

Gauging ADF Application Performance: Instrumenting Your Code



COLLABORATE15

TECHNOLOGY AND APPLICATIONS FC
FOR THE ORACLE COMMUNITY



APRIL 12-16, 2015
MANDALAY BAY
RESORT & CASINO

#C15LV

Presented by: John Jay King
Download this paper from:
<http://www.kingtraining.com>

Copyright @ 2015, John Jay King





COLLABORATE15
TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Session Objectives

- Learn how Oracle and Oracle WebLogic server support tuning instrumentation
- Be able to instrument ADF to track performance
- Use performance tracking data to improve ADF application performance

Who Am I?

- John King – Partner, King Training Resources
- Oracle Ace Director 
- Member Oak Table Network 
- Providing training to Oracle and IT community for over 25 years – <http://www.kingtraining.com>
- “Techie” who knows Oracle, ADF, SQL, Java, and PL/SQL pretty well (along with other topics)
- Member of AZORA, ODTUG, IOUG, and RMOUG

Planes Have Instruments: Why?

- Most of us have been aircraft passengers (probably recently)
 - Aren't you glad the plane's crew had instruments to monitor things?
 - Would you trade the cost/complexity of the instruments for the loss of information?



Code Instrumentation

- Instrumentation is the thoughtful act of creating code that allows monitoring and measurement of execution to facilitate debugging and performance improvement
- Instrumentation provides:
 - Meaningful information about what, where, and why something is happening
 - Timing information in useful increments
 - Logs are a key tool used to provide instrumentation output

- Two (among many) Oracle performance experts who advocate instrumentation:
 - Cary Millsap
<https://method-r/fogbuz.com/default.asp?W265>
 - Tom Kyte
<http://tkyte.blogspot.com/2005/06/instrumentation.html>

- How do you know if your application is running properly?
 - Correctness of input/output processing
 - Execution speed within user targets
- When is the cost too high?
 - Tracking everything all the time provides mountains of not-very-useful data and can impede normal execution
 - Maybe; tracking should be “switchable” to be enabled when needed and disabled when not

Tuning

- What is acceptable performance? Have your users provided reachable targets?
- If something “takes too long” – it’s important to know where time is spent
- Instrumentation must provide meaningful information about when processes begin, when they end, and how long the activities being performed last
- Once problem areas are highlighted specific issues may be addressed

- You need a strategy
 - Too little instrumentation; can't use it
 - Too much instrumentation; masses of data that are hard to use
 - How will instrumentation take place?
 - Home-grown?
 - Using Oracle built-in facilities?
 - Using vendor-provided facilities?
 - some combination of the above?

Say “No” to System.out.println

- Many Java developers use console output via System.out.println for rudimentary instrumentation during testing
 - Production system consoles are often unmonitored
 - Production system consoles (today) are often part of a virtualized server; never to be seen...
- Instrumentation needs to survive the ups and downs of the JVM and be broadly available; logging provides many options

- You shouldn't use online debug in production
- Traditionally, IT uses logging as a preferred method for collecting information about application effectiveness and efficiency
- ADF exists in the Java world where several logging tools are available including:
 - Java SE `java.util.logging`
 - Log4J
 - Apache Commons Logging
 - ADFLogger built into ADF



Planning for Logs

- Where will logs go? (console/XML/text/etc.)
- What level of Detail/Content?
 - Name of package & name of module
 - Name of method/procedure/function
 - Variable and parameter values in use at time
 - Applicable error messages/codes
(Messages and/or codes?)
 - Date and time
 - Who will consume the log? Is translation needed? (user/admin/dba/support)

ADFLogger

- Oracle's ADF team recognized the need for instrumentation and provided "ADFLogger"
- ADFLogger provides a log mechanism fully integrated with ADF via `java.util.logging` "under the covers" with added functionality
- ADFLogger works effectively both within JDeveloper and from Enterprise Manager
- ADFLogger may be switched on and off without restart

ADFLogger Levels

- Like most logging tools; ADFLogger divides log entries into several classifications:
 - SEVERE (fewest log entries)
 - WARNING
 - INFO
 - CONFIG
 - FINE
 - FINER
 - FINEST (most log entries)

- ADFLogger provides some methods of its own in addition to those inherited from `java.util.logging`; including:
 - `begin()`
 - `end()`
 - `log()`
 - `severe()`, `warning()`, `info()`, `config()`, `fine()`, `finer()`, `finest()`

