Progress

Electronic Magazine

An Amduus™ Information Works, Inc. Publication

You can find an archive of these E-Zines here: http://amduus.com/OpenSrc/FreePublications/

Table of Contents

The information contained in this document represents the current view of the community or Amduus on the issues discussed as of the date of publication. Because the community or Amduus must respond to changing market and technological conditions, it should not be interpreted to be a commitment on the part of the community or Amduus, and the community or Amduus cannot guarantee the accuracy of any information presented . This paper is for informational purposes only. The community or Amduus MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. Product and company names mentioned herein may be the trademarks of their respective owners.

Publisher's Statement

I will finish off the set of data structures I mentioned in previous issues of a stack, queue, and in this issue - the list. Sometimes a comma delimited string manipulated by ENTRY () and NUM-ENTRIES() will do the trick – other times – a more sophisticated list management tool needs to be used. While the former way allows one to add items to the front and the back, the list object allows one to add items in two additional ways. And have you ever try to delete a specific entry? The list object allows one to get past the limitations of a REPLACE() which might replace everything!

Am I saying to remove all your coding that uses ENTRY() and NUM-ENTRIES()? Nope. If that fits the bill, then that is the way you should go. I always believe in the "Less is More" mantra and one should use the simplest methods to achieve a goal. But – when you find yourself twisting and bending and convoluted around those functions to achieve your goal – you may want to consider a more sophisticated list management tool.

Also in this issue, I talk about adopting open source in your organization. Many who read this E-Zine know I am big on open source – nearly a zealot in some eyes. But not all open source is good open source – especially in the context of your company or organization. In this issue I mention some things one might want to think about when developing an open source policy for the company.

Have fun!

Scott Auge
sauge at amduus dot com

Developing an Open Source Policy

By Scott Auge
sauge @amduus.com

So undoubtedly you have been hearing about this free software that is available out on the Internet.

Now Open Source is finding it's way into the functions and infrastructure of companies. A lot of start up companies are taking the plunge with Open Source simply because the venture capitalists are not knocking at anyone's door these days.

Here are some thoughts about what to take into consideration when implementing open source in your company.

Determine the languages accepted

One of the most important aspects for considering the use of an open source project is what language is the application/tool written in. Is the language in the skill set of the developers in your organization? Can you find developers with those language skills? If not, then it might be awfully hard to modify the code to meet your changing business requirements or to correct software defects.

Some other considerations:

- Does the language have any books on it?
- Are there other projects written in that language?
- Is it a common language inside your organization?
- Is it a common language outside your organization?

If you have developer's willing to learn the language, it might certainly be worth walking off into a new area for your organization.

Determine the licenses accepted

Your policy should also explore the acceptable licensing for the software and any of it's ancillary required software. Most of these licenses can be seen at www.opensource.com and most are quite clear. Once you start reading the licenses, you will discover most of them are reworks of the following base licenses: GPL, LGPL, BSD, and the MIT licenses. Once you understand those base licenses, you will have a gist of what the other licenses are about.

These licenses have different legal requirements should you choose to redistribute the code outside of your organization in a commercial or free-for-all manner.

If you do not plan on redistributing the code outside of your organization, then one only needs to take into account if there is pricing or use restrictions[1] on the license.

Determine the support level

Once you start using the package, you will undoubtedly come to the point where you will need some kind of help.

One should consider if there are books out there on the given application – for example a browse on http://amazon.com or http://bn.com should tell you that. The existence of a book signifies a very mature software project. One should consider both paper and “electronic” books. (After all, you are reading an “electronic” magazine!) Some example projects that have books associated with them are Apache, Linux, Sendmail, Yacc, etc.

Does the project have a mailing list of some kind or web forum? This is also a good sign for a project. One can also gage the amount of help that might be out there and the

[1] I came across of piece of software with a use restriction against being used on adult websites or alcohol related web sites. I did complain to the license holder about these “morality” clauses and how one could make a whole new legal mess regarding licensing should they become more common. They were removed for that project. Phew!

amount of activity on the software. (One should remember that simple or very mature software may “just work” and so the support activity on web sites and mailing lists could be low.)

Does the project come with a manual, HOW TO, or at least INSTALL directions? There are some people out there who put together practically unusable software simply because they did not take the time to build a manual for the software. If you figure it out – maybe that can be your contribution!

Is the project supported commercially? Some companies will support an application or set of applications for a fee. If there are multiple organizations – all the better. Often the original authors or the project managers of the project will be willing to offer support on a contract/contractor basis. If they can't themselves (prior obligations, geography, or simply “we don't do that”) they probably will know someone who would be willing.

