GOLANG UNITED

Introduction lecture

Golang United Основные разработчики языка:

Ken Thompson

- Designed and implemented the original **Unix OS**
- One of the creators and early developers of the **Plan 9 OS**
- Invented the **B programming language**
- contributions included his work on **regular expressions**
- Worked on **UTF-8**

Rob Pike

- member of the **Unix** team
- Involved in the creation of the **Plan 9**
- **Sam** and **Acme** text editors
- Co-author of **The Practice of Programming** and **The Unix Programming Environment** books.
- Co-creator of **UTF-8**

Robert Griesemer

- The Google's **V8** JavaScript engine
- The **Sawzall** language
- The **Java HotSpot** virtual machine
- The **Strongtalk** system.

Ребята контрибьютили в Unix, создавали свои ЯП, в то время это было модно — считает что их опыт это хорошо.

Golang United

12 years of jorney

2009 Public release

Golang become open-source project

2015 1.5 release

In 1.4 and 1.5 version internals of the language was refactored, all language tooling was rewritten in Go.

2022 1.18 release

Generic types and fuzzing testing

2007 Development started

Development started as internal project at Google

2012 v1.0 release

Golang released v1.0, language team promised compatibility of all 1.* versions

2018 v1.11 release

Go modules release, this change improved dependency management a lot

Go в пределах мажорного релиза обратно совместим; самокомпилируем.

Why Golang is so good?

Typing, Compilation, Concurrency, Standard library

async | thread | многопоточность?

- Строгая типизация
- Многословность (?), выразительность
- Быстр, легок, хорош на server side
- Хорош в многопоточность за счет

goroutine - некая высокоуровневая абстракция над тредами.

Golang United

No legacy.

Golang is used a lot for cloud native development

Golang is simple and fast

Concurrent by design

На go написаны:

- k8s
- prometheus
- grafana

and etc...

Golang United

Huge standart library(https://pkg.go.dev/std):

- https://pkg.go.dev/compress@go1.17.6 – compression via different algorithms
- https://pkg.go.dev/crypto@go1.17.6 – supports many cryptographic algorithms
- https://pkg.go.dev/encoding@go1.17.6 – supports many encoding formats
- https://pkg.go.dev/net/http@go1.17.6 – built-in http server with http1.1/http2 support
- https://pkg.go.dev/html@go1.17.6 – built-in html template engine
- https://pkg.go.dev/text@go1.17.6 – built-in text template engine
- https://pkg.go.dev/database/sql@go1.17.6 – built-in sql library

The key differences of Golang

Type system, Interfaces, Concurrency

На самом деле, слайды $\rightarrow$ на текущем этапе немного бесполезна, наверное их цель придать inspiration

Golang United

Golang is explicit language

Code is the main documentation of itself

By default, all parameters passed by value not a reference

Using value type in most cases faster then use references

Code generation over generic programming (So far)

Data structures is simple, there is no such Collection framework (so far)

↑скоро добавят

Golang United

Strong typing — every variable in golang has type

```go
var d1 int64
var d2 int32
var d3 int
```

