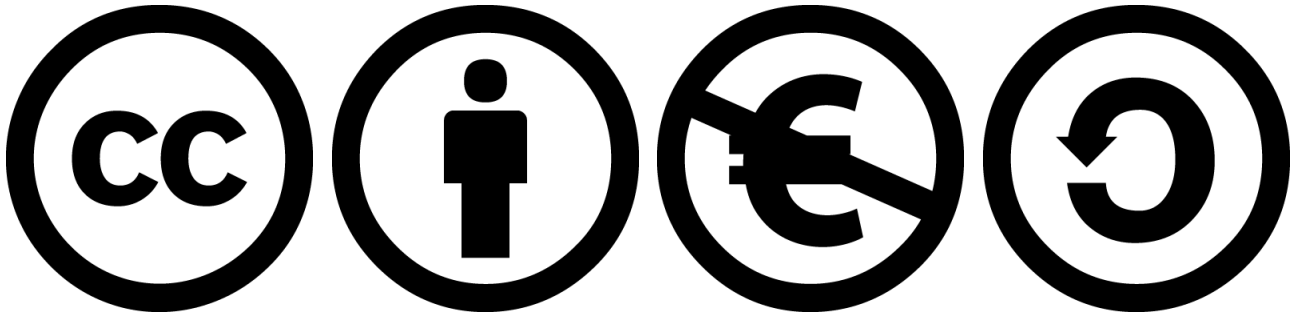
Le langage Go

http://golang.org/

Orléans Tech Talks - Stéphane Legrand - Février 2015

Licence de ce document



Sommaire

- Historique
- Objectifs
- Hello world
- Quelques spécificités
- Exemple API REST
- Conclusion

Historique

- Projet interne Google à partir de 2007
- Open source depuis 2009
- Notamment Rob Pike et Ken Thompson

- Aujourd’hui, version 1.4

Objectifs

- Souplesse d'un langage interprété
- Efficacité et sécurité d'un langage compilé
- Moderne : réseau et multi-processeurs
- Rapide : compilation et exécution
- Portable : FreeBSD (v8 et sup.), Linux (noyau 2.6.23 et sup.), Mac OS X (v10.6 et sup.), Windows (XP et sup.)

Hello world

```
package main

import "fmt"

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

```
% go run hello.go
hello world
% go build hello.go
% ./hello
hello world
```

Typage statique

```
package main
type t1 string
type t2 string
func f (v1 t1, v2 t2) (s string) {
  s = v1 + v2
  return
}
func main() {
  f("v1", "v2")
}
```

```
% go build typage.go
./typage.go:5: invalid operation: v1 + v2 (mismatched types t1
and t2)
```

Structures

Collection de champs regroupés sous un type

```
package main
import "fmt"
type utilisateur struct {
  identifiant string
  groupe int
}
func main() {
  fmt.Println(utilisateur{identifiant: "Toto", groupe: 42})
}

% go run structure.go
{Toto 42}
```

Méthodes

```
func (u utilisateur) incGroupeVal() { u.groupe++ }
func (u *utilisateur) incGroupePoint() { u.groupe++ }
func main() {
  u := utilisateur{identifiant: "Toto", groupe: 42}
  u.incGroupeVal()
  fmt.Println(u)
  u.incGroupePoint()
  fmt.Println(u)
}

% go run methode.go
{Toto 42}
{Toto 43}
```

Interfaces / Définition

Interface = à la fois

- un ensemble de méthodes
- un type

```
type Animal interface {
    Parle() string
}
```

Interfaces / OK

```
type Chien struct {}
func (c Chien) Parle() string { return "Wouf !" }
type Chat struct {}
func (c Chat) Parle() string { return "Miaou !" }
func main() {
    animaux := []Animal{Chien{}, Chat{}}
    for _, animal := range animaux { fmt.Println(animal.Parle()) }
}

% go build interface.go                    // COMPILATION OK
% ./interface
Wouf !
Miaou !
```

Interfaces / KO

```
type Chien struct {}
func (c Chien) Parle() string { return "Wouf !" }

type Chat struct {}           // chat muet, pas de méthode Parle()

func main() {
    animaux := []Animal{Chien{}, Chat{}}
    for _, animal := range animaux { fmt.Println(animal.Parle()) }
}

% go build interface.go
./interface.go:16: cannot use Chat literal (type Chat) as type
Animal in array element:
Chat does not implement Animal (missing Parle method)
```

Programmation concurrente

```
package main
import ("fmt"; "time"; "math/rand")
func f (i int, msg string) {
    rnd := rand.Intn(2000)
    time.Sleep(time.Duration(rnd) * time.Millisecond)
    fmt.Println(i, ": ", msg)
}
func main() {
    for i := 0; i < 100000; i++ {
        go f(i, "Go !")
    }
    var input string // attente appui d'une touche
    fmt.Scanln(&input)
}
```

Outils intégrés / Tests

