Busy Programmers' Guide to Java

Part 2

JEE & Web

- Adam Haskell
- Software Architect at The Kroger Co.
- Corporate Champion for Adobe ColdFusion
- Trainer & Consultant

FUSEBOX

Agenda

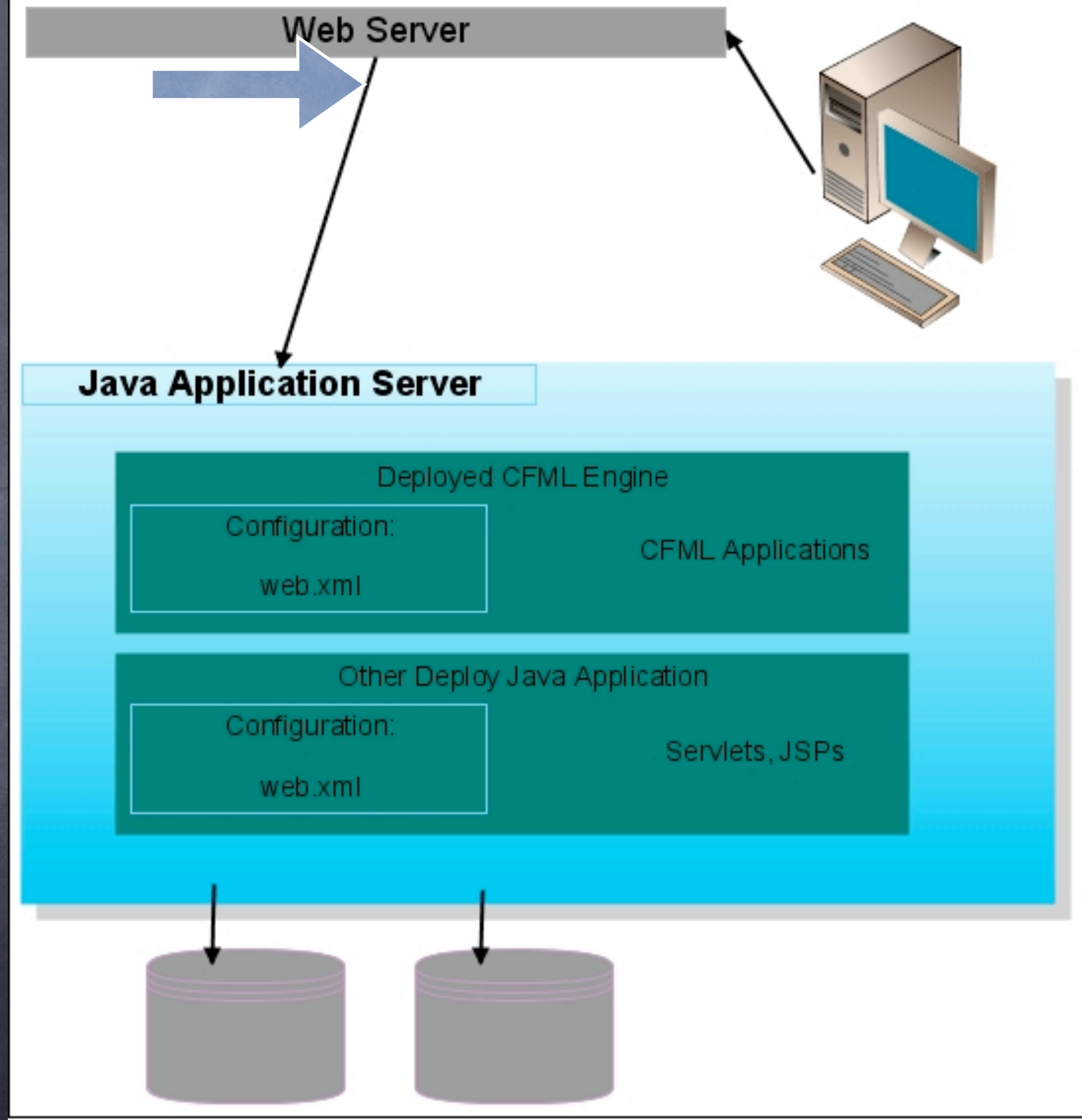
Introduce the SimpleObject Java Files

ColdFusion is JEE

We've All been running in JEE for years now
Adobe has done a great job hiding it
This is all about peeking under the covers and understanding how it is pieced together
No expectation to write Java to service the request just enhance development

Shared Hosting

Almost everything is off limits
Javaloader is (possibly) an option.
Limited functionality...

