Lab Guide

BEA WebLogic Server 8.1: Troubleshooting Methodologies

WLS-A33-81-02

The lab exercises were validated against BEA WebLogic Platform 8.1, SP4

Education Services, BEA Systems, Inc.
2315 North First Street, San Jose, CA 95131
1-800-232-4858

BEA Ed.Lab™ License Statements

Contents

Lab 1. Install BEA Software

(20 minutes)

Skills Learned

At the end of this exercise, you will be able to install BEA WebLogic Platform 8.1 software.

Problem Statement

To address the issues in the current Dizzyworld architecture, the application infrastructure platform must enable you to efficiently design, implement, and maintain highly integrated, high-performance, reliable enterprise applications.

BEA WebLogic Platform 8.1 increases productivity and lowers the cost for enterprise IT organizations, by providing a unified, simplified, extensible platform for application development, deployment, and management. BEA WebLogic Platform combines technology from proven BEA products that are used by thousands of customers worldwide to deliver a unified application infrastructure platform.

Design

You will install the BEA WebLogic Platform software for use with the lab exercises. BEA WebLogic Platform 8.1 (Figure 1) offers a common application architecture that includes the following:

- A set of integrated technology frameworks that provide a solutions-oriented starting point for addressing your project needs
- A single runtime framework for your entire software platform that abstracts complexity from the underlying software, simplifying the development of complex tasks and enhancing developer productivity for all aspects of the project
- A common development paradigm, providing the means to apply one skill set to multiple projects
- A unified, simplified management architecture, empowering developers and administrators to realize business objectives in an environment that is populated with distributed, heterogeneous technologies and platforms
- A highly-reliable, available, scalable, extensible, standards-based and high-performing foundation—WebLogic Server—that allows you to incorporate flexibility, and choice into your IT solutions

This course provides hands-on experience building various applications using BEA WebLogic Platform 8.1 products, with BEA WebLogic Workshop 8.1 as the development environment.

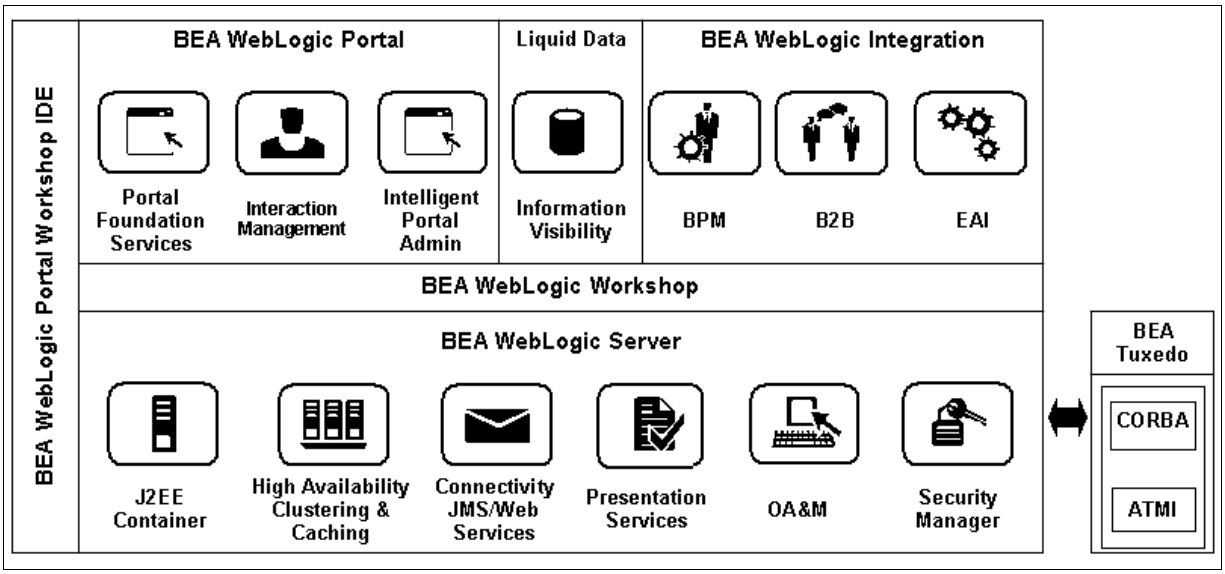


Figure 1. BEA WebLogic Platform Overview

Detailed Instructions

This section explains how to install BEA WebLogic Platform on a computer that has no previous versions of the product installed.

Install BEA WebLogic Platform 8.1	
1.	Insert the product CD-ROM in your computer and display its contents.
2.	Run the installation program on the CD-ROM: Windows: `installation_Platform814_win\platform814_win32.exe` Unix/Linux: `installation_Platform814_lnx\platform814_linux32.bin` **Note**: To successfully install WebLogic software and BEA Ed.Lab software to work together, use the same user ID when you install them. Make a note of your user ID.
3.	In the Welcome window, click **Next**.
4.	Accept the license agreement and click **Next**.
5.	Create a BEA home directory. Name the directory **bea**. Click **Next**. **Tip:** Typically, the BEA home directory is Windows: `c:\bea` Unix/Linux: `/root/bea`
6.	Select **Complete**. Click **Next**.
7.	Select the BEA product directory. Accept the default option. Click **Next**. BEA WebLogic Platform installs itself.
8.	In the dialog, clear **Run QuickStart**. Click **Done**.

Lab 2. Set Up the BEA Ed.Lab Environment

(10 minutes)

Skills Learned

At the end of this exercise, you will be able to install and configure BEA Ed.Lab.

Problem Statement

To address the issues in the current Dizzyworld architecture, you must hone your design, development and administration skills without wasting time configuring a complicated, non-intuitive lab environment or completing trivial, unrealistic lab exercises.

BEA Ed.Lab, offered exclusively by BEA, is a revolutionary learning environment that eliminates the gap between technical training and real-world implementation.

BEA Ed.Lab is:

- A safe, structured “sandbox” in which to practice skills
- A consistent business scenario
- A comprehensive lab framework that incorporates practical IT situations

Design

You will install and configure the BEA Ed.Lab environment and learn how to use the environment for developing and running the hands-on exercises.

Directory Reference Variables

The following table shows the key file directories that you refer to throughout the exercises.

Reference	Description	Value (Example)
<STUDENT>	The student directory contains exercises and other resources for the course.	Windows: C:\student\course_wls_troubleshooting D:\student\course_wls_troubleshooting
		Unix/Linux: {yourhome}/student/course_wls_troubleshooting
<BEA_HOME>	The installation directory for BEA products.	Windows: C:\bea D:\bea
		Unix/Linux: /opt/bea
<WEBLOGIC_HOME>	The installation directory for BEA WebLogic Platform 8.1.	Windows: C:\bea\weblogic81 or D:\bea\weblogic81
		Unix/Linix: /opt/bea/weblogic81

Detailed Instructions

Install and configure BEA Ed.Lab

1. If WebLogic Server is running, shut it down.

   NOTE: *To successfully install WebLogic software and BEA Ed.Lab software to work together, you must use the same user ID when you install them.*

2. Insert the student CD-ROM in your computer and display its contents.

3. Run the BEA Ed.Lab installation script; it is located at the `labs` subdirectory of the CD:
   Windows: Run the `install.cmd` script.

   Unix/Linux: Run the `install.sh` script. Detailed tasks:
   A. Ensure the course CD-ROM is mounted
   B. Open a new command prompt window and change directory to the CD-ROM installation location. (Usually `/mnt/cdrom`). Change directory to `labs`.
   C. Execute the `install.sh` file using the following (mind the leading period):
      `./install.sh`
   D. If you are prompted for a valid JDK 1.4 directory, type in your `<BEA_HOME>/jdk142_05` value (for example: `/opt/bea/jdk142_05`).

   To successfully install BEA Ed.Lab, select your own locations for BEA Ed.Lab and <BEA_HOME> in the dialogue. ***Defaults may not apply to your file structure****.*

4. Verify that the defaults are OK for your system, or change them.
   The installation wizard finishes and closes.

5. Windows: To open a command prompt, use the `promptBEA.cmd` file located in the `<STUDENT>\bin` directory.
   Unix/Linux: To open a command prompt, use (mind the period and space in front) `./promptBEA.sh` located in the `<STUDENT>/bin` directory.

   This script ensures that your environment variables PATH and CLASSPATH in this window have been correctly initialized.

Update file attributes (Windows XP only)

Perform the steps in this section only if your computer operating system is Windows XP.

6. If WebLogic Server is running, shut it down.
   If WebLogic Workshop is open, close it.

7. Open Windows Explorer.
   Right-click on the `<STUDENT>` directory and select Properties.
   Deselect the Read-only attribute. Click Apply.
   In the Confirm Attribute Changes dialog, select Apply changes this folder, subfolder and files. Click OK.

   Tip: *The Applying Changes dialog indicates that the operation can take over an hour to perform, but it typically takes only a few minutes.*

Lab 3. Troubleshoot a Server Core Dump

(20 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Identify a core file
- Know how to read a core file to determine what caused the crash
- Know how to identify what code needs to be changed to ensure that the crash does not occur again

Problem Statement

When an error occurs in the Java Virtual Machine (JVM) or in libraries that are bound to the JVM through JNI (Java Native Interface), it is possible to crash the JVM. When the JVM crashes, it dumps a core file. Understanding how to read and deal with a core file is a key skill in diagnosing the problem.

Design

In this lab you will use a purpose built application that crashes the WebLogic Server and will then try to determine how it crashes the server and what code is causing the crash.

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	

Administrator Username	Administrator Password
system	weblogic

Detailed Instructions

Start the WebLogic Server instance

1. a. Windows: If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: If not already running, start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Build and deploy the application

2. a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

3. Change to the main directory of the current lab and type:

   ant

   This builds and deploys the lab application.