```
// fichier interface_test.go
package main
import "testing"

func TestChienParle(t *testing.T) {
    chien := Chien{}
    parole := chien.Parle()
    if parole != "Wouf !" {
        t.Error("S'attendait à Wouf, parole = ", parole)
    }
}

% go test interface
ok      interface  0.002s
```

Outils intégrés / Divers

- Comprend les gestionnaires de version

```
% go get github.com/julienschmidt/httprouter
```

- Indentation automatique du code

```
% go fmt …
```

- Générateur de documentation à partir des commentaires

```
% godoc ...
```

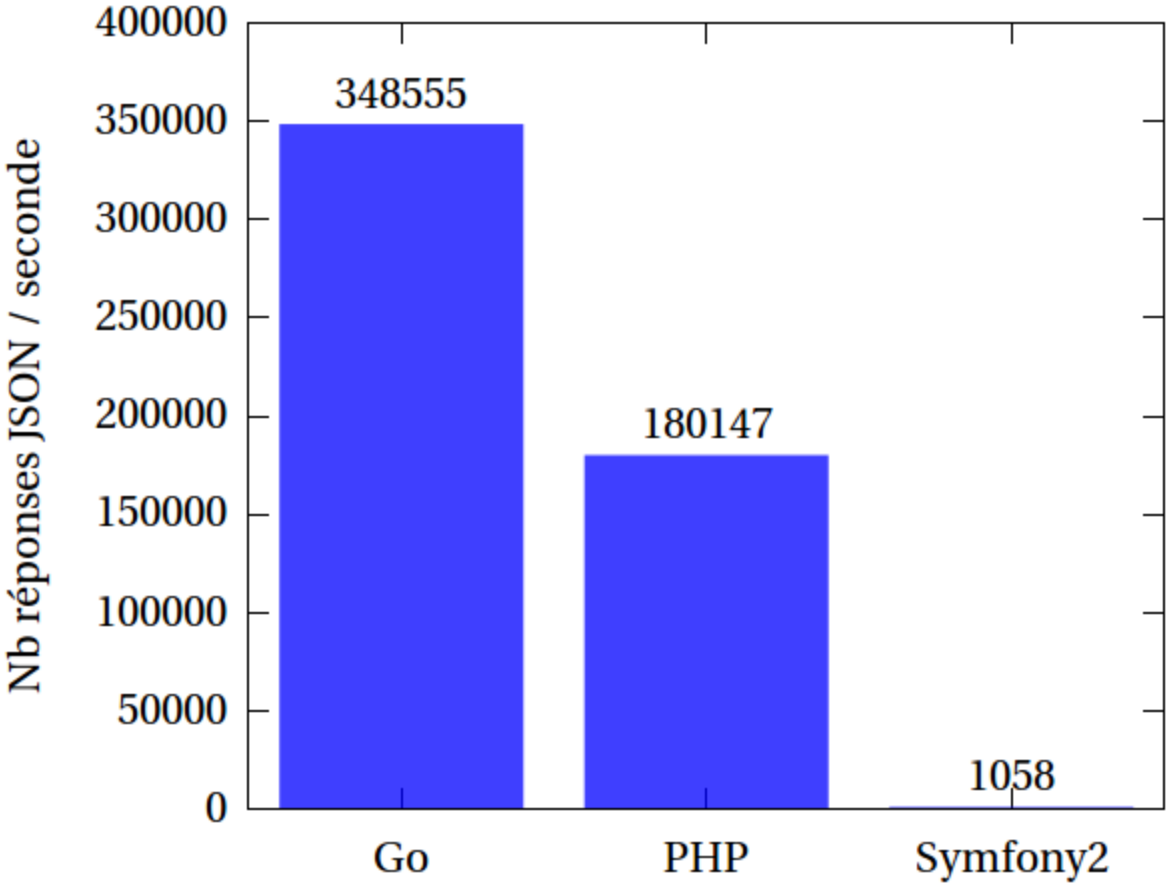
Rapidité de compilation

Pas trouvé de benchmarks (désolé)...

Mais il suffit de l'essayer
pour le constater.

VRAIMENT, VRAIMENT RAPIDE !

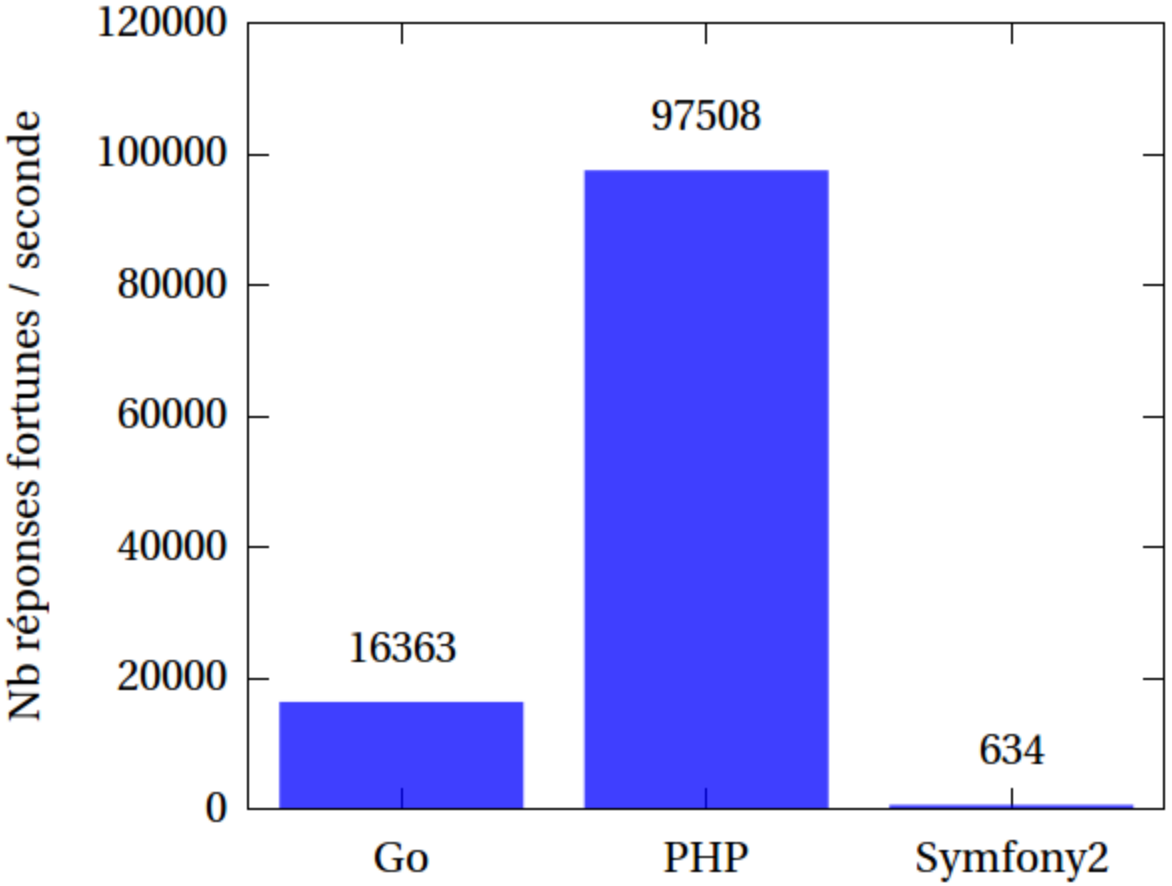
Rapidité d'exécution / 1



http://www.techempower.com/benchmarks/#section=data-r9&hw=peak&test=json&l=dfk