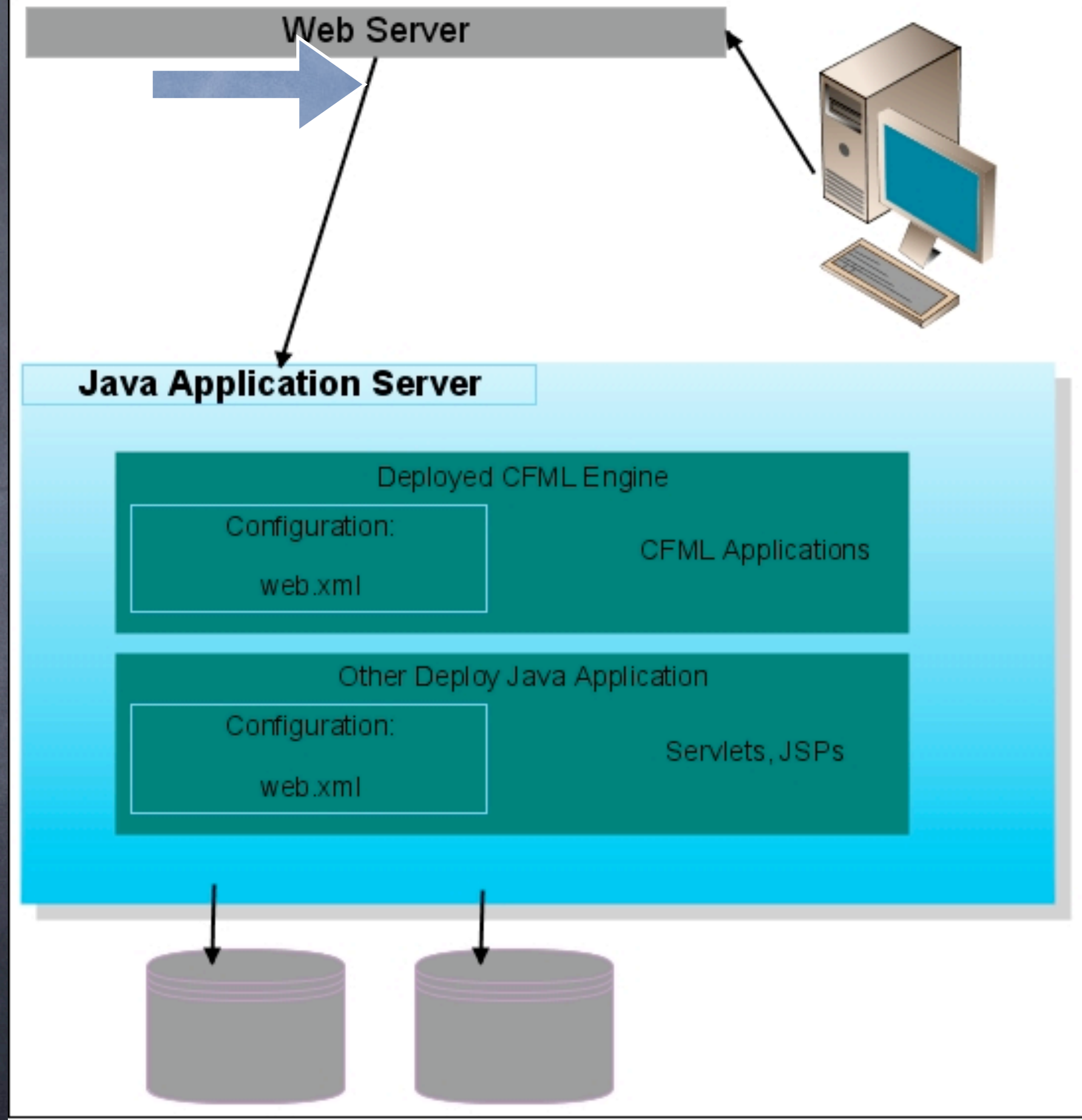


Mention: Logical Diagram
Take a request through the system
brevity is key, each section has its own slide

Webserver

- Apache httpd
- IIS
- nginX
- lighttpd
- Connecting to the Server
  - Special Connector
  - Proxy Connector

Short...this is not JEE specific but people always seem to have questions about this...
Mention: many Web Containers have a port listener that can field HTTP requests.