Cause the WebLogic Server to crash

4. Windows:

   a. Visit the lab application web site at *http://localhost:7001/CORE/weblogicCoreDump*

   b. Click **Crash WebLogic Server**.

   c. You should see an error page after clicking the button.

   Linux:

   a. Visit the lab application web site at *http://localhost:7001/CORE_LINUX/NativeServlet*

   b. *Click Reload/Refresh on your browser to reload the same page again.*

5. Check that the server has indeed failed either by trying to open or use the Admin Console, checking to see the message in the server console (if it exists at all) or by simply trying to reload the web page.

6. a. Windows: Shut down the PointBase window started as part of the server startup script by simply closing (not minimizing) the PointBase window.

   b. Linux: Kill the PointBase database process if it is still running in the background.

   Hint: *use ps –ef | grep database to find the PointBase process id and kill it.*

Restart the WebLogic Server instance

7. a. Windows: Restart the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Restart the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Examine the server core file

8. Find the server core file that the JVM dropped when it crashed. Look in the directory where the server was started from, `<STUDENT>/bin`. The filename will be of the form `hs_err_pid<WLSpid>.log`, where `<WLSpid>` is the WebLogic Server process ID.

9. View the contents of the file in a text editor.

10. What does this file tell you about the crash and its cause?

Perform a thread dump on the running server

11. Get a thread dump on the running server using the following methods:

    Windows:

    a. Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Get the thread dump using:
    java weblogic.Admin –url t3://localhost:7001
    –username system –password weblogic THREAD_DUMP

    c. The thread dump will be printed to the WebLogic server window.

    Linux:

    a. Open a Terminal window.

    b. Find the WebLogic Server process PID using **ps** command. Send a SIGQUIT signal by typing

    kill -3 pid

    c. The thread dump will be printed to the WebLogic server window.

    Hint: *Use ps –ef | grep weblogic to find the WebLogic Server process id.*

12. What other method can you use to take a thread dump on the server running on the Windows platform?

    Is there any difference between the two methods?

Modify the WebLogic Server startup script

13. Next time the server crashes you would like to take a thread dump at the point of failure. How can you do this?

14. To ensure the server gives you the chance to take a thread dump just before it crashes:

    a. Windows: Edit the `startServer.cmd` script in `<STUDENT>\bin`.

    Linux: Edit the `startServer.sh` script in `<STUDENT>/bin`.

    b. Add the switch to the java command:

    `-XX:+ShowMessageBoxOnError`

    Note : *Add it anywhere after 'java' and before 'weblogic.Server'*

Restart the WebLogic Server instance

15. a. Windows: Shut down the PointBase window started as part of the server startup script by simply closing (not minimizing) the PointBase window.

    b. Linux: Kill the PointBase database process running in the background.

    Hint: *use ps –ef | grep database to find the PointBase process id and kill it.*

16. a. Windows: Stop the Administration Server and then restart it using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Stop the Administration Server and then restart it using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Perform a thread dump on a crashing WebLogic Server

17. Crash the server using the lab application.

18. A dialog box (Windows) or a prompt (Linux) should appear giving you the opportunity to take a thread dump.

    a. Do *not* click yes or no**.**

    b. Take a thread dump of the server.

Examine the thread dump

19. Examine the thread dump and see if you can answer the following questions:

    a) What is the error that is occurring and causing the WebLogic Server instance to crash?

    b) Where is the error coming from, that is, what module?

    c) What java class and method is calling the native code, which is making the WebLogic Server instance crash?

    d) How would you fix this problem?

    Once you have identified the cause of the crash, get the developers of the application of the JNI plug-in involved to help resolve the problem.

Lab cleanup

20. If the dialog box is still open, click **no**.

21. If you have time, remove the switch you added to the server startup script.

22. Undeploy the lab application:

    a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    To undeploy the application, change to the main directory of the current lab and type:

    ant undeploy

Lab 4. Investigating a JVM Crash with No Core Dump

(20 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Discuss some of the potential issues for core files not being produced when the JVM crashes.
- Understand the processes involved in identifying and resolving the issues.

Problem Statement

WebLogic Server runs within a JVM. If the JVM crashes then the clues regarding the reason for the crash can be found in the core file that is produced. This lab provides example scenarios for figuring out why a core file may not be produced when the JVM crashes.

Detailed Instructions

Expect the unexpected	
1.	You are the WebLogic Server administrator and the WLS has crashed. More precisely, the JVM that WLS was running in has crashed (you can see the message on the console output). You need to look at the core file to figure out if it was the JNI code that was recently developed or if it was a problem with the JVM. You look into the directory that WLS was started in and to your horror; there is no core file to be found.
Investigate the mystery of missing core	
2.	First you check to see how much disk space is free. You realize that there is only 125 KB free on the volume, and the script you have been meaning to write to clean up old log files has not been written yet. What should you do?

3. After a general clean up, there are now several gigabytes available on the volume. You start up WebLogic Server thinking that the crash may have been caused by the lack of disk space.

   At five o'clock that evening as you are about to head home, you get a call that the WebLogic Server has crashed again. Once again the core file is missing.

   There must be a reason that core files cannot be created. Surely it cannot be bigger than the free space on the volume (several gigabytes).

   Is it a system limitation? Where should you start looking next?

4. You decide to look at system limits.

   For Unix based systems you type:

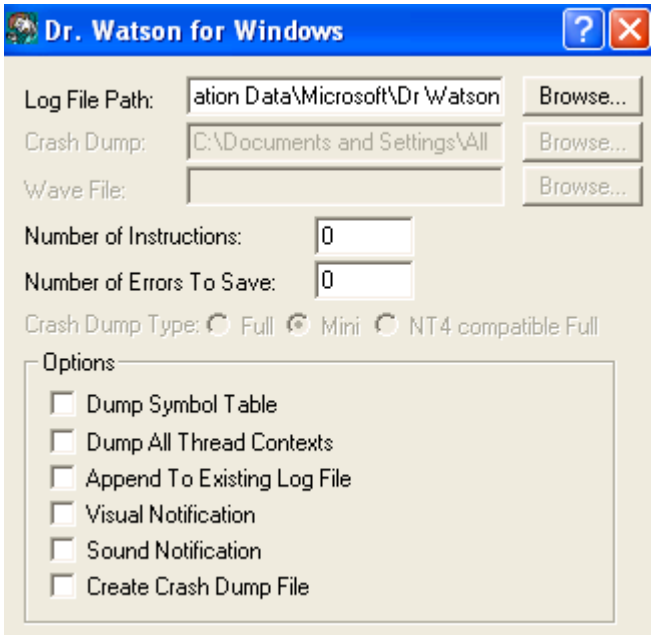
   `ulimit –c | grep core` yields:

```
core file size (blocks)       0
```

   `grep core /etc/security/limits.conf` yields:

```
*               soft    core            0
```

   In Windows, you open the Msinfo32.exe tool, select the **Tools** menu and click **Dr Watson**.



   What should you do?

5. In Unix based systems:

   You have changed the limits.conf file to read:

   ```
   *               soft    core            unlimited
   ```

   In Windows:

   You have changed the properties to allow more errors to save, to dump all the thread contexts and to create a crash dump file.

   This will enable core files system wide. You start up a new shell (so the changes can take effect) and boot WebLogic Server. Is there anything else you could have checked to ensure core files will be produced?

Phew!

6. Temporary relief. The system crashed just as you were heading out for lunch but at least a core file was produced. Now that we have the file, we can start investigating what the real problem is.

   Suggest some reasons why the WebLogic Server's JVM is crashing?

7. After you investigated the core file, you noticed a file in the same directory called `hs_err_1234.log`.

   Could this file be useful in tracking down the problem?

Discuss your answers with the class and instructor

8. Once everyone has completed this exercise, work through your answers with the group and discuss the various options.

Lab 5. Troubleshoot a Generic Server Hang

(30 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Identify a server hang
- Determine the cause of a server hang
- Understand the processes involved in identifying and resolving the issues.

Problem Statement

A server hang can cause an application to become unusable and can directly impact business. Understanding why this happens and finding the cause are vital skills in the resolving the problem.

Design

This lab will simulate a server hang caused by a lack of execute threads. A servlet in the application is designed to sleep for a long time. When it is run many times concurrently, all the execute threads will be consumed leaving none for other requests. This particular server hang will unravel because the logic of the servlet code will unravel, eventually freeing the thread, but until that happens, nothing else can happen on the server.

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	

Administrator Username	Administrator Password
system	weblogic

Detailed Instructions

Start the WebLogic Server instance

1. a. Windows: If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: If not already running, start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Build and deploy the application

2. a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

3. Change to the main directory of the current lab and type:

   ant

   This builds and deploys the lab application.

Cause the server to hang

4. Visit the lab application web site at *http://localhost:7001/HANG/genericServerHang*. No web page will be returned because the application is sleeping. However, the server log will provide a message indicating what is happening with the system.

5. Run a script to occupy all the server's thread:

   a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

   c. Change to the current lab directory and type:

      ant -f RunLab.xml hangServer

Investigate the server hang

6. Try pinging the server.

   a. Windows: Open another Command Prompt window using the `PromptBEA.cmd` script.

   b. Linux: Open another Terminal window using the `promptBEA.sh` script.

   c. Type the following command to ping the server:

      java weblogic.Admin -url t3://localhost:7001 -username system -password weblogic PING

7. Did the server respond?

   If not, why could this be?

   It is time to look at the thread dumps.

