging, JNDI, deployment - not a Java EE server
- ◆ **standard**: Java EE 5 certified config (call-by-value, isolated deployments, RMI-IIOP)
- ◆ **web**: Lightweight Web container, Java EE 6 Web profile based
- ◆ **production (EAP-only)**: Similar to all – tuned for production use

Notes

- ◆ The **minimal** configuration is not of much use by itself - it has no Java EE services deployed
 - It is, however, a very small compact core that you can add your own services to
 - For example, someone might want to use it as the core of an embedded controller, and add in their own services to perform whatever they need
 - ◆ The **default** config is close to a standard Java EE configuration - with some modifications to make it smaller and easier to use
 - ◆ The **standard** config is the configuration that has been tested for JavaEE compliance
 - It is new in JBoss 5.x
 - The major differences with the previously existing configurations (default, all) is that call-by-value and deployment isolation are enabled by default, along with support for RMI-IIOP and jUDDI (taken from the all config) [JBoss 5.0 release notes]
 - ◆ The **web** config is a new experimental lightweight configuration created around JBoss Web that will follow the developments of the JavaEE 6 web profile
 - Except for the servlet/JSP container it provides support for JTA/JCA and JPA. It also limits itself to allowing access to the server only through the HTTP port. Please note that this configuration is not JavaEE certified and will most likely change in the following releases.
- [JBoss 5.0 release notes]

Zip Installation

- ◆ Zip available at www.jboss.org with name like: **jboss-5.1.0.GA.zip**
- ◆ Unzip to location with no spaces in directory name (see notes)
 - Creates the server root directory - e.g. *jboss-5.1.0.GA*
 - For EAP the server root is similar to: *jboss-eap-5.1/jboss-as*
- ◆ The server root directory will have the structure shown below, with the following subdirectories
 - **bin**: Scripts to start/stop JBoss
 - **client**: Client side libraries (jars)
 - **common**: Shared components (jars)
 - new in JBoss 5
 - **docs**: Documentation and sample configuration files
 - **lib**: Microcontainer (boot) libraries
 - **server**: Set of configurations for starting JBoss AS



Notes

- ◆ Avoid installing JBoss into a directory hierarchy where there are spaces in the directory names
 - This may cause problems
- ◆ Common directories to install under are:
 - Windows: Root directory (C:\)
 - Unix: For class, you can unzip to your home directory
 - /usr/local is a common location if you're on a machine where you have permission
- ◆ The <jboss>\lib directory is for microcontainer jar files only
 - You should NOT put any of your own application jars in this directory
- ◆ The server structure has changed slightly from JBoss 4.x
 - The common\lib directory is new
 - It contains jar files shared between configurations (e.g. default and standard)
 - This minimizes the size of the overall installation by eliminating redundant jars in each configuration

Graphical Installer

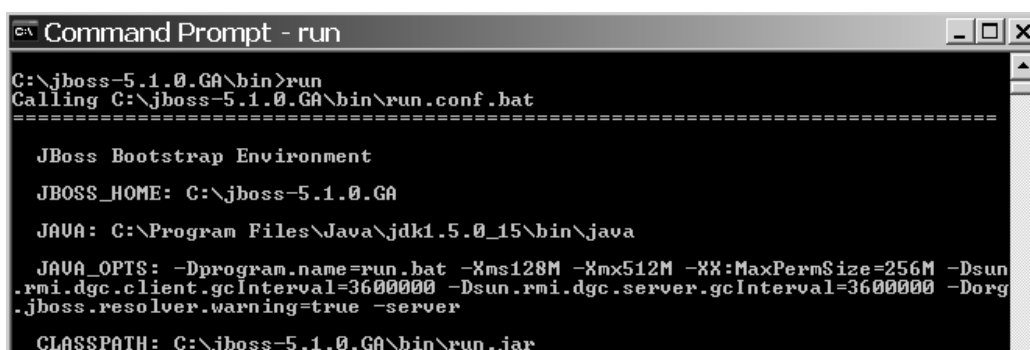
- ◆ There is a graphical installer available as part of the JBoss Enterprise Application Platform (EAP)
 - The installer allows configuration of things like Installation location, Security (e.g. JMX console security), and more
 - EAP is a for-sale supported bundle of a number of different products from JBoss
- ◆ Previously, an open source project, the JEMS installer also provided this capability
 - The latest release (from early 2008) supports JBoss AS 4.2.2
 - This project no longer appears to be actively supported

Notes

- ◆ The installer will provide graphical windows that includes configuration of:
 - **Installation location** (best not to install to a directory with spaces in its name)
 - **JMX Security** - There are various access points to JBoss AS that should be secured, we'll talk about these later
 - **Shortcuts** on the desktop for starting/stopping JBoss AS

Starting JBoss AS

- ◆ To start JBoss AS, you simply run a supplied script - in `<jboss>\bin`
 - After setting any necessary environment variables
- ◆ ***run.bat*** for Windows, ***run.sh*** for Unix variants *
- ◆ By default, the script will start the ***default*** server configuration
 - Use the `-c` option to pass in a different configuration to run - e.g.
 Windows: ***run -c minimal***
 Unix: ***./run.sh -c minimal***
 - These scripts first check the environment, then print out important system properties, as shown below (Windows example shown)



```

C:\jboss-5.1.0.GA\bin>run
Calling C:\jboss-5.1.0.GA\bin\run.conf.bat
=====
JBoss Bootstrap Environment
JBOSS_HOME: C:\jboss-5.1.0.GA
JAVA: C:\Program Files\Java\jdk1.5.0_15\bin\java
JAVA_OPTS: -Dprogram.name=run.bat -Xms128M -Xmx512M -XX:MaxPermSize=256M -Dsun
.rmi.dgc.client.gcInterval=36000000 -Dsun.rmi.dgc.server.gcInterval=36000000 -Dorg
.jboss.resolver.warning=true -server
CLASSPATH: C:\jboss-5.1.0.GA\bin\run.jar
  
```

Notes

- ◆ When running in a command prompt for Unix, you'll likely need to invoke the script as `./run.sh`
 - Unix installations usually do not have dot (.) in the path
- ◆ If you want to start JBoss from other than the `<jboss>\bin` directory, then you can set the `JBOSS_HOME` environment variable to point to your JBoss installation
 - You can also place `<jboss>\bin` on your path
 - With this configuration, you can start JBoss from anyplace on your system
- ◆ Boot time is somewhat slower when compared to the 4.2.x series, due to the extensive annotation scanning that goes on behind the scenes
 - There are ways to optimize the annotation scanning, especially for large deployments
 - Check the JBoss Wiki for more information.

[JBoss 5 release notes]
- ◆ Note that for JBoss EAP 5, the default profile will be started if no `-c` argument is given to run
 - This is different from JBoss EAP 4, which started the production profile if no argument was given to run

