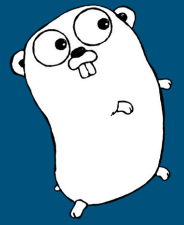
Capacitación de Golang:

Introducción a Go

Por:
Esteban Martínez S.

Historia



- Creado por Google en 2007 y lanzado públicamente en 2009.
- Nació y sigue siendo un proyecto de código abierto.
- Entre 2016 y 2018 alcanza su pick de popularidad, cambiando de branding.
- Con el paso de los años añaden la programación genérica. En 2018 se intentó integrar un paquete de control de errores hecho por la comunidad, el que fue rechazado posteriormente.

Importantes

- Go es un lenguaje de programación de carácter concurrente y compilado.
- Crea programas ejecutables específicos en conjunto y para cada sistema operativo.
- Se pueden desarrollar desde aplicaciones para sistemas(Desk Apps, CLI, ORM’s, etc..), hasta Web apps nativas en la nube (*Cloud native)*.
- Posee un Toolkit como ningún otro.

Importantes

- Listo para producción desde 2012.
- Basado en el desempeño y paradigma de C y C++, junto al dinamismo de tipado que posee Python.
- La mentalidad detrás del lenguaje, es completamente diferente:
  - Orientado a objetos, pero no admite clases.
  - Invocaciones de Objetos, pero no admite herencia.
  - No admite puntero, uniones de conjuntos, conversión implícita de tipos ni afirmaciones.
- Sistema de Tipos: las estructuras son tipos de datos predefinidos, esto para reducir errores por formatos o interacciones entre tipos de datos incompatibles.

Lexer y Types

Lexer: verificador de la sintaxis común de GO, como la indentación, paréntesis, ***puntos y coma*** entre otros. Coloca los puntos y comas automáticamente si fue olvidado.

Types: Elementos sensibles a mayúsculas(métodos públicos y privados). Para trabajar con variables se deben definir previamente, además del ***tipo*** de dato que es. Casi **todo** es un Type.

String, Bool, Int(6 tipos), Float(2 tipos), Complex(i), Array, Slices, Maps, Structs…

ref: https://go.dev/ref/spec#Lexical_elements

¡Hola Mundo!

1. Instalar Go desde el código fuente y el instalador en binario
2. Instalar extensión de **Go** oficial de Google en VSCode
3. Crea una carpeta llamada “mygolang”, luego dentro crea otra carpeta llamada “01hello”. Una vez dentro crea un archivo llamado “main.go”.
4. En el terminal, dentro del directorio “01hello”, ejecutar: *go mod init hello*. (Al ejecutar este comando, puede que VSCode pregunte por instalar el toolkit, se debe aceptar todos los paquetes sugeridos)
5. Insertar código de Hola Mundo —>
6. Compilar el script con *go run main.go*

```go
package main

import "fmt"

func main() {
  fmt.Println("Hola, mundo.")
}
```

Explicando un poco...

- El comando *go mod init hello* crea un archivo con la información de configuración, tales como dependencias y variables de entorno, para esa aplicación en específico. *go mod* es parte del toolkit de **Go**.

- El comando *go run main.go* compila nuestro código en el momento y ejecutando el script sin necesidad de hacer un build y crear un ejecutable para probar nuestro código. *go run* es también parte del toolkit

- Estos son solo dos de los múltiples comandos integrados en el toolkit, algunos nos permiten hacer test, hacer debugging, instalar paquetes de terceros, crear ejecutables...entre otros.

Definiendo Variables

Variables válidas para cada función depende de la definición de ésta.

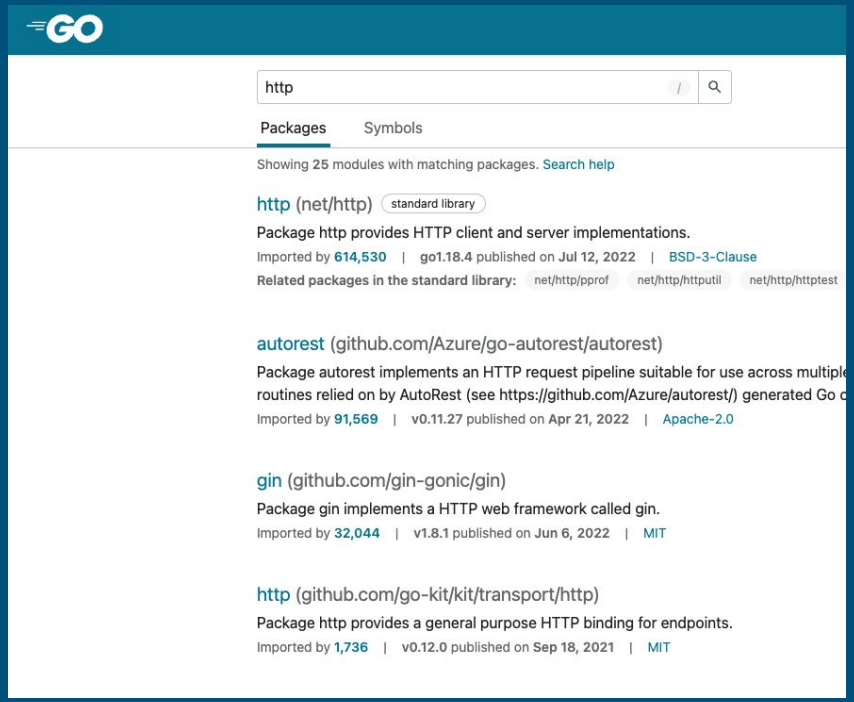
```
var website = "google.cl" // intrinsic declaration

var anotherVariable int // aliases

var smallVal int = 256 // for most use cases

const LoginToken string = "iandasdmp" // Public constant
```

