Performance analysis and troubleshooting for WebSphere Portlet Factory applications

WebSphere Portlet Factory Development Team

Topics

- Introduction
- Summary of diagnostic and analysis tools
  - Tools for developers
  - Tools for use during load testing and deployment
- Detailed information on diagnostic and analysis tools
- Some best practices for troubleshooting and analysis

What's covered here

- This presentation focuses on analysis and troubleshooting for peformance and scalability
  - Other debugging and troubleshooting topics are not covered here
- The tools discussed are the ones that are most relevant for Portlet Factory application code
  - Other performance tools for application server, operating system, and database are not covered

About Portlet Factory and application performance

- Portlet Factory is a high-level code generation and application execution framework
- During execution, most of the low-level work is typically done in the low-level framework
  - The low-level framework is tuned by the Portlet Factory team
- Application analysis and tuning focuses on higher-level application elements
  - How Portlet Factory builders are used and combined
  - Application actions and memory use
  - Utilization of external data and services
- Portlet Factory provides several tools for analyzing application-level elements

Summary of diagnostic and analysis tools

Tool	Description	Where used
WPF Server stats logging	Records numerous statistics about portlet activity every five minutes	Deployment and load test
Verbose GC logs	Records garbage collector activity for server	Deployment and load test
WPF Event logs	Shows all exceptions encountered, with stack traces	All environments
WPF Model action tracing	Shows all actions within each request, with latency (response) for each sub-action	Development and test
WPF Session size tracing	Shows relative sizes of session variables	Development and test
WPF other built-in logging	Service call, SQL, Profile selection, Builder calls, regen	Development, limited in deployment
Application logging	Records application-specific information	Development, limited in deployment
Load test tools	Simulates multiple users and measures performance	Load test
Performance monitoring tools (Wily, Tivoli, e.g.)	Monitors running systems and captures performance-related information	Deployment and load test
Heap dump	Shows detailed information about heap use at a point in time (triggered manually or on OutOfMemory)	Deployment and load test
Java profiling tools	Provides low-level information about Java code execution and object allocation hotspots	Development and test

WPF Server stats logging

- Captures periodic snapshot of statistics on portlet activity
- Logged every 5 minutes by default
- Recorded in each Portlet Factory WAR folder, in the file:
  WEB-INF/logs/serverStats.txt
- Enabled by default and should always be left enabled

Use in deployment and load test for:

- See what's happening at the portlet level for an application under load
- Check portlet response – overall and for individual portlets
- Check back end response – DB, services, etc.
- See if exceptions occurring – if so, check event.log for details
- Check for cache tuning issues

Server stats example (description on following slides)

```
*-- TIME: [2009-03-12 07:52:32,674] --*
Category: bowstreet.system.server.logging.serverStats.default
Priority: INFO
Thread:   ServerStatsThread
Msg:      Sessions: 301
RestoredSessions: 0
ModelCacheRegenEntries: 10
Regens: 785
RegensFromCache: 785
OutputCacheHits: 0
OutputCacheMisses: 0
MemTotal: 1757411840
MemFree: 348386120
MemInUse: 1409025720
ErrorsLogged: 1220
SevereErrorsLogged: 0
WarningsLogged: 0
PeakSessions: 301
ParallelModelRequests: 0
WebAppRequests: 908 Latency: 2440
WebAppRequests/myproject/MyPortlet: 908 Latency: 3201
WebAppRequests/myproject/MyPortlet/_bowstreet_show_current_page: 607 Latency: 1510
WebAppRequests/myproject/MyPortlet/MyLJO.doSearch: 301 Latency: 4700
WebAppSOAPRequests: 0
WebAppMethodClassWritten: 0
WebAppJSPSourceWritten: 0
WebAppsInstantiated: 785
SchemaCacheHits: 4466
SchemaCacheMisses: 0
SchemaCacheEntries: 44
ProfileCacheHits: 0
ProfileCacheMisses: 0
ProfileSetCacheHits: 0
ProfileSetCacheMisses: 0
SoapRequests: 565 Latency: 694
SoapRequests/mysoapserver.mycompany.com: 565 Latency: 694
SoapRequests/mysoapserver.mycompany.com/GetEmployeeDetails: 565 Latency: 694
```

