PASOE – Performance Tuning

Martyn Kemp
Senior Consultant
Consultingwerk Software Services Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and **related technology**
- Located in Cologne, Germany, subsidiaries in UK and Romania
- Customers in Europe, North America, Australia and South Africa
- Vendor of developer tools and consulting services
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration
- Experts in OpenEdge Application Modernization
Martyn Kemp

- Senior Consultant
- 20 plus years of development experience in various roles and using various technologies
- Initially started using Progress just before the release of version 7.
- First started using the .Net Framework with the introduction of ProDatasets (circa 2002). During those times and upto the current time, Martyn has developed using OpenEdge GUI for .Net, C#, VB .Net and more recently, OE Mobile, OE REST, Spring IO, PASOE, Angularjs 1.x & Angular 2 and Swagger. Martyn currently works as Senior Consultant for Consultingwerk.
Agenda

- **Introduction**
- Understanding PASOE
- Use the latest versions
- Configure and fine tuning for your application
- Testing
- Monitoring
- Memory
- Summary
Introduction

- **What is PASOE?**
  - Progress Application Server for OpenEdge (PAS for OpenEdge) is a web server preconfigured to expose your Advanced Business Language (ABL) business and web-UI applications to HTTP clients in an internet/intranet environment.

  - PAS for OpenEdge is constructed on top of a web server, it will not be configured, monitored or tuned like the classic Progress® OpenEdge® Application Server that you may be used to, nor will PAS for OpenEdge behave like it.
Agenda

- Introduction
- **Understanding PASOE**
- Use the latest versions
- Configure and fine tuning for your application
- Testing
- Monitoring
- Memory
- Summary
PASOE – Components
PASOE – Components

- Just one Multi-Session Agent able to utilize all kind of clients and operation modes
PASOE – Architecture

- Progress proprietary network protocol has been dropped
- Tomcat does not speak TCP
PASOE – Architecture

Before you begin tuning your PASOE, it really helps to understand the PASOE architecture, how your ABL application relates to it and where certain critical subsystems operate.

ABL applications running in PASOE are comprised of the following subsystems:

1. The OS process and its configured limits
2. The Java Virtual Machine (JVM)
3. The PAS based on Apache Tomcat
4. The Multi-Session ABL language agent (MS-Agent)
5. The ABL application source code and architecture (ABL application)
PASOE – OS Process Limits

- PASOE and spawned subcomponents are affected by the imposed resource maximums for memory, files, etc.
- CPU and Memory
  - The number and speed of the OS CPUs greatly influence the concurrent execution speed of client requests, and the memory influences the number of concurrent client requests.
PASOE – Java Virtual Machine

- The JVM supplies all of the memory management, threads and I/O resources. The JVM is the point at which you make critical tuning decisions that affect every aspect of server performance, such as determining which web applications you deploy, how many clients it will support and how fast it processes those client requests.
PASOE – Apache Tomcat

- PAS is an adaptation of the Apache Tomcat product that runs in a Java JVM and therefore is subject to the JVM tuning of process threads, memory allocation and file/network I/O.

- PASOE Startup
  - consumes both heap and permgen space at startup time.
  - PAS does not become fully available for client access until it completes the loading and starting of all deployed web application. At the time when PAS is fully available, the web applications are still not fully loaded. Incremental loading of web application classes and memory allocations continues as client HTTP requests begin arriving for the web application to execute. It is advisable that any CPU and memory measurements begin after a warmup period of client traffic across all deployed web applications.
PASOE – Architecture: Multi-Session Agent

The ABL language cannot physically execute in the PAS JVM process, the physical ABL sessions that execute client requests are hosted in external MS-Agents.

