Subject: Case Insensitive Hash but still preserve cases of original keys
Posted by SonicKenking on Mon, 14 Jul 2014 06:24:08 GMT
View Forum Message <> Reply to Message

Hello Community,

I tried to implement a special Hash object with following features
1. Keys are of String type
2. Keys should be case insensitive
3. The original cases of keys should be preserved and reported when the keys() method is called.

Let's say it is called SpeicalHash. It should support following operations:

```
h = SpecialHash()
h['X'] = 42
print, h.keys() ; output 'X'
print, h['x'], h['X'] ; output 42, 42
print, h.haskey('x'), h.haskey('X')  ; output 1, 1

h['x'] = 1337
print, h['x'], h['X'] ; output 1337, 1337
```

I came up with an implementation (attached at the end of this post) and it meets all of the above requirements. However, when I tried to print the variable h, it reports error of "Key does not exist"

```
print, h ; Output % Key does not exist: "X"
```

I found out the PRINT command calls Hash::_overloadPrint to do the job and it in turn calls Keys() method to get the keys. I guess that is where the key "X" comes in. However, I don't understand how PRINT gets the value of a key because none of the overload Bracket methods are called (the cases of keys are taken care of in SpecialHash's bracket methods). I can only think of two possibilities:

1. The Hash::_overloadPrint method calls Hash's own bracket methods to get the value of a key. Therefore the SpecialHash's bracket methods did not get called. If this is true, it seems to be a bug in IDL's Hash implementation and should be fixed.

2. The Hash::_overloadPrint method calls some other hidden method to get the value of a key. If this is the case, can someone show me how it is done?

Thanks!

Yang

```
; ============== SpecialHash Implementation Starts Here ==============


function SpecialHash::keys
```

```
    return, list(self.keylist, /extract)
end

function SpecialHash::hasKey, _keys
    keys = strlowcase(_keys)
    return, self->Hash::hasKey(keys)
end

function SpecialHash::_overloadBracketsRightSide, isRange, $
    sub1, sub2, sub3, sub4, sub5, sub6, sub7, sub8

    sub1 = strlowcase(sub1)
    return, self->hash::_overloadBracketsRightSide(isRange, sub1, sub2, sub3, sub4, sub5, sub6,
sub7, sub8)

end

pro SpecialHash::_overloadBracketsLeftSide, objref, value, isrange, $
    _sub1, sub2, sub3, sub4, sub5, sub6, sub7, sub8

    sub1 = strlowcase(_sub1)
    self->Hash::_overloadBracketsLeftSide, objref, value, isrange, sub1, sub2, sub3, sub4, sub5,
sub6, sub7, sub8
    if (self->Hash::keys()).count() eq self.keylist.count() + 1 then self.keylist.add, _sub1

end

function SpecialHash::init
    if ~self->Hash::init() then return, 0
    self.keylist = list()
    return, 1
end

pro SpecialHash__define
    class = {SpecialHash, inherits hash, $
        keylist: list() }
end


; =============== End of Code ==============
```