Starting JBoss AS - the Microcontainer

- ◆ The microcontainer is then booted, and core services are started (such as logging, a virtual file system, etc.)
 - These services are deployed statically from an XML configuration file (*conf\bootstrap.xml*) that we'll cover later
 - Info on the boot process is output to the console, as shown below

```

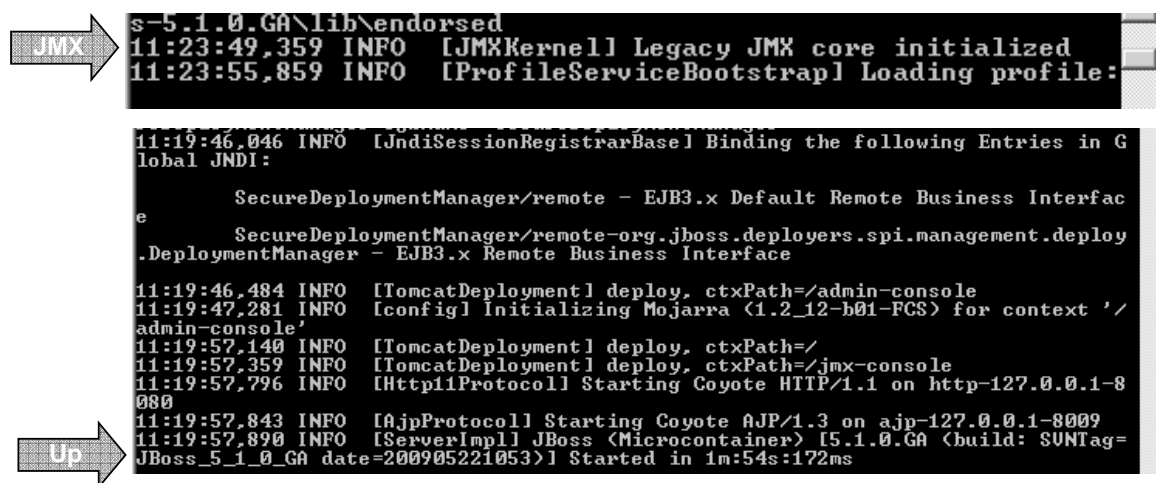
=====
11:23:41.625 INFO [ServerImpl] Starting JBoss <Microcontainer>...
11:23:41.625 INFO [ServerImpl] Release ID: JBoss [The Oracle] 5.1.0.GA <build: SUNTag=JBoss_5_1_0_GA
11:23:41.625 INFO [ServerImpl] Bootstrap URL: null
11:23:41.625 INFO [ServerImpl] Home Dir: C:\jboss-5.1.0.GA
11:23:41.625 INFO [ServerImpl] Home URL: file:/C:/jboss-5.1.0.GA/
11:23:41.625 INFO [ServerImpl] Library URL: file:/C:/jboss-5.1.0.GA/lib/
11:23:41.625 INFO [ServerImpl] Patch URL: null
11:23:41.625 INFO [ServerImpl] Common Base URL: file:/C:/jboss-5.1.0.GA/common/
11:23:41.625 INFO [ServerImpl] Common Library URL: file:/C:/jboss-5.1.0.GA/common/lib/
11:23:41.625 INFO [ServerImpl] Server Name: default
11:23:41.625 INFO [ServerImpl] Server Base Dir: C:\jboss-5.1.0.GA\server
11:23:41.625 INFO [ServerImpl] Server Base URL: file:/C:/jboss-5.1.0.GA/server/
11:23:41.625 INFO [ServerImpl] Server Config URL: file:/C:/jboss-5.1.0.GA/server/default/conf/
11:23:41.625 INFO [ServerImpl] Server Home Dir: C:\jboss-5.1.0.GA\server\default
11:23:41.625 INFO [ServerImpl] Server Home URL: file:/C:/jboss-5.1.0.GA/server/default/
11:23:41.640 INFO [ServerImpl] Server Data Dir: C:\jboss-5.1.0.GA\server\default\data
11:23:41.640 INFO [ServerImpl] Server Library URL: file:/C:/jboss-5.1.0.GA/server/default/lib/
11:23:41.640 INFO [ServerImpl] Server Log Dir: C:\jboss-5.1.0.GA\server\default\log
11:23:41.640 INFO [ServerImpl] Server Native Dir: C:\jboss-5.1.0.GA\server\default\tmp\native
11:23:41.640 INFO [ServerImpl] Server Temp Dir: C:\jboss-5.1.0.GA\server\default\tmp
11:23:41.640 INFO [ServerImpl] Server Temp Deploy Dir: C:\jboss-5.1.0.GA\server\default\tmp\deploy
11:23:42.921 INFO [ServerImpl] Starting Microcontainer. bootstrapURL=file:/C:/jboss-5.1.0.GA/server/
ml
11:23:44.875 INFO [UFSCacheFactory] Initializing UFSCache [org.jboss.virtual.plugins.cache.CombinedU
11:23:44.875 INFO [UFSCacheFactory] Using UFSCache [CombinedUFSCache[real-cache: null]]

```

Notes

Starting JBoss AS - the Microkernel

- ◆ The legacy microkernel is added into the MC
 - MK services are also deployed statically - from *conf/jboss-service.xml*
- ◆ Other services are deployed, and eventually the server has booted
 - You can browse to ***http://localhost:8080*** to check if it is up
 - This shows the default web page for the Web container

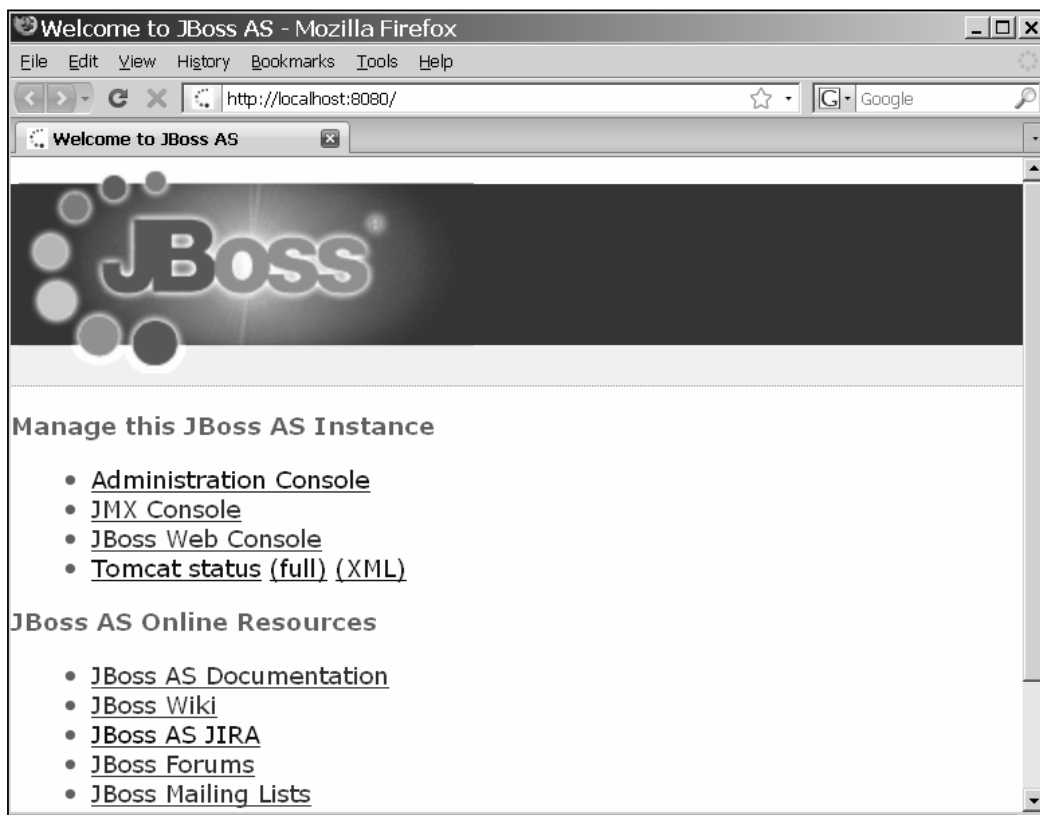


```

s-5.1.0.GA\lib\endorsed
11:23:49,359 INFO    [JMXKernel] Legacy JMX core initialized
11:23:55,859 INFO    [ProfileServiceBootstrap] Loading profile:
11:19:46,046 INFO    [JndiSessionRegistrarBase] Binding the following Entries in Global JNDI:
        SecureDeploymentManager/remote - EJB3.x Default Remote Business Interface
        SecureDeploymentManager/remote-org.jboss.deployers.spi.management.deploy
        .DeploymentManager - EJB3.x Remote Business Interface
11:19:46,484 INFO    [TomcatDeployment] deploy, ctxPath=/admin-console
11:19:47,281 INFO    [config] Initializing Mojarra (1.2_12-b01-FCS) for context '/admin-console'
11:19:57,140 INFO    [TomcatDeployment] deploy, ctxPath=/
11:19:57,359 INFO    [TomcatDeployment] deploy, ctxPath=/jmx-console
11:19:57,796 INFO    [Http11Protocol] Starting Coyote HTTP/1.1 on http-127.0.0.1-8080
11:19:57,843 INFO    [AjpProtocol] Starting Coyote AJP/1.3 on ajp-127.0.0.1-8009
11:19:57,890 INFO    [ServerImpl] JBoss (Microcontainer) [5.1.0.GA (build: SUNTag=JBoss_5_1_0_GA date=200905221053)] Started in 1m:54s:172ms
  
```

Notes

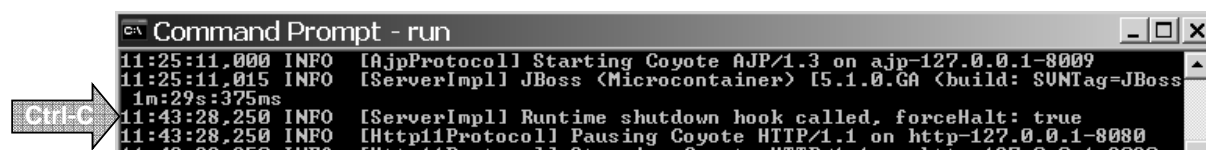
The Server Is UP !