http://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=go&lang2=php

Rapidité d'exécution / 2



http://www.techempower.com/benchmarks/#section=data-r9&hw=peak&test=fortune&l=dfk

Richesse librairie standard

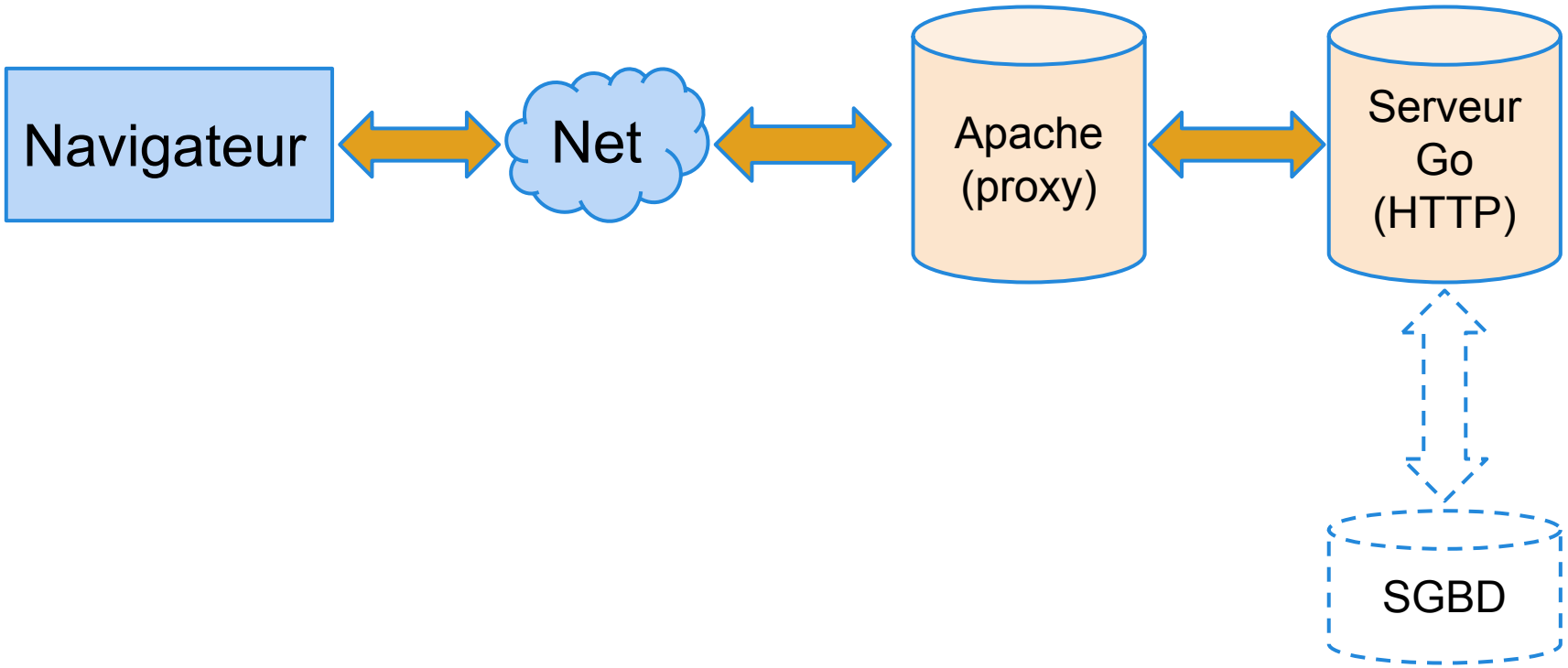
- (dé)compression : gzip, bzip2, zlib…
- (dé)chiffrement : aes, des, hmac, md5, rsa, sha512…
- bases de données SQL (drivers tiers).
- encodage : base64, csv, json, xml…
- serveur HTTP
- expressions régulières
- opérations sur dates et heures
- Unicode
- ...

Bizarreries...

- Pas d'exceptions (mais `v,err := f()`)
- Pas de type générique
- Erreur de pointeur nul possible
- Pas de librairie dynamique
- Pas d'héritage

http://tmikov.blogspot.fr/2015/02/you-dont-like-googles-go-because-you.html

Exemple REST / Architecture



Exemple REST / Objectif

Si le serveur reçoit une requête GET

/nom/toto

Doit renvoyer une chaîne JSON

{"Nom":"toto"}

Exemple REST / Code - partie 1