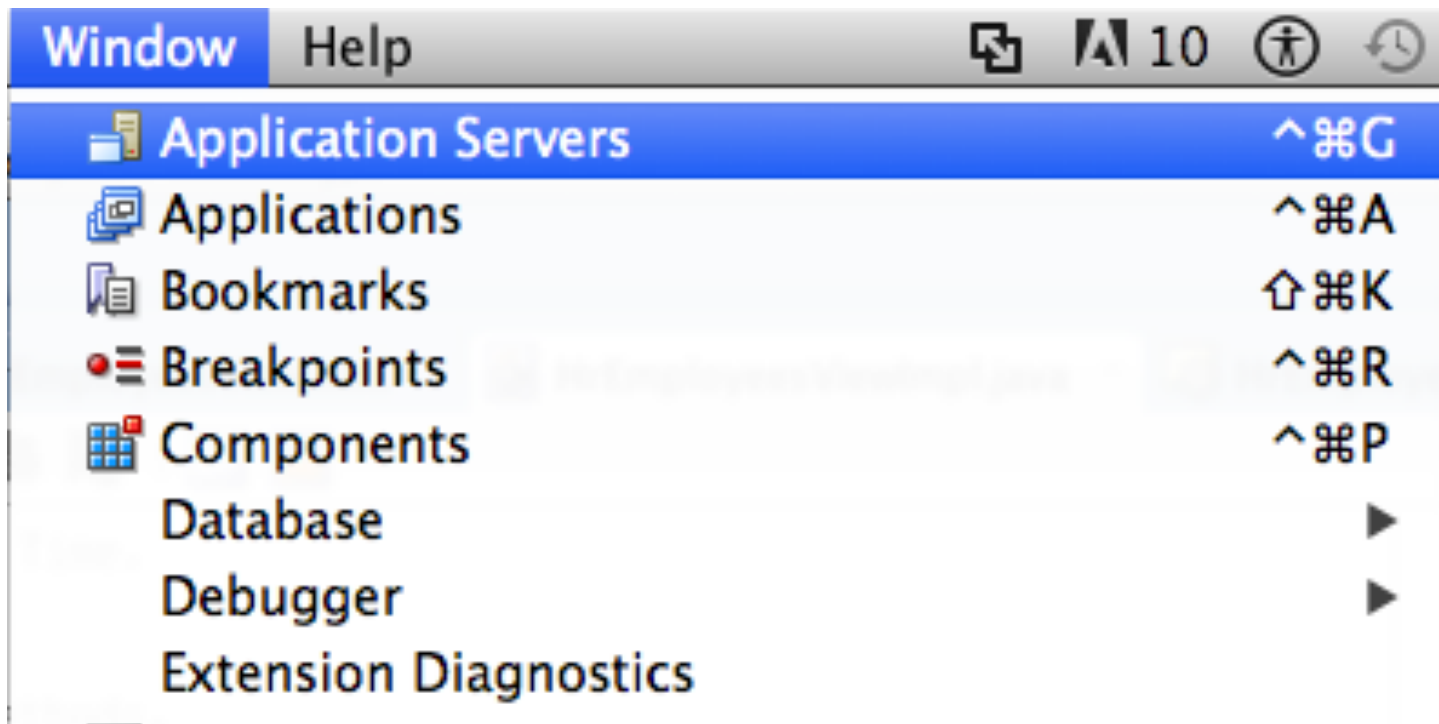
- Several ADFLogger methods and constructors require HashMap input parameters
- It is probably simplest to create a utility method, interface and/or superclass class for your team rather than having everyone build basic logging logic themselves

ADF & WebLogic

- ADF is fully integrated into WebLogic; including ADFLogger and its tooling
 - A “logging.xml” file describe ADFLoggers, their default level, and the handlers used for them
 - JDeveloper has dialog-based configuration via “Oracle Diagnostics Logging Configuration”
 - JDeveloper also has “Oracle Diagnostic Log Analyzer” tool to review log output
 - Production support via Enterprise Manager
(farm->WebLogicDomain->appcluster->YOURSERVER->logs)

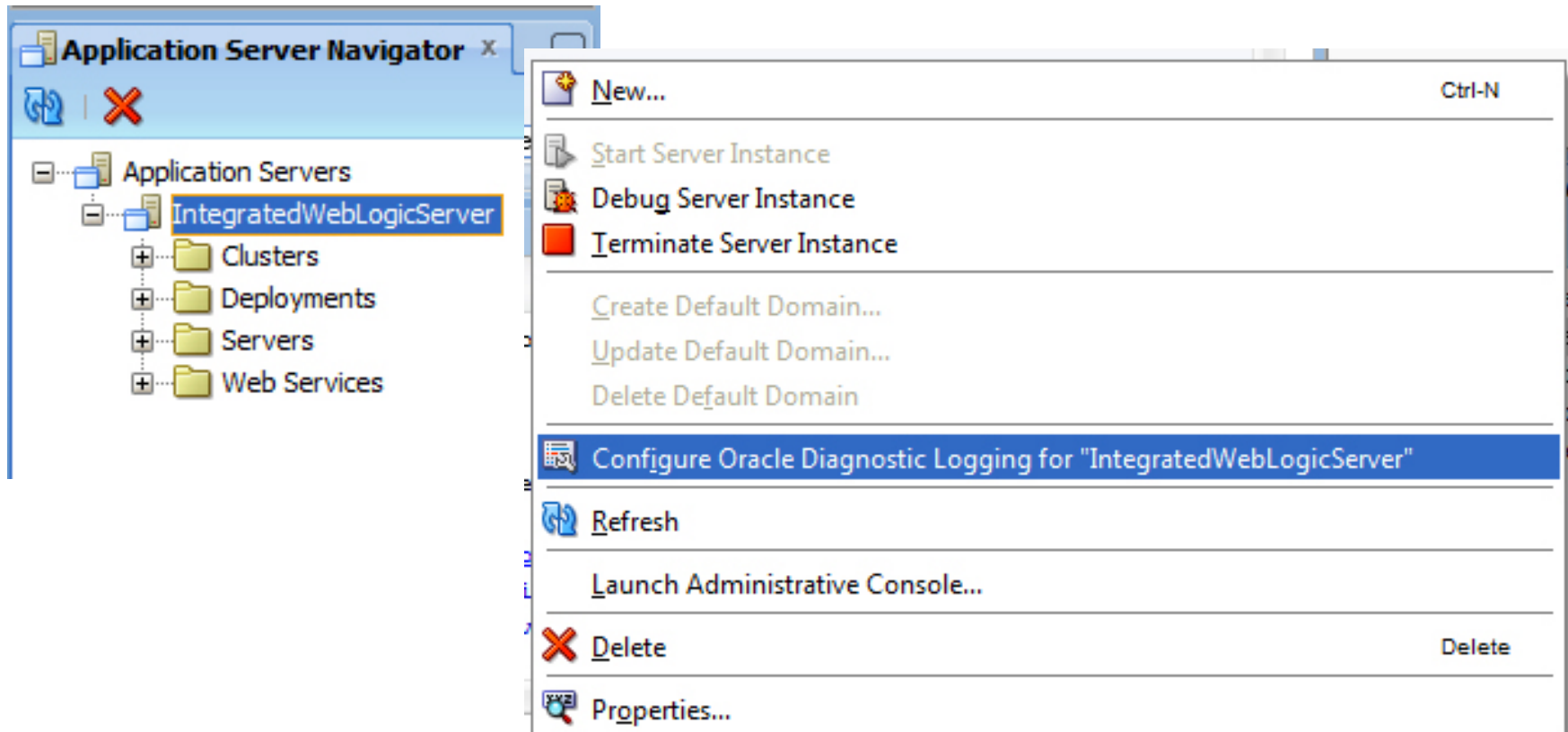
JDeveloper Logging: 1

- First open Window->Application Servers if it is not already open
(11g: View-> Application Server Navigator)



JDeveloper Logging, 2

- Next use IntegratedWebLogicServer's context menu (right-click) and choose "Configure Oracle Diagnostics Logging"



- ADFLogger is configured with “logging.xml” file; click the “source” tab to manipulate the XML or use the panel displayed (preferred)

Oracle Diagnostics Logging Configuration

Control logging behavior for specified [loggers](#). If the server is running, changes take effect immediately. Any transient instance loggers you add exist only in the context of the running server, and are not persisted to the logging.xml file.