Perform a thread dump on the running server

8. Try to get a thread dump on the running server using the following methods:

   Windows:

   - Using `weblogic.Admin`

     a. Open a Command Prompt/Terminal window using the `PromptBEA.cmd` (Windows) OR `promptBEA.sh` (Unix/Linux) script.

     b. Get a thread dump by typing:
     java weblogic.Admin –url t3://localhost:7001 –username system –password weblogic THREAD_DUMP

   - OS Specific Method

     a. Switch to the server console window. Get a thread dump by pressing CTRL + BREAK.

   Linux:

   - OS Specific Method

     a. Open a Terminal window window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

     b. Find the WebLogic Server process PID using **ps** command. Send a SIGQUIT signal by typing

     Kill -3 pid

   Hint: *Use ps –ef | grep weblogic to find the WebLogic Server process id.*

9. Gather three or more thread dumps several seconds apart.

10. Look at the output of each to determine what is happening.

    Hint: Look at the execute threads to see what they are doing and if they change over time. Are any of them idle? Look for Object.wait() only in the weblogic.kernel.Default execute queue thread pool.

Configure the WebLogic Server with more execute threads

11. To try and resolve the problem increase the number of execute threads.
    a. Open a web browser.
    b. Enter the URL: *http://localhost:7001/console*
    c. Enter the system administration username and password.
    d. In the Administration Console, browse to:
       dizzyworld -> Servers -> adminserver -> Configuration tab -> **General** tab
    e. Click **Advanced Options [Show]** at the bottom of the page.
    f. Click **Configure Execute Queues** at the bottom of the page.
    g. Click the weblogic.Kernel.Default queue.
    h. Click the **Configuration** tab.
    i. Increase the **Thread Count** to **25**.
    j. Click **Apply**.

12. Stop and restart the Administration Server.

    Note: Close the PointBase window before you restart the server.

Repeat the exercise with more execute threads

13. Open a Command Prompt/Terminal window using the `PromptBEA.cmd` (Windows) OR `promptBEA.sh` (Unix/Linux) script and use the `runlab.xml` ant script again to hang the server and then attempt a PING.

    Did anything change?

    Perform a thread dump and have a look at the execute queues. What has changed?

14. You have just increased the execute thread count of the server for the sake of the exercise. In a real application, would it be wise to increase the execute thread count for a server that has long-running service calls?

Will this solve the problem? What can be done about application code that hogs the execute threads?

15. Using the WebLogic Server Administration Console, how would you increase the number of execute threads that can act as socket readers?

Would it be better if a fixed number of threads can be allocated to this task or is the current strategy of using a percentage of the total execute threads the best way to define this parameter?

Why?

16.	In this lab we observed a server hang because the execute threads were all consumed by an application. What are some other reasons WebLogic Server may hang (or appear hung) and how do you think you could diagnose (and fix) these problems?

Discuss your answers with the class and instructor	
17.	Once everyone has completed this exercise, work through your answers with the group and discuss the various options.

Lab cleanup	
18.	Change the execute thread count back to **15** once you have completed the exercise.
19.	Undeploy the lab application: a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory. To undeploy the application, change to the main directory of the current lab and type: **ant undeploy**

Lab 6. Troubleshoot a High CPU Usage Problem

(20 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Identify a high CPU usage WebLogic Server instance.
- Perform an analysis of the server instance to determine what threads are causing the issue.
- Use this information to identify areas where performance tuning is required.

Problem Statement

High CPU usage is a common problem that can result from inefficient code or under configured WebLogic Server servers. Being able to identify and resolve high CPU usage problems is a key skill required to ensure that the WebLogic Server system runs smoothly and quickly.

Design

The objective of this lab is to gain experience dealing with a WebLogic Server system showing unusually high levels of CPU load.

To achieve the high levels of CPU load, the example application uses performance-testing code taken from a benchmark developed by Christopher W. Cowell-Shah. Full details on the benchmark can be found at *http://osnews.com/story.php?news_id=5602.*

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	

Administrator Username	Administrator Password
system	weblogic

Detailed Instructions

Start the WebLogic Server instance

1. Windows:

   a. If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   Linux:

   a. If already running, stop the Administration server.

   b. Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

   c. Set the environment variable "`LD_ASSUME_KERNEL`" by typing the following command on the prompt:

   export LD_ASSUME_KERNEL=2.4.1

   d. Start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Build and deploy the application

2. a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

3. Change to the main directory of the current lab and type:

   ant

   This will build and deploy the lab application.

Create CPU load

4. Visit the lab application web site at *http://localhost:7001/HCPU/highCPUUsage.*

5. Generate CPU usage with the application. Enter the number of times that you would like to run the load test, where each cycle will generally take around four to six minutes, depending on the power of your machine.

Monitor the CPU load

6. While the CPU is under load, monitor the CPU.

   Windows:

   a. Open the Windows Task Manager by pressing the keys CTRL+ALT+DEL.

   b. Select the **Processes** tab.

   c. Click the **CPU** column header to see the process using the highest amount of CPU.

   Linux:

   a. Use Top utility to find the process id (PID) for the thread that is using up the CPU. Type the following command on a terminal window :

      top

   (***Note:*** *Linux treats every thread as a process*)

7. Which process is currently taking the most CPU load?

   Write down the PID of the process that is consuming the most CPU for the next step.

   PID = _____

   Note (Windows ONLY): *If you do not see the PID, you may need to click* ***View*** *on the menu and then Select* ***Columns.*** *Select* ***PID*** *(Process Identifier) as one of the columns to be displayed.*

8. Windows ONLY:

   Determine which of the process's threads is creating the load.

   Use either the Windows Process Explorer or pslist, both of which have been downloaded for you and supplied with the lab material in the `<STUDENT>/bin` directory.

   To use the pslist tool:

   a. Open a Command Prompt window using the `PromptBEA.cmd` script.

   b. Start the pslist application by typing:

      pslist –d <*PID*>

   Where <*PID*> is the PID of the WLS instance from the previous step.

   Note: *You can download the Process Explorer and pslist application from the www.sysinternals.com site.*

9. Windows ONLY:

   Write down the thread ID consuming most of the processing time in the server.

   THREAD ID = _____

10. Convert the Thread ID (Windows) OR PID (Linux) to a hexadecimal representation.

    Windows: You can use the Windows Calculator to do this.

    a. Start the Calculator.
    b. On the **View** menu, select **Scientific**.
    c. Enter the decimal value and then click **Hex**.

    Linux: You can use the following command to print the hexadecimal representation of the PID from step-7:

    a. **echo | awk '{printf("%x\n",<PID>)}'**

    Where <*PID*> is the PID of the WLS instance from Step-7.

    HEXADECIMAL THREAD ID = _____

11. Windows:

    Increase the screen buffer size.

    a. In the running server console, click on the upper left hand corner and select **Properties**.
    b. Click the **Layout** tab and change the **Height** of the **Screen Buffer Size** to **9999**.

    Linux:

    Find the WebLogic Server process id

    b. Use ps command to find the WebLogic Server process id

       ps -ef | grep java | more

    c. Ignore the first java process that gets listed which will be database process.

       (***Hint:*** *The database process will be listed as :*

       java -Ddatabase.home=/root/student/course_wls_troubleshooting/bin/databases ...)

    d. Note the PID of the next java process listed which is WebLogic Server.

       (***Hint:*** *The WebLogic server main thread will start with :*

    java -Dweblogic.security.SSL.verbose=false...)

    WLS_PID = _____

12. To get a thread dump on the running server, use *one* of the following methods.

Windows:

- Using `weblogic.Admin`

c. Open a Command Prompt/Terminal window using the `PromptBEA.cmd` (Windows) OR `promptBEA.sh` (Unix/Linux) script.

d. Get a thread dump by typing:
java weblogic.Admin –url t3://localhost:7001 –username system –password weblogic THREAD_DUMP

- OS Specific Method

b. Switch to the server console window. Get a thread dump by pressing CTRL + BREAK.

Linux:

- OS Specific Method

c. Open a Terminal window.

d. Find the WebLogic Server process PID using **ps** command. Send a SIGQUIT signal by typing

Kill -3 pid

Hint: *Use the WebLogic Server process id from Step-11.*

13. Copy the results of the thread dump into a text editor and search for the hexadecimal representation of the thread identifier to find out what the thread (represented by the hexadecimal thread id from Step-10) is doing.

14. Can you answer the following questions?

What java classes are causing the load on the WebLogic Server instance?

How would you go about resolving this problem?

Discuss your answers with the class and instructor

15. Once everyone has completed this exercise, work through your answers with the group and discuss the various options.

Lab cleanup

16. Undeploy the lab application:

    a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    To undeploy the application, change to the main directory of the current lab and type:

    ant undeploy

Lab 7. Troubleshoot an Out of Memory Problem

(20 minutes, or 30 minutes to complete the optional component)

Skills Learned

At the end of this lab, you will be able to:

- Identify a WebLogic Server instance that has run out of heap memory.
- Adjust the WebLogic Server system to allocate more memory on the heap
- Use the BEA JRockit tools to help further refine the location of the problem.

Problem Statement

WebLogic Server instances can run out of memory for any number of reasons. Most commonly the out of memory error is caused by an application that has simply allocated more memory than what is available on the heap.

Design

This lab uses a small application that allows for a large amount of data to be allocated on to the Java heap. The challenge of this lab is to modify the WebLogic Server configuration so that there is more memory for the WebLogic Server instance to handle the additional space required.

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	
BEA_HOME	Windows: `c:\bea`	The installation directory for BEA products.
	Unix/Linux: `/opt/bea`	

Administrator Username	Administrator Password
system	weblogic

Detailed Instructions

Start the WebLogic Server instance

1. a. Windows: If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: If not already running, start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Build and deploy the application

2. a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

3. Change to the main directory of the current lab and type:

   ant

   This builds and deploys the lab application.

