

Khoá học Golang Vue.js cho fresher Open Commerce

cuong@techmaster.vn

Phương pháp học

- Không căng thẳng, không cố nhớ cú pháp Golang
- Không phân nản cú pháp Golang khác với ngôn ngữ bạn quen dùng. Mà giữ một bản cheat sheet so sánh ngôn ngữ bạn quen và Go.
- Nên chọn sớm một ý tưởng làm đồ án. Không cần hoành tráng mà phải chạy được đã.
- Học Go routing, pointer... khi bạn cảm thấy cần dùng nó.
- Luôn đặn ghi chép lại và viết blog quá trình học – lập trình – làm sản phẩm với Golang

Kế hoạch đào tạo

- Một tuần học 3 buổi. Học Vue và Golang xen kẽ nhau.
- Sau 10 buổi Vue, 10 buổi Golang các bạn sẽ bảo vệ đồ án.
- Tìm hiểu thật kỹ web site bán hàng đơn giản bằng Golang + Vue.js để có kinh nghiệm sau đó lập trình sản phẩm khác.
- 4 buổi Golang cuối cùng sẽ học nâng cao về: RabbitMQ, JWT, GRPC, Concurrency

Các tài liệu tự học Golang

- <https://tour.golang.org/>
- <https://yourbasic.org/golang/go-java-tutorial/>
- <https://gobyexample.com/>
- <https://tutorialedge.net/golang/>
- Các khoá học trên Udemy cực nhiều tha hồ tìm.

Chuẩn bị buổi thứ 20 bảo vệ sản phẩm

- Hãy bắt đầu tích cực ngay từ ngày hôm nay
- Ý tưởng cho đề án là gì? Có hữu ích gì đối với hoạt động của OCG?
- Đề án giúp rèn luyện những kỹ năng gì?
- Một ngày bạn dành ra bao nhiêu tiếng để lập trình đề án?
- Bạn đã có những kinh nghiệm gì trước khi vào OCG và cần bổ xung gì để làm sản phẩm

Ý tưởng cho đồ án

- Web site đặt vé xem phim qua mạng kiểu như CGV.vn
- Web site cho phép tạo ra CV online
- Web site mua bán, cho thuê bất động sản
- Web site tuyển và nhận làm việc part time

Nhập môn Golang

Cài đặt Golang

- Mac
 - Cài đặt `$ brew install go`
 - Gỡ bỏ `$ brew uninstall go`
- Ubuntu
 - <https://www.cyberciti.biz/faq/how-to-install-gol-ang-on-ubuntu-linux/>
- Windows
 - Tải về, cài đặt

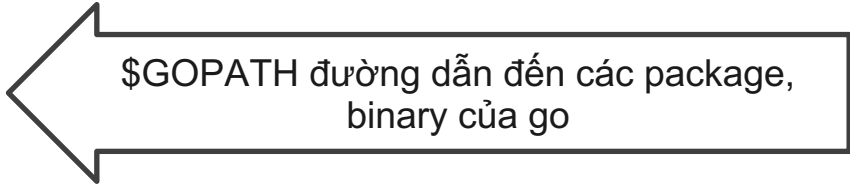
Kiểm tra môi trường

```
$ go version  
go version go1.16.3 darwin/amd64
```



Kiểm tra phiên bản Go cài trên máy

```
$ echo $GOPATH  
/Users/techmaster/golang
```



