Subject: Re: Smart way to extract the same attribute from a list of objects?
Posted by PMan on Thu, 04 Oct 2012 16:53:54 GMT
View Forum Message <> Reply to Message

Well, I will share with you what I did. Not sure about if it qualitfies as better/faster/more-elegant, but here you go.

I created a new object that is just a list:

```
pro newObj__define
  compile_opt idl2

  _ = {newObj, $
    inherits list $
    }
```

Then, in my init, I created a struct like

```
function newObj::init, nSamples = nSamples
  compile_opt idl2

  newObjStruct = {newObjStruct, $
    radiance:0, $
    channel:0}

  self->add, replicate(newObjStruct, nSamples), /extract

  return, 1
end
```

Then I overloaded the setProperty and getProperty functions.

```
pro newObj::setProperty, _extra=extra
  compile_opt idl2

  if self->isEmpty() then return

  if (~n_elements(extra)) then return

  num = n_elements((tags=tag_names(extra)))

  lArray = self->toArray()
  self->remove, /all

  for ii=0, num-1 do begin
    ok = execute("lArray."+tags[ii]+ " = extra.(ii)")
  endfor
```

```
  self.add, lArray, /extract
end

pro newObj::getProperty, _ref_extra=extra
  compile_opt idl2

  if self->isEmpty() then return

  if (~n_elements(extra)) then return

  lArray = self.toArray()

  for ii=0, n_elements(extra)-1 do begin
    ok = execute("(scope_varfetch(extra[ii], /ref_extra)) = lArray."+extra[ii])
  endfor

end
```

Subject: Re: Smart way to extract the same attribute from a list of objects?
Posted by Paul Van Delst[1] on Thu, 04 Oct 2012 18:11:08 GMT
View Forum Message <> Reply to Message

Hello,

Thanks. I will have a closer look at your (and Mark's) technique.

In the meantime, what I came up with for my particular object is down
below. I reopened the List class to do it; a faux pas I know but...

```
-----%<-----
FUNCTION List::Get_Channel_Data, $
   data_name    , $ ; Input
   Debug = debug    ; Input keyword

   ; Set up
   @rtsolution_func_err_handler
   @rtsolution_parameters
   ; ...Check that *all* list members are RTSolution objects
   FOREACH element, self DO BEGIN
     IF ( ~ ISA(element,'RTSolution') ) THEN $
       MESSAGE, 'Non-RTSolution object found in channel list.', $
                NONAME=MsgSwitch, NOPRINT=MsgSwitch
   ENDFOREACH

   ; Construct the data retrieval command
   cmd_string = "self[i]->RTSolution::Get_Property, "+data_name+"=x"
```

```
  ; Define the return array
  n_channels = self.Count()
  data = MAKE_ARRAY(n_channels, VALUE = ZERO)

  ; Loop over list elements
  FOR i = 0L, n_channels-1 DO BEGIN
    ; ...Execute the command
    result = EXECUTE(cmd_string)
    IF ( result NE TRUE ) THEN $
      MESSAGE, "Error retrieving " + data_name + $
               " data for channel index " + STRTRIM(i,2), $
               NONAME=MsgSwitch, NOPRINT=MsgSwitch
    ; ...Save the channel data
    data[i] = x
  ENDFOR


  ; Done
  RETURN, data

END
-----%<-----
```

So once I've read my list-of-lists in

```
IDL> rtsolution_readfile, 'iasi616_metop-a.RTSolution.bin'
% RTSOLUTION_READFILE: Reading RTSolution profile #1
% RTSOLUTION_READFILE: Reading RTSolution profile #2
% RTSOLUTION_READFILE: Number of profiles/channels read from
iasi616_metop-a.RTSolution.bin : 2/616
```

I have my list of atmospheric profile results (the "profile" list),

```
IDL> help, rts
RTS             LIST  <ID=1  NELEMENTS=2>
```

each of which contains sensor spectral results (a "channel" list for each "profile" entry),

```
IDL> help, rts[0]
<Expression>    LIST  <ID=2  NELEMENTS=616>
```

each of which contains the main object

```
IDL> help, (rts[0])[0]
<Expression>    OBJREF    = <ObjHeapVar3(RTSOLUTION)>
```

And now I can extract the same data from each object in the "channel" list:

```
IDL> help, rts[0].get_channel_data('radiance')
% Compiled module: LIST::GET_CHANNEL_DATA.
<Expression>    DOUBLE    = Array[616]
IDL> help, rts[1].get_channel_data('surface_emissivity')
<Expression>    DOUBLE    = Array[616]
```

etc..

Uff da. I really need to do more IDL metaprogramming.... :o)

cheers,

paulv

On 10/04/12 12:53, Paul Mallas wrote:
> Well, I will share with you what I did. Not sure about if it qualitfies as better/faster/more-elegant, but here you go.
>
> I created a new object that is just a list:
>
```
> pro newObj__define
>   compile_opt idl2
>
>   _ = {newObj, $
>     inherits list $
>     }
```
>
> Then, in my init, I created a struct like
>
```
> function newObj::init, nSamples = nSamples
>   compile_opt idl2
>
>   newObjStruct = {newObjStruct, $
>     radiance:0, $
>     channel:0}
>
>   self->add, replicate(newObjStruct, nSamples), /extract
>
>   return, 1
> end
```
>
> Then I overloaded the setProperty and getProperty functions.
>
```
> pro newObj::setProperty, _extra=extra
```

```
>    compile_opt idl2
>
>    if self->isEmpty() then return
>
>    if (~n_elements(extra)) then return
>
>    num = n_elements((tags=tag_names(extra)))
>
>    lArray = self->toArray()
>    self->remove, /all
>
>    for ii=0, num-1 do begin
>      ok = execute("lArray."+tags[ii]+ " = extra.(ii)")
>    endfor
>
>    self.add, lArray, /extract
> end
>
> pro newObj::getProperty, _ref_extra=extra
>    compile_opt idl2
>
>    if self->isEmpty() then return
>
>    if (~n_elements(extra)) then return
>
>    lArray = self.toArray()
>
>    for ii=0, n_elements(extra)-1 do begin
>      ok = execute("(scope_varfetch(extra[ii], /ref_extra)) = lArray."+extra[ii])
>    endfor
>
> end
```