The MS Agent is constructed to perform:

- Multiple client requests at the same time
- Supports both Session-Managed and Session-Free request simultaneously
- Manages database shared memory connections
- Uses far fewer system resources
- Handles multiple more transactions than the single session AppServer agents
Agenda

- Introduction
- Understanding PASOE
- **Use the latest versions**
- Configure and fine tuning for your application
- Testing
- Monitoring
- Memory
- Summary
PASOE – Latest versions

- >= 11.7.3
- it's state of the art and well maintained so stay up to date ...
  - Allows you to keep up to date with New features and bug fixes
Agenda

- Introduction
- Understanding PASOE
- Use the latest versions
- **Configure and fine tuning for your application**
- Testing
- Monitoring
- Memory
- Summary
PASOE – Default Settings

Tomcat

maxthreads=300
maxqueuesize=100

PASOE

maxAgents=10
maxConnectionsPerAgent=16
Total 160

Database

-n 100
PASOE – Default Settings

maxthreads=200
maxqueueSize=100
maxAgents=2
maxConnectionsPerAgent=100
Total 202

-n 202 +
PASOE – Default Settings

Firstly, let’s clarify this point “The defaults are always wrong!”

- One major point to remember, never go into production without “Testing” your PAS OE Instance under the expected loads and modifying the configuration accordingly.

- Tomcat settings
  - tcman config - will list all of the configurable values for Tomcat
  - tcman config psc.as.executor.maxthreads=300 – Will set Tomcat incoming threads to 300 (the default by the way)
  - psc.as.https.maxqueueoffset=100 – thread overflow if executor maxthreads are full
Configuration and Fine Tuning - JVM

The most important aspect of tuning your PASOE instance is the JVM stack / heap/ permgen memory allocations and the garbage-collection.

- The default JVM configuration parameters
  - `{PASOE Instance}/conf/jvm.properties`.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Xms&lt;size&gt;</td>
<td>512m</td>
<td>Initial Java heap size</td>
</tr>
<tr>
<td>-Xmx&lt;size&gt;</td>
<td>1024m</td>
<td>Maximum Java heap size</td>
</tr>
<tr>
<td>-XX:NewSize</td>
<td>64m</td>
<td>Initial space used for short duration objects and indirectly how often garbage collection runs</td>
</tr>
<tr>
<td>-XX:MaxNewSize</td>
<td>64m</td>
<td>Maximum space used for short duration objects</td>
</tr>
<tr>
<td>-XX+DisableExplicitGC</td>
<td>-</td>
<td>Disable explicit garbage collection</td>
</tr>
</tbody>
</table>
Configuration and Fine Tuning - JVM

Tuning Tips:

- Reduce the frequency of garbage collection by starting the JVM with a larger maximum heap memory space (-Xmx)
- Reduce repeated reallocation of heap memory by setting the initial heap memory space equal to the maximum amount (-Xms == -Xmx)
Configuration and Fine Tuning - PAS

Tuning the PAS involves controlling the HTTP client TCP/IP connections. The properties for many of the network connection properties are defined as Java system properties in:

- `{PASOE Instance}/conf/catalina.properties` file

The contents here are too numerous and to date, I’ve not had a reason to change any of the settings. Fine tuning the properties I feel, requires a Web Server expert and certain a lot of trial and error.
# Configuration and Fine Tuning – MS-Agents

## The OpenEgde parameter settings

- `{PASOE Instance}/conf/openedge.properties`

<table>
<thead>
<tr>
<th>Property</th>
<th>Default</th>
<th>Description (per ABL Application)</th>
</tr>
</thead>
<tbody>
<tr>
<td>numInitialAgents</td>
<td>1</td>
<td>Number of MS-Agent processes to create at server startup</td>
</tr>
<tr>
<td>maxAgents</td>
<td>10</td>
<td>Maximum number of MS-Agent processes that can exist</td>
</tr>
<tr>
<td>minAgents</td>
<td>0</td>
<td>0 is off, &gt;1 maintain this number</td>
</tr>
<tr>
<td>agentStartLimit</td>
<td>1</td>
<td>only start one agent at a time</td>
</tr>
<tr>
<td>maxConnectionsPerAgent</td>
<td>16</td>
<td>Maximum number of network connections between Session Manager and an MS-Agent</td>
</tr>
<tr>
<td>maxABLSessionsPerAgent</td>
<td>200</td>
<td>Maximum number of ABL sessions per MS-Agent process</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Default</th>
<th>Description (per ABL Application)</th>
</tr>
</thead>
<tbody>
<tr>
<td>numInitialSessions</td>
<td>5</td>
<td>Number of ABL sessions started at MS-Agent startup time</td>
</tr>
</tbody>
</table>
Configuration and Fine Tuning – MS-Agents

