Gunosy APIチーム

Go開発フロー

自己紹介

- Twitter: @k_yokomi （きょこみ）※1
- 2015年1月にGunosyへ転職
- Goエンジニアやってます
- 以前は、渋谷の某ゲーム会社で
  Cocos2d-xを使ったスマホゲーム開発
- 最近go : generateがマイブーム

※1 アイコンは月姫の琥珀さんです。

目次

- 概要
- 開発フロー
- 今後について

概要

Goの開発フロー内で、どんなツールや技術を使っているのか等、一般的な開発フローにそって順番に紹介します。

対象者:

- これからGoを導入しようとしている
- すでに導入していて他ではどのように開発しているか知りたい
- 新規アサインしたメンバー

開発フロー（概要）

- 環境構築・エディタ
- コーディング・実装・テスト
- ステージング環境・デプロイ戦略
- その他（ライブラリ選定、パッケージ/ファイル分け）
- 運用・監視

環境構築・エディタ

環境構築

- Go環境は事前にbrew等で構築済みという前提
- プロジェクトのREADME.mdに環境構築手順を記載している
- PCはMacが多いが、Ubuntuで開発していたメンバーもいた
- 環境の違いでのトラブルが起きたケースは今のところない
- 今のところWindowsで開発している人はいない

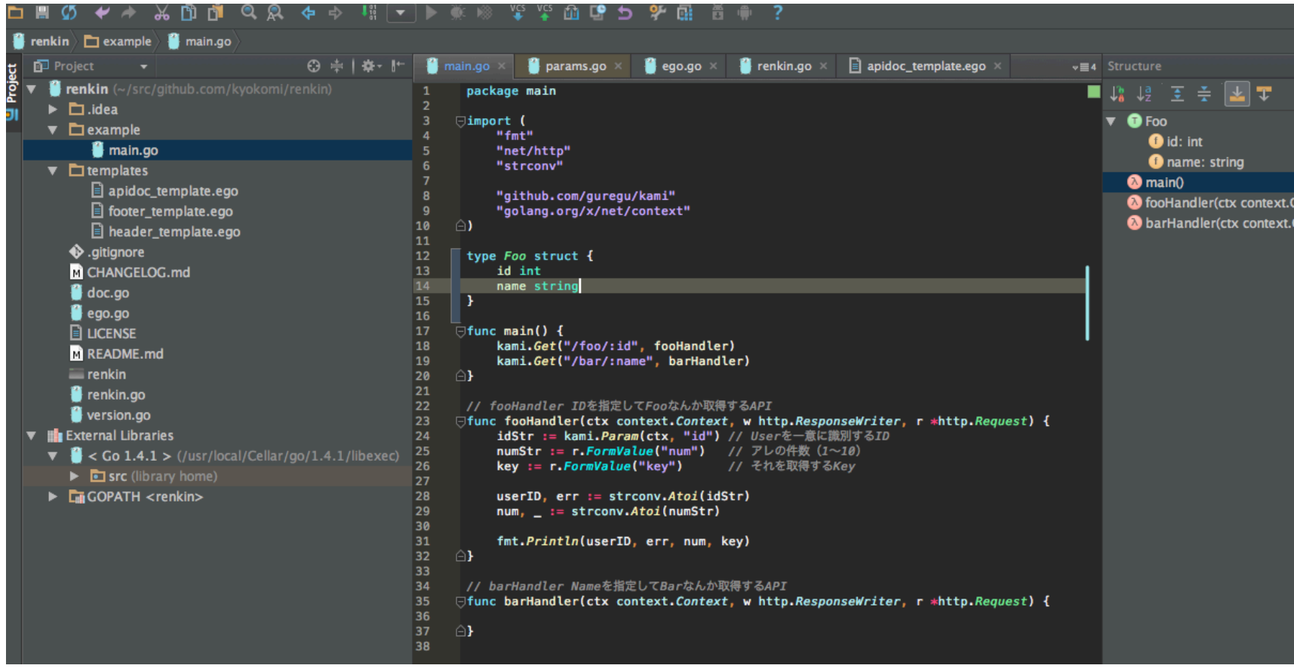
エディタ

- IntelliJ IDEA Community Edition
- Sublime
- Vim
- Emacs

`gofmt -s`とか`goimports`を行っているので、基本的にエディタの違いで困ったケースはない。

IntelliJ IDEAでGo

- 自分はIntelliJ IDEA推し
- Community Edition（無料）
- 最近、Golang Pulginの開発がIntelliJ本家の人？に変わったらしい
- ver0.9の頃とくらべて、開発も活発になり、かなり使い勝手良くなった
- package補完・コード補完・参照元検索などなど