Mention: Connection to Web server, reinforce previous slide last point
Interactions between web server and Application server
Brevity is Key.

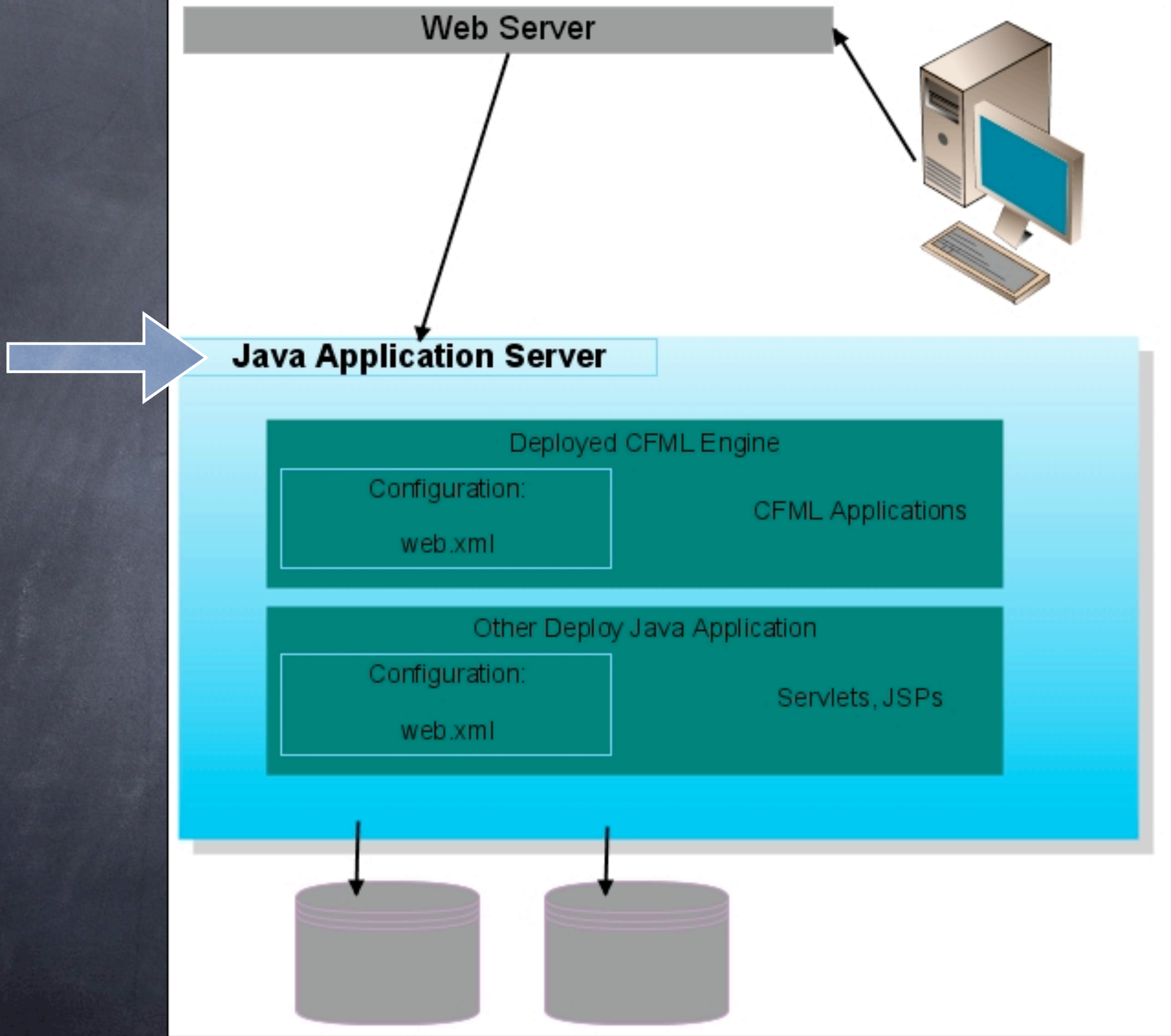
Application Server

Middleware Java Application
Provide Execution Environment for Enterprise and Web applications
Provide hooks into other systems
JDBC pooling for Database
Proxy or AJP(binary) connection to external systems like webservers