Tuning Tips:

- The general rule of thumb is:
  - Set the MaxAgents = No Of CPU’s
- During Load Testing you may want to try decreasing / increasing the following:
  - requestWaitTimeout
  - idleSessionTimeout
  - idleAgentTimeout
  - maxABLSessionsPerAgent
PASOE – Openedge Settings

- **-q Quick Request**
  - Use Quick Request (-q) to tell the AVM to search PROPATH directories only on the first use of a procedure.

Ordinarily in an ABL procedure, when the RUN statement is used to run a subprocedure, the AVM searches the directories named by the PROPATH environment variable, looking for a procedure of the same name with a .r extension. If it finds a file with a .r extension (an r-code file), it checks to make sure the r-code file has not changed since that r-code file was created.

This search is very useful in a development environment where procedures change regularly and you want to make sure you are always running the most current version of your application. However, in a production environment, you might want to bypass this search.

With Quick Request (-q), after the initial search, if the procedure still resides in memory or in the local session-compiled file, the AVM uses that version of the procedure rather than searching the directories again. However, the AVM always checks whether Data Dictionary definitions related to a procedure were modified. If they were modified, the AVM displays an error when it tries to retrieve the procedure.
PASOE – Openedge Settings

- `reusableObjects` Re-usable Objects Cache

  - Use Re-usable Objects Cache (`reusableObjects`) to specify the number of deleted class objects that the AVM stores for later re-initialization.

<table>
<thead>
<tr>
<th>Operating System and syntax</th>
<th>UNIX / Windows</th>
<th>-reusableObjects <code>cache-size</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max Value</td>
<td>Min Value</td>
</tr>
<tr>
<td>Client Session</td>
<td>-</td>
<td>Working directory</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Working directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multi-user default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Single-user default</td>
</tr>
</tbody>
</table>

`Cache-size` Number of deleted class objects to store for later re-initialization.

This parameter can improve the performance of applications that use ABL classes. When a class object is deleted, either by the `DELETE OBJECT` statement or through garbage collection, the AVM caches the object. When the AVM instantiates the same class later, the stored object is re-initialized and removed from the cache. This re-use saves much of the overhead of instantiating a class.
PASOE – Openedge Settings

- **-T Temporary Directory**
  - Use Temporary Directory (-T) to specify a directory for temporary files.

<table>
<thead>
<tr>
<th>Operating System and syntax</th>
<th>UNIX / Windows</th>
<th>-T dirname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max Value</td>
<td>Min Value</td>
</tr>
<tr>
<td>Client Session</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

If you do not use this parameter, OpenEdge creates temporary files in the current working directory.

The database broker passes the setting of -T to any servers it spawns. Manual servers and secondary brokers can specify their own -T or take the default; they do not inherit the setting from the primary broker. Remote Client connections can also specify a -T setting different from the server it is connecting to. Temporary space needed in the remote client is stored in its own -T directory. Temporary space needed in the server to process a request for a remote client is stored in the server’s -T directory.
PASOE – Openedge Settings

- **-D Directory Size**
  - Use Directory Size (-D) to change the number of compiled procedure directory entries.

<table>
<thead>
<tr>
<th>Operating System and syntax</th>
<th>UNIX / Windows</th>
<th>-D n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max Value</td>
<td>Min Value</td>
</tr>
<tr>
<td>Client Session</td>
<td>2,147,483,647</td>
<td>5</td>
</tr>
</tbody>
</table>


\[ n \]

The number of compiled procedure directory entries.

Each compiled procedure executed during an OpenEdge session requires a directory entry. A compiled procedure can be a session-compiled version of a procedure or a precompiled r-code version.

