NEWTON Q&A: ASK THE LLAMA

Q *Here's something that's been puzzling me a bit: I want to pop up something like a copyright me for ten seconds when my application starts up. So I drew up a layout called Presents with a protoFloater containing all the necessary text. In my main layout is a link to Presents called presentsLink. The following is the viewShowScript for the topmost view in my application:*

```
func()
begin
   presentsLink:open();
   AddDelayedAction(presentsLink:close(), nil, 10000)
end
```

This says to me: open the linked view, wait ten seconds, and close it. And that's exactly what it does, except that after the view is closed, there's an exception. What am I doing wrong and how fix it?

A The short answer is that the second argument to the AddDelayedAction function is of the wrong type. This argument is supposed to be an array of parameters to be passed to the delayed functi and nil is not an array. The proper syntax for no arguments is [] instead of nil.

But there's more: Although the closure you supplied in the first argument works in this case, yo should get out of the habit of using that type of function call for delayed or deferred actions. Yo better off providing a full closure and sending in the view you want closed, as in this:

```
func()
begin
   // Define a closure to use in the delayed action.
   local myClose := func(whichView)
      whichView:Close();

   presentsLink:Open();
   AddDelayedAction(myClose, [presentsLink], 10000);
end
```

Unfortunately, things do not end there. You also need to make sure that the myClose function i internal RAM. The best way to do this is to use DefConst to define the function:

```
// In your ProjectData file:
DefConst('kDelayedClose, func(whichView) whichView:Close());
```

/
/

```
This changes the function above:
func()
begin
   presentsLink:Open();
   AddDelayedAction(EnsureInternal(kDelayedClose),
      [presentsLink], 10000);
end
```

However, there is an easier solution. Instead of doing a delayed action you can use the view idl mechanism to do what you want. All you need to do is add a viewIdleScript and viewIdleFrequency to the presentsLink top-level view. The viewIdleScript simply sends a Clos message. The viewIdleFrequency is set to the desired delay (10000 in this case).

The really short answer is that you can just use protoGlance. This proto has the show and disappear behavior built in. You can set the viewIdleFrequency to 10000 to get the behavior yo want.

Q *I have an alphabetically sorted list of items in a protoTable and I want to use protoa2z to quickly move through the table (like the cardfile overview). How do I do it?*

A The first thing you need to do is find the index in the protoTable of the correct item (that is, find right item in the def.tabValues array of the protoTable). How you do this will depend on what t of data you're representing.

Once you have the index, you can figure out how high each item in the protoTable is, set the VC slot of the table to the correct line, and force the protoTable to redraw. You probably want to m sure that you don't scroll off the bottom of the table. Below is a method you can add to your protoTable that will do what you want. You can then send the message from your a2zChanged method (make sure you send the message to the protoTable).

```
func(index)
begin
   // Figure out the height of an item in the table.
   local childHeight := if def.tabProtos.viewFont exists then
         fontHeight(def.tabProtos.viewFont)
      else
         fontHeight(viewFont);

   // Make sure that the table will not scroll off the end
   // by calculating the index of the bottommost item that
   // can be displayed.
   local largestIndex :=
```

```
ef.tabDown - ((:LocalBox().bottom - :LocalBox().top)
         DIV childHeight) - 1;

   // Use the bottommost item (largestIndex) to make sure
   // the table has no empty space on the bottom.
   vOrg := MIN(index, largestIndex);

   // Now force the table to redraw.
   :RedoChildren();
end
```

Q *What is your quest?*

A To answer the questions of those who develop for Newton.

Q *I would like to use the protoa2z sample code in my application, but I can't figure out how to se highlighted letter when the user scrolls through my data.*

A This is one of those "Oh, of course!" questions, at least once you realize that protoa2z is based protoPictIndexer. All you have to do is a SetValue of the currIndex slot to the correct index. Th will change the highlighting of the protoa2z.

Note that using SetValue will not call the IndexClickScript, but this is probably what you want you do want it to be called, you'll have to manually unhighlight the current selection, set the currIndex slot, and then highlight the new item. The appropriate code would be

```
a2z:Unhilite();
a2z.currIndex := newIndex;
a2z:Hiliter(newIndex);
```

Q *How do I create my own class of binary object?*

A To get a binary object of your own class, you first need to create a binary object, then change it class to your own. The easiest way to do this is to create a string that's the same length as your intended binary object and then change (coerce) the class of this new object to your own class. can use the string class as your basic binary object.