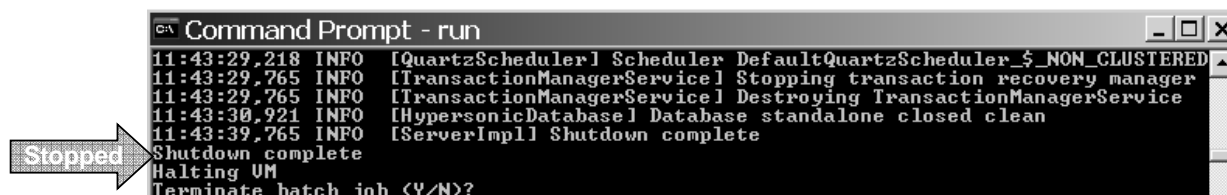
Notes

Stopping JBoss

- ◆ Stopping JBoss is easy - one way is to press Ctrl-C * in the console window - which will shut it down in an orderly manner
 - JBoss catches the Ctrl-C, and does a normal shutdown
- ◆ You can also use *shutdown.bat/sh* in the bin directory *
 - You can also shut it down from the Web console, covered later



```
CA Command Prompt - run
11:25:11,000 INFO [AjpProtocol] Starting Coyote AJP/1.3 on ajp-127.0.0.1-8009
11:25:11,015 INFO [ServerImpl] JBoss (Microcontainer) 15.1.0.GA (build: SUNTag=JBoss
1m:29s:375ms [ServerImpl] Runtime shutdown hook called, forceHalt: true
11:43:28,250 INFO [Http11Protocol] Pausing Coyote HTTP/1.1 on http-127.0.0.1-8080
11:43:28,250 INFO [Http11Protocol] Stopping Coyote HTTP/1.1 on http-127.0.0.1-8080
```



```
CA Command Prompt - run
11:43:29,218 INFO [QuartzScheduler] Scheduler DefaultQuartzScheduler_$_NON_CLUSTERED
11:43:29,765 INFO [TransactionManagerService] Stopping transaction recovery manager
11:43:29,765 INFO [TransactionManagerService] Destroying TransactionManagerService
11:43:30,921 INFO [HypersonicDatabase] Database standalone closed clean
11:43:39,765 INFO [ServerImpl] Shutdown complete
Shutdown complete
Halting VM
Terminate batch job (Y/N)?
```

Notes

- ◆ Ctrl-C means pressing the control key and the C key at the same time
- ◆ The shutdown script *shutdown.sh* should be used for Unix systems



Lab 1.1 – Setting Up JBoss AS

Notes

Lab 1.1 – Setting Up JBoss AS



- ◆ **Overview:** In this lab, we will install JBoss AS (if needed) and test it to make sure it is running well
 - We will also become familiar with the lab structure
- ◆ **Objectives:**
 - Install and run JBoss AS
 - Set up our environment, and become familiar with the lab structure
- ◆ **Builds on previous labs:** None
- ◆ **Approximate Time:** 20-30 minutes

Notes

Information Content and Task Content



- ◆ Within a lab, information only content is presented in the normal way – the same as in the student manual pages
 - Like these bullets at the top of the page
- ◆ Tasks that the student needs to perform are in a box with a slightly different look – to help you identify them
- ◆ An example appears below

Tasks to Perform

- ◆ Look at these instructions, and notice the different look of the box as compared to that above
 - Make a note of how it looks, as future labs will use this format
- ◆ OK – Now **get out your setup files**; we're ready to start working

Notes

Working with MS Windows or *nix



Tasks to Perform

- ◆ We support both Windows and *nix (Linux, other Unix variants) in the course
- ◆ You'll see instructions for both, often differentiated inline, e.g.
 - Windows: ***somewindowsCommand***
 - *nix: ***someUnixCommand***
- ◆ Sometimes we'll use the notes, e.g.
 - Run the following for Windows (see notes for *nix)
somewindowsCommand
- ◆ Sometimes we'll have separate slides for each
 - Which will be clearly marked in the title of the slide
- ◆ Bottom Line: Pay attention to make sure you are looking at the correct instructions for your system

Notes



Java EE Security Overview

Java EE Security Overview

JBoss AS Security

Encrypting Datasource Passwords

TLS/SSL and HTTPS

Securing JBoss Services

Notes

Security Requirements

- ◆ There are a number of key aspects that are important when dealing with the security of Java EE applications
- ◆ **Authentication:** Identifying a user
 - There must be a way of reliably identifying a user
 - For example with a name and password
- ◆ **Authorization:** Granting of access rights
 - Different users need to have differing levels of access to resources
- ◆ **Data Integrity:** Information must be protected from tampering
 - Generally done by encryption of some sort, e.g. SSL
- ◆ **Data Privacy:** Information must be protected from unauthorized access
 - Also generally done using encryption

Notes

Java EE Security in JBoss AS

- ◆ **Authentication:** Support for various mechanisms to authenticate the identity of a user
 - Web: HTTP Basic Authentication, Form Based Authentication, Digest Authentication, Client Certificate Authentication
 - Java Client: JAAS
- ◆ **Authorization:** Role-based access model to resources
 - Users mapped to roles
 - Access to resources configured in deployment descriptor (DD) in terms of roles
- ◆ **Data Integrity and Confidentiality:** Supports the use of SSL to protect data
 - Configured in DD for Web resources
 - Configured at invoker level for EJB access from remote clients

Notes

Transport Level Security with HTTPS/SSL

- ◆ Provides data encryption, server authentication, message integrity using Secure Sockets Layer (SSL) for TCP/IP
- ◆ Uses a combination Public Key Encryption (PKE) for an initial handshake, and symmetric encryption for transferring data
 - In PKE, two keys exist a **public key** and a **private key**
 - When, for example, a browser tries to open an SSL connection to a server, the server sends a certificate with its public key
 - The browser encrypts data for the server with the server's public key, and the server decrypts it with the private key
 - Very difficult to decrypt without knowing the private key
- ◆ Public keys are shared via certificates that are verified (signed) by some trusted authority (e.g. VeriSign)
 - Usually using an X.509 based certificate

Notes

- ◆ HTTPS also can optionally require a client certificate
- ◆ PKE is used for secure communications to exchange information to set up session keys that are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity
 - Symmetric key encryption is much faster than PKE
 - However, PKE provides better authentication techniques, which is why it is used for the initial handshake
- ◆ There are many more details to SSL, PKE, and symmetric key encryption, but most of them are not pertinent to what you need to do to manage the server

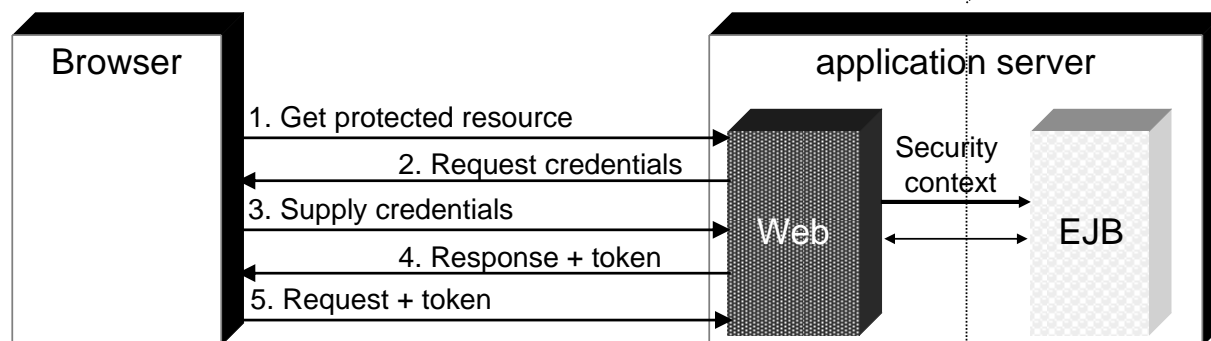
Java EE Security Overview

- ◆ Integrates application security into the Java EE architecture and programming model
 - Security policies can be set in deployment descriptors (configuration) rather than being hard-coded at development time (source code)
 - Works with existing security systems
- ◆ When clients use a resource, they authenticate to the server
 - By one of the methods described earlier - e.g. HTTP Basic
 - Or may be assigned a default identity, typically "guest" or some variant
- ◆ The container keeps track of the client identity
 - Identity is stored "container-wide" so that once you are logged in, your identity is known to all apps in the container
 - The security identity (or principal) is propagated to other resources (e.g. from Web tier to EJB calls)
 - Re-authentication required only when security policy boundary crossed

Notes

Java EE Security Example

- ◆ In a typical Java EE application, a Web application from an unauthenticated user will request a protected resource (1)
 - The Web container will prompt the user for security credentials (e.g. name/password) through one of the standard methods (2)
 - The user supplies credentials (3), is authenticated in the Web tier, and a security context is associated with the user
 - The Web container may make requests of other resources, and will propagate the security context if it does
 - The Web tier responds (4) and returns a token (e.g. session cookie), which the browser includes on subsequent requests (5)



Notes

- ◆ Web tier security provides a number of ways for the client to authenticate
 - Basic, form-based, digest, etc.

