going global with i18n4go

dr.max @maximilien
ibm cloud labs
maximilien.org

v0.4.1
April 23rd, 2015

Agenda

- motivations
- what? how? and when?
- goals and design points
- overview of approach
- demo
- technical details and contributing
- what next? and future work

motivations

- **IBM Bluemix** (based on CloudFoundry) is **worldwide PaaS**
- CloudFoundry **CLI is written in Golang** & is main entry to CF
- Golang support for **internationalization (i18n)** is minimal
- existing i18n for Golang mainly runtime libraries
- how to convert existing and green field Golang code to i18n?
- supporting **i18n** requires **lifecycle tooling**:
  - code, test, continuous integration (CI)
  - deployment and maintenance

what? how? and when?

- **what?**
  - internationalization (i18n) tooling for the Golang: **github.com/maximilien/i18n4go**
  - supports lifecycle i18n workflows for both existing and green field Golang programs
- **how?**
  - written entirely in Golang
  - makes heavy use of Go's AST package to manipulate program source code
- **when?**
  - created by IBM and Pivotal in July 2014 and in used since by CloudFoundry CLI
  - available under Apache 2.0 OSS license; email: i18n4go at the Gmail domain

seven goals or design points

1. **automate** as much as possible

2. support both: **pre-** and **post-facto** i18n **use cases**

3. maintain Golang's **static compile-time checks**

4. keep **code readable** (as much as possible)

5. minimal external **dependencies**

6. **JSON** and **PO** resource **file types**

7. implement in **Golang**

hello world - original

```go
package main

import (
    "fmt"
)

const VERSION = "v0.0.1"

func main() {
    fmt.Println("Hello from Goffer land and i18n4go")
    fmt.Println("")
    fmt.Printf("Version %s\n", VERSION)
}

```

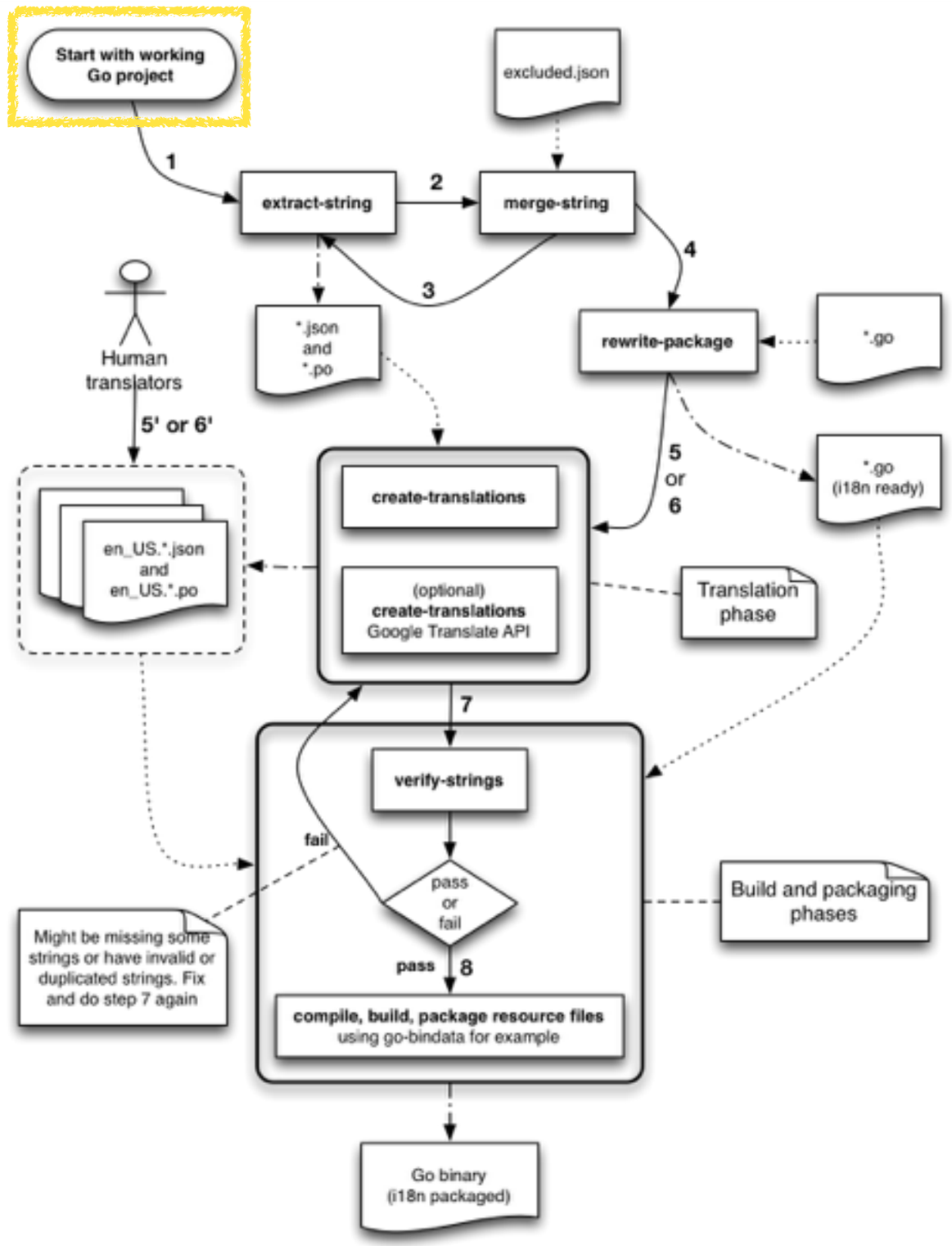
hello world - desiderata

```go
package main

import (
    "fmt"
)

const VERSION = "v0.0.1"

func main() {
    fmt.Println(T("Hello from Goffer land and i18n4go"))
    fmt.Println("")
    fmt.Printf(T("Version {{.Arg0}}\n", map[string]interface{}{"Arg0": VERSION}))
}