Loggers:

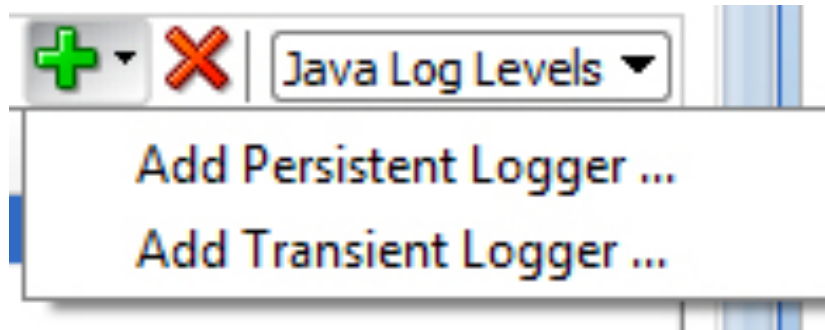
☐ Hide Transient Loggers + - ✕ Java Log Levels ▾

Name	Level	Declares Handlers
Root Logger (default)	WARNING	
com		
global		
javax		
jrf		
oracle	WARNING	
org		
sun		
weblogic		

Overview Source

Add Custom Logger, 1

- To add your own logger click the green “plus sign” icon and choose whether to create a persistent or transient logger



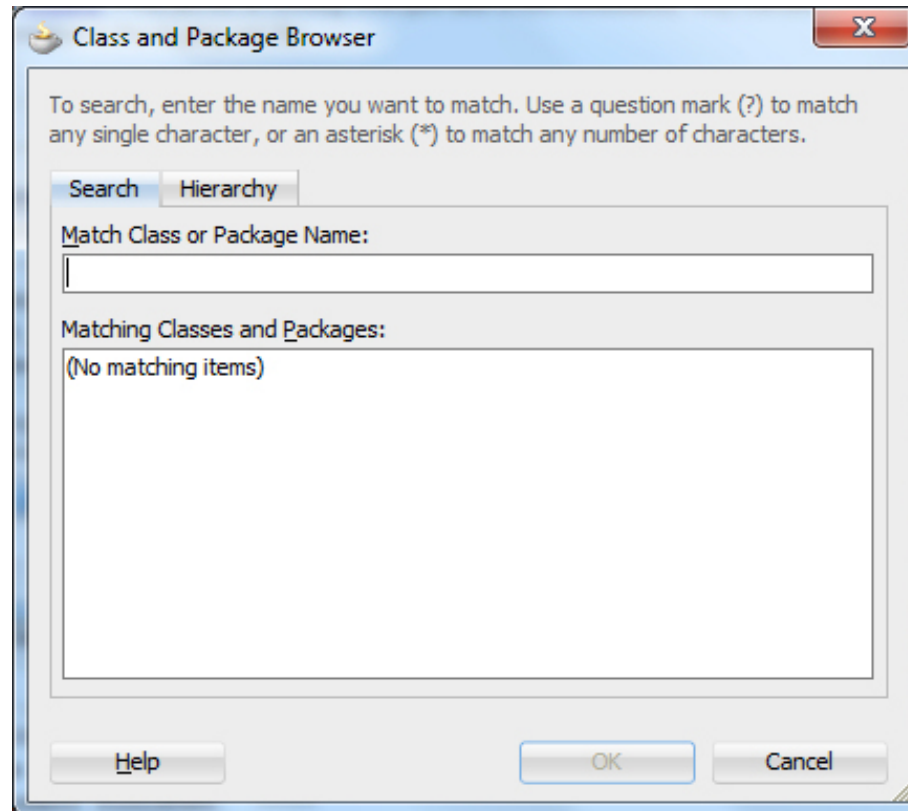
- Persistent logger
- Transient logger

Stays after WebLogic shutdown

Available until WebLogic shutdown

Add Custom Logger, 2

- Name the logger; if you enter a Java class name the class name and its package name will be used to identify log entries



Add Custom Logger, 3

- You may also use a textual name; take time to plan what serves you best
- Choose the default level for messages associated with this logger

Add Persistent Logger

Persistent Loggers added while the server is running take effect as soon as you click OK, and are written to the configuration file.

Logger Name:

Loggers are conventionally named with the corresponding class or package name when declared.
Example: oracle.adf.share

Logger Level:

[Help](#)

Adding Custom Logger, 4

- Custom logger(s) show in the configuration dialog with the Oracle-provided loggers

DemoApplication1 Overview x Demo1.jsf x logging.xml x

Oracle Diagnostics Logging Configuration

Control logging behavior for specified [loggers](#). If the server is running, changes take effect immediately. Any transient instance loggers you add exist only in the context of the running server, and are not persisted to the logging.xml file.

