Subject: HASH with case insensitive keys versus a new DICT class
Posted by chris_torrence@NOSPAM on Tue, 11 Dec 2012 20:14:56 GMT
View Forum Message <> Reply to Message

Hi all,

I've been toying around with adding support for the "." operator to HASH. So, for example, you could do the following:

```
h = HASH()
h.field1 = "my data"  ; adds the key "FIELD1" with data "my data"
print, h.field1
```

So, in essence, this would be a "dynamic" structure.

The only problem with this approach is that the keys become all uppercase, because of the case insensitivity of IDL variables. Also, if you add keys during initialization (or using the square brackets), then they would need to be valid IDL variable names, otherwise you couldn't access them using the ".". So no spaces or special characters.

I see three possible solutions:

1. Add a keyword to HASH() that forces it into "valid IDL variable" mode. Keys can only be strings. The keys could all be stored as the user provided them, but internally the actual hash would be done with uppercase versions of the keys. The Hash could throw errors if the key wasn't a valid IDL variable name.
Advantage: still uses the HASH interface, the case of keys can be preserved & returned to the user
Disadvantage: confusing - you could have 2 hashes in your program that behave differently, depending upon a creation keyword that you might not even know was set.

2. Change the HASH behavior, so if a key is a string, then internally it constructs its hash using an "IDL_Validname()" version of the key. Again, we would store the original keys, so they could be returned intact. Numeric keys would be unchanged.
Advantage: No weird keyword to have to explain - just a single hash class.
Disadvantage: Backwards compatibility issues - could no longer have 2 keys that differed only in their case. Would need to explain that if you want to use "." then you have to be "careful" with your key names.

3. Add a new "DICT" class, that behaves differently than HASH. Keys can only be strings. Keys could still be stored (and returned) with mixed case, errors are thrown if a key isn't a valid IDL variable name. Internally, the actual hash would be done with uppercase versions of the keys.
Advantage: No backwards compatibility issues. Documentation is very clear.
Disadvantage: Yet another class.

Just to reiterate, in all 3 cases, the original "mixed" case keys would be stored, and could be returned with the hash.Keys() method.

Thoughts? Is anyone storing 2 keys that differ only in their case? Are you using HASH, and if so, for what purpose?

Thanks!
-Chris
ExelisVIS

Subject: Re: HASH with case insensitive keys versus a new DICT class
Posted by Craig Markwardt on Thu, 13 Dec 2012 04:37:02 GMT
View Forum Message <> Reply to Message

On Wednesday, December 12, 2012 10:10:51 PM UTC-5, Chris Torrence wrote:
> On Wednesday, December 12, 2012 12:09:34 PM UTC-7, Craig Markwardt wrote:
>
>> On Wednesday, December 12, 2012 1:05:08 AM UTC-5, Chris Torrence wrote:
>
>>
>
>>> Okay, I'll bite.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I have a specific use case for this, where using a "dynamic" structure would be very helpful. Users have also been asking for dynamic, extendable structures for years.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>

>>>
>
>>
>
>>> One more data point: It would take less than a day to make this update. Adding a new widget system would take more than a day.
>
>>
>
>>
>
>>
>
>> I would go at it from the other end and make the structure access operator "." overloadable from within the IDL language. Once that key language feature is added, then your particular use case of HASH as the backing store can be wired up with a few lines of IDL code, and you can choose whatever idiosyncracies you wish.
>
>>
>
>>
>
>>
>
>> Craig
>
>
>
> Hi Craig,
>
>
>
> Actually, it is already overloadable. As long as your class inherits from IDL_Object, then using a "." will call GetProperty (if it is on the right-hand-side of the equals), or SetProperty (if it is on the left-hand-side).
>
>
>
> And you're right, it was indeed just a couple of lines of code. The trickiness comes in because the IDL parser has already made the "field" name uppercase by the time it reaches the internal C structure code. I really don't want to mess with the IDL parser.

OK, then if I were you I would implement an *alternate* case-insensitive hash algorithm when you initialize a new hash, such that h['MYSTRING'] returns the same as h['mystring'], as h['MyString']. Turn that flag on for your special class, and you're done. My point being, separate the "IDL variable"ness from the case-insensitiveness.

Craig

Subject: Re: HASH with case insensitive keys versus a new DICT class
Posted by David Fanning on Fri, 14 Dec 2012 19:51:41 GMT
View Forum Message <> Reply to Message

Chris Torrence writes:

> Well, bad news David. I spent a day looking at this, and it turns out to be a Microsoft Windows vendor bug. Sorry.

Weird! Obviously, I use this dialog all day, every day,
with every application I work with. IDL is the only
one I've ever noticed clipping the file name. But,
now you mention it, when I OPEN a file with this
dialog, none of my applications gives me the option
to specify a "starting" file name. They all open
with the filename blank.

OK, well, thanks. I guess I'll just stick to the
work-around I use in cgPickfile. :-)

Cheers,

David

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

Subject: Re: HASH with case insensitive keys versus a new DICT class
Posted by wlandsman on Fri, 14 Dec 2012 20:09:06 GMT
View Forum Message <> Reply to Message

But shouldn't it be possible for Exelis to use the same work-around in Dialog_Pickfile that David uses in CgPickfile?, i.e. the full filename does appear correctly when writing an output file name. Or are there other issues involved?

On Friday, December 14, 2012 2:51:41 PM UTC-5, David Fanning wrote:
> Chris Torrence writes:
>

>
> OK, well, thanks. I guess I'll just stick to the
>

> work-around I use in cgPickfile. :-)
>
>
>
> Cheers,
>
>
>
> David
>
>
>
>
>
>
>
>
> --
>
> David Fanning, Ph.D.
>
> Fanning Software Consulting, Inc.
>
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>
> Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: HASH with case insensitive keys versus a new DICT class
Posted by Mark Piper on Fri, 14 Dec 2012 21:46:03 GMT
View Forum Message <> Reply to Message

On Friday, December 14, 2012 1:09:06 PM UTC-7, wlandsman wrote:
> But shouldn't it be possible for Exelis to use the same work-around in Dialog_Pickfile that David uses in CgPickfile?, i.e. the full filename does appear correctly when writing an output file name. Or are there other issues involved?
>

When prompting to open (not save) a file, the button on the dialog says "Save."

mp

Subject: Re: HASH with case insensitive keys versus a new DICT class
Posted by chris_torrence@NOSPAM on Sat, 15 Dec 2012 02:19:34 GMT
View Forum Message <> Reply to Message

On Friday, December 14, 2012 2:46:03 PM UTC-7, Mark Piper wrote:

> On Friday, December 14, 2012 1:09:06 PM UTC-7, wlandsman wrote:
>
>> But shouldn't it be possible for Exelis to use the same work-around in Dialog_Pickfile that David uses in CgPickfile?, i.e. the full filename does appear correctly when writing an output file name. Or are there other issues involved?
>
>>
>
>
>
> When prompting to open (not save) a file, the button on the dialog says "Save."
>
>
>
> mp

Mark's correct. They are completely different Windows controls. You can't change the text on the Save button or on the "Save filter".

The weird thing is that if you click on a different application, and then back on the dialog, it will then display the entire filename.

Luckily, it is usually rare for a GUI application to need to supply an initial filename on an "Open". It's almost always on "Save" that you supply a filename.

That's probably why MS hasn't fixed this bug.

-Chris