```
package main

import (
  // installation : go get github.com/julienschmidt/httprouter
  "github.com/julienschmidt/httprouter"
  "encoding/json"
  "log"
  "net/http"
)

type utilisateur struct {
  Nom   string // exporté pour JSON
}
```

Exemple REST / Code - partie 2

```
func main() {
  router := httprouter.New()
  router.GET("/nom/:nom", nom)
  log.Fatal(http.ListenAndServe("localhost:8080", router))
}
```

Exemple REST / Code - partie 3

```
func nom(w http.ResponseWriter, r *http.Request, ps httprouter.Params)
{
  u := utilisateur{}
  u.Nom = ps.ByName("nom")
  j, err := json.Marshal(u)
  if err == nil {
    w.Header().Set("Content-Type", "application/json")
    w.Write(j)
  } else {
    log.Fatal(err)
  }
}
```

Exemple REST / Exécution

```
% go build rest.go
% ./rest


% curl 'http://localhost:8080/nom/toto'
{"Nom":"toto"}
% curl 'http://localhost:8080/nom/titi'
{"Nom":"titi"}
```

Exemple REST / Benchmark

```
% siege -q -b -t 60S 'http://localhost:8080/nom/toto'
Transactions:              449160 hits
Availability:              100.00 %
Elapsed time:              59.09 secs
Data transferred:          6.00 MB
Response time:             0.00 secs
Transaction rate:          7601.81 trans/sec
Throughput:                    0.10 MB/sec
Concurrency:               14.81
Successful transactions:449160
Failed transactions:    0
Longest transaction:    0.02
Shortest transaction:   0.00
```

Conclusion

- Pragmatique !
- Beaucoup de potentiel
- Utiliser “golang” pour recherches sur internet

Merci

Merci de votre attention

Questions ?