Loggers:

☐ Hide Transient Loggers + - X Java Log Levels ▾

Name	Level	Declares Handlers
Root Logger (default)	WARNING	
com		
demoLogger	INFO	
global		
javax		
jrf		
oracle	WARNING	
org		
sun		
weblogic		

Overview | Source

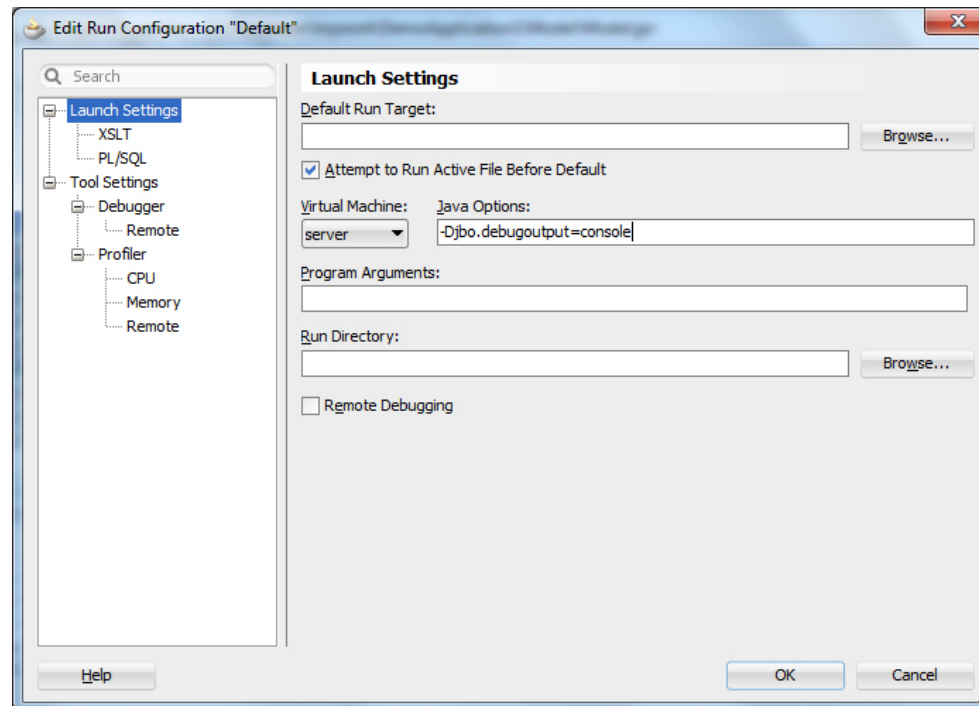
Enable Built-In Loggers

- Oracle has instrumented just about everything; probably too many choices
- A good base set is oracle.adf, oracle.adfinternal, and oracle.jbo (set level)

Name	Level	Declares Handlers
Root Logger (default)	WARNING	
com		
demologger	INFO	
global		
jaxax		
jrf		
orade	WARNING	
orade.adf	CONFIG	
orade.adfdt		
orade.adfinternal		
orade.adfinternal	CONFIG	
orade.as		

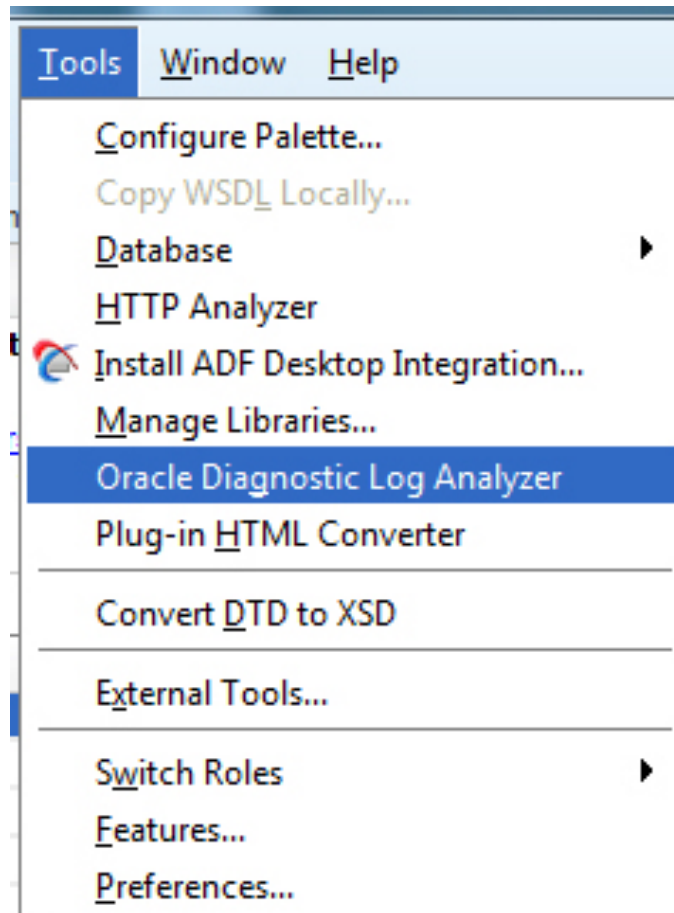
Step No Longer Needed?

- When using the built-in configuration tool; a runtime setting is automatically set; in earlier versions of ADF this required a restart of WebLogic (sometime still useful)



Built-In Log Analyzer, 1

- Use the “Tools” menu and select the “Oracle Diagnostic Log Analyzer” to open



Built-In Log Analyzer, 2

- Choose which loggers and levels to view:

DemoApplication1 Overview x Demo1.jsf x logging.xml x **Oracle Diagnostic Log Analyzer** x

Log: C:\Users\John\AppData\Roaming\JDeveloper\system11.1.2.4.39.64.36.1\DefaultDomain\servers\DefaultServer\|...

By ADF Request By Log Message

Search

Most Recent: 10 Requests

Log Time: Most Recent 1 Hours

Request Header Contains

Search Reset

Results:

(To begin, select Oracle Diagnostic Log files and search criteria then click Search button.)

Using Tracking Data, 1

- Select “By ADF Request” for details, times, and percentages

The screenshot shows the Oracle Diagnostic Log Analyzer interface. The 'By ADF Request' tab is selected. The search criteria are set to 'Most Recent' (10 Requests) and 'Log Time' (Most Recent, 1 Hours). The search results show 37 requests found, with 10 displayed. The results are sorted by time and percent.

