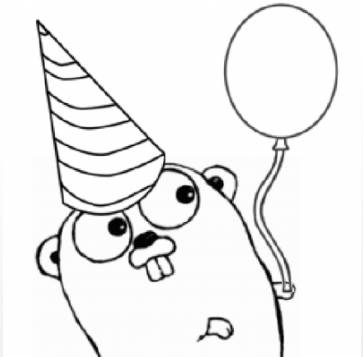
Framework

Revel (v0.12.0)

Framework WEB à haute productivité pour Go

présenté par
Jonathan Masmejean
Lead Developper chez Padawan Group

Et surtout fan de Go :-D



Société éditant des sites de petites annonces à l'international

- 24 sites dans 24 pays
- Plus de 130 Millions d'annonces
- 60 employés dispersés entre la France, le Royaume Uni et l'Inde

On fait :

- Du Symfony2
- Du NodeJS
- Du Go
- Et beaucoup d'autres super technos :-D

Revel

Est :

- Un Framework FullStack à
  - Hautes performances
  - Haute productivité
- Du code auto-compilé
- Pratique pour l’internationalisation

N’est pas :

- Un micro framework
- Une machine à café
- Du legacy code

Est :

- Exhaustif (routing, parameter parsing, validation, session/flash, templating, caching, job running, framework de testing, internationalization.)
- Stateless (chaque requête est gérée par une goroutine)
- Modulaire (grâce aux “filters” que nous verrons plus tard)

N’est pas :

- Compliqué à prendre en main
- Adapté à tout type de projet (WEB App vs API)

Quick Start

Revel met à disposition de nombreux exemples d'applications pour démontrer les usages les plus courant.

Pour débuter un projet Revel, il suffit de lancer ces différentes commandes (après avoir correctement installé et configuré Go (GOPATH, GOROOT…)

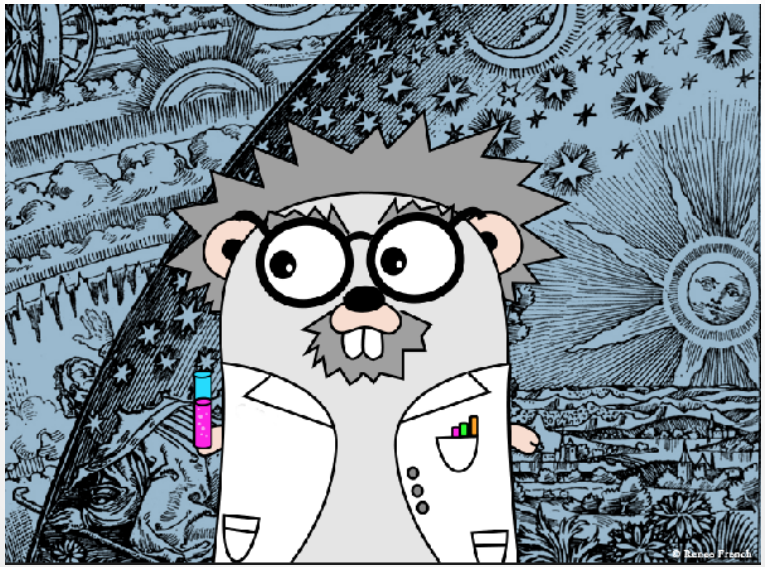
```
# get revel framework
go get github.com/revel/revel

# get 'revel' command
go get github.com/revel/cmd/revel

# create a new app and run
revel new github.com/myaccount/my-app
```

Architecture d'un projet Revel

- `my_gocode/` - GOPATH root
  - `src/` - GOPATH src/ directory
    - `github.com/revel/revel/` - Revel source code
    - `bitbucket.org/me/sample/` - Sample app root
      - `app/` - app sources
        - `controllers/` - app controllers
          - `init.go` - interceptor registration
        - `models/` - app domain models
        - `routes/` - reverse routes (generated code)
        - `views/` - templates
      - `tests/` - test suites
      - `conf/` - configuration files
        - `app.conf` - main configuration file
        - `routes` - routes definition file
      - `messages/` - i18n message files
      - `public/` - static/public assets
        - `css/` - stylesheet files
        - `js/` - javascript files
        - `images/` - image files

Revel

Maintenant que vous connaissez les grandes lignes, voici des petits zooms sur certains aspects du Framework



Le Routing

```
# conf/routes file
# This file defines all application routes (Higher priority routes first)
GET     /login                  Application.Login        # A simple path
GET     /hotels/                Hotels.Index             # Matches with or without trailing slash
GET     /hotels/:id             Hotels.Show              # Extract an embedded argument
WS      /hotels/:id/feed        Hotels.Feed              # WebSockets.
POST    /hotels/:id/:action     Hotels.:action           # Automatically route some actions.
GET     /public/*filepath       Static.Serve("public")   # Assets served from /public/...
*       /:controller/:action    :controller.:action      # Catch all; Automatic URL generation
```

Revel utilise une syntaxe déclarative pour gérer le routing.

