Ten Tips For Java Developers with Oracle WebLogic Server

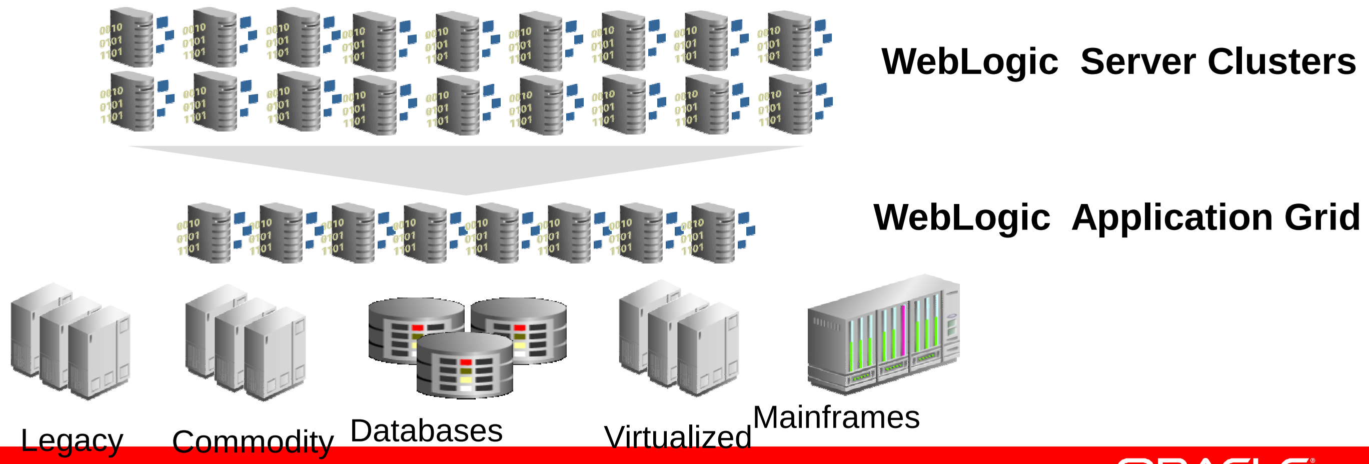
Roger Freixa
EMEA App Grid PM (WebLogic, Java)
Roger.freixa@oracle.com

Oracle WebLogic Server

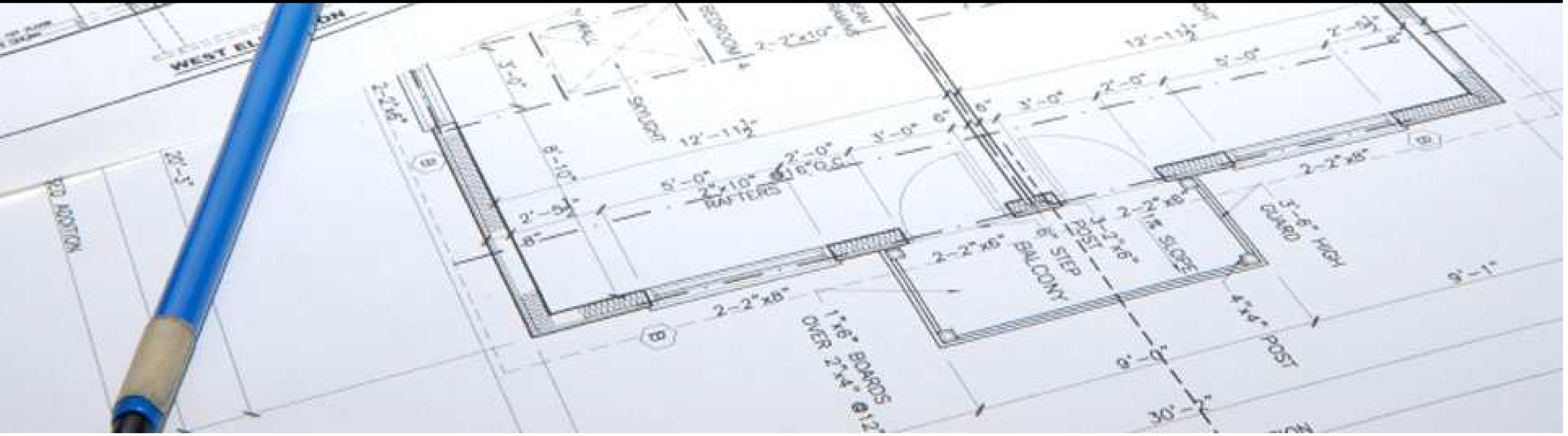
Converged Infrastructure for the Oracle Platform

- The Number #1 Java EE application server, designed for the most Mission-Critical of applications
- Developer-friendly – productive, standards-based development
- Focus on **quality of service** – performance, scalability, reliability, availability
- Built-in **manageability** – configuration, monitoring , diagnostics, maintenance

WebLogic Differentiator: the “ilities”



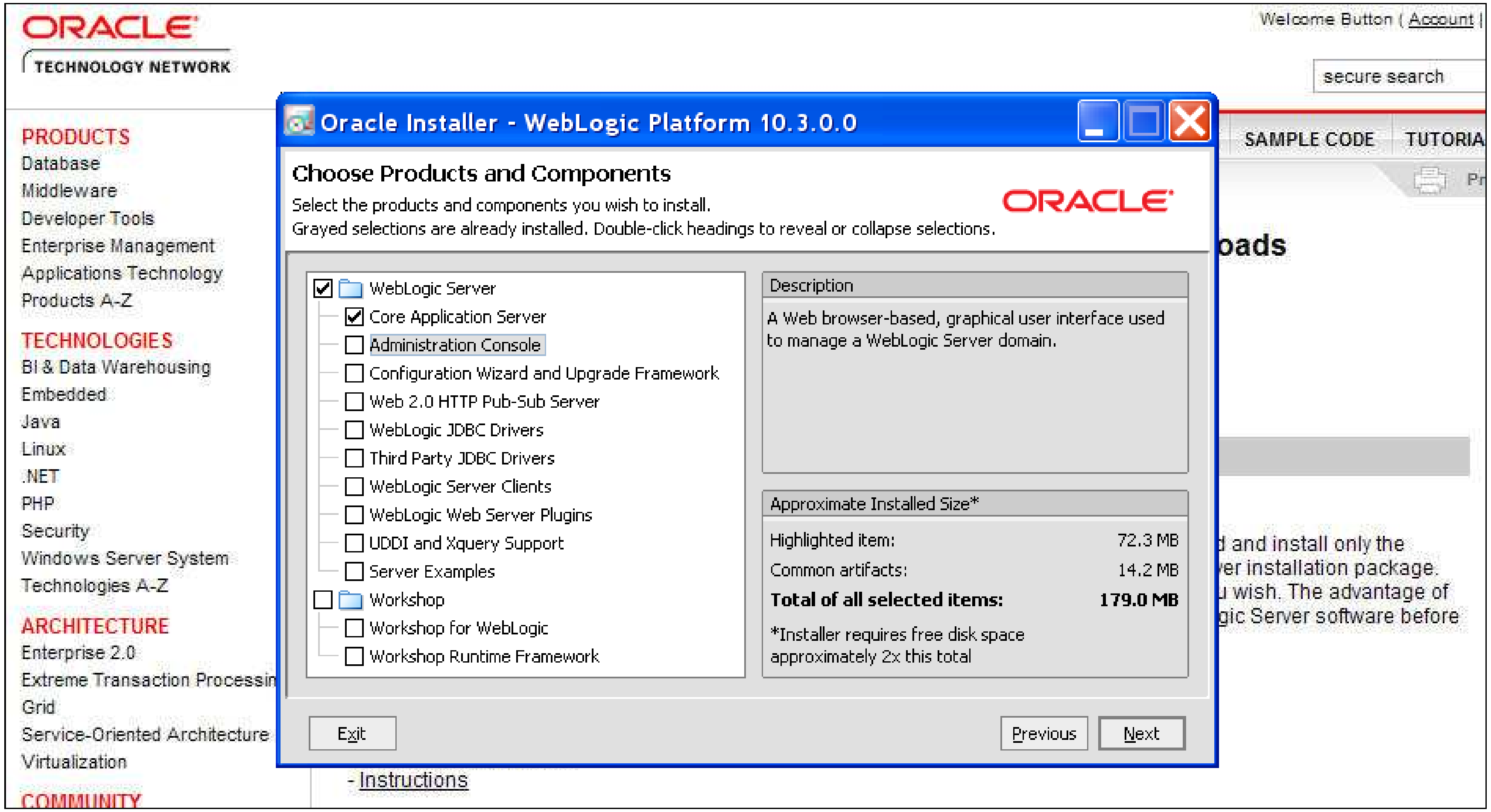
#1 Utilize the Lightweight Server



Lightweight Installation

- Install and configure a WebLogic Server 10.3 instance as quickly as possible
- Utilize network installer
  - Initial download is just 39 Mb
  - Select what you want to install
  - Downloads then installs components
- Minimum server installed < 179 Mb
  - Using pre-installed JDK < 151 Mb
- Core Server Installer For ISVs < 250 Mb

Lightweight Installation



Welcome Button (Account |

secure search

SAMPLE CODE TUTORIA

PRODUCTS
- Database
- Middleware
- Developer Tools
- Enterprise Management
- Applications Technology
- Products A-Z

TECHNOLOGIES
- BI & Data Warehousing
- Embedded
- Java
- Linux
- .NET
- PHP
- Security
- Windows Server System
- Technologies A-Z

ARCHITECTURE
- Enterprise 2.0
- Extreme Transaction Processin
- Grid
- Service-Oriented Architecture
- Virtualization

COMMUNITY

oads

d and install only the
er installation package.
u wish. The advantage of
gic Server software before

- Instructions

Oracle Installer - WebLogic Platform 10.3.0.0

Choose Products and Components

Select the products and components you wish to install.
Grayed selections are already installed. Double-click headings to reveal or collapse selections.

- [✓] WebLogic Server
  - [✓] Core Application Server
  - [ ] Administration Console
  - [ ] Configuration Wizard and Upgrade Framework
  - [ ] Web 2.0 HTTP Pub-Sub Server
  - [ ] WebLogic JDBC Drivers
  - [ ] Third Party JDBC Drivers
  - [ ] WebLogic Server Clients
  - [ ] WebLogic Web Server Plugins
  - [ ] UDDI and Xquery Support
  - [ ] Server Examples
- [ ] Workshop
  - [ ] Workshop for WebLogic
  - [ ] Workshop Runtime Framework

Description

A Web browser-based, graphical user interface used to manage a WebLogic Server domain.

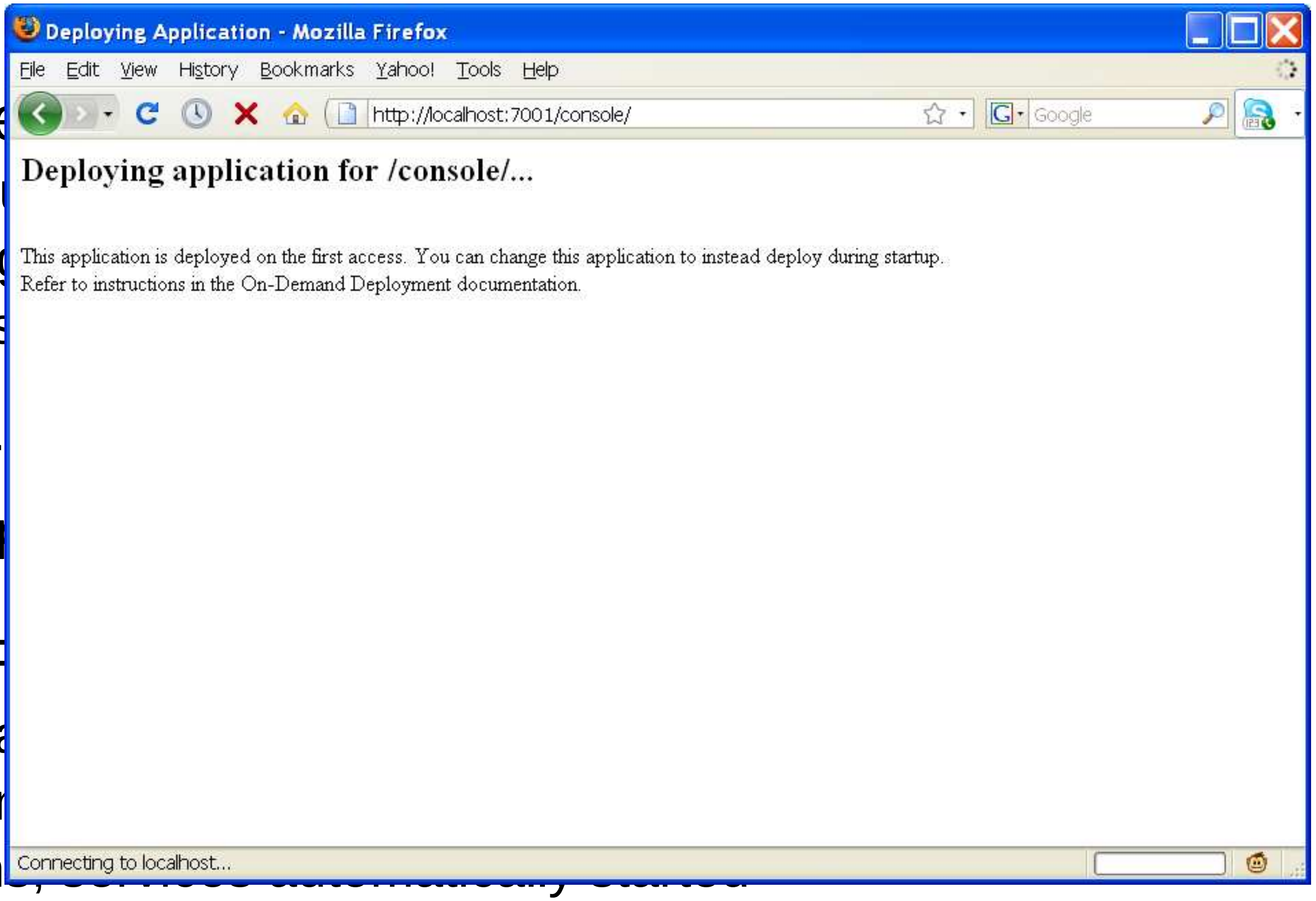
Approximate Installed Size*

Highlighted item:	72.3 MB
Common artifacts:	14.2 MB
Total of all selected items:	**179.0 MB**

