JavaOne℠

Sun's 2000 Worldwide Java Developer Conference℠

JavaOne℠
Sun's 2000 Worldwide Java Developer Conference℠

Remote Debugging of Server Side Java™ Technology

Elizabeth Vinsel Looney
Principal Member of Technical Staff
Oracle Corporation

Raghu Kodali
Senior Product Manager
Oracle Corporation

Agenda

- Server-side Java debugging techniques
- Java Platform Debugger Architecture
- Apache JServ
  - Java Servlet API
  - JavaServer Pages™ technology
- BEA WebLogic Server
  - Java Servlet API
  - Enterprise JavaBeans™ technology
  - JavaServer Pages technology

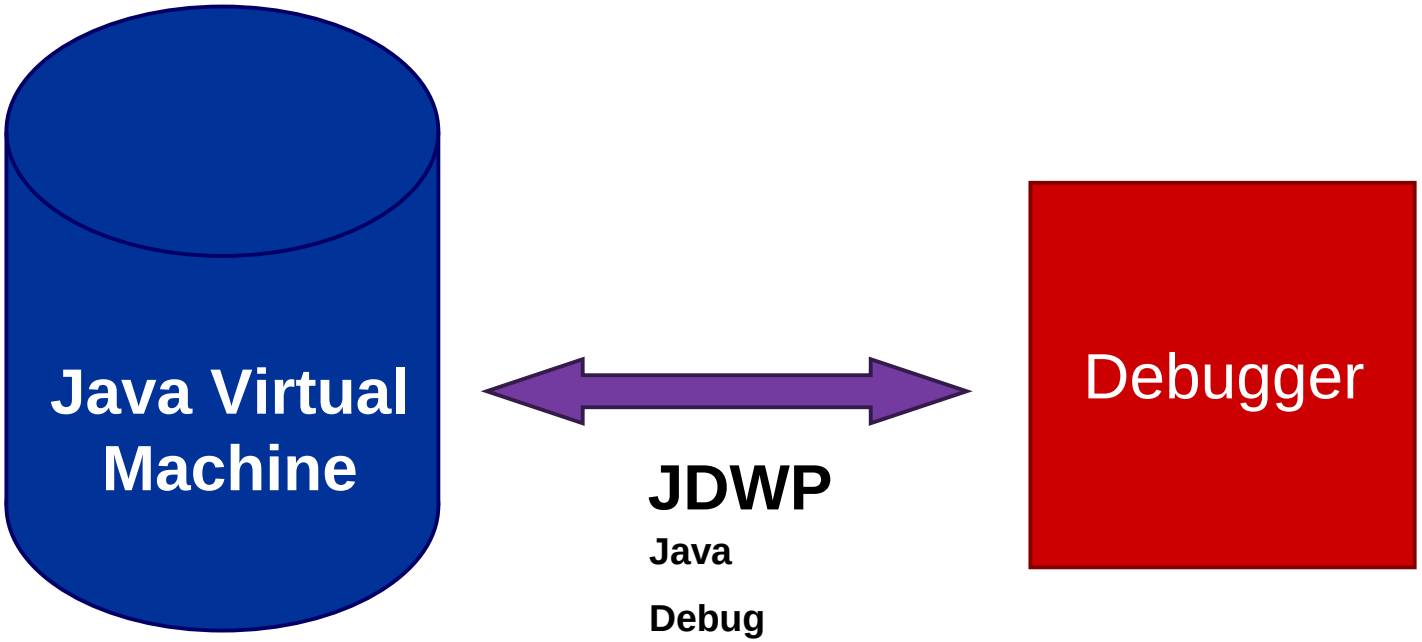
Ways to Debug Server Side Java™ Technology

- Add System.out.println statements
- Add fake application entry points
- Do local debugging within an IDE
- Do remote debugging with a real server

Key Features for Remote Debugging Server Side Java Technology

- Ability to start a process running with debugging enabled, but not suspended
- Ability to attach the debugger to the running process
- Ability to detach the debugger without terminating the process

Java™ Platform Debugger Architecture (JPDA)