When the AVM creates a session-compiled version of a procedure and there is no available space in the directory, it discards the oldest inactive compilation of a procedure. The next time the discarded procedure is run, the AVM must recompile it if it was a session compile, or reopen and reread it if it was a precompiled r-code version.

The -D limit is a soft limit. If your application needs to exceed the limit, OpenEdge automatically increases the number of directory entries by 50 percent and dynamically allocates memory for the expanded array.
PASOE – Openedge Settings

- **-Bt Number of Buffers for Temporary Tables**
  - Use Number of Buffers for Temporary Tables (-Bt) to specify the number of buffers in the temporary table database pool.

<table>
<thead>
<tr>
<th>Operating System and syntax</th>
<th>UNIX / Windows</th>
<th>-Bt n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max Value</td>
<td>Min Value</td>
</tr>
<tr>
<td>Client Session</td>
<td>50,000</td>
<td>10</td>
</tr>
</tbody>
</table>

Use **Temporary Table Database Block Size (-tmpbsize)** to specify the temporary table database block size, which dictates the size of each buffer in the temporary table database buffer pool for the current OpenEdge session.

If you do not have enough free memory (virtual or physical) for the buffer pool, reduce the value of -Bt. To approximate the amount of memory required (in kilobytes) for the buffer pool, use the following formula:

\[
1.1 \times (\text{the value of } -\text{Bt}) \times (\text{the value of } -\text{tmpbsize})
\]
PASOE – Openedge Settings

- TB Speed Sort

- Use Speed Sort (-TB) primarily to improve sort performance, particularly during index rebuild operations.

<table>
<thead>
<tr>
<th>Operating System and syntax</th>
<th>UNIX / Windows</th>
<th>-TB blocksize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max Value</td>
<td>Min Value</td>
</tr>
<tr>
<td>Client Session</td>
<td>31</td>
<td>1</td>
</tr>
</tbody>
</table>

If you increase the block size beyond 8KB, OpenEdge sorts records faster but uses more memory and disk space in the process. If system memory is severely limited, you might actually choose to lower the block size.

Sort space is allocated in the SRT file, a temporary session file having a system-generated unique name with the prefix srt. OpenEdge uses the SRT file to store session-compile modules, to swap r-code segments into and out of the in-memory execution buffer, and as temporary work space during sorting (including rebuilding indexes). Use -TB to set the SRT file block size (that is, the increments by which the SRT file grows).
PASOE – Openedge Settings

- **-TM Merge Number**
  - Use Merge Number (-TM) to increase the speed of the merge phase of the sort process (at the cost of increased temporary memory usage).

<table>
<thead>
<tr>
<th>Operating System and syntax</th>
<th>UNIX / Windows</th>
<th>-TM n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max Value</td>
<td>Min Value</td>
</tr>
<tr>
<td>Client Session</td>
<td>32</td>
<td>1</td>
</tr>
</tbody>
</table>

- $n$ The number of blocks or streams to be simultaneously merged during the sort process.
PASOE – Openedge Settings

- Setting that have worked well for heavy loads
  - -T c:\temp
  - -rereadnolock
  - -errorstack
  - -reusableObjects 1000000
  - -q
  - -TB 31
  - -TM 32
  - -l 2000
  - -nb 500
  - -mmax 65534
  - -D 100000
  - -Bt 50000
  - -tmpbsize 8
Agenda

▪ Introduction
▪ Understanding PASOE
▪ Use the latest versions
▪ Configure and fine tuning for your application
▪ Testing
▪ Monitoring
▪ Memory
▪ Summary
PASOE – Test

- In an ideal world, you would run your tests on a Production Like Environment.
  - PASOE on one machine and your databases on a separate machine….
  - DB local or remote accessed via Proxy.