コーディング・実装

ローカルでの動作確認

- `go test`する
- もしくはbuildして実行する
- gdb等のデバッガは基本的に使っていない
- printデバッグで十分
- MySQL,Redis,MongoDB等は、ローカルのものを利用

コーディングガイド

今は、コーディングガイドはない。以下くらいで十分な認識。

- `A tour of go`をひと通りやっておく
- EffectiveGoを一読する
- 鵜飼文敏さんの「Goに入ってはGoに従え」を一読する※2
- golintでのfiled名は対応する（Id -> ID, hogeUrl -> hogeURL等）

※2 http://ukai-go-talks.appspot.com/2014/gocon.slide#1

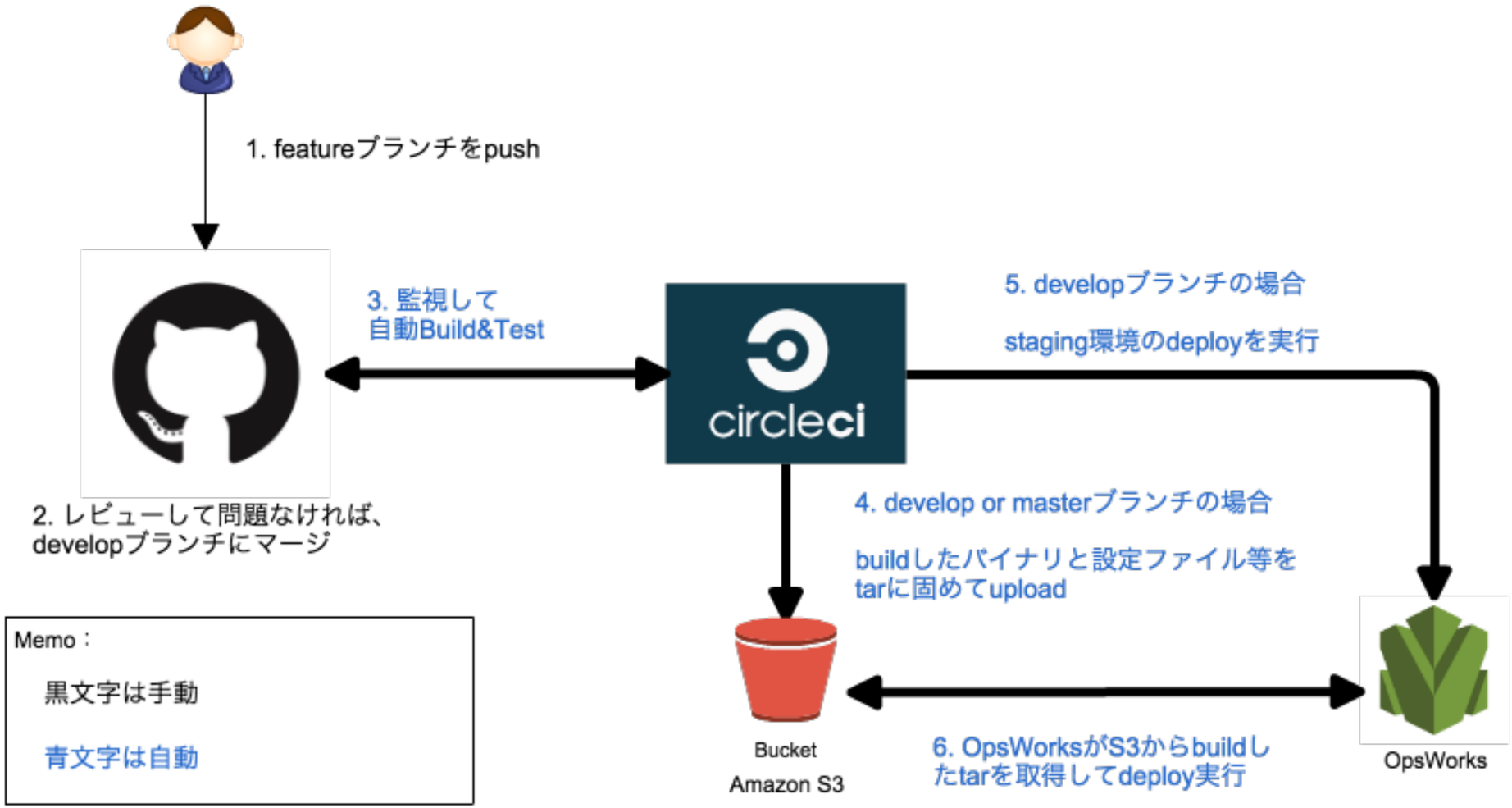
テスト

- `guregu/mogi`を使ってsqlはmockしてる
- RedisやMongoDBは、ローカルのtest用に接続先を変えている
- Handlerのtestには、普通にnet/httptestを使ってる
- testingフレームワークは使ってない（Gunosy.go#10を参照[※3]）

[※3] Gunosy.go#10 testnight http://gunosygo.connpass.com/event/8485/

ステージング環境・デプロイ戦略

かっこいいdeployフロー


1. featureブランチをpush
2. レビューして問題なければ、
developブランチにマージ
3. 監視して
自動Build&Test
circleci
4. develop or masterブランチの場合
buildしたバイナリと設定ファイル等を
tarに固めてupload
5. developブランチの場合
staging環境のdeployを実行
Bucket
Amazon S3
6. OpsWorksがS3からbuildし
たtarを取得してdeploy実行
OpsWorks
Memo：
黒文字は手動
青文字は自動

その他

（ライブラリ選定、パッケージ/ファイル分け）

ライブラリ選定

- できるだけgopkg.inでバージョン固定しているものを利用
- Godep等のパッケージライブラリは利用していない
- 導入する際に担当者がちゃんと調査・検討して他のメンバーと相談してから入れる
- もしくは自作する
- 外に出せるものは、どんどんライブラリ化してGitHubで公開

GitHub公開しているライブラリ

主なライブラリを抜粋（他にも色々あります）

- gopkg.in/guregu/null.v2: null可能SQLを扱う
- github.com/guregu/toki: SQLTIME型のnull値可能
- github.com/guregu/buildver: build version（4.1.10.5）とかを扱う
- github.com/guregu/mogi: sqlのスタブ
- github.com/guregu/kami: gojiベースのWebフレームワーク
- github.com/kyokomi/goebi: errbitクライアント

