Subject: Treating an object as a structure
Posted by natha on Wed, 04 Mar 2009 20:20:46 GMT
View Forum Message <> Reply to Message

If I've a struct I can do something like that:

```
a={s:'string', l: 0l}
print, n_tags(a)
print, tag_names(a)
help, a, /str
etc.
```

But if I've an object I can't do the same... Is it possible to parse
all the attributes inside the object using something similar ?
I mean, I don't want to use the GetProperty method, I only want to
know which attributes are stored in the definition of the object.

```
pro object__define
   str ={ object, $
          attribute_s: '', $
          attribute_l: 0l  }
end
```

The object saves a struct. It could be possible to take information
about this without the use of the getproperty method.

thanks

Subject: Re: Treating an object as a structure
Posted by David Fanning on Wed, 04 Mar 2009 23:54:10 GMT
View Forum Message <> Reply to Message

Michael Galloy writes:

> Just use STRUCT_ASSIGN:

Well, shoot, I knew that! How come I couldn't remember it? :-(

Probably because I didn't write it down on my web page,
my (almost) guaranteed way of remembering these sorts of
things. Look for an article soon. ;-)

Cheers,

David
--
David Fanning, Ph.D.

Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Treating an object as a structure
Posted by natha on Thu, 05 Mar 2009 00:30:25 GMT
View Forum Message <> Reply to Message

Cool !

Lately, we've learned a lot with my questions.

llo

Subject: Re: Treating an object as a structure
Posted by natha on Thu, 05 Mar 2009 00:35:25 GMT
View Forum Message <> Reply to Message

Remember that STRUCT_ASSIGN is so useful too to clone object inside
the Init method

Subject: Re: Treating an object as a structure
Posted by David Fanning on Thu, 05 Mar 2009 00:52:21 GMT
View Forum Message <> Reply to Message

llo writes:

> Lately, we've learned a lot with my questions.

Yes, but be careful. Karma is a funny thing, and
if you learn too much you will be made to
listen to other people ask clever questions that
you can only hope to answer. ;-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Treating an object as a structure
Posted by Michael Galloy on Thu, 05 Mar 2009 16:01:27 GMT
View Forum Message <> Reply to Message

llo wrote:
> On Mar 4, 4:59 pm, David Fanning <n...@dfanning.com> wrote:
>> llo writes:
>>> It works but the problem is that the struct is not the object itself.
>> No, that's pretty much the point of objects. :-)
>
> That's not true David. You can get all the object attributes doing
> this:
>
> a=OBJ_NEW('object')
> str=CREATE_STRUCT(NAME=OBJ_CLASS(a))
> STRUCT_ASSIGN, a, str
>
> Now, ypu have all the attributes of a in the struct str.
>
> Cheers,
>
> Bernat
>
>

Not so fast! That trick only works inside an object's methods:

```
IDL> a=OBJ_NEW('idlgrmodel')
IDL> str=CREATE_STRUCT(NAME=OBJ_CLASS(a))
IDL> STRUCT_ASSIGN, a, str
% STRUCT_ASSIGN: Object instance data is not visible outside class
methods: A
% Execution halted at: $MAIN$
```

Mike
--
www.michaelgalloy.com
Associate Research Scientist
Tech-X Corporation

Subject: Re: Treating an object as a structure
Posted by natha on Thu, 05 Mar 2009 16:05:47 GMT
View Forum Message <> Reply to Message

Ups

Sorry ! I only tried inside a class method. thanks for you comment !

Subject: Re: Treating an object as a structure
Posted by JDS on Fri, 06 Mar 2009 22:36:13 GMT
View Forum Message <> Reply to Message

I use this technique a lot:

```
 snapshot=create_struct(NAME=obj_class(self))
```

or simply

```
 snapshot={MYCLASS}
 struct_assign,self,snapshot
```

later apply snapshot:

```
 struct_assign,snapshot,self
```

This allows me to make snapshots of object data for the purpose of undoing changes, or examining how it was at a certain point in time. Keep a pointer to a list of these, and you have multiple undo. Add a text phrase when snapshotting ("Frobnoid change") and you can advertise "Undo Forbnoid change" and so on.

The lack of garbage collection starts getting painful at this point though (since older versions could refer to data newer versions have deleted).

Remember that snapshots of heap pointers (at whatever level in the heirarchy) are not immutable. If you want a deep copy, you can use the save/restore trick, first calling the new HEAP_NOSAVE routine on items you won't need a deep copy of. Or you can simply replicate certain heap data by hand (new=ptr_new(*old)). This reminds me that we're still lacking deep copy capability without hitting the disk.

JD