Allocate some memory in the Java heap

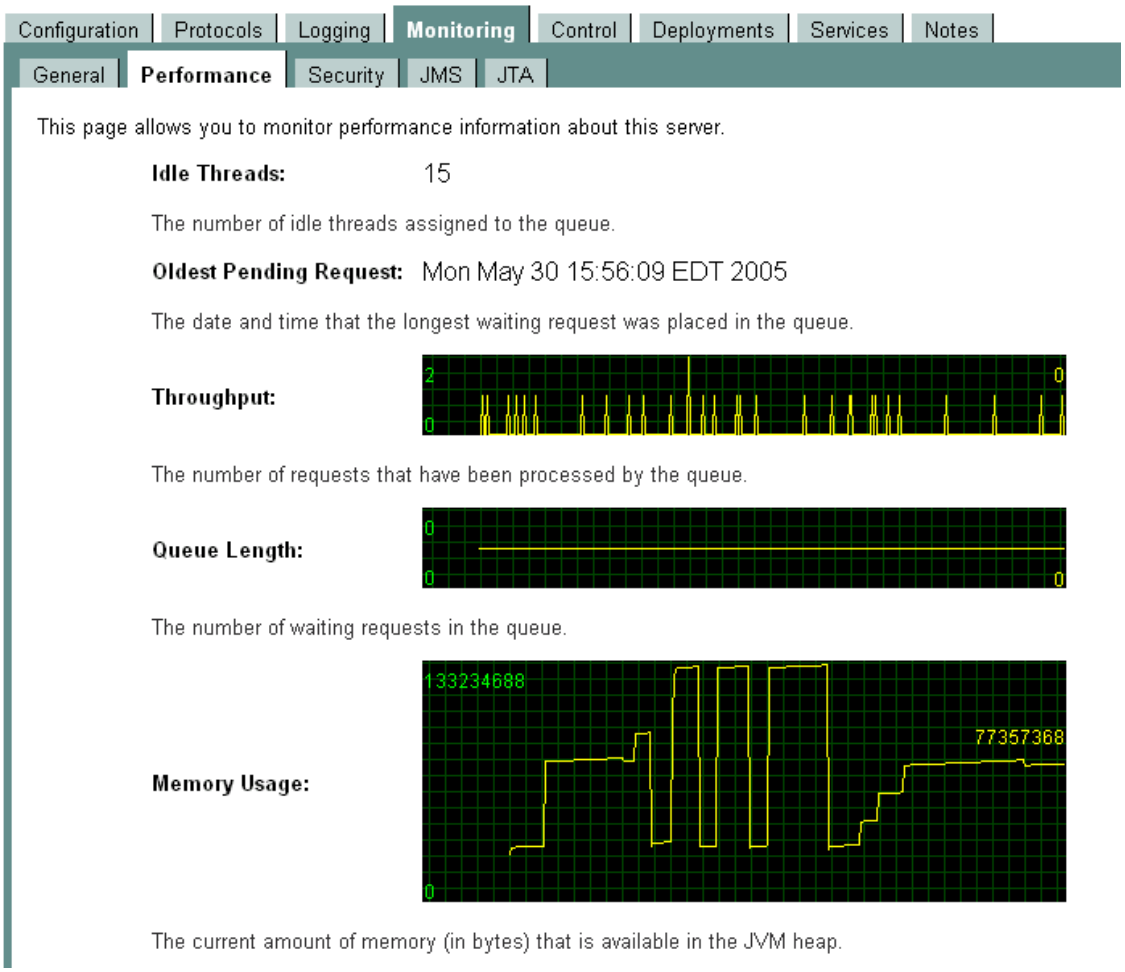
4. Visit the lab application web site at *http://localhost:7001/OOME/outOfMemory* **.**

5. Have the application allocate memory in the Java heap by entering a small number and clicking **Allocate this MB of data**. The lab application then allocates memory on to the Java heap in 1MB chunks.

Monitor Java heap

6. Monitor the heap usage via the Admin Console.
   a. Open a Web browser.
   b. Enter the URL: *http://localhost:7001/console.*
   c. Enter the system administration **username** and **password**.
   d. Browse to
   labdomain > Servers > adminserver > Monitoring tab > **Performance** tab

7. A screen containing performance charts appears. The memory usage chart shows the current size of the heap as well as the amount of heap space currently in use.



8. Use the lab application to allocate chunks of memory until you see an error in the server console window. Take a note of how much memory it took to cause the error.

9. What error do you receive on your browser window?

What do you see in the server console window?

10. Examine the memory usage in the administration console. What is happening with the heap usage?

    Allocate a smaller amount of memory to the heap using the lab application.

    Does the heap usage go down when you reduce the amount of memory allocated on to the heap by the test application?

    If not, why does this happen?

    Sometimes Out of Memory errors occur simply because the Java heap is not big enough. We will now increase the size of the Java heap to see if that makes a difference.

Increase the heap size and turn on verbose garbage collection

11. Shut down the server and close the PointBase window.

12. Increase the heap size allocated to the server when it starts.

    a. Windows: Edit the `startServer.cmd` file in the `<STUDENT>/bin` directory.

    b. Linux: Edit the `startServer.sh` file in the `<STUDENT>/bin` directory.

    c. Locate the java startup line similar to the following:

    ```
    java -Dweblogic.security.SSL.verbose=false
    -Dssl.debug=false -Dweblogic.StdoutDebugEnabled=true
    -ms64m -mx64m -hotspot …
    ```

    d. Change the minimum and maximum heap size to 128MB.

    `–ms128m –mx128m`

13. Turn on verbose garbage collection by adding the –verbosegc switch to the java startup line. The java command should now look like:

    ```
    java -Dweblogic.security.SSL.verbose=false
    -Dssl.debug=false -Dweblogic.StdoutDebugEnabled=true
    -ms64m -mx64m –verbosegc -hotspot …
    ```

14. Restart the administration server.

Retest with more heap space

15. Use the lab application to allocate chunks of memory in order to trigger garbage collection.

16. Look in the server console window for the verbose garbage collection output. You should see something similar to this:

    [memory] 1641.650-1641.951: GC 56282K->52382K (131072K), 301.000 ms

    What does this show about GC?

17. How much memory can you now allocate using the test application?

    Has this increased over how much memory you could allocate previously?

18. Increase the amount of memory being allocated until the server throws an Out of Memory exception. Check the server console and look for the garbage collection just before the Out of Memory exception was thrown.

    What do you notice about the last garbage collection directly before the exception was thrown?

    How much memory was freed in this last garbage collection run? What does this tell you?

    The previous steps show a WebLogic Server system that is running of memory and how to identify such a situation. Normally, once you have reached this point, developers will need to help determine what part of the code is leaking the memory.

19. If you are finished with the lab, please skip to the cleanup instructions. If you still have time, you may try the optional section of the lab.

(Optional) Use the BEA JRockit JRA Tool

The following instructions will work only if you are running WebLogic Server 8.1 SP3 or higher on a Windows or Linux platform. BEA JRockit is not available on other platforms and earlier versions of BEA JRockit do not have the necessary tools to provide the following functionality.

BEA JRockit provides a basic implementation of a profiling system that you can use to help identify where memory is being used in the JVM. For a more detailed analysis, typically a JVMPI tool (such as OptimizeIT or JProbe) would be used. However these tools are more complex to install and hence outside the scope of this course. The following instructions use the BEA JRockit tool to provide some basic analysis of the environment and the potential problem.

20. Stop the server and close the PointBase window.

21. Change the JVM from Sun to the BEA JRockit.

Windows:

a. Open the `<STUDENT>\bin\setenvBEA.cmd` file.

b. Comment out the line:

```
set BEA_JDK=%BEA_HOME%\jdk142_05
```

c. Add the line:

```
set BEA_JDK=%BEA_HOME%\jrockit81sp4_142_05
```

Linux:

a. Open the `<STUDENT>/bin/setEnvBEA.sh` file.

b. Comment out the line:

```
BEA_JDK=$BEA_HOME/jdk142_05
```

c. Add the line:

```
BEA_JDK=$BEA_HOME/jrockit81sp4_142_05
```

Note: If you are running a different version of WebLogic Server, the name of the BEA JRockit directory may be different. In any case, use the latest BEA JRockit installation that is in your `BEA_HOME` directory and make sure that the BEA JRockit release is at least 1.4.2_04 or later.

22. Set JVM specific switches.

    a. Windows: Edit the `startServer.cmd` file in the `<STUDENT>/bin` directory.

    b. Linux: Edit the `startServer.sh` file in the `<STUDENT>/bin` directory.

    c. Locate the java startup line similar to the following:

```
java -Dweblogic.security.SSL.verbose=false
-Dssl.debug=false -Dweblogic.StdoutDebugEnabled=true
-ms128m –mx128m -hotspot …
```

    d. Add the **–Xmanagement** switch to the java start command.

    e. Remove the **–hotspot** switch if it is currently being used.

```
java -Dweblogic.security.SSL.verbose=false
-Dssl.debug=false -Dweblogic.StdoutDebugEnabled=true
–Xmanagement -ms128m –mx128m –verbosegc …
```

23. Start the server.

24. Once the server has started, open the BEA JRockit console.

    Windows:

    a. Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Change to the `<BEA_HOME>\jrockit81sp4_142_05\bin` directory.

    c. Open the JRockit Management Console window by typing:

    d. **console.exe**

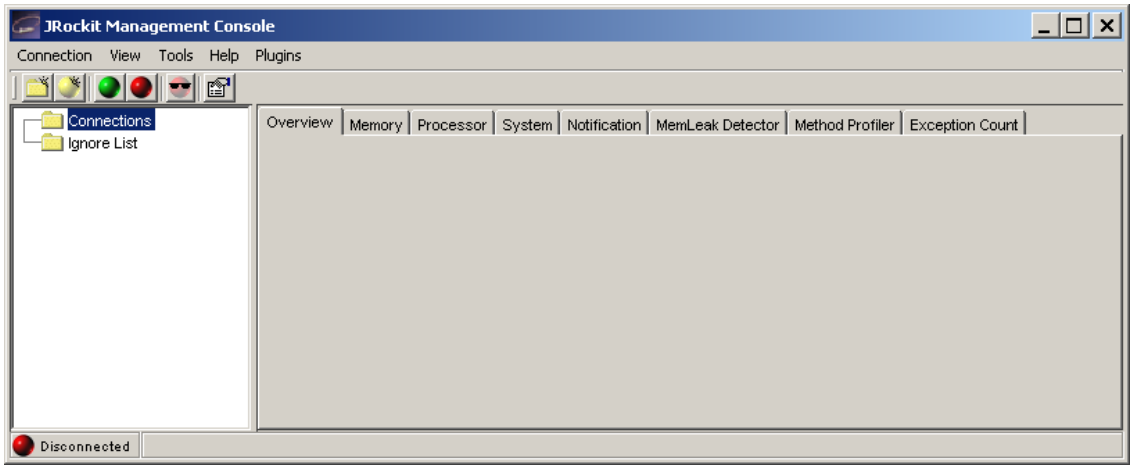
    Linux:

    a. Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    b. Change to the `<BEA_HOME>/jrockit81sp4_142_05/bin` directory.