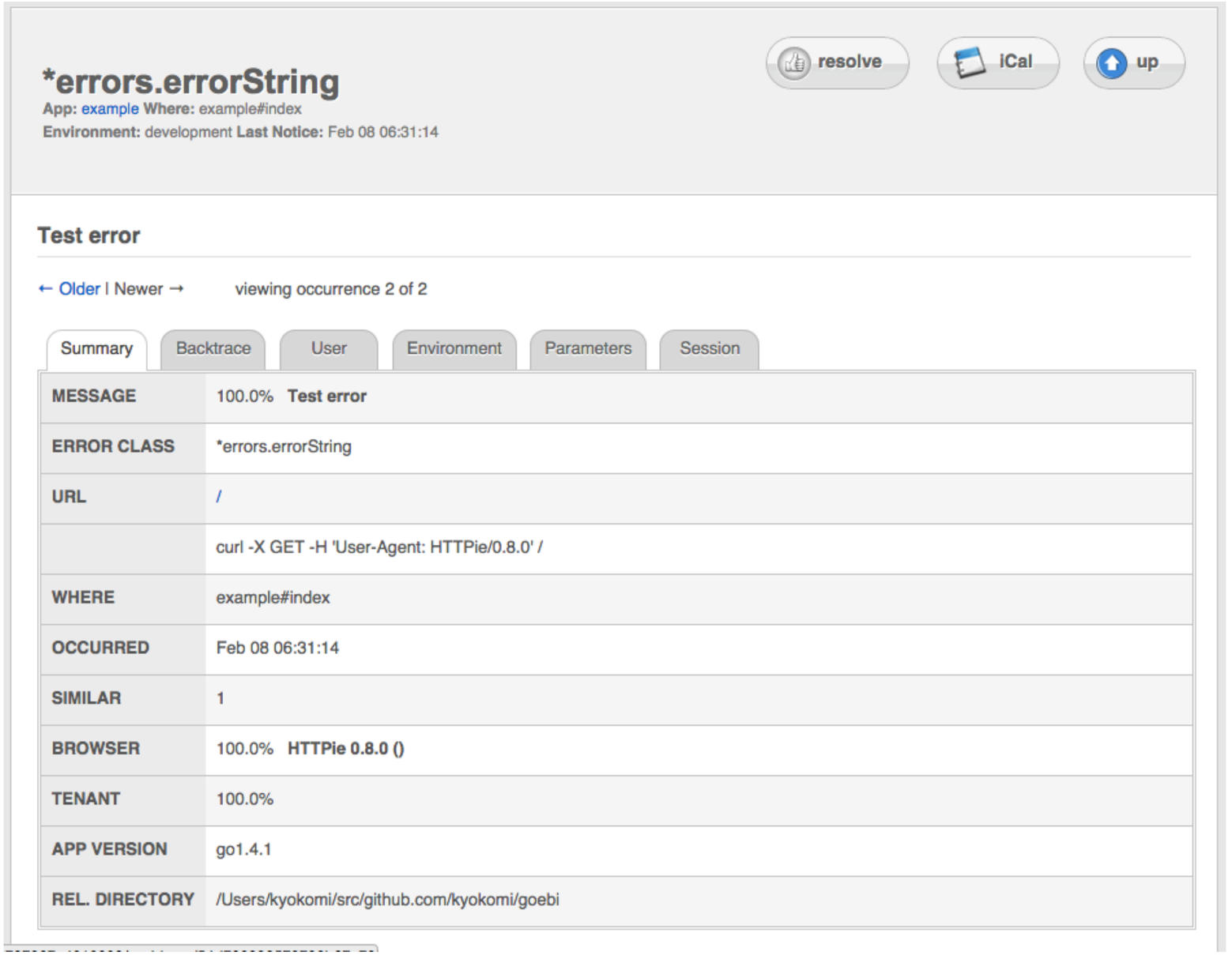
パッケージ分け、ソースファイル分け

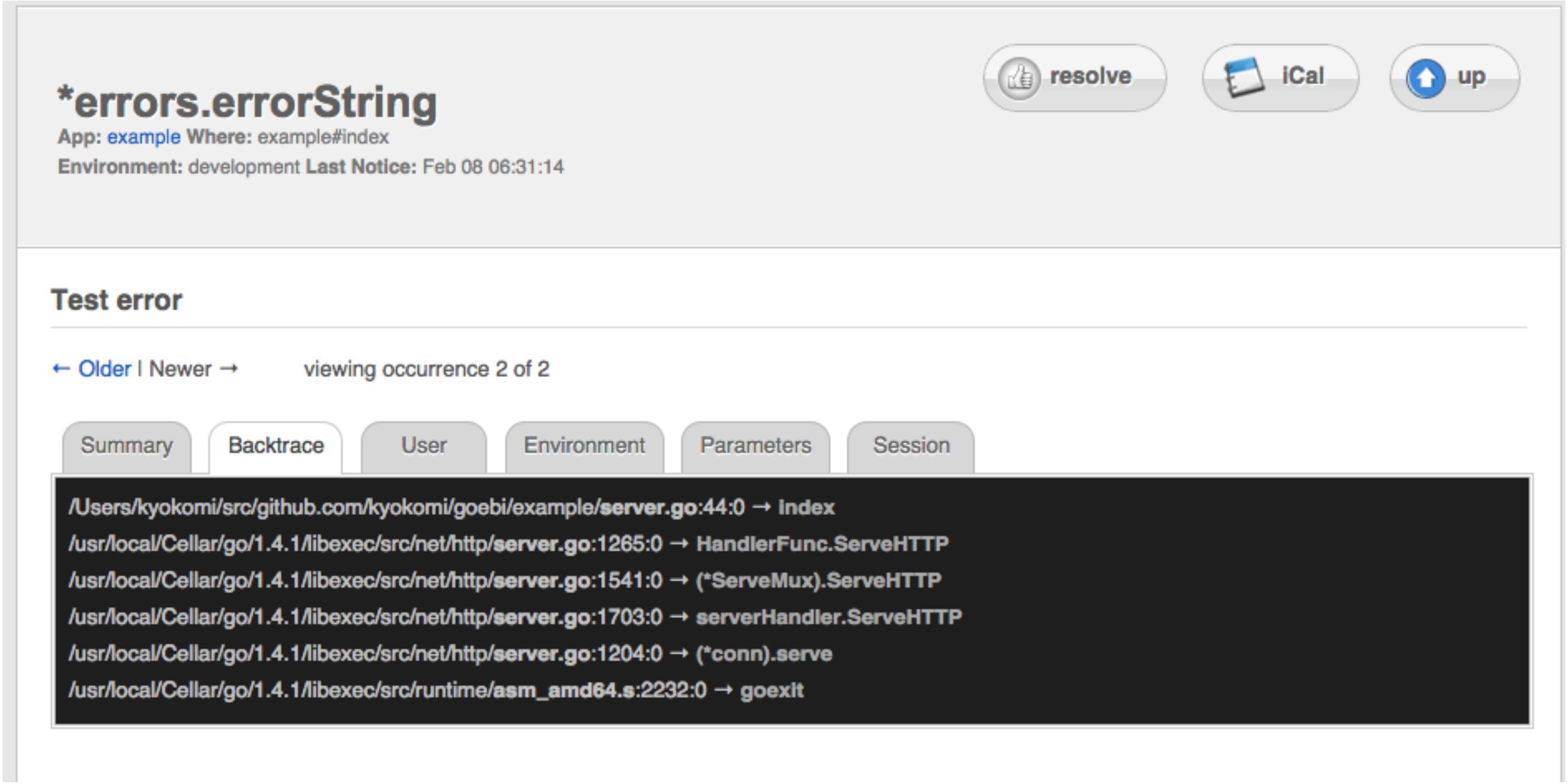
- 社内共通パッケージを別リポジトリに立てている: `github.com/gunosy/go`
- ディレクトリ構成というかパッケージ分けは未だに課題
  - これの時はこういう構成がいいといった決定版がまだない
  - パッケージ分け、ファイル分けの知見を共有する勉強会やりたいと思ってる

運用・監視

運用・監視

- panicのHandlerでpanicしたら、mail通知している（errbit移行したのでやめる予定）
- 標準出力をlogファイルに出力
- 最近、errbitを導入してエラー監視するようにした


*errors.errorString
App: example Where: example#index
Environment: development Last Notice: Feb 08 06:31:14
resolve
iCal
up
Test error
← Older | Newer →
viewing occurrence 2 of 2
Summary
Backtrace
User
Environment
Parameters
Session
MESSAGE
100.0% Test error
ERROR CLASS
*errors.errorString
URL
/
curl -X GET -H 'User-Agent: HTTPie/0.8.0' /
WHERE
example#index
OCCURRED
Feb 08 06:31:14
SIMILAR
1
BROWSER
100.0% HTTPie 0.8.0 ()
TENANT
100.0%
APP VERSION
go1.4.1
REL. DIRECTORY
/Users/kyokomi/src/github.com/kyokomi/goebi


