The Progress Electronic Magazine

In this issue:

Did you sign up to receive this E-Zine? Send email to sauge@amduus.com to subscribe or fill out the forms at http://www.amduus.com/online/dev/ezine/EZineHome.html ! It's free! (Though donations are certainly welcome – whatever you feel is fair!)

Though intended for users of the software tools provided by Progress Software Corporation, this document is NOT a product of Progress Software Corporation.

Publisher's Statement:

Last issue's management article on scoping progress applications gave me the idea of sharing tips on how to get up to speed on a progress application. Hopefully people will write in with their tips about how to get up to speed on programming that can be shared with everyone else to make programming with Progress tools all the more productive.

Also I see there are many questions on the PEG about using cookies within web applications. This issue examines the use of cookies to identify a user and to determine if that user should be on the web page in question. It gives some sample code to work with to get functional with cookies in an application right away.

I am looking for people who are interested in reselling Amduus Information Works applications. Some are done (as if software is ever done) and some are not. Please contact me if your interested at sauge@amduus.com. Also, if you have any Webspeed development work that you would like to farm out – I am looking for that kind of activity also. My resume is available here: http://www.amduus.com/Resumes/ScottAuge.html

In the future, Amduus is going to be giving away actual applications of significant functionality. Service Express is the first one to be tried out. The idea is to see if Open Source generates programming hours for the people out there capable of modifying it. It appears from the survey taken that nearly 50% of the people using an Open Source application would hire the people that wrote it for any modifications. An interesting experiment indeed and I will let you all know how it goes.

If you have not already, please be sure to take this survey about Open Source:

Click http://www.amduus.com/online/survey/PrmPresentSurvey.html?SurveyID=aQGs91zXeI16 to take the survey. Click http://www.amduus.com/online/survey/PrmSurveyResult.html?SurveyID=aQGs91zXeI16&PassCode=Yw7aVifx6h17 to see the results.

To your success,

Scott Auge
Founder, Amduus Information Works, Inc.
sauge@amduus.com

Coding Article: Using cookies to login and identify a web user for security purposes

Written by Scott Auge

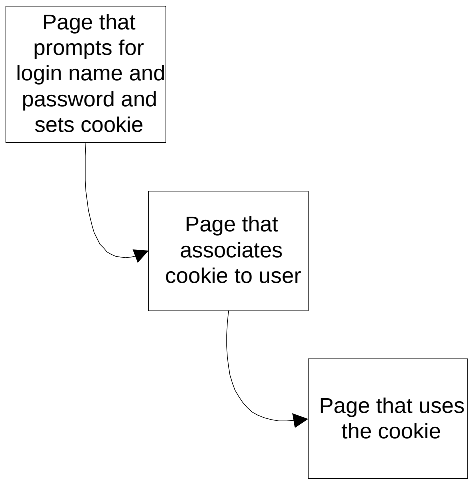
Often there are questions about how to use a cookie to identify a user, as well what exactly is identified of the user by the cookie.

> *Myth control: People often want to turn cookies off for "privacy." If these cookies, are turned off, then one must use hidden fields and name-value pairs in URLs to continue to identify the user. The fact is, the data used to identify the user is merely in a different form, instead of being in a cookie data structure, it is in a hidden field or URL data structure. Without these values, your website cannot identify individual users.*

This article will explain how to use a cookie to login in a user, and how to continue to identify that user in subsequent web pages of your web application.

This article assumes you understand the use of FORMs and Cookies in an HTML environment. It focuses on the Webspeed[1] implementations to get information from these constructs within the Webspeed environment.

The WebState table is used, this can be referenced in Issue Two of the Progress E-Zine. In short, the general architecture of the pages are as shown:


Page that prompts for login name and password and sets cookie
Page that associates cookie to user
Page that uses the cookie

[1] Blue Diamond and many other Webspeed alternatives also use these constructs.