Java EE Declarative Security

- ◆ Java EE security is **declarative** and **role based**
 - Specified in the deployment descriptor using **roles**, **access control** and **integrity/confidentiality** constraints
 - Defines a **logical** security structure - i.e., roles are logical names mapped to physical users and groups
- ◆ A **security role** is a **grouping of users** for the purpose of enforcing a common security policy
 - For example, the manager role vs. the customer role
 - A user may serve in more than one role
 - Role names (e.g., “Customer”) are specified and used within the standard deployment descriptors to set the basic security policy
- ◆ The logical structure in the deployment descriptor is **mapped** to an app server's specific security policy
 - The security policy is **enforced by the Java EE container** at runtime

Notes

- ◆ Role-based security is not unique to Java EE. It is used in various security systems throughout the industry.
 - Security is specified in terms of user roles, rather than programmed into your components
 - It is enforced by the container
- ◆ The J2EE specifications note that it is important to keep in mind that the security roles in the deployment descriptor are used to define the logical security view of an application. Roles defined in the J2EE deployment descriptors should not be confused with the user groups, users, principals, and other concepts that exist in the target enterprise's operational environment. The deployment descriptor roles are application constructs with application domain-specific names. For example, a banking application might use role names such as BankManager, Teller, or Customer. [JBoss AS Server Configuration Guide 8.1.3]

JMX Console - Security Role Declaration

- ◆ **Role declaration** in the *web.xml* for JMX Console (below)
 - The fragments from the next few slides are taken from the actual JMX Console deployment descriptor (DD)

```
<!-- Much detail omitted throughout this file -->
<web-app>
  <description>JMX console html adaptor</description>
  <servlet>
    <servlet-name>HtmlAdaptor</servlet-name>
    ...
  </servlet>

  <!-- Security role is used in a security constraint -->
  <security-role>
    <role-name>JBossAdmin</role-name>
  </security-role>

  <!-- Continued in later slide -->
```

Notes

Specifying Security Constraints

- ◆ **Security constraints** specify the protection of Web resources in the Web app DD
- ◆ A security constraint can include the following:
 - **Web resource collection:** Set of URL patterns / HTTP methods
 - Describes a set of resources to be protected
 - **Authorization constraint:** Set of security roles
 - Users must belong to at least one listed role to access resources in the web resource collection
 - Users not in any of the roles are denied access
 - **User data constraint:** Requirements for the transport layer
 - **Integrity** - preventing data tampering while in transit
 - **Confidentiality** - preventing reading while in transit
 - The container must use SSL (HTTPS) for resources marked *INTEGRAL* or *CONFIDENTIAL*

Notes

Security Constraints - Deployment Descriptor

```
<!-- Continued from earlier slide -->
<!-- Uncomment the WEB-INF/jboss-web.xml/security-domain element
to enable secured access to the HTML JMX console. -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>HtmlAdaptor</web-resource-name>
    <description>An example security config that only allows
      users with the role JBossAdmin to access the HTML JMX
      console web application
    </description>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
<!-- Continued on next slide -->
```

- ◆ **Important Note:** The sample in *web.xml* contains the elements below, and is a major security leak - Don't include them (see notes)

```
<http-method>GET</http-method>
<http-method>POST</http-method>
```

Notes

- ◆ For more on the security issue mentioned in the slide, see:
<http://kbase.redhat.com/faq/docs/DOC-30741>
- ◆ We summarize it below:
- ◆ The default setup of the JMX console as shipped with these products defines the following security constraints in *deploy/jmx-console.war/WEB-INF/web.xml*:

```
<!-- Most detail omitted -->
<url-pattern>/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
```
- ◆ This means that any GET or POST requests without proper authentication to the JBossAdmin realm are blocked and get a 401 error.
- ◆ The HTTP protocol lists other verbs besides GET and POST, for instance HEAD, PUT and DELETE. As those are not listed in the security constraint, requests with these verbs are still allowed to be directed to the server without a security check and are executed by the default GET handler if no verb specific handler is specified.
- ◆ To manually prevent the remaining allowed verbs to trigger the GET handler the security constraint needs to be adjusted by removing the explicit denial of GET and POST which blocks all verbs by default
 - This is shown in the slide

Security Constraints - Deployment Descriptor

```
<!-- Continued from previous slide -->



<!-- Only users in JBossAdmin role can access jmx-console -->
<auth-constraint>
  <role-name>JBossAdmin</role-name>
</auth-constraint>

<!-- Keep data private (via HTTPS/SSL) -->
<user-data-constraint>
  <transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>

</security-constraint>

</web-app>
```

Notes



JBoss AS Security

- Java EE Security Overview
- JBoss AS Security**
- Encrypting Datasource Passwords
- TLS/SSL and HTTPS
- Securing JBoss Services

Notes

Overview of Security in JBoss AS

- ◆ Many areas in JBoss AS itself where you need to be concerned about security; these include
 - **Applications:** Especially administration points like the JMX/Web/Admin consoles
 - **Registries:** This includes stored user information (e.g. user/password info in a DB or file), as well as login information in the JBoss configuration files (e.g. a DB password for a DataSource)
 - **Physical Access Points:** Wire level access
 - **Web Tier:** Various security points such as HTTPS and configuration of the Web server
 - **Invokers:** Multiple access points to the JBoss server itself, as well as services running in it
- ◆ We'll first take a look at the general JBoss security model, then look at how it's used to secure these various areas

Notes

- ◆ One of the problems with the default JBoss installation (as created using the zip installation) is that it is mostly unsecured
 - Many management and access points are totally unsecured in the default installation created from the zip
 - You can configure security for these later, but you have to know about them, and remember to secure them
- ◆ The graphical installer, available as part of the for-sale EAP platform, allows you to configure security for these sensitive services while you are doing the installation

JBoss Security Mechanism Overview

- ◆ JBoss Security (**JBossSX**) lets you flexibly map logical security roles (e.g. from Java EE) to your environment
 - Provides pluggable **login modules** for user authentication and mapping users to roles
 - Allows easy integration with different security architectures (e.g. LDAP or DB-based)
 - Built on top of JAAS (a standard Java security API) *
- ◆ Uses **security domains** that are configured via XML, with login module(s) for authentication
 - This WIKI has a lot of info:
<http://www.jboss.org/community/wiki/JBossSX>
- ◆ Applications declare security by naming a security domain
 - Usually in the JBoss specific deployment descriptor (e.g. *jboss.xml* for EJB, or *jboss-web.xml* for Web apps)

Notes

- ◆ JAAS (Java Authentication and Authorization Service)
 - Standard Java API that allows you to authenticate users in a pluggable way (via Pluggable Authentication Modules)
 - JAAS is a fairly complex API, and the JBossSX framework that uses it has quite a bit of complexity
 - Most of this is transparent to the day-to-day administration of the server
 - You would only need to deal with the low-level details if you wanted to modify the way JBoss provided security

Security Domains and Login Modules

- ◆ In JBoss, **security domains** abstract the underlying mechanisms to authenticate and authorize users
 - Basically they're **collections of users** with authentication policies
 - Each security domain defines a stack of **login modules** to authenticate and authorize users
- ◆ A **login module** specifies an **authentication policy**
 - Usually accessing a data store (e.g. DB) holding user login/credential (e.g. name/password) and user/role associations
 - You configure a login module to access the data store, for example:
 - Connection info for an LDAP server, object names that hold needed data
 - Datasource lookup name for a relational database, SQL queries
 - File names/locations for a simple file
 - By default, login modules are configured in ***conf/login-config.xml***
 - The modules are loaded statically - once at server startup

Notes

- ◆ Login modules are part of the pluggable architecture of JBossSX
 - They allow you to plug in different security policies
 - They are JAAS based in their implementation, but that is really not important from the point of view of daily management of the server
 - You mostly work with the *login-config.xml* file, and usually don't need to care how it is implemented

JBoss Login Modules

- ◆ JBoss provides a number of different login modules, including:
 - **UsersRolesLoginModule**: Simple file-based login module supporting multiple users/roles
 - Very convenient for testing of security mechanisms
 - **DatabaseServerLoginModule**: JDBC (Relational DB) based login module
 - **LdapLoginModule**: LDAP server based login module
 - **IdentityLoginModule**: Simple login module that associates a hard-coded user name to any subject it authenticates
 - **BaseCertLoginModule**: Authenticates via X509Certificates
 - **CertRolesLoginModule** & **DatabaseCertLoginModule** extend the behavior to obtain authorization roles from a properties file or DB
 - **Custom login module**: JBoss provides support for writing your own login modules - requires Java programming

Notes

- ◆ The *BaseCertLoginModule* only does authentication
 - It needs to be combined with another module that acquires the authorization roles for the user
 - *CertRolesLoginModule* & *DatabaseCertLoginModule* do just that (via properties files like *UsersRolesLoginModule*, and database info respectively)
- ◆ Each of these login modules is implemented by a different JBoss class
 - You configure the application policy by specifying one of these classes, along with configuration information (e.g. the names of the files for the *UsersRolesLoginModule*)

UsersRolesLoginModule - File Based

- ◆ **UsersRolesLoginModule** configures security via properties files
 - **Username-to-password** mapping file: Contains users and passwords in **username=password** format
 - **Username-to-roles** mapping file: Contains users and roles in **username=role[, role]** format
 - Usually used for testing, and not production
- ◆ The files can appear in the server config's conf directory, or on the classpath of the application using the module *
- ◆ The files below (from the jmx-console domain) define one user
 - The user name is **admin**, and the password is **admin**
 - **admin** is in the **JBossAdmin** and **HttpInvoker** roles

```
# jmx-console-users.properties: Username=password  
admin=admin
```

```
# jmx-console-roles.properties: Username=role[, role]  
admin=JBossAdmin,HttpInvoker
```