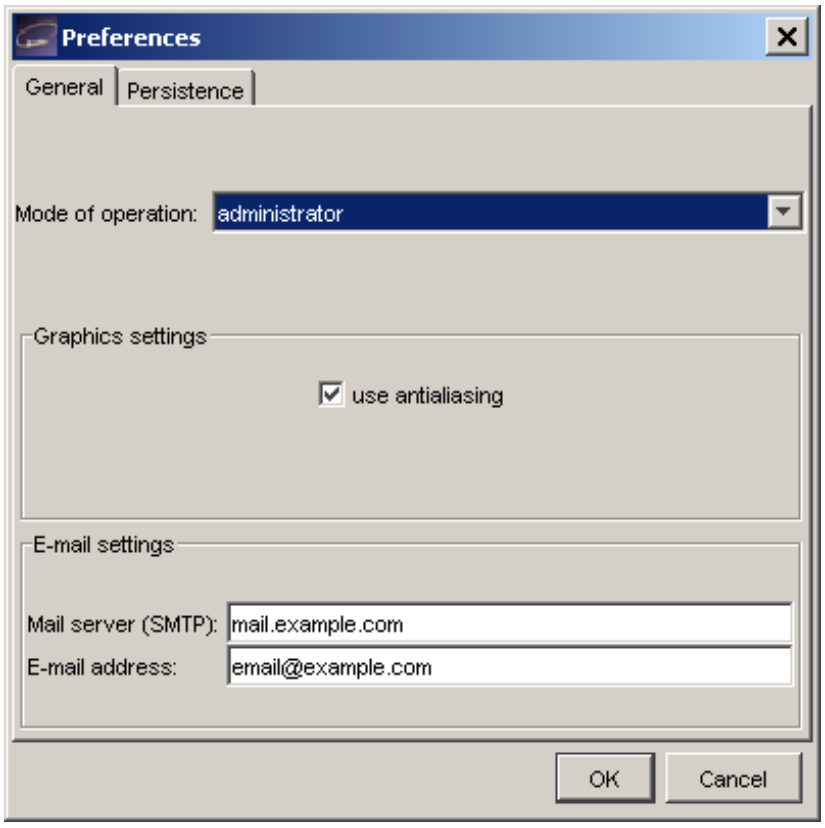
    c. Open the JRockit Management Console window by typing:

    d. ./**console** (mind the leading period)

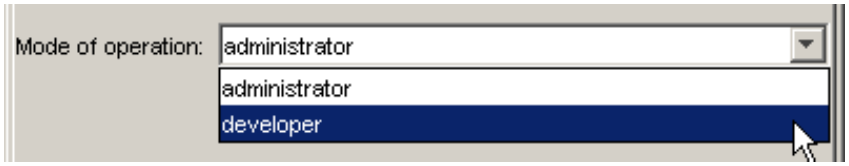


25. Create a new connection to the local JRockit instance by clicking the **yellow** button. Leave the port numbers at the default. You should now be connected. If not, connect to the JRockit instance by selecting the connection and clicking the **green** button.

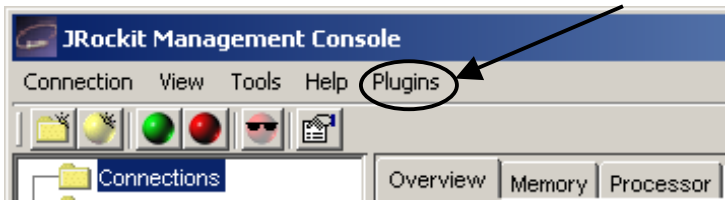
26. On the **Tools** menu, click **Preferences**.



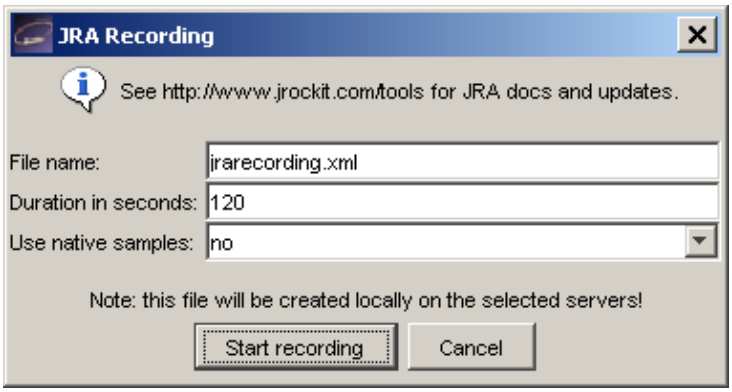
27. On the **General** tab, in the **Mode of operation** box, select **developer**. Click **OK**.



28. The **Plugins** menu appears on the menu bar.



29. Click the Plugins menu and select **Start JRA recording.**



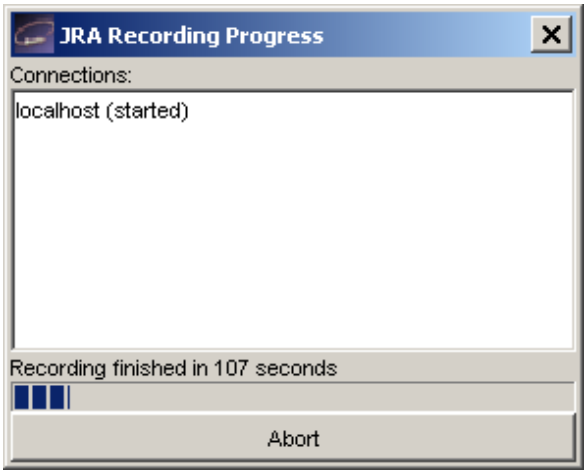
The JRA recording creates a snapshot of the performance of the application over a period of time. We can this use this snapshot to do an analysis and see if we can determine patterns that can help resolve the issues the system is having.

30. Enter a name for the recording. The file is created in the directory where the server was originally started. The file will be overwritten if it already exists.

    a. Set the duration of the recording to **60** seconds.

    b. Click **Start recording**.

    The following box appears:



This box indicates that the recording has started. You will see confirmation messages halfway through the recording and when the recording is finished.

Note: *If the start recording action fails and if you receive a jrockit.license.LicenseException consult with your instructor to update the JRockit license file.*

31. Using the lab application, allocate chunks of memory until you get the Out of memory exception. Make sure you allocate memory fast enough in order to get the exception before the recording has finished.

32. Once the recording has finished, close the JRockit Management Console and open the BEA JRockit Runtime Analyzer.

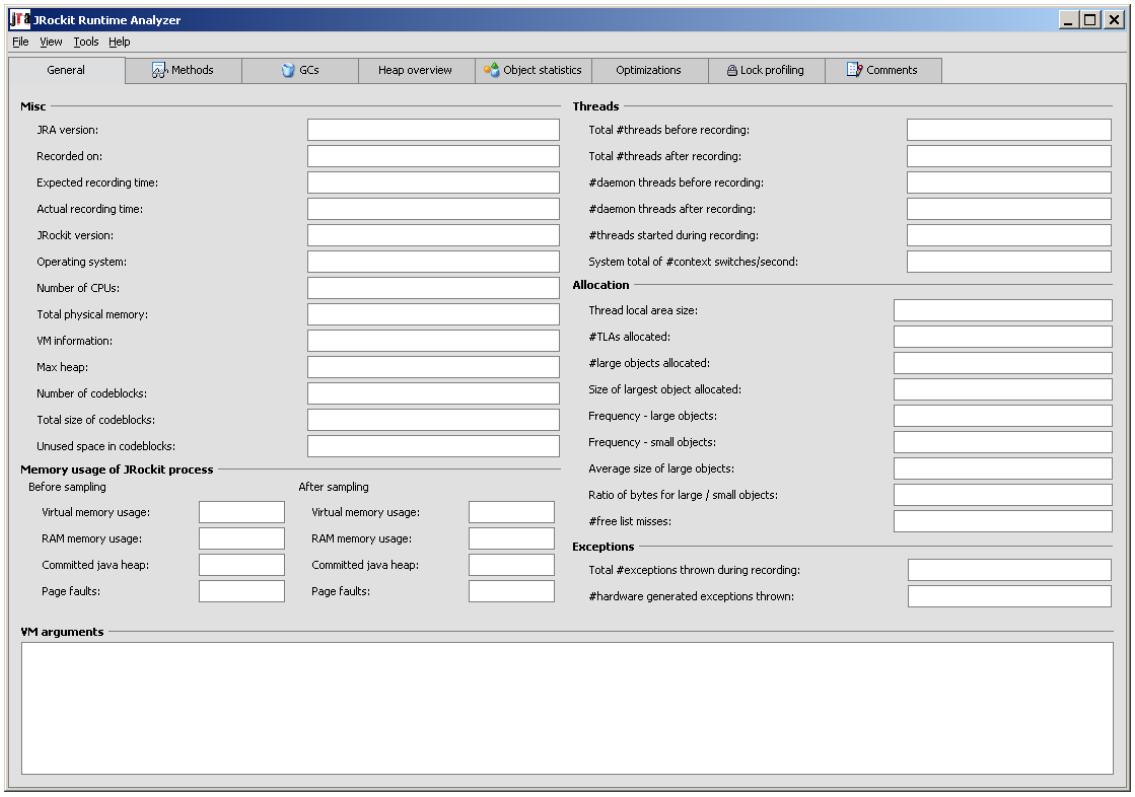
    a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    c. Change to the `<STUDENT>/lib` directory.