*Installer requires free disk space approximately 2x this total

Lightweight, Friendly Server

- Selective Se
  - Choice of f
  - Or lightwei
  - Specify -Ds property
- Fast Server
  - Internal ap mode
- Developer F
  - Disable cha
  - Disable co
  - Application
  - 50% Improved response times over previous releases

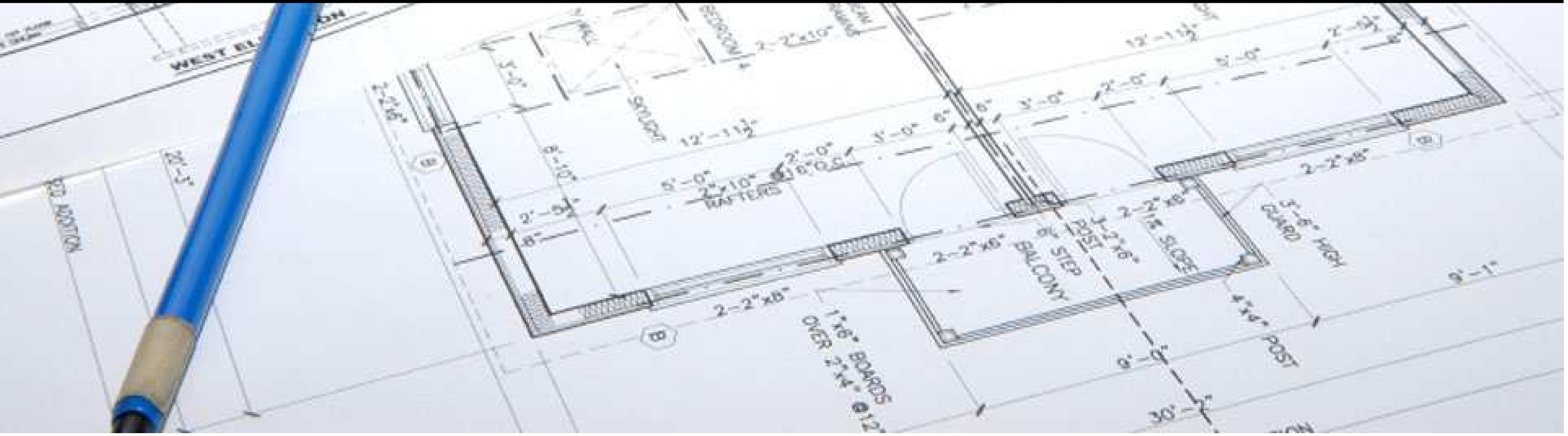

Deploying Application - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://localhost:7001/console/
Google

Deploying application for /console/...

This application is deployed on the first access. You can change this application to instead deploy during startup. Refer to instructions in the On-Demand Deployment documentation.

Connecting to localhost...

#2 Adopt Java Enterprise Edition 5.0



WebLogic Server Standards

Java EE 5.0 and Java SE 5.0/6.0

Java EE 5.0 APIs	Support
JSP 2.1	√
JSF 1.2	√
Servlet 2.5	√
EJB 3.0	√
JAX-WS 2.0	√
JMS 1.1	√
JNDI 1.2	√
JCA 1.5	√
JTA 1.1	√
JACC/JAAS 1.0	√
JMX 1.2	√
J2EE Deployment 1.2	√
J2EE Management 1.1	√
JDBC 3.0	√

√

- Standards Compliant
- Certified JEE 5.0 Compatible

Java EE5 Developer Productivity

- Primary theme of Java EE5 is developer productivity
- Makes extensive use of annotations instead of XML
  - Build naked applications – no deployment descriptors!
- Dependency injection
  - Annotation based container injection of resources
  - No more JNDI lookups
- EJB 3.0 == simpler, usable EJB model
  - Far less coding infrastructure – only need a business interface and implementation class
  - Beans and behavior defined with annotations
  - Interceptors
- Dedicated lightweight persistence API
  - Best of industry breeding, remove all complexity from EJB 2.x CMP
  - Entities are POJO based, use annotations for O-R mapping
  - EntityManager API to create, query, remove entity bean instances
  - Extensive query support – Named, SQL, EJBQL

EJB 3.0 Entity Bean

```
 @Entity
 @Table(name = "EMPLOYEES")
 @NamedQuery(name="Employee.findAll", query="se       o from Employee
o")

 public class Employee implements Seriali
   @Id
   @Column(name = "EMPNO")
   private int empNo;
   private String eName;
   ...
   @ManyToOne
   @JoinColumn(name="MANA          referencedColumnName="EMPLOYEE_ID")
   private Employee ma

   public int getE
     return this.emp
   }

 public Employee getManager() {
     return this.manager;
   }
 }
```

No XML Required

EJB 3.0 Session Bean

```
  @Remote
  public interface EmployeeManager {
    Employee createEmployee(int empNo, String na
    Employee findEmployeeByEmpNo(int empNo);
  }
@Stateless(mappedName="EmployeeManager")
public class EmployeeManagerBean impl            ployeeManager {
 @PersistenceContext private Entit          em;

 public Employee findEmployee          nt empNo) {
   return ((Employee) em.f         oyee.class, empNo));
 }
 public Employee cre           ee(int empNo, String eName) {
   Employee emp = n          oyee(empNo, eName);
   em.persist(emp);
   return emp;
 }
}
```

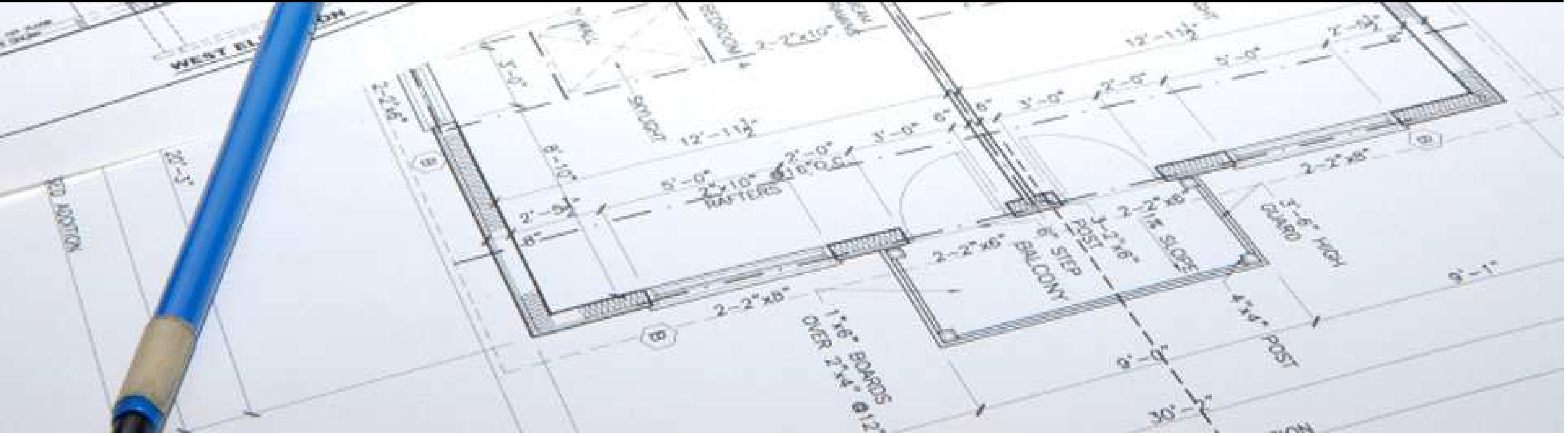
No XML Required

Servlet 2.5 with Dependency Injection

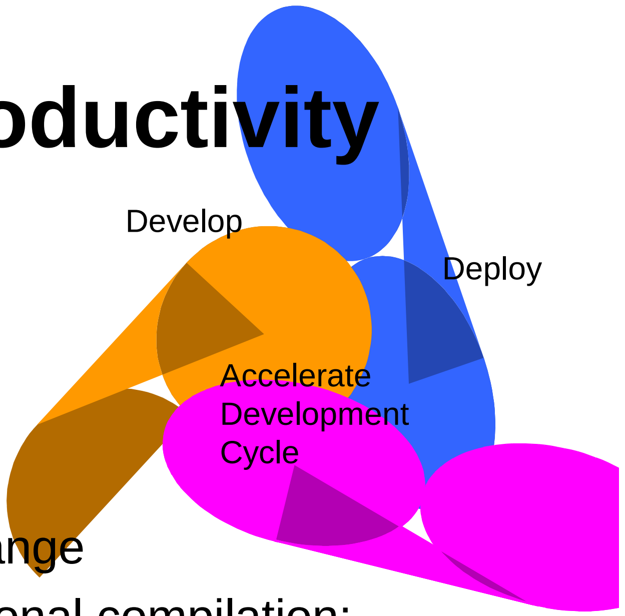
```
 public class HRFaceServlet extends HttpServlet {

   // Inject the CalculatorBean
   @EJB(name=“EmployeeManager")
   EmployeeManager mgr;

   public void doGet(HttpServletReq[illegible], HttpServletResponse
res)
          throws ServletExcepti[illegible]ception {
      ...
      // Create new Employ[illegible]
      Employee newEmp = [illegible]teEmployee(910377, “bill.bloggs”);

      // Assign Emp[illegible] Manager
      Employee mgr[illegible]findEmployee(mgrId);
      mgr.addEmployee[illegible]ewEmp);
      ...
  }
```

No XML Required

#3 Employ FastSwap for Rapid Develop/Test Cycles



FastSwap for Maximum Productivity



- Traditional JEE development cycle:

Edit > Build > Deploy > Test

- Developers must complete cycle for every code change
- Modern IDEs remove the Build step through conditional compilation:

Edit > Deploy > Test

- FastSwap's goal is to eliminate the Deploy step

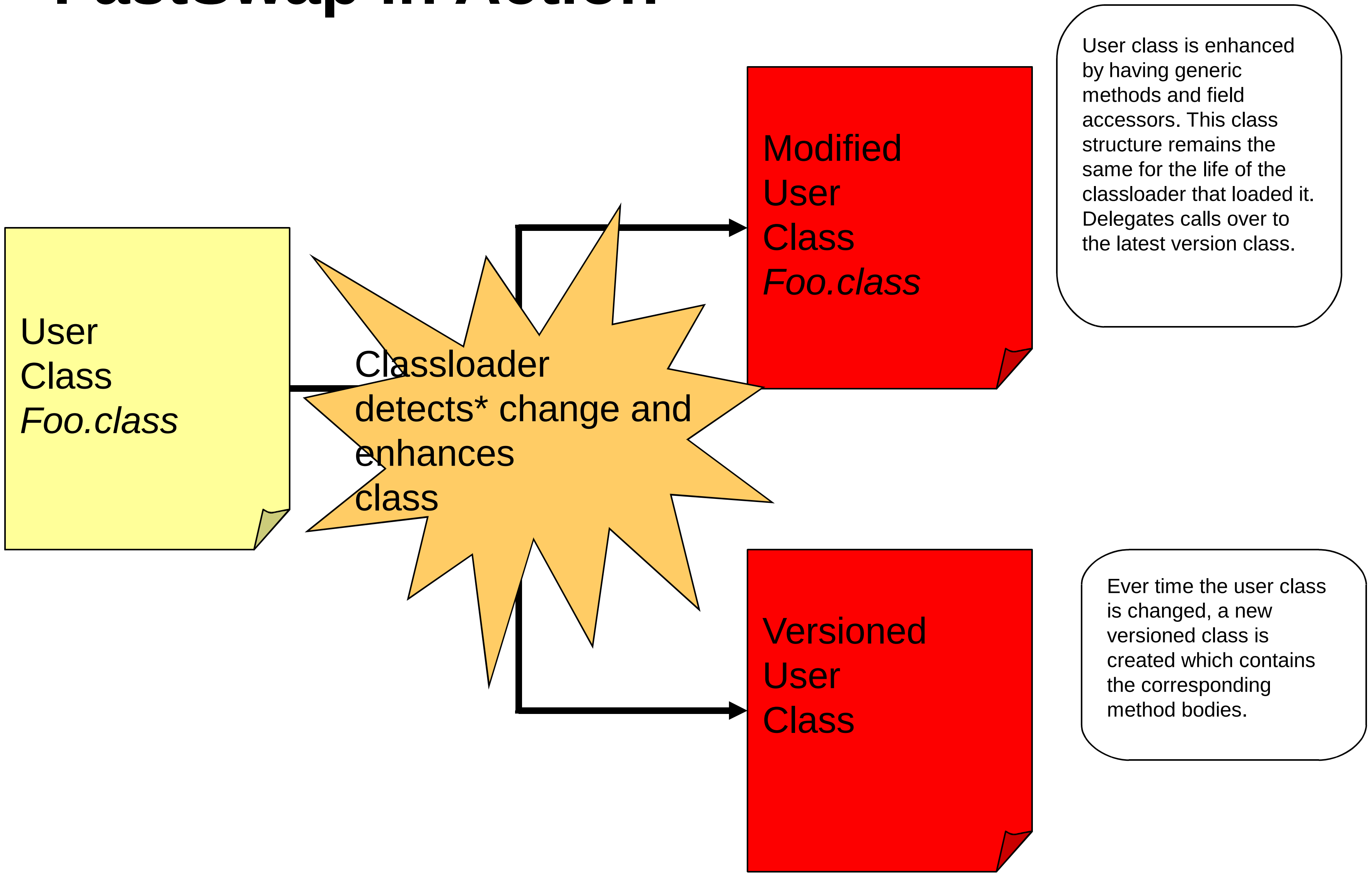
Edit > Test

- A web developer changes his code, refreshes his browser, and immediately sees the results

FastSwap for Maximum Productivity

- Next step in the evolution
  - Redeployment – requires reloading the entire application
  - “Hot” Deploy
  - Partial Redeployment / Split Directory – required dropping and recreating classloaders
  - **FastSwap**
- Enabled by Java 6 Enhancements
  - java.lang.instrumentation.
    Instrumentation.retransformClasses(Class...)
- Preserves the State of Application
  - Replaces the byte code for just the modified methods
  - Maintains instance variables
- Enabled via setting in deployment descriptor
  - Development mode only

FastSwap in Action


User
Class
Foo.class
Classloader detects* change and enhances class
Modified
User
Class
Foo.class
User class is enhanced by having generic methods and field accessors. This class structure remains the same for the life of the classloader that loaded it. Delegates calls over to the latest version class.
Versioned
User
Class
Ever time the user class is changed, a new versioned class is created which contains the corresponding method bodies.

