Zammi

A Propath Oriented Compiler Tool

Scott Auge
sauge@amduus.com

Amduus Information Works, Inc.
http://www.amduus.com

Table of Contents

Introduction

In the AppBuilder there is a propath compiler – of sorts. It is more like a collection of paths and it is up to the programmer to sort it all out. Once you move onto another project – that's right, you loose all those configurations should you need to come back.

This is something distinctly missing from the web based programming tools too. There doesn't seem to be a tool that will walk up and down the propath looking for HTML to convert and .p/.w's to compile. I have encountered applications with so many files in them, the File Tools in Web Tools funks out if one tries to compile them all at the same time.

There are other propath based compilers out there – but often they are parts of other tools and that means a lot of baggage for one piece of functionality. I don't like to wade through a bunch of other stuff (often poorly documented) only to do 1% of what it is capable of doing. The UNIX way is to have a certain tool to do a certain piece of functionality.

Another thing is with the AppBuilder, one cannot have nightly compiles (aka "builds.") Sometimes it takes an hour or so to compile stuff up. Might as well do it at night and have a report made available in the morning.

Sometimes on a tricky install, you simply want to compile the whole works in case the client decides to add fields to tables, etc. on their own.

If you can understand a script similar to this:

```
./compile \
 -propath "/appl/schdeve/work:/appl/schdeve/source" \
 -xref \
 -listing \
 -dlc /prg/101a \
 -pf /appl/schdeve/parm/all.pf
```

You pretty much have it under control.

System Requirements

The first release of this software is oriented to UNIX operating systems in some small ways. In the future this code can be enhanced for Windows.

So basically, for the software that comes with this manual you need:

- A UNIX Oriented Operating System
- OE Developers License

OR

- A UNIX Oriented Operating System
- 4GL Development License

This code has been tested on Progress Version 10 (aka OpenEdge) though it should work alright on Progress Version 9 installations.

License

This toolkit is licensed under the GPL Version 2 license. Basically it means, you can use it on how many machines you want (this doesn't work for Progress licenses remember) and you can redistribute the software as you please provided you provide source code and recognition to previous contributors.

Using this on an application YOU write doesn't mean YOUR application is covered by the license. You can do with YOUR code what ever you please. :)

Changing Zammi DOES fall under the GPL, so if you make changes to the /script or /src directories, then your changes fall under the GPL and it's requirements.

Installing

Putting The Software On The Disk

Installing is simple. Download the latest build from Amduus' open source directory (usually http://www.amduus.com/OpenSrc/SrcLib) It should be under the Zammi sub-directory.

The files will be named zammi.yyyyjjjhhmmss.zip where

yyyy represents the year of the build
jjj represents the day of the year of the build
hh represents the hour
mm represents the minute
ss represents the seconds

of the build.

Unzip the file and it will create a propathcompiler directory with src, doc, and script sub-directories.

Compiling the software

The software does not come compiled so it can work on multiple versions of Progress.

Simply make your working directory where the propathcompiler/src directory is installed. Then

```
RUN compile_project.p.
```

It should compile up into some r-code for you to use.

Configuration Of The Software

While you don't need to install the software along with your application code, you do need to configure the compile script a little.

In the section:

```
/*************************************************************************/
/* You will need to put the location of this toolkit's src here.         */
/*************************************************************************/
```

```
export TOOLKIT=/appl/propathcompiler/src
```

Simply tell the script where it can find the r-code for the compiler.

Invoking

Arguments To The Command

Invoking the software is as easy as a command line. If you invoke compile with no arguments it will tell you what it wants:

```
gaius:/appl/propathcompiler/script # ./compile
./compile
-pf          Where are the connection parameters found
-xref        Generate an XREF listing for each compiled file
-list        Generate a listing for each compiled file
-dlc         Where is your progress installation
-propath     What is your propath
```

An Example

An example is as from the introduction:

```
./compile \
 -propath "/appl/schdeve/work:/appl/schdeve/source" \
 -xref \
 -listing \
 -dlc /prg/101a \
 -pf /appl/schdeve/parm/all.pf
```

This is a short script called `img_compile` on my system that lets me compile an entire set of directories with one simple command.

Remember, the propath is for the application you wish to compile – not for the compiler tool it's self. That portion you set in the `compile` script.

A Log Of Activities

Once executing the command will put out a listing of what it did. For large sets of files it may be a good idea to redirect the output into a file.

The first thing it does is try to find files in the PROPATH that are qualified[1] to compilation. It will also pop out a date and time at the top so you know which day the log was created.

1 Qualification is done by a CAN-DO in walk_propath.p

When used under UNIX, the `find` command is used to make the list. It's invocation is recorded in the log.

Some example lines:

```
Time: 00:50:29
Date: 18/05/07
Using command: find /appl/schdeve/work -type f
Checking /appl/schdeve/work/RCS/applyipbannedrules.p,v
Checking /appl/schdeve/work/RCS/applyipbannedrules.r,v
Checking /appl/schdeve/work/paypayperiod.html
Adding /appl/schdeve/work/paypayperiod.html To Compile List
Checking /appl/schdeve/work/xlsreport.i
Checking /appl/schdeve/work/updpayweekno.html
Adding /appl/schdeve/work/updpayweekno.html To Compile List
```

Then, it will wander through this list. If it is a straight up .w or .p, it should compile it as normal.

```
Compiling /appl/schdeve/work/tchrheading.p
```

If it is an E4GL/SpeedScript based HTML file, it will convert the file and compile[2] it that way.

```
Compiling /appl/schdeve/source/grdstutransupdgpa.html
Converting HTML into .p for /appl/schdeve/source/grdstutransupdgpa.html
Object Type: web-object
Stored .p in /appl/schdeve/source/grdstutransupdgpa.p
Removing File /appl/schdeve/source/grdstutransupdgpa.p
```

An example of compile errors are shown in this set of lines:

```
Compiling /appl/schdeve/source/rptenrollment.p
** "rtfreport.i" was not found. (293)
** /appl/schdeve/source/rptenrollment.p Could not understand line 113. (193)
** Unable to understand after -- "RUN". (247)
** Invalid statement. (254)
** /appl/schdeve/source/rptenrollment.p Could not understand line 322. (198)
** Unable to understand after -- "RUN". (247)
** Invalid statement. (254)
** /appl/schdeve/source/rptenrollment.p Could not understand line 373. (198)
** Unable to understand after -- "RUN". (247)
** Invalid statement. (254)
```

2 This is done in compile_file.p