Configuring the Server

- Java™ technology-based Servlets ("Java Servlets) and Enterprise JavaBeans™ architecture-based components
  - Specify which VM for the Java™ platform ("Java virtual machine")
  - Specify debug arguments
- JavaServer Pages™ API (JSP™)
  - All of the above
  - Specify classpath
  - Specify JSP engine
  - Specify parameters for JSP engine

Configuring Apache JServ—Java Servlet API

- Specify which Java virtual machine
  - In jserv.properties, set wrapper.bin:

```
# The Java Virtual Machine interpreter.
# Syntax: wrapper.bin=[filename] (String)
# Note: specify a full path if the interpreter is not
# visible in your path.
wrapper.bin=c:\jdk1.3\bin\java.exe
```

Configuring Apache JServ—Java Servlet API

- Specify debug arguments
  - In jserv.properties, set wrapper.bin.parameters:

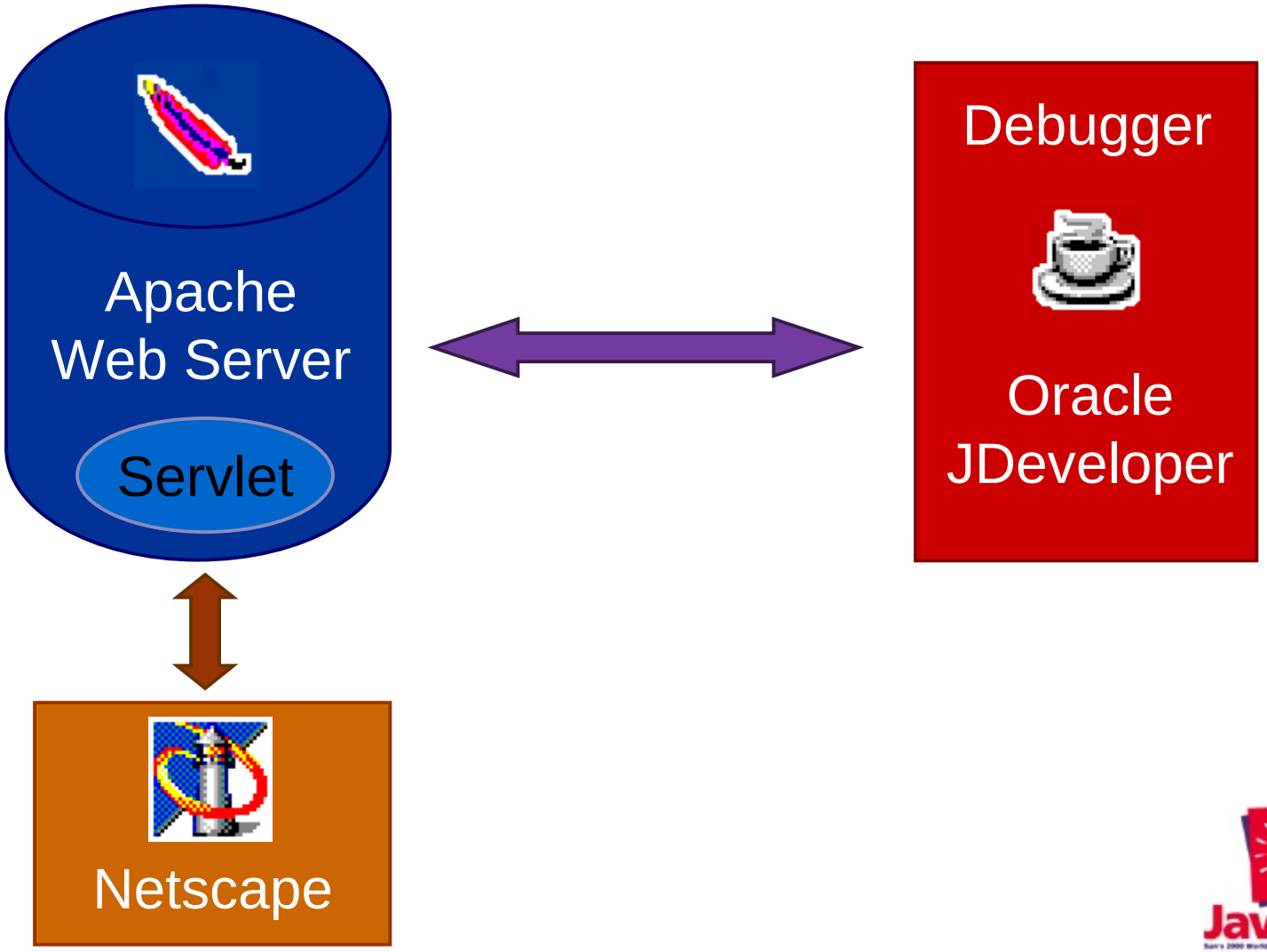
```
# Arguments passed to Java interpreter (optional)
# Syntax: wrapper.bin.parameters=[parameters] (String)
# Default: NONE
wrapper.bin.parameters=-classic
wrapper.bin.parameters=-Xdebug
wrapper.bin.parameters=-Xnoagent
wrapper.bin.parameters=-Djava.compiler=NONE
wrapper.bin.parameters=-Xrunjdwp:transport=dt_socket,
    server=y,suspend=n,address=5000
```

