



21.01.2020

Berlin / GoDays 2020 Meetup

Using Go coming from Java

INNOQ

The logo for InnoQ, featuring the word "INNOQ" in a bold, sans-serif font. The "INNO" part is white, and the "Q" is red. The logo is set against a dark blue background.

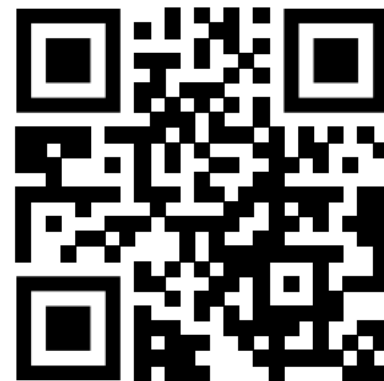
INNOQ



Philipp Haußleiter

Senior Consultant
innoQ Deutschland GmbH

philipp.haussleiter@innoq.com
@phaus

The logo for InnoQ, featuring the word "INNOQ" in a bold, sans-serif font. The "INNO" part is white, and the "Q" is red. The logo is set against a dark blue rectangular background.

Philipp Haußleiter

Senior Consultant
innoQ Deutschland GmbH

philipp.haussleiter@innoq.com
@phaus

coming from Java...

Typical history

- Java backend developer
- JPA/EJB
- Spring (not Boot)
- Play(1/2)
- Monoliths

Why Go?

Typical history

- Java backend developer
- JPA/EJB
- Spring (not Boot)
- Play(1/2)
- Monoliths
- Dev-(OPs)
- Tools/Clients
- Software Distribution
- Microservices

My Reasons to Consider Go Coming from Java

May 7, 2019 - 7 min



[Thanks for stopping by! Follow me on Twitter!](#)

I hate listicles, and I am highly hoping that this doesn't become one of them. Yet, I somehow felt the need to write down the reasons why I am giving Go a second (or maybe, third 🤔) chance. To the Java developers out there, searching for a new weapon of choice, I am hoping that this will give you a bit of perspective. This is not a description of Go-specific features. For those, readers can find plenty of information online.



Preslav Rachev



Go focuses on “less is more”

Go apps are fast and small

The tooling and the standard library are outstanding

The community

present 10 things
in 30 minutes

9 Topics

Server

Testing

Plugins

Client

Updates

X-Compiling

Tooling

Build-Flags

Building

9 Topics

Server

Client

Tooling

Testing

Updates

Build-Flags

Plugins

X-Compiling

Building

Add-On



Middleware

UI

Assets



Server

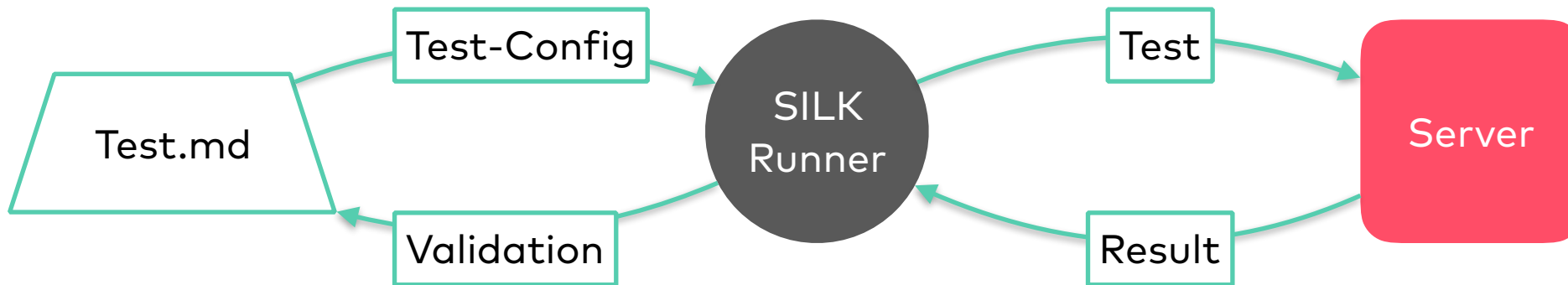
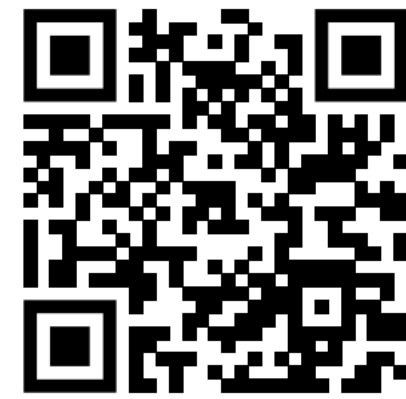
1

Testing

Problem

- I need to test my Applications
- I want to have a documentation of my APIs
- Sometimes, people doing the testing, are no developers

Silk - document-driven RESTful API testing.



example.silk.md — Edited ▾

47 Words

Comments

POST /comments

Create a new comment.