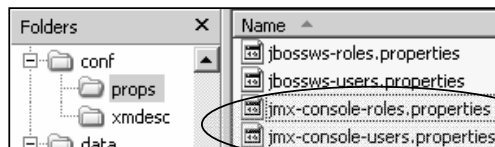
Notes

- ◆ The properties files can be in any of the following places:
 - The conf directory
 - Any directory on the JBoss AS server or system classpath
 - Within the deployment archive
 - e.g. Under the classes directory of a WAR file

UsersRolesLoginModule Example

- ◆ A security domain using this module includes:
 - **<application-policy>**: Root element, **name** attribute specifies the security domain name (bound in JNDI as **java:/jaas/domain-name**)
 - **<login-module>**: Specifies how security information is accessed
 - **code** attribute specifies the classname of the login module implementation
 - **flag="required"** specifies that the login module must succeed
 - **<module-option>**: Configuration information for the module (in this case, the names of the files for the *UsersRolesLoginModule*)

```
<application-policy name="jmx-console"> <!-- From login-config.xml -->
  <authentication>
    <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
      flag="required">
      <module-option name="usersProperties">
        props/jmx-console-users.properties
      </module-option>
      <module-option name="rolesProperties">
        props/jmx-console-roles.properties
      </module-option>
    </login-module>
  </authentication>
</application-policy>
```



Notes

- ◆ The *jmx-console* security domain shown in the slide comes preconfigured in the *login-config.xml* file for the default, standard, web, and all configurations
- ◆ The login module's **flag** attribute controls the overall behavior of the authentication stack
 - **flag=required** means the login module is required to succeed for the authentication to be successful.
 - Other choices are requisite, sufficient, optional
 - You can have multiple login modules within a security domain
 - They may use other choices for the flag attribute
- ◆ Each login module has its own set of configuration options
 - These are set as name/value pairs using the module-option elements.

Specifying Security Domains

- ◆ **Specifying a security domain** for an application ties it into the JBoss security system
 - Done in ***jboss-web.xml*** for a Web app (shown below - this is a JBoss specific deployment descriptor)
 - ***<security-domain>*** specifies the JNDI name of the security domain holding the authentication/authorization information
 - Note: the name starts with ***java:/jaas***, followed by the name for the domain that is configured in *login-config.xml* (jmx-console)
- ◆ EJBs support a similar configuration in ***jboss.xml***
 - *jboss.xml* is the JBoss DD for EJB

```
<jboss-web>
  <!-- Security-domain to enable security. You will need to edit
        the jmx-console login configuration to setup the login
        modules used to authentication users. -->
  <security-domain>java:/jaas/jmx-console</security-domain>
</jboss-web>
```

Notes

- ◆ JNDI is used by JBoss to make the security domains configured in *login-config.xml* available to applications that need it
 - The ***java:/jaas*** prefix is the standard location that JBoss uses for the security domain bindings
- ◆ To secure an EJB, you would use a ***<security-domain>*** element in the *jboss.xml* file, as a child of the top level ***<jboss>*** element

Encrypting User Passwords

- ◆ Login modules can be configured to use hashed passwords *
 - A hash function is used to encrypt passwords
 - They're stored in encrypted form in the data store (e.g. the DB)
 - To authenticate, the password from the user is hashed and compared with the stored hash – no clear text password stored
- ◆ The login module properties for hashing include:
 - ***hashAlgorithm***: *java.security.MessageDigest* algorithm to use to hash the password. Typical values are MD5 and SHA.
 - ***hashEncoding***: The string format for the hashed pass - must be either base64, hex or rfc2617. The default is base64.

Notes

- ◆ A hash function takes a string of any length as input and produces a fixed length string as output, sometimes termed a message digest or a digital fingerprint
 - In general, it is very difficult to compute the clear text password from the stored hash (this is the basis of cryptographic hashing)
 - Because of this, the hashed version of the password is safe to store
 - The user still has to present their original password, (which is then hashed) so security is maintained
- ◆ Additional login module properties for hashing include:
 - ***hashCharset***: The encoding used to convert the clear text password to a byte array. The platform default encoding is the default.
 - ***hashStorePassword***: This indicates that the hashing algorithm should be applied to the password stored on the server side. This is used for digest authentication - see the Server Configuration Guide for more info

Configuring Password Hashing

- ◆ Below, the `jmx-console` security domain is shown configured for hashed passwords
 - Note the options for setting up hashing
 - The passwords in the `jmx-console-user.properties` file would now need to be stored in hashed form

```
<application-policy name="jmx-console">
  <authentication>
    <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
      flag="required">
      <module-option name="usersProperties">
        props/jmx-console-users.properties
      </module-option>
      <module-option name="rolesProperties">
        props/jmx-console-roles.properties
      </module-option>
      <module-option name="hashAlgorithm">SHA-256</module-option>
      <module-option name="hashEncoding">base64</module-option>
    </login-module>
  </authentication>
</application-policy>
```

Notes

- ◆ The configuration above does the following
 - Hashes it with the SHA-256 algorithm
 - Encodes it using base64 encoding
 - base64 encoding's main purpose is to enable the transfer of binary data using only 7-bit ASCII characters
 - It is used because certain protocols, such as SMTP and NNTP may not yet support other encodings
 - It does not provide any security benefits
- ◆ We can use these same options for other login modules also
 - For example, if we were using a login module that stores passwords in a database (as shown in a later section) we'd use the exact same `hashAlgorithm` and `hashEncoding` properties
 - The only difference would be that our hashed passwords would be stored in the DB instead of in a properties file

Generating Hashed Passwords

- ◆ ***org.jboss.security.Base64Encoder*** provides a convenient utility to hash strings and encode them using Base 64 encoding
 - This is exactly what is needed to support password hashing
- ◆ Below we hash the string 'admin' using the SHA-256 algorithm
 - Note how we set up the classpath first

```
java -classpath %JBOSS_HOME%\client\jbossa11-client.jar  
org.jboss.security.Base64Encoder admin SHA-256
```

- The returned string is:

```
[jG125bVBBBW96Qi9Te4V37Fnqchz/Eu4qB9vKrRIqRg=]
```

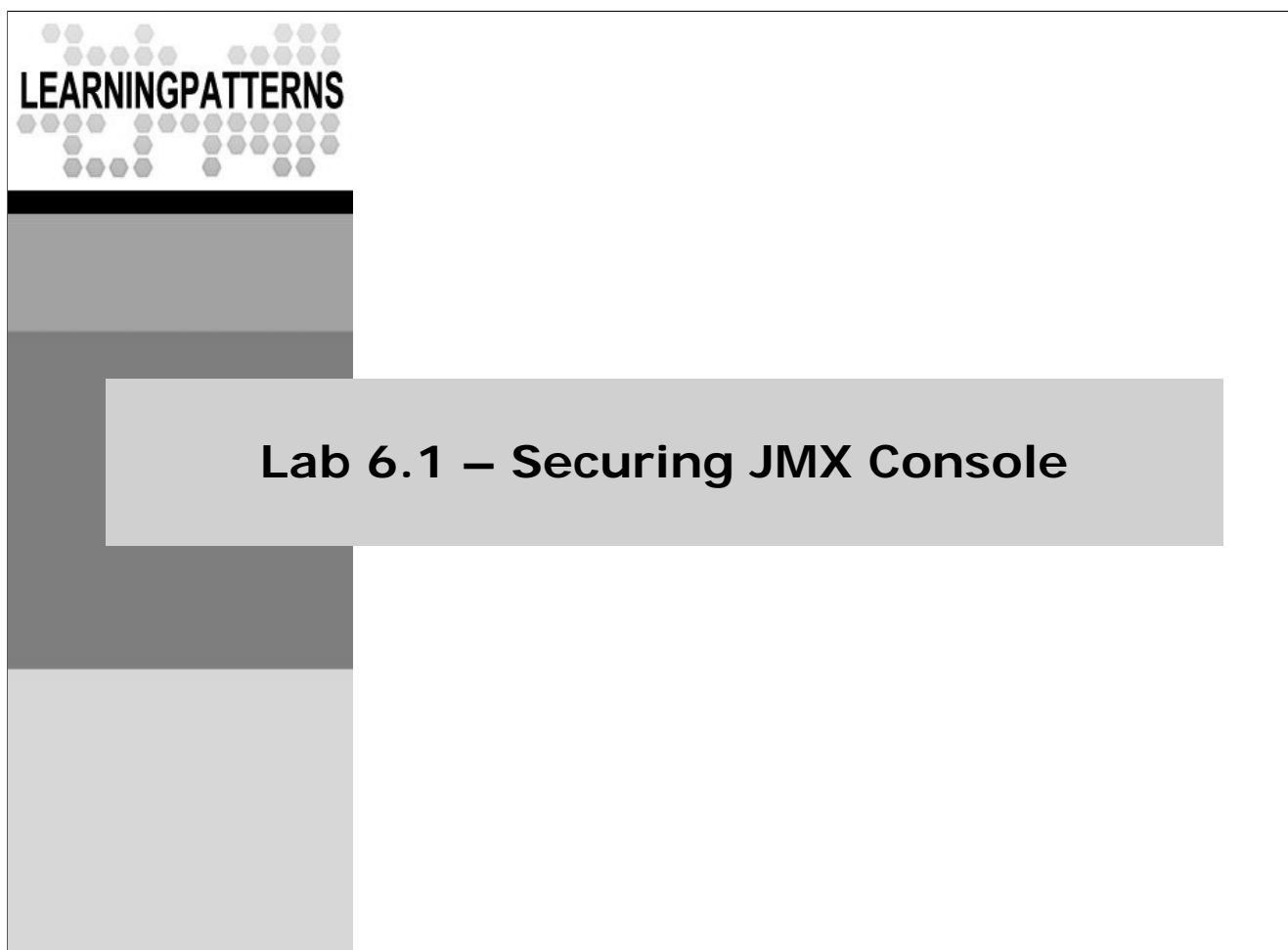
- Note - the enclosing brackets are not part of the hashed value