Server stats example, part 1

```
Msg:        Sessions: 301
RestoredSessions: 0
ModelCacheRegenEntries: 10
Regens: 785
RegensFromCache: 785
OutputCacheHits: 0
OutputCacheMisses: 0
MemTotal: 1757411840
MemFree: 348386120
MemInUse: 1409025720
ErrorsLogged: 1220
SevereErrorsLogged: 0
WarningsLogged: 0
```

Number of portlet sessions (one for each portlet in user's session

Failover sessions restored

Entries in regen cache

Total regens requested and how many from cache (all, in this case)

Hits and misses for Cache Control caches (Cache Control not used in this case)

Java heap – total, free, and in use (these are not after GC as you would see in VGC log)

Errors and warnings – see event.log file for details of all 1220 errors.

Server stats example, part 2

```
WebAppRequests: 901 Latency: 2640
WebAppRequests/myproject/MyPortlet: 901 Latency: 3231
WebAppRequests/myproject/MyPortlet/_bowstreet_show_current_page: 600 Latency:
  1500
WebAppRequests/myproject/MyPortlet/MyLJO.doSearch: 301 Latency: 4700
```

WebAppRequests = Number of portlet requests to this WAR

Latency = response time in ms for this portlet (not including Portal page or other portlets on page)

WebAppRequests/modelName = requests and latency for a single portlet model

WebAppRequests/modelName/action = requests and latency for a single top-level action in a single portlet model

Observations:

- In this time period, average server response for portlets was 2.6 seconds
- The method MyLJO.doSearch was by far the slowest action at 4.7 seconds

Server stats example, part 3

```
WebAppSOAPRequests: 0
WebAppMethodClassWritten: 0
WebAppJSPSourceWritten: 0
WebAppsInstantiated: 785
SchemaCacheHits: 4466
SchemaCacheMisses: 0
SchemaCacheEntries: 44
ProfileCacheHits: 0
ProfileCacheMisses: 0
ProfileSetCacheHits: 0
ProfileSetCacheMisses: 0
```

Only used for SOAP server (not portlets)

Java source files written/compiled

JSP files written/compiled

Number of models initialized

Schema cache activity – number of entries in cache, number of hits/misses to cache

Profile cache activity

Server stats example, part 4

```
SoapRequests: 565 Latency: 694
SoapRequests/mysoapserver.mycompany.com: 565 Latency: 694
SoapRequests/mysoapserver.mycompany.com/GetDetails: 307 Latency: 754
SoapRequests/mysoapserver.mycompany.com/GetEmployees: 258 Latency: 623
```

SoapRequests = Number of outgoing SOAP web service calls from this WAR
Latency = response time in ms for constructing inputs, calling web service, and processing results
SoapRequests/endpointServer = requests and latency for all requests to one server
SoapRequests/endpointServer/service = requests and latency for a single service