- * Content-Type: "application/json"
- * Accept: "application/json"

Include the ``name`` and ``comment`` text in the body:

```
...  
{  
  "name": "Mat",  
  "comment": "Writing tests is easy"  
}  
...
```

===

Example response

- * Status: 201
- * Content-Type: "application/json"

```
...  
{  
  "id": "123",  
  "name": "Mat",  
  "comment": "Writing tests is easy"  
}  
...
```

Comments

POST /comments

Create a new comment.

- Content-Type: "application/json"
- Accept: "application/json"

Include the `name` and `comment` text in the body:

```
{  
  "name": "Mat",  
  "comment": "Writing tests is easy"  
}
```

Example response

- Status: 201
- Content-Type: "application/json"

```
{  
  "id": "123",  
  "name": "Mat",  
  "comment": "Writing tests is easy"  
}
```

```
package project_test
```

```
import (  
    "testing"  
    "github.com/matryer/silk/runner"  
)
```

```
func TestAPIEndpoint(t *testing.T) {  
    // start a server  
    s := httptest.NewServer(yourHandler)  
    defer s.Close()  
  
    // run all test files  
    runner.New(t, s.URL).RunGlob(filepath.Glob("../testfiles/failure/*.silk.md"))  
}
```

2

Plugins

Problem

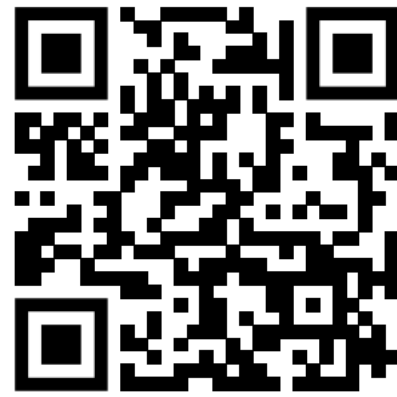
- I want to have my Application expandable
- I want to be able to install/reload/update plugins during runtime
- Third parties should be able to also provide plugins

hashicorp/go-plugin



- Go (golang) plugin system over RPC
- initially created for Packer, it is additionally in use by Terraform, Nomad, and Vault.
- gRPC-based plugins enable plugins to be written in any language
- dynamic loading