- ◆ We would use hashed passwords in the data store
 - So, for example, our *jmx-console-user.properties* file would be:

```
# jmx-console-users.properties: Username=encryptedPassword  
admin=jG125bVBBBW96Qi9Te4V37Fnqchz/Eu4qB9vKrRIqRg=
```

Notes

- ◆ Base64 encoding is a MIME standard for taking binary data, and translating into a base 64 representation
 - It is not an encryption technology, it is only used to facilitate the easy transfer of binary data
- ◆ The Base64Encoder class can hash a value with a supplied algorithm before encoding it using Base64
- ◆ The SHA-256 algorithm is one of a family of SHA algorithms (the SHA-2 family)
 - SHA-256 produces a 256 bit message digest



Notes

Lab 6.1 – Securing JMX Console



- ◆ **Overview:** In this lab, we will configure the JMX Console Web application to require authentication
 - Most of the required configuration elements are already in the configuration files (just commented out)
- ◆ **Objectives:**
 - Become more familiar with configuring JBoss security
- ◆ **Builds on previous labs:** none
- ◆ **Approximate Time:** 15-20 minutes
- ◆ **Note:** We do not secure the Web console in this lab, but it would be done in exactly the same manner
 - **It is important to secure the Web console** also

Notes

- ◆ The Web console gives just as much access to the server as the JMX console
 - You should apply the same security measures to the Web console that you do to the JMX console
 - Otherwise, your server is still wide open

Review



- ◆ A **security domain** with the name *jmx-console* is already set up in *conf/login-config.xml*
 - It uses *UsersRolesLoginModule*, and the files *props/jmx-console-users.properties* and *props/jmx-console-roles.properties*
 - This entry already exists, NO need to modify *login-config.xml*
 - The domain name in *login-config.xml* is **jmx-console**, so the JNDI name to use for it is **java:/jaas/jmx-console**
- ◆ You will need to specify the following for *jmx-console.war*
 - In **WEB-INF/web.xml** specify what parts of the web app are secure using existing security elements you can uncomment
 - In **WEB-INF/jboss-web.xml** specify the security domain
- ◆ We give detailed instructions for all of this

Notes

Configure the JMX Console



Tasks to Perform

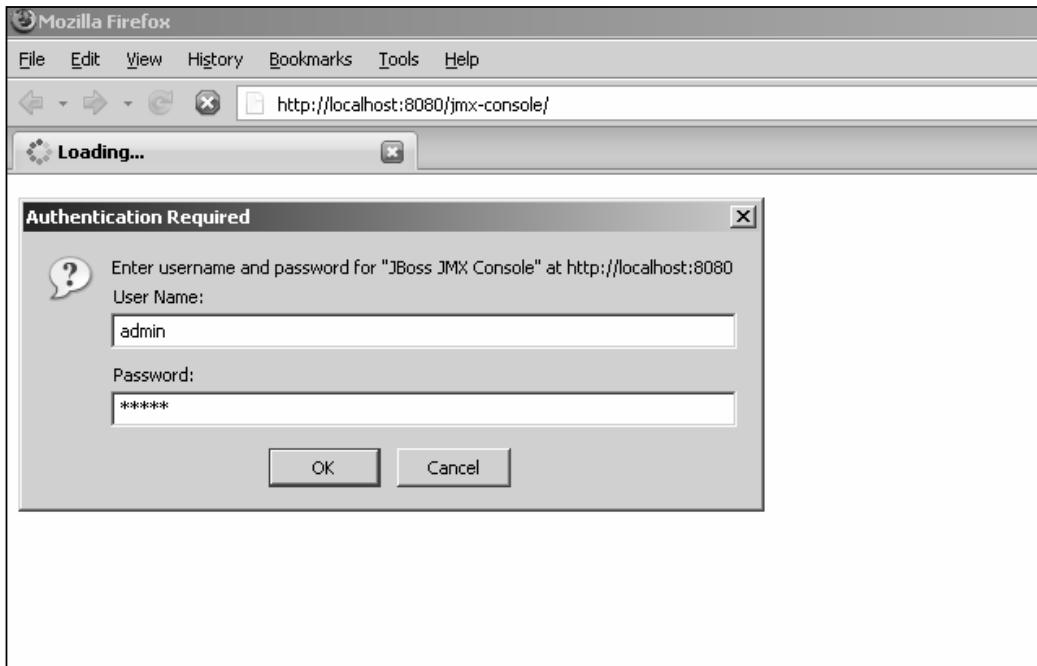
- ◆ Open *deployjmx-console.war\WEB-INF\jboss-web.xml* for editing
 - **Uncomment** the `<security-domain>` element * and save the file
 - This element configures the JMX Console to use the **jmx-console** security domain - which is already present in *conf\login-config.xml*
- ◆ Open *deployjmx-console.war\WEB-INF\web.xml* for editing
 - **Uncomment** the `<security-constraint>` element near the bottom of the file - this defines the security requirements for the Web app *
 - **Make sure you remove** the explicit specification of the GET and POST verbs (if present), as discussed earlier in the course
 - Save the file (this will redeploy the jmx-console Web app)
- ◆ Browse to <http://localhost:8080/jmx-console/>
 - You should be prompted for a username/password
 - Enter **admin/admin** - you should be able to access the console

Notes

- ◆ When un-commenting the `<security-domain>` element, all you need to do is move the "end of comment mark `-->`" to before the `<security-domain>` element, so it looks like that below


```
<jboss-web>
<!-- Uncomment the security-domain to enable security. You will need
to edit the htmladaptor login configuration to setup the login
modules used to authentication users.
-->
<security-domain>java:/jaas/jmx-console</security-domain>
</jboss-web>
```
- ◆ Similarly, when un-commenting the `<security-constraint>` element, move the "end of comment mark `-->`" from after `</security-domain>` to before the opening `<security-domain>` element
 - Also, delete the two explicit elements specifying the GET and POST verbs, if present
- ◆ Note that the JMX console uses BASIC authentication, which is not really recommended
 - Form-based authentication is much more flexible, and allows you to log users out (by deleting the session)
 - A discussion on this is beyond the scope of the course

Logging into JMX Console Web app



Notes

[Optional] Use Hashed Passwords



[Optional] Tasks to Perform

- ◆ In this optional part, we will configure the jmx-console login module, and modify the user/password file, to use hashed passwords
- ◆ Save an unchanged copy of *default\conf\login-config.xml* then open it for editing
 - Find the *jmx-console* security domain, and modify it as follows.
 - Add the following two elements before its `</login-module>`
`<module-option name="hashAlgorithm">SHA-256</module-option>`
`<module-option name="hashEncoding">base64</module-option>`
 - This specifies that the passwords are stored in the users file in encrypted form, and how they are encrypted/encoded
 - You can copy these elements from: **Lab06.1\login-config.xml.txt**
- ◆ Save an unchanged copy of *jmx-console-users.properties* (in *default\conf\props*), then copy the *jmx-console-users.properties* in the Lab06.1 dir to *default\conf\props*
 - The lab file contains hashed passwords instead of clear text

Notes

- ◆ There is another file, *hash.bat*, that sets up the classpath for *Base64Encoder*, and invokes it to encrypt a value with a given algorithm
 - For example, to encrypt the string 'admin' using SHA-256, you would invoke it as:
hash admin SHA-256
 - This was used to create the hashed password in the *jmx-console-users.properties* file we supply
 - You shouldn't need to use this, since we supply the users file with the password already hashed

[Optional] Use Hashed Passwords



Tasks to Perform

- ◆ **Restart** the server so the changes take affect
- ◆ Shut all browser windows, go to the JMX Console and log in again
 - Use admin/admin as the username/password, as before
- ◆ Your login should work exactly the same as before
 - The only difference is that the passwords are not stored in clear text
- ◆ If you want, open up a command prompt in the Lab06.1 dir
 - The *hash* command file is set up to hash any passed in args
 - Execute it, and pass in the string admin, and algorithm SHA-256
 - You should see the password hash we used in the lab as output