\$GOPATH đường dẫn đến các package,
binary của go

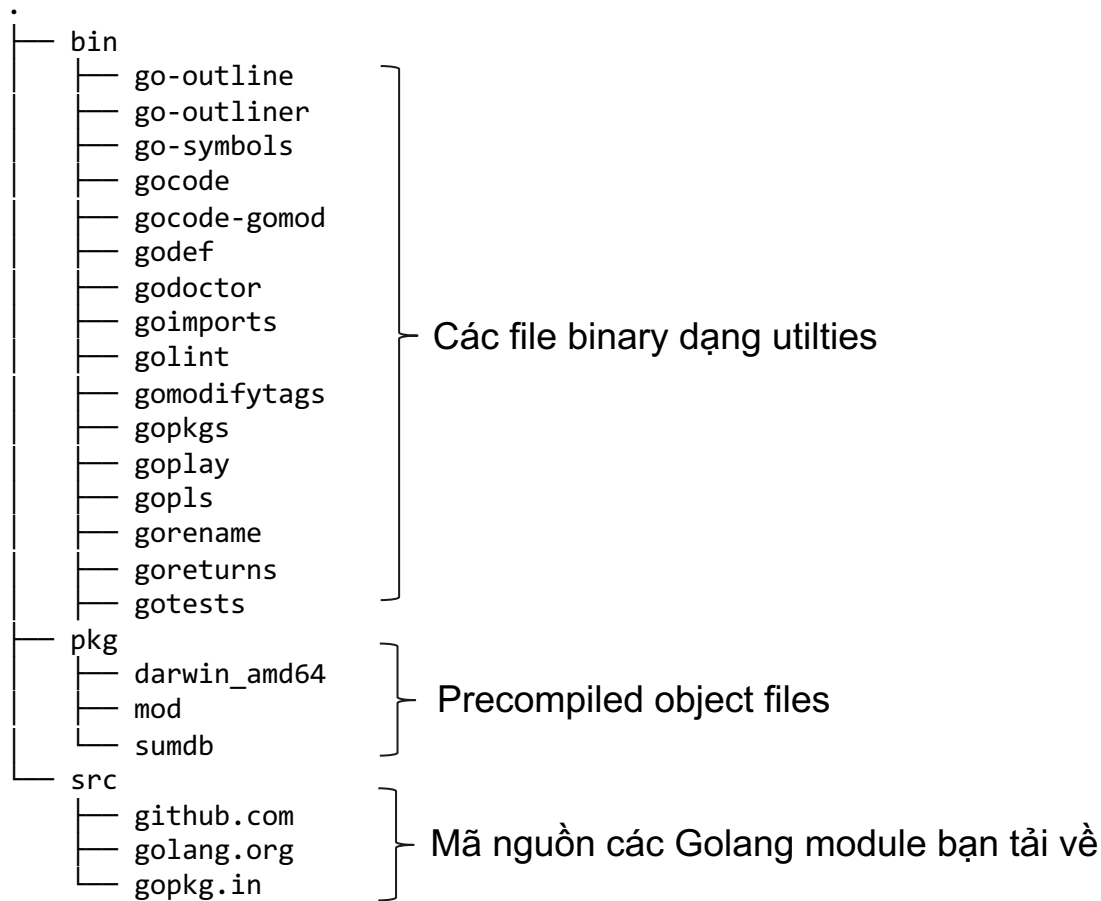
```
$ cd $GOPATH
```

```
$ ls  
bin pkg src
```

Trong \$GOPATH có gì?

```
$ cd $GOPATH
```

```
$ tree -L 2
```

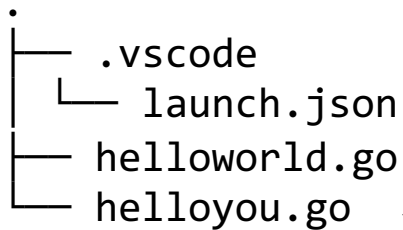


IDE để lập trình Golang

- VSCode + Extension đủ dùng
- JetBrains Goland mất phí

Tạo ứng dụng Go đầu tiên

- `go run`, `go build`
- Launch file vs Launch package
- `go mod init`
- `go mod tidy`



```
package main

import "fmt"

func main() {
    fmt.Println("hello world")
    Say("Hello my friend")
}
```

```
package main

import "fmt"

func Say(msg string) {
    fmt.Println(msg)
}
```

Go vs Java

Hello World

JAVA

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```

GO

```
package main  
import "fmt"  
func main() {  
    fmt.Println("hello world")  
}
```

Khai báo biến

JAVA

```
int a;  
int b = 1;  
var c = 1;  
String s = "Hello";  
final var hello = "Hello";
```

GO

```
var s string = "Hello" // outside method  
var a int  
var b = 1  
c := 1 // inside methods  
d := "Hello"  
const hello = "Hello"
```


String

JAVA

```
var name = "John";  
var lastName = "Smith";  
var text = "My name is: " + name + " " + lastName;  
var text2 = String.format("My names is: %s %s", name, lastName);  
var otherText = "My name is: " + name.substring(2);
```

GO

```
var name = "John"  
var lastName = "Smith"  
var text = "My name is: " + name + " " + lastName;  
var text2 = fmt.Sprintf("My names is: %s %s", name, lastName)  
var otherText = "My name is: " + name[2:len(name)]
```