Suppose you wanted to have a binary class called CharID for an ID consisting of four ASCII characters. You could write a NewID function that would create the object and optionally initial it, like this:

```
// This will appear in your ProjectData file.
                                                                    /
                                                                    /
```

```
Define a constant for a default CharID object. This constant
// can be cloned at run time. kDefaultCharIDObj will be a CharID
// object with 4 bytes that are set to 0x00.
DefConst('kDefaultCharIDOBj, SetClass(SetLength("", 4), 'CharID));

// CharString is a string of 4 characters or nil.
NewCharID := func(CharString)
begin
   // Create a binary object of the correct length.
   local newObj := Clone(kDefaultCharIDObj);

   // Optionally initialize it.
   if CharString then
      for i := 0 to 3 do
         StuffChar(newObj, i, CharString[i]);

   // Return the new object.
   newObj ;
end;
```

To see this code in action, you can type it into the Inspector window in the Newton Toolkit and evaluate it. Note that you cannot use DefConst in the Inspector since it's a compile-time functio Just substitute the second argument in the DefConst function for kDefaultCharIDObj in the func to evaluate it. Then you can try things like this:

```
x := :NewCharID(nil);
#440DD01  <CharID, length 4>
ExtractChar(x, 2);
#6        $\00
x := NewCharID("abcd");
#4410FC9  <CharID, length 4>
ExtractChar(x, 2);
#636      $c
ExtractByte(x, 2);
#18C      99
```

Q *I tried to use a protoPictRadioButton in my application but the highlight rectangle isn't the right I know that I need to write some code to draw the correct-sized highlight, but where do I hook*

A Minimally you need to override the viewDrawScript of the protoPictRadioButton. You may als want to change the viewFormat since it defaults to a thick rounded rectangle border.

Assuming that you wanted some sort of rectangle highlight around the selected button, you cou use th follow

viewDrawScript:

```
func()
begin
   // If the button is selected, highlight it.
   if viewValue then
   begin
      // Get the bounds of the protoPictRadioButton.
      local b := :LocalBox();

      // Inset the bounds.
      b.top := b.top + 2;
      b.left := b.left + 2;
      b.bottom := b.bottom - 2;
      b.right := b.right - 2;

      // Now draw a rectangle.
      :DrawShape(MakeRect(b.left, b.top, b.right, b.bottom), nil);
   end;
end
```

Q *What is your favorite color?*

A Llama fur beige.

Q *In my application I have a clPictureView that can display a variable number of pictures. Right n create a bunch of picture slots in my application and then make another slot at run time that is ar of those items. There must be an easier way.*

A You're right; there is an easier way. You can use the GetPictAsBits function in your ProjectDat file to read in the bitmaps. Note that you'll first have to open the resource file that contains the pictures.

```
// HOME is defined to be the project directory pathname
// by NTK; use that in opening the resource file.
r := OpenResFileX(HOME & "Pictures");

// Get an array of pictures.
myPictures := [
   GetPictAsBits("TARDIS", nil),
   GetPictAsBits("Planet", nil)
];
```

```
 Now close the resource file.
CloseResFileX(r);
```

Once you have the myPictures array, you can create a slot of type Evaluate in your application a just type “myPictures” in the editor for the slot. Then you can use SetValue to set the icon slot c the clPicture view to one of the elements of the array.

Q *What is the AutoClose checkbox in the Newton Toolkit for? Why should I use it?*

A The AutoClose flag causes all other AutoClose applications to be closed when your application clicked. The effect is that only one “auto-close” application can be open at one time. You should *always* make your application auto-close, to help conserve memory and other resources, unless providing special functionality to other applications (like the built-in Calculator or Styles application).

Q *How do I put my own default person in the fax information slip?*

A You can set up a default person in your SetupRoutingSlip method, which is called before the fa slip is shown. The argument to that method is used to set up the particular routing slip. In the ca of a fax slip, there’s a slot called “alternatives” which is an array of cardfile entries for the possi people to fax to. If there’s just one entry, that’s the person. The fax number will be set from the cardfile entry. So your SetupRoutingSlip method would look like this:

```
func(fields)
begin
   // Check for a fax.
   if fields.category = 'faxSlip then
      fields.alternatives := SmartCFQuery("Llama");

   // Do other stuff like put a title for the out box.
   . . .
end
```

Note that the fields.category is set to the same value you set in the routeSlip slot of a frame in y application’s entry in the global routing frame. The SmartCFQuery function returns an array of cardfile entries that have strings starting with the string passed in.

Q *I noticed that every soup entry has a _uniqueID slot. Just how unique is it?*

A The _uniqueID slot is only guaranteed to be unique within that soup on that particular store. The will never be reused in that soup on that store. However, it’s not necessarily unique across stor (say, RAM and a PCMCIA card).

ground velocity of an unladen llama climbing a five percent grade?

A Mexican or Venezuelan?

The llama is
the unofficial mascot of the Developer Technical Support group in Apple's Personal Interactive Electronics (PIE) division. Send your Newton-related questions to the PIE llama at llama@blammo.apple.com on the Internet. The first time we use a question from you, we'll send you a T-shirt.

Thanks to
Glen Raphael and our PIE Partners for the questions used in this column, and to
jXopher, Todd Courtois, Bob Ebert, Mike Engber, Kent Sandvik, Maurice Sharp, and Scott ("Zz") Zimmerman for the answers.

Have more questions?
Need more answers? Take a look at PIE Developer Info on AppleLink.