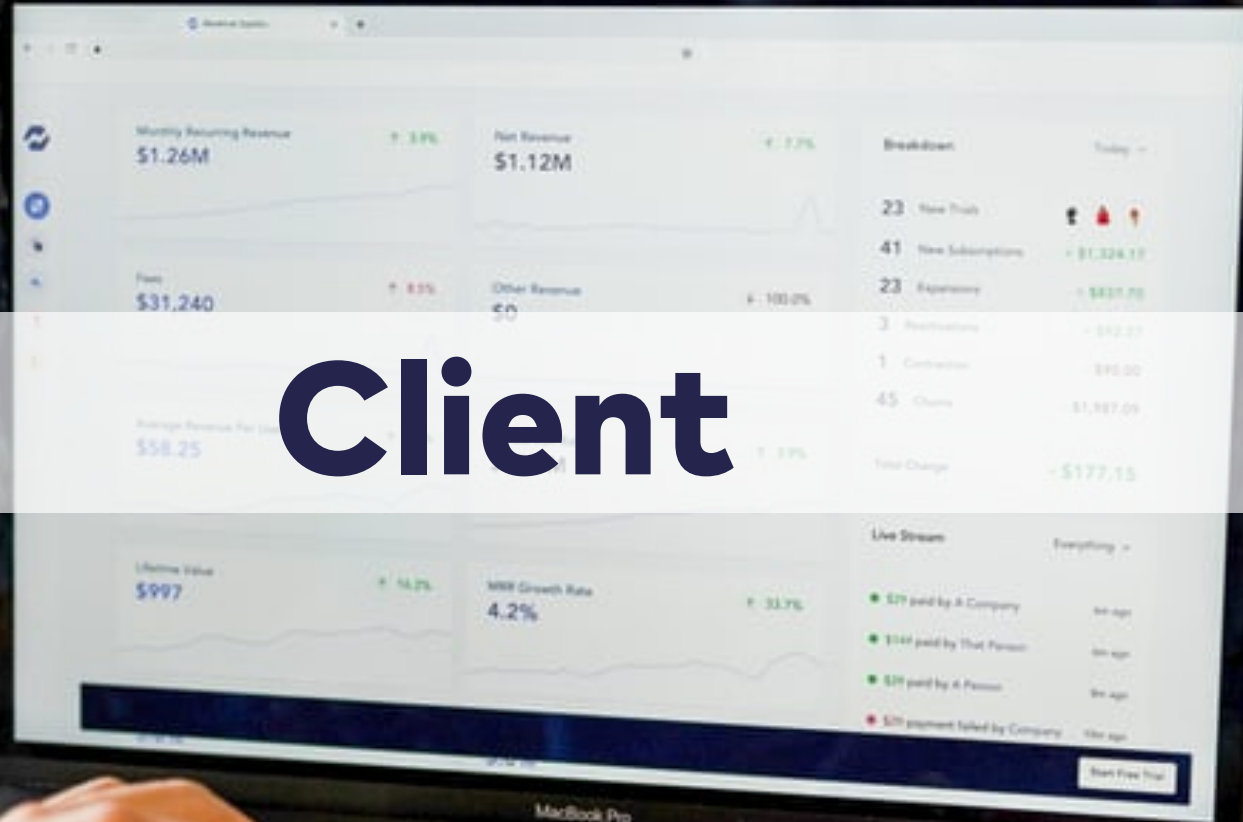
robertkrimen/otto



- A JavaScript interpreter in Go (golang)
- Use Go functions in Javascript
- dynamic loading

```
vm := otto.New()
vm.Run(`
  abc = 2 + 2;
  console.log("The value of abc is " + abc); // 4
`)
```

Client



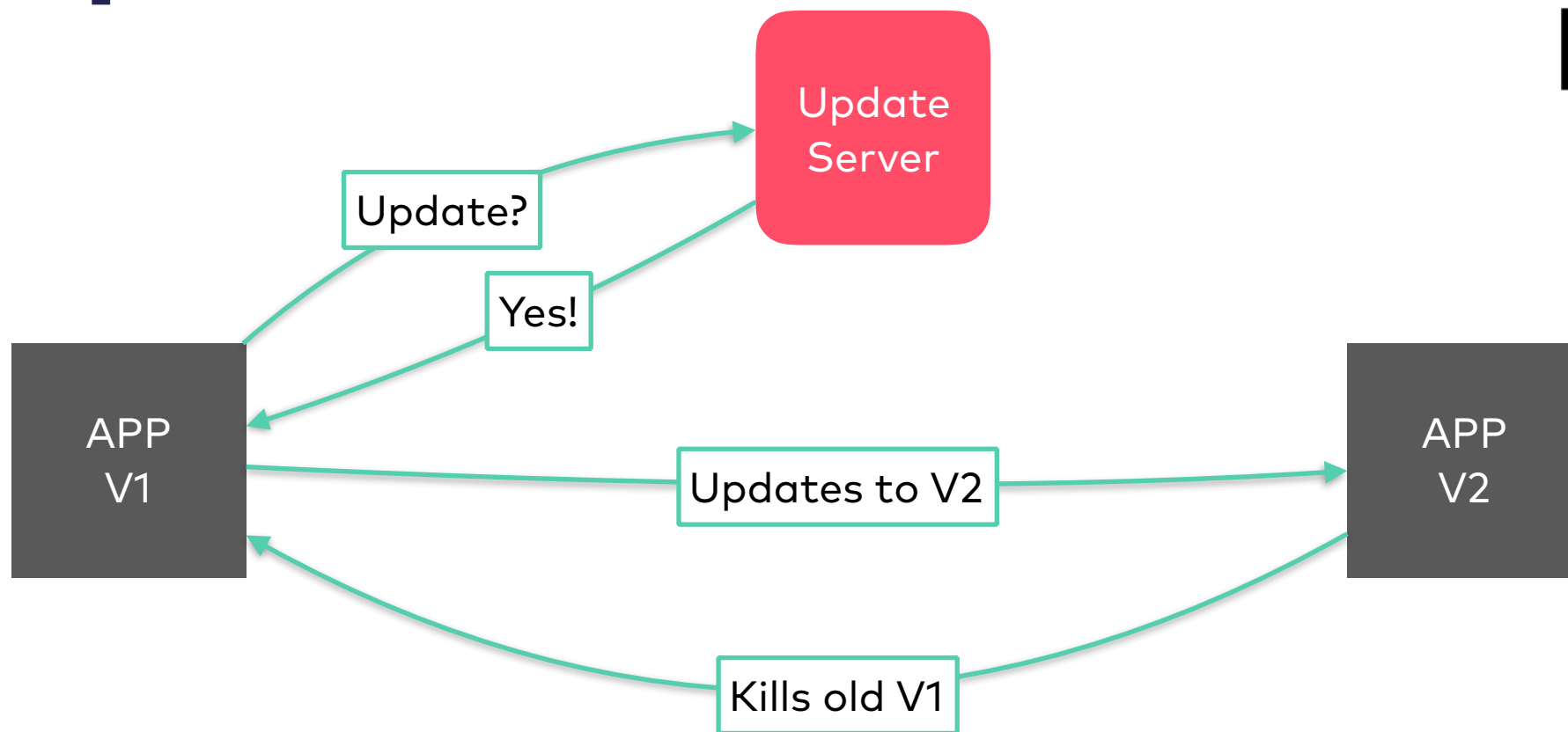
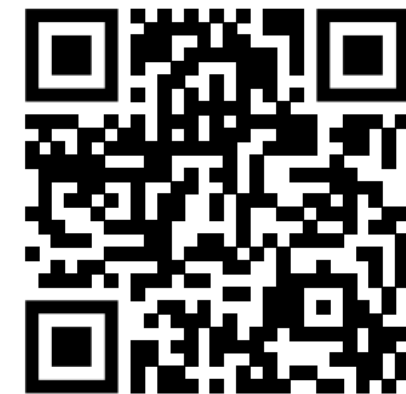
3