Sample code of a web page that sets the cookie value

```
DEF VAR lCookieValue    AS CHARACTER NO-UNDO.

{MakeID2.i}

/* We set the cookie value here, so that it is available in later pages. */
/* This helps reduce programmer stress.  Later on in the AdminHome.html  */
/* page, the cookie value is associated to a user via the WebState table.*/

PROCEDURE OUTPUT-HEADERS:

  ASSIGN lCookieValue = MakeID2(20).

  SET-COOKIE("UserTrack", lCookieValue, ?, ?, ?, ?, ?).

END. /* PROCEDURE OUTPUT-HEADERS */
```

Under Webspeed E4GL programming style, the use of the OUTPUT-HEADERS procedure is to be called before any other code. This program allows output of the cookie syntax in the header of the HTTP message before outputting the HTML. If SET-COOKIE() is called after the header of the HTTP message, the text of the cookie will be rendered on the window of the browser, instead of being interpreted by the browser as setting a cookie. See `sga_util.i` in Blue Diamond for more information about how a Cookie is set in the HTTP Header of a HTTP message.

Generally a random string of some characters and digits is created and deposited on the user's browser from the page that they are accessing the web application on. I do this because the value of the cookie is immediately available in following pages. It is on the following pages, that accept the login name and password of the user, that will take their cookie value and associate it to a user.

Sample code to differentiate a login or checking of the user

```
DEF VAR lUserID     AS CHARACTER NO-UNDO.
DEF VAR lPassword   AS CHARACTER NO-UNDO.

/* Determine if the user is available */

ASSIGN
lUserID = GET-VALUE("UserID")
lPassword = GET-VALUE("Password").

/* If we are coming in from the Login.html page, then the UserID will be */
/* populated, run the LoginUser procedure to get them straightened out.  */

IF lUserID <> "" THEN RUN LoginUser.

/* Otherwise, they are coming in from some other page, so we need the */
/* cookie to determine who they are and if they have any business on  */
/* this page.

ELSE RUN CheckUser.

/* Render the page */
-->
```

The user may come into a menu page for the first time to be logged in, or from a link to the menu page from inside the application. The above code determines if there is a NVP (Name Value Pair) for UserID and uses that to determine if it is a login or merely a return to the page.

Code to login the user

Logging in a user determines if the user has a record recording them as available to the application and if they have any business being on the web page. The mechanics below determine if a pre-existing SecUser record is available for the user. I have a table Person to identify people, and SecUser to identify people who have access to the application. This is because a person may or may not be a user of the application. (Ie, a person could be merely a contact for an entity in the application.) I have found that keeping data related to users of an application in it's own table versus part of an overall table for "people" much more flexible and easy to understand.

```
PROCEDURE LoginUser:

  FIND SecUser NO-LOCK
  WHERE SecUser.Login = lUserID
    AND SecUser.Password = lPassword
  NO-ERROR.

  IF NOT AVAILABLE SecUser THEN DO:

    RUN ipRejectPage ("Sorry, cannot find your user/password combination.").
    RETURN.

  END.  /* IF NOT AVAILABLE SecUser */

  /* Associate the user to the cookie */

  ASSIGN lCookieValue = GET-VALUE("UserTrack").

  RUN SetUserCookie.p (INPUT lCookieValue,
                       INPUT SecUser.SecUserID).

  /* Determine if the user has any business on this page */

  RUN SecurePage.p (INPUT SecUser.SecUserID,
                    INPUT "AdminHome.html",
                    OUTPUT lIsOK).

  IF NOT lIsOK THEN DO:

    RUN ipRejectPage ("Sorry, your userid cannot access this page.").
    RETURN.
  END.  /* IF NOT lIsOK */

END. /* PROCEDURE LoginUser */
```

In short, the code looks for a pre-existing record for the userid and password. If not available, then the user is sent to a reject page which has the proper HTML formatting to elegantly tell them they are not allowed into the system.

If we did find them, we associate their cookie value with the user. This is done by making an entry in the Webstate table using the cookie value as the WebState.SessionID and the data value the unique identifier for the user. It is at this point that the user is “logged in” as we can reference the cookie value to a SecUser record. See Issue Two of the E-Zine for more information about the use of the WebState table and it’s other fields.

