Go 语言环境安装 Go 语言基础语法

Go 语言结构

在我们开始学习 Go 编程语言的基础构建模块前，让我们先来了解 Go 语言最简单程序的结构。

Go Hello World 实例

Go 语言的基础组成有以下几个部分：

- 包声明
- 引入包
- 函数
- 变量
- 语句 & 表达式
- 注释

接下来让我们来看下简单的代码，该代码输出了"Hello World!":

实例

```
package main

import "fmt"

func main() {
   /* 这是我的第一个简单的程序 */
   fmt.Println("Hello, World!")
}
```

让我们来看下以上程序的各个部分：

1. 第一行代码 *package main* 定义了包名。你必须在源文件中非注释的第一行指明这个文件属于哪个包，如：package main。package main表示一个可独立执行的程序，每个 Go 应用程序都包含一个名为 main 的包。
2. 下一行 *import "fmt"* 告诉 Go 编译器这个程序需要使用 fmt 包（的函数，或其他元素），fmt 包实现了格式化 IO（输入/输出）的函数。
3. 下一行 *func main()* 是程序开始执行的函数。main 函数是每一个可执行程序所必须包含的，一般来说都是在启动后第一个执行的函数（如果有 init() 函数则会先执行该函数）。
4. 下一行 /*...*/ 是注释，在程序执行时将被忽略。单行注释是最常见的注释形式，你可以在任何地方使用以 // 开头的单行注释。多行注释也叫块注释，均已以 /* 开头，并以 */ 结尾，且不可以嵌套使用，多行注释一般用于包的文档描述或注释成块的代码片段。

5. 下一行 *fmt.Println(...)* 可以将字符串输出到控制台，并在最后自动增加换行字符 \n。

   使用 fmt.Print("hello, world\n") 可以得到相同的结果。

   Print 和 Println 这两个函数也支持使用变量，如：fmt.Println(arr)。如果没有特别指定，它们会以默认的打印格式将变量 arr 输出到控制台。

6. 当标识符（包括常量、变量、类型、函数名、结构字段等等）以一个大写字母开头，如：Group1，那么使用这种形式的标识符的对象就可以被外部包的代码所使用（客户端程序需要先导入这个包），这被称为导出（像面向对象语言中的 public）；标识符如果以小写字母开头，则对包外是不可见的，但是他们在整个包的内部是可见并且可用的（像面向对象语言中的 protected ）。

执行 Go 程序

让我们来看下如何编写 Go 代码并执行它。步骤如下：

1. 打开编辑器如Sublime2，将以上代码添加到编辑器中。
2. 将以上代码保存为 *hello.go*
3. 打开命令行，并进入程序文件保存的目录中。
4. 输入命令 *go run hello.go* 并按回车执行代码。
5. 如果操作正确你将在屏幕上看到 *"Hello World!"* 字样的输出。

```
$ go run hello.go
Hello, World!
```

注意

需要注意的是 `{` 不能单独放在一行，所以以下代码在运行时会产生错误：

实例

```
package main

import "fmt"

func main()
{  // 错误，{ 不能在单独的行上
    fmt.Println("Hello, World!")
}
```

← Go 语言环境安装　　Go 语言基础语法 →

当前的调试部分可以使用 **`go run filename.go`** 来执行。

可以生成一个 **build.sh** 脚本，用于在指定位置产生已编译好的 可执文件:

```
#!/usr/bin/env bash

CURRENT_DIR=`pwd`
OLD_GO_PATH="$GOPATH"  #例如: /usr/local/go
OLD_GO_BIN="$GOBIN"    #例如: /usr/local/go/bin

export GOPATH="$CURRENT_DIR"
export GOBIN="$CURRENT_DIR/bin"

#指定并整理当前的源码路径
gofmt -w src

go install test_hello

export GOPATH="$OLD_GO_PATH"
export GOBIN="$OLD_GO_BIN"
```

糊涂的爸爸 1年前 (2017-12-01)