Finally, a developer can usually make heads or tails out of a piece of software if given enough time to become familiar with it. This is one of the real values of open source – the ability to see what makes it tick, and how to make it tick for you. The other value is, if the originating company goes belly-up, or through a “PeopleSoft” event – you are still OK for using that software until you can find a replacement (or become the new originating organization!)

Determine the packaging sophistication

By packaging sophistication I don't mean the box the software might have come in! I mean is it easy to install and remove from your system?

Can you set up multiple versions of the software on the same system? (That way you can test/modify a new version while the previous version is busy doing what it is doing.)

A package manager such as rpm or “what have you” may not be the be all and end all requirement to meet this. If the software has it's own install and removal tools[2] that can work also.

Determine ancillary packages needed

One thing you should get use to, is that in the Open Source world, a lot of work sometimes is dependent on other work. It is the whole "Standing on the shoulders of giants" thing.

Sometimes a program will be dependent on another program. For example, one may like PHP programs a lot (such as the service desk app SEIII put out by Amduus!) – but a lot of them depend on a database. Let's say you have OK'ed the use of MySQL, but the program authors say PostgreSQL is really the way to go with their software. You should check out what you are getting into with PostgreSQL then. (You will probably be a lot happier.)

You may need to soften up your standards to achieve the real value of the original program. But you should also determine what you are getting into in terms of license, support, and skills needed. You can look at it as a means of increasing your technology skills set.

Determine the process for bringing Open Source into the company

This is really up to the company. Hopefully the company will make up a policy document that takes these considerations I have been writing about and gives answers to them.

The document should list the languages allowed, the licenses allowed, the ancillary programs that might be allowed, the amount of support that is out there – etc. It should also include the way to amend the document for different languages etc. One shouldn't penalize the business for lack of skills and understanding in IT.

[2] I am a compile from the source kind of guy. I am quite happy when the make file has a "clean," "install", and "remove" option to it. I am especially happy when I can specify the binary and library directories to use for compilation and for installation.

Determine the process for releasing Open Source from the company

One of the things one really does not want to do is release source code that is of value to the company on the software market[3]. Let me get you in on a secret about open source. Often it is created by an organization or company that actually uses it as a base for something else they consider their value proposition.

For example, the Apache web server wasn't made by Web Server companies – it was made by people who make web pages – who want to take the web to the next level as a means of competitive and strategic advantage compared to people on “old technology.” It was made by the people who create web pages – not by the people who create web servers[4].

Often Open Source is created by people who are on the “next level” of the chain of value. It might seem strange that competitors might collaborate on a web server – but they really don't consider themselves competitors at “that level.” They are sure enough about the web page content they create and look for the opportunity to share the base technology at a reduced price.

If your company is not in the business of making software, but is in the business of making widgets – often releasing open source code is not a big deal. It is not like one was going to package and sell it to begin with. Hopefully someone will install it in their place and add to it. Then they share their goodies and all are the better for it. The one caveat might be if that software release can tell the competition some secret sauce you use in your processes. *The truth is the competition often thinks their way of doing things is better anyhow.*

Finally, of course, releasing source code and an application DOES reflect on your

[3] Some people DO release open source that depend on commercial tools. For example, there is a building of routines for the Progress 4GL world, but of course Progress is not an open source software product.

[4] As is often with most commercial companies – trying to get something into their product is like trying to move the Pyramids of Giza with a mule. With open source you simply plop it in there.

company to a degree. Keep the code nice and tidy and make a little manual to help people implement it. You don't have to become a support company to release it. Believe me, the people who adopt it will create a support network if your code is really useful[5].

When people adopt it, you will start seeing all kinds of functionality and probably bug fixes you have never encountered being added to the software. Hey – can you complain about free programming by giving a little?

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.

[5] The Progress Email List and other web based forums are examples of this.

Donations

Do you find something useful in the E-Zine now and then? Do you think it holds value for your education in the 4GL and to learn what is out there?

It does take money to produce this electronic magazine. Sure I can cut corners by not needing paper, ink, or postage – but bandwidth still costs x amount of dollars every month.

Even with Linux server, hardware does cost money – and the server is five years old. It's due for an upgrade.

Yes, I use OpenOffice.org to edit the document on, but throw in the laptop and that is a few more dollars.

Throw in a developer's Progress license – well... we all know how much that puppy is! It sure is useful for writing code with! I would like to get on version 10.

All of the above is going to cost about $10,000.00 this year. Bet you didn't know it costs so much for a free publication!

We all know there are not that many publications for Progress based technology. You can help yourself, and others, by donating to “the cause.”

Your Name: ____

Organization: ____

___ $5.00 ___ $10.00 ___$20.00

Consider this address good until June 2005:

Scott Auge
222 East Riverside #302
Austin, TX 78704
United States of America

Thanks for your support!

