Ruby简明入门和提高

作者chenge （chenge3k at qq.com）

目录

第一章 简介

欢迎开启有趣的Ruby之旅!

Ruby语言的作者Matz在《代码之美》一书中讲述了他的设计思想，就是简洁、保守、简单、灵活性，以及平衡性等五大原则。

```
puts 'hello world'
```

例子中那一行代码，可以说明简洁、保守（puts继承自C语言）和简单。灵活性和平衡性需要在更复杂的代码中可以体会到。

学习计划

针对初次编程者和初次学习Ruby的程序员，建议你分两步，先学习初初级部分，有兴趣继续的话学中级。中级部分需要一本参考书,可自选书籍或者参考这个电子版基础教程。

其他参考资料： Ruby中文官网,欢迎推荐更好的。

实验

可以复制代码，去“rubyfiddle”在线运行

或者参考wiki自行安装Ruby。

学习社区

http://exercism.io 是很好的英文的互动编程社区。

第二章 初级：类和对象

分类是符合人的思维的。比如人类，每个人就是人类的一个实例，约定都翻译成对象。

```
class Person
  def initialize(name)
     @name = name
  end

  def show_name
    puts @name
  end
end

zhao = Person.new 'zhao yun '
qian = Person.new 'qian jin '

zhao.show_name
qian.show_name
```

简单解释一下，class是定义类Person，def定义了方法。@name是实例变量的写法。 new是生成对象，两个new就生成了两个对象，会调用initialize这个方法，name是参数。最后两句是调用方法show_name, puts的意思就是显示一段文字。

如果你第一次编程，也许你对这些术语还有些陌生，不要紧，多看几次，慢慢就会熟悉起来了。

第三章 块

块是ruby的特色。

```
  3.times do
    puts 'hello world'
  end

  people = ['zhao', 'qian']

  people.each do |x|
    puts x
  end
```

以上就是两种块的写法，第一个是无参数，后一个带参数x， []是数组。块可以看成独立的函数，与块前面的方法协同工作，就像二人转。

第四章 模块

模块也是Ruby的特色。

```
module Show
  def show_msg
    puts self.class
  end
  Pi = 3.14
end

class Person
  include Show
end

class Desk
  include Show
end

Pi = 2
puts Show::Pi #3.14

Person.new.show_msg #Person
Desk.new.show_msg   #Desk
```

主要有两个作用，一个是作为命名空间，避免名字冲突，比如例子中的Pi。另一个是共享代码，例子中Person和Desk共享Show的代码。

第五章 中级：单元测试

> *补充说明一下：如果你不是很有经验的话，看这个例子会有困难。可以结合这个例子和简介里提到的基础教程来学习。*

这个例子不能在线运行，需要在本地测试。

关系示意图如下

－－incoming-－》测试对象－－－outgoing－－－》依赖对象

测试对象和依赖对象，依赖对象一般用mock来写。 incoming测试状态，outgoing command测试行为。outgoing query不用测试。 参见代码，test_msg方法先是测试了outgoing的行为，用的是mock和verify。后面测试incoming的状态。

```
require 'minitest'
require 'minitest/autorun'

class TestMan <  MiniTest::Test
    def test_msg
        manager = MiniTest::Mock.new
        manager.expect(:get_name!,  'john',[1])
        man = Man.new(1, manager)
        manager.verify

        msg = man.msg
        assert_equal msg, 'hi john'
    end

    def test_manager_name
        manager = Manager.new
        assert_equal manager.get_name!(01), 'john'
    end
end

class Man
    def initialize(i,manager)
        @name = manager.get_name! i
    end
    def msg
        "hi #{@name}"
    end
end

class Manager
    def get_name!(i)
        @checked ||= []
        @checked[i] = true

        ['joe', 'john'][i]

    end
end
```

第六章 高级：元编程和装饰器

那天看了coolshell陈皓先生分享的一关于python装饰器的文章，我在ruby-china发起了一个讨论。这个例子来自这次讨论，征得原作者piecehealth先生的同意发布在这里。

《Ruby元编程》这本书里讲的一些窍门，这个例子用了好几个。send method, define method, alias chain, class_eval。如果看起来吃 力的话，可以去看那本书。需要在本地测试。

```
module Decrorator

    def wrap method_name
        WarppedObj.new method_name, self
    end

    class WarppedObj

        def initialize method_name, context
            @method_name = method_name
            @context = context
            context.class_eval do
                @@_warped_counter ||= {}
                @@_warped_counter[method_name] ||= 0
            end
        end

        def with method_name, *args
            warpped_method = @method_name
            @context.class_eval do
                @@_warped_counter[warpped_method] += 1
                alias_method :"#{@@_warped_counter[warpped_method]}_#{warpped_method}",
                warpped_method
                i = @@_warped_counter[warpped_method]
                define_method warpped_method do |*_args, &_blk|
                    ret = __send__ :"#{i}_#{warpped_method}", *_args, &_blk
                    __send__(method_name, *args) {ret}
                end
            end
        end
    end

end


class HTMLHelper
    extend Decrorator

    def makeHtmlTag tag, opts
        "<#{tag} #{opts.map {|key, value| %Q{#{key}="#{value}"}}.join(' ')}>" + yield +
""
    end

    def hello
        'hello world'
    end

    wrap(:hello).with :makeHtmlTag, 'i', class: 'italic_css'
    wrap(:hello).with :makeHtmlTag, 'b', class: 'bold_css'
end

puts HTMLHelper.new.hello
```

附录 学习Ruby的一些小窍门

必备工具

irb

查祖先

1.9.3-p545 :023 > String.ancestors => [String, Comparable, Object, Kernel, BasicObject]

String的前面有四个上级

过滤方法

Ruby的方法非常多，以至于不得不用grep了。

1.9.3-p545 :049 > [].methods.grep /^me/ => [:member?, :methods, :method]

查方法来源

1.9.3-p545 :018 > {}.method :select => #<Method: Hash#select> 1.9.3-p545 :019 > {}.method :reduce => #<Method: Hash(Enumerable)#reduce>

method方法可以实现。

文档

如果找明确的方法，可以用ri，方便快捷。 ri String.sub

对象模型

各种对象语言的原理是类似的，但是内部实现模型是不一样的。Ruby用起来简单，内部很复杂的。

当你感觉Ruby好用的时候，其实应该感谢Matz的工作。

Kernel模块是核心，很多重要的方法都在里面。推荐看《Ruby元编程》，书中有详细介绍。

告别:

希望这本小书成为你学习Ruby的起点，再见！

2015-1 修订

请我喝一杯，手机支付宝