    d. To start the Runtime Analyzer, type:

    java –jar RuntimeAnalyzer.jar



33. Load the recorded file in to the BEA JRockit Runtime Analyzer:

    a. Select **Open File** from the **File** menu, and then open the saved file.

    b. The recorded file will have been saved in: `<STUDENT>/bin/jrarecording.xml.encr.zip`

34. Once the file has loaded, click the **Object Statistics** tab. Look for the class (at the end of the recording) that increased its KB allocated by a significant amount. This may be the source of the memory leak.

    In the case of the test application, memory is used by allocating large arrays of bytes of data. Since an array of bytes is a Java primitive (that is, not a Java class), the class name shown for an array of bytes is [B.

    Can you see the memory that has been allocated for the array of bytes?

    Typically, information from tools such as BEA JRockit Runtime Analyzer, JProbe or OptimizeIT is then feed back to the development team to help resolve the issue.

    Do not use any JVMPI tool (such as JProbe or OptimizeIT) in production environments as they can cause significant performance issues due to the monitoring overhead.

Lab cleanup

35. None of the other labs for this course are designed to run with BEA JRockit. Roll back the changes made to switch the system to using the BEA JRockit JVM:

    a. Edit the `setenvBEA.cmd` (Windows) OR `setEnvBEA.sh` (Unix/Linux)

    b. Change the JVM back by resetting the `BEA_JDK` environment variable.

    c. Edit the `startServer.cmd` (Windows) OR `startServer.sh` (Unix/Linux).

    d. Remove the **`-Xmanagement`** switch from the Java line.

    Note: *–Xmanagement is not a switch that is recognized by the Sun JVM and if you try to start the Sun JVM with this switch, it will fail.*

36. Undeploy the lab application:

    a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    To undeploy the application, change to the main directory of the current lab and type:

    ant undeploy

37. Stop the server and close the PointBase window.

Lab 8. Investigating Server Class Cast Exceptions (Optional)

(20 minutes)

Skills Learned

At the end of this lab, you will:

- Understand class inheritance and class casting issues.
- Understand basic class loader issues.

Problem Statement

ClassCastExceptions can occur for various reasons ranging from simple coding (and design) issues to complicated class loading problems. This lab explains how ClassCastExceptions can occur and how to avoid them.

Detailed Instructions

The class hierarchy

1. You have an interface class, Flower, and this class has several descendants that implement it: Hyacinth, Daisy and Rose.

   Assume the following declarations:

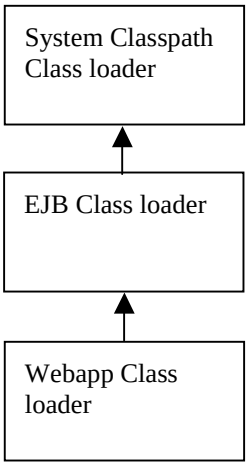
   ```
   Flower f1 = null;
   Hyacinth h1 = null;
   Hyacinth h2 = new Hyacinth();
   Daisy d1 = new Daisy();
   Rose r1 = new Rose();
   ```

   Mark each line of the following code snippet indicating if it is valid or if it will cause a ClassCastException (and explain why):

   ```
   f1 = h2;
   h1 = f1;
   h1 = (Hyacinth) f1;
   h2 = (Hyacinth) f1;
   r1 = f1;
   d1 = (Daisy) f1;
   h1 = (Flower) h2;
   ```

WebLogic Server class loading

2. When WebLogic Server deploys an EAR, it will create a hierarchy of class loaders, each of which inherits the loaded classes of its parent. EJB modules are loaded before Web modules and the Web modules' classloader is a child class loader of the EJB modules' class loader.

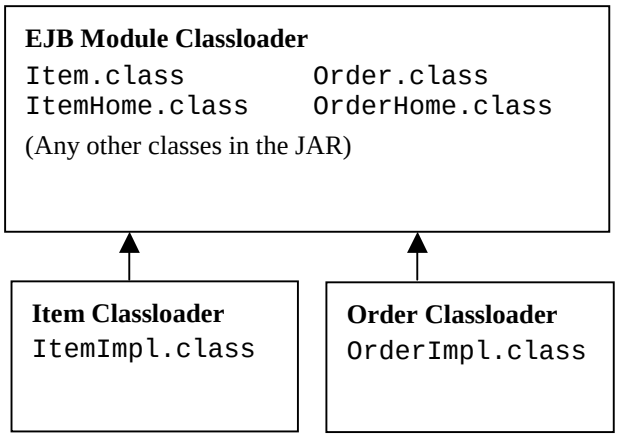


This allows Web modules to access EJB classes without actually having those classes packaged inside the Web module.

Can an EJB class create an instance of a class that is packaged only in a Web module? Why?

Can a servlet in the Web module use classes in the system classpath class loader?

3. Now assume the following situation has been configured for your EJB module:



What advantages does this give you (**Hint:** Reloading of modules requires a new classloader)?

Can `OrderImpl` directly access `ItemImpl`? Why?

Can `OrderImpl` access `ItemImpl` using an EJB call? Explain how?

Can `ItemImpl` be modified and redeployed without affecting `OrderImpl`?

4. For more information on class loading in WebLogic Server, see:

   http://e-docs.bea.com/wls/docs81/programming/classloading.html

Discuss your answers with the class and instructor

5. Once everyone has completed this exercise, work through your answers with the group and discuss the various options.

Lab 9. Troubleshoot a JDBC Problem

(30 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Identify potential issues with JDBC configuration.
- Troubleshoot a JDBC connection pool problem.
- Discuss various JDBC issues.

Problem Statement

JDBC connection pools are the intermediary between the application and the database. Problems can occur with the application and the connection pool if they are not configured properly. The database itself has the potential for many problems, such as a lack of connections, a lock on the table, or the system going down.

Design

This lab has two parts. The first involves the use of a servlet that retrieves JDBC connections from a DataSource, holds them open (or closes them) and possibly might leak connections. By using this tool you will see how JDBC connection pools work and how you can respond to real world connection pool issues.

The second part assumes that you have had a week's holiday and the CEO's nephew filled in for you. Since then many users have been complaining about database connectivity. It is your job to figure out what has happened and how it can be fixed.

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	

Administrator Username	Administrator Password
system	weblogic

Detailed Instructions

Start the WebLogic Server instance

1. a. Windows: If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: If not already running, start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Build and deploy the application

2. a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

3. Change to the main directory of the current lab and type:

   ant

   This builds and deploys the lab application.

Monitor the connection pool

4. Open the system administration console.
   a. Open a Web browser.
   b. Enter the following URL: *http://localhost:7001/console*
   c. Log in using the system user ID and password.

5. View the connection pool. Navigate to dizzyworld > Services > JDBC > Connection Pools > BankConnectionPool > Monitoring

   If the pool is not running, start the pool:
   a. Select the **Target and Deploy** tab.
   b. Select the **AdminServer** as a target.

6. Visit the lab application Web site at *http://localhost:7001/GJDB/jdbcConnections*. Open a small number of JDBC connections.

7. Check the BankConnectionPool monitoring tab in the administration console.

   What happened?

8. Now open zero connections and look at the console. What happened?

9. Enter **–1** (minus 1) as the number of connections to open. This tells the lab application to "leak" a connection. A leaked connection can *not* be closed by the application and it will be permanently lost (unless some mechanism of WLS cleans it up).

   What happens when you monitor the connection pool now?

   Open three more connections (*not* leaked connections) and look at the pool's monitoring data.

   How many connections is the pool using?

   Open zero connections using the lab application, thereby closing the three connections you just opened.

   What happens to the monitoring screen? How many connections are open in the pool? Explain what is happening?

10. Try to open 10 connections.

    Could you do it?

    If not, why?

    How can you remedy this situation?

11. It seems the connection pool is too small to handle the number of connections you want to obtain. Increase the maximum capacity:

    a. Navigate to `dizzyworld > Services > JDBC > Connection Pools > BankConnectionPool > Configuration > Connections`

    b. Increase the **Maximum Capacity** value to **20**.

    c. Apply the change.

    How many connections can you get now?

    What are some potential drawbacks of making such a configuration change?

    Note: *The Maximum Capacity is usually set to the expected maximum number of connections required during peak processing (plus a little more for elbow room and contingencies). The number may also be limited by the number of connections allowed by the license of your database server. In this scenario, we are expecting no more than 20 concurrent connections from this pool.*

Configuration problems

12. The CEO's nephew spent a week managing the server and now you are getting calls from many disgruntled users. They say that they cannot access the database anymore.

    You realize that the high volume transaction users are not complaining and it is only the people running large reports and the batch processing staff that are experiencing the issues.

    Do you have any ideas what might be going on?

