Ruby on Rails 入门之：(10) Ruby 中的对象

1. 类的定义与使用

1.1 类的定义

Ruby 是一个完全的面向对象的语言，在 Ruby 中所有的一切的数据类型都是对象，然而 Ruby 是一种弱类型的语言，也就是说变量在使用之前不许要定义，而且不用分别变量的类型。

Ruby 中类的定义如代码所示：

[ruby] view plaincopy

```ruby
#encoding:gbk
puts "test of class in Ruby!";

class Animal
    puts "i amm an animal!"

end
```

在类中的 puts 语句会直接输出，不像 C#中的类没有作用，这里的类在运行的时候 puts 语句会运行。

在 Ruby 中要求类的第一个字符必须是大写字符，如果第一个字符不是大写字符，那么程序在运行的时候会出现错误。

如果一个类中有多个输出语句，将会按照顺序进行输出。

[ruby] view plaincopy

```ruby
#encoding:gbk
puts "test of class in Ruby!";

class Animal
    puts "i amm an animal!"

    def cal
        puts "calculate...";
    end

    puts "the second puts";
end
```

1.2 self 关键字

c++中使用 this 关键字来表示当前对象，在 Ruby 中使用 self 表示当前对象。

[ruby] view plaincopy

```ruby
#encoding:gbk
puts "test of class in Ruby!";

class Animal
    puts "i amm an animal!"

    def cal
        puts "calculate...";
    end

    #puts "the second puts";

    puts self;
    puts self.class;
end
```

输出结果为：

[html] view plaincopy

```
watkins@watkins:~/temp/workspace/ruby$ ruby class.rb
test of class in Ruby!
i amm an animal!
Animal
Class
watkins@watkins:~/temp/workspace/ruby$
```

我们再看看在一个具体的对象里输出的 self 是什么，代码如下：

[ruby] view plaincopy

```ruby
#encoding:gbk
puts "test of class in Ruby!";

class Animal
    puts "i amm an animal!"

    def cal
        puts "calculate...";
    end

```

```
11.     #puts "the second puts";
12.
13.     puts self;
14.     puts self.class;
15.
16.     def put
17.         puts self;
18.         puts self.class;
19.     end
20. end
21.
22. a = Animal.new;
23. a.put;
```

输出结果为：

[html] view plaincopy

```
1.  watkins@watkins:~/temp/workspace/ruby$ ruby class.rb
2.  test of class in Ruby!
3.  i amm an animal!
4.  Animal
5.  Class
6.  #<Animal:0xb77c272c>
7.  Animal
8.  watkins@watkins:~/temp/workspace/ruby$
```

之所以会出现这样的情况，是因为我们在类中直接使用 puts 语句输出 self 的时候，我们作用的对象是一个类，不是一个具体的对象，当我们作用于 a.put 的时候，我们的具体的操作的是一个具体的对象，所以会输出这个对象的地址。那么这个对象的类型也是 Animal 类型。这里需要特别注意。

1.3 追加类

在使用类的时候，如果有两个或多个同名的类，系统会自动的合并这几个类。

[ruby] view plaincopy

```
1.  #encoding:gbk
2.  puts "test of class in Ruby!";
3.
4.  class Animal
5.      puts "i amm an animal!"
6.
7.      def cal
```

```
8.          puts "calculate...";
9.      end
10.
11.     #puts "the second puts";
12.
13.     puts self;
14.     puts self.class;
15.
16.     def put
17.         puts self;
18.         puts self.class;
19.     end
20. end
21.
22. class Animal
23.     def sayHello
24.         puts "hello";
25.     end
26. end
27.
28. a = Animal.new;
29. a.put;
30. a.sayHello;
```

1.4 嵌套类

嵌套类的使用要使用::双冒号来引用里面的嵌套类。

[ruby] view plaincopy

```
1.  #encoding:gbk
2.
3.  class Animal
4.      class Head
5.          def put
6.              puts "this is class Head's method put";
7.          end
8.      end
9.  end
10.
11. h = Animal::Head.new
12. h.put;
```

还可以在类的外部直接定义嵌套类：

[ruby] view plaincopy

```ruby
#encoding:gbk

class Animal
    class Head
        def put
            puts "this is class Head's method put";
        end
    end
end

class Animal::Body
    def put
        puts "this is class Body's method put";
    end
end

h = Animal::Head.new
h.put;

b = Animal::Body.new
b.put;
```

1.5 特殊类

在 Ruby 中还可一位某个特定的对象追加一些方法和属性，这些方法和属性封装到一个类中直接追加到对象中。这个类没有特定的类名，所以不能作为普通类使用，只能当作特殊类使用。

[ruby] view plaincopy

```ruby
#encoding:gbk

class Animal
    #empty
end

a = Animal.new;

class << a
    def put
        puts "added method";
    end
end

a.put;
```

这里我们定义了一个空的类 Animal，然后为对象 a 添加了一个方法。特殊类追加方法和属性只能给具体的对象追加，不能给类追加。