The Progress Electronic Magazine

Table of Contents

Publisher's Statement:

As some of you may have known from my postings on the peg, I have been exploring using version 9 progress for object oriented programming.

In this issue I will explore a bit more with what I have discovered and provide – of course – usable coding you can build off of.

I will be pretty short on the description on the meaning and benefits of OO programming, but one can find out about that from other sources. This article is focused on the 4GL implementation of OOP.

Hopefully you will see the benefits to using this programming paradigm in your progress applications. You will find it easier to manage pieces of functionality and data with it.

The E-Zine should be distributed with the source code, but if you get it from another location, be sure to hit http://amduus.com/OpenSrc/FreePublications/EZineArchive to pick up the zip file with this document and the source code mentioned.

I also want to announce that Service Express is good to for use by companies that might be interested in a help desk software package. Be sure to read http://amduus.com/serviceexpress for more information about this offering.

This software tool can be used easily by people in the IT department, building maintenance department (think approaching your land lord!), and service calls for pieces of equipment.

I also plan on adding companies where users are associated to a customer company and contracts/warranties for equipment.

Lets get on with the fun!

Scott Auge

Object Oriented Programming with the Progress 4GL

By Scott Auge

About Object Oriented Programming

(You will need Progress Version 9 or better to perform this type of programming.)

So what is this object oriented programming? First in Progress we had this procedural programming. Then came event oriented programming. Now Scott is bringing up this use of the language for object oriented programming!

Hey – it's a good thing. The language is adapting. It might be as awkward as say Object Oriented Fortran programming (http://www.cs.rpi.edu/~szymansk/OOF90/F90_Objects.html is an entertaining read) but it can be made to look good.

Let's look at structured programming. We try to reduce tasks to their component tasks. Then we look for tasks with redundant steps and make them into routines. We have two distinct things: data and procedures. We usually look at things in either of

those contexts. We either use procedures to massage data, or we send data to procedures to see something happen.

Object Oriented Programming (OOP) melds the data and the procedures together. Now, we can say, “Oh we have a set of procedures that works on a set of data!” and that is close to OOP, but it just ain't it yet.

OOP allows an even higher level of abstraction for your programming needs. It lets you focus on various “things” in your business or project. For example, one could have a thing called a “Bill of Material.” (BOM) One can do certain things with a BOM – associate it to an order, “explode” it out against inventory, add items to it, subtract items from it. One might even want to put two BOMs together to make yet a third BOM.

You can better focus on the interaction between these things. One could create a sales order object, and then associates a customer object to it. Notice how I am not mentioning records? A sales order could be represented in the database as a set of records – but with OO as far as the programmer is concerned, a sales order is a routine one starts up and calls procedures in. Since the data “sticks” to a persistent procedure, we can start up multiple persistent procedures each of which are manipulating different sets of data.

This can be a godsend for very large programs. I won't try to explain OOP in detail here – the whole idea is to show how to use OOP ideas with the 4GL. Lets just say it is a good thing to manage source code for data and activities in a convenient manner.

I do plan on writing another article giving a discussion of a workable template for an object definition in a persistent procedure.

An Object Manager

To aid us with all the objects that we will be creating, it would be good to have an object manager. The reason for this is:

1. We can dynamically create as many objects as we need. Because the manager works the handles, we don't need to worry about making handle variables.
2. The object manager lets us name the objects, so this is more like creating instances of the objects in an OO oriented language. Passing in the name of the object to other objects managed by this code will be in effect making the object available to those receiving objects.
3. The object manager will automatically call into our object's constructor and destructor routines. Since the 4GL is not a pure OO language, we need to do somethings a little more manually. More about that later.

Since this is the top most procedure file, we include the object manager code with an argument of NEW. This is because the object manager uses a temp table that is

shared amongst everything that refers to that objmgr file. Any called procedures that what to use this tool will need to include it without the NEW argument.

We can see here we create two instances of an object. It's procedures and internal data are described by the 4gl file obja.p with the routines OMAdd from the objmgr code. We name them ObjectA and ObjectB. Note that OMAdd actually calls into the Init routine found in the object definition file.

```
DEFINE VARIABLE cText  AS CHARACTER NO-UNDO.
DEFINE VARIABLE cText1 AS CHARACTER NO-UNDO.

{objmgr.i NEW}

/* Create two instances of the obja object */

RUN OMAdd("ObjectA", "obja.p").
RUN OMAdd("ObjectB", "obja.p").

/* Lets change the attribute in the first and second instances */

RUN SetA IN OMGH("ObjectA") ("A1.1").
RUN SetA IN OMGH("ObjectB") ("A2").

/* Lets see what comes back from both of them */

RUN GetA IN OMGH("ObjectA") (OUTPUT cText).
disp ctext.
RUN GetA IN OMGH("ObjectB") (OUTPUT cText1).
disp ctext1.

disp OM_cErrCode OM_cErrMsg FORMAT "x(20)".

/* Clean up */

RUN OMDel ("ObjectA").
RUN OMDel ("ObjectB").
```

Next to show the objects actually support using different data with the same routine, we use the SetA routine for each object to set a variable in that object. To determine the appropriate handle for the objects, we use the OMGH (Object Manager Get Handle) function from the objmgr tool. Given a name of the object, the OMGH will return the handle to that object.