Type system

```go
var (
    d1 int64
    d2 int32
)

fmt.Println(d1 == d2)
```

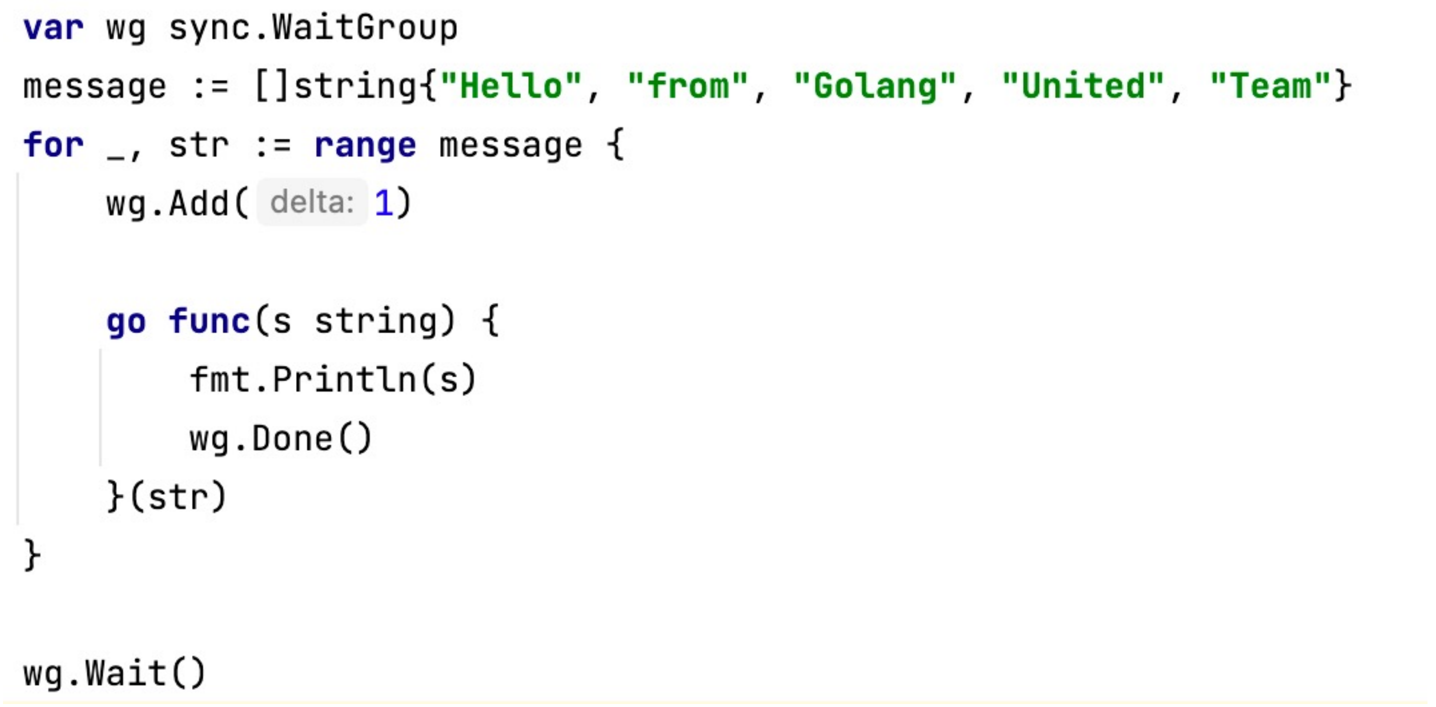
Compilator do the work for you

```
# github.com/burov/snipets/webserver
./main.go:22:17: invalid operation: d1 == d2 (mismatched types int64 and int32)
```

Компилятор помогает отлавливать ошибки к-эп :)

Golang United

Concurrency dramatically simple



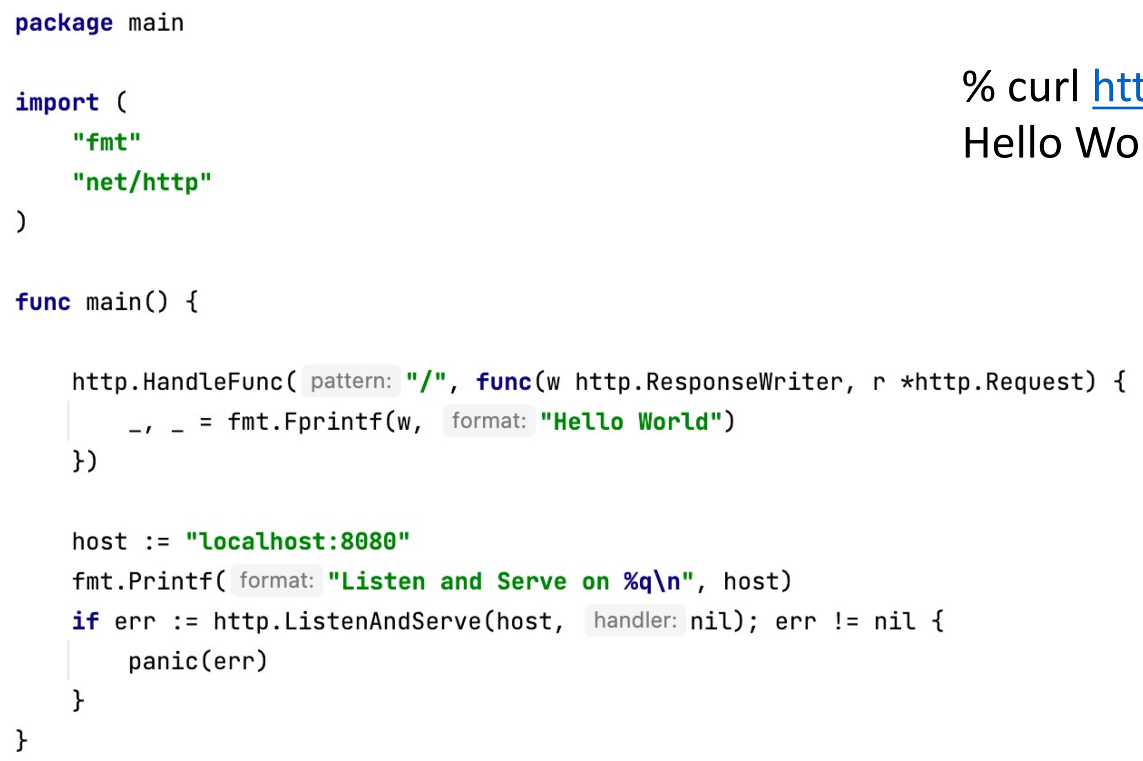
```go
var wg sync.WaitGroup
message := []string{"Hello", "from", "Golang", "United", "Team"}
for _, str := range message {
    wg.Add( delta: 1)

    go func(s string) {
        fmt.Println(s)
        wg.Done()
    }(str)
}

wg.Wait()
```