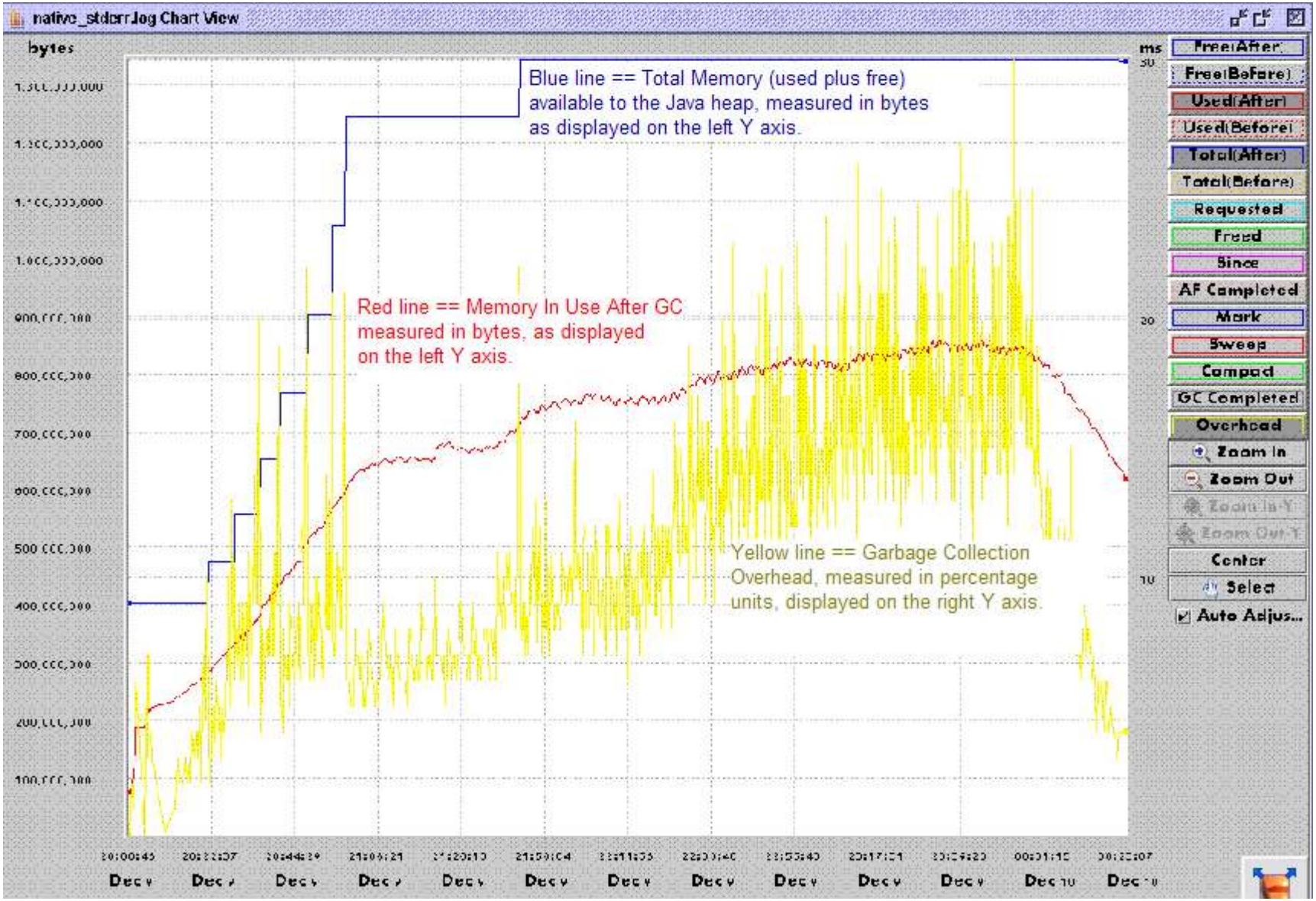
Observations:
- In this time period, average service call response was around .7 seconds, including result processing
- The two different services had similar response times

Verbose GC logs

- Records JVM garbage collector activity for server
- Enabled using WebSphere Application Service console
  - We recommend leaving this enabled all the time since it is not costly
- File: logs/<portalServer>/native_stderr.log
- Logs can be viewed with PMAT tool (Pattern Modeling and Analysis Tool for Java Garbage Collector) from IBM developerWorks

Use in deployment and load test for:

- Get a picture of overall heap activity and health over time
- See total heap and used heap
- See if too much CPU time is spent in GC (GC Overhead)
- Monitor frequency and size of GC activity


native_stderr.log Chart View
bytes
Blue line == Total Memory (used plus free) available to the Java heap, measured in bytes as displayed on the left Y axis.
Red line == Memory In Use After GC measured in bytes, as displayed on the left Y axis.
Yellow line == Garbage Collection Overhead, measured in percentage units, displayed on the right Y axis.
ms
FreeAfter
FreeBefore
UsedAfter
UsedBefore
TotalAfter
TotalBefore
Requested
Freed
Since
AF Completed
Mark
Sweep
Compact
GC Completed
Overhead
Zoom In
Zoom Out
Center
Select
Auto Adjus...

WPF Event logs

- Logs all exceptions and errors from Portlet Factory execution
- Recorded in each Portlet Factory WAR folder, in the file:
  WEB-INF/logs/event.log
- Typically includes full stack trace of exceptions

Use during development, test, and deployment for:

- Determining root cause of errors

WPF Model action tracing

- Shows the detailed sub-actions for every portlet request
- Shows how much time spent in each sub-action
- Enabled with property or with Designer “Run” command
- Highly recommended for use during development – confirm program flow and look for performance issues
- Best used for single-user runs (manual or automated test script)
- Recorded in each Portlet Factory WAR folder, in the file:
  WEB-INF/logs/modelActions.txt

Use for:

- Examining program flow, for debugging and to locate inefficiencies
- Finding slow methods in execution
- Finding unexpected execution code paths (e.g., duplicated calls to back end)
- Note: model action tracing should never be left enabled in deployment under load

Model action trace – portlet action phase

```
*-- TIME: [2009-02-25 10:10:19,415] --*
Category: bowstreet.system.modelActions.myproject_MyPortlet
Priority: INFO
Thread:   WebContainer : 1
Msg:
    Session:  q9ASYT_XasdfB7KzHrQMvug
    Model:    myproject/MyPortlet
    User Name:              cn=myuser,o=mycompany,c=us
       104   580    Start Request: WebAppRunner.doRequest
         0     0    [myproject/MyPortlet]
         0     5    Method: _handleOnRequestEvent
         5     5    .Method: OnReqAction
        47   471    Method: MyLJO.doSearch
        13   424    .Method: MyServiceConsumerEmployeeGetDetails
         8   411    ..Method: MyServiceConsumer.executeOperation
         0     0    ...Method: MyServiceConsumer_createHelper
         0     0    ...[myproject/service/EmployeeDetailsService]
         1     1    ...Instantiate: myproject/service/EmployeeDetailsService
         0     0    ....Method: EmployeeImplOnLoadHandler
         0     0    ....Method: EmployeeDetailsImplOnLoadHandler
        11   402    ...Method: EmployeeGetDetailsExecute
         2   391    ....Method: EmployeeImpl.invoke
       386   389    .....Method: EmployeeImpl.invokeInternal
         0     0    ......Method: _IRResolver_1
         0     1    ......Method: _IRResolver_11
         1     1    .......Method: MyLJO.getDataFromCache
         0     0    ......Method: _IRResolver_2
         0     0    ......Method: _IRResolver_3
         0     0    ......Method: _IRResolver_4
         0     0    ......Method: _IRResolver_5
         0     0    ......Method: _IRResolver_6
         0     0    ......Method: _IRResolver_7
         0     2    ......Method: _IRResolver_8
         2     2    .......Method: MyUtilsLJO.getStuff
```

First column: elapsed time in ms for one sub-action

Second column: elapsed time for a sub-action and all child actions

47ms was spent in this method, not including child actions

Most of the time in this trace is spent calling an external web service (invokeInternal is the web service execution method name)