In amduus databases, each record has a unique identifier on it – sort of an embedded RowID for the record. This is so individual records can be keyed on one field when need be. Unlike a RowID, when a dump and load of the database is performed, the value is guaranteed to be the same (this is why I use the field, instead of storing the ROWID of a record.)

Finally, we determine if the user has any business being on that certain page. This is done by calling a module called SecurePage.p naming the user and naming the page the user is on. Note this code does not determine the data the user has access to, only the page.

A very simple method is to make database entries in the SecPage table matching user to page name. If there exists a record for the user on that page, then SecurePage.p should return a YES answer stating they are good to go. If not, it should send a NO answer and then the reject page routines will come up.

To check a logged in user

Checking who and if a logged in user has any business on the page is similar to the steps of logging in a user. Instead of associating a user to a cookie, we use a cookie to find the user. This is done by retrieving the cookie value and looking up the user in the WebState table.

If the user cannot be found, then the program will proceed to the rejection page.

If the user is found, it proceeds to the determination if the user has any business on the page.

```
PROCEDURE CheckUser:

  DEF VAR lSecUserID   AS CHARACTER NO-UNDO.
  DEF VAR lCookieValue    AS CHARACTER NO-UNDO.
  /* Determine who the user is based on the cookie */

  ASSIGN lCookieValue = GET-VALUE("UserTrack").

  RUN GetUserCookie.p (INPUT lCookieValue,
                       OUTPUT lSecUserID).

  FIND SecUser NO-LOCK
  WHERE SecUser.SecUserID = lSecUserID
  NO-ERROR.

  IF NOT AVAILABLE SecUser THEN DO:

    RUN ipRejectPage ("Sorry, your cookie is not valued correctly").
    RETURN.

  END.  /* IF NOT AVAILABLE SecUser */

  /* Determine if the user has any business on this page */

  RUN SecurePage.p (INPUT SecUser.SecUserID,
                    INPUT "AdminHome.html",
                    OUTPUT lIsOK).

  IF NOT lIsOK THEN DO:

    RUN ipRejectPage ("Sorry, your userid cannot access this page.").
    RETURN.

  END.  /* IF NOT lIsOK */

END. /* PROCEDURE CheckUser */
```

Ancillary code:

MakeID2 function

The MakeID2 function generates a usually unique value to be used by a program. See the Open Source area of http://http://www.amduus.com or Issue One of the E-Zine for more on the `RandomString.p` procedure.

```
/* MakeID.i
 *
 * Function returns a pure random string.  Not guaranteed to be unique however,
 * but most likely will be.
 */

DEF VAR RCSVersionMakeSeq AS CHARACTER INIT "$Header:
/home/appl/survey/src/RCS/
MakeID2.i,v 1.1 2001/10/15 07:18:49 sauge Exp $" NO-UNDO.

FUNCTION MakeID2 RETURNS CHARACTER (INPUT pLength AS INTEGER):

  DEF VAR i AS CHARACTER NO-UNDO.
  RUN RandomString.p (INPUT pLength, OUTPUT i).

  ASSIGN i = i + STRING(ETIME).

  RETURN i.

END. /* FUNCTION MakeID2() */
```

GetUserCookie.p

```
/* GetUserCookie.p */

DEF VAR RCSVersion AS CHARACTER INIT "$Header:
/home/appl/survey/src/RCS/GetUser
Cookie.p,v 1.1 2001/10/27 00:11:04 sauge Exp sauge $" NO-UNDO.

DEF INPUT  PARAMETER pSessionID  AS CHARACTER NO-UNDO.
DEF OUTPUT PARAMETER pSecUserID  AS CHARACTER NO-UNDO.

FIND WebState NO-LOCK
WHERE WebState.SessionID = pSessionID
  AND WebState.Category = "Login"
  AND WebState.Name = "SecUser"
NO-ERROR.

IF NOT AVAILABLE WebState THEN DO:
  ASSIGN pSecUserID = ?.
  RETURN.
END.

ASSIGN pSecUserID = WebState.Data.
```