ADF Message	Message Data	Time (ms)	Percent
ADF web request	URL =http://localhost:7101/DemoApplication1-ViewC...	77	
JSF Phase RESTORE_VIEW 1		29	
JSF Phase RENDER_RESPONSE 6		32	
ADF Web Request: 2014-05-28 18:05:55 /faces/Demo1.jsf		14087 ms	
ADF Web Request: 2014-05-28 18:05:55 /faces/Demo1.jsf		89 ms	
ADF Web Request: 2014-05-28 18:05:54 /faces/Demo1.jsf		333 ms	
ADF Web Request: 2014-05-28 18:05:50 /faces/Demo1.jsf		89 ms	
ADF Web Request: 2014-05-28 18:05:49 /faces/Demo1.jsf		142 ms	
ADF Web Request: 2014-05-28 18:05:48 /faces/Demo1.jsf		702 ms	

Using Tracking Data, 2

- Select “By Log Message” to see very detailed specifics

The screenshot shows the Oracle Diagnostic Log Analyzer application. The 'Log' field is set to 'C:\Users\John\AppData\Roaming\JDeveloper\system11.1.2.4.39.64.36.1\DefaultDomain\servers\DefaultServ...'. The 'By ADF Request' and 'By Log Message' tabs are visible, with 'By Log Message' selected. The 'Search' section includes a 'Java Log Level' dropdown, checkboxes for log levels (Severe, Warning, Info, Config, Fine, Finer, Finest, Unknown), a 'Log Time' section with 'Most Recent', '1', and 'Hours' dropdowns, and a 'Message' section with 'Contains' and a search input field. The 'Search' and 'Reset' buttons are at the bottom right of the search section. The results section shows 'Results: 1000+ found (1000 displayed). May 28, 2014 6:08 PM' and a 'Group by Id' checkbox. The results table has columns: Time, Message, Module, Related, Mess..., Type, and Appli... The table lists several log entries, including '[1567] Releasing iterator binding:vcRowsIterator', '[1568] Releasing iterator binding:variableIterator', '[1563] Releasing iterator binding:DeptView1Iterator', and '[1564] released: ROOT node binding:noCtrl_oracle_adfinternal_view_face...'. The first row is highlighted in blue.

Time	Message	Module	Related	Mess...	Type	Appli...
May 28, 2014 6:...	[1567] Releasing iterator binding:vcRowsIterator				FINEST	DemoA...
May 28, 2014 6:...	[1568] Releasing iterator binding:variableIterator				FINEST	DemoA...
May 28, 2014 6:...	[1563] Releasing iterator binding:DeptView1Iterator				FINEST	DemoA...
May 28, 2014 6:...	[1564] released: ROOT node binding:noCtrl_oracle_adfinternal_view_face...				FINEST	DemoA...
May 28, 2014 6:...	[1565] released: ROOT node binding:noCtrl_oracle_adfinternal_view_face...				FINEST	DemoA...
May 28, 2014 6:...	[1566] Releasing iterator binding:EmpView2Iterator				FINEST	DemoA...

Using Tracking Data, 3

#C15LV

```

755 <ADFLogger> <begin> Evaluate Expression
756 <ADFLogger> <end> Evaluate Expression
757 <ADFLogger> <begin> Executing iterator binding
758 <ViewRowSetImpl> <execute> [589] EmpView2 ViewRowSetImpl.execute caused params to be "un"changed
759 <ViewRowSetImpl> <initQueryCollection> [590] Carrying over CappedRowCount:-1for ViewRowSet:EmpView2
760 <QueryCollection> <createColumnList> [591] Column count: 8
761 <ViewRowSetImpl> <execute> [592] executeQueryForCollection ViewObject:EmpView2, RowSet:EmpView2
762 <ADFLogger> <begin> Execute query
763 <ViewObjectImpl> <closeStatementsResetRowSet> [593] ViewObject: [samples.model.EmpView]DemolAppModule.EmpView2
764 <ViewObjectImpl> <getPreparedStatement> [594] ViewObject: [samples.model.EmpView]DemolAppModule.EmpView2
765 <ViewObjectImpl> <buildQuery> [595] EmpView2:#q computed SQLStmtBufLen: 246, actual=191, storing=221
766 <ViewObjectImpl> <buildQuery> [596] SELECT Emp.EMPNO, Emp.ENAME, Emp.JOB, Emp.MGR
767 <ViewObjectImpl> <bindParametersForCollection> [597] Bind params for ViewObject: [samples.model.EmpView]DemolAppModule.EmpView2
768 <OracleSQLBuilderImpl> <bindParamValue> [598] Binding param "Bind_Deptno": 10
769 <ADFLogger> <addContextData> Execute query
770 <ADFLogger> <addContextData> Execute query
771 <ADFLogger> <end> Execute query
772 <EntityImpl> <registerExprValueSupplierOverride> [599] Entity with key:oracle.jbo.Key[7782 ] owned by row
773 <ADFLogger> <end> Executing iterator binding
774 <JUCtrlValueBinding> <startEvents> [600] noCtrl_oracle_adfinternal_view_faces_model_binding_FacesCtrlHierarchy
775 <JUCtrlValueBinding> <startEvents> [601] Empno: IGNORING Start events as it is not in active/push mode
776 <JUCtrlValueBinding> <startEvents> [602] Ename: IGNORING Start events as it is not in active/push mode
777 <JUCtrlValueBinding> <startEvents> [603] Job: IGNORING Start events as it is not in active/push mode