Output:

```
Team
Hello
United
Golang
from
```

Golang United

An example of simple web-server



```go
package main

import (
    "fmt"
    "net/http"
)

func main() {

    http.HandleFunc( pattern: "/", func(w http.ResponseWriter, r *http.Request) {
        _, _ = fmt.Fprintf(w,  format: "Hello World")
    })

    host := "localhost:8080"
    fmt.Printf( format: "Listen and Serve on %q\n", host)
    if err := http.ListenAndServe(host,  handler: nil); err != nil {
        panic(err)
    }
}
```

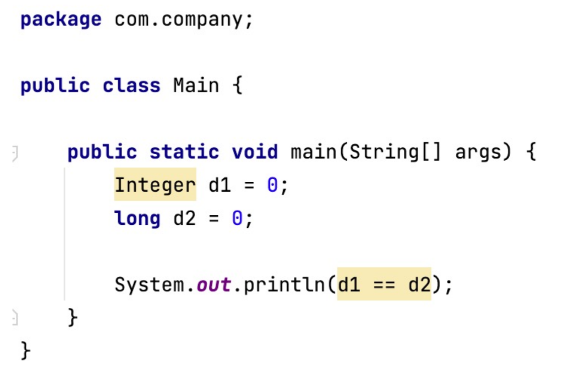
```
% curl http://localhost:8080/
Hello World
```

Много букв, но зато мы понима-
ем, что происходит. Ну, ок

Golang comparison

Golang United

```
Strong typing – every variable in golang has type
```



```
package com.company;

public class Main {

    public static void main(String[] args) {
        Integer d1 = 0;
        long d2 = 0;

        System.out.println(d1 == d2);
    }
}
```

Output

```
true
```

Тут предлагается вспомнить приколы из JavaScript

Golang United

Strong typing – every variable in golang has type

```go
var d1 int64
var d2 int32
var d3 int
```

Type system

```go
var (
    d1 int64
    d2 int32
)

fmt.Println(d1 == d2)
```

Compilator do the work for you

```
# github.com/burov/snipets/webserver
./main.go:22:17: invalid operation: d1 == d2 (mismatched types int64 and int32)
```