FastSwap Operation

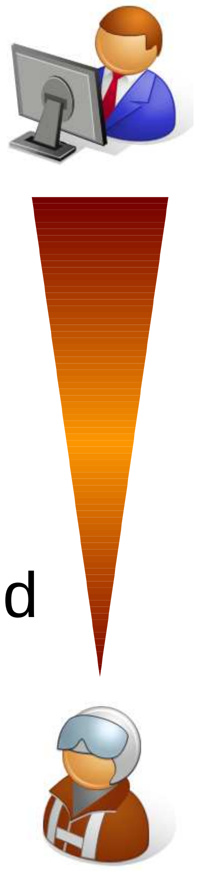
- Detects changes to class files
  - Looks for changes while processing HTTP requests using the FastSwapFilter servlet filter
  - Manual trigger with JMX interface for “headless” applications
  - Works only on `classes` directory: no archives
- Redefines changed classes
  - Automatic for detected class file changes
- Non invasive
  - No dropped classloaders, Servlets, no loss of session state
- Development mode **only**
  - Automatically disabled in production mode

FastSwap Supported Class Changes

- Add, remove constructors & methods
  - Includes static methods
  - Includes final methods
  - Addition and removal of finalize method not supported
- Add, remove fields
  - Includes static fields
  - Includes final fields
- Change constructor and method code
- Change constructor, method, and field modifiers
- Add methods to interfaces

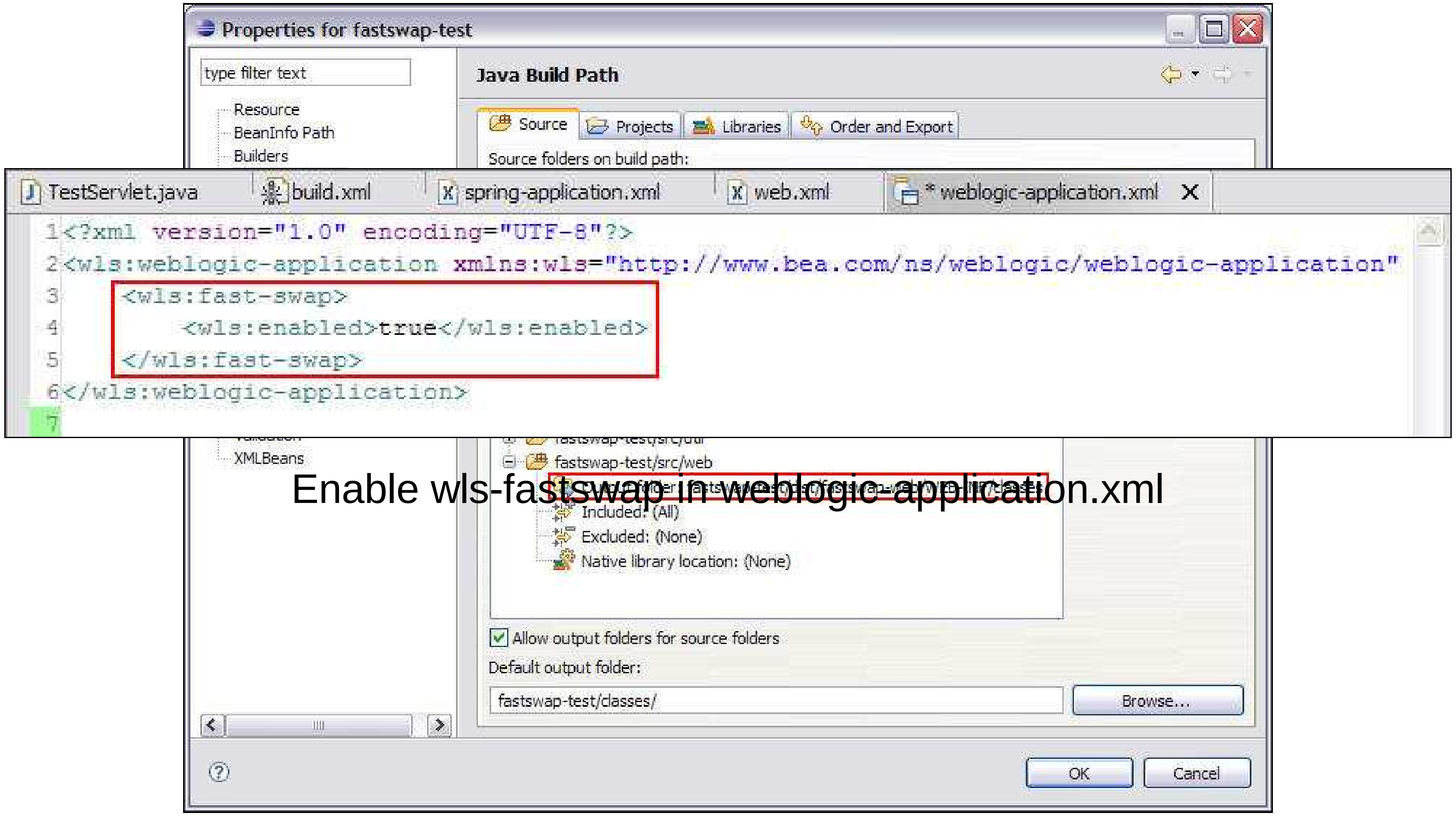
Using FastSwap

1. Create an exploded deployment structure
   - Enable FastSwap in weblogic-application.xml
2. Compile source code modules into respective directories in exploded deployment structure
   - No archives, must be unpackaged classes
3. Deploy application to WebLogic Server using exploded deployment structure
4. Configure IDE to directly compile classes exploded deployment structure
5. Develop/edit   auto-compile with IDE   test immediately



ORACLE

Enabling FastSwap with Eclipse



```xml
<?xml version="1.0" encoding="UTF-8"?>
<wls:weblogic-application xmlns:wls="http://www.bea.com/ns/weblogic/weblogic-application"
    <wls:fast-swap>
        <wls:enabled>true</wls:enabled>
    </wls:fast-swap>
</wls:weblogic-application>
```

Source folders on build path:

fastswap-test/src/web
- Included: (All)
- Excluded: (None)
- Native library location: (None)

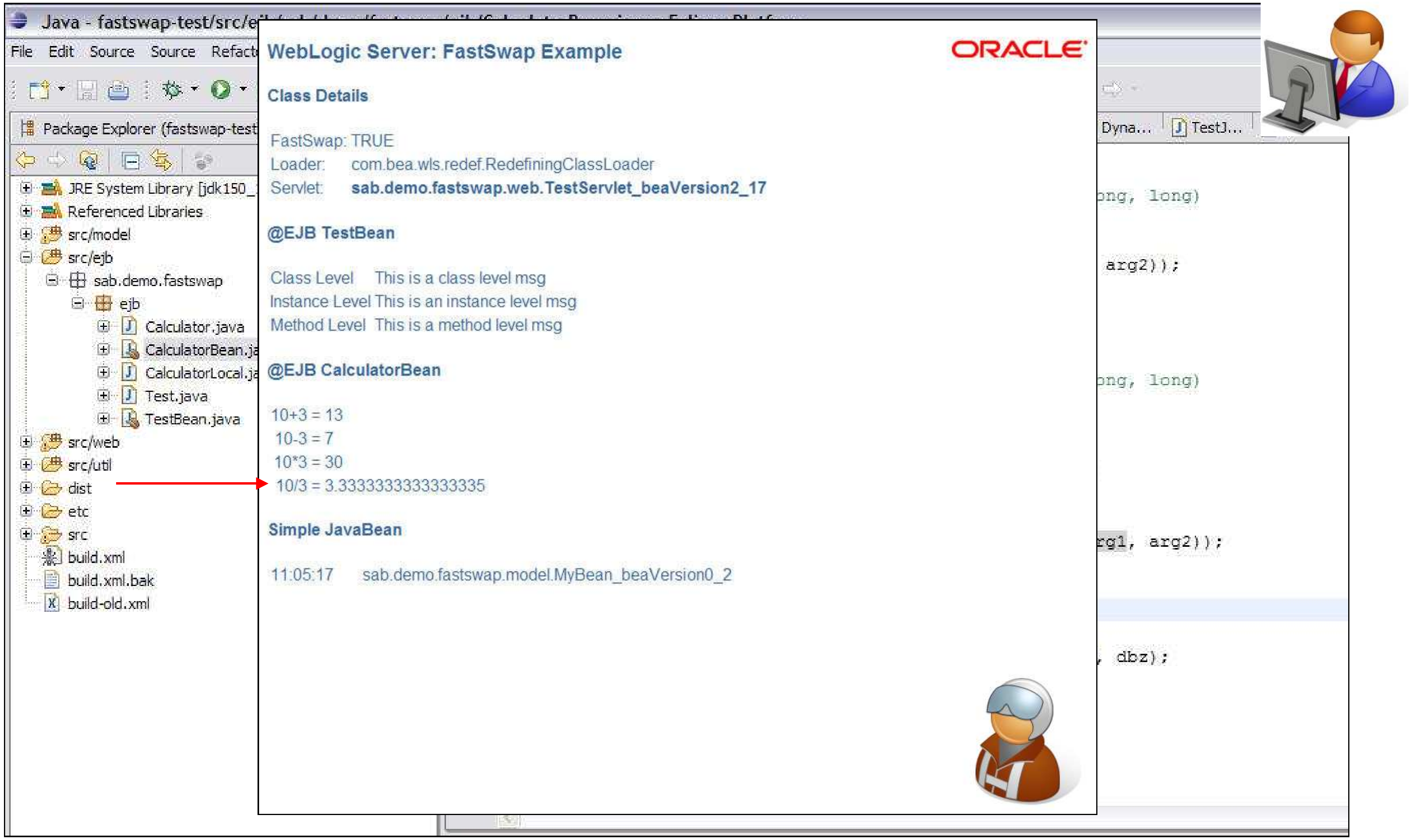
[✓] Allow output folders for source folders

Default output folder: fastswap-test/classes/

Enable wls-fastswap in weblogic-application.xml

Configure project to compile into exploded directories

Using FastSwap with Eclipse



WebLogic Server: FastSwap Example

Class Details

FastSwap: TRUE
Loader: com.bea.wls.redef.RedefiningClassLoader
Servlet: **sab.demo.fastswap.web.TestServlet_beaVersion2_17**

@EJB TestBean

Class Level This is a class level msg
Instance Level This is an instance level msg
Method Level This is a method level msg

@EJB CalculatorBean

10+3 = 13
10-3 = 7
10*3 = 30
10/3 = 3.3333333333333335

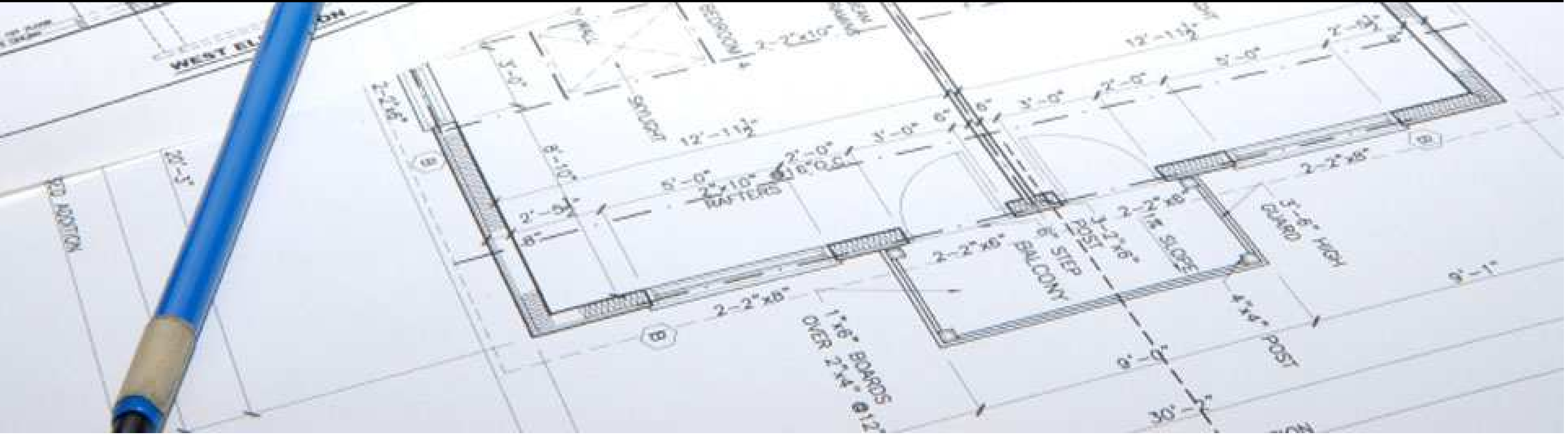
Simple JavaBean

11:05:17 sab.demo.fastswap.model.MyBean_beaVersion0_2

```
ong, long)
 arg2));
ong, long)
rg1, arg2));
, dbz);
```

FastSwap Demonstration

#4 Use Split Development



WebLogic Server Split Development

- Use WebLogic Split Development for starting new projects
- Development by convention
  - Prescribed directory layout for EJBs, web modules, common libraries
  - Prescribed locations for source code, deployment descriptors, HTML, images, …
- Tooling to automatically generate build files
  - Examines directory structure to infer module types
  - Uses Ant tasks that help you repeatedly build, change, and deploy Java EE applications.
  - Separates project source from generated artifacts
- Provides
  - Fast development and deployment
  - Simplified build scripts
  - Easy integration with source control systems

Develop By Convention

```
+---APP-INF
|   +---classes
|   |       applicationresource.properties
|   +---lib
|           GenericResourceLoader.jar
|
+---appUtils
|   +---examples\hello\apputils\AppUtils.java
|
+---helloEJB
|   +---examples\hello\ejb\HelloBean.java
|   +---examples\hello\ejb\Hello.java
|
+---helloWebApp
|   |   index.jsp
|   |   wls_examples.css
|   +---WEB-INF
|       |   web.xml
|       |   weblogic.xml
|       +---src
|           +---examples\hello\utils\WebAppUtils.java
|
+---META-INF
        application.xml
        weblogic-application.xml
```

- Create project directory structures
- Similar to JEE archive format
- Modules are located at top level
- Deployment descriptors placed in META-INF, WEB-INF directories
- Source code placed in package structure under specific modules
- Web module source placed in WEB-INF/src directory