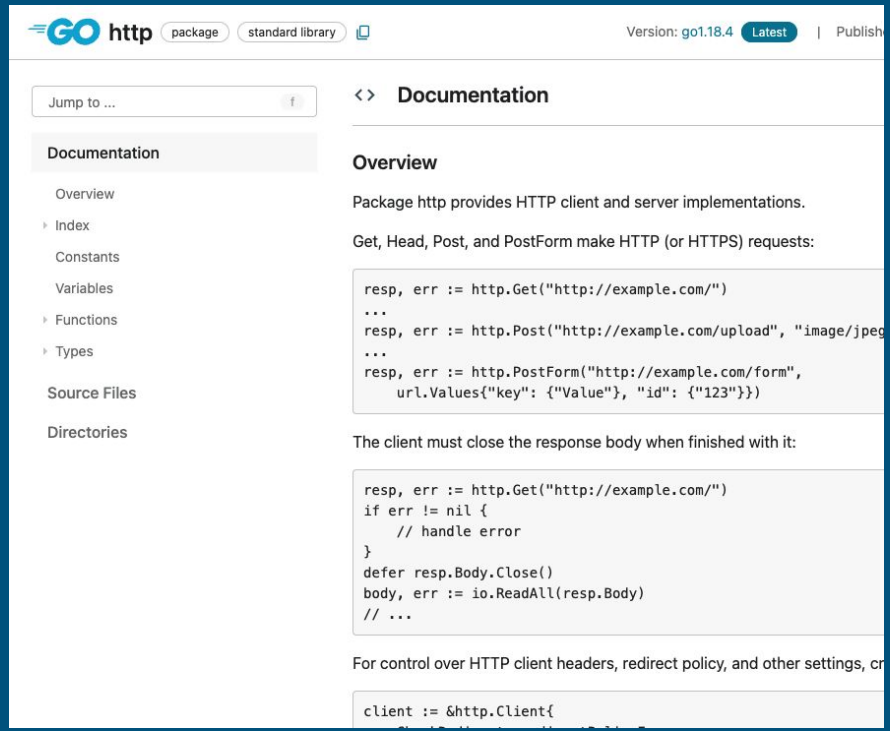
```
var isLoggedIn bool = true  // normal declaration

bigNumber := 30000 // warlus operation
```

```
uint8       the set of all unsigned  8-bit integers (0 to 255)
uint16      the set of all unsigned 16-bit integers (0 to 65535)
uint32      the set of all unsigned 32-bit integers (0 to 4294967295)
uint64      the set of all unsigned 64-bit integers (0 to 18446744073709551615)

int8        the set of all signed  8-bit integers (-128 to 127)
int16       the set of all signed 16-bit integers (-32768 to 32767)
int32       the set of all signed 32-bit integers (-2147483648 to 2147483647)
int64       the set of all signed 64-bit integers (-9223372036854775808 to 9223372036854775807)

float32     the set of all IEEE-754 32-bit floating-point numbers
float64     the set of all IEEE-754 64-bit floating-point numbers

complex64   the set of all complex numbers with float32 real and imaginary parts
complex128  the set of all complex numbers with float64 real and imaginary parts

byte        alias for uint8
rune        alias for int32
```

Go Packages

https://pkg.go.dev/





comma, ok | *comma, error*

- **Es el equivalente a la sentencia -> *try - catch***

```
//  variable, error accion    :=    action

input       ,  _              :=    reader.ReadString('\n')
```

```
//  variable, error accion    :=    action

    _       ,  error          :=    reader.ReadString('\n')

    if error != nil {
        // Handle error...
    }
```

Que se viene para después...

- Build y Memory Management
- Manejo de estructura de datos
- Declaraciones, Funciones, Métodos y **Defer**