Programación concurente en Go

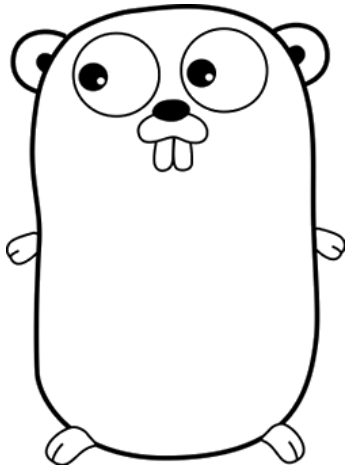
TechMeetup 2014

Agenda

- Introducción a Go
  - Estructura de una aplicación
  - Tipos de datos
- Modelo de concurrencia
- Rutinas
- Sincronización

Conociendo Go

- Lenguaje nacido en Google
- Strongly typed
- Compilado
- Garbage collected
- Performance comparable a C
- Orientado a concurrencia



Un “hola mundo” diferente

- Sin punto y coma final
- El ejemplo muestra:
  - Arrays y slicing
  - Mapas (hash)
  - Asignación múltiple
  - Declaración implícita
  - Uso de biblioteca estándar
- El punto de entrada es main.main()

```go
package main

import (
    "fmt"
    "os"
)

var birthYear = map[string]uint{
    "Einstein": 1879,
    "Newton":   1643,
}

func main() {
    for _, name := range os.Args[1:] {
        if year, present := birthYear[name]; present {
            fmt.Println(name, "nació en", year)
        } else {
            // No tenemos datos para name
            fmt.Println("No se quién es", name)
        }
    }
}
```

Tipos de datos y literales

- Tipos base (lo usual)
- Slices, arrays y mapas
- Estructuras
- Punteros
- Funciones
- Interfaces
- Canales

```
var gender = []string{
    "Male",
    "Female",
    "Custom",
    "Neither",
}

var balanceInBillions = map[string]float64{
    "United":  3.75,
    "Copa":    0.74,
    "Pluna":   -1.1,
}

type Passenger struct {
    name   string
    gender string
    weight float64
}

type Flight map[string]*Passenger

type SeatChooser func(*Passenger) string
```

Declaración y uso de variables

- La memoria siempre está inicializada.
- Declaración implícita con operador :=
- El cero del tipo ya es un valor listo para usar.
- Slices, maps y canales requieren construcción con make (o literal).

```
func firstClass() []*Passenger {
    onboard := make(map[string]*Passenger)

    amy := &Passenger{
        name:   "Amy",
        gender: "Female",
    }
    onboard[amy.name] = amy
    loadOthers(onboard) // maps are ref. types

    firstClass := []*Passenger{}

    for _, p := range onboard {
        if p.gender == "Female" {
            firstClass = append(firstClass, p)
        }
    }
    if p, present := onboard["Gustavo"]; present {
        firstClass = append(firstClass, p)
    }
    return firstClass[:]
}
```

Interfaces

- Concepto: duck typing
- Los métodos:
  - Se declaran con “receivers”
  - Se promueven desde tipos embebidos
- No es OOP
- No hay generics

```
var ErrOutOfSpace = errors.New("out of space")

type Transport interface {
    Capacity() int
}

type Plane struct {
    capacity int
    weight   float64
}

func (p Plane) Capacity() int {
    return p.capacity
}

func Carry(t Transport, people int) error {
    if people > t.Capacity() {
        return ErrOutOfSpace
    }
    // Do stuff
    return nil
}
```

Concurrencia

- Concurrencia no es paralelismo
- Las rutinas (goroutines) se disparan via go f()
- Hay una relación *n*:*m* con threads del sistema
- Go gestiona con un scheduler interno
- No son identificables
- No se puede forzar su finalización

Sincronización y canales

- Opciones:
  - Canales (estándar)
  - Paquete sync
- El canal debe ser creado con make()
- Hay de tipos:
  - Un-buffered
  - Buffered

```
func square(in <-chan int) <-chan int {
    out := make(chan int)

    go func() {
        defer close(out)
        for n := range in {
            out <- n * n
        }
    }()
    return out
}

func printSquared(in <-chan int) {
    squared := square(in)

    for n := range squared {
        fmt.Println(n)
    }
}
```

Estructura y compilación

- Los directorios reflejan jerarquía de paquetes
- Es preciso ubicarlos bajo $GOPATH/src
  - No es estrictamente necesario para main
- No se soporta gestión de dependencias
  - Se puede importar de GitHub, Google Code y otros via go get
  - Ver manejadores de paquetes en http://goo.gl/1vHZ4o
  - Recomiendo especialmente Johnny Deps :)
- Se compila via go build

A trabajar!

- Go: golang.org/dl
- Repositorio: github.com/gkristic/tech-meetup-2014
- Hay dos aplicaciones:
  - rsum: suma SHA1 sobre archivos y directorios
  - fdup: encuentra archivos duplicados
- Vamos a modificarlas para que sean concurrentes

Gracias!