SetUserCookie.p

```
/* SetUserCookie.p */

DEF VAR RCSVersion AS CHARACTER INIT "$Header:
/home/appl/survey/src/RCS/SetUser
Cookie.p,v 1.1 2001/10/27 00:11:04 sauge Exp sauge $" NO-UNDO.

DEF INPUT PARAMETER pSessionID  AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER pSecUserID  AS CHARACTER NO-UNDO.

/* This will be a problem if the SessionIDs are not unique between users */
/* In effect, the user has become another user!  Make sure when creating */
/* the session id, that sessionid does not already exist.                */

FIND WebState NO-LOCK
WHERE WebState.SessionID = pSessionID
NO-ERROR.

IF AVAILABLE WebState THEN RETURN.

CREATE WebState.
```

```
ASSIGN
WebState.SessionID = pSessionID
WebState.Category = "Login"
WebState.Name = "SecUser"
WebState.Data = pSecUserID.
```

Database tables used in this article:

```
==========================================================================
============================= Table: SecUser ============================

              Table Flags: "f" = frozen, "s" = a SQL table


Table                          Dump     Table Field Index Table
Name                           Name     Flags Count Count Label
------------------------------ -------- ----- ----- ----- -------------------
SecUser                        secuser            6     3 SecUser

  Description: Security information for a system user
 Storage Area: N/A


============================= FIELD SUMMARY =============================
============================= Table: SecUser ============================

Flags: <c>ase sensitive, <i>ndex component, <m>andatory, <v>iew component

Order Field Name                Data Type   Flags Format          Initial
----- ------------------------- ----------- ----- --------------- ----------
   10 SecUserID                 char        i     x(8)
   20 PersonID                  char        i     x(8)
   30 Login                     char              x(8)
   40 Password                  char              x(8)
   50 SecProfileID              char        i     x(8)
   60 Confirmed                 logi              yes/no          no

Field Name                     Label                 Column Label
------------------------------ --------------------- ---------------------
SecUserID                      SecUserID             SecUserID
PersonID                       PersonID              PersonID
Login                          Login                 Login
Password                       Password              Password
SecProfileID                   SecProfileID          SecProfileID
Confirmed                      Confirmed             Confirmed
```

```
============================= INDEX SUMMARY =============================
============================= Table: SecUser ============================

Flags: <p>rimary, <u>nique, <w>ord, <a>bbreviated, <i>nactive, + asc, - desc

Flags Index Name                       Cnt Field Name
----- -------------------------------- --- ---------------------------------
pu    key1                               1 + SecUserID

      key2                               1 + PersonID

      key3                               1 + SecProfileID

** Index Name: key1
 Storage Area: N/A
** Index Name: key2
 Storage Area: N/A
** Index Name: key3
 Storage Area: N/A


============================= FIELD DETAILS =============================
============================= Table: SecUser ============================

** Field Name: Confirmed
  Description: User confirmed to be who they say they are
         Help: User confirmed to be who they say they are

=========================================================================
============================= Table: SecPage ============================

              Table Flags: "f" = frozen, "s" = a SQL table


Table                          Dump     Table Field Index Table
Name                           Name     Flags Count Count Label
------------------------------ -------- ----- ----- ----- -------------------
SecPage                        secpage            3     2 SecPage

  Description: Pages Users are allowed access to
 Storage Area: N/A


============================= FIELD SUMMARY =============================
============================= Table: SecPage ============================
Flags: <c>ase sensitive, <i>ndex component, <m>andatory, <v>iew component

Order Field Name                Data Type   Flags Format          Initial
----- ------------------------- ----------- ----- --------------- ----------
   10 SecPageID                 char        i     x(8)
   20 PageName                  char        i     x(8)
   30 SecUserID                 char        i     x(8)

Field Name                     Label                 Column Label
------------------------------ --------------------- ----------------------
SecPageID                      SecPageID             SecPageID
PageName                       PageName              PageName
SecUserID                      SecUserID             SecUserID


============================= INDEX SUMMARY =============================
============================= Table: SecPage ============================

Flags: <p>rimary, <u>nique, <w>ord, <a>bbreviated, <i>nactive, + asc, - desc

Flags Index Name                       Cnt Field Name
----- -------------------------------- --- ---------------------------------
u     key1                               2 + SecUserID
                                           + PageName

pu    pukey                              1 + SecPageID

** Index Name: key1
 Storage Area: N/A
** Index Name: pukey
 Storage Area: N/A


============================= FIELD DETAILS =============================
============================= Table: SecPage ============================
```