Updates

Problem

- You are building a tool/script
- You distribute the tool
- Now it is "out there"
- How do you update?

Go-update



```
import (  
    "fmt"  
    "net/http"  
  
    "github.com/inconshreveable/go-update"  
)  
  
func doUpdate(url string) error {  
    resp, err := http.Get(url)  
    if err != nil {  
        return err  
    }  
    defer resp.Body.Close()  
    err := update.Apply(resp.Body, update.Options{})  
    if err != nil {  
        // error handling  
    }  
    return err  
}
```

4

X-Compiling

Problem

- I want to have my clients on all platforms
- I want to have it build and packaged automatically
- I want to have one Code-Base


```
# Build
```

```
declare -a TARGETS=(darwin linux solaris freebsd, windows)
```

```
for target in ${TARGETS[@]} ; do
```

```
    export GOOS=${target}
```

```
    export GOARCH=amd64
```





```
    output="client-${target}"
```

```
    echo "Building for ${target}, output bin/${output}"
```


```
    go build -o bin/${output}
```

```
done
```





Build

- ✓ linux-binary32 
- ✓ linux-binary64 
- ✓ mac-binary 
- ✓ windows-binary 

Optimize





- ✓ optimize-clients 

Package

- ✓ linux-app-image 
- ✓ mac-dmg 
- ✓ mac-dmg-64bit 
- ✓ windows-app-i... 

Deploy

- ✓ collect 

 client-i686.ApplImage	1.07 MB
 client-setup.exe	3.94 MB
 client-x86_64.ApplImage	1.12 MB
 client.dmg	2.29 MB
 client.x86_64.dmg	1.26 MB



Tooling

5

Build-Flags

Problem

- Parts of your code are platform depended
- You want to have as much abstraction as possible
- You want to have only one source tree

Build Constraints



```
(linux OR darwin) AND 386  
// +build linux darwin  
// +build 386
```

To build a file only when using cgo, and only on Linux and OS X:
// +build linux,cgo darwin,cgo

Build Constraints

If a file's name, after stripping the extension and a possible `_test` suffix, matches any of the following patterns:

- *_GOOS
- *_GOARCH
- *_GOOS_GOARCH

info

- |-- info_darwin.go
- |-- info_linux.go
- `-- info_windows.go

Build Constraints

If a file's name, after stripping the extension and a possible `_test` suffix, matches any of the following patterns:

- *_GOOS
- *_GOARCH
- *_GOOS_GOARCH

```
info
|-- info_darwin.go
|-- info_linux.go
`-- info_windows.go
```

```
import (
    ...
    "collector/client/info"
    ...
)

func main() {
    fmt.Printf("Result: %s\n", info.Analyse())
}
```

```
./collector
windows $ Result: running windows
macOS $ Result: running darwin
linux $ Result: running linux
```

6

Building

Problem

- I want to have fast deployment
- I would like to have Apps following the Single Responsibility Principle
- I want to deploy with Docker
- I don't want to have build dependencies in my production environments

Good Tooling

- Multi-stage builds
- Images from scratch
- UPX!

Docker all the things!

```
FROM golang:alpine3.8
```

```
RUN apk --update add git upx && \  
    rm -rf /var/lib/apt/lists/* && \  
    rm /var/cache/apk/*
```

```
WORKDIR /app  
COPY . /app
```

```
RUN go build -o bin/data-service && \  
    /usr/bin/upx /app/bin/data-service
```

```
FROM alpine:3.8
```

```
WORKDIR /  
ENTRYPOINT ["/app/server"]  
COPY --from=0 /app/bin/service /app/server
```

Docker all the things!