- Start with a low client payload

- What to check during Performance Testing
  - round-trip time
  - System Resources
  - PASOE process memory (java)
  - MSAgent process memory (_mproapsv)
  - MSAgent Session Memory (oemanager REST API)
PASOE – Test

- Hardware Upgrade!! When?
  - Memory is cheap – so don’t cut corners
  - Round-trip time’s start to deteriorate
  - CPU load matches CPU number
  - Memory is fully utilised and starts to swap
Agenda

- Introduction
- Understanding PASOE
- Use the latest versions
- Configure and fine tuning for your application
- Testing
- Monitoring
- Memory
- Summary
PASOE - Monitoring

- JConsole
- OE Manager REST API
- Tomcat Manager
- JMX Query
- Log File Monitoring
PASOE - Monitoring

Jconsole

- GUI based tool for accessing the Tomcat and PASOE JVM directly
- Direct access to all JMX beans and objects in the Tomcat/PASOE Session Manager
PASOE - Monitoring

OE Manager REST API

- In order to use the OE Manager, it must be installed for your PASOE Instance.
  - This can achieved utilising the tcman command
    - tcman deploy {progress directory}/servers/pasoe/extras/oemanager.war
    - The result should create an oemanager within your webapps “../webapps/oemanager”
  - With the OE Manager deployed, you acces information about your PASOE Instance via HTTP / REST calls.
PASOE - Monitoring

Demo - OE Manager REST API


Similar to “asbman –query”
PASOE - Monitoring

Demo - OE Manager REST API

- http://localhost:8820/oemanager/applications/smartpas_stream/sessions

All current client sessions in the Session Manager
PASOE - Monitoring

Demo - OE Manager REST API


```json
{  "result": {    "agents": [      {        "agentId": "D4IP_TvTva_iNj5ZAJcdA",        "pid": "28508",        "state": "AVAILABLE"      }    ],    "errmsg": "",    "versionStr": "v11.7.2 (2017-12-29)",    "versionNo": 1,    "outcome": "SUCCESS",    "operation": "GET AGENTS"  }
}
```
PASOE - Monitoring

Demo - OE Manager REST API


Shows all agents ABL sessions, start and end times, Session memory
PASOE - Monitoring

Demo - OE Manager REST API


Shows all current requests in the agent
PASOE - Monitoring

Tomcat Manager

- Web based management and monitoring
  - Tomcat process
  - ABL and Web Applications
- Data on current requests to the web server
- Data on memory size of the web server (including the Session Manager)
PASOE - Monitoring

JMX Query

- Script based access to the Tomcat JMX beans
  - Allows gathering information locally without opening ports
  - Like Jconsole, without GUI interface
- Available since 11.7.2
- ../bin/oejmx.sh(.bat)
  - Fully documented in the script / batch file
Agenda

- Introduction
- Understanding PASOE
- Use the latest versions
- Configure and fine tuning for your application
- Testing
- Monitoring
- Memory
- Summary
PASOE - Memory

Memory: MSAgent

- Memory usage will appear to be multiplied
- We use less memory in total, but it's all in 1 executable
- If your classic AppServer agent routinely used 2 Gig of memory
- Moving to PASOE and setting the maxConnectionsPerAgent=100
- Results in an msagent somewhere under 200 (historically 140) Gig of memory
PASOE - Memory

Memory: MSAgent

- If you use the setting “maxConnectionsPerAgent=100”
  - Having 100 ABL sessions in a single agent (versus 1 session in Classic)
  - Memory issues and leaks are multiplied by 100 times
PASOE – Memory MSAgent

Tools for finding memory leaks.

- memory-checker
  - Ability to dump memory allocation of object to a file per ABL session
  - Note: run with one agent and one session

```
DEF NEW GLOBAL SHARED VARIABLE iactcnt AS INT NO-UNDO.

ASSIGN iactcnt = iactcnt + 1.
IF iactcnt = 50 THEN
  _memory:START-MEM-CHECK(LEAK-DETECTION,TRUE,?).
ELSE IF iactcnt = 200 THEN
  _memory:STOP-MEM-CHECK().
```
PASOE – Memory MSAgent

Tools for finding memory leaks.