Web(Servlet) Container

jetty://

Explain difference with Application Server. Why would someone want an application server?
Will not deploy all JEE components
Typically Limited to WARs
Lack “Enterprisie” Features
JMS (ColdFusion has own JMS functionality)
EJB
JPA (CF9 has it built in, hibernate)

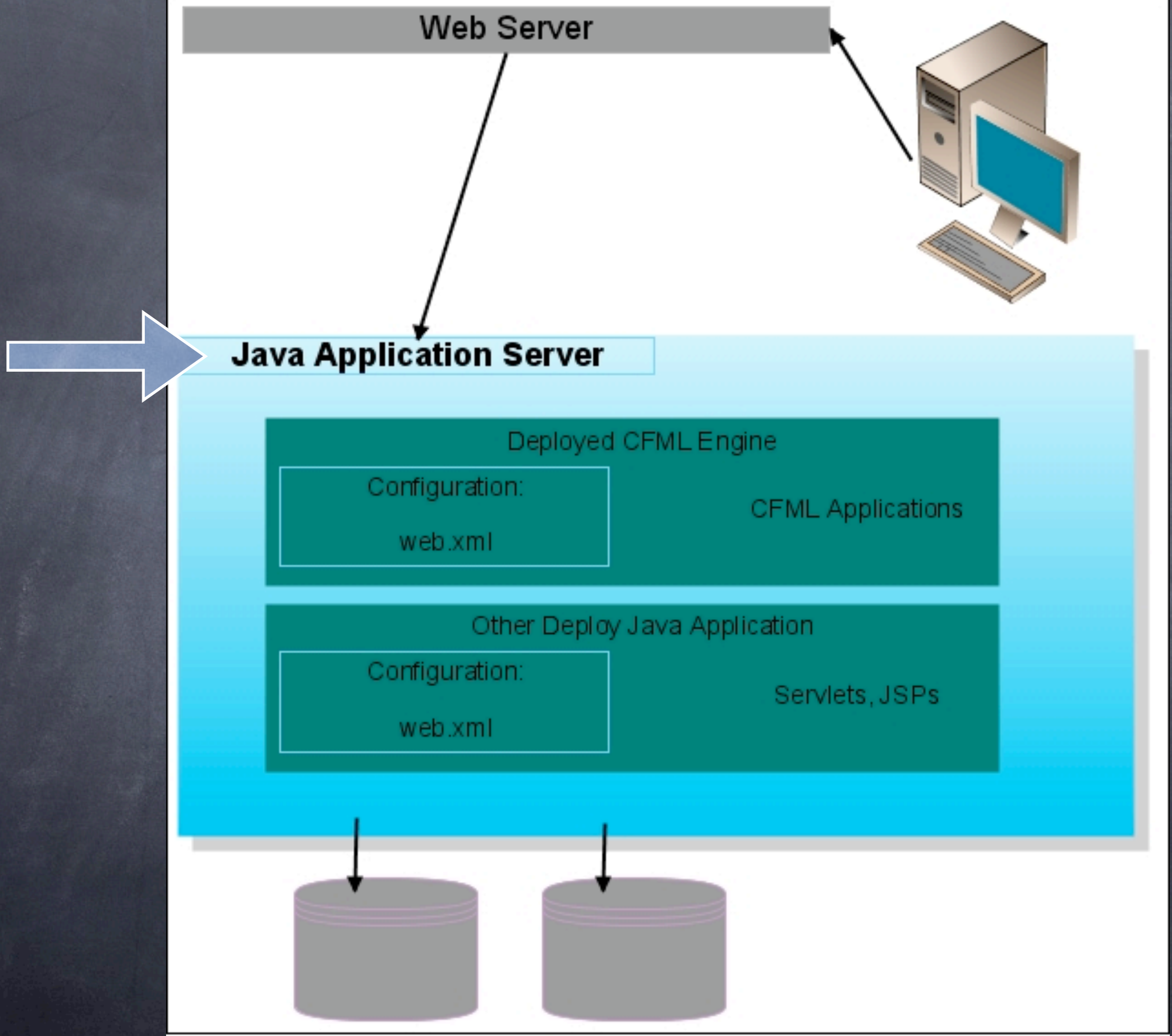


Mention: Web Container, hand off from App server to web container to servlet container
Brevity is Key.

Deployment Packages

- Self-contained
- ColdFusion Hides this pretty well
- WAR
  - The typical JEE deployment style
- EAR
  - EJB
  - Jars

WEB-INF files are not accessible via web requests
Resources vs templates



Mention: Web Container, hand off from App server to web container to servlet container
Brevity is Key.

Servlets

- Standard mechanism to respond to web requests
- Typical Java apps have multiple Servlets
- CFML engines nothing more than a (set of) servlets

```
package sample.contact;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Serv extends HttpServlet {
    /**
     *
     */
    private static final long   serialVersionUID    = 1L;

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 "
                + "Transitional//EN\">\n" + "<html>\n"
                + "<head><title>Hello WWW</title></head>\n" + "<body>\n"
                + "<h1>Hello WWW</h1>\n" + "</body></html>");
    }
}
```


Container loads, calls init(), service client requests, destroy()

web.xml

```
<!-- CF Monitoring Filter  -->
<filter>
    <filter-name>CFMonitoringFilter</filter-name>
    <filter-class>coldfusion.bootstrap.BootstrapFilter</filter-class>
    <init-param>
        <param-name>filter.class</param-name>
        <param-value>coldfusion.monitor.event.MonitoringServletFilter</param-value>
    </init-param>
</filter>
```

```
<filter-mapping>
    <filter-name>CFMonitoringFilter</filter-name>
    <servlet-name>CfmServlet</servlet-name>