We then use the GetA routines to obtain the values from the different objects and display them.

When we are done with the objects, we use OMDel to clean up the memory allocation by the objects and to call Destroy routines in the objects automatically.

A Log File Object

Here is an example object that is simple and usable by everyone. It is an object that allows one to manage log files.

One might say "A log file is easy – all I have to do is output a stream." Well that is true. But one might want to have multiple log files – perhaps more than a single procedures five streams can handle. I know I like my log files to have certain information in them too.

Then we might want to have some tools available to delete or copy those log files. All that manipulation might as well be in the log file object code so that it is working with that log file immediately. (Less data management for us like, what is it's name? It's stream name? Et cetera.)

We might want to define one log file as having entries of a given log level, while the other one has a different log level.

As you may be guessing, the coding around this becomes a but more complex if your writing code to manage each and every stream to a log file you might require. If you want to create yet a third log file, that is a whole new set of variables to define, manipulate, and keep track of.

Under objects it is simply the creation of another log file instance and then setting it. From there, it is all one call when using it.

Lets look at the creation of two different log objects and their use in the code below:

```
{objmgr.i NEW}

/*************************************************************/
/* Here we create instances and name them.                   */
/*************************************************************/

RUN OMAdd("GeneralLogFile", "obj_log.p").
RUN OMAdd("MailDetailsLogFile", "obj_log.p").

/*************************************************************/
/* Here we set the log level for each log object.            */
/*************************************************************/

RUN SetLogLevel IN OMGH("GeneralLogFile") (INPUT 3).
RUN SetLogLevel IN OMGH("MailDetailsLogFile") (INPUT 4).

/*************************************************************/
/* Here we set the log file name the object should work with.*/
/*************************************************************/
RUN initFile IN OMGH("GeneralLogFile") (INPUT "/tmp/general_log").
RUN initFile IN OMGH("MailDetailsLogFile") (INPUT "/tmp/objsmtpmail_log").

/*************************************************************/
/* Here we write some entries, denoting which log level they apply  */
/* to.                                                       */
/*************************************************************/

RUN WrtEntry IN OMGH("GeneralLogFile") (INPUT 1, "Level 1").
RUN WrtEntry IN OMGH("GeneralLogFile") (INPUT 2, "Level 2").
RUN WrtEntry IN OMGH("GeneralLogFile") (INPUT 3, "Level 3").
RUN WrtEntry IN OMGH("GeneralLogFile") (INPUT 4, "Level 4").
RUN WrtEntry IN OMGH("GeneralLogFile") (INPUT 5, "Level 5").

RUN WrtEntry IN OMGH("MailDetailsLogFile") (INPUT 1, "Level 1").
RUN WrtEntry IN OMGH("MailDetailsLogFile") (INPUT 2, "Level 2").
RUN WrtEntry IN OMGH("MailDetailsLogFile") (INPUT 3, "Level 3").
RUN WrtEntry IN OMGH("MailDetailsLogFile") (INPUT 4, "Level 4").
RUN WrtEntry IN OMGH("MailDetailsLogFile") (INPUT 5, "Level 5").

/*************************************************************/
/* We could call DoCloseLogFile in each object, but our destructor  */
/* procedure called by OMDel will automatically call DoCloseLogFile */
/* for us, closing each stream appropriately.                */
/*************************************************************/

RUN OMDel("GeneralLogFile").
RUN OMDel("MailDetailsLogFile").
```