Les Contrôleurs

Un contrôleur est n'importe quel *type* qui intègre le pointeur **revel.Controller*. *(voir l'héritage en GO)*

```
type MyAppController struct {
    *revel.Controller
}
type MyOtherController struct {
    *revel.Controller
    OtherStuff string
    MyNo int64
}
```

Revel

Les “Interceptors”

Les “interceptors” sont des fonctions qui sont instanciées “avant” ou “après” une instanciation d’action.

Ce qui peut apporter une notion de POO qu’on a tous plus ou moins l’habitude d’ utiliser comme :

- Request logging
- Error handling
- Statistics logging

Les “Interceptors” suite

Dans Revel, il existe 2 type d’interceptors :

1. Le type “function interceptor”

Ci dessous un exemple d’une définition et d’une assignation d’un “function interceptor” pour tous les contrôleurs.

```
func checkUser(c *revel.Controller) revel.Result {
    if user := connected(c); user == nil {
        c.Flash.Error("Please log in first")
        return c.Redirect(App.Index)
    }
    return nil
}
func init() {
    revel.InterceptFunc(checkUser, revel.BEFORE, &App{})
}
```

- La fonction utilise l’interface `InterceptorFunc`
- Elle n’a pas accès au contrôleur d’application spécifique instancié
- Peut s’appliquer à un ou tous les contrôleurs d’une application

Les “Interceptors” suite

2. Le type “method interceptor”

La signature d’un “method interceptor” doit correspondre à une de ses 2 formes

```
func (c AppController) example() revel.Result
func (c *AppController) example() revel.Result
```

Voici le même exemple mais qui n’opère que sur le contrôleur de l’application (ici Hotels)

```
func (c Hotels) checkUser() revel.Result {
    if user := connected(c); user == nil {
        c.Flash.Error("Please log in first")
        return c.Redirect(App.Index)
    }
    return nil
}
func init() {
    revel.InterceptMethod(Hotels.checkUser, revel.BEFORE)
}
```

- Une méthode de contrôleur n’acceptant pas d’argument et retournant `revel.Result`
- Ne peut intercepter que des appels liés à ce contrôleur
- Peut modifier à souhait le contrôleur invoqué

Les “Filters”

Les “Filters” représentent la partie middleware et sont répartis en fonctions individuelles qui constituent le pipeline de traitement des demandes.

Toutes les fonctionnalités du Framework (ou presque) sont conçues avec ces fameux “Filters”.

Voici la signature d’un filter :

```
type Filter func(c *Controller, filterChain []Filter)
```

Chaque filtre est responsable du chargement et de l’instanciation du prochain filtre de la chaîne. Donc pour implémenter un filtre il ne faut pas oublier les autres de la chaîne !

```
var MyFilter = func(c *revel.Controller, fc []revel.Filter) {
    // .. do some pre-processing ..

    fc[0](c, fc[1:]) // Execute the next filter stage.

    // .. do some post-processing ..
}
```

Informations pratiques

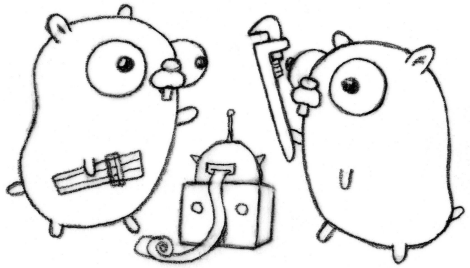
Revel est le plus vieux Framework Fullstack du moment pour le langage Go (4ans)

1 168 commits

96 contributeurs

En version 0.12.0

106 tickets open



La communauté est grandissante, et grandit en même temps que la communauté Go.

Revel

Des questions ?

Revel

Merci !

Café ? :-D

Sources

You can find all you need to know about Revel here :

http://revel.github.io/

http://revel.github.io/manual/index.html

Issues listed here : https://github.com/revel/revel/issues