Configuring Apache JServ—Java Servlet API

- Add the servlet we want to debug
  - Put our servlet classes in the repository as specified in zone.properties:

```
# The list of servlet repositories controlled by this
# servlet zone
# Syntax: repositories=[repository],[repository]...
# Default: NONE
# Note: The classes you want to be reloaded upon
#       modification should be put here.
repositories=C:\Program Files\Apache JServ 1.1\servlets
```

Demo Apache Web Server—Java Servlet API



Configuring Apache JServ—JavaServer Pages™ API

- Specify classpath
  - In jserv.properties, set wrapper.classpath:

```
# CLASSPATH environment value passed to the JVM
# Syntax: wrapper.classpath=[path] (String)
wrapper.classpath=C:\Program Files\Apache JServ 1.1\
     ApacheJServ.jar
wrapper.classpath=C:\jsdk2.0\lib\jsdk.jar
wrapper.classpath=C:\Program Files\Oracle\JDeveloper 3.1\
     lib\ojsp.jar
wrapper.classpath=C:\Program Files\Oracle\JDeveloper 3.1\
     jswdk-1.0.1\lib\servlet.jar
wrapper.classpath=C:\Program Files\Oracle\JDeveloper 3.1\
     lib\ojc.jar
```

Configuring Apache JServ—JavaServer Pages API

- Specify servlet for receiving requests for JSP technology-based pages (“JSP engine”)
  - In jserv.conf, set ApJServAction .jsp:

```
# Executes a servlet passing filename with proper
# extension in PATH_TRANSLATED property of servlet
# request.
# Syntax: ApJServAction [extension] [servlet-uri]
# Defaults: NONE
# Notes: This is used for external tools.
ApJServAction .jsp /servlet/oracle.jsp.JspServlet
```

Configuring Apache JServ—JavaServer Pages API

- Specify parameters for JSP engine
  - In zone.properties:

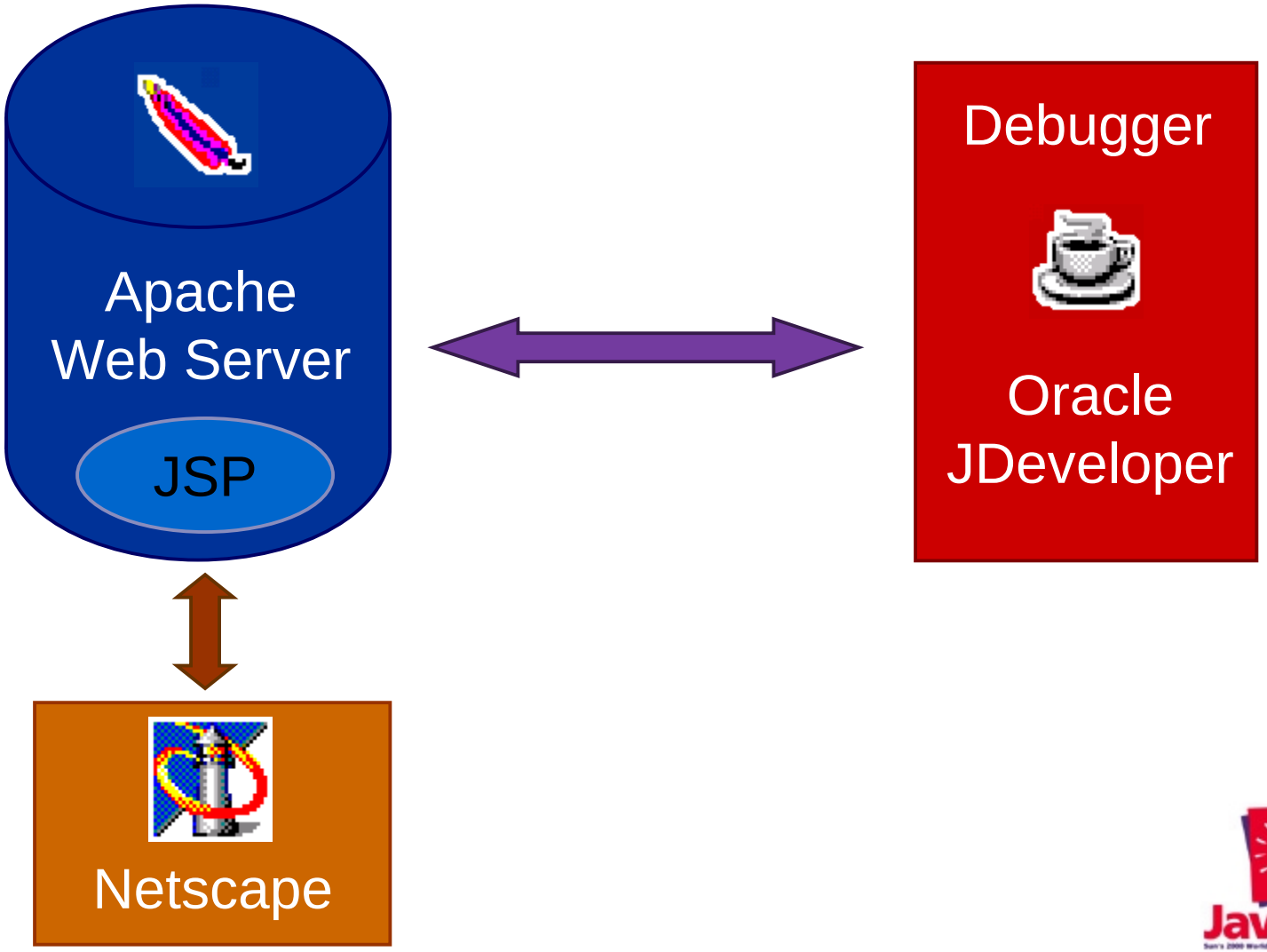
```
# Servlet Init Parameters
############################
# These properties define init parameters for each
# servlet that is invoked by its classname.
# Syntax: servlet.[classname].initArgs=[name]=[value],
# [name]=[value],...
# Default: NONE
servlet.oracle.jsp.JspServlet.initArgs=emit_debuginfo=true
servlet.oracle.jsp.JspServlet.initArgs=jspcompiler=
     oracle.jdeveloper.jsp.JspOjcCompiler
servlet.oracle.jsp.JspServlet.initArgs=classpath=
     c:\jdk1.3\jre\lib\rt.jar
```

Configuring Apache JServ—JavaServer Pages API

- Add the JSP™ component we want to debug
  - Put our JSP component in the document root as specified in httpd.conf:

```
# DocumentRoot: The directory out of which you will
# serve your documents. By default, all requests are
# taken from this directory, but symbolic links and
# aliases may be used to point to other locations.
DocumentRoot "C:/Program Files/Apache Group/Apache/htdocs"
```

Demo Apache Web Server—JavaServer Pages API



Configuring BEA WebLogic Server—Java Servlet API

- Specify which Java™ virtual machine
  - In startWebLogic.cmd, set JAVA_HOME:

```
SETLOCAL

@rem Enable debugging
set JAVA_HOME=C:\jdk1.3

@rem Set user-defined variables. Note that JAVA_HOME
@rem will be taken  from the environment, if it is
@rem already defined.
if "%JAVA_HOME%" == "" set JAVA_HOME=.\jre1_2\jre
```