About the author: Scott Auge is the founder of Amduus Information Works, Inc. He has been programming in the Progress environment since 1994. His works have included E-Business initiatives and focuses on web applications on UNIX platforms.
sauge@amduus.com

Management Article: Getting a programmer up to speed on an application

Written by Scott Auge sauge@amduus.com

As a software contractor, I found it a requirement to become familiar with complex applications very quickly. Some applications had nearly half a million lines of code in them, and it is a challenge to become functional in a short amount of time. So in order to not go insane, I came up with some methods to "get up to speed" with what is going on.

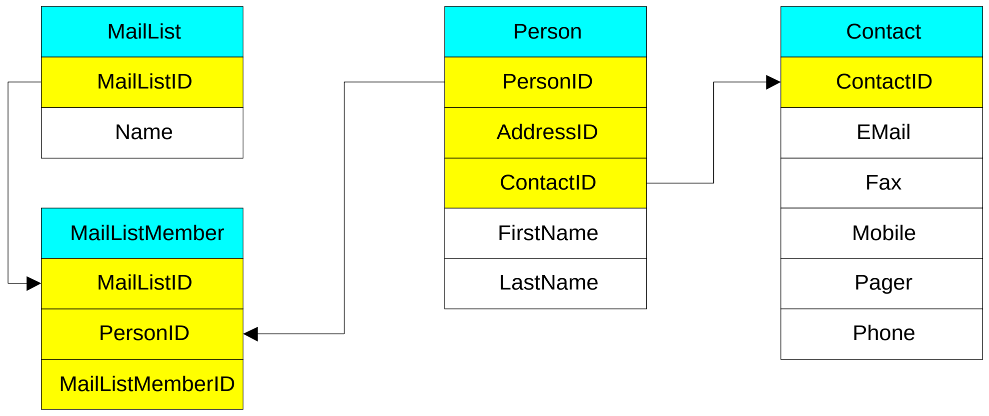
Examine Tables in the database

The core purpose of most database oriented applications is the tables within the database. Learning the database tables, fields, and relations will help the programmer learn what the end result data is that the programs have available to them to work with as well as create.

The first step is finding out the databases that are available for applications to connect with. Often there are multiple copies for development, testing, and production and knowing their names and connection parameters are important. Hence one should become familiar with the scripts (or icons) used to access the DB and application.

Once the DBs are known I will use the Data Dictionary to create a Progress Data Dictionary Report of the entire database to a disk file. By using a search feature in a text editing program, one can quickly get to the detail of a certain table. By using vi, I generally enter `/Table: Address` to zip right to the Address table report within the document. I have found it much easier to search on different items throughout the report under a text editor than to continuously use the Data Dictionary Report over and over. Plus it is easier to cut and paste with an editor.

Having that, I begin drawing up ER diagrams. These are the most useful things for a programmer, and sub-sets of diagrams can be drawn for certain functions or modules of the application. An example ER diagram is below:



Notice that the name of the table is in blue, fields that are indexed are in yellow, and pure attribute fields are in white.

Read the user manuals

Having a clue about what the application does is important. Time will need to be set aside to read the manuals of the application. And one should not expect the time to become familiar with the manuals to be any less than time than one would need to read a book (or two or three with some applications.)