Golang United

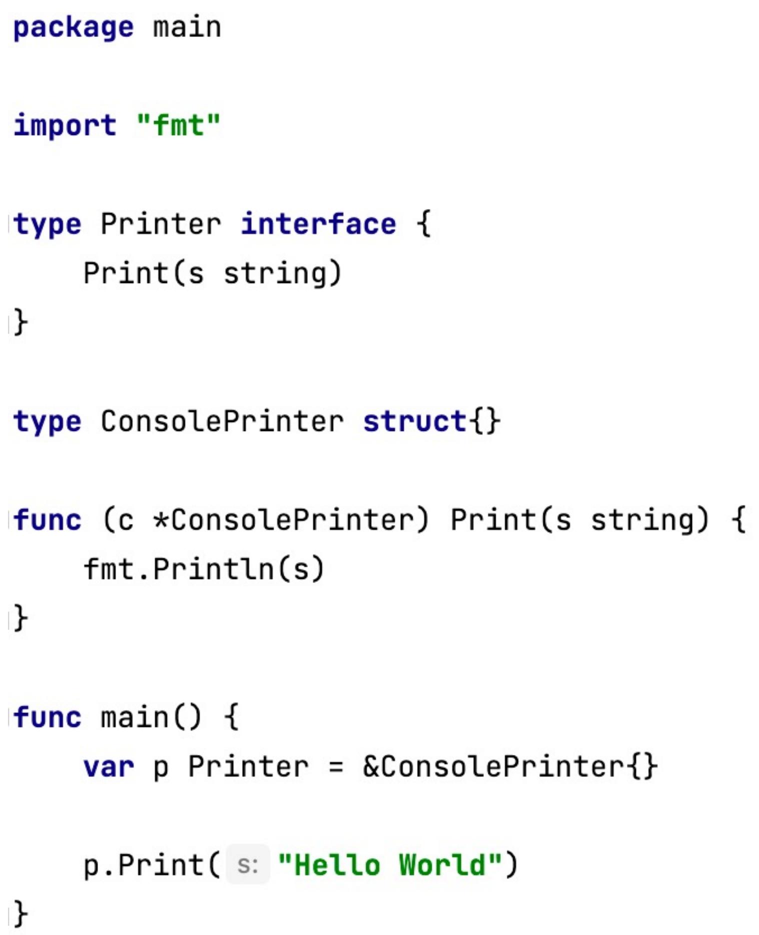
Java:

- Types system is very complex, some of the decisions have to be made, due to the initial design weakneses

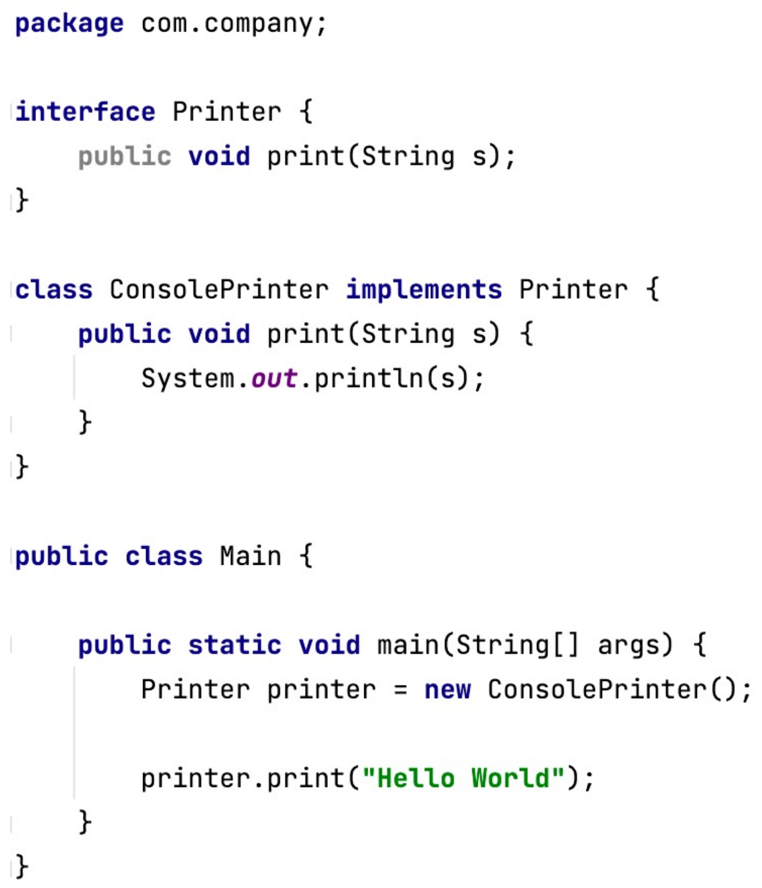
Golang:

- Type system realy simple and explicit, there no hacks in it

Golang United



```go
package main

import "fmt"

type Printer interface {
    Print(s string)
}

type ConsolePrinter struct{}

func (c *ConsolePrinter) Print(s string) {
    fmt.Println(s)
}

func main() {
    var p Printer = &ConsolePrinter{}

    p.Print( s: "Hello World")
}
```



```java
package com.company;

interface Printer {
    public void print(String s);
}

class ConsolePrinter implements Printer {
    public void print(String s) {
        System.out.println(s);
    }
}

public class Main {

    public static void main(String[] args) {
        Printer printer = new ConsolePrinter();

        printer.print("Hello World");
    }
}
```

Golang United

Java:

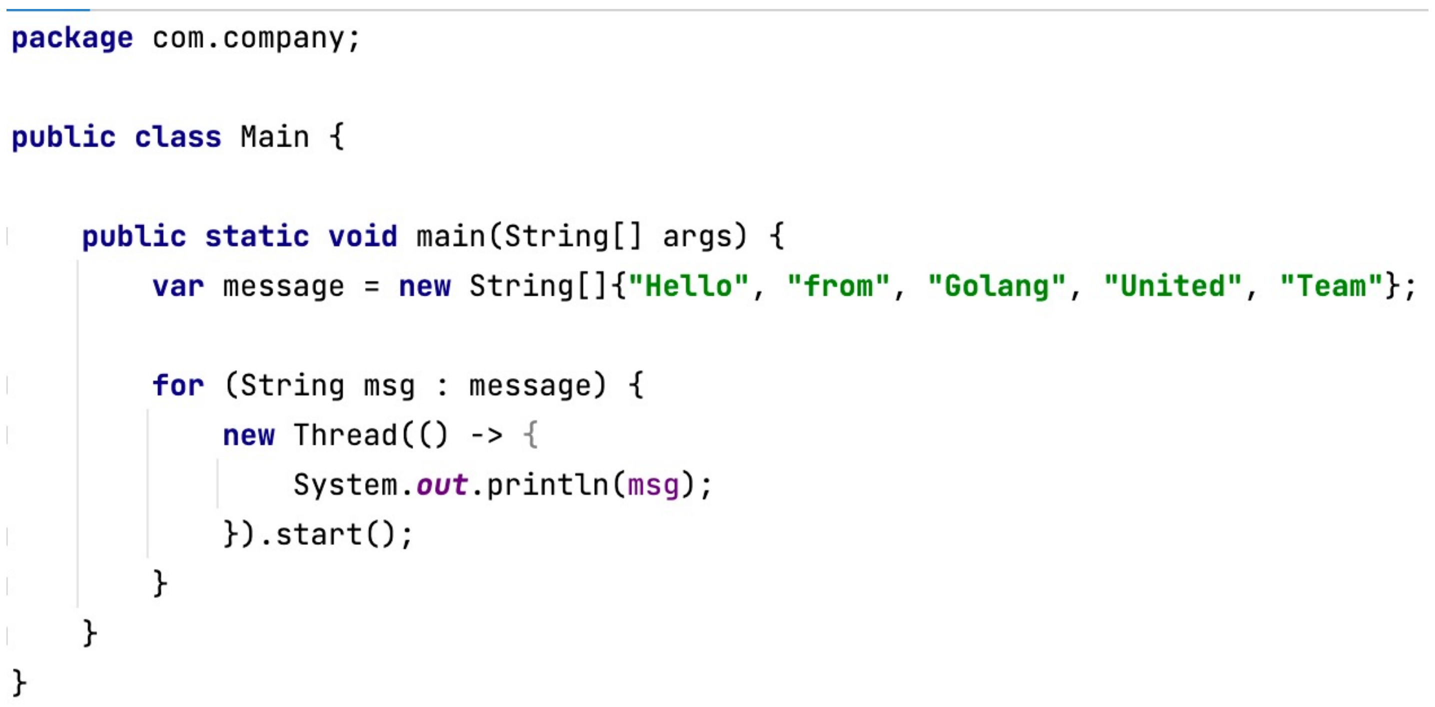
- Interface should be defined by producer
- Interface should be explicitly implemented by Class

Golang:

- Interface might be defined by consumer or producer
- Interface implicitly implemented if type has all the methods defined in interface (Duck typing)

Golang United

Concurrency in Java



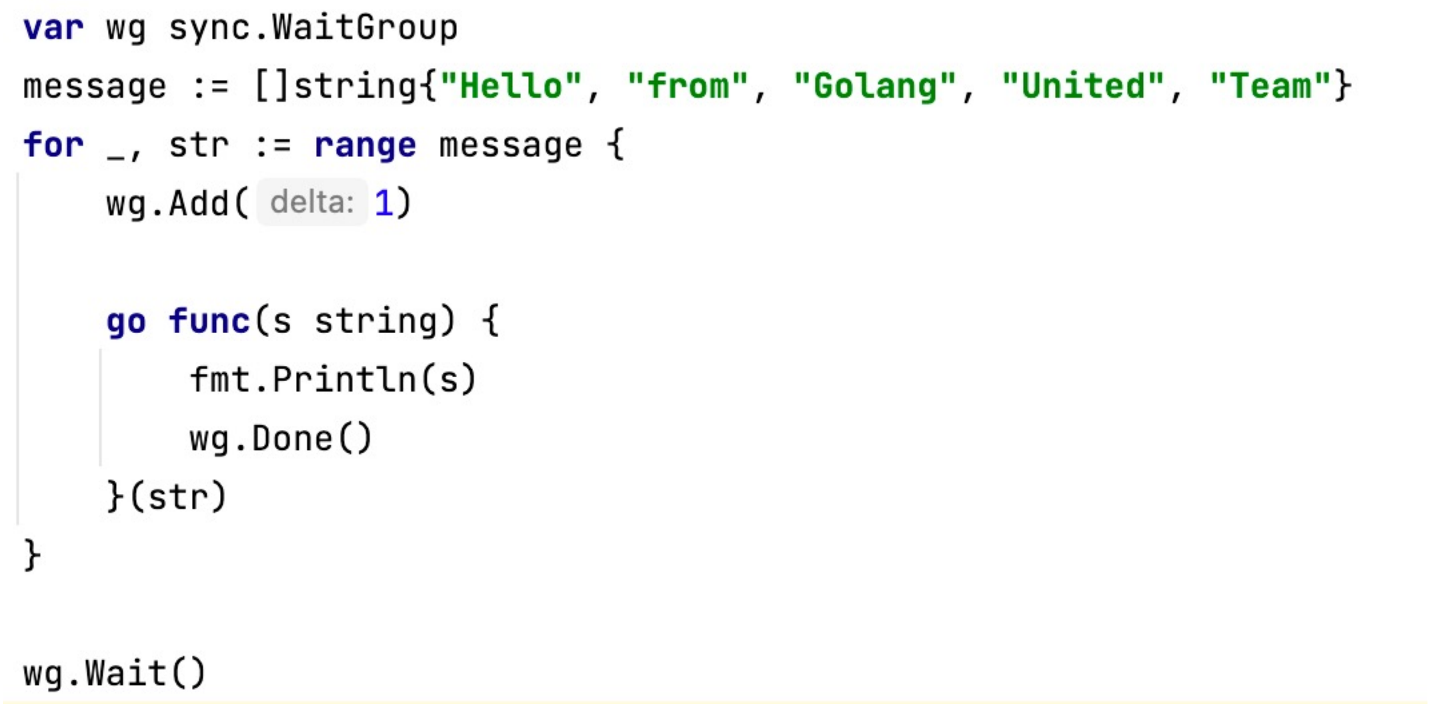
```java
package com.company;

public class Main {

    public static void main(String[] args) {
        var message = new String[]{"Hello", "from", "Golang", "United", "Team"};

        for (String msg : message) {
            new Thread(() -> {
                System.out.println(msg);
            }).start();
        }
    }
}
```

Output:

```
Team
Hello
United
Golang
from
```

Golang United

Concurrency dramatically simple



```go
var wg sync.WaitGroup
message := []string{"Hello", "from", "Golang", "United", "Team"}
for _, str := range message {
    wg.Add( delta: 1)

    go func(s string) {
        fmt.Println(s)
        wg.Done()
    }(str)
}

wg.Wait()
```