Model action trace – portlet render phase

```
*-- TIME: [2009-02-25 10:10:19,618] --*
Category: bowstreet.system.modelActions.myproject_MyPortlet
Priority: INFO
Thread:   WebContainer : 1
Msg:
    Session:            q8ASYT_XasdfB7KzHrQMvug
    Model:  myproject/MyPortlet
    User Name:          cn=myuser,o=mycompany,c=us
        40    61    Start Request: WebAppRunner.doRequest
         0     0    [myproject/MyPortlet]
         0     5    Method: _handleOnRequestEvent
         2     5    .Method: OnReqAction
         3     3    ..Method: MyLJO.DoSomething
        10    16    Page: MySearchPage
         1     1    .Method: MyLJO.getInfo
         1     1    .Method: MyLJO.getMoreInfo
         3     3    .Method: MyLJO.getSomeOtherStuff
         1     1    .Page: MyOtherPage
         0     0    .Method: MyLJO.isAvailable
```

WPF Session size tracing

- Shows the relative size of all session variables
- Enabled via property (temporarily!)
- Information about variables is logged periodically at the end of a request
- Recorded in each Portlet Factory WAR folder, in the comma separate value file:

  WEB-INF/logs/sessionsize.csv
- Best used for single-user runs (manual or automated test)
- Recommendation: find a single snapshot in CSV file and read that into spreadsheet for sorting
  - Often the last logged request is used, so all portlets are in session
- Note: Sizes are rough relative sizes, not accurate absolute sizes
  - Java provides no simple way to find object sizes
  - Some string data used in XML structures may be shared objects

Use for:

- Identifying large session variables, to help in reducing session memory use
- Note: session size tracing should never be left enabled in deployment under load

Session size tracing

"*-- 2009-03-02 18:10:26,406 --*"
Session Size Info for user wasadmin
Model,Name,Type,Scope,Size,StringSize
lwm/ess/lifeEvents/ChangeName,LocaleInfo,com.bowstreet.services.base.TaggedData,Session,464,150
lwm/ess/lifeEvents/ChangeName,lwmLogInfo,com.bowstreet.services.base.TaggedData,Session,466,151
lwm/ess/lifeEvents/ChangeName,NameViewUtil,com.ibm.lwm.NameViewUtil,Session,0,33
lwm/ess/lifeEvents/ChangeName,ModelInitializationData,java.lang.String,Session,42,5
lwm/ess/lifeEvents/ChangeName,EmployeeId,java.lang.String,Session,42,5
lwm/ess/lifeEvents/ChangeName,NameDataTranslateInfo,com.bowstreet.services.base.TaggedData,Session,464,150
lwm/ess/lifeEvents/ChangeName,EditDataTranslateInfo,com.bowstreet.services.base.TaggedData,Session,464,150
lwm/ess/lifeEvents/ChangeName,changeNameVar,com.bowstreet.services.base.TaggedData,Session,5080,1452
lwm/ess/lifeEvents/ChangeName,rowShadingUtil,com.ibm.lwm.RowShadingUtil,Session,0,35
lwm/ess/lifeEvents/ChangeName,prefixIndexVar,java.lang.Integer,Session,16,1
lwm/ess/lifeEvents/ChangeName,PIMProducerGetPersonalDataInputs,com.bowstreet.services.base.TaggedData,Session,442,208
lwm/ess/lifeEvents/ChangeName,PIMProducerGetPersonalDataResults,com.bowstreet.services.base.TaggedData,Session,5160,1430
lwm/ess/lifeEvents/ChangeName,PIMProducerNamePrefixHelpResults,com.bowstreet.services.base.TaggedData,Session,1076,221
lwm/ess/lifeEvents/ChangeName,PIMProducerNameSuffixHelpResults,com.bowstreet.services.base.TaggedData,Session,22160,4491
lwm/ess/lifeEvents/ChangeName,PIMProducer,com.bowstreet.builders.webapp.ServiceConsumer2DataHelper,Session,0,65
lwm/ess/lifeEvents/ChangeName,suffixIndexVar,java.lang.Integer,Session,16,1
lwm/ess/lifeEvents/ChangeName,editPageError,com.bowstreet.builderutilities.PageAutomationMessages,Session,0,1327
lwm/ess/lifeEvents/ChangeName,bowstreet.method.class,genjava.lwm.ess.lifeEvents._ChangeName,Session,0,47
lwm/ess/lifeEvents/ChangeName,_moduleInitializationComplete,java.lang.Boolean,Session,16,4
lwm/myresources/Widget_MyLifeResources,LocaleInfo,com.bowstreet.services.base.TaggedData,Session,464,150
lwm/myresources/Widget_MyLifeResources,ActivateWire,com.ibm.lwm.ActivateWire,Session,0,33
lwm/myresources/Widget_MyLifeResources,xmlEventTypes,com.bowstreet.services.base.TaggedData,Session,742,209
lwm/myresources/Widget_MyLifeResources,getMyResourcesFromKBAccessKBESSActionsInputs,com.bowstreet.services.base.TaggedData,Session,290,169
lwm/myresources/Widget_MyLifeResources,getMyResourcesFromKBAccessKBESSActionsResults,com.bowstreet.services.base.TaggedData,Session,21866,545