Configuring BEA WebLogic Server—Java Servlet API

- Specify debug arguments
  - In startWebLogic.cmd, set and use DEBUG_OPTIONS:

```
@rem Enable debugging
set DEBUG_OPTIONS=-classic
set DEBUG_OPTIONS=%DEBUG_OPTIONS% -Xdebug
set DEBUG_OPTIONS=%DEBUG_OPTIONS% -Xnoagent
set DEBUG_OPTIONS=%DEBUG_OPTIONS% -Djava.compiler=NONE
set DEBUG_OPTIONS=%DEBUG_OPTIONS% -Xrunjdwp:transport=
     dt_socket,server=y,suspend=n,address=5000

%JAVA_HOME%\bin\java %DEBUG_OPTIONS% -ms64m -mx64m …
```

Configuring BEA WebLogic Server—Java Servlet API

- Add the servlet we want to debug (step 1)
  - Put our servlet classes in the servlet classpath as specified in weblogic.properties

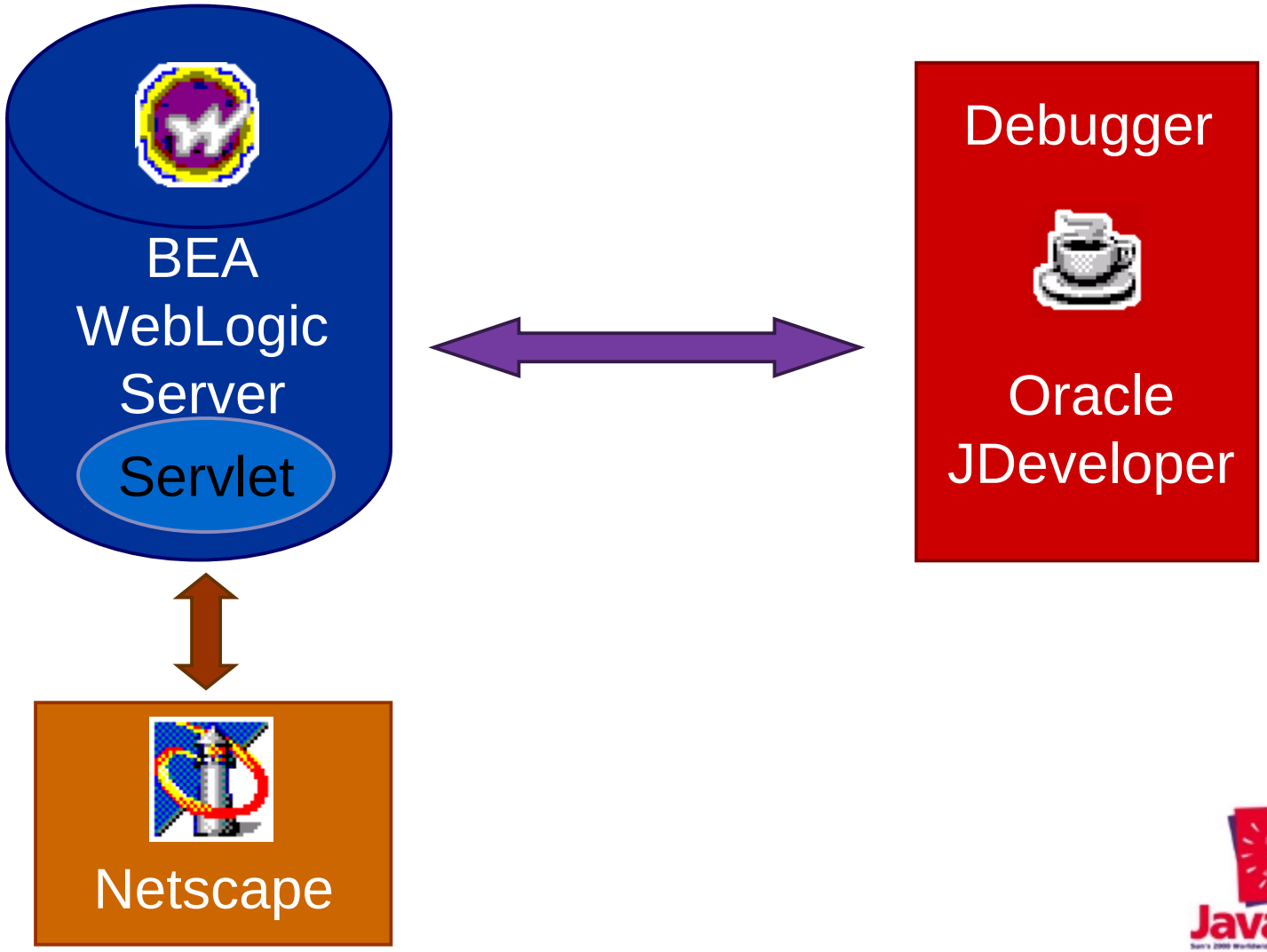
```
# Servlet reload properties
# ----------------------------------------
# Put servlets and classes they depend on below this
# directory to take advantage of dynamic reloading.
weblogic.httpd.servlet.classpath=
     C:/weblogic/myserver/servletclasses
```