Command Prompt

```
C:\StudentWork\JBossAdmin\workspace\Lab06.1>hash admin SHA-256  
[jG125b0BBBBW96Qi9Te4U37Fnqchz/Eu4qB9vKrRIqRg=]  
C:\StudentWork\JBossAdmin\workspace\Lab06.1>
```



Notes

DatabaseServerLoginModule - DB Based

- ◆ **DatabaseServerLoginModule**: JDBC based login module that authenticates against a relational database
- ◆ Based on two **logical** tables
 - Table Principals(PrincipalID text, Password text)*
 - Table Roles(PrincipalID text, Role text, RoleGroup text)*
 - The Principals table associates an id with a password
 - The Roles table associates an id with its set of roles
- ◆ You specify the SQL queries the module uses to retrieve the data from your DB (hence the term logical tables)
 - The SQL queries can be customized to your schema, as long as the *ResultSet*s have the appropriate structure:
 - Result set for user query contains password elements
 - Result set for roles query contains [role | roleGroup] pairs
 - You can have different table names/structure in your DB

Notes

- ◆ The tables are called "logical" because they can actually have any structure
 - As long as you specify queries that return result sets structured as shown in the slide, it will work

Security Domain - DatabaseServerLoginModule

- ◆ The **messaging** domain using *DatabaseServerLoginModule* is shown below, and supports the following module options
 - ***unauthenticatedIdentity***: Identity assigned to unauthenticated id
 - ***dsJndiName***: JNDI name of datasource for the database
 - ***principalsQuery***: Prepared statement query for user/password
 - ***rolesQuery***: Prepared statement query for user/role
 - ***ignorePasswordCase***: Should password comparison ignore case

```
<application-policy name="messaging">
  <authentication>
    <login-module code="org.jboss.security.auth.spi.DatabaseServerLoginModule"
      flag = "required">
      <module-option name = "unauthenticatedIdentity">guest</module-option>
      <module-option name = "dsJndiName">java:/DefaultDS</module-option>
      <module-option name = "principalsQuery">
        SELECT PASSWD FROM JBM_USER WHERE USER_ID=?
      </module-option>
      <module-option name = "rolesQuery">
        SELECT ROLE_ID, 'Roles' FROM JBM_ROLE WHERE USER_ID=?
      </module-option>
    </login-module>
  </authentication>
</application-policy>
```

Notes

- ◆ This security domain is taken from the one defined in *deploy\messaging\messaging-jboss-beans.xml*
- ◆ Prepared statements use the symbol ? as placeholders for query parameters
 - These placeholders are filled in with actual values when the queries are run

Tables and Queries for Messaging Example

- ◆ The DB tables for the previous example are shown at bottom
 - ***principalsQuery*** just selects the password for the given id:

```
SELECT PASSWD FROM JBM_USER WHERE USER_ID=?
```

- ***rolesQuery*** has to take into account the fact that there is no *RoleGroup* column in our JBM_ROLE table (remember the logical structure of the ResultSet is [Role | RoleGroup])
- This is accomplished by hard coding a role group in the query (note that the role group value is case sensitive)

```
SELECT ROLE_ID, 'Roles' FROM JBM_ROLE WHERE USER_ID=?
```

JBM_USER	
PK	USER_ID
	PASSWD
	CLIENTID

JBM_ROLE	
PK	ROLE_ID
PK, FK	USER_ID

Notes

- ◆ You can see examples of the SQL used to create these tables for various databases in docs\examples\jms
 - If you don't provide SQL to create the tables, then the default (for Hypersonic) is used

LdapLoginModule - LDAP Based

- ◆ **LdapLoginModule**: Login module authenticating against LDAP server
 - Used if your user/role information is stored in an LDAP server
 - The LDAP server is accessed via a JNDI LDAP provider
- ◆ There are a number of configuration options for connecting to the LDAP server, including:
 - **java.naming.factory.initial**: InitialContext factory to use
 - **java.naming.provider.url**: URL for LDAP server
 - **java.naming.security.protocol**: Protocol for secure access (e.g. SSL)
 - **java.naming.security.authentication**: Security level to use
 - **java.naming.security.credentials**: Login credential (e.g. password)
- ◆ Authentication of a user is done by connecting to the LDAP server using the configuration options above

Notes

- ◆ The connectivity configuration options are fairly standard LDAP JNDI properties
- ◆ These would be specified using `<module-option>` elements with the names shown in the slides, and the values appropriate for your LDAP server – as in the example below

```
<module-option name="java.naming.factory.initial">
    com.sun.jndi.ldap.LdapCtxFactory
</module-option>
<module-option name="java.naming.provider.url"> ldap://ldapserver/ </module-option>
<module-option name="java.naming.security.authentication"> simple </module-option>
```
- ◆ To add SSL to the LDAP login module, you could use the following option

```
<module-option name="java.naming.security.protocol">ssl</module-option>
```

Using the LdapLoginModule

- ◆ Authenticating against an LDAP server happens in two phases
 - Binding to the directory, using the configuration previously described, and the userDN (which can be constructed from the user's login name)
 - Searching the directory for role objects according to additional configuration information
- ◆ *LdapLoginModule* has configuration properties that let you configure how to construct the *userDN*, and how to search the LDAP directory - for example:
 - The *principalDNPrefix*, and *principalDNSuffix* properties allow you to specify prefixes and suffixes to the user name in order to form the userDN
 - The *rolesCtxDN* property - specifies the fixed distinguished name to the context to search for user roles
- ◆ The details of the (substantial) configuration information for this module are available in the JBoss documentation

Notes

- ◆ To work with the LDAP login module, you need to be able to do the following:
 - Find the user objects in the server, Find the group objects (which contain role information) for each user, for example using the following options:


```
<module-option name="principalDNPrefix">uid=</module-option>
<module-option name="principalDNSuffix">,ou=People,dc=jboss,dc=org</module-option>
<module-option name="rolesCtxDN">ou=Roles,dc=jboss,dc=org</module-option>
<module-option name="uidAttributeID">member</module-option>
<module-option name="matchOnUserDN">true</module-option>
<module-option name="roleAttributeID">cn</module-option>
```
- ◆ The documentation can be found in:
 - The JBoss 4 Server Configuration Guide - Chapter 8: Security on JBoss, Section 5.3.5 - The LdapLoginModule
 - This does NOT appear to be in the JBoss 5 docs as of the time of this writing
 - The LdapLoginModule Wiki page: <http://wiki.jboss.org/community/wiki/LdapLoginModule>

JBoss Security MBeans

- ◆ Two important security related beans
- ◆ ***XMLLoginConfig*** bean in *deploy\security\security-jboss-beans.xml*
 - Specifies the file defining security domains (*login-config.xml*)
- ◆ ***JaasSecurityManager*** MBean in *conf\jboss-service.xml*
 - Configures a number of security related properties, including:
 - ***DefaultUnauthenticatedPrincipal***: Default identity for unauthenticated users
 - ***DefaultCacheTimeout***: Timeout for security info in security cache
- ◆ Note that the security system **caches security information**
 - If user registry has changed "underneath" the security manager, **it will only be reflected when the security cache is refreshed**
 - The JMX Console provides some management for this service, such as listing and flushing the cache for a security domain

Notes

- ◆ To disable security caching, you can set the *DefaultCacheTimeout* to 0
- ◆ Note: *session.invalidate()* removes the users credentials for the cache - for example consider the following scenario
 - User logs in Web app
 - User changes password via the Web app – which updates the user registry, for example the USERS table
 - User logs out - *session.invalidate()* called by the Web app
 - User logs in with the new password (which works)
 - No need to flush authentication cache
- ◆ However, consider the scenario where you use a DB query tool to go into the DB and change the values in the user registry
 - The security cache may still hold old values – which will remain there until they time out or the cache is flushed

The JaasSecurityManager MBean

List of MBean attributes:

Name	Type	Access	Value	Description
SecurityManagerClassName	java.lang.String	RW	org.jboss.security.plugin	MBean Attribute.
ServerMode	boolean	RW	<input checked="" type="radio"/> True <input type="radio"/> False	MBean Attribute.
StateString	java.lang.String	R	Started	MBean Attribute.
DefaultUnauthenticatedPrincipal	java.lang.String	RW	anonymous	MBean Attribute.
SecurityProxyFactoryClassName	java.lang.String	RW	org.jboss.security.Subje	MBean Attribute.
State	int	R	3	MBean Attribute.
CallbackHandlerClassName	java.lang.String	RW	org.jboss.security.auth.c	MBean Attribute.
AuthenticationCacheJndiName	java.lang.String	RW	java/timedCacheFacto	MBean Attribute.
DeepCopySubjectMode	boolean	RW	<input type="radio"/> True <input checked="" type="radio"/> False	MBean Attribute.
DefaultCacheTimeout	int	RW	1800	MBean Attribute.
DefaultCacheResolution	int	RW	60	MBean Attribute.
Name	java.lang.String	R	JaasSecurityManagerService	MBean Attribute.

