Templates 101 - Concepts of template writing

Tom Moseley

President, The Moseley Group

Writing templates is hard work. Oh, sure, you can go in and muck about with the shipping templates, but are you sure you know what a change does? And what about writing your own? Do you know what happens when, and why?

In this discussion (it's not a lecture), we'll talk about some basic concepts you should to know to write templates.

It's a matter of time

There are five distinct time periods in the use-cycle of your template:

Population Time

When an instance of a template is encountered by the Application Generator, either by populating the template, or loading the application.

#GLOBALDATA, LOCALDATA, Screen Controls

Symbols have no values

Design Time

When your template user (the application writer) is mucking about in the application generator.

Can edit Controls

Generation Time

When the templates are doing their thing, generating code.

Gen'd Code

Compile Time

When the Clarion compiler is compiling the code your templates have written.

Run Time

When the program written with your templates is running.

Symbol Basics, part 1

Your templates don't use variables in the Clarion sense, but variables exist in the system.

Single Valued symbols vs. Multi-Valued symbols

Single valued symbols hold a single value, and operate like string (or long, or real) variables. Multi-valued symbols operate like a queue, with a twist, in that the symbol itself is also a variable.

Symbol (not simple) Dependencies

Just as files can have parent-child relationships, so can symbols. You can have any number of symbols, either single or multi-valued, dependent on a multi-valued "parent" symbol.

Getting a record of a multi-valued symbol, and when to use FIX and FIND

There are several different ways to get a symbol value, either through direct access to a record of the symbol, or by fixing the symbol to a value. A "mega-fix" command, called FIND, is also provided. Be wary of FIND, though. If misused, it can make your application generate in all sorts of interesting, and inappropriate, ways.

Symbol Basics, part 2

Now that you know what kind of symbols there are, it's time to look at how symbols are declared.

Built-in Symbols

These are symbols built in to the Clarion application generator.

User-defined symbols

These are symbols that your templates declare as they are processing.

Prompts

These are symbols, are effectively a hybrid of built-in and user-defined, whose value can be set by your template user at design-time.

EMBED Basics

The EMBED point is the workhorse of the template system. Without these, your templates wouldn't, couldn't work.

Calling All Code

The important thing to remember about an EMBED point is that it is entirely passive. The EMBED point doesn't need to be used at all, and there's no error checking to see if it's used. It's basically a broadcast mechanism, saying... **"I'm at this place in the code! Does anyone have anything for me?"** The code from your templates, and the template user's EMBED code, are written into the current piece of code, and then generation continues. What this means is...

There's NO error checking for EMBED names

As I said, EMBEDs are passive. If a misspelling occurs in an EMBED point, or in an AT accessing an EMBED point, the only clue you'll get is that your code won't generate.

Using #AT

This is the other half of the EMBED picture. An #AT section is the template section that answers the call of the wild EMBED.

If there's one piece of advice I could give you...

Small Steps

While the Application Generator is a solid piece of work, you should never assume that just because something should work, it does.
The best way to avoid confusion (and loads of lost work) is to take baby steps when writing your templates. In other words...

Write your prompts

Get the user interface portion of your template right, takes awhile

Design your controls

Figure out what controls you have and write the CONTROLS section of a control template.

Use ATSTART

ATSTART is the first AT section, processed when a procedure begins to generate. Capture all of the info you need here, then...

Use AT(%GatherSymbols)

This is the first EMBED in your generated code. Check to see if you've captured the right values in ATSTART.

Write the rest of the code in EMBED points in AppGen

And get it right before translation.

Transfer the code, one EMBED point at a time, to your template

Nuff said.