Output:

```
Team
Hello
United
Golang
from
```

Golang United

Java:

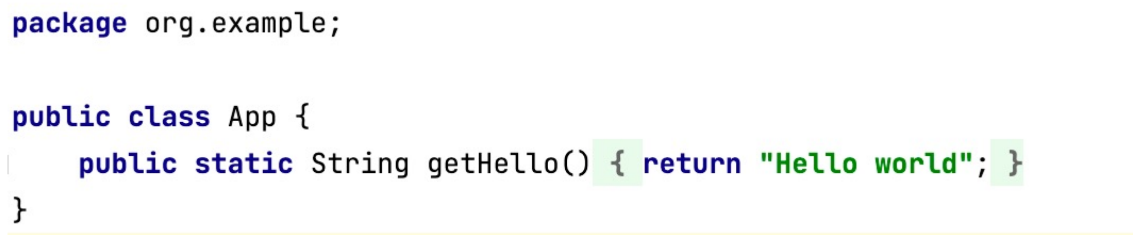
- Create system thread for each Thread object
- Static stack
- Communication by shared memory

Golang:

- Go routines is a local object, no system thread creation
- Dynamic stack, each go routine can have stack size up to a few gigabytes
- Communication by shared memory or built-in messaging (channels)

Golang United

An example of simple web-server in Java



```java
package org.example;

public class App {
    public static String getHello() { return "Hello world"; }
}
```

```
% curl http://localhost:8080/
Hello World
```

Golang United

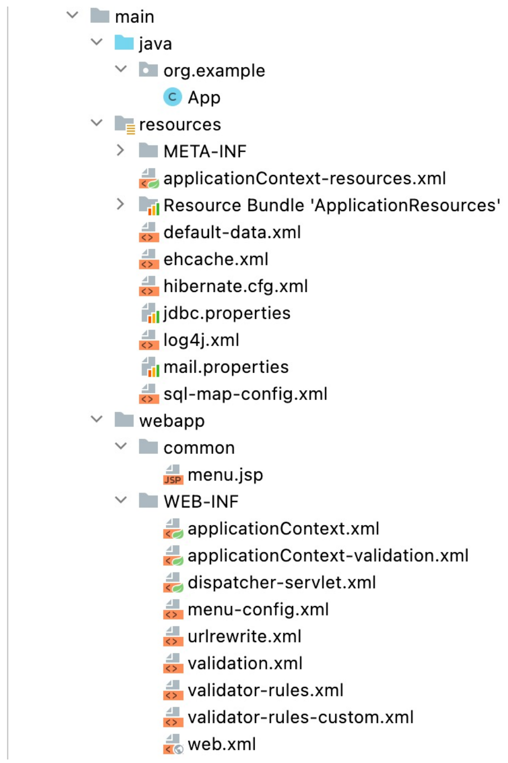
An example of simple web-server

```go
package main

import (
    "fmt"
    "net/http"
)

func main() {

    http.HandleFunc( pattern: "/", func(w http.ResponseWriter, r *http.Request) {
        _, _ = fmt.Fprintf(w,  format: "Hello World")
    })

    host := "localhost:8080"
    fmt.Printf( format: "Listen and Serve on %q\n", host)
    if err := http.ListenAndServe(host,  handler: nil); err != nil {
        panic(err)
    }
}
```

```
% curl http://localhost:8080/
Hello World
```

Golang United

Web project configuration in Java



```
main
  java
    org.example
      App
  resources
    META-INF
    applicationContext-resources.xml
    Resource Bundle 'ApplicationResources'
    default-data.xml
    ehcache.xml
    hibernate.cfg.xml
    jdbc.properties
    log4j.xml
    mail.properties
    sql-map-config.xml
  webapp
    common
      menu.jsp
    WEB-INF
      applicationContext.xml
      applicationContext-validation.xml
      dispatcher-servlet.xml
      menu-config.xml
      urlrewrite.xml
      validation.xml
      validator-rules.xml
      validator-rules-custom.xml
      web.xml
```

Golang United

Golang:

- Built-in server
- Additional configuration isn’t required
- No external depedencies
- Build just with plain compiler

Java:

- Additional server required before servlets container
- Required configuration for frameworks
- Servlets containers and JavaEE libs required
- Gradle or Maven requied to build project