Generate Ant Build File

- Use weblogic.BuildXMLGen to generate Ant build file
  - Inspects directory structure, generates corresponding build file with appropriate targets
  - Uses Oracle Ant tasks

```
Usage: java weblogic.BuildXMLGen [options] <src_dir>

where options include:
   -help               Print the standard usage message.
   -version            Print version information.
   -projectName <project name> name of the ANT project.
   -d <directory>      directory where build.xml is created.
                       Default is the current directory.
   -file <build.xml> name of the generated build file.
   -username <username> user name for deploy commands.
   -password <password> password for the user.
   -adminurl <url>     Administatrion Server URL.
   -librarydir <directories> Comma-separated list of directories

$ java weblogic.BuildXMLGen -projectName helloWorld -d . -username weblogic -password
  weblogic .
```

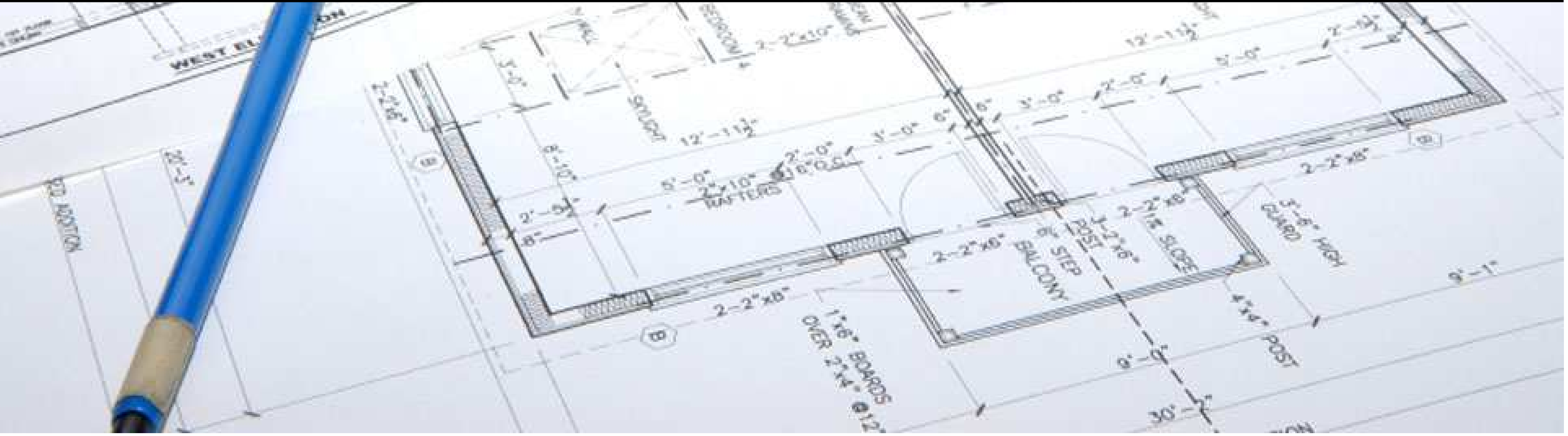
Generated Ant Build File Targets

```
$ant -p

Main targets:

appc                  Runs weblogic.appc on your application
build                 Compiles helloWorld application and runs appc
clean                 Deletes the build directory
compile               Only compiles helloWorld application, no appc
compile.appUtils      Compiles just the appUtils module of the application
compile.helloEJB      Compiles just the helloEJB module of the application
compile.helloWebApp   Compiles just the helloWebApp module of the application
config.server         Configure server with resources required by application
deploy                Deploys (and redeploys) the entire hellowWorld application
ear                   Package a standard JEE EAR for distribution
ear.exploded          Package a standard exploded JEE EAR
redeploy.appStartup   Redeploys just the appStartup module of the application
redeploy.appUtils     Redeploys just the appUtils module of the application
redeploy.helloEJB     Redeploys just the helloEJB module of the application
redeploy.helloWebApp  Redeploys just the helloWebApp module of the application
undeploy              Undeploys the entire helloWorld application
```

Executing Build, Package

```
$ ant ear
Buildfile: build.xml

compile:
 [javac] Compiling 1 source file to D:\splitdev-builds\helloWorld\APP-INF\classes
 [javac] Compiling 7 source files to D:\splitdev-builds\helloWorld\helloWorldWeb\WEB-INF\classes
 [javac] Compiling 3 source files to D:\splitdev-builds\helloWorld\helloWorldEJB\
 ...

appc:
 [wlappc] <24/11/2008 01:27:28 PM CST> <Info> <J2EE> <BEA-160186> <Compiling EAR module
   ‘helloWorldWeb'>
 [wlappc] [JspcInvoker]Checking web app for compliance.
 [wlappc] <24/11/2008 01:27:30 PM CST> <Info> <HTTP> <BEA-101047> <[Compliance Checker]
   Validating  the servlet element with servlet-name named “HelloWorldServlet".>
 ...
 [wlappc] <24/11/2008 01:27:30 PM CST> <Info> <HTTP> <BEA-101047> <[Compliance Checker] Checking
   servlet-mapping for servlet name : “HelloWorldServlet".>
 [wlappc] [jspc]  -webapp specified, searching . for JSPs
 [wlappc] [jspc] Compiling /index.jsp
 [wlappc] [jspc] Compiling /systemproperties.jsp
 [wlappc] Compilation completed successfully.

build:

ear:
     [jar] Building jar: D:\splitdev-builds\dist\helloWorld.ear
```

#5 Leverage WebLogic Server Ant Tasks



WebLogic Server Ant Tasks

- Easily integrate WebLogic Server operations into your Ant build scripts
- Tasks to start, stop, restart server
- Tasks to deploy, undeploy, redeploy
- Tasks to call out to execute WLST scripts

Ant Task Definition

```
<path id="WLS_CLASSPATH">
  <pathelement path="${WLS_HOME}/server/lib/wlfullclient.jar"/>
</path>
<property name="WLS_CLASSPATH" refid="WLS_CLASSPATH"/>

<taskdef name="wldeploy"
  classname="weblogic.ant.taskdefs.management.WLDeploy"
  classpathref="WLS_CLASSPATH"/>

<taskdef name="wlserver"
  classname="weblogic.ant.taskdefs.management.WLServer"
  classpathref="WLS_CLASSPATH"/>

<taskdef name="wlst"
  classname="weblogic.ant.taskdefs.management.WLSTTask"
  classpathref="WLS_CLASSPATH"/>
```

<wldeploy/>

- Deploy an application to the target server

```
<target name="server_deploy" depends="all">
  <wldeploy
    action="deploy"
    adminurl="${wls.adminurl}" targets="${wls.server}"
    user="${wls.user}" password="${wls.password}"
    source="${dist.dir}" name="${app.name}"
    verbose="${wls.verbose}" debug="${wls.debug}"/>
</target>
```

<wlserver/>

Create a new server definition and start it

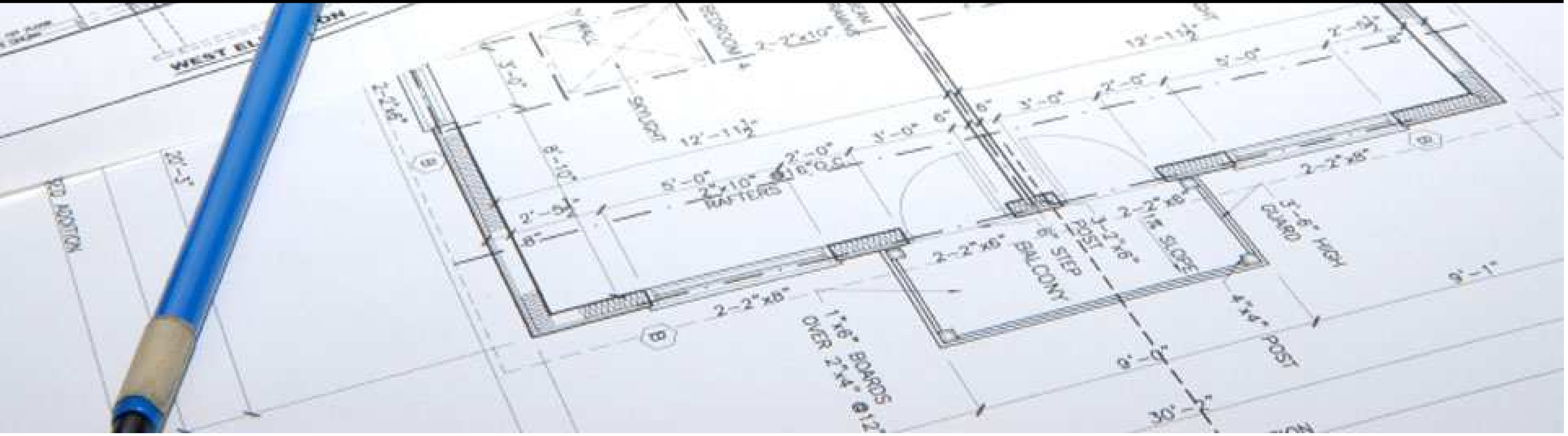
```
<target name="testserver_create">
  <delete dir="testserver”/>
  <mkdir dir="testserver"/>
  <wlserver dir="testserver" generateconfig="true“
          port="7001" action="start“
            servername="AdminServer“
          username="weblogic" password="weblogic“
          verbose="true"/>
</target>
```

<wlst/>

- Call a WLST script to populate a test instance

```
<target name="testserver_config" depends="testserver_create">
  <wlst debug="false" failOnError="false“
      executeScriptBeforeFile="true"
        fileName="${TEST_HOME}/config/testserver.py">
    <script>
      connect('weblogic','weblogic','t3://localhost:7001')
    </script>
  </wlst>
</target>
```

#6 Make Use of WebLogic Server Tooling



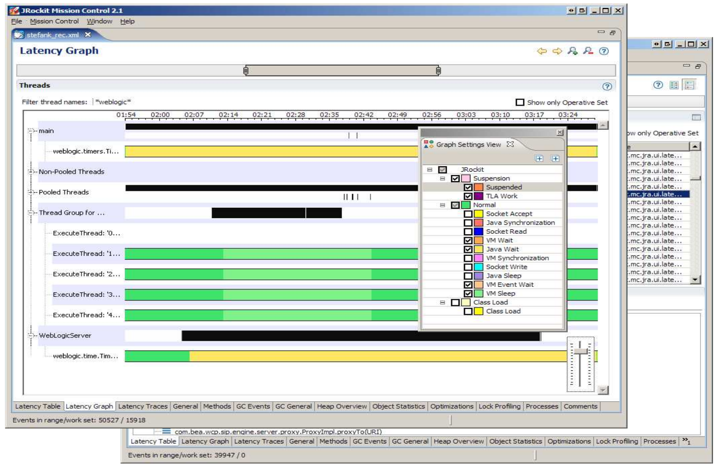
WebLogic Server Tooling

- WebLogic Server provides a wide array of helpful tooling to help developers
  - Incorporate into development process as necessary
- Set environment before executing
  - <DOMAIN_HOME>/bin/setDomainEnv.cmd

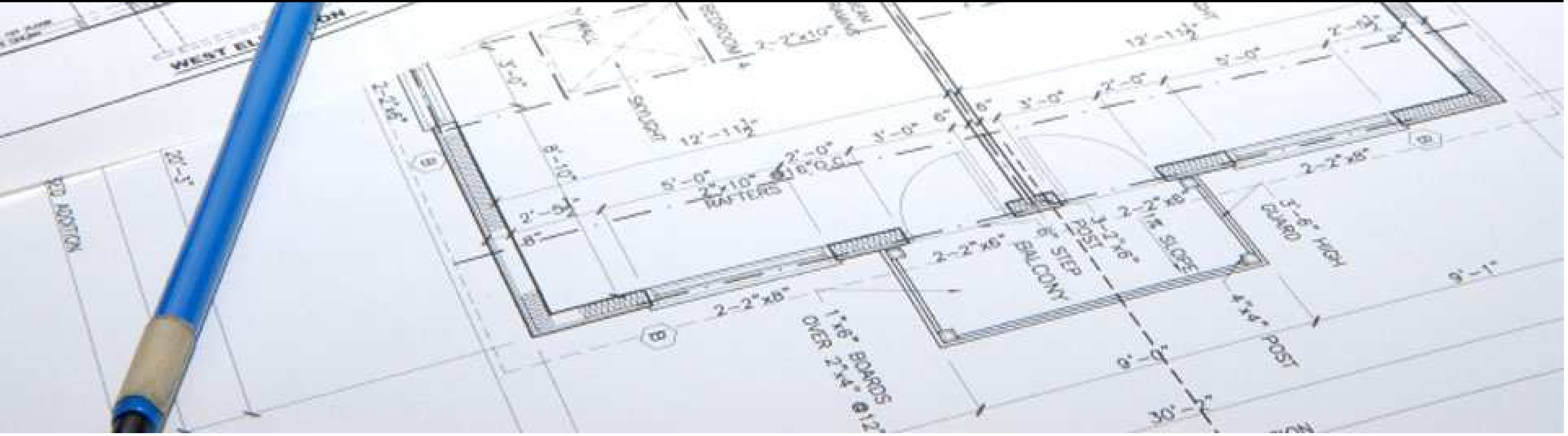
weblogic.appc	Compiles JSPs, EJB, validates deployment descriptors
weblogic.Deployer	Command line deployment utility
weblogic.PlanGenerator	Generates a template deployment plan for an application
weblogic.DDConverter	Convert deployment descriptors to current WLS version
weblogic.marathon.ddinit.EarInit	Generate EAR level deployment descriptors
weblogic.marathon.ddinit.WebInit	Generate Web module deployment descriptors

JRockit Mission Control

- An extension to JRockit which provides **profiling, monitoring, managing and diagnostics** of your Java applications at runtime
- Exposed through JRockit Mission Control GUI
  - JRockit Management Console
  - JRockit Runtime Analyzer (JRA)
  - Memory Leak Detector
  - Latency Analysis
- Integrated in the JVM
  - Near zero overhead
  - Available on-demand, no instrumentation needed