13. Upon investigating the programs that are failing, you find that they get a connection from the connection pool, issue several long running SQL statements, as well as perform complex processing between SQL calls.

    a. Turn on JDBC debugging in the administration console. Navigate to `dizzyworld > Servers > AdminServer > Logging > JDBC`. Check **Enable JDBC Logging**.

    b. Change the location of the **JDBC Log File Name** to a location of your choice. **Make sure the directory exists so that the JDBC log file will be created.**

    c. Close the Pointbase window and restart the Admin server.

14. Use the lab application to open a couple of JDBC connections then wait. You should see something happen within 10 - 20 seconds.

    What happens to the monitoring information? What happened to the connections?

15. WebLogic Server cleans up idle connections very quickly. You remember seeing an Inactive Connection Timeout setting in the Advanced Options section of the Connection Pool Configuration.

    Does this setting have anything to do with the problem? What would be a better value?

    After talking to the users that are experiencing the problems, you determine that the longest expected idle time is about 60 seconds on large reports. The applications have no history of connection leaks so you decide it would be safe to set the **Inactive Connection Timeout** to five minutes (**300** seconds).

    Apply this change and use the lab application to test this fix.

16. Look at the other Advanced Options for the connection pool.

    Are they set to the appropriate values? Which settings may have an impact on performance and why?

    In particular, discuss the shrinking, profiling, and connection testing options.

    Are any of these values going to potentially cause a performance problem for this system?

Discuss your answers with the class and instructor

17. Once everyone has completed this exercise, work through your answers with the group and discuss the various options.

Lab cleanup

18. Turn off the JDBC logging.

19. Undeploy the lab application:

    a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    To undeploy the application, change to the main directory of the current lab and type:

    ant undeploy

Lab 10. Troubleshoot a Too Many Open Files Problem

(15 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Identify a WebLogic system that has too many files open.
- Use available tools to determine which component has too many files open and what the files are.

Problem Statement

It is possible with a WebLogic system to open too many file handles and reach an operating system limit preventing the system from opening any more. Typically a WebLogic Server system will not require a large number of open file handles, so having a large number present may indicate a leak or bug. This lab helps you to identify the issue and the core of the problem.

Design

In this lab you will use a custom-built application that deliberately creates a large number of file handles and attempts to overload the number of file handles available for an application on the operating system.

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	

Administrator Username	Administrator Password
system	weblogic

Detailed Instructions

Start the WebLogic Server instance

1. a. Windows: If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: If not already running, start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

Build and deploy the application

2. a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

3. Change to the main directory of the current lab and type:

   ant

   This builds and deploys the lab application.

Create too many file handles

4. Visit the lab application web site at *http://localhost:7001/TMOF/tooManyOpenHandles.*

5. Enter a small number of files, for example 10. Does anything happen?

6. Open a large number of files, for example 10000.

7. What is the error that you receive and what does this mean?

Investigate the currently open files

8. Windows ONLY: Shut down the PointBase window started as part of the server startup script by simply closing (not minimizing) the PointBase window.

9. Get a list of the files that are currently open by the WebLogic Server instance:

   Windows:

   a. Open a Command Prompt window using the `PromptBEA.cmd` script.

   b. Get a list of open files by typing:

      handle -p java.exe >output.txt

   c. Open the `output.txt` document and inspect the data. It will be located in the directory from which you ran the **handle** program.

   Note: *The handle tool has been downloaded and included in the student lab files for use in these labs. You can also download the tool from the www.sysinternals.com site.*

   Linux:

   a. Open a Terminal window window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

   b. Get a list of open files by typing:

      lsof –p <wls-pid> > lsof_output.txt

   c. Get the open files count by typing:

      lsof –p <wls-pid> | wc – l > lsof_count.txt

   d. Open the `lsof_utput.txt, lsof_count.txt` documents and inspect the data. It will be located in the directory from which you ran the `lsof` command.

10. How could you use the information from the **handle** tool OR `lsof` command?

11. Windows ONLY:

    Start Process Explorer by opening a Command Prompt window using the `PromptBEA.cmd` script, changing to the bin directory and typing:

    procexp.exe

    Note: *The Process Explorer has been downloaded and included in the student lab files for use in this lab. You can also download the Process Explorer from the www.sysinternals.com site.*

12. Windows:

Using Process Explorer, determine the following:

- How many file handles does WebLogic Server currently have open?
(***Hint:*** *Look for the java.exe process and look at its properties*)

- How many handles do you currently have open?
(***Hint:*** *Check the lab application web page*)

- Roughly how many handles are taken by WebLogic Server in just running the server?
(***Hint:*** *Look for handles with names pointing to your WebLogic directory*)

Linux:

Using `lsof` output files, determine the following:

- How many file handles does WebLogic Server currently have open?
(***Hint:*** *Look at the lsof_count.txt file*)

- How many handles do you currently have open?
(***Hint:*** *Check the lab application web page*)

- Roughly how many handles are taken by WebLogic Server in just running the server?
(***Hint:*** *Look for handles with names pointing to your WebLogic directory in the lsof_output.txt file*)

13. When debugging a WebLogic Server instance that has too many files open, you will use a combination of the information that you have gathered in the previous steps to analyze the environment and to try to determine the cause of the problem.

    Based on the information that you have gathered, develop a short outline of what you think the problem may be?

    What hints do you have that you can take to the development team or BEA to help try and pin-point the source of the problem?

    What would be your next steps?

Discuss your answers with the class and instructor

14. Once everyone has completed this exercise, work through your answers with the group and discuss the various options.

Lab cleanup

15. Undeploy the lab application:

    a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    To undeploy the application, change to the main directory of the current lab and type:

    ant undeploy

16. Stop the server, to clear all the files.

Lab 11. Troubleshoot a Proxy Plug-in Problem

(15 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Switch on debugging for the WebLogic Server proxy
- Know where to look in the WebLogic Server Console and proxy logs for issues

Problem Statement

WebLogic Server proxy issues can occur for a number of reasons. This lab looks at how to enable debugging in the WebLogic Server proxy so that an administrator can more closely monitor the performance of the WebLogic Server proxy plug-in.

Design

To allow the labs to be deployed in a simple manner, this lab uses the WebLogic Server proxy plug in. However, the same result can be achieved using the Apache, IIS or NES (Netscape Enterprise Server / IPlanet) plug-ins for WebLogic Server.

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	

Administrator Username	Administrator Password
system	weblogic

Detailed Instructions

Start the servers

1. a. Windows: If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: If not already running, start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.

2.	a. Windows: If not already running, start the ServerA using the `startServerA.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: If not already running, start the ServerA using the `startServerA.sh` script found in the `<STUDENT>/bin` directory.
3.	a. Windows: If not already running, start the ServerB using the `startServerB.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: If not already running, start the ServerB using the `startServerB.sh` script found in the `<STUDENT>/bin` directory.
4.	a. Windows: If not already running, start the ProxyServer using the `startProxyServer.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: If not already running, start the ProxyServer using the `startProxyServer.sh` script found in the `<STUDENT>/bin` directory.
Build and deploy the application	
5.	a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.
6.	Change to the main directory of the current lab and type: **ant** This builds and deploys the lab application.
Ensure the lab application is deployed	
7.	Open a web browser and browse to *http://localhost:7001/console*. Log in using the system user ID and password.
8.	Click **Deployments** in the menu tree and then click **Web Application Modules**.
9.	Select the **GDPP.war** file in the menu tree and click the **targets** tab at the top of the panel. Ensure that the application is targeted at all servers in the cluster. Target and apply if necessary.

10. Make sure that the lab application is deployed to all servers in the cluster by clicking the **Deploy** tab. Redeploy if necessary.

Test the load balancing

11. Open two separate Web browser windows.

12. In the first Web browser, access the following URL:
http://localhost:8888/GDPP/sessionReplication

    The server instance running on port 8888 is the ProxyServer instance. This server instance takes the request and forwards it on to one of the other server instances, in this case, the ServerA instance or the ServerB instance.

13. In the second browser, access the same URL using the icon, not by selecting **New** in the **File** menu.

14. In each Web browser, put a value in the session and check the server consoles for ServerA and ServerB. You should see a message indicating the use of each server by each browser instance.

Turn on logging

15. Switch debugging on for the proxy application.

    a. Open the file:
    `<STUDENT>/domains/dizzyworld/applications/proxy/WEB-INF/web.xml`

    b. Add the following lines in the `<servlet>` section:

```
<init-param>
  <param-name>Debug</param-name>
  <param-value>true</param-value>
</init-param>
```

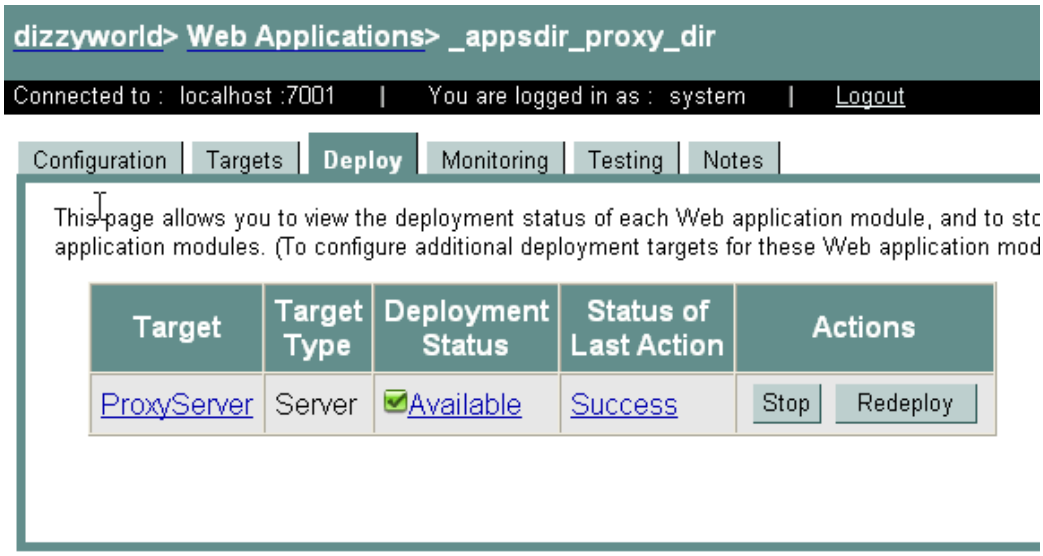
    c. Check to see if you have a `temp` directory in your drive C. If you do not, create one.