</filter-mapping>
```

```
<servlet id="coldfusion_servlet_3">
    <servlet-name>CfmServlet</servlet-name>
    <display-name>CFML Template Processor</display-name>
    <description>Compiles and executes CFML pages and tags</description>
    <servlet-class>coldfusion.bootstrap.BootstrapServlet</servlet-class>
    <init-param id="InitParam_1034013110656ert">
        <param-name>servlet.class</param-name>
        <param-value>coldfusion.CfmServlet</param-value>
    </init-param>
    <load-on-startup>4</load-on-startup>
</servlet>
```

```
<servlet id="coldfusion_servlet_3">
    <servlet-name>CfmServlet</servlet-name>
    <display-name>CFML Template Processor</display-name>
    <description>Compiles and executes CFML pages and tags</description>
    <servlet-class>coldfusion.bootstrap.BootstrapServlet</servlet-class>
    <init-param id="InitParam_1034013110656ert">
        <param-name>servlet.class</param-name>
        <param-value>coldfusion.CfmServlet</param-value>
    </init-param>
    <load-on-startup>4</load-on-startup>
</servlet>
```

```
<listener>
    <listener-class>coldfusion.bootstrap.HttpFlexSessionBootstrap</listener-class>
</listener>
```


Hit what each does/Can do on first round
Demo Filters

JSP vs CFM

Resources v Templates

JSP compiles to a Servlet. Just a Templating, accessible via a Web URL

CFM/CFC each have a servlet dedicated to processing web URLs that match the pattern respectively

Classpaths & Classloaders

- Classloader App/ Container Implemented
- Classpaths are different Stacks of Jars
- New Classloader Instance
- OSGi

Complex Beasts - Entire presentations on Classloaders
CD == JAR
Song == Class

Java Object Demo

Keep Track of Me

- Twitter: AHaskell
- IRC: #ColdFusion #OpenBD
- Blog: http://cfrant.blogspot.com
- E-Mail/Messanger: a.haskell@gmail.com
- Slides: http://dl-client.getdropbox.com/u/64114/JavaPt2.pdf
- CFConversations!!!