Configuring BEA WebLogic Server—Java Servlet API

- Add the servlet we want to debug (step 2)
  - Register our servlet in weblogic.properties:

```
# USER-WRITTEN AND DEMO SERVLET REGISTRATIONS
# ------------------------------------------------
# Set ACLs for these as desired
# ------------------------------------------------
weblogic.httpd.register.servlet1=Servlet1
```

Demo BEA WebLogic Server—Java Servlet API

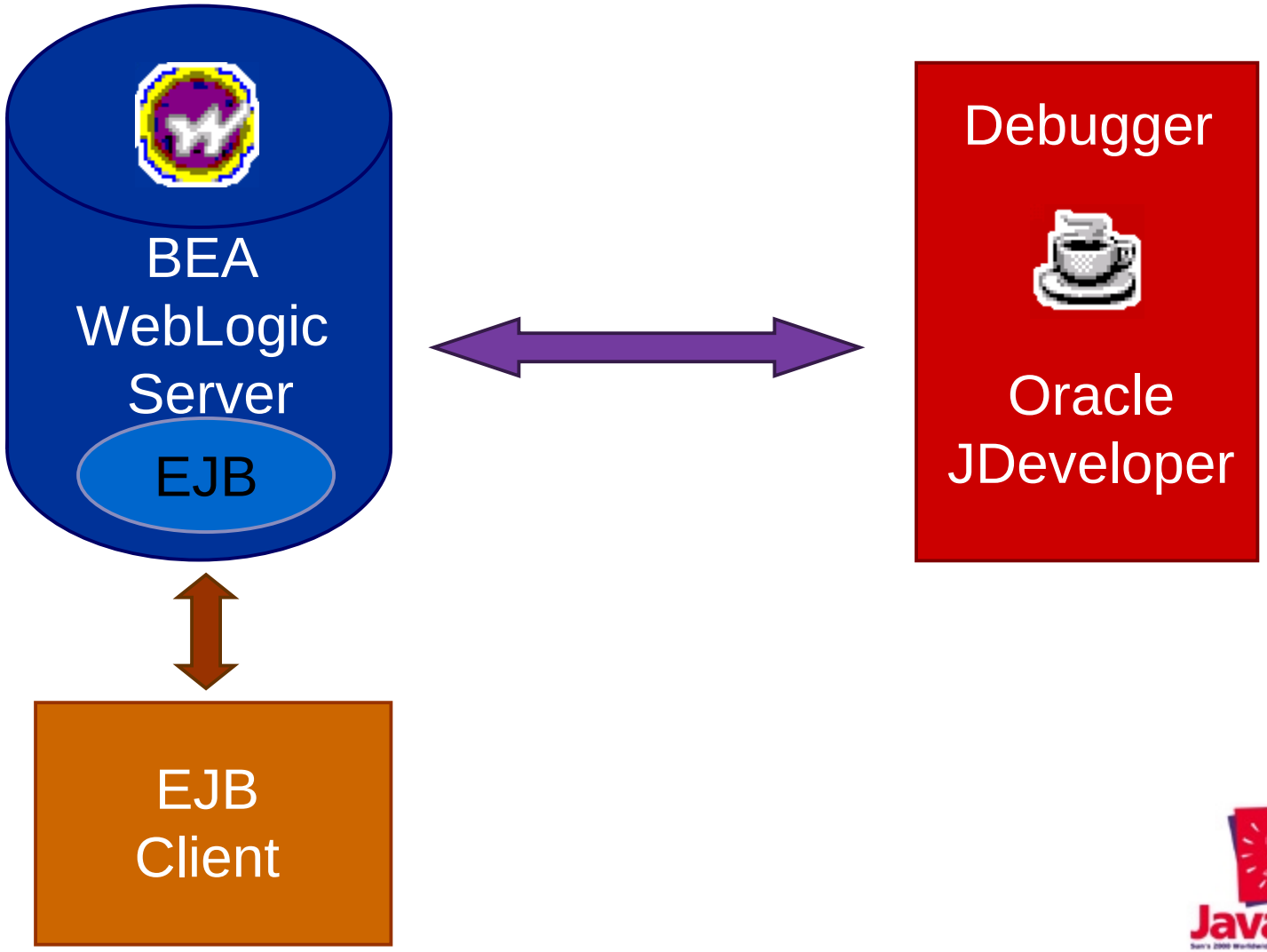


Configuring BEA WebLogic Server—EJB™ Components

- Add the Enterprise JavaBeans™ Architecture-based Components ("EJB™ components") we want to debug
  - Register our EJB component in weblogic.properties:

```
# Deploys EJBeans.
weblogic.ejb.deploy=D:/weblogic/myserver/MyEJB.jar
```

Demo BEA WebLogic Server—EJB Components



Configuring BEA WebLogic Server—JavaServer Pages API

- Specify classpath
  - In startWebLogic.cmd, set JAVA_CLASSPATH:

```
:runWebLogicJava
@rem Enable debugging
set JAVA_CLASSPATH=%JAVA_CLASSPATH%;C:\PROGRA~1\Oracle\JDEVEL~1.1\
     lib\ojsp.jar
set JAVA_CLASSPATH=%JAVA_CLASSPATH%;C:\PROGRA~1\Oracle\JDEVEL~1.1\
     JSWDK-~1.1\lib\servlet.jar
set JAVA_CLASSPATH=%JAVA_CLASSPATH%;C:\PROGRA~1\Oracle\JDEVEL~1.1\
     lib\ojc.jar
```

Configuring BEA WebLogic Server—JavaServer Pages API

- Specify JSP engine
  - In weblogic.properties, set weblogic.httpd.register.*.jsp

```
# WEBLOGIC JSP PROPERTIES
# ----------------------------------------------
weblogic.httpd.register.*.jsp=\
   oracle.jsp.JspServlet
```

Configuring BEA WebLogic Server—JavaServer Pages API

- Specify parameters for JSP engine
  - In weblogic.properties, set weblogic.httpd.initArgs.*.jsp:

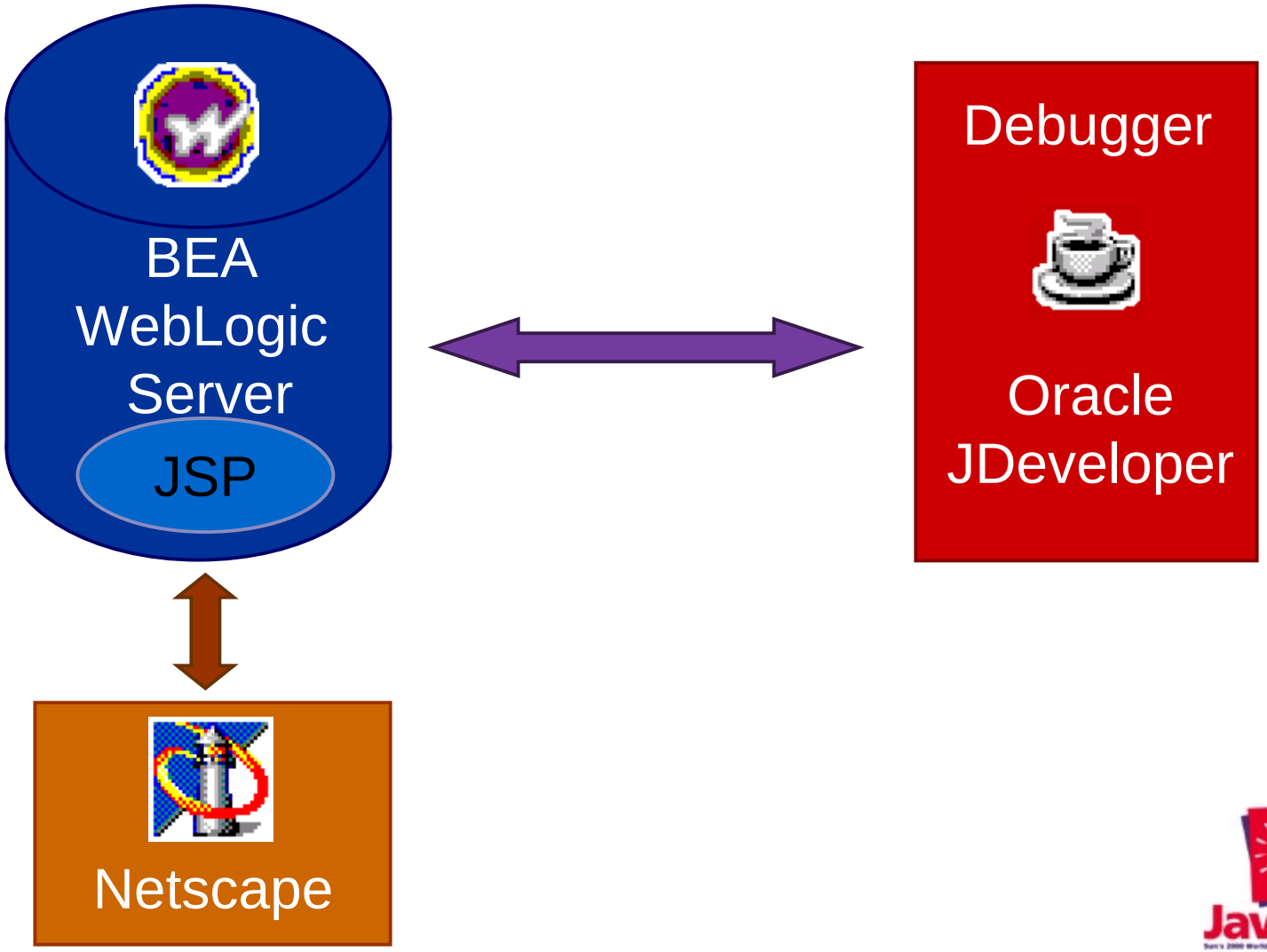
```
# WEBLOGIC JSP PROPERTIES
# ----------------------------------------
weblogic.httpd.register.*.jsp=\
   oracle.jsp.JspServlet
weblogic.httpd.initArgs.*.jsp=\
   emit_debuginfo=true,\
   jspcompiler=oracle.jdeveloper.jsp.JspOjcCompiler,\
   classpath=C:/jdk1.3/jre/lib/rt.jar
```

Configuring BEA WebLogic Server—JavaServer Pages API

- Add the JSP component we want to debug
  - Put our JSP component in the document root as specified in weblogic.properties

```
# DocumentRoot configuration
# ----------------------------------------
# The documentRoot is by default set to this directory;
# change as desired.
weblogic.httpd.documentRoot=public_html/
```

Demo BEA WebLogic Server—JavaServer Pages API



Summary

- Java™ Platform Debugger Architecture specifies the foundation for debugging
  - JPDA enables remote debugging of server side Java technology
- Remote debugging can be done with tools which support JPDA
  - Oracle JDeveloper allows you to debug Java servlets, JSP components, and EJB components, while they are running in a real server

Info

Visit Oracle Technology Network for White papers, Online Demos and Discussion Forums

http://otn.oracle.com

http://otn.oracle.com/products/jdev

JavaOne℠

Sun's 2000 Worldwide Java Developer Conference℠