Ruby 类案例 Ruby 运算符

Ruby 变量

变量是持有可被任何程序使用的任何数据的存储位置。

Ruby 支持五种类型的变量。

- 一般小写字母、下划线开头：变量（Variable）。
- $开头：全局变量（Global variable）。
- @开头：实例变量（Instance variable）。
- @@开头：类变量（Class variable）类变量被共享在整个继承链中
- 大写字母开头：常数（Constant）。

您已经在前面的章节中大概了解了这些变量，本章节将为您详细讲解这五种类型的变量。

Ruby 全局变量

全局变量以 $ 开头。未初始化的全局变量的值为 *nil*，在使用 -w 选项后，会产生警告。

给全局变量赋值会改变全局状态，所以不建议使用全局变量。

下面的实例显示了全局变量的用法。

实例

```
#!/usr/bin/ruby
# -*- coding: UTF-8 -*-
$global_variable = 10
class Class1
def print_global
puts "全局变量在 Class1 中输出为 #$global_variable"
end
end
class Class2
def print_global
puts "全局变量在 Class2 中输出为 #$global_variable"
end
end
class1obj = Class1.new
class1obj.print_global
class2obj = Class2.new
class2obj.print_global
```

尝试一下 »

在这里，$global_variable 是全局变量。这将产生以下结果：

注意：在 Ruby 中，您可以通过在变量或常量前面放置 # 字符，来访问任何变量或常量的值。

```
全局变量在 Class1 中输出为 10
全局变量在 Class2 中输出为 10
```

Ruby 实例变量

实例变量以 @ 开头。未初始化的实例变量的值为 *nil*，在使用 -w 选项后，会产生警告。

下面的实例显示了实例变量的用法。

实例

```
#!/usr/bin/ruby
class Customer
def initialize(id, name, addr)
@cust_id=id
@cust_name=name
@cust_addr=addr
end
def display_details()
puts "Customer id #@cust_id"
puts "Customer name #@cust_name"
puts "Customer address #@cust_addr"
end
end
# 创建对象
cust1=Customer.new("1", "John", "Wisdom Apartments, Ludhiya")
cust2=Customer.new("2", "Poul", "New Empire road, Khandala")
# 调用方法
cust1.display_details()
cust2.display_details()
```

尝试一下 »

在这里，@cust_id、@cust_name 和 @cust_addr 是实例变量。这将产生以下结果：

```
Customer id 1
Customer name John
Customer address Wisdom Apartments, Ludhiya
Customer id 2
Customer name Poul
Customer address New Empire road, Khandala
```

Ruby 类变量

类变量以 @@ 开头，且必须初始化后才能在方法定义中使用。

引用一个未初始化的类变量会产生错误。类变量在定义它的类或模块的子类或子模块中可共享使用。

在使用 -w 选项后，重载类变量会产生警告。

下面的实例显示了类变量的用法。

实例

```ruby
#!/usr/bin/ruby
class Customer
@@no_of_customers=0
def initialize(id, name, addr)
@cust_id=id
@cust_name=name
@cust_addr=addr
end
def display_details()
puts "Customer id #@cust_id"
puts "Customer name #@cust_name"
puts "Customer address #@cust_addr"
end
def total_no_of_customers()
@@no_of_customers += 1
puts "Total number of customers: #@@no_of_customers"
end
end
# 创建对象
cust1=Customer.new("1", "John", "Wisdom Apartments, Ludhiya")
cust2=Customer.new("2", "Poul", "New Empire road, Khandala")
# 调用方法
cust1.total_no_of_customers()
cust2.total_no_of_customers()
```

尝试一下 »

在这里，@@no_of_customers 是类变量。这将产生以下结果：

```
Total number of customers: 1
Total number of customers: 2
```

Ruby 局部变量

局部变量以小写字母或下划线 _ 开头。局部变量的作用域从 class、module、def 或 do 到相对应的结尾或者从左大括号到右大括号 {}。

当调用一个未初始化的局部变量时，它被解释为调用一个不带参数的方法。

对未初始化的局部变量赋值也可以当作是变量声明。变量会一直存在，直到当前域结束为止。局部变量的生命周期在 Ruby 解析程序时确定。

在上面的实例中，局部变量是 id、name 和 addr。

Ruby 常量

常量以大写字母开头。定义在类或模块内的常量可以从类或模块的内部访问，定义在类或模块外的常量可以被全局访问。

常量不能定义在方法内。引用一个未初始化的常量会产生错误。对已经初始化的常量赋值会产生警告。

实例

```ruby
#!/usr/bin/ruby
# -*- coding: UTF-8 -*-
class Example
VAR1 = 100
VAR2 = 200
def show
puts "第一个常量的值为 #{VAR1}"
puts "第二个常量的值为 #{VAR2}"
end
end
# 创建对象
object=Example.new()
object.show
```

尝试一下 »

在这里，VAR1 和 VAR2 是常量。这将产生以下结果：

```
第一个常量的值为 100
第二个常量的值为 200
```

Ruby 伪变量

它们是特殊的变量，有着局部变量的外观，但行为却像常量。您不能给这些变量赋任何值。

- **self:** 当前方法的接收器对象。
- **true:** 代表 true 的值。
- **false:** 代表 false 的值。
- **nil:** 代表 undefined 的值。
- **__FILE__:** 当前源文件的名称。
- **__LINE__:** 当前行在源文件中的编号。

← Ruby 类案例 Ruby 运算符 →