Building

```
FROM golang:alpine3.8
```

```
RUN apk --update add git upx && \  
    rm -rf /var/lib/apt/lists/* && \  
    rm /var/cache/apk/*
```

```
WORKDIR /app  
COPY . /app
```

```
RUN go build -o bin/data-service && \  
    /usr/bin/upx /app/bin/data-service
```

Packaging

```
FROM alpine:3.8
```

```
WORKDIR /  
ENTRYPOINT ["/app/server"]  
COPY --from=0 /app/bin/service /app/server
```

Optimise more - meet UPX



Ultimate Packer for eXecutables

Copyright (C) 1996 - 2017

UPX 3.94 Markus Oberhumer, Laszlo Molnar & John Reiser May 12th 2017

File size	Ratio	Format	Name
-----	-----	-----	-----
13410621 ->	6680188	49.81%	linux/amd64 data-service

<https://blog.filippo.io/shrink-your-go-binaries-with-this-one-weird-trick>

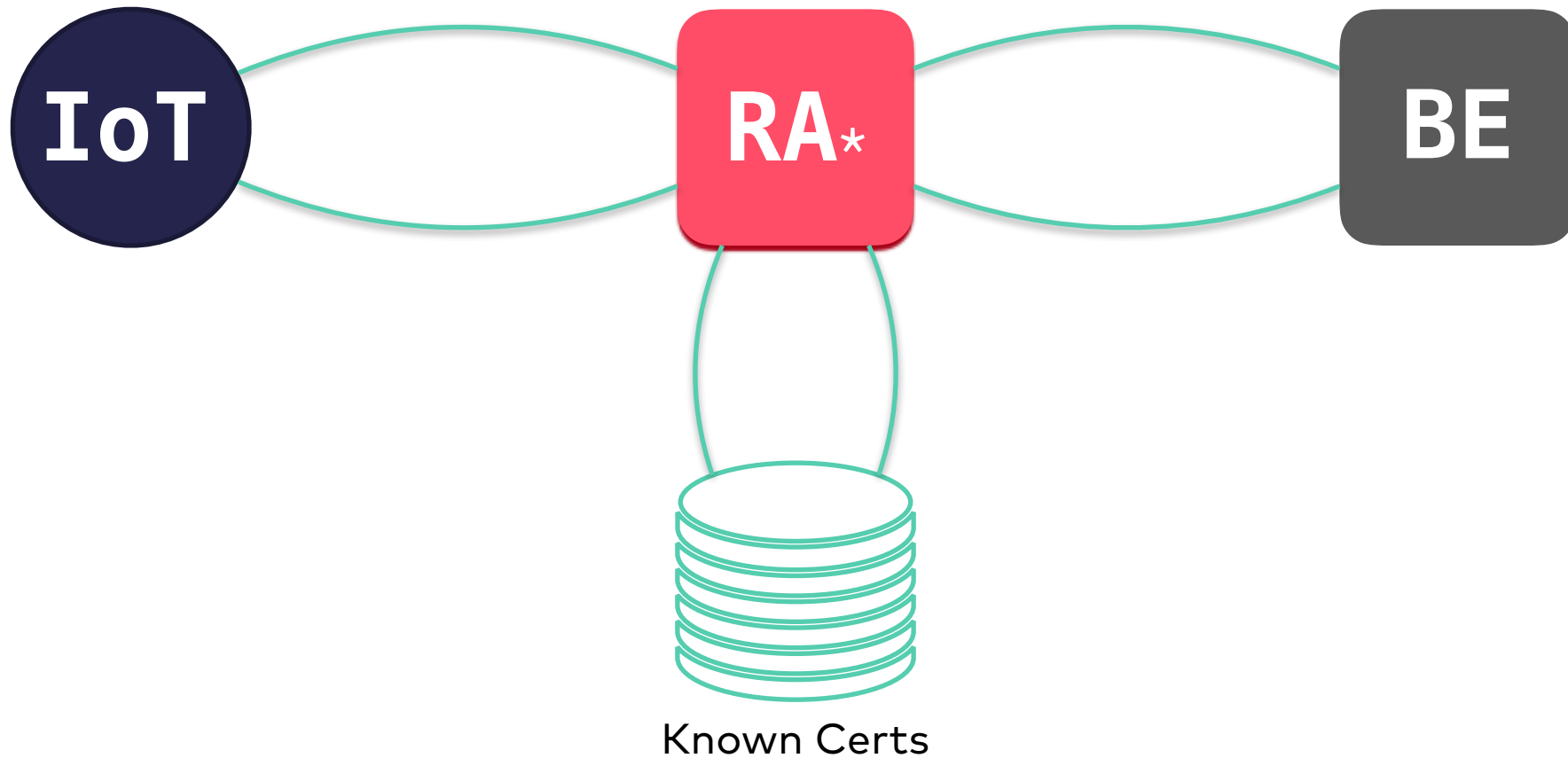


Add-On

7

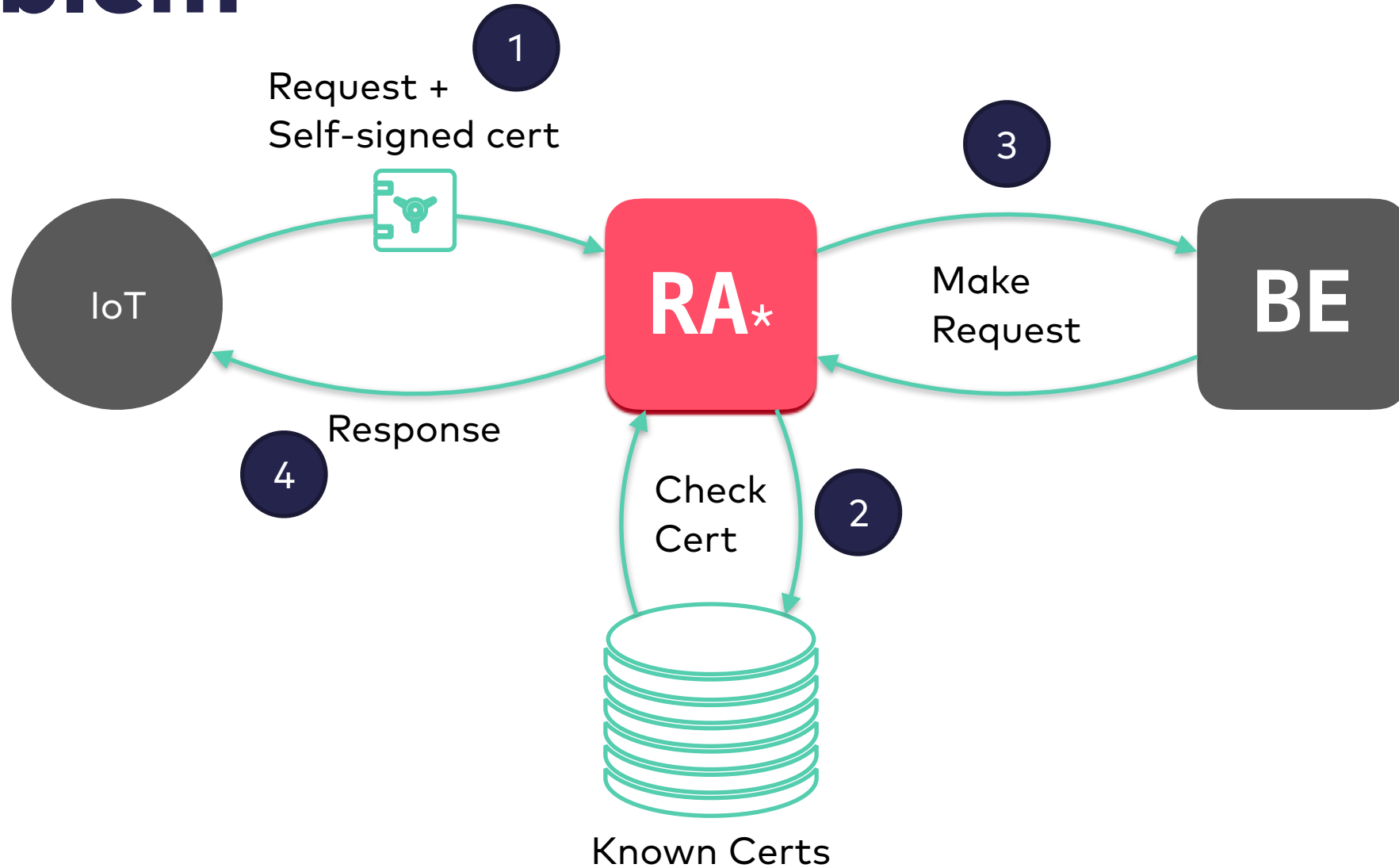
Middleware

Problem



* Request Authority

Problem



* Request Authority

```
package main
```

```
import (  
    "crypto/tls"  
    "crypto/x509"  
    "net/http"  
)
```

```
func verifyCert(  
    rawCerts [][]byte,  
    x509Certs [][]*x509.Certificate)  
    error {  
  
    if validCert(rawCerts) {  
        return nil  
    }  
  
    return errors.New("Cert is invalid!")  
}
```