When reading the manuals, one usually does not need to become familiar with the entire application. But one should focus on an overview of what the application is attempting to accomplish. Learn how to set up and view default configurations and codes as these will influence your algorithms (like including them!) The section you will need to work on as a programmer should be paid special attention.

Get out the yellow post it notes to mark important chapters and defaults configurations. Get your own copy of the manuals so you can write marginalia.

Read Technical Documentation/Specifications

If there are technical documentation available about the module of code you are working on, these should be read. They usually will document sub-procedures and functions you will have available to you when writing new functionality, as well where to look when you want to modify the functionality of the application.

An important difference between "reading the code" and reading the specifications, is that reading the code states exactly what the program *is* doing, while the technical documentation usually states what the program _should_ do. These can be two very different meanings when an application has a problem in it.

Technical documentation[2] about the programs in the application can be a quick way to learn what procedure or function does what. As an example:

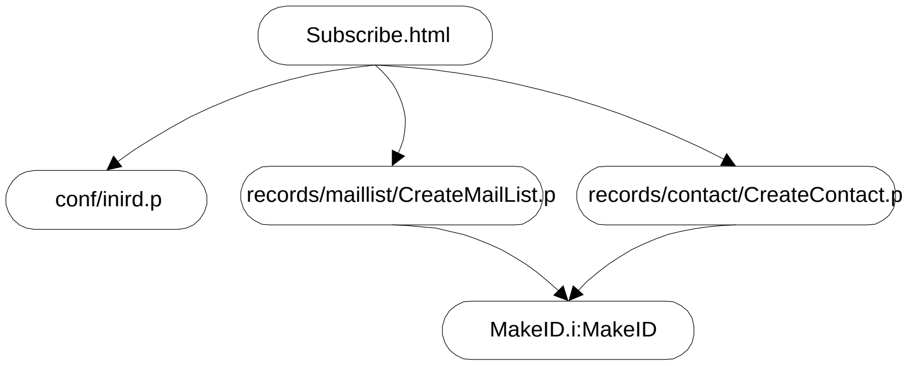
```
FUNCTION MakeID RETURNS CHARACTER (INPUT Seq AS INTEGER):
```

Returns an unique string based on the Sequence number for the ID field of a table.
Example:

```
{MakeID.i}
…
ASSIGN Address.AddressID = MakeID(NEXT-VALUE(AddressID)).
```

Read the code

When one reads the code, one wants to accomplish a few things other than slogging through line after line of source. Some of the things I do right away are "calling trees" which document how the code calls other code. An example is as below:


Subscribe.html
conf/inird.p
records/maillist/CreateMailList.p
records/contact/CreateContact.p
MakeID.i:MakeID

Note that if a call is made into an include that contains an internal procedure or function, the include file name is prefixed followed by a colon.

One also wants tools to determine how a field is used within the application. The use of the UNIX command find combined with grep through the directory structure can be useful for this:

[2] Amduus Information Works is creating a database web based requirements and documentation application. It would be helpful to know if such a thing would be considered by the reader's organization.

```
find . –name "*" –exec grep –il "expression" {} \;
```

An example with results for the expression "Street1" can be found with:

```
$ find /appl/amduus/src -name "*" -exec grep -il "Street1" {} \; 2> /dev/null
/appl/amduus/src/records/address/ModifyAddress.p
/appl/amduus/src/records/address/CreateAddress.p
```

This way when messing with a field value such as Street1, one can become more familiar with how it is used by other parts of the program as well how it might become populated.

With practice, this expression can be useful for all sorts of things. For example, when changing something in an include file, a search of the directory structure could be made to find what other files are influenced by that include file. Then more proper testing can be done not only on what your intended change is, but what unintended changes might occur in other parts of the application could be.

Become familiar with Xref files. This is a topic for a future article.