```

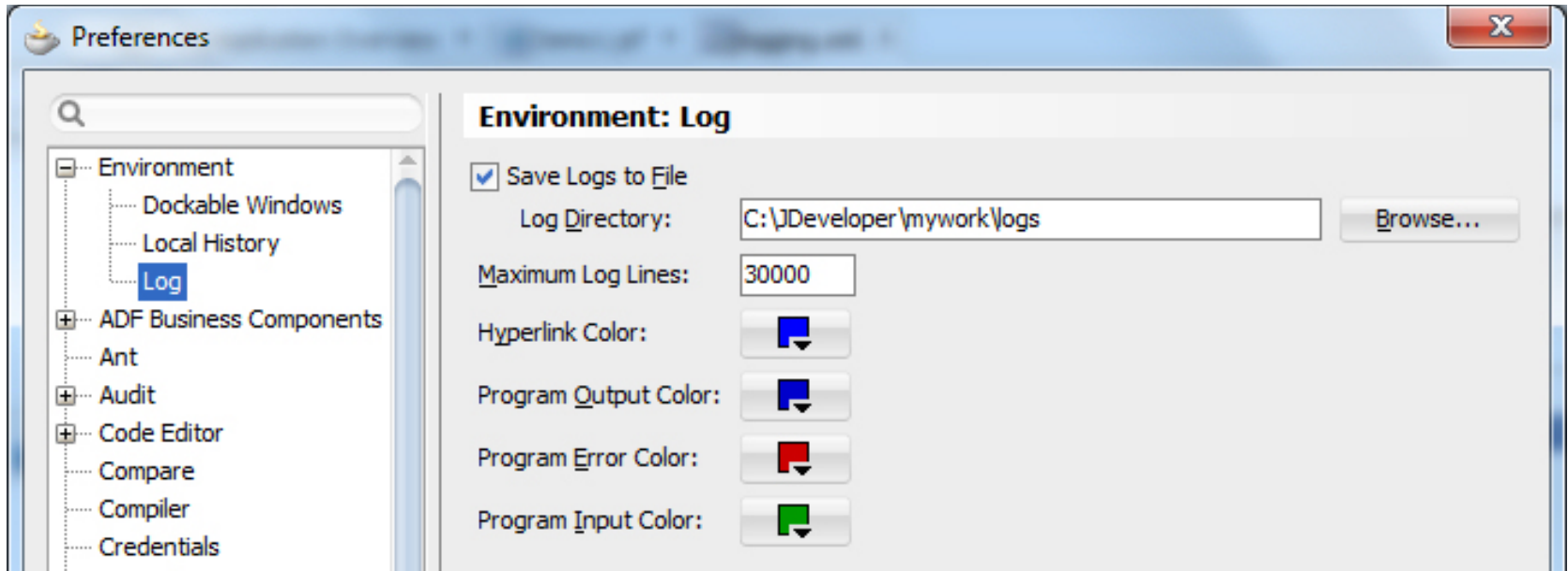
Normal text file length: 406246 lines: 5097 Ln: 766 Col: 37 Sel: 0 Doc:\Windows ANSI BNC



Save Logs to File

- Use JDeveloper's preferences to save logs to a file in text form (probably increase maximum log lines, default is 300)

**JDeveloper->Preferences->Environment->Log
(11g: Tools->Preferences->Environment->Log)**



- ADF generates a text file that looks like:

```
<DCExecutableBinding> <refreshIfNeeded> [815] Invoke refresh for :DeptView1Iterator
<DCIteratorBinding> <refresh> [816] Executing and syncing on IteratorBinding.refresh
from :DeptView1Iterator
<ADFLogger> <begin> Attaching an iterator binding to a datasource
<DCIteratorBinding> <getViewObject> [817] Resolving V0:DeptView1 for iterator
binding:DeptView1Iterator
<ADFLogger> <addContextData> Attaching an iterator binding to a datasource
<ADFLogger> <end> Attaching an iterator binding to a datasource
<ADFLogger> <begin> Executing iterator binding
<ADFLogger> <end> Executing iterator binding
<DCExecutableBinding> <refreshIfNeeded> [818] Invoke refresh for :EmpView2Iterator
<DCIteratorBinding> <refresh> [819] Executing and syncing on IteratorBinding.refresh
from :EmpView2Iterator
<ADFLogger> <begin> Attaching an iterator binding to a datasource
<ADFLogger> <addContextData> Attaching an iterator binding to a datasource
<ADFLogger> <end> Attaching an iterator binding to a datasource
<ADFLogger> <begin> Executing iterator binding
<ADFLogger> <end> Executing iterator binding
<ADFLogger> <end> Refreshing binding container
<DCBindingContainerState> <validateToken> [820] Process BindingContainer state token(decompressed
state):BCST:=0%V%=NDeptView1Iterator=-D-,EmpView2Iterator=-D-,
%R=0%EmpView2Query=BCST:=0;#;eAF1Uj1vE0EQHScESCKQFSrNDQUEc3dDzBC0n8pJ85ngx0ruhRhfd01uzeHnt7zoUCCR
qEREED
LR0VLT2iBVHyL2j4A2b0sZFBYqdY7eyb90bz5uNPWMs0VI68MZswS7D4x0oMxxia6pvvh+/L6R2x
ApAnALBKuN0jT2kWCrTGQ2UNm0ZsKLD65eWjzx9e7d9bQEuphr05kEUjS6oIhTXkccSJv1FffLoy
EXfbP75+0twIi10nt5boQyWliv+oF0AXN979+vb2dbgK2zXYaXUetp3+seu30sf0wHE9p+Y1a3Bz
0e93+v/90/Bd3+27jucGzUYAl2Wda3Neg+tyn6UtpSUzbjxSAexKN23giMcY0cUyYQZMBjAJenj
WQBxpa+MnwkrWjBssjTLE2a4in3ECCMXNmRXY8hTSrkk0wuZwBpsyomkYgaLEXr0qqvYYG7aLDFQ
mflhZ4Ylm5o5pWTvg2tyWd7A1gWqcM2eu+bBumzmiW50mDCwM7fLJrvsRXpG1MwNxtRcgxn2BJ7B
FQ/W5H1y6G/WntFKGVVsy07mJxcibM4v80ol7VwKm3rr0k55amZL4UFZ9pA2RPCnq0uCxliIrXuw
LfuaxSnH2BQ9Ic1Lxb0/PCnBlKJEQWc6nT5fSebnbA8qVBc+fpChPieLETmEEZT/3dwkT5Lfhfvz
4w==
: # ,
<LoopDiagnostic> <dump> [821] variableIterator variables activated <<< [U]:TrackQueryPerformed=null
<ADFLogger> <end> ADF Phase prepareModel
```