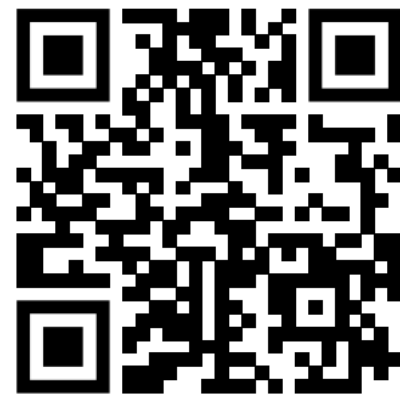
```
func main() {  
    tlsConfig := &tls.Config{  
        ClientAuth:          tls.RequestClientCert,  
        VerifyPeerCertificate: verifyCert,  
    }  
  
    server := &http.Server{  
        Addr:                  ":8443",  
        ReadTimeout:          10 * time.Second,  
        WriteTimeout:         10 * time.Second,  
        MaxHeaderBytes:       1 << 20,  
        TLSConfig:            tlsConfig,  
        Handler:              router,  
    }  
  
    server.ListenAndServeTLS(  
        serverCert,  
        serverKey)  
}
```

8
UI

Problem

- **My App need to have an UI**
- **Should be written in Go**
- **Should be commonly known**
- **Should not look like an alien on my platform**

webview



- UI with JavaScript/HTML
- Support for Windows/MacOS/Linux (Browsers)
- "Electron" for Go


```
package main

import "github.com/zserge/webview"

func main() {
    // Open wikipedia in a 800x600 resizable window
    webview.Open("Minimal webview example",
        "https://en.m.wikipedia.org/wiki/Main_Page", 800, 600, true)
}
```



Today's featured article



O. kokomoensis

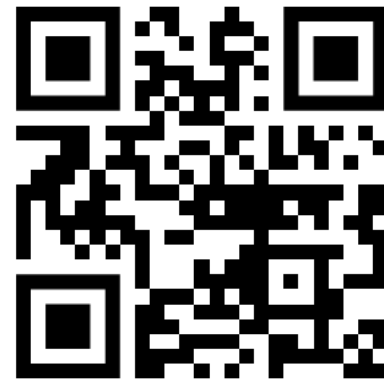
Onychopterella was a [predatory](#) aquatic [arthropod](#) of the [order](#) of [eurypterids](#), often called sea scorpions. [Fossils](#) of the [species](#) *O. kokomoensis* (*pictured*) and *O. pumilus* have been found in the United States, and fossils of *O. augusti* in South Africa. *Onychopterella* (from Greek for 'claw wing') lived from the Late [Ordovician](#) to the Late [Silurian](#), from 444 to 422 million years ago. The head was almost rectangular, with bean-shaped [compound eyes](#). The [limbs](#) were generally long and narrow with a [spine](#) on the tip, and the body was [ornamented](#) with small, pointed [scales](#). Lengths ranged from 16 cm (6.3 in) for *O. kokomoensis* to 4 cm (1.6 in) for *O. pumilus*. *Onychopterella* was able to swim, and probably able to walk on the seabed with its spines and dig with its head. The best-preserved specimens of *O. augusti* show similarities to modern [scorpions](#) in their [alimentary canal](#), limb musculature and [respiratory system](#). ([Full article...](#))

Recently featured:

[Buzz Aldrin](#) [Jill Valentine](#) [Coldrum Long Barrow](#)

[Archive](#) [By email](#) [More featured articles](#)