Multiple lines String

JAVA

```
var text = "First Line\n" +  
"Second Line\n" +  
"Third Line";
```

```
var textjdk15 = ""  
this is  
multiline in JDK15"";
```

GO

```
var text = `First Line  
Second Line  
Third Line`
```

Multiple lines String

JAVA

```
var text = "First Line\n" +  
"Second Line\n" +  
"Third Line";
```

```
var textjdk15 = ""  
this is  
multiline in JDK15"";
```

GO

```
var text = `First Line  
Second Line  
Third Line`
```

If else

```
public int min(int x, int y) {  
    if (x < y) {  
        return x;  
    } else {  
        return y;  
    }  
}
```

Java

```
public int minTernary(int x, int y) {  
    return (x < y) ? x : y;  
}
```

```
func min(x int, y int) int {  
    if x < y {  
        return x  
    } else {  
        return y  
    }  
}
```

Go

Go có không có ternary condition

Go if with statement

```
func BMIndex(weight float32, height float32) string {  
    if bmi := weight / (height * height); bmi < 18.5 {  
        return "Underweight"  
    } else if bmi < 25 {  
        return "Normal"  
    } else {  
        return "Overweight"  
    }  
}
```

Đặc sản !

Switch in Java

```
String language = "French";  
switch (language) {  
    case "Spanish":  
        System.out.println("Buenos dias!");  
        break;  
    case "French":  
        System.out.println("Bonjour!");  
        break;  
    default:  
        System.out.println("Hello!");  
}
```

```
switch (month) {  
    case JANUARY, FEBRUARY, MARCH -> System.out.println("1st Quarter");  
        //no break needed  
    case APRIL, MAY, JUNE -> System.out.println("2nd Quarter");  
    case JULY, AUGUST, SEPTEMBER -> System.out.println("3rd Quarter");  
    case OCTOBER, NOVEMBER, DECEMBER -> System.out.println("4th Quarter");  
    default -> System.out.println("Unknown Quarter");  
}
```

```
String quarter = switch (month) {  
    case JANUARY, FEBRUARY, MARCH -> "First Quarter"; //must return single value  
    case APRIL, MAY, JUNE -> "Second Quarter";  
    case JULY, AUGUST, SEPTEMBER -> "Third Quarter";  
    case OCTOBER, NOVEMBER, DECEMBER -> "Forth Quarter";  
    default -> "Unknown Quarter";  
};
```

Switch in Go

```
func Greeting() {  
    switch hour := time.Now().Hour(); {  
    case hour < 12:  
        fmt.Println("Good morning!")  
    case hour < 17:  
        fmt.Println("Good afternoon!")  
    default:  
        fmt.Println("Good evening!")  
    }  
}
```

```
func Quarter(month string) string {  
    switch month {  
    case "Jan", "Feb", "Mar":  
        return "First Quarter"  
    case "Apr", "May", "Jun":  
        return "Second Quarter"  
    case "Jul", "Aug", "Sep":  
        return "Third Quarter"  
    case "Oct", "Nov", "Dec":  
        return "Forth Quarter"  
    default:  
        return "Unknown Quarter"  
    }  
}
```


Array

JAVA

```
String cars[] = new String[]{"Toyota", "Mercedes", "BMW"};  
System.out.println(cars[0]); // Toyota
```

GO

```
cars := [3]string{"Toyota", "Mercedes", "BMW"}  
fmt.Println(cars[0]) // Toyota
```

For loop array

JAVA

```
for (String car : cars) {  
    System.out.println(car);  
}
```

GO

```
for _, car := range cars {  
    fmt.Println(car)  
}
```

2 dimensions array

JAVA

```
String langs[][] = new String[][]{{"C#", "C", "Python"},  
{"Java", "Scala", "Perl"},  
{"C++", "Go", "HTML"}};
```

GO

```
langs := [3][3]string{{"C#", "C", "Python"},  
{"Java", "Scala", "Perl"},  
{"C++", "Go", "HTML"}}
```

Nested loop

JAVA

```
for (String[] arr : langs) {  
    for (String lang : arr) {  
        System.out.print(lang + " ");  
    }  
    System.out.println();  
}
```