#7 Use Smart IDE Features

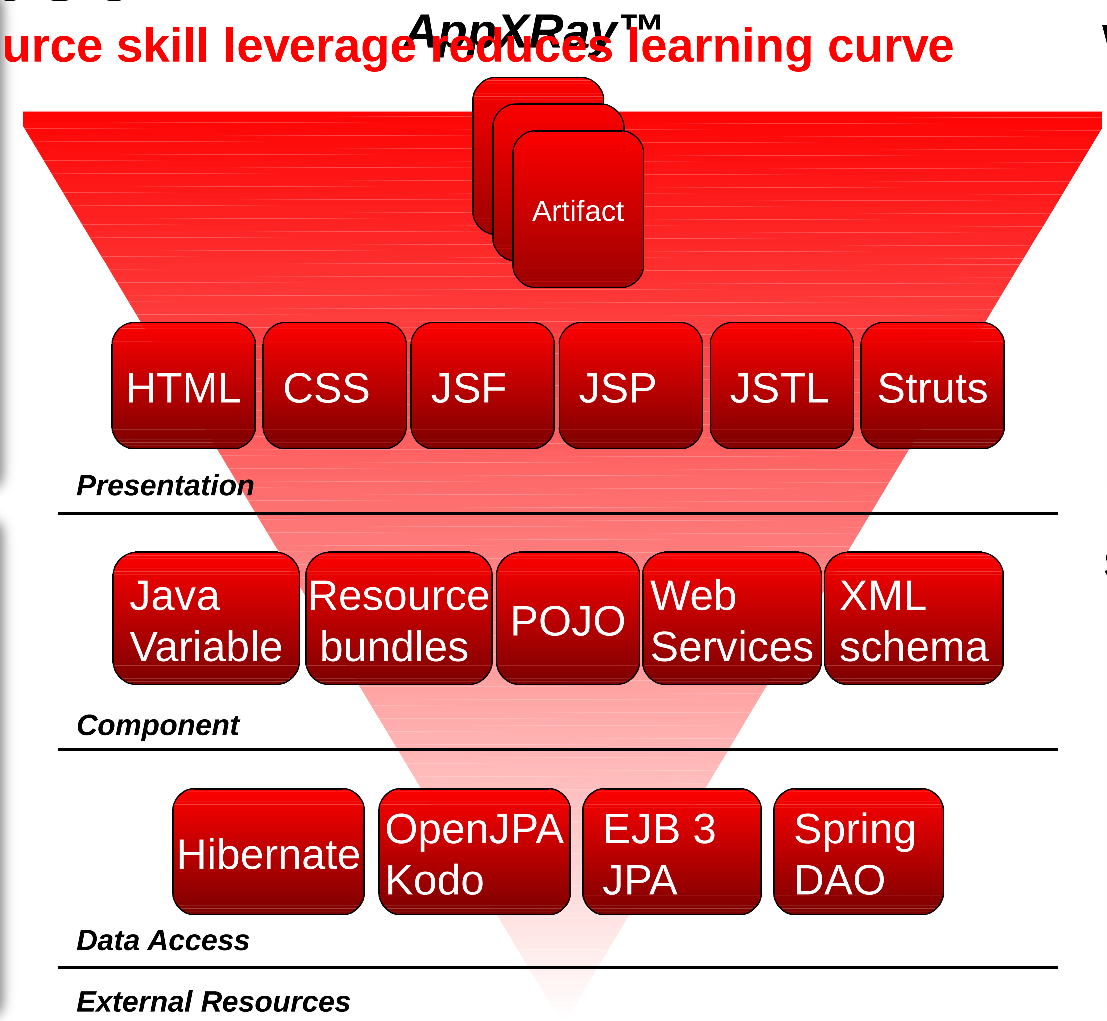


Oracle Enterprise Pack for Eclipse

Open Source skill leverage reduces learning curve

Java / Java EE

Web Services

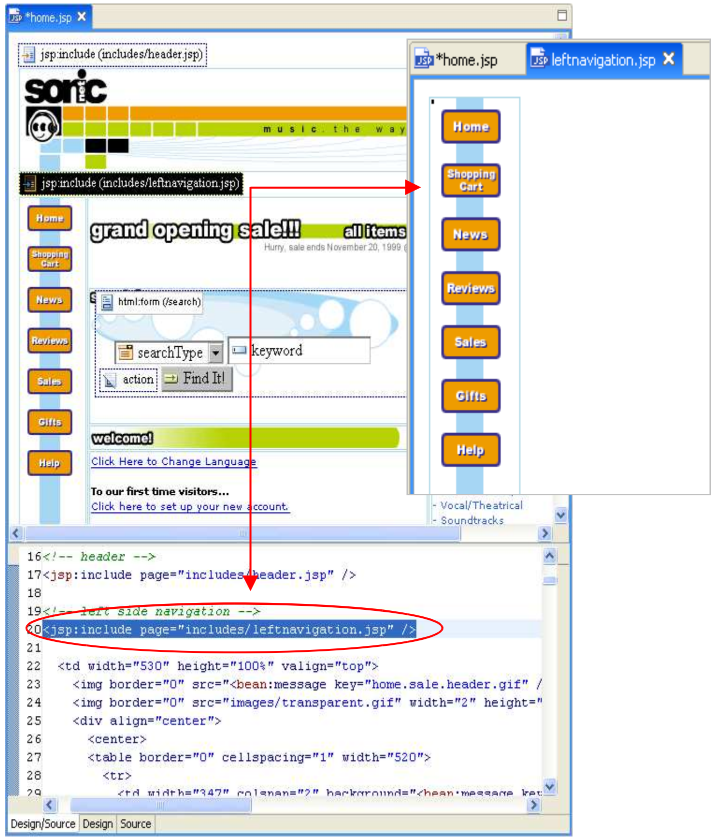


WebLogic Server

Spring, ORM, DB

OEPE Overview

- Classic Workshop Features
  - Based on Eclipse 3.4 + WTP 3.0
  - AppXRay
  - WYSIWYG for JSP, JSF, Struts
  - EJB 2 & EJB 3 tools
  - Database and ORM tools
  - Web Services (JAX-WS & JAX-RPC)
  - Spring Beans
  - Full support for WLS 10.4 and older versions
  - Support for other servers (Tomcat, JBoss, Websphere, Oracle, etc)
  - Upgrade
- Integration with existing Oracle Eclipse initiatives
  - Web Tools Web Page Designer
  - Web Tools JSF Project
  - Dali/DB Tools/EclipseLink



```
<!-- header -->
<jsp:include page="includes/header.jsp" />

<!-- left side navigation -->
<jsp:include page="includes/leftnavigation.jsp" />

  <td width="530" height="100%" valign="top">
    <img border="0" src="<bean:message key="home.sale.header.gif" /
    <img border="0" src="images/transparent.gif" width="2" height="
    <div align="center">
      <center>
      <table border="0" cellspacing="1" width="520">
        <tr>
          <td width="347" colspan="2" background="<bean:message key
```

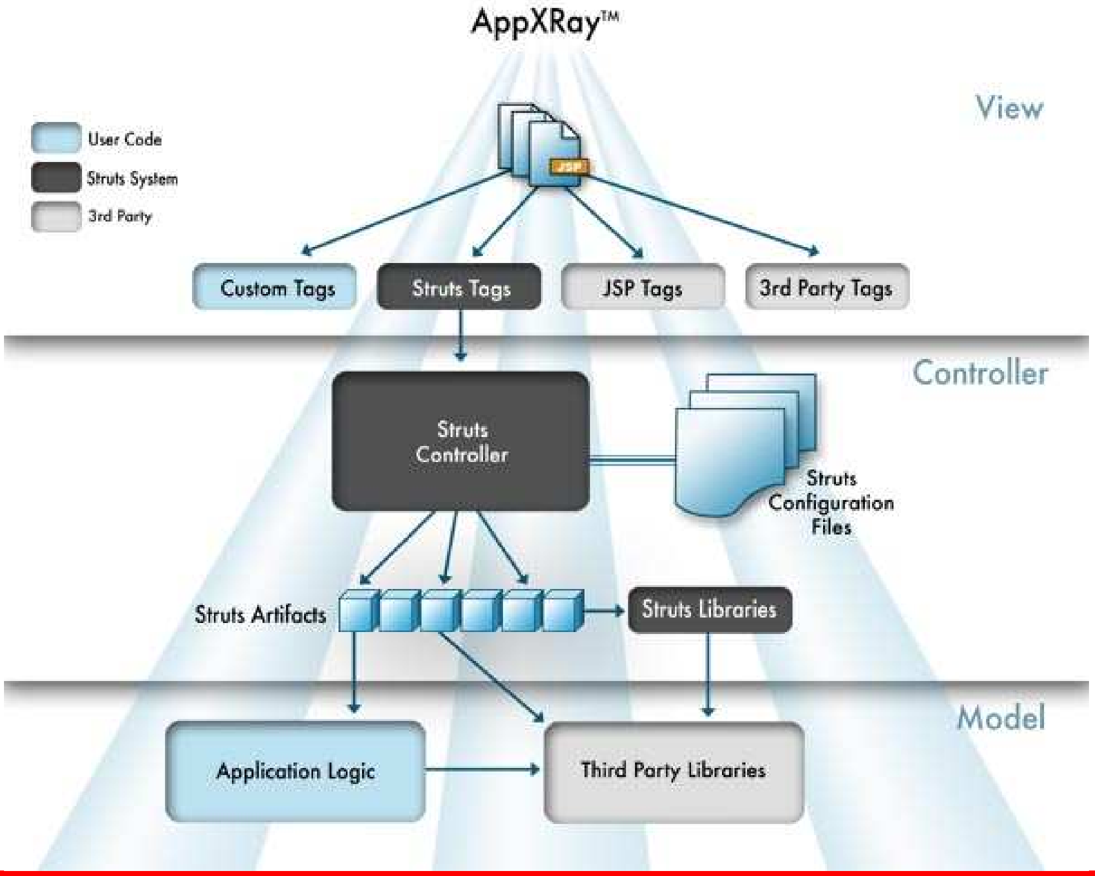
ORACLE

AppXRay™

Oracle’s AppXRay™ provides as-you type, compiler level awareness of much more than java at design time, offering unique capabilities in code and annotation completion, code navigation, dependency visualization, consistency checking with generated classes and configuration files, pre-build error checking, and validation that understands your entire application.

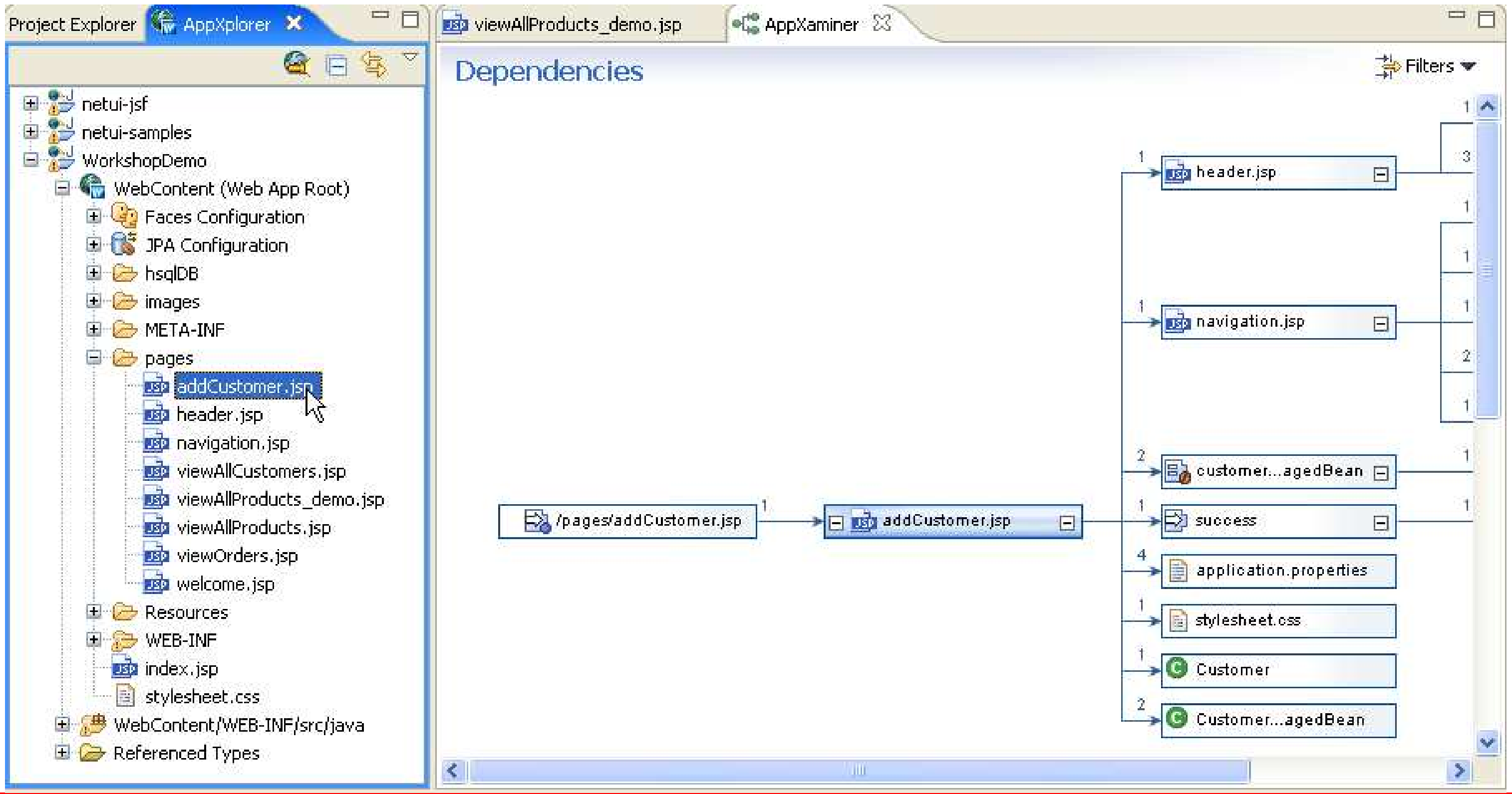
Provides design-time compiler awareness for:

- Java, HTML, CSS
- JSP/JSTL, Struts, Tiles, JSF
- EJB3, Oracle Kodo, Hibernate
- Java Resource Bundles, Variables



AppXRay: AppXaminer

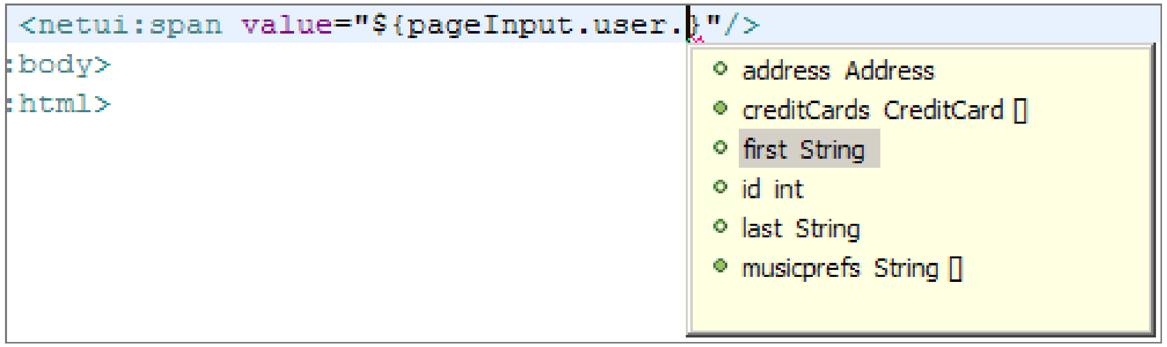
Developers who inherit code or applications developed by others will appreciate AppXaminer. View the relationships between **all** design time artifacts with a simple right click gesture, then filter out what you don't want to see. AppXaminer allows navigation through specific instances of dependences as well.



