

# Intro to JEE Servlet Containers for CFML Developers

cf.Objective() 2010

Jamie Krug

Blog: [jamiokrug.com](http://jamiokrug.com)

Twitter: [@jamiokrug](https://twitter.com/jamiokrug)

# What We'll Cover

- Web Server vs. Application Server
- Java Platform & Specifications  
(servlet, servlet container, Java SE, Java EE, EAR/WAR)
- Application isolation techniques, benefits
- Application configuration/deployment  
(EAR/WAR, context path, virtual hosting)
- Application design & code implications
- Multiple "CFML Engines"  
(Adobe ColdFusion, Open BlueDragon, Railo)
- Development environment techniques, benefits

# Web Server vs. Application Server

## Web Server:

"a computer program that delivers (serves) content, such as web pages, using the Hypertext Transfer Protocol."

--Wikipedia

Apache (httpd), IIS

## Application Server:

"a software framework dedicated to the efficient execution of procedures (scripts, routines, programs, ...) for supporting the construction of applications."

--Wikipedia

*E.g., JBoss, JRun, Apache Geronimo, Resin, WebSphere (IBM), WebLogic (Oracle), GlassFish, Tomcat\*, Jetty\**

CFML/CFScript == Language

ColdFusion/OpenBD/Railo == Java Web Application

- J2EE-compliant
- Servlet-powered

ColdFusion "Server" == ...

ColdFusion Java Web Application (EAR/WAR)

+

J2EE Application Server

+

Web Server (optional)

# Java Platform & Specifications

- Java Community Process (JCP)
- Java Specification Request (JSR)
  - Specification
  - Implementation
- Java Platform, Standard Edition (Java SE)
  - Base APIs
  - J2SE from 1.2 until 1.5
- Java Platform, Java Enterprise Edition (Java EE)
  - Extends functionality of base Java SE APIs
  - Enterprise JavaBean (EJB), transaction (JTA), XML streams, message service (JMS), Java Server Faces (JSF), persistence

# Servlet

"A Servlet is a Java class which conforms to the Java Servlet API, a protocol by which a Java class may respond to http requests." --Wikipedia

- Dynamic Web content via Java platform
- State maintained via cookies, session variables

Sound Familiar?

...ColdFusion does this, with style & ease!

# Servlet Container

- Implements specs for Servlet and JSP
- Loads Servlet classes
- Manages Servlet request/response

*Example: Apache Tomcat*

Tomcat is made up of 3 main components:

- Catalina: servlet container
- Coyote: HTTP connector
- Jasper: JSP engine

# Java EE Application Server

Java EE certified application server  
=  
Web/servlet container  
+  
full Java EE platform APIs

# Java Web App Archives

## Web Application Archive (WAR)

- WEB-INF -- core directory of WAR
  - web.xml -- deployment descriptor
  - lib -- Java Archives (.jar files)
- Other assets

## Enterprise Application Archive (EAR)

- META-INF -- core directory of EAR
  - application.xml -- deployment descriptor
  - One or more WARs

# EAR/WAR Deployment

## Drag & Drop!

Servlet containers generally contain a default deployment directory (e.g., /tomcat/webapps/).

By default, WAR name is context root (exception: ROOT).

Virtual hosts may be used to change context, application path/location, etc. (more on this, later...)

Be aware of compressed vs. exploded WARs.

**Reminder:** ColdFusion is a Java Web application!

**EXAMPLES...**

# "Instances" & Application Isolation

- Multiple application server instances
- Single application server instance, multiple ColdFusion WAR/EAR "instances" (Java Web applications)
- Single application server instance, single ColdFusion WAR/EAR, multiple ColdFusion Web applications

**EXAMPLES...**

# Virtual Hosting

To allow multiple logical/virtual hosts to live on one physical host, use name-based or IP-based hosting.

Name-based hosting is very flexible for both development and production use.

Virtual hosts are crucial to achieving root context for multiple Web applications on single physical host.

**EXAMPLES (Tomcat server.xml)...**

# Web Server & Application Server

- Web server excels at serving **static** content
- Application server serves **dynamic** content
- Multiple application server instances can be exposed by one Web server (i.e., all via HTTP on port 80)
- Connector/proxy configuration allows Web server to delegate requests for dynamic content

**EXAMPLES...**

# App Code/Design Implications

Web Server and Application Server can have different notion of "root" (e.g., context path may need to be included in URLs and other HTML/requests, but not in cinclude).

URL rewriting can be used to hide/mask context.

**EXAMPLES...**

# IDE Server Config/Management

Leverage your IDE (CFEclipse, IntelliJ IDEA, ColdFusion Builder, etc.):

- Start/stop servlet container right from IDE
- View logs within IDE (debugging, etc.)
- Skip Web/application server configuration overhead
- Compile/deploy/reload Java from IDE

*Consult IDE docs; Joe Rinehart blogged good, quick screencast example with CFEclipse/Tomcat/CF9*

# More to think about...

- Multiple CFML engines (OSS testing, unique features, etc.)
- No need for one CF instance to auto-start and "take over" your dev machine
- Complete client/project isolation; including deployment, staging, etc.
- Choose the application server that suits your project/use case (e.g., Tomcat for small VPS, JBoss for larger CFML/Java app, etc.)
- Troubleshooting: logs, logs logs!  
(also, consider JVM performance tuning)
- Adobe ColdFusion installer/JRun connector "magic"

# Recap

- Web Server vs. Application Server
- Java Platform & Specifications  
(servlet, servlet container, Java SE, Java EE, EAR/WAR)
- Application isolation techniques, benefits
- Application configuration/deployment  
(EAR/WAR, context path, virtual hosting)
- Application design & code implications
- Multiple "CFML Engines"  
(Adobe ColdFusion, Open BlueDragon, Railo)
- Development environment techniques, benefits

Questions?

Thank You!

Jamie Krug

Blog: [jamiokrug.com](http://jamiokrug.com)

Twitter: [@jamiokrug](https://twitter.com/jamiokrug)