GO

```
for _, v := range langs {  
    for _, lang := range v {  
        fmt.Print(lang, " ")  
    }  
    fmt.Println()  
}
```

List vs Slice

JAVA

```
List<String> letters = new ArrayList<>(List.of("a", "b", "c", "d"));  
  
letters.add("e");  
  
int length = letters.size();
```

GO

```
letters := []string{"a", "b", "c", "d"}  
  
letters = append(letters, "e")  
  
length := len(letters)
```

List vs Slice

JAVA

```
List<String> letters = new ArrayList<>(List.of("a", "b", "c", "d"));  
letters.forEach(value -> System.out.println(value));  
letters.forEach(System.out::println);
```

GO

```
letters := []string{"a", "b", "c", "d"}  
for _, letter := range letters {  
    fmt.Println(letter)  
}
```

Map

JAVA

```
Map<String, String> mapA = new HashMap<>();
mapA.put("MSFT", "Microsoft");
mapA.put("APPL", "Apple");
mapA.remove("APPLE");
if (mapA.containsKey("MSFT")) {
    System.out.println(mapA.get("MSFT"));
}
```

GO

```
mapA := map[string]string{"MSFT":"Microsoft"}
mapA["APPL"] = "Apple"
delete(mapA, "APPL")
if value, ok := mapA["MSFT"]; ok == true {
    fmt.Println(value)
}
```

Loop through Map

JAVA

```
Map<String, String> mapA = new HashMap<>();  
mapA.put("MSFT", "Microsoft");  
mapA.put("APPL", "Apple");  
mapA.forEach((key, value) -> System.out.println(key + ":" + value));
```

GO

```
mapA := map[string]string{"MSFT": "Microsoft", "APPL": "Apple"}  
for key, value := range mapA {  
    fmt.Println(key + ":" + value)  
}
```


Private vs Public function in Java

```
public class Util {  
    private void doInternally() {  
        System.out.println("this is private method");  
    }  
  
    public int add(int ...a) {  
        int result = 0;  
        for (int x: a) {  
            result = result + x;  
        }  
        return result;  
    }  
}
```

Access modifier keywords:
private, **public**, **protected**

```
public class App {  
    public static void main(String[] args) {  
        Util util = new Util();  
        System.out.println(util.add(1, 2, 3, 4, 5));  
    }  
}
```

Private and Public Function in Go

```
func doInternally() {  
    fmt.Println("this is private func")  
}  
  
func Add(a ...int) int {  
    result := 0  
    for _, x := range a {  
        result = result + x  
    }  
    return result  
}
```

- Ký tự đầu chữ **t** thường **private**
- Ký tự đầu chữ **H** hoa **public**

```
func main() {  
    fmt.Println(Add(1, 2, 3, 4, 5))  
}
```

Java Class

```
public class Person {  
    private String firstName;  
    private String lastName;  
    private int age;  
  
    public Person(String firstName, String lastName, int age) {  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.age = age;  
    }  
  
    public String getFullName() {  
        return firstName + " " + lastName;  
    }  
    @Override  
    public String toString() {  
        return getFullName() + " is " + age + " years old";  
    }  
}
```

```
Person tom = new Person("Tom", "Sawyer", 15);  
System.out.println(tom);
```

Go Struct

```
type Person struct {  
    FirstName string  
    LastName  string  
    Age       int  
}
```

```
func (p *Person) FullName() string {  
    return p.FirstName + " " + p.LastName  
}
```

```
func (p Person) String() string {  
    return fmt.Sprintf("%v is %v years old", p.FullName(), p.Age)  
}
```

```
tom := Person{"Tom", "Sawyer", 15}  
fmt.Println(tom)
```

Golang không thực sự có constructor

```
func NewPerson(firstName string, lastName string, age int) *Person {  
    if age < 0 {  
        return nil  
    }  
    p := new(Person)  
    p.FirstName = firstName  
    p.LastName = lastName  
    p.Age = age  
    return p  
}
```

```
var tom = NewPerson("Tom", "Sawyer", -15)  
fmt.Println(tom)
```

```
func BuildPerson() *Person {  
    return new(Person)  
}  
  
func (p *Person) WithFirstName(firstName string) *Person {  
    p.FirstName = firstName  
    return p  
}  
  
func (p *Person) WithLastName(lastName string) *Person {  
    p.LastName = lastName  
    return p  
}  
  
func (p *Person) WithAge(age int) *Person {  
    p.Age = age  
    return p  
}
```