Session size tracing

Model	Name	Type	Scope	Size	StringSize
lwm/ess/lifeEvents/ChangeName	LocaleInfo	com.bowstreet.services.base.TaggedData	Session	464	150
lwm/ess/lifeEvents/ChangeName	lwmLogInfo	com.bowstreet.services.base.TaggedData	Session	466	151
lwm/ess/lifeEvents/ChangeName	NameViewUtil	com.ibm.lwm.NameViewUtil	Session	0	33
lwm/ess/lifeEvents/ChangeName	ModelInitializationData	java.lang.String	Session	42	5
lwm/ess/lifeEvents/ChangeName	EmployeeId	java.lang.String	Session	42	5
lwm/ess/lifeEvents/ChangeName	NameDataTranslateInfo	com.bowstreet.services.base.TaggedData	Session	464	150
lwm/ess/lifeEvents/ChangeName	EditDataTranslateInfo	com.bowstreet.services.base.TaggedData	Session	464	150
lwm/ess/lifeEvents/ChangeName	changeNameVar	com.bowstreet.services.base.TaggedData	Session	5080	1452
lwm/ess/lifeEvents/ChangeName	rowShadingUtil	com.ibm.lwm.RowShadingUtil	Session	0	35
lwm/ess/lifeEvents/ChangeName	prefixIndexVar	java.lang.Integer	Session	16	1
lwm/ess/lifeEvents/ChangeName	PIMProducerGetPersonalDataInputs	com.bowstreet.services.base.TaggedData	Session	442	208
lwm/ess/lifeEvents/ChangeName	PIMProducerGetPersonalDataResult	com.bowstreet.services.base.TaggedData	Session	5160	1430
lwm/ess/lifeEvents/ChangeName	PIMProducerNamePrefixHelpResults	com.bowstreet.services.base.TaggedData	Session	1076	221
lwm/ess/lifeEvents/ChangeName	PIMProducerNameSuffixHelpResults	com.bowstreet.services.base.TaggedData	Session	22160	4491
lwm/ess/lifeEvents/ChangeName	PIMProducer	com.bowstreet.builders.webapp.ServiceConsume	Session	0	65
lwm/ess/lifeEvents/ChangeName	suffixIndexVar	java.lang.Integer	Session	16	1
lwm/ess/lifeEvents/ChangeName	editPageError	com.bowstreet.builderutilities.PageAutomationM	Session	0	1327
lwm/ess/lifeEvents/ChangeName	bowstreet.method.class	genjava.lwm.ess.lifeEvents._ChangeName	Session	0	47
lwm/ess/lifeEvents/ChangeName	_moduleInitializationComplete	java.lang.Boolean	Session	16	4
lwm/myresources/Widget_MyLifeRe	LocaleInfo	com.bowstreet.services.base.TaggedData	Session	464	150
lwm/myresources/Widget_MyLifeRe	ActivateWire	com.ibm.lwm.ActivateWire	Session	0	33
lwm/myresources/Widget_MyLifeRe	xmlEventTypes	com.bowstreet.services.base.TaggedData	Session	742	209
lwm/myresources/Widget_MyLifeRe	getMyResourcesFromKBAccessKB	com.bowstreet.services.base.TaggedData	Session	290	169
lwm/myresources/Widget_MyLifeRe	getMyResourcesFromKBAccessKB	com.bowstreet.services.base.TaggedData	Session	21866	545