- Dynamic Object leakchecker.p
  - Details can be found at: [https://knowledgebase.progress.com/articles/Article/P133306](https://knowledgebase.progress.com/articles/Article/P133306)

- Dump a ABL session stack (like prostack) via the oemanager REST API
PASOE – Memory MSAgent

Memory leakchecking

- Dynamic Object leakchecker.p
  - Turn on Dynamic Object Logging
    - In the configuration file, openedge.properties
      - agentLogEntryTypes=ASPlumbing, DB. Connects, DynObjects.*
        DynObjects.Class
        DynObjects.DB
        DynObjects.Other
        DynObjects.XML
        DynObjects.UI
PASOE – Memory MSAgent

Memory leakchecking

- Dynamic Object leakchecker.p
  - In the sessionActivateProc (place the below code within the session activate procedure)

```plaintext
DEF NEW GLOBAL SHARED VARIABLE iLoop AS INT NO-UNDO.
DEF NEW GLOBAL SHARED VARIABLE clogentries AS CHARACTER NO-UNDO.

ASSIGN iLoop = iLoop + 1.
IF iLoop = 50 THEN DO:
  ASSIGN clogentries = log-manager:log-entry-types
  log-manager:log-entry-types = log-manager:log-entry-types + ",DYNOBJECTS.*:4".
END.
ELSE IF iLoop = 200 THEN DO:
  assign log-manager:log-entry-types = clogentries.
END.
```
PASOE – Memory MSAgent

Memory leakchecking
- Once the information is logged to the {PASOE_Instance}.agent.log
- Run the routine leakchecker.p
- The routine shall iterate through the log file and show any leaking objects (objects that have been created but not deleted).
PASOE – Memory MSAgent

ABL Session Stack

- This will dump ABL session information for an ABL session
- If the session is using large amounts of memory
- This will dump the current ABL stack for review
PASOE – Memory MSAgent

ABL Session Stack

- http://localhost:8820/oemanager/applications/smartpas_stream/agents/28508/sessions/7/stacks
PASOE – Reclaiming System Memory

- First try removing high memory Agent sessions
  - curl -v -X DELETE -u tomcat:tomcat
    http(s)://host:port/oemanager/applications/agents/agentId/sessions/sessionId
- If that doesn’t work you can “stop” an Agent
  - This will return memory to the system
  - This REST API allows 10 seconds for all currently running requests to finish before stopping
    - https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/pasoeadmin%2Fstop-an-agent.html%23
- Set minAgents=(a number 2 or greater)
- This will restore agents to the pool when an agent is stopped (or crashes)
# PASOE – Resource Timeout

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value (milliseconds)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentListenerTimeout</td>
<td>300000</td>
<td></td>
</tr>
<tr>
<td>agentWatchdogTimeout</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>connectionWaitTimeout</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>idleAgentTimeout</td>
<td>300000</td>
<td>If running with a “backup” agent may want to disable this value (0)</td>
</tr>
<tr>
<td>idleConnectionTimeout</td>
<td>300000</td>
<td></td>
</tr>
<tr>
<td>idleResourceTimeout</td>
<td>0</td>
<td>Must be set to &gt;0 to enable, NO timeouts enabled when set to 0</td>
</tr>
<tr>
<td>idleSessionTimeout</td>
<td>300000</td>
<td></td>
</tr>
<tr>
<td>requestWaitTimeout</td>
<td>15000</td>
<td></td>
</tr>
<tr>
<td>socketTimeout</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>sessionTimeout</td>
<td>180</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

- Introduction
- Understanding PASOE
- Use the latest versions
- Configure and fine tuning for your application
- Testing
- Monitoring
- Memory
- Summary
PASOE - Summary

To summarize the Performance and Tuning of a PASOE, the main issue that most software companies shall encounter shall most of the time be Memory Leakages.

Although we all like to think that we write code that is optimal and performs well, it is imperative to ensure that we also check our code.

- Check for Orphaned Objects
  - This is by far the most common issue
  - Objects (Temp Tables, Datasets, Classes) which are created / instantiated should always be Garbage Collected when no longer needed.
PASOE - Demo

Demo of memory leakage
Questions