Associate Screens to code

User interfaces to the code of the application are important, and having a screen that makes the data more human readable is useful. When one needs to make a change to the screen of an application, one can quickly garner the complexity of the change by the screen's calling tree. This helps with determining the effort that might be needed, as well as other programs that could require modification based on the modification of the original program.

With Webspeed[3] applications the HTML page is a good starting point and very simple.

With CHUI/GUI[4] apps one usually needs to identify which procedure renders the screen (and in most cases, which part of the screen.) When doing so, one will want to become familiar with the menu-ing method for the application.

Some applications merely hard code the screens into a menu bar or the like.

[3] And Blue Diamond!

[4] CHUI is Character User Interface, GUI is Graphical User Interface

Other applications use a database table to identify screens, menu options, and the program that needs to be called. Under a database driven menu-ing system, one will want to print out that data most likely for quick reference.

Associate background processes to code

Often applications will have background processes associated with them. Since these are also places where data can be manipulated, one will want to become familiar with them and what their function is for. Particularly of they play a role in integrations. An addition of a field may require changes to processes not seen by the user that funnel data around the application landscape.

About the author: Scott Auge is the founder of Amduus Information Works, Inc. He has been programming in the Progress environment since 1994. His works have included E-Business initiatives and focuses on web applications on UNIX platforms.
sauge@amduus.com

Contractors/Consulting Companies Available:

If you do work in the Progress world – let me know and I will be able to include you! Price is $10.00 per listing per issue.

Analysts Express Inc.

The Reliable Source for I.T. Consulting and Recruiting
Contact: James Arnold @ 888/889-9091

02-1

Amduus Information Works, Inc.
http://www.amduus.com
scott_auge@yahoo.com sauge@amduus.com

Creation of modules and products for re-sale as well customized Internet/Intranet programming for E-Business in the marketing/manufacturing/service and law enforcement industries.

Community Announcements:

A place to announce Progress User Groups, Open Source Exchanges, etc. No charge!

Wonder where to find those Progress marketing articles?

http://www.progress.com/analyst/
http://www.progress.com/profiles/index.htm
http://www.progress.com/success/index.htm

Latest Progress in the News sighting!

http://www.cw360.com/article&rd=&i=&ard=107155&fv=1

Product Announcements:

Survey Software

Amduus Information Works, Inc. is creating survey software. This software can be used on a web site to query a population of people about their views and needs. The population could be internal to a company or external to yield a better understanding of the marketplace.

The software ships with source code for better adaptability to your company's application landscape and needs. The software was developed and designed on Linux with Progress Version 9 and Blue Diamond (Compatible with Progress Webspeed). The software can operate on AIX, Linux, Solaris, HP-UX and UNIXWare.

Customers and resellers are welcome to contact Scott Auge at sauge@amduus.com for more information. Street price is $2,000 per machine without Progress licenses.

The software can be rented out at 10 cents per response or $25.00 per survey (whichever greater) per week of taking results. See http://www.amduus.com/HTML_Manuals/RentingSurveyExpress.html for information about how to set up a survey with us.

Mail List Software

The publishing of this E-Zine has created the need to create some software that can handle the creation of mailing lists, web pages to subscribe and unsubscribe as well as a command line to distribute this file. This software is available for $50.00 per site. The User Guide is available at http://www.amduus.com/Manuals/mls.doc . Purchase before

October 15th and get a free update that will have a subscription verification feature and a list server.

Security SDK

The SDK works on E4GL source applications on Webspeed, Blue Diamond, or other Webspeed alternatives that use E4GL coding. Just one more thing brewing up at Amduus Information Works!

Publishing Information:

Scott Auge publishes this document. I can be reached at scott_auge@yahoo.com.

Currently there are over 400 subscribers and companies that receive this mailing! This mailing is not sent unsolicited, so it is not SPAM.

Amduus Information Works, Inc. assists in the publication of this document:

Amduus Information Works, Inc.
925 South Wolfe Road #26
Sunnyvale, CA 94086-8843
408-730-4588
http://www.amduus.com

Article Submission Information:

Please submit your article in Microsoft Word 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.