These are the two largest session variables for this application

Other Portlet Factory built-in logging

- Several other types of logging can be enabled via log4j.properties settings
  - Service call – with inputs and response
  - SQL calls – with SQL parameter (input) logging
  - Profile selection
  - Builder calls
  - Model regeneration
- These should only be enabled for trouble-shooting and not left enabled in production

Application logging

- Application models and Java code will often use logging APIs to enable logging of application-specific information
- Use APIs such as log4j categories to support enabling/disabling of logging
- Important: in Java code, always check whether logging is enabled before doing logging work such as constructing log strings

Load test tools

- Used to measure performance and capacity of complete Portal or web application
- Examples:
  - IBM Rational Performance Tester
  - Apache JMeter
  - HP LoadRunner
- Portlet Factory applications are load-tested like any other Portal application
  - There are no particular Portlet Factory issues with these tools
- Refer to other resources for information on these

Heap dump

- Snapshot of all Java heap object use
- Triggered manually or when heap exhausted (OutOfMemory)
- Analyze with IBM HeapAnalyzer or Eclipse Memory Analyzer
  - Lots of memory is needed to run these tools – we often need to run on 64-bit JVM to analyze multi-gigabyte heap dumps.
- Typically, most objects in heap are WPF objects or String/char[] owned by WPF objects – this is normal
  - Analysis of these objects may require WPF expertise

Use in deployment and load test for:

- Analysis of WPF, Portal, and WAS objects by IBM team
- Analysis of application objects by application team
- Identifying causes of memory leak or excessive memory use

Java profiling tools

- These tools can provide low-level information about Java execution hot spots
- Both CPU and memory use can be profiled
- Examples:
  - JPROF
  - JProbe
  - JProfiler
  - YourKit Java Profiler
- When using a high-level framework such as Portlet Factory, most hot spots found in profiling are in framework code and objects
  - This is normal and expected
  - The primary use of these profiling tools is for WPF and Portal teams in tuning the framework itself
  - Interpreting results typically requires detailed knowledge of the framework

Some best practices for troubleshooting and analysis

- Keep an eye on exceptions and event.log
  - Exceptions and exception logging can skew application performance results
  - Always try to achieve exception-free execution
- Use verbose GC log to get a nice indicator of heap “health”
- If session size is a problem, use session size tracing to find the largest variables, and target those for improvement
- If particular portlets and portlet actions are slow, use model action tracing to see what’s going on
- Some logs should always be left enabled since they are low overhead and provide valuable information about system health:
  - Verbose GC, server stats, event log
- Other logs should be left disabled in production and load test except when needed:
  - Model action traces, session size traces, service call logging, application logging

Additional resources

- Performance Best Practices page on the Portlet Factory wiki
  - http://www-10.lotus.com/ldd/pfwiki.nsf/dx/performance-best-practices
- Using Portlet Factory analysis tools
  - http://www-01.ibm.com/support/docview.wss?rs=3044&context=SSRUWN&dcÛ520&uid=swg21268497&loc=en_US&cs=UTF-8&lang=en&rss=ct3044websphere
- PMAT tool – for analyzing verbose GC logs
  - http://www.alphaworks.ibm.com/tech/pmat