*errors.errorString
App: example Where: example#index
Environment: development Last Notice: Feb 08 06:31:14
resolve
iCal
up
Test error
← Older | Newer →
viewing occurrence 2 of 2
Summary
Backtrace
User
Environment
Parameters
Session
/Users/kyokomi/src/github.com/kyokomi/goebi/example/server.go:44:0 → Index
/usr/local/Cellar/go/1.4.1/libexec/src/net/http/server.go:1265:0 → HandlerFunc.ServeHTTP
/usr/local/Cellar/go/1.4.1/libexec/src/net/http/server.go:1541:0 → (*ServeMux).ServeHTTP
/usr/local/Cellar/go/1.4.1/libexec/src/net/http/server.go:1703:0 → serverHandler.ServeHTTP
/usr/local/Cellar/go/1.4.1/libexec/src/net/http/server.go:1204:0 → (*conn).serve
/usr/local/Cellar/go/1.4.1/libexec/src/runtime/asm_amd64.s:2232:0 → goexit

今後について

コーディングガイド作成

社内コーディングガイドの作成を考えている。

- 良くない例
- レビューで指摘があったもの

ガイドというほどのものではないが上記のようなものを見やすくしてまとめておくだけでもよさそう。

ドキュメント管理

Qiita:TeamとかwikiとかにかくとIいちいち開かないといけなくて更新が辛い。

APIドキュメントの自動生成（GoDocベース）

現在進行中（今はkami専用だが、70%くらいできてる）

github.com/kyokomi/renkin [※4]

[※4] https://github.com/kyokomi/renkin

```go
package main

import (
    "fmt"
    "net/http"
    "strconv"

    "github.com/guregu/kami"
    "golang.org/x/net/context"
)

func main() {
    kami.Get("/foo/:id", fooHandler)
    kami.Get("/bar/:name", barHandler)
}

// fooHandler IDを指定してFooなんか取得するAPI
func fooHandler(ctx context.Context, w http.ResponseWriter, r *http.Request) {
    idStr := kami.Param(ctx, "id") // Userを一意に識別するID
    numStr := r.FormValue("num")   // アレの件数（1〜10）
    key := r.FormValue("key")      // それを取得するKey

    userID, err := strconv.Atoi(idStr)
    num, _ := strconv.Atoi(numStr)

    fmt.Println(userID, err, num, key)
}

// barHandler Nameを指定してBarなんか取得するAPI
func barHandler(ctx context.Context, w http.ResponseWriter, r *http.Request) {
    // hogehoge
}
```

```
// NOTE: THIS FILE WAS PRODUCED BY THE
// RENKIN APIDOC GENERATION TOOL (github.com/kyokomi/renkin)
// DO NOT EDIT

/*
RENKIN APIDOC GENERATION TOOL (https://github.com/kyokomi/renkin).


Get fooHandler ・・・ IDを指定してFooなんか取得するAPI


Path:
    "/foo/:id"

Params:
    1. `"id"`: Userを一意に識別するID (kami.Param)
    2. `"num"`: アレの件数（1〜10） (r.FormValue)
    3. `"key"`: それを取得するKey (r.FormValue)


file at example/main.go.

Get barHandler ・・・ Nameを指定してBarなんか取得するAPI


Path:
    "/bar/:name"

Params:
    none

file at example/main.go.


*/
package main
```

Command renkin

Get fooHandler ・・・ IDを指定してFooなんか取得するAPI
Get barHandler ・・・ Nameを指定してBarなんか取得するAPI
Subdirectories

RENKIN APIDOC GENERATION TOOL (https://github.com/kyokomi/renkin).

Get fooHandler ・・・ IDを指定してFooなんか取得するAPI

Path:

```
"/foo/:id"
```

Params:

```
1. `"id"`: Userを一意に識別するID (kami.Param)
2. `"num"`: アレの件数 (1〜10) (r.FormValue)
3. `"key"`: それを取得するKey (r.FormValue)
```

file at example/main.go.

Get barHandler ・・・ Nameを指定してBarなんか取得するAPI

Path:

```
"/bar/:name"
```

Params:

```
none
```

file at example/main.go.

テスト

kyokomi/renkinのフェーズ２を検討中。

- 各APIのテストコードひな形をgo:generate自動生成
    - コピペして少し直す手間を減らしたい
- ステージング環境向けの総合テストのコード自動生成
    - 全APIを叩くイメージ

おわり