Dm25: Apps can create methods and controls

Creating at design time

We are used to creating new methods in the Class Browser with Methods | New Jade method and we are used to creating new controls, such as labels, on a form in the Painter.

Both these are design-time actions, not run-time actions.

Creating at run time

As we grow more expert, can we ask our programs—our applications—to create new methods and new controls as they run, responding to a changing situation or to a user's needs?

Yes we can. These are 'metaschema' actions.

Creating a method at run-time

The Process class has methods for compiling and executing source dynamically, as shown in the following example:[1]

```
vars                                /*  Example of a method to compile a method */
    src             : String;
    meth            : JadeMethod;
    err, pos, len   : Integer;
begin
    src :=          /* we supply the source code for a method */
"vars
  begin
    write 'Hello World';
  end;";
    meth := process.createTransientMethod( 'temp', class, currentSchema, src, true, null, err, pos, len );
    if err = null   then
        process.executeTransientMethod( meth, self );
    else
        write 'Compile error in: ' & CrLf & src;
    endif;
end;
```

This is a simple example, but it opens the way to a program constructing and executing a method of arbitrary complexity.

[1] Thanks to Geoff Davies

Creating a control at run-time

Suppose that we discover a need to create buttons, or labels (say) in response to the situations that the program finds itself in. Here is an example of creating labels at run time.

```
vars                                /*  Example of a method to add new controls */
    label       : Label;
begin
    create label transient;
    label.parent        := self;
    label.caption       := “Fresh label”;
    label.width         := getTextExtent( label.caption );
    label.top           := 50;
    label.left          := 150;
    addControl ( label );
end;
```

Creating the two together at run-time

Suppose that we could not compose the dynamic execution method until the need arose to create the control? Here is how to associate a new method with a new control:

First we use our create-a-method-dynamically skill to create a method on the form with a signature similar to a normal button_click method:

```
clickForNewButton (btn : Button input ) updating;

begin
    write “Hello World”;
end;
```

Then we map the desired _click behaviour for our new control onto that method, in this manner:

```
load ( ) updating;
 vars btn : Button;
begin
    create btn transient;
    addControl ( btn );
    btn.setEventMapping ( "click", "clickForNewButton");
end;
```