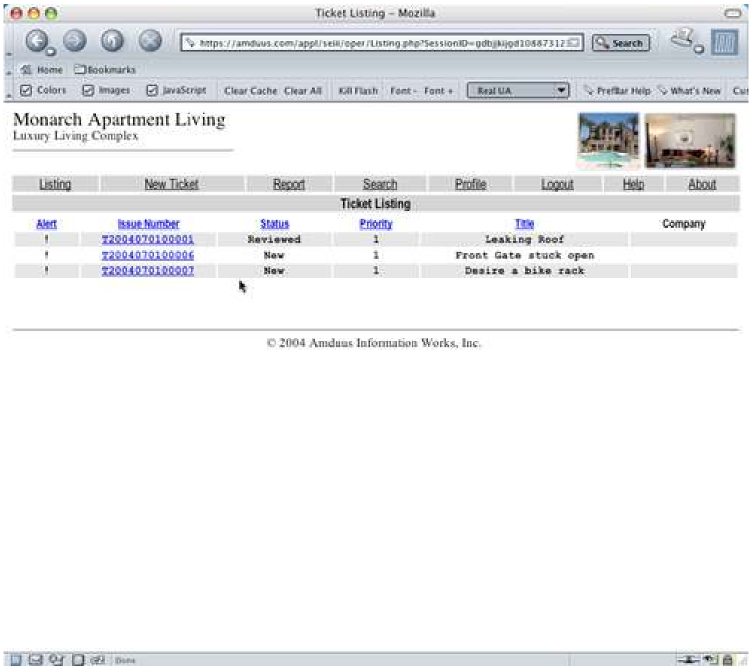
Scott

Advertisement

Service Express Is Now Open Source under the GPL license!

Service Express is golden and ready for use. Below find Service Express configured for an apartment management system, though it is flexible enough to be used by help desks in nearly any kind of industry for smaller businesses.

SAVE MONEY! SAVE STRESS!



Monarch Apartment Living
Luxury Living Complex

Ticket Listing

Alert	Issue Number	Status	Priority	Title	Company
!	T2004070100001	Reviewed	1	Leaking Roof	
!	T2004070100006	New	1	Front Gate stuck open	
!	T2004070100007	New	1	Desire a bike rack	

© 2004 Amduus Information Works, Inc.

- 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 or see http://amduus.com/serviceexpress

Service Express is available in three ways:

1. As an Application Service from Amduus Information Works, Inc.
2. As a leased application on an Amduus provided machine.
3. As a GPL Open Source licensed program for use on your machine (Free!!!)

Implementing A List With Persistent Procedures

By Scott Auge

When we go to school to learn computer science or mathematics, we often take that first Comp 101 course studying data structures. If you don't know about data structures, I recommend you hit amazon http://www.amazon.com/exec/obidos/ASIN/0262032937/qid=1106878827/sr=2-1/ref=pd_ka_b_2_1/104-9886269-2431926 for "Introduction to Algorithms" or of course, Volume I of "The Art Of Computer Programming" at http://www.amazon.com/exec/obidos/tg/detail/-/0201485419/qid=1112814362/sr=8-1/ref=pd_csp_1/102-4111809-5836960?v=glance&s=books&n=507846

Unlike the stack and the queue, a list data structure is a little more complicated. There are more "directions" of travel in a list. In a list, one needs tends to slide around from front to back and in between. There are more ways of inserting and deleting data from the list than a simple pop() or dequeue() routine. One also has to come to terms with the idea of a cursor of sorts. Lets take a quick look at the operations available:

GoFront – Move the current list cursor position to the front of the list.

GoRear – Move the current list cursor position to the end of the list.

GoNext – Move the current list cursor position one step to the right.

GoPrev – Move the current list cursor position one step to the left.

JumpTo – Move the cursor to a specific position.

GetLoc – What location is the cursor at?

FindLoc – For a given value, find it's location in the list.

Remove – Remove the value at the current cursor location. Items are shifted to the left.

AddBefore – Add a value before the current cursor location.

AddAfter – Add a value after the current cursor location.

AddFront – Move the cursor to the front of the list and insert the value before the cursor.

AddRear – Move the cursor to the end of the list and insert the value after the cursor.

ReadLoc – Given a location return it's value. Cursor is left at that location.

ReadValue – Obtain the value in the list where the current cursor is located.

Count – How many items are in the list.