AppXRay Code Completion

Example: JSP Source Completion

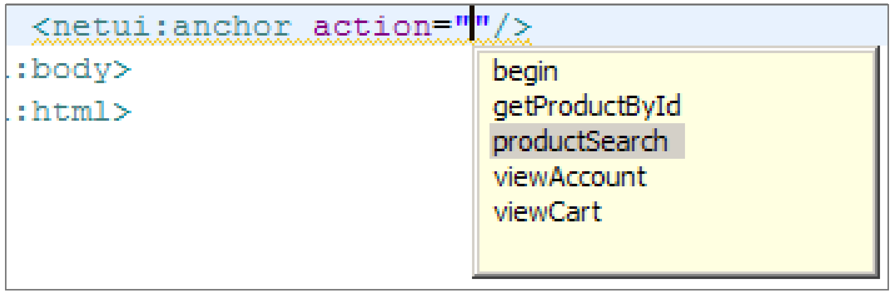
- Works for any framework supported by AppXRay
- Expression completion for JSP2.0, JSF EL



```
<netui:span value="${pageInput.user.}"/>
:body>
:html>
```

```
address  Address
creditCards  CreditCard []
first  String
id  int
last  String
musicprefs  String []
```

- Tag attribute values



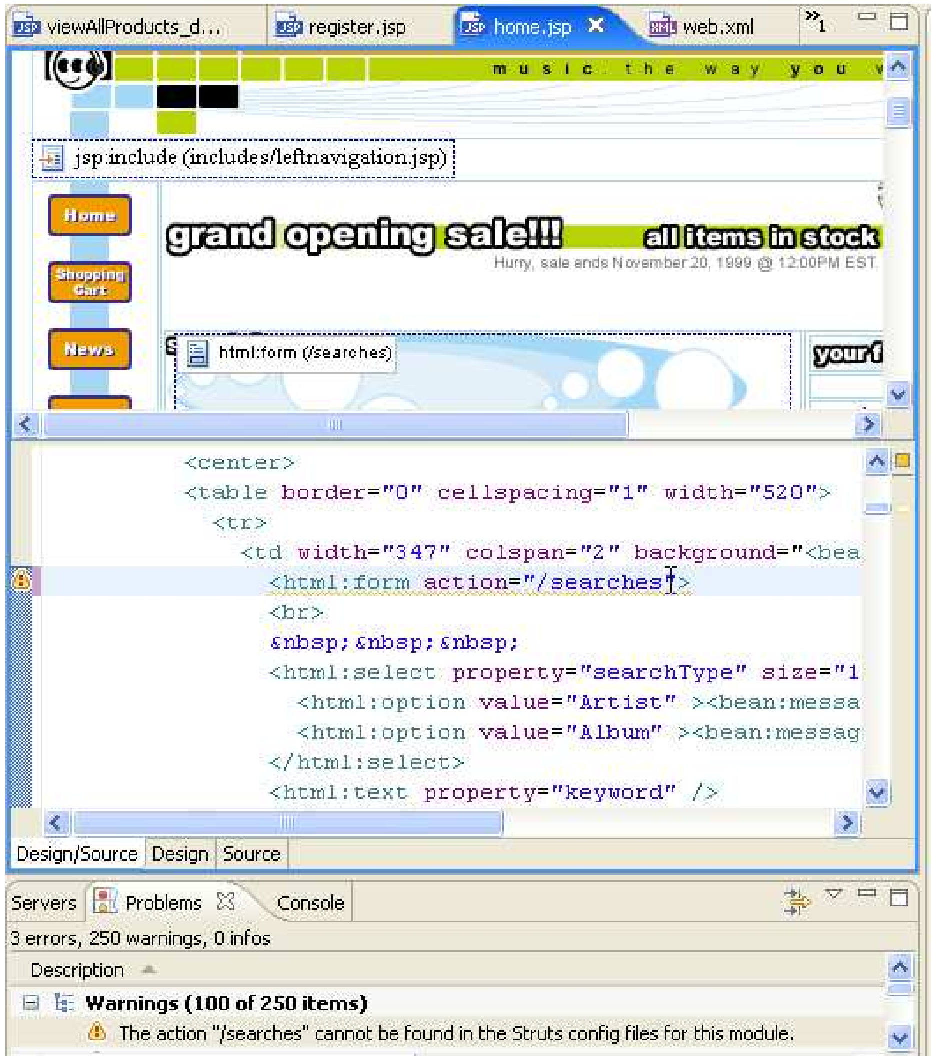
```
<netui:anchor action=""/>
.:body>
.:html>
```

```
begin
getProductById
productSearch
viewAccount
viewCart
```

AppXRay Validation

Example: Struts Validation

- Syntax checking for any framework AppXRay supports
  - Tag usage in pages
  - Code syntax against spec (eg. EJB3, JSF)
  - Validation against current framework configuration file
- Validation on any dependencies between framework related artifacts



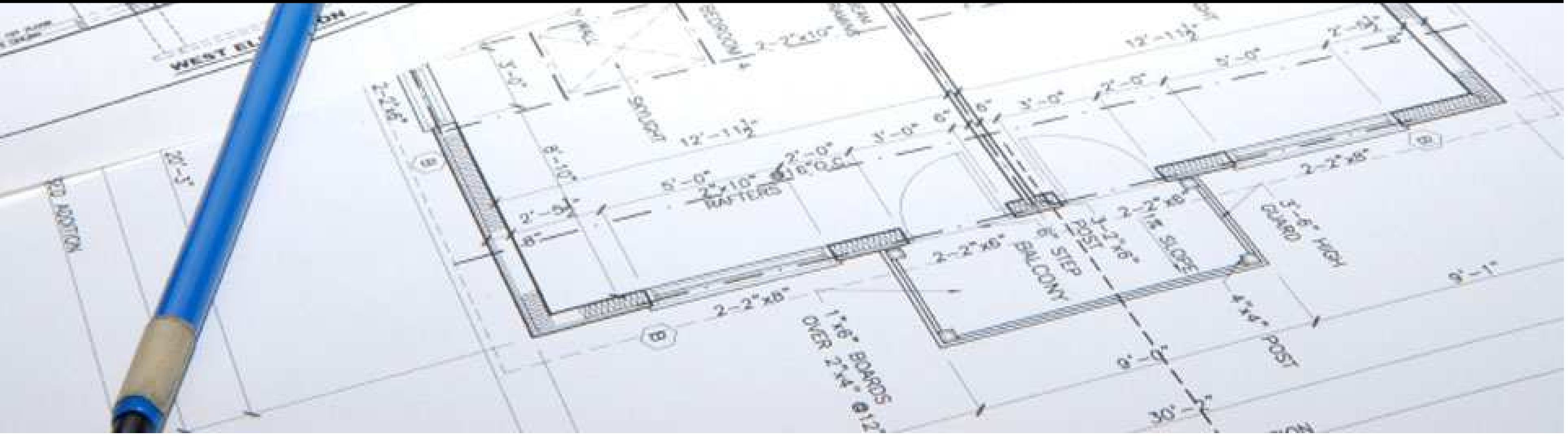
```
<center>
<table border="0" cellspacing="1" width="520">
  <tr>
    <td width="347" colspan="2" background="<bea
      <html:form action="/searches">
      <br>
      &nbsp;&nbsp;&nbsp;
      <html:select property="searchType" size="1
        <html:option value="Artist" ><bean:messa
        <html:option value="Album" ><bean:messag
      </html:select>
      <html:text property="keyword" />
```

3 errors, 250 warnings, 0 infos

Warnings (100 of 250 items)

The action "/searches" cannot be found in the Struts config files for this module.

#8 Automate Creation of Dev/Test Environments

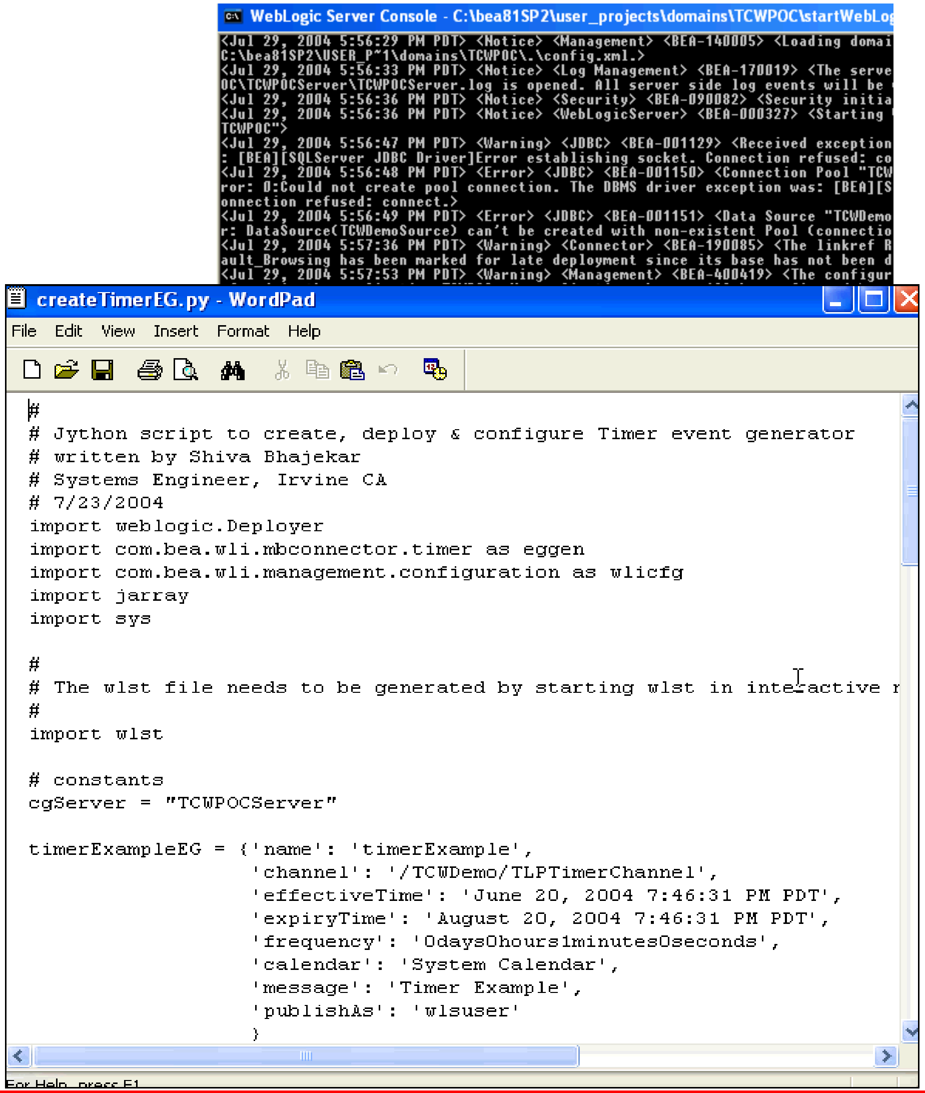


Automation of Dev/Test Environments

- Creating development, integration and general test servers can be a burden
  - Need to create domain(s)
  - Need to populate it with resources (jdbc, jms, libraries)
  - Need to be able to reset it and recreate it
- A manual approach leads to errors, inconsistencies
  - Not easy to guarantee outcome
- Requires a a repeatable, automatable solution
  - Executable from command line, Ant via <wlst> , continuous integration server product
- WebLogic Server provides a scripting solution to service this need with WLST

WebLogic Scripting Tool (WLST)

- Command-line scripting interface for managing and monitoring WebLogic Server instances, clusters, and domains
- Based on 100% pure Java implementation of Python
- Modes of operation – (script / interactive) ↔ (online / offline)
- Provides a repeatable process for propagating configuration changes across environments
- Quickly allows environments to be replicated and migrated


WebLogic Server Console - C:\bea81SP2\user_projects\domains\TCWPOC\startWebLog
createTimerEG.py - WordPad
File Edit View Insert Format Help
For Help, press F1

