

Lua 教程
Lua 教程
Lua 环境安装
Lua 基本语法
Lua 数据类型
Lua 变量
Lua 循环
Lua 流程控制
Lua 函数
Lua 运算符
Lua 字符串
Lua 数组
Lua 选代器
Lua table(表)
Lua 模块与包
Lua 表(Metatable)
Lua 协同程序(coroutine)
Lua 文件 I/O
Lua 错误处理
Lua 调试(Debug)
Lua 垃圾回收
Lua 面向对象
Lua 数据库访问
Lua5.3 参考手册

<- Lua 垃圾回收

Lua 数据库访问 →

Lua 面向对象

面向对象编程 (Object Oriented Programming , OOP) 是一种非常流行的计算机编程架构。

以下几种编程语言都支持面向对象编程 :

- C++
- Java
- Objective-C
- Smalltalk
- C#
- Ruby

面向对象特征

- 1) 封装 : 能够把一个实体的信息、功能、响应都放入一个单独的对象中的特性。
- 2) 继承 : 继承的方法允许在不改动原程序的基础上对其进行扩充，这样使得原功能得以保存，而新功能也得以扩展。这有利于减少重复编码，提高软件的开发效率。
- 3) 多态 : 同一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果。在运行时，可以通过指向基类的指针，来调用实现派生类中的方法。
- 4) 抽象 : 抽象(Abstract)是简化复杂的现实问题的途径，它可以为具体问题找到最恰当的类定义，并且可以在最恰当的继承级别解释问题。

Lua 中面向对象

我们知道，对象由属性和方法组成。Lua中最基本的结构是

Lua 中的 function 可以用来表示方法。那么Lua中的类可以通过 table + function 模拟出来。

至于继承，可以通过 metatable 模拟出来（不推荐用，只模拟最基本的对象大部分时间够用了）。

Lua 中的类不仅在语义上是一种对象，像对象一样，表也有状态（成员变量）；也有与对象的值独立的本性，特别是拥有两个不同值的对象 (table) 代表两个不同的对象；一个对象在不同的时候也可以有不同的值，但他始终是一个对象；与对象类似，表的生命周期与其由什么创建、在哪创建没有关系。对象有他们的成员函数，表也有：

```
Account = {balance = 0}
function Account:withdraw (v)
    Account.balance = Account.balance - v
end
```

这个定义创建了一个新的函数，并且保存在Account对象的withdraw域内，下面我们可以这样调用：

```
Account.withdraw(100.00)
```

一个简单实例

以下简单的类包含了三个属性：area, length 和 breadth，printArea方法用于打印计算结果：

实例

```
-- 元类
Rectangle = {area = 0, length = 0, breadth = 0}

-- 派生类的方法 new
function Rectangle:new (o,length,breadth)
    o = o or {}
    setmetatable(o, self)
    self.__index = self
    self.length = length or 0
    self.breadth = breadth or 0
    self.area = length*breadth
    return o
end

-- 派生类的方法 printArea
function Rectangle:printArea ()
    print("矩形面积为 ",self.area)
end
```

创建对象

创建对象是为类的实例分配内存的过程。每个类都有属于自己的内存并共享公共资源。

```
r = Rectangle:new(10,10,20)
```

访问属性

我们可以使用点号(.)来访问类的属性：

```
print(r.length)
```

访问成员函数

我们可以使用冒号(:)来访问类的成员函数：

```
r:printArea()
```

内存对象初始化时分配。

完整实例

以下我们演示了 Lua 面向对象的完整实例：

```
-- 元类
Shape = {area = 0}

-- 基础类方法 new
function Shape:new (o,side)
    o = o or {}
    setmetatable(o, self)
    self.__index = self
    side = side or 0
    self.area = side*side;
    return o
end

-- 基础类方法 printArea
function Shape:printArea ()
    print("面积为 ",self.area)
end
```

-- 创建对象

myshape = Shape:new(10,10)

myshape:printArea()

执行以上程序，输出结果为：

面积为 100

Lua 继承

继承是指一个对象直接使用另一对象的属性和方法，可用于扩展基础类的属性和方法。

以下演示了一个简单的继承实例：

```
-- Meta class
Shape = {area = 0}
-- 基础类方法 new
function Shape:new (o,side)
    o = o or {}
    setmetatable(o, self)
    self.__index = self
    side = side or 0
    self.area = side*side;
    return o
end

-- 基础类方法 printArea
function Shape:printArea ()
    print("面积为 ",self.area)
end
```

接下来的实例，Square 对象继承了 Shape 类

```
-- Derived class method new
function Square:new (o,side)
    o = o or Shape:new(o,side)
    setmetatable(o, self)
    self.__index = self
    return o
end
```

-- 派生类方法 new

function Square:new (o,side)
 o = o or Shape:new(o,side)
 setmetatable(o, self)
 self.__index = self
 return o
end

-- 派生类方法 printArea

function Square:printArea ()
 print("正方形面积为 ",self.area)
end

-- 创建对象

mysquare = Square:new(10,10)

mysquare:printArea()

Rectangle = Shape:new()

-- 派生类方法 new

function Rectangle:new (o,length,breadth)
 o = o or {}
 setmetatable(o, self)
 self.__index = self
 self.length = length or 0
 self.breadth = breadth
 self.area = length * breadth
 return o
end

-- 派生类方法 printArea

function Rectangle:printArea ()
 print("矩形面积为 ",self.area)
end

-- 创建对象

myrectangle = Rectangle:new(10,10,20)

myrectangle:printArea()

执行以上代码，输出结果为：

面积为 100

正方形面积为 100

矩形面积为 200

函数重写

Lua 中我们可以重写基础类的函数，在派生类中定义自己的实现方式：

```
-- 派生类方法 printArea
function Square:printArea ()
    print("正方形面积 ",self.area)
end
```

<- Lua 垃圾回收

Lua 数据库访问 →

8 篇笔记

写笔记

视频通话SDK

声网Agora.io , API接口 , 4行代码接入。每月1万分钟免费。 www.agora.io

广告

分类导航

HTML / CSS

JavaScript

服务端

数据库

移动端

XML 教程

ASP.NET

Web Service

开发工具

网站建设

Advertisement

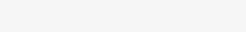
CSS HTML JS

学习资料

你的WEB前端的

这里有

我要听课



扫描二维码