SQL Statement Log

- Default oracle.jbo logging shows SQL

```
Running_IntegratedWebLogicServer_20140529092602.log
755 <ADFLogger> <begin> Evaluate Expression
756 <ADFLogger> <end> Evaluate Expression
757 <ADFLogger> <begin> Executing iterator binding
758 <ViewRowSetImpl> <execute> [589] EmpView2 ViewRowSetImpl.execute caused params to be "un"changed
759 <ViewRowSetImpl> <initQueryCollection> [590] Carrying over CappedRowCount:-1for ViewRowSet:EmpView2
760 <QueryCollection> <createColumnList> [591] Column count: 8
761 <ViewRowSetImpl> <execute> [592] executeQueryForCollection ViewObject:EmpView2, RowSet:EmpView2
762 <ADFLogger> <begin> Execute query
763 <ViewObjectImpl> <closeStatementsResetRowSet> [593] ViewObject: [samples.model.EmpView]DemolAppModule.EmpView2
764 <ViewObjectImpl> <getPreparedStatement> [594] ViewObject: [samples.model.EmpView]DemolAppModule.EmpView2
765 <ViewObjectImpl> <buildQuery> [595] EmpView2>#q computed SQLStmntBufLen: 246, actual=191, storing=221
766 <ViewObjectImpl> <buildQuery> [596] SELECT Emp.EMPNO, Emp.ENAME, Emp.JOB, Emp.MGR
767 <ViewObjectImpl> <bindParametersForCollection> [597] Bind params for ViewObject: [samples.model.EmpView]DemolAppModule.EmpView2
768 <OracleSQLBuilderImpl> <bindParamValue> [598] Binding param "Bind_Deptno": 10
769 <ADFLogger> <addContextData> Execute query
770 <ADFLogger> <addContextData> Execute query
771 <ADFLogger> <end> Execute query
772 <EntityImpl> <registerExprValueSupplierOverride> [599] Entity with key:oracle.jbo.Key[7782 ] owned by row
773 <ADFLogger> <end> Executing iterator binding
774 <JUCtrlValueBinding> <startEvents> [600] noCtrl_oracle_adfinternal_view_faces_model_binding_FacesCtrlHierl
775 <JUCtrlValueBinding> <startEvents> [601] Empno: IGNORING Start events as it is not in active/push mode
776 <JUCtrlValueBinding> <startEvents> [602] Ename: IGNORING Start events as it is not in active/push mode
777 <JUCtrlValueBinding> <startEvents> [603] Job: IGNORING Start events as it is not in active/push mode
```

Normal text file length: 496246 lines: 5097 Ln: 766 Col: 37 Sel: 0 Dos\Windows ANSI INS

Custom Logging

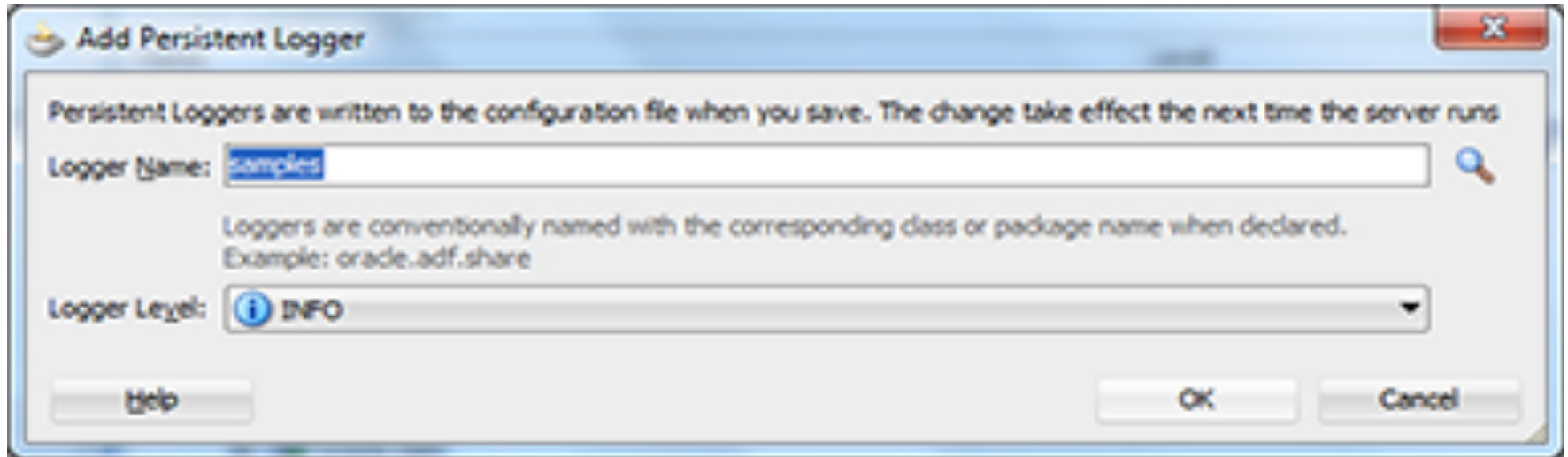
- Here is some code added to view object implementation class when salary changes

```
private static ADFLogger myLogger =
    ADFLogger.createADFLogger(EmpImpl.class);

public void setSal(BigDecimal value) {
    myLogger.info("Salary for employee "
        + this.getEmpno().toString()
        + " changed to "
        + value.toString());
    setAttributeInternal(SAL, value);
}
```

Enable Custom Logging

- Use Oracle Diagnostics Logging Config.



Oracle Diagnostics Logging Configuration

Control logging behavior for specified [loggers](#). If the server is running, changes take effect immediately. Otherwise, saved changes take effect the next time the server runs.

