Día tres

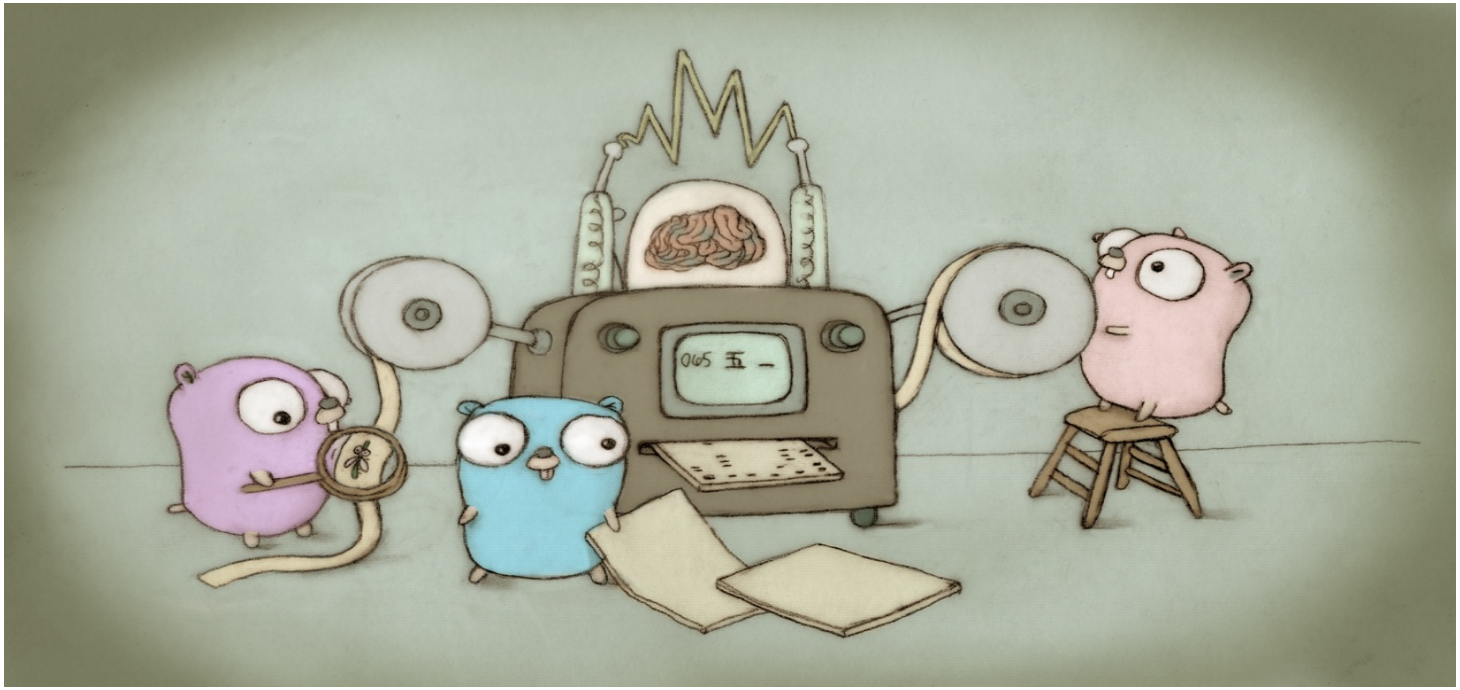
Curso avanzado de Go

6 abril de 2016

Iván Jaimes
Backend developer, Iver

Objetivos

1.1. Del curso



Conocer el lenguaje desarrollando un chat server

Objetivos

2.1. Del día

- Agregar lógica en backend para chat
- Uso de goroutines (concurrencia)

- Pruebas en revel
- Uso de websocket

Vistazo a "container/list"

```
package main

import (
    "container/list"
    "fmt"
)

func main() {
    // Crea una lista y pone número en ella
    l := list.New()
    // Agregamos elementos a la lista
    e4 := l.PushBack(4)
    e1 := l.PushFront(1)
    // Insertamos elementos en la lista
    l.InsertBefore(3, e4)
    l.InsertAfter(2, e1)

    // Recorremos la lista y se imprime el contenido.
    for e := l.Front(); e != nil; e = e.Next() {
        fmt.Println(e.Value)
    }

}
```

Run

Concurrencia

Concurrencia

5.1. Como es en Go

- Un solo código
- Una instancia en ejecución (si, solo una)

- Memoría compartida
- Desarrolladores describen que pasa al mismo tiempo
- Go decide en tiempo de ejecución como pasa
- Concurrencia no es paralellismo - En inglés (https://www.youtube.com/watch?v=cN_DpYBzKso)

Tres elementos principales

- Goroutines (go keyword) describe las unidades que lo ejecutan independientemente.
- Channels (chan type) comunicación entre goroutines
- Select elige entre los channels

Ejemplos

7.1. Multiples goroutine

```
func main() {
    go func() {
        time.Sleep(time.Duration(rand.Intn(30)) * time.Millisecond)
```

```
        fmt.Println("Function 1 says hi")
    }()
    go func() {
        time.Sleep(time.Duration(rand.Intn(30)) * time.Millisecond)
        fmt.Println("Function 2 says hi")
    }()
    time.Sleep(time.Duration(rand.Intn(30)) * time.Millisecond)
    fmt.Println("Main says hi")
}
```

Run

El programa termina cuando la función main termina

WaitGroup

```
func main() {
    wg := sync.WaitGroup{}
    wg.Add(2)
    go func() {
        defer wg.Done() // Run this when the function returns
        time.Sleep(time.Duration(rand.Intn(30)) * time.Millisecond)
        fmt.Println("Function 1 says hi")
    }()
    go func() {
        defer wg.Done()
        time.Sleep(time.Duration(rand.Intn(30)) * time.Millisecond)
        fmt.Println("Function 2 says hi")
    }()
    time.Sleep(time.Duration(rand.Intn(30)) * time.Millisecond)
    fmt.Println("Main says hi")
    wg.Wait() // Wait for both of the goroutines to finish
}
```

Run

Elementos del chat

- Usuario
- Evento
- Chatroom

```
- Suscripción
- Publicación
- Desuscripción
```

Tipos de actualización de mensaje

- Refresh
- Polling
- WebSocket

Manos a la obra

Thank you

6 abril de 2016
Tags: Go (#ZgotmplZ)

Iván Jaimes
Backend developer, Iver

http://iver.mx (http://iver.mx)
http://github.com/ivan-iver/ (http://github.com/ivan-iver/)
@iver14 (http://twitter.com/iver14)