```

overview of approach

- use @**NickSnyder**'s **go-i18n** project on Github for runtime
- **created tooling** for
  1. **extracting** and **merging strings** from Golang programs
  2. **rewriting Golang** code to enable i18n translations
  3. **verify strings** and different i18n resources
  4. **initialize locale** by auto discovery (depends on OS)
- **i18n4go** enables **all of the above** and **more**

overview of approach (*cont.*)



hello world - initial program

```
package main

import (
    "fmt"
)

const VERSION = "v0.0.1"

func main() {
    fmt.Println("Hello from Goffer land and i18n4go")
    fmt.Println("")
    fmt.Printf("Version %s\n", VERSION)
}
```

hello world - extracted strings (en_US)

```
[
    {
        "id": "Hello from Goffer land and i18n4go",
        "translation": "Hello from Goffer land and i18n4go",
        "modified": false
    },
    {
        "id": "Version {{.Arg0}}\n",
        "translation": "Version {{.Arg0}}\n",
        "modified": false
    }
]
```

hello world - rewritten code

```
package main

import (
    "fmt"
)

const VERSION = "v0.0.1"

func main() {
    fmt.Println(T("Hello from Goffer land and i18n4go"))
    fmt.Println("")
    fmt.Printf(T("Version {{.Arg0}}\n",
                map[string]interface{}{"Arg0": VERSION}))
}
```

hello world - i18n init code (partial)

```
…
func Init(detector Detector) goi18n.TranslateFunc {
    var T goi18n.TranslateFunc
    var err error

    var userLocale string
    userLocale, err = initWithUserLocale(detector)
    if err != nil {
        userLocale = mustLoadDefaultLocale()
    }

    T, err = goi18n.Tfunc(userLocale, DEFAULT_LOCALE)

    if err != nil {
        panic(err)
    }

    return T
}
…
```

hello world - resulting code structure

```
➜  i18n4go git:(master) tree examples/demo1-i18n
examples/demo1-i18n
├── build
├── demo1.go
├── demo1.go.en.json
├── en.all.json
├── excluded.json
├── generate-language-resources
├── i18n
│   └── resources
│       ├── de.all.json
│       ├── de_DE.all.json
│       ├── en_US.all.json
│       ├── es.all.json
│       ├── es_ES.all.json
│       ├── fr.all.json
│       ├── fr_FR.all.json
│       ├── zh.all.json
│       └── zh_Hans.all.json
├── i18n_init.go
├── i18n_resources.go
└── out
    └── demo1

3 directories, 18 files
```

demo

---

live demo

https://github.com/maximilien/i18n4go/tree/master/examples/demo1

technical details

- heavy use of **golang.org/pkg/go/ast** package
  - **`extract-strings`** and **`rewrite-package`**
  - **`go-fmt`** to reformat code (uses AST as well)
- auto **`create-translations`** is OK but humans way better
- need to **package** i18n JSON as **binary**, e.g., go-bindata
- **`verify-strings`** and **`checkup`** and **`fixup`** for maintenance

what next?

- **being used** for about one year now in **CF CLI**
- can we **simplify basic workflow** even more?
- other **auto-translators**? e.g., IBM Watson, Bing
- support strings with localized **dates**, **currency**, numbers
- **socialize** to Golang communities, e.g., GoSF
- we welcome **contributors**…

credit: http://knowyourmeme.com/photos/522333-language

@UsedRecently

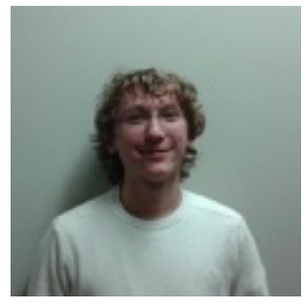
@maximilien

@LeftSaidTim

Various at Pivotal

Simon Leung

@jberkhahn



Dan Lavine