Apply Changes

Flush JMX Console security cache

void flushAuthenticationCache()

MBean Operation.

Param	ParamType	ParamValue	ParamDescription
p1	java.lang.String	jmx-console	(no description)
Invoke			

Notes

- ◆ Flushing the security cache for a domain will remove all cached information
 - This ensures that any security operations will use the latest information from a domain
 - This is useful if you update a security domain, and want the changes to take effect immediately
 - The screenshots are from the JBoss 4.2 console, but JBoss 5.x has exactly the same capabilities
- ◆ The *getAuthenticationCachePrincipals()* operation is also useful
 - It will show what information is in the cache for a given security domain
- ◆ For some more info, see:
 - <http://www.jboss.org/community/wiki/JaasSecurityManagerService>
 - <http://www.jboss.org/community/wiki/SecurityFAQ>
 - <http://www.jboss.org/community/wiki/JBossSXTTheBigPicture>



Lab 6.2 –DatabaseServerLoginModule

Notes

Lab 6.2 – DatabaseServerLoginModule



- ◆ **Overview:** In this lab, we will configure the jmx-console security domain to use a relational database for its security information
 - We'll need to change the jmx-console security domain definition
 - The jmx-console Web app will remain the same
 - The database we created earlier (accessed via a datasource with name **CatalogDS**) also has security information in it that we'll use in the new jmx-console security domain
- ◆ **Objectives:**
 - Work with *DatabaseServerLoginModule*
- ◆ **Builds on previous labs:** Setup in the Datasource lab
- ◆ **Approximate Time:** 20-30 minutes

Notes

- ◆ Normally, we would probably keep the security information for the server separate from the application information in CatalogDS
 - However, to minimize setup for the labs, we've put everything together in the same database

Configure the JMX Console



Tasks to Perform

- ◆ Open *default\conf\login-config.xml* for editing
 - Find and comment out the existing jmx-console security domain
- ◆ Open *workspace\Lab06.2\login-config.xml.txt*, and copy the complete **<application-policy>** element
- ◆ Paste this below the old jmx-console security domain, and modify it:
 - Set the name to **jmx-console**, and the **dsJndiName** for **CatalogDS**
 - Modify the **principalsQuery** and **rolesQuery** to conform to the database tables below, using the SQL shown, then save the file
 - **principalsQuery**: SELECT PASSWORD FROM JMX_CONSOLE_USERS WHERE USERID=?
 - **rolesQuery**: SELECT ROLE, 'Roles' FROM JMX_CONSOLE_ROLES WHERE USERID=?

JMX_CONSOLE_USERS	
PK	USERID
	PASSWORD

JMX_CONSOLE_ROLES	
PK	ROLE
PK, FK	USERID

Notes

- ◆ Remember that the dsJndiName should start with java:/
- ◆ The tables above are present in the Derby database we've set up
 - You can look at the file *StudentWork\JBossAdmin\Derby\dbCreate.sql* to see the SQL that creates these tables

Test the JMX-Console Login



Tasks to Perform

- ◆ **Restart the server** - this is required for the changes in *login-config.xml* to be picked up
- ◆ **Shut down all your browser windows** (JMX Console uses basic authentication, and the browser will cache the login info)
- ◆ Try browsing to the JMX Console:
 - <http://localhost:8080/jmx-console/>
- ◆ You should be prompted for login information
 - Enter ***SomeUser*** as the user name, and ***password*** as the password (this user is in the database with the role JBossAdmin)
 - You should be able to view the JMX Console with this login information
 - Note that the previous admin user (with password=admin) is also in the database

Notes

[Optional] Use Hashed Passwords



[Optional] Tasks to Perform

- ◆ The database has a table, *JMX_CONSOLE_ENC_USERS* that contains the passwords in encrypted form - which we can use for *jmx-console*
- ◆ Open *default\conf\login-config.xml* for editing
 - Find the *jmx-console* security domain, and modify it as follows.
 - Set the *principalsQuery* value to be:
`SELECT PASSWORD FROM JMX_CONSOLE_ENC_USERS WHERE USERID=?`
 - The *rolesQuery* value can remain the same
 - Add the following to elements after the *rolesQuery* definition - you can copy these from Lab06.1\login-config.xml.txt
`<module-option name="hashAlgorithm">SHA-256</module-option>`
`<module-option name="hashEncoding">base64</module-option>`
- ◆ **Restart** the server so the changes take affect
- ◆ Shut all browser windows, go to the JMX Console and log in again - it should work exactly as before

Notes

Management and Documentation



Tasks to Perform

- ◆ Browse to the JMX console, search for the *JaasSecurityManager* MBean (in the ***jboss.security*** domain), and view it
 - Review the MBean attributes
- ◆ Find the ***getAuthenticationCachePrincipals()*** method, and invoke it with an argument of ***jmx-console***
 - This will show you what information is in the cache
- ◆ Find the ***flushAuthenticationCache()*** method, and invoke it with an argument of ***jmx-console***
 - This flushes the cache - check this by listing the principals again
- ◆ **Look at the Documentation:**
 - Find the **JBossSX & SecureJBoss** Wiki pages (from main Wiki page)
 - There are many other Wiki pages accessible from these - look at some of them, making sure to look at some of the login modules

Notes

[Optional] Use twiddle to Manage Cache

Lab

Tasks to Perform

- ◆ If you want to flush a security cache programmatically you can do so with twiddle – we'll do it from a command line
- ◆ Close all browser windows, reopen one and log into the jmx-console
 - So we have something in the cache
- ◆ Open a command prompt in `<jboss>\bin` and execute the following to check the cache (it should be all on one line)
`twiddle invoke jboss.security:service=JaasSecurityManager getAuthenticationCachePrincipals jmx-console`
 - You should see the admin user listed in the output
- ◆ Now execute the following to empty the cache (all on one line)
`twiddle invoke jboss.security:service=JaasSecurityManager flushAuthenticationCache jmx-console`
- ◆ Check the cache again using twiddle
 - It should be empty !

STOP

Notes



Encrypting Datasource Passwords

Java EE Security Overview

JBoss AS Security

Encrypting Datasource Passwords

TLS/SSL and HTTPS

Securing JBoss Services

Notes

Encrypting Database Login Passwords

- ◆ It's bad practice to store database passwords in clear text
 - `org.jboss.resource.security.SecureIdentityLoginModule` supports the encryption of datasource passwords
 - You can use this as a login module when defining a security domain
 - It contains a single user and an encrypted password, as shown below
 - The **`managedConnectionFactoryName`** property value will be based on the JMX name of the datasource

```
<application-policy name="derby-encrypted-pseudouser">
  <authentication>
    <login-module code="org.jboss.resource.security.SecureIdentityLoginModule"
      flag="required">
      <module-option name="userName">guest</module-option>
      <module-option name="password"> <!-- The encrypted password -->
        5dfc52b51bd35553df8592078de921bc
      </module-option>
      <module-option name="managedConnectionFactoryName">
        jboss.jca:name=CatalogDS,service=LocalTxCM
      </module-option>
    </login-module>
  </authentication>
</application-policy>
```

Notes

- ◆ `SecureIdentityLoginModule` uses a hard coded algorithm / key to encrypt/decrypt the datasource password
 - Therefore, it really isn't completely secure
 - Someone can get the source to `SecureIdentityLoginModule` and use it to decrypt your password
 - Still, it is better than storing it in clear text

Using SecureIdentityLoginModule

- ◆ *SecureIdentityLoginModule* is also used to generate the encrypted password
 - After setting up the classpath appropriately, we can simply run it as a program, and pass in the password we want to encrypt
java org.jboss.resource.security.SecureIdentityLoginModule password
 - The result is: ***5dfc52b51bd35553df8592078de921bc***, which we used in the security domain configuration on the previous page
- ◆ Your datasource configuration can now reference this security domain instead of holding a clear text password, as shown below

```
<local-tx-datasource>
  <jndi-name>CatalogDS</jndi-name>
  <!-- Much detail omitted -->
  <!-- Use our security domain instead of user-name/password -->
  <security-domain>derby-encrypted-pseudouser</security-domain>
  <!-- ... -->
</local-tx-datasource>
```

Notes

- ◆ The clear text version held the login info directly in the configuration file, e.g.

```
<user-name>guest</user-name>
<password>password</password>
```

 - Using *SecureIdentityLoginModule*, we remove those elements, and include `<security-domain>` instead, as shown on the slide
 - The referenced security domain includes the password in encrypted form
- ◆ The *JaasSecurityDomain* class may also be used to encrypt passwords
 - Its use is somewhat obscure, and it's also not actually truly safe, as the passwords can be decrypted by looking at the source code for this class
 - In the Wiki page for *JaasSecurityDomain*, it's called "Security by Obscurity" since it's unlikely that anyone will know how or bother to decrypt passwords encrypted in this way
 - See the following Wiki pages for more info:
 - <http://wiki.jboss.org/community/wiki/JaasSecurityDomain>