Golang cũng viết
được Fluent API

```
bob := BuildPerson().WithFirstName("Bob").WithLastName("Aladin").WithAge(37)  
fmt.Println(bob)
```

```
var x *Person = Person{}
```

- Khai báo và khởi tạo biến trong và ngoài function
- Phạm vi: package, global, local
- Khai báo có thể tách khởi tạo
- Có thể bổ xung kiểu

```
x := Person{}
```

- Chỉ dùng để khai báo và khởi tạo biến trong function
- Phạm vi: local trong function
- Khai báo luôn đi cùng khởi tạo
- Không được bổ xung kiểu. Kiểu xác định khi trong lệnh gán

Pointer receiver hay Value receiver ?

```
func (p *Person) FullName() string {    //Pointer receiver
    return p.FirstName + " " + p.LastName
}
```

```
func (p Person) String() string {        //Value receiver
    return fmt.Sprintf("%v is %v years old", p.FullName(), p.Age)
}
```


Pointer Receiver

- Tránh không phải copy đối tượng mỗi khi gọi hàm
- Phù hợp khi cần thay đổi thuộc tính bên trong đối tượng
- Tối ưu khi kích thước đối tượng lớn
- Không thread safe (go routine safe) vì nó có thể thay đổi đối tượng (mutable)

Value Receiver

- Copy đối tượng khi truyền
- Không thay đổi thuộc tính bên trong đối tượng (immutable)
- Kém hiệu quả khi kích thước đối tượng lớn
- Go routing safe vì immutable

Java Exception

- Sử dụng throw, try catch exception
- Cơ chế Stack Unwinding
- Check Exception yêu cầu khai báo throws trong method
- Non Check Exception không yêu cầu khai báo throws
- Có thể tạo Custom Exception

Go Error

- Không có exception, chỉ trả về error
- Lỗi ở hàm nào, hàm đó phải xử lý
- Có 3 cách tạo error
 - String Error
 - Format String Error
 - Custom Error

Golang không throw exception mà return error

```
package main
```

```
import (  
    "errors"  
    "fmt"  
    "math"  
)
```

```
func Sqrt(f float64) (float64, error) {  
    if f < 0 {  
        return 0, errors.New("math: square root of negative number")  
    }  
    return math.Sqrt(f), nil  
}
```

```
func main() {  
    result, err := Sqrt(-1)  
    if err != nil {  
        fmt.Println(err)  
    }  
    fmt.Println(result)  
}
```

Java đọc file, in ra từng dòng

```
public void readAFile (String fileName) throws IOException {  
    FileReader file = new FileReader(fileName);  
  
    // Try with resource  
    try (BufferedReader bufferReader = new BufferedReader(file)) {  
        String thisLine;  
        while ((thisLine = bufferReader.readLine()) != null) {  
            System.out.println(thisLine);  
        }  
    }  
}
```

Sử dụng kỹ thuật try close resource

Golang đọc file, in ra từng dòng

```
file, err := os.Open("sample.txt") //trả về lỗi err

if err != nil {
    log.Fatalf("failed to open")
}

defer file.Close() //kỹ thuật defer

scanner := bufio.NewScanner(file)

for scanner.Scan() {
    fmt.Println(scanner.Text())
}
```

3 cách tạo Error trong Golang

1

```
errors.New("math: square root of negative number") //plain string error
```

2

```
fmt.Errorf("math: square root of negative number %g", f) //formatted error string
```

```
type error interface {  
    Error() string  
}
```

A diagram showing two structs, `SyntaxError` and `InternalError`, with arrows pointing from them to a central `error` interface definition box above. This indicates that both structs implement the `error` interface.

```
type SyntaxError struct {  
    Line int  
    Col  int  
}  
  
func (e *SyntaxError) Error() string {  
    return fmt.Sprintf("%d:%d: error",  
        e.Line, e.Col)  
}
```

```
type InternalError struct {  
    Path string  
}  
  
func (e *InternalError) Error() string {  
    return fmt.Sprintf("error at %v",  
        e.Path)  
}
```

Cần bổ xung thêm

Con trỏ

Kiểu Date Time

Kiểu Enumeration

Interface

Generics