Module

ObjectSpace

The ObjectSpace module contains a number of routines that interact with the garbage collection facility and allow you to traverse all living objects with an iterator.

ObjectSpace also provides support for object finalizers. These are procs that will be called when a specific object is about to be destroyed by garbage collection.

```
include ObjectSpace
a, b, c = "A", "B", "C"
puts "a's id is #{a.object_id}"
puts "b's id is #{b.object_id}"
puts "c's id is #{c.object_id}"
define_finalizer(a, lambda {|id| puts "Finalizer one on #{id}" })
define_finalizer(b, lambda {|id| puts "Finalizer two on #{id}" })
define_finalizer(c, lambda {|id| puts "Finalizer three on #{id}" })
```

produces:

```
a's id is 333080
b's id is 332950
c's id is 332880
Finalizer three on 332880
Finalizer two on 332950
Finalizer one on 333080
```

Module methods

_id2ref

ObjectSpace._id2ref(*object_id*) → *obj*

Converts an object ID to a reference to the object. May not be called on an object ID passed as a parameter to a finalizer.

```
s = "I am a string"           # =>   "I am a string"
oid = s.object_id             # =>   335730
r = ObjectSpace._id2ref(oid)  # =>   "I am a string"
r                             # =>   "I am a string"
r.equal?(s)                   # =>   true
```

count_objects

ObjectSpace.count_objects → *histogram_hash*

1.9

Returns a hash where the keys are the interpreter-specific internal object types and the values are the number of objects of each type.

```
ObjectSpace.count_objects   # =>   {:TOTAL=>10639, :FREE=>412,
                                   :T_OBJECT=>7, :T_CLASS=>406,
                                   :T_MODULE=>18, :T_FLOAT=>5,
                                   :T_STRING=>2357, :T_REGEXP=>11,
                                   :T_ARRAY=>304, :T_HASH=>9,
                                   :T_STRUCT=>32, :T_BIGNUM=>2,
                                   :T_FILE=>4, :T_DATA=>153, :T_MATCH=>1,
                                   :T_COMPLEX=>1, :T_NODE=>6898,
                                   :T_ICLASS=>19}
```

define_finalizer ObjectSpace.define_finalizer(*obj*, *a_proc*=proc())

Adds *a_proc* as a finalizer, called when *obj* is about to be destroyed. Note that if you use lambda to create the proc object, you must remember to include a parameter wih the block. If you don't, the invocation of the lambda will silently fail when the finalizer is called because of a mismatch in the expected and actual parameter count.

each_object ObjectSpace.each_object(⟨ *class_or_mod* ⟩) {| *obj* | *block* } → *fixnum*

Calls the block once for each living, nonimmediate object in this Ruby process. If *class_or_mod* is specified, calls the block for only those classes or modules that match (or are a subclass of) *class_or_mod*. Returns the number of objects found. Immediate objects (Fixnums, Symbols true, false, and nil) are never returned. In the following example, each_object returns both the numbers we defined and several constants defined in the Math module:

```
a = 102.7
b = 95       # Fixnum: won't be returned
c = 12345678987654321
count = ObjectSpace.each_object(Numeric) {|x| p x }
puts "Total count: #{count}"
```

produces:

```
12345678987654321
102.7
(0+1i)
2.71828182845905
3.14159265358979
9223372036854775807
2930980666147050870094753580906653300 6
2.22044604925031e-16
1.79769313486232e+308
2.2250738585072e-308
Total count: 10
```

garbage_collect ObjectSpace.garbage_collect → nil

Initiates garbage collection (see module GC on page 532).

undefine_finalizer ObjectSpace.undefine_finalizer(*obj*)

Removes all finalizers for *obj*.