Loggers:

<input type="checkbox"/> Hide Transient Loggers				Java Log Levels ▼
Name	Level	Declares Handlers		
Root Logger (default)	WARNING			
demologger	INFO			
oracle	WARNING			
samples	INFO			

Custom Logging Output

The screenshot shows two Oracle WebLogic Server windows. The top window is the 'Oracle Diagnostic Log Analyzer', which has a search bar with the path 'C:\Users\John\AppData\Roaming\JDeveloper\system11.1.2.4.39.64.36.1\DefaultDomain\servers\DefaultServ...'. It features tabs for 'By ADF Request' and 'By Log Message'. Under the 'By Log Message' tab, there is a 'Search' section with a 'Java Log Level' dropdown set to 'Info' (with checkboxes for Severe, Warning, Info, Config, Fine, Finer, Finest, and Unknown). Below this is a 'Log Time' section with a 'Most Recent' dropdown, a text input '10', and a 'Minutes' dropdown. There is also a 'Message' section with a 'Contains' dropdown and an empty text input. 'Search' and 'Reset' buttons are at the bottom right. The 'Results' section is empty, with a 'Group by Id' checkbox. The bottom window is 'Running: IntegratedWebLogicServer - Log', showing a search for 'salary'. The log output is as follows:

```
<ADFLogger> <end> Evaluate Expression
<EmpImpl> <setSal> Salary for employee 7782 changed to 2450.99
<ADFLogger> <begin> ADF Phase validateModelUpdates
```


Related Output

- Use drop-down to select related output

Results: 1 found. May 29, 2014 12:04 PM ☐ Group by Id

Time	Message	Module	Related	Mess...	Type	Appli...
May 29, 2014 12...	Salary for employee 7782 changed to 2450.99	samples.model.E...			INFO	DemoA...

Related By Request
 Related By Time
 Related By ADF Request

[Results Messages](#) > Related by Time:Before 30 seconds

Time	Message	Module	Message Id	Type	Application
May 29, 2014 12:00:11 PM MST	Salary for emplo...	samples.model.EmpImpl		INFO	DemoApplication 1
May 29, 2014 12:00:11 PM MST	Evaluate Expres...	oracle.jbo.common.ADF...		CONFIG	DemoApplication 1
May 29, 2014 12:00:11 PM MST	Evaluate Expres...	oracle.jbo.common.ADF...		CONFIG	DemoApplication 1
May 29, 2014 12:00:11 PM MST	Evaluate Expres...	oracle.jbo.common.ADF...		CONFIG	DemoApplication 1
May 29, 2014 12:00:11 PM MST	Evaluate Expres...	oracle.jbo.common.ADF...		CONFIG	DemoApplication 1
May 29, 2014 12:00:11 PM MST	Evaluate Expres...	oracle.jbo.common.ADF...		CONFIG	DemoApplication 1



Custom Timing Entries

- When the built-in instrumentation isn't what you need; you can add timing yourself

```

    }
    @Override
    public void executeQuery() {
        long startMillis = System.currentTimeMillis();
        long endMillis = 0L;
        long elapsedMillis = 0L;
        myLogger.info("EMP query: Before EMP query ");
        try {
            super.executeQuery();
        }
        finally {
            endMillis = System.currentTimeMillis();
            elapsedMillis = endMillis - startMillis;
            myLogger.info("EMP query: Elapsed time for EMP query SQL = "
                + elapsedMillis + " milliseconds");
        }
    }

```

Results: 1 found. May 29, 2014 3:11 PM							<input type="checkbox"/> Group by Id
Time	Message	Mod...	Related	Me...	Type	Ap...	
May 29, 201...	EMP query: Elapsed time for EMP query SQL = 32 milliseconds	sampl...			 I...	Demo...	

- Checking if Logging enabled

```
public void executeQuery() {
    Long startMillis = System.currentTimeMillis();
    Long endMillis = 0L;
    Long elapsedMillis = 0L;
    super.executeQuery();
    if (myLogger.isLoggable(Level.INFO)) {
        endMillis = System.currentTimeMillis();
        elapsedMillis = endMillis - startMillis;
        myLogger.log("SQL execution "+ elapsedMillis);
    }
}
```


- When executing in production environments, Enterprise Manager provides the ability to view WebLogic's logging data:
 1. Open the server farm
 2. Select WebLogicDomain
 3. Select appcluster
 4. Choose the desired WebLogic server
 5. Select logs to see options to turn logging on/off or to view logs

Use | Make | Buy

- Instrumentation is easy, we have 3 choices:
 - Use Oracle's built-in logging is vast and provides great detail
 - Make Use ADFLogger to "grow your own" (use superclasses to make it reusable)
 - Buy Frank Houweling from Amis has created an ADF Performance Monitor Tool (I have not tested it thoroughly; but it seems to work well)
- More information is available at:

<http://www.amis.nl/ADFperformancemonitor>

Wrapping it all Up

- Instrumentation is the key to debugging, tracking, and tuning ADF applications
- ADFLogger provides comprehensive logging ability “out of the box”
- JDeveloper provides GUI-based support for ADFLogger configuration and use
- Occasionally, it will be useful to create a “home-grown” ADFLogger; but, the built-in tools will work with that too

RMOUG Training Days 2016

February 9-11, 2016

(Tuesday-Thursday)

Denver Convention Center



Tracks

- Application Development
- Business Intelligence
- Database Administration
- DBA Deep Dive
- Database Tools of the Trade
- Hyperion
- Middleware
- Professional Empowerment

PHOTO CREDIT: Mike Landrum, SQL Developer and the "Data Tsunami" from i-Behavior

www.rmoug.org



COLLABORATE 16 – IOUG Forum

April 10 – 14, 2016

**Mandalay Bay
Las Vegas, NV**





ODTUG
Kscope15

HOLLYWOOD, FLORIDA • JUNE 21-25



Session 779

Gauging ADF Application Performance: Instrumenting Your Code

To contact the author:

John King

King Training Resources

P. O. Box 1780

Scottsdale, AZ 85252 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com



Thanks for your attention!

Today's slides are on the web:

<http://www.kingtraining.com>

- End

