Inside the Ruby Object Model

pluskid@freecity
An overview of how **objects** are represented in memory
Contrast

<table>
<thead>
<tr>
<th></th>
<th>Static Language (C++)</th>
<th>Dynamic Language (Ruby)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
<td>Cells with variable size</td>
<td>Cells with fixed size</td>
</tr>
<tr>
<td></td>
<td>Can hold pointers to heap objects</td>
<td>Can hold pointers to heap objects</td>
</tr>
<tr>
<td></td>
<td>Can hold any kind of objects</td>
<td>Can only hold special embedded objects*</td>
</tr>
<tr>
<td>Heap</td>
<td>Cells with variable size</td>
<td>Cells with fixed size</td>
</tr>
<tr>
<td></td>
<td>Can hold any kind of objects</td>
<td>Can only hold non-embedded objects</td>
</tr>
</tbody>
</table>

* Embedded objects include: nil, true, undef, false, Fixnum, Symbol
<table>
<thead>
<tr>
<th>Stack</th>
<th></th>
<th>Heap</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static Language (C++)</strong></td>
<td><strong>Cells with variable size</strong></td>
<td><strong>Cells with variable size</strong></td>
<td><strong>Cells with fixed size</strong></td>
</tr>
<tr>
<td></td>
<td>Can hold pointers to</td>
<td></td>
<td>Can hold any kind of objects</td>
</tr>
<tr>
<td></td>
<td>heap objects</td>
<td>Can only hold special embedded objects*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Can hold any kind of objects</td>
<td></td>
<td>Can only hold non-embedded objects</td>
</tr>
<tr>
<td><strong>Dynamic Language (Ruby)</strong></td>
<td><strong>Cells with fixed size</strong></td>
<td><strong>Cells with fixed size</strong></td>
<td></td>
</tr>
</tbody>
</table>

* Embedded objects include: nil, true, undef, false, Fixnum, Symbol
It will be clear later.
3 Requirements of an Object

1. Being able to differentiate itself from the rest (having an identity)
2. Keeping an internal state (instance variables)
3. Being able to reply to requests (methods)
Duck Typing

If It Walks Like a Duck and Quacks Like a Duck, I Would Call It a Duck.
typedef unsigned long VALUE;
struct RObject {
    struct RBasic basic;
    struct st_table *iv_tbl;
};

struct RBasic {
    unsigned long flags;
    VALUE klass;
};
struct RClass {
    struct RBasic basic;
    struct st_table *iv_tbl;
    struct st_table *m_tbl;
    VALUE super;
};
• Class is object.
• Everything is object.
Basic OO Concepts (from Java)

• Inheritance
  – Single inheritance + Singleton Class + Mixin

• Encapsulation
  – Object level access permission

• Polymorphism
  – Polymorphism? Duck typing!
Singleton Class

• Give an object **Unique** behavior
Why Singleton class?

• All classes are instances of the class **Class**
  – Each class would have its own methods – class methods (a.k.a static methods)

```ruby
class Foo
  def Foo.foo  # or def self.foo
    # ...
  end
end

class Bar
end

Foo.foo  # => OK
Bar.foo  # => NoMethodError
```
How is it implemented?

Before

After

Object

Class

Object
class

Class

super class

Singleton Class
Mixin

module Cool
  def be_awesome!
    # ...
  end
end

class Saber
  include Cool
end

class Ninja
  include Cool
end

a = Ninja.new
b = Saber.new
a.be_awesome!
b.be_awesome!
How is it implemented?

Before

- Object
  - super class
    - Ninja

After

- Object
  - super class
    - include class
      - super class
      - mtbl
  - Ninja
  - method table
    - be_awesome!
    - ...

Cool

- mtbl
class Monkey
  include Cool
end

Monkey.new.be_awesome!
class Numeric
  def days
    o = Object.new
    o.instance_variable_set("@time", Time.now)
    o.instance_variable_set("@diff", self * 3600 * 24)
  end

  def o.ago
    @time - @diff
  end

  def o.from_now
    @time + @diff
  end

  return o
end
end

Time.now    # => Tue May 06 20:58:12 +0800 2008
5.days.ago  # => Thu May 01 20:58:08 +0800 2008
5.days.from_now  # => Sun May 11 20:58:17 +0800 2008
method (way) missing
irb(main):001:0> foo.be_awesome!
NoMethodError: undefined method `be_awesome!' for #<Foo:0x525cbb4>
from (irb):1
class Article
  def self.method_missing(method, *args)
    if method.to_s =~ /^find_by_(\w+)/$ and args.size == 1
      puts "SELECT * FROM articles WHERE #{\$1} LIKE '#{args.first}"
    elsif method == :be_awesome!
      puts "Awesome!"
    else
      super
    end
  end
end

Article.find_by_title("foo")
# => SELECT * FROM articles WHERE title LIKE 'foo'
Article.find_by_content("ruby")
# => SELECT * FROM articles WHERE content LIKE 'ruby'
Article.be_awesome!
# => Awesome!
Article.be_cool!
# => NoMethodError: undefined method `be_cool!' for Article:Class
thank_you!

NoMethodError: undefined method `thank_you!' for main:Object