andlabs/ui



- Native UI elements
- Support for Windows/MacOS/Linux

```
func setupUI() {  
    mainwin = ui.NewWindow(  
        "libui Control Gallery",  
        640, 480, true)  
  
    mainwin.OnClosing(func(*ui.Window) bool {  
        ui.Quit()  
        return true  
    })  
  
    ui.OnShouldQuit(func() bool {  
        mainwin.Destroy()  
        return true  
    })  
  
    tab := ui.NewTab()  
    mainwin.SetChild(tab)  
    mainwin.SetMargined(true)
```

```
    tab.Append("Basic Controls",  
        makeBasicControlsPage())  
    tab.SetMargined(0, true)  
  
    tab.Append("Numbers and Lists",  
        makeNumbersPage())  
    tab.SetMargined(1, true)  
  
    tab.Append("Data Choosers",  
        makeDataChoosersPage())  
    tab.SetMargined(2, true)  
  
    mainwin.Show()  
}  
  
func main() {  
    ui.Main(setupUI)  
}
```

```

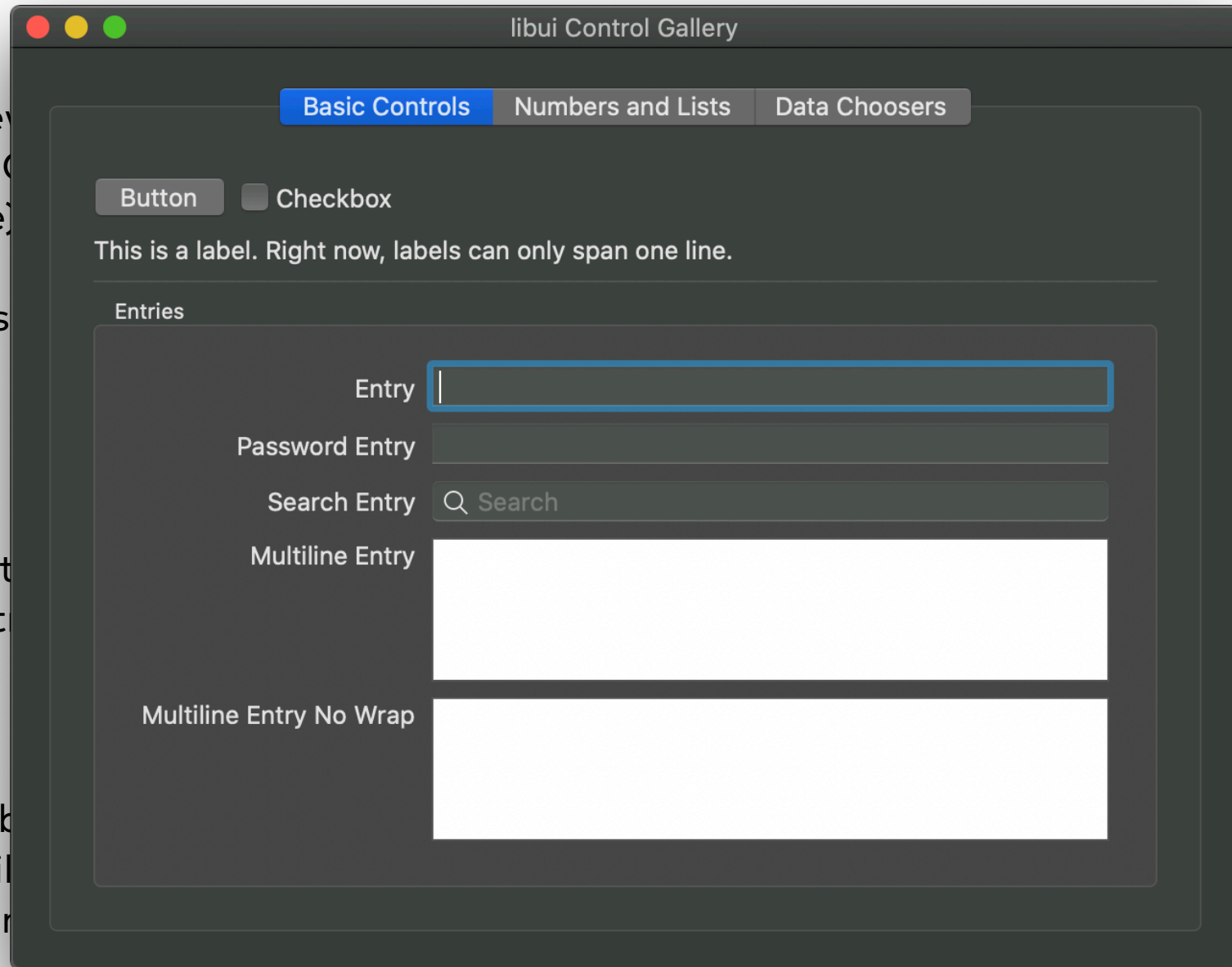
func setupUI() {
    mainwin = ui.NewWindow(
        "libui Control Gallery",
        640, 480, true)

    mainwin.OnCloseEvent(func() {
        ui.Quit()
        return true
    })

    ui.OnShouldQuit(func() {
        mainwin.Destroy()
        return true
    })

    tab := ui.NewTabView()
    mainwin.SetChild(tab)
    mainwin.SetMargin(10)

```



9

Assets

Problem

- There are a lot of additional Assets I need to distribute
- I want to have only one Binary
- I want to be sure, that the Asset can't be change so easy

jteeuwen/go-bindata



- This package converts any file into manageable Go source code.
- Useful for embedding binary data into a go program.
- The file data is optionally gzip compressed before being converted to a raw byte slice.


```
$ go get -u github.com/jteeuwen/go-bindata/...
```

```
data
```

```
`-- pub  
  |-- img  
  | `-- favicon.ico  
  |-- script  
  | `-- main.js  
  `-- style  
    `-- foo.css
```

```
$ go-bindata data/...
```

```
// generated asset.go file in main package.
```

```
// access asset data, via  
Asset(string) ([]byte, error) function
```

```
data, err := Asset("pub/style/foo.css")  
if err != nil {  
    // Asset was not found.  
}
```

```
// use asset data
```



Summary

1. A lot can be done with **onboard** tooling
2. There are **unique go-specific** libraries
3. Developing in Go is **very efficient**



Questions?