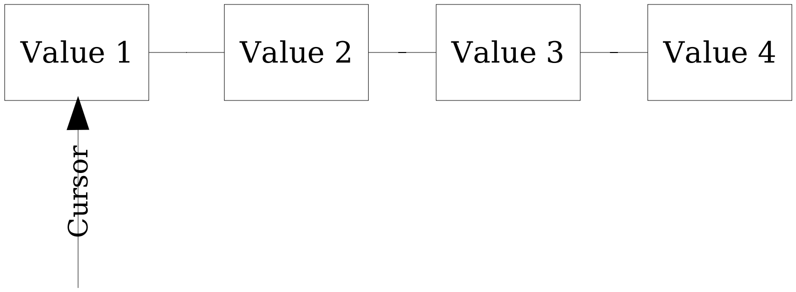
IsEmpty - Is the list empty?

ClearList – Delete all entries in the list.

Note the following errors could happen:

```
100 – List Is Empty
101 – Beyond List Length
```

It is easier to think of the list as a series of boxes. In each box contains a value and each box is the left or the right of the others. The front of the list is to the left[6] and the rear of the list is to the right.



Some code that might create this condition is:

```
RUN AddRear IN OMGH(“Test”) (“Value 1”).
RUN AddRear IN OMGH(“Test”) (“Value 2”).
RUN AddRear IN OMGH(“Test”) (“Value 3”).
```

[6] Sorry middle eastern readers! As a condolence understand direction really isn't that important and you can think of it as right to left. The operations will be the same “direction” you entered the data into.

```
RUN AddRear IN OMGH(“Test”) (“Value 4”).
RUN GoFront IN OMGH(“Test”) .
```

One basically moves the cursor around with the GoFront, GoRear, GoNext or GoPrev routines.

One adds data with the AddRear, AddFront, AddBefore and AddAfter routines. Remember that the cursor places it's self at these new entries.

Here is some testing code to help you comprehend how to use the list object:

```
DEFINE VARIABLE hList AS HANDLE NO-UNDO.
DEFINE VARIABLE i AS INTEGER NO-UNDO.
DEFINE VARIABLE c AS CHARACTER NO-UNDO.
DEFINE VARIABLE x AS CHARACTER NO-UNDO.

RUN obj_list.p PERSISTENT SET hList.
RUN Init IN hList ("NoName").

/* DCBA */

RUN AddFront IN hList ("A").
RUN AddFront IN hList ("B").
RUN AddFront IN hList ("C").
RUN AddFront IN hList ("D").

RUN DisplayList.

/* ABCD */

RUN ClearList IN hList.

RUN AddRear IN hList ("A").
RUN AddRear IN hList ("B").
RUN AddRear IN hList ("C").
RUN AddRear IN hList ("D").

RUN DisplayList.

RUN ReadLoc IN hList (3, OUTPUT x).
DISPLAY x.

/* Create a list ADCB */

RUN ClearList IN hList.

RUN AddAfter IN hList ("A").
RUN AddAfter IN hList ("B").
RUN AddAfter IN hList ("C"). /* At A, so moves B over */
RUN AddAfter IN hList ("D"). /* At A, so moves C over */

RUN DisplayList.


/* Create DCBA */

RUN ClearList IN hList.

RUN AddBefore IN hList("A").
RUN AddBefore IN hList("B").
RUN AddBefore IN hList("C").
RUN AddBefore IN hList("D").

RUN DisplayList.

/* BeforeD,D,AfterD,C,B,A */

RUN GoFront IN hList.
RUN AddBefore IN hList ("Before D").
RUN GoNext IN hList.
RUN AddAfter IN hList ("After D").

/* RUN DisplayList. */

/* Remove D as we are "left there" from above                    */
/* Remember that DisplayList moves our "position" around in the list! */

RUN Remove IN hList.
RUN GetLoc IN hList (OUTPUT i).

DISPLAY i.
RUN DisplayList.

RUN FindLoc IN hList (INPUT "B", OUTPUT i).

display i.
RUN JumpTo IN hList (i).
RUN ReadValue IN hList (OUTPUT c).
DISPLAY c.

DELETE PROCEDURE hList.

PROCEDURE DisplayList:

  DEFINE VARIABLE iCount AS INTEGER NO-UNDO.
  DEFINE VARIABLE iListCount AS INTEGER NO-UNDO.
  DEFINE VARIABLE c AS CHARACTER NO-UNDO.

  RUN Count IN hList (OUTPUT iListCount).

  RUN GoFront IN hList.
  DO WHILE iCount < iListCount:

    RUN ReadValue IN hList (OUTPUT c).
    DISPLAY C.
    DOWN.

    RUN GoNext IN hList.
    ASSIGN iCount = iCount + 1.

  END.

END.
```

You can find this source code at: http://amduus.com/OpenSrc/SrcLib/OOP/list.zip

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.

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 similar 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[7] 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!

[7] 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.