And this is the results of the program:

```
[/tmp]$ cat general_log
```

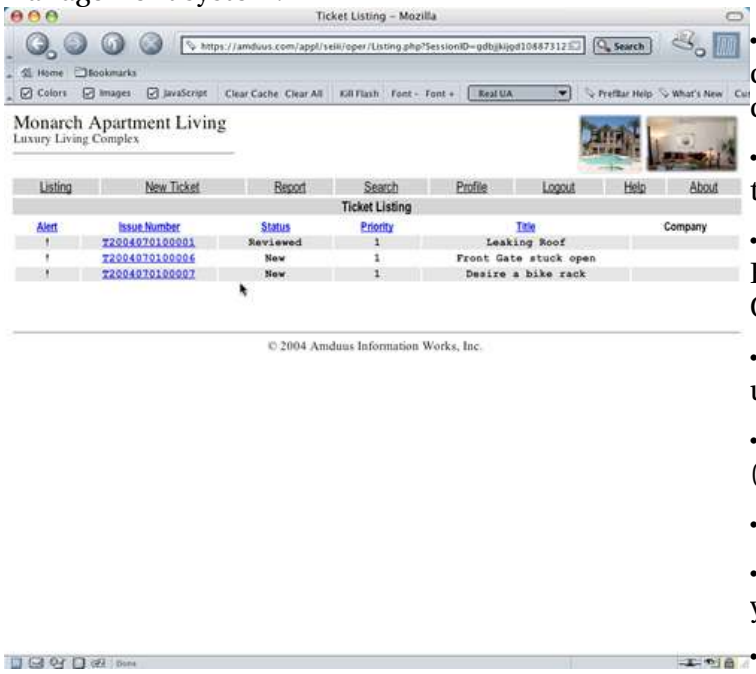
```
13/07/2004 20:50:04 Level 1
13/07/2004 20:50:04 Level 2
13/07/2004 20:50:04 Level 3
```

The source code mentioned in this article, as well as more OO source, can be found in http://amduus.com/OpenSrc/SrcLib/OOP/.

Scott Auge is the founder of Amduus information Works. He has been working with Progress technologies since Version 6. He works with UNIX platforms dealing with integration and web based applications.

Service Express

SEIII is golden and ready for use. Below find SEIII configured for an apartment management system.



- Allow your external customers to manage and create their tickets.
- Internal users manage all tickets.
- Web based – use Internet Explorer, Mozilla, Safari, or Opera.
- Easy to use, easy to understand.
- Configurable statuses (workflow)
- Configurable priorities
- Configurable HTML areas for your look and feel.
- For more information contact Scott Auge at sauge@amduus.com.

I received the following word on a job available!!!

Hi Scott, I have a client located in Brecksville, Ohio which is a southern suburb of Cleveland, Ohio. They are looking for 6 Progress developers to work on their long-term assignment (at least a year). If you are interested in hearing more please email me your resume along with a time that we can talk? Or if you know of any Progress developers and would like to pass this email on to them I would appreciate it. Have a great day! Thanks Dave Hinderschied Chagrin Consulting Services, Inc. 216-514-3301x24 216-514-3302 Fax dhinderschied@chagrinconsulting.com

Publishing Information:

Scott Auge publishes this document. I can be reached at sauge@amduus.com.

Amduus Information Works, Inc. assists in the publication of this document by providing an internet connection and web site for redistribution:

Amduus Information Works, Inc.

1818 Briarwood

Flint, MI 48507

http://www.amduus.com

Other Progress Publications Available:

This document focuses on the programming of Progress applications. If you wish to read more business oriented articles about Progress, be sure to see the Profile's magazine put out by Progress software http://www.progress.com/profiles/

There are other documents/links available at http://www.peg.com .

There is a web ring of sites associated with Progress programming and consultants available at http://i.webring.com/hub?ring=prodev&id=38&hub .

White Star Software publishes a commercial document called "Progressions." It is simular to this document but with different content. More information can be found at http://wss.com/. White Star also publishes Progress Programming books!

Article Submission Information:

Please submit your article in OpenOffice[1] format or as text. Please include a little bit about yourself for the About the Author paragraph.

Looking for technical articles, *marketing Progress* articles, articles about books relevant to programming/software industry, white papers, etc.

Send your articles to sauge@amduus.com! Thanks!

1 OpenOffice is a freely available Office Suite for Windows, Apple, and *NIX based operating systems. You can download it at http://openoffice.org. This document is edited on OpenOffice.

Order Form for Progress Open Source CD-ROM

COUPON 001A

This is an offer for the CD-ROM at lower than list savings!

This is a great way to support the E-Zine too!

Mail this form to:

Amduus Information Works, Inc.

1818 Briarwood

Flint, MI 48507

Please send _____ copies of the Open Source CD-ROM at $35.00 per disk to:

Name	_____
Company	_____
Address	_____
City	_____
State	_____
Zip	_____

Please make your checks/money orders out to: Amduus Information Works, Inc. Cash works too!

This offer only valid in the United States of America and those countries with postal agreements with the United States Post Office.

The CD-ROM includes (all source code included):

- Blue Diamond/IRIS – Webspeed alternatives
- Survey Express – easily create text templates of surveys and then have the program generate the web pages automatically
- Service Express – Web based Help Desk.
- The Progress E-Zines, books on learning to program in Webspeed (PDF/Word/HTML)
- Denkh HTML Reporter – web based report writer
- CMS – a web content management system
- DB Email – Use pop3 to download emails into a Progress database
- Neural Networks – experiments in spam recognition and text message classification
- Denkh – create PDF file reports for Webspeed/UNIX CHUI!
- More!