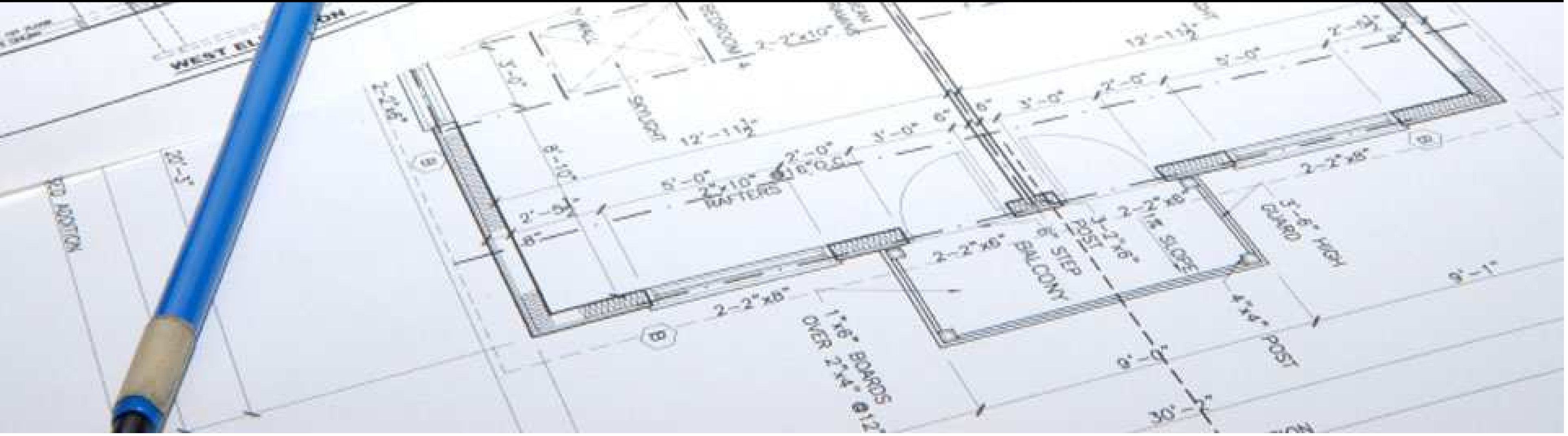
```
<Jul 29, 2004 5:56:29 PM PDT> <Notice> <Management> <BEA-140005> <Loading domai
C:\bea81SP2\USER_P~1\domains\TCWPOC\.\config.xml.>
<Jul 29, 2004 5:56:33 PM PDT> <Notice> <Log Management> <BEA-170019> <The serve
OC\TCWPOCServer\TCWPOCServer.log is opened. All server side log events will be
<Jul 29, 2004 5:56:36 PM PDT> <Notice> <Security> <BEA-090082> <Security initia
<Jul 29, 2004 5:56:36 PM PDT> <Notice> <WebLogicServer> <BEA-000327> <Starting
TCWPOC">
<Jul 29, 2004 5:56:47 PM PDT> <Warning> <JDBC> <BEA-001129> <Received exception
: [BEA][SQLServer JDBC Driver]Error establishing socket. Connection refused: co
<Jul 29, 2004 5:56:48 PM PDT> <Error> <JDBC> <BEA-001150> <Connection Pool "TCW
ror: 0:Could not create pool connection. The DBMS driver exception was: [BEA][S
onnection refused: connect.>
<Jul 29, 2004 5:56:49 PM PDT> <Error> <JDBC> <BEA-001151> <Data Source "TCWDemo
r: DataSource(TCWDemoSource) can't be created with non-existent Pool (connectio
<Jul 29, 2004 5:57:36 PM PDT> <Warning> <Connector> <BEA-190085> <The linkref R
ault_Browsing has been marked for late deployment since its base has not been d
<Jul 29, 2004 5:57:53 PM PDT> <Warning> <Management> <BEA-400419> <The configur
```

```
#
# Jython script to create, deploy & configure Timer event generator
# written by Shiva Bhajekar
# Systems Engineer, Irvine CA
# 7/23/2004
import weblogic.Deployer
import com.bea.wli.mbconnector.timer as eggen
import com.bea.wli.management.configuration as wlicfg
import jarray
import sys

#
# The wlst file needs to be generated by starting wlst in interactive r
#
import wlst

# constants
cgServer = "TCWPOCServer"

timerExampleEG = {'name': 'timerExample',
                  'channel': '/TCWDemo/TLPTimerChannel',
                  'effectiveTime': 'June 20, 2004 7:46:31 PM PDT',
                  'expiryTime': 'August 20, 2004 7:46:31 PM PDT',
                  'frequency': '0days0hours1minutes0seconds',
                  'calendar': 'System Calendar',
                  'message': 'Timer Example',
                  'publishAs': 'wlsuser'
                  }
```

Resource Creation with WLST

```
# Database configuration
dataSources = splitMap(env, "datasource.")

for (dataSourceName, properties) in dataSources.items():
  cd("/")
  jdbcSystemResource = JDBCSystemResource(dataSourceName)
  dataSources[dataSourceName] = jdbcSystemResource
cd("JDBCSystemResource/%s" % jdbcSystemResource.name)
  cd("JdbcResource/%s" % jdbcSystemResource.name)

  JDBCDriverParams("ignored", driverName = "oracle.jdbc.OracleDriver",
                   passwordEncrypted = properties["password"],
                   url = properties["url"])

 JDBCConnectionPoolParams("ignored", initialCapacity = 1,
 maxCapacity = int(properties["capacity"]),
   secondsToTrustAnIdlePoolConnection = 10,
   shrinkFrequencySeconds = 0, statementCacheSize = 100,
   testConnectionsOnReserve = 1, testFrequencySeconds = 0,
   testTableName = "SQL SELECT 1 FROM dual WHERE 1=2")
```

#9 Application Class Loading



Application Class Loading

- As a developer you are responsible for developing and assembling applications
  - Make use of many sources of code/libraries within applications
  - In-house libraries + Spring, Xerces, Log4J, apache-commons-*, …
- Understanding how class loading works is important to make correct and efficient use of Java libraries
  - What classes get loaded from where
  - Efficiently reusing shared libraries
  - Avoiding ClassCastExceptions
- WebLogic Server class loading is a powerful mechanism that can be used to good effect
  - Reuse of common shared libraries
  - Filtering Classloader to control library visibility
  - Construction of custom module classloader hierarchies

Application Class Paths

- EAR application classpath
  - APP-INF/classes/
  - APP-INF/lib/*.jar
  - Manifest classpath
  - (EAR-library-classpath)*
  - (JAR-library-classpath)*
- WAR application classpath
  - WEB-INF/classes/
  - WEB-INF/*.jar
  - Manifest classpath
  - (WAR-library-classpath)*
  - (JAR-library-classpath)*
  - (EAR-Application-classpath)

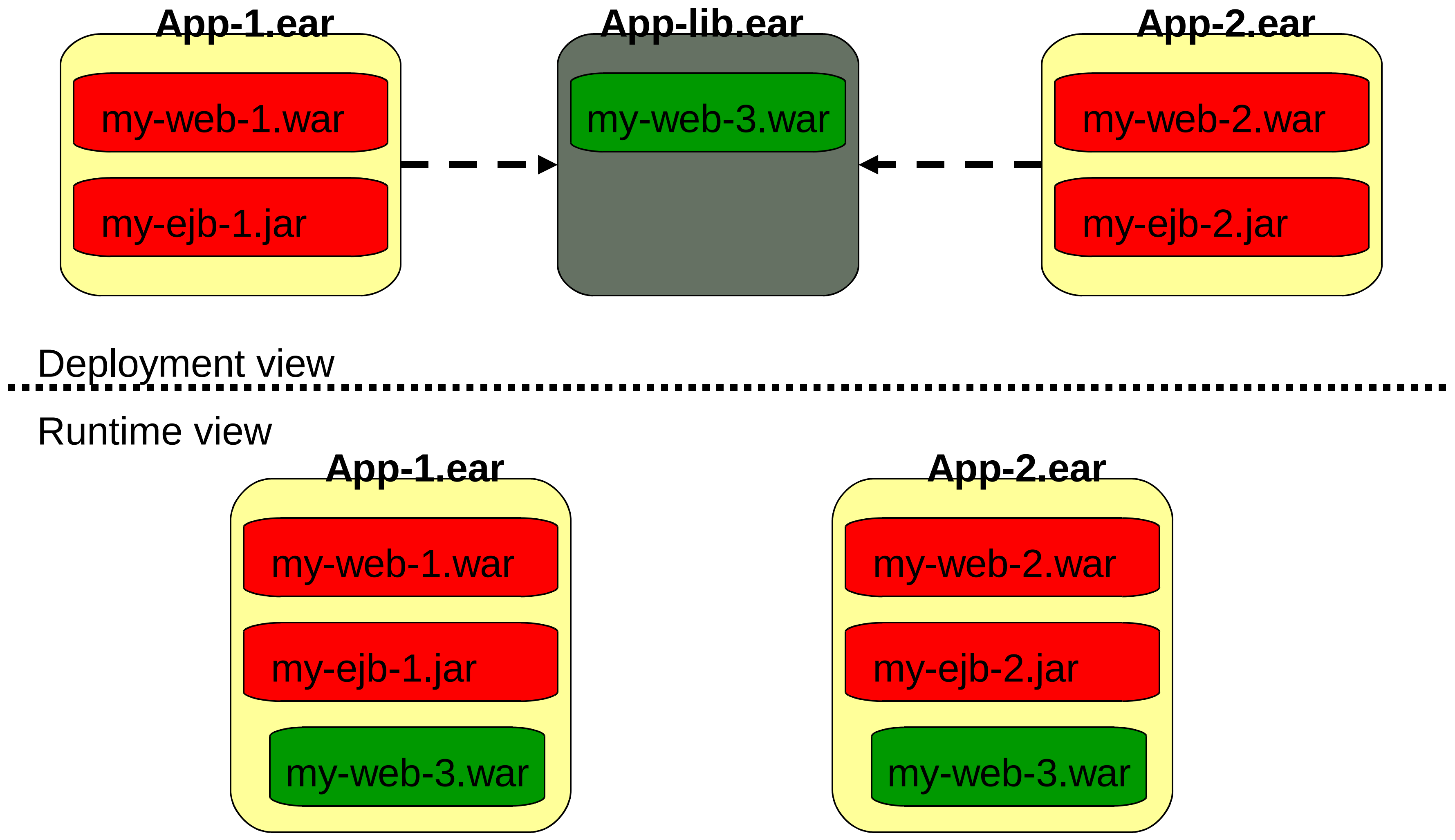
Ways of Sharing Libraries

- System Classpath
  - $DOMAIN/lib automatically added to classpath
  - Modify Claspath setting in setDomainEnv, commEnv
- Application level
  - APP-INF/lib for packaged libraries
  - APP-INF/classes for unpackaged classes
  - META-INF/Manifest.mf/Class-Path
- Java EE Libraries
  - Deploy reusable modules as libraries
  - Reference libraries (name, version) in deployment descriptors

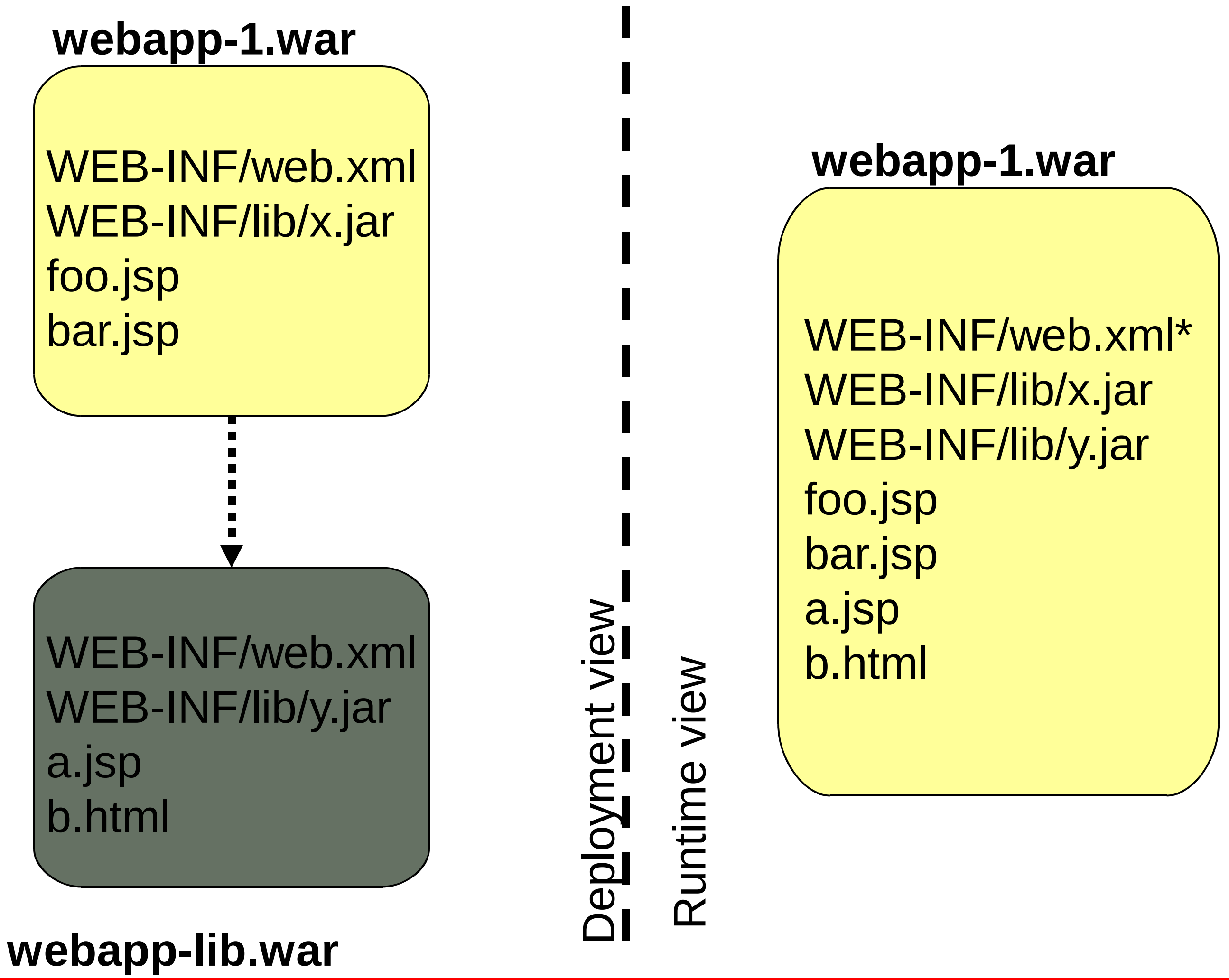
Best Practices for Sharing Libraries

- System classpath
  - Need to be visible to all/many applications
  - Reloading of library is not common
- Application level
  - Library not shared with any other application
  - Reloading of library is required
- Deployed library
  - Need to be shared among many applications
  - Library evolves differently than application classes
  - Reloading of library is required