    Note: For other proxy plug-ins, the Debug setting is usually set to ALL; however when using the WebLogic Server proxy application, the Debug parameter should be set to true.

16. Redeploy the proxy application in the administration console.

    a. Select the proxy application by browsing to:
    Deployments > Web Application Modules >_appsdir_proxy_dir

    b. Click the **Deploy** tab and redeploy the application.



Check the log

17. In a browser, access the following URL:

    http://localhost:8888/GDPP/sessionReplication

18. Check the `c:\temp` (Windows) OR `/tmp` directory for a file called `wlproxy.log`. Open the log and examine it to see what it contains.

Troubleshooting the Problem	
19.	Typically, when you are debugging problems with the wlproxy, switching on the Debug and then looking at the logs is a good place to start.
Discuss your answers with the class and instructor	
20.	Once everyone has completed this exercise, work through your experiences with the group and discuss any questions raised.
Lab cleanup	
21.	Remove the debug switch from the web.xml in the proxy application. Redeploy the application.
22.	Undeploy the lab application: a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory. To undeploy the application, change to the main directory of the current lab and type: **ant undeploy**
23.	Shut down ServerA, ServerB, and ProxyServer.

Lab 12. Investigating HTTP Session Replication Failures

(30 minutes)

Skills Learned

At the end of this lab, you will be able to:

- Understand how the WebLogic Server proxy server works with respect to HTTP session replication and failover.
- Discuss various aspects of HTTP session replication configuration and operation.

Problem Statement

WebLogic Server can run with a cluster of servers servicing client requests. This can assist in areas such as failover, security, and performance. For failover, it is desirable for a WLS Server to replicate its HTTP session data on another node. If the server fails, the other node can seamlessly take over the failed server's clients.

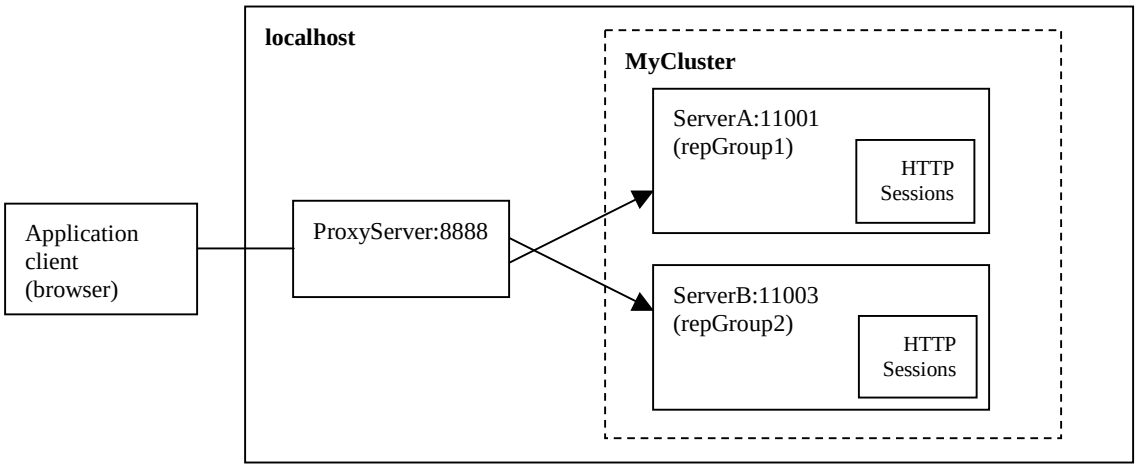
For this to happen, a proxy server is required as the contact point for the clients. The proxy server will redirect the requests from the failed server to the failover server.

Design

This lab consists of a WLS administration server, AdminServer, and two remote managed servers, ServerA and ServerB. These two servers are configured in a cluster and they replicate HTTP sessions between each other for failover. This can only happen if the *Preferred Secondary Group* for each server's cluster configuration specifies the other server's replication group.

Another WLS server, ProxyServer, is also configured as a proxy server with a round-robin strategy for load balancing. The HTTP session replication for this lab is in-memory replication.

For the purposes of this lab, all of the servers are running on the local host but each server will have its own port number. The following diagram shows the setup as well as the port numbers and the replication groups:



A lab application has been created to demonstrate the round-robin balancing and how it is affected by session replication. Each time it is invoked, it will print a message to the console of the server that serviced the request. You can instruct it to put a value in the session so you can then observe the routing and failover activity when a server fails.

Two servers in the cluster nominate the other server's replication group as their "Preferred Secondary Group" in the cluster configuration for the server.

Environment Variable	Example	Description
STUDENT	Windows: `c:\student\course_wls_troubleshooting`	The installation directory for the labs.
	Unix/Linux: `{yourhome}/student/course_wls_troubleshooting`	

Administrator Username	Administrator Password
`system`	`weblogic`

Detailed Instructions

Start the WebLogic Server instance	
1.	a. Windows: If not already running, start the Administration Server using the `startServer.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: If not already running, start the Administration Server using the `startServer.sh` script found in the `<STUDENT>/bin` directory.
Start and access the WebLogic Server cluster and proxy	
2.	a. Windows: If not already running, start the ServerA using the `startServerA.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: If not already running, start the ServerA using the `startServerA.sh` script found in the `<STUDENT>/bin` directory.
3.	a. Windows: If not already running, start the ServerB using the `startServerB.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: If not already running, start the ServerB using the `startServerB.sh` script found in the `<STUDENT>/bin` directory.
4.	a. Windows: If not already running, start the ProxyServer using the `startProxyServer.cmd` script found in the `<STUDENT>\bin` directory. b. Linux: If not already running, start the ProxyServer using the `startProxyServer.sh` script found in the `<STUDENT>/bin` directory.

Build and deploy the application

5. a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

   b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

6. Change to the main directory of the current lab and type the following:

   ant

   This builds and deploys the lab application.

7. Visit the lab application at *http://localhost:8888/HSRF/sessionReplication.* Check the console output from the two servers in the cluster to verify where the application executed.

   Refresh the page a couple of times and recheck the consoles.

   What do you notice?

Play with the HTTP session

8. Select **Put value in session** and click **Submit** to submit the form. This puts the current date/time into the HTTP session.

   Look at the console outputs and notice which server executed the request. This server will be the primary server for that client now. The other server should be the secondary.

9. Select **Get value from session** and click **Submit**. You will see the time that the Put value in session operation was performed. Perform multiple get and put operations and look at the console output for them.

   Do you understand what is going on? Why isn't the round-robin balancing happening?

10. Select **Clear value from session** and click **Submit**. This removes the date value we put into the session earlier. The session is now empty.

    Try getting the value from the session.

    What happens? What is the round-robin activity?

Inspect the configuration

11. Display cluster data for ServerA.

    a. Open a Command Prompt/Terminal window using the `PromptBEA.cmd` (Windows) OR `promptBEA.sh` (Linux) script.

    b. Run the following:
    java weblogic.Admin -url t3://localhost:11001 -username system -password weblogic GET -type ClusterRuntime -pretty

12. Repeat the preceding steps for ServerB and ProxyServer. You will need to change the port number for each server.

13. What do you notice?

    Is ProxyServer part of the cluster?

14. View the cluster configuration for ServerA. Browse to dizzyworld > Servers > ServerA > Cluster in the administration console.

15. Repeat the previous step for ServerB. This is where the replication groups are defined.

    Explain how the replication group setup works.

Time to failover

16. Using the lab application, put some more values into the session.

17. Shut down the server that has been processing the requests.

18. Using the lab application, get the session value.

    Were any messages displayed by the lab application on the other server's console?

    Did you get the same date/time that was put in the session on the original server?

    What has happened?

Optional

19. Turn on extra replication debugging.

    a. Open a Command Prompt/Terminal window using the `PromptBEA.cmd` (Windows) OR `promptBEA.sh` (Linux) script.

    b. Run the following commands:

    java weblogic.Admin -url t3://localhost:7001 -username system -password weblogic SET -type ServerDebug -property DebugCluster true

    java weblogic.Admin -url t3://localhost:7001 -username system -password weblogic SET -type Server -property StdoutDebugEnabled true

    ***Note**: The property can be DebugCluster, DebugClusterAnnouncements, DebugFailOver, DebugReplication or DebugReplicationDetails.*

20. Restart the server that was shut down. After it has started successfully, visit the lab application and put a value in the session.

21. Shut down the other server (the one that now holds the HTTP session data) and the session should switch to (a failover) to the newly booted server.

    Did the extra debugging give you any more information?

Discuss your answers with the class and instructor

22. Once everyone has completed this exercise, work through your answers with the group and discuss the various options.

Lab cleanup

23. Undeploy the lab application:

    a. Windows: Open a Command Prompt window using the `PromptBEA.cmd` script found in the `<STUDENT>\bin` directory.

    b. Linux: Open a Terminal window using the `promptBEA.sh` script found in the `<STUDENT>/bin` directory.

    To undeploy the application, change to the main directory of the current lab and type:

    ant undeploy

24. Shut down ServerA, ServerB, and ProxyServer.