Application Shared Libraries



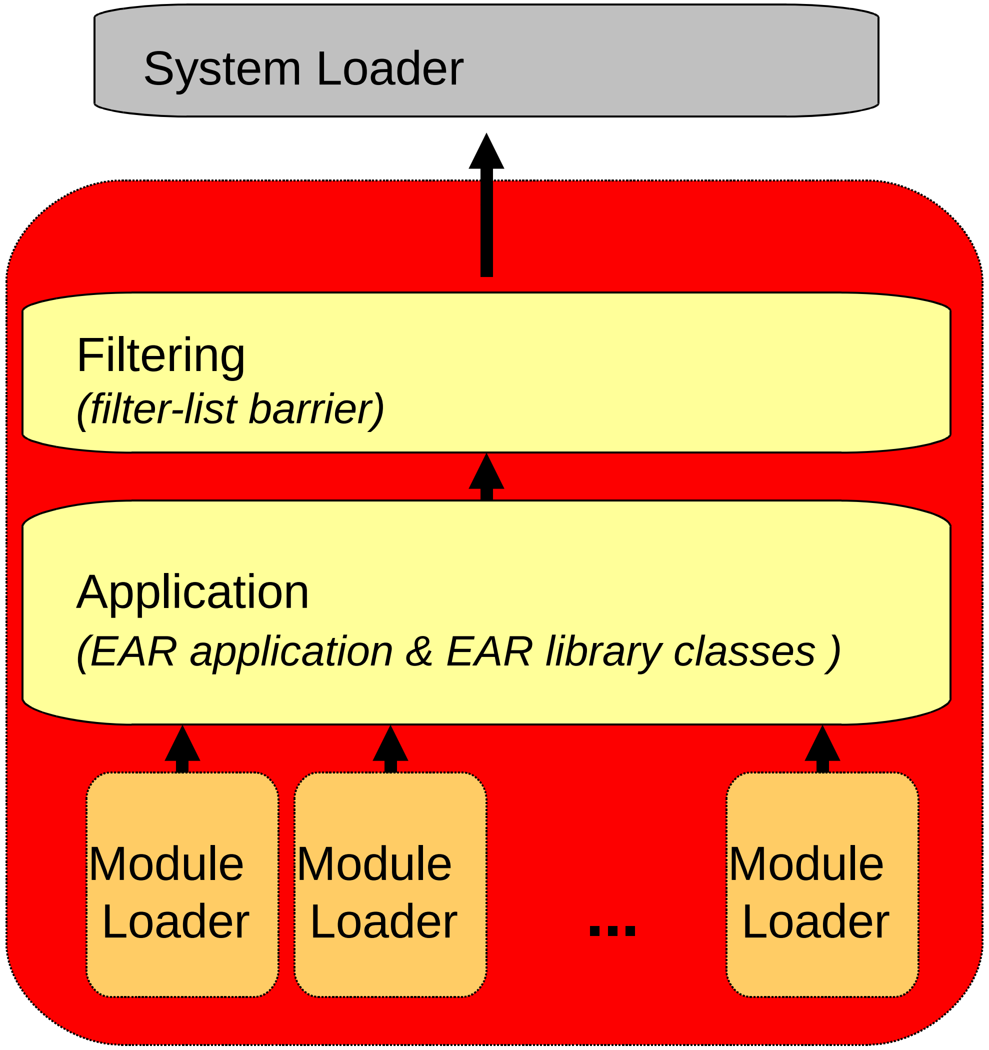
Web-App Shared Libraries



Filtering Classloader

- Enables classes to be loaded from an application supplied library first
  - Changes the delegation model from parent to child first
  - Works as a barrier to prevent parent from supplying class
  - Does not load classes itself
- Useful in scenarios where application needs to use a different version of a framework that is already bundled with the server
  - Xerces, Spring, Ant, Commons-Logging, etc.

Filtering Classloader Hierarchy



- *Filtering classloader sits between Application and System classloaders*
- *Does not load classes itself*
- *Prevents classes from being loaded from system if they match the filter-list*
- *Returns **ClassNotFoundException** from the parent so child assumes loading duties*

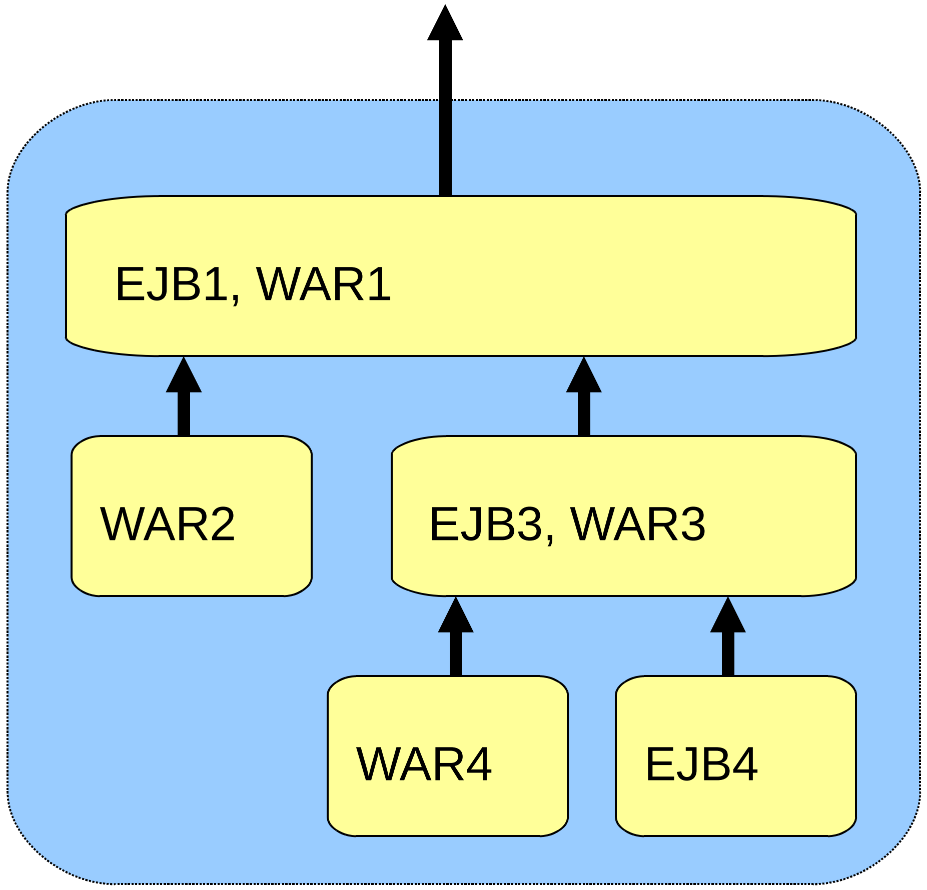
Filtering Classloader Configuration

- List of packages to load from the application is specified in application level deployment descriptor

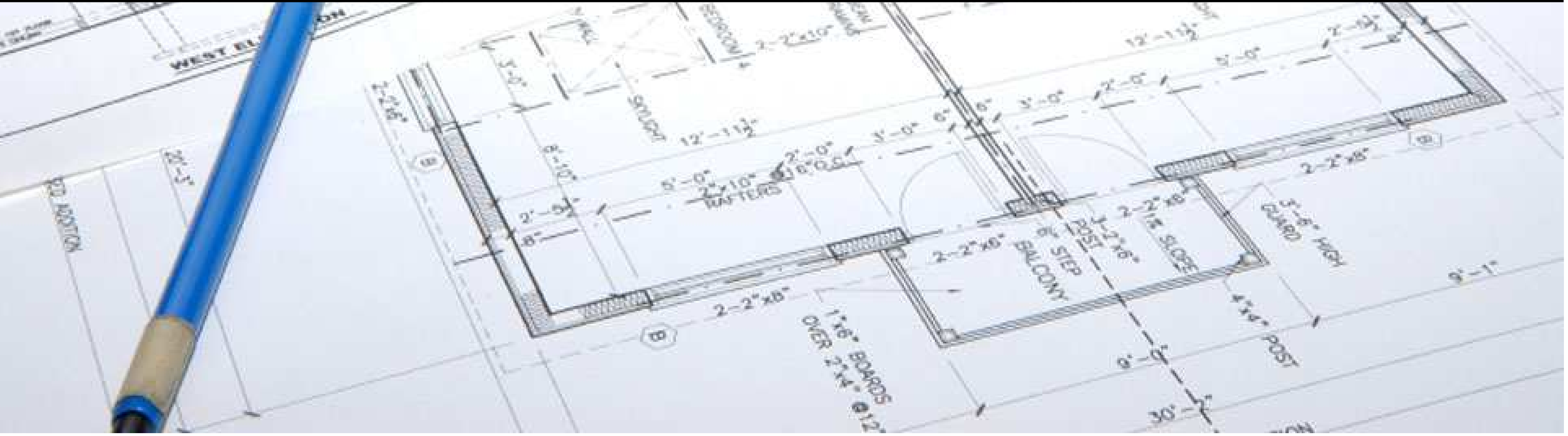
```
<weblogic-application>
  ...
  <prefer-application-packages>
    <package-name>org.apache.xerces.*</package-name>
    <package-name>org.apache.commons.*</package-name>
    <package-name>org.apache.log4j.*</package-name>
  </prefer-application-packages>
  ...
</weblogic-application>
```

Custom Classloader Hierarchies

- Custom classloader hierachies can be constructed
  - Change the hierarchy of classloaders in an application(EAR)
  - Control inter-module visibility
- Modules in an EAR can be organized for flexibility in module redeployments
- Specified in weblogic-application.xml using <classloader-structure> element



#10 Application Logging



Applications Using java.util.logging

- Application level logging using java.util.logging is widespread
  - Can configure standard Handlers to handle them from the command line
  - -Djava.util.logging.config.file
- WebLogic Server does not automatically handle log messages from application loggers
  - WebLogic Server creates its own Root level logger
- To direct application logs, craft a custom Handler
  - Get reference to WLS server logger and log messages there

Java Logging Handler for WLS

```
package sab.demo.fastswap.logging;

import java.util.logging.*;

public class WLSServerHandler extends Handler {
  final Logger wlsLogger =
weblogic.logging.LoggingHelper.getServerLogger();

  @Override
  public synchronized void setLevel(java.util.logging.Level newLevel)
    throws SecurityException {
    super.setLevel(newLevel);
  }

  @Override
  public void publish(LogRecord record) {
    // Push record into WLS Server Logger
    wlsLogger.log(record);
  }
}
```

Configuring Logging Handler

- Configure logging properties in logging.properties file

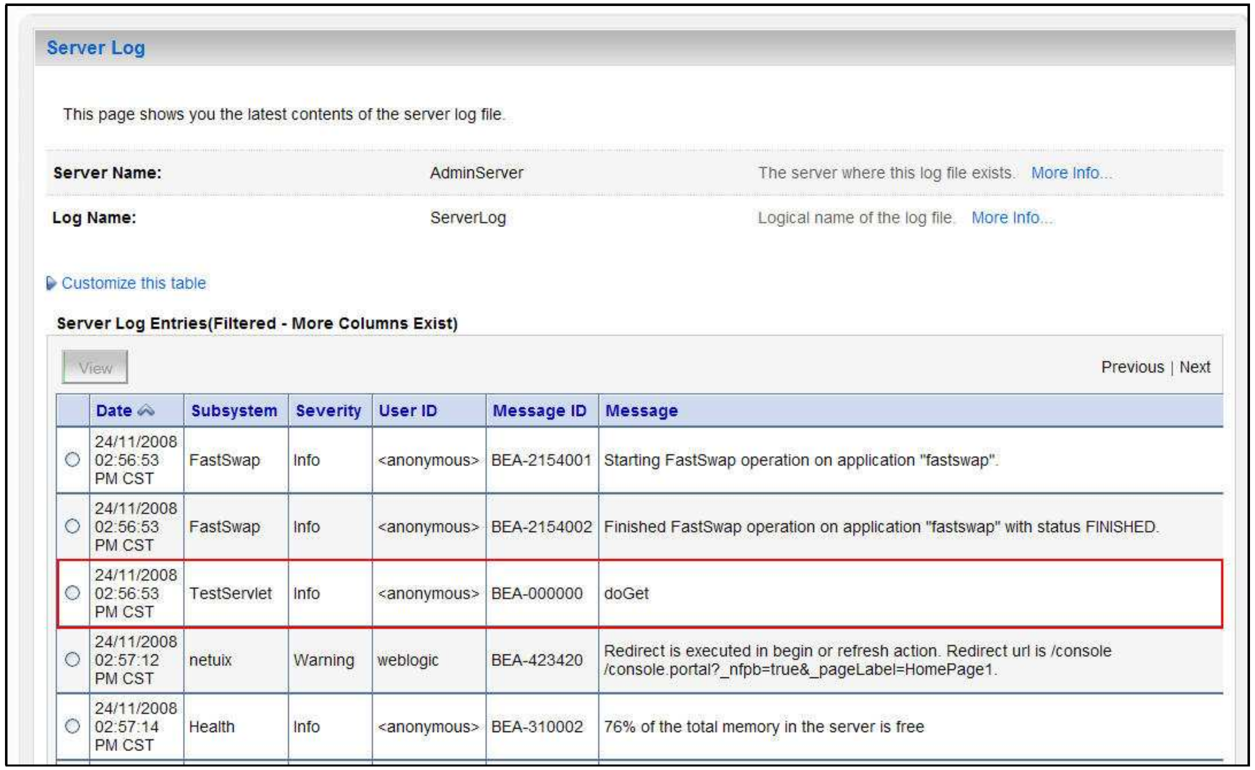
```
handlers=sab.demo.fastswap.logging.WLSServerHandler
sab.demo.fastswap.logging.WLSServerHandler.level=FINER
TestServlet.level=FINER
CalculatorBean.level=FINER
```

Specify logging.properties file in setDomainEnv.cmd

```
set JAVA_OPTIONS=%JAVA_OPTIONS%
-Djava.util.logging.config.file=d:\bea\user_projects\domians\research\wls-
logging.properties
```

Console Log Viewer

- Log messages written to server.log of domain
- Tail the log file or view in console



This page shows you the latest contents of the server log file.

Server Name:	AdminServer	The server where this log file exists. More Info...
Log Name:	ServerLog	Logical name of the log file. More Info...

Server Log Entries(Filtered - More Columns Exist)

	Date	Subsystem	Severity	User ID	Message ID	Message
○	24/11/2008 02:56:53 PM CST	FastSwap	Info	<anonymous>	BEA-2154001	Starting FastSwap operation on application "fastswap".
○	24/11/2008 02:56:53 PM CST	FastSwap	Info	<anonymous>	BEA-2154002	Finished FastSwap operation on application "fastswap" with status FINISHED.
○	24/11/2008 02:56:53 PM CST	TestServlet	Info	<anonymous>	BEA-000000	doGet
○	24/11/2008 02:57:12 PM CST	netuix	Warning	weblogic	BEA-423420	Redirect is executed in begin or refresh action. Redirect url is /console /console.portal?_nfpb=true&_pageLabel=HomePage1.
○	24/11/2008 02:57:14 PM CST	Health	Info	<anonymous>	BEA-310002	76% of the total memory in the server is free

Summary

- WebLogic Server provides many conveniences to developers
- Highly reduced dev and test cycles with FastSwap`
- Easy integration with Ant build environments
- Repeatable dev and test environment creation
- Advanced and highly configurable class loader options

More Resources

- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/api.html
- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/programming/